

UNIVERSITY OF QUEBEC AT MONTREAL

STATISTICAL METHODS FOR ANALYSIS AND CORRECTION OF HIGH-  
THROUGHPUT SCREENING DATA

DISSERTATION  
PRESENTED  
AS A PARTIAL REQUIREMENT  
OF THE DOCTORATE IN COMPUTER SCIENCE

BY

PLAMEN DRAGIEV

NOVEMBER 2012

UNIVERSITÉ DU QUÉBEC À MONTRÉAL  
Service des bibliothèques

Avertissement

La diffusion de cette thèse se fait dans le respect des droits de son auteur, qui a signé le formulaire *Autorisation de reproduire et de diffuser un travail de recherche de cycles supérieurs* (SDU-522 – Rév.01-2006). Cette autorisation stipule que «conformément à l'article 11 du Règlement no 8 des études de cycles supérieurs, [l'auteur] concède à l'Université du Québec à Montréal une licence non exclusive d'utilisation et de publication de la totalité ou d'une partie importante de [son] travail de recherche pour des fins pédagogiques et non commerciales. Plus précisément, [l'auteur] autorise l'Université du Québec à Montréal à reproduire, diffuser, prêter, distribuer ou vendre des copies de [son] travail de recherche à des fins non commerciales sur quelque support que ce soit, y compris l'Internet. Cette licence et cette autorisation n'entraînent pas une renonciation de [la] part [de l'auteur] à [ses] droits moraux ni à [ses] droits de propriété intellectuelle. Sauf entente contraire, [l'auteur] conserve la liberté de diffuser et de commercialiser ou non ce travail dont [il] possède un exemplaire.»

UNIVERSITÉ DU QUÉBEC À MONTRÉAL

MÉTHODES STATISTIQUES POUR L'ANALYSE ET LA CORRECTION DES  
DONNÉES DE CRIBLAGE À HAUT DÉBIT

THÈSE  
PRÉSENTÉE  
COMME EXIGENCE PARTIELLE  
DU DOCTORAT EN INFORMATIQUE

PAR  
PLAMEN DRAGIEV

NOVEMBRE 2012

To my two lovely daughters Zlatina and Aleksandra

## ACKNOWLEDGMENTS

This thesis would not have been possible without the help and the support of many people. Above all, I am heartily thankful to my two supervisors, Professor Vladimir Makarenkov from Université du Québec à Montréal and Professor Robert Nadon from McGill University for their guidance, encouragement and invaluable assistance throughout my studies that enabled me to develop understanding of the subject, to advance in and complete my research work. Special thanks, also, to my colleagues and friends at UQAM and Genome Quebec, who gave me their moral support, shared literature and assisted me with research materials. I would like to express my gratitude to all professors and administrative staff at UQAM who in one way or another helped me to conclude my studies. I am also grateful to Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT) for the financial support.

On a personal level, I owe thanks to my wife, my two daughters and my parents for their love, understanding and most of all for their patience during this multi-year adventure.

## TABLE OF CONTENTS

LIST OF FIGURES.....	V
LIST OF TABLES .....	XI
LIST OF ABBREVIATIONS AND ACRONYMS .....	XIV
RÉSUMÉ.....	XV
ABSTRACT .....	XVII
CHAPTER I	
INTRODUCTION.....	1
1.1 High-Throughput Screening.....	1
1.2 HTS experimental errors .....	5
1.3 Hit selection .....	6
1.4 False positive and false negative hits .....	8
1.5 Sensitivity and specificity of a statistical model .....	8
1.6 Types of experimental error .....	9
1.7 Within-plate HTS controls .....	10
1.8 Systematic error correction and data normalization.....	11
1.8.1 Percent of control.....	11
1.8.2 Normalized percent inhibition .....	11
1.8.3 Z-score .....	12
1.8.4 Assay quality and validation.....	12
1.8.5 B-score .....	15
1.8.6 Well correction .....	16
1.8.7 Other HTS-related research .....	17
1.9 Machine learning algorithms in HTS .....	21
1.10 Decision trees .....	23
1.11 Neural networks .....	24

CHAPTER II	
SYSTEMATIC ERROR DETECTION IN EXPERIMENTAL HIGH-THROUGHPUT SCREENING .....	27
2.1 Abstract .....	29
2.1.1 Background.....	29
2.1.2 Results.....	29
2.1.3 Conclusions.....	29
2.2 Background .....	30
2.3 Materials And Methods.....	35
2.3.1 Data description .....	35
2.3.2 Generating systematic error .....	36
2.3.3 Systematic error detection tests .....	38
2.4 Results And Discussion.....	42
2.4.1 Simulation 1: Detecting systematic error in individual plates .....	42
2.4.2 Simulation 2: Detecting systematic error on hit distribution surfaces .....	47
2.4.3 Application to the McMaster data .....	50
2.5 Conclusions .....	54
2.6 Supplementary Figures.....	57
CHAPTER III	
TWO EFFECTIVE METHODS FOR CORRECTING EXPERIMENTAL HIGH-THROUGHPUT SCREENING DATA .....	76
3.1 Abstract .....	78
3.2 Introduction .....	78
3.3 Methods.....	81
3.3.1 Data preprocessing in HTS.....	81
3.3.2 Two new data correction methods .....	83
3.4 Results And Discussion.....	89
3.4.1 Simulation study .....	89
3.4.2 Analysis of the McMaster Test assay .....	98
3.5 Conclusion.....	108
3.6 Supplementary Materials.....	110
3.6.1 t-test applied in the HTS context .....	110
3.6.2 $\chi^2$ goodness-of-fit test applied in the HTS context .....	111

CHAPTER IV	
STATISTICAL METHODS FOR THE ANALYSIS OF EXPERIMENTAL HIGH-THROUGHPUT SCREENING DATA .....	112
4.1 Abstract .....	114
4.2 Introduction .....	114
4.3 Analysis of the hit and no hit data in the McMaster <i>Test dataset</i> .....	115
4.4 Polynomial RDA to establish relationships between hit/no hit outcomes and values of chemical descriptors .....	118
4.5 Prediction of experimental HTS results using decision trees and neural networks.....	120
4.6 Probability-based hit selection method .....	122
4.7 New measures for assay quality estimation depending on the number of replicates .....	131
4.8 Discussion and future developments.....	140
CONCLUSION .....	142
APPENDIX A	
HTS HELPER SOFTWARE.....	148
A.1 HTS Helper Utility .....	148
A.2 Using HTS Helper Utility.....	150
A.3 HTS Helper Command Line Interface .....	152
Syntax .....	152
Parameters.....	152
APPENDIX B	
SOURCE CODE OF THE HTS HELPER UTILITY (VER 1.0) .....	154
Program.cs.....	154
UserInterface.cs.....	154
CommandLine.cs .....	155
HTSData.cs .....	161
Plate.cs .....	164
Dataset.cs .....	173
WellData.cs .....	177
G.cs.....	178
HTSFileReader.cs .....	182
HTSFileWriter.cs .....	193
HTSHelper.cs .....	198

REFERENCES.....204

## LIST OF FIGURES

Figure	Page
1.1 Drug development process (source: Malo et al. 2006).....	2
1.2 Size of corporate screening collections over time. This figure shows that the screening collections of four pharmaceutical companies from 2001 differ dramatically from those from 2009. Data taken from GlaxoSmithKline, Novartis, Sanofi-Aventis and Wyeth (now part of Pfizer) companies (source: Macarron et al. 2011).....	3
1.3 High-throughput screening equipment (source: <a href="http://www.ingenesys.co.kr">http://www.ingenesys.co.kr</a> ).....	4
1.4 HTS plates with: 96 wells (8 rows × 12 columns), 384 wells (16 rows × 24 columns) and 1536 wells (32 rows × 48 columns), (source: Mayr and Fuerst 2008).....	5
1.5 A typical HTS plate layout (source: Malo et al. 2006) for a 96-well plate.....	10
1.6 An example of a decision tree where $c_1$ and $c_2$ are constants such that $c_1 > c_2$ .....	24
1.7 An example of a multilayer feed-forward network.....	25
2.1 Systematic error in experimental HTS data. Hit distribution surfaces for the McMaster (cases (a) and (b) - 1250 plates – Elowe et al. 2005) and Princeton (cases (c) and (d) - 164 plates – (Helm et al. 2003) Universities experimental HTS assays. Values deviating from the plate means for more than 2 standard deviations - cases (a) and (c), and for more than 3 standard deviations - cases (b) and (d) were selected as hits. The well, row and column positional effects are shown (the wells containing controls are not presented).....	34
2.2 Simulation 1, Plate Size: 96 wells – Cohen’s Kappa vs Error Size Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: (a) - (c): $\alpha = 0.01$ ; Second column: (d) - (f): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test. ....	44
2.3 Simulation 1, Plate Size: 384 wells – Cohen’s Kappa vs Error Size Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: (a) - (c): $\alpha = 0.01$ ; Second column: (d) - (f): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test. ....	45
2.4 Simulation 1, Plate Size: 1536 wells - Cohen’s Kappa vs Error Size. Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: (a) - (c): $\alpha = 0.01$ ; Second column: (d) - (f): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test. ....	46
2.5 Simulation 2, Plate Size: 96 wells, Cohen’s Kappa vs Hit Percentage. Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) -	

- (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test..... 48
- 2.6 Simulation 2, Plate Size: 384 wells, Cohen's Kappa vs Hit Percentage. Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test..... 49
- 2.7 Simulation 2, Plate Size: 1536 wells, Cohen's Kappa vs Hit Percentage. Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test..... 50
- 2.8 Intersections between the original set of hits (96 hits in total) and the sets of hits obtained after the application of the *B-score* (411 hits in total; the method was carried out only on the plates where systematic error was detected) and *Well correction* methods (102 hits in total) computed for *McMaster Test assay*. The  $\mu - 2.29SD$  hit selection threshold was used to select hits..... 54
- 2.9 Simulation 1, Plate Size: 96 wells – Sensitivity (True Positive Rate). Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (c):  $\alpha = 0.01$ ; Second column: cases (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test..... 57
- 2.10 Simulation 1, Plate Size: 384 wells – Sensitivity (True Positive Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (c):  $\alpha = 0.01$ ; Second column: cases (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test..... 58
- 2.11 Simulation 1, Plate Size: 1536 wells – Sensitivity (True Positive Rate). Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (c):  $\alpha = 0.01$ ; Second column: cases (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test..... 59
- 2.12 Simulation 1, Plate Size: 96 wells – Specificity (True Negative Rate). Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test..... 60
- 2.13 Simulation 1, Plate Size: 384 wells – Specificity (True Negative Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test..... 61
- 2.14 Simulation 1, Plate Size: 1536 wells – Specificity (True Negative Rate). Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test..... 62

2.15 Simulation 2, Plate Size: 96 wells - Sensitivity (True Positive Rate). Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (b): $\alpha = 0.01$ ; Second column: cases (c) - (d): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ ) $\chi^2$ goodness-of-fit test. ....	63
2.16 Simulation 2, Plate Size: 384 wells - Sensitivity (True Positive Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (b): $\alpha = 0.01$ ; Second column: cases (c) - (d): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ ) $\chi^2$ goodness-of-fit test.....	64
2.17 Simulation 2, Plate Size: 1536 wells - Sensitivity (True Positive Rate). Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (b): $\alpha = 0.01$ ; Second column: cases (c) - (d): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ ) $\chi^2$ goodness-of-fit test.....	65
2.18 Simulation 2, Plate Size: 96 wells - Specificity (True Negative Rate). Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e): $\alpha = 0.01$ ; Second column: cases (f) - (j): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ ) $\chi^2$ goodness-of-fit test.....	66
2.19 Simulation 2, Plate Size: 384 wells - Specificity (True Negative Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e): $\alpha = 0.01$ ; Second column: cases (f) - (j): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ ) $\chi^2$ goodness-of-fit test.....	67
2.20 Simulation 2, Plate Size: 1536 wells - Specificity (True Negative Rate). Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e): $\alpha = 0.01$ ; Second column: cases (f) - (j): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ ) $\chi^2$ goodness-of-fit test.....	68
2.21 Simulation 1, Plate Size: 96 wells – Success Rate. Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e): $\alpha = 0.01$ ; Second column: cases (f) - (j): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.....	69
2.22 Simulation 1, Plate Size: 384 wells – Success Rate. Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e): $\alpha = 0.01$ ; Second column: cases (f) - (j): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.....	70
2.23 Simulation 1, Plate Size: 1536 wells – Success Rate. Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e): $\alpha = 0.01$ ; Second column: cases (f) - (j): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.....	71
2.24 Simulation 2, Plate Size: 96 wells - Success Rate. Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e): $\alpha = 0.01$ ; Second column: cases (f) - (j): $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ ) $\chi^2$ goodness-of-fit test.....	72

- 2.25 Simulation 2, Plate Size: 384 wells - Success Rate. Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test..... 73
- 2.26 Simulation 2, Plate Size: 1536 wells - Success Rate. Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test..... 74
- 2.27 Data distribution before and after the application of the Discrete Fourier Transform (DFT) method. Data from one of the simulated 96-well plates before and after the application of Discrete Fourier Transform. The raw data followed a normal distribution and contained *random error only* (i.e., systematic error was not added). The raw data show agreement with the normal distribution, both graphically (case a) and by the Kolmogorov-Smirnov test (KS = 0.03, p = 0.5). However, after the application of Discrete Fourier Transform, the data deviate from normality as shown in the graph (case b) and by the Kolmogorov-Smirnov test (KS = 0.06, p = 0.0018).... 75
- 3.1 Hit maps showing the presence of positional effects in the McMaster 1250-plate assay (Elowe et al. 2005) - (a) whole assay background surface, (b) plate 1036 measurements; and in the Princeton 164-plate assay (Helm et al. 2003) - (c) whole assay background surface, (d) plate 144 measurements. Color intensity is proportional to the compounds' signal levels (higher signals - potential target inhibitors, are shown in red)..... 80
- 3.2 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 96-well plate assays estimated under the condition that at most two columns and two rows of each plate were affected by systematic error. Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction ( $\circ$ ), B-score ( $\Delta$ ), MEA ( $\square$ ), t-test and MEA ( $\diamond$ ), SMP (+), t-test, and SMP ( $\times$ )..... 92
- 3.3 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 384-well plate assays estimated under the condition that at most four columns and four rows of each plate were affected by systematic error. Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction ( $\circ$ ), B-score ( $\Delta$ ), MEA ( $\square$ ), t-test and MEA ( $\diamond$ ), SMP (+), t-test, and SMP ( $\times$ )..... 93
- 3.4 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 1536-well plate assays estimated under the condition that at most eight columns and two eight of each plate were affected by systematic error. Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the

- results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction ( $\circ$ ), B-score ( $\Delta$ ), MEA ( $\square$ ), t-test and MEA ( $\diamond$ ), SMP (+), t-test, and SMP ( $\times$ )..... 94
- 3.5 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 96-well plate assays estimated under the condition that systematic error affects up to 50% of rows and columns of each plate (the exact number of affected rows and columns on each plate was determinate randomly according to the uniform distribution). Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of  $1.2SD$ . Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction ( $\circ$ ), B-score ( $\Delta$ ), MEA ( $\square$ ), t-test and MEA ( $\diamond$ ), PMP (+), t-test, and PMP ( $\times$ )..... 95
- 3.6 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 384-well plate assays estimated under the condition that systematic error affects up to 50% of rows and columns of each plate (the exact number of affected rows and columns on each plate was determinate randomly according to the uniform distribution). Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of  $1.2SD$ . Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction ( $\circ$ ), B-score ( $\Delta$ ), MEA ( $\square$ ), t-test and MEA ( $\diamond$ ), PMP (+), t-test, and PMP ( $\times$ )..... 96
- 3.7 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 1536-well plate assays estimated under the condition that systematic error affects up to 50% of rows and columns of each plate (the exact number of affected rows and columns on each plate was determinate randomly according to the uniform distribution). Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of  $1.2SD$ . Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction ( $\circ$ ), B-score ( $\Delta$ ), MEA ( $\square$ ), t-test and MEA ( $\diamond$ ), PMP (+), t-test, and PMP ( $\times$ )..... 97
- 3.8 Hit distribution surfaces of the McMaster *Test dataset* for the hit selection thresholds  $\mu - SD$  (cases a, c, e, g, i and k) and  $\mu - 2.29SD$  (cases b, d, f, h, j and l) obtained for: the raw (i.e., uncorrected) data (a, b), and the data corrected by B-score (c, d), MEA (e, f), SMP (g, h), Well Correction + MEA (i, j), and Well Correction + SMP (k, l). .. 107
- 4.1 The relationship between the HTS measurement values (varying from 0 to 200) and the molecular weight (varying from 200 to 600) depicted at 6 different levels of the *ClogP* descriptor. There is virtually no difference in the six presented patterns. That suggests that *ClogP* does not have any influence over the way molecular weight is related to the HTS measurements..... 117
- 4.2 Polynomial RDA correlation biplot for the McMaster *Test dataset*. Triangles represent the three types of samples (*D-R Hits* - the consensus hits showing good dose-response behavior; *No D-R Hits* - the consensus hits not showing good dose-response behavior; *No-Hits* - samples that are not hits). Dashed arrows represent two

- binary response variables *Hit/No-Hit* and *D-R*. Solid arrows represent the chemical descriptors (10 descriptors from Table 4.1 plus 2 extra descriptors  $Hdon^2$  and  $Hdon \times Hacc$  that show the biggest correlations with *Hit/No-Hit* and *D-R* variables, respectively). The lengths of the chemical descriptor arrows were multiplied by 10; this does not change the interpretation of the diagram. .... 119
- 4.3 ROC curves for four different group sizes used for training and test: 48 hits + 100 (case a), 500 (case b), 1000 (case c) and 2000 (case d) no hits. The abscissa axis represents (1-*Specificity*), and the ordinate axis represents *Sensitivity*. The results of the decision tree method are depicted by squares and those of the neural network method by triangles. .... 121
- A.1 *HTS Helper*, Windows Forms executable (version 1.0, March 22<sup>nd</sup>, 2012)..... 148

## LIST OF TABLES

Table	Page
1.1 Categorization of an HTS assay quality depending on the value of <i>Z-factor</i> .....	13
2.1 Five types of HTS datasets containing different kinds of systematic and/or random error generated and tested in this study. ....	37
2.2 Number of rows, columns and plates (where at least one row or column contains systematic error) of McMaster <i>Test assay</i> in which the t-test reported the presence of systematic error, depending on the $\alpha$ parameter. Only 8 rows and 10 columns of McMaster <i>Test assay</i> were examined because the first and twelfth columns of the (8 by 12) plates were used for controls.....	51
2.3 Number of hits selected in McMaster <i>Test assay</i> for the $\mu-3SD$ threshold after the application of the B-score correction, depending on the $\alpha$ parameter. The t-test was carried out to detect systematic error.....	51
2.4 Number of hits selected in McMaster <i>Test assay</i> for the $\mu-2.29SD$ threshold (i.e., threshold used by the McMaster competition organizers to select the 96 original average hits) after the application of the B-score correction, depending on the $\alpha$ parameter. The t-test was carried out to detect systematic error. ....	52
2.5 Number of hits selected in McMaster <i>Test assay</i> for the $\mu-3SD$ and $\mu-2.29SD$ thresholds after the application of the Well Correction method.....	53
3.1 Number of hits selected by the six data correction methods for the McMaster <i>Test dataset</i> . The hit selection threshold of $\mu-2.29SD$ was used.....	99
3.2 Hit distribution of the raw McMaster dataset computed for the $\mu-SD$ threshold (mean value of hits per well is 37.69 and standard deviation is 24.27). The $\chi^2$ goodness-of-fit test provided the following results ( $\chi^2$ value = 1234.23, critical value 111.14 for $\alpha = 0.01$ ; $H_0$ is rejected and data correction is recommended).....	100
3.3 Hit distribution of the raw McMaster dataset computed for the $\mu-2.29SD$ threshold (mean value of hits per well is 1.20 and standard deviation is 1.19). The $\chi^2$ goodness-of-fit test provided the following results ( $\chi^2$ value = 94.0, critical value 111.14 for $\alpha = 0.01$ ; $H_0$ is not rejected and data correction is optional).....	100
3.4 Hit distribution of the McMaster dataset after applying B-score normalization and computed for the $\mu-SD$ threshold (mean value of hits per well is 39.48 and standard deviation is 14.67). The $\chi^2$ goodness-of-fit test provided the following results ( $\chi^2$ value = 430.96, critical value 111.14 for $\alpha = 0.01$ ; $H_0$ is rejected and data correction is recommended). ....	101
3.5 Hit distribution of the McMaster dataset after applying B-score normalization and computed for the $\mu-2.29SD$ threshold (mean value of hits per well is 2.33 and	

- standard deviation is 1.89). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 121.10, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended). ..... 101
- 3.6 Hit distribution of the McMaster dataset after applying Matrix Error Amendment (MEA) method and computed for the  $\mu$ -SD threshold (mean value of hits per well is 38.48 and standard deviation is 23.96). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 1178.53, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended). ..... 102
- 3.7 Hit distribution of the McMaster dataset after applying Matrix Error Amendment (MEA) method and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.25 and standard deviation is 1.24). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 96.80, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional). ..... 102
- 3.8 Hit distribution of the McMaster dataset after applying Partial Mean Polish (PMP) method and computed for the  $\mu$ -SD threshold (mean value of hits per well is 39.90 and standard deviation is 22.41). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 994.27, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended). ..... 103
- 3.9 Hit distribution of the McMaster dataset after applying Partial Mean Polish (PMP) method and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.44 and standard deviation is 1.32). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 95.78, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional). ..... 103
- 3.10 Hit distribution of the McMaster dataset after applying Well Correction followed by Matrix Error Amendment (MEA) method and computed for the  $\mu$ -SD threshold (mean value of hits per well is 35.48 and standard deviation is 9.55). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 203.18, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended). ..... 104
- 3.11 Hit distribution of the McMaster dataset after applying Well Correction followed by Matrix Error Amendment (MEA) method and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.36 and standard deviation is 1.37). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 108.98, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional). ..... 104
- 3.12 Hit distribution of the McMaster dataset after applying Well Correction followed by Partial Mean Polish (PMP) method and computed for the  $\mu$ -SD threshold (mean value of hits per well is 35.64 and standard deviation is 9.47). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 198.68, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended). ..... 105
- 3.13 Hit distribution of the McMaster dataset after applying Well Correction followed by Partial Mean Polish (PMP) method and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.36 and standard deviation is 1.37). The  $\chi^2$  goodness-of-fit

test provided the following results ( $\chi^2$ value = 108.98, critical value 111.14 for $\alpha = 0.01$ ; $H_0$ is not rejected and data correction is optional).....	105
4.1 Average values of the measurements and 10 chemical descriptors for four types of samples (consensus hits – 42 samples, average hits – 96 samples, hits having well-behaved dose-response curves – 26 samples and the whole dataset – 50,000 samples) from the McMaster University <i>Test dataset</i> . In bold, the values of chemical descriptors showing important variation. ....	116
4.2 Results obtained after applying the new probability-based hit selection method on the raw McMaster <i>Test dataset</i> . Results are shown for different values of the probability-based hit selection thresholds, $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list. ....	126
4.3 Results obtained after applying the new probability-based hit selection method on the McMaster <i>Test dataset</i> corrected by the Matrix Error Amendment method. Results are shown for different values of the probability-based hit selection thresholds, $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list. ....	127
4.4 Results obtained after applying the new probability-based hit selection method on the McMaster <i>Test dataset</i> corrected by the Partial Mean Polish method. Results are shown for different values of the probability-based hit selection thresholds, $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list. ....	128
4.5 Results obtained after applying the new probability-based hit selection method on the McMaster <i>Test dataset</i> corrected by the Well Correction followed by Matrix Error Amendment methods. Results are shown for different values of the probability-based hit selection thresholds, $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list. ....	129
4.6 Results obtained after applying the new probability-based hit selection method on the McMaster <i>Test dataset</i> corrected by the Well Correction and Partial Mean Polish methods. Results are shown for different values of the probability-based hit selection thresholds, $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list. ....	130
4.7 The 96 average hits from the McMaster Test assay with their MAC IDs, the first (M1) and the second (M2) measurement values, hit probability computed from theoretical distribution, and false positive (FPch) and false negative (FNcr) change rates also computed from theoretical distributions. The consensus hit are italicized....	139
A.1 HTShelper Command Line Parameters Description .....	153

## LIST OF ABBREVIATIONS AND ACRONYMS

ANN	Artificial neural networks
DFT	Discrete Fourier Transform
FN	False negative
FP	False positive
HTS	High-throughput screening
kNN	k-nearest neighbors
MAD	Median absolute deviation
MEA	Matrix Error Amendment
PMP	Partial Mean Polish
RDA	Redundancy Analysis
SAR	Structure-activity relationship
Se	Sensitivity
Sp	Specificity
SVM	Support vector machines
TN	True negative
TP	True positive

## RÉSUMÉ

Durant le criblage à haut débit (High-throughput screening, HTS), la première étape dans la découverte de médicaments, le niveau d'activité de milliers de composés chimiques est mesuré afin d'identifier parmi eux les candidats potentiels pour devenir futurs médicaments (i.e., hits). Un grand nombre de facteurs environnementaux et procéduraux peut affecter négativement le processus de criblage en introduisant des erreurs systématiques dans les mesures obtenues. Les erreurs systématiques ont le potentiel de modifier de manière significative les résultats de la sélection des hits, produisant ainsi un grand nombre de faux positifs et de faux négatifs. Des méthodes de correction des données HTS ont été développées afin de modifier les données reçues du criblage et compenser pour l'effet négatifs que les erreurs systématiques ont sur ces données (Heyse 2002, Brideau et al. 2003, Heuer et al. 2005, Kevorkov and Makarenkov 2005, Makarenkov et al. 2006, Malo et al. 2006, Makarenkov et al. 2007).

Dans cette thèse, nous évaluons d'abord l'applicabilité de plusieurs méthodes statistiques servant à détecter la présence d'erreurs systématiques dans les données HTS expérimentales, incluant le  $\chi^2$  goodness-of-fit test, le t-test et le test de Kolmogorov-Smirnov précédé par la méthode de Transformation de Fourier. Nous montrons premièrement que la détection d'erreurs systématiques dans les données HTS brutes est réalisable, de même qu'il est également possible de déterminer l'emplacement exact (lignes, colonnes et plateau) des erreurs systématiques de l'essai. Nous recommandons d'utiliser une version spécialisée du t-test pour détecter l'erreur systématique avant la sélection de hits afin de déterminer si une correction d'erreur est nécessaire ou non.

Typiquement, les erreurs systématiques affectent seulement quelques lignes ou colonnes, sur certains, mais pas sur tous les plateaux de l'essai. Toutes les méthodes de correction d'erreur existantes ont été conçues pour modifier toutes les données du plateau sur lequel elles sont appliquées et, dans certains cas, même toutes les données de l'essai. Ainsi, lorsqu'elles sont appliquées, les méthodes existantes modifient non seulement les mesures expérimentales biaisées par l'erreur systématique, mais aussi de nombreuses données correctes. Dans ce contexte, nous proposons deux nouvelles méthodes de correction d'erreur systématique performantes qui sont conçues pour modifier seulement des lignes et des colonnes sélectionnées d'un plateau donné, i.e., celles où la présence d'une erreur systématique a été confirmée. Après la correction, les mesures corrigées restent comparables avec les valeurs non modifiées du plateau donné et celles de tout l'essai. Les deux nouvelles méthodes s'appuient sur les résultats d'un test de détection d'erreur pour déterminer quelles lignes et colonnes de chaque plateau de l'essai doivent être corrigées. Une procédure générale pour la correction des données de criblage à haut débit a aussi été suggérée.

Les méthodes actuelles de sélection des hits en criblage à haut débit ne permettent généralement pas d'évaluer la fiabilité des résultats obtenus. Dans cette thèse, nous décrivons une méthodologie permettant d'estimer la probabilité de chaque composé chimique d'être un hit dans le cas où l'essai contient plus qu'un seul réplicat. En utilisant la nouvelle méthodologie, nous définissons une nouvelle procédure de sélection de hits basée sur la probabilité qui permet d'estimer un niveau de confiance caractérisant chaque hit. En plus, de

nouvelles mesures servant à estimer des taux de changement de faux positifs et de faux négatifs, en fonction du nombre de réplifications de l'essai, ont été proposées.

En outre, nous étudions la possibilité de définir des modèles statistiques précis pour la prédiction informatique des mesures HTS. Remarquons que le processus de criblage expérimental est très coûteux. Un criblage virtuel, *in silico*, pourrait mener à une baisse importante de coûts. Nous nous sommes concentrés sur la recherche de relations entre les mesures HTS expérimentales et un groupe de descripteurs chimiques caractérisant les composés chimiques considérés. Nous avons effectué l'analyse de redondance polynomiale (Polynomial Redundancy Analysis) pour prouver l'existence de ces relations. En même temps, nous avons appliqué deux méthodes d'apprentissage machine, réseaux de neurones et arbres de décision, pour tester leur capacité de prédiction des résultats de criblage expérimentaux.

Mots-clés : criblage à haut débit (HTS), modélisation statistique, modélisation prédictive, erreur systématique, méthodes de correction d'erreur, méthodes d'apprentissage automatique

## ABSTRACT

During the high-throughput screening (HTS), an early step in the drug discovery process, the activity levels of thousands of chemical compounds are measured in order to identify the potential drug candidates, called *hits*. A number of environmental and procedural factors can affect negatively the HTS process, introducing systematic error in the obtained experimental measurements. Systematic error has the potential to alter significantly the outcome of the hit selection procedure, thus generating eventual false positives and false negatives. A number of systematic error correction methods have been developed for compensating for the effect of this error in experimental HTS (Heyse 2002, Brideau et al. 2003, Kevorkov and Makarenkov 2005, Heuer et al. 2005, Makarenkov et al. 2006, Malo et al. 2006, Makarenkov et al. 2007).

In this thesis, we first evaluate the applicability of several statistical procedures for assessing the presence of systematic error in experimental HTS data, including the  $\chi^2$  goodness-of-fit test, Student's t-test and Kolmogorov-Smirnov test preceded by the Discrete Fourier Transform method. We show that the detection of systematic error in raw HTS data is achievable, and that it is also possible to determine the most probable assay locations (rows, columns and plates) affected by systematic error. We conclude that the t-test should be preferably used prior to the hit selection in order to determine whether an error correction is required or not.

Typically, systematic error affects only a few rows and/or columns of the given plate (Brideau et al. 2003, Makarenkov et al. 2007). All the existing error correction methods have been designed to modify all the data on the plate on which they are applied, and in some cases, even all the data in the assay. Thus the existing methods modify not only the error-biased measurements, but also the error-free measurements. We propose two new error correction methods that are designed to modify (i.e., correct) only the measurements of the selected rows and columns where the presence of systematic error has been confirmed. After the correction, the modified measurements remain comparable with the unmodified ones within the given plate and across the entire assay. The two new methods rely on the results from an error detection test to determine which plates, rows and columns should be corrected. Our simulations showed that the two proposed methods generally outperform the popular B-score procedure (Brideau et al. 2003). We also describe a general correction procedure allowing one to correct both plate-specific and screen-specific systematic error.

The hit selection methods used in the modern HTS do not allow for assessing the reliability of the selected hits. In this thesis, we describe a methodology for estimating the probability of each compound to be a hit when the assay contains more than one replicate. Using the new methodology, we define a new probability-based hit selection procedure that allows one to estimate the probability of each considered compound to be a hit based on the available replicate measurements. Furthermore, new measures for computing the false positive and false negative change rates depending on the number of experimental assay replicates (i.e., how these two rates would change if an additional screen replicate will be performed), are introduced.

Further, we investigate the possibility of developing accurate statistical models for computer prediction of HTS measurements. Note that the experimental HTS process is very

expensive. The use of virtual, *in silico*, HTS instead of experimental HTS could lead to an important cost decline. We first focus on finding relationships between experimental HTS measurements and a group of descriptors characterizing the given chemical compounds. We carried out Polynomial Redundancy Analysis (Polynomial RDA) to prove the existence of such relationships. Second, we evaluate the applicability of two machine learning methods, neural networks and decision trees, for predicting the hit/non-hit outcomes for selected compounds, based on the values of their chemical descriptors.

Keywords: high-throughput screening (HTS), statistical modeling, predictive modeling, systematic error, error correction methods, machine learning methods

## CHAPTER I

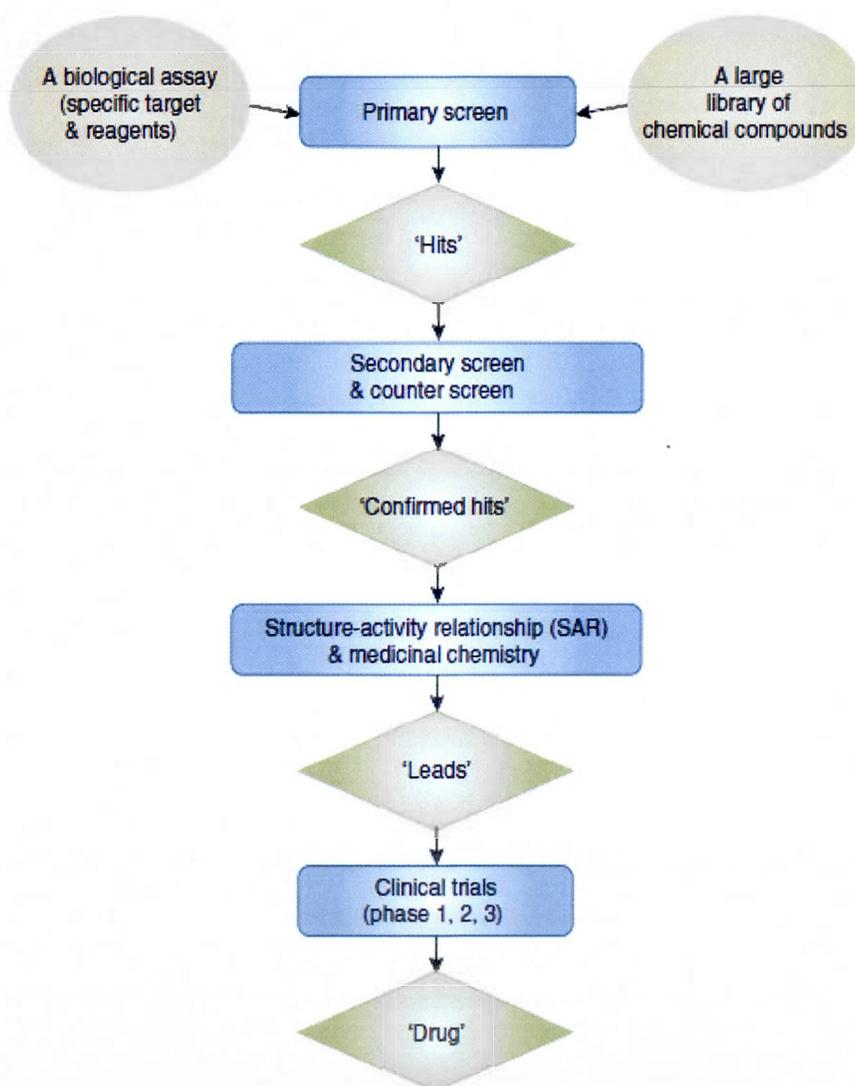
### INTRODUCTION

#### **1.1 High-Throughput Screening**

High-throughput screening (HTS) is a large-scale highly automated process used widely within the pharmaceutical industry (Broach and Thorner 1996, Carnero 2006, Malo et al. 2006, Janzen and Bernasconi 2009). It is employed in the early stages of the drug discovery process during which a huge number of chemical compounds are screened and the compounds activity against a specific target measured. The goal of HTS is to identify among all tested compounds those showing promising “drug-like” properties. As specified in Sirois et al. (2005) and Malo et al. (2006), the drug discovery could be described as a multi-step process (Figure 1.1):

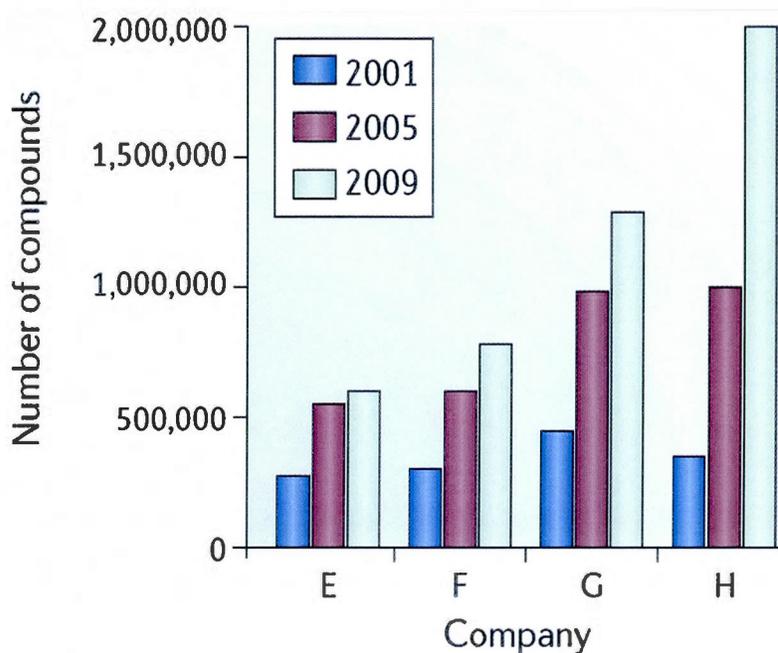
1. Hit identification: target selection (i.e., researchers typically focus on enzymes or proteins that are essential to the survival of an infectious agent), assay preparation, primary screening;
2. Hit verification: re-testing, secondary screening and dose response curve generation;
3. Lead identification: structure-activity relationships (SAR) analysis, establishing and confirming the mechanism of action;
4. Clinical studies: drug effectiveness evaluation, drug-to-drug interactions, safety assessment studies;
5. Regulatory approval for a new drug.

High-throughput screening is the backbone of the first step of the drug discovery process. At that stage, thousands of chemical compounds are tested in an initial primary screen. The identified compounds are marked for follow-up in the second step of the process when HTS is employed again for performing secondary screens of a group of pre-selected (i.e., hit) compounds (e.g., 1% of the most active compounds from the primary screen, Nelson and Yingling 2004). Typically, at least duplicate compound measurements are recommended (Malo et al. 2006).



**Figure 1.1 Drug development process (source: Malo et al. 2006)**

HTS is a relatively new technology that continues to advance at fast pace. The recent progress in computer technologies and robotic automation reduced significantly the cost of operating high-throughput screening facility and made the technology feasible for many small and moderate-sized organizations. The price of experimental screening per unit also dropped significantly bringing a considerable increase in the number of tested compounds. Figure 1.2 (Macarron et al. 2011) shows that the use of HTS in the four selected pharmaceutical companies more than doubled for three of them and increased by 10 times for the fourth company during the period from 2001 to 2009.

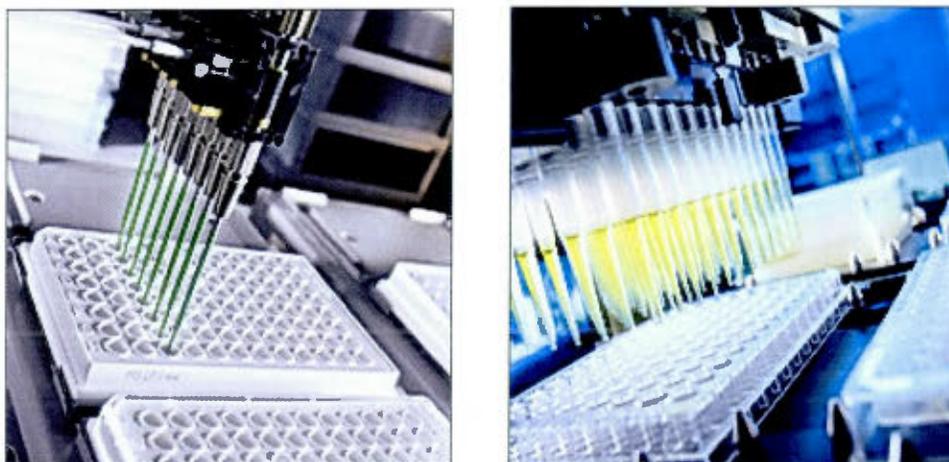


**Figure 1.2 Size of corporate screening collections over time. This figure shows that the screening collections of four pharmaceutical companies in 2001 differ dramatically from those in 2009. Data taken from GlaxoSmithKline [E], Novartis [F], Sanofi-Aventis [G] and Wyeth [H] (now part of Pfizer) companies (source: Macarron et al. 2011)**

The steadily decreasing prices, as well as a number of governmental programs and initiatives (Austin 2008) allowed HTS to penetrate rapidly into the academic settings (Stein 2003, Verkman 2004, Kaiser 2008, Silber 2010). In May 2012, the website of the US Society for Laboratory Automation and Screening listed 93 academic institutions that have

established HTS facilities (SLAS 2012), while the Molecular Libraries Initiative has generated a collection of more than 360,000 compounds in a screening library for academic investigators (MLP 2012).

The modern drug discovery is no more exclusively based on chemistry. It is a product of interdisciplinary cooperation of engineers, chemists, biologists and statisticians. Massive libraries of millions of compounds have been created using combinatorial chemical synthesis and are used as a drug-candidate base. The contemporary HTS technology is an almost fully automated process. Robots are used in all steps of the process. They retrieve compounds from the libraries, combine them with the selected biological target, convey the compound solutions through the HTS readers and record the measured activity levels. The recent dramatic improvements in miniaturization, low-volume liquid-handling and multimodal readers increased drastically the throughput of the screening process. Figure 1.3 depicts two typical high-throughput screening systems.



**Figure 1.3 High-throughput screening equipment (source: <http://www.ingenesis.co.kr>)**

The contemporary HTS systems are capable of screening more than 100,000 compounds per day (Mayr and Fuerst 2008). At the same time, the screening-collection sizes constantly increase and are expected to grow, given the fact that the size of the “drug like” chemical space is estimated to be greater than  $1 \times 10^{30}$  compounds (Fink and Reymond 2007). High-throughput screening involves many steps, such as target selection and characterization, assay development, reagent preparation and compound screening. Each of those steps adds

cost proportional to the number of the compounds in the assay. Thus, despite the technological advances, HTS remains an expensive technology.

High-throughput screening assays are organized as a sequence of *plates*. HTS plate is a small container, usually disposable and made of plastic. It includes a grid of small, open concavities called *wells*. During the assay preparation, every compound of interest is placed in a separate well.



**Figure 1.4 HTS plates with: 96 wells (8 rows  $\times$  12 columns), 384 wells (16 rows  $\times$  24 columns) and 1536 wells (32 rows  $\times$  48 columns), (source: Mayr and Fuerst 2008)**

A typical HTS plate consists of 96 wells arranged in 8 rows and 12 columns. Along with the tendency of miniaturization in HTS, plates with 384 and 1536 were also created (see Figure 1.5). The plates with more wells usually have the same dimensions but contain smaller wells. During the screening process, all the wells on the plate are examined and the activity level of their content is registered as a single numeric value. Thus, the results of screening one plate can be represented as a matrix of numeric values where the measurements are arranged in the same layout as the compounds on the plate.

## 1.2 HTS experimental errors

Since the appearance of the first HTS systems, 15-20 years ago, a lot of efforts have been made to improve the speed, the capacity and the precision of HTS equipment (Macarron 2006, Pereira and Williams 2007, Houston et al. 2008). The contemporary HTS systems are capable to rapidly handle great number of compounds while detecting even slightest

differences in the compounds' activity levels. That remarkable sensitivity, otherwise an asset, makes the HTS process prone to errors. Many environmental factors, including variations in the electricity, temperature, air flow and light intensity, may affect HTS readers (Makarenkov et al. 2007). Equipment malfunctions, such as robotic failures, poor pipette delivery, glass fogging or needle clogging, may also cause erroneous readings. Assay miniaturization and especially work with very small liquid volumes can cause fast and significant concentration changes due to reagent evaporation. Therefore, procedural factors such as differences in the time compounds await to be processed create inequalities among the experimental measurements. In the context of HTS, an experimental error (or simply an error) denotes the case when the measurement obtained during the screening does not reflect the real compound activity either because of a failure during the reading procedure or because the conditions at which the activity was measured differed from the planned ones. Systematic error affects the HTS data by either over- or underestimating the measurements of some of the compounds. With the appearance of the second generation HTS systems in 2001 (Mayr and Fuerst 2008), a strong focus was put on the quality of the selected hits, i.e., to ensure lower readout artifacts. HTS laboratories have pioneered and implemented rigorous quality assurance methods (Macarron et al. 2011), such as liquid handler and reader performance monitoring (Taylor et al. 2002), Z trend monitoring (Zhang et al. 1999) and regular usage of pharmacological standards (Coma et al. 2009). Despite the remarkable quality improvements, the experimental error remains a major hindrance in high-throughput screening, and handling this error becomes even of a greater importance as our dependence on the technology increases.

### 1.3 Hit selection

Hit selection is the last step of the high-throughput screening process. At this step, the compounds with the best activity values are selected for further investigation. The compounds selected in such a way are called *hits* (Malo et al. 2006). The hit selection is based on comparability of the activity measurements. Depending on the type of the screened assay the researches may be interested either in the compounds showing the highest activation properties (activation assays - the goal here is to activate the given target) or the highest inhibition properties (inhibition assays - the goal here is to inhibit the given target).

There exist four different strategies for selecting hits (Malo et al. 2006):

1. Hits are selected as a fixed percentage of all compounds, those having the highest activity levels (for example: top 1%). This strategy, and especially the number of selected hits, is dictated by resource availability and other project constraints. This method is not usually recommended because a fixed, in advance, number of compounds does not usually reflect the number of really active compounds in the assay;
2. Hits are identified as compounds whose activity measurements exceed some fixed threshold. The hit-cutting threshold can be specified as an absolute value or as a percentage of the maximum possible activity level. In order to achieve quality results using such a threshold, the advanced knowledge about the expected activity levels is required;
3. Similar to the second strategy, hits are selected to exceed a hit-cutting threshold, but the threshold is calculated to reflect the specifics of the actual data. This threshold is usually expressed in terms of the mean value  $\mu$  (the median can be also used) and the standard deviation  $\sigma$  of the measurements, with the most widely used threshold of  $\mu-3\sigma$  (inhibition assays) and  $\mu+3\sigma$  (activation assays). This strategy can be applied globally for the whole assay when the mean and standard deviation are calculated using all measurements of the assay, or alternatively, on the plate-by-plate basis when the mean and the standard deviation are calculated separately for each plate using only the associated plate's measurements.
4. Statistical testing with replicates was recommended by Malo et al. (2006) for a better hit identification. More specifically, to get around the small sample size problem that is known to produce serious problems with the t-test, Malo et al. (2006) recommended using "shrinkage tests" (see also Allison et al. 2006). They provided a specific example of one particular shrinkage test – the Random Variance Model.

#### 1.4 False positive and false negative hits

The presence of error in experimental HTS data may affect the accuracy of the hit selection campaign. Working with the measurements that do not correctly represent the real compound activity levels may cause the situation where some compounds are mistakenly selected as hits: this situation is known as Type I error and the selected compounds are called *false positives (FP)*. It is also possible that some active compounds are overlooked, i.e., they have not been selected as hits: this situation is known as Type II error and the affected compounds are called *false negatives (FN)*. All the active compounds that are correctly identified as hits are defined as *true positives (TP)* and the correctly identified inactive compounds are defined as *true negatives (TN)*.

#### 1.5 Sensitivity and specificity of a statistical model

With its ability to separate the compounds into two groups, hits and non-hits, the high-throughput screening process constitutes a binary classifier. *Sensitivity (Se)* and *Specificity (Sp)* are two statistical measures used to evaluate the performance of such classifiers. The sensitivity of the model can be defined as follows:

$$Se = \frac{TP}{TP + FN}$$

In the context of HTS, Sensitivity represents the proportion of the active compounds that were selected as hits. The sensitivity value of 1 (or 100% if expressed as a percentage) indicates that all active compounds were successfully identified as hits, whereas the value of 0 means that all active compounds were overlooked as false negatives.

The value of Specificity can be calculated using the following formula:

$$Sp = \frac{TN}{TN + FP}$$

In the context of HTS, Specificity represents the proportion of the non-hit compounds that were not selected as hits. Specificity value of 1 (or 100% if expressed as a percentage) indicates that no inactive compounds were incorrectly identified as hits.

### 1.6 Types of experimental error

Momentary variations in the HTS experimental conditions introduce random errors in the screening data that affect single or limited number of compounds only. Random error unpredictably lowers or raises the values of some of the screened measurements. Random errors can affect as well the hit selection process (Kevorkov and Makarenkov 2005). If a further increase of the HTS precision is required, in regards to random error, then replicated measurements should be used, i.e., multiple instances of the compounds should be screened (Lee et al. 2000, Nadon and Shoemaker 2002). It is also important to note that this type of error is usually very difficult to detect in HTS using standard quality control and performance monitoring techniques.

Another type of error, systematic error, can as well be present in HTS data. Systematic error biases experimental HTS results by systematically over- or underestimating the compounds true activity levels and affects multiple compounds of the given plate or given assay. Systematic error is usually location dependent (Brideau et al. 2003). It generates repeatable local artifacts that may affect only a single plate, a group of several plates or even all plates of the assay. Within the plates, the compounds affected by systematic error are usually located in the same row or column, very often at the edges of the given plate - situation known as *edge effect* (Cheneau et al. 2003, Iredale et al. 2005, Carralot et al. 2012). It is also possible that systematic error affects only the compounds located at the specific well locations on all the plates or majority of plates across the assay (Makarenkov et al. 2007).

Systematic error has the potential to affect significantly the hit selection process, thus resulting in the generation of false positive and false negative hits (Kevorkov and Makarenkov 2005, Makarenkov et al. 2007). Therefore, eliminating or at least reducing the impact of systematic error on experimental HTS data was recognized as one of the major goals for achieving reliable, high quality high-throughput screening results (Brideau et al. 2003, Makarenkov et al. 2007). Several specialized statistical data correction methods have

been proposed for correcting different types of systematic errors (Brideau et al. 2003, Kevorkov and Makarenkov 2005, Makarenkov et al. 2007, Malo et al. 2010, Carralot et al. 2012).

### 1.7 Within-plate HTS controls

The controls are substances with stable and well-known activity levels for the specific assay. They are placed in separate wells within plates and then processed and measured as regular compounds of interest. Controls with different reference activity levels can be used. The most commonly used controls are *positive controls*, with the observed high activity effects, and *negative controls*, with the observed low activity effects (or no activity at all). It is also possible that some wells of the plate are left empty and are used as negative controls (Figure 1.5). It is a common practice that the controls are placed in the first and in the last columns on the plates while the regular compounds are located in the inner wells. Figure 1.5 shows a typical 96-well HTS plate layout with 80 compounds and the first and the last columns occupied by control.

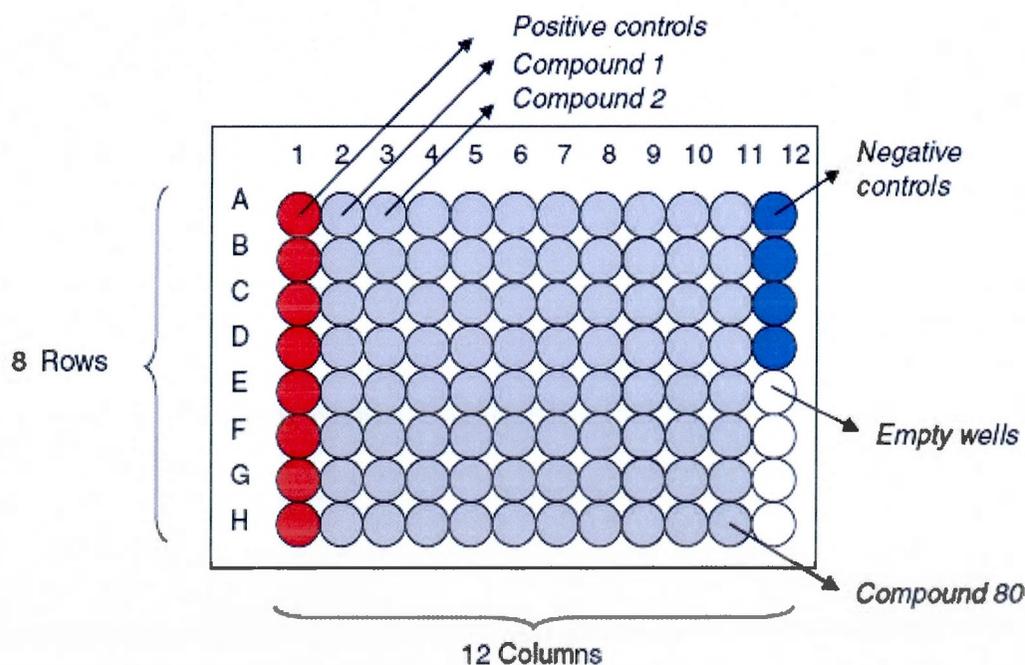


Figure 1.5 A typical HTS plate layout (source: Malo et al. 2006) for a 96-well plate

The within-plate controls can be used for both quality control and data normalization. They are most commonly used for the measurements normalization needed to account for the plate-to-plate differences throughout the assay. The values of the controls are often omitted when HTS results are presented. For example, the screening result for the HTS plate on Figure 1.5 are usually reported as for a plate that has wells arranged in 8 rows and 10 columns containing regular compounds only.

## 1.8 Systematic error correction and data normalization

Normalization of raw HTS data allows one to remove systematic plate-to-plate variations and makes the results comparable across plates. Here we present the three most popular normalization methods used in HTS:

### 1.8.1 Percent of control

*Percent of control* is a simple normalization procedure that relies on the presence of positive controls in every plate. It allows for elimination of the plate-to-plate environmental differences by aligning the measurements of the positive controls on the plates. As the method name suggests, the compounds measurements are reported as a percentage of the control measurements. The normalized measurements are calculated as follows:  $\hat{x}_{ij} = \frac{x_{ij}}{\mu_{pos}}$ , where  $x_{ij}$  is the raw measurement of the compound in well  $(i, j)$  of the plate,  $\hat{x}_{ij}$  is the new normalized value, and  $\mu_{pos}$  is the mean of the positive controls.

### 1.8.2 Normalized percent inhibition

*Normalized percent inhibition* is another normalization procedure that relies on the presence of both positive and negative controls in every plate. It allows for elimination of the plate-to-plate environmental differences by aligning the measurements of both the positive and negative controls on the plates. The normalized measurements are calculated using the

following formula:  $\hat{x}_{ij} = \frac{x_{ij} - \mu_{neg}}{\mu_{pos} - \mu_{neg}}$ , where  $x_{ij}$  is the raw measurement of the compound in well  $(i, j)$  on the plate,  $\hat{x}_{ij}$  is the new normalized value,  $\mu_{pos}$  is the mean of positive controls and  $\mu_{neg}$  is the mean of the negative controls.

### 1.8.3 Z-score

*Z-score* is a well known statistical normalization procedure that does not require the presence of controls. It ensures comparability of the measurements on different plates by aligning the mean values of the measurements of the plates and by eliminating the plate-to-plate variations. The normalization is carried out using the following formula:  $\hat{x}_{ij} = \frac{x_{ij} - \mu}{\sigma}$ , where  $x_{ij}$  is the raw measurement of the compound in well  $(i, j)$  on the plate,  $\hat{x}_{ij}$  is the new normalized value,  $\mu$  is the mean of all the measurements within the plate, and  $\sigma$  is the standard deviation of all the measurements within the plate. After the Z-score normalization, the plate's data are zero-centered, i.e., their mean value is 0, and their standard deviation is 1.

### 1.8.4 Assay quality and validation

A lot of efforts were put for validating and improving quality of experimental high-throughput screening assays. Zhang et al. (1999) explored the criteria for evaluating the suitability of an HTS assay for hit identification. The latter authors argued that the classical approach for quality assessment by examining the signal-to-noise (S/N) ratio and signal-to-background (S/B) ratio is difficult to apply in the context of experimental HTS. Zhang and colleagues defined a new screening coefficient called *Z-factor*, which reflects both the assay signal dynamic range and the data variation. This makes it suitable for comparison and evaluation of the assays quality, including assay validation. Zhang et al. (1999) defined *Z-factor* as follows:

$$Z = 1 - \frac{3(\sigma_s - \sigma_c)}{|\mu_s - \mu_c|}$$

where  $\mu_s$  is the mean value of the screened data,  $\mu_c$  is the mean value of the controls,  $\sigma_s$  is the standard deviation of the screened data and  $\sigma_c$  is the standard deviation of the controls. These authors stated that in the case of activation assays, positive controls should be used, whereas in the case of inhibition assays, negative controls should be employed. In the same study, Zhang et al. (1999) also defined the *Z'-factor* coefficient that can be calculated as follows using the control data only:

$$Z' = 1 - \frac{3(\sigma_{pos} - \sigma_{neg})}{|\mu_{pos} - \mu_{neg}|},$$

where  $\mu_{pos}$  is the mean value of the positive controls,  $\mu_{neg}$  is the mean value of the negative controls,  $\sigma_{pos}$  is the standard deviation of the positive controls and  $\sigma_{neg}$  is the standard deviation of the negative controls. Zhang et al. (1999) stated that *Z-factor* can be used for evaluating the quality and the performance of any HTS assay, while *Z'-factor* can be used to assess the quality of the assay design and development process. Zhang and colleagues also provided directions for how *Z-factor* can be used to assess the quality of an HTS assay (see Table 1.1).

<i>Z-factor</i>	<i>Screening</i>
1	An ideal assay. <i>Z-factors</i> can never exceed 1.
$0.5 \leq Z < 1$	An excellent assay.
$0 < Z < 0.5$	A marginal assay.
0	A "yes/no" type assay.
$Z < 0$	Screening essentially impossible. There is too much overlap between positive and negative controls for the assay to be useful.

**Table 1.1 Categorization of an HTS assay quality depending on the value of *Z-factor*.**

Since its introduction, *Z-factor* has become the most commonly used criterion for the evaluation and validation of HTS experiments. The publication of Zhang et al. (1999) is now one of the most cited papers in the field of HTS (Sui and Wu 2007). Most of the large laboratories, including the National Institutes of Health Chemical Genomics Center

(<http://ncgc.nih.gov>) and the National Screening Laboratory for the Regional Centers of Excellence in Biodefense and Emerging Infectious Diseases (<http://nsrb.med.harvard.edu>) in the United States, recommend  $Z \geq 0.5$  as an indication of proper assay optimization.

In Inversen et al. (2006), the authors conducted a simulation study and compared the performance of *Z-factor* to that of the signal window and assay variability ratio, and recommended *Z-factor* as a preferred assay performance measure.

The negative impact of systematic error on the HTS hit selection process and on the overall assay quality was ascertained in many scientific publications (Woodroffe and Ginsberg 1998, Brideau et al. 2003, Cheneau et al. 2003, Iredale et al. 2005). Many efforts were put on solving the problem by altering assay preparation methodologies and procedures (Lundholt et al. 2003, Nelson et al. 2004) as well as on developing new data correction or normalization techniques (Heyse 2002, Brideau et al. 2003, Kevorkov and Makarenkov 2005, Makarenkov et al. 2006, Makarenkov et al. 2007, Birmingham et al. 2009, Malo et al. 2010, Carralot et al. 2012).

Heyse (2002) performed a comprehensive study of the complexity and statistical practice in data analysis of high-throughput screening data as a valuable raw material for the drug-discovery process. He concluded that the quantity, complexity and heterogeneity of the HTS data require novel, sophisticated approaches of data analysis. The latter author outlined five major steps which, according to him, are of high importance for performing an HTS data analysis:

1. Quality Assurance: Checking data for experimental artifacts and eliminating low quality data;
2. Biological Profiling: Clustering and ranking compounds based on their biological activity, taking into account specific characteristics of HTS data;
3. Rule-based Classification: Applying user-defined rules to biological and chemical properties, and providing hypotheses for the biological mode-of-action of compounds;

4. Joint Biological-Chemical Analysis: Associating chemical compounds data to HTS data, providing hypotheses for structure-activity relationships;
5. Integration with genomic and gene expression data and assessing the compounds' modes-of-action, toxicity, and metabolic properties.

In the same publication, Heyse stated that “screening data tend to be very complex and they must be analyzed with care”, and also “thorough quality control is a must” and HTS data should be “preprocessed in a suitable way”.

### 1.8.5 *B-score*

A well-known paper by Brideau et al. (2003) was one of the first to propose a systematic error correction method in the context of experimental HTS. The latter authors argued that processing HTS data requires more than simple control-based or basic statistical normalizations such as *Z-score*. Brideau et al. (2003) pointed out the main drawbacks of the *Z-score* normalization and especially its failure to deal with positional effects, i.e., systematic error. They presented a new statistics score, *B-score* (i.e., “Better score”), a robust nonparametric analog of the *Z-score* normalization which is also resistant to outliers. Assuming the following data model:

$$x_{ijp} = \mu_p + R_{ip} + C_{jp} + \varepsilon_{ijp},$$

where  $x_{ijp}$  is the measurement in row  $i$  and column  $j$  of plate  $p$ ,  $\mu_p$  is the mean value of the plate measurements,  $R_{ip}$  is the row  $i$  effect,  $C_{jp}$  is the column  $j$  effect and  $\varepsilon_{ijp}$  is the random noise affecting the well  $(i, j)$  of plate  $p$ . The *B-score* method starts by carrying out a two-way median polish procedure (Tukey 1977) to account for row and column effects of the plate. The median polish procedure has been preferred over several alternative methods, like ANOVA, because of its robustness regarding the outliers. The residual ( $r_{ijp}$ ) of the measurement in row  $i$  and column  $j$  of the plate  $p$ , is obtained as follows by a two-way median polish:

$$r_{ijp} = x_{ijp} - \hat{x}_{ijp} = x_{ijp} - (\hat{\mu}_p + \hat{R}_{ip} + \hat{C}_{jp}).$$

The residual is defined as the difference between the observed measurement ( $x_{ijp}$ ) and its fitted value  $\hat{x}_{ijp}$ , which is defined as a sum of the estimated average of the plate  $p$ , ( $\hat{\mu}_p$ ), the estimated systematic measurement offset ( $\hat{R}_{ip}$ ) for row  $i$  of  $p$  and the estimated systematic measurement column offset ( $\hat{C}_{jp}$ ) for column  $j$  of  $p$ .

