The "chemometrics" Package in R – Application in Multivariate Calibration and Classification

Peter Filzmoser

Department of Statistics and Probability Theory
Vienna University of Technology, Austria

Chemometrics Workshop, Wien 2010

June 28, 2010

Vienna University of Technology

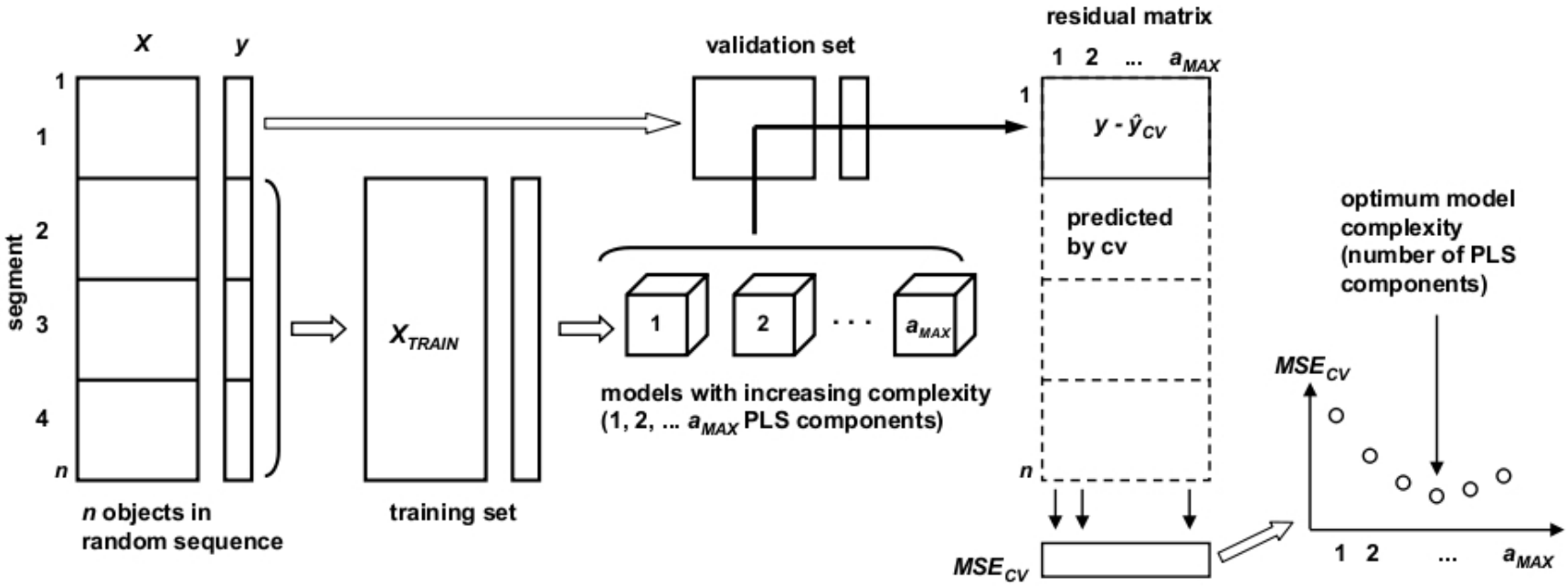
What is available in R?

. . . nearly "everything" with respect to multivariate calibration and classification. For calibration, the package "pls" is very valuable.

Main contributions of the "chemometrics" package":

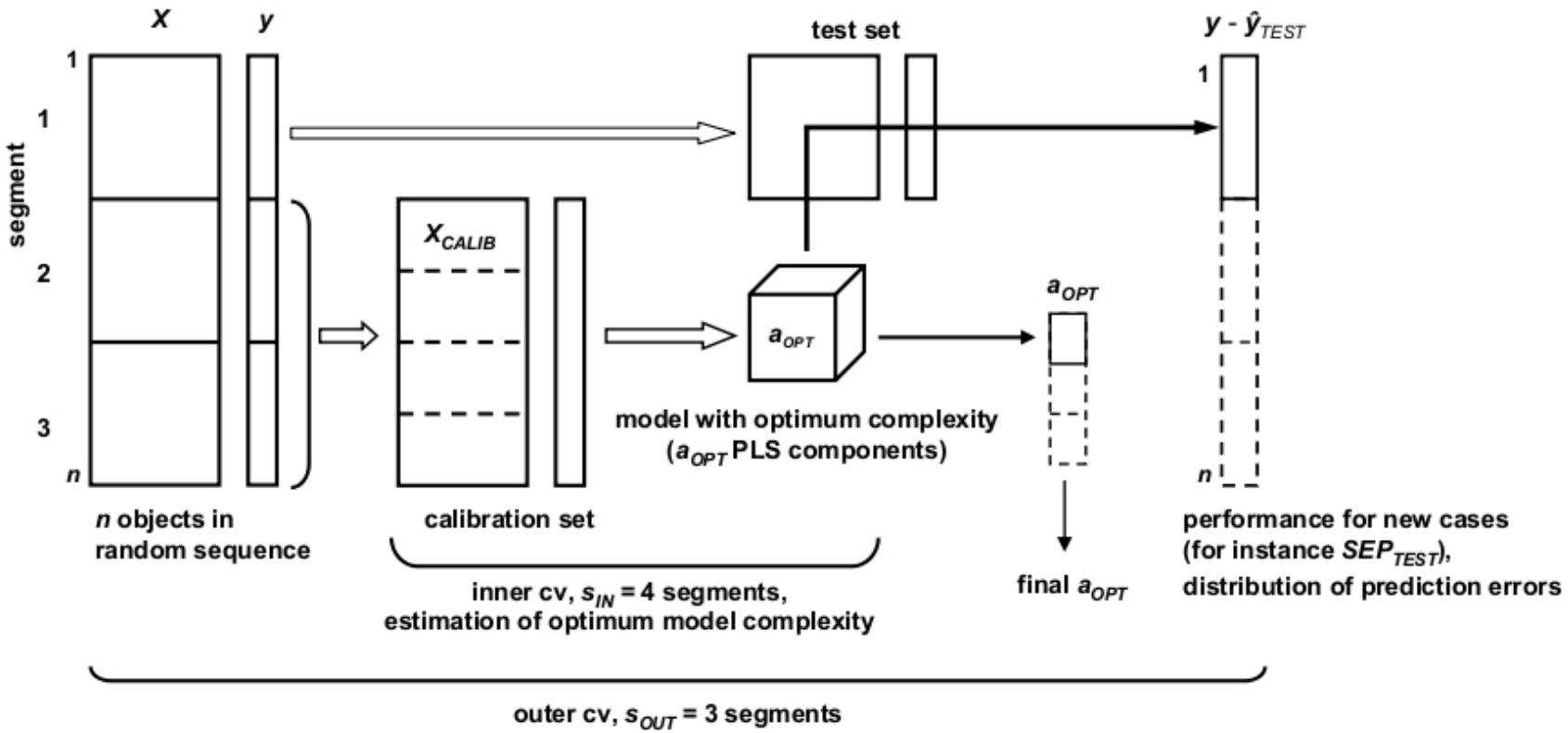
- Robustness (e.g. robust PLS)
- Model evaluation (e.g. repeated double cross-validation)
- Unified evaluation tools for parameter selection
- Diagnostic tools (e.g. for choice of number of components, visualizing effect of outliers)
- Example data sets

Cross Validation



Peter.Filzmoser@tuwien.ac.at

Repeated Double Cross Validation



Peter.Filzmoser@tuwien.ac.at

Multivariate Calibration

Example data:

Gas chromatographic retention indices of polycyclic aromatic compounds:

We consider $n = 209$ polycyclic aromatic compounds (PAC):

y-vector: GC retention index;

X-matrix: $m = 467$ descriptors of the molecular structure (Corina, Dragon).

```
> library(chemometrics)
> data(PAC)
> str(PAC)

List of 2
 $ y: num [1:209] 197 197 197 200 201 ...
 $ X: num [1:209, 1:467] 6.51 6.51 6.01 7.12 8.95 6.62 7.32 7.6 7.6 6.77 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:209] "1" "2" "3" "4" ...
  .. ..$ : chr [1:467] "AMW" "Me" "Mp" "Ms" ...
```

PLS (and PCR)

Idea: reduce the number of regressor variables to a few components
(PCR: using only the X-data; PLS: using both X and y data).

```
> pls_dcv <- mvr_dcv(y~X,ncomp=50,data=PAC,method="simpls")
             # PLS with repeated double cross validation
             # Default: 100 repetitions, 4 outer and 10 inner segments
             # for PCR use: method="svdpc"
```

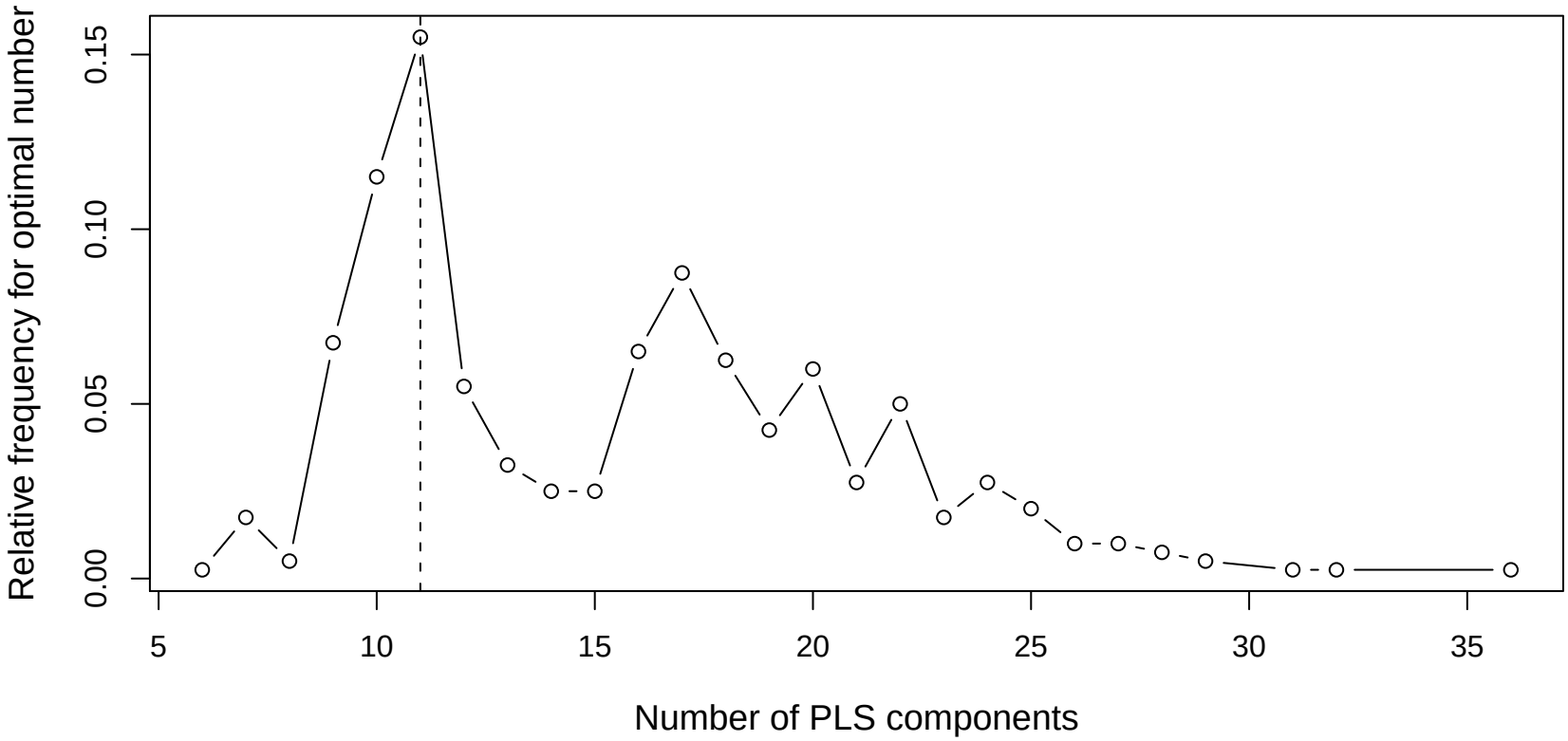
Output:

```
 $ resopt  : num [1:209, 1, 1:100]
 $ predopt : num [1:209, 1, 1:100]
 $ optcomp : int [1:4, 1:100]
 $ pred    : num [1:209, 1, 1:50, 1:100]
 $ SEPopt  : num 12
 $ sIQRopt : num 8.4
 $ sMADopt : num 8.39
 $ MSEPopt : num 144
 $ afinal  : num 11
 $ SEPfinal: Named num [1:50]
```

