UniL Master Course
“Case studies in bioinformatics”

Module 2 “, "How to make valid prognostic models with gene expression signatures? ?”

Mauro.Delorenzi @ unil.ch

Ludwig Center for Cancer Research
Dpt of Oncology
University of Lausanne

SIB Swiss Institute of Bioinformatics

UNIL | Université de Lausanne

SIB Swiss Institute of Bioinformatics

BCF

SCHEDULE

Monday 14 Nov 2015, 13:15-16:00 (BIO/1929)

Monday 21 Nov 2015, 9:00-12:00 (BIO/1929)

Monday 21 Nov 2015, 13:15-16:00 (BIO/1929)

REPORTING

Select a module => hand-in a report => pass / score

10 students, 4 modules : 2 modules * 2 students + 2m * 3s

What do have in the report ?

REPORTING-2

Report should include:

- A **summary** of essential points in the slides, to introduce / understand what was done
- The **results** that were obtained during analyses in R (tables, figures) with explanatory text and comments including conclusions learned from the results (especially the results of the **own work session** at the end)
- **Appendix** with code that you have used for generating the results and that can be executed
- **Discussion**: insights gained, open questions that could be further investigated

Report should be handed-in by mail as a file, formatted text (Open office, Word, latex-pdf), appendix with code as flat text or Rstudio file.
The report is personal, each participant submits his/her own report.

JNCi commentary paper "Pitfalls" by Simon et al.

COMMENTARY

Pitfalls in the Use of DNA Microarray Data for Diagnostic and Prognostic Classification

Richard Simon, Michael D. Radmacher, Kevin Dobbin, Lisa M. McShane

DNA microarrays have made it possible to estimate the level of expression of thousands of genes for a sample of cells. Although biomedical investigators have been quick to adopt this powerful new research tool, accurate analysis and interpretation of the data have provided unique challenges. Indeed, many investigators are not experienced in the analytical steps needed to convert tens of thousands of noisy data points into reliable and interpretable biologic information. Although some investigators recognize the importance of collaborating with experienced biostatisticians to analyze microarray data, the number and availability of experienced biostatisticians is inadequate. Consequently, investigators are using available software to analyze their data, many seemingly without knowledge of potential pitfalls. Because of serious problems associated with the analysis and reporting of some DNA microarray studies, there is great interest in guidance on valid and effective methods for analysis of DNA microarray data.

The design and analysis strategy for a DNA microarray experiment should be determined in light of the overall objectives of the study. Because DNA microarrays are used for a wide variety of objectives, it is not feasible to address the entire range of design and analysis issues in this commentary. Here, we address statistical issues that arise from the use of DNA microarrays for an important group of objectives that has been called "class prediction" *(1)*. Class prediction includes derivation of predictors of prognosis, response to therapy, or any phenotype or genotype defined independently of the gene expression profile.

Experimental Objectives Drive Design and Analysis

Good DNA microarray experiments, although not based on gene-specific mechanistic hypotheses, should be planned and conducted with clear objectives. Three commonly encountered types of study objectives are "class comparison," "class prediction," and "class discovery" *(1)*.

Class comparison is the comparison of gene expression in different groups of specimens. The major characteristic of class comparison studies is that the classes being compared are defined independently of the expression profiles. The specific objectives of such a study are to determine whether the expression profiles are different between the classes and, if so, to identify the differentially expressed genes. One example of a class comparison study is the comparison of gene expression profiles of stage I breast cancer patients who are long-term survivors with the gene expression profiles of those who have recurrent disease. Another example is the comparison between gene expression profiles in breast cancer patients with and without germline BRCA1 mutations *(2)*.

Class prediction studies are similar to class comparison studies in that the classes are predefined. In class prediction studies, however, the emphasis is on developing a gene expression-based multivariate function (referred to as the predictor) that accurately predicts the class membership of a new sample on the basis of the expression levels of key genes. Such predictors can be used for many types of clinical management decisions, including risk assessment, diagnostic testing, prognostic stratification, and treatment selection. Many studies include both class comparison and class prediction objectives.

Class discovery is fundamentally different from class comparison or class prediction in that no classes are predefined. Usually the purpose of class discovery in cancer studies is to determine whether discrete subsets of a disease entity can be defined on the basis of gene expression profiles. This purpose is different from determining whether the gene expression profiles correlate with some already known diagnostic classification. Examples of class discovery are the studies by Bittner et al. *(3)* that examined gene expression profiles for advanced melanomas and by Alizadeh et al. *(4)* that examined the gene expression profiles of patients with diffuse large B-cell lymphoma. Often the purpose of class discovery is to identify clues regarding the heterogeneity of disease pathogenesis.

Limitations of Cluster Analysis for Class Prediction

One of the most common errors in the analysis of DNA microarray data is the use of cluster analysis and simple fold change statistics for problems of class comparison and class prediction. Although cluster analysis is appropriate for class discovery, it is often not effective for class comparison or class prediction. Cluster analysis refers to an extensive set of methods for partitioning samples into groups on the basis of the similarities and differences (referred to as distances) among their gene expression profiles. Because there are many ways of measuring distances among gene expression profiles involving thousands of genes and because there are many algorithms for partitioning, cluster analysis is a very subjective analysis strategy.

Cluster analysis is considered an unsupervised method of analysis because no information about sample grouping is used. The distance measures are generally computed with regard to the complete set of genes represented on the array that are measured with sufficiently high signals, or with regard to all the genes that show meaningful variation across the sample set. Because relatively few genes may distinguish any particular class, the distances used in cluster analysis will often not reflect the influence of these relevant genes. This feature accounts for the poor results often obtained in attempting to use cluster analysis for class prediction studies.

Cluster analysis also does not provide statistically valid quantitative information about which genes are differentially expressed between classes. Investigators often use simple average fold change measures or visual inspection of a cluster image display to identify differentially expressed genes. However, average fold change indices do not account for variability in gene expression across samples within the same class; some twofold average effects represent statistically significant differences and some do not. Neither fold change indices nor visual inspection of cluster image displays enable the investigator to deal with multiple comparison issues in a statistically valid manner. For example, in examining expression levels of thousands of randomly varying genes, there may be many genes that spuriously appear to be differentially expressed between two classes on the basis of visual inspection or fold change thresholds.

Class Prediction Using Supervised Methods

For class prediction studies, it is more appropriate to use a supervised method (i.e., one that makes distinctions among the specimens on the basis of predefined class label information) than an unsupervised method, such as cluster analysis. Supervised class prediction is usually based on the assumption that a collection of differentially expressed genes is associated with class distinction.

The first step toward constructing the class predictor (sometimes called the classifier) is to select the subset of informative genes. The second step is often to assign weights related to the individual predictive strengths of these informative genes. Predictors based on linear combinations of the weighted intensity measurements of the informative genes have been proposed *(1,5)*. One alternative method is to use a dimension reduction technique such as principal components analysis or partial least squares on the informative genes and to base the prediction on the resulting factors *(6–8)*. Many other methods for defining a multivariate predictor have been described *(9,10)*. The final step in constructing the classifier is to define the prediction rule. For example, in a two-group classification where a single predictor is computed, the classification rule may simply be a threshold value; a specimen is classified as being in one group if the derived predictor value is less than the threshold and classified as being in the other group if the derived predictor value is more than the threshold.

One major limitation of supervised methods is overfitting the predictor. Overfitting means that the number of parameters of the model is too large relative to the number of cases or specimens available. Because the model parameters are optimized for the data, the model will fit the original data but may predict poorly for independent data. This happens because the model fits random variations within the original data that do not represent true relationships that hold for independent data. Consequently, it is essential to obtain an unbiased estimate of the true error rate of the predictor (i.e., the probability of incorrectly classifying a randomly selected future case).

Methods for obtaining unbiased estimates of a predictor's error rate include leave-one-out cross-validation or application of the prediction rule developed from a supervised analysis of one dataset to an independent dataset. By using these techniques, it is possible not only to evaluate overfitting the predictor, but also to compare various prediction methods and assess which ones are less prone to overfitting. The appropriate use of leave-one-out cross-validation and validation of independent datasets is discussed in the next two sections.

Cross-Validation of Prediction Accuracy

The performance of a class prediction rule is best assessed by applying the rule created on one set of data (the training set) to an independent set of data (the validation set). Because most clinical research laboratories have access to only a limited number of tumor samples, withholding a substantial proportion of the samples from the training set for the sake of creating a validation set may considerably reduce the performance of the prediction rule. Cross-validation procedures use the data more efficiently. A small number of specimens are withheld, and most of the specimens are used to build a predictor. The predictor is used to predict class membership for the withheld specimens. This process is iterated, leaving out a new set of specimens at each step, until all specimens have been classified. In leave-one-out cross-validation, for example, each specimen is excluded from the training set one at a time and then classified on the basis of the predictor built from the data for all of the other specimens. The leave-one-out cross-validation procedure provides a nearly unbiased estimate of the true error rate of the classification procedure. The estimated error rate applies to the procedure used to build the classifier rather than to the specific prediction model based on all the data, because there is a different classifier for each leave-one-out training set *(11,12)*. Other cross-validation methods omit more than one specimen at a time *(13)* and also produce nearly unbiased estimates.

In the previous section, three common components of class prediction methods were listed: 1) selection of informative genes, 2) computation of weights for selected informative genes, and 3) creation of a prediction rule. It is important that all three steps undergo the cross-validation procedure. Failure to cross-validate all steps may lead to substantial bias in the estimated error rate.

We performed a simulation to examine the bias in estimated error rates for a class prediction study with various levels of cross-validation (*see* supplemental information at http://jncicancerspectrum.oupjournals.org/jnci/content/vol95/issue1/index.shtml and at http://linus.nci.nih.gov/~brb for a full description of the simulation). We considered two types of leave-one-out cross-validation: one with removal of the left-out specimen before selection of differentially expressed genes and one with removal of the left-out specimen after gene selection but before computation of gene weights and application of the prediction rule. We also computed the resubstitution estimate of the error rate (this estimate results from building the predictor on the full dataset and then reapplying it to each specimen for classification purposes). In each simulated dataset, 20 expression profiles of 6000 genes were randomly generated from the same distribution. Ten profiles were arbitrarily assigned to class 1 and the other 10 profiles to class 2, creating an artificial separation of the profiles into two classes. Because no true underlying difference existed between the two classes, the class prediction should perform no better than a random guess, with

Affiliations of authors: R. Simon, K. Dobbin, L. M. McShane, Biometric Research Branch, National Cancer Institute, National Institutes of Health, Bethesda, MD; M. D. Radmacher, Departments of Biology and Mathematics, Kenyon College, Gambier, OH.

Correspondence to: Richard Simon, D.Sc., National Cancer Institute, 9000 Rockville Pike, MSC 7434, Bethesda, MD 20892-7434 (e-mail: rsimon@nih.gov).

See "Note" following "References."

Student list 2016

Virginie Ricci <Virginie.Ricci@unil.ch>,

Nastassia Gobet <nastassia.gobet@unil.ch>,

Mirjam Mattei <mirjam.mattei@unil.ch>

Shaoline Sheppard <shaoline.sheppard@unil.ch>,

Titouan Laessle <Titouan.Laessle@unil.ch>,

Anthony Sonrel <Anthony.Sonrel@unil.ch>,

Jonas Garessus <Jonas.Garessus@unil.ch>,

Cyril Matthey-Doret <Cyril.Matthey-Doret@unil.ch>,

Joaquim Claivaz <Joaquim.Claivaz@unil.ch>,

Karim Hamidi <Karim.Hamidi@unil.ch>

How was the classifier is constructed?

