GPQR

Installation

Like many other R packages, the simplest way to obtain GPQR is to install it from github. Type the following command in R console:

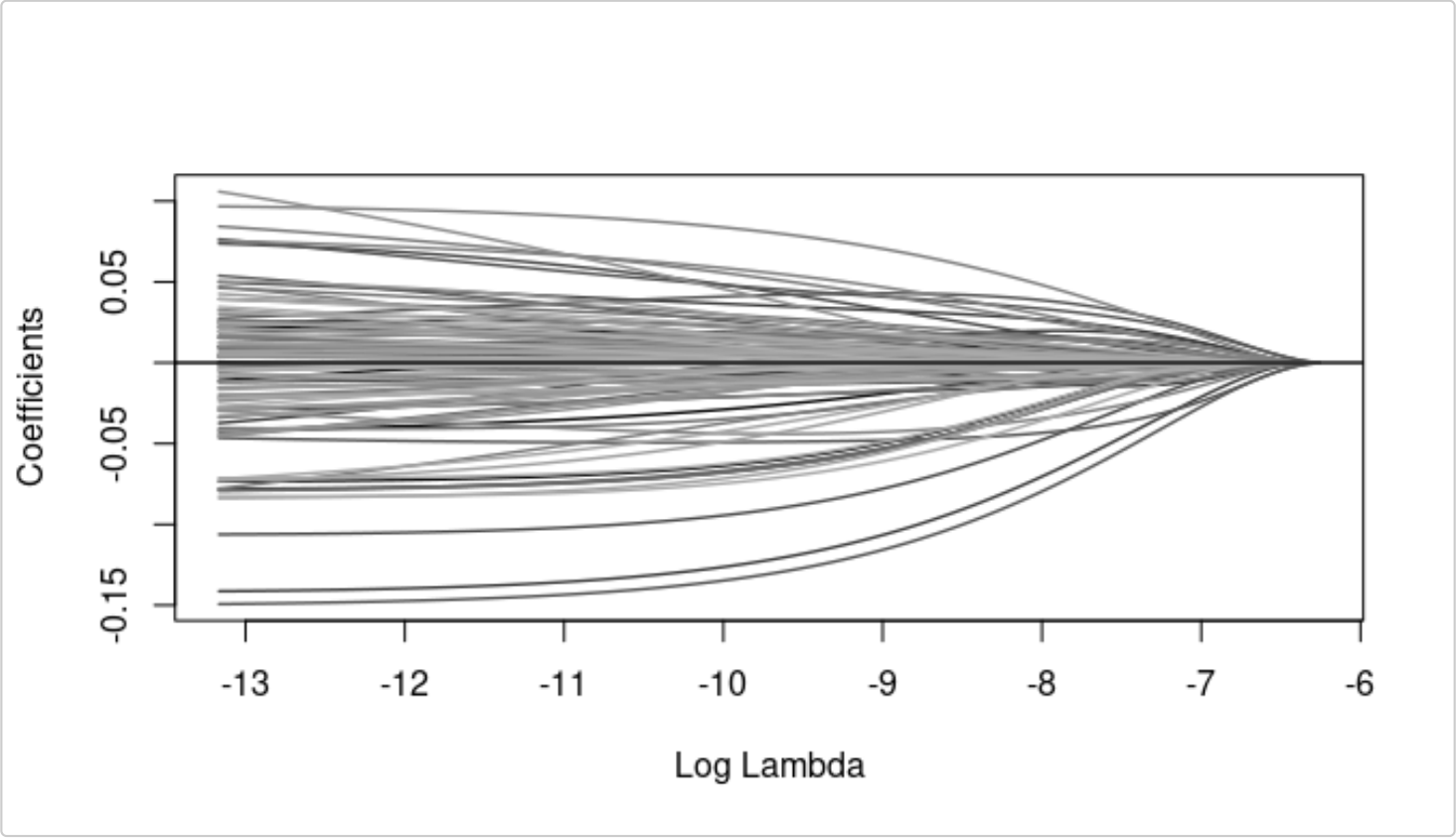
```
library(devtools)
#> Le chargement a nécessité le package : usethis
#devtools::install_github("https://github.com/ouhourane/GPQR.git")
```

In this vignette, we demonstrate how to use the GSQR() and cv.GSQR() functions in the GPQR package to fit the regularization path of parametric quantile regression with grouped penalties. This includes group selection methods such as group Lasso, group Mcp, and group Scad. The others function: predict(), coef(), cv.predict, cv.coef, ... are minor modifications or directly copied from the glmnet package.

