High-dimensional data analysis using penalized regression methods

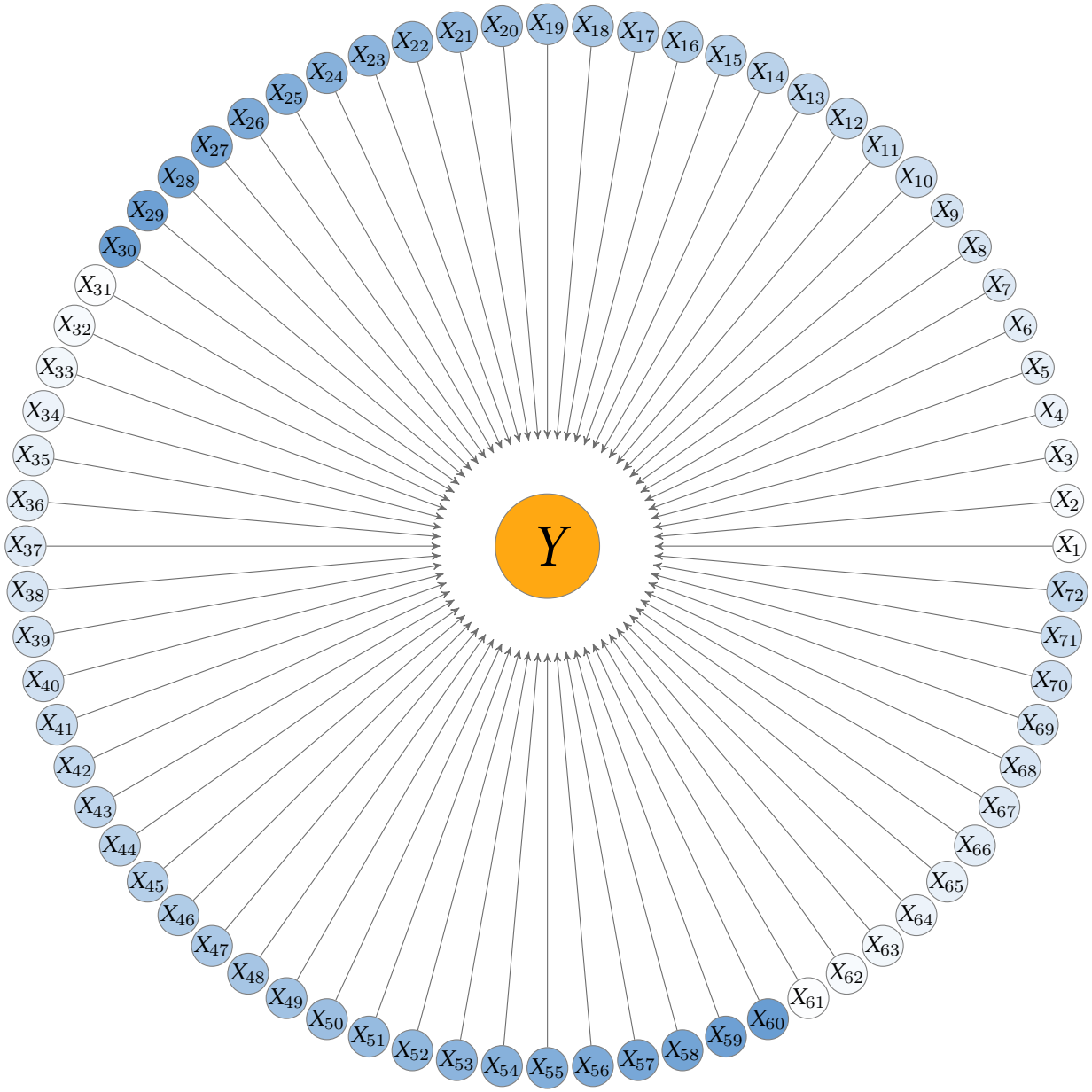
Sahir Rai Bhatnagar

Department of Epidemiology, Biostatistics, and Occupational Health
Department of Diagnostic Radiology