PLS (and PCR)

Diagnostic plots: optimal number of components (based on 4×100 values)

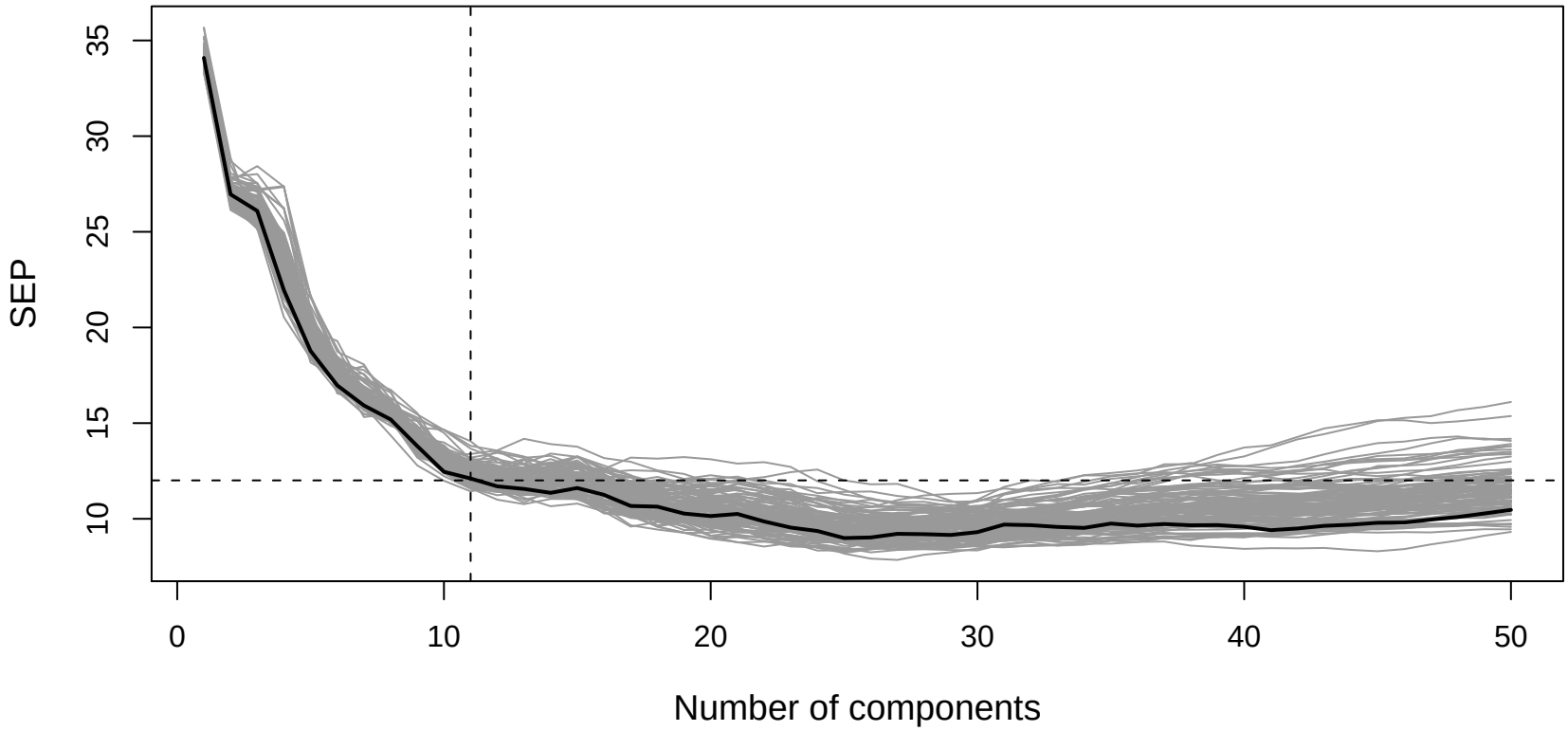
```
> plotcompmvr(pls_dcv)
```



PLS (and PCR)

Diagnostic plots: SEP for $1, \ldots, 50$ components

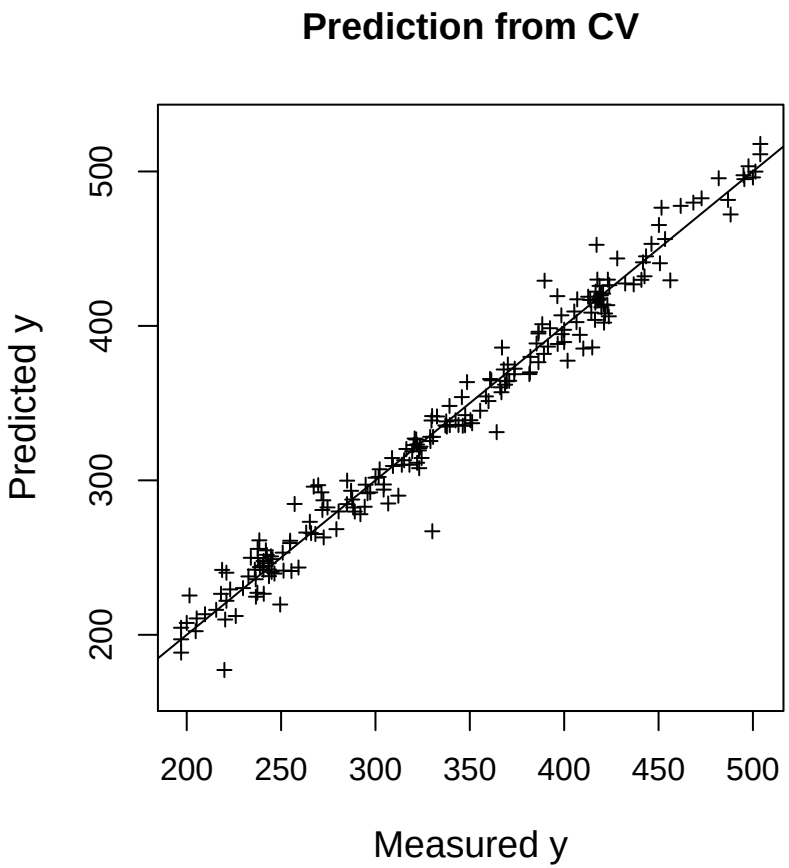
```
> plotSEPmvr(pls_dcv,optcomp=11,y=PAC$y,X=PAC$X,method="simpls")
```

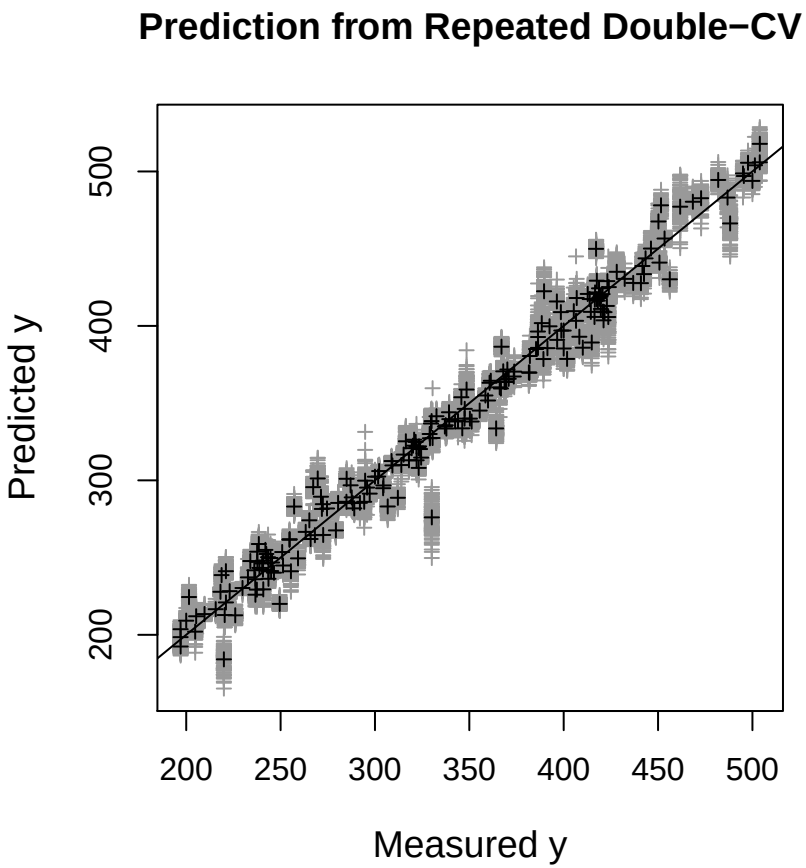


PLS (and PCR)

Diagnostic plots: predicted values from 100 models with 11 components

```
> plotpredmvr(pls_dcv,optcomp=11,y=PAC$y,X=PAC$X,method="simpls")
```