The residuals obtained after carrying out the median polish procedure within each plate are then scaled by the plate's *Median Absolute Deviation* ( $MAD_p$ ). For each plate  $p$ , the adjusted median absolute deviation is obtained from the  $r_{ijp}$ 's as a robust estimate of the spread of the  $r_{ijp}$ 's values:

$$MAD_p = \text{median}\left\{ \left| r_{ijp} - \text{median}(r_{ijp}) \right| \right\}.$$

The *B-score* of the compound in row  $i$  and column  $j$  of plate  $p$  is then calculated as follows:

$$B\text{-score} = \frac{r_{ijp}}{MAD_p}.$$

### 1.8.6 Well correction

Makarenkov et al. (2007) proposed a new advanced systematic error correction method, called *Well Correction*, which was designed to remove row and column systematic biases as well as systematic error that affects compounds located at the same well location and repeats for all of the assay plates. As described by Makarenkov et al. (2007), the Well Correction method consists of the two following main steps:

- 1) Least-squares approximation of the data carried out separately for each well location of the assay;
- 2) *Z-score* normalization of the data within each well location of the assay.

In the simulations carried out by Makarenkov et al. (2007), the Well Correction method generally outperformed the Median polish and B-score methods as well as the classical hit selection procedure not implying any data correction. All the conducted tests suggested that the Well Correction procedure is a robust method for systematic error correction and the authors recommended its use prior to the hit selection process.

### ***1.8.7 Other HTS-related research***

Jenkins et al. (2003) introduced a method for improving the quality of HTS hit lists by using computationally-based virtual screening (VS), which, according to the authors, typically provides a large percentage of false positives and requires costly follow-ups to distinguish between the active and inactive compounds. The proposed method is based on the expectation that the false positive hits are not likely to dock well to the target's active sites unlike to the truly active compounds. Therefore, the biological false positives could be identified computationally with minimal time and expense. The latter authors tested their hypothesis in a study using the *angiogenin* enzyme, screened for small-molecule inhibitors. Jenkins et al. (2003) also used VS as a tool to enrich given chemical libraries prior to performing experimental HTS. They reported that VS can be a highly effective and facile tool for enriching the hit rate from high-throughput screens, noting that VS methods may be less valuable as a pre-HTS filter than as a tool for minimizing false positives in HTS. It is worth noting that, except this paragraph, the term "false positive" refers to statistical false positives everywhere else in the thesis. From a statistical perspective, if a feature is selected as a hit, it is a false positive if its true signal is null. If its true signal is non-null, then it is a true positive. However, from a biological perspective, a statistical true positive may nonetheless be a biological false positive. This would be the case if the true signal did not reflect the biological mechanism of interest. By pre-selecting molecules with desired molecular structures via VS methods, researchers may minimize the number of biological false positives.

Hu and Sung (2004) described a method for data mining and outlier detection in the HTS context. The proposed method uses a local trimmed mean approach for estimating spatial outlier factors. Spatial outliers are different from non-spatial outliers in the way that

they are outlying in their local neighbourhood, though, they may not be outlying globally. The described approach allows for estimating and removing spatial trend, i.e., systematic error, in HTS data what can be used to improve the quality of the hit selection process.

Kelley (2005) presented a software package using Microsoft Excel that can detect spatially correlated artefacts, for example: “row effects”, “column effects” or “edge effects”, in HTS data. The software presented by Kelley, and called *Array Validator*<sup>TM</sup>, is based on the Discrete Fourier Transform method and the analysis of the resulting periodograms. It provides a sensitive assessment of the randomness of an array of values. According to the author, *Array Validator*<sup>TM</sup> provides a way for an automatic validation of screening data and generation of alert messages when spatial non-randomness, i.e., systematic error, is detected.

Kevorkov and Makarenkov (2005) described a method for evaluation and topological analysis of the background surface of HTS data. The described method allows for determination of trends and local fluctuations in experimental HTS assays and can also be used for the elimination of systematic error.

Gagarin et al. (2006) discussed the hit selection process in high-throughput screening. Three new clustering techniques, designed to identify quality hits in the observed measurements were proposed in the publication and can be used to improve the hit selection in experimental HTS. The three proposed methods are the following: k-means partitioning, sum of the average squared inside-cluster distances (SASD) and average inter-cluster distance (AICD). The authors claimed that two of the proposed techniques: k-mean partitioning and SASD can bring a significant improvement to the hit selection process.

Wu et al. (2008) reviewed several widely-used normalization and error correction methods. Two new statistical procedures, called BZ-score and R-score, were also proposed in this article. Both BZ-score and R-score methods are based on the classical B-score method (Brideau et al. 2003). The BZ-score procedure is a modification of the B-score method in which an extra step is added after the application of the 2-way Median Polish procedure. During that step Z-score is carried out to normalize the obtained residual values. R-score method is a variant of the B-score method in which the model fitting Median Polish procedure is replaced by the alternative *Robust Linear Model (RLM)*. After fitting the data

model, the obtained residuals can be standardized by the scale estimate from RLM in order to generate a Z statistic, called R-score by the authors. The value of R-score can be used to decide if the positional effects are statistically significant and whether an error correction is needed. The authors underlined the importance of the last step, because their experiments showed that “when there is absolutely no positional effects, the R-score adjustment sacrifices the ability to detect hits”. In their study, Wu et al. (2008) used a rich set of experimental HTS data in which all compounds were tested 42 times over a wide range of 14 concentrations. The latter authors compared the new R-score and BZ-score procedure with the classical B-score method, Z-score and control-based methods. The authors reported that the R-score performed significantly better than the other competing methods, followed by BZ-score whose performance was very close to the performance of the classical B-score method. In the same article, the authors evaluated the following three hit selection methods in the case of duplicate screenings, i.e., when for each compound  $X$ , two measurements  $x_1$  and  $x_2$  are available:

1. “Either-or” – the compound is selected as a hit if at least one of the measurements exceeds the hit-cutting threshold, or equivalently, the combined score for the compound is  $x = \max(x_1, x_2)$ ;
2. “Both” – the compound is selected as a hit if both measurements exceed the hit-cutting threshold, or equivalently, the combined score for the compound is  $x = \min(x_1, x_2)$ ;
3. “Average” – the compound is selected as a hit if the average compound’s measurement exceeds the hit-cutting threshold, or equivalently, the combined score for the compound is  $x = (x_1 + x_2)/2$ .

Wu et al. (2008) concluded that the “Average” hit selection strategy is the best way to combine duplicate measurements in order to identify hits.

Malo et al. (2010) presented a novel approach for maximizing true-positive rates in an HTS study without increasing the false-positive rate. The latter authors stated that they “have noted recently that the B-score (Brideau et al. 2003) method can potentially generate

excessive false positives because normally distributed null data generate long-tailed B-score distributions". In this paper, Malo et al. (2010) proposed a modification of the B-score method. The authors claimed that the robustness of the B-score method could be increased by replacing the originally used model fitting procedure, the two-way median polish, with a trimmed-mean polish. The authors reported that they have achieved particularly good results when a trim value of 0.1 (10%) was used. Malo et al. (2010) recommended that the trim parameter should be chosen to reflect the expected maximum number of hits in the rows and columns of the assay (10% in their simulations) and also that the proposed modification is equivalent to the original B-score method when a trim value of 0.5 (50%) is used. The simulations were conducted using the commercial S-Plus software (TIBCO Spotfire, Somerville, MA).

Shun et al. (2011) reviewed the current practice in high-throughput screening with respect to ensuring appropriate hit selection quality. In this study, a 3-step statistical decision methodology is described for achieving quality HTS results. At the first step, the criteria for quality assessment should be established by calculating for each plate of the assays a number of quality measures: *Z-factor*, *Z'-factor* (Zhang et al. 1999), the *percent coefficient of variation* (the standard deviation of the compounds expressed as a percentage of the standard deviation of the controls). Then, a 2-way ANOVA test should be carried out to assess the presence of the row/column effects. The second step consists of selecting an appropriate HTS data processing method by using the following rules:

- B-score or BZ-score methods should be used if systematic error is present or if the data do not follow normal distribution.
- Z-score method should be used for normally distributed data of bad quality, determined by *Z-factor* or percent coefficient of variation, which are not affected by systematic error.
- Control-based methods should be used for normally distributed data of good quality data which are not affected by systematic error.

During the final step of the proposed method, the data quality of each plate should be retested. The plates that failed the test should be screened again, whereas those that passed the test should be used for the hit selection.

Carralot et al. (2012) developed of a novel edge effect correction algorithm suitable for RNA interference (RNAi) screening. The authors stressed that their goal was providing a specific method that targeted a recurrent type of artefact known as *edge effect*, and not a generic method for correcting any type of systematic error that may corrupt experimental HTS data. The proposed method estimates edge effects for each assay plate separately by using the data from a single control column. The computations are based on a diffusion model. Multiplicative bias values have been assumed and datasets with different signal-to-noise ratios were used in the study. The authors reported that the quality of the test datasets, measured using the Z-factor coefficient, had improved significantly after applying the new method. Carralot et al. (2012) concluded that their method can be successfully used to detect systematic error and obtain a robust estimation of the edges-related spatial biases in RNAi screening data.

### 1.9 Machine learning algorithms in HTS

The cost of pharmaceutical development has increased dramatically in recent years, and many approaches have been developed to decrease both the time and the cost associated with bringing a drug to the market. Among the proposed methods, it is worth noting the use of *in silico* screening of compounds for detecting new drug leads, commonly referred to as virtual high-throughput screening (VHTS) (Sirois et al. 2005). One of the promising branches of VHTS is the application of machine learning methods for *in silico* prediction of the compounds biological activity based on their chemical properties and previous experiences of screening similar compounds.

Briem and Günther (2005) evaluated the three following machine learning techniques to distinguish between kinase inhibitors and other molecules with no reported activity on any protein kinase: support vector machines (SVM), artificial neural networks (ANN) and k-nearest neighbors (kNN). Using the majority vote for all tested techniques, the authors

concluded that ANN provided the best prediction of experimental results, followed by SVM. On the other hand, Müller et al. (2005) described an application of SVM to the problem of assessing the “drug-likeness” of a compound based on a given set of molecular descriptors. The authors concluded that in the drug-likeness analysis the polynomial SVM with a high polynomial degree ( $d = 11$ ) allows for a very complex decision surface which could be used for prediction.

Harper and Pickett (2006) reviewed the current state of the application of data mining techniques in the field of HTS. Data mining has been described as “the exploration and analysis, by automatic or semi-automatic means, of large quantities of data to discover meaningful patterns and rules”. In their publication, Harper and Pickett stated that there has been a recent increase in the application of data-mining techniques in high throughput screening. The authors argued that given the large quantity of data generated during an HTS campaign and the importance of analyzing those data effectively, the use of data-mining techniques in HTS data analysis is expected to increase steadily.

Burton et al. (2006) carried out a recursive partitioning, based on decision trees, for predicting the CYP1A2 and CYP2D6 inhibition. The latter authors reported that with mixing 2D and 3D descriptors, they were able to achieve 2% to 5% gain in accuracy compared to the 3D descriptors alone.

Fang et al. (2006) presented results of a type I *methionine aminopeptidases* (MetAPs) inhibition study. Fang and colleagues employed support vector machines for mining HTS data, while testing a compound library of 43,736 small organic molecules. The authors discovered that half of the active molecules could be recovered after screening only 7% of compounds of the test set.

Plewczynski et al. (2007) reported that an SVM model was able to achieve classification rates up to 100% in evaluating compounds activity with respect to specific protein targets. In the latter study, the authors concluded that the obtained sensitivities for all targets exceeded 80%.

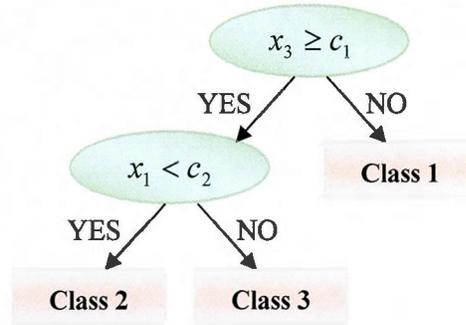
Simmons et al. (2008a) compared 10 different machine learning methods by employing them to develop classifiers on data derived from an *in vivo* HTS campaign. The authors compared the methods' predictive performances in terms of the number of false negatives and false positives. The set of descriptors used by Simmons et al. (2008a) consisted of 825 numerical values, representing 55 possible atom-type pairs mapped to 15 distance ranges. The same team of researchers (Simmons et al. 2008b) described an ensemble-based decision tree model used to virtually screen and prioritize compounds for acquisition.

Mballo and Makarenkov (2010) analyzed the Test assay from McMaster University Data Mining and Docking Competition (Elowe et al. 2005) using binary decision trees, neural networks, support vector machines, linear discriminant analysis, *k*-nearest neighbors and partial least squares. The authors first compared separately the sets of molecular and atomic descriptors in order to establish which of them provides a better prediction. Then, the comparison of the six considered machine learning methods was carried out in terms of false positives and false negatives, method's sensitivity and enrichment factor. Finally, a variable selection procedure allowing one to improve the method's sensitivity was presented and applied in the framework of polynomial SVM.

Each of these studies was conducted in particular statistical and experimental contexts, i.e., the sampling strategies, the type of HTS data, the proportions of confirmed hits in the data and the available descriptors differed between the studies, making the comparison of the obtained results very difficult.

### **1.10 Decision trees**

Here we present two machine learning approaches we applied in our study. Decision trees are hierarchical data structures implementing the divide-and-conquer strategy. They can be used as an efficient classification method (Breiman et al. 1984). A decision tree is composed of internal decision nodes and terminal leaves as shown below (Figure 1.6):



**Figure 1.6** An example of a decision tree where  $c_1$  and  $c_2$  are constants such that  $c_1 > c_2$

Each  $f_m(X)$  defines a discriminant in the input space dividing it into smaller regions which are further subdivided as we descend down toward the leaves. Different decision tree strategies assume different models  $f_m(X)$ . Those models define the shape of the discriminants and the regions. A leaf defines a region in the input space where the instances falling in this region are classified as belonging to the same class.

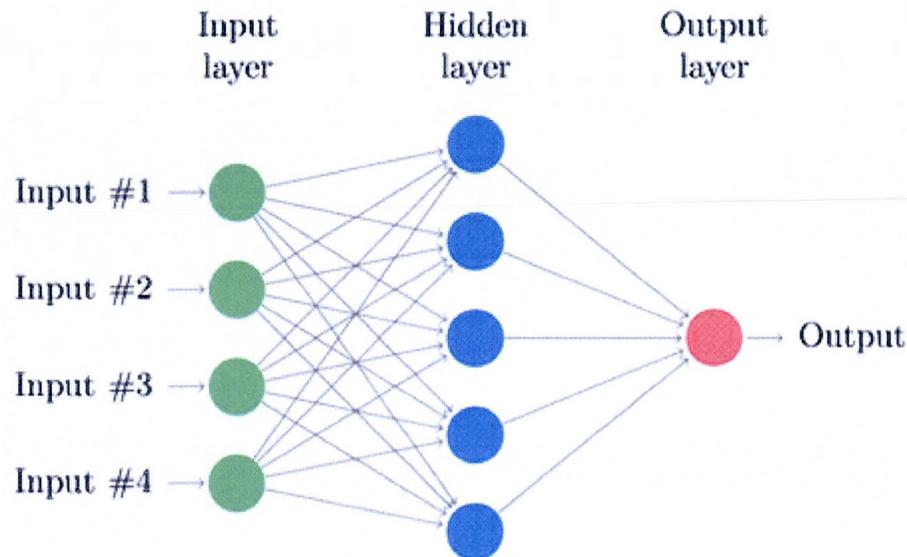
### 1.11 Neural networks

Artificial Neural Networks (ANN), or simply neural networks (NN), are a powerful modeling tool which is largely used in various disciplines such as chemistry, economics, medicine and pharmaceutical science. Inspired by the power and adaptability of the human brain, NN models are composed of numbers of small units, called *neurons*, interconnected and operating in parallel (Bishop 1995, Gurney 1997, Ripley 1996). Mathematical model of a neuron states that a neuron has multiple inputs  $x_j \in R, j=1, 2, \dots, m$ . Associated with each input is a connection weight  $w_j \in R$ . The total input received by a neuron can be computed as the weighted sum of all its inputs:

$$v = \sum_{j=1}^m w_j x_j + w_0,$$

where  $w_0$  is a bias associated with the given neuron. It is convenient to think of the bias as of the weight for an extra input  $x_0$  coming from a neuron whose output value is always equal to 1. A neuron has one output and implements a monotone activation function  $g$  that maps  $v$  into

$g(v)$ , which is the output value of the neuron. While there are numerous different neural network architectures that have been studied by researchers, the most successful NN applications in data mining and artificial intelligence have been multilayer feed-forward networks, Figure 1.7.



**Figure 1.7 An example of a multilayer feed-forward network**

In a typical neural network there exist an input layer consisting of nodes needed to define the input values and successive layers of nodes that are neurons as depicted above. The outputs of neurons in a layer are the inputs to neurons in the next layer. The last layer is called the output layer. Layers between the input and output layers are called hidden layers. When a neural network is used to predict a numerical quantity, there is one neuron in the output layer and its output is the prediction. When the network is used for classification, the output layer typically has as many nodes as the number of classes and the output layer node with the largest output value gives the network's estimate of the class for a given input.

An important characteristic of NNs is that they can be trained, to recognize inputs from different classes. It is said that NN learns by example. During the training a number of instances and their expected output values are supplied. A machine learning algorithm is used

to calculate the input weights of all neurons in a way that minimizes the total error while predicting the training data.

Once trained, the neural network is used for prediction. During this phase test, the data are entered as an input to the network, which then generates an output. The output value is a prediction to which class the test data belong.

## CHAPTER II

### SYSTEMATIC ERROR DETECTION IN EXPERIMENTAL HIGH-THROUGHPUT SCREENING

This chapter is a reproduction of the following article:

Dragiev P., Nadon R. and Makarenkov V. (2011) Systematic error detection in experimental high-throughput screening. *BMC Bioinformatics*, 12:25.

This chapter is at the fundament of the research work presented in this thesis. It focuses on the problems addressing the quality and reliability of the high-throughput screening (HTS) results. As an important first step of the complex and expensive drug development process, HTS identifies among thousands and sometimes even millions of compounds, those, called *hits*, that have the potential of becoming a successful medicine. Any wrong decision during the HTS hit selection, i.e., choosing incorrect compounds for further development or overlooking some promising drug candidates, turns out to be very costly. Errors during the compound screening cause the activity levels of some compounds to be systematically over- or underestimated. The effect of the systematic error is usually location dependent. The affected compounds are typically located at the edges of one or several plates of the assay. Working with systematically biased measurements during the hit selection magnifies the number of false positives and false negatives and, in general, undermines the quality of the obtained screening results.

Because of the considerable negative effects that systematic error has on the whole drug development process, many efforts have been made to improve the reliability and the

robustness of the HTS equipment. Despite of recent technological advances, it has become clear that systematic error is often caused by procedural and environmental factors beyond the control of the equipment. The increase in assay sizes and duration of HTS experiments makes it very difficult to ensure that all the measurements are taken at absolutely the same conditions, the main of which are temperature, humidity, light intensity and solution concentrations.

Different methods have been developed for processing raw HTS data before the hit selection (Brideau et al. 2003, Kevorkov and Makarenkov 2005, Makarenkov et al. 2007, Malo et al. 2006). Each of the existing methods modifies the data in order to eliminate or reduce the effect of a given type of systematic error, or to compensate for differences between the plates in the assay. Despite their power to diminish the error effects if systematic error is present, all existing methods have one significant disadvantage – when applied on data not containing any error or data affected by an error of a different type than the considered method is intended to treat – they introduce an important bias affecting the correct HTS measurements (Makarenkov et al. 2007).

Chapter II focuses on the development of the tests for estimating the presence of systematic error in raw HTS data. The ability to detect systematic error is of a significant practical importance. The error detection tests can be used for decreasing the risk of a wrong application of the error correction methods. The proposed tests not only detect the presence of systematic error in general but also pinpoint the most probable locations of the error, i.e., indicate which plates, rows and columns are affected. The information provided by these tests can be used to decide whether the considered data require error correction to be carried out and, if so, which error correction method is the most appropriate for the dataset in hand. We will describe the application of Student's t-test, the  $\chi^2$  goodness-of-fit test and the Kolmogorov-Smirnov test combined with the Discrete Fourier Transform method in the context of experimental HTS and compare the obtained results.

## **2.1 Abstract**

### ***2.1.1 Background***

High-throughput screening (HTS) is a key part of the drug discovery process during which thousands of chemical compounds are screened and their activity levels measured in order to identify potential drug candidates (i.e., hits). Many technical, procedural or environmental factors can cause systematic measurement error or inequalities in the conditions in which the measurements are taken. Such systematic error has the potential to critically affect the hit selection process. Several error correction methods and software have been developed to address this issue in the context of experimental HTS (Brideau et al. 2003, Heuer et al. 2005, Heyse 2002, Kevorkov and Makarenkov 2005, Makarenkov et al. 2006, Makarenkov et al. 2007, Malo et al. 2006). Despite their power to reduce the impact of systematic error when applied to error perturbed datasets, those methods also have one disadvantage – they introduce a bias when applied to data not containing any systematic error (Makarenkov et al. 2007). Hence, we need first to assess the presence of systematic error in a given HTS assay and then carry out systematic error correction method if and only if the presence of systematic error has been confirmed by statistical tests.

### ***2.1.2 Results***

We tested three statistical procedures to assess the presence of systematic error in experimental HTS data, including the  $\chi^2$  goodness-of-fit test, Student's t-test and Kolmogorov-Smirnov test (D'Agostino and Stephens 1986) preceded by the Discrete Fourier Transform (DFT) method (Cooley and Tukey 1965). We applied these procedures to raw HTS measurements, first, and to estimated hit distribution surfaces, second. The three competing tests were applied to analyze simulated datasets containing different types of systematic error, and to a real HTS dataset. Their accuracy was compared under various error conditions.

### ***2.1.3 Conclusions***

A successful assessment of the presence of systematic error in experimental HTS assays is possible when the appropriate statistical methodology is used. Namely, the t-test

should be carried out by researchers to determine whether systematic error is present in their HTS data prior to applying any error correction method. This important step can significantly improve the quality of selected hits.

## 2.2 Background

High-throughput screening (HTS) is a modern technology used by drug researchers to identify pharmacologically active compounds (Dove 2003). HTS is a highly automated early-stage mass screening process. Contemporary HTS equipment allows for testing more than 100,000 compounds a day. HTS serves as a starting point for rapid identification of primary hits that are then further screened and evaluated to determine their activity, specificity, and physiological and toxicological properties (Heuer et al. 2005). As a highly sensitive test system, HTS requires both precise measurement tools and dependable quality control. The absence of standardized data validation and quality assurance procedures is recognized as one of the major hurdles in modern experimental HTS (Gunter et al. 2003, Kaul 2005, Malo et al. 2006). Acknowledging the importance of automatic quality assessment and data correction systems, many researchers have offered methods for eliminating experimental systematic artifacts which, if left uncorrected, can obscure important biological or chemical properties of screened compounds (false negatives) and can seemingly indicate biological activity when there is none (false positives) (Brideau et al. 2003, Dove 2003, Gagarin et al. 2006, Gunter et al. 2003, Heuer et al. 2005, Heyse 2002, Kaul 2005, Kevorkov and Makarenkov 2005, Makarenkov et al. 2006, Makarenkov et al. 2007, Malo et al. 2006, Malo et al. 2010, Zhang et al. 1999, Zhang et al. 2000).

Systematic error may be caused by various factors, including robotic failures and reader effects, pipette malfunction or other liquid handling anomalies, unintended differences in compound concentrations due to agent evaporation or variation in the incubation time and temperature differences, and lighting or air flow present over the course of the entire screen (Heuer et al. 2005, Makarenkov et al. 2007). Unlike random error that produces measurement noise and usually has minimal impact on the whole process, systematic error produces measurements that are systematically over- or underestimated. Systematic error may be time dependent, introducing biases in individual plates or subsets of consecutive plates, but it may

also affect an entire HTS assay (i.e., all screened plates). In practice, systematic error is almost always location related. The under- or overestimation affects compounds located in the same row or column or in the same well location across the screened plates. The row and column effects may be persistent across the assay affecting repeatedly the same rows and columns on different plates or may vary from plate to plate, perturbing some rows and columns within a particular plate only (Makarenkov et al. 2007). Plate controls are used in HTS to ensure the accuracy of the activity measurements being taken. Controls are substances with stable well-known activity levels. They might be positive (i.e., a strong activity effect is observed) or negative (i.e., no any activity effect is observed). Controls help to detect plate-to-plate variability and determine the level of background noise.

The following normalization and pre-processing methods have been widely used in experimental HTS to remove plate-to-plate variation and make plate measurements comparable across plates (Malo et al. 2006, Makarenkov et al. 2007):

- *Percent of control* – the following formula is used:

$$\hat{x}_{ij} = \frac{x_{ij}}{\mu_{pos}}, \text{ where } x_{ij} \text{ is the raw measurement of the compound in well } (i, j), \hat{x}_{ij} \text{ is}$$

the normalized value of  $x_{ij}$ , and  $\mu_{pos}$  is the mean of positive controls.

- *Control normalization* (known also as *normalized percent inhibition transformation*) is based on the following formula:

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_{neg}}{\mu_{pos} - \mu_{neg}}, \text{ where } x_{ij} \text{ is the raw measurement of the compound in well } (i, j),$$

$\hat{x}_{ij}$  is the normalized value of  $x_{ij}$ ,  $\mu_{pos}$  is the mean of positive controls, and  $\mu_{neg}$  is the mean of negative controls.

- *Z-score* normalization is carried out as follows:

$$\hat{x}_{ij} = \frac{x_{ij} - \mu}{\sigma}, \text{ where } x_{ij} \text{ is the raw measurement of the compound in well } (i, j), \hat{x}_{ij} \text{ is}$$

the normalized value of  $x_{ij}$ ,  $\mu$  is the mean of all the measurements of the given plate, and  $\sigma$  is the standard deviation of all the measurements of the given plate.

- *B-score* (i.e., *Best score* normalization (Brideau et al. 2003) is carried out as follows: First, a two-way median polish procedure (Tukey 1977) is performed to account for row and column effects of the plate. The resulting residuals within each plate are then divided by their median absolute deviation, *MAD*. It is worth noting that there is an additional smoothing step that could be applied across plates (see the original article (Brideau et al. 2003) for a description of the smoothing). This optional smoothing step was not applied however in (Makarenkov et al. 2006, Makarenkov et al. 2007, Tukey 1977).

The residual ( $r_{ijp}$ ) of the measurement in row  $i$  and column  $j$  on the  $p^{\text{th}}$  plate is obtained as follows by a two-way median polish procedure (Equation 2.1):

$$r_{ijp} = x_{ijp} - \hat{x}_{ijp} = x_{ijp} - (\hat{\mu}_p + \hat{R}_{ip} + \hat{C}_{jp}). \quad (2.1)$$

The residual is defined as the difference between the observed result ( $x_{ijp}$ ) and the fitted value  $\hat{x}_{ijp}$ , defined as the estimated average of the plate ( $\hat{\mu}_p$ ) plus the estimated systematic measurement offset ( $\hat{R}_{ip}$ ) for row  $i$  of plate  $p$  and plus the estimated systematic measurement column offset ( $\hat{C}_{jp}$ ) for column  $j$  of plate  $p$ . For each plate  $p$ , the adjusted median absolute deviation ( $MAD_p$ ) is then obtained from the  $r_{ijp}$ 's.

*Median absolute deviation (MAD)* – a robust estimate of spread of the  $r_{ijp}$ 's values is computed as follows:  $median\{|r_{ijp} - median(r_{ijp})|\}$ .

The B-score normalized measurements are then calculated as follows:

$$B\text{-score} = \frac{r_{ijp}}{MAD_p}. \quad (2.2)$$

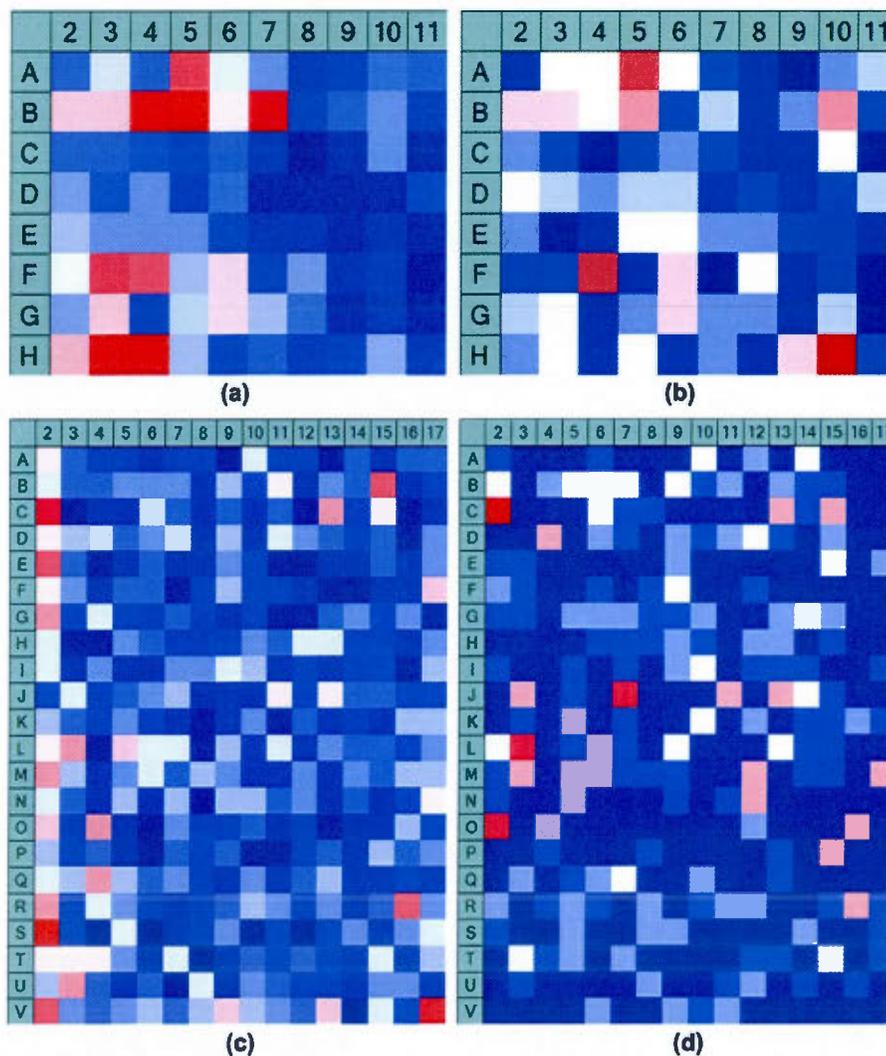
The B-score normalization was introduced by a team of *Merck Frosst* researchers (Brideau et al. 2003) as a systematic error correction method.

- *Well correction* is another advanced systematic error correction technique (Makarenkov et al. 2006, Makarenkov et al. 2007) used to remove systematic biases affecting the assay's wells, rows or columns, and spread across all the plates of the assay. It consists of two main steps:
  1. Least-squares approximation of the data carried out separately for each well location of the assay;
  2. Z-score normalization of the data within each well location of the assay (i.e., the Z-score normalization is performed across all the plates of the assay).

In the HTS workflow, the normalization/data correction phase is usually followed by the *hit selection* process. During this process the most active compounds are identified as *hits* and selected for additional screens. A predefined threshold is usually established to select hits (Malo et al. 2006). Depending on the specifics of the research study, one may be looking for compounds whose activity level is greater than the defined threshold (i.e., activation assay) or interest may lie in the compounds whose measurements are below the defined threshold (i.e., inhibition assay). In this study, we always assume the latter case where the hits are the compounds with the smallest measurement values. The threshold for defining hits is usually expressed using the mean value and standard deviation of the considered measurements. The most widely used threshold is  $\mu - 3\sigma$ , where  $\mu$  is the mean value and  $\sigma$  is the standard deviation of the considered measurements. Hits can be selected globally, over the whole assay, when the mean and standard deviation of all assay compounds are calculated, or on a plate-by-plate basis, when the mean and standard deviation of the compounds of each single plate are considered (Makarenkov et al. 2007, Malo et al. 2006).

The presence of systematic error in a HTS assay can be identified and visualized using its *hit distribution surface* (Kevorkov and Makarenkov 2005, Makarenkov et al. 2007). Such a surface can be computed by determining the number of selected hits for each well location. In the ideal case when systematic error is absent, we expect that the hits are evenly distributed over the well locations. However, this expectation is not always fulfilled in real datasets (see Figure 2.1). This figure presents the hit distribution surfaces computed for two hit selection thresholds,  $\mu - 2\sigma$  and  $\mu - 3\sigma$ , of two experimental HTS screens performed at

McMaster (Figure 2.1 a, b – Elowe et al. 2005) and Princeton (Figure 2.1 c, d – Helm et al. 2003) Universities. The row and column effects in the hit distributions across plates are easily noticeable here, especially in the case of a lower (i.e.,  $\mu-2\sigma$ ) hit selection threshold.



**Figure 2.1 Systematic error in experimental HTS data. Hit distribution surfaces for the McMaster (cases (a) and (b) - 1250 plates – Elowe et al. 2005) and Princeton (cases (c) and (d) - 164 plates – (Helm et al. 2003) Universities experimental HTS assays. Values deviating from the plate means for more than 2 standard deviations - cases (a) and (c), and for more than 3 standard deviations - cases (b) and (d) were selected as hits. The well, row and column positional effects are shown (the wells containing controls are not presented).**

The dataset provided by the Chemistry Department of Princeton University consists of a screen of compounds that inhibit the glycosyltransferase MurG function of *E. coli* (Helm et al. 2003). The experimental data for 164 plates were considered. According to the ChemBank description, this assay has been obtained during a screen that measured the binding of MurG to a fluorescent (fluorescein-labelled) analogue of UDP-GlcNAc. Positives were defined as compounds that inhibit binding of GlcNAc to MurG. The McMaster assay was originally used as a benchmark in *McMaster Data Mining and Docking Competition* (Elowe et al. 2005). The McMaster dataset, which will be examined in detail in this study, consists of compounds intended to inhibit the *E. coli Dihydrofolate reductase* (DHFR). The screen of 50,000 training molecules selected by the organizers of McMaster Competition yielded 96 primary hits, then, 12 potent hits (i.e., hits confirmed by dose response analysis), the majority of which were novel DHFR inhibitors that fell into 3 broad structural classes (Elowe et al. 2005).

It is worth noting that the application of sophisticated pre-processing HTS techniques does not always guarantee data improvement. Moreover, the application of systematic error correction methods on error-free HTS assays will produce data in which certain activity measurements will be biased (Makarenkov et al. 2007). The result of such a misuse of data pre-processing methods can lead to a dramatically inaccurate hit selection. Makarenkov et al. 2007 (see Figure 2 and Figure 4, cases a and c, in Makarenkov et al. 2007) showed that all data correction methods introduce a bias when applied to error-free HTS data. This bias can be less important (e.g., in the case of the Well correction procedure) or very significant (e.g., in the case of the B-score method). Hence, the data correction methods should be applied with caution and only in situations when the presence of systematic error in the given assay has been demonstrated by an appropriate statistical methodology. Assessing the presence of systematic error in experimental HTS is the main focus of this article.

## **2.3 Materials And Methods**

### ***2.3.1 Data description***

In this study we consider an experimental assay provided by the HTS laboratory of McMaster University. This assay was called *Test assay* and used as a benchmark in McMaster Data Mining and Docking Competition (Elowe et al. 2005). McMaster *Test assay*

consists of 50,000 different chemical compounds whose potential to inhibit the *E. coli* DHFR was tested. Each of the 50,000 considered compounds was screened in duplicate; two copies of each of the 625 plates were run through the HTS equipment; 1250 plates in total, with wells arranged in 8 rows and 12 columns, were screened; columns 1 and 12 of each plate were used for positive and negative controls and were, therefore, not considered in our study. Thus, every plate comprised 80 different compounds. The exact experimental conditions of *Test assay* are reported in (Elowe et al. 2005). The competition organizers defined as *primary hits* the compounds that reduced the DHFR of *E. coli* to 75% of the average residual activity of the high controls. Two lists of hits were published (for more details, the reader is referred to: [http://hts.mcmaster.ca/Downloads/82BFBE4-F2A4-4934-B6A8-804CAD8E25A0\\_files/experimental\\_actives.pdf](http://hts.mcmaster.ca/Downloads/82BFBE4-F2A4-4934-B6A8-804CAD8E25A0_files/experimental_actives.pdf)). The first list, called a *consensus hits list*, contained all compounds that were classified as hits in both of their replicate measurements (i.e., both measurement values were lower than or equal to 75% of the reference controls). Only 42 of all the 50,000 tested compounds were declared consensus hits. The second list, called an *average hits list*, contained 96 compounds classified as hits when the average value of the two HTS measurements was lower than or equal to 75% of the reference controls. Obviously, all consensus hits were also average hits. A secondary screening of the 96 average hits was also performed in order to determine their activity in different concentrations. As result of the secondary screening, 12 of the average hits were identified as *D-R hits* (i.e., hits having *well-behaved dose-response curves*).

### 2.3.2 Generating systematic error

We simulated data in order to evaluate the performances of the systematic error detection tests. First, we generated error-free datasets consisting of random normally distributed data. The basic data format adopted here was that of the McMaster dataset - 1250 plates, each containing 96 wells arranged in 8 rows and 12 columns. In addition, we also generated two other basic datasets which were 4 and 16 times bigger. They also included 1250 plates, each of them comprising 384 (16 x 24) and 1536 (32 x 48) wells, respectively. It is worth noting that 96, 384 and 1536-well plates are the most typical plate formats used in the modern HTS.

An assay was defined as an ordered set of plates  $PL_p$ , where  $p$  ( $1 \leq p \leq 1250$ ) is the plate number. Each plate,  $PL_p$ , can be viewed as a matrix of experimental HTS measurements  $x_{ijp}$ , where  $i$  ( $1 \leq i \leq N_R$ ) is the row number,  $j$  ( $1 \leq j \leq N_C$ ) is the column number, and  $N_R$  and  $N_C$  are, respectively, the number of rows and columns in  $PL_p$ . The generated values  $x_{ijp}$ 's followed the standard normal distribution  $\sim N(0, 1)$ .

Error type	Generation of error-affected measurements
A. Datasets with both <i>column and row systematic errors</i> which are constant across all assay plates.	$x'_{ijp} = x_{ijp} + r_i + c_j + Rand_{ijp}$ , $1 \leq i \leq 8, 1 \leq j \leq 12, 1 \leq p \leq 1250$ .
B. Datasets with the column systematic error only which is constant across all plates.	$x'_{ijp} = x_{ijp} + c_j + Rand_{ijp}$ , $1 \leq i \leq 8, 1 \leq j \leq 12, 1 \leq p \leq 1250$ .
C. Datasets with the well systematic error which is constant across all plates.	$x'_{ijp} = x_{ijp} + w_{ij} + Rand_{ijp}$ , $1 \leq i \leq 8, 1 \leq j \leq 12, 1 \leq p \leq 1250$ .
D. Datasets with the variable column and row systematic error which are different for each plate.	$x'_{ijp} = x_{ijp} + r_{ip} + c_{jp} + Rand_{ijp}$ , $1 \leq i \leq 8, 1 \leq j \leq 12, 1 \leq p \leq 1250$ .
E. Datasets with the random error only (i.e., systematic error was absent).	$x'_{ijp} = x_{ijp} + Rand_{ijp}$ , $1 \leq i \leq 8, 1 \leq j \leq 12, 1 \leq p \leq 1250$ .
<p>where: <math>x'_{ijp}</math> is the error-affected value in well <math>ij</math> (row <math>i</math>, column <math>j</math>) of plate <math>p</math>.  <math>x_{ijp}</math> is the original value in well <math>ij</math> of plate <math>p</math> in the error-free dataset.  <math>r_i</math> is the systematic error in row <math>i</math> (constant over all plates); it had a normal distribution with the parameters <math>\sim N(0, C)</math>.  <math>c_j</math> is the systematic error in column <math>j</math> (constant over all plates); it had a normal distribution with the parameters <math>\sim N(0, C)</math>.  <math>w_{ij}</math> is the systematic error that affects well <math>ij</math> (row <math>i</math>, column <math>j</math>) and is the same for all plates; it had a normal distribution with the parameters <math>\sim N(0, C)</math>.  <math>r_{ip}</math> is the systematic error in row <math>i</math> of plate <math>p</math>; it had a normal distribution with the parameters <math>\sim N(0, C)</math>.  <math>c_{jp}</math> is the systematic error in column <math>i</math> of plate <math>p</math>; it had a normal distribution with the parameters <math>\sim N(0, C)</math>.  <math>Rand_{ijp}</math> is the random error affecting well <math>ij</math> (row <math>i</math>, column <math>j</math>) of plate <math>p</math>; it had a normal distribution with the parameters <math>\sim N(0, 0.3SD)</math>.            Datasets for <math>C = 0, 0.6SD, 1.2SD, 1.8SD, 2.4SD</math> and <math>3SD</math> were generated and tested, where <math>\mu</math> is the mean and <math>SD</math> is the standard deviation of the error-free dataset.</p>	

**Table 2.1 - Five types of HTS datasets containing different kinds of systematic and/or random error generated and tested in this study.**

Then, the hits were added to the datasets. Several hit percentages,  $h$ , were tested in our simulations:  $h = 0.5, 1, 2, 3, 4$  and  $5\%$ . The locations and values of hits were chosen randomly. The probability of each well in each plate to contain a hit was  $h\%$ . The values of hits followed a normal distribution with the parameters  $\sim N(\mu - 5SD, SD)$ , where  $\mu$  and  $SD$  are the mean value and standard deviation of the error-free dataset.

In total, five types of datasets, presented in Table 2.1, containing different kinds of systematic and/or random error were generated and tested. In order to render our simulation study more realistic, we limited the number of rows, columns and wells affected by systematic error. Typically, in real HTS assays only some of the error parameters (i.e.,  $r_i, c_j, w_{ij}, r_{ip}$  and  $c_{jp}$ , see Table 2.1) are non null and only a few columns and rows are biased by systematic error. In datasets of types A and B, the number of rows and columns affected by systematic error as well as their locations were chosen randomly. These parameters were identical for all the plates of the assay. In datasets of type D, the number of rows and columns affected by systematic error as well as their locations were also randomly selected, but these parameters were different for different plates of the assay. In datasets of type C, the number of biased wells and their locations were randomly selected and were the same for all assay plates. The datasets used in our simulations were subject to the following constraints. For the 96-well plates, at most 2 rows and 2 columns (cases A, B and D), and not more than 10% of the wells (case C) were affected by systematic error. For the 384-well plates, the limits were 4 rows, 4 columns and 10% of the wells, whereas for the 1536-well plates, systematic error affected at most 8 rows, 8 columns and 10% of wells.

### **2.3.3 Systematic error detection tests**

Three systematic error detection methods, including the t-test, the  $\chi^2$  goodness-of-fit test and Discrete Fournier Transform procedure followed by the Kolmogorov-Smirnov test, were examined in this study in the context of experimental HTS.

#### **2.3.3.1 t-test**

The first systematic error detection test was based on the classical two-sample Student's t-test for the case of samples with different sizes. In Simulation 1, we carried out

this test on every row and every column of each assay. In Simulation 2, we applied it to the rows and columns of the assay's hit distribution surfaces. In both cases, we divided the data into two independent subsets (i.e., samples). The first subset contained the measurements of the tested row or column while the second subset consisted of all remaining plate measurements. In this test, the null hypothesis  $H_0$ , was that the selected row or column does not contain systematic error. If systematic error is absent, then the mean of the given row or column is expected to be close to the mean of the rest of the data in the given plate or hit distribution surface. For the two samples in hand:  $S_1$  with  $N_1$  elements and  $S_2$  with  $N_2$  elements, we first calculated the two sample variances  $s_1^2$  and  $s_2^2$ , and then their weighted average (Equation 2.3):

$$s_p^2 = \frac{(N_1 - 1) \times s_1^2 + (N_2 - 1) \times s_2^2}{N_1 + N_2 - 2}. \quad (2.3)$$

The value of the *t*-statistic was then obtained as presented in Equation 2.4:

$$t = \frac{\mu_1 - \mu_2}{\sqrt{s_p^2 \left( \frac{1}{N_1} + \frac{1}{N_2} \right)}}, \quad (2.4)$$

where  $\mu_1$  is the mean of the sample  $S_1$  and  $\mu_2$  is the mean of the sample  $S_2$ . The calculated *t*-statistic was then compared to the corresponding critical value for the chosen statistical significance level  $\alpha$  (the  $\alpha$  values equal to 0.01 and 0.1 were used in our simulations) in order to decide whether or not  $H_0$  should be rejected. While assuming homogeneity of variance in the construction of the *t*-test, the computation can be optimized using the equivalent contrasts in the context of an analysis of variance.

### 2.3.3.2 $\chi^2$ goodness-of-fit test

The second tested method was the  $\chi^2$  goodness-of-fit test. This test was performed in Simulation 2 only in order to assess the presence of systematic error in the hit distribution surfaces. It was first recommended in (Makarenkov et al. 2007) in order to identify systematic error in HTS data. The null hypothesis  $H_0$ , here, is that no systematic error is

present in the data. If  $H_0$  is true, then the hits are evenly distributed across the well locations and the observed counts of hits  $x_{ij}$  in each row  $i$  and each column  $j$  of the hit distribution surface is not significantly different from the expected value calculated as the total counts across the entire surface divided by the number of wells. The rejection region of  $H_0$  is  $P(\chi^2 > C_\alpha) > \alpha$ , where  $C_\alpha$  is the  $\chi^2$  distribution critical value corresponding to the selected  $\alpha$  parameter (the  $\alpha$  values equal to 0.01 and 0.1 were tested here) and to the number of degrees of freedom of the model.

For a hit distribution surface with  $N_R$  rows and  $N_C$  columns, we can assess the presence of systematic error in a given row  $r$  by computing the test statistic  $\chi_r^2$  by means of Equation 2.5:

$$\chi_r^2 = \sum_{j=1}^{N_C} \frac{(x_{rj} - E)^2}{E}, \quad (2.5)$$

where  $E$  is the total hits count of the whole hit distribution surface divided by the number of wells ( $N_R \times N_C$ ) with the number of degrees of freedom equal to  $N_R - 1$ .

Similarly, the columns of the hit distribution surface affected by systematic error can be identified by calculating the test statistic  $\chi_c^2$ , using Equation 2.6 below:

$$\chi_c^2 = \sum_{i=1}^{N_R} \frac{(x_{ic} - E)^2}{E}, \quad (2.6)$$

where  $E$  is the total hits count of the whole hit distribution surface divided by the number of wells ( $N_R \times N_C$ ) with the number of degrees of freedom equal to  $N_C - 1$ .

The presence of systematic error in the assay can be detected even if systematic error affects particular wells of the assay, not necessarily located in the same row or column. We can achieve it by calculating the test statistic  $\chi^2$  over all well locations of the given hit distribution surface (Equation 2.7):

$$\chi^2 = \sum_{i=1}^{N_R} \sum_{j=1}^{N_C} \frac{(x_{ij} - E)^2}{E}, \quad (2.7)$$

where  $E$  is the total hits count of the whole hit distribution surface divided by the number of wells ( $N_R \times N_C$ ) with the number of degrees of freedom equal to  $N_R \times N_C - 1$ .

### 2.3.3.3 Discrete Fourier Transform and Kolmogorov-Smirnov test

The third tested method consisted of the Discrete Fourier Transform (DFT) procedure (Cooley and Tukey 1965) followed by the Kolmogorov-Smirnov goodness-of-fit test (D'Agostino and Stephens 1986). DFT has been widely used in the frequency analysis of signals and, in particular, for building the signal's density spectrum. The power density spectrum shows the energy contained in each frequency component existing in the signal. In order to apply DFT to HTS data we need first to unroll a plate measurement matrix into a linear sequence of measurements. There are two natural ways to do so: (a) to build the sequence starting by the first row of the plate, followed by the second row, then third one, and so on, and (b) to start by the first column of the plate, followed by the second column, third one, and so on. The analysis of sequences (a) and (b) would allow us to detect column and row effects, respectively. DFT detects frequencies of signals that repeat every two, three, four, and so on, positions in the sequence. DFT calculates the amplitudes of every possible frequency component. Let  $y_k^p$  ( $1 \leq k \leq N$ ) be the power density spectrum generated by the DFT analysis for the plate  $p$  with  $N$  wells.

As a second step of this method, we carry out the Kolmogorov-Smirnov test to compute the probability of the density spectrum  $y_k^p$  occurring under the null hypothesis of no effect. The test statistic  $D$  can be calculated as follows:

$$D = \max_{1 \leq k \leq N} \left( F(y_k^p) - \frac{k-1}{N}, \frac{k}{N} - F(y_k^p) \right), \quad (2.8)$$

where  $F(y_k^p)$  is defined as the number of values in the density spectrum that are lower than or equal to  $y_k^p$ , i.e.,  $F(y_k^p) \equiv \left\| \{y_l^p, 1 \leq l \leq N, y_l^p < y_k^p\} \right\|$ . Big values of  $D$  lead to the rejection

of the null hypothesis (i.e.,  $x_{ijp}$ 's have been drawn from random normally distributed data). The method consisting of the DFT analysis followed by the Kolmogorov–Smirnov test was included in some commercial software focusing on the detecting systematic error in experimental data (e.g. in Array Validator described in Kelley 2005).

## 2.4 Results And Discussion

### 2.4.1 Simulation 1: Detecting systematic error in individual plates

Simulation 1 consisted of the detection of systematic error on a plate-by-plate basis. Artificial HTS data for three different plate sizes: 96 wells – 8 rows and 12 columns, 384 wells – 16 rows and 24 columns, and 1536 wells – 32 rows and 48 columns were first generated. We started by creating basic error-free datasets for which the well measurements followed a standard normal distribution  $\sim N(0,1)$ . For all datasets the number of plates was set to 1250 – the same as in McMaster *Test assay* (Elowe et al. 2005). Then, we added 1% of hits to each of the generated basic datasets. The hits were added in such a way that the probability that a given well contained a hit was 1%. All the hit values followed a normal distribution with the parameters  $\sim N(\mu-5SD, SD)$ , where  $\mu$  and  $SD$  are the mean value and standard deviation of the basic dataset (without hits).

Using these error-free datasets, we generated datasets comprising different types of systematic error, labelled A to E, as reported in Table 2.1. Systematic error was added only to some of the assay rows (columns, wells). The number of rows (columns, wells) affected by systematic error as well as the indexes of the affected rows, columns and wells were determined randomly for each considered dataset. Six types of error-affected sets were produced for each error-free dataset by varying the standard deviation of systematic error. The following values of the systematic error standard deviation were used: 0, 0.6SD, 1.2SD, 1.8SD, 2.4SD and 3.0SD, where  $SD$  is the standard deviation of the basic dataset. The t-test and K-S test were then applied to error-affected data. Both tests produced a binary result for each row and column of each plate: Systematic error was detected or not detected in this row or column. The output was then compared to the information from the data generation phase to determine whether the result of the test was correct.

Cohen's kappa coefficient (Cohen 1960, Fleiss 1981) was calculated to estimate the accuracy of both statistical tests. Cohen's kappa is a measure of inter-rater agreement or inter-annotator agreement. The kappa coefficient, which takes into account the agreement occurring by chance is computed as follows (Equation 2.9):

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)}, \quad (2.9)$$

where  $\Pr(a)$  is the relative observed agreement among raters (i.e., statistical tests in our study) and  $\Pr(e)$  is the hypothetical probability of chance agreement. If the raters are in complete agreement, then  $\kappa = 1$ . If there is no agreement among the raters, other than what would be expected by chance, then  $\kappa \leq 0$ .

In our HTS context,  $\Pr(a)$  and  $\Pr(e)$  were calculated as follows:

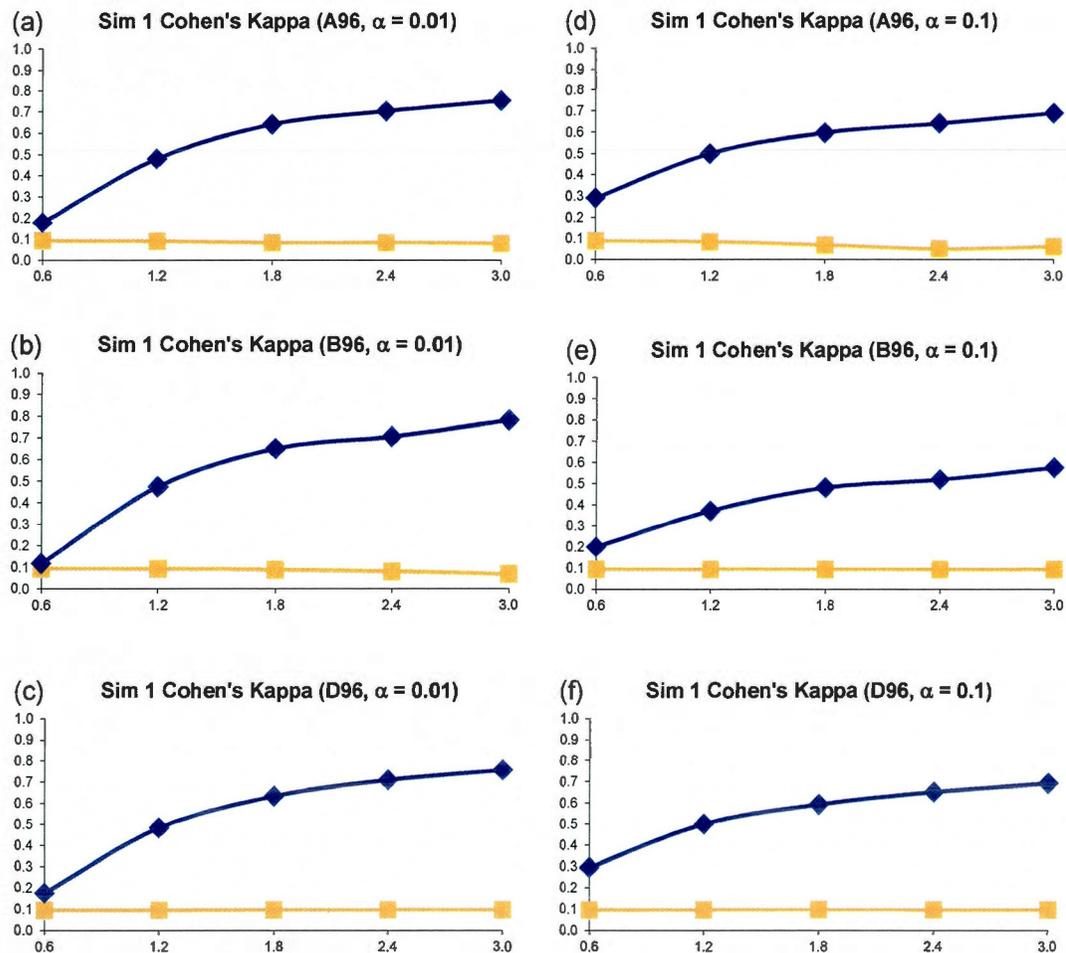
$$\Pr(a) = \frac{TP + TN}{P \times (N_R + N_C)} \text{ and } \Pr(e) = \frac{(TP + FN) \times (TP + FP) + (TN + FN) \times (TN + FP)}{(P \times (N_R + N_C))^2}, \text{ where}$$

$P$  is the number of plates in the assay,  $N_R$  and  $N_C$ , are, respectively, the number of rows and columns per plate,  $TP$  (*true positives*) is the sum of the numbers of rows and columns where systematic error was added during the data generation and then detected by the test,  $FP$  (*false positives*) is the sum of the numbers of rows and columns where systematic error was not added but detected by the test,  $TN$  (*true negatives*) is the sum of the numbers of rows and columns where systematic error was not added and not detected by the test, and  $FN$  (*false negatives*) is the sum of the numbers of rows and columns where systematic error was added but not detected by the test.

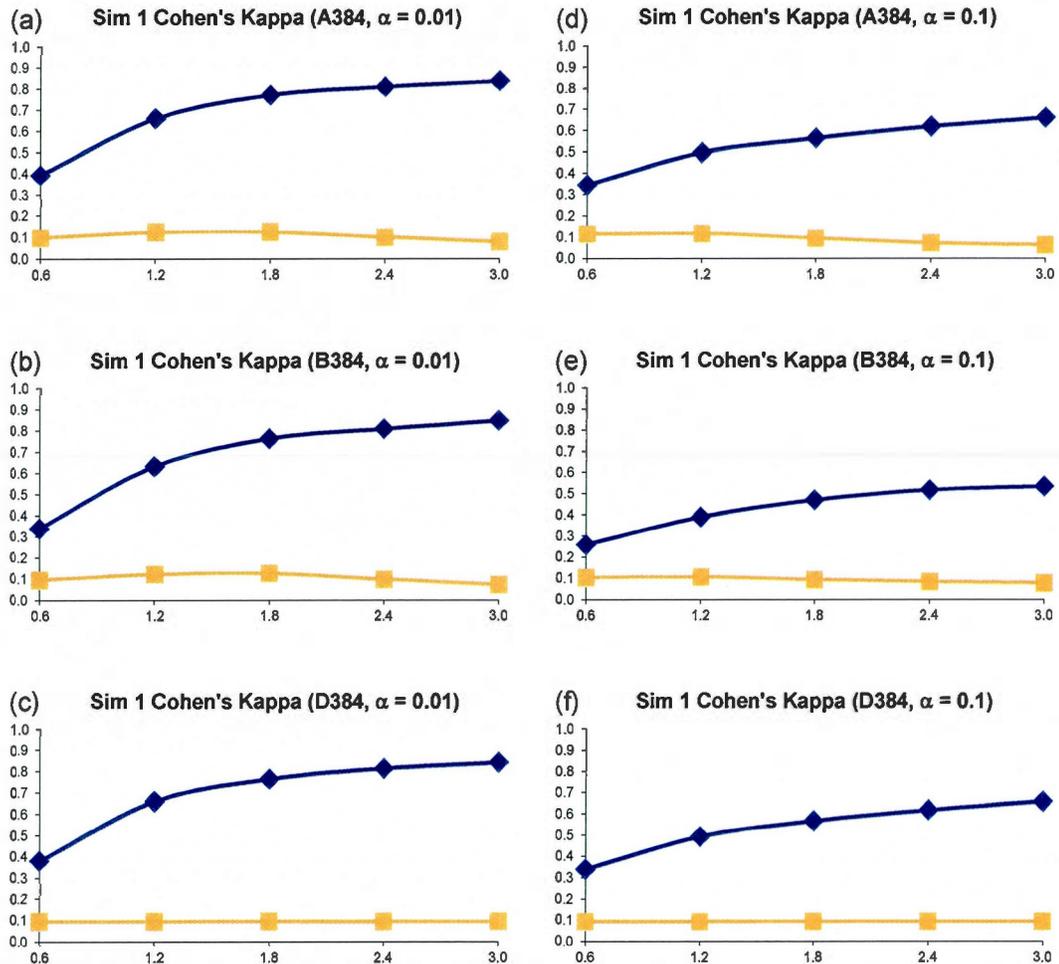
For all generated variants of error-affected data, 500 different sets were created. The averages of obtained Cohen's kappa coefficients are represented in Figures 2.2, 2.3 and 2.4 (for the 96, 384 and 1536-well plates, respectively). Also, the sensitivity (Figures 2.9, 2.10 and 2.11, see the section Supplementary Figures), specificity (Figures 2.12, 2.13 and 2.14) and success rate (Figures 2.21, 2.22 and 2.23) of the two tests are depicted. The sensitivity and specificity of the two tests were calculated as follows (Equations 2.10):

$$\text{Sensitivity} = \frac{TP}{TP + FN}, \quad \text{Specificity} = \frac{TN}{TN + FP}. \quad (2.10)$$

Since datasets of types C and E did not contain row or column systematic error, the sensitivity and Cohen's kappa coefficient of both competing statistical tests for these data were undefined (i.e.,  $TP = FN = 0$  for these data types).