- Find genes that discriminate between the groups
- Use a statistics that measures the difference in expression
- Rank genes, Take the top n

Feature Selection

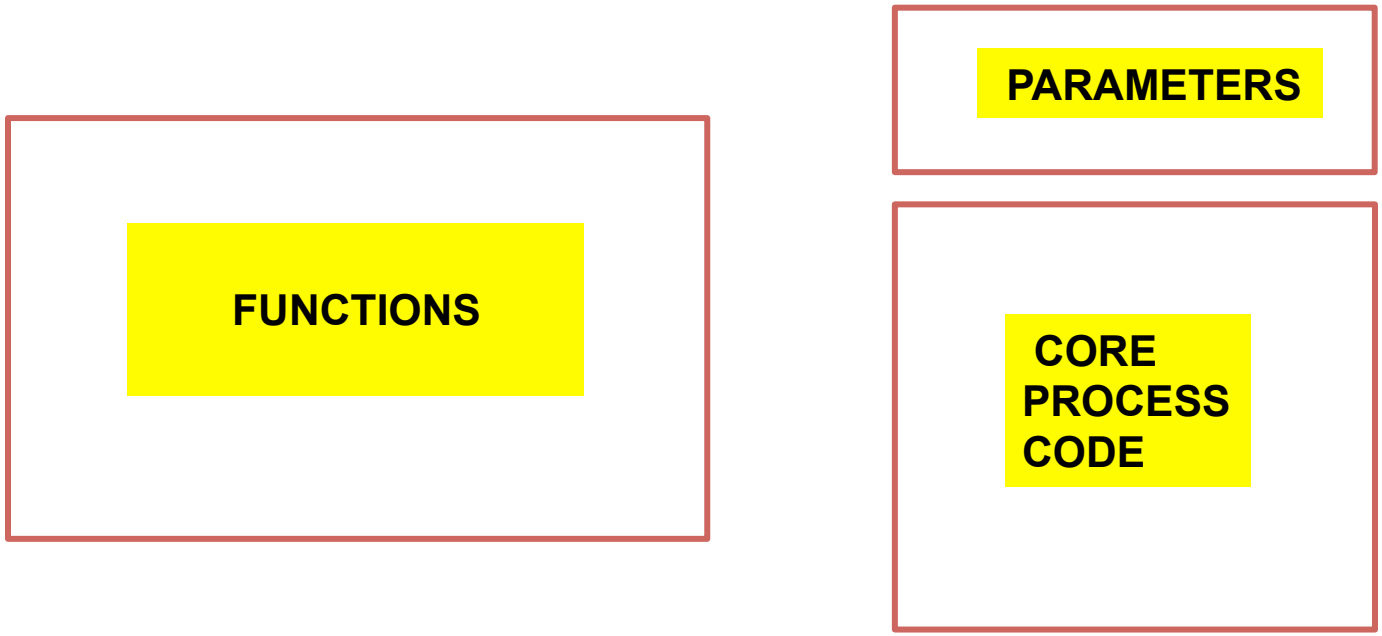
- Decide how to use the n genes to predict the group of a new sample:
- Coefficients
- Cutoff

Classifier

- Estimate how well the prediction works

Testing assess performance

CODE OVERVIEW

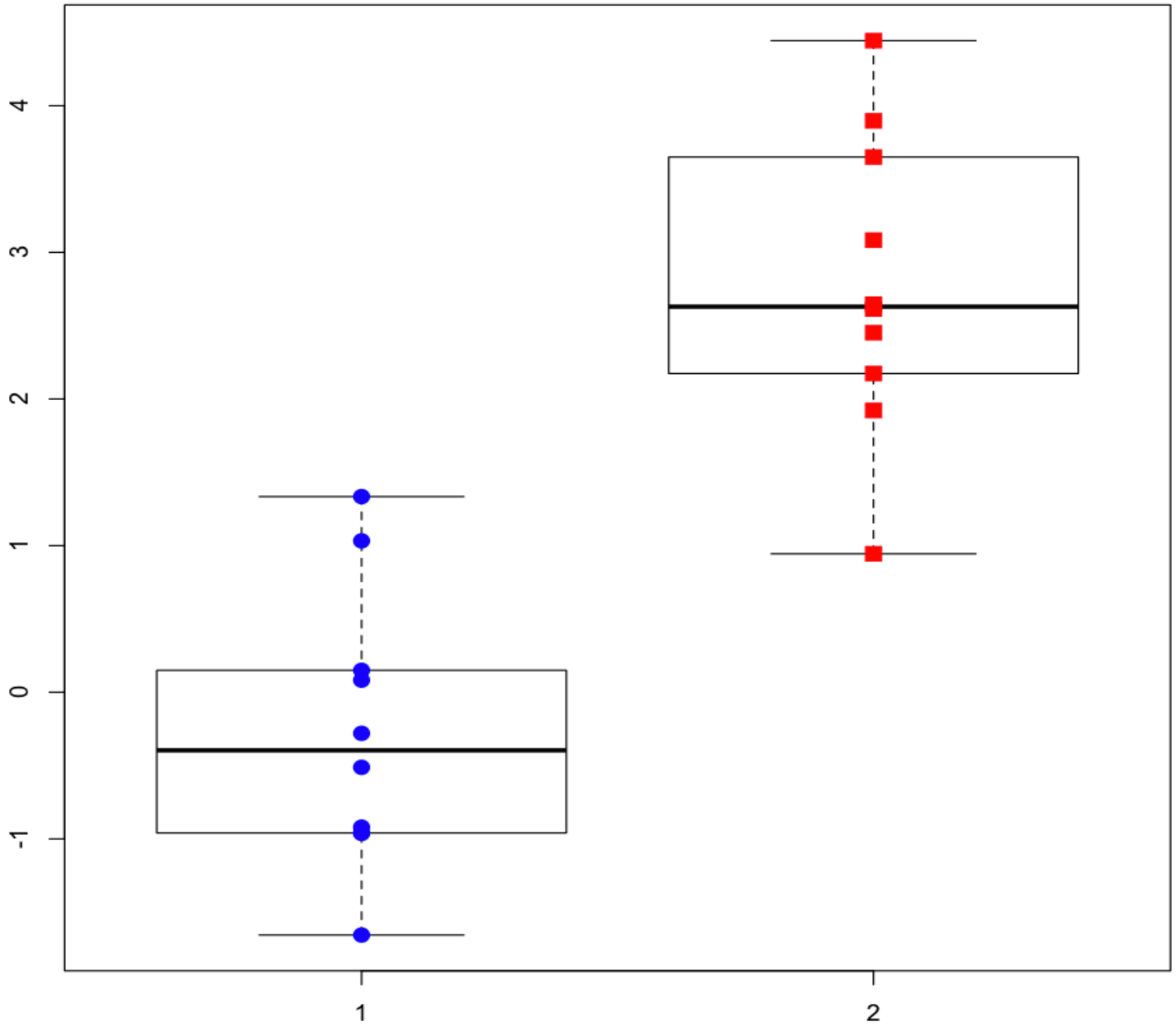


Simulation

- Scenario
  - 2 groups of each 10 patients
  - 2000 genes,
  - 1 informative (discriminative gene)

Simulation

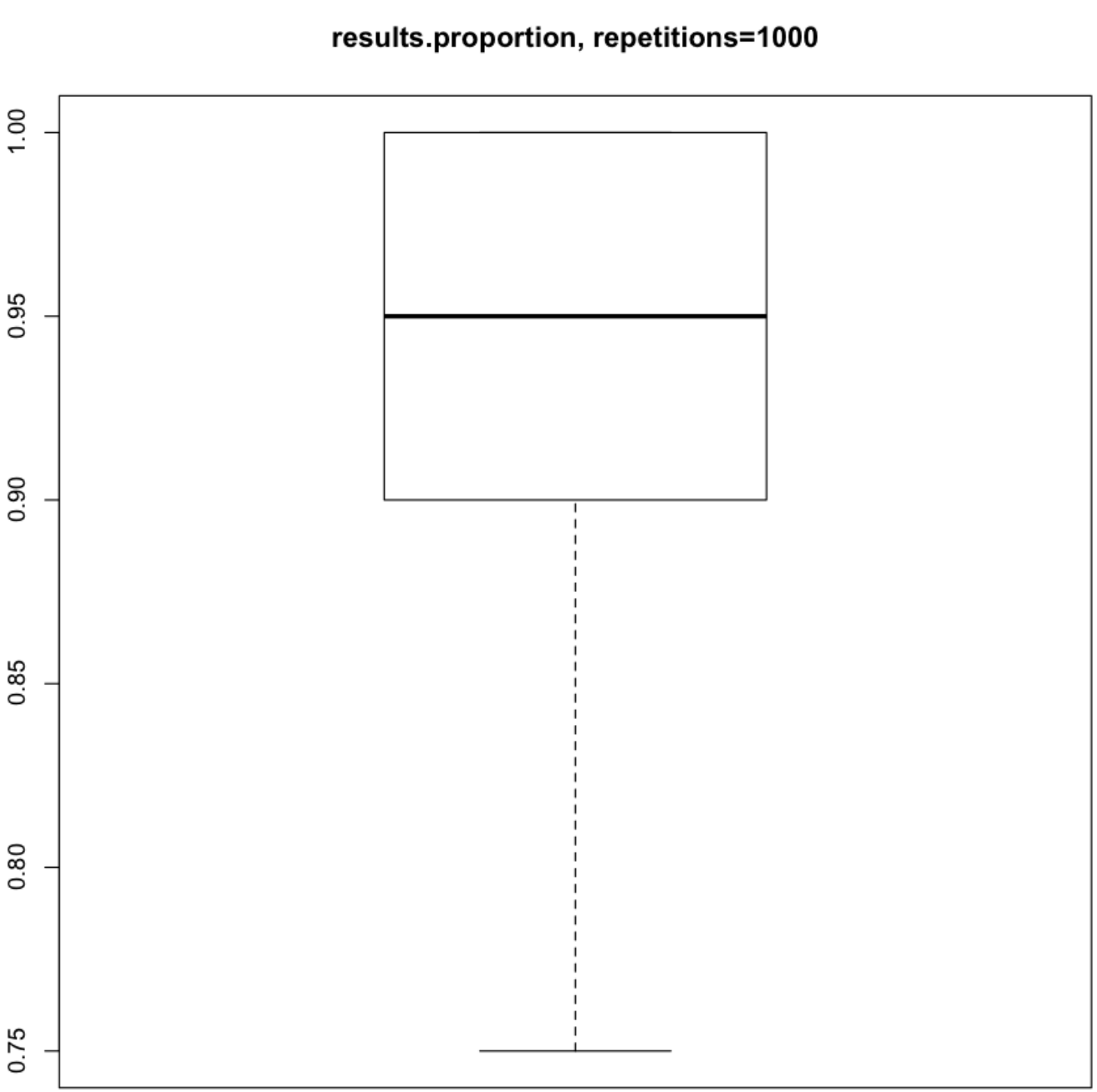
```
Ngroup <- 10
effect.mean <- 3
y0 <- rnorm(Ngroup , 0, 1)
y1 <- rnorm(Ngroup , effect.mean, 1)
tt <- t.test(y0, y1)
Tstat <- tt$statistic
cutpoint <- (mean(y0) + mean(y1)) / 2
scores <- Tstat * c(y0, y1)
```