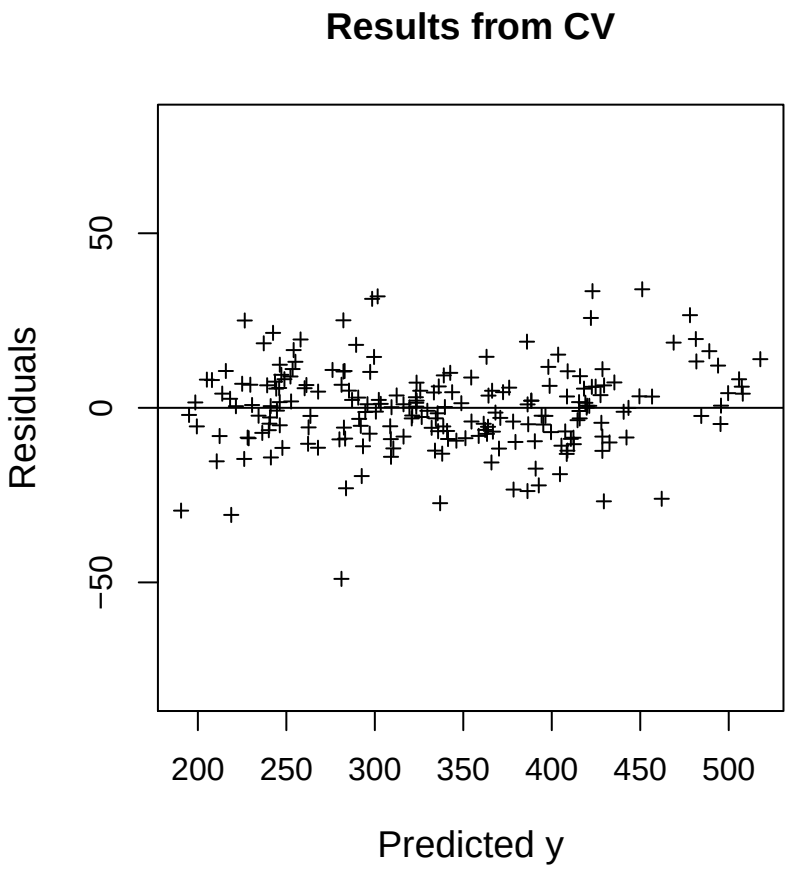


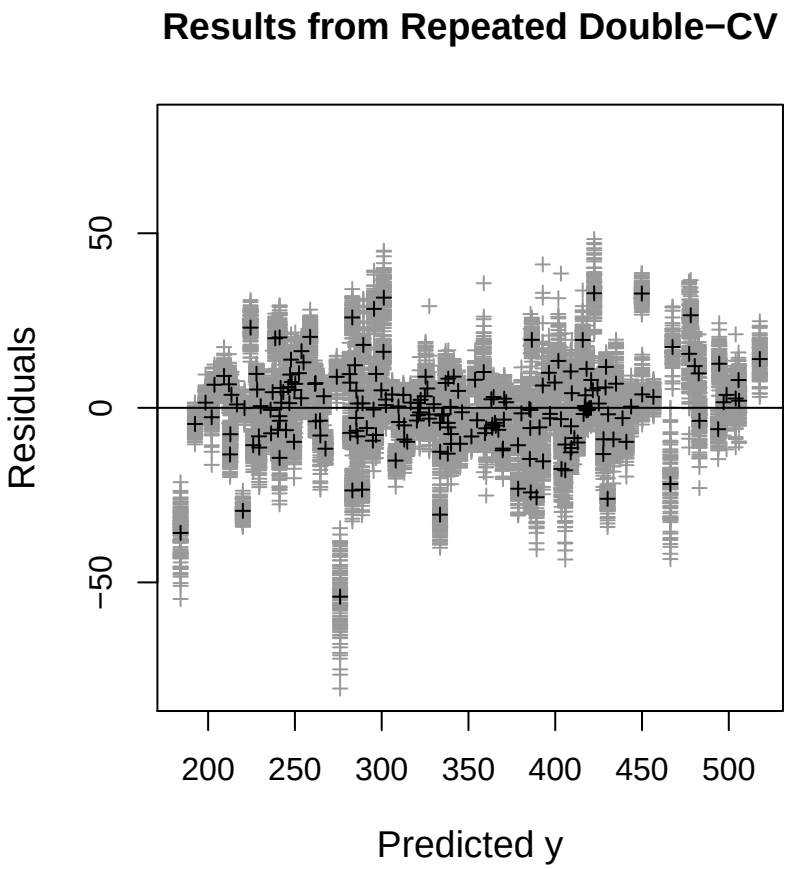
Peter.Filzmoser@tuwien.ac.at

PLS (and PCR)

Diagnostic plots: residuals from 100 models with 11 components

```
> plotresmvr(pls_dcv,optcomp=11,y=PAC$y,X=PAC$X,method="simpls")
```

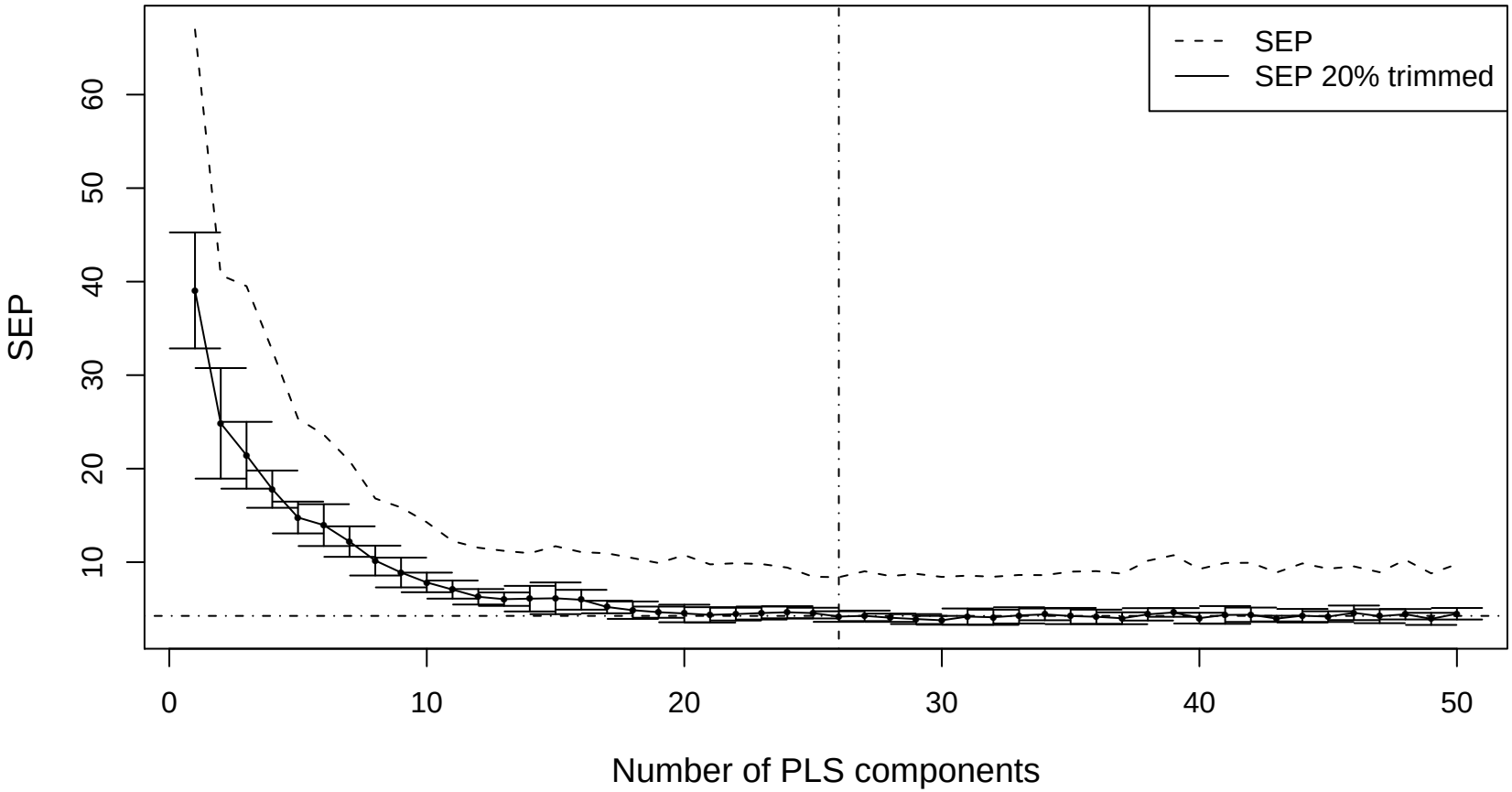




Robust PLS

PRM: Serneels, Croux, Filzmoser, Van Espen (ChemoLab, 2005)

```
> prm_cv(PAC$X,PAC$y,a=50,trim=0.2,plot.opt=TRUE)
```

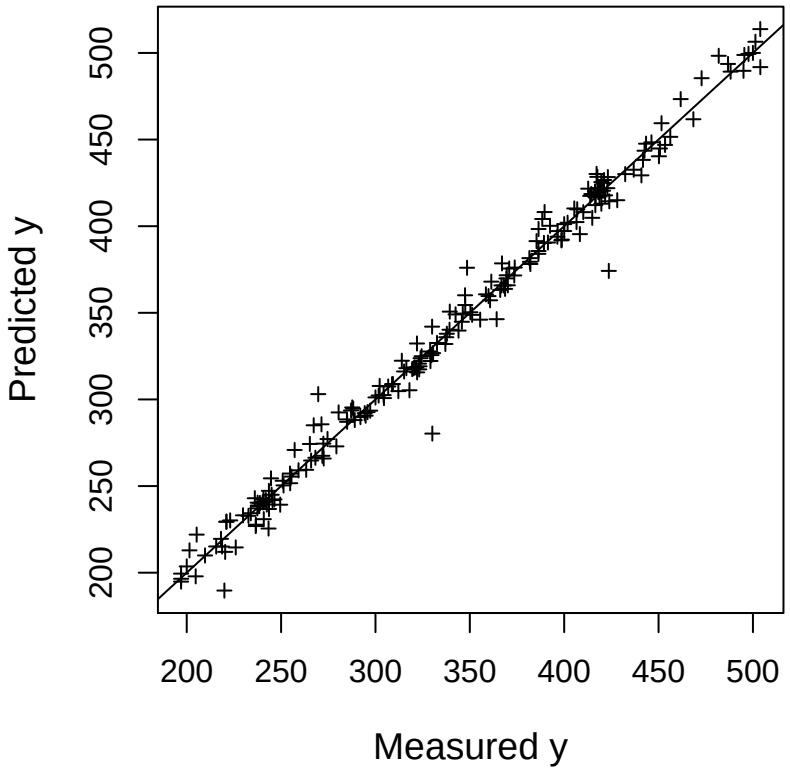


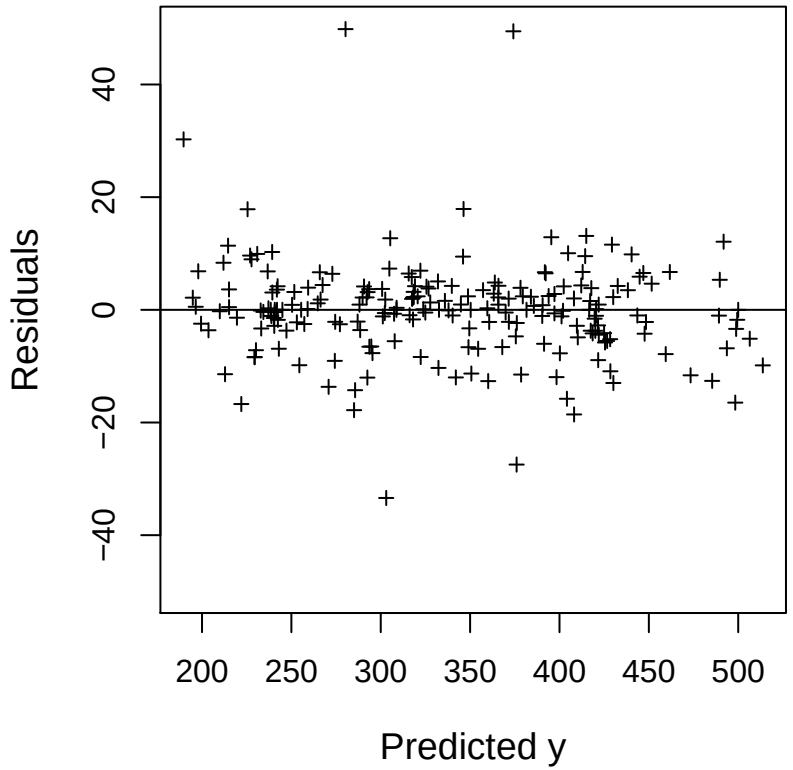
Peter.Filzmoser@tuwien.ac.at

Robust PLS

Diagnostic plots: predicted values and residuals using 26 components

```
> plotprm(resprmcv,PAC$y)
```