Bardet dataset

Gene expression data (20 genes for 120 samples) from the microarray experiments of mammalian eye tissue samples of Scheetz et al. (2006). This data set contains 120 samples with 100 predictors (expanded from 20 genes using 5 basis B-splines, as described in Yang, Y. and Zou, H. (2012)).

```
library(GPQR)
#load bardet dataset
data(bardet)
group <- rep(1:20,each=5)
#run GPQR for group Lasso penaly, taux = 0.5 and  with the penalty group Lasso
#and the first pseudo-quantile approximation loss function
fit <- GPQR(x=bardet$x,y=bardet$y,group=group,method="GLasso",check="f1",taux=0.5)
# To produce a coefficient profile plot of the coefficient paths for a fitted GPQR object.
plot(fit)
```



"fit" is an object of class GPQR that contains all the relevant information of the fitted the QR for further use. To extract the extract the information from "fit", various functions are provided for the this object such as plot, print, coef and predict that enable us to run easily those tasks.

Example 1 of our paper: Group Penalized Quantile Regression

In this example, our goal is to illustrate graphically the key advantages of using group penalized quantile regression approaches to detect heterogeneous effects of predictors, as alternatives to group penalized Least-Square (LS) regression methods.

We set the sample size to $n = 100$ observations and $p = 20$ predictors. The predictors $X_j, j = 1, \ldots, 20$, were generated as follows:

Thus, we set the predictors' effects to be

$$\beta = (\underbrace{3,3,3,3}_{G_1}, \underbrace{2,2,2,2}_{G_2}, \underbrace{-1,-1,-1,-1}_{G_3}, \underbrace{0,\ldots,0}_{G_4-G_{11}})^\top$$

and $\sigma = 3$. The response Y is generated from the following location-scale linear regression model

$$Y = \sum_{j=1}^{20} \beta_j X_j + \Phi(X_{20})\epsilon, \qquad \epsilon \sim N(0,3),$$

where $\Phi(.)$ is the cumulative distribution function of the standard normal distribution.

```
library("MASS")
library("grpreg")
#>
#> Attachement du package : 'grpreg'
#> L'objet suivant est masqué depuis 'package:MASS':
#>
#>     select
# Simulation data
xlm=c(-2,4)
Ng=11
# A vector of consecutive integers describing the grouping of the coefficients
group=c(1,1,1,1,2,2,2,2,3,3,3,3,4:Ng)
n=100
p=length(group)
X=NULL
sig=3
betac=c(rep(3,4),rep(2,4),rep(-1,4),rep(0,p-12))
cc1=c(1,2,3,4);cc2=c(5,6,7,8);cc3=c(9,10,11,12);cc4=c(13:(p-1));
MuVec<-rep(0,p);v<-rep(1,p);SigmaMat<-diag(v)
for(i in 1:p){for(j in 1:p){SigmaMat[i,j]<- 0.5^abs(i-j);SigmaMat[i,i]<- 1}}
set.seed(123456789)
x<-mvrnorm(2*n,MuVec,SigmaMat,tol = 1e-6, empirical = FALSE)
# Simulation of a matrix of predictors
xx = NULL
for (h in 1:3) for (k in 1:4) xx=cbind(xx,x[,h]+rnorm(2*n,0,0.1))
xx = cbind(xx,x[,4:Ng])
Xtr=xx[1:n,]
# Simulation of a response variable
Ytr<-Xtr%*%betac+pnorm(Xtr[,p])*rnorm(n,0,sig)
dd=20;ddd=15
taux = 0.95
# Get lambda.min and lambda.1se
getmin <-function(lambda, cvm, cvsd) {
    cvmin <- min(cvm)
    idmin <- cvm <= cvmin
    lambda.min <- max(lambda[idmin])
    idmin <- match(lambda.min, lambda)
    semin <- (cvm + cvsd)[idmin]
    idmin <- cvm <= semin
    lambda.1se <- max(lambda[idmin])
    list(lambda.min = lambda.min, lambda.1se = lambda.1se)
}
```