https://sahirbhatnagar.com/

McGill Summer School in Health Data Analytics
June 1, 2021

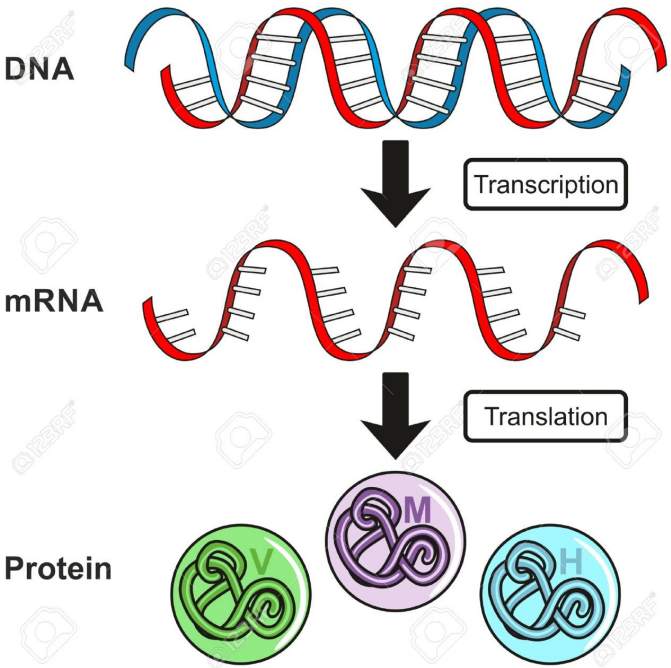
Setting

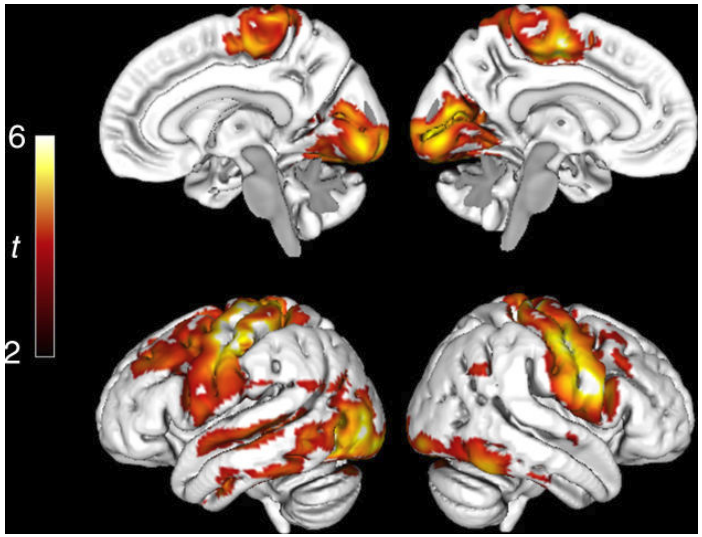

Y
X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
X13
X14
X15
X16
X17
X18
X19
X20
X21
X22
X23
X24
X25
X26
X27
X28
X29
X30
X31
X32
X33
X34
X35
X36
X37
X38
X39
X40
X41
X42
X43
X44
X45
X46
X47
X48
X49
X50
X51
X52
X53
X54
X55
X56
X57
X58
X59
X60
X61
X62
X63
X64
X65
X66
X67
X68
X69
X70
X71
X72

Setting

- This lecture concerns the analysis of data in which we are attempting to predict a vector outcome $y \in \mathbb{R}^n$ using a number of explanatory factors $\mathbf{X} = (X_1, X_2, X_3, \ldots, X_p) \in \mathbb{R}^{n \times p}$, some of which may not be particularly useful
- Although the methods we will discuss can be used solely for prediction (i.e., as a "black box"), I will adopt the perspective that we would like the statistical methods to be interpretable and to explain something about the relationship between the $\mathbf{X}$ and y
- Regression models are an attractive framework for approaching problems of this type, and the focus today will be on extending classical regression modeling to deal with high-dimensional data

High-dimensional data ($n << p$)





$$\mathbf{X}_{n\times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{12} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & x_{np} \end{bmatrix}$$