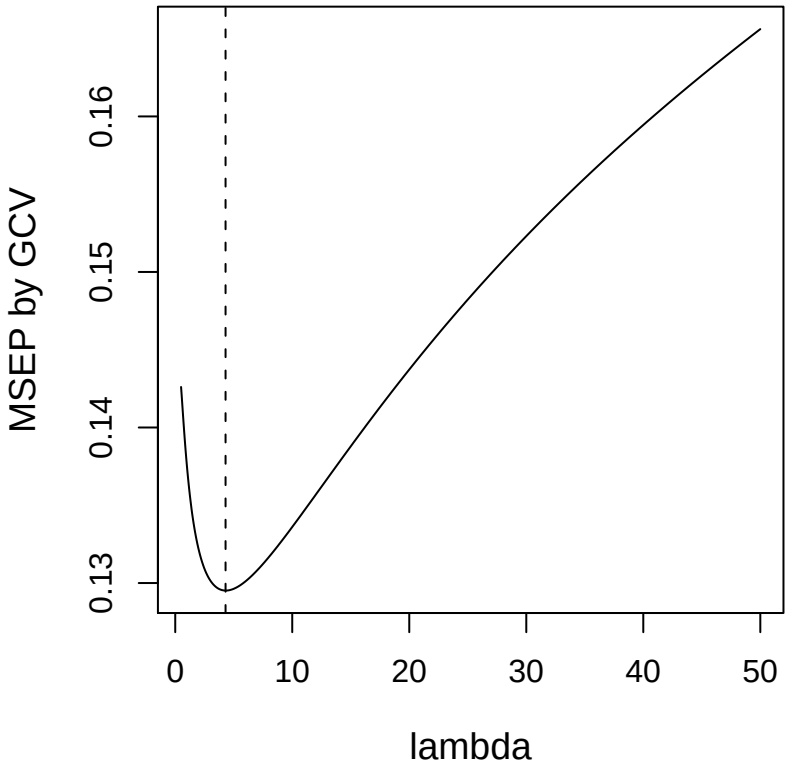


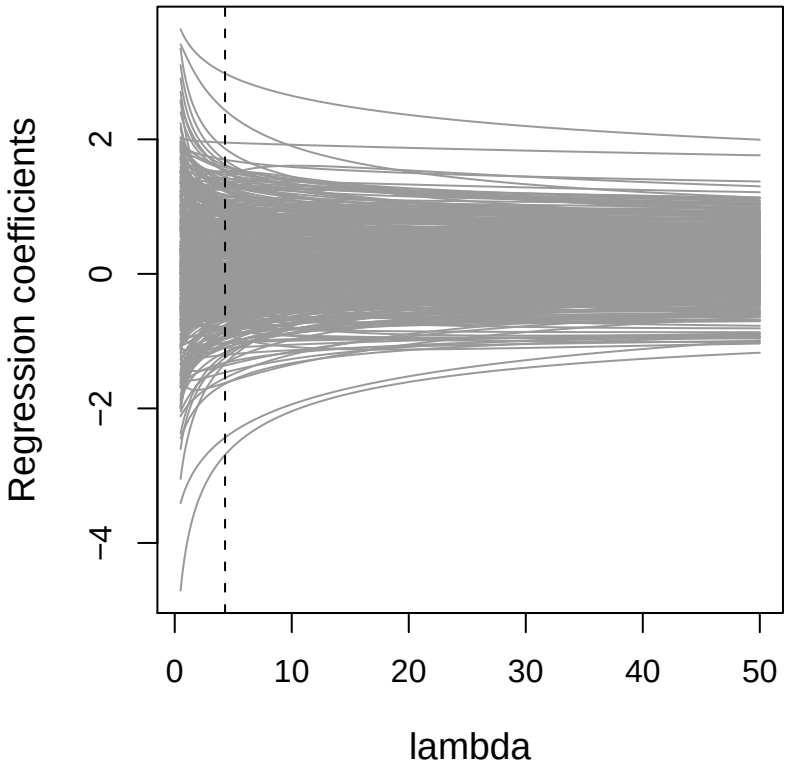
Peter.Filzmoser@tuwien.ac.at

Ridge Regression

Diagnostic plot: choice of ridge parameter

```
> resR <- plotRidge(y~X,data=PAC,lambda=seq(0.5,50,by=0.05))
```



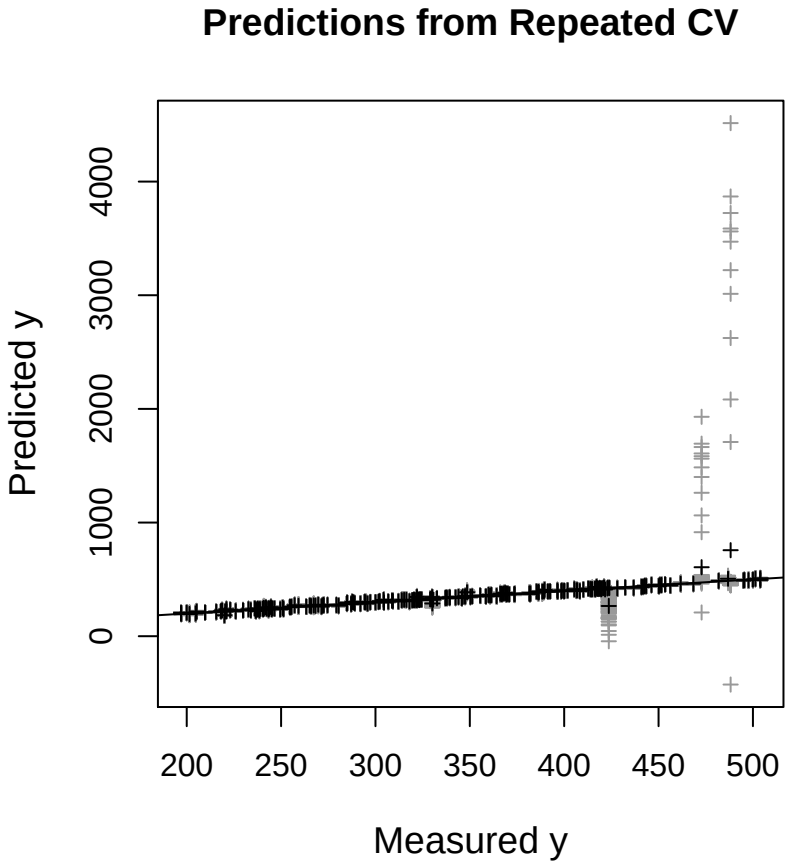


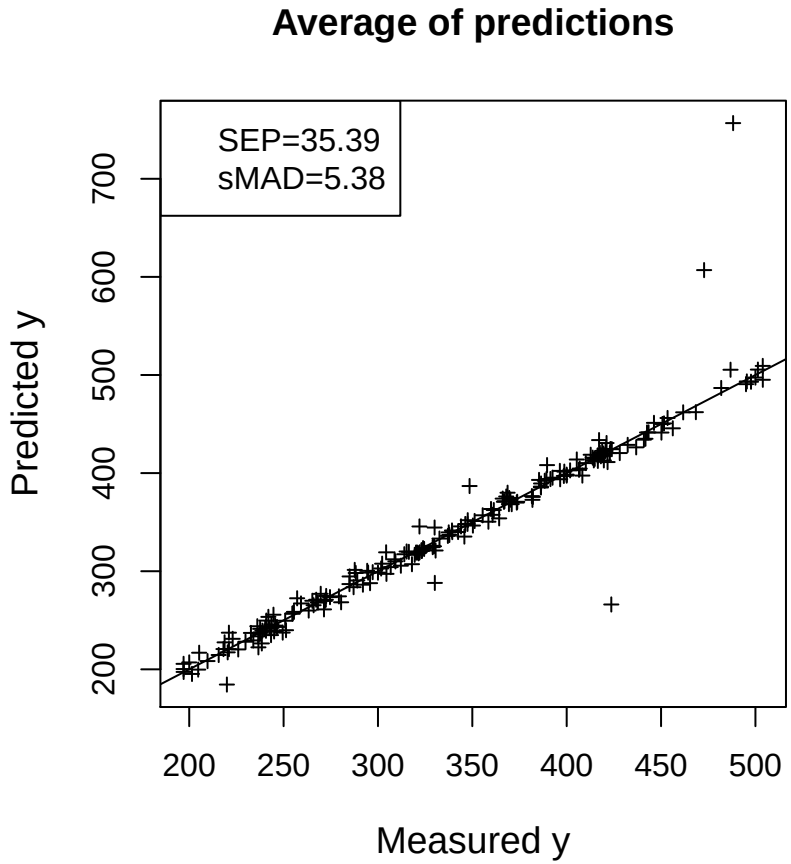
Peter.Filzmoser@tuwien.ac.at

Ridge Regression

Diagnostic plots: from repeated cross validation

```
> resRcv <- ridgeCV(y~X,data=PAC,repl=100,lambda=resR$lambdaopt)
```