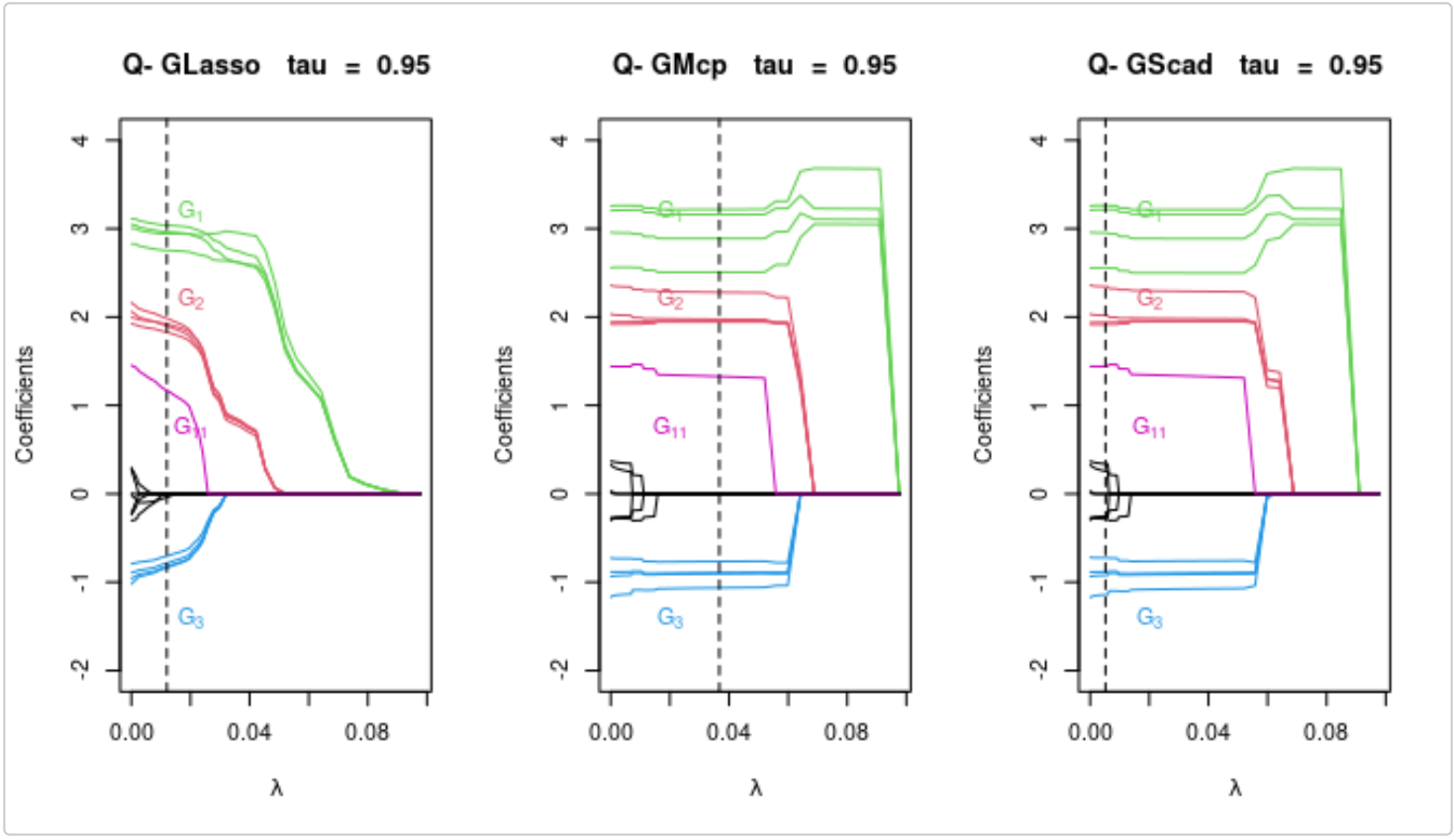
This function produce a coefficient profile plot of the coefficient paths for a fitted GPQR object.

```
GPQR_illustration <-function(penalty, taux){
  cv <- cv.GPQR(x=Xtr,y=Ytr,group=group,method=penalty,check="f1",taux=taux)
fittGL=t(cv$finalfit$beta)
```

```
seqLambda=cv$lambda
seqLambda=cv$lambda
ll = getmin(cv$lambda, cv$cv, cv$cv.error)
l = which(cv$lambda == ll$lambda.min)
main_lab = paste("Q-",penalty," ",expression(tau)," = ", taux)
matplot(seqLambda,fittGL[,cc1], type = "l",col = 3,lty = 1,ylim=xlm,lwd=1,
        ylab="Coefficients",main=main_lab, xlab = expression(lambda))
matlines(seqLambda,fittGL[,cc2], type = "l",col = 2,lty = 1,lwd=1)
matlines(seqLambda,fittGL[,cc3], type = "l",col = 4,lty = 1,lwd=1)
matlines(seqLambda,fittGL[,cc4], type = "l",col = 1,lty = 1,lwd=1)
matlines(seqLambda,fittGL[,20], type = "l",col = 6,lty = 1,lwd=1)
abline(v=seqLambda[l],col=1,lty = 2)
text(0.02,0.75,expression("G"[11]),col=6)
text(0.02,3.2,expression("G"[1]),col=3)
text(0.02,2.2,expression("G"[2]),col=2)
text(0.02,-1.4,expression("G"[3]),col=4)
}
```