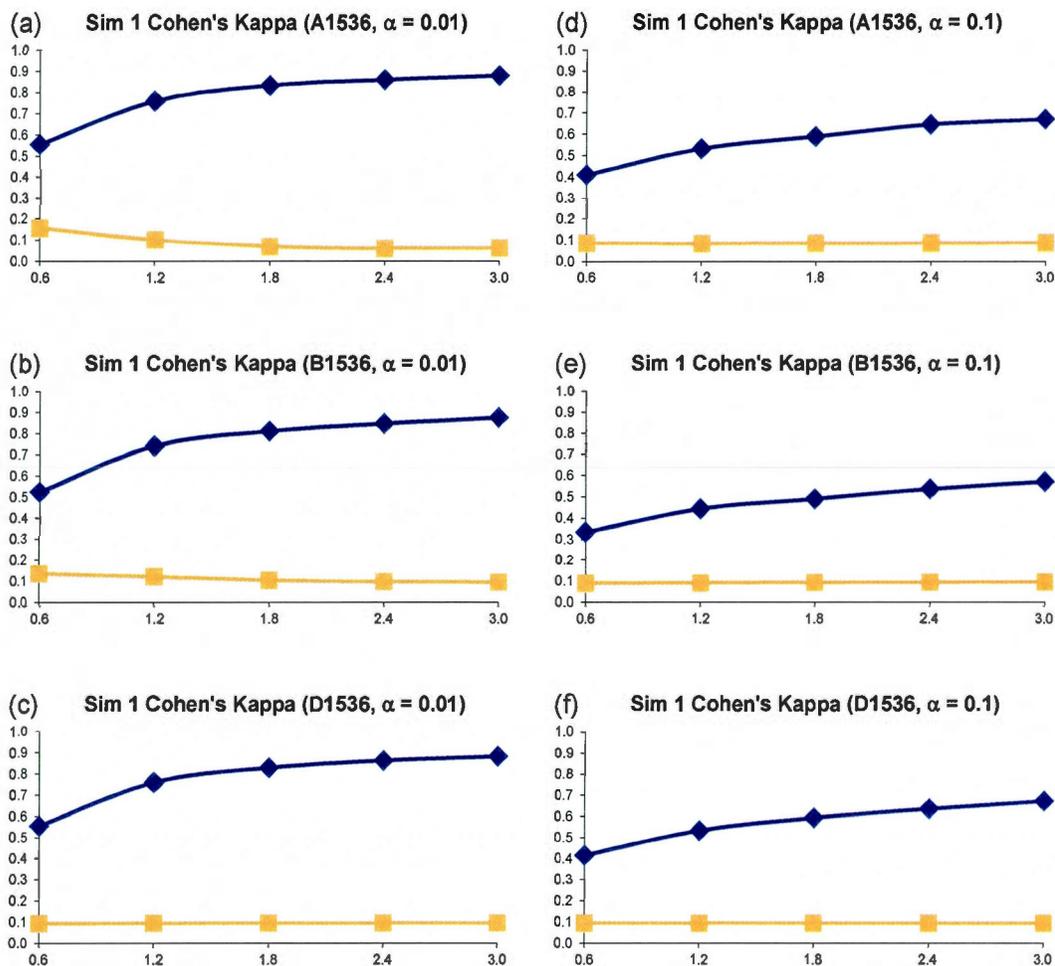


**Figure 2.2 Simulation 1, Plate Size: 96 wells – Cohen's Kappa vs Error Size Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: (a) - (c):  $\alpha = 0.01$ ; Second column: (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



**Figure 2.3 Simulation 1, Plate Size: 384 wells – Cohen's Kappa vs Error Size Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: (a) - (c):  $\alpha = 0.01$ ; Second column: (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**

The kappa coefficient curves in Figures 2.2, 2.3 and 2.4 show that the t-test clearly outperforms DFT followed by the K-S test for all selected sizes of systematic error, confidence levels and plate sizes. The accuracy of the t-test grows as the size of systematic error increases. It also grows slightly as the plate size increases. The accuracy of the K-S test remains very low and usually varies between 0.0 and 0.1, thus suggesting a very poor systematic error recovery by this test.



**Figure 2.4 Simulation 1, Plate Size: 1536 wells - Cohen's Kappa vs Error Size.**  
**Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: (a) - (c):  $\alpha = 0.01$ ; Second column: (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**

Figures 2.21, 2.22 and 2.23 indicate that the success rate of the t-test is largely independent of the systematic error variance and remains very steady for all tested types of systematic error and plate sizes. In contrast, the success rate of the K-S test decreases as the standard deviation of systematic error increases. The performance of the K-S test is also affected by the size of the plate (Figures 2.2, 2.3 and 2.4). The K-S test success rate decreases significantly, and often falls below 50%, for larger plates (Figure 2.23). The chosen

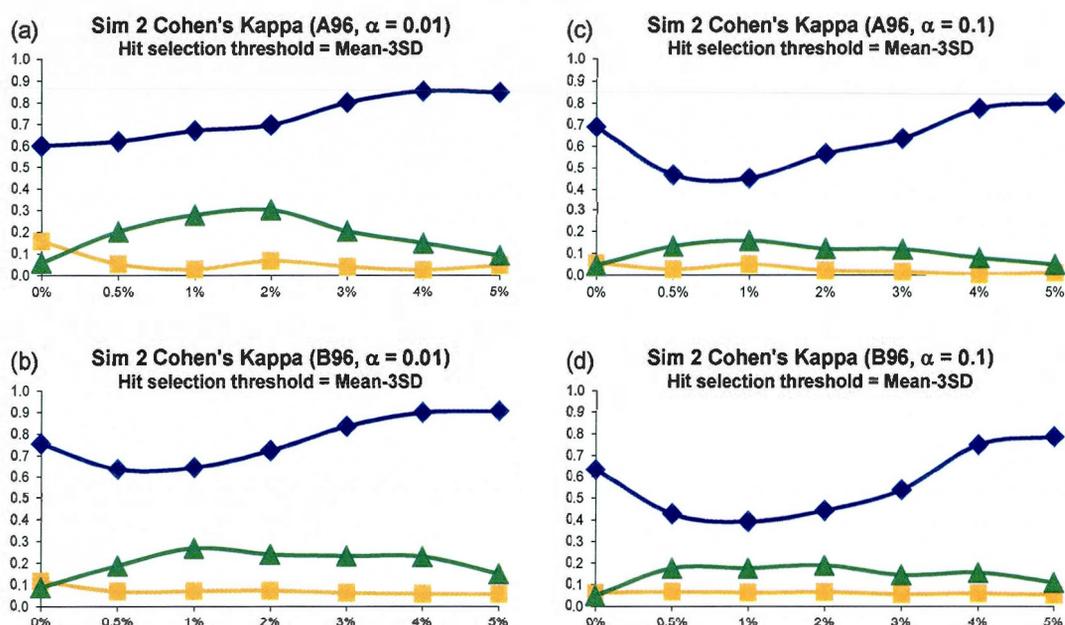
confidence level  $\alpha$  affects the accuracy of both statistical tests. For instance, the use of  $\alpha = 0.1$  generally causes a decrease in the kappa coefficient (the decrease of 0.2 on average, see Figures 2.2, 2.3 and 2.4) and in the success rate (the decrease of 10% on average, see Figures 2.21, 2.22 and 2.23) of the t-test, when compared to  $\alpha = 0.01$ . The sensitivity charts (Figures 2.9, 2.10 and 2.11) show that the increase in the variance of systematic error leads to the increase in sensitivity of both tests. In terms of sensitivity, the t-test outperforms the K-S test for all data types and all sizes of systematic error, the only exception being large plates tested with the confidence level  $\alpha = 0.1$  (Figure 2.11).

Similarly to real HTS assays, our artificially generated datasets had systematic error in only a few rows and/or columns. They contained many negative and only a few positive samples. Such an imbalance between positive and negative samples implies that the overall accuracy of the tests will depend much more on the test specificity than on its sensitivity. Figures 2.12, 2.13 and 2.14 confirm this observation – most of the specificity charts resemble the corresponding success rate charts (see Figures 2.21, 2.22 and 2.23).

#### ***2.4.2 Simulation 2: Detecting systematic error on hit distribution surfaces***

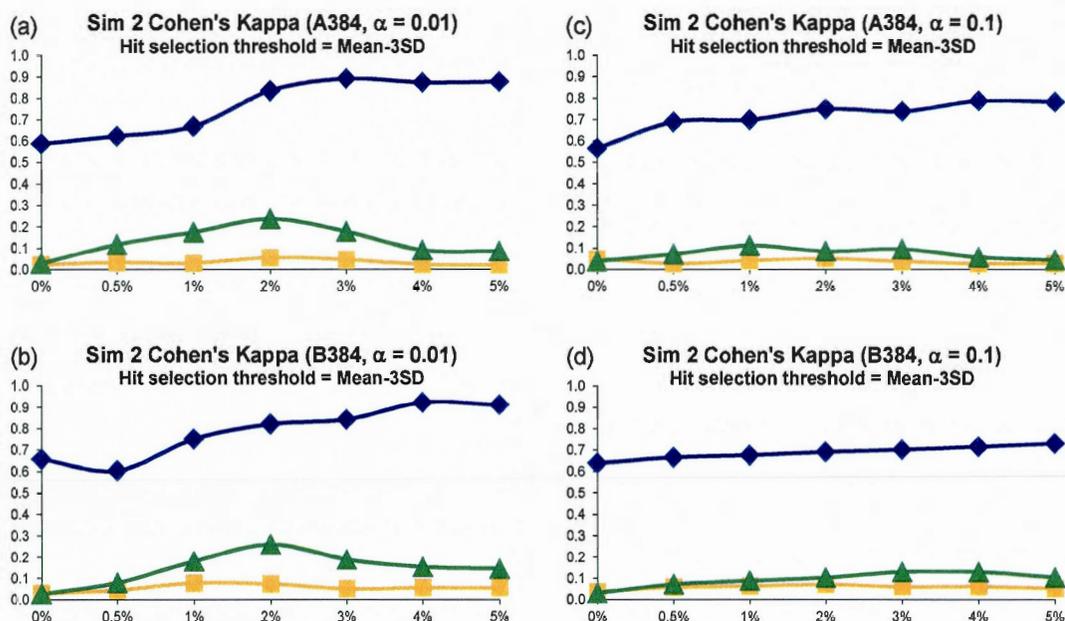
The second simulation, Simulation 2, consisted of the detection of systematic error on the hit distribution surfaces. The recommendation to use statistical tests to examine hit distribution surfaces of experimental HTS assays was first formulated in (Makarenkov et al. 2007), in the case of the  $\chi^2$  test. In Simulation 2, we also considered artificially generated assays with plates of three different sizes (i.e., 96-, 384- and 1536-well plates as well as 1250-plate assays) with the measurements following the standard normal distribution. From every basic dataset we generated 6 error-free datasets comprising 0.5%, 1%, 2%, 3%, 4% and 5% of hits. All the hit values followed a normal distribution with the parameters  $\sim N(\mu - 5SD, SD)$ . Using the error-free datasets, we generated assays containing different types of systematic error (i.e., from A to E). Systematic error, added to some of the assay rows (columns, wells) only, followed the normal distribution with the mean value of 0 and the standard deviation of  $1.2SD$ . For each such an assay, we calculated its hit distribution surface for the hit selection threshold of  $\mu - 3\sigma$ . Then we applied, in turn, the t-test, and the K-S and  $\chi^2$  goodness-of-fit tests to detect the presence of systematic error.

For each error variant, 500 different datasets were generated and the averages of obtained Cohen's kappa coefficients were plotted in Figures 2.5, 2.6 and 2.7. The sensitivity and specificity of the three tests were depicted in Figures 2.15 to 2.20, and the success rate in Figures 2.24, 2.25 and 2.26. The hit distribution surfaces for the assays of types C, D and E (these assays don't contain systematic error that repeats along all assay plates) cannot be used to retrace row or column systematic error. Hence, the sensitivity and Cohen's kappa coefficient for datasets of types C, D and E were undefined.



**Figure 2.5 - Simulation 2, Plate Size: 96 wells, Cohen's Kappa vs Hit Percentage. Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**

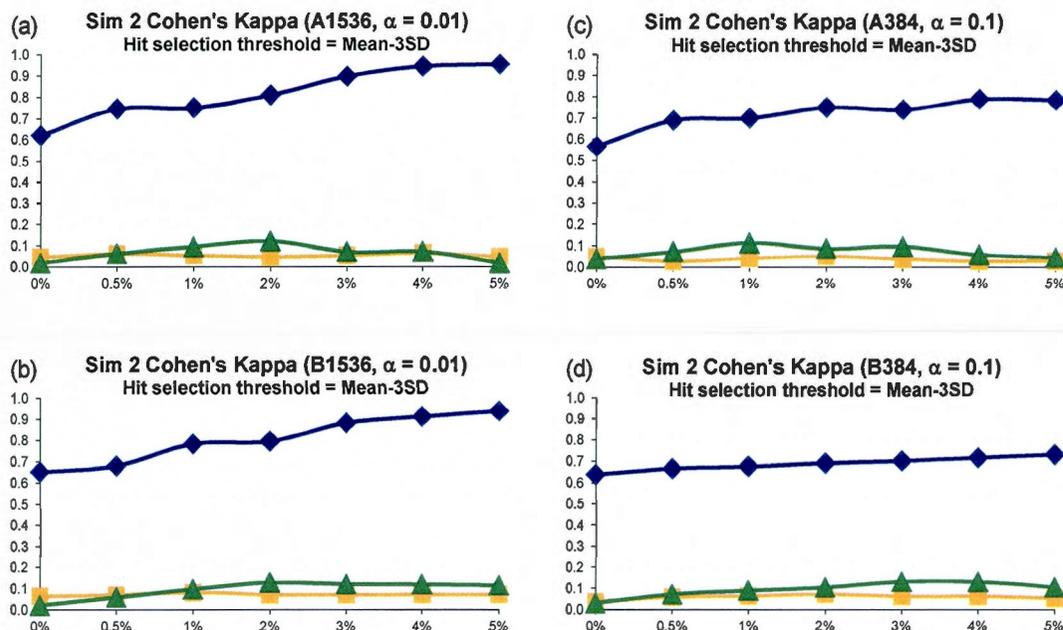
The kappa coefficient curves presented in Figures 2.5, 2.6 and 2.7 illustrate that the t-test clearly outperforms the  $\chi^2$  goodness-of-fit test as well as the combination of DFT and the K-S test for all selected sizes of systematic error, confidence levels and plate sizes. The accuracy of the t-test generally grows as the size of systematic error increases, but this trend is not as steady as in Simulation 1: The curve's minimum is not always associated with the lowest systematic noise (e.g., see cases *c* and *d* in Figure 2.5).



**Figure 2.6 Simulation 2, Plate Size: 384 wells, Cohen's Kappa vs Hit Percentage. Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**

The kappa values for the  $\chi^2$  and K-S tests usually varies between 0.0 and 0.25, thus suggesting a poor systematic error recovery provided by both of them. As in Simulation 1, the success rate of the t-test is largely independent of the systematic error variance (Figures 2.24, 2.25 and 2.26). Moreover, the success rate of the t-test varies between 90% and 100% in the most of simulated experiments. At the same time, the accuracy of the K-S test is extremely low in almost all of the considered situations. The success rate analysis of the  $\chi^2$  goodness-of-fit test suggests that this test follows different patterns for different types of data. For datasets of types D and E, whose hit distribution surfaces did not contain systematic error, the accuracy of the  $\chi^2$  test is very close to that of the t-test (Figures 2.24, 2.25 and 2.26, cases d, e, i and j). However, for the datasets that contained row and/or column systematic error and well systematic error, the success rate of the  $\chi^2$  goodness-of-fit test is significantly lower than that of the t-test (Figures 2.24, 2.25 and 2.26, cases a to c and f to h) and shows a tendency to deteriorate when the percentage of hits in the data increases. The sensitivity patterns shown in Figures 2.15, 2.16 and 2.17 demonstrate that the sensitivity of the three

statistical tests grows as the percentage of hits contained in the data increases. Similarly to Simulation 1, choosing a bigger value of  $\alpha$  led to a decrease in the accuracy of all tests.



**Figure 2.7 Simulation 2, Plate Size: 1536 wells, Cohen's Kappa vs Hit Percentage. Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**

### 2.4.3 Application to the McMaster data

As a final step in our study we applied the three discussed systematic error detection tests on real HTS data. We examined the impact that the presented methodology would have on the hit selection process in McMaster Data Mining and Docking Competition *Test assay* (Elowe et al. 2005). Similarly to Simulations 1 and 2 carried out with artificial data, we performed two types of analysis. First, we studied the raw HTS measurements, and then calculated and analyzed the hit distribution surfaces of *Test assay*.

We carried out the t-test on every plate of *Test assay*, scanning all rows and columns of each plate for the presence of systematic error. We performed the calculation for several

confidence levels including:  $\alpha = 0.01, 0.05, 0.1$  and  $0.2$ . In each case, we counted the number of rows and columns in which the test reported the presence of systematic error and also the number of plates in which at least one row or column contained systematic error. The collected results are presented in Table 2.2.

$\alpha$	Plates	Rows	Rows %	Columns	Columns %
0.01	159	76	0.76%	94	0.75%
0.05	814	575	5.76%	606	4.86%
0.1	1121	1148	11.50%	1296	10.38%
0.2	1241	2242	22.46%	2583	20.70%

**Table 2.2 - Number of rows, columns and plates (where at least one row or column contains systematic error) of McMaster *Test assay* in which the t-test reported the presence of systematic error, depending on the  $\alpha$  parameter. Only 8 rows and 10 columns of McMaster *Test assay* were examined because the first and twelfth columns of the (8 by 12) plates were used for controls.**

$\alpha$	Original hits	Obtained hits	Preserved hits	Added hits	Removed hits
0.01	96	123	57	66	39
0.05	96	125	55	70	41
0.1	96	126	52	74	44
0.2	96	130	55	75	41

**Table 2.3 - Number of hits selected in McMaster *Test assay* for the  $\mu-3SD$  threshold after the application of the B-score correction, depending on the  $\alpha$  parameter. The t-test was carried out to detect systematic error.**

The obtained results suggest that the number of positives for the row and column effects is almost exactly what we would expect by chance (e.g., approximately 1% when we used  $\alpha = 0.01$ , 5 % when we used  $\alpha = 0.05$ , etc.). This means that there is no statistical evidence of bias for columns and rows in McMaster *Test assay*.

$\alpha$	Original hits	Obtained hits	Preserved hits	Added hits	Removed hits
0.01	96	357	79	278	17
0.05	96	419	79	340	17
0.1	96	411	79	332	17
0.2	96	417	76	341	20

**Table 2.4 - Number of hits selected in McMaster *Test assay* for the  $\mu-2.29SD$  threshold (i.e., threshold used by the McMaster competition organizers to select the 96 original average hits) after the application of the B-score correction, depending on the  $\alpha$  parameter. The t-test was carried out to detect systematic error.**

For comparative purposes, we corrected the raw McMaster data using the B-score method in all plates where systematic error was detected by the t-test. Unlike the artificially generated data used in the simulation study, McMaster *Test assay* contained replicated plates - every compound of the assay was screened twice (Elowe et al. 2005). We adjusted our hit selection procedure to search for *average hits*. Thus, we first calculated the average of the two compound measurements and then used it in the hit selection process. If systematic error was detected only in first plate and, therefore, corrected using the B-score method, then the residuals produced by B-score were incomparable with the values of the second (i.e., replicated) plate. In order to make the measurements in both plates comparable, we normalized both plates by means of the Z-score method prior to calculating the average compound activity. Using the corrected dataset, we determined the assay hits for two hit selection thresholds:  $\mu-3SD$  - the most popular hit cutting threshold employed in HTS, and  $\mu-2.29SD$  - the threshold used by the McMaster competition organizers to identify the original 96 average hits. The obtained results are reported in Tables 2.3 and 2.4, respectively. A comparison between the original set of hits and the newly selected hits is also made in these tables. In fact, these tables report how many of the original hits remained hits, how many of them were removed and how many new hits were selected. For the threshold  $\mu-3SD$ , only about half of the original hits were preserved, whereas for the threshold  $\mu-2.29SD$  about four times more hits were selected for the B-score corrected data. The presented results

demonstrate how significantly the selected error correction method and confidence level  $\alpha$  can affect the hit selection process in experimental HTS.

In our second experiment, we computed and analyzed the hit distribution surfaces of McMaster *Test assay* for the hit selection thresholds:  $\mu-3SD$  and  $\mu-2SD$ . We assessed the presence of systematic error in the assay by applying the three discussed systematic error detection tests: t-test, K-S test and  $\chi^2$  goodness-of-fit test. All three tests detected the presence of systematic error in both surfaces for both considered confidence levels  $\alpha = 0.01$  and 0.1. While the hit distribution surface is useful for detecting the presence of overall bias, it does not capture the variability of the bias on a plate-by-plate basis.

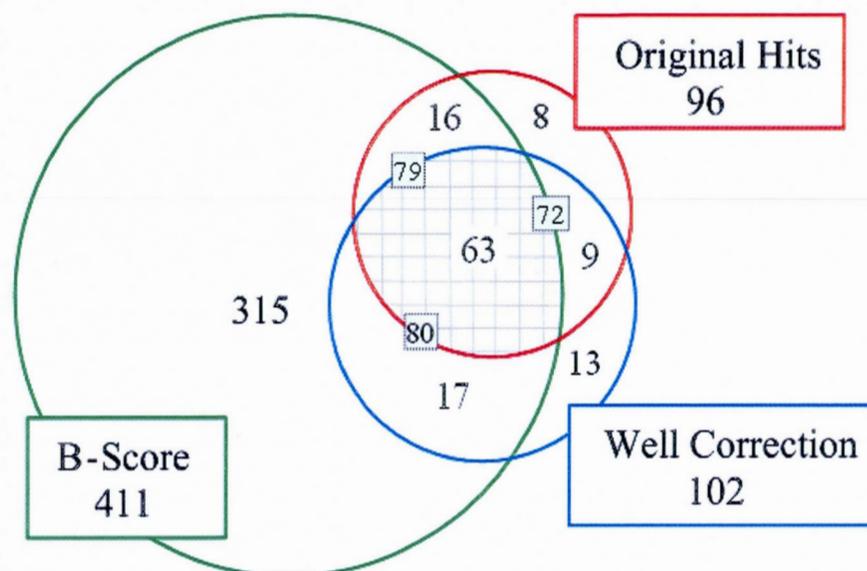
Threshold	Original hits	Obtained hits	Preserved hits	Added hits	Removed hits
$\mu-3SD$	96	26	26	0	70
$\mu-2.29SD$	96	102	72	30	24

**Table 2.5 - Number of hits selected in McMaster *Test assay* for the  $\mu-3SD$  and  $\mu-2.29SD$  thresholds after the application of the Well Correction method.**

Finally, we also applied the Well correction method to remove systematic error from McMaster *Test assay*. After Well correction was performed, the hit selection was carried out again for the hit selection thresholds:  $\mu-3SD$  and  $\mu-2.29SD$ . Table 2.5 reports the comparative results of the two hit selections. When analyzing the obtained hits for the  $\mu-2.29SD$  threshold, one can notice that 24 of the original hits were not detected and, at the same time, 30 new compounds were selected as hits.

Figure 2.8 presents a summary of our experiments conducted with McMaster *Test assay*. The pairwise intersections between the three obtained sets of hits are presented. The dashed grey area in the middle represents the intersections between the three hit sets and thus defines the *consensus hits* for McMaster *Test assay*. The results provided by the B-score method (414 hits in total) shows that this data correction procedure tends to overestimate, at least when compared to Z-score and Well correction, the number of hit compounds. On the other hand, the results provided by the Well correction method suggest that about one third of

the original hits could be, in fact, false positives and that about the same percentage of false negatives could be ignored if systematic error present in the raw McMaster data is not identified and removed adequately.



**Figure 2.8** Intersections between the original set of hits (96 hits in total) and the sets of hits obtained after the application of the *B-score* (411 hits in total; the method was carried out only on the plates where systematic error was detected) and *Well correction* methods (102 hits in total) computed for McMaster *Test assay*. The  $\mu - 2.29SD$  hit selection threshold was used to select hits.

## 2.5 Conclusions

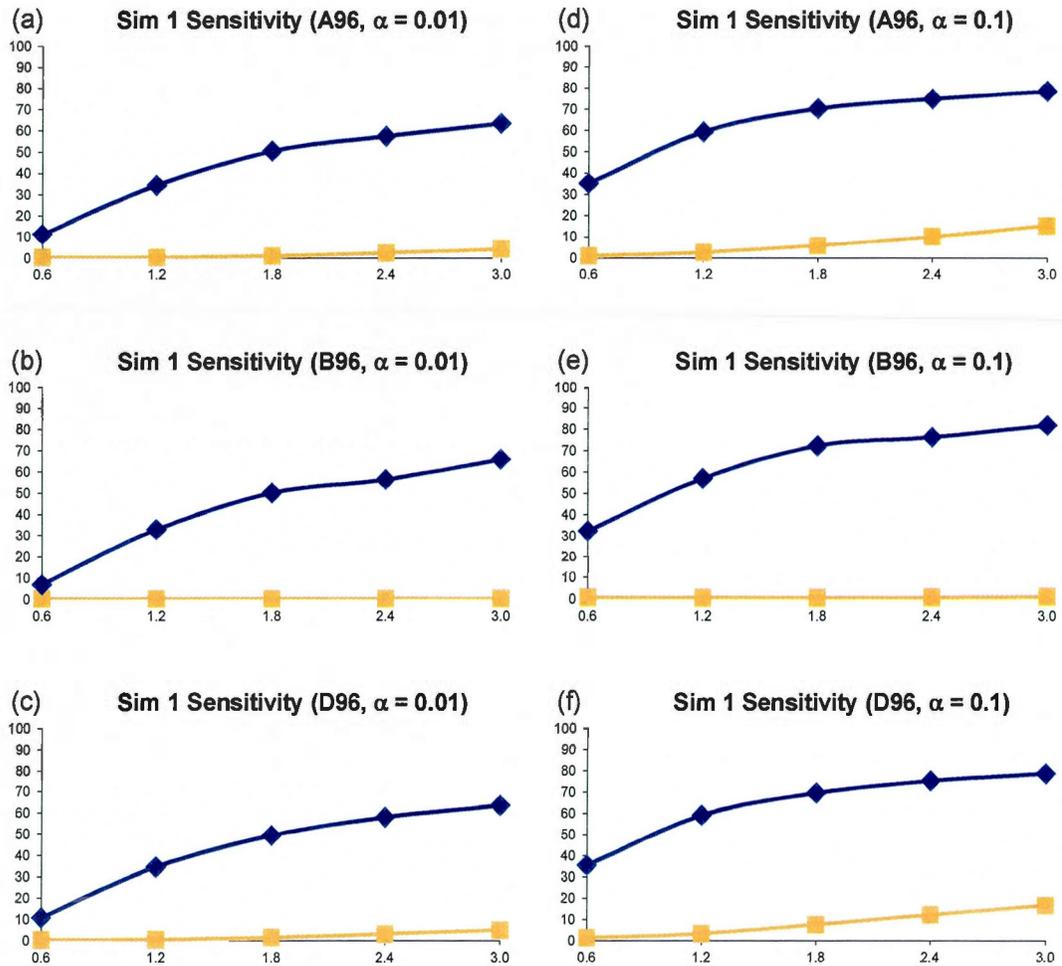
In this article we discussed and tested three methods for detecting the presence of systematic error in experimental HTS assays. We conducted a comprehensive simulation study with artificially generated HTS data, constructed to model a variety of real-life situations. The variants of each dataset, comprising different hit percentages and various types and levels of systematic error, were examined. The experimental results show that the

method performances depend on the assay parameters - plate size, hit percentage, and type and variance of systematic error. We found that the simplest and computationally fastest method, the t-test, outperformed the Kolmogorov-Smirnov (K-S) and  $\chi^2$  goodness-of-fit tests in most of the practical situations. The t-test demonstrated a high robustness when applied on a variety of artificial datasets. The success rate of the t-test was, in most situations, well above 90%, regardless the plate size, noise level and type of systematic error, while the values of Cohen's kappa coefficient computed for this test suggested its superior performance especially in the case of large plates and high level of systematic noise. We can thus recommend the t-test as a method of choice in experimental HTS. On the contrary, advocated in some works (Kelley 2005, Root et al. 2003) Discrete Fourier Transform followed by the K-S test yielded very disappointing results. Moreover, the latter technique required a lot of computational power but provided the worst overall performance among the three competing statistical procedures. The K-S test can still be used to examine HTS data located in small plates (i.e., 96-well plates), but we strongly recommend not using it for the analysis of large plates (i.e., 384 and 1536-well plates) and hit distribution surfaces. The main reason for such a disappointing performance of the K-S test is that it was applied, as recommended in (Kelley 2005), on the data already transformed by the Discrete Fourier method. Figure 2.27 presents an example of data from one of the simulated 96-well plates before and after the application of Discrete Fourier Transform. The raw data followed a normal distribution and contained random error only (i.e., systematic error was not added). The raw data did not deviate from the normal distribution, as shown both graphically (Figure 2.27a) and by the K-S test ( $KS = 0.03$ ,  $p = 0.5$ ). However, after the application of Discrete Fourier Transform, the data deviate from normality as shown in the graph (Figure 2.27a) and by the K-S test ( $KS = 0.06$ ,  $p = 0.0018$ ). The third method, the  $\chi^2$  goodness-of-fit test suggested in (Makarenkov et al. 2007), can be employed to assess hit distribution surfaces for the presence of systematic error. In general, its performances were lower than those of the t-test and were very sensitive to the type of systematic error as well as to its variance. The  $\chi^2$  goodness-of-fit test could be recommended, especially to analyze HTS assays with small plate sizes, but we suggest carrying out the t-test as well to confirm its results.

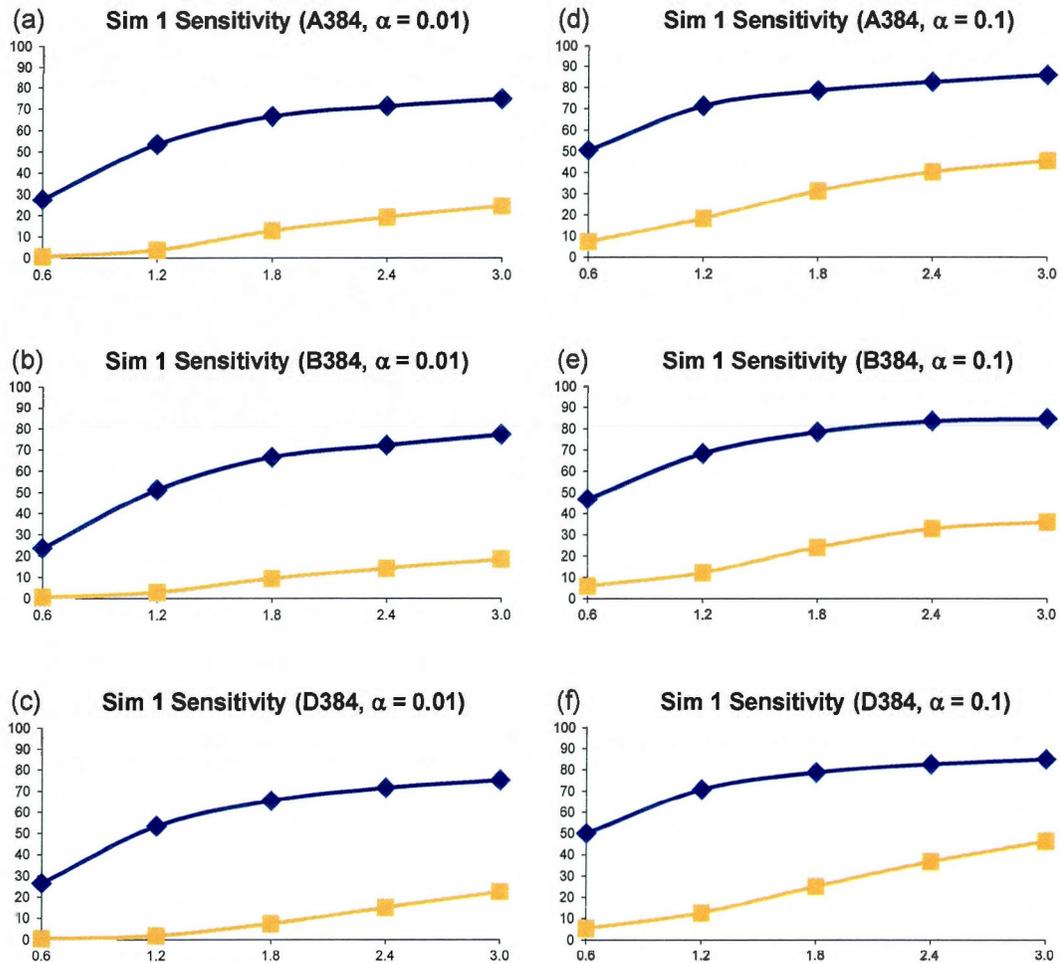
In addition to the experiments with simulated data, we applied the three discussed systematic error detection tests to real HTS data. Our goal was to evaluate the impact of systematic error on the hit selection process in experimental HTS. The obtained results (see Tables 2.2-2.5 and Figure 2.8) confirm the following fact: If raw HTS data are not treated properly for eliminating the effect of systematic error, then many (e.g., about 30% of hits in the case of McMaster *Test assay*, as reported in Table 2.5) of the selected hits may be due to the presence of systematic error and, at the same time, many promising compounds may be missed during hit selection. A special attention should be paid to control the results of aggressive data normalization procedures, such as B-score, that could easily do more damage by introducing biases in raw HTS data and, therefore, lead to the selection of many false positive hits even in the situations when the data don't contain any kind of systematic error.

Our general conclusion is that a successful assessment of the presence of systematic error in experimental HTS assays is achievable when the appropriate statistical methodology is used. Namely, the t-test should be carried out by HTS researchers to pre-process raw HTS data. This test should help improve the "quality" of selected hits by discarding many potential false positives and suggesting new, and eventually real, active compounds. The t-test should be used in conjunction with data correction techniques such as: Well correction (Makarenkov et al. 2006, Makarenkov et al. 2007), when row or column systematic error (detected by the test) repeats across all plates of the assay, and B-score (Brideau et al. 2003) or trimmed-mean polish score (Malo et al. 2010), when systematic error varies across plates. Thus, we recommend adding an extra preliminary systematic error detection and correction step in all HTS processing software and using consensus hits in order to improve the overall accuracy of HTS analysis.

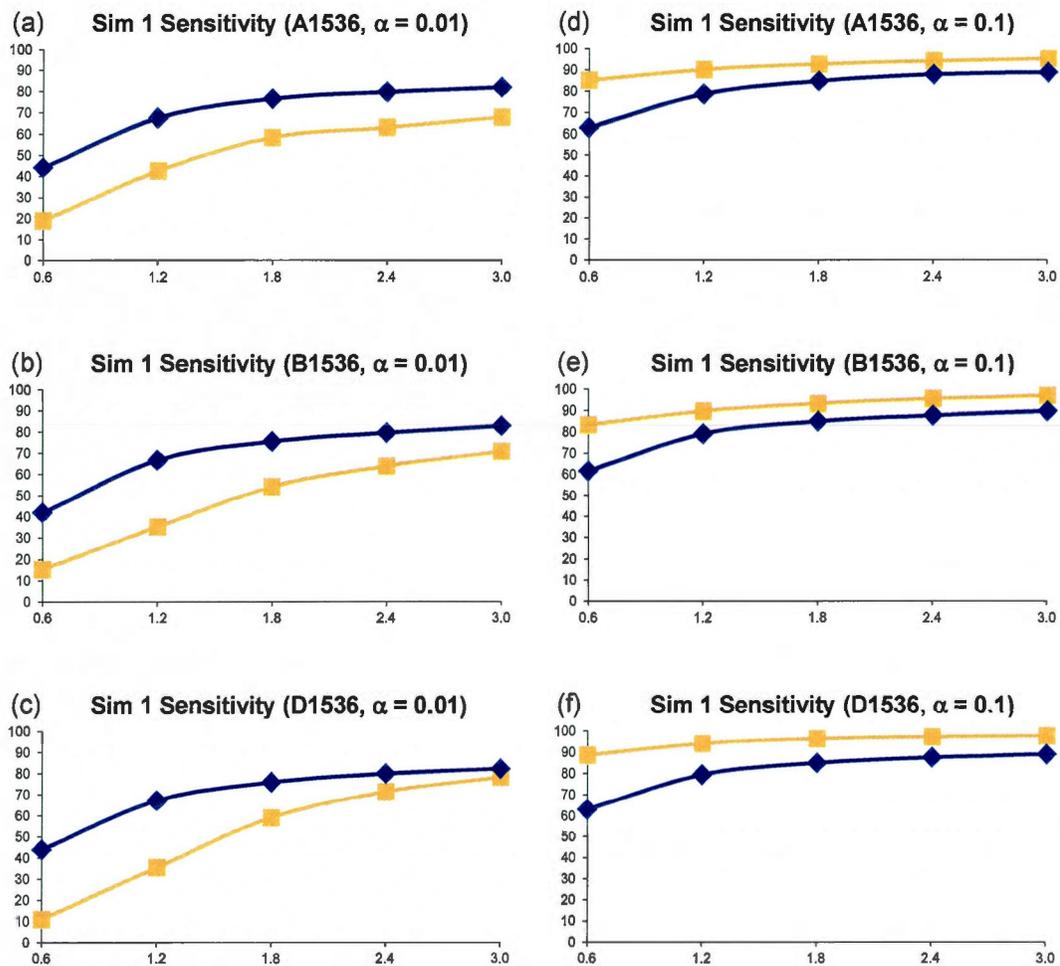
## 2.6 Supplementary Figures



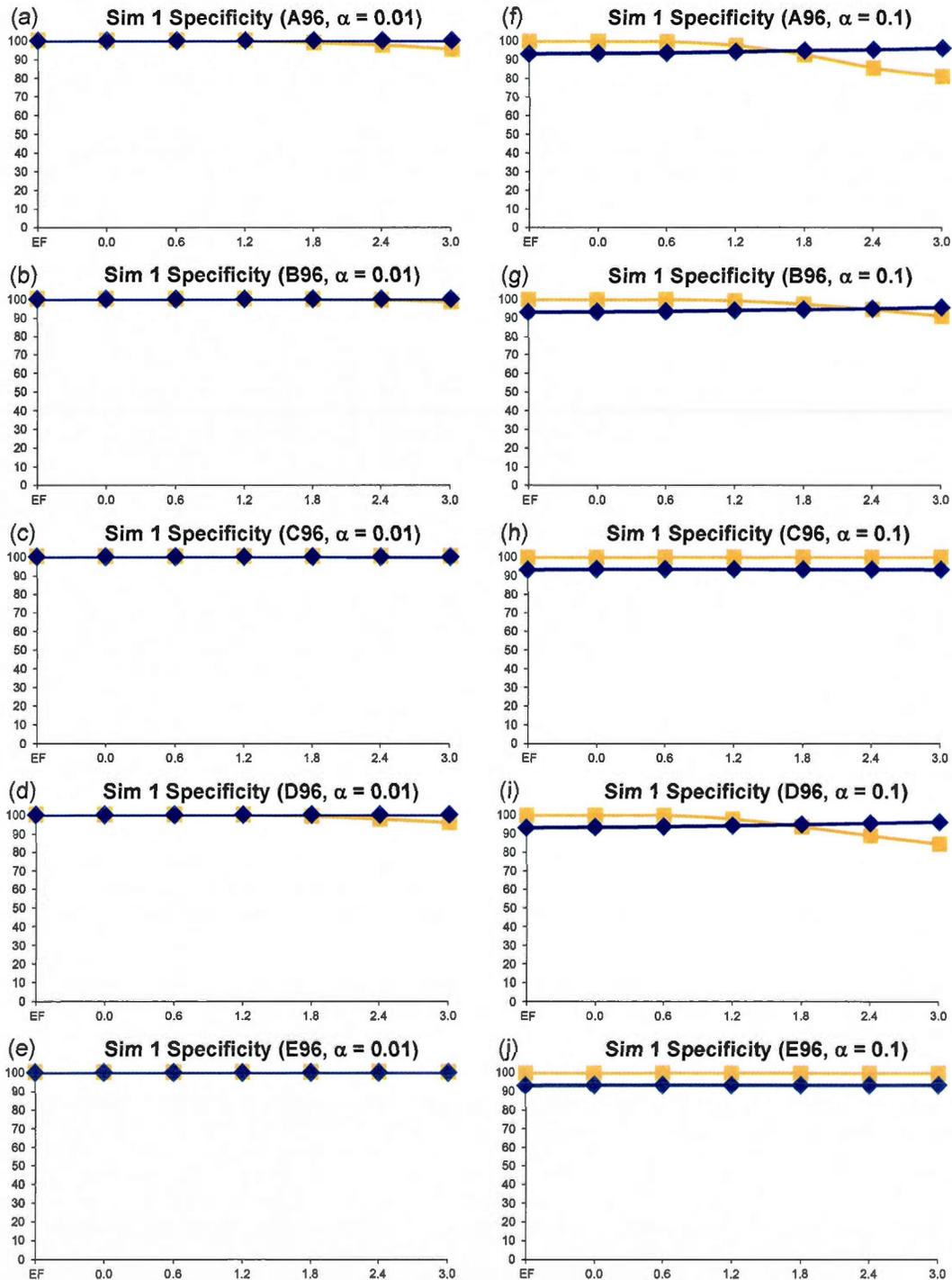
**Figure 2.9 Simulation 1, Plate Size: 96 wells – Sensitivity (True Positive Rate). Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (c):  $\alpha = 0.01$ ; Second column: cases (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



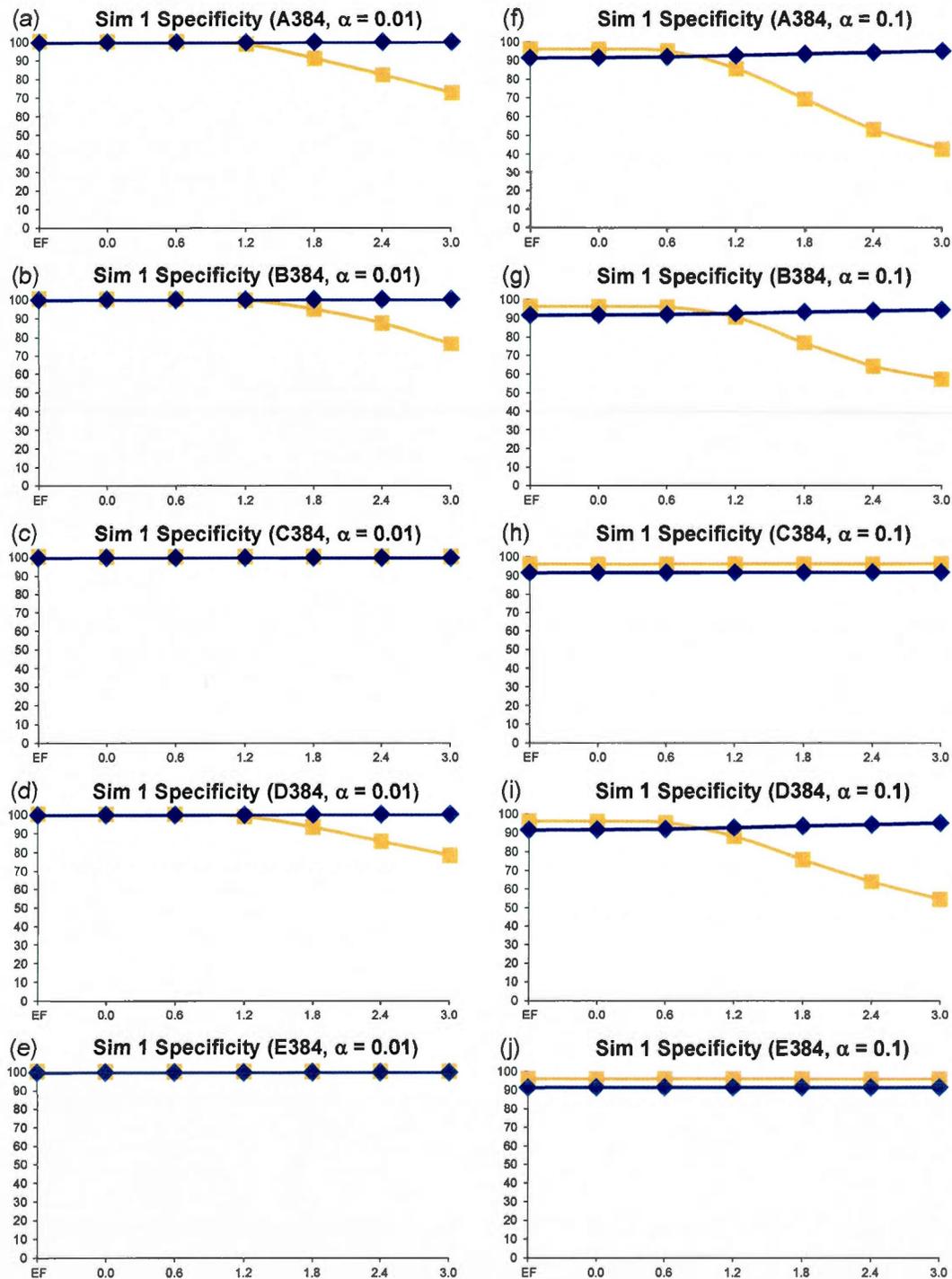
**Figure 2.10 Simulation 1, Plate Size: 384 wells – Sensitivity (True Positive Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (c):  $\alpha = 0.01$ ; Second column: cases (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



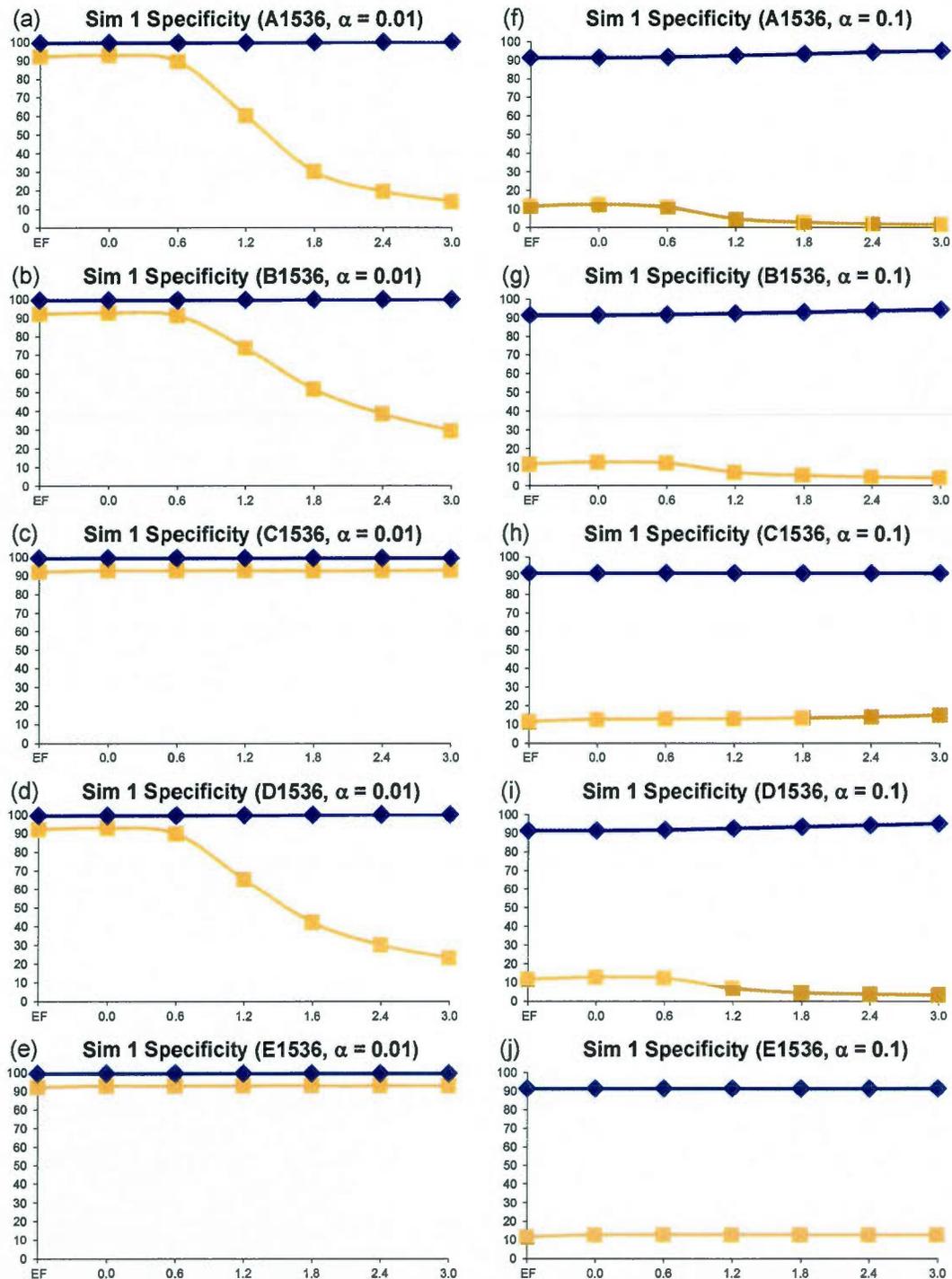
**Figure 2.11 Simulation 1, Plate Size: 1536 wells – Sensitivity (True Positive Rate). Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (c):  $\alpha = 0.01$ ; Second column: cases (d) - (f):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



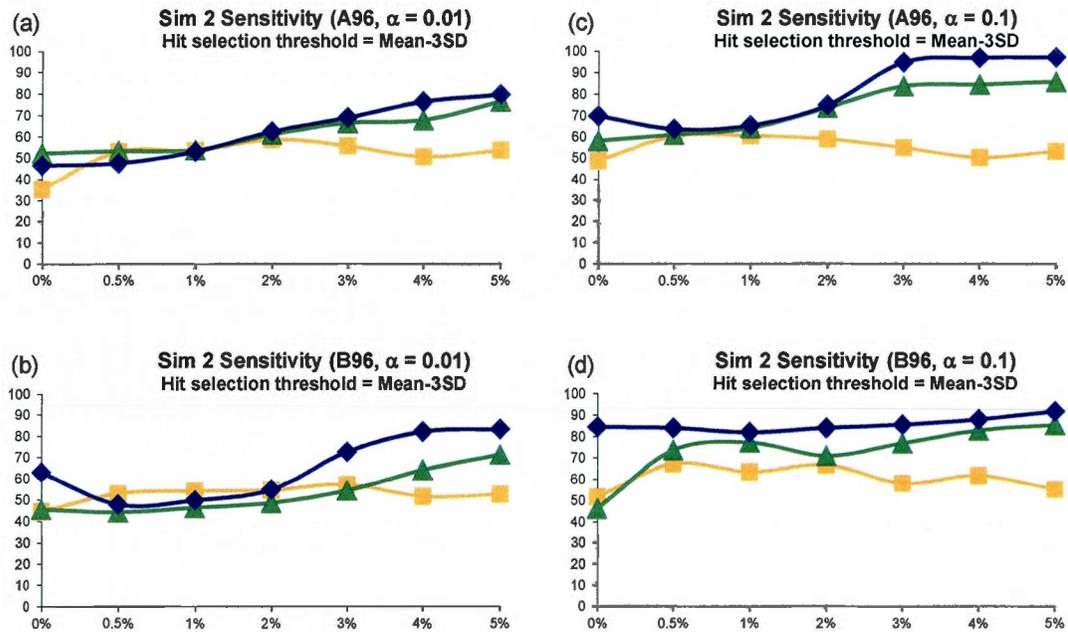
**Figure 2.12 Simulation 1, Plate Size: 96 wells – Specificity (True Negative Rate). Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



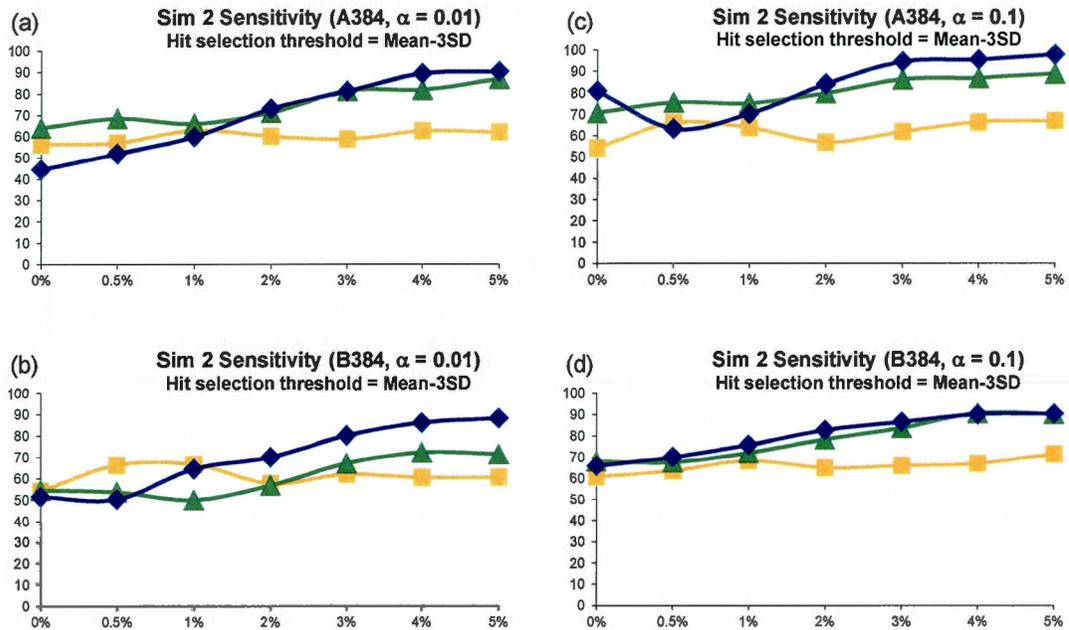
**Figure 2.13 Simulation 1, Plate Size: 384 wells – Specificity (True Negative Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



**Figure 2.14 Simulation 1, Plate Size: 1536 wells – Specificity (True Negative Rate). Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



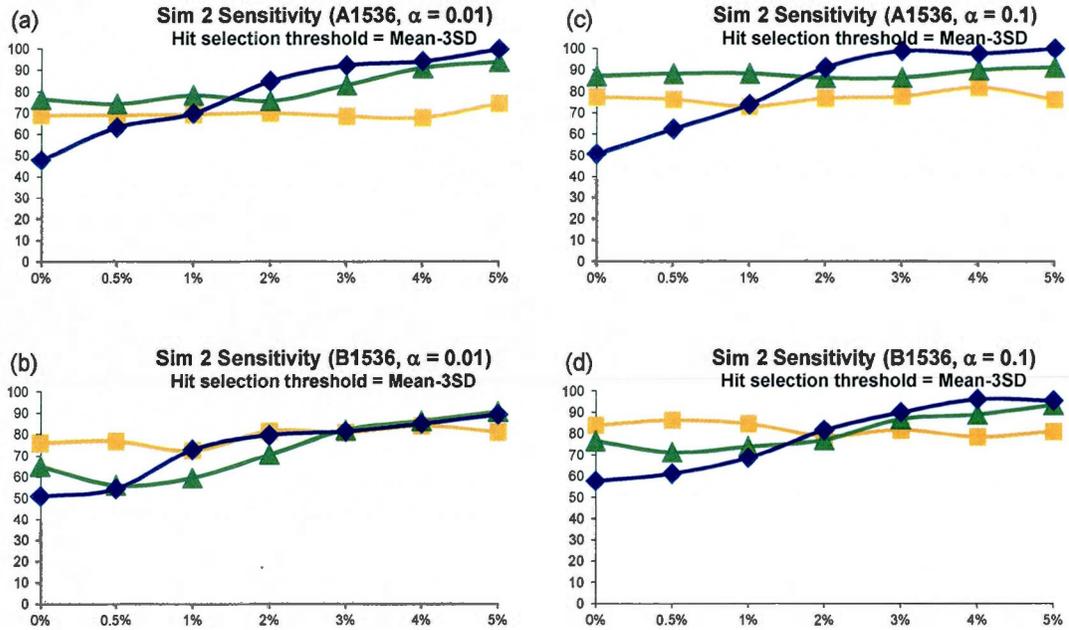
**Figure 2.15 Simulation 2, Plate Size: 96 wells - Sensitivity (True Positive Rate).**  
**Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases**  
**(a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection**  
**Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\triangle$ )  $\chi^2$  goodness-of-fit test.**



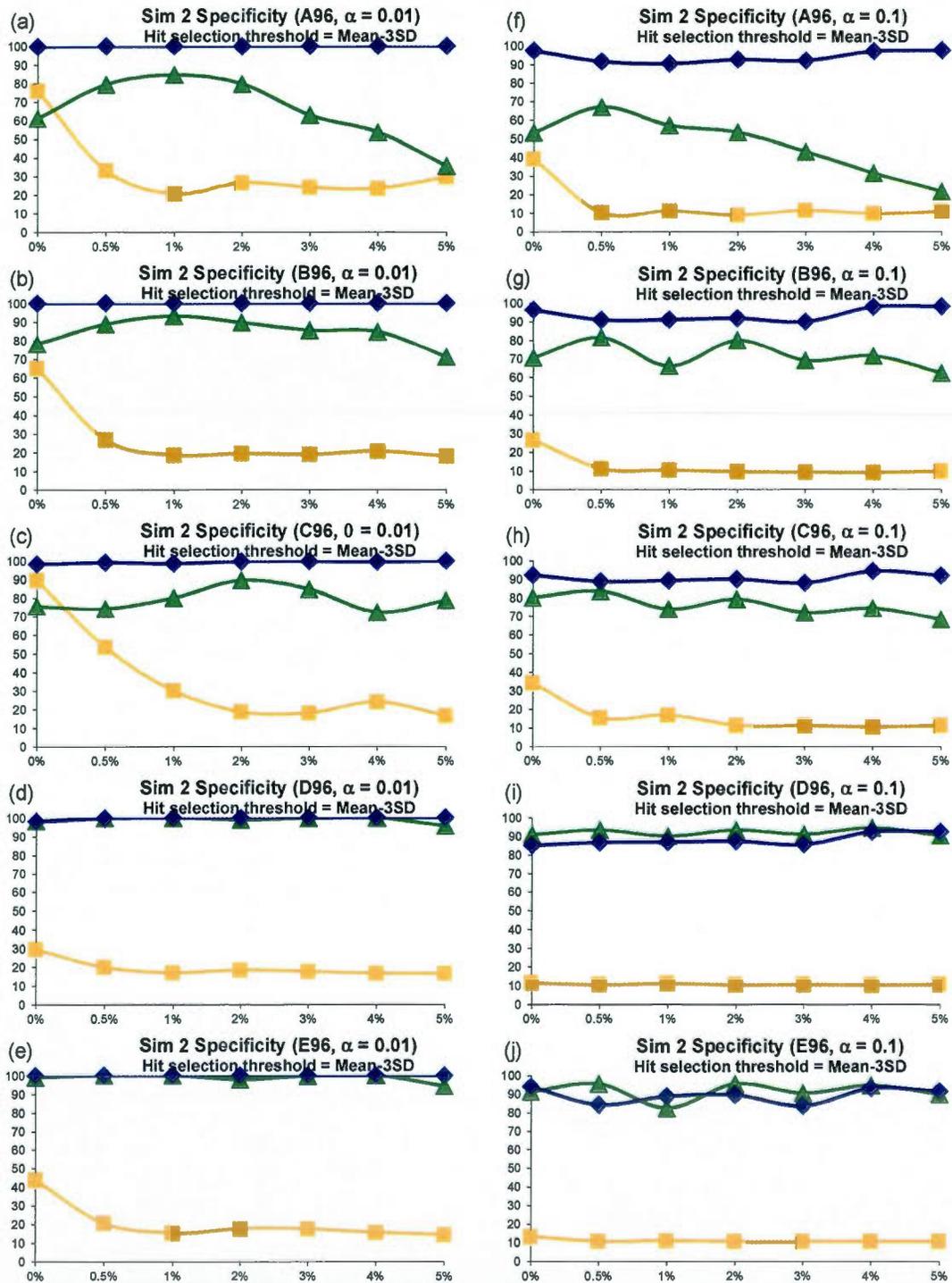
**Figure 2.16 Simulation 2, Plate Size: 384 wells - Sensitivity (True Positive Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases**