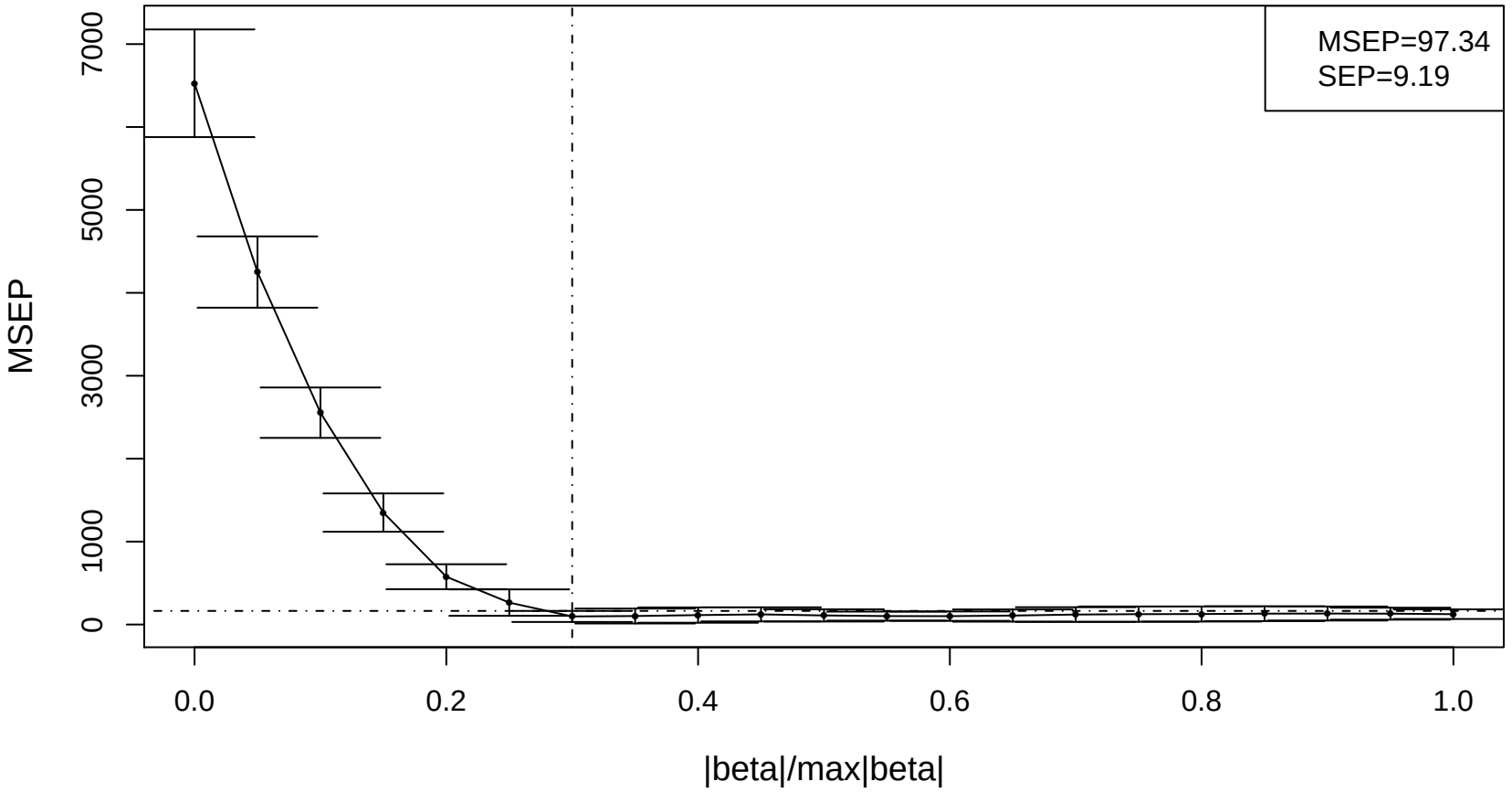




Lasso Regression

Diagnostic plot: choice of Lasso parameter

```
> resL <- lassoCV(y~X,data=PAC,K=10,fraction=seq(0,1,by=0.05))
```

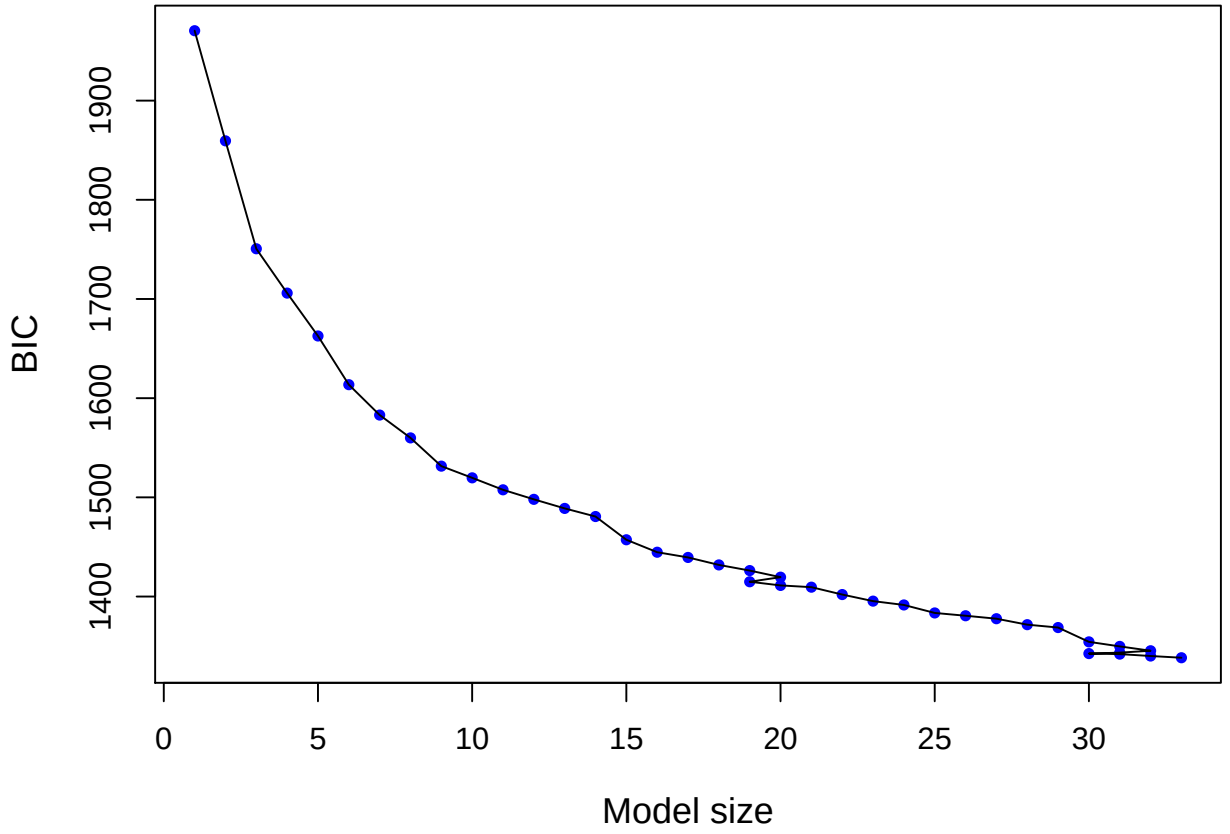


Peter.Filzmoser@tuwien.ac.at

Variable Selection by Stepwise Regression

Diagnostic plot: choice of model

```
> resS <- stepwise(y~X,data=PAC)
```

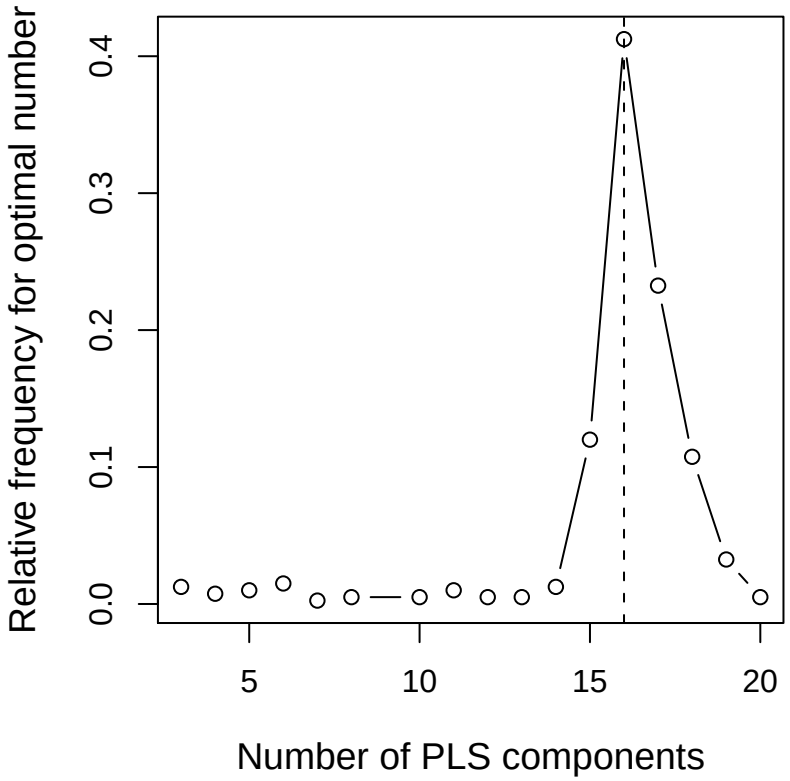


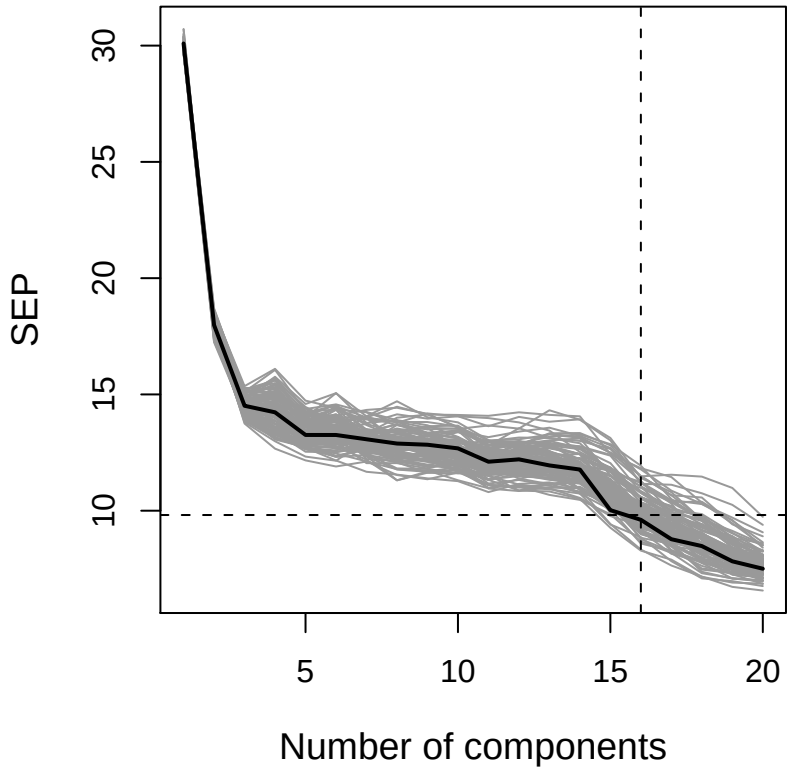
Peter.Filzmoser@tuwien.ac.at

Stepwise Regression + PLS

Diagnostic plot: choice of number of PLS components

```
> resSdcv <- mvr_dcv(y~.,ncomp=20,data=PACred,method="simpls")
```





Peter.Filzmoser@tuwien.ac.at

Comparison of Results

Method	**m***	**a**	**SEP$_{Test}$**	**SEP$_{CV}$**	**SEP$^{0.2}$**
PCR	467	21	14.2	—	7.9
PLS	467	11	12.0	—	5.7
Robust PLS	467	26	—	8.9	4.0
Ridge regression	467	—	—	28.4	4.0
Lasso regression	145	—	—	7.7	5.0
Stepwise variable selection + PLS	33	16	9.6	—	4.4

Peter.Filzmoser@tuwien.ac.at

Classification Methods

- **Linear classification methods:** linear discriminant analysis (LDA), logistic regression (LR)
- **Kernel and prototype methods:** Gaussian mixture models, k-NN classification
- **Classification trees and random forests**
- **Artficial neural networks**
- **Support vector machines**
- ...