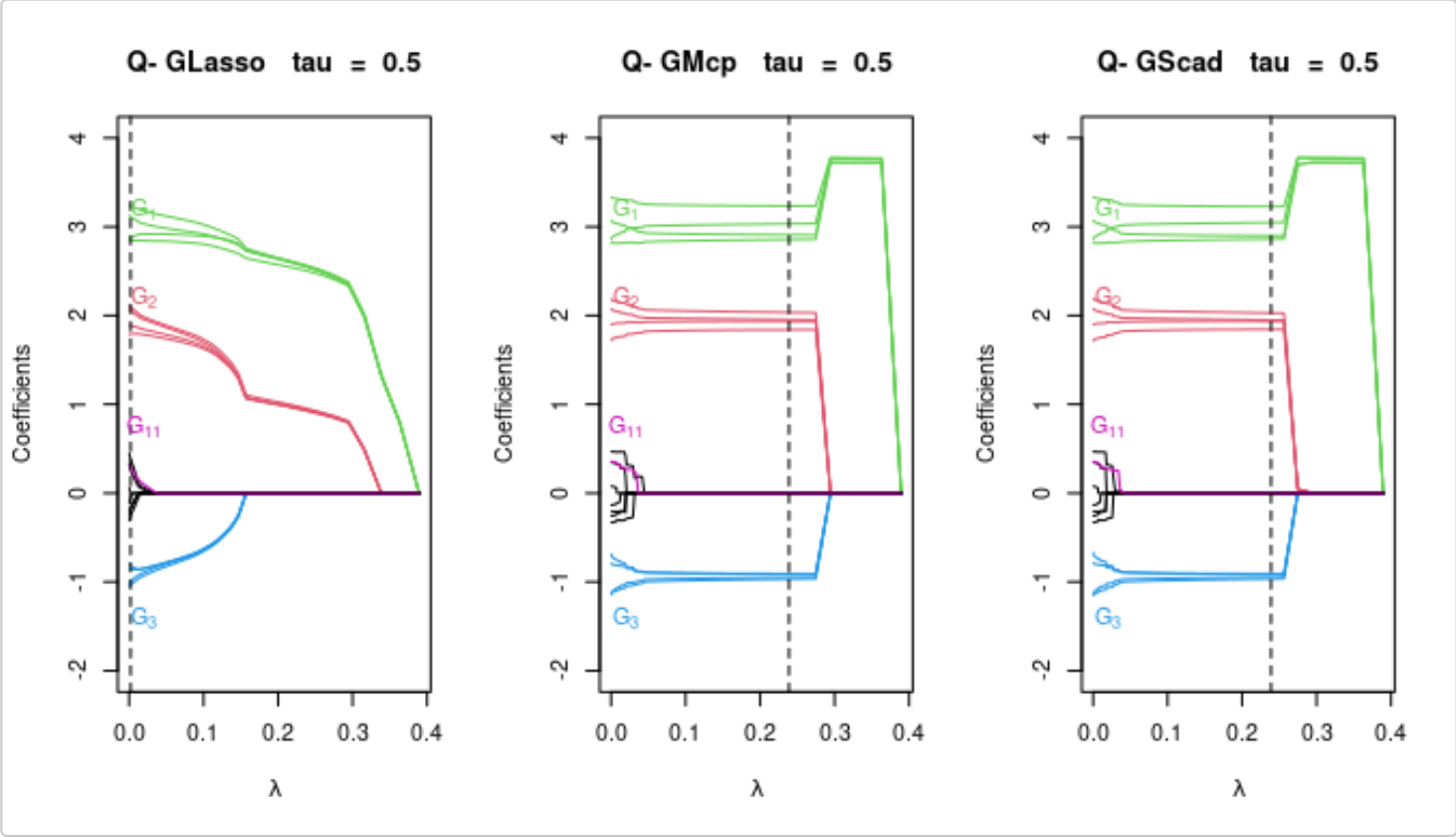
We run the function “GPQR_illustration” with different parameters.

```
par(mfrow = c(1,3))
GPQR_illustration("GLasso", 0.95)
GPQR_illustration("GMcp", 0.95)
GPQR_illustration("GScad", 0.95)
```




```
par(mfrow = c(1,3))
GPQR_illustration("GLasso", 0.50)
```

```
GPQR_illustration("GMcp", 0.50)
GPQR_illustration("GScad", 0.50)
```