**(a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection**

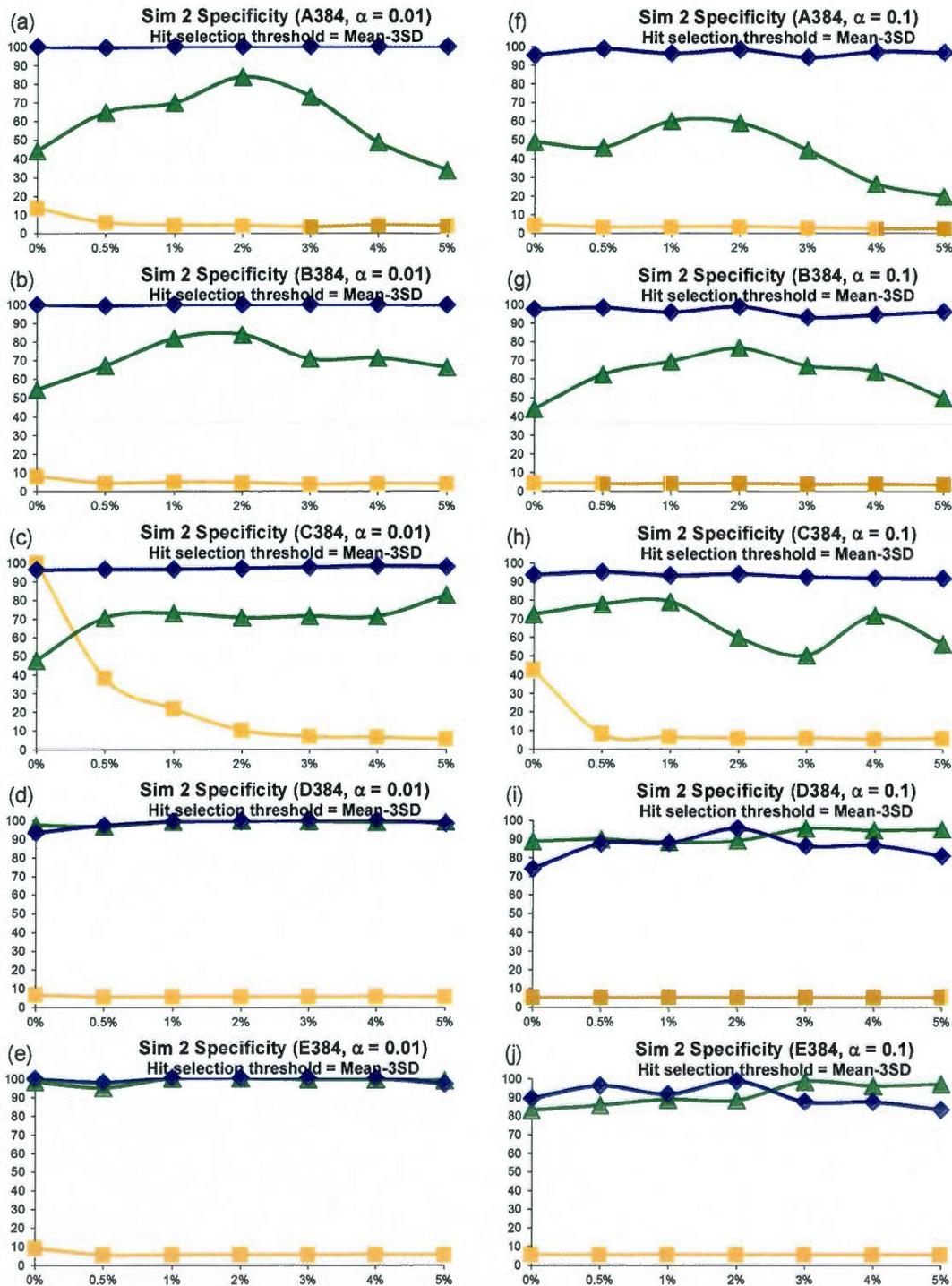
**Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**



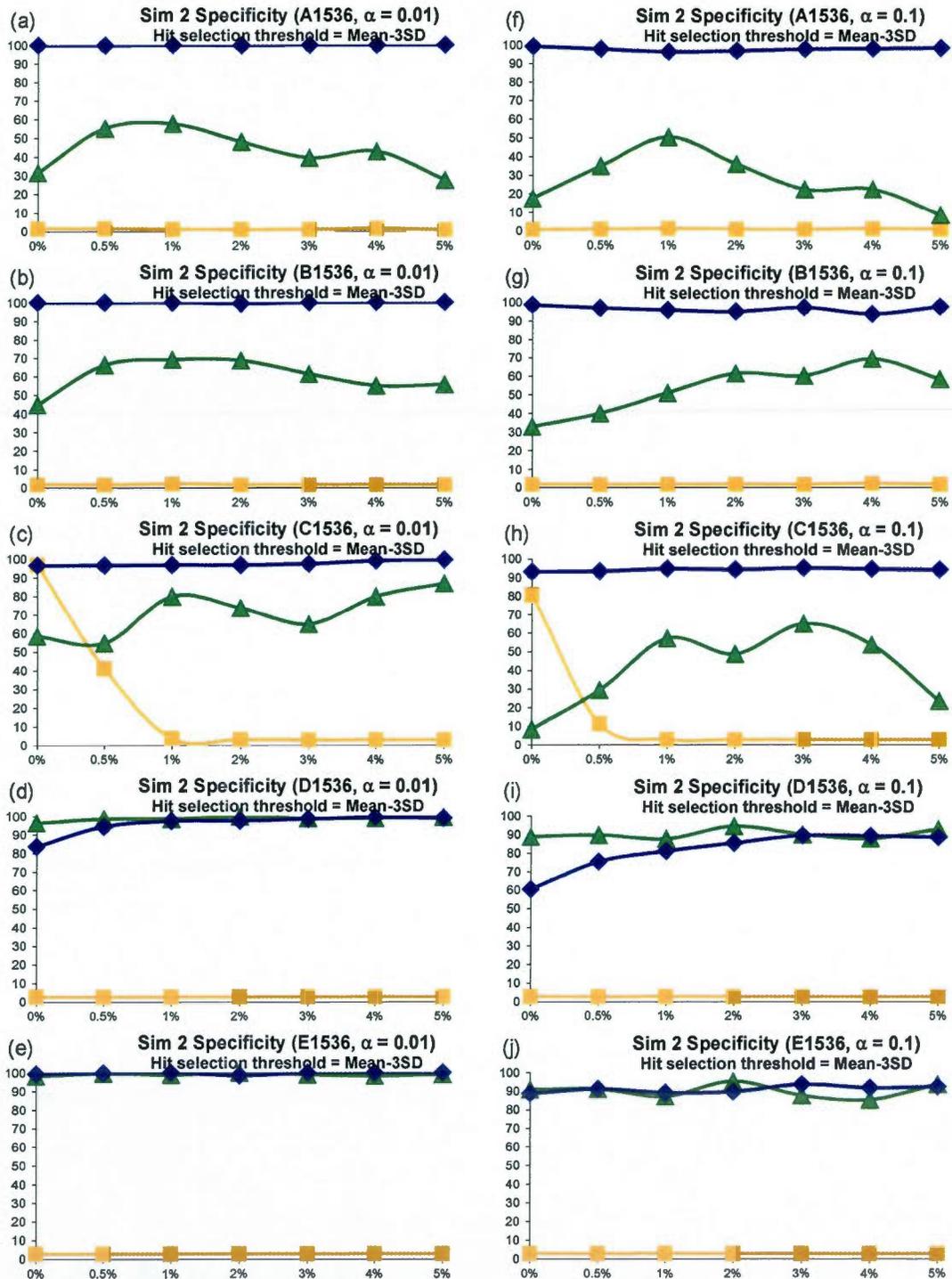
**Figure 2.17 Simulation 2, Plate Size: 1536 wells - Sensitivity (True Positive Rate). Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (b):  $\alpha = 0.01$ ; Second column: cases (c) - (d):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**



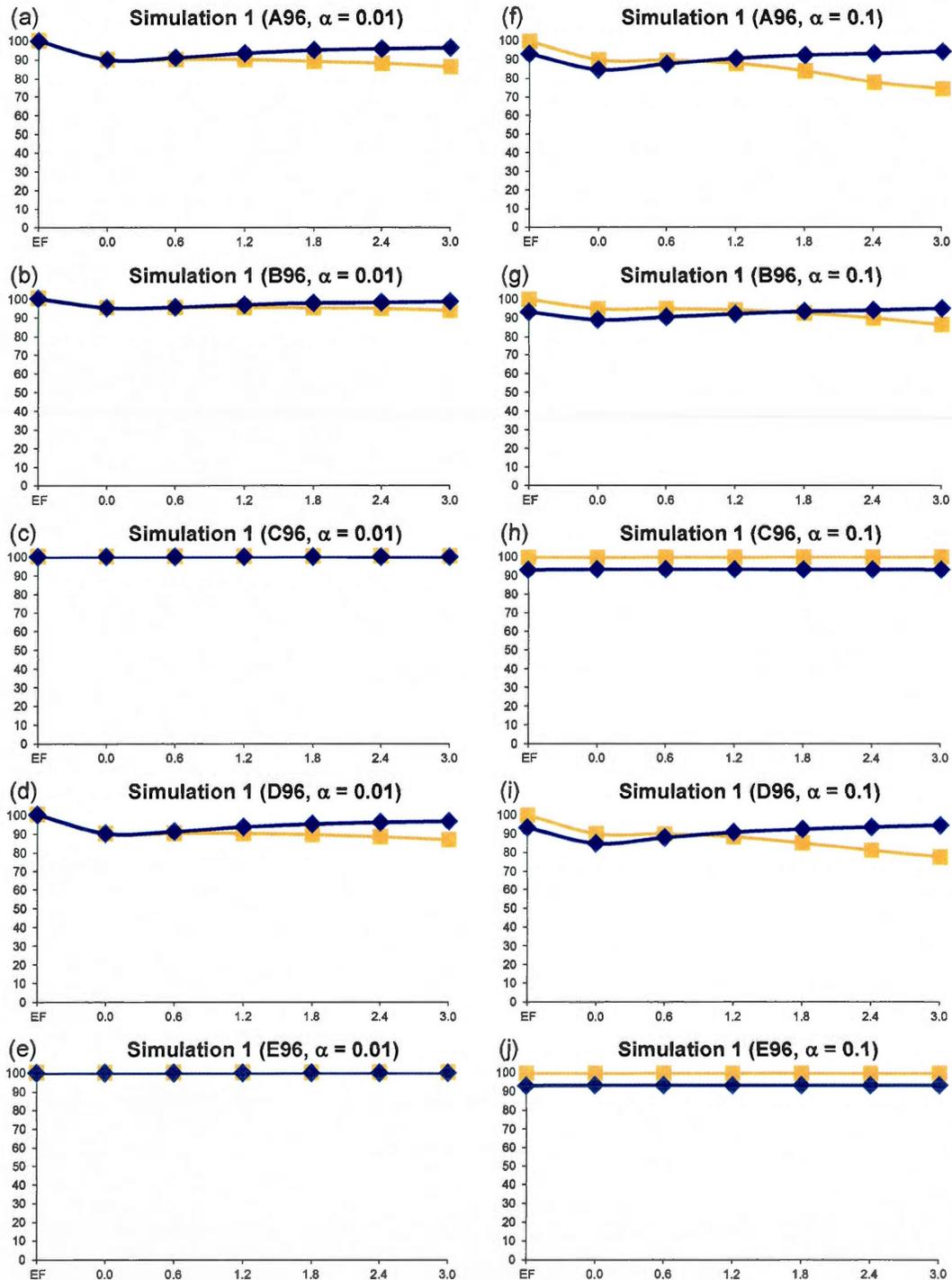
**Figure 2.18 Simulation 2, Plate Size: 96 wells - Specificity (True Negative Rate).** Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.



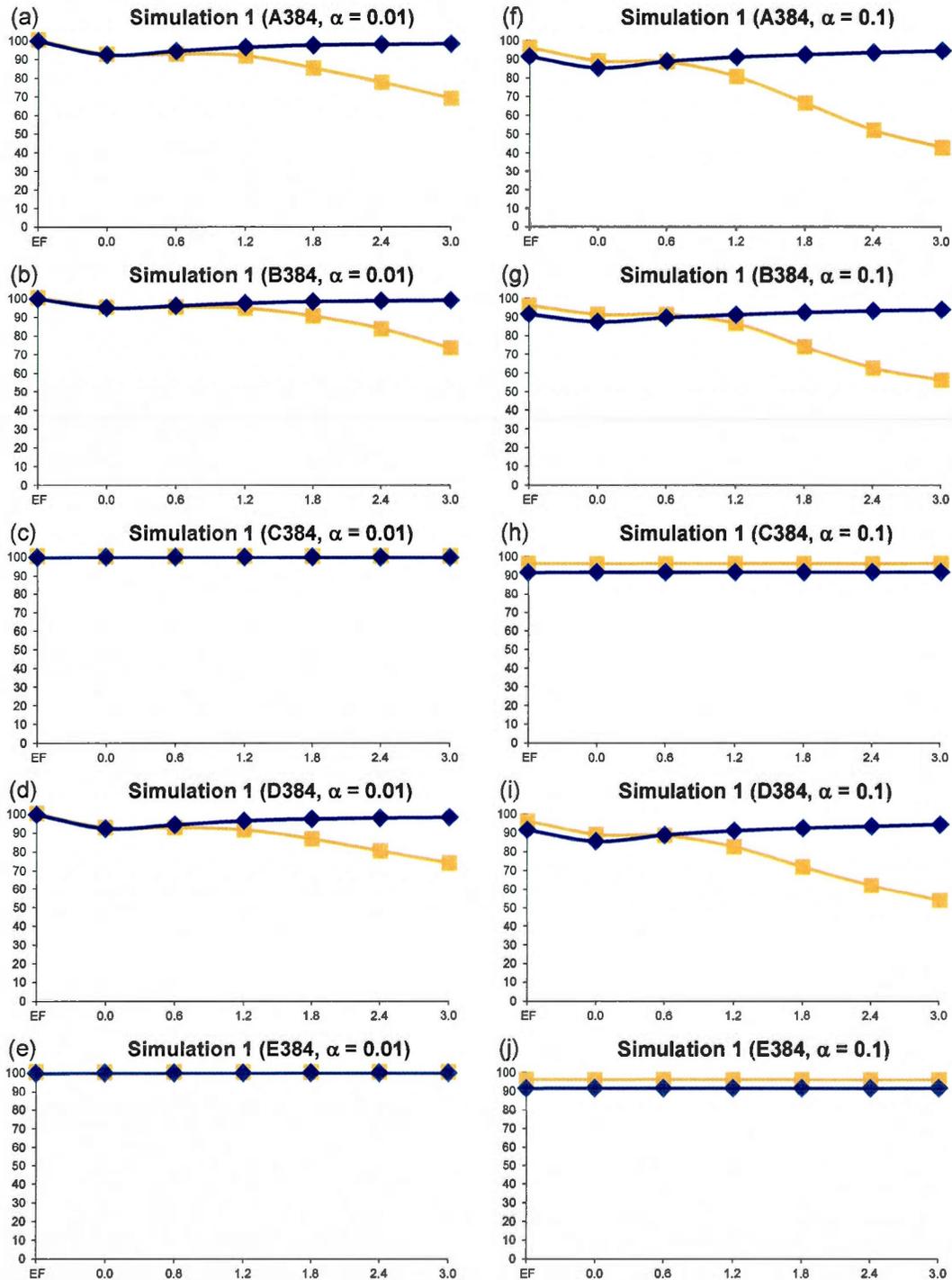
**Figure 2.19 Simulation 2, Plate Size: 384 wells - Specificity (True Negative Rate). Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**



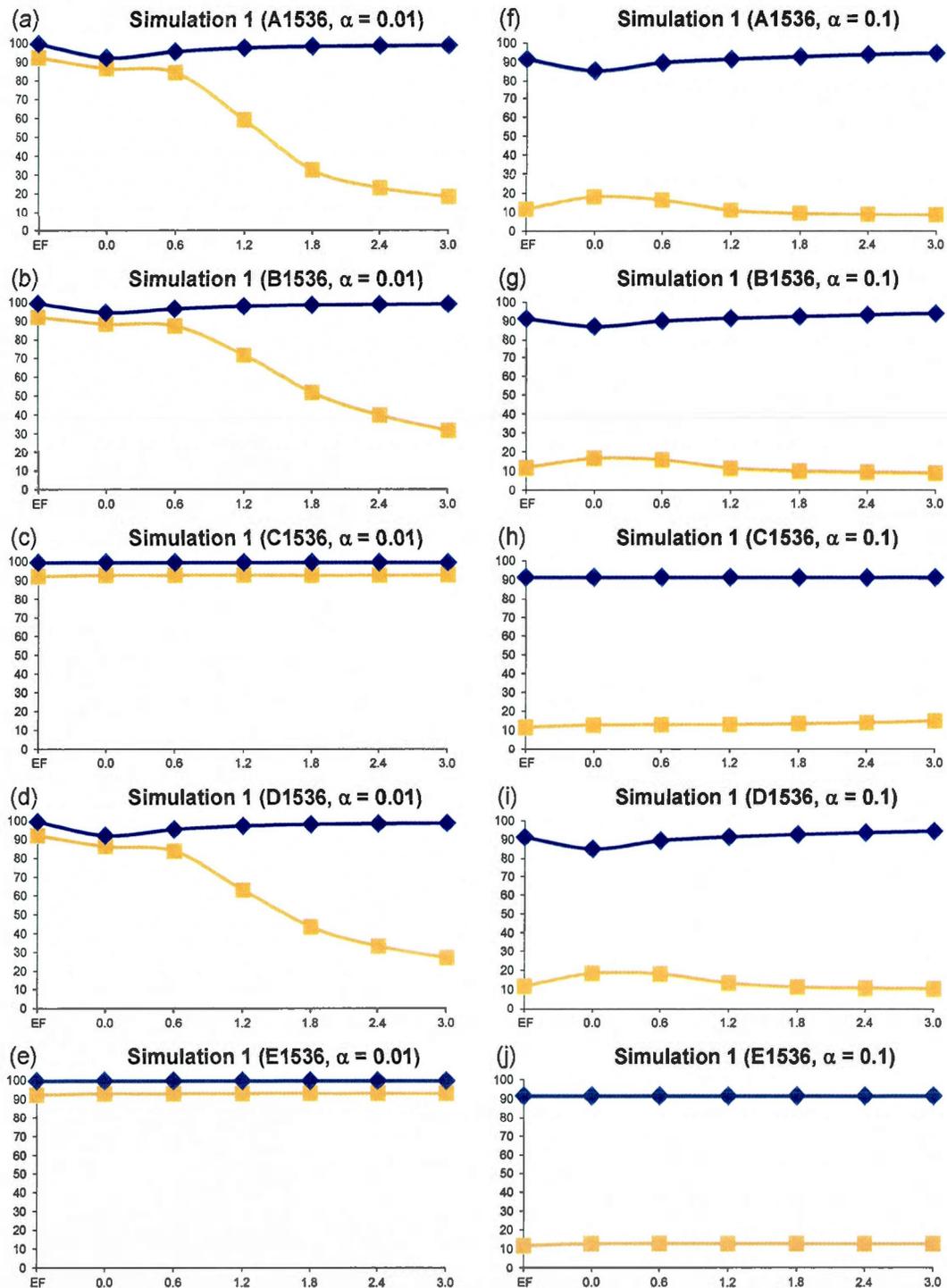
**Figure 2.20 Simulation 2, Plate Size: 1536 wells - Specificity (True Negative Rate).** Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.



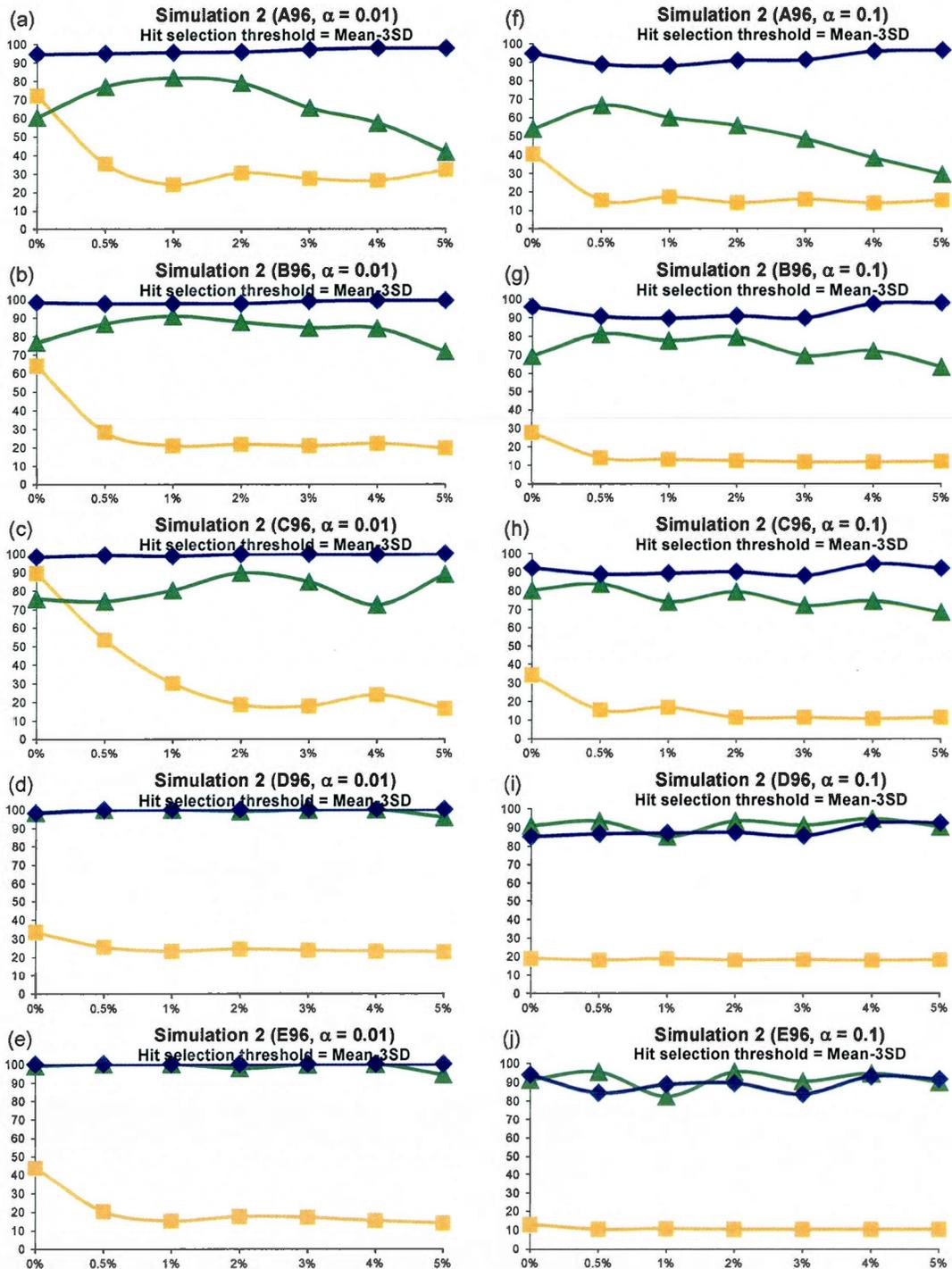
**Figure 2.21 Simulation 1, Plate Size: 96 wells – Success Rate. Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



**Figure 2.22 Simulation 1, Plate Size: 384 wells – Success Rate. Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



**Figure 2.23 Simulation 1, Plate Size: 1536 wells – Success Rate. Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test and ( $\square$ ) K-S test.**



**Figure 2.24 Simulation 2, Plate Size: 96 wells - Success Rate. Systematic error size: 10% (at most 2 columns and 2 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**

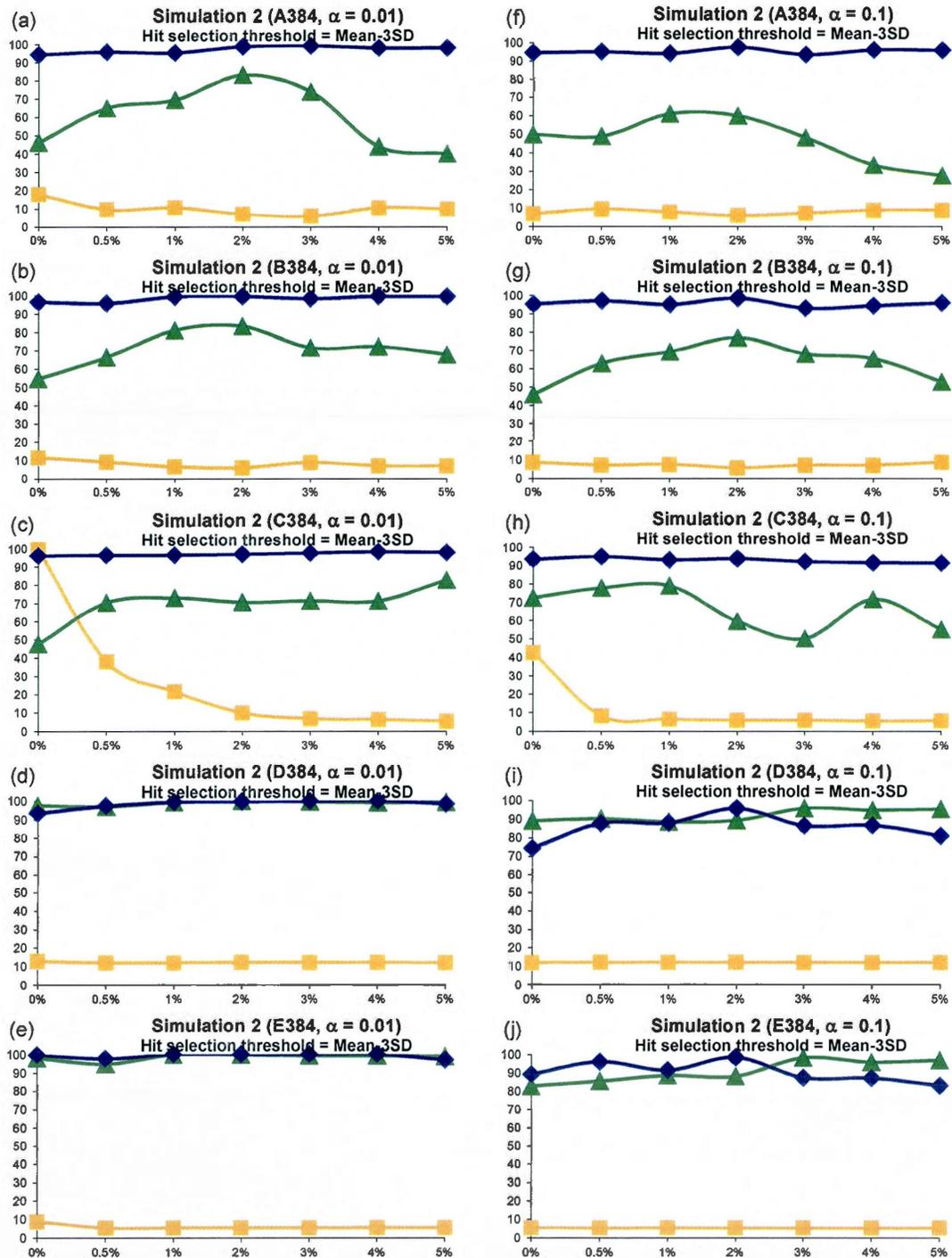
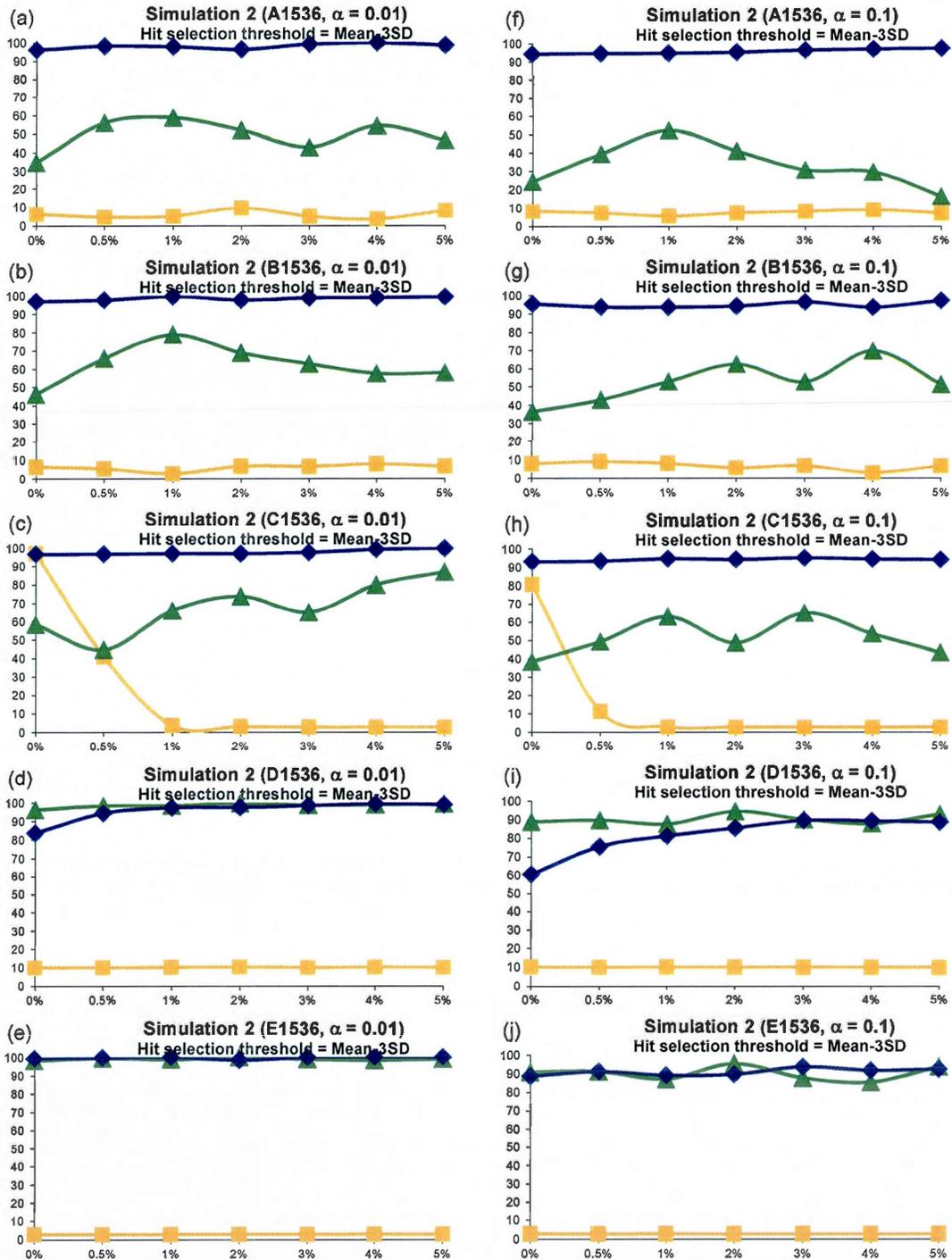
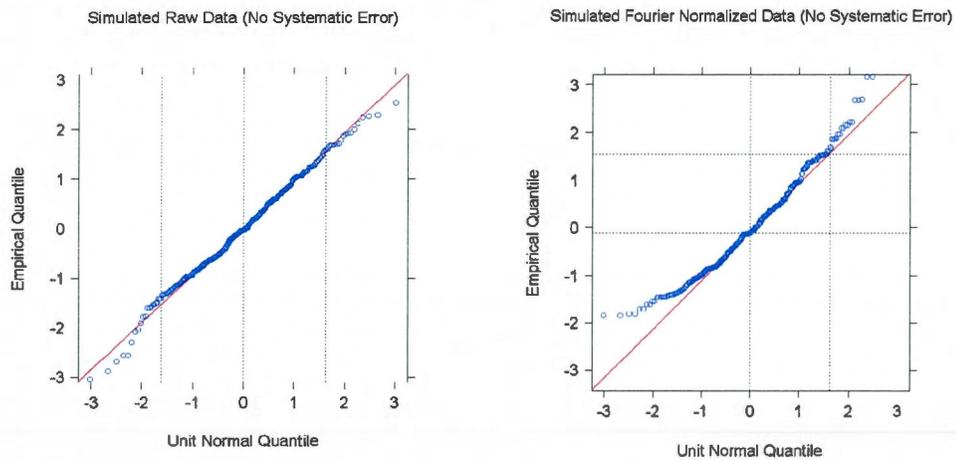


Figure 2.25 Simulation 2, Plate Size: 384 wells - Success Rate. Systematic error size: 10% (at most 4 columns and 4 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.



**Figure 2.26 Simulation 2, Plate Size: 1536 wells - Success Rate. Systematic error size: 10% (at most 8 columns and 8 rows affected). First column: cases (a) - (e):  $\alpha = 0.01$ ; Second column: cases (f) - (j):  $\alpha = 0.1$ . Systematic Error Detection Tests: ( $\diamond$ ) t-test, ( $\square$ ) K-S test and ( $\Delta$ )  $\chi^2$  goodness-of-fit test.**



**Figure 2.27** Data distribution before and after the application of the Discrete Fourier Transform (DFT) method. Data from one of the simulated 96-well plates before and after the application of Discrete Fourier Transform. The raw data followed a normal distribution and contained *random error only* (i.e., systematic error was not added). The raw data show agreement with the normal distribution, both graphically (case a) and by the Kolmogorov-Smirnov test ( $KS = 0.03$ ,  $p = 0.5$ ). However, after the application of Discrete Fourier Transform, the data deviate from normality as shown in the graph (case b) and by the Kolmogorov-Smirnov test ( $KS = 0.06$ ,  $p = 0.0018$ ).

## CHAPTER III

### TWO EFFECTIVE METHODS FOR CORRECTING EXPERIMENTAL HIGH-THROUGHPUT SCREENING DATA

This chapter is a reproduction of the following article:

Dragiev P., Nadon R. and Makarenkov V. (2012) Two effective methods for correcting experimental high-throughput screening data. *Bioinformatics*, 28 (13), 1775–1782.

Latest scientific and technical progress allowed millions of chemical compounds to be tested in a single HTS campaign. Working at large scale makes it difficult to ensure that the experimental conditions remain constant throughout the whole experiment. Many technical and procedural factors as well as changes in the environmental factors can cause the situation when the activity levels of certain compounds are systematically over- or underestimated. The presence of systematic error in the raw HTS data affects negatively the hit selection process, generating false positives and false negative hits (Makarenkov et al. 2007).

Different error correction methods have been developed to eliminate or reduce the effect of systematic error in experimental HTS (Brideau et al. 2003, Kevorkov and Makarenkov 2005, Makarenkov et al. 2007, Malo et al. 2010, Carralot et al. 2012). All these methods modify all the data of the given plate or the given assay. Therefore, when applied to error-free data, such an error correction introduces a bias into the data (see also Chapter II). In addition, the most widely-used error correction methods, like B-score (Brideau et al.

2003), change the scale of the corrected data, making them incomparable with the remaining unmodified data of the assay.

Chapter III presents two new systematic error correction methods meant to address the above-mentioned issues. Here, we assume that the assay plates affected by systematic error are known. Those plates and the error locations can be determined using the error detection tests described in Chapter II. The two new methods were designed to modify only the data affected, or supposed to be affected, by systematic error. The error correction is carried in such a way that after the correction, the corrected measurements remain on the same scale with the original raw data. Moreover, we also propose a general data correction framework capable of correcting screen-based and plate-based types of systematic error.

### 3.1 Abstract

**Motivation:** Rapid advances in biomedical sciences and genetics have increased the pressure on drug development companies to promptly translate new knowledge into treatments for disease. Impelled by the demand and facilitated by technological progress, the number of compounds evaluated during the initial high-throughput screening (HTS) step of drug discovery process has steadily increased. As a highly-automated large-scale process, HTS is prone to systematic error caused by various technological and environmental factors. A number of error correction methods have been designed to reduce the effect of systematic error in experimental HTS (Brideau et al. 2003, Carralot et al. 2012, Kevorkov and Makarenkov 2005, Makarenkov et al. 2007, Makarenkov et al. 2007, Malo et al. 2010). Despite their power to correct systematic error when it is present, the applicability of those methods in practice is limited by the fact that they can potentially introduce a bias when applied to unbiased data. We describe two new methods for eliminating systematic error from HTS data based on a prior knowledge of the error location. This information can be obtained using a specific version of the t-test or of the  $\chi^2$  goodness-of-fit test as discussed in Dragiev et al. (2011). We will show that both new methods constitute an important improvement over the standard practice of not correcting for systematic error at all as well as over the B-score correction procedure (Brideau et al. 2003) which is widely used in the modern HTS. We will also suggest a more general data preprocessing framework where the new methods can be applied in combination with the Well Correction procedure (Makarenkov et al. 2007). Such a framework will allow for removing systematic biases affecting all plates of a given screen as well as those relative to some of its individual plates.

### 3.2 Introduction

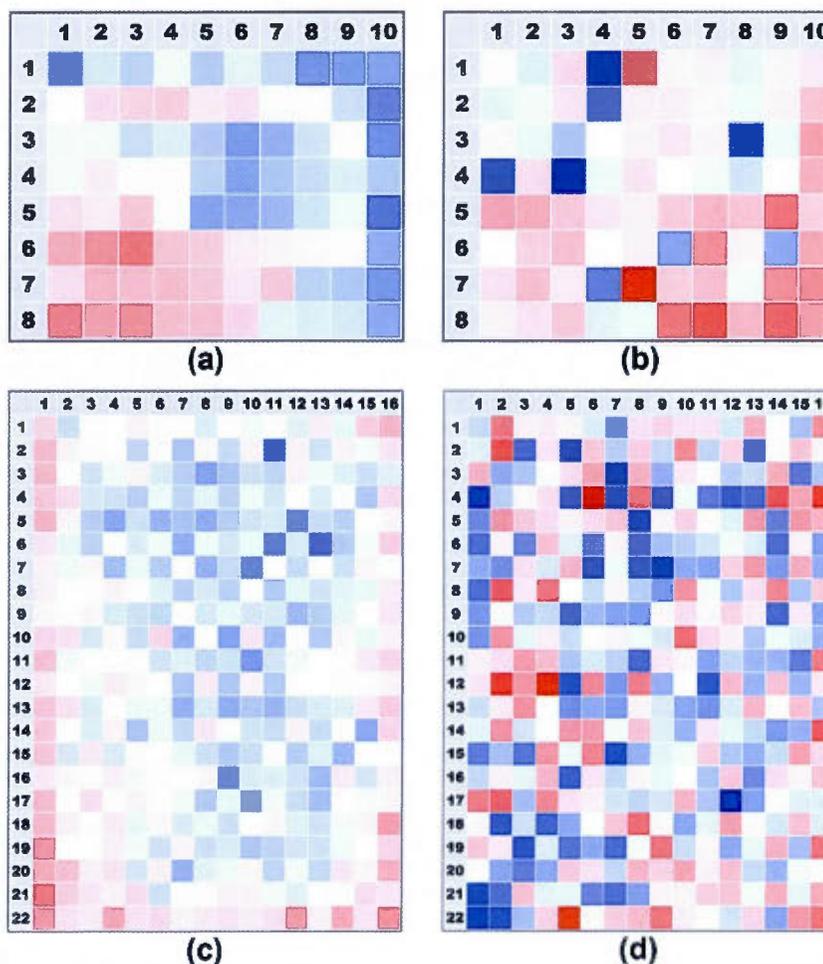
A typical drug development project starts with a candidate identification phase in which a large chemical compound library is tested against a given biological target (Malo et al. 2006). Complex high-throughput screening equipment is employed at this stage to obtain precise estimates of compound activity levels. The collected data are then used to identify the compounds that show the most promising “drug-like” activity behavior (Brideau et al. 2003, Malo et al. 2006). The selected compounds, called hits, typically undergo further testing to confirm their reproducibility and suitability for drug development. Depending on the nature

of the study, the hits may be compounds with the highest activation capacity (i.e., activation assays), inhibition capacity (i.e., inhibition assays), or both. The hit selection process assumes that the measurements taken by HTS equipment accurately represent the activity levels of the tested compounds. An important consideration for this to be true is that experimental conditions are the same for all compounds of the screen. Biases in the measurements can nonetheless appear, due to inconsistencies in the environmental factors, such as electricity, temperature, humidity or lighting changes (Heyse 2002, Makarenkov et al. 2007). Organizational factors can also have a significant systematic impact on the results of an HTS campaign. For example, differences in the incubation time allow the solvent evaporation to cause unintended variations in the solution concentrations. Highly sensitive readers in particular can detect subtle differences among the tested molecules which misdirect follow-up efforts when they are due to bias rather than to biology.

As a result of systematic bias causing under- or over-estimation of biological activity, inactive compounds may be incorrectly selected as hits (false positives), while promising (active) compounds may remain undetected (false negatives). In HTS, systematic error is usually column or row dependent (Brideau et al. 2003, Makarenkov et al. 2007). It is important to note that systematic error can either affect compounds placed in the same well, column or row location in all plates of the screen (i.e., screen-specific error) or affect a column or row of a specific single plate of the screen (i.e., plate-specific error).

Figure 3.1 illustrates the presence of positional effects in two publicly available experimental HTS datasets: McMaster Test dataset, used as a benchmark for the McMaster Data Mining and Docking Competition (Elowe et al. 2005; it contained the compounds intended to inhibit the *E. coli* Dihydrofolate reductase, DHFR) and a dataset provided by the Chemistry Department of Princeton University and consisting of a screen of compounds meant to inhibit the glycosyltransferase *MurG* function of *E. coli* (Helm et al. 2003). Figures 3.1a and 3.1c show activity levels averaged across all plates (i.e., assay background surfaces), whereas Figures 3.1b, and 3.1d show the activity levels of two selected single plates (from the McMaster and Princeton datasets, respectively). These examples demonstrate that systematic biases in HTS may have different screen-specific and plate-specific systematic deviations. For instance, in the McMaster dataset, the measurements in the column 10 are

globally over-estimated (Figure 3.1a), but in plate 1036 they are rather under-estimated (Figure 3.1b).



**Figure 3.1** Hit maps showing the presence of positional effects in the McMaster 1250-plate assay (Elowe et al. 2005) - (a) whole assay background surface, (b) plate 1036 measurements; and in the Princeton 164-plate assay (Helm et al. 2003) - (c) whole assay background surface, (d) plate 144 measurements. Color intensity is proportional to the compounds' signal levels (higher signals - potential target inhibitors, are shown in red).

Similarly, Figure 3.1c reveals apparent “edge effects” in the Princeton dataset with the values of the outer rows and columns being below the screen average. This effect was not observed, however, for all plates of the Princeton screen, with an evident over-estimation of the first column measurements detected in plate 144 (Figure 3.1d). Thus, systematic error

correction methods should be able first to recognize the character of systematic error affecting the data at hand and then remove it either from the whole assay and/or only from the specific plates where it was detected. In this article we describe two new methods for eliminating plate-specific systematic error and show how these methods can be applied in a more general correction framework that also includes the Well Correction procedure (Makarenkov et al. 2007) which allows for removing screen-specific systematic biases.

### 3.3 Methods

#### 3.3.1 Data preprocessing in HTS

In order to analyze experimental HTS assays, a data preprocessing treatment should be performed before the hit selection. Several data normalization and correction techniques, including the step of the quality control, have been proposed to preprocess experimental HTS data (Brideau et al. 2003, Carralot et al. 2012, Chapter II or Dragiev et al. 2011, Kevorkov and Makarenkov 2005, Makarenkov et al. 2007, Malo et al. 2006, Malo et al. 2010, Shun et al. 2011, Zhang et al. 1999, Zhang 2008). The most popular data normalization procedures used in HTS are as follows (see Chapter I): Percent of control that normalizes the measurements of the given compounds relative to the mean value of the plate's positive controls, Normalized percent inhibition in which the normalization is carried out relative to both positive and negative controls, and Z-score that consists in a zero mean and unit standard deviation normalization of the plate's measurements (Malo et al. 2006). Regarding data correction, mention the B-score (Brideau et al. 2003) and Well Correction (Makarenkov et al. 2006, Makarenkov et al. 2007) methods which will be considered in this study. Their main steps of these methods are as follows:

*B-score* (Brideau et al. 2003) is a robust normalization procedure commonly used in experimental HTS. Similarly to the above-mentioned normalizations, B-score sensibly handles plate-to-plate variability. In addition, it also corrects the raw plate measurements by removing the existing row and column positional effects. It assumes the following statistical model of HTS measurements (equation 3.1):

$$x_{ijp} = \mu_p + R_{ip} + C_{jp} + \varepsilon_{ijp}, \quad (3.1)$$

where  $x_{ijp}$  is the raw measurement of the compound in well  $(i, j)$  of a given plate  $p$ ,  $\mu_p$  is the plate average,  $R_{ip}$  is the systematic error affecting row  $i$ ,  $C_{jp}$  is the systematic error affecting column  $j$  and  $\varepsilon_{ijp}$  is the random noise affecting well  $(i, j)$  of this plate. B-score first employs a 2-way median polish procedure (Tukey 1977) to obtain the estimated values of  $x_{ijp}$ ,  $\mu_p$ ,  $R_{ip}$  and  $C_{jp}$  (equation 3.2):

$$\hat{x}_{ijp} = \hat{\mu}_p + \hat{R}_{ip} + \hat{C}_{jp}. \quad (3.2)$$

The residual,  $r_{ijp}$ , for the measurement in well  $(i, j)$  is then calculated as the difference between the raw measurement  $x_{ijp}$  and its fitted value  $\hat{x}_{ijp}$ :  $r_{ijp} = x_{ijp} - \hat{x}_{ijp}$ . Finally, the raw compound measurement is replaced with the corresponding residual adjusted by the plate's median absolute deviation ( $MAD_p$ , equation 3.3):

$$x'_{ijp} = \frac{r_{ijp}}{MAD_p}, \quad MAD_p = \text{median} \{ |r_{ijp} - \text{median}(r_{ijp})| \}, \quad (3.3)$$

where  $x'_{ijp}$  is the normalized measurement value.

*Well Correction* (Makarenkov et al. 2006, Makarenkov et al. 2007) is another combined data normalization and correction method designed to compensate for positional effects affecting rows, columns or individual wells, and appearing in all plates of the screen (i.e., screen-specific error). Well Correction includes the two following steps:

1. For each well location of the screen, a linear or polynomial least-squares approximation is carried out for the compound measurements located in that well over all plates of the screen. This approximation is performed separately for each well location.
2. The approximated entities within the same well location are then normalized over all plates of the screen using Z-score. This normalization is performed separately for each well location.

Once the data normalization and correction steps are completed, a hit selection procedure, meant to identify the compounds that will be promoted to leads, is carried out. The most popular strategy for hit selection proceeds by the identification of the compounds whose activity levels exceed a predefined threshold (Malo et al. 2006). Typically, the hit selection threshold is expressed in terms of the mean,  $\mu$ , and the standard deviation,  $SD$ , of the observed measurements. A commonly used approach selects as hits the compounds whose activity levels deviate from the mean value  $\mu$  for more than  $3SD$ .

Despite their ability to eliminate systematic error, HTS preprocessing techniques cannot guarantee the recovery of correct hits. In our previous works (Makarenkov et al. 2007, Chapter II or Dragiev et al. 2011), we showed that a misapplication of error correction methods on error-free HTS data introduces a significant bias that affects very negatively the accuracy of the hit selection process. For instance, a simulation study described in Makarenkov et al. 2007) suggests that the B-score method is unable to cope with screen-specific systematic error (see Figures 2 and 3 in the latter article) and that the Well Correction method is not suited for eliminating plate-specific systematic error (see Figure 4 in the latter article). Hence, error correction methods should be used with caution and only when the presence of systematic noise in the data has been confirmed by statistical tests. In our recent work (see Chapter II or Dragiev et al. 2011), we described how individual HTS plates can be assessed for presence of systematic error, thus facilitating the decision regarding the application of data correction techniques.

### ***3.3.2 Two new data correction methods***

Here we present two new methods for HTS systematic error correction, called *Matrix Error Amendment* (MEA) and *Partial Mean Polish* (PMP). Both methods rely on prior information concerning the location of rows and columns of individual plates that are systematically over- or underestimated. Such information might be available through the analysis of an individual plate (or entire screen) background (Kevorkov and Makarenkov 2005) or can be acquired using a specific version of the t-test or of the  $\chi^2$  goodness-of-fit test (Chapter II or Dragiev et al. 2011; see also the Supplementary Materials section for the

application of these tests in the HTS context). Both MEA and PMP methods are applied on a plate-by-plate basis.

Let  $X$  be a plate of HTS measurements with  $m$  rows and  $n$  columns. Let  $x_{ij}$  be the measurement of the compound located in well  $(i, j)$  of  $X$  and let  $\mu$  be the mean value of all measurements of plate  $X$  that are not affected by systematic error.

In the case when plate  $X$  is free of systematic error, we can expect that the mean of the values in a given row  $i$  ( $i = 1, 2, \dots, m$ ) does not deviate substantially from  $\mu$ , which in this case is the mean of all measurements on the plate:  $\sum_{j=1}^n x_{ij} \approx n\mu$ . Similarly, for a given column

$j$  ( $j = 1, 2, \dots, n$ ) of  $X$ , we expect that:  $\sum_{i=1}^m x_{ij} \approx m\mu$ .

Assume that  $X$  is affected by systematic error. Let  $r_1, r_2, \dots, r_p$  ( $p < m$ ) be the set of rows of  $X$ , and  $c_1, c_2, \dots, c_s$  ( $s < n$ ) be the set of columns of  $X$ , where the presence of systematic error has been confirmed. It is worth noting that the set  $r_1, r_2, \dots, r_p$  can represent any subset of the complete set of rows  $1, 2, \dots, m$  and the set  $c_1, c_2, \dots, c_s$  can represent any subset of the complete set of columns  $1, 2, \dots, n$  of plate  $X$ . The only necessary condition for the application of the new methods is the presence in  $X$  of at least one row and at least one column not affected by systematic error. Let  $e_{r_i}$  be the unknown value of systematic error affecting row  $r_i$  and  $e_{c_j}$  be the unknown value of systematic error affecting column  $c_j$ . The following fourfold set of linear equations can be composed:

$$\sum_{j=1}^n x_{r_i j} - n e_{r_i} - \sum_{j=1}^s e_{c_j} = n\mu, \quad (3.4)$$

$$\sum_{i=1}^m x_{i c_j} - m e_{c_j} - \sum_{i=1}^p e_{r_i} = m\mu, \quad (3.5)$$

$$\sum_{j=1}^n x_{ij} - \sum_{j=1}^s e_{c_j} = n\mu, \quad (3.6)$$

$$\sum_{i=1}^m x_{ij} - \sum_{i=1}^p e_{r_i} = m\mu, \quad (3.7)$$

where equation (3.4) corresponds to rows  $r_1, r_2, \dots, r_p$  affected by row systematic error, equation (3.5) to columns  $c_1, c_2, \dots, c_s$  affected by column systematic error, equation (3.6) to rows not affected by row systematic error, and equation (3.7) to columns not affected by column systematic error.

### 3.3.2.1 Matrix Error Amendment Method

Systematic error in HTS does not typically affect all the columns and rows of a plate. The affected columns and rows are often those located on the plate edges (Brideau et al. 2003, Kevorkov and Makarenkov 2005). Thus, typically,  $p$  is much smaller than  $m$  and  $s$  is much smaller than  $n$ . The presence of rows and columns not affected by systematic error allows us to estimate  $\mu$  and leaves  $e_{r_i}$  and  $e_{c_j}$  the only unknowns in the linear system of equations (3.4–3.7), which have  $m+n$  equations and fewer than  $m+n$  unknowns.

The *Matrix Error Amendment* method consists of the two following steps:

1. Estimate the values of the row and column systematic errors  $\hat{e}_{r_i}$  and  $\hat{e}_{c_j}$  ( $i = 1, 2, \dots, p$  and  $j = 1, 2, \dots, s$ ), independently for every plate of the assay, by solving the system of linear equations (3.4–3.7).
2. Adjust the measurements of all compounds located in rows and columns of the plates affected by systematic error using the error estimates  $\hat{e}_{r_i}$  and  $\hat{e}_{c_j}$  determined in step 1.

Two approaches of solving the system of linear equations (3.4–3.7) were tested in our study. First, by combining all equations (3.4–3.7), we composed an overdetermined system of linear equations  $\mathbf{Ae} = \mathbf{b}$  with  $m+n$  equations and fewer than  $m+n$  unknowns, where  $\mathbf{A}$  was the matrix of the coefficients for the unknowns  $e_{r_i}$  and  $e_{c_j}$  ( $i = 1, 2, \dots, p$  and  $j = 1, 2, \dots, s$ ) combined in the vector  $\mathbf{e}$  of size  $p+s$ , and  $\mathbf{b}$  was the vector of free terms. We found that in all cases the matrix  $\mathbf{A}^T\mathbf{A}$  was singular, thus rendering inapplicable the standard least-square approximation method for solving overdetermined systems of linear equations. We were able, however, to find an approximate solution of this system by using the singular value

decomposition (SVD) method. Second, we also tested a simpler and computationally less intensive approach consisting of combining only equations (3.4) and (3.5) into the linear system (3.8), having exactly  $m+n$  equations and  $m+n$  unknowns. When  $m+n > 5$  the system (3.8) always has a unique solution which can be found using standard methods for solving linear equations systems (e.g., Gaussian elimination).

$$\begin{pmatrix} n & 0 & \dots & 0 & 0 & 1 & 1 & \dots & 1 & 1 \\ 0 & n & \dots & 0 & 0 & 1 & 1 & \dots & 1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & n & 0 & 1 & 1 & \dots & 1 & 1 \\ 0 & 0 & \dots & 0 & n & 1 & 1 & \dots & 1 & 1 \\ 1 & 1 & \dots & 1 & 1 & m & 0 & \dots & 0 & 0 \\ 1 & 1 & \dots & 1 & 1 & 0 & m & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 1 & \dots & 1 & 1 & 0 & 0 & \dots & m & 0 \\ 1 & 1 & \dots & 1 & 1 & 0 & 0 & \dots & 0 & m \end{pmatrix} \begin{pmatrix} e_{r_1} \\ e_{r_2} \\ \vdots \\ e_{r_{p-1}} \\ e_{r_p} \\ e_{c_1} \\ e_{c_2} \\ \vdots \\ e_{c_{s-1}} \\ e_{c_s} \end{pmatrix} = \begin{pmatrix} b_{r_1} \\ b_{r_2} \\ \vdots \\ b_{r_{p-1}} \\ b_{r_p} \\ b_{c_1} \\ b_{c_2} \\ \vdots \\ b_{c_{s-1}} \\ b_{c_s} \end{pmatrix}, \quad (3.8)$$

where  $b_{r_i} = \sum_{j=1}^n x_{r_i j} - n\mu$  and  $b_{c_j} = \sum_{i=1}^m x_{i c_j} - m\mu$ .

According to our simulation study, the second approach, which requires less computer power, generally provided better results in terms of systematic error identification (i.e., it yielded a higher hit detection rate, see the section Simulation study). Thus, its detailed results are presented in the section Results and Discussion.

The final step of the MEA method proceeds by subtracting the obtained systematic error estimates  $\hat{e}_{r_i}$  and  $\hat{e}_{c_j}$  from the raw plate measurements (equations 3.9-3.10). For all rows  $r_i$  ( $i = 1, 2, \dots, p$ ) affected by systematic error, we have:

$$x'_{r_i j} = x_{r_i j} - \hat{e}_{r_i}, \quad \text{for all } j: 1 \leq j \leq n, \quad (3.9)$$

and for all columns  $c_j$  ( $j = 1, 2, \dots, s$ ):

$$x'_{i c_j} = x_{i c_j} - \hat{e}_{c_j}, \quad \text{for all } i: 1 \leq i \leq m. \quad (3.10)$$

### 3.3.2.2 Partial Mean Polish Method

Denote by  $\mu_i$  the mean value of all measurements in row  $i$  and by  $\mu_j$  the mean value of all measurements in column  $j$  of plate  $X$ :

$$\mu_i = \frac{1}{n} \sum_{j=1}^n x_{ij} \quad \text{and} \quad \mu_j = \frac{1}{m} \sum_{i=1}^m x_{ij}$$

Equations (3.4) and (3.5) can be rewritten as equations (3.11) and (3.12):

$$ne_{r_i} = \sum_{j=1}^n x_{r_i j} - n\mu - \sum_{j=1}^s e_{c_j}, \quad (3.11)$$

$$me_{c_j} = \sum_{i=1}^m x_{i c_j} - m\mu - \sum_{i=1}^p e_{r_i}, \quad (3.12)$$

where  $\mu$  is the mean value of all measurements of  $X$  not affected by systematic error.

Dividing equations (3.11) and (3.12) by  $n$  and  $m$ , respectively, we obtain:

$$e_{r_i} = \mu_{r_i} - \mu - \frac{1}{n} \sum_{j=1}^s e_{c_j}, \quad (3.13)$$

$$e_{c_j} = \mu_{c_j} - \mu - \frac{1}{m} \sum_{i=1}^p e_{r_i}. \quad (3.14)$$

Since systematic error usually affects only a few columns and rows of HTS plates (e.g., row and column measurements on plate edges are often biased; for more details see Brideau et al. 2003 or Kevorkov and Makarenkov 2005) and causes an over or under-estimation of the affected measurements (i.e., the error values can be negative or positive), we can assume that the term consisting of the total column error divided by the number of columns has a negligible impact compared to the other terms in equation (3.13) and thus that the row systematic error of row  $r_i$  can be estimated as the difference between the mean value

of the entities in that row and the mean value  $\mu$  of the plate measurements that are not affected by systematic error:

$$\hat{e}_{r_i} = \mu_{r_i} - \mu. \quad (3.15)$$

Similarly, for the column  $c_j$ , we can expect that:

$$\hat{e}_{c_j} = \mu_{c_j} - \mu. \quad (3.16)$$

Based on the assumptions above, we can formulate the Partial Mean Polish iterative procedure (only a part of the plate's rows and columns, i.e., those affected by systematic bias, will be "polished" by the method). The means in this procedure can be easily replaced by the medians giving rise to Partial Median Polish method which could be viewed as an extension of a well-known Median Polish procedure by Tukey 1977 for the case when the error locations are known.

The main steps of the Partial Mean Polish method are the following:

1. Compute the mean value  $\mu$  of all entities of the given plate that are not affected by systematic error:

$$\mu = \frac{\sum_{i \notin R, j \notin C} x_{ij}}{(m-p)(n-s)}, \quad (3.17)$$

where  $R = \{r_1, r_2, \dots, r_p \mid 0 \leq p < m\}$  is a set of rows of  $X$  affected by systematic error and  $C = \{c_1, c_2, \dots, c_s \mid 0 \leq s < n\}$  is a set of columns of  $X$  affected by systematic error.

2. For each  $i$  ( $1 \leq i \leq p$ ), compute the mean value  $\mu_{r_i}$  of row  $r_i$  as:  $\mu_{r_i} = \frac{1}{n} \sum_{j=1}^n x_{r_i j}$ , and

then, using equation (3.15), the estimate of the row bias  $\hat{e}_{r_i}$  as:  $\hat{e}_{r_i} = \mu_{r_i} - \mu$ .

For each  $j$  ( $1 \leq j \leq s$ ) compute the mean value  $\mu_{c_j}$  of column  $c_j$  as:  $\mu_{c_j} = \frac{1}{m} \sum_{i=1}^m x_{ic_j}$ ,

and then, using equation (3.16), the estimate of the column bias  $\hat{e}_{c_j}$  as:

$$\hat{e}_{c_j} = \mu_{c_j} - \mu.$$

3. For all rows affected by systematic bias, adjust their measurements using the error estimates determined in step 2, i.e., for each  $i$  ( $1 \leq i \leq p$ ), and for each  $j$  ( $1 \leq j \leq n$ ):

$$x_{r_i j} = x_{r_i j} - \hat{e}_{r_i}.$$

For all columns affected by systematic error, adjust their measurements using the error estimates determined in step 2, i.e., for each  $j$  ( $1 \leq j \leq s$ ), and for each  $i$  ( $1 \leq i \leq$

$$m$$
):  $x_{ic_j} = x_{ic_j} - \hat{e}_{c_j}$ .

4. Compute the value of the convergence parameter  $\delta$ :  $\delta = \sum_{i=1}^p |\hat{e}_{r_i}| + \sum_{j=1}^s |\hat{e}_{c_j}|$ .

5. If  $\delta < \varepsilon$ , where  $\varepsilon$  is a selected convergence threshold, or if a fixed maximum number of iterations has been already carried out then return  $X$ , otherwise, repeat steps 2 to 5.

### 3.4 Results And Discussion

To evaluate the performances of the two introduced systematic error correction methods we first carried out simulations with artificially generated HTS measurements. We also applied both MEA and PMP methods to analyze the 1250-plate HTS screen produced at the HTS Laboratory of McMaster University (i.e., the Test dataset proposed as a benchmark for the McMaster Data Mining and Docking Competition, see Figure 3.1 and Elowe et al. 2005).

#### 3.4.1 Simulation study

The simulated data also consisted of 1250-plate assays. Plate sizes were 96-well plates (8 rows  $\times$  12 columns), 384-well plates (16 rows  $\times$  24 columns), and 1536-well plates (32 rows  $\times$  48 columns). Inactive compound measurements were generated according to the

standard normal distribution. Active compounds (hits) were added randomly to the plates to form assays with the following hit percentages: 0%, 0.5%, 1%, 2%, 3%, 4%, and 5%. Hit locations were chosen randomly within each plate (i.e., the probability that a given well contained a hit compound was the same for all wells of the plate, regardless of the well location within the plate). The hit measurements were generated according to the normal distribution with parameters  $\sim\mathcal{N}(\mu-5SD, SD)$ , where  $\mu$  and  $SD$  were the mean and standard deviation of the original dataset (obtained before the addition of hits; i.e.,  $\mu = 0$  and  $SD = 1$ ). Systematic row and column errors were added to randomly selected rows and columns of each plate. The rows and columns affected by systematic error were selected separately for each plate, and thus their locations differed from plate to plate. The values of systematic bias followed a normal distribution with parameters  $\sim\mathcal{N}(0, C)$ . The following values of the error standard deviation,  $C$ , were considered to generate assays affected by different degree of systematic error: 0,  $0.6SD$ ,  $1.2SD$ ,  $1.8SD$  and  $2.4SD$ . In order to mimic empirical HTS data, in our first simulations the effect of systematic error was limited to a few rows and columns only. Thus, at most 2 rows and 2 columns for 96-well plates, at most 4 rows and 4 columns for 384-well plates, and at most 8 row and 8 columns for 1536-well plates were affected by systematic bias. A small random error was also added to both hit and non hit measurements. The random error in all datasets followed a normal distribution with parameters  $\sim\mathcal{N}(0, 0.6SD)$ .