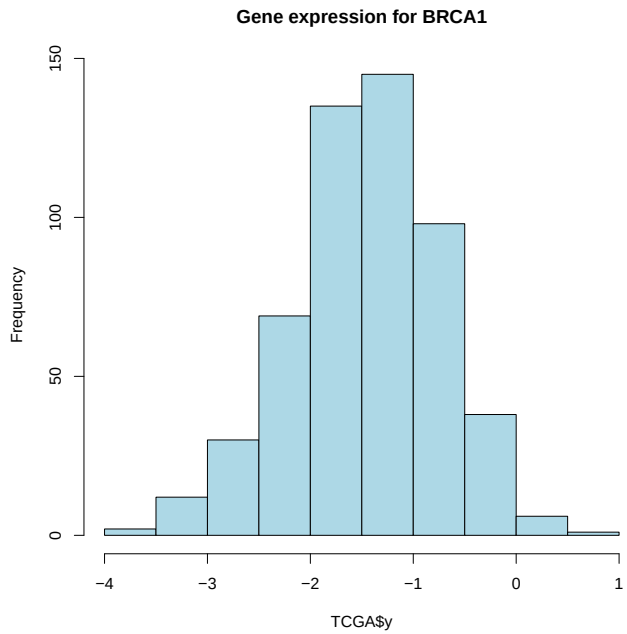
Motivating Example: The Cancer Genome Atlas (TCGA)

- The response variable in our analysis is expression of BRCA1, the first gene identified to increase the risk of early onset breast cancer
- In the dataset, expression measurements of 17,322 additional genes from 536 patients are available (and measured on the log scale)
- Because BRCA1 is likely to interact with many other genes, including tumor suppressors and regulators of the cell division cycle, it is of interest to find genes with expression levels related to that of BRCA1

```r
# install.packages("pacman")
pacman::p_load_gh('sahirbhatnagar/mcgillHDA')
library(mcgillHDA)
data(TCGA)
# help(TCGA)
str(TCGA)
```

```
## List of 3
##  $ X    : num [1:536, 1:17322] -1.45 -2.3 -1.94 -2.1 -1.28 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:17322] "15E1.2" "2'-PDE" "7A5" "A1BG" ...
##  $ y    : num [1:536] -1.661 -1.388 -1.925 -1.656 -0.358 ...
##  $ fData:'data.frame':^^I17322 obs. of  2 variables:
##   ..$ chromosome: chr [1:17322] NA NA NA "19" ...
##   ..$ gene_name : chr [1:17322] NA NA NA "alpha-1-B glycoprotein" ...
```

```r
hist(TCGA$y, col = 'lightblue', main = "Gene expression for BRCA1")
```



Multivariable Linear Regression on Training Set

```r
set.seed(101) # for reproducibility
sample <- sample.int(n = nrow(TCGA$X), size = floor(.80*nrow(TCGA$X)), replace = F) # 80% training / 20% testing
X.train <- TCGA$X[sample, ] ; dim(X.train)
```

```
## [1]   428 17322
```

```r
X.test  <- TCGA$X[-sample, ] ; dim(X.test)
```

```
## [1]   108 17322
```

```r
y.train <- TCGA$y[sample]
y.test <- TCGA$y[-sample]

# fit linear regression on training
fit.train <- lm.fit(x = X.train, y = y.train)
beta_hat_lm <- coef(fit.train)
table(is.na(beta_hat_lm))
```

```
##
## FALSE  TRUE
##   428 16894
```

```r
all.equal(fitted(fit.train), y.train)
```