Multivariate Classification

Example data: Origin of glass samples:

- $n = 214$ glass samples
- 6 different glass types (e.g. windows, headlamps, tableware, containers)
- $m = 9$ variables (refractive index, mass-% of Al, Ba, Ca, Fe, K, Mg, Na, Si)

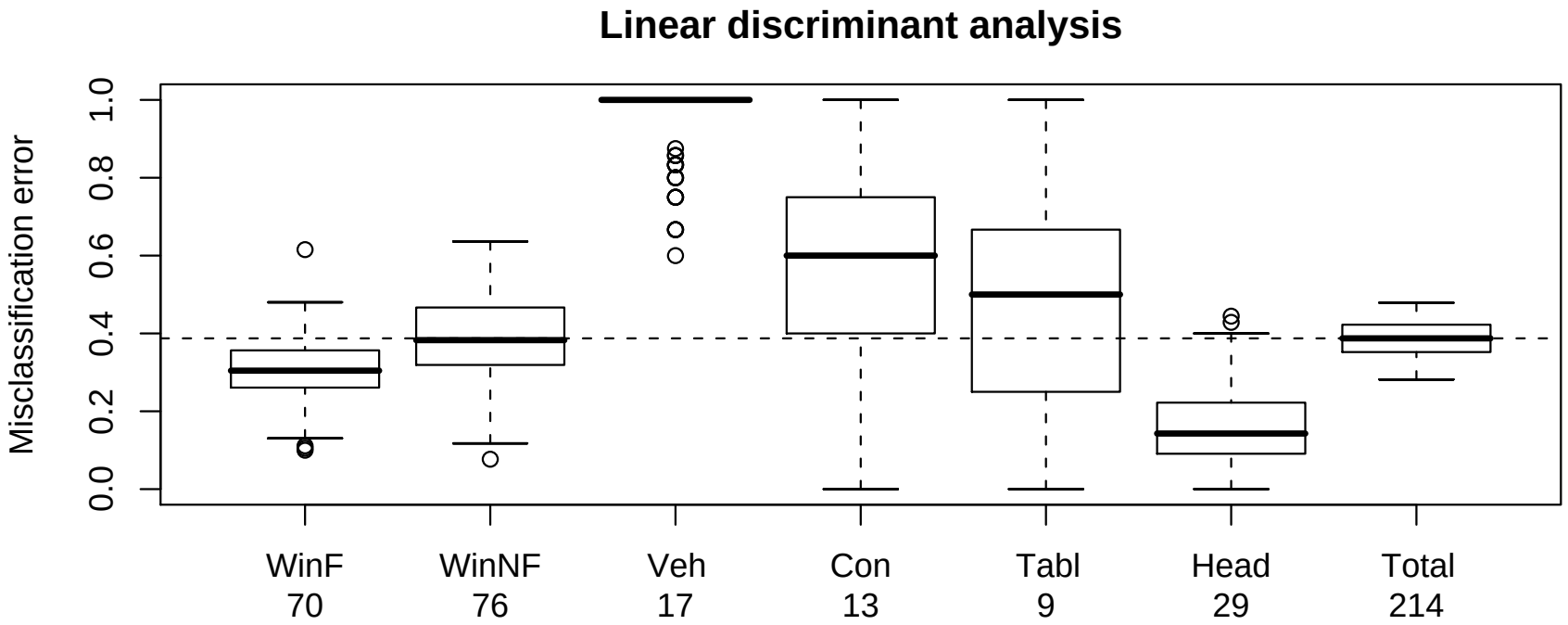
```
> library(MASS)
> data(fgl)
> grp=fgl$type
> X <- scale(fgl[,1:9])
> dim(X)

[1] 214   9
```

LDA (Linear Discriminant Analysis)

LDA: obtain LDA-rule for training data, apply to test data; repeat

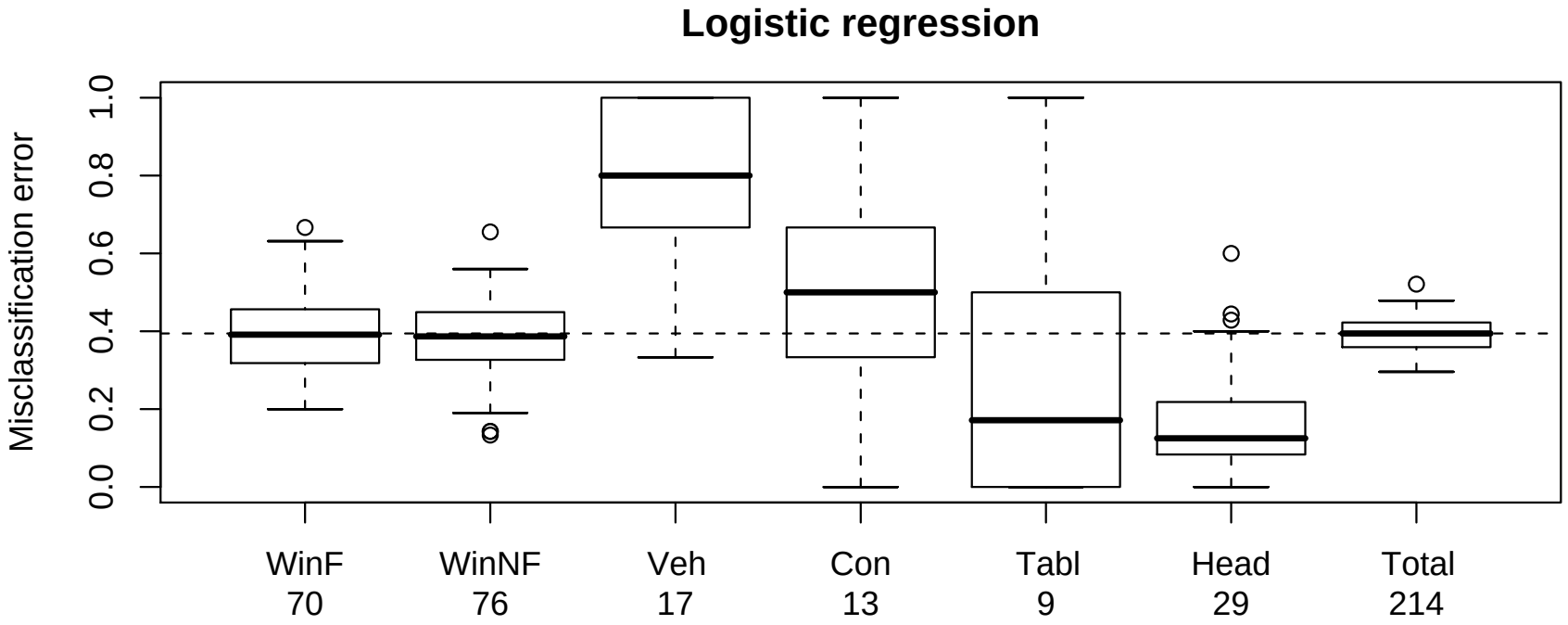
```
> train <- sample(1:n,ntrain)
> reslda <- lda(X[train,],grp[train])
> pred <- predict(reslda,newdata=X[-train,])$class
> tab <- table(grp[-train],pred)
```



LR (Logistic Regression)

LR: obtain LR-rule for training data, apply to test data; repeat

```
> train <- sample(1:n,ntrain)
> reslr <- vglm(grp ~ .,data=dat[train,],family=multinomial)
> predmix <- predict(reslr,dat[-train,],type="response")
> predgrp <- apply(predmix,1,which.max)
> tab <- table(grp[-train],predgrp)
```