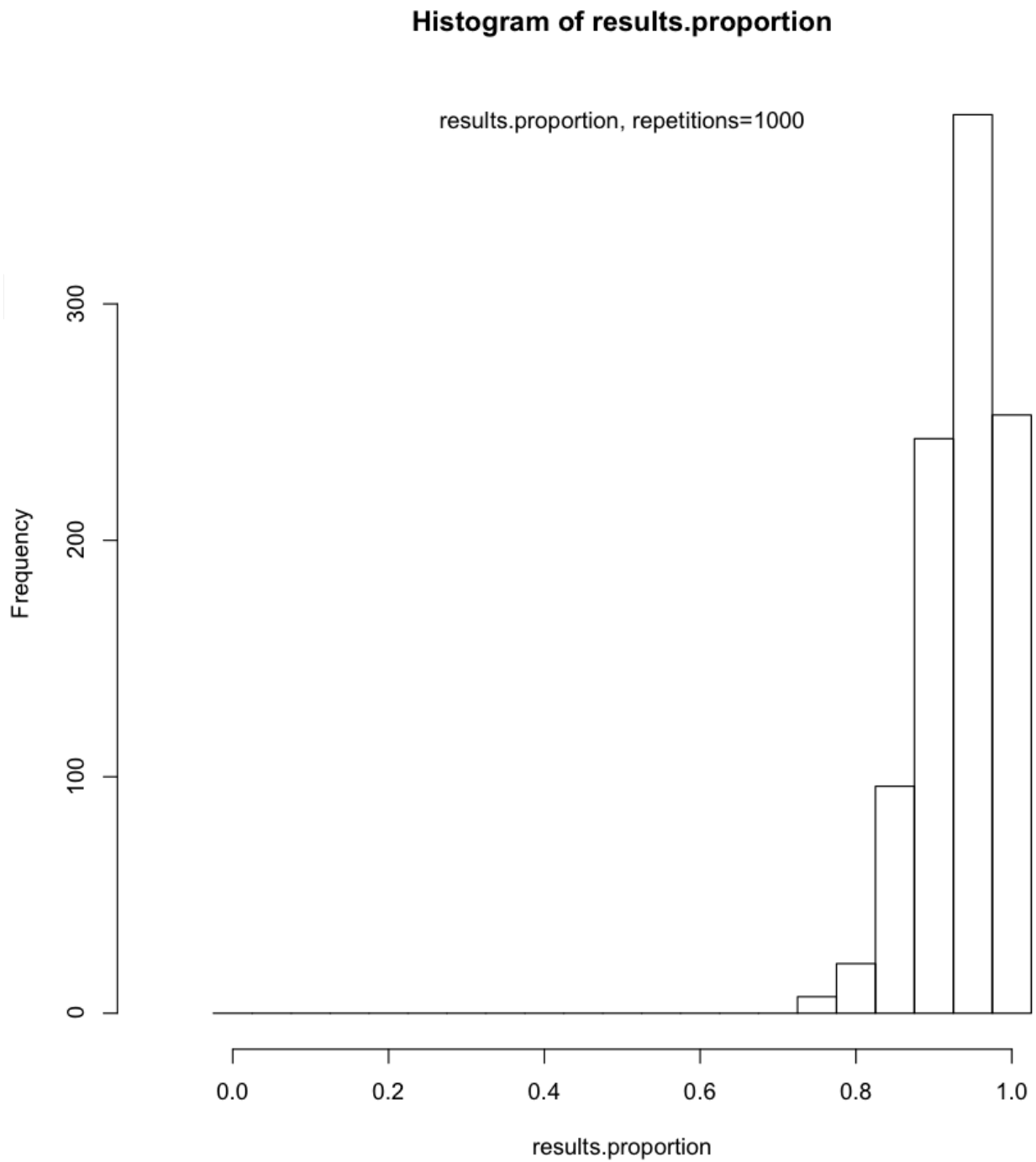

4
3
2
1
0
-1
1
2

Exercises

How un-**stable** is the correct classification rate across example datasets ?

How **accurate** is the correct classification rate obtained in the example ?


results.proportion, repetitions=1000
1.00
0.95
0.90
0.85
0.80
0.75


Histogram of results.proportion
results.proportion, repetitions=1000
Frequency
0
100
200
300
0.0
0.2
0.4
0.6
0.8
1.0
results.proportion

```
# EXAMPLE SOLUTION E1A

Ngroup <- 10; effect.mean <- 3
Y <- c( rep(0,Ngroup), rep(1,Ngroup) )
repetitions <- 1000
# do the simulation
SEED <- 444
for (i in  c(1:repetitions) )
{
SEED <- SEED+1; set.seed(SEED);
y0 <- rnorm(Ngroup , 0, 1)
y1 <- rnorm(Ngroup , effect.mean, 1)
y <- c(y0,y1);
# how strong the separation ? How to visualize ?
tt <- t.test(y0, y1)
Tstat <- tt$statistic
scores <- Tstat * y
cutpoint <- (mean(scores[Y==0]) + mean(scores[Y==1])) / 2
pred <- sign(scores < cutpoint)
correct <- sum(pred == Y)
results[i] <- correct
if ( (i %% 100) == 0)  {cat("\n i=",i)}
}
```

```
# EXAMPLE SOLUTION E1A

# analyze the simulation results

results.proportion <- results / (2*Ngroup)

summary(results.proportion )
table(results.proportion )
boxplot(results.proportion)
hist(results.proportion, xlim=c(-0.1,1.1), breaks=-0.025+c(0:21)*0.05 )

Myname <- "XY”
pdf(paste (Myname , "exercise1b.pdf"))
hist(results.proportion, breaks=-0.025+c(0:21)*0.05, xlim=c(-0.1,1.1)  )
dev.off()
```

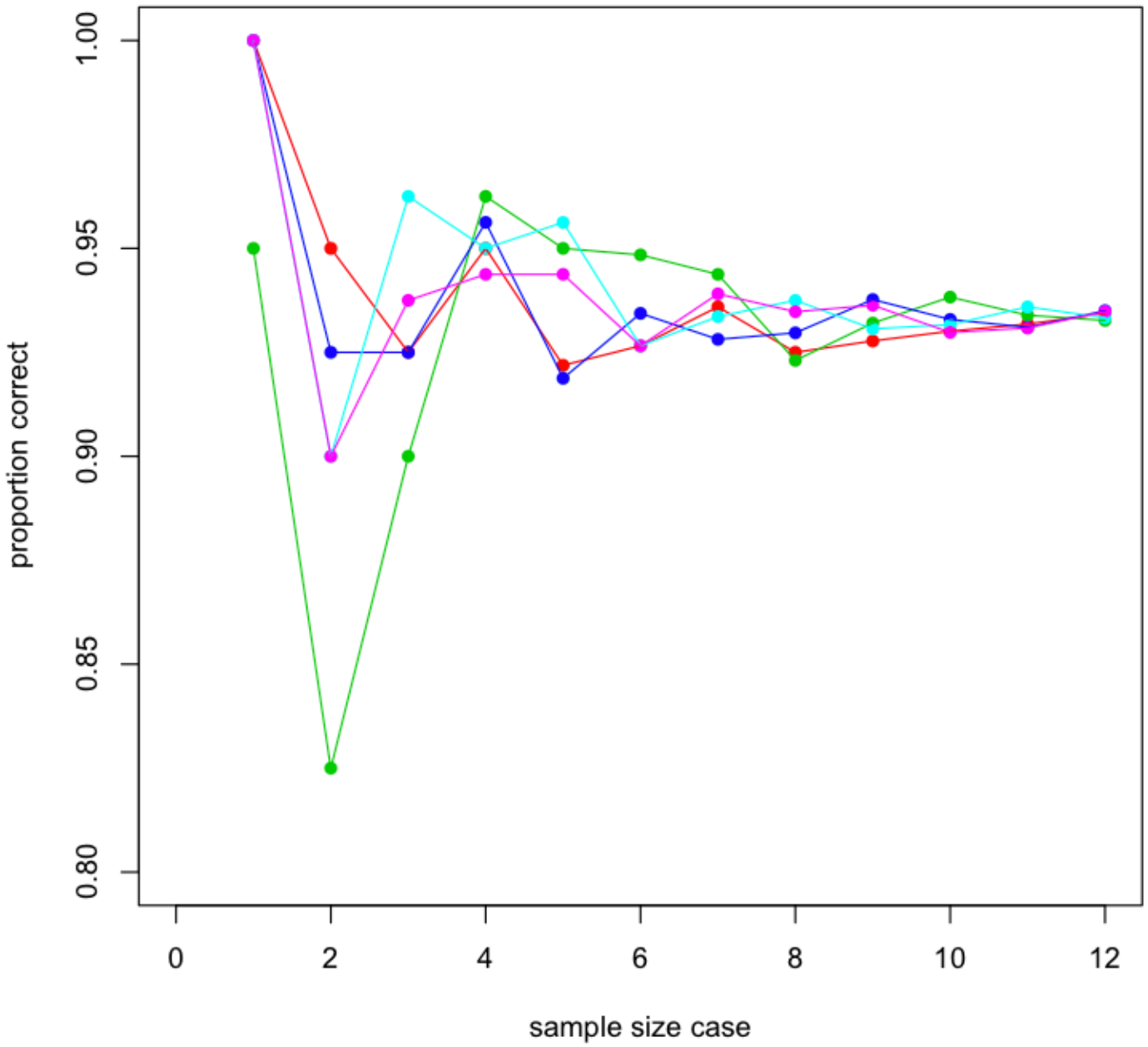
```
# write a function to simplify the loop etc.

determine.perf.fu <- function(localSEED, Ngroup=10, effect.mean=3)
{
set.seed(localSEED);
Y <- c( rep(0,Ngroup), rep(1,Ngroup) );
y0 <- rnorm(Ngroup , 0, 1); y1 <- rnorm(Ngroup , effect.mean, 1); y <- c(y0,y1);
# how strong the separation ? How to visualize ?
tt <- t.test(y0, y1); Tstat <- tt$statistic
scores <- Tstat * y;
cutpoint <- (mean(scores[Y==0]) + mean(scores[Y==1])) / 2
pred <- sign(scores < cutpoint)
correct <- sum(pred == Y)
return(correct)
}

# do the simulation
SEED <- 444
for (i in  c(1:repetitions) )  {
SEED <- SEED+1;
results2[i]  <- determine.perf.fu (SEED)
}
```

```
# EXAMPLE SOLUTION E2a

# MODIFY THE PREVIOUS FUNCTION
determine.perf.fu2 <- function(localSEED, Ngroup=10, effect.mean=3)
{
set.seed(localSEED);
Y <- c( rep(0,Ngroup), rep(1,Ngroup) );
y0 <- rnorm(Ngroup , 0, 1); y1 <- rnorm(Ngroup , effect.mean, 1); y <- c(y0,y1);
# how strong the separation ? How to visualize ?
tt <- t.test(y0, y1); Tstat <- tt$statistic
scores <- Tstat * y;
cutpoint <- (mean(scores[Y==0]) + mean(scores[Y==1])) / 2
pred <- sign(scores < cutpoint)
correct <- sum(pred == Y)
return(correct / (2*Ngroup) )
}
```

 # now with more genes say 2'000 and generalize further
# of which most or all will not be different btw the two groups

GENERATE ARTIFICIAL DATA , first random and equal for groups then add a shift for informative genes

```
# parameters
Ngroup <- 10; Nlearn <- 2*Ngroup;
Y <- c( rep(0,Ngroup), rep(1,Ngroup) )
Ngenes <- 2000;

# VALUE DISTRIBUTION
mean.x <- 0; sd.x <- 1
effect.mean <- 3
```