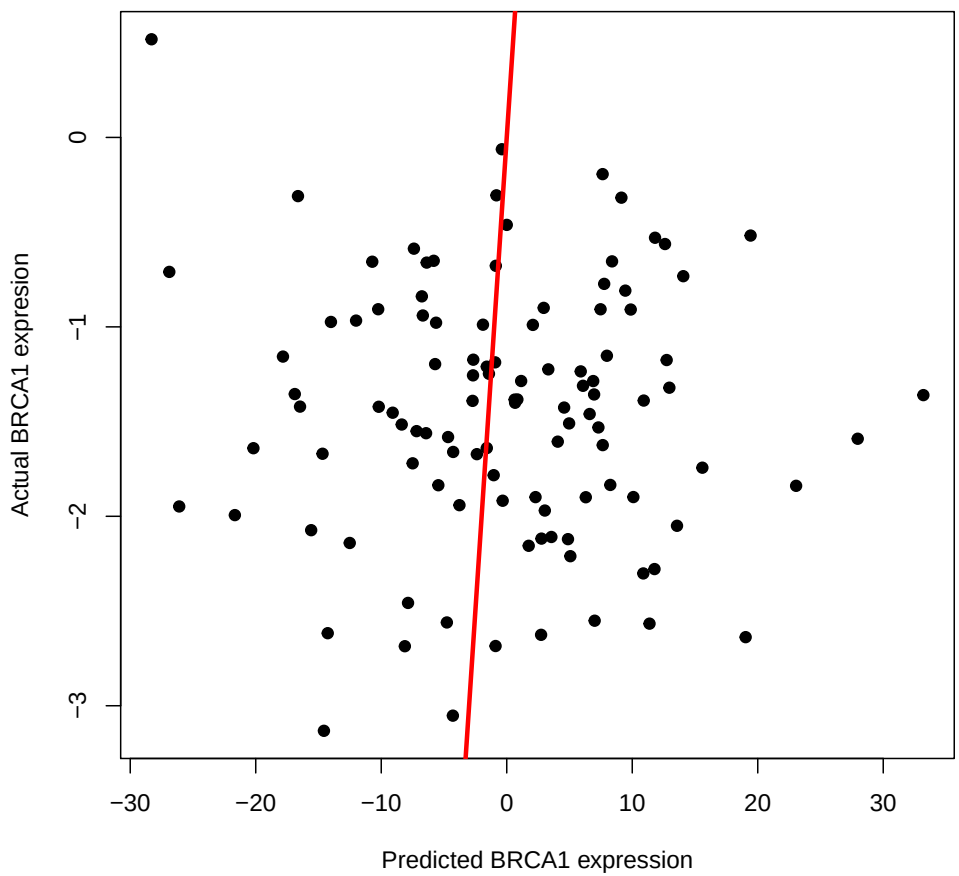
```
## [1] TRUE
```

```r
residuals(fit.train) # y_actual - y_predicted
```

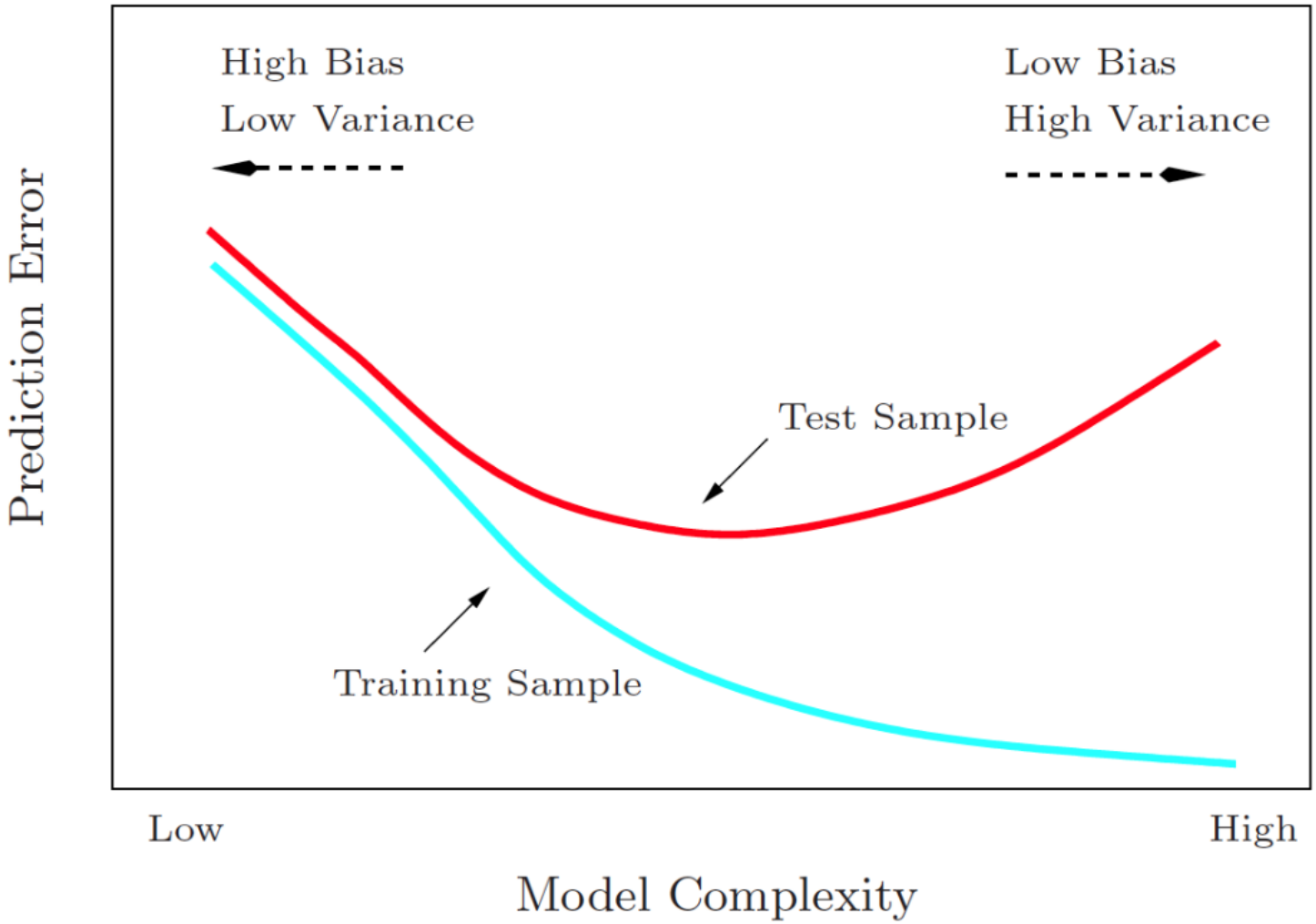
```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [186] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [223] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [260] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [297] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Multivariable Linear Regression on Testing Set