Formula 3.18 specifies the model we used to generate an error-affected measurement of the compound located in well  $(i, j)$  of plate  $p$ :

$$x'_{ijp} = x_{ijp} + e_{r_{ip}} + e_{c_{jp}} + rand_{ijp}, \quad (3.18)$$

where  $x'_{ijp}$  is the resulting measurement value,  $x_{ijp}$  is the original error-free measurement,  $e_{r_{ip}}$  is the systematic error affecting row  $i$  of plate  $p$ ,  $e_{c_{jp}}$  is the systematic error affecting column  $j$  of plate  $p$  and  $rand_{ijp}$  is the random error in well  $(i, j)$  of plate  $p$ .

Six data correction/hit selection methods were tested in our simulations. All tested methods comprised an identical hit selection step, but differed in the way the data were

processed before the hit selection. The hits were selected globally for each assay using the hit selection threshold of  $\mu_{hs} - 3SD_{hs}$  (i.e., all compounds with the measurements lower than  $\mu_{hs} - 3SD_{hs}$  were declared hits, where  $\mu_{hs}$  and  $SD_{hs}$  were respectively the mean and standard deviation of the entire assay after the addition of hits and systematic error). The six methods evaluated in our simulation study were the following:

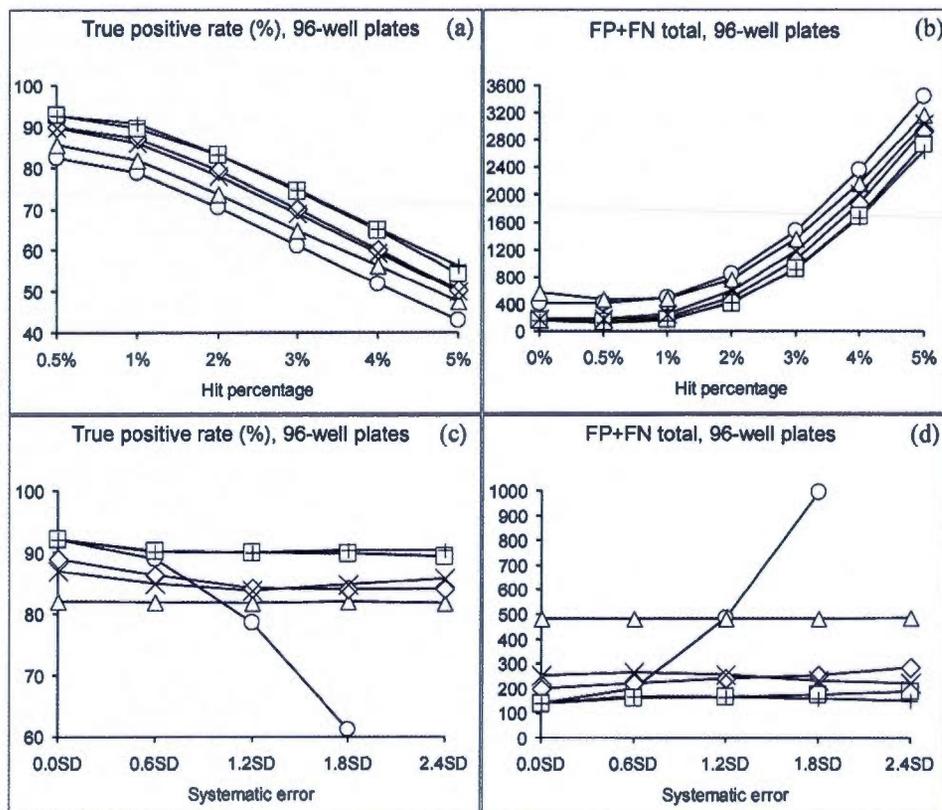
- Original data processing without any data correction;
- B-score correction method (Brideau et al. 2003);
- MEA method performed under the assumption that the exact locations of the error-affected rows and columns on each plate of the assay are known;
- MEA method performed for the rows and columns where systematic error was detected by the t-test (for more details, see Chapter II or Dragiev et al. 2011);
- PMP method performed under the assumption that the exact locations of the error-affected rows and columns on each plate of the assay are known;
- PMP method performed for the rows and columns where systematic error was detected by the t-test (for more details, see Chapter II or Dragiev et al. 2011).

In all experiments, we assessed the performances of the six data preprocessing methods by measuring the total number of false positives and false negatives, and by estimating the methods hit detection rate (i.e., true positive rate).

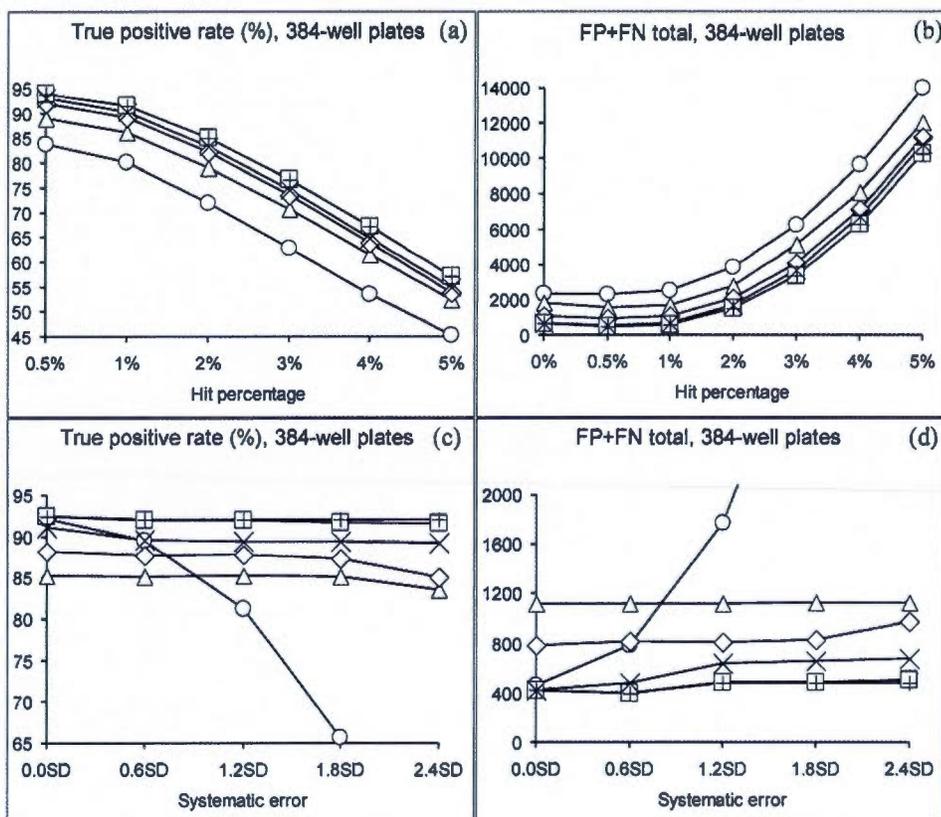
We conducted two series of experiments to evaluate the methods' performances depending on the hit percentage and the variance of systematic error. The first series of experiments used datasets with the fixed systematic error standard deviation of  $1.2SD$  and the hit percentage rate varying from 0% to 5% (there are no true positives for the case of 0% of hits; see Figures 3.2-3.4a).

The second series of experiments considered datasets with the fixed hit percentage of 1% and the systematic error standard deviation varying from 0 to  $2.4SD$ . Some 500 datasets were generated for both series of experiments and for each parameter combination. Figures 3.2, 3.3 and 3.4 present the average results obtained for the two series of experiments for the

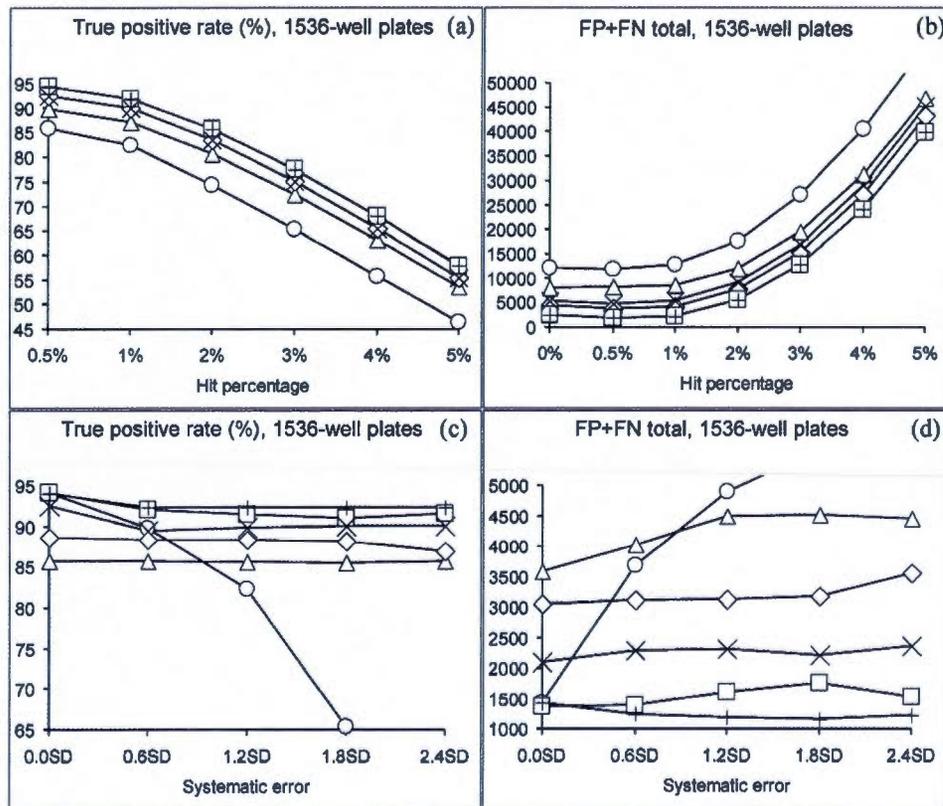
96-well, 384-well and 1536-well plates, respectively. Furthermore, we conducted additional simulations in order to assess the performances of the MEA and PMP methods in the situation when up to 50% of the plates' rows and columns were affected by systematic bias. The graphics depicting relative performances of the MEA, PMP, B-score and no-correction strategies in this case are presented in Figures 3.5 to 3.7.



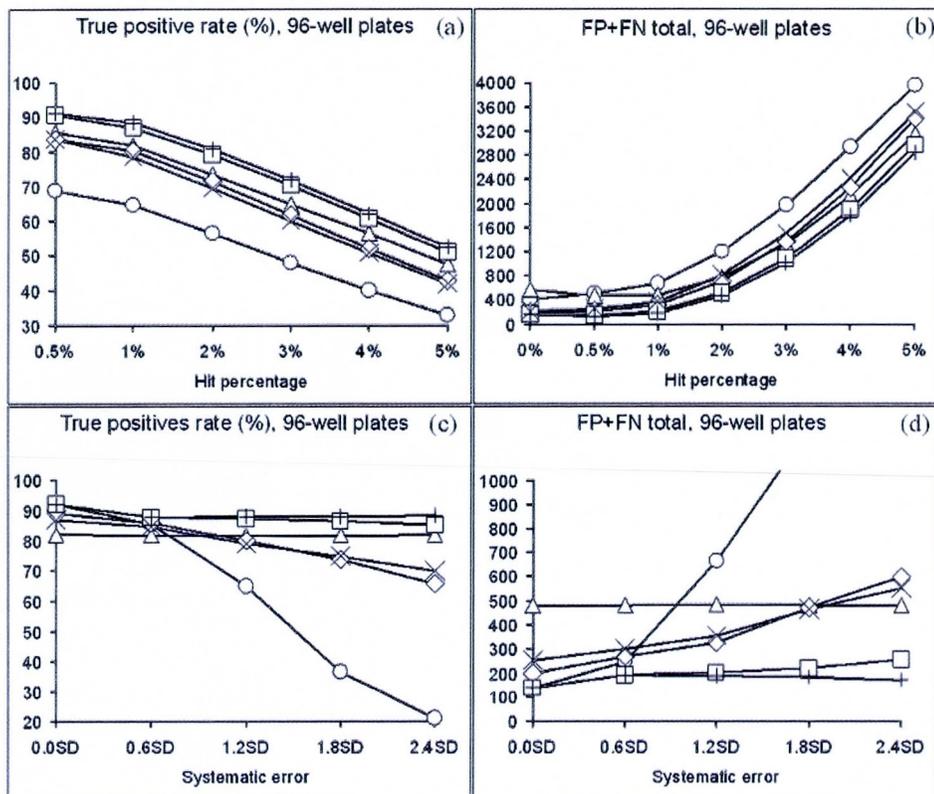
**Figure 3.2** True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 96-well plate assays estimated under the condition that at most two columns and two rows of each plate were affected by systematic error. Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction (○), B-score (Δ), MEA (□), t-test and MEA (◇), SMP (+), t-test, and SMP (×).



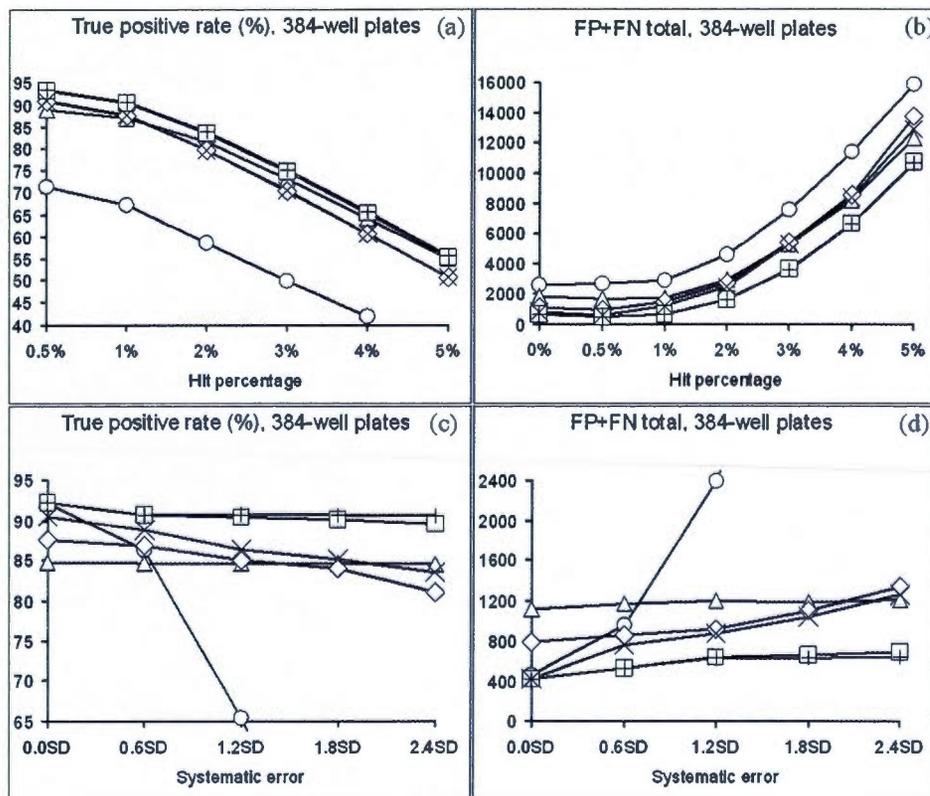
**Figure 3.3 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 384-well plate assays estimated under the condition that at most four columns and four rows of each plate were affected by systematic error. Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction (○), B-score (△), MEA (□), t-test and MEA (◇), SMP (+), t-test, and SMP (×).**



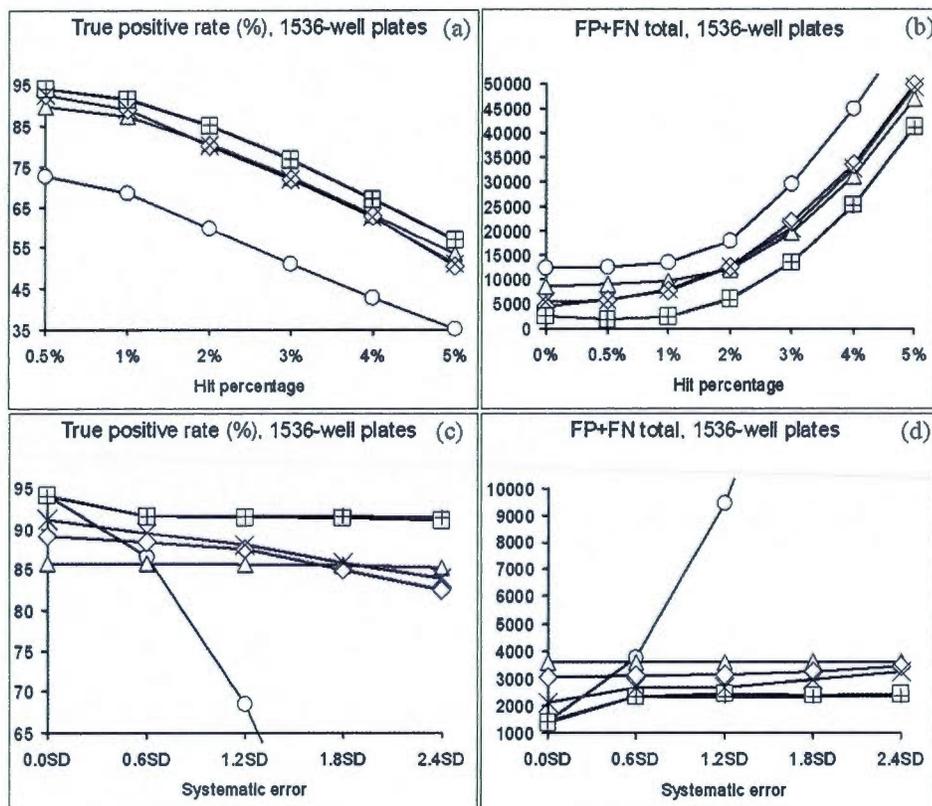
**Figure 3.4 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 1536-well plate assays estimated under the condition that at most eight columns and two eight of each plate were affected by systematic error. Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction (○), B-score (△), MEA (□), t-test and MEA (◇), SMP (+), t-test, and SMP (×).**



**Figure 3.5 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 96-well plate assays estimated under the condition that systematic error affects up to 50% of rows and columns of each plate (the exact number of affected rows and columns on each plate was determinate randomly according to the uniform distribution). Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction (○), B-score (Δ), MEA (□), t-test and MEA (◇), PMP (+), t-test, and PMP (×).**



**Figure 3.6 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 384-well plate assays estimated under the condition that systematic error affects up to 50% of rows and columns of each plate (the exact number of affected rows and columns on each plate was determinate randomly according to the uniform distribution). Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction (○), B-score (Δ), MEA (□), t-test and MEA (◇), PMP (+), t-test, and PMP (×).**



**Figure 3.7 True positive rate and total number of false positive and false negative hits (i.e., total number of false conclusions) per assay for 1536-well plate assays estimated under the condition that systematic error affects up to 50% of rows and columns of each plate (the exact number of affected rows and columns on each plate was determinate randomly according to the uniform distribution). Panels (a) and (b) present the results obtained for datasets with the fixed systematic error standard deviation of 1.2SD. Panels (c) and (d) present the results for datasets with the fixed hit percentage rate of 1%. Methods legend: No Correction (○), B-score (△), MEA (□), t-test and MEA (◇), PMP (+), t-test, and PMP (×).**

The simulation results suggest that both proposed methods outperformed the B-score and no-correction procedures when the number of the plate's rows and columns affected by systematic error was low (e.g., in case of commonly observed edge effects), regardless of plate size, hit rate and systematic error variance (see Figures 3.2 to 3.4). In the situations when the number of affected rows and columns of each plate affected by systematic bias

could attain 50% of the plate's total number of rows and columns (see Figures 3.5 to 3.7), the MEA and PMP methods generally yielded better results than B-score when the hit percentage was under 3% (see Figures 3.5 to 3.7, cases a and b) or when the level of systematic error was under  $1.8SD$  (see Figures 3.5 to 3.7, cases c and d). However, in the situations when the hit percentage or systematic error variance was high, the B-score procedure generally showed a more stable behaviour than the new methods. This was largely due to the fact that the performance of the t-test, carried out prior to MEA and PMP, decreases as the amount of data affected by systematic error grows (Chapter II or Dragiev et al. 2011). In general, the MEA method turned out to be the best performing method for correcting systematic error within 96-well plates when the systematic error variance or the hit percentage was low (see Figures 3.2 and 3.5), whereas the PMP method provided better results than MEA for the 96-well plates when the systematic error variance or the hit percentage was elevated as well as for the 384 and 1536-well plates (see Figures 3.3, 3.4, 3.6 and 3.7). It is worth noting that the B-score method was very prone to generating false positives.

#### **3.4.2 Analysis of the McMaster Test assay**

We also carried out the MEA and PMP methods to analyze the McMaster Data Mining and Docking Competition Test assay (see Elowe et al. 2005 and Figures 3.1a and b). We examined their impact on the hit identities determined during the HTS phase of the project. This dataset consisted of 625, 96-well plates (with 8 rows and 12 columns) screened in duplicate. Columns 1 and 12 of all plates contained controls and thus were not considered in our study. The assay conditions were identical for all plates. They were as follows: Each 200  $\mu$ L reaction mixture contained 40  $\mu$ M NADPH, 30  $\mu$ M DHF, 5 nM DHFR, 50 mM Tris (pH 7.5), 0.01% (w/v) Triton and 10 mM  $\beta$ -mercaptoethanol. The compounds from the screening library were added to the reaction before initiation by enzyme at a final concentration of 10  $\mu$ M. All measurements were taken at 25° C.

The threshold of  $\mu - 2.29SD$  was used to identify hits. This threshold led to the identification of 96 average hits which were reported by the competition organizers (Elowe et al. 2005). Our previous works showed that the measurements in the McMaster Test dataset were affected by systematic error (Makarenkov et al. 2007, Chapter II or Dragiev et al. 2011),

especially when some higher hit selection thresholds were used (e.g.,  $\mu-SD$  or  $\mu-2SD$ ). The hit sets provided by the six following methods were compared: uncorrected data processing, B-score, and the introduced MEA and PMP methods applied as such and in the combination with the Well Correction procedure (Makarenkov et al. 2007) allowing for removing screen-specific systematic error. Both MEA and PMP methods were carried out on a plate-by-plate basis and were preceded by the t-test, which was necessary to recover systematic error row and column locations. The t-test was performed with the  $\alpha$  parameter value set to 0.01 (see Supplementary Materials). As the McMaster Test dataset contained replicates, the hit selection procedure was adjusted to search for average hits (i.e., the average of the two measurements of every compound was calculated and the obtained result was supplied to the hit selection procedure). The totals of hits retraced by the six considered methods are presented in Tables 3.1 and 3.2-3.13 (the detailed results).

Data correction method	Number of hits
No Correction	96
B-score	186
Matrix Error Amendment (MEA)	100
Partial Mean Polish (PMP)	115
Well Correction + Matrix Error Amendment (MEA)	109
Well Correction + Partial Mean Polish (PMP)	109

**Table 3.1** Number of hits selected by the six data correction methods for the McMaster Test dataset. The hit selection threshold of  $\mu-2.29SD$  was used.

Both proposed methods identified more potential hits (100 for MEA and 115 for PMP) than the organizers of the McMaster competition (i.e., 96 hits for the uncorrected dataset), while rejecting a few of the original hits as false positives. The MEA method found 8 extra hits, while rejecting 4 of the original hits as false positives. The PMP method extended the set of original hits with 24 new hits, while rejecting only 5 of them. In contrast, the B-score method rejected 28 original hits, and provided 118 new potential hits (according to our simulation results, many of those new hits can be in fact false positives). The total overlap of all the six considered methods consisted in 55 consensus average hits that could be recommended for further testing including the structure-activity relationships (SAR) analysis

Row\Column	1	2	3	4	5	6	7	8	9	10
1	19	39	32	59	50	41	26	14	21	24
2	42	63	81	82	69	63	34	30	17	16
3	27	30	22	27	25	14	18	25	29	7
4	31	45	41	21	22	13	21	15	19	12
5	42	40	62	30	18	18	16	16	26	5
6	51	86	97	52	47	29	38	22	21	5
7	40	76	65	65	73	45	48	16	20	9
8	107	85	101	71	56	48	23	23	24	13

**Table 3.2 Hit distribution of the raw McMaster dataset computed for the  $\mu$ -SD threshold (mean value of hits per well is 37.69 and standard deviation is 24.27). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 1234.23, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended).**

Row\Column	1	2	3	4	5	6	7	8	9	10
1	1	2	1	5	1	0	0	0	2	1
2	1	1	3	2	1	0	1	1	5	1
3	0	0	0	0	3	0	0	0	3	1
4	2	1	1	1	1	1	1	0	2	1
5	2	0	0	1	2	1	2	0	0	0
6	0	1	4	2	1	0	3	0	0	1
7	2	3	1	2	3	3	2	1	1	0
8	3	1	2	0	0	0	1	1	3	0

**Table 3.3 Hit distribution of the raw McMaster dataset computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.20 and standard deviation is 1.19). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 94.0, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional).**

and various clinical trials. As shows the example of the consensus hits set of the McMaster Test assay (see Elowe et al. 2005 or Table 9SM in Makarenkov et al. 2007), consensus hits can also contain an important percentage of false negatives and false positives. The consensus hits list of this assay, which included 42 hit compounds in total, comprised only 14 of 26 hit compounds confirmed by the SAR analysis conducted by the McMaster competition

organizers (i.e., 12 of 26 confirmed hits false negatives and 28 of 42 consensus hits were false positives). Thus, SAR investigations should be always conducted in conjunction with data correction and hit selection techniques in order to confirm the selected hits.

Row\Column	1	2	3	4	5	6	7	8	9	10
1	21	41	33	61	57	61	50	34	43	60
2	39	51	47	64	58	73	43	47	31	28
3	31	35	31	33	35	32	28	38	58	32
4	39	43	39	33	31	21	25	30	34	47
5	59	37	54	40	25	26	29	26	50	24
6	39	55	51	34	45	27	27	24	29	10
7	30	39	36	42	67	44	45	20	21	15
8	97	48	61	46	51	44	27	25	29	23

**Table 3.4 Hit distribution of the McMaster dataset after applying B-score normalization and computed for the  $\mu$ -SD threshold (mean value of hits per well is 39.48 and standard deviation is 14.67). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 430.96, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended).**

Row\Column	1	2	3	4	5	6	7	8	9	10
1	1	2	1	10	5	1	1	0	2	3
2	3	2	4	4	2	3	1	1	4	1
3	1	0	1	1	3	1	1	0	7	2
4	5	4	2	2	2	3	1	1	1	4
5	4	1	2	2	3	2	4	1	0	2
6	3	1	5	1	3	0	5	1	1	1
7	2	3	0	3	5	5	2	1	2	2
8	4	2	1	1	0	3	2	3	9	1

**Table 3.5 Hit distribution of the McMaster dataset after applying B-score normalization and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 2.33 and standard deviation is 1.89). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 121.10, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended).**

Row\Column	1	2	3	4	5	6	7	8	9	10
1	23	42	30	61	51	45	30	14	24	26
2	42	61	80	82	67	66	32	32	16	16
3	28	29	24	28	27	16	20	29	29	7
4	31	45	41	21	24	13	21	15	20	13
5	44	37	60	31	20	20	17	17	26	5
6	52	87	93	53	49	29	39	22	20	6
7	39	76	64	68	74	50	52	16	21	11
8	108	87	99	70	58	51	24	23	24	15

**Table 3.6** Hit distribution of the McMaster dataset after applying Matrix Error Amendment (MEA) method and computed for the  $\mu$ -SD threshold (mean value of hits per well is 38.48 and standard deviation is 23.96). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 1178.53, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended).

Row\Column	1	2	3	4	5	6	7	8	9	10
1	1	2	1	5	1	0	0	0	1	1
2	2	1	2	2	1	0	1	1	5	1
3	0	0	0	0	3	0	0	0	3	1
4	2	1	1	2	1	1	1	0	2	1
5	2	0	0	1	2	2	3	0	0	0
6	0	1	4	3	1	0	3	0	0	1
7	3	3	1	2	3	3	2	1	1	0
8	4	1	2	0	0	0	1	1	2	0

**Table 3.7** Hit distribution of the McMaster dataset after applying Matrix Error Amendment (MEA) method and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.25 and standard deviation is 1.24). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 96.80, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional).

Row\Column	1	2	3	4	5	6	7	8	9	10
1	27	49	33	64	49	47	33	16	26	32
2	41	60	81	78	67	67	34	29	18	22
3	30	28	24	29	27	16	24	30	36	17
4	32	41	42	21	25	15	21	13	20	22
5	45	41	61	32	20	21	20	19	29	14
6	51	85	93	50	49	34	41	25	25	10
7	42	72	65	69	72	52	47	15	25	19
8	104	85	99	70	55	55	22	26	24	23

**Table 3.8 Hit distribution of the McMaster dataset after applying Partial Mean Polish (PMP) method and computed for the  $\mu$ -SD threshold (mean value of hits per well is 39.90 and standard deviation is 22.41). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 994.27, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended).**

Row\Column	1	2	3	4	5	6	7	8	9	10
1	1	2	1	5	3	0	0	1	2	1
2	2	1	3	2	1	0	1	1	5	1
3	0	1	0	0	2	0	0	0	3	0
4	2	1	1	1	1	1	1	0	2	1
5	1	0	0	1	3	3	2	0	0	0
6	1	1	4	2	2	0	3	0	0	1
7	2	4	2	2	4	4	3	1	2	2
8	4	2	2	0	1	0	1	1	4	0

**Table 3.9 Hit distribution of the McMaster dataset after applying Partial Mean Polish (PMP) method and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.44 and standard deviation is 1.32). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 95.78, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional).**

It is worth also noting that MEA and PMP agreed on most of the hits they selected (i.e., 92 of the hits identified by MEA were also detected by PMP). Furthermore, after the application of Well Correction, the MEA and PMP methods provided an identical set of 109 hits. Figure 3.8

and Tables 3.2 to 3.13 present the hit distribution surfaces (i.e., hit totals obtained for each well location and computed over all plates of the given assay) of the Master Test assay obtained for the hit selection thresholds  $\mu-SD$  and  $\mu-2.29SD$ .

Row\Column	1	2	3	4	5	6	7	8	9	10
1	31	22	45	50	22	31	42	42	41	27
2	22	44	44	52	33	45	37	44	31	31
3	44	40	29	34	35	51	26	39	43	14
4	25	40	42	38	22	35	39	26	30	35
5	20	38	44	37	32	34	35	32	45	17
6	46	54	55	40	46	41	40	30	39	24
7	27	33	34	50	19	34	36	22	29	32
8	40	41	55	31	41	47	21	23	31	20

**Table 3.10 Hit distribution of the McMaster dataset after applying Well Correction followed by Matrix Error Amendment (MEA) method and computed for the  $\mu-SD$  threshold (mean value of hits per well is 35.48 and standard deviation is 9.55). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 203.18, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended).**

Row\Column	1	2	3	4	5	6	7	8	9	10
1	1	0	1	5	0	0	1	1	2	0
2	0	1	2	2	0	0	1	2	6	1
3	2	0	0	0	3	1	0	0	7	0
4	0	2	1	2	1	2	1	0	2	2
5	0	0	1	1	2	3	3	2	1	0
6	1	1	4	3	2	0	3	1	3	1
7	0	3	0	2	2	1	2	1	1	2
8	2	0	2	0	0	2	1	1	2	1

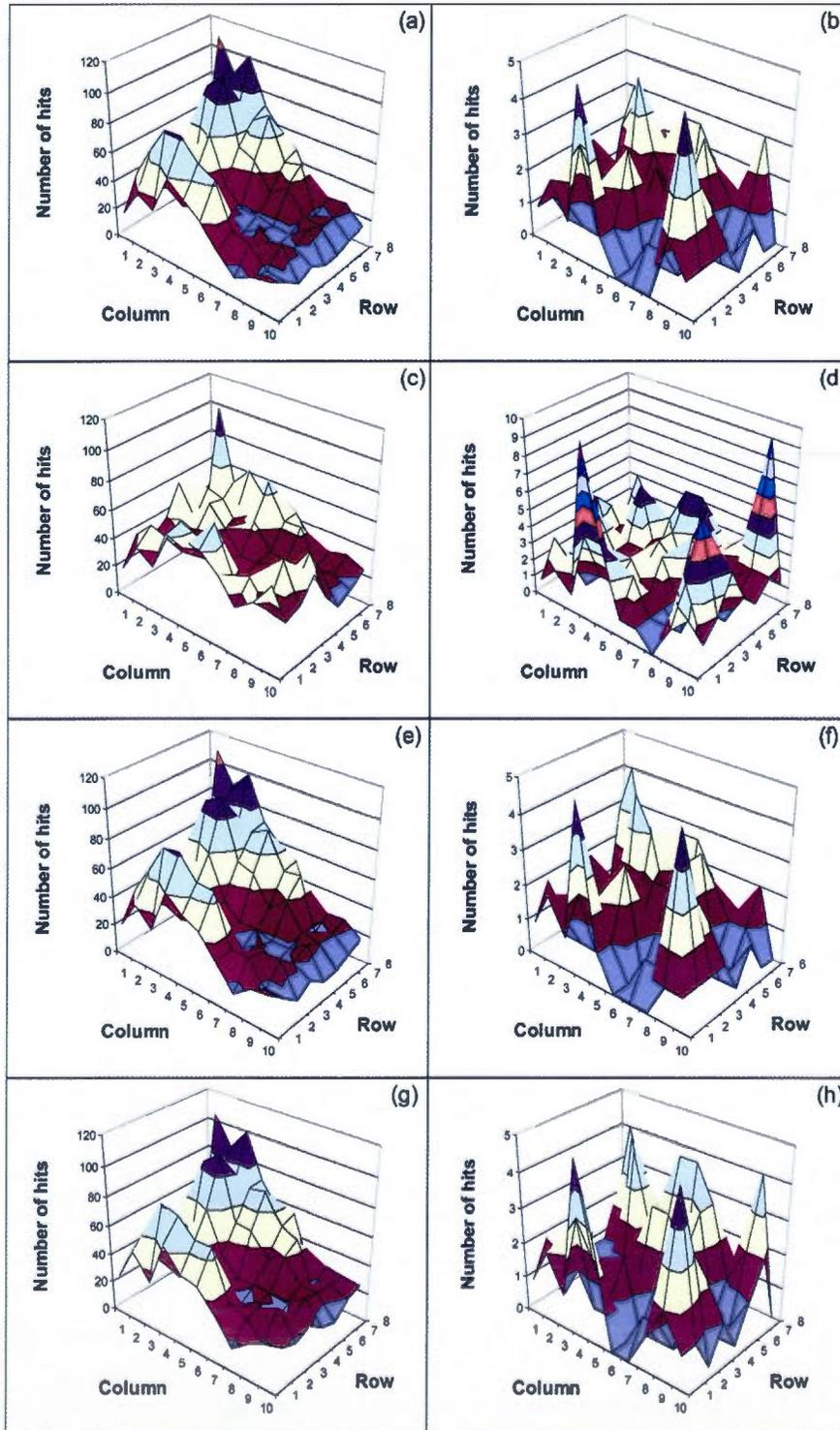
**Table 3.11 Hit distribution of the McMaster dataset after applying Well Correction followed by Matrix Error Amendment (MEA) method and computed for the  $\mu-2.29SD$  threshold (mean value of hits per well is 1.36 and standard deviation is 1.37). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 108.98, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional).**

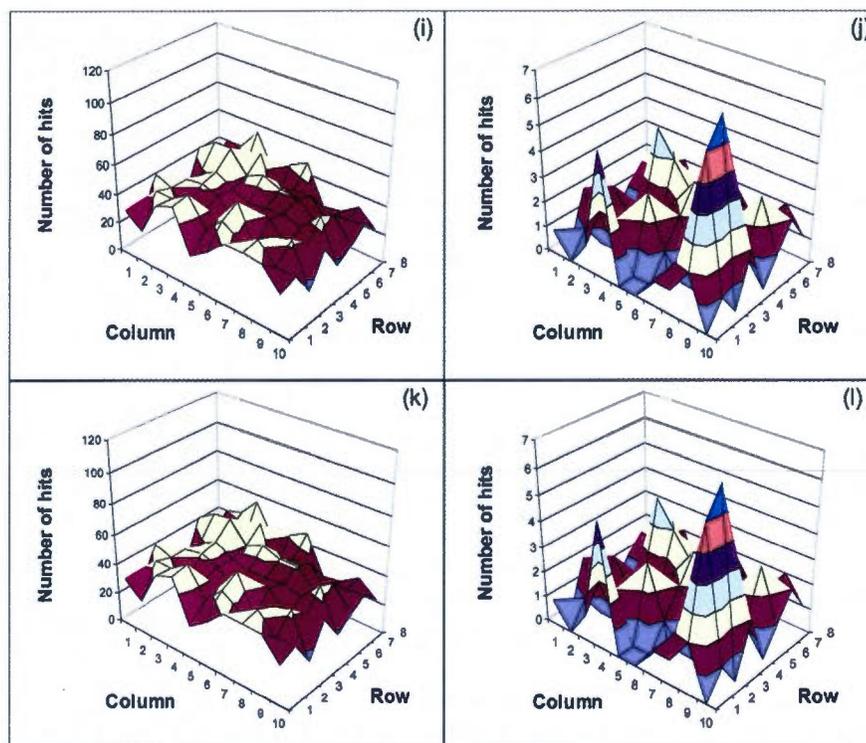
Row\Column	1	2	3	4	5	6	7	8	9	10
1	32	22	45	50	22	31	42	42	41	27
2	22	44	44	52	33	45	37	44	31	32
3	44	40	29	34	35	51	26	39	43	14
4	25	40	42	38	23	36	39	26	30	35
5	20	38	44	37	34	34	35	32	45	18
6	46	53	55	40	46	41	41	30	40	25
7	27	34	34	50	19	34	37	22	29	31
8	40	43	55	31	41	47	21	23	31	21

**Table 3.12** Hit distribution of the McMaster dataset after applying Well Correction followed by Partial Mean Polish (PMP) method and computed for the  $\mu$ -SD threshold (mean value of hits per well is 35.64 and standard deviation is 9.47). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 198.68, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is rejected and data correction is recommended).

Row\Column	1	2	3	4	5	6	7	8	9	10
1	1	0	1	5	0	0	1	1	2	0
2	0	1	2	2	0	0	1	2	6	1
3	2	0	0	0	3	1	0	0	7	0
4	0	2	1	2	1	2	1	0	2	2
5	0	0	1	1	2	3	3	2	1	0
6	1	1	4	3	2	0	3	1	3	1
7	0	3	0	2	2	1	2	1	1	2
8	2	0	2	0	0	2	1	1	2	1

**Table 3.13** Hit distribution of the McMaster dataset after applying Well Correction followed by Partial Mean Polish (PMP) method and computed for the  $\mu$ -2.29SD threshold (mean value of hits per well is 1.36 and standard deviation is 1.37). The  $\chi^2$  goodness-of-fit test provided the following results ( $\chi^2$  value = 108.98, critical value 111.14 for  $\alpha = 0.01$ ;  $H_0$  is not rejected and data correction is optional).





**Figure 3.8** Hit distribution surfaces of the McMaster *Test dataset* for the hit selection thresholds  $\mu - SD$  (cases a, c, e, g, i and k) and  $\mu - 2.29SD$  (cases b, d, f, h, j and l) obtained for: the raw (i.e., uncorrected) data (a, b), and the data corrected by B-score (c, d), MEA (e, f), SMP (g, h), Well Correction + MEA (i, j), and Well Correction + SMP (k, l).

The consecutive application of two data correction methods, Well Correction and MEA (Figures 3.8i and j) or Well Correction and PMP (Figures 3.8k and l), allowed us to eliminate screen-specific systematic error, first, and plate-specific systematic error, second (see Tables 3.11 and 3.13). For instance, the MEA and PMP hit distribution surfaces provide better fits to the corresponding plain surfaces (which represent a perfect uniform distribution of the assay hits across all well locations) when Well Correction is applied beforehand (Figures 3.8i and k). After the application of Well Correction, the hit distribution surface  $\chi^2$  goodness-of-fit statistic for the hit selection thresholds  $\mu - SD$  decreased from 1178.53

(Figures 3.8e and Table 3.6) to 203.18 for MEA (Figures 3.8i and Table 3.10) and from 994.27 (Figures 3.8g and Table 3.8) to 198.68 for PMP (Figures 3.8k and Table 3.12).

### 3.5 Conclusion

We described two new methods, called Matrix Error Amendment (MEA) and Partial Mean Polish (PMP), allowing for elimination of plate-specific systematic error from experimental HTS data. Both methods rely on the prior information concerning the location of the rows and columns of the given plate affected by systematic bias. Such information can be obtained by using the methodology described in Chapter II or Dragiev et al. 2011.

We conducted a simulation study with different HTS plate sizes, hit percentages and systematic error magnitudes. In this study, the MEA and PMP methods were compared to the B-score (Brideau et al. 2003) and no-correction strategies. Both new methods always outperformed the B-score and no-correction procedures when the number of the plate's rows and columns affected by systematic error was low (Figures 3.2 to 3.4). In the simulations where the number of rows and columns affected by systematic error could reach 50% of the plate's total number of rows and columns (Figures 3.5 to 3.7), the MEA and PMP methods generally yielded better results than B-score when the hit percentage was under 3% (in a typical HTS campaign the hit percentage is usually under 1%) or when the level of systematic error was under  $1.8SD$ . The B-score method showed a more stable behaviour than MEA and PMP only when the number of rows and columns affected by systematic error, hit percentage and systematic error variance were high (mainly due to a mediocre performance of the t-test in this case). MEA was generally the best method for correcting systematic error within 96-well plates, while PMP performed better for 384 and 1536-well plates.

The analysis of the McMaster Data Mining and Docking Competition Test assay (Elowe et al. 2005) showed that the new methods can be also applied in the combination with the Well Correction technique (Makarenkov et al. 2007) aiming to remove screen-specific systematic error. Hence, a general data correction phase in HTS, permitting for the elimination of both screen- and plate-specific systematic biases, can be conducted in the following way:

1. **Normalize** the raw measurements using Percent of control, Normalized percent inhibition or Z-score transformation. This normalization step can be carried out either on a plate-by-plate basis or for all assay measurements together (i.e., when all plates have been processed under the same experimental conditions);
2. **Perform** the t-test or  $\chi^2$  goodness-of-fit test on the hit distribution surface for the selected hit selection threshold;
3. **If** systematic error is detected **then carry out** the Well Correction method;
4. **Perform** the t-test or  $\chi^2$  goodness-of-fit test on each individual plate of the assay to identify its rows and columns affected by systematic error as well as the error locations;
5. **For** all plates where systematic error is detected **correct** the plate measurements by carrying out the PMP or MEA method (or, alternatively, the B-score procedure).

In this study we addressed the issue of the commonly considered additive systematic artifact that can be described using equation (3.17). It is worth noting that the multiplicative type of systematic bias affecting well  $(i, j)$  of plate  $p$  and defined by equation (3.18):

$$x'_{ijp} = x_{ijp} \times e_{\eta p} \times e_{c_j p} + rand_{ijp}, \quad (3.19)$$

can be also treated using the proposed methods. While the MEA method should undergo substantial changes in order to treat multiplicative type of systematic error because the linear equations systems (3.4 to 3.7) and (3.8) will be transformed into the corresponding nonlinear equations systems, the PMP method can be easily adapted for the identification and correction of multiplicative bias by adding the following equations:  $\hat{e}_{\eta} = \mu_{\eta}/\mu$  and  $\hat{e}_{c_j} = \mu_{c_j}/\mu$  to step 2, and then  $x_{\eta j} = x_{\eta j}/\hat{e}_{\eta}$  and  $x_{ic_j} = x_{ic_j}/\hat{e}_{c_j}$  to step 3, of the method instead of the corresponding equations containing the subtraction sign.

A version of the PMP method, in which a median is used instead of the mean, could be viewed as a direct extension of the well-known median polish (MP) algorithm (Tukey 1977), applicable in the situations when the exact error location is known (the traditional median polish assumes that systematic error is present in all rows and columns of the given matrix). Another advantage of the PMP method over MP and its B-score analog is that our method does not reduce the original data to residuals, keeping the corrected data on the same scale with the original ones, and not modifying the unbiased data at all. Moreover, both of the proposed methods could be interesting in general, from the statistical point of view, and applied as data correction methods in any other field.

A new program implementing the two data correction methods described in this article, and including also the Well Correction, B-score and Z-score procedures, is freely available at the following URL: [http://www.info2.uqam.ca/~makarenkov\\_v/HTS\\_Helper](http://www.info2.uqam.ca/~makarenkov_v/HTS_Helper).

### 3.6 Supplementary Materials

#### 3.6.1 *t*-test applied in the HTS context

This test is based on the classical two-sample Student's *t*-test in the case of samples with different sizes (for more details, see Chapter II or Dragiev et al. 2011). The test was carried out separately for each row and column of each assay plate. We divided the data into two independent subsets. The first subset contained the measurements of the tested row or column, while the second consisted of all remaining plate measurements. In this test, the null hypothesis  $H_0$  was that the selected row or column does not contain systematic error. If systematic error is absent, then the mean of the given row or column is expected to be close to the mean of the rest of the data in the given plate or hit distribution surface. For the two samples in hand:  $S_1$  with  $N_1$  elements and  $S_2$  with  $N_2$  elements, we first calculated the two sample variances  $s_1^2$  and  $s_2^2$ , and then their weighted average (Equation 3.20):

$$s_p^2 = \frac{(N_1 - 1) \times s_1^2 + (N_2 - 1) \times s_2^2}{N_1 + N_2 - 2}. \quad (3.20)$$

The value of the *t*-statistic can then be obtained as follows (Equation 3.21):

$$t = \frac{\mu_1 - \mu_2}{\sqrt{s_p^2 \left( \frac{1}{N_1} + \frac{1}{N_2} \right)}}, \quad (3.21)$$

where  $\mu_1$  is the mean of the sample  $S_1$  and  $\mu_2$  is the mean of the sample  $S_2$ . The calculated *t*-statistic was then compared to the corresponding critical value for the chosen statistical significance level  $\alpha$ . The  $\alpha$  parameter equal to 0.01 was used in our simulations.

### 3.6.2 $\chi^2$ goodness-of-fit test applied in the HTS context

The  $\chi^2$  goodness-of-fit test in the HTS context was recommended in Makarenkov et al. 2007 and Chapter II or Dragiev et al. 2011 in order to identify systematic error in HTS hit distribution surfaces. The null hypothesis,  $H_0$ , in this test is that no systematic error is present in the data. If  $H_0$  is true, then the hits are supposed to be evenly distributed across the well locations and the observed number of hits  $x_{ij}$  in row  $i$  and column  $j$  of the hit distribution surface is not significantly different from the expected value calculated as the total number of hits in the assay divided by the number of wells per plate. The rejection region of  $H_0$  is  $P(\chi^2 > C_\alpha) > \alpha$ , where  $C_\alpha$  is the  $\chi^2$  distribution critical value corresponding to the selected  $\alpha$  parameter (in this study, we used  $\alpha = 0.01$ ) and to the number of degrees of freedom of the model.

The test statistic  $\chi^2$  over all well locations of the given hit distribution surface can be computed as follows (Equation 3.22):

$$\chi^2 = \sum_{i=1}^m \sum_{j=1}^n \frac{(x_{ij} - E)^2}{E}, \quad (3.22)$$

where  $m$  is the number of rows per plate,  $n$  is the number of columns per plate,  $\square E$  is the total number of hits on the entire hit distribution surface (i.e., whole assay) divided by the number of wells per plate ( $m \times n$ ). The number of degrees of freedom here equals to  $m \times (n-1)$ . For the McMaster dataset considered in this study  $m = 8$  and  $n = 10$ .

## CHAPTER IV

### STATISTICAL METHODS FOR THE ANALYSIS OF EXPERIMENTAL HIGH-THROUGHPUT SCREENING DATA

This chapter is a reproduction of the following article:

Dragiev P., Makarenkov V., Mballo C. and Nadon R. (2012) Statistical Methods for the Analysis of Experimental High-Throughput Screening Data. *Advances in Data Analysis and Classification*, Springer, submitted.

High-throughput screening (HTS) is a large scale process intended to evaluate a vast number of compounds in order to determine the molecules with the best activity levels. Preparing and testing a large number of compounds requires various technical resources and involves very high costs. Continuous use of HTS generates large databases containing millions of tested compounds. Often the same compounds are tested more than once in different concentrations.

Chapter IV is devoted to techniques intended to reduce the overall cost of HTS by employing statistical methods and simulations to decrease the number of compounds that require experimental HTS testing. We first investigate the possibility of employing information collected during the previous tests for predicting the outcomes of the current ones. We consider the case that all compounds are characterized by a set of chemical descriptors which we can use to establish similarities and differences between the current compound and the compounds that have been already tested. We also explore the applicability of two machine learning methods, decision trees and neural networks, for

predicting the hit/non-hit outcomes of HTS experiments based on the compounds similarity with the compounds that have been already tested. The ability to predict correct outcomes of screening campaigns, i.e., performing *in silico* HTS instead of experimental HTS, can contribute to the reduction of the overall cost of HTS.

Often the researchers, not confident in the quality of the obtained results, repeat the same tests one or more times multiplying, in this way, the cost of the experiments. However, the use of replicates increases the quality of HTS results (Malo et al. 2006). Currently, there is no formal model that can be applied to decide how many replicates should be tested in order to ensure a selected false positive or false negative rate. The decisions are taken by the researchers subjectively. In Chapter IV, we propose a new probability-based model, based on the analysis of replicated measurements, which allows us to estimate for every compound of the assays its probability to be a hit. Using such a model, we can define a new probability-based hit selection procedure. Furthermore, we developed a methodology allowing one to estimate the effect that one extra experimental replicate would have on the quality of HTS results, providing objective information to the researchers for deciding whether additional assay screens will significantly decrease the compounds false positive and false negative rates. The systematic error detection and elimination methods described in Chapters II and III will be applied in Chapter IV to refine the obtained experimental results.

#### 4.1 Abstract

High-throughput screening (HTS) is a modern technology actively used to identify pharmacologically active compounds. HTS is a highly automated early-stage process that allows thousands of chemical compounds to be tested in a single study. It classifies all promising compounds as hits that need to be further investigated. As the experimental HTS is a very costly process, the development of accurate statistical models for virtual prediction of HTS measurements could lead to an important cost decline. In this article we focus on finding the relationships between experimental HTS measurements and a group of descriptors characterizing chemical compounds. Polynomial redundancy analysis (Polynomial RDA) along with neural networks and decision trees methods were applied to discover these relationships. We also describe a new hit selection method based on the estimation of the hit-outcome probabilities of the given chemical compounds.

#### 4.2 Introduction

HTS is a modern technology currently available in many pharmacological laboratories worldwide. It serves for automated early identification of active chemical compounds, called *hits*. The hits discovered during the primary screening are used as starting points for further optimization and development. HTS is also employed for the determination of activity, toxicological and physiological properties of chemical compounds as well as for the verification of structure-activity hypotheses (Heyse 2002). The advances in robotic methods, parallel processing and miniaturization of the assays have highly increased the screening throughput (Malo et al. 2006). A typical HTS center in the pharmaceutical industry can generate millions of data points per year (Heuer et al. 2005). Along with the augmented throughput, the research costs have also dramatically increased. This work is aimed to present a methodology for reducing the cost of HTS by using computer aided statistical methods to accurately predict the compound measurement values. Such a methodology will allow one to decrease the number of experimentally screened compounds and thus lower the total cost of an HTS campaign. Our approach will use already collected experimental measurements and the similarity between compounds as a base for prediction. We will first attempt to establish the relationships between the obtained experimental measurements of an HTS assay and the values of 10 physicochemical and structural molecular descriptors widely

employed by chemists. The polynomial RDA analysis (Makarenkov and Legendre 2002) will be carried out in order to discover those relationships. Second, two machine learning methods, decision trees (Breiman et al. 1984) and neural networks (Haykin 1999), will be applied and compared to each other. Finally, we will describe a new hit selection method allowing one to estimate the probability of each considered compound to be a hit. This probability will be assessed based on the available replicated measurements of the screened chemical compounds. The obtained probabilities can be used as a benchmark for deciding which compounds should be tested in further experimental HTS trials.

#### **4.3 Analysis of the hit and no hit data in the McMaster *Test dataset***

In this study, we consider an experimental HTS dataset provided by the HTS Laboratory of McMaster University. We analyze real screening data proposed originally as *Test dataset* and used as a benchmark for the McMaster Data Mining and Docking Competition (Elowe et al. 2005). This dataset consists of screening data of compounds that inhibit the *Escherichia coli* dihydrofolate reductase. Each of 50,000 different chemical compounds located in plates were screened in duplicate; two copies of each of the 625 plates were run through the screening equipment; 1250 plates in total, with wells arranged in 8 rows and 12 columns, were screened; columns 1 and 12 of each plate were used as high and low controls (i.e., compounds with the well-known properties whose values are used for the data normalization) and, therefore, not considered in this study. The exact experimental conditions are reported in the article of Elowe et al. 2005.

The competition organizers defined the primary hits as compounds that reduced the DHFR activity to 75% of the average residual activity of the high (uninhibited) controls. Two lists of hits were published (for more details, see: [http://hts.mcmaster.ca/Downloads/82BFEB4-F2A4-4934-B6A8-804CAD8E25A0\\_files/experimental\\_actives.pdf](http://hts.mcmaster.ca/Downloads/82BFEB4-F2A4-4934-B6A8-804CAD8E25A0_files/experimental_actives.pdf)). The first list, called a *consensus hit list*, contained all compounds that were classified as hits in both of their replicated measurements (i.e., both obtained measurement values were lower or equal to 75% of the reference controls). Only 42 of all the 50,000 tested compounds were consensus hits. The second list, called an *average hits list*, contained 96 compounds classified as hits when the average value of the two HTS measurements was lower or equal to 75% of the reference controls. Obviously, all consensus

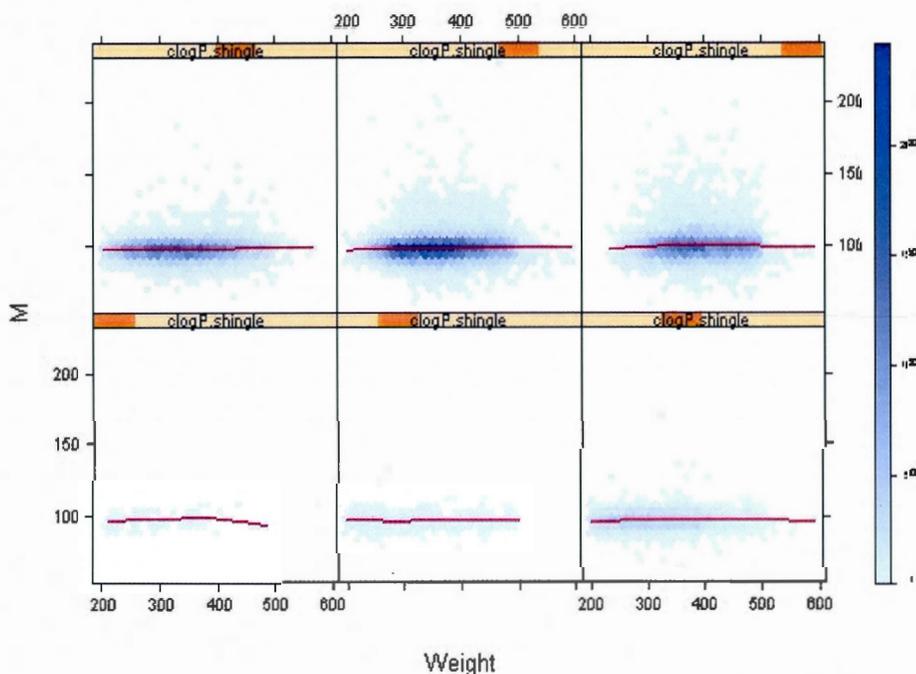
hits were also average hits. A secondary screening of the 96 average hits was also performed in order to determine their activity in different concentrations. As result of the secondary screening, 26 of the average hits were identified as D-R hits (i.e., hits having *well-behaved dose-response curves*, see Elowe et al. 2005).

Data type / Average values of chemical descriptors	Consensus hits (42 samples)	D-R hits (26 samples)	Average hits (96 samples)	All data (50,000 samples)
<i>Measurement</i>	68.13	69.73	70.38	98.88
<b><i>ClogP</i></b>	<b>4.61</b>	<b>4.56</b>	<b>4.35</b>	<b>3.68</b>
<i>Hdon</i>	1.19	1.04	1.18	1.06
<i>Hacc</i>	3.48	3.62	3.28	3.50
<i>RB</i>	4.17	4.27	3.94	3.81
<i>tPSA</i>	66.58	63.09	64.60	64.58
<b><i>MW</i></b>	<b>392.60</b>	<b>381.35</b>	<b>379.21</b>	<b>358.69</b>
<b><i>SlogP</i></b>	<b>4.72</b>	<b>4.61</b>	<b>4.50</b>	<b>3.76</b>
<b><i>nSSSR</i></b>	<b>3.29</b>	<b>3.31</b>	<b>3.13</b>	<b>2.82</b>
<b><i>logP_5</i></b>	<b>0.31</b>	<b>0.38</b>	<b>0.36</b>	<b>0.14</b>
<b><i>SumFlag</i></b>	<b>0.12</b>	<b>0.00</b>	<b>0.09</b>	<b>0.08</b>

**Table 4.1 Average values of the measurements and 10 chemical descriptors for four types of samples (consensus hits – 42 samples, average hits – 96 samples, hits having well-behaved dose-response curves – 26 samples and the whole dataset – 50,000 samples) from the McMaster University Test dataset. In bold, the values of chemical descriptors showing important variation.**

In this study we used a set of 10 chemical descriptors considered as primary by many researchers: molecular weight (*MW*), number of H-accepting (*Hacc*) and H-donating (*Hdon*) atoms, number of rotatable bonds (*RB*), topologic polar surface area (*tPSA*), three flavors of *log* of the octanol/water partition coefficient (*ClogP*, *SLogP* and *LogP\_5*), number of rings (*nSSSR*) and the variable representing the basic properties violation score of a molecule (*SumFlag*) – the bigger the value of *SumFlag*, the less “drugable” is the molecule (see Sirois et al. 2005 for more details). In order to evaluate the usability of the selected descriptors for distinguishing the hits from no hits, we calculated and compared the mean values of four different groups of compounds: consensus hits, average hits, average hits with well-behaved dose response curves and whole dataset. The resultant values are reported in Table 4.1. This

table shows that there are important variations in the mean measurement values of the hit and no hit compounds for the six chemical descriptors put in bold in Table 4.1.



**Figure 4.1** The relationship between the HTS measurement values (varying from 0 to 200) and the molecular weight (varying from 200 to 600) depicted at 6 different levels of the *ClogP* descriptor. There is virtually no difference in the six presented patterns. That suggests that *ClogP* does not have any influence over the way molecular weight is related to the HTS measurements.

We also looked for the relationships between the values of experimental HTS measurements and the associated values of chemical descriptors. Our analyses showed no evidence of simple (e.g. linear) relationship between the predictors and the compound measurements. Figure 4.1 presents one example of potential interactions. It depicts the relationship between the experimental measurements (denoted by *M*) and *MW* (denoted by *Weight*) at six different levels of the *ClogP* descriptor. The relationship between the experimental measurements and *MW* does not differ across levels of *ClogP* (i.e., there is no *MW* by *ClogP* interaction in predicting the measurements). The other interaction graphs also

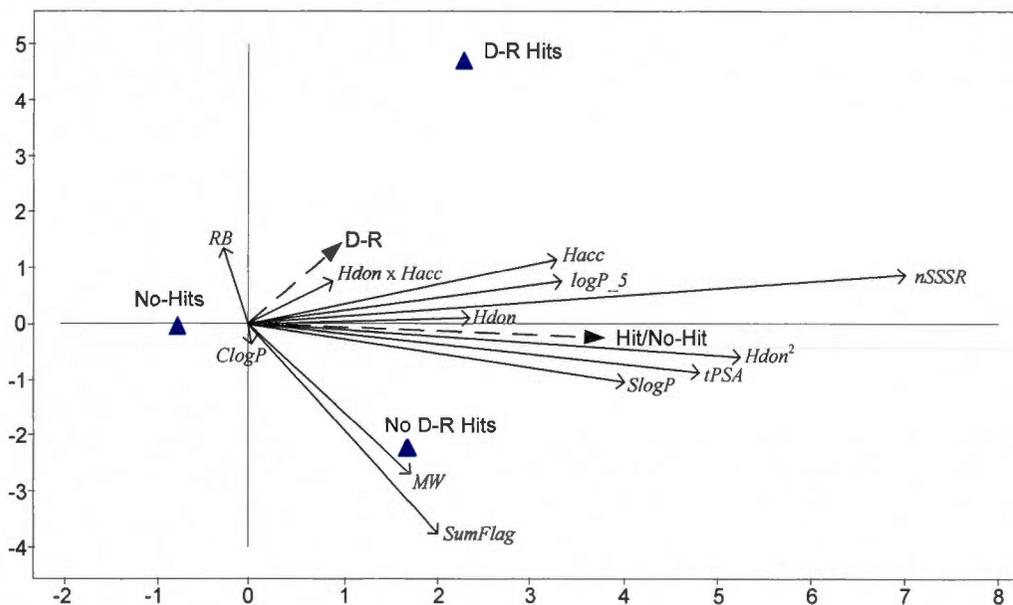
showed the absence of simple linear relationships between the assay measurements and the associated values of chemical descriptors.