proportion correct
1.00
0.95
0.90
0.85
0.80
0
2
4
6
8
10
12
sample size case

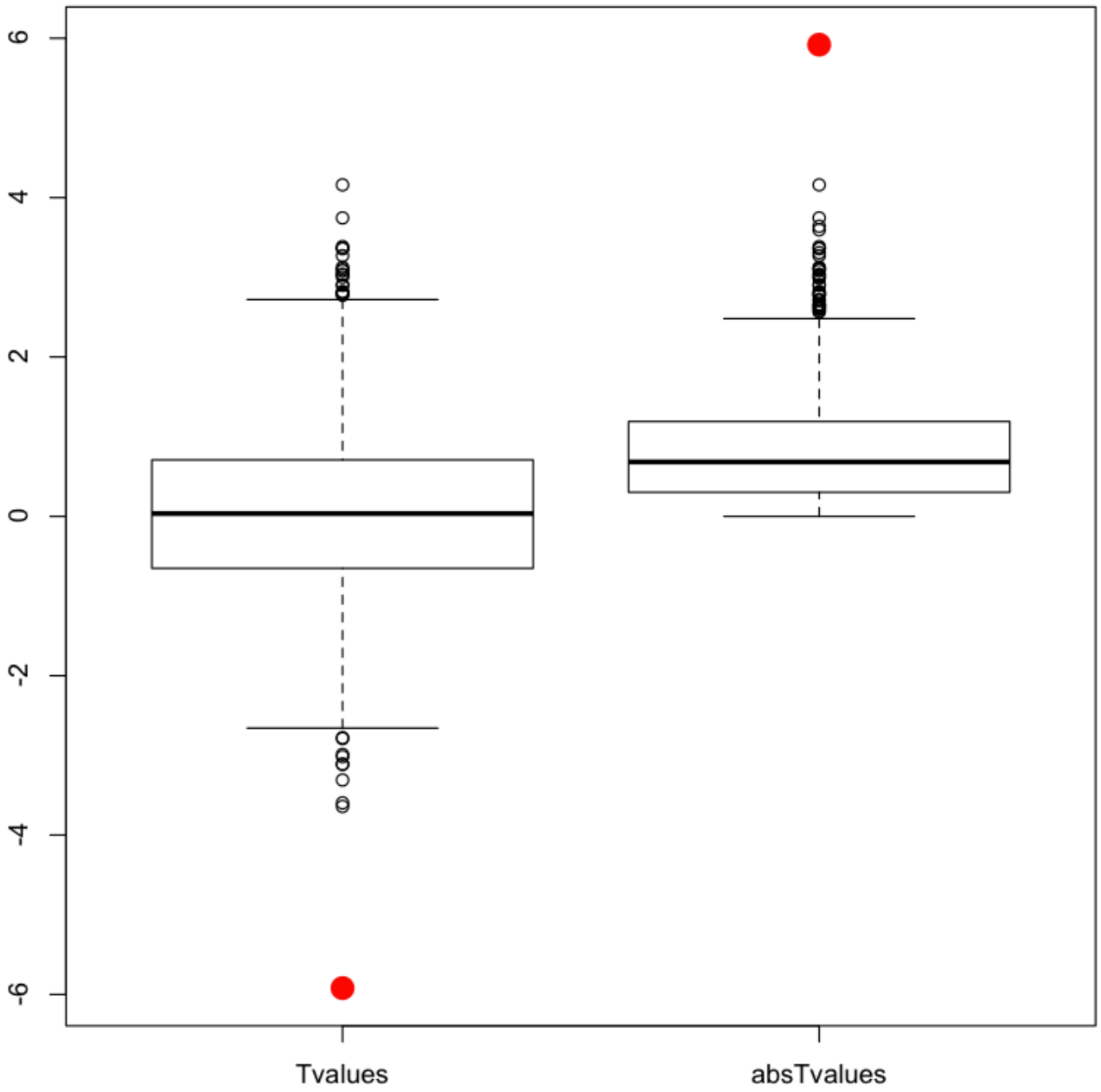
```
# SIMULATED GENE EXPRESSION VALUES IN A MATRIX X
# variables /FEATURES IN COLUMNS, observations / SAMPLES IN ROWS

XS <- matrix( rnorm(( Ngenes * Nlearn), mean=mean.x, sd=sd.x), ncol= Ngenes,
nrow= Nlearn)


#  GENERATE CLASS LABELS
(JUST FOR PRINCIPLE: RANDOM ORDER LABELS AS IN PAPER)

YS <- rep(1,Nlearn);
group1 <- sample(x=1:Nlearn, Ngroup)
YS[group1] <- 0
table(YS)
```

```
# LEARNING - SELECTION OF FEATURES AND COEFFICIENTS FOR THE CLASSIFIER
# SELECTION OF FEATURES : top genes in a t-test


Tvalues <- apply(X=XS, MARGIN=2, FUN=testfunction.t, Z=YS)

# where YS are the group identifiers and testfunction.t is a function we write
# and that returns the t-value for each gene
# ***EXERCISE E3:  WRITE THE FUNCTION testfunction.t ****


boxplot(Tvalues, abs(Tvalues) )
```

```
# ADD DISCRIMINATIVE GENE

effect.group <- 1
effect.mean <- 3

# for simplicity take always the gene "1"
sel <- (YS == effect.group);
XS[sel, 1 ] <- XS[sel, 1 ] + effect.mean;

Tvalues <- apply(X=XS, MARGIN=2, FUN=testfunction.t, Z=YS)

boxplot( list(Tvalues=Tvalues, absTvalues=abs(Tvalues) ) )
points(1, Tvalues[1], pch=16, col="red", cex=2)
points(2, abs(Tvalues[1]), pch=16, col="red", cex=2)
```

 # EXAMPLE SOLUTION E3

```
testfunction.t <- function(u, Z)
# receiving from caller two parameters, a vector of values u and a vector of labels Z
# these can be provided by "apply", the first by its main mechanism, the second as a named parameter
{
# this can be run for each gene in a matrix using the function call inside the apply function
u1 <- u[Z==0]     #  values for group "0"
u2 <- u[Z==1]       #  values for group "1"
st <- t.test (x=u1, y=u2)  #  object returned from the t-test function, positive when mean u1 HIGHER
return (st$statistic)  #  return to caller the t-stat value from the object st
}
```


6
4
2
0
-2
-4
-6
Tvalues
absTvalues

*EXERCISE **E4a**: GENERALIZE THE CODE FOR MORE THAN ONE GENE

# ***EXERCISE **E4b**: WRITE A FUNCTION FOR DATA GENERATION ****
# that returns generated data as a list , ans is called like this:
data <- **datageneration.function**(localSEED=SEED, Ngroup=Ngroup, Ngenes=NBclassifierFeatures, meanvalue=mean.x , sdvalue=sd.x , effect=effect.mean)

# ***EXERCISE **E4c**: WRITE A FUNCTION FOR CONSTRUCTION OF THE CLASSIFIER ****
# that returns a classifier as a list , ans is called like this:
classifier <- **classifier.function (**mydata, NBgenes.selected, NBobservations)
# where the data is a list returned by the datageneration.function of E4b

EXERCISE **E4d**: PUT THINGS TOGETHER TO COMPUTE CORRECT CLASSIFICATION PROPORTION *
test it for the case that the classifier uses two genes

SCHEDULE

WORK ON EXERCISES

```
# EXAMPLE SOLUTION E4a


NBgenes.selected <- 2

selgenes <- (order(abs(Tvalues), decreasing = TRUE))[1:NBgenes.selected ]

model.coefficients <- Tvalues[selgenes]

scores <-  XS[, selgenes]  %*%  model.coefficients
```

 # EXAMPLE SOLUTION E4b

```
datageneration.function <-  function (localSEED, Ngroup=10, Ngenes=2000,
meanvalue=0, sdvalue=1, effect=3, shiftgroup=1)
{
set.seed(localSEED)
Nlearn <- 2*Ngroup;
XS <- matrix( rnorm(( Ngenes * Nlearn), mean=meanvalue, sd=sdvalue), ncol=
Ngenes, nrow= Nlearn)
XS[sel, 1 ] <- XS[sel, 1 ] + effect.mean;
# remark: always only the values of gene 1 are shifted, to easy keep track or
reproduce in new data
YS <- rep(1,Nlearn); group1 <- sample(x=1:Nlearn, Ngroup); YS[group1] <- 0
datalist <- list (X=XS, Y=YS)
return (datalist)
}
```

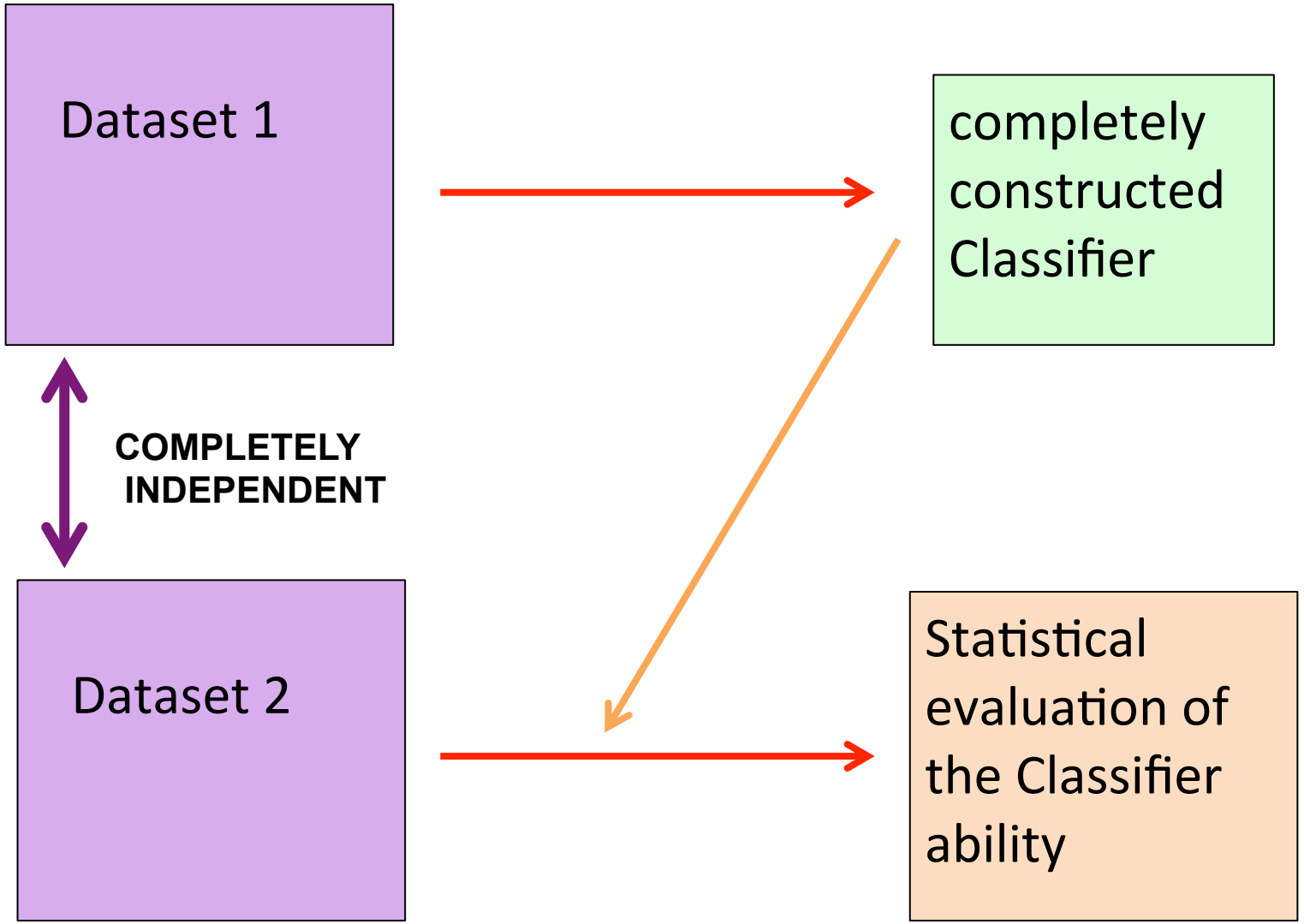
 # EXAMPLE SOLUTION E4c

```
classifier.function <-  function (mydata, NBgenes, NBobservations)
{
Tvalues <- apply(X=mydata$X, MARGIN=2, FUN=testfunction.t, Z=mydata$Y)
selgenes <- (order(abs(Tvalues), decreasing = TRUE))[1:NBgenes.selected ]
model.coefficients <- Tvalues[selgenes]
scores <- mydata$X[, selgenes] %*% model.coefficients
if(NBgenes.selected > 1) {scores <- mydata$X[, selgenes] %*%
model.coefficients}
      else {scores <- mydata$X[, selgenes] * model.coefficients}
cutpoint <- ( mean( scores[mydata$Y==0] ) + mean( scores[mydata$Y==1] ) ) /  2
pred <- sign(scores < cutpoint)
corr.prop <- sum(pred == mydata$Y) / NBobservations
classifier <- list(features=selgenes, coefficients=model.coefficients,
cutoff=cutpoint, predicted=pred, correct.proportion=corr.prop)
return (classifier)
}
```