```
active_beta_ind <- which(!is.na(beta_hat_lm)) # which genes are active i.e. not NA
yhat.test <- X.test[,active_beta_ind] %*% beta_hat_lm[active_beta_ind] # predicted BRCA1 expression in test set
(mse.lm <- mean((yhat.test - y.test)^2)) # test set mean squared error

## [1] 123.895

plot(yhat.test, y.test, ylab = "Actual BRCA1 expresion", xlab = "Predicted BRCA1 expression", pch = 19)
abline(a=0,b=1, col = "red", lwd = 3)
```

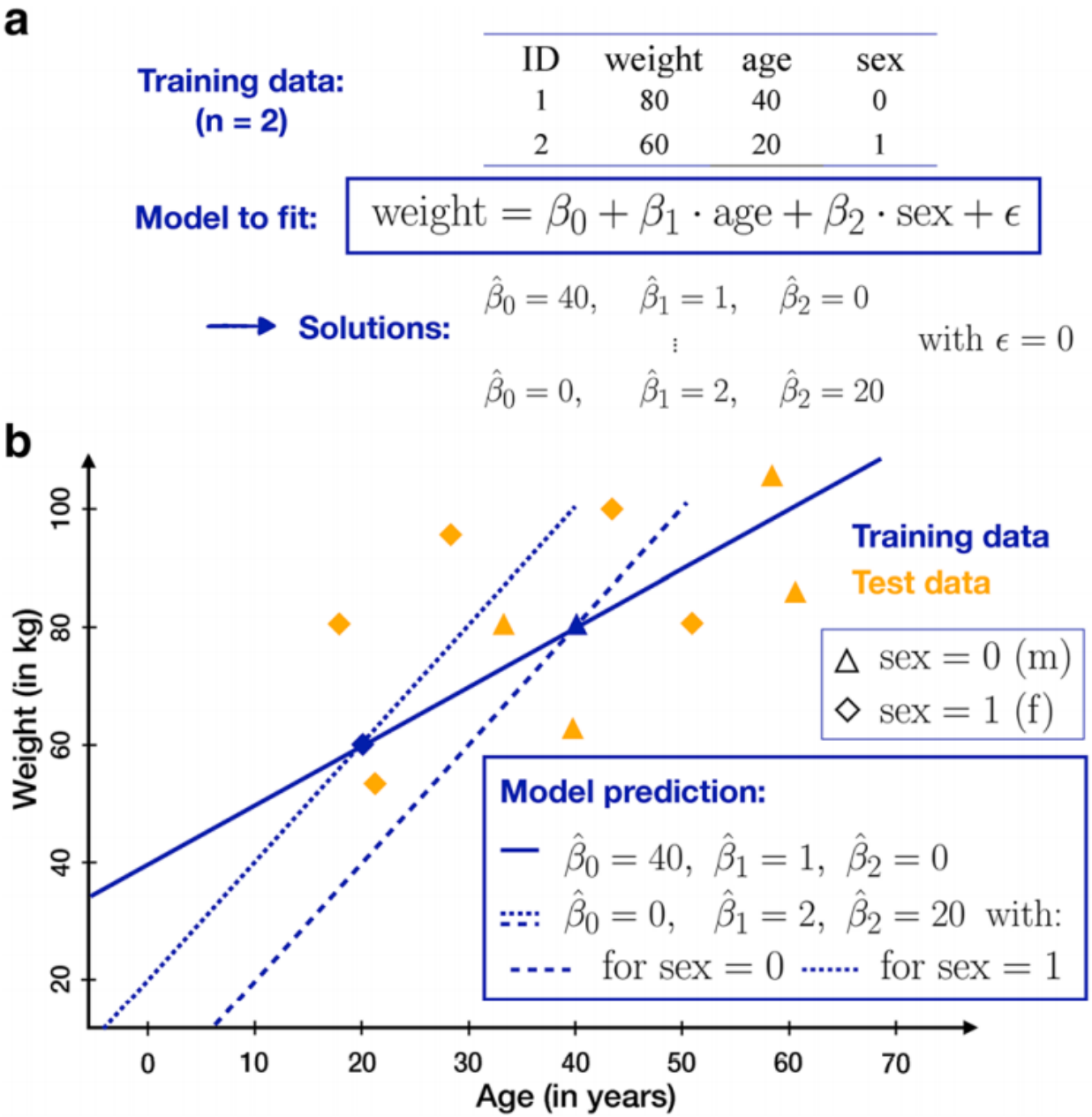


A fundamental picture for data science



ESL, Hastie et al. 2009

Issue with Linear Regression on High-dimensional data?



Boulesteix et al., Human Genetics, 2019

High-dimensional data ($n << p$)

- Throughout the course, we will let
  - n denote the number of independent sampling units (e.g., number of patients)
  - p denote the number of features recorded for each unit
- In high-dimensional data, p is large with respect to n
  - This certainly includes the case where $p > n$
  - However, the ideas we discuss in this course are also relevant to many situations in which $p < n$; for example, if $n = 100$ and $p = 80$, we probably don't want to use ordinary least squares

Classical Methods

- A nice and powerful toolbox for analyzing the more traditional datasets where the sample size (N) is far **greater than** the number of covariates (p):
  - linear regression, logistic regression, LDA, QDA, glm,
  - regression spline, smoothing spline, kernel smoothing, local smoothing, GAM,
  - Neural Network, SVM, Boosting, Random Forest, ...