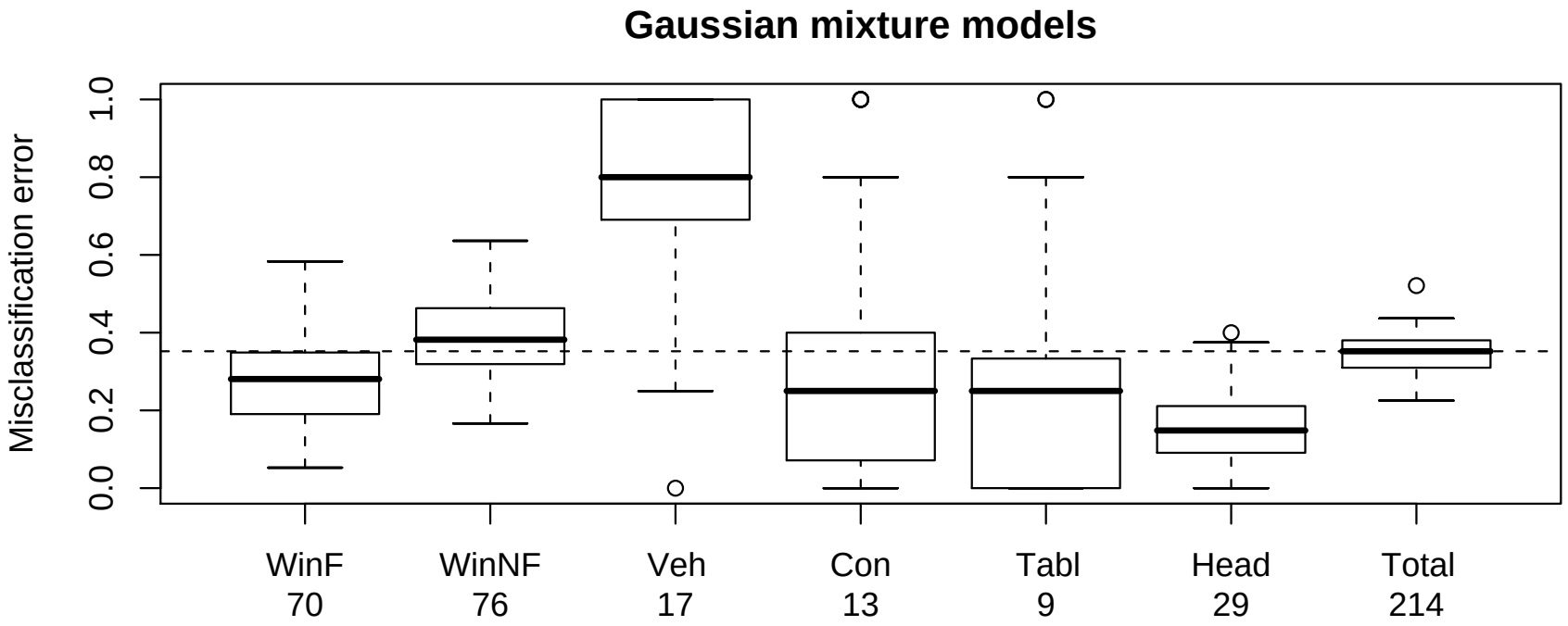


Peter.Filzmoser@tuwien.ac.at

Mix: Gaussian Mixture Models

Mix: obtain models for training data, apply to test data; repeat

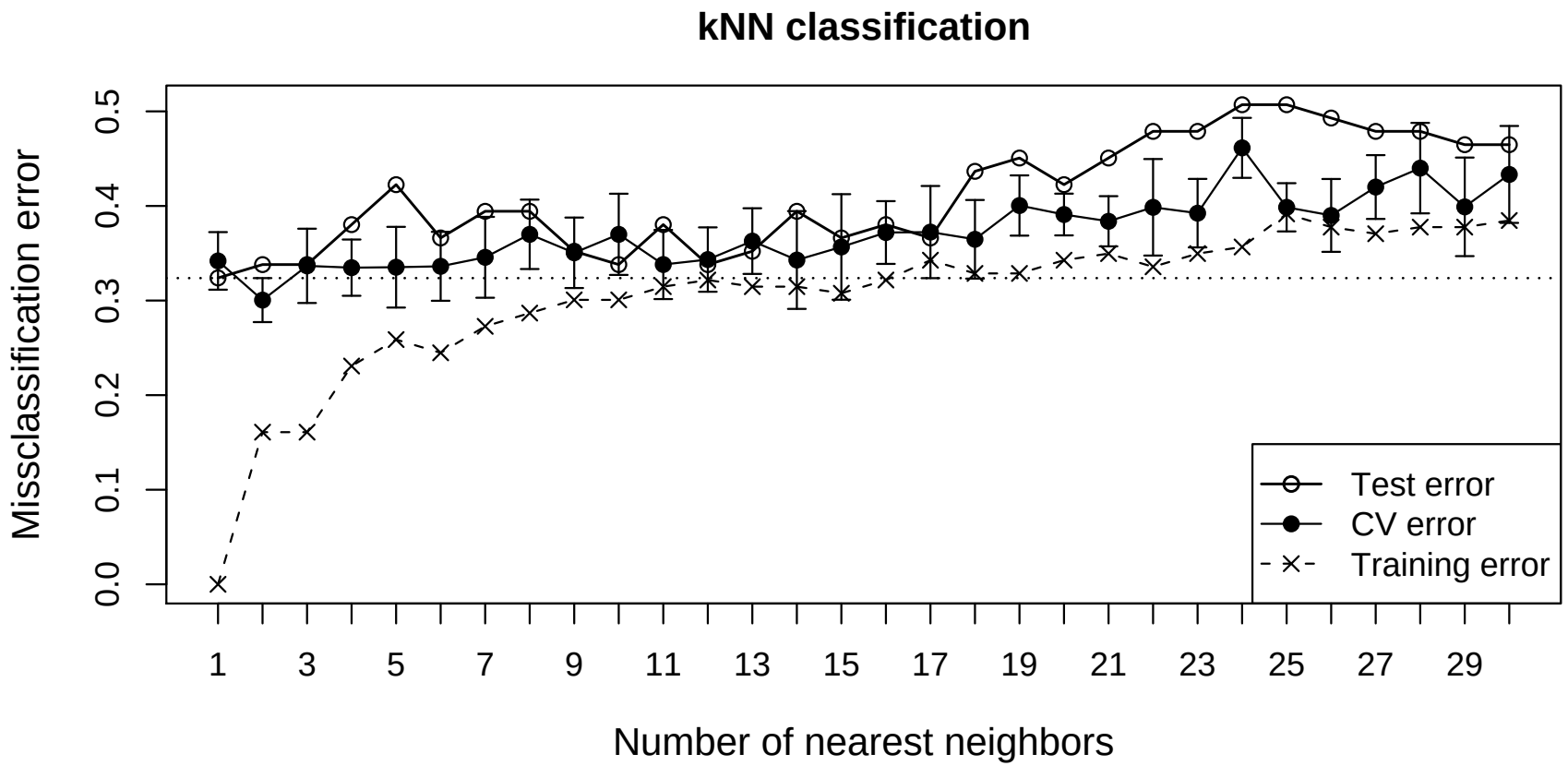
```
> train <- sample(1:n,ntrain)
> resgmm <- mda(grp ~ .,data=dat[train,])
> predgmm <- predict(resgmm,dat[-train,],type="post")
> predgrp <- apply(predgmm,1,which.max)
> tab <- table(grp[-train],predgrp)
```



kNN: k-nearest-neighbor classification

kNN: select tuning parameter "k" (number of neighbors)

```
> train <- sample(1:n,ntrain)
> resknn <- knnEval(X,grp,train,knnvec=seq(1,30,by=1))
```

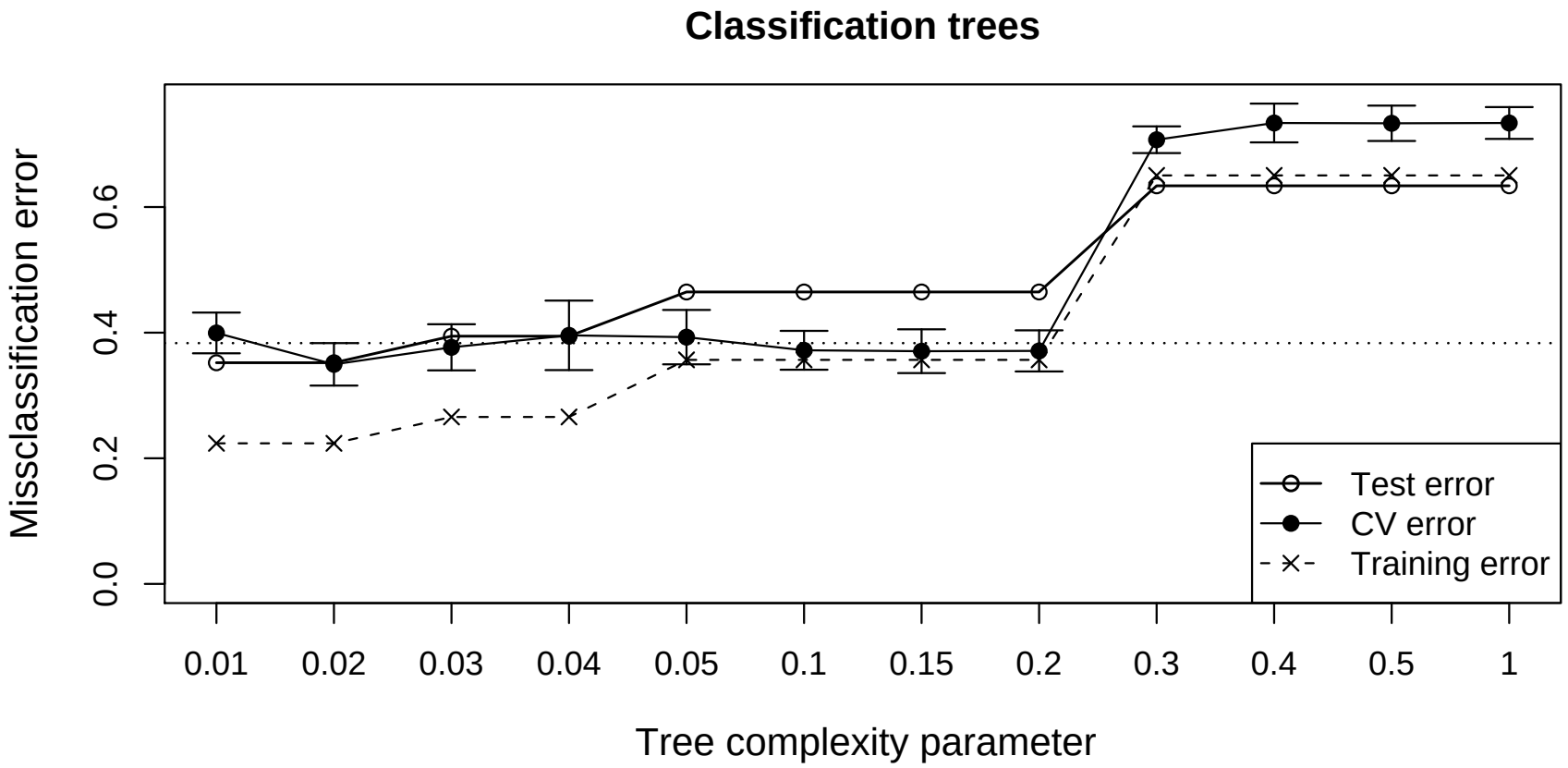


Peter.Filzmoser@tuwien.ac.at

Tree: classification trees

Tree: select tuning parameter "cp" (tree complexity)

```
> train <- sample(1:n,ntrain)
> cptry <- c(0.01,0.02,0.03,0.04,0.05,0.1,0.15,0.2,0.3,0.4,0.5,1)
> restree <- treeEval(X,grp,train,cp=cptry)
```

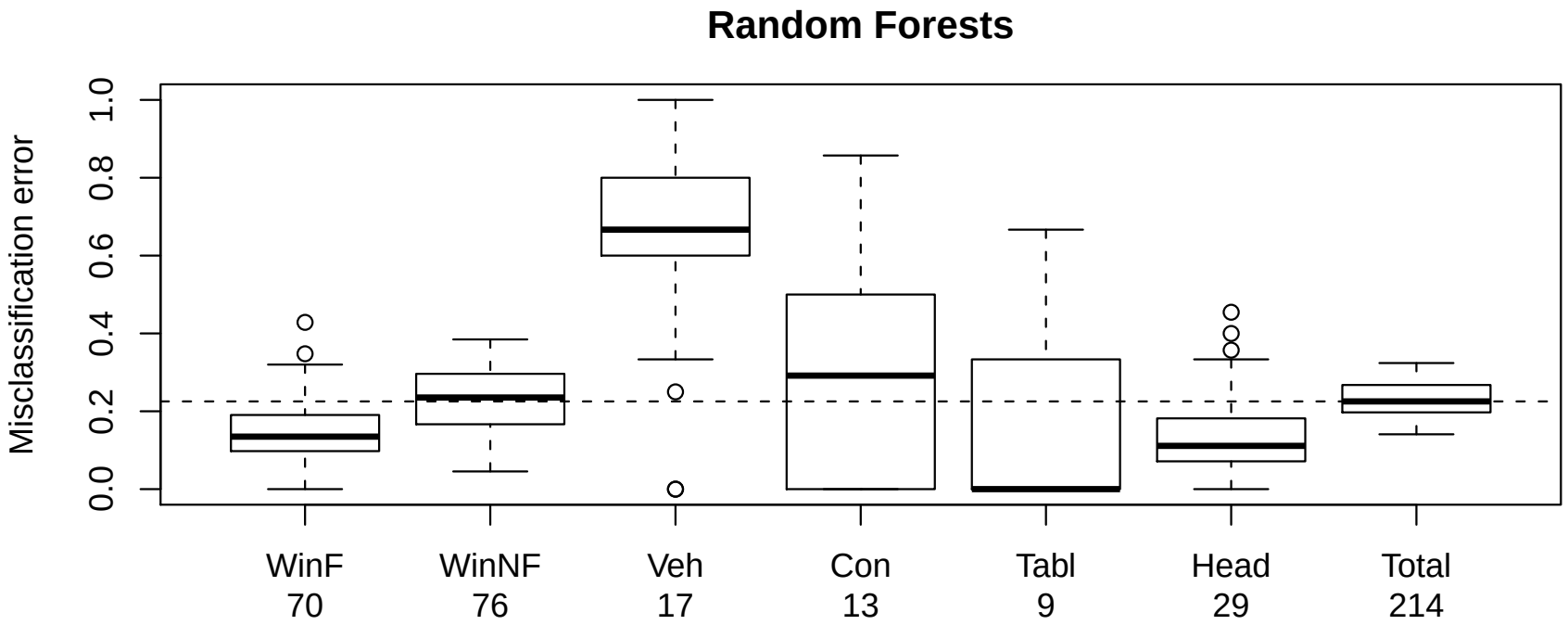


Peter.Filzmoser@tuwien.ac.at

RF: Random Forests

RF: obtain rule for training data, apply to test data; repeat

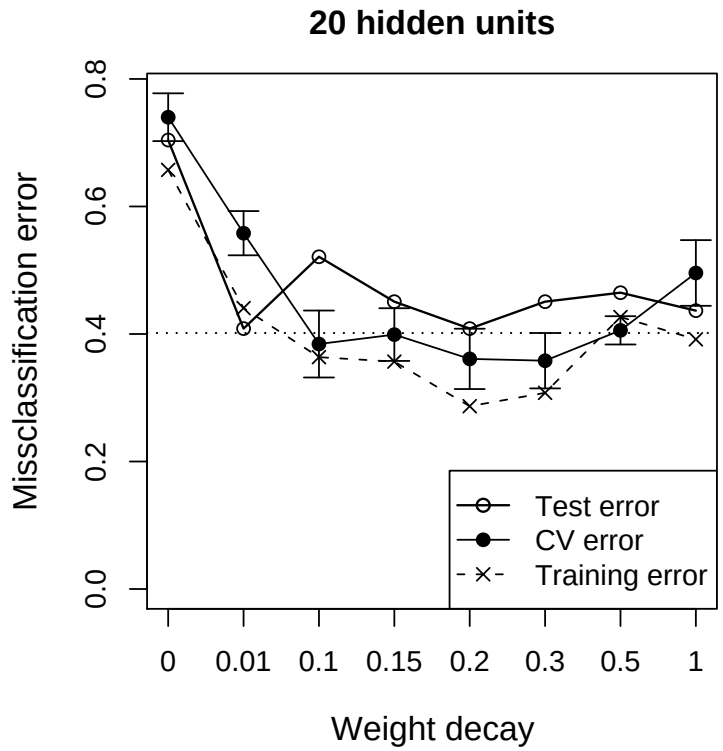
```
> train <- sample(1:n,ntrain)
> resRF <- randomForest(grp~.,data=dat,subset=train)
> predRF <- predict(resRF, dat[-train,])
> table(grp[-train],predRF)
```