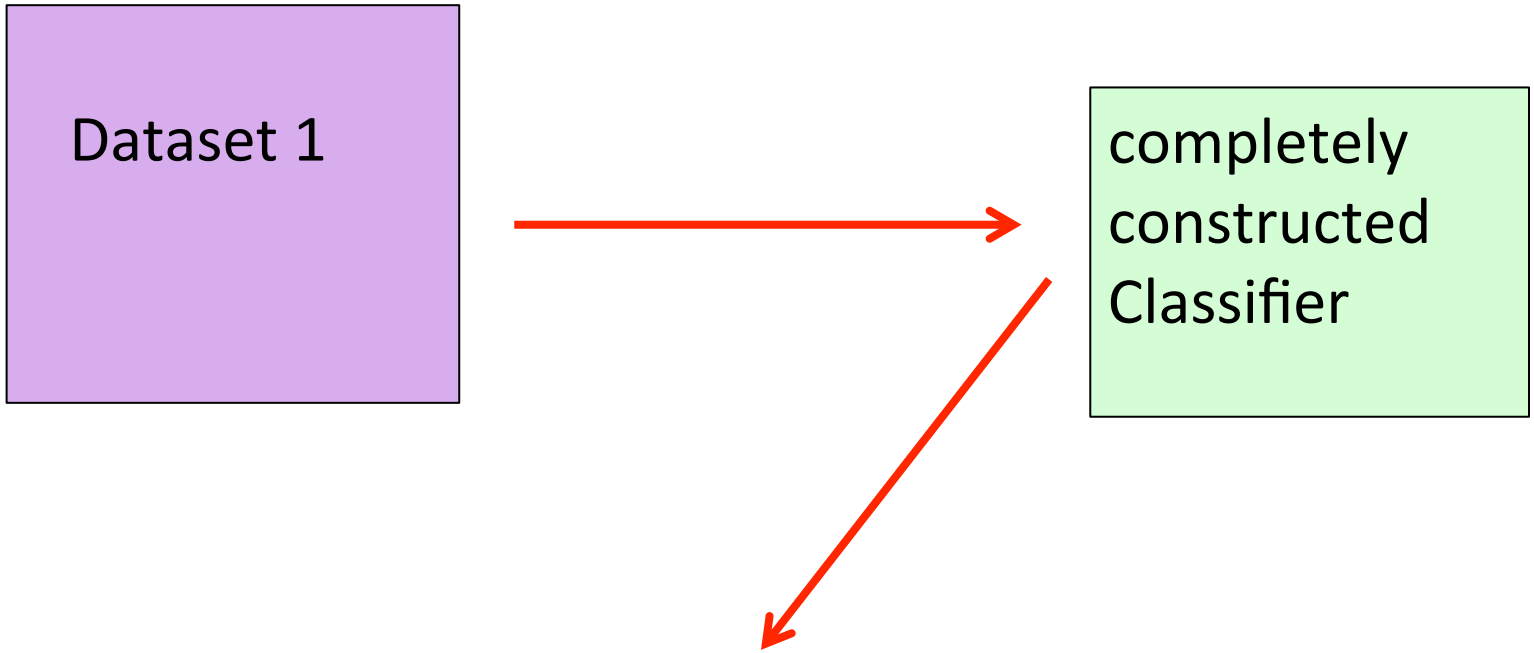
 # EXAMPLE SOLUTION E4d

```
# parameters
Ngroup <- 10;
NBgenesMeasured <- 2000; # genes in data
mean.x <- 0; sd.x <- 1
effect.mean <- 3
SEED <- 444
NBclassifierFeatures <- 2 # 1

data <- datageneration.function(localSEED=SEED, Ngroup=Ngroup,
Ngenes=NBgenesMeasured, meanvalue=mean.x, sdvalue=sd.x,
effect=effect.mean )
str(data)

classifier <- classifier.function(mydata=data, NBgenes=NBclassifierFeatures,
NBobservations=2*Ngroup)
str(classifier)

classifier$correct.proportion
```

Use of data – “split-set” strategy



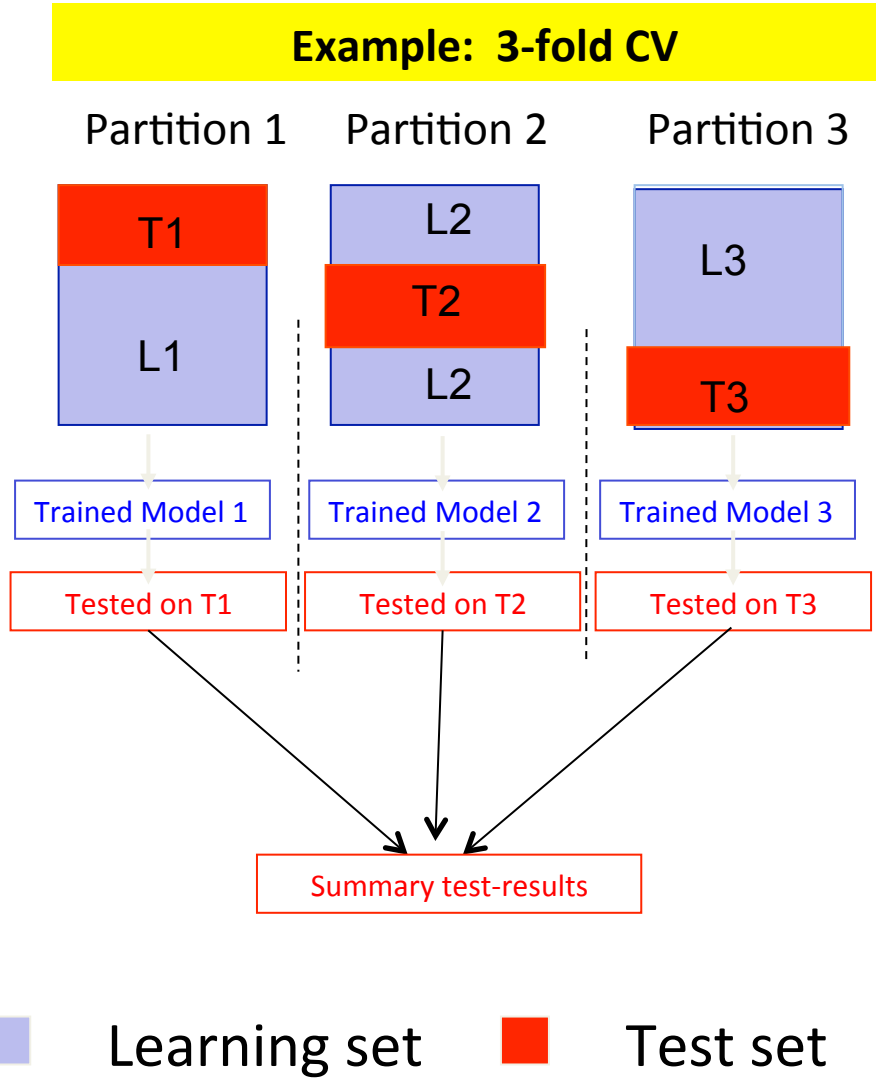
Use of data – typical case



Only later can test on new data,
New data will only be generated if the classifier appears to be promising for useful applications:

How to assess this ?

Cross-Validation Experimental Design



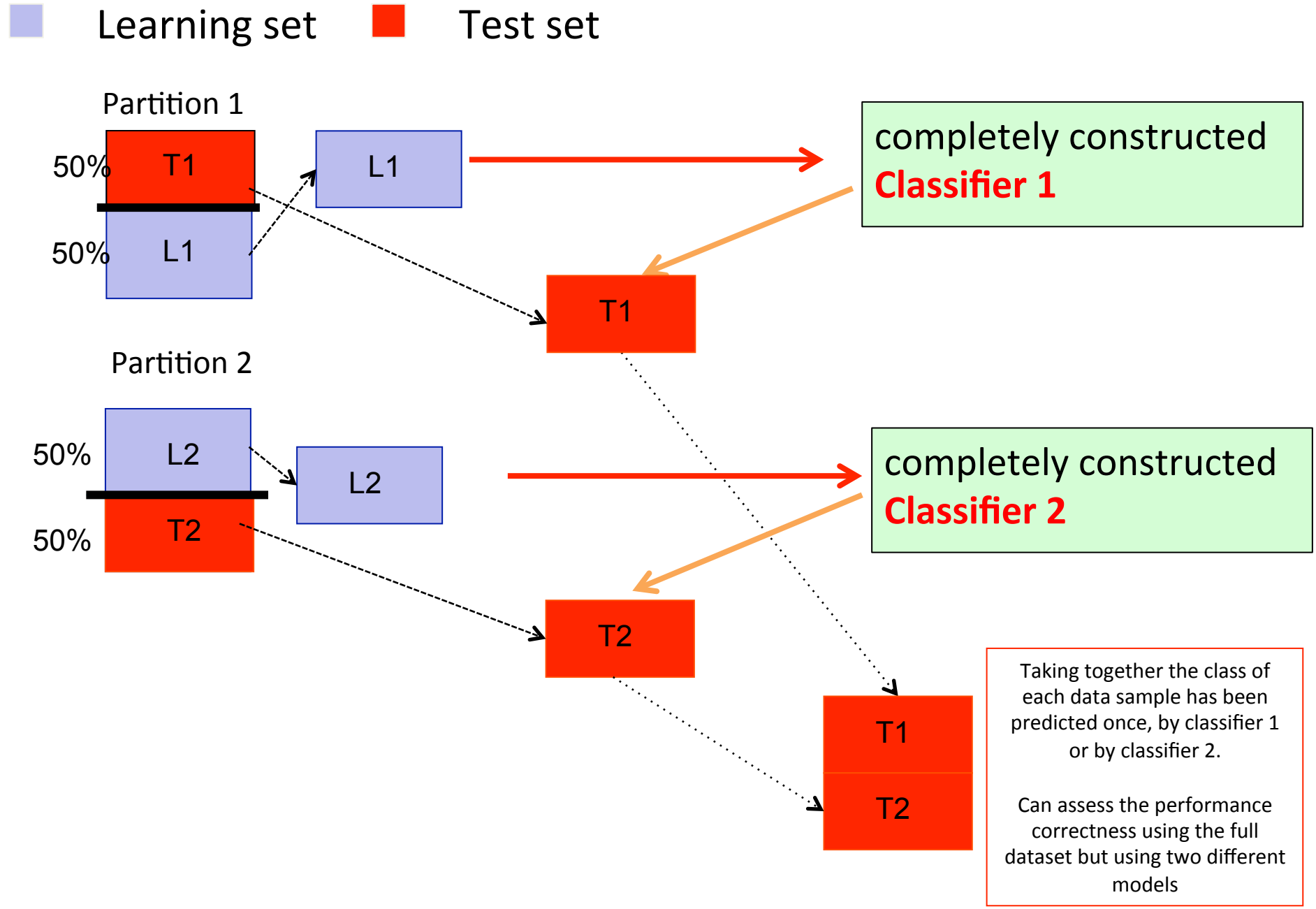
- The **learning sets** **overlap** each other (except 2-fold CV, half-half)

- The **test sets** **do not overlap**
- The test results are joined together, together each case is tested once and only once: the maximally-possible testing set

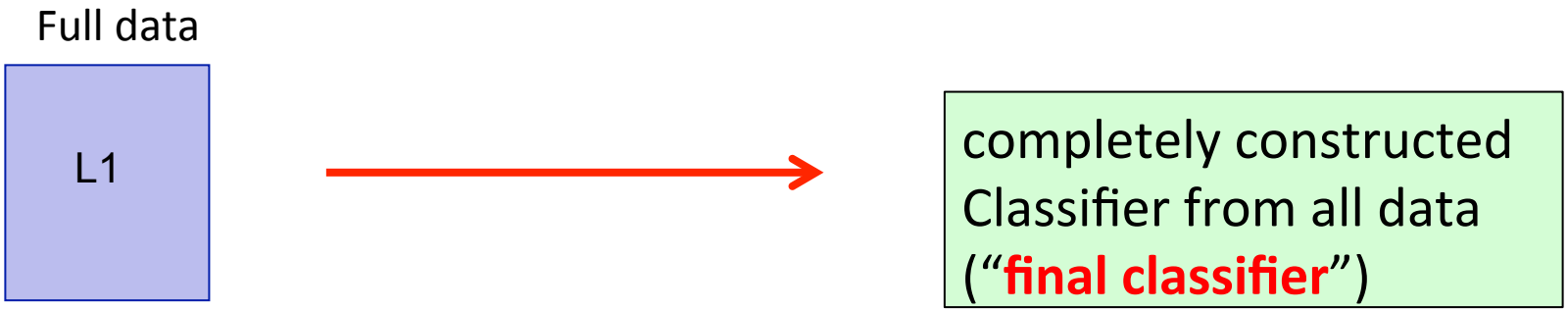
- The summary test results combines the prediction from **different classifiers**

- The procedure can be repeated with the same data but a different partition defined by the test sets:
- Except in the case of leave-one-out CV

Special case: 2-fold Cross-Validation, “half-half”



Cross-Validation's final classifier



- The performance observed in the cross-validation is intended to **estimate the future performance** of the classifier built on all data when it will be applied to new completely independent data (of the same type)
- The final classifier is constructed using all data while the classifiers used during the cross-validation are constructed using only part of the data, a part that should not be too small
- In the case of **leave-one-out crossvalidation** the learning sets used are almost as large as the full dataset

Cross-Validation Experimental Design

- The **final classifier** for future use is constructed using all data, it is **a different classifier than those constructed during the cross-validation**

- The prediction performance obtained in the cross-validation is a good estimated of the **expected future performance** of the **final classifier** on new data

- The procedure can be repeated with the same data taking **a different partition**, but the variability observed by doing so is **not** very useful to estimate of the uncertainty in the future performance

Special case: **Leave-one-out** Cross-Validation Procedure

- Classifier rules built on all data except the **leave-one-out record**
- Apply classifier on the **leave-one-out record** and store the prediction result
- Repeat leaving out each data point once
- Assess the correctness of the prediction results

JNCi commentary paper "Pitfalls" by Simon et al.