#### **4.4 Polynomial RDA to establish relationships between hit/no hit outcomes and values of chemical descriptors**

The traditional redundancy analysis (RDA) is a direct extension of multiple linear regression to the modeling of multivariate response data (Legendre and Legendre 1998). Makarenkov and Legendre 2002 proposed a method based on polynomial regression, called a *polynomial RDA*, that allows one to do away with the assumption of linearity in modeling the relationships between the explanatory (chemical descriptors in our case) and response (hit/no hit outcomes in our case) variables. To carry out polynomial RDA, we considered 42 existing consensus hits and 100 no hit samples selected randomly from the set of 50,000 available experimental measurements from the McMaster *Test dataset*. The values of the 10 considered chemical descriptors formed the matrix of explanatory variables **X** and the two binary variables, consisting of the hit/no hit and positive/negative dose-response behavior outcomes formed the matrix of response variables **Y**. The biplot ordination diagram in Figure 4.2 helps interpret the ordination of samples in terms of **Y** and **X**. Here we used the correlation biplot to represent the relationships between the samples and the variables in **X** and **Y**. In such a biplot the angles between the variables from the sets **X** and **Y** reflect their correlations; projecting a sample at right angle on a response variable *y* approximates the value of the sample along this variable; projecting a sample at right angle on an explanatory variable *x* approximates the value of the sample along this variable.

The  $R^2$  coefficient for the polynomial regression of **X** on **Y** was equal to 0.55. The percentage of variance of **Y** accounted for was 62.69% for the polynomial regression and only 45.51% for the multiple linear regression. The first canonical axis accounted for 56.24% of the total variance explained by the polynomial RDA, while the second axis accounted for 6.45% only. We also conducted the permutation tests for both polynomial and linear regressions as well as for the differences between them. The difference test was carried out to estimate the possibility of overfitting by the polynomial regression (see Makarenkov and Legendre 2002). The tests carried out with 999 permutations showed that both models, as well as the differences between them, were highly significant (*p*-values of 0.001 for each of

the three tests were obtained). Given the high significance of the difference test, the use of the polynomial regression and the polynomial RDA can be recommended for this dataset.



**Figure 4.2 Polynomial RDA correlation biplot for the McMaster Test dataset.** Triangles represent the three types of samples (*D-R Hits* - the consensus hits showing good dose-response behavior; *No D-R Hits* - the consensus hits not showing good dose-response behavior; *No-Hits* - samples that are not hits). Dashed arrows represent two binary response variables *Hit/No-Hit* and *D-R*. Solid arrows represent the chemical descriptors (10 descriptors from Table 4.1 plus 2 extra descriptors  $Hdon^2$  and  $Hdon \times Hacc$  that show the biggest correlations with *Hit/No-Hit* and *D-R* variables, respectively). The lengths of the chemical descriptor arrows were multiplied by 10; this does not change the interpretation of the diagram.

The following trends can be also noticed while observing the polynomial RDA biplot in Figure 4.2: the *Hit - No Hit* outcome variable is strongly positively correlated with the four following chemical parameters  $Hdon$ ,  $Hdon^2$ ,  $tPSA$  and  $SlogP$ ; the variable corresponding to the positive dose-response outcome, *D-R*, is strongly correlated with the combined  $Hdon \times Hacc$  variable; the *D-R Hits* as well as *No Hits* showed no particular relationships with the

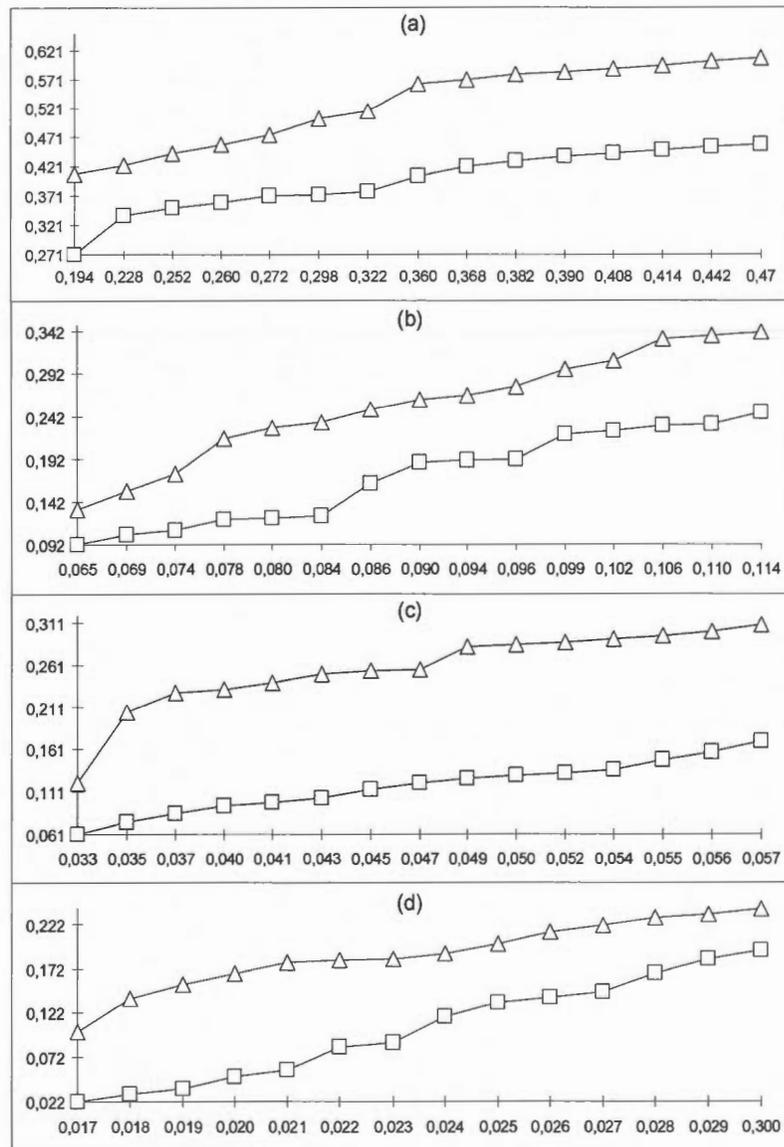
considered chemical descriptors, but *No D-R Hits* had usually bigger molecular weight (*MW*) and *SumFlag* values.

#### 4.5 Prediction of experimental HTS results using decision trees and neural networks.

In this section we present the results of our analysis of the McMaster HTS data set using the machine learning approach. Based on our discovery that the relationships between the predictors and the compound measurements in McMaster dataset are not simple (e.g. linear) we chose to use decision trees (Breiman et al. 1984) and neural networks (Haykin 1999) as powerful analytic tools capable to discover complex data dependencies. Our objective here was to build and train decision trees and neural networks in order to predict whether a given compound belongs to the partitions of *Hits* or *No Hits* using as input the values of the 10 molecular descriptors reported in Table 4.1. First, we tested several pre-processing techniques consisting of scaling and normalization of the input data. The best prediction results were obtained while scaling all input variables to the [0, 1] interval. In each training experiment we computed the numbers of false positive and false negative hits obtained for each test data set, and then determined the specificity and sensitivity of the model (Equation 4.1). It is worth noting that given a very unbalanced distribution of the *Hits* and *No Hits* partitions, only 96 hits (active compounds) and 49,904 no hits were present in the considered McMaster data set, we did not expect to obtain "perfect results" with either decision trees or neural networks methods.

To build the training model, we considered four different data groups including each time 48 hits (randomly selected from the whole set of 96 average hits) and then, in turn, 100, 500, 1000 and 2000 no hits (randomly selected from the whole set of 49,904 no hits). Four data groups of the same size (48 remaining hits and, respectively, 100, 500, 1000 and 2000 randomly selected no hits), but containing different hit and no hit compounds were used to make up the test data sets. To build our decision trees and neural networks, we used the R2008a version of MATLAB with the Levenberg-Marquardt training function (*trainlm*) for neural networks and the *classregtree* function for decision trees (for more details of these functions, see *MATLAB User's Guide*; Demuth et al. 2005). In the neural networks, the number of neurons in the hidden layer was set to 10 (and, then respectively, to 50, 100 and 200) for the groups containing 100 no hits (and, then respectively, to 500, 1000 and 2000 no

hits); 100 replicated datasets were generated for each data size and the average percentages of false positive and false negative hits were calculated.



**Figure 4.3** ROC curves for four different group sizes used for training and test: 48 hits + 100 (case a), 500 (case b), 1000 (case c) and 2000 (case d) no hits. The abscissa axis represents *(1-Specificity)*, and the ordinate axis represents *Sensitivity*. The results of the decision tree method are depicted by squares and those of the neural network method by triangles.

The ROC curve (Fawcett 2001) presentation was chosen to illustrate the results of the two competing machine learning methods (Figure 4.3). Equation 4.1 was used to compute the specificity ( $Sp$ ) and sensitivity ( $Se$ ) of both methods:

$$Sp = \frac{TN}{TN + FP} \quad \text{and} \quad Se = \frac{TP}{TP + FN}, \quad (4.1)$$

where TP is the number of true positives, FN – the number of false negatives, TN – the number of true negatives and FP – the number of false positives. Figure 4.3 shows the ROC curves obtained for each of the four considered group sizes; the results of the neural network method are indicated by triangles and those of the decision tree method by squares. The proportions of false positive and false negative hits increase as the size of No Hits partition grows. For a given group, these proportions vary in the opposite directions: the number of false positives increases as the number false negatives decreases. When comparing the cases a, b, c and d in Figure 4.3, one can notice that the sensitivity of the methods decreases as the size of the No Hits partition increases. The sensitivity was higher for neural networks than for decision trees for all four considered group sizes (Figure 4.3). The sensitivity was very low when the number of no hits in the group was large (Figure 4.3c and 4.3d). Its greatest average value, obtained for the neural network method, was about 0.61 for the group containing 100 no hits (Figure 4.3a).

#### 4.6 Probability-based hit selection method

The final step in the HTS process consists of a hit selection procedure allowing one to identify the most promising of the tested compounds which will be selected for further analysis. The hit selection procedure identifies the compounds with the highest inhibition (inhibition assay) or activation (activation assay) properties regarding the given target (e.g. selected protein of a bacterium). There exist several hit selection strategies the most known of which are the following (Malo et al. 2006): hits can be determined as a fixed number, or percentage, of all tested compounds (e.g. 100, or 1%, of the most active compounds) or as compounds whose measurements exceed a given threshold, usually expressed as a function of the mean  $\mu$  and the standard deviation  $\sigma$  of the obtained measurements, with the most commonly used thresholds of  $\mu - 3\sigma$  for inhibition assays and  $\mu + 3\sigma$  for activation assays.

Despite many recent technological advances and collected experience, the highly automated HTS process is prone to measurement errors (Makarenkov et al. 2007). Many environmental and technical factors can cause inaccuracy of the screening measurements or undesired variances in the experimental conditions, thus affecting negatively the hit selection process. Erroneous HTS data may result in some compounds incorrectly being selected as hits – false positives (FP) or mistakenly overlooked – false negatives (FN). Both types of misclassification may be extremely expensive and can undermine the results of an entire HTS campaign. In order to avoid uncertainties and reduce the overall impact of the error, many researches use replicates in their experiments, e.g. several samples of each compound are tested, limiting in this way the effect that a single error may cause to the hit selection routine (Malo et al. 2006).

The current practice of selecting hits in HTS suffers from the absence of probability models. There is no way of estimating confidence behind the decision to classify the given compound as a hit or no hit when the traditional hit selection methods are employed. We developed a method allowing one to assess the probability of each compound to be a hit. By using this method, one can select as hits the compounds whose probabilities are greater than a predefined probability-based threshold  $P_{hit}$  (e.g.  $P_{hit}$  can be set to 0.5). Given that HTS is very costly process, further experimental screens could be carried out only with compounds whose hit-outcome probabilities exceed  $P_{hit}$ .

The new hit selection method and the procedure for estimating the probability of a compound to be a hit require the presence of experimental assay replicates. Assume that we already have the screening results for the  $N$  ( $N \geq 2$ ) assay replicates. The probability-based hit selection method consists of the five following steps:

- 1) From the  $N$  available experimental replicates, we first estimate the mean,  $\tilde{\mu}_{ijp}$ , and the standard deviation,  $\tilde{\sigma}_{ijp}$ , of each tested compound  $x_{ijp}$ , where  $p$  is the plate number,  $i$  is the row number and  $j$  is the column number.
- 2) We generate *in silico*  $K$  ( $K$  was set to 1000 in our simulations described below) additional assay replicates  $R_1, R_2, \dots, R_K$ , using the estimates of each compound,  $\tilde{\mu}_{ijp}$  and  $\tilde{\sigma}_{ijp}$ , obtained

in Step 1. In our simulations all compound measurements were normally distributed, i.e.,

$$R_k = [x_{ijp(k)} \sim N(\tilde{\mu}_{ijp}, \tilde{\sigma}_{ijp})], 1 \leq k \leq K.$$

3) For all assay replicates  $R_k$  ( $1 \leq k \leq K$ ), we identify the hits using the conventional hit selection method and a fixed hit-cutting threshold (in our simulations the hit cutting threshold was set to  $\mu - 3\sigma$ , where  $\mu_k$  was the mean and  $\sigma_k$  was the standard deviation of the assay  $R_k$ , computed over all measurements of the assay).

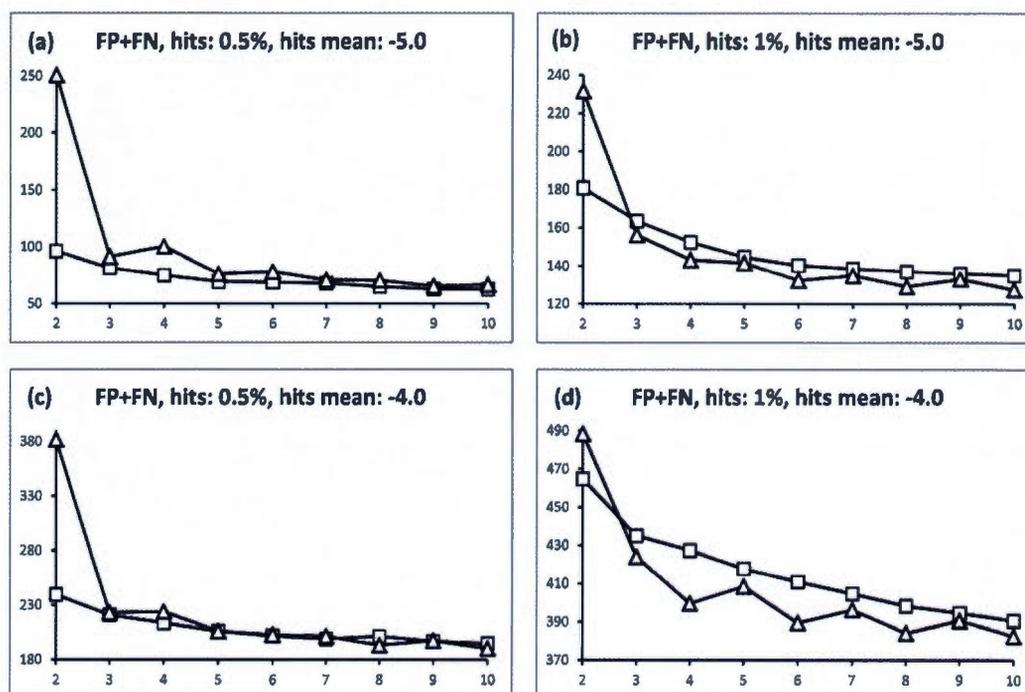
4) For each compound  $x_{ijp}$ , we count the number of times,  $c_{ijp}$ , when it was identified as a hit and estimate the overall probability of the compound  $x_{ijp}$  to be a hit as follows:

$$pr_{ijp} = c_{ijp} / K.$$

5) We select as hits the compounds with the associated probability values  $pr_{ijp} \geq P_{hit}$ , where  $P_{hit}$  is a predefined probability-based hit selection threshold computed using replicate measurements.

In order to evaluate the performances of our new hit selection method we carried out experiments with simulated data. We generated inhibition assays with  $N$  replicates,  $N = 2, 3, \dots, 10$ , following the format of the McMaster *Test dataset* – 1250 plates with 80 wells arranged in 8 rows and 10 columns. All datasets had fixed percentage of hits  $h\%$ . Values of  $h = 0.5\%$  and  $h = 1\%$  were used in our simulations. The locations of the hits were chosen randomly in such a way that the probability of each well on each plate to contain a hit was  $h\%$ . The standard hit selection was performed using the threshold of  $\mu - 3\sigma$  (i.e., all the compounds whose measurements were smaller than or equal to  $\mu - 3\sigma$  were declared hits). For each compound  $x_{ijp}$  (located in row  $i$ , column  $j$  and plate  $p$ ), we generated  $N$  original (i.e., experimental) replicated measurements  $x_{ijp(n)}$ ,  $1 \leq n \leq N$ , such that  $x_{ijp(n)}$ 's values were normally distributed with parameters  $\sim N(\mu_{ijp}, \sigma_{ijp})$ . For the no hit compounds, the  $\mu_{ijp}$ 's values followed standard normal distribution. Two types of datasets were examined: those in which the  $\mu_{ijp}$ 's values of the hit compounds followed normal distribution with parameters  $\sim N(-4, 1)$  and those with parameters  $\sim N(-5, 1)$ . The standard deviations,  $\sigma_{ijp}$ , for both hits and no hit

compounds were selected following standard normal distribution. The five steps of the hit selection method presented above were then carried out.



**Figure 4.4** Total number of false positives (*FP*) and false negatives (*FN*) obtained for the simulated data with the following parameters: (a) Hit mean =  $-5.0$ , hit percentage =  $0.5\%$ ; (b) Hit mean =  $-5.0$ , hit percentage =  $1\%$ ; (c) Hit mean =  $-4.0$ , hit percentage =  $0.5\%$ ; (d) Hit mean =  $-4.0$ , hit percentage =  $1\%$ . The results are shown for the following numbers of in silico replicates:  $K = 0$  (depicted by triangles) and  $K = 1000$  (depicted by squares) and represent the averages obtained after 100 iterations (i.e., 100 different initial assays). The x-axis represents the number of experimental replicates.

Figure 4.4 shows the results of our simulations. Cases (a) and (c) of Figure 4.4 present the results obtained for the datasets with the hit percentage of  $0.5\%$ , and cases (b) and (d) those for the datasets with the hit percentage of  $1\%$ . Typically, HTS data contain a very small hit percentage. For example, the official average hits in the McMaster Test dataset accounts for less than  $0.2\%$  of the total number of screened compounds. In the simulations, we considered the two following cases: when there was a clear distinction between the activities of the hit and no hit compounds (i.e., hit mean set to  $-5$  in cases (a) and (b)) and when this

difference was much smaller (i.e., hit mean set to -4 in cases (c) and (d)). We studied how the new method performed depending on the number of experimental replicates  $N$  (depicted on the  $x$ -axis). We compared the results yielded by the new method for the case of 1000 *in silico* replicates (i.e.,  $K = 1000$ ) with those provided by the strategy based on the experimental replicates only (i.e.,  $K = 0$ ) by measuring the totals of false positives and false negatives in each case. The probability-based hit selection threshold,  $P_{hit}$ , was set to 0.5.

$P_{hit}$	Number of hits	New hits	Removed hits	Preserved hits
$\geq 0.95$	7	0	89	7
$\geq 0.90$	10	0	86	10
$\geq 0.85$	15	0	81	15
$\geq 0.80$	22	0	74	22
$\geq 0.75$	25	0	71	25
$\geq 0.70$	27	0	69	27
$\geq 0.65$	33	0	63	33
$\geq 0.60$	37	0	59	37
$\geq 0.55$	44	0	52	44
$\geq 0.50$	55	0	41	55
$\geq 0.45$	77	4	23	73
$\geq 0.40$	110	29	15	81
$\geq 0.35$	142	59	13	83
$\geq 0.30$	186	100	10	86
$\geq 0.25$	275	188	9	87
$\geq 0.20$	393	304	7	89
$\geq 0.15$	684	593	5	91
$\geq 0.10$	1187	1096	5	91
$\geq 0.05$	2201	2108	3	93

**Table 4.2 Results obtained after applying the new probability-based hit selection method on the raw McMaster *Test dataset*. Results are shown for different values of the probability-based hit selection thresholds,  $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list.**

Analyzing the results illustrated in Figure 4.4, we can notice that *in silico* experiments bring an important improvement in the case of two experimental replicates. It is worth noting that two-replicate screens are the most frequent case in HTS campaigns. Also, for a lower hit

percentage cases (a) and (c), what represents a real-life situation in HTS, the results provided for a high number of *in silico* replicates were often equivalent to those obtained with one additional experimental replicate of the assay. The strategy using *in silico* replicates yields better results when the hits are clearly distinguishable (cases (a) and (b)). The unstable (i.e., zigzag-like) behavior of the strategy not using *in silico* data generation is due to a low number of experimental replicates (2 to 10) and to the discreteness of the established probability-based threshold  $P_{hit} = 0.5$ .

$P_{hit}$	Number of hits	New hits	Removed hits	Preserved hits
$\geq 0.95$	6	0	90	6
$\geq 0.90$	10	0	86	10
$\geq 0.85$	16	0	80	16
$\geq 0.80$	22	0	74	22
$\geq 0.75$	25	0	71	25
$\geq 0.70$	27	0	69	27
$\geq 0.65$	31	0	65	31
$\geq 0.60$	34	1	63	33
$\geq 0.55$	44	1	53	43
$\geq 0.50$	57	3	42	54
$\geq 0.45$	75	4	25	71
$\geq 0.40$	110	29	15	81
$\geq 0.35$	151	68	13	83
$\geq 0.30$	206	121	11	85
$\geq 0.25$	284	199	11	85
$\geq 0.20$	431	341	6	90
$\geq 0.15$	727	637	6	90
$\geq 0.10$	1254	1164	6	90
$\geq 0.05$	2332	2240	4	92

**Table 4.3 Results obtained after applying the new probability-based hit selection method on the McMaster *Test dataset* corrected by the Matrix Error Amendment method. Results are shown for different values of the probability-based hit selection thresholds,  $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list.**

In addition to the experiments with artificial data, we also applied our new probability-based hit selection method to analyze empirical HTS data. We conducted five separate

experiments (see Tables 4.2 to 4.6) using the McMaster two-replicate Test dataset described earlier in the article (for more details, see also Elowe et al. 2005). In all experiments we carried out the hit selection at different hit probability thresholds  $P_{hit}$  ranging from 0.05 up to 0.95 (with a step of 0.05). For each value of  $P_{hit}$ , we counted the number of selected hits and then compared them with the list of 96 average hits published by the organizers of McMaster Data Mining and Docking Competition (see Section 4.3). We determined how many of the average hits were preserved with the new hit selection, how many of them were removed and how many new hits were identified. As in our simulation study, the number of *in silico* replicates  $K$  was set to 1000.

$P_{hit}$	Number of hits	New hits	Removed hits	Preserved hits
$\geq 0.95$	9	3	90	6
$\geq 0.90$	13	3	86	10
$\geq 0.85$	17	6	85	11
$\geq 0.80$	26	6	76	20
$\geq 0.75$	29	6	73	23
$\geq 0.70$	33	6	69	27
$\geq 0.65$	39	8	65	31
$\geq 0.60$	44	9	61	35
$\geq 0.55$	49	9	56	40
$\geq 0.50$	64	13	45	51
$\geq 0.45$	89	20	27	69
$\geq 0.40$	135	56	17	79
$\geq 0.35$	187	105	14	82
$\geq 0.30$	268	183	11	85
$\geq 0.25$	392	306	10	86
$\geq 0.20$	579	493	10	86
$\geq 0.15$	931	842	7	89
$\geq 0.10$	1515	1426	7	89
$\geq 0.05$	2633	2543	6	90

**Table 4.4 Results obtained after applying the new probability-based hit selection method on the McMaster Test dataset corrected by the Partial Mean Polish method. Results are shown for different values of the probability-based hit selection thresholds,  $P_{hit}$  varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list.**

$P_{hit}$	Number of hits	New hits	Removed hits	Preserved hits
$\geq 0.95$	13	0	83	13
$\geq 0.90$	17	0	79	17
$\geq 0.85$	22	0	74	22
$\geq 0.80$	23	0	73	23
$\geq 0.75$	25	0	71	25
$\geq 0.70$	29	0	67	29
$\geq 0.65$	32	0	64	32
$\geq 0.60$	37	1	60	36
$\geq 0.55$	45	3	54	42
$\geq 0.50$	60	6	42	54
$\geq 0.45$	81	19	34	62
$\geq 0.40$	114	44	26	70
$\geq 0.35$	155	81	22	74
$\geq 0.30$	221	143	18	78
$\geq 0.25$	308	227	15	81
$\geq 0.20$	457	375	14	82
$\geq 0.15$	749	666	13	83
$\geq 0.10$	1284	1198	10	86
$\geq 0.05$	2393	2305	8	88

**Table 4.5 Results obtained after applying the new probability-based hit selection method on the McMaster Test dataset corrected by the Well Correction followed by Matrix Error Amendment methods. Results are shown for different values of the probability-based hit selection thresholds,  $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list.**

In the first experiment, we used the raw McMaster Test dataset without any error correction. The obtained results are presented in Table 4.2. In the second and third experiments, we used *Matrix Error Amendment (MEA)* and *Partial Mean Polish (PMP)* methods combined with the t-test (see Chapters II and III) to correct the raw HTS data prior carrying out the hit selection. The results of the two tests are shown in tables 4.3 and 4.4, respectively. In the last two experiments we used a two-step error correction procedure. As it has been suggested that McMaster dataset contains well systematic error (Kevorkov and Makarenkov 2005, Makarenkov et al. 2007), as first step we used the *Well Correction (WC)*

procedure (Makarenkov et al. 2007) to correct for the effect of well-located systematic error and then, similarly to the second and third experiments, we applied MEA and PMP methods to compensate for the plate-located systematic error (see Chapter III for more details on these combined systematic error correction techniques). The results of our fourth and fifth experiments are reported in Tables 4.5 and 4.6, respectively.

$P_{hit}$	Number of hits	New hits	Removed hits	Preserved hits
$\geq 0.95$	13	0	83	13
$\geq 0.90$	17	0	79	17
$\geq 0.85$	22	0	74	22
$\geq 0.80$	22	0	74	22
$\geq 0.75$	25	0	71	25
$\geq 0.70$	28	0	68	28
$\geq 0.65$	34	0	60	34
$\geq 0.60$	37	1	60	36
$\geq 0.55$	45	3	54	42
$\geq 0.50$	61	7	42	54
$\geq 0.45$	82	21	35	61
$\geq 0.40$	114	45	27	69
$\geq 0.35$	155	81	22	74
$\geq 0.30$	221	143	18	78
$\geq 0.25$	304	223	15	81
$\geq 0.20$	463	381	14	82
$\geq 0.15$	753	670	13	83
$\geq 0.10$	1290	1204	10	86
$\geq 0.05$	2402	2314	8	88

**Table 4.6 Results obtained after applying the new probability-based hit selection method on the McMaster *Test dataset* corrected by the Well Correction and Partial Mean Polish methods. Results are shown for different values of the probability-based hit selection thresholds,  $P_{hit}$ , varying from 0.05 to 0.95. For each threshold value, the list of selected hits was compared to the official McMaster average hit list.**

A closer examination of the raw McMaster data results suggest that only 55 out of the 96 official average hits (i.e., average hits identified by the McMaster Competition Organizers) have the probability to be a hit which is greater than or equal to 0.5. Comparing the results in Table 4.2 to the results in Tables 4.3 to 4.6, we can notice the impact of

systematic error on the hit selection process. The number of the preserved hits decreases to 54 when systematic error correction was carried out (to 51, when the PMP method was carried out alone). Tables 4.3 to 4.6 show that the removal of systematic error allowed some compounds to be identified as hits with the probabilities greater than or equal to 0.5: 3 new hits when MEA method was carried out, 13 when the PMP method was carried out, 6 when the WC+MEA methods were carried out and 7 when the WC+PMP methods were carried out.

The results provided for the probability level  $P_{hit}=0.3$  suggest that 10 of the original average hits have the probability to be a hit which is lower than 0.3 and they may be in fact false positives (Table 4.2). We can notice that applying MEA or PMP methods alone increase that number by 1 only (Tables 4.2 and 4.3), while using Well Correction in combination with the Matrix Error Amendment or Partial Mean Polish methods almost double the suggested number of false positives (18 removed hits, see Tables 4.5 and 4.6). The obtained results for  $P_{hit}=0.3$  level show that 100 other compounds have the probability to be a hit lower than 0.5 (Table 4.2). These compounds were missed during the hit selection performed by the McMaster Competition organizers. We think that some of these compounds should have been identified as hits if a third experimental screen of the McMaster Test assay would have been performed. The error correction of the raw data increases significantly the number of compounds with the probability to be a hit greater than 0.3, i.e., from 100 to 121 in the case of the MEA method (Table 4.3), to 143 in the cases of WC+MEA and WC+PMP (Tables 4.5 and 4.6), and to 183 when the error was corrected using the PMP method (Table 4.4).

#### **4.7 New measures for assay quality estimation depending on the number of replicates**

In HTS the replicates are used for improving the quality of hit selection results (Malo et al. 2006). Due to the excessive cost of experimental HTS, the number of replicates used in a single HTS study is usually limited. In practice, in order to ensure the optimal use of the project resources, the researchers need to decide whether carrying out an additional assay screen is appropriate or not, i.e., whether an extra assay replicate would bring an important improvement to the results quality. Here, we describe a methodology that can be used to estimate how an additional experimental replicate would affect the false positive and false negative rates of the screen.

Let  $DS$  be a dataset including  $N$  ( $N \geq 2$ ) complete, experimentally screened replicates  $S_n$ , i.e.,  $DS = \{S_n, 1 \leq n \leq N\} = \{x_{ijp(n)} : 1 \leq i \leq N_{rows}, 1 \leq j \leq N_{cols}, 1 \leq p \leq N_{pl}, 1 \leq n \leq N\}$ , where  $N_{pl}$  is the number of the plates in each assay replicate,  $N_{rows}$  is the number of rows and  $N_{cols}$  is the number of the columns on each plate.

The effect that testing an additional replicate would have on the quality of the selected hits can be estimated following the steps below:

1. Using the  $N$  available experimental (i.e., *in vitro*) replicates  $S_n$ ,  $1 \leq n \leq N$ , we estimate the mean  $\tilde{\mu}_{ijp}$  and the standard deviation  $\tilde{\sigma}_{ijp}$  of each compound  $x_{ijp}$  in  $DS$ .
2. Using the estimates obtained in Step 1, we generate  $K$  *in silico* (e.g.,  $K$  was equal to 1000 in our simulations) complete assay replicates  $R_1, R_2, \dots, R_K$ . The simulated measurements of each compound  $x_{ijp}$  will be normally distributed with the mean  $\tilde{\mu}_{ijp}$  and the standard deviation  $\tilde{\sigma}_{ijp}$ , i.e.,  $R_k = \{x_{ijp(k)} \sim N(\tilde{\mu}_{ijp}, \tilde{\sigma}_{ijp}), k = 1, 2, \dots, K\}$ .
3. The expected *false positive change rate* ( $FP_{ijp}$ ) and *false negative change rate* ( $FN_{ijp}$ ) of the compound  $x_{ijp}$  can be defined in the following way:

$$FP_{ijp} = \text{Max} \left( \frac{1}{K} \sum_{k=1}^K \delta_{ijp(k)} - \frac{1}{N} \sum_{n=1}^N \lambda_{ijp(n)}; 0 \right) \times 100\%, \quad (4.2)$$

$$FN_{ijp} = \text{Max} \left( \frac{1}{N} \sum_{n=1}^N \lambda_{ijp(n)} - \frac{1}{K} \sum_{k=1}^K \delta_{ijp(k)}; 0 \right) \times 100\%, \quad (4.3)$$

where  $\lambda_{ijp(n)}$  is equal to 1 if the compound  $x_{ijp}$  is selected as hit in the *in vitro* replicate  $S_n$  and equal to 0 otherwise, and  $\delta_{ijp(k)}$  is equal to 1 if the compound  $x_{ijp}$  is selected as hit in the *in silico* replicate  $R_k$  and equal to 0 otherwise.

4. Let  $N_{FP}$  be the number of compounds  $x_{ijp}$  whose false positive change rate value  $FP_{ijp}$  is positive, i.e.,  $N_{FP} = \{x_{ijp} : FP_{ijp} > 0, 1 \leq i \leq N_{rows}, 1 \leq j \leq N_{cols}, 1 \leq p \leq N_{pl}\}$ , and let  $N_{FN}$  be the number of compounds  $x_{ijp}$  whose false negative change rate value  $FN_{ijp}$  is positive, i.e.,  $N_{FN} = \{x_{ijp} : FN_{ijp} > 0, 1 \leq i \leq N_{rows}, 1 \leq j \leq N_{cols}, 1 \leq p \leq N_{pl}\}$ .

The expected *false positive change rate* ( $FP_{N+1}$ ) and *false negative change rate* ( $FN_{N+1}$ ) of the assay regarding an additional *in vitro* replicate  $N+1$  can be defined as follows:

$$FP_{N+1} = \frac{\sum_{p=1}^{N_{pl}} \sum_{i=1}^{N_{rows}} \sum_{j=1}^{N_{cols}} FP_{ijp}}{N_{FP}} \quad \text{and} \quad (4.4)$$

$$FN_{N+1} = \frac{\sum_{p=1}^{N_{pl}} \sum_{i=1}^{N_{rows}} \sum_{j=1}^{N_{cols}} FN_{ijp}}{N_{FN}}. \quad (4.5)$$

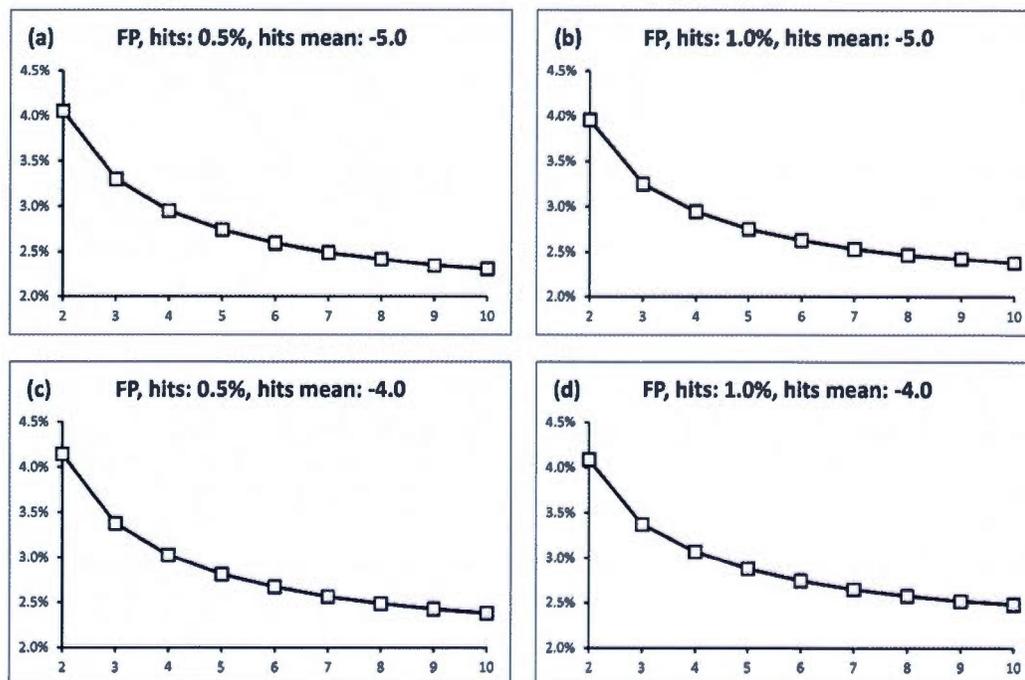
5. Finally, the *average probability of hit/non-hit outcome change rate*  $POC_{N+1}$  of the assay regarding an additional *in vitro* replicate  $N+1$  can be defined as follows:

$$POC_{N+1} = \frac{\sum_{p=1}^{N_{pl}} \sum_{i=1}^{N_{rows}} \sum_{j=1}^{N_{cols}} \left| \frac{1}{N} \sum_{n=1}^N \lambda_{ijp(n)} - \frac{1}{K} \sum_{k=1}^K \delta_{ijp(k)} \right|}{N_{rows} \times N_{cols} \times N_{pl}}. \quad (4.6)$$

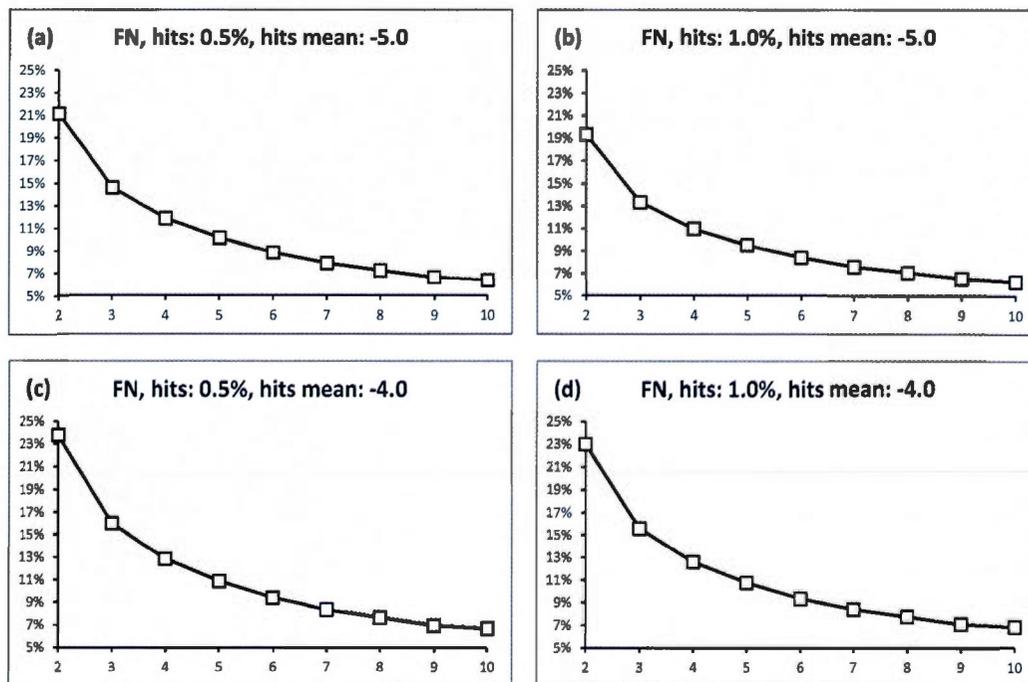
We conducted experiments with simulated HTS data in order to evaluate the proposed assay quality estimators. As in our previous simulations, we generated inhibition assays with  $N$  initial replicates,  $N = 2, 3, \dots, 10$ , following the format of the McMaster *Test dataset* – 1250 plates with 80 wells arranged in 8 rows and 10 columns. The hit percentage of all datasets was fixed to  $h\%$  where  $h$  was equal to either 0.5% or 1%. The hits were placed at randomly chosen locations within the plates in such a way that the probability of each well on each plate to contain a hit was  $h\%$ . The standard hit selection procedure was carried out to select hits using the threshold of  $\mu - 3\sigma$  (i.e., all compounds with measurements smaller than or equal to  $\mu - 3\sigma$  were declared hits).

For each compound  $x_{ijp}$  (located in row  $i$ , column  $j$  and plate  $p$ ), we generated  $N$  original (i.e., experimental) replicated measurements  $x_{ijp(n)}$ ,  $1 \leq n \leq N$ , such that  $x_{ijp(n)}$ 's values followed a normal distribution with parameters  $\sim N(\mu_{ijp}, \sigma_{ijp})$ . For the *non-hit compounds*, the  $\mu_{ijp}$ 's values followed standard normal distribution. Two types of datasets were examined: those in which the  $\mu_{ijp}$ 's values of the *hit compounds* followed normal distribution with

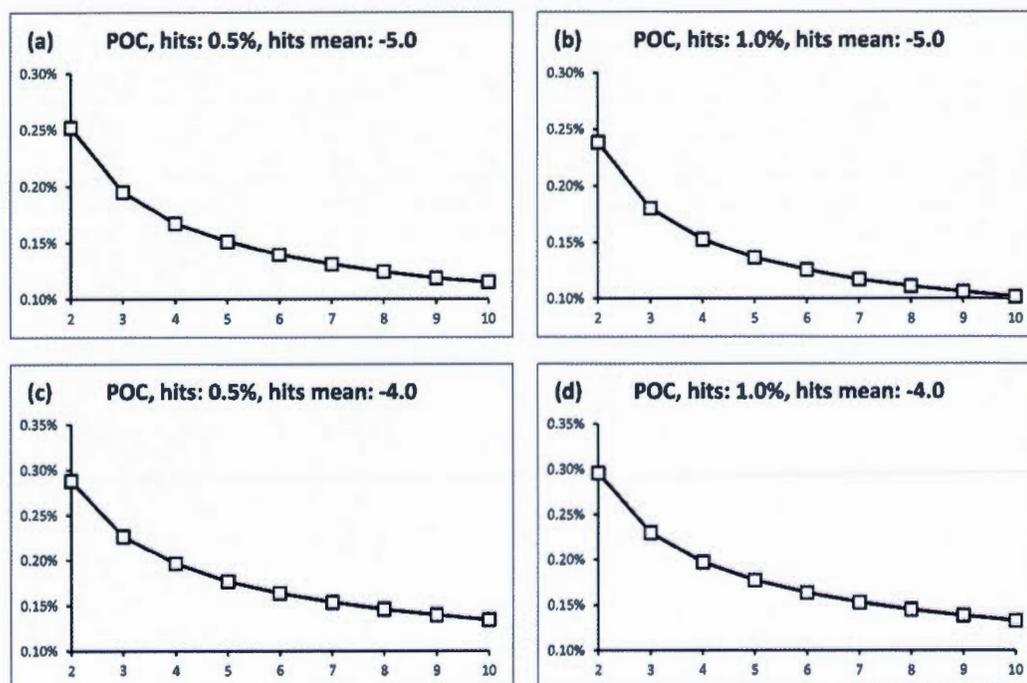
parameters  $\sim N(-4,1)$  and those with parameters  $\sim N(-5,1)$ . The standard deviations,  $\sigma_{ijps}$  for both hits and non-hit compounds were selected following standard normal distribution.



**Figure 4.5** False positive change rate ( $FP_{N+1}$ ) obtained for the simulated data with the following parameters: (a) Hit mean =  $-5.0$ , hit percentage =  $0.5\%$ ; (b) Hit mean =  $-5.0$ , hit percentage =  $1\%$ ; (c) Hit mean =  $-4.0$ , hit percentage =  $0.5\%$ ; (d) Hit mean =  $-4.0$ , hit percentage =  $1\%$ . The results are shown for 1000 *in silico* replicates and represent the averages obtained after 100 iterations (i.e., 100 different initial assays). The x-axis represents the number of experimental (*in vitro*) replicates.



**Figure 4.6** False negative change rate ( $FN_{N+1}$ ) obtained for the simulated data with the following parameters: (a) Hit mean =  $-5.0$ , hit percentage = 0.5%; (b) Hit mean =  $-5.0$ , hit percentage = 1%; (c) Hit mean =  $-4.0$ , hit percentage = 0.5%; (d) Hit mean =  $-4.0$ , hit percentage = 1%. The results are shown for 1000 *in silico* replicates and represent the averages obtained after 100 iterations (i.e., 100 different initial assays). The *x*-axis represents the number of experimental (*in vitro*) replicates.



**Figure 4.7** Average probability of hit/non-hit outcome change ( $POC_{N+1}$ ) obtained for the simulated data with the following parameters: (a) Hit mean =  $-5.0$ , hit percentage =  $0.5\%$ ; (b) Hit mean =  $-5.0$ , hit percentage =  $1\%$ ; (c) Hit mean =  $-4.0$ , hit percentage =  $0.5\%$ ; (d) Hit mean =  $-4.0$ , hit percentage =  $1\%$ . The results are shown for 1000 *in silico* replicates and represent the averages obtained after 100 iterations (i.e., 100 different initial assays). The x-axis represents the number of experimental (*in vitro*) replicates.

For all generated datasets, the values of the *false positive change rate* ( $FP_{N+1}$ ), *false negative change rate* ( $FN_{N+1}$ ) and *average probability of hit/non-hit outcome change rate* ( $POC_{N+1}$ ) were calculated following the proposed methodology. The obtained results are shown on Figures 4.5, 4.6 and 4.7, respectively. In general, all three quality assessment measures demonstrate similar behavior suggesting that the assay quality is highly dependent on the number of the experimental (*in vitro*) replicates. The highest changes in the values of the proposed statistics were obtained for the case of two and three experimental replicates; these values decreased rapidly when an extra replicate was tested till the number of replicates reached five. For five and more replicates, the change rates decrease, accounting for positive effects of an additional experimental replicate, was much slower. Comparing the four

considered practical situations (see panels (a) to (d) in Figures 4.5-4.7), we notice that the impact of an additional replicate is more important when the hits are more easily distinguishable from the non-hits (see panels (a) and (b) in Figures 4.5-4.7). Also, the increase of the hit percentage in the assay causes the decrease of the positive impact of an extra replicate in terms of the false positive rate (Figures 4.5).

MAC ID	Plate	Row	Column	M1	M2	Hit probability	FPer	FNcr
MAC-0120363	1	8	9	91.78	50.75	0.57281	0.073	0.000
MAC-0121481	3	8	9	97.12	50.98	0.51694	0.017	0.000
MAC-0121668	5	8	9	93.7	53.25	0.53064	0.031	0.000
MAC-0122467	16	5	1	84.45	65.61	0.5	0.000	0.000
MAC-0126835	35	5	6	41.57	100.29	0.55553	0.056	0.000
MAC-0128437	55	7	9	90.84	52.81	0.56693	0.067	0.000
MAC-0112108	91	1	3	71.49	62.82	0.96536	0.000	0.035
MAC-0112764	97	2	9	69.06	64.16	0.99971	0.000	0.000
MAC-0114479	100	1	9	81.65	59.31	0.65812	0.158	0.000
MAC-0113020	103	6	4	65.85	81.38	0.5723	0.072	0.000
MAC-0114615	107	5	5	70.83	67.44	0.99975	0.000	0.000
MAC-0114159	112	2	8	67.08	72.61	0.96962	0.000	0.030
MAC-0115084	116	4	10	75.2	74.21	0.74427	0.244	0.000
MAC-0115794	131	6	5	68.94	69.46		0.000	0.000
MAC-0119733	156	2	2	59.68	69.88	0.97777	0.000	0.022
MAC-0128921	161	7	5	74.21	68.5	0.90099	0.000	0.099
MAC-0130772	185	7	2	68.53	73.86	0.92493	0.000	0.075
MAC-0130492	187	4	1	76.43	70.79	0.69271	0.193	0.000
MAC-0130938	188	3	5	64.82	70.58	0.99454	0.000	0.005
MAC-0131221	192	6	3	72.12	72.22	1	0.000	0.000
MAC-0132669	209	8	2	73.82	70.75	0.96313	0.000	0.037
MAC-0133856	224	2	4	73.24	72.19	1	0.000	0.000
MAC-0134063	226	2	9	69.87	79.64	0.52245	0.022	0.000
MAC-0134899	232	4	7	76.39	56.65	0.80571	0.306	0.000
MAC-0135007	240	8	1	54.51	89.2	0.57262	0.073	0.000
MAC-0136174	243	7	4	67.68	74.32	0.8876	0.000	0.112
MAC-0135559	243	8	1	63.46	79	0.6876	0.188	0.000
MAC-0136881	247	2	9	86.23	61.32	0.54013	0.040	0.000
MAC-0136292	254	2	3	50.17	96.6	0.52825	0.028	0.000
MAC-0136229	254	7	7	80.05	64.31	0.64137	0.141	0.000
MAC-0137495	267	1	10	72.17	76.71	0.60253	0.103	0.000

MAC-0138159	269	2	9	70.63	78.74	0.5339	0.034	0.000
MAC-0139408	273	6	3	65.62	70.86	0.99522	0.000	0.005
MAC-0140284	285	7	4	60.16	79.77	0.69727	0.197	0.000
MAC-0139275	287	7	6	87.72	55.3	0.58595	0.086	0.000
MAC-0140548	292	2	10	56.26	80.66	0.70489	0.205	0.000
MAC-0141166	306	4	4	96.16	51.14	0.52444	0.024	0.000
MAC-0144065	325	2	3	77.28	67.31	0.70838	0.208	0.000
MAC-0144510	332	1	1	62.2	84.51	0.55968	0.060	0.000
MAC-0145361	333	1	4	69.55	71.72	1	0.000	0.000
MAC-0147323	349	6	3	76.7	67.48	0.73818	0.238	0.000
MAC-0148399	352	2	7	90.11	50.7	0.59278	0.093	0.000
MAC-0103980	382	6	7	67.81	74.83	0.85474	0.000	0.145
MAC-0106260	396	7	1	76.44	73.58	0.50558	0.006	0.000
MAC-0107974	401	3	5	75.15	74.64	0.70174	0.202	0.000
MAC-0108994	404	4	3	66.56	66.31	1	0.000	0.000
MAC-0110039	408	1	4	65.28	65.37	1	0.000	0.000
MAC-0110027	409	6	3	70.12	68.1	1	0.000	0.000
MAC-0110562	410	2	3	53.25	72.71	0.89222	0.000	0.108
MAC-0112179	415	4	9	68.87	50.68	0.95326	0.000	0.047
MAC-0112287	415	6	4	63.42	74.15	0.87779	0.000	0.122
MAC-0114842	426	5	7	67.27	54.76	0.98747	0.000	0.013
MAC-0115469	429	1	4	74.24	62.11	0.87082	0.000	0.129
MAC-0116655	432	8	8	58.59	71.82	0.93126	0.000	0.069
MAC-0117240	434	8	3	69.57	74.02	0.92702	0.000	0.073
MAC-0117820	435	5	4	74.98	50.57	0.84233	0.000	0.158
MAC-0117987	435	7	8	64.99	78.94	0.66982	0.170	0.000
MAC-0122178	448	2	5	77.42	66.88	0.70764	0.208	0.000
MAC-0122411	448	7	6	76.19	73.68	0.53017	0.030	0.000
MAC-0122661	449	4	9	74.53	69.98	0.88873	0.000	0.111
MAC-0122330	449	8	1	76.9	63.95	0.76152	0.262	0.000
MAC-0122959	450	2	4	69.1	71.16	1	0.000	0.000
MAC-0122586	451	1	4	62.32	64.19	1	0.000	0.000
MAC-0123662	453	4	2	51.74	79.75	0.74633	0.246	0.000
MAC-0124103	454	2	1	76.62	67.35	0.7444	0.244	0.000
MAC-0124476	455	7	3	84.03	65.32	0.51514	0.015	0.000
MAC-0125372	457	7	7	85.45	62.59	0.53521	0.035	0.000
MAC-0127264	460	3	9	71.76	69.5	0.99995	0.000	0.000
MAC-0137845	475	7	5	54.01	81.18	0.70791	0.208	0.000
MAC-0140989	479	7	2	74.94	72.25	0.85699	0.000	0.143

<i>MAC-0140910</i>	483	5	7	63.47	68.77	0.99961	0.000	0.000
MAC-0143736	488	7	1	58.47	90.69	0.51114	0.011	0.000
MAC-0144119	490	5	1	75.4	72.27	0.77744	0.277	0.000
MAC-0144345	492	1	2	87.73	55.54	0.58353	0.084	0.000
<i>MAC-0144586</i>	492	3	9	73.78	70.55	0.96197	0.000	0.038
<i>MAC-0145030</i>	493	6	10	68.41	64.36	1	0.000	0.000
MAC-0147938	513	1	5	83.04	65.26	0.53943	0.039	0.000
<i>MAC-0149343</i>	518	7	5	68.77	64.42	0.99995	0.000	0.000
MAC-0150159	525	6	7	77.25	67.42	0.70826	0.208	0.000
MAC-0149121	526	6	7	66.52	80.39	0.58983	0.090	0.000
<i>MAC-0150029</i>	528	7	6	72.84	73.31	1	0.000	0.000
MAC-0101192	545	1	9	68.86	78.85	0.59299	0.093	0.000
MAC-0101115	546	1	4	76.33	50.75	0.8155	0.316	0.000
MAC-0101665	548	7	2	75.65	60.34	0.82095	0.321	0.000
MAC-0105044	557	2	9	76.51	56.61	0.80269	0.303	0.000
<i>MAC-0104038</i>	563	8	3	71.72	70.03	1	0.000	0.000
<i>MAC-0104867</i>	567	6	2	66.49	57.06	0.99753	0.000	0.002
MAC-0105511	571	3	10	70.68	79.26	0.50558	0.006	0.000
MAC-0106449	574	3	9	90.04	55.41	0.55295	0.053	0.000
MAC-0106706	578	4	1	59.64	86.55	0.55718	0.057	0.000
MAC-0107801	585	4	6	79.16	60.07	0.71475	0.215	0.000
<i>MAC-0107329</i>	587	5	5	67.75	52.62	0.97514	0.000	0.025
MAC-0111457	591	4	5	75.72	69.42	0.78258	0.283	0.000
MAC-0109304	607	1	2	77.26	69.12	0.6744	0.174	0.000
<i>MAC-0110019</i>	609	3	5	68	71.29	0.99947	0.000	0.001
<i>MAC-0109949</i>	609	8	7	70.79	65.17	0.99394	0.000	0.006

**Table 4.7 The 96 average hits from the McMaster Test assay with their MAC IDs, the first (M1) and the second (M2) measurement values, hit probability computed from theoretical distribution, and false positive (FPch) and false negative (FNcr) change rates also computed from theoretical distributions. The consensus hit are italicized.**

We also evaluated the proposed assay quality estimators on an experimental HTS data - the McMaster Test dataset. The calculated values of *hit probability*, *false positive change rate (FPch)*, *false negative change rate (FNch)* of all original 96 average hits are shown in Table 4.7. Our calculations identified 12 of the original hits as “absolute” hits with *hit probability* equal to 1 and *FPch* and equal to 0. We determined that more than one third, 33,

of the original hits have *hit probability* greater than 90% of which 29 have *hit probability* greater than 95% and 21 greater than 99%. The calculations showed that the maximum *false positive change rate* of an original McMaster hit is 0.321 and the maximum *false negative change rate* is 0.158. The average values of *FPch* and *FNch* for all 96 hits are 0.074 and 0.017 respectively. The hit probability in Table 4.7 was computed from the theoretical distribution.

#### 4.8 Discussion and future developments

In this article, we discussed the applications of the three well-known and one new statistical methods to the analysis of experimental high-throughput screening data. First, we carried out the polynomial RDA in order to discover relationships between the experimental HTS measurements and the values of ten chemical descriptors characterizing the associated compounds. The polynomial RDA analysis was effective in finding relationships between the hit/no hit outcomes and the values of chemical descriptors when small HTS datasets were considered (e.g. with 42 hits and 100 no hits).

Second, we showed that neural networks are much more effective than decision trees regarding the *in silico* prediction of HTS results. Neural networks were usually more discriminate than decision trees when the model's parameters were well fitted. We established that the increase in the number of no hits leads to a drastic drop in the sensitivity of machine learning methods; this trend is particularly noticeable for the decision tree method.

HTS is often used to screen a huge number of compounds, millions in some cases, but it yields only a few hits for further investigation. For instance, in the considered McMaster *Test dataset* only 96 of 50,000 tested compounds were identified as hits. This disproportion suggests that the McMaster *Test dataset* is extremely unbalanced: the number of hits is less than 0.2% of the total number of tested compounds. Having too many negative samples in the training set biases a neural network decision towards the negative samples. Several approaches are possible to address this problem. One of them consists of a random selection of balanced same-size partitions used to build and train a number of separate networks. All obtained networks can be used to predict the proposed test set and their outputs can be

averaged to produce the final result. Another approach, one could use to produce balanced training sets, is the bootstrapping hit samples. In this way, the number of hits and no hits in the set would be equal, while some of the hit samples may have been selected several numbers of times. Our experiences with the *McMaster Test dataset* using the above-mentioned approaches showed that, as expected, they can efficiently reduce the bias caused by the imbalance in the dataset, but are not powerful enough to correct the main problem of HTS: lack of positive samples for achieving a satisfactory level of neural networks or decision trees training. Thus, our future developments will be focused on the examination of the results provided by neural networks with higher numbers of hits (e.g. by adding to the set of hits, all no hit compounds whose experimental measurements are close to the hit measurements) and chemical predictors (e.g. by adding to the set of 10 predictors the docking scores provided by certain HTS software) in the considered training and test data sets.

We also presented a new hit selection method allowing one to assess the probability of each considered compound to be a hit. This probability is computed by estimating the mean value and the standard deviation of each compound from the available experimental replicates of the given HTS assay and then by generating, using those estimates, a sufficient number of *in silico* replicates of each compound. The obtained probabilities can be used for limiting the number of compounds that should be tested in further experimental trials, thus reducing the cost of the associated experimental HTS campaign.

Finally, we proposed a new methodology for estimating the effect of an additional replicate on the quality of the obtained HTS results. Such a methodology allows researchers to use the available experimental data for evaluating how an additional replicate would affect the current false positive and false negative rates. We tested our new methodology on simulated data. The results of our experiments suggest that screening another replicate could be especially beneficial when no more than five *in vitro* replicated screens have been carried out previously as well as in the cases when the number of hits in the assay was small and the hits were easily distinguishable from the inactive compounds.

## CONCLUSION

High-throughput screening is a relatively new technology that was rapidly embraced both by the pharmaceutical industry and the academia. The complexity and the novelty of HTS present many challenges, which require attention, in order to derive benefits from the evolving technology. The importance of HTS is well recognized, and is expected that high-throughput screening will remain in the focus of the research community in the coming years.

In this thesis, we addressed two fundamental issues in High-Throughput Screening (HTS): negative effect that systematic error has over the hit selection process and very high cost associated with HTS experiments. We investigated the possibility of detecting the presence of systematic bias in raw HTS data and determining its exact location. Three well-known statistical tests were considered in the context of experimental HTS. Their ability to detect systematic error was evaluated in various practical situations. Then, we presented our efforts in developing new error correction methods that focus on modifying only the measurements located in the plate's rows and columns affected by systematic error and leaving the rest of the data unchanged. Thus, we formulated two new error correction techniques that efficiently eliminate systematic error from raw HTS data if it is present while minimizing the unintended side effects of the error correction process. HTS Helper software implementing the new methods was made available to the scientific community. Furthermore, we addressed the issue of the high cost of experimental HTS campaigns by offering HTS researchers a methodology for deciding whether an additional assay replicate should be carried out for improving the results quality. The proposed methodology also includes a newly developed procedure allowing one to assess the probability of each compound of the assay to be a hit. We then used the latter procedure to define a new probability-based hit selection method that guarantees that all selected compounds have the probability to be a hit above a chosen in advance level. Moreover, we considered the use of virtual HTS, i.e., *in silico* methods, for predicting the hit/non-hit outcomes of HTS tests, in order to reduce the number of compounds that need to be tested experimentally. We carried out Polynomial RDA analysis to establish first whether there exist relationships between the

compounds and a set of available molecular descriptors and then to determine if the detected relationships are sufficient for successful outcome prediction. We studied, as well, the usability of two machine learning methods for constructing and training prediction models in the HTS context. The new methods described in this thesis represent our scientific contribution intended for accomplishing quality improvements of the high-throughput screening hit selection procedures as well as for decreasing the overall cost associated with experimental HTS.

In Chapter II, we discussed and tested three statistical methods for systematic error detection in experimental HTS data. We studied Student's t-test, the  $\chi^2$  goodness-of-fit test and Discrete Fourier Transform followed by the Kolmogorov-Smirnov goodness-of-fit test. We examined the performances of the three tests on a wide range of artificially generated high-throughput screening data constructed to recreate a variety of real-life situations. The data parameters used in our simulations included: plate size, systematic error magnitude, hit percentage and systematic error location. We determined that the t-test, which is the simplest and computationally fastest of the three tests, outperformed, in most cases, the  $\chi^2$  goodness-of-fit and Kolmogorov-Smirnov tests. The t-test demonstrated a robust behavior, which was often independent of the simulations parameters. The calculated values of Cohen's kappa coefficient suggested a good performance of this test for all plate sizes, hit percentages and noise magnitudes, and, in particular, for large plates and high level of systematic error. Hence, we can recommend the t-test as a method of choice for systematic error detection in experimental HTS.