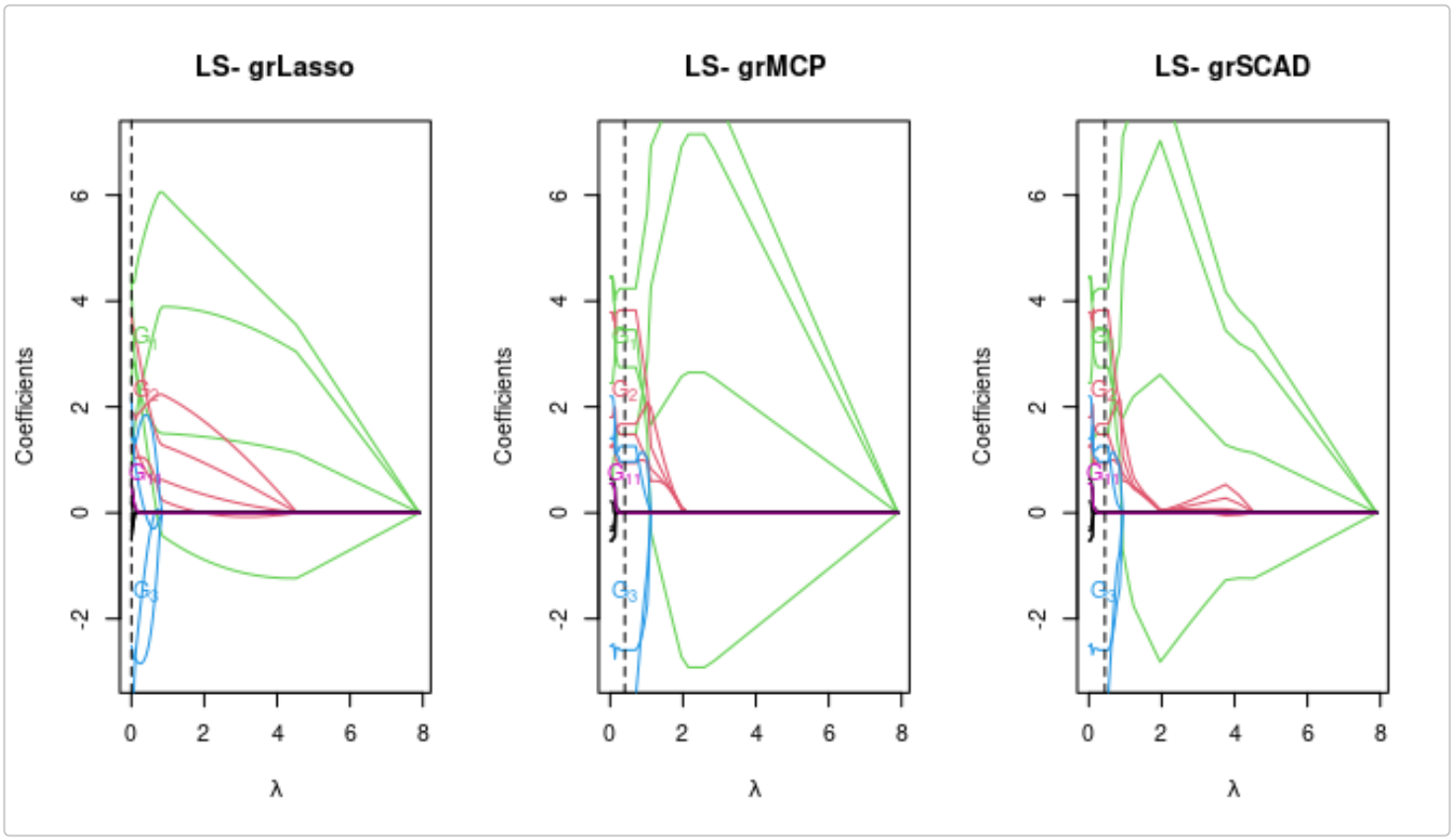


This function produce a coefficient profile plot of the coefficient paths for a fitted grpreg object.

```
grpreg_illustration <-function(penalty){
cv <- cv.grpreg(Xtr, Ytr, group, penalty=penalty)
fittL=t(cv$fit$beta)[,-1]
seqLambdaL=cv$lambda
l=cv$min
matplot(seqLambdaL,fittL[,cc1], type = "l",col = 3,lty = 1,ylim=xlm,lwd=1,
        ylab="Coefficients",main=paste("LS-",penalty) ,xlab = expression(lambda))
matlines(seqLambdaL,fittL[,cc2], type = "l",col = 2,lty = 1,lwd=1)
matlines(seqLambdaL,fittL[,cc3], type = "l",col = 4,lty = 1,lwd=1)
matlines(seqLambdaL,fittL[,cc4], type = "l",col = 1,lty = 1,lwd=1)
matlines(seqLambdaL,fittL[,20], type = "l",col = 6,lty = 1,lwd=1)
###abline(v=seqLambdaL[dd],col = 1,lty = 3)
##abline(v=seqLambdaL[dd+ddd],col = 2,lty = 3)
abline(v=seqLambdaL[l],col=1,lty = 2)
text(0.4,0.75,expression("G"[11]),col=6)
text(0.4,3.3,expression("G"[1]),col=3)
text(0.4,2.3,expression("G"[2]),col=2)
text(0.4,-1.5,expression("G"[3]),col=4)
}
```

We run the function “grpreg_illustration” with different penalities:

```
xlm=c(-3,7)
par(mfrow = c(1,3))
grpreg_illustration("grLasso")
grpreg_illustration("grMCP")
grpreg_illustration("grSCAD")
```



The figures above show that the coefficients' profiles of the GPQR with $\tau \in \{0.50, 0.95\}$ tend to be smooth, however, the LS paths fluctuate widely, and some coefficients are in opposite directions/signs to their true values.

Gene-based analysis of the DNA methylation data near the {} gene

This real example illustrates the GPQR approach performance for binary classification using DNA methylation around the {} gene, located in chromosome 8, to detect differentially methylated regions (DMRs). DMRs refer to genomic regions with significantly different methylation levels between two groups of samples. The data consists of methylation levels of 5,986 cytosine-guanine dinucleotides within a genomic region of 2 million base pairs. The methylation levels in these CpG sites (predictors) are measured in 40 samples using bisulfite sequencing

Importing "PCEV.RData" data into R from Github

```
#load "PCEV.RData" from github
load(url("https://github.com/ouhourane/GPQR/raw/main/data/PCEV.RData"))
ylab = expression(beta~'value')
```