COMMENTARY

Pitfalls in the Use of DNA Microarray Data for Diagnostic and Prognostic Classification

Richard Simon, Michael D. Radmacher, Kevin Dobbin, Lisa M. McShane

DNA microarrays have made it possible to estimate the level of expression of thousands of genes for a sample of cells. Although biomedical investigators have been quick to adopt this powerful new research tool, accurate analysis and interpretation of the data have provided unique challenges. Indeed, many investigators are not experienced in the analytical steps needed to convert tens of thousands of noisy data points into reliable and interpretable biologic information. Although some investigators recognize the importance of collaborating with experienced biostatisticians to analyze microarray data, the number and availability of experienced biostatisticians is inadequate. Consequently, investigators are using available software to analyze their data, many seemingly without knowledge of potential pitfalls. Because of serious problems associated with the analysis and reporting of some DNA microarray studies, there is great interest in guidance on valid and effective methods for analysis of DNA microarray data.

The design and analysis strategy for a DNA microarray experiment should be determined in light of the overall objectives of the study. Because DNA microarrays are used for a wide variety of objectives, it is not feasible to address the entire range of design and analysis issues in this commentary. Here, we address statistical issues that arise from the use of DNA microarrays for an important group of objectives that has been called "class prediction" *(1)*. Class prediction includes derivation of predictors of prognosis, response to therapy, or any phenotype or genotype defined independently of the gene expression profile.

Experimental Objectives Drive Design and Analysis

Good DNA microarray experiments, although not based on gene-specific mechanistic hypotheses, should be planned and conducted with clear objectives. Three commonly encountered types of study objectives are "class comparison," "class prediction," and "class discovery" *(1)*.

Class comparison is the comparison of gene expression in different groups of specimens. The major characteristic of class comparison studies is that the classes being compared are defined independently of the expression profiles. The specific objectives of such a study are to determine whether the expression profiles are different between the classes and, if so, to identify the differentially expressed genes. One example of a class comparison study is the comparison of gene expression profiles of stage I breast cancer patients who are long-term survivors with the gene expression profiles of those who have recurrent disease. Another example is the comparison between gene expression profiles in breast cancer patients with and without germline BRCA1 mutations *(2)*.

Class prediction studies are similar to class comparison studies in that the classes are predefined. In class prediction studies, however, the emphasis is on developing a gene expression-based multivariate function (referred to as the predictor) that accurately predicts the class membership of a new sample on the basis of the expression levels of key genes. Such predictors can be used for many types of clinical management decisions, including risk assessment, diagnostic testing, prognostic stratification, and treatment selection. Many studies include both class comparison and class prediction objectives.

Class discovery is fundamentally different from class comparison or class prediction in that no classes are predefined. Usually the purpose of class discovery in cancer studies is to determine whether discrete subsets of a disease entity can be defined on the basis of gene expression profiles. This purpose is different from determining whether the gene expression profiles correlate with some already known diagnostic classification. Examples of class discovery are the studies by Bittner et al. *(3)* that examined gene expression profiles for advanced melanomas and by Alizadeh et al. *(4)* that examined the gene expression profiles of patients with diffuse large B-cell lymphoma. Often the purpose of class discovery is to identify clues regarding the heterogeneity of disease pathogenesis.

Limitations of Cluster Analysis for Class Prediction

One of the most common errors in the analysis of DNA microarray data is the use of cluster analysis and simple fold change statistics for problems of class comparison and class prediction. Although cluster analysis is appropriate for class discovery, it is often not effective for class comparison or class prediction. Cluster analysis refers to an extensive set of methods for partitioning samples into groups on the basis of the similarities and differences (referred to as distances) among their gene expression profiles. Because there are many ways of measuring distances among gene expression profiles involving thousands of genes and because there are many algorithms for partitioning, cluster analysis is a very subjective analysis strategy.

Cluster analysis is considered an unsupervised method of analysis because no information about sample grouping is used. The distance measures are generally computed with regard to the complete set of genes represented on the array that are measured with sufficiently high signals, or with regard to all the genes that show meaningful variation across the sample set. Because relatively few genes may distinguish any particular class, the distances used in cluster analysis will often not reflect the influence of these relevant genes. This feature accounts for the poor results often obtained in attempting to use cluster analysis for class prediction studies.

Cluster analysis also does not provide statistically valid quantitative information about which genes are differentially expressed between classes. Investigators often use simple average fold change measures or visual inspection of a cluster image display to identify differentially expressed genes. However, average fold change indices do not account for variability in gene expression across samples within the same class; some twofold average effects represent statistically significant differences and some do not. Neither fold change indices nor visual inspection of cluster image displays enable the investigator to deal with multiple comparison issues in a statistically valid manner. For example, in examining expression levels of thousands of randomly varying genes, there may be many genes that spuriously appear to be differentially expressed between two classes on the basis of visual inspection or fold change thresholds.

Class Prediction Using Supervised Methods

For class prediction studies, it is more appropriate to use a supervised method (i.e., one that makes distinctions among the specimens on the basis of predefined class label information) than an unsupervised method, such as cluster analysis. Supervised class prediction is usually based on the assumption that a collection of differentially expressed genes is associated with class distinction.

The first step toward constructing the class predictor (sometimes called the classifier) is to select the subset of informative genes. The second step is often to assign weights related to the individual predictive strengths of these informative genes. Predictors based on linear combinations of the weighted intensity measurements of the informative genes have been proposed *(1,5)*. One alternative method is to use a dimension reduction technique such as principal components analysis or partial least squares on the informative genes and to base the prediction on the resulting factors *(6–8)*. Many other methods for defining a multivariate predictor have been described *(9,10)*. The final step in constructing the classifier is to define the prediction rule. For example, in a two-group classification where a single predictor is computed, the classification rule may simply be a threshold value; a specimen is classified as being in one group if the derived predictor value is less than the threshold and classified as being in the other group if the derived predictor value is more than the threshold.

One major limitation of supervised methods is overfitting the predictor. Overfitting means that the number of parameters of the model is too large relative to the number of cases or specimens available. Because the model parameters are optimized for the data, the model will fit the original data but may predict poorly for independent data. This happens because the model fits random variations within the original data that do not represent true relationships that hold for independent data. Consequently, it is essential to obtain an unbiased estimate of the true error rate of the predictor (i.e., the probability of incorrectly classifying a randomly selected future case).

Methods for obtaining unbiased estimates of a predictor's error rate include leave-one-out cross-validation or application of the prediction rule developed from a supervised analysis of one dataset to an independent dataset. By using these techniques, it is possible not only to evaluate overfitting the predictor, but also to compare various prediction methods and assess which ones are less prone to overfitting. The appropriate use of leave-one-out cross-validation and validation of independent datasets is discussed in the next two sections.

Cross-Validation of Prediction Accuracy

The performance of a class prediction rule is best assessed by applying the rule created on one set of data (the training set) to an independent set of data (the validation set). Because most clinical research laboratories have access to only a limited number of tumor samples, withholding a substantial proportion of the samples from the training set for the sake of creating a validation set may considerably reduce the performance of the prediction rule. Cross-validation procedures use the data more efficiently. A small number of specimens are withheld, and most of the specimens are used to build a predictor. The predictor is used to predict class membership for the withheld specimens. This process is iterated, leaving out a new set of specimens at each step, until all specimens have been classified. In leave-one-out cross-validation, for example, each specimen is excluded from the training set one at a time and then classified on the basis of the predictor built from the data for all of the other specimens. The leave-one-out cross-validation procedure provides a nearly unbiased estimate of the true error rate of the classification procedure. The estimated error rate applies to the procedure used to build the classifier rather than to the specific prediction model based on all the data, because there is a different classifier for each leave-one-out training set *(11,12)*. Other cross-validation methods omit more than one specimen at a time *(13)* and also produce nearly unbiased estimates.

In the previous section, three common components of class prediction methods were listed: 1) selection of informative genes, 2) computation of weights for selected informative genes, and 3) creation of a prediction rule. It is important that all three steps undergo the cross-validation procedure. Failure to cross-validate all steps may lead to substantial bias in the estimated error rate.

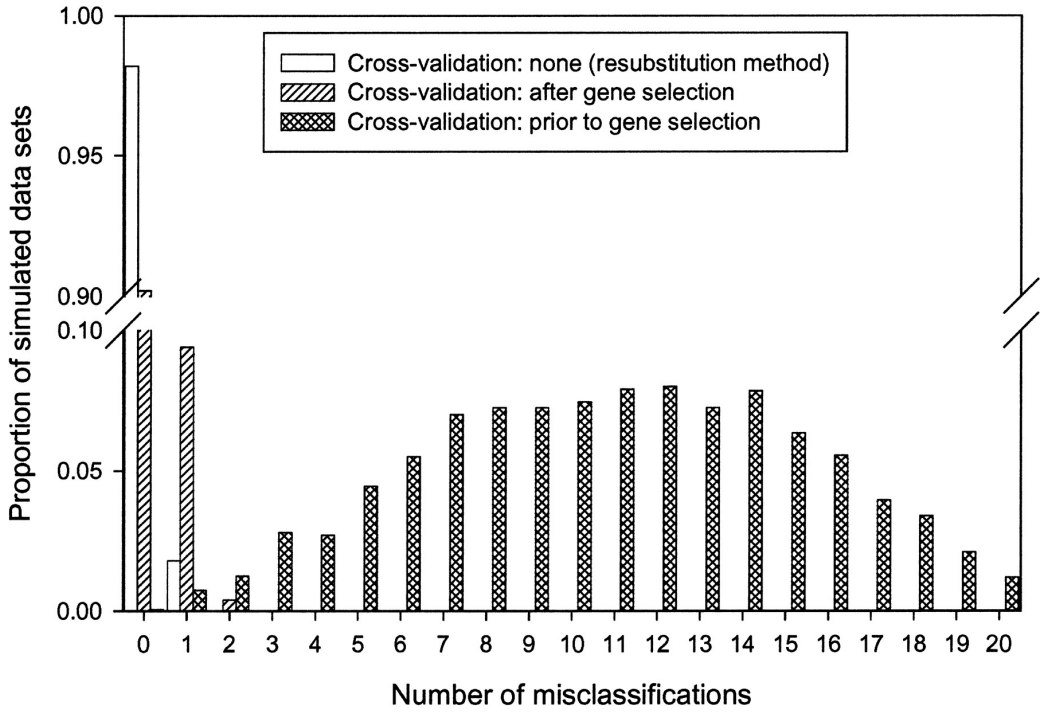
We performed a simulation to examine the bias in estimated error rates for a class prediction study with various levels of cross-validation (*see* supplemental information at http://jncicancerspectrum.oupjournals.org/jnci/content/vol95/issue1/index.shtml and at http://linus.nci.nih.gov/~brb for a full description of the simulation). We considered two types of leave-one-out cross-validation: one with removal of the left-out specimen before selection of differentially expressed genes and one with removal of the left-out specimen after gene selection but before computation of gene weights and application of the prediction rule. We also computed the resubstitution estimate of the error rate (this estimate results from building the predictor on the full dataset and then reapplying it to each specimen for classification purposes). In each simulated dataset, 20 expression profiles of 6000 genes were randomly generated from the same distribution. Ten profiles were arbitrarily assigned to class 1 and the other 10 profiles to class 2, creating an artificial separation of the profiles into two classes. Because no true underlying difference existed between the two classes, the class prediction should perform no better than a random guess, with

Affiliations of authors: R. Simon, K. Dobbin, L. M. McShane, Biometric Research Branch, National Cancer Institute, National Institutes of Health, Bethesda, MD; M. D. Radmacher, Departments of Biology and Mathematics, Kenyon College, Gambier, OH.