On the contrary, highlighted in some works (Kelley 2005, Root et al. 2003) Discrete Fourier Transform followed by the Kolmogorov-Smirnov goodness-of-fit test strategy provided very disappointing results. It is worth noting that the latter combined strategy was both the most computation-intensive and the worst performing method among the three tested tests. Our simulations showed that the Kolmogorov-Smirnov test can still be used to examine small-sized HTS plates (i.e., 96-well plates), but we strongly suggest not using it for larger plates (i.e., 384 and 1536-well plates). A deeper analysis showed that the main reason for such a poor performance of the Kolmogorov-Smirnov test was the fact that the original

normally distributed data deviated from the normality after the application of Discrete Fourier Transform.

The third method, the  $\chi^2$  goodness-of-fit test, suggested in Makarenkov et al. 2007), showed a lower than the t-test performance but can still be employed for detecting the presence of systematic error in HTS assays using small plate sizes. The simulation results demonstrated that unlike the t-test, the  $\chi^2$  goodness-of-fit test appears to be very sensitive to the type and variance of systematic error. The main advantage of the  $\chi^2$  goodness-of-fit test is that it can be used to assess the assay's hit distribution surface for the presence of systematic error affecting compounds located at the same well location across all plates of the assay.

In addition to the experiments carried out with simulated data, we also applied the three considered tests for the analysis of real HTS data. Our goal was to study how systematic error affects the hit selection process in HTS. The obtained results confirmed that when uncorrected HTS data are used for hit selection, several selected hits (about 30%, in the case of the considered McMaster Test assay) may be in fact false positives. We also observed that the application of some aggressive data normalization procedures, such as B-score (Brideau et al. 2003), can easily bias the results of the hit selection process by introducing an important number of false positives and false negatives.

The conducted experiments demonstrated that the presence of systematic error can be successfully assessed by employing the t-test. Hence, it is also possible to improve the quality of experimental HTS results by adding a preliminary systematic error detection step to the HTS workflow. The information collected during that extra step should be used to determine the most appropriate error correction method for the given situation thus permitting the error correction step to be ruled out if no presence of systematic error has been detected.

In Chapter III, we formulated two new methods, called Matrix Error Amendment (MEA) and Partial Mean Polish (PMP), for eliminating plate-specific systematic error from experimental HTS data. Both new methods assume that the exact locations (plates, rows and columns) affected by systematic error are known. This information can be acquired using the t-test or the  $\chi^2$  goodness-of-fit test as described in Chapter II or Dragiev et al. (2011). Both

methods modify only the measurements of the compounds at the locations affected by systematic error. The adjusted compounds measurements remain on the same scale with raw data, thus permitting a global hit selection to be performed when only a part of the compounds of the assay have been treated for the removal of systematic error.

We conducted comprehensive experiments with artificially generated HTS data by considering different plate sizes, hit percentages and systematic error magnitudes. In all our experiments we compared the proposed MEA and PMP methods with the B-score (Brideau et al. 2003) procedure and hit selection based on uncorrected data. We observed that both new methods outperformed the B-score and no-correction procedures in all cases when the number of the plate's rows and columns affected by systematic error was low. We observed, however, the situations in which B-score method yielded better results. The B-score method showed a more stable behaviour than MEA and PMP only when the hit percentage was above 3%, or the number of rows and columns affected by systematic error were high. We should underline that the latter cases have mainly theoretical value as in a typical HTS campaign the hit percentage is usually under 1% (for example the McMaster Test dataset has a hit percentage of 0.19%). Comparing the two new methods between them, we noticed that MEA was generally a better method for correcting systematic error within 96-well plates, while PMP performed better for 384- and 1536-well plates.

Moreover, we evaluated the MEA and PMP methods on real data (McMaster Test assay). Our study showed that the new methods can be applied in combination with the Well Correction procedure (Makarenkov et al. 2007). We agreed on the following recommended way to treat experimental HTS data. First, if controls were used, the data should be normalized using Percent of control or Normalized percent inhibition transformation (see chapter I). Second, the hit distribution surface should be tested for presence of systematic error affecting the assay globally. If such an error is detected, it should be corrected using the Well Correction method. Third, the t-test should be carried out on the plate-by-plate basis to detect if local row or column systematic error is present in each plate of the assay given plate. All the plate's measurements affected by systematic error should then be treated using the MEA or PMP methods. We also showed how, with simple modifications, the newly proposed PMP method can be adapted for the correction of multiplicative systematic bias (PMP and

MEA were originally developed for the correction of additive type of systematic error in experimental HTS).

Furthermore, we provided a program that implements the two new methods, Matrix Error Amendment and Partial Mean Polish, as well as the B-score, Z-score and Well Correction methods. The executables and the source code of our software can be downloaded from: [http://www.info2.uqam.ca/~makarenkov\\_v/HTS\\_Helper](http://www.info2.uqam.ca/~makarenkov_v/HTS_Helper).

In Chapter IV, we presented a new probability-based hit selection procedure. We developed a methodology allowing one to assess for every compound of the assay its probability to be a hit. By estimating empirically the distribution parameters of every compound from the available experimental assay replicates, we were able to simulate, *in silico*, additional assay replicates. That allowed us to estimate the probability of each compound to be a hit. We then used that information to identify as hits the compounds whose probability was higher than a predefined threshold. Using the simulated assay replicates, we were also able to estimate the current false positive and false negative change rates, allowing the researchers to assess the effect that an additional experimental assay replicate will have on the false positive and false negative rates, and thus to decide whether screening another assay replicate is justified or not.

The use of replicated high-throughput screens has become a common practice (Malo et al. 2006) especially during the secondary screening when relatively small number of compounds is tested. We believe that further work is needed for developing more precise probability models for evaluating the “drug-likeness” of the selected hits. The methodology described in Chapter IV assumes normally distributed data, however, it can be easily extended to include an extra step of assessing the distribution of the considered experimental data and to adjust *in silico* simulations for achieving more precise results.

In Chapter IV, we also investigated the possibility of using a group of ten chemical descriptors for predicting the outcome of HTS experiments. We carried out polynomial RDA in order to discover relationships between experimental HTS measurements and the available values of ten chemical descriptors. Our analysis showed that several descriptors were

strongly correlated with the hit/non-hit outcomes. Thus, enough information was often present to distinguish between hits and non-hits.

Moreover, we evaluated two machine learning methods, neural networks and decision trees, for building prediction models. Our experiments showed that neural networks are much more effective than decision trees in terms of *in silico* prediction of HTS results. High-throughput screening is usually used to screen large and highly unbalanced datasets that include thousands of non-hits and only a few hits. We established that the increase in the number of non-hits causes a significant decrease in the sensitivity of both machine learning methods. This trend was particularly noticeable for the decision trees. We tested several approaches to address that problem. For instance, we conducted training with balanced sets, constructing training sets by randomly selecting balanced same-size partitions to build and train a number of separate neural networks. The obtained networks were used for prediction and their outputs were averaged to produce the final result. Another approach, we applied, was based on the bootstrapping the hits samples. In this way, the number of hits and non-hits in each training set was equal, while some of the hit samples were selected several times. Despite the use of balanced training sets, what, as expected, allowed us to reduce the bias in the decisions of the predicting models, such an adjustment was not powerful enough to correct the main hurdle for predicting correct hit/non-hit outcomes in experimental HTS: the lack of positive samples for satisfactory training of the neural networks and the decision tree classifiers. Thus, future developments could be focused on improving the training of the prediction models and extending the set of chemical predictors as well as evaluating other machine learning and classification methods that are less sensitive to the hit/non-hit unbalance.

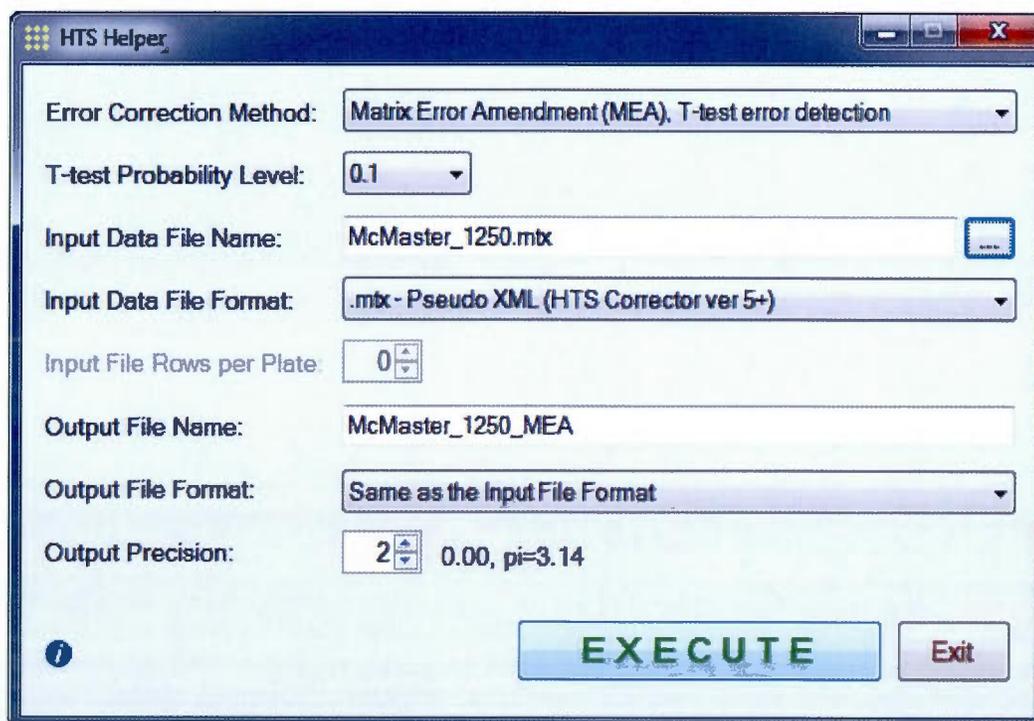
In our experiments, we considered the values of only ten common chemical descriptors. Some additional work should be done for extending the number of descriptors used for prediction. Burton et al. (2006) showed that not all chemical descriptors have the identical prediction power. Further research is needed for determining the optimal set of descriptors to be used in virtual HTS.

## APPENDIX A

### HTS HELPER SOFTWARE

#### A.1 HTS Helper Utility

This appendix presents a handy freeware utility, *HTS Helper*, which can be used to analyse high-throughput screening data. The *HTS Helper* utility and the content of this appendix are available online at: [http://www.info2.uqam.ca/~makarenkov\\_v/HTS\\_Helper](http://www.info2.uqam.ca/~makarenkov_v/HTS_Helper).



The screenshot shows the HTS Helper application window. The title bar reads "HTS Helper". The interface includes the following fields and controls:

- Error Correction Method:** A dropdown menu set to "Matrix Error Amendment (MEA), T-test error detection".
- T-test Probability Level:** A dropdown menu set to "0.1".
- Input Data File Name:** A text field containing "McMaster\_1250.mtx" with a browse button (three dots) to its right.
- Input Data File Format:** A dropdown menu set to ".mtx - Pseudo XML (HTS Corrector ver 5+)".
- Input File Rows per Plate:** A spin box set to "0".
- Output File Name:** A text field containing "McMaster\_1250\_MEA".
- Output File Format:** A dropdown menu set to "Same as the Input File Format".
- Output Precision:** A spin box set to "2" followed by the text "0.00, pi=3.14".

At the bottom of the window, there is an information icon (i) on the left, a large blue button labeled "EXECUTE" in the center, and a smaller button labeled "Exit" on the right.

Figure A.1 *HTS Helper*, Windows Forms executable (version 1.0, March 22<sup>nd</sup>, 2012)

High-throughput screening (HTS) is a large-scale, greatly automated early step in drug development during which thousands of chemical compounds are screened and their activity levels measured in order to identify potential drug candidates (i.e., hits). Many technical, environmental and procedural factors can cause systematic measurement error or inequalities in the conditions in which the measurements are taken. Systematic error introduces inaccuracy in the data by over- or underestimating compounds' true activity levels and thus it has the potential to critically disturb the hit selection process. The systematic error is almost always location related. Usually, it affects compounds located in the same row or column on the screening plate and it may affect only a single plate or a sequence of plates in the HTS assay. It is of high importance to eliminate the effect of the systematic error on the HTS data before the hit selection process in order to ensure high quality results.

The *HTS Helper* utility has been created to facilitate the systematic error correction of experimental HTS data. It implements several error correction and normalization methods:

- Matrix Error Amendment
- Partial Mean Polish
- Well Correction
- Z-score normalization
- B-score normalization

Z-score and B-score are two widely used normalization methods designed to compensate for plate-to-plate differences, ensuring comparability of all measurements throughout the assay. Well Correction is a systematic error correction method designed to eliminate error that affects compounds located at the same well location (row and column) within the plates across the assay. Most of the error correction and normalization methods have one serious drawback - if they are applied on data that is not affected by any systematic error, they may alter unnecessarily the data causing effect similar to the effect of the systematic error itself and thus impeding the hit selection process. Matrix Error Amendment (MEA) and Partial Mean Polish (PMP) are two novel methods designed to modify only those rows and columns in each plate that are affected by systematic error while preserving their

comparability with the remaining unmodified measurements. In order to determine the location of the systematic error *HTS Helper* uses methodology developed earlier by its author and based on the well-known t-test used in statistics.

*HTS Helper* utility is available both as a Windows Form application and as a Command Line (Console) application. It was developed using Microsoft .NET Framework version 4. The source code of *HTS Helper* utility is also freely available.

## A.2 Using HTS Helper Utility

By design, *HTS Helper* completes its work in three steps. First, it reads an HTS dataset from the input data file. Second, it applies the selected data processing method, if any. And finally, it saves the modified dataset into the output data file. Several parameters are available to control the whole process:

- What action *HTS Helper* should perform on the HTS dataset can be specified using the first drop down list in the Windows Forms Application (WFA) or using `-a` parameter in the Command Line Application (CLA). *HTS Helper* can apply *Matrix Error Amendment (MEA)*, *Partial Mean Polish (PMP)*, *Well Correction (WC)*, *B-score* and *Z-score* methods. Two additional composite actions are also supported, in which, MEA and PMP methods are applied on the data after it has been first treated using the WC method. A special extra option is available that instructs *HTS Helper* utility not to alter the data in any way, ensuring that it will be saved unchanged to the output file. This option can be used for converting HTS data from one format to another or for changing measurements' numeric precision, while preserving the same data format.
- When MEA or PMP method is to be performed (alone or in combination with WC) *HTS Helper* uses t-test to estimate the location of the systematic error within each plate. The required t-test probability level can be specified using the correspondent field in WFA or `-tta` parameter in CLA. The probability level should be a number between 0.0001 and 0.9999.

- It is mandatory to provide the name of the input file containing the HTS data. In WFA that can be done by entering the name in the corresponding field or by pressing the [...] button and selecting the file using the standard Windows 'Open File' dialog. In CLA the input file is specified using the `-i` parameter.
- *HTS Helper* supports 4 different input file formats. It determines the input data format based on the file extension. In case that the data is stored in file with non-standard file extension then the input file format can be explicitly specified. In CLA the `-iff` parameter should be used. Note also that *HTS Helper* can distinguish between the two .csv file formats based on the file content, therefore this option is rarely used.
- In case that the input file is in CSV data format *HTS Helper* offers two ways for importing the data: first, by explicitly specifying the number of rows per plate and second, in the case that plates are separated by empty lines in the input file *HTS Helper* can automatically determine the number of rows per plate. In CLA the number of rows per plate can be explicitly specified with the `-rows` parameter. Note that if the number of rows is explicitly specified then all empty lines in the input file are ignored.
- Specifying an output file name is optional. In CLA, that can be done using the `-o` parameter. If no output file name is specified *HTS Helper* saves the data in the same folder as the input data file. An output file name is automatically generated from the input file name by adding a short suffix corresponding to the selected action.
- *HTS Helper* can save data in 5 different output file formats. The desired file format can be selected in the corresponding drop down list in WFA or in CLA by using `-off` parameter. If no output file format is specified *HTS Helper* will save the result dataset in the same file format as the input data.

- *HTS Helper* allows the user to control the numeric precision of the output HTS data. In CLA it is specified by using the `-pre` parameter. The default numeric precision is 2 digits after the decimal point.

### A.3 HTS Helper Command Line Interface

#### *Syntax*

```
HTSHelper -a ACTION -i INPUT_FILE [-o OUTPUT_FILE] [other parameters ...]
```

#### *Parameters*

Parameter	Description
-a ACTION	Specifies what action should be performed on the HTS data -a 0: Convert data from one format to another, do not modify the data -a 1: <i>B-score</i> normalization method -a 2: <i>Well Correction (WC)</i> method -a 3: <i>Matrix Error Amendment (MEA)</i> method with T-test systematic error detection -a 4: <i>Partial Mean Polish (PMP)</i> method with T-test systematic error detection -a 5: <i>Well Correction (WC)</i> method followed by <i>Matrix Error Amendment (MEA)</i> method with T-test systematic error detection -a 6: <i>Well Correction (WC)</i> method followed by <i>Partial Mean Polish (PMP)</i> with T-test systematic error detection -a 7: <i>Z-score</i> normalization method
-tta ALPHA	Specifies the probability level of the T-test systematic error detection. ALPHA should be a number between 0.00001 and 0.99999. It can be used with -a 3, -a 4, -a 5 and -a 6.
-i INPUT_FILE	Specifies the name of the input data file
-iff INPUT_FORMAT	Specifies what is the format of the input data file -iff 2: .mtr file format, a tabbed value format used in HTS Corrector software, version 5+. -iff 3: .mtx file format, a XML-like format used in HTS Corrector software, version 5+. -iff 4: .csv data file format, every line in the input file represents a row of HTS data. The measurement values are comma separated.

	-iff 5: .csv database file format, every line in the input file represents the <i>Value</i> of a single compound measurement and its location in the assay ( <i>Plate, Row, Column</i> ).
-rows N	Defines the number of rows per plate. It can be used with the -iff 4 option.
-o OUTPUT_FILE	Specifies the name of the output data file
-off OUTPUT_FORMAT	Specifies the format of the output data file -off 2: .mtr file format, a tabbed value format used in HTS Corrector software. -off 3: .mtx file format, a XML-like format used in HTS Corrector software, version 5+. -off 4: .csv data file format, every row of HTS data is saved as a separate line of comma separated values in the output file. -off 5: .csv database file format, every compound measurement is saved in a separate record/line with attributes <i>Plate, Row, Column, Value</i> . -off 6: .html Web Page. Every plate is saved as a separate HTML table.
-pre P	Specifies the precision with which the measurement values are saved in the output file. P defines the number of digits after the decimal point. P should be a number between 1 and 15.

**Table A.1 HTSHelper Command Line Parameters Description**

## APPENDIX B

### SOURCE CODE OF THE HTS HELPER UTILITY (VER 1.0)

#### Program.cs

```
// File: Program.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

namespace HTSHelper
{
    class Program
    {
        /// <summary>
        /// HTSHelper program entry point
        /// </summary>
        /// <param name="args">Command Line Parameters</param>
        static void Main(string[] args)
        {
            UserInterface UI = new CommandLine(args);

            UI.Run();
        }
    }
}
```

#### UserInterface.cs

```
// File: UserInterface.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

using System;
using System.Collections.Generic;

namespace HTSHelper
{
    /// <summary>
    /// An abstract class that defines how HTSHelper interacts with the user interface.
    /// Base class for both the command line and Windows GUI interface.
    /// </summary>
    abstract class UserInterface
```

```

{
    /// <summary>
    /// This method is called to start the User Interface from Program's entry point
    /// </summary>
    public abstract void Run();

    /// <summary>
    /// This method is called by HTSHelper when the execution should be aborted
    /// </summary>
    public abstract void Abort();

    /// <summary>
    /// This method is called by HTSHelper when some text needs to be added
    /// to the execution log.
    /// </summary>
    /// <param name="Text">Text to be written to the log</param>
    public abstract void LogWrite(string Text);

    /// <summary>
    /// This method is called by HTSHelper when some text needs to be added to the
    /// execution followed by a new line.
    /// </summary>
    /// <param name="Text">Text to be written to the log</param>
    public abstract void LogWriteLine(string Text);

    /// <summary>
    /// This method is called by HTSHelper when a warning message should be shown
    /// to the user.
    /// </summary>
    /// <param name="Text">The text of the warning message</param>
    /// <returns>Returns false if the execution should continue and true if the execution
    /// should be aborted</returns>
    public abstract bool Warning(string Text);

    /// <summary>
    /// This method is called by HTSHelper when an error message should be shown to the user.
    /// </summary>
    /// <param name="Text">The text of the error message</param>
    public abstract void Error(string Text);
}
}

```

## CommandLine.cs

```

// File: CommandLine.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//
using System;

namespace HTSHelper
{
    /// <summary>
    /// Implements a simple command line user interface
    /// </summary>
    class CommandLine : UserInterface
    {
        // data fields
        protected string[] Args;

        // temporary variables used to parse the parameters
        protected string ActionStr;
        protected string IFormatStr;
        protected string OFormatStr;
        protected string RowsStr;
        protected string PrecisionStr;
        protected string AlphaStr;
    }
}

```

```

// variables used to store the parameter values
protected int Action;
protected string InputFileName;
protected string OutputFileName;
protected int IFormat;
protected int OFormat;
protected int Rows;
protected int Precision;
protected double Alpha;

// An instance of the main worker object
protected HTSHelper HTS;

/// <summary>
/// Constructor of the User interface object.
/// </summary>
/// <param name="args">Command Line Parameters</param>
public CommandLine(string[] args)
{
    Args = args;
    Action = -1;
    IFormat = -1;
    OFormat = -1;
    Rows = -1;
    Precision = -1;
    Alpha = -1.0;
}

/// <summary>
/// This method is called to start the User Interface from Program's entry point
/// </summary>
public override void Run()
{
    PrintPrompt();

    if (ParseParameters() && ValidateParameters())
    {
        ExecuteAction();
    }
    else
    {
        Console.Out.WriteLine();
        PrintUsage();
    }
}

/// <summary>
/// This method is called by HTSHelper if the execution should be aborted
/// </summary>
public override void Abort()
{
    Environment.Exit(3);
}

/// <summary>
/// This method is called by HTSHelper when some text needs to be added to
/// the execution log.
/// </summary>
/// <param name="Text">Text to be writteh to the log</param>
public override void LogWrite(string Text)
{
    Console.Out.Write(Text);
}

/// <summary>
/// This method is called by HTSHelper when a text line needs to be added to
/// the execution log.
/// </summary>
/// <param name="Text">Text to be writteh to the log as a separate line</param>
public override void LogWriteLine(string Text)
{
    Console.Out.WriteLine(Text);
}

```

```

/// <summary>
/// This method is called by HTSHelper when a warning message should be shown
/// to the user.
/// </summary>
/// <param name="Text">The text of the warning message</param>
/// <returns>Returns false if the execution should continue and true if the
/// execution should be aborted</returns>
public override bool Warning(string Text)
{
    Console.Out.WriteLine("WARNING: " + Text);
    return false;
}

/// <summary>
/// This method is called by HTSHelper when an error message should be shown
/// to the user.
/// </summary>
/// <param name="Text">The text of the error message</param>
public override void Error(string Text)
{
    Console.Error.WriteLine("ERROR: " + Text);
}

/// <summary>
/// Prints a welcome/about message at program's start
/// </summary>
private void PrintPrompt()
{
    Console.Out.WriteLine("HTSHelper utility. Version 1.0 / Mar 14, 2012." +
        " Written by Plamen Dragiev.");
    Console.Out.WriteLine();
}

/// <summary>
/// Prints short on screen instructions how to use the program.
/// </summary>
private void PrintUsage()
{
    Console.Out.WriteLine("USAGE:");
    Console.Out.WriteLine("HTSHelper -a ACTION -i INPUT_FILE [-o OUTPUT_FILE] [other
parameters ...]");
    Console.Out.WriteLine(" [-a ACTION] Specifies an action to be performed");
    Console.Out.WriteLine(" -a 0: None, convert data from one format to another.");
    Console.Out.WriteLine(" -a 1: BScore method.");
    Console.Out.WriteLine(" -a 2: Well Correction method.");
    Console.Out.WriteLine(" -a 3: T-test + Matrix Error Amendment method.");
    Console.Out.WriteLine(" -a 4: T-test + Partial Mean Polish method.");
    Console.Out.WriteLine(" -a 5: Well Correction + T-test + Matrix Error Amendment
method.");
    Console.Out.WriteLine(" -a 6: Well Correction + T-test + Partial Mean Polish
method.");
    Console.Out.WriteLine(" -a 7: Z-score normalization method.");
    Console.Out.WriteLine(" [-i INPUT_FILE] Specifies the name of the input file.");
    Console.Out.WriteLine(" [-o OUTPUT_FILE] Specifies the name of the output file.");
    Console.Out.WriteLine(" [-iff FILE_FORMAT] Specifies the format of the input file.");
    // .asy format is not supported in ver. 1.0
    // Console.Out.WriteLine(" -iff 1: .asy file format.");
    Console.Out.WriteLine(" -iff 2: .mtr file format.");
    Console.Out.WriteLine(" -iff 3: .mtx file format.");
    Console.Out.WriteLine(" -iff 4: .csv data file format.");
    Console.Out.WriteLine(" -iff 5: .csv database file format.");
    Console.Out.WriteLine(" [-off FILE_FORMAT] Specifies the format of the output
file.");
    // .asy format is not supported in ver. 1.0
    // Console.Out.WriteLine(" -off 1: .asy file format.");
    Console.Out.WriteLine(" -off 2: .mtr file format.");
    Console.Out.WriteLine(" -off 3: .mtx file format.");
    Console.Out.WriteLine(" -off 4: .csv data file format.");
    Console.Out.WriteLine(" -off 5: .csv database file format.");
    Console.Out.WriteLine(" -off 6: .html file format.");
    Console.Out.WriteLine(" [-rows N] Specifies the number of rows per plate. Used
with -iff 4 option.");
    Console.Out.WriteLine(" [-pre N] Specifies the real number precision. 1 < N < 16.");
    Console.Out.WriteLine(" [-tta X] Specifies T-test probability level, actions

```

```

3,4,5 or 6. 0.00001<=X<1");
    Console.Out.WriteLine();
}

/// <summary>
/// Parses the command line parameters. Checks for if the input follows the basic syntax
/// rules, like for example if parameter -a is followed by an action code and etc. It
/// does validate the values of the provided parameters, only if they are present or not.
/// </summary>
/// <returns>Returns false if a syntax error was detected and true otherwise</returns>
private bool ParseParameters()
{
    int i = 0;
    while (i < Args.Length)
    {
        if (Args[i] == "-a")
        {
            i++;
            if (i == Args.Length || Args[i].StartsWith("-"))
            {
                Error("Please, specify an action!");
                return false;
            }
            ActionStr = Args[i];
        }
        else if (Args[i] == "-i")
        {
            i++;
            if (i == Args.Length || Args[i].StartsWith("-"))
            {
                Error("Please, specify an input file!");
                return false;
            }
            InputFileName = Args[i];
        }
        else if (Args[i] == "-o")
        {
            i++;
            if (i == Args.Length || Args[i].StartsWith("-"))
            {
                Error("Please, specify an output file!");
                return false;
            }
            OutputFileName = Args[i];
        }
        else if (Args[i] == "-iff")
        {
            i++;
            if (i == Args.Length || Args[i].StartsWith("-"))
            {
                Error("Please, specify an input file format!");
                return false;
            }
            IFormatStr = Args[i];
        }
        else if (Args[i] == "-off")
        {
            i++;
            if (i == Args.Length || Args[i].StartsWith("-"))
            {
                Error("Please, specify an output file format!");
                return false;
            }
            OFormatStr = Args[i];
        }
        else if (Args[i] == "-rows")
        {
            i++;
            if (i == Args.Length || Args[i].StartsWith("-"))
            {
                Error("Please, specify the number of rows per plate!");
                return false;
            }
            RowsStr = Args[i];
        }
    }
}

```

```

    }
    else if (Args[i] == "-pre")
    {
        i++;
        if (i == Args.Length || Args[i].StartsWith("-"))
        {
            Error("Please, specify the output precision!");
            return false;
        }
        PrecisionStr = Args[i];
    }
    else if (Args[i] == "-tta")
    {
        i++;
        if (i == Args.Length || Args[i].StartsWith("-"))
        {
            Error("Please, specify T-test probability level!");
            return false;
        }
        AlphaStr = Args[i];
    }
    else
    {
        Warning("Unknown/ignored parameter: '" + Args[i] + "'.");
    }
    i++;
}
return true;
}

/// <summary>
/// Validates the correctness of the provided command line parameters or if a mandatory
/// parameter is omitted.
/// </summary>
/// <returns>Returns false if an incorrect parameter value is detected and true
otherwise</returns>

private bool ValidateParameters()
{
    if (ActionStr == null)
    {
        Error("Please, specify an action!");
        return false;
    }
    else
    {
        Action = G.ParseInt(ActionStr, -1);
        if (Action < 0 || Action > 7)
        {
            Error("Bad action code: '" + ActionStr + "'.");
            return false;
        }
        else
        {
            if (InputFileName == null)
            {
                Error("Please, specify an input data file!");
                return false;
            }
            else
            {
                if (IFormatStr != null)
                {
                    IFormat = G.ParseInt(IFormatStr);
                    if (IFormat < 1 || IFormat > 5)
                    {
                        Error("Bad input file format: '" + IFormatStr + "'.");
                        return false;
                    }
                }
            }
            if (Action == 0 && OFormatStr == null)
            {

```

```

        Error("You should specify an output file format.");
        return false;
    }
    else if (OFormatStr != null)
    {
        OFormat = G.ParseInt(OFormatStr);
        if (OFormat < 1 || OFormat > 6)
        {
            Error("Bad output file format: " + OFormatStr + ".");
            return false;
        }
    }
    if (PrecisionStr != null)
    {
        Precision = G.ParseInt(PrecisionStr, -1);
        if (Precision < 1 || Precision > 15)
        {
            Error("Bad numeric precision specified: " + PrecisionStr +
                ". A number between 1 and 15 is expected.");
            return false;
        }
    }
    if (RowsStr != null)
    {
        Rows = G.ParseInt(RowsStr, -1);
        if (Rows < 4 || Rows > 1000)
        {
            Error("Bad number or rows per plate specified: " + RowsStr +
                ". A number between 4 and 1000 is expected.");
            return false;
        }
    }
    if (AlphaStr != null)
    {
        Alpha = G.ParseDouble(AlphaStr, -1);
        if (Alpha < 0.00001 || Alpha >= 1.0)
        {
            Error("Bad number T-test probability level specified: " +
                AlphaStr + ". A number between 0.00001 and 1.0 is expected.");
            return false;
        }
    }
    return true;
}
}
}

/// <summary>
/// Creates a worker instance and passes the control to it in order to execute the
/// requested action. Only the parameters explicitly specified on the command line are
/// passed to the worker object If an option is not defined, the default value defined
/// in the worker object (HTSHelper) is used.
/// </summary>
private void ExecuteAction()
{
    HTS = new HTSHelper();

    HTS.UI = this;

    if (InputFileName == null) InputFileName = "";
    HTS.InputFileName = InputFileName;

    if (OutputFileName == null) OutputFileName = "";
    HTS.OutputFileName = OutputFileName;

    if (IFormat > -1)
    {
        FileFormats InputFormat = (FileFormats)IFormat;
        HTS.InputFormat = InputFormat;
    }
}

```

```

        if (OFormat > -1)
        {
            FileFormats OutputFormat = (FileFormats)OFormat;
            HTS.OutputFormat = OutputFormat;
        }

        if (Rows>-1) HTS.PlateRows = Rows;
        if (Precision > -1) HTS.Precision = Precision;
        if (Alpha > 0.0) Plate.TTest_Alpha = Alpha;

        Actions Act = (Actions)Action;

        bool Res = HTS.Execute(Act);

        if (Res)
            Console.WriteLine("Action completed.");
        Console.WriteLine("Bye.");
    }
}

```

## HTSData.cs

```

// File: HTSData.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

using System;
using System.Collections.Generic;

namespace HTSHelper
{
    /// <summary>
    /// An abstract class representing a set of HTS measurements
    /// </summary>
    abstract class HTSData
    {
        protected const double ZSCORE_EPSILON = 0.000001;

        /// <summary>
        /// Returns the number of items in the set of HTS measurements
        /// </summary>
        public abstract int NumItems { get; }

        /// <summary>
        /// Enumerates all HTS measurements
        /// </summary>
        /// <returns>An enumerable object for all HTS measurements</returns>
        public abstract IEnumerable<double> AllItems();

        /// <summary>
        /// Calculates the sum of all HTS measurements
        /// </summary>
        /// <returns>Returns the sum of all HTS measurements</returns>
        public virtual double CalcSum()
        {
            double Sum = 0.0;
            foreach (double X in AllItems()) Sum += X;
            return Sum;
        }

        /// <summary>
        /// Calculates the mean value of all measurements
        /// </summary>
    }
}

```

```

/// <returns>The mean value of all measurements</returns>
public virtual double CalcMean()
{
    if (NumItems > 0)
        return CalcSum() / NumItems;
    return 0.0;
}

/// <summary>
/// Calculates the variance of all measurements
/// </summary>
/// <param name="Mean">The mean of all measurements</param>
/// <returns>Returns the variance of all measurements</returns>
public virtual double CalcVariance(double Mean)
{
    double Var = 0.0, Diff;
    foreach (double X in AllItems())
    {
        Diff = X - Mean;
        Var += Diff * Diff;
    }
    return Var;
}

/// <summary>
/// Calculates the standard deviation of all measurements
/// </summary>
/// <returns>Returns the standard deviation of all measurements</returns>
public virtual double CalcSD()
{
    if (NumItems > 1)
        return Math.Sqrt(CalcVariance(CalcMean()) / (NumItems - 1)); ;
    return 0.0;
}

/// <summary>
/// Calculates the mean and the standard deviation of all measurements
/// </summary>
/// <param name="Mean">On return it contains the mean of all measurements</param>
/// <param name="SD">on return it contains the standard deviation of all measurements
</param>
public virtual void CalcMeanAndSD(out double Mean, out double SD)
{
    Mean = CalcMean();
    if (NumItems > 1)
        SD = Math.Sqrt(CalcVariance(Mean) / (NumItems - 1));
    else
        SD = 0.0;
}

/// <summary>
/// Calculates the sum of an enumerable set of measurements
/// </summary>
/// <param name="Set">The enumerable set of measurements</param>
/// <returns>Returns the sum of all measurements</returns>
public static double CalcSum(IEnumerable<double> Set)
{
    double Sum = 0.0;
    foreach (double X in Set) Sum += X;
    return Sum;
}

/// <summary>
/// Calculates the mean value of an enumerable set of measurements
/// </summary>
/// <param name="Set">The enumerable set of measurements</param>
/// <returns>Returns the mean of all measurements</returns>
public static double CalcMean(IEnumerable<double> Set)
{
    double Sum = 0.0;
    int Count = 0;
    foreach (double X in Set)
    {
        Count++;
    }
}

```

```

        Sum += X;
    }
    return Count>0? Sum / Count : 0.0;
}

/// <summary>
/// Calculates the variance of an enumerable set of measurements
/// </summary>
/// <param name="Mean">The mean of measurements</param>
/// <param name="Set">The enumerable set of measurements</param>
/// <returns>Returns the variance of all measurements</returns>
public static double CalcVariance(IEnumerable<double> Set, double Mean)
{
    double Var = 0.0, Diff;
    foreach (double X in Set)
    {
        Diff = X - Mean;
        Var += Diff * Diff;
    }
    return Var;
}

/// <summary>
/// Calculates the standard deviation of an enumerable set of measurements
/// </summary>
/// <param name="Mean">The mean of all measurements</param>
/// <param name="Set">The enumerable set of measurements</param>
/// <returns>Returns the standard deviation of all measurements</returns>
public static double CalcSD(IEnumerable<double> Set, double Mean)
{
    double Var = 0.0, Diff;
    int Count = 0;
    foreach (double X in Set)
    {
        Count++;
        Diff = X - Mean;
        Var += Diff * Diff;
    }
    if (Count > 1)
        return Math.Sqrt(Var / (Count - 1)); ;
    return 0.0;
}

/// <summary>
/// Calculates the mean and the standard deviation of an enumerable set of measurements
/// </summary>
/// <param name="Mean">On return - the mean of all measurements</param>
/// <param name="SD">On return - the standard deviation of all measurements</param>
/// <param name="Set">The enumerable set of measurements</param>
public static void CalcMeanAndSD(IEnumerable<double> Set, out double Mean, out double SD)
{
    Mean = CalcMean(Set);
    SD = CalcSD(Set, Mean);
}

/// <summary>
/// Adds X to all measurements
/// </summary>
/// <param name="X">A value to be added to all measurements</param>
public abstract void Add(double X);

/// <summary>
/// Subtracts X from all measurements
/// </summary>
/// <param name="X">A value to be subtracted from all measurements</param>
public virtual void Sub(double X)
{
    Add(-X);
}

/// <summary>
/// Multiplied all measurements by X
/// </summary>

```

```

public abstract void Mul(double X);

/// <summary>
/// Divides all measurements by X
/// </summary>
public virtual void Div(double X)
{
    Mul(1.0 / X);
}

/// <summary>
/// For all measurements M, M:= (M-A)/X
/// </summary>
public virtual void SubDiv(double A, double X)
{
    Sub(A);
    Div(X);
}

/// <summary>
/// Applies Z-score method to the measurements
/// </summary>
public virtual void Zscore()
{
    double Mean, SD;

    CalcMeanAndSD(out Mean, out SD);

    if (SD > ZSCORE_EPSILON) SubDiv(Mean, SD);
    else Sub(Mean);
}
}
}
}

```

## Plate.cs

```

// File: Plate.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//
using System;
using System.Collections.Generic;

namespace HTSHelper
{
    /// <summary>
    /// A class representing a HTS plate of compounds
    /// </summary>
    class Plate : HTSData
    {
        // Dimensions of the plate
        protected int Rows;
        protected int Columns;

        // Compound measurements for every well of the plate
        protected double[,] Wells;

        // BScore constants
        protected double BSCORE_EPSILON = 0.00005;
        protected double BSCORE_EPS_PERC = 0.0001;
        protected int BSCORE_MAX_ITERATIONS = 20;

        // PMP constants
        protected double PMP_EPSILON = 0.01;
        protected int PMP_MAX_ITERATIONS = 50;

        // T-test constants
    }
}

```

```

public static double TTest_Alpha = 0.1;

/// <summary>
/// Default Constructor
/// </summary>
protected Plate()
{
    // Empty
}

/// <summary>
/// Constructor
/// </summary>
public Plate(int Rows, int Columns)
{
    this.Rows = Rows;
    this.Columns = Columns;
    Wells = new double[Rows, Columns];
}

/// <summary>
/// Well indexer
/// </summary>
public double this[int Row, int Column]
{
    get
    {
        return Wells[Row, Column];
    }
    set
    {
        Wells[Row, Column] = value;
    }
}

/// <summary>
/// Returns the number of items/wells on the plate
/// </summary>
public override int NumItems
{
    get { return Rows * Columns; }
}

/// <summary>
/// Enumerates all items on the plate
/// </summary>
public override IEnumerable<double> AllItems()
{
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            yield return Wells[i, j];
        }
    }
}

/// <summary>
/// Overwritten only for speed optimization
/// </summary>
/// <returns>Returns the sum of all items in the plate</returns>
public override double CalcSum()
{
    double Sum = 0.0;
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            Sum += Wells[i, j];
        }
    }
    return Sum;
}

```

```

/// <summary>
/// Overwritten only for speed optimization
/// </summary>
/// <returns>Returns the variance of all items in the plate</returns>
public override double CalcVariance(double Mean)
{
    double Var = 0.0;
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            double Diff = Wells[i, j] - Mean;
            Var += Diff * Diff;
        }
    }
    return Var;
}

/// <summary>
/// Returns the number of rows of the plate
/// </summary>
public int NumRows
{
    get { return Rows; }
}

/// <summary>
/// Returns the number of columns of the plate
/// </summary>
public int NumColumns
{
    get { return Columns; }
}

/// <summary>
/// Enumerates all items/compounds in a given row of the plate
/// </summary>
public IEnumerable<double> RowItems(int Row)
{
    for (int j = 0; j < Columns; j++)
    {
        yield return Wells[Row, j];
    }
}

/// <summary>
/// Enumerates all items/compounds in a given column of the plate
/// </summary>
public IEnumerable<double> ColumnItems(int Column)
{
    for (int i = 0; i < Rows; i++)
    {
        yield return Wells[i, Column];
    }
}

/// <summary>
/// Enumerates all items/compounds on the plate except the ones located in a given row
/// of the plate
/// </summary>
public IEnumerable<double> NotRowItems(int Row)
{
    for (int i = 0; i < Rows; i++)
    {
        if (i == Row) continue;
        for (int j = 0; j < Columns; j++)
        {
            yield return Wells[i, j];
        }
    }
}

/// <summary>
/// Enumerates all items/compounds on the plate except the ones located in a given column

```

```

/// of the plate
/// </summary>
public IEnumerable<double> NotColumnItems(int Column)
{
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            if (j == Column) continue;
            yield return Wells[i, j];
        }
    }
}

/// <summary>
/// Adds X to all measurements in the plate
/// </summary>
public override void Add(double X)
{
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            Wells[i, j] += X;
        }
    }
}

/// <summary>
/// Multiplies by X all measurements on the plate
/// </summary>
public override void Mul(double X)
{
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            Wells[i, j] *= X;
        }
    }
}

/// <summary>
/// Divides by X all measurements on the plate
/// </summary>
public override void Div(double X)
{
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            Wells[i, j] /= X;
        }
    }
}

/// <summary>
/// Replaces all measurements M on the plate with (M - A) / X
/// </summary>
public override void SubDiv(double A, double X)
{
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            Wells[i, j] = (Wells[i, j] - A) / X;
        }
    }
}

/// <summary>
/// Copies the measurements in row N of the plate to an array
/// </summary>
public double[] RowValues(int N)

```

```

{
    double[] R = new double[Columns];
    for (int j = 0; j < Columns; j++) R[j] = Wells[N, j];
    return R;
}

/// <summary>
/// Copies the measurements in column N of the plate to an array
/// </summary>
public double[] ColumnValues(int N)
{
    double[] C = new double[Rows];
    for (int i = 0; i < Rows; i++) C[i] = Wells[i, N];
    return C;
}

/// <summary>
/// Applies B-score method to the measurements on the plate
/// </summary>
public void Bscore()
{
    double[] MRow = new double[Rows];
    double[] MCol = new double[Columns];

    double[] R = new double[Rows];
    double[] C = new double[Columns];

    Array.Clear(R, 0, Rows);
    Array.Clear(C, 0, Columns);

    int i, j, k;
    int Iteration = BSCORE_MAX_ITERATIONS;
    double OldSum = 0.0;
    bool converge = false;

    do
    {
        // Rows
        for (i = 0; i < Rows; i++)
            R[i] += MRow[i] = G.Median(RowValues(i), true);

        for (i = 0; i < Rows; i++)
            for (j = 0; j < Columns; j++)
                Wells[i, j] -= MRow[i];

        double RMed = G.Median(MRow);
        for (i = 0; i < Rows; i++) R[i] -= RMed;

        // Columns
        for (j = 0; j < Columns; j++)
            C[j] += MCol[j] = G.Median(ColumnValues(j), true);

        double WellSum = 0.0;
        for (i = 0; i < Rows; i++)
            for (j = 0; j < Columns; j++)
            {
                Wells[i, j] -= MCol[j];
                WellSum += Math.Abs(Wells[i, j]);
            }
        double CMed = G.Median(MCol);
        for (j = 0; j < Columns; j++) C[j] -= CMed;

        converge = WellSum < BSCORE_EPSILON || Math.Abs(WellSum - OldSum) <
BSCORE_EPS_PERC * WellSum;
        OldSum = WellSum;
    } while (--Iteration > 0 && !converge);

    double[] Resid = new double[Rows * Columns];
    for (k = i = 0; i < Rows; i++)
        for (j = 0; j < Columns; j++)
            Resid[k++] = Wells[i, j];

    double ResMed = G.Median(Resid, true);
    for (i = 0; i < Resid.Length; i++) Resid[i] = Math.Abs(Resid[i] - ResMed);
}

```

```

double MAD = G.Median(Resid, true);

// the following line's been added for compatibility with HTS Corrector
// see Makarenkov V, Zentilli P, Kevorkov D, Gagarin A, Malo N and Nadon R:
// An efficient method for the detection and elimination of systematic error
// in high-throughput screening. Bioinformatics 2007, 23:1648-1657
MAD *= 1.4826;

if (MAD > 0.0001)
    for (i = 0; i < Rows; i++)
        for (j = 0; j < Columns; j++)
            Wells[i, j] /= MAD;
}

/// <summary>
/// Applies Matrix Error Amendment (MEA) method to the measurements on the plate
/// </summary>
public void MEA(List<int> ERows, List<int> EColumns)
{
    int NR = ERows!=null? ERows.Count : 0;
    int NC = EColumns!=null? EColumns.Count : 0;
    int N = NR + NC;

    // Is there any row column affected by systematic error?
    if (N == 0) return;

    bool[] RFlag = new bool[Rows];
    bool[] CFlag = new bool[Columns];

    Array.Clear(RFlag, 0, Rows);
    Array.Clear(CFlag, 0, Columns);
    foreach (int r in ERows) RFlag[r] = true;
    foreach (int c in EColumns) CFlag[c] = true;

    double Mu = 0.0;

    int i, j;

    for (i = 0; i < Rows; i++)
    {
        if (RFlag[i]) continue;
        for (j = 0; j < Columns; j++)
        {
            if (CFlag[j]) continue;
            Mu += Wells[i, j];
        }
    }

    Mu /= (Rows - NR) * (Columns - NC);

    // Exact Solution ...
    double[] X = new double[N]; // The solution - the first NR values are the
                                // row errors, the rest are the column errors

    double[,] A = new double[N, N]; // A * X = B
    double[] B = new double[N]; //

    for (i = 0; i < NR; i++)
    {
        int r = ERows[i];
        A[i, i] = Columns;
        for (j = NR; j < N; j++) A[i, j] = 1.0;
        B[i] = -Columns * Mu;
        for (int k = 0; k < Columns; k++)
        {
            B[i] += Wells[r, k];
        }
    }

    for (i = NR; i < N; i++)
    {
        int c = EColumns[i - NR];
        A[i, i] = Rows;
    }
}

```

```

        for (j = 0; j < NR; j++) A[i, j] = 1.0;
        B[i] = -Rows * Mu;
        for (int k = 0; k < Rows; k++)
        {
            B[i] += Wells[k, c];
        }
    }

    G.InvertMatrix(A);

    // X = Inv(A) * B, X is the estimated row and column error
    for (i = 0; i < N; i++)
    {
        X[i] = 0.0;
        for (j = 0; j < N; j++)
        {
            X[i] += A[i, j] * B[j];
        }
    }

    // Remove the systematic error from the plate measurements
    for (i = 0; i < NR; i++)
    {
        int r = ERows[i];
        for (j = 0; j < Columns; j++) Wells[r, j] -= X[i];
    }

    for (i = NR; i < N; i++)
    {
        int c = EColumns[i - NR];
        for (j = 0; j < Rows; j++) Wells[j, c] -= X[i];
    }
}

/// <summary>
/// Applies Matrix Error Amendment (MEA) method to the measurements on the plate
/// </summary>
public void MEA()
{
    List<int> ERows;
    List<int> EColumns;

    if (TTest(out ERows, out EColumns))
        MEA(ERows, EColumns);
}

/// <summary>
/// Applies Partial Mean Polish (PMP) method to the measurements on the plate
/// </summary>
public void PMP()
{
    List<int> ERows;
    List<int> EColumns;

    if (TTest(out ERows, out EColumns))
        PMP(ERows, EColumns);
}

/// <summary>
/// Applies Partial Mean Polish (PMP) method to the measurements on the plate
/// </summary>
public void PMP(List<int> ERows, List<int> EColumns)
{
    int NR = ERows != null ? ERows.Count : 0;
    int NC = EColumns != null ? EColumns.Count : 0;
    int N = NR + NC;

    // Is there any row column affected by systematic error?
    if (N == 0) return;

    bool[] RFlag = new bool[Rows];
    bool[] CFlag = new bool[Columns];

```

```

Array.Clear(RFlag, 0, Rows);
Array.Clear(CFlag, 0, Columns);
foreach (int r in ERows) RFlag[r] = true;
foreach (int c in EColumns) CFlag[c] = true;

double Mu = 0.0;

int i, j;

for (i = 0; i < Rows; i++)
{
    if (RFlag[i]) continue;
    for (j = 0; j < Columns; j++)
    {
        if (CFlag[j]) continue;
        Mu += Wells[i, j];
    }
}

Mu /= (Rows - NR) * (Columns - NC);

double[] RMu = new double[Rows];
double[] CMu = new double[Columns];

Array.Clear(RMu, 0, Rows);
Array.Clear(CMu, 0, Columns);

int Loop = 1;
double Converge = 0.0;

do
{
    double Diff;
    Converge = 0.0;
    for (i = 0; i < Rows; i++)
    {
        if (!RFlag[i]) continue;
        for (j = 0; j < Columns; j++)
        {
            RMu[i] += Wells[i, j];
        }
        RMu[i] /= Columns;
    }

    for (j = 0; j < Columns; j++)
    {
        if (!CFlag[j]) continue;
        for (i = 0; i < Rows; i++)
        {
            CMu[j] += Wells[i, j];
        }
        CMu[j] /= Rows;
    }

    for (i = 0; i < Rows; i++)
    {
        if (!RFlag[i]) continue;
        Diff = Mu - RMu[i];
        Converge += Math.Abs(Diff);
        for (j = 0; j < Columns; j++)
        {
            Wells[i, j] += Diff;
        }
    }

    for (j = 0; j < Columns; j++)
    {
        if (!CFlag[j]) continue;
        Diff = Mu - CMu[j];
        Converge += Math.Abs(Diff);
        for (i = 0; i < Rows; i++)
        {
            Wells[i, j] += Diff;
        }
    }
}

```

```

    }
    ] while (Converge > PMP_EPSILON && Loop++ < PMP_MAX_ITERATIONS);
}

/// <summary>
/// Performs T-test for all rows and columns of the plate in ordet to assess
/// if they are affected by systematic error
/// </summary>
public bool TTest(out List<int> ERows, out List<int> EColumns)
{
    ERows = new List<int>();
    EColumns = new List<int>();

    bool Res = false;

    // T-test by rows
    for (int i = 0; i < Rows; i++)
    {
        double RowMean = CalcMean(RowItems(i));
        double RowVar = CalcVariance(RowItems(i), RowMean);

        double RestMean = CalcMean(NotRowItems(i));
        double RestVar = CalcVariance(NotRowItems(i), RestMean);

        double N1 = Columns;
        double N2 = Columns * (Rows - 1);

        double Sp = (RowVar + RestVar) / (N1 + N2 - 2);

        double TStat = Math.Abs((RowMean - RestMean) / Math.Sqrt(Sp * (N1+N2) / (N1*N2)));

        double CV = G.TTestCV(Columns, TTest_Alpha);

        if (TStat > CV)
        {
            // there's error detected by the t-test
            Res = true;
            ERows.Add(i);
        }
    }

    // T-test by Columns
    for (int j = 0; j < Columns; j++)
    {
        double ColMean = CalcMean(ColumnItems(j));
        double ColVar = CalcVariance(ColumnItems(j), ColMean);

        double RestMean = CalcMean(NotColumnItems(j));
        double RestVar = CalcVariance(NotColumnItems(j), RestMean);

        double N1 = Rows;
        double N2 = Rows * (Columns - 1);

        double Sp = (ColVar + RestVar) / (N1 + N2 - 2);

        double TStat = Math.Abs((ColMean - RestMean) / Math.Sqrt(Sp * (N1+N2) / (N1*N2)));

        double CV = G.TTestCV(Rows, TTest_Alpha);

        if (TStat > CV)
        {
            // there's error detected by the test
            Res = true;
            EColumns.Add(j);
        }
    }

    return Res;
}
}

```

## Dataset.cs

```

// File: Dataset.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

using System;
using System.Collections.Generic;

namespace HTSHelper
{
    /// <summary>
    /// A class representing a HTS dataset
    /// </summary>
    class Dataset : HTSData
    {
        // The size of the plates in this dataset
        protected int Rows;
        protected int Columns;

        // The list of plates in the dataset
        protected List<Plate> Plates;

        /// <summary>
        /// Default Constructor
        /// </summary>
        protected Dataset()
        {
            // Empty
        }

        /// <summary>
        /// Creates a dataset with 0 plates
        /// </summary>
        /// <param name="NRows">Number of rows</param>
        /// <param name="NCColumns">Number of columns</param>
        public Dataset(int NRows, int NCColumns)
        {
            Rows = NRows;
            Columns = NCColumns;
            Plates = new List<Plate>();
        }

        /// <summary>
        /// Creates a dataset with <i>NPlates</i> plates
        /// </summary>
        /// <param name="NPlates">Number of plates</param>
        /// <param name="NRows">Number of rows</param>
        /// <param name="NCColumns">Number of columns</param>
        public Dataset(int NPlates, int NRows, int NCColumns)
        {
            Rows = NRows;
            Columns = NCColumns;
            Plates = new List<Plate>(NPlates);
            for (int p = 0; p < NPlates; p++)
            {
                Plates.Add(new Plate(NRows, NCColumns));
            }
        }

        /// <summary>
        /// A plate indexer
        /// </summary>
        public Plate this[int plate]
        {
            get
            {
                return Plates[plate];
            }
        }
    }
}

```

```

        set
        {
            Plates[plate] = value;
        }
    }

    /// <summary>
    /// A well indexer
    /// </summary>
    public double this[int Plate, int Row, int Column]
    {
        get
        {
            return Plates[Plate][Row, Column];
        }
        set
        {
            Plates[Plate][Row, Column] = value;
        }
    }

    /// <summary>
    /// Returns the number of items/wells in the dataset
    /// </summary>
    public override int NumItems
    {
        get { return Plates.Count * Rows * Columns; }
    }

    /// <summary>
    /// Returns the number of rows in the plates of the dataset
    /// </summary>
    public int NumRows
    {
        get { return Rows; }
    }

    /// <summary>
    /// Returns the number of columns in the plates of the dataset
    /// </summary>
    public int NumColumns
    {
        get { return Columns; }
    }

    /// <summary>
    /// Returns the number of plates in the dataset
    /// </summary>
    public int NumPlates
    {
        get { return Plates.Count; }
    }

    /// <summary>
    /// Enumerates all items/wells in the dataset
    /// </summary>
    public override IEnumerable<double> AllItems()
    {
        for (int p = 0; p < Plates.Count; p++)
        {
            Plate PL = Plates[p];
            for (int i = 0; i < Rows; i++)
            {
                for (int j = 0; j < Columns; j++)
                {
                    yield return PL[i, j];
                }
            }
        }
    }

    /// <summary>
    /// Enumerates all items/wells located at a given well location (Row, Columns)
    /// on all plates in the dataset

```

```

/// </summary>
public IEnumerable<double> WellItems(int Row, int Column)
{
    for (int p = 0; p < Plates.Count; p++)
    {
        yield return Plates[p][Row, Column];
    }
}

/// <summary>
/// Enumerates all plates in the dataset
/// </summary>
public IEnumerable<Plate> AllPlates()
{
    for (int p = 0; p < Plates.Count; p++)
    {
        yield return Plates[p];
    }
}

/// <summary>
/// Adds a plate to the dataset
/// </summary>
/// <param name="aPlate">A plate to be added to the dataset</param>
public void AddPlate(Plate aPlate)
{
    Plates.Add(aPlate);
}

/// <summary>
/// Adds an empty plate to the dataset and returns a reference to it
/// </summary>
/// <returns>A reference to the newly added plate</returns>
public Plate AddPlate()
{
    Plate PL = new Plate(Rows, Columns);
    Plates.Add(PL);
    return PL;
}

/// <summary>
/// Returns a reference to the last plate in the dataset
/// </summary>
/// <returns>A reference to the last plate of the dataset</returns>
public Plate LastPlate()
{
    if (Plates.Count > 0)
        return Plates[Plates.Count - 1];
    return null;
}

/// <summary>
/// Overwritten only for speed optimization
/// </summary>
/// <returns>Returns the sum of all items in the plate</returns>
public override double CalcSum()
{
    double Sum = 0.0;
    foreach (Plate PL in Plates) Sum += PL.CalcSum();
    return Sum;
}

/// <summary>
/// Overwritten only for speed optimization
/// </summary>
/// <returns>Returns the variance of all items in the plate</returns>
public override double CalcVariance(double Mean)
{
    double Var = 0.0;
    foreach (Plate PL in Plates) Var += PL.CalcVariance(Mean);
    return Var;
}

/// <summary>

```

```

/// Adds X to all measurements in the dataset
/// </summary>
public override void Add(double X)
{
    foreach (Plate PL in Plates) PL.Add(X);
}

/// <summary>
/// Multiplies all measurements in the dataset by X
/// </summary>
public override void Mul(double X)
{
    foreach (Plate PL in Plates) PL.Mul(X);
}

/// <summary>
/// Divides all measurements in the dataset by X
/// </summary>
public override void Div(double X)
{
    foreach (Plate PL in Plates) PL.Div(X);
}

/// <summary>
/// For all measurements M in the dataset it subtracts first A and the result divides
/// by X, i.e.,  $M := (M - A) / X$ 
/// </summary>
public override void SubDiv(double A, double X)
{
    foreach (Plate PL in Plates) PL.SubDiv(A, X);
}

/// <summary>
/// Applies Z-score method on the dataset, i.e. it applies it on each plate
/// </summary>
public override void Zscore()
{
    foreach (Plate PL in Plates) PL.Zscore();
}

/// <summary>
/// Applies B-score method on the dataset, i.e. it applies it on each plate
/// </summary>
public void Bscore()
{
    foreach (Plate PL in Plates) PL.Bscore();
}

/// <summary>
/// Applies Well Correction method on the dataset
/// </summary>
public void WellCorrection()
{
    for (int i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            WellData WData = new WellData(this, i, j);
            WData.Zscore();
        }
    }
}

/// <summary>
/// Applies Matrix Error Amendment (MEA) method on the dataset,
/// i.e., it applies it on each plate
/// </summary>
public void MEA()
{
    foreach (Plate PL in Plates) PL.MEA();
}

/// <summary>
/// Applies Partial Mean Polish (PMP) method on the dataset,

```

```

    /// i.e., it applies it on each plate
    /// </summary>
    public void PMP()
    {
        foreach (Plate PL in Plates) PL.PMP();
    }
}
}
}

```

## WellData.cs

```

// File: Plate.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

using System;
using System.Collections.Generic;

namespace HTSHelper
{
    /// <summary>
    /// A class used to access and modify data located at a specific well location
    /// across the plates of an dataset
    /// </summary>
    class WellData : HTSData
    {
        // The dataset and the well location
        protected Dataset Data;
        protected int Row;
        protected int Column;

        /// <summary>
        /// Constructor
        /// </summary>
        public WellData(Dataset DSet, int aRow, int aColumn)
        {
            Data = DSet;
            Row = aRow;
            Column = aColumn;
        }

        /// <summary>
        /// Returns the number of items in this set (one on every plate)
        /// </summary>
        public override int NumItems
        {
            get { return Data.NumPlates; }
        }

        /// <summary>
        /// Enumerates all items/compounds in this set
        /// </summary>
        public override IEnumerable<double> AllItems()
        {
            for (int p = 0; p < Data.NumPlates; p++)
            {
                yield return Data[p, Row, Column];
            }
        }

        /// <summary>
        /// Adds X to all measurements in the set
        /// </summary>
        public override void Add(double X)
        {
            for (int p = 0; p < Data.NumPlates; p++)
            {

```

```

        Data[p, Row, Column] += X;
    }
}

/// <summary>
/// Multiplies all items by X
/// </summary>
public override void Mul(double X)
{
    for (int p = 0; p < Data.NumPlates; p++)
    {
        Data[p, Row, Column] *= X;
    }
}

/// <summary>
/// Replaces all measurements M with (M - A) / X
/// </summary>
public override void SubDiv(double A, double X)
{
    for (int p = 0; p < Data.NumPlates; p++)
    {
        Data[p, Row, Column] = (Data[p, Row, Column] - A) / X;
    }
}
}
}
}

```

## G.cs

```

// File: G.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

using System;

namespace HTSHelper
{
    /// <summary>
    /// Global utility methods and constants. Static methods only.
    /// </summary>
    class G
    {
        /// <summary>
        /// Converts a string to an integer value. Returns -1 if there is a syntax error.
        /// </summary>
        /// <param name="Str">String representation of an integer number</param>
        /// <returns>Returns the integer value or -1</returns>
        public static int ParseInt(string Str)
        {
            int X;
            try { X = Int32.Parse(Str); }
            catch { X = -1; }
            return X;
        }

        /// <summary>
        /// Converts a string to an integer value. Returns the value of ErrorValue parameter
        /// if there is a syntax error.
        /// </summary>
        /// <param name="Str">String representation of an integer number</param>
        /// <param name="ErrorValue">A value to be returned in case of an error</param>
        /// <returns>Returns the integer value</returns>
        public static int ParseInt(string Str, int ErrorValue)
        {
            int X;
            try { X = Int32.Parse(Str); }