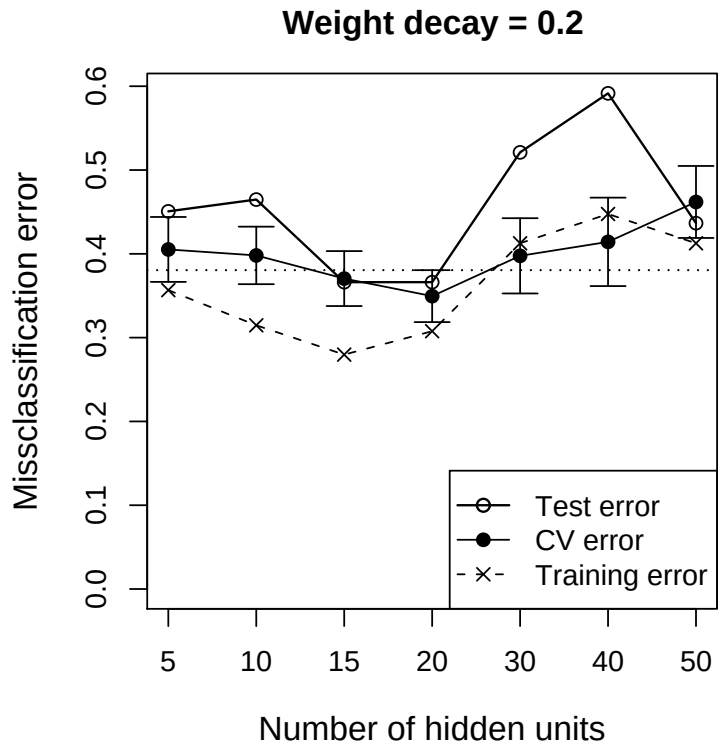


ANN: Artifician Neural Networks

ANN: select tuning parameters "weight decay" and "number of hidden units"

```
> train <- sample(1:n,ntrain)
> wd <- c(0,0.01,0.1,0.15,0.2,0.3,0.5,1)
> sz <- c(5,10,15,20,30,40,50)
> resnnet=nnetEval(X,grp,train,decay=wd,size=20)
> resnnet=nnetEval(X,grp,train,decay=0.2,size=sz)
```



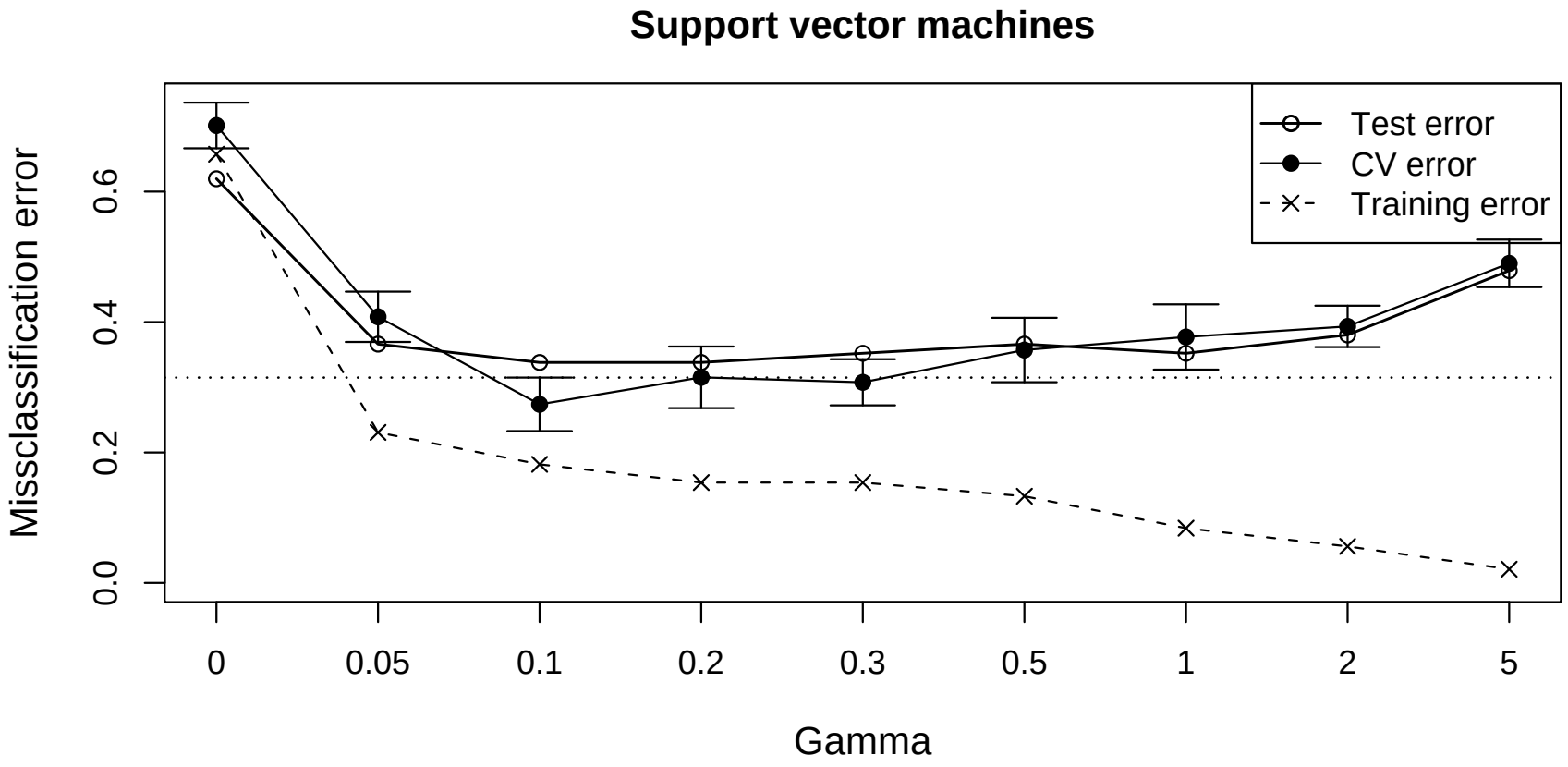


Peter.Filzmoser@tuwien.ac.at

SVM: Support Vector Machines

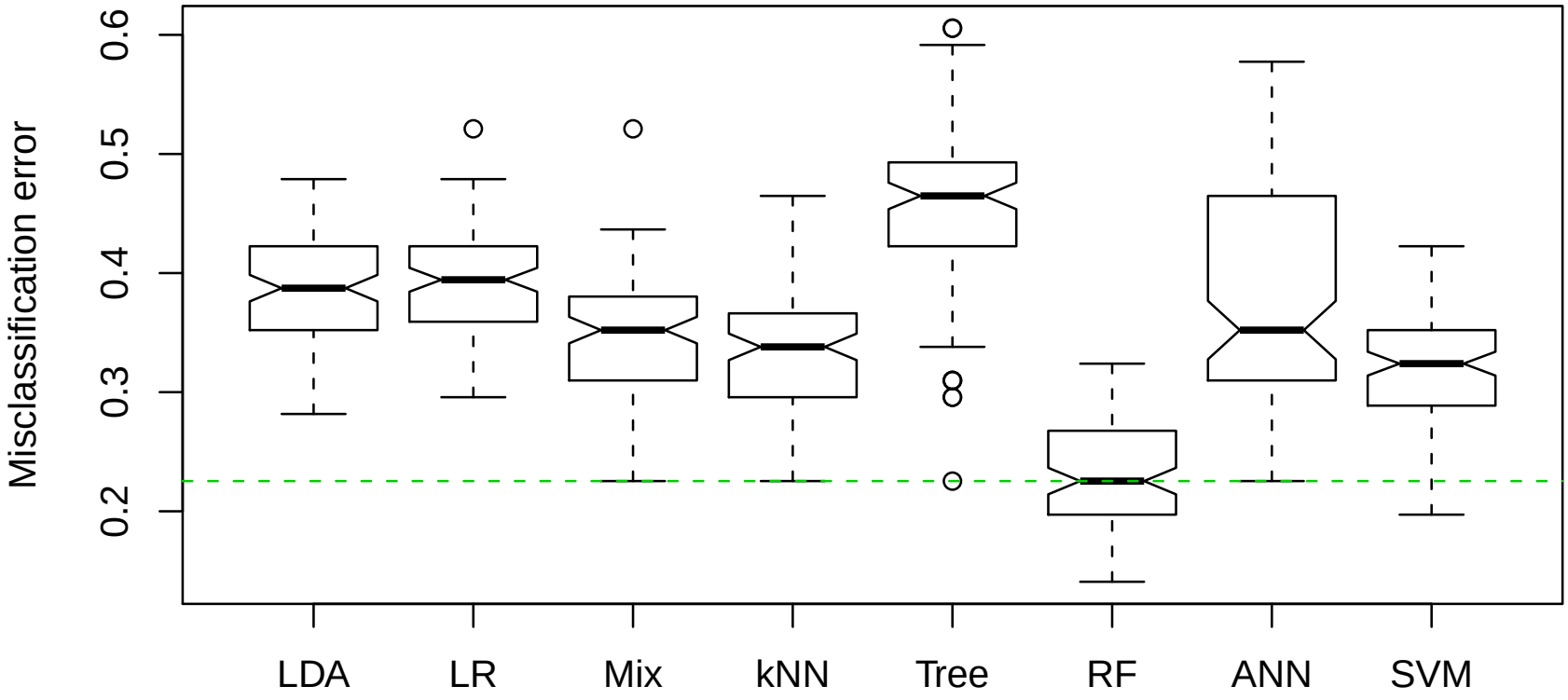
SVM: select tuning parameter "γ" (size constraint of slack variables)

```
> train <- sample(1:n,ntrain)
> gv <- c(0,0.05,0.1,0.2,0.3,0.5,1,2,5)
> ressvm <- svmEval(X,grp,train,gamvec=gv)
```



Peter.Filzmoser@tuwien.ac.at

Overall Comparison



Random Forests often work surprisingly well!

Peter.Filzmoser@tuwien.ac.at

Conclusions

- **Multivariate calibration and classification:** many algorithms are available in R – the code is always "quite" similar

  In the **chemometrics** library we tried to **unify** the code and the evaluation procedures.

- **Evaluation:** should be done carefully, but *not* only for one specific method

- **Computation time:** careful evaluation requires more time–at least for larger data sets (data collection also needs time)

Peter.Filzmoser@tuwien.ac.at