Correspondence to: Richard Simon, D.Sc., National Cancer Institute, 9000 Rockville Pike, MSC 7434, Bethesda, MD 20892-7434 (e-mail: rsimon@nih.gov).

See "Note" following "References."

Figure in Publication



- To get a honest estimation of the performance

- We cannot apply the model constructed on all data on the same data (**resubstitution**): the apparent performance is much too high

- A **cross-validation** approach can give a much better assessment of performance

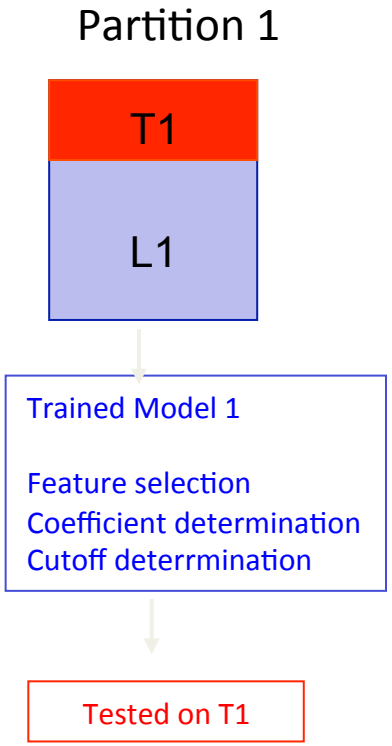
- But if some steps of the classifier construction are done with all the data (**incomplete cross-validation**) then this might not be the case, as shown here

- Demonstrated **here**: incomplete CV predicts a good performance in random data where by construction the two groups are the same and no gene has a "real" generally valid difference, so no classifier can be "significantly" better than making 50% errors on average

Complete and incomplete Leave-one-out Cross-Validation Procedure

Complete

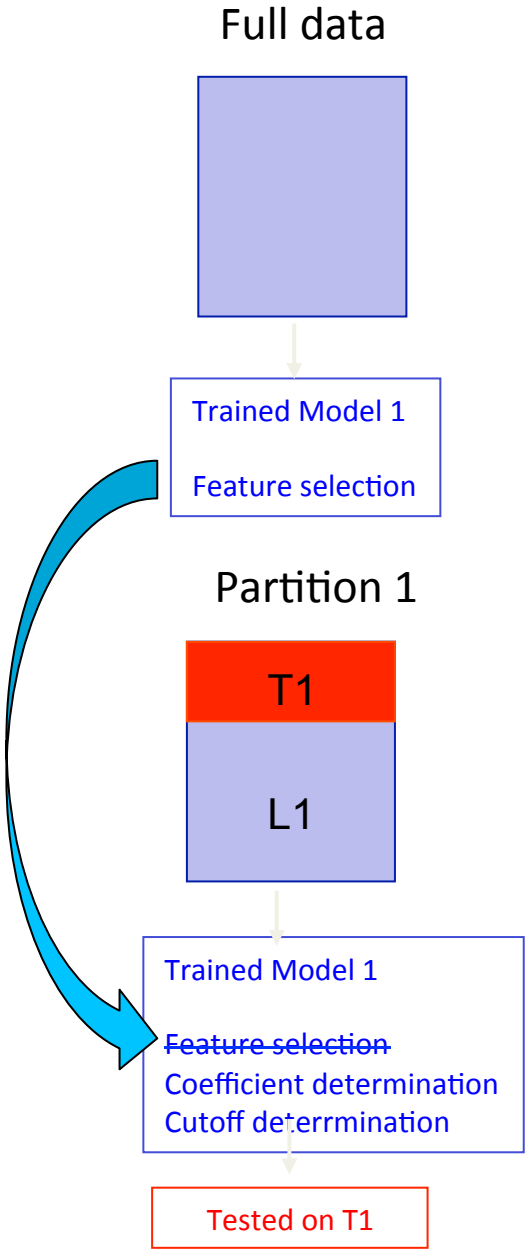
Classifier rules built on all data except the leave-one-out record, this is done including all steps of the classifier construction



Incomplete

Classifier rules built on all data except the leave-one-out record, this is done including for some steps of the classifier construction

other steps are done on the full data and taken over “as is”



Parameters in the publication

- Two groups, each class 10 samples
- For each gene use a t-test btw the two groups
- Take the top **10 genes** with the best t-statistics (equivalent: best p-values)
- Define a discriminatory score using the t-statistics as coefficients, so that the genes high in the first group are taken positively, summed for the score, while the genes high in the second group are taken negatively, subtracted to calculate the score

  Score = $c_i = \sum_j t_j x_{ij}$,
- Define a cutoff for the score, new samples with a score above the cutoff are classified as group one, the others as group two,

  Cutoff = mean btw. (score average in group 1, score average in group 2)
- The simulation was **repeated** 2'000 times, each time with a new dataset generated at random: need to repeat many many times to get a good picture of what happens and to estimate the average perf.

needed for the report

EXERCISE E5: REPRODUCE THE ANALYIS DONE IN THE PAPER AND GENERATE A VERSION OF THEIR FIGURE *
(If the simulation is too slow for 2'000 repetitions and 6'000 genes reduce the number of simulations in the lab and run the fuller version later at home)

***EXERCISE E5a: write code for one full cross-validation

***EXERCISE E5b: write code for one incomplete cross-validation

***EXERCISE E5c: do an appropriate number of repetitions to collect results for resubstitution, full and incomplete cross-validation

supplementary suggested additional exercises

EXERCISE E6: Scenario: there is one informative gene and only one gene is selected for use in the classifier: Generate and interpret the analogous figure (set effect.mean = 2) *
(An illustration of "Winner's bias")
Extension E6b: test the classifiers also on an independent second (large) dataset and compare the performances in selected cases.

EXERCISE E7: In the situation of the exercise E6: *
How large should the sample size of the dataset be so that one can build a classifier that reaches at least 90% of the maximal performance in at least 90% of the cases ?
(An illustration of a "Power Study")

SCHEDULE

WORK ON EXERCISES

```
# EXAMPLE SOLUTION E5a
# one full LOO crossvalidation :

pred.loo <- rep(NA, Nlearn)

for ( loo in c(1:Nlearn) )
{
learndata <- data
learndata$X <- learndata$X[-loo, ]
learndata$Y <- learndata$Y[-loo]

classifier.loo <- classifier.function(mydata=learndata,
NBgenes.selected=NBclassifierFeatures, NBobservations=(Nlearn-1) )

score.loo <- as.numeric( data$X[ loo, classifier.loo$features] %*% classifier.loo
$coefficients )

pred.loo[loo] <- sign(score.loo <  classifier.loo$cutoff)
}

corr.prop.loo <- sum(pred.loo == data$Y) / Nlearn
```

```
# EXAMPLE SOLUTION E5b
# select genes on all data :
classifier <- classifier.function(mydata=data,
NBgenes.selected=NBclassifierFeatures, NBobservations=Nlearn) # this is the
original classifier as for resubstitution, trained on all data
selectedgenes <- classifier$features

# one LOO crossvalidation loop:
pred.iCV <- rep(NA, Nlearn)
for ( loo in c(1:Nlearn) )
{
learndata <- data
testdata.loo <- learndata$X[loo, selectedgenes]
learndata$X <- learndata$X[-loo, selectedgenes]
learndata$Y <- learndata$Y[-loo]
classifier.icv <- classifier.function(mydata=learndata,
NBgenes.selected=NBclassifierFeatures, NBobservations=(Nlearn-1) )
score.iCV <- as.numeric( testdata.loo[classifier.icv$features] %*%
classifier.icv$coefficients)
pred.iCV[loo] <- sign(score.iCV <  classifier.icv$cutoff)
}
corr.prop.iCV <- sum(pred.iCV == data$Y) / Nlearn
```

 # EXAMPLE SOLUTION E5c-1

```
# parameters
Ngroup <- 10; Nlearn <- 2*Ngroup;
NBgenesMeasured <- 6000; # genes in data
NBclassifierFeatures <- 10
mean.x <- 0; sd.x <- 1
effect.mean <- 0; effect.group <- 1

# other variables
repetitions  <- 200
corr.prop.resub <- rep(NA, repetitions)
corr.prop.loo <- rep(NA, repetitions)
corr.prop.iCV <- rep(NA, repetitions)
```

 # EXAMPLE SOLUTION E5c-2

```
SEED <- 443
for (i in  c(1:repetitions) )
{
SEED <- SEED+1; set.seed(SEED);

cat("\n data")
data <- datageneration.function(localSEED=SEED, Ngroup=Ngroup,
Ngenes=NBgenesMeasured, meanvalue=mean.x , sdvalue=sd.x ,
effect=effect.mean )

cat("\n resubstitution case")
classifier <- classifier.function(mydata=data,
NBgenes.selected=NBclassifierFeatures, NBobservations=Nlearn)
corr.prop.resub[i] <- classifier$correct.proportion
selectedgenes <- classifier$features
```

 # EXAMPLE SOLUTION E5c-3

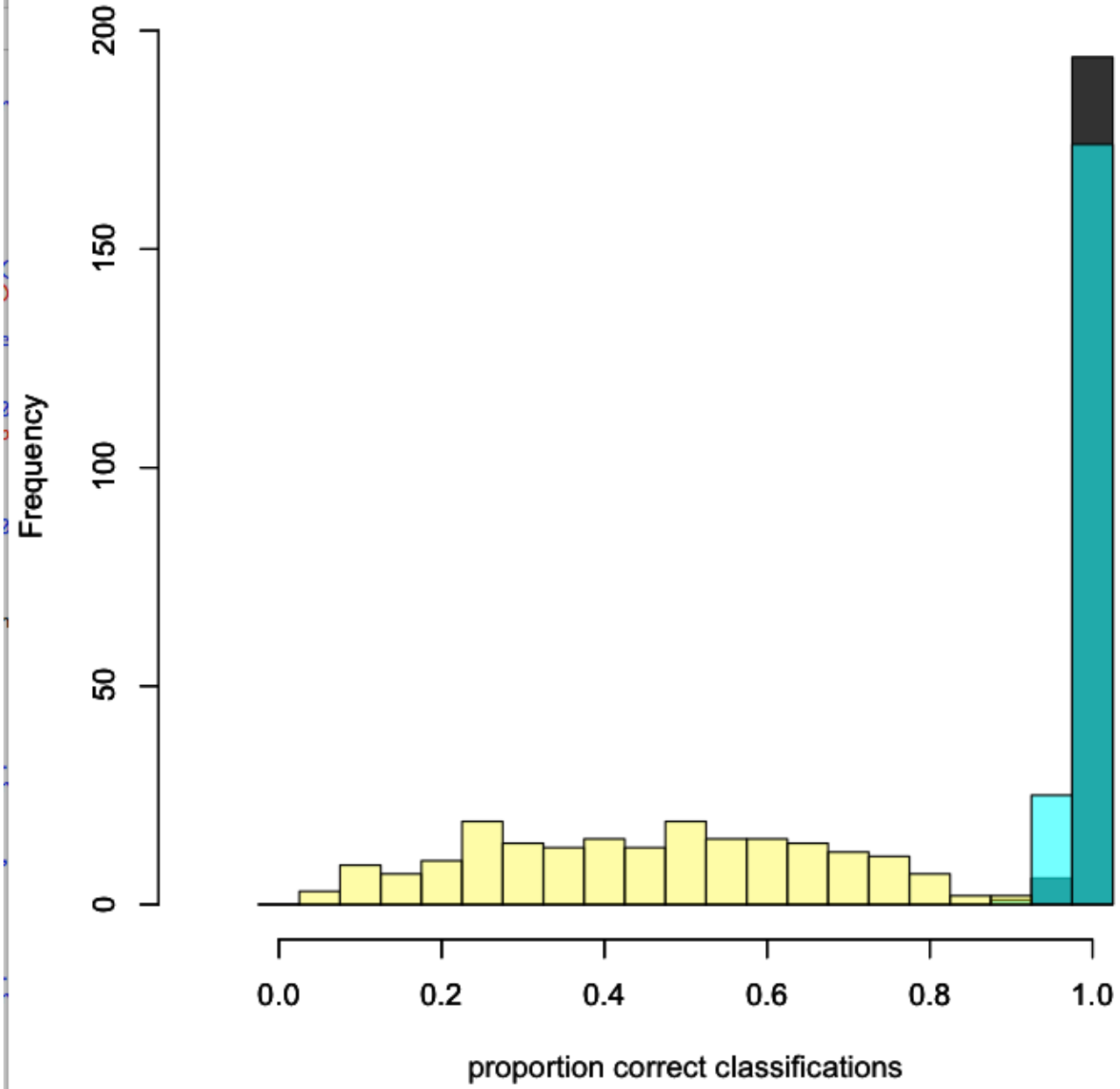
```
cat("\n incomplete LOO crossvalidation")
# incomplete LOO crossvalidation : # could put this section into a function
pred.iCV <- rep(NA, Nlearn)
for ( loo in c(1:Nlearn) )
{
learndata <- data
testdata.loo <- learndata$X[loo, selectedgenes]
learndata$X <- learndata$X[-loo, selectedgenes]
learndata$Y <- learndata$Y[-loo]
classifier.icv <- classifier.function(mydata=learndata,
NBgenes.selected=NBclassifierFeatures, NBobservations=(Nlearn-1) )
score.iCV <- as.numeric( testdata.loo[classifier.icv$features] %*% classifier.icv
$coefficients)
pred.iCV[loo] <- sign(score.iCV <  classifier.icv$cutoff)
}
corr.prop.iCV[i] <- sum(pred.iCV == data$Y) / Nlearn
```