```

```

        catch { X = ErrorValue; }
        return X;
    }

    /// <summary>
    /// Converts a string to an integer value and stores it in the output parameter Value.
    /// Value is set to -1.0 if there is a syntax error.
    /// </summary>
    /// <param name="Str">String representation of an integer number</param>
    /// <returns>Returns true if the conversion is successful and false otherwise</returns>
    public static bool ParseInt(string Str, out int Value)
    {
        try { Value = Int32.Parse(Str); }
        catch { Value = -1; return false; }
        return true;
    }

    /// <summary>
    /// Converts a string to a double number.
    /// Returns Double.MinValue if there is a syntax error.
    /// </summary>
    /// <param name="Str">String representation of a real number</param>
    /// <returns>Returns the double value or Double.MinValue</returns>
    public static double ParseDouble(string Str)
    {
        double X;
        try { X = Double.Parse(Str); }
        catch { X = Double.MinValue; }
        return X;
    }

    /// <summary>
    /// Converts a string to a double number.
    /// Returns the value of ErrorValue parameter if there is a syntax error.
    /// </summary>
    /// <param name="Str">String representation of a real number</param>
    /// <param name="ErrorValue">A value to be returned in case of an error</param>
    /// <returns>Returns the double number</returns>
    public static double ParseDouble(string Str, double ErrorValue)
    {
        double X;
        try { X = Double.Parse(Str); }
        catch { X = ErrorValue; }
        return X;
    }

    /// <summary>
    /// Converts a string to a double number and stores it in the output parameter Value.
    /// Value is set to Double.MinValue if there is a syntax error.
    /// </summary>
    /// <param name="Str">String representation of an real number</param>
    /// <returns>Returns true if the conversion is successful and false otherwise</returns>
    public static bool ParseDouble(string Str, out double Value)
    {
        try { Value = Double.Parse(Str); }
        catch { Value = Double.MinValue; return false; }
        return true;
    }

    /// <summary>
    /// Ensures that a number is in the interval [Min..Max].
    /// </summary>
    /// <param name="Value">An integer number</param>
    /// <param name="Min">The minimal value</param>
    /// <param name="Max">The maximum value</param>
    /// <returns>Returns the value of Min if Value is less than Min,
    /// returns Max if Value is greater than Max and Value otherwise</returns>
    public static int Limit(int Value, int Min, int Max)
    {
        if (Value > Max) Value = Max;
        if (Value < Min) Value = Min;
        return Value;
    }
}

```

```

/// <summary>
/// Ensures that a number is in the interval [Min..Max].
/// </summary>
/// <param name="Value">A double number</param>
/// <param name="Min">The minimal value</param>
/// <param name="Max">The maximum value</param>
/// <returns>Returns the value of Min if Value is less than Min,
/// returns Max if Value is greater than Max and Value otherwise</returns>
public static double Limit(double Value, double Min, double Max)
{
    if (Value > Max) Value = Max;
    if (Value < Min) Value = Min;
    return Value;
}

/// <summary>
/// Calculates the format string that can be used to format a double number
/// with a specified precision
/// </summary>
/// <param name="Precision">Specifies the desired precision</param>
/// <returns>Returns a format string to be used to format a double</returns>
public static string NumFormat(int Precision)
{
    Precision = Limit(Precision, 1, 15);
    return "0.0000000000000000".Substring(0, Precision + 2);
}

/// <summary>
/// Determines the extension of a file name
/// </summary>
/// <param name="FileName">A file name</param>
/// <returns>File name's extension</returns>
public static string FileExtension(string FileName)
{
    int i = FileName.LastIndexOf(".");
    if (i >= 0 && i < FileName.Length - 1)
        return FileName.Substring(i + 1);
    return "";
}

/// <summary>
/// Removes the extension from a file name
/// </summary>
/// <param name="FileName">A file name</param>
/// <returns>The file name without its extension</returns>
public static string FileNameWithoutExtension(string FileName)
{
    int i = FileName.LastIndexOf(".");
    if (i >= 0 && i < FileName.Length - 1)
        return FileName.Substring(0, i);
    return FileName;
}

/// <summary>
/// Calculates the median value of a set of numbers
/// </summary>
/// <param name="X">A set of numbers whose median value needs to be determined</param>
/// <param name="InPlace">True if the content of X parameter can be destroyed and false
/// if X should be preserved unchanged</param>
/// <returns>Returns the median of the numbers in X</returns>
public static double Median(double[] X, bool InPlace)
{
    double M = 0.0;
    if (X != null)
    {
        int Len = X.Length;
        if (Len > 0)
        {
            if (!InPlace)
                X = (double[])X.Clone();
            Array.Sort(X);
            if ((Len & 1) == 1)
                M = X[Len / 2];
            else

```

```

        {
            Len /= 2;
            M = (X[Len] + X[Len + 1]) / 2.0;
        }
    }
    return M;
}

/// <summary>
/// Calculates the median value of the numbers in X parameter without altering X
/// </summary>
/// <param name="X">A set of numbers whose median value needs to be determined</param>
/// <returns>Returns the median of the numbers in X</returns>
public static double Median(double[] X)
{
    return Median(X, false);
}

/// <summary>
/// Calculates the inverse matrix of matrix A
/// </summary>
/// <param name="A">The matrix that needs to be inverted, Inv(A) on return</param>
public static void InvertMatrix(double[,] A)
{
    if (A == null) return;
    int N = A.GetLength(0);
    if (N != A.GetLength(1)) return;

    double e;
    for (int k = 0; k < N; k++)
    {
        e = A[k, k];
        A[k, k] = 1.0;

        for (int j = 0; j < N; j++)
        {
            A[k, j] = A[k, j] / e;
        }

        for (int i = 0; i < N; i++)
        {
            if (i != k)
            {
                e = A[i, k];
                A[i, k] = 0;
                for (int j = 0; j < N; j++)
                {
                    A[i, j] = A[i, j] - e * A[k, j];
                }
            }
        }
    }
}

#region T-distribution critical value calculation
/// <summary>
/// Calculates the critical value of T distribution for a given degree of freedom (DF) and
/// probability level (Alpha)
/// </summary>
/// <param name="DF">Degree of freedom parameter</param>
/// <param name="Alpha">Probability level, a value in the interval (0, 1)</param>
/// <returns>Returns the calculated critical value</returns>
public static double TTestCV(int DF, double Alpha)
{
    double X = 0.5;
    double DeltaX = 0.5;
    double CV = 0.0;
    while (DeltaX > 0.000001)
    {
        CV = 1.0 / X - 1.0;
        DeltaX /= 2;
        if (TTest_P(CV, DF) > Alpha)
            X -= DeltaX;
    }
}

```

```

        else
            X += DeltaX;
    }
    return CV;
}

/// <summary>
/// A helper function used to calculate T-distribution critical values
/// </summary>
private static double TTest_P(double X, int DF)
{
    X = Math.Abs(X);
    double F = Math.Atan(X/Math.Sqrt(DF));
    if (DF==1) return 1-2*Math.PI;
    double E = Math.Sin(F);
    double D = Math.Cos(F);
    if ((DF & 1) == 1)
        return 1.0 - 2 * (F + E * D * TTest_Z(D * D, 2, DF - 3)) / Math.PI;
    return 1.0 - E * TTest_Z(D * D, 1, DF - 3);
}

/// <summary>
/// A helper function used to calculate T-distribution critical values
/// </summary>
private static double TTest_Z(double Q, int i, int j)
{
    double ZStep = 1.0;
    double Z = 1.0;
    for (int k = i; k <= j; k += 2 )
    {
        ZStep *= (Q * k) / (k + 1);
        Z += ZStep;
    }
    return Z;
}
#endregion
}
}
}

```

## HTSFileReader.cs

```

// File: HTSFileReader.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Text.RegularExpressions;

namespace HTSHelper
{
    class HTSFileReader
    {
        // The name and the format of the input file
        protected string Name;
        protected FileFormats Format;

        // Errors and Warnings produced during the import of the HTS data
        public List<string> Messages;
        public List<string> Warnings;

        // The content of the input file
        protected string[] Lines;
        protected string Text;
    }
}

```

```

// used in case of .csv datafile
public int RowsPerPlate = -1;

protected Dataset Data;

/// <summary>
/// Constructor
/// </summary>
/// <param name="FileName">The name of the input file</param>
/// <param name="FileFormat">The format of the input file</param>
public HTSFileReader(string FileName, FileFormats FileFormat)
{
    Name = FileName;
    Format = FileFormat;
}

/// <summary>
/// Reads all lines of the input text file into Lines field
/// </summary>
/// <returns>True on success, false in case of an error</returns>
protected bool ReadAllLines()
{
    try
    {
        Lines = File.ReadAllLines(Name);
    }
    catch (Exception ex)
    {
        Messages.Add(ex.Message);
        Messages.Add("Cannot read the input file '" + Name + "'");
        Lines = null;
        return false;
    }
    return true;
}

/// <summary>
/// Reads the content of the input text file into Text field
/// </summary>
/// <returns>True on success, false in case of an error</returns>
protected bool ReadAllText()
{
    try
    {
        Text = File.ReadAllText(Name);
    }
    catch (Exception ex)
    {
        Messages.Add(ex.Message);
        Messages.Add("Cannot read the input file '" + Name + "'");
        Text = null;
        return false;
    }
    return true;
}

/// <summary>
/// Reads data from the input file and construct a Dataset object.
/// Dispatches to specialized methods depending on the input file format
/// </summary>
/// <returns>A Dataset object on success, null in case of an error</returns>
public Dataset ReadData()
{
    Data = null;

    Messages = new List<string>();
    Warnings = new List<string>();

    switch(Format)
    {
        case FileFormats.Asy:
            if (ReadAllLines())
                Data = ProcessAsyFile();
    }
}

```

```

        break;

    case FileFormats.Csv:
        if (ReadAllLines())
            Data = ProcessCsvFile();
        break;

    case FileFormats.CsvData:
        if (ReadAllLines())
            Data = ProcessCsvDataFile();
        break;

    case FileFormats.CsvDB:
        if (ReadAllLines())
            Data = ProcessCsvDBFile();
        break;

    case FileFormats.Mtr:
        if (ReadAllLines())
            Data = ProcessMtrFile();
        break;

    case FileFormats.Mtx:
        if (ReadAllText())
            Data = ProcessMtxFile();
        break;

    default:
        Messages.Add("Unknown input data format: " + (int)Format);
        break;
    }

    if (Data != null)
    {
        Messages.AddRange(Warnings);
    }
    return Data;
}

/// <summary>
/// .asy format support is not implemented in version 1.0
/// </summary>
/// <returns>A Dataset object on success, null in case of an error</returns>
protected Dataset ProcessAsyFile()
{
    return null;
}

/// <summary>
/// Reads data from a .csv file. Detects if the input file is in CSV data or
/// CSV database file format.
/// </summary>
/// <returns>A Dataset object on success, null in case of an error</returns>
protected Dataset ProcessCsvFile()
{
    Regex R = new Regex(@"\w", RegexOptions.IgnoreCase);

    foreach (string Line in Lines)
    {
        Match M = R.Match(Line);
        if (M.Success)
        {
            if (Char.IsDigit(M.Value[0]))
            {
                return ProcessCsvDataFile();
            }
            else
            {
                return ProcessCsvDBFile();
            }
        }
    }
}
Messages.Add("Empty file or bad .csv file format: " + Name + ".");

```

```

        return null;
    }

    /// <summary>
    /// Parses a line of double numbers separated by commas
    /// </summary>
    /// <param name="Line">A string of comma separated values, single line</param>
    /// <param name="RowData">A list of double numbers</param>
    /// <param name="Msg">An error message in case of an error</param>
    /// <returns>True on success, false in case of an error</returns>
    protected bool ParseCommaDataLine(string Line, out List<double> RowData, out string Msg)
    {
        char[] Delim = ",".ToCharArray();
        string[] Arr = Line.Split(Delim);
        RowData = new List<double>();

        if (Arr.Length == 1)
        {
            Msg = "Syntax error - not comma separated values.";
            return false;
        }
        for (int i = 0; i < Arr.Length; i++)
        {
            string Str = Arr[i].Trim();
            double V;
            if (G.ParseDouble(Str, out V))
                RowData.Add(V);
            else
            {
                if (Str == "") Msg = "Missing value";
                else Msg = "Bad number: '" + Str + "'";
                Msg += ", well " + (1+RowData.Count);
                return false;
            }
        }
        Msg = "";
        return true;
    }

    /// <summary>
    /// Parses a multi-line string of double numbers separated by tab character
    /// </summary>
    /// <param name="Line">A string of tab separated values, multi-lines</param>
    /// <param name="Plate">A Plate object containing the number values</param>
    /// <param name="Msg">An error message in case of an error</param>
    /// <returns>True on success, false in case of an error</returns>
    protected bool ParseTabbedData(string Str, Plate PL, out string Msg)
    {
        char[] NewLine = "\r\n".ToCharArray();
        string[] Rows = Str.Split(NewLine, StringSplitOptions.RemoveEmptyEntries);

        if (Rows.Length != PL.NumRows)
        {
            Msg = "Bad rows number: " + Rows.Length + " instead of the expected " + PL.NumRows;
            return false;
        }
        for (int i = 0; i < Rows.Length; i++)
        {
            string[] Arr = Rows[i].Split("\t".ToCharArray(), StringSplitOptions.
RemoveEmptyEntries);

            if (Arr.Length != PL.NumColumns)
            {
                Msg = "Bad number of items in row " + (i+1) + ": " + Arr.Length +
                    " instead of the expected " + PL.NumColumns;
                return false;
            }
            for (int j = 0; j < Arr.Length; j++)
            {
                double V;
                if (G.ParseDouble(Arr[j], out V))
                    PL[i, j] = V;
                else
                {

```

```

        if (Str == "") Msg = "Missing value";
        else Msg = "Bad number: " + Str + "";
        Msg += ", row " + (1 + i) + ", column " + (j+1);
        return false;
    }
}
}
Msg = "";
return true;
}

/// <summary>
/// Reads data from a .csv file on CSV data format.
/// </summary>
/// <returns>A Dataset object on success, null in case of an error</returns>
protected Dataset ProcessCsvDataFile()
{
    int LastPlateProcessed = -1;
    int CurrPlate = 0;
    int Rows = 0;
    int Columns = 0;
    Dataset Data = null;
    Plate PL = null;
    bool DefinedSize = RowsPerPlate > 0;

    // if the number of rows is not specified - calculate it
    if (RowsPerPlate < 0)
    {
        foreach (string Line in Lines)
        {
            if (Line.Trim().Length > 0) Rows++;
            else if (Rows > 0)
            {
                RowsPerPlate = Rows;
                if (Rows < 4)
                {
                    Messages.Add("Bad data format. Plates with less than 4 rows.");
                    return null;
                }
                break;
            }
        }
    }

    int LNum = 1;
    int NRows = 0;

    foreach (string Line in Lines)
    {
        List<double> RowData;
        string Msg;

        if (Line.Trim().Length > 1)
        {
            if (ParseCommaDataLine(Line, out RowData, out Msg))
            {
                if (CurrPlate == 0 && NRows == 0)
                {
                    Columns = RowData.Count;
                    if (Columns < 4)
                    {
                        Messages.Add("Bad data format. Plates with less than 4 columns.");
                        return null;
                    }
                }

                if (Data == null)
                    Data = new Dataset(RowsPerPlate, Columns);

                if (PL == null)
                    PL = Data.AddPlate();

                if (RowData.Count < Columns)
                {

```

```

        Messages.Add("Plate " + (CurrPlate + 1) + ", row " + (NRows + 1) +
" is too short - " + RowData.Count + " instead of " + Columns + " items.");
        return null;
    }
    else if (RowData.Count > Columns)
    {
        Messages.Add("Plate " + (CurrPlate + 1) + ", row " + (NRows + 1) +
" is too long - " + RowData.Count + " instead of " + Columns + " items.");
        return null;
    }

    if (NRows == RowsPerPlate)
    {
        if (!DefinedSize)
        {
            Messages.Add("Plate " + (CurrPlate + 1) + " has too many rows,
more than " + RowsPerPlate + ".");
            return null;
        }
        else
        {
            CurrPlate = ++LastPlateProcessed + 1;
            NRows = 0;
            PL = null;
        }
    }

    if (PL == null)
        PL = Data.AddPlate();

    for (int j = 0; j < Columns; j++)
    {
        PL[NRows, j] = RowData[j];
    }
    NRows++;
}
else
{
    Messages.Add("Line " + LNum + ": " + Msg);
    return null;
}
}
else
{
    if (!DefinedSize)
    {
        if (NRows > 0)
        {
            if (NRows < RowsPerPlate)
            {
                Messages.Add("Plate " + (CurrPlate + 1) + " has too few rows,
less than " + RowsPerPlate + ".");
                return null;
            }
            CurrPlate = ++LastPlateProcessed + 1;
            NRows = 0;
            PL = null;
        }
    }
}
LNum++;
}
if (NRows > 0 && NRows < RowsPerPlate)
{
    Messages.Add("The last plate has too few rows, less than " + RowsPerPlate + ".");
    return null;
}
return Data;
}

/// <summary>
/// Reads data from a .csv file on CSV database format.
/// </summary>
/// <returns>A Dataset object on success, null in case of an error</returns>

```

```

protected Dataset ProcessCsvDBFile()
{
    if (Lines.Length < 1)
    {
        Messages.Add("Empty file: " + Name + ".");
        return null;
    }

    int i;
    string[] Keys;
    Dictionary<string, string> Dict = new Dictionary<string, string>();

    char[] Comma = new char[] { ',' };
    string Str = Lines[0].ToLower();
    Keys = Str.Split(Comma);

    for (i = 0; i < Keys.Length; i++)
    {
        Dict.Add(Keys[i] = Keys[i].Trim(), "0");
    }

    StringBuilder SB = new StringBuilder();

    if (!Dict.ContainsKey("plate")) SB.Append("'Plate'");
    if (!Dict.ContainsKey("row")) ( if (SB.Length > 0) SB.Append(", "); SB.Append("'Row'"); )
    if (!Dict.ContainsKey("column")) { if (SB.Length > 0) SB.Append(", "); SB.Append(  ↵
"'Column'"); }
    if (!Dict.ContainsKey("value")) { if (SB.Length > 0) SB.Append(", "); SB.Append(  ↵
"'Value'"); }

    if (SB.Length > 0)
    {
        Messages.Add("Bad file format - missing column(s): " + SB.ToString());
        return null;
    }

    List<int> PlateList = new List<int>();
    List<int> RowList = new List<int>();
    List<int> ColList = new List<int>();
    List<double> ValueList = new List<double>();

    for (i = 1; i < Lines.Length; i++)
    {
        Str = Lines[i].Trim();
        if (Str.Length < 1) continue;
        string[] Arr = Lines[i].Split(Comma);
        if (Arr.Length != 4)
        {
            Messages.Add("Format error on line " + (i+1));
            return null;
        }
        for (int j = 0; j < Arr.Length; j++) Dict[Keys[j]] = Arr[j].Trim();

        bool ok;
        int X; double V;
        if (ok = G.ParseInt(Str = Dict["plate"], out X))
        {
            if (ok = (X >= 1 && X <= 100000))
            {
                PlateList.Add(X-1);
                if (ok = G.ParseInt(Str = Dict["row"], out X))
                {
                    if (ok = (X >= 1 && X <= 1000))
                    {
                        RowList.Add(X-1);
                        if (ok = G.ParseInt(Str = Dict["column"], out X))
                        {
                            if (ok = (X >= 1 && X <= 1000))
                            {
                                CollList.Add(X-1);
                                if (ok = G.ParseDouble(Str = Dict["value"], out V))
                                {
                                    ValueList.Add(V);
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
    }
    }
    }
    }
    if (!ok)
    {
        Messages.Add("Format error on line " + (i+1) + ". Bad number: " + Str + "");
        return null;
    }
}

int Plates = PlateList.Max()+1;
int Rows = RowList.Max()+1;
int Columns = ColList.Max()+1;

if (Rows < 4)
{
    Messages.Add("Bad data. Less than 4 rows.");
    return null;
}
if (Columns < 4)
{
    Messages.Add("Bad data. Less than 4 columns.");
    return null;
}

Dataset Data = new Dataset(Plates, Rows, Columns);
bool[,,,] Flag = new bool[Plates, Rows, Columns];

for (i = 0; i < PlateList.Count; i++)
{
    if (Flag[PlateList[i], RowList[i], ColList[i]])
    {
        Messages.Add("Repeating value: plate " + PlateList[i] + ", row " +
RowList[i] + ", column " + ColList[i]);
        return null;
    }
    Data[PlateList[i], RowList[i], ColList[i]] = ValueList[i];
    Flag[PlateList[i], RowList[i], ColList[i]] = true;
}

for (int p = 0; p < Plates; p++)
{
    for (i = 0; i < Rows; i++)
    {
        for (int j = 0; j < Columns; j++)
        {
            if (Flag[p, i, j] == false)
            {
                Messages.Add("Missing value: plate " + (p+1) + ", row " + (i+1) +
", column " + (j+1));
                return null;
            }
        }
    }
}

return Data;
}

/// <summary>
/// Reads data from a .mtx file in HTS Corrector pseudo XML format.
/// </summary>
/// <returns>A Dataset object on success, null in case of an error</returns>
protected Dataset ProcessMtxFile()
{
    Regex AssayTag = new Regex(@"<s*assay\s+([\^<]*)\s*>", RegexOptions.IgnoreCase);
    Match M = AssayTag.Match(Text);

    int Plates = 0;
    int Rows = 0;
    int Columns = 0;
}

```

```

int Pos = 0;

string Msg;

if (M.Success)
{
    string PlatesStr = "";
    string RowsStr = "";
    string ColumnsStr = "";

    Pos = M.Index + M.Value.Length;

    string S = M.Groups[1].Value;
    Regex PlatesAttr = new Regex(@"plates\s*=\s*" + S + @"\s*(\d+)\s*" + S + "", RegexOptions.
IgnoreCase);
    M = PlatesAttr.Match(S);
    if (M.Success)
    {
        PlatesStr = M.Groups[1].Value;
        Plates = G.ParseInt(PlatesStr);
    }
    else
    {
        Messages.Add("Bad format! ASSAY tag without 'plates' attribute!");
        return null;
    }

    Regex RowsAttr = new Regex(@"rows\s*=\s*" + S + @"\s*(\d+)\s*" + S + "", RegexOptions.
IgnoreCase);
    M = RowsAttr.Match(S);
    if (M.Success)
    {
        RowsStr = M.Groups[1].Value;
        Rows = G.ParseInt(RowsStr);
    }
    else
    {
        Messages.Add("Bad format! ASSAY tag without 'rows' attribute!");
        return null;
    }

    Regex ColumnsAttr = new Regex(@"column?\s*=\s*" + S + @"\s*(\d+)\s*" + S + "", RegexOptions.
IgnoreCase);
    M = ColumnsAttr.Match(S);
    if (M.Success)
    {
        ColumnsStr = M.Groups[1].Value;
        Columns = G.ParseInt(ColumnsStr);
    }
    else
    {
        Messages.Add("Bad format! ASSAY tag without 'columns' attribute!");
        return null;
    }

    if (Plates < 1 || Plates > 100000)
    {
        Messages.Add("Bad number of plates: " + PlatesStr + "! An integer number
between 1 and 100,000 is expected.");
        return null;
    }
    if (Rows < 4 || Rows > 1000)
    {
        Messages.Add("Bad number of rows: " + RowsStr + "! An integer number
between 4 and 1000 is expected.");
        return null;
    }
    if (Columns < 4 || Columns > 1000)
    {
        Messages.Add("Bad number of rows: " + ColumnsStr + "! An integer number
between 4 and 1000 is expected.");
        return null;
    }
}
}

```

```

else
{
    Messages.Add("Bad format! Missing ASSAY tag!");
    return null;
}

Data = new Dataset(Plates, Rows, Columns);
int CurrPlate = 0;
Regex PlateTag = new Regex(@"<s*plate\s+([\^<]*)\s*>", RegexOptions.IgnoreCase);

bool[] PMap = new bool[Plates];

M = PlateTag.Match(Text, Pos);
while (M.Success)
{
    int PNum = 0;

    Pos = M.Index + M.Value.Length;
    string S = M.Groups[1].Value;

    Regex NoAttr = new Regex(@"no\s*=\s*" + S + "\s*(\d+)\s*" + S, RegexOptions.IgnoreCase);
    M = NoAttr.Match(S);
    if (M.Success)
    {
        string NoStr = M.Groups[1].Value;
        PNum = G.ParseInt(NoStr);
        if (PNum < 1 || PNum > Plates)
        {
            Messages.Add("Bad plate number: " + PNum + "! An integer number
between 1 and " + Plates + " is expected.");
            return null;
        }
        PNum--;
    }
    else
        PNum = CurrPlate + 1;

    int Ind = Text.IndexOf("<", Pos);

    string Str = (Ind >= Pos) ? Text.Substring(Pos, Ind - Pos) : Text.Substring(Pos);

    if (PMap[PNum])
    {
        Messages.Add("Repetition of plate number: " + (PNum+1) + "!");
        return null;
    }

    if (!ParseTabbedData(Str, Data[PNum], out Msg))
    {
        Messages.Add("Plate " + (PNum+1) + ": " + Msg);
        return null;
    }
    PMap[PNum] = true;

    CurrPlate = PNum;
    M = PlateTag.Match(Text, Pos);
}

for (int i = 0; i < Plates; i++)
{
    if (PMap[i]==false)
    {
        Messages.Add("Missing plate number " + (i+1) + "!");
        return null;
    }
}

return Data;
}

/// <summary>
/// Reads data from a .mtr file in HTS Corrector file format.
/// </summary>
/// <returns>A Dataset object on success, null in case of an error</returns>

```

```

protected Dataset ProcessMtrFile()
{
    Dataset Data = null;

    char[] SPACE = { ' ', '\t' };

    int NPlates = 0;
    int NRows = 0;
    int NColumns = 0;

    if (Lines == null || Lines.Length < 2)
    {
        Messages.Add("Bad format or empty file: " + Name + ".");
        return null;
    }

    int Len = Lines.Length;
    string[] Arr;

    Arr = Lines[0].Split(SPACE);

    if (Arr != null)
    {
        if (Arr.Length > 0)
        {
            NPlates = G.ParseInt(Arr[0], -1);
            if (NPlates < 1)
            {
                Messages.Add("Bad number of plates: " + Arr[0] + ".");
                return null;
            }
        }
        if (Arr.Length > 1)
        {
            NRows = G.ParseInt(Arr[1], -1);
            if (NRows < 2 || NRows > 1000)
            {
                Messages.Add("Bad number of rows: " + Arr[1] + ". An integer between 2
and 1000 is expected.");
                return null;
            }
        }
        if (Arr.Length > 2)
        {
            NColumns = G.ParseInt(Arr[2], -1);
            if (NColumns < 2 || NColumns > 100)
            {
                Messages.Add("Bad number of columns " + Arr[2] + ". An integer between
2 and 1000 is expected.");
                return null;
            }
        }
    }

    if (Lines.Length != 1 + NRows * NColumns)
    {
        Messages.Add("Bad file format. It contains " + Lines.Length + " lines instead of
the expected " + (1 + NRows * NColumns));
        return null;
    }

    Data = new Dataset(NPlates, NRows, NColumns);

    for (int l = 1; l < Lines.Length; l++)
    {
        int row = (l - 1) % NRows;
        int col = (l - 1) / NRows;

        Arr = Lines[l].Split(SPACE, StringSplitOptions.RemoveEmptyEntries);

        if (Arr == null || Arr.Length != NPlates)
        {
            Messages.Add("Bad file format. Line " + (l+1) + " contains " + Arr.Length +

```

```

" numbers instead of the expected " + NPlates);
        return Data = null;
    }

    int p = 0;
    foreach (string s in Arr)
    {
        try
        {
            double X = Double.Parse(s.Trim());
            Data[p, row, col] = X;
        }
        catch
        {
            Data[p, row, col] = 0.0;
            Warnings.Add("Bad number '" + s + "' on line " + (l+1) + ". Ignored! 0.0 ⚡
assumed.");
        }
        p++;
    }
}

return Data;
}
}
}

```

## HTSFileWriter.cs

```

// File: HTSFileWriter.cs, HTSHelper project
// Written by Plamen Dragiev
// Last updated: Mar 14, 2012
// History:
//   Mar 3, 2012 - file created
//   Mar 14, 2012 - version 1.0
//

using System;
using System.Collections.Generic;
using System.Text;
using System.IO;

namespace HTSHelper
{
    class HTSFileWriter
    {
        // The name and the format of the output file
        protected string Name;
        protected FileFormats Format;

        // Errors and Warnings produced during the export of the HTS data
        public List<string> Messages;
        public List<string> Warnings;

        protected string NumFormat;
        protected int FLUSH_SIZE = 100000;

        protected StringBuilder Buf;

        protected Dataset Data;

        /// <summary>
        /// Constructor
        /// </summary>
        /// <param name="FileName">The name of the output file</param>
        /// <param name="FileFormat">The format of the output file</param>
        public HTSFileWriter(string FileName, FileFormats FileFormat)
        {
            Name = FileName;
            Format = FileFormat;
        }
    }
}

```

```

        NumFormat = G.NumFormat(6);
    }

    /// <summary>
    /// Specifies the number format to be used for outputing numbers
    /// </summary>
    /// <param name="Format">A number format to be used</param>
    public void SetNumFormat(string Format)
    {
        NumFormat = Format;
    }

    /// <summary>
    /// Creates the output file if it does not exist or truncates it if exists
    /// </summary>
    /// <returns>Returns true on success and false in case of an error</returns>
    protected bool CreateOrTruncate()
    {
        try
        {
            File.WriteAllText(Name, "");
        }
        catch (Exception E)
        {
            Messages.Add(E.Message);
            Messages.Add("Cannot write to '" + Name + "'.");
            return false;
        }
        return true;
    }

    /// <summary>
    /// Writes the memory buffer to the disk
    /// </summary>
    /// <returns>Returns true on success and false in case of an error</returns>
    protected bool Flush()
    {
        if (Buf != null && Buf.Length > 0)
        {
            try
            {
                File.AppendAllText(Name, Buf.ToString());
                Buf.Length = 0;
            }
            catch
            {
                return false;
            }
        }
        return true;
    }

    /// <summary>
    /// Checks if the size of the memory buffer is over the limit and writes it
    /// to the disk if it is
    /// </summary>
    /// <returns>Returns true on success and false in case of an error</returns>
    protected bool CheckFlush()
    {
        if (Buf.Length >= FLUSH_SIZE)
            return Flush();
        return true;
    }

    /// <summary>
    /// Writes text to the output file
    /// </summary>
    /// <param name="Str">Text to be written</param>
    /// <returns>Returns true on success and false in case of an error</returns>
    protected bool Write(string Str)
    {
        Buf.Append(Str);
        return CheckFlush();
    }
}

```

```

    /// <summary>
    /// Writes a text line to the output file
    /// </summary>
    /// <param name="Str">The text of the line that should be written to the output file
</param>
    /// <returns>Returns true on success and false in case of an error</returns>
    protected bool WriteLine(string Str)
    {
        Buf.Append(Str);
        Buf.Append(Environment.NewLine);
        return CheckFlush();
    }

    /// <summary>
    /// Exports the data of a Dataset to the output file
    /// </summary>
    /// <param name="DataSet">Data to be written</param>
    /// <returns>Returns true on success and false in case of an error</returns>
    public bool WriteData(Dataset DataSet)
    {
        Data = DataSet;

        Messages = new List<string>();
        Warnings = new List<string>();

        Buf = new StringBuilder((int)(FLUSH_SIZE*1.1));

        if (CreateOrTruncate())
        {
            bool Res;
            switch(Format)
            {
                case FileFormats.Asy:
                    Res = WriteAsyFile();
                    break;

                case FileFormats.CsvData:
                    Res = WriteCsvDataFile();
                    break;

                case FileFormats.CsvDB:
                    Res = WriteCsvDBFile();
                    break;

                case FileFormats.Mtr:
                    Res = WriteMtrFile();
                    break;

                case FileFormats.Mtx:
                    Res = WriteMtxFile();
                    break;

                case FileFormats.Html:
                    Res = WriteHtmlFile();
                    break;

                default:
                    Messages.Add("Unknown output data format: " + (int)Format);
                    Res = false;
                    break;
            }

            if (Res) Messages = Warnings;
            return Res;
        }
        return false;
    }

    /// <summary>
    /// .asy file format is not supported in version 1.0
    /// </summary>
    /// <returns>Returns true on success and false in case of an error</returns>
    protected bool WriteAsyFile()

```

```

{
    return true;
}

/// <summary>
/// Writes a Dataset to a CSV data file
/// </summary>
/// <returns>Returns true on success and false in case of an error</returns>
protected bool WriteCsvDataFile()
{
    for (int p = 0; p < Data.NumPlates; p++)
    {
        for (int i = 0; i < Data.NumRows; i++)
        {
            for (int j = 0; j < Data.NumColumns; j++)
            {
                if (j > 0) Buf.Append(", ");
                Buf.Append(Data[p, i, j].ToString(NumFormat));
            }
            Buf.AppendLine();
            if (!CheckFlush()) return false;
        }
        Buf.AppendLine();
    }
    return Flush();
}

/// <summary>
/// Writes a Dataset to a CSV database file
/// </summary>
/// <returns>Returns true on success and false in case of an error</returns>
protected bool WriteCsvDBFile()
{
    Buf.AppendLine("Plate, Row, Column, Value");
    for (int p = 0; p < Data.NumPlates; p++)
    {
        for (int i = 0; i < Data.NumRows; i++)
        {
            for (int j = 0; j < Data.NumColumns; j++)
            {
                Buf.Append((p + 1).ToString()).Append(", ");
                Buf.Append((i + 1).ToString()).Append(", ");
                Buf.Append((j + 1).ToString()).Append(", ");
                Buf.AppendLine(Data[p, i, j].ToString(NumFormat));
            }
        }
        if (!CheckFlush()) return false;
    }
    Buf.AppendLine();
    return Flush();
}

/// <summary>
/// Writes a Dataset to a MTR data file (HTS Corrector)
/// </summary>
/// <returns>Returns true on success and false in case of an error</returns>
protected bool WriteMtrFile()
{
    Buf.Append(Data.NumPlates.ToString()).Append(" ");
    Buf.Append(Data.NumRows.ToString()).Append(" ");
    Buf.Append(Data.NumColumns.ToString()).AppendLine();

    for (int j = 0; j < Data.NumColumns; j++)
    {
        for (int i = 0; i < Data.NumRows; i++)
        {
            for (int p = 0; p < Data.NumPlates; p++)
            {
                if (p > 0) Buf.Append("\t");
                Buf.Append(Data[p, i, j].ToString(NumFormat));
            }
            Buf.AppendLine();
            if (!CheckFlush()) return false;
        }
    }
}

```

```

    }
    }
    Buf.AppendLine();
    return Flush();
}

/// <summary>
/// Writes a Dataset to a MTX data file (HTS Corrector)
/// </summary>
/// <returns>Returns true on success and false in case of an error</returns>
protected bool WriteMtxFile()
{
    Buf.AppendFormat("<ASSAY plates=\"{0}\" rows=\"{1}\" columns=\"{2}\"
columns=\"{2}\">", Data.NumPlates, Data.NumRows, Data.NumColumns);
    Buf.AppendLine().AppendLine();

    for (int p = 0; p < Data.NumPlates; p++)
    {
        Buf.AppendLine("<Plate name=\"" + (p+1).ToString() + "\" no=\"" +
(p+1).ToString() + "\">");
        for (int i = 0; i < Data.NumRows; i++)
        {
            for (int j = 0; j < Data.NumColumns; j++)
            {
                if (j > 0) Buf.Append("\t");
                Buf.Append(Data[p, i, j].ToString(NumFormat));
            }
            Buf.AppendLine();
            if (!CheckFlush()) return false;
        }
        Buf.AppendLine("</Plate>").AppendLine();
    }
    Buf.Append("</ASSAY>").AppendLine().AppendLine();
    return Flush();
}

/// <summary>
/// Writes a Dataset to a html file
/// </summary>
/// <returns>Returns true on success and false in case of an error</returns>
protected bool WriteHtmlFile()
{
    Buf.Append("<html>").AppendLine();
    Buf.Append("<head>").AppendLine();
    Buf.Append("<style>").AppendLine();
    Buf.Append(" .pltclass { font-family: Arial, Helvetica; font-size: 12pt; font-
weight: bold; }").AppendLine();
    Buf.Append(" .tblclass { font-family: Arial, Helvetica; font-size: 11pt; border:
1px solid #777777; }").AppendLine();
    Buf.Append(" .hdrclass { font-family: Arial, Helvetica; font-size: 11pt; border:
1px solid #777777; font-weight: bold; background-color: #F0F0F0; text-align: center; padding-
left: 8pt; padding-right: 8pt; }").AppendLine();
    Buf.Append(" .cellclass { font-family: Arial, Helvetica; font-size: 11pt; border:
1px solid #777777; text-align: center; padding-left: 3pt; padding-right: 3pt; }").AppendLine();
    Buf.Append("</style>").AppendLine();
    Buf.Append("</head>").AppendLine();
    Buf.Append("<body>").AppendLine();
    Buf.Append("<center>").AppendLine();

    for (int p = 0; p < Data.NumPlates; p++)
    {
        Buf.AppendLine("<div class=\"pltclass\">Plate " + (p + 1).ToString() + "</div>");
        Buf.AppendLine("<table class=\"tblclass\" cellspacing=\"0\">");

        Buf.AppendLine("<tr>");
        Buf.Append("<td class=\"hdrclass\"> # </td>");
        for (int k = 0; k < Data.NumColumns; k++)
        {
            Buf.Append("<td class=\"hdrclass\"> " + (k + 1) + " </td>");
        }
        Buf.AppendLine().AppendLine("</tr>");
        for (int i = 0; i < Data.NumRows; i++)
        {
            Buf.AppendLine("<tr>");

```



```

        OutputFileName = "";

        InputFormat = FileFormats.Auto;
        OutputFormat = FileFormats.Auto;

        PlateRows = -1;

        Precision = 6;
        NumFormat = G.NumFormat(Precision);
    }

    public void SetUI(UserInterface anUI)
    {
        UI = anUI;
    }

    /// <summary>
    /// Determines the file format based on its extension
    /// </summary>
    /// <param name="FileName">A file name</param>
    /// <returns>Returns the format of the file</returns>
    public static FileFormats FileFormatFromFileName(string FileName)
    {
        string Ext = G.FileExtension(FileName);
        Ext = Ext.ToLower();
        if (Ext == "asy")
            return FileFormats.Asy;
        if (Ext == "mtr")
            return FileFormats.Mtr;
        if (Ext == "mtx")
            return FileFormats.Mtx;
        if (Ext == "csv")
            return FileFormats.Csv;
        return FileFormats.Unknown;
    }

    /// <summary>
    /// Returns the default extension for the specified file format
    /// </summary>
    /// <param name="Format">A file format</param>
    /// <returns>The default extension for the format</returns>
    public static string FileFormatExtension(FileFormats Format)
    {
        if (Format == FileFormats.Asy)
            return "asy";
        if (Format == FileFormats.Mtr)
            return "mtr";
        if (Format == FileFormats.Mtx)
            return "mtx";
        if (Format == FileFormats.Html)
            return "html";
        if (Format == FileFormats.CsvDB)
            return "csv";
        if (Format == FileFormats.Csv || Format == FileFormats.CsvData)
            return "csv";
        return "";
    }

    /// <summary>
    /// Adds an extension to the file name
    /// </summary>
    /// <param name="FileName">A file name</param>
    /// <param name="Format">The format of the file</param>
    /// <returns>Returns a full name with extension</returns>
    public static string AddFormatExtension(string FileName, FileFormats Format)
    {
        string Ext = FileFormatExtension(Format);
        if (Ext != "")
        {
            if (G.FileExtension(FileName).ToLower() != Ext)
                FileName += "." + Ext;
        }
        return FileName;
    }
}

```

```

/// <summary>
/// Builds an output file name based on the input file name and the action performed
/// </summary>
/// <param name="FileName">The input File name</param>
/// <param name="Action">The action performed</param>
/// <returns>Returns the constructed output file name</returns>
public static string BuildOutputFileName(string FileName, Actions Action)
{
    string Name = G.FileNameWithoutExtension(FileName);
    string Sfx = "";
    if (Action == Actions.Convert) Sfx = "_CONVERTED";
    else if (Action == Actions.Bscore) Sfx = "_BSCORE";
    else if (Action == Actions.WC) Sfx = "_WC";
    else if (Action == Actions.WC_MEA) Sfx = "_WC_MEA";
    else if (Action == Actions.WC_PMP) Sfx = "_WC_PMP";
    else if (Action == Actions.MEA) Sfx = "_MEA";
    else if (Action == Actions.PMP) Sfx = "_PMP";
    else if (Action == Actions.Zscore) Sfx = "_ZSCORE";
    return Name + Sfx;
}

/// <summary>
/// Checks the validity of the supplied parameters
/// </summary>
/// <returns>Returns true on success and if a problem is detected</returns>
public bool CheckParameters()
{
    NumFormat = G.NumFormat(Precision);

    if (InputFormat == FileFormats.Auto)
    {
        InputFormat = FileFormatFromFileName(InputFileName);
        if (InputFormat == FileFormats.Unknown || InputFormat == FileFormats.Auto)
        {
            UI.Error("Unknown input file format!");
            return false;
        }
        if (InputFormat == FileFormats.Asy)
        {
            UI.Error(".asy file format is not supported in version 1.0!");
            return false;
        }
    }

    if (OutputFormat == FileFormats.Auto)
    {
        if (OutputFileName != "")
        {
            OutputFormat = FileFormatFromFileName(OutputFileName);
        }
        if (OutputFormat == FileFormats.Unknown || OutputFormat == FileFormats.Auto)
        {
            OutputFormat = InputFormat;
        }
        if (OutputFormat == FileFormats.Asy)
        {
            UI.Error(".asy file format is not supported in version 1.0!");
            return false;
        }
    }

    if (OutputFileName == "")
    {
        OutputFileName = BuildOutputFileName(InputFileName, Action);
    }

    OutputFileName = AddFormatExtension(OutputFileName, OutputFormat);
    return true;
}

/// <summary>
/// Executes the required action
/// </summary>

```

```

/// <param name="anAction">The code of the action that should be performed</param>
/// <returns>Returns true on succes and false in case of an error</returns>
public bool Execute(Actions anAction)
{
    this.Action = anAction;

    if (CheckParameters())
    {
        if (ReadInputFile() && Data != null)
        {
            switch (Action)
            {
                case Actions.Bscore:
                    DoBscore();
                    break;

                case Actions.Zscore:
                    DoZscore();
                    break;

                case Actions.MEA:
                    DoMEA();
                    break;

                case Actions.PMP:
                    DoPMP();
                    break;

                case Actions.WC:
                    DoWC();
                    break;

                case Actions.WC_MEA:
                    DoWCandMEA();
                    break;

                case Actions.WC_PMP:
                    DoWCandPMP();
                    break;

                case Actions.Convert:
                    // Do nothing
                    break;

                default:
                    UI.Error("Unknown/Unsupported action requested!");
                    Data = null;
                    break;
            }
        }

        if (Data != null)
            return WriteOutputFile();
    }
    return false;
}

/// <summary>
/// Reads the data from the input file
/// </summary>
/// <returns>Returns true on succes and false in case of an error</returns>
protected bool ReadInputFile()
{
    HTSFileReader Reader = new HTSFileReader(InputFileName, InputFormat);

    if (PlateRows > 0)
        Reader.RowsPerPlate = PlateRows;

    Data = Reader.ReadData();

    if (Data == null)
    {
        if (Reader.Messages == null || Reader.Messages.Count == 0)
        {

```

```

        UI.Error("Failed to read the input data!");
    }
    else
    {
        foreach (string Msg in Reader.Messages)
        {
            UI.Error(Msg);
        }
    }
}
else
{
    if (Reader.Messages != null)
    {
        foreach (string Msg in Reader.Messages)
        {
            UI.Warning(Msg);
        }
    }
}
return Data != null;
}

/// <summary>
/// Writes data to the output file
/// </summary>
/// <returns>Returns true on succes and false in case of an error</returns>
protected bool WriteOutputFile()
{
    HTSFileWriter Writer = new HTSFileWriter(OutputFileName, OutputFormat);
    Writer.SetNumFormat(NumFormat);

    if (!Writer.WriteData(Data))
    {
        if (Writer.Messages == null || Writer.Messages.Count == 0)
        {
            UI.Error("Failed to write the output data!");
        }
        else
        {
            foreach (string Msg in Writer.Messages)
            {
                UI.Error(Msg);
            }
        }
        return false;
    }
    else
    {
        if (Writer.Messages != null)
        {
            foreach (string Msg in Writer.Messages)
            {
                UI.Warning(Msg);
            }
        }
    }
    return true;
}

/// <summary>
/// Applies B-score method on the dataset
/// </summary>
protected void DoBscore()
{
    Data.Bscore();
}

/// <summary>
/// Applies Z-score method on the dataset
/// </summary>
protected void DoZscore()
{
    Data.Zscore();
}

```

```

}

/// <summary>
/// Applies Matrix Error Amendment (MEA) method on the dataset
/// </summary>
protected void DoMEA()
{
    Data.MEA();
}

/// <summary>
/// Applies Partial Mean Polish (PMP) method on the dataset
/// </summary>
protected void DoPMP()
{
    Data.PMP();
}

/// <summary>
/// Applies Well Correction method on the dataset
/// </summary>
protected void DoWC()
{
    Data.WellCorrection();
}

/// <summary>
/// Applies Well Correction method followed by Matrix Error Amendment (MEA) method on
the dataset
/// </summary>
protected void DoWCandMEA()
{
    Data.WellCorrection();
    Data.MEA();
}

/// <summary>
/// Applies Well Correction method followed by Partial Mean Polish (PMP) method on the
dataset
/// </summary>
protected void DoWCandPMP()
{
    Data.WellCorrection();
    Data.PMP();
}
}

```

## REFERENCES

- Allison D. B., Cui X., Page G. P. and Sabripour M. (2006) Microarray data analysis: from disarray to consolidation and consensus. *Nature Reviews Genetics*, volume 7, pages 55-65.
- Austin C. P. (2008) The completed human genome: implications in chemical biology. *Current Opinion in Chemical Biology*, volume 7, pages 511-515.
- Berry M. J. A. and Linoff G. (1997) Data mining techniques for marketing, sales, and customer support. John Wiley and Sons.
- Bishop C. M. (1995) Neural Networks for Pattern Recognition. Oxford University Press, Oxford.
- Breiman L., Friedman J.H., Stone R.A. and Olshen C.J. (1984) Classification and regression trees. Chapman and Hall, New York.
- Brideau C., Gunter B., Pikounis W., Pajni N. and Liaw A. (2003) Improved statistical methods for hit selection in HTS. *Journal of Biomolecular Screening*, volume 8, pages 634-647.
- Briem H. and Günther J. (2005) Classifying "Kinase Inhibitor-Likeness" by using machine-learning methods. *European Journal of Chemical Biology ChemBioChem*, volume 6, pages 558-566.
- Broach J. R., Thorner J. (1996) High-throughput screening for drug discovery. *Nature*, volume 384, pages 14-6.
- Burton J., Ijjaali I., Barberan O., Petitet F., Vercauteren D. P. and Michel A. (2006) Recursive partitioning for the prediction of cytochromes P450 2D6 and 1A2 inhibition: importance of the quality of the dataset. *Journal of Medicinal Chemistry*, volume 49, pages 6231-6240.
- Carnero A. (2006) High throughput screening in drug discovery. *Clinical and Translational Oncology*, volume 8(7), pages 482-490.
- Carralot J.P., Ogier A., Boese A., Genovesio A., Brodin P., Sommer P. and Dorval T. (2012) A novel specific edge effect correction method for RNA interference screenings. *Bioinformatics*, volume 28, pages 261-268.
- Cheneau E., Wolfram R., Leborgne L. and Waksman R. (2003) Understanding and preventing the edge effect. *Journal of Interventional Cardiology*, volume 16(1), pages 1-7.

- Cohen J. (1960) A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, volume 20, pages 37–46.
- Coma I., Clark L., Diez E., Harper G., Herranz J., Hofmann G., Lennon M., Richmond N., Valmaseda M. and Macarron R. (2009) Process Validation and Screen Reproducibility in High-Throughput Screening. *Journal of Biomolecular Screening*, volume 14(1), pages 66-76.
- Cooley J.W. and Tukey J.W. (1965) An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, volume 19, pages 297-301.
- D'Agostino R. and Stephens M. (1986) Goodness-of-Fit Techniques. Marcel Dekker, Inc.
- Demuth H., Beale M. and Hagan M. (2005) Neural networks toolbox TM 6. User's guide.
- Dove A. (2003) Screening for content – the evolution of high throughput. *Nature Biotechnology*, volume 21, pages 859-864.
- Dragiev P., Nadon R. and Makarenkov V. (2011) Systematic error detection in experimental high-throughput screening. *BMC Bioinformatics*, volume 12, pages 25.
- Dragiev P., Nadon R. and Makarenkov V. (2012) Two effective methods for correcting experimental high-throughput screening data. *Bioinformatics*, 28 (13), 1775–1782.
- Dragiev P., Makarenkov V., Mballo C. and Nadon R. (2012) Statistical Methods for the Analysis of Experimental High-Throughput Screening Data. *Advances in Data Analysis and Classification*, Springer, submitted.
- Elowe N.H., Blanchard J.E., Cechetto J.D., Brown E.D. (2005) Experimental Screening of Dihydrofolate Reductase Yields a “Test Set” of 50,000 Small Molecules for a Computational Data-Mining and Docking Competition. *Journal of Biomolecular Screening*, volume 10, pages 653-657.
- Fang J., Dong Y., Lushington G. H., Ye Q. Z., Georg G. I. (2006) Support Vector Machines in HTS data mining: Type I MetAPs inhibition study. *Journal of Biomolecular Screening*, volume 11, pages 138-144
- Fawcett T. (2001) Using rule sets to maximize ROC performance. In: *Proceedings of the IEEE International Conference on Data Mining*, IEEE Computer Society, pages 131-138.
- Fink T. and Reymond J.-L. (2007) Virtual exploration of the chemical universe up to 11 atoms of C, N, O, F: assembly of 26.4 million structures (110.9 million stereoisomers) and analysis for new ring systems, stereochemistry, physicochemical properties, compound classes, and drug discovery. *Journal of Chemical Information and Modeling*, volume 47, pages 342–353.

- Fleiss J.L. (1981) Statistical methods for rates and proportions, 2<sup>nd</sup> edition John Wiley & Sons, Inc. New York.
- Gagarin A., Makarenkov V. and Zentilli P. (2006) Clustering techniques to improve the hit selection in HTS. *Journal of Biomolecular Screening*, volume 11, pages 903-914.
- Gunter B., Brideau C., Pikounis B., Pajni N. and Liaw A. (2003) Statistical and graphical methods for quality control determination of HTS data. *Journal of Biomolecular Screening*, volume 8, pages 624-633.
- Gurney K. (1997) An Introduction to Neural Networks. Routledge, London.
- Harper G. and Pickett S. (2006) Methods for mining HTS data. *Drug Discovery Today*, volume 11, pages 694-699
- Haykin S. (1999) Neural networks: a comprehensive foundation. 2<sup>nd</sup> edition, Prentice Hall.
- Helm J.S., Hu Y., Chen L., Gross B., Walker S. (2003) Identification of active-site inhibitors of MurG using a generalizable, high-throughput glycosyltransferase screen. *J. Am. Chem. Soc.*, volume 125, pages 11168-11169.
- Heuer C., Haenel T., Prause B. (2005) A novel approach for quality control and correction of HTS data based on artificial intelligence. *Pharmaceutical discovery & development report*. PharmaVentures.
- Heyse S. (2002) Comprehensive analysis of high-throughput screening data. In: *Proc. of SPIE 2002*, volume 4626, pages 535-547.
- Houston J. G., Banks M. N., Binnie A., Brenner S., O'Connell J., Petrillo E. W. (2008) Case study: impact of technology investment on lead discovery at Bristol-Myers Squibb, 1998-2006. *Drug Discovery Today*, volume 13, pages 44-51.
- Hu T. and Sung S. Y. (2004) A trimmed mean approach to finding spatial outliers. *Journal of Intelligent Data Analysis*, volume 8(1), pages 79-95.
- Iredale R., Jones L., Gray J. and Deaville J. (2005) 'The edge effect': an exploratory study of some factors affecting referrals to cancer genetic services in rural Wales. *Health & place*, volume 11(3), pages 197-204.
- Iversen P. W., Eastwood B. J., Sittampalam G. S., Cox K. L. (2006) A comparison of assay performance measures in screening assays: signal window, Z'-factor, and assay variability ratio. *Journal of Biomolecular Screening*, volume 11, pages 247-252.
- Janzen W. P. and Bernasconi P. (2009) High throughput screening. Methods and protocols. 2<sup>nd</sup> edition. *Methods Mol Biol*, volume 565, pages v-vii.

- Jenkins J. L., Kao R. Y. and Shapiro R. (2003) Virtual screening to enrich hit lists from high-throughput screening: a case study on small-molecule inhibitors of angiogenin. *Proteins* volume 50(1), pages 81-93.
- Kaiser J. (2008) Industrial-style screening meets academic biology. *Science*, volume 321, pages 764–766.
- Kaul A. (2005) The impact of sophisticated data analysis on the drug discovery process. *Business briefing: future drug discovery*.
- Kelley B.P. (2005) Automated Detection of Systematic Errors in Array Experiments. *Software Report*. Datect Inc., Santa Fe.
- Kevorkov D. and Makarenkov V. (2005) Statistical analysis of systematic errors in HTS. *Journal of Biomolecular Screening*, volume 10, pages 557-567.
- Lee M. L., Kuo F. C., Whitmore G. A. and Sklar J. (2000) Importance of replication in microarray gene expression studies: statistical methods and evidence from repetitive cDNA hybridizations. *Proceedings of the National Academy of Sciences of USA*. Volume 97, pages 9834–9839.
- Legendre P., Legendre L. (1998) Numerical ecology. 2<sup>nd</sup> English edition, Amsterdam: Elsevier Science BV, 739-746.
- Lundholt B. K., Scudder K. M. and Pagliaro L. (2003) A Simple Technique for Reducing Edge Effect in Cell-Based Assays. *Journal of Biomolecular Screening*, volume 8(5), pages 566-570.
- Macarron R. (2006) Critical review of the role of HTS in drug discovery. *Drug Discovery Today*, volume 11, pages 277-279.
- Macarron, R., Banks M., Bojanic D., Burns D. J., Cirovic D. A., Garyantes T., Green D. V., Hertzberg R. P., Janzen W. P., Paslay J. W., Schopfer U. and Sittampalam G. S. (2011). Impact of high-throughput screening in biomedical research. *Nature Reviews Drug Discovery*, volume 10(3), pages 188-195.
- Makarenkov V., Dragiev P. and Nadon R. (2012) A new effective method for elimination of systematic error in experimental high-throughput screening. In: *Proceedings of IFCS 2011 Studies in Classification, Data Analysis, and Knowledge Organization*. Springer Verlag, to appear.
- Makarenkov V., Kevorkov D., Zentilli P., Gagarin A., Malo N. and Nadon R. (2006) HTS-Corrector: new application for statistical analysis and correction of experimental data. *Bioinformatics*, volume 22, pages 1408-1409.

- Makarenkov V., Legendre P. (2002) Nonlinear redundancy analysis and canonical correspondence analysis based on polynomial regression. *Ecology*, volume 83, pages 1146-1161.
- Makarenkov V., Zentilli P., Kevorkov D., Gagarin A., Malo N. and Nadon R. (2007) An efficient method for the detection and elimination of systematic error in high-throughput screening. *Bioinformatics*, volume 23, pages 1648-1657.
- Malo N., Hanley J., Carlile G., Jing L., Pelletier J., Thomas D. and Nadon R. (2010) Experimental design and statistical methods for improved hit detection in high-throughput screening. *Journal of Biomolecular Screening*, volume 15, pages 990-1000.
- Malo N., Hanley J. A., Cerquozzi S., Pelletier J. and Nadon R. (2006) Statistical practice in high-throughput screening data. *Nature Biotechnology*, volume 24, pages 167-175.
- Mayr L. M. and Fuerst P. (2008) The future of high-throughput screening. *Journal of Biomolecular Screening*, volume 13(6), pages 443-448.
- Mballo C. and Makarenkov V. (2010) Using machine learning methods to predict experimental high-throughput screening data, *Combinatorial Chemistry & High Throughput Screening*, volume 13, pages 430-441.
- MLP (2012) Molecular Libraries Program. MLPCN probes Web table. <http://mli.nih.gov/mli/mlp-probes/>.
- Müller K. R., Rätsch G., Sonnenburg S., Mika S., Grimm M. and Heinrich N. (2005) Classifying "drug-likeness" with kernel-based learning methods. *Journal of Chemical Information and Modeling*, volume 45, pages 249-253.
- Nadon R. and Shoemaker J. (2002) Statistical issues with microarrays: processing and analysis. *Trends in Genetics*, volume 18, pages 265-271.
- Nelson R. and Yingling J. (2004) Introduction to High-Throughput Screening for Drug Discovery, *IBC USA Conferences Inc.*, San Diego, CA.
- Nelson R. M., Yingling J. D., Lundholt B. K., Scudder, K. M. and Pagliaro, L. (2004) A simple technique for reducing edge effect in cell-based assays. *Introduction to High-Throughput Screening for Drug Discovery*. volume 8(5), pages 566-570.
- Pereira D. A., Williams J. A. (2007) Origin and evolution of high throughput screening. *British Journal of Pharmacology*, volume 152, pages 53-61.
- Plewczynski D., Von G. M., Spieser S. A., Rychlewski L., Wyrwicz L. S., Ginalski K. and Koch U. (2007) Target specific compounds identification using a support vector machine. *Combinatorial Chemistry & High Throughput Screening*, volume 10, pages 189-196

- Ripley B. (1996) *Pattern Recognition and Neural Networks*, Cambridge.
- Root D. E., Kelley B. P., Stockwell B. R. (2003) Detecting spatial patterns in biological array experiments. *Journal of Biomolecular Screening*, volume 8, pages 393-398.
- Shun T. Y., Lazo J. S., Sharlow E. R and Johnston P. A. (2011) Identifying actives from HTS data sets: practical approaches for the selection of an appropriate HTS data processing method and quality control review. *Journal of Biomolecular Screening*, volume 16, pages 1-14.
- Silber B. M. (2010) Driving drug discovery: the fundamental role of academic labs. *Science Translational Medicine*, volume 2, pages 30.
- Simmons K., Kinney J., Owens A., Kleier D., Bloch K., Argentar D., Walsh A. and Vaidyanathan G. (2008a) Comparative study of machine learning and chemometric tools for analysis of in-vivo high throughput screening data. *Journal of Chemical Information and Modeling*, volume 48, pages 1663-1668.
- Simmons K., Kinney J., Owens A., Kleier D., Bloch K., Argentar D., Walsh A., Vaidyanathan G. (2008b) Practical Outcomes of Applying Ensemble Machine Learning Classifiers to High-Throughput Screening (HTS) Data Analysis and Screening. *Journal of Chemical Information and Modeling*, volume 48, pages 2196-2206.
- Sirois S., Hatzakis G., Wei D., Du Q. and Chou K.C. (2005) Assessment of chemical libraries for their druggability. *Computational Biology and Chemistry*, volume 29, pages 55-67.
- SLAS (2012) Society for Laboratory Automation and Screening. Screening facilities. <http://www.slas.org/screeningFacilities/facilityList.cfm>.
- Stein R. L. (2003) High-throughput screening in academia: The Harvard experience. *Journal of Biomolecular Screening*, volume 8(6), pages 615-619.
- Sui Y. and Wu Z. (2007) Alternative statistical parameter for high-throughput screening assay quality assessment. *Journal of Biomolecular Screening*, volume 12(2), pages 229-234.
- Taylor P. B., Ashman S., Baddeley S. M., Bartram S. L., Battle C. D., Bond B. C., Clements Y. M., Gaul N. J., McAllister W. E., Mostacero J. A., Ramon F., Wilson J. M., Hertzberg R. P., Pope A. J. and Macarron R. (2002) A Standard Operating Procedure for Assessing Liquid Handler Performance in High-Throughput Screening. *Journal of Biomolecular Screening*, volume 7(6), pages 554-569.
- Tukey J. W. (1977) *Exploratory Data Analysis*, Reading Massachusetts: Addison-Wesley.
- Verkman A. S. (2004) Drug discovery in academia. *American Journal of Physiology - Cell Physiology*, volume 286, pages C465-C474.

- Woodroffe and Ginsberg (1998) Edge effects and the extinction of populations inside protected areas. *Science*, volume 280, pages 2126-2128.
- Wu Z., Liu D. and Sui Y. (2008) Quantitative Assessment of Hit Detection and Confirmation in Single and Duplicate High-Throughput Screenings. *Journal of Biomolecular Screening*, volume 13(2), pages 159-167.
- Zhang J. H., Chung T. D. Y. and Oldenburg K. R. (1999) A simple statistic parameter for use in evaluation and validation of HTS assays. *Journal of Biomolecular Screening*, volume 4, pages 67-73.
- Zhang J. H., Chung T. D. Y. and Oldenburg K. R. (2000) Confirmation of primary active substances from HTS of chemical and biological populations: a statistical approach and practical considerations. *Journal of Combinatorial Chemistry*, volume 2, pages 258-265.
- Zhang, X. D. (2008) Novel analytic criteria and effective plate designs for quality control in genome-scale RNAi screens. *Journal of Biomolecular Screening*, volume 13, pages 363-377.