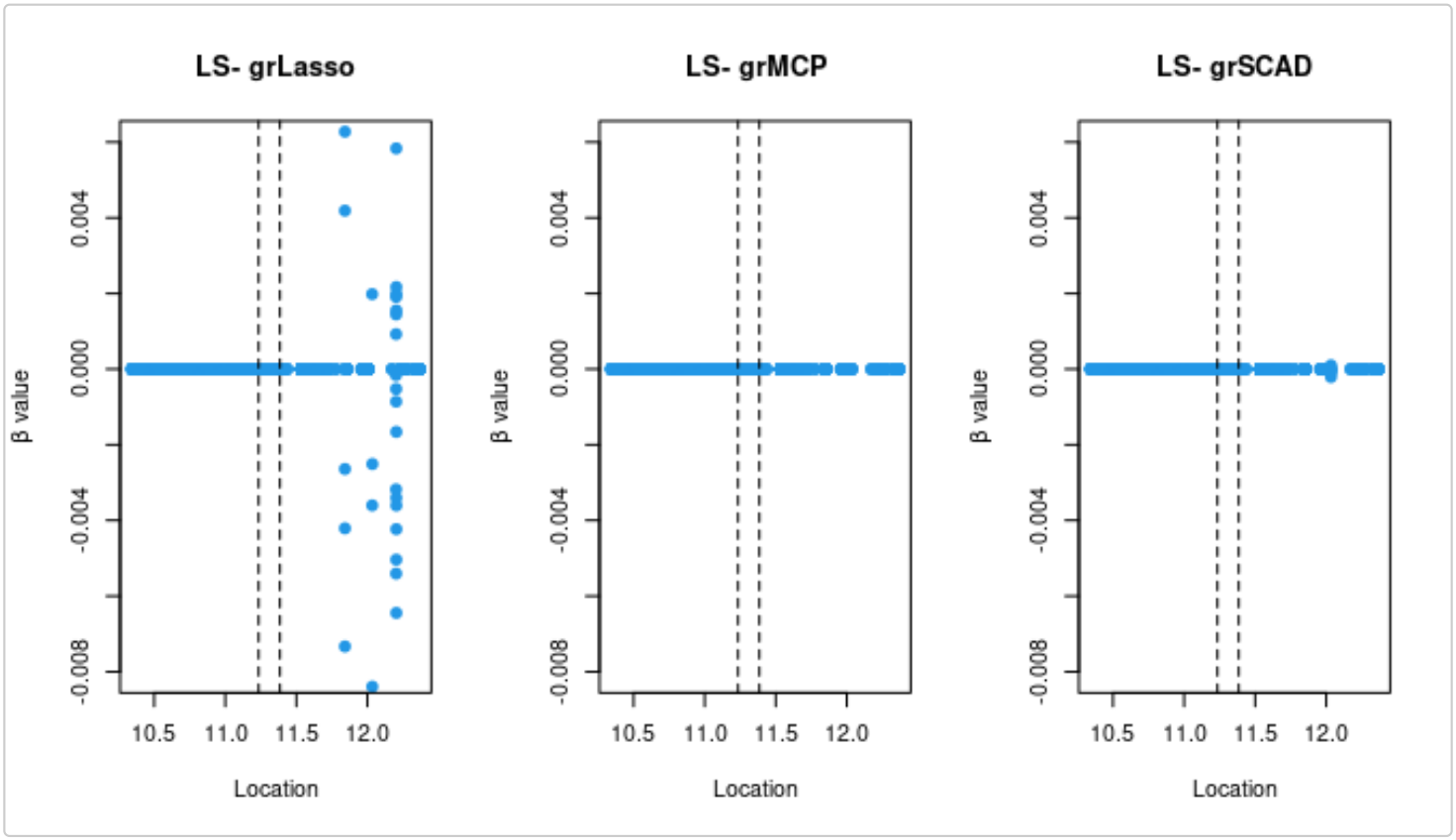
```
par(mfrow = c(1, 3))
set.seed(123)
```

The function “plot_grpreg” produce the optimal value for the regression coefficients of the grpreg-methods with the three group penalties (GMCP, GSCAD, and Glasso) are shown as a function of the genomic position.

```
plot_grpreg <- function (penalty){
  cvfit = cv.grpreg(Xc, pheno, group, penalty=penalty)
  coefGM=predict(cvfit, Xc, type="coefficients")[-1]
  matplot(positionC/1e6,coefGM,col=4, type =
        "p",pch=19,cex=1,ylab=ylab,xlab="Location",main=paste("LS-
        ",penalty),ylim=c(-0.008,0.006))
  abline(v=11.235,lty=2)
  abline(v=11.385,lty=2)
}
```

We run the function “plot_grpreg” with different penalties:

```
plot_grpreg("grLasso")
plot_grpreg("grMCP")
plot_grpreg("grSCAD")
```