 # EXAMPLE SOLUTION E5c-4

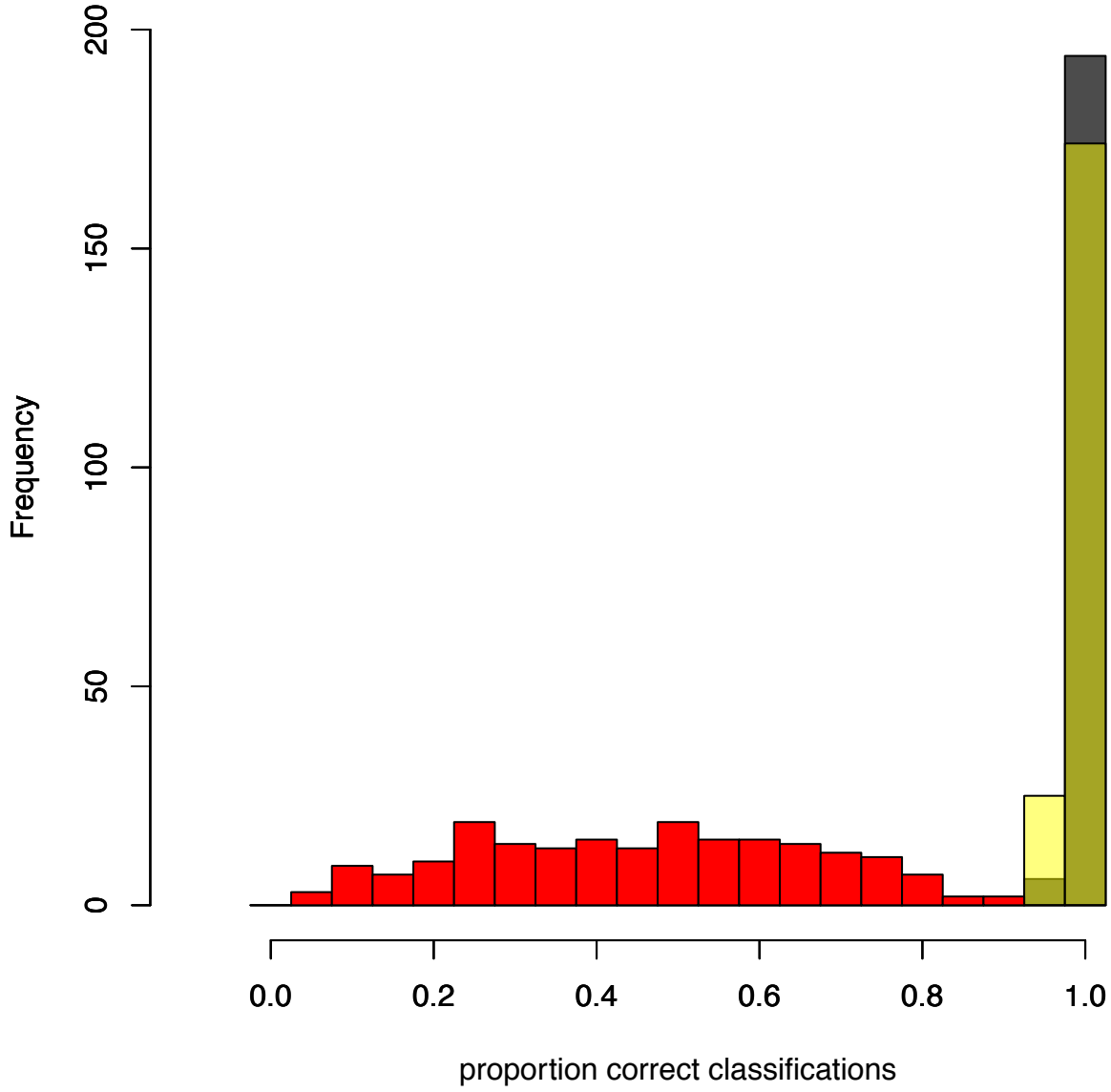
```
cat("\n full LOO crossvalidation")
# full LOO crossvalidation : # could put this section into a function
pred.loo <- rep(NA, Nlearn)
for ( loo in c(1:Nlearn) )
{
learndata <- data
learndata$X <- learndata$X[-loo, ]
learndata$Y <- learndata$Y[-loo]
classifier.loo <- classifier.function(mydata=learndata,
NBgenes.selected=NBclassifierFeatures, NBobservations=(Nlearn-1) )
score.loo <- as.numeric( data$X[ loo, classifier.loo$features] %*% classifier.loo
$coefficients )
pred.loo[loo] <- sign(score.loo <  classifier.loo$cutoff)
if ( (loo %% 5) == 0)  {cat("\n loo=",loo)}
}
corr.prop.loo[i] <- sum(pred.loo == data$Y) / Nlearn
```

 # EXAMPLE SOLUTION E5c-5

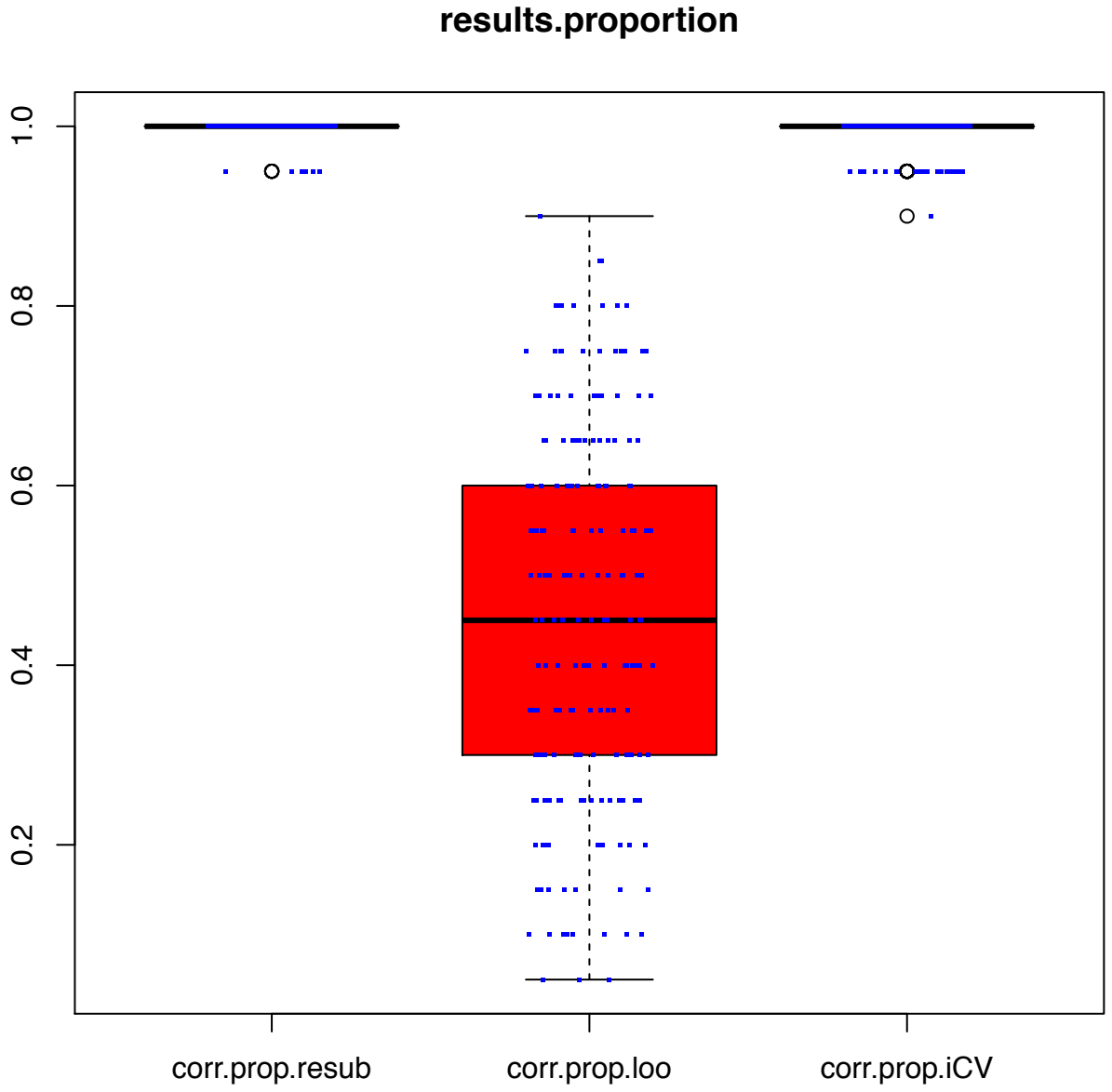
```
if ( (i %% 10) == 0)  {cat("\n\n repetitions i=",i)}
}


boxplot( list ( corr.prop.resub=corr.prop.resub,
corr.prop.loo=corr.prop.loo, corr.prop.icv=corr.prop.iCV ))
```


Frequency
0
50
100
150
200
0.0
0.2
0.4
0.6
0.8
1.0
proportion correct classifications

Exercises - Results



Exercises - Results



Exercises - Results

Results
6000 genes, 200 repetitions

```
> table(corr.prop.resub)
corr.prop.resub
0.95    1
  6  194

> table(corr.prop.loo)
corr.prop.loo
0.05  0.1 0.15  0.2 0.25  0.3 0.35  0.4 0.45  0.5 0.55  0.6 0.65  0.7 0.75  0.8 0.85  0.9
  3    9    7   10   19   14   13   15   13   19   15   15   14   12   11    7    2    2

> table(corr.prop.iCV)
corr.prop.iCV
 0.9 0.95    1
   1   25  174
```

Exercises - Results

Results
6000 genes, 200 repetitions

```
> summary(corr.prop.resub)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.9500  1.0000  1.0000  0.9925  1.0000  1.0000

> summary(corr.prop.loo)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1000  0.3000  0.5000  0.4325  0.5000  0.8500

> summary(corr.prop.iCV)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   0.95    0.95    1.00    0.98    1.00    1.00
```