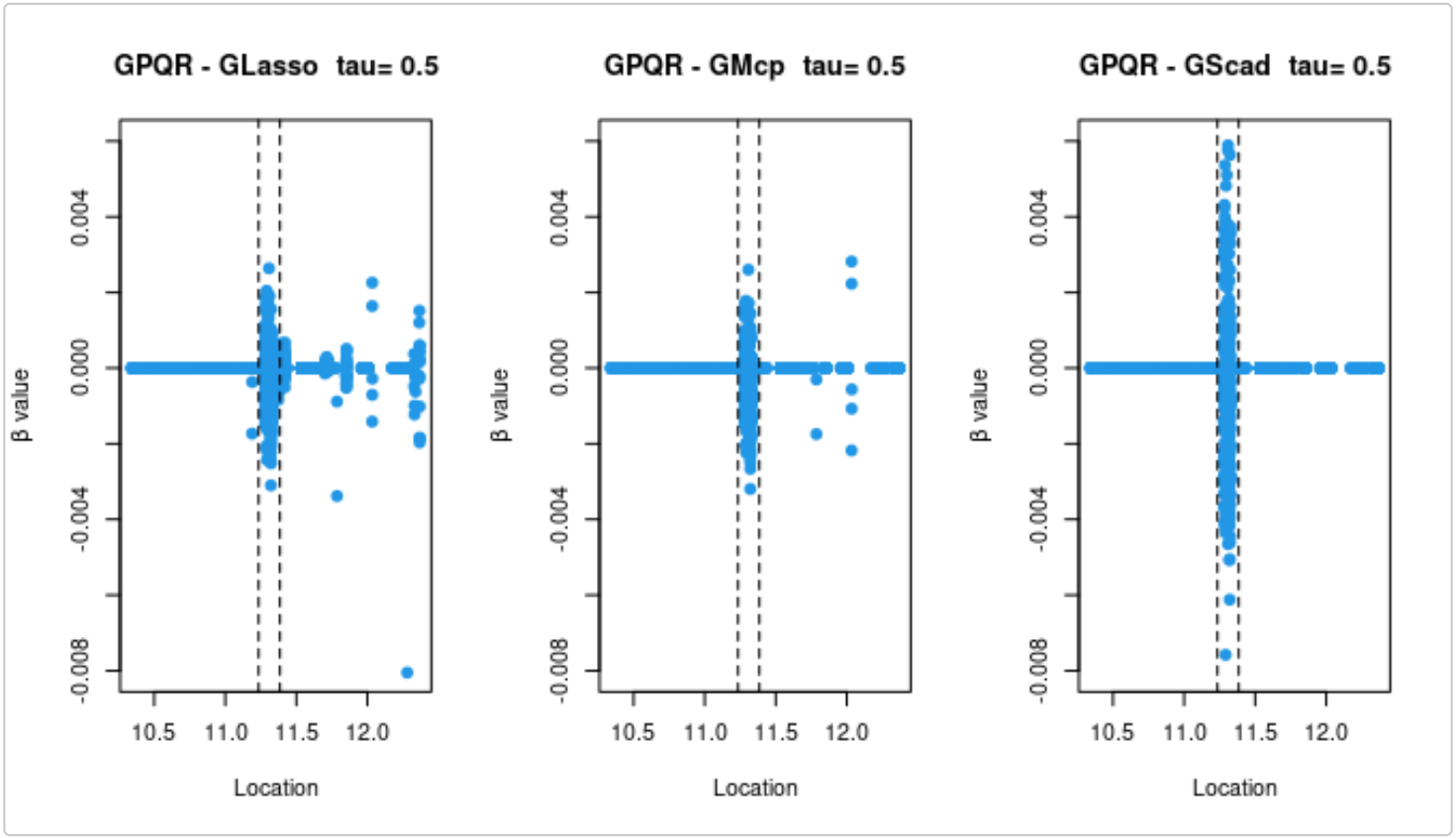
The function “plot_GPQR” produce the optimal value for the regression coefficients of the GPQR-methods with the three group penalties (GMCP, GSCAD, and Glasso) are shown as a function of the genomic position.

```
plot_GPQR <- function (penalty,taux){
  cvGS=cv.GPQR(Xc, pheno, group, Kfold = 5, taux = taux, check ="f1",method =penalty,plot.it=F)
```

```
  l=min(which(cvGS$cv==min(cvGS$cv)))
  nbeta <- c(cvGS$finalfit$b0[l], cvGS$finalfit$beta[,l])
  matplot(positionC/1e6,nbeta[-1],col=4, type = "p",pch=19,cex=1,ylab=ylab,xlab="Location",
          main= paste("GPQR -",penalty," tau=",taux),ylim=c(-0.008,0.006))
  abline(v=11.235,lty=2)
  abline(v=11.385,lty=2)
}
```

We run the function "plot_GPQR" for $\tau = 0.5$ and three penalties:

```
plot_GPQR("GLasso",0.5)
plot_GPQR("GMcp",0.5)
plot_GPQR("GScad",0.5)
```



As we can see, the region around 11.3 Mb with size 150kb (i.e., the region delimited by the two vertical lines) is detected/selected by the quantile regression methods, but not with the LS approach.