$$\mathbf{X}_{n\times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{12} & \cdots & x_{1p} \\ x_{31} & x_{12} & \cdots & x_{1p} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{12} & \cdots & x_{np} \end{bmatrix}$$

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa

Classical Linear Regression

Data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ iid from

$$y = \mathbf{x}^T \boldsymbol{\beta} + \epsilon$$

where $E(\epsilon|\mathbf{x}) = 0$, and $\dim(x) = p$. To include an intercept, we can set $\mathbf{x}_1 \equiv 1$. Using Matrix notation:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \epsilon$$

The least squares estimator

$$\widehat{\boldsymbol{\beta}}_{LS} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

$$\widehat{\boldsymbol{\beta}}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

- **Question:** How to find the important variables $\mathbf{x}_j$?

Best-subset Selection (Beal et al. 1967, Biometrika)

Predictor set	model
None of $x_1\ x_2\ x_3\ x_4$	$E(Y) = \beta_0$
x_1	$E(Y) = \beta_0 + \beta_1 x_1$
x_2	$E(Y) = \beta_0 + \beta_2 x_2$
x_3	$E(Y) = \beta_0 + \beta_3 x_3$
x_4	$E(Y) = \beta_0 + \beta_4 x_4$
$x_1\ x_2$	$E(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2$
$x_1\ x_3$	$E(Y) = \beta_0 + \beta_1 x_1 + \beta_3 x_3$
$x_1\ x_4$	$E(Y) = \beta_0 + \beta_1 x_1 + \beta_4 x_4$
$x_2\ x_3$	$E(Y) = \beta_0 + \beta_2 x_2 + \beta_3 x_3$
$x_2\ x_4$	$E(Y) = \beta_0 + \beta_2 x_2 + \beta_4 x_4$
$x_3\ x_4$	$E(Y) = \beta_0 + \beta_3 x_3 + \beta_4 x_4$
$x_1\ x_2\ x_3$	$E(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$
$x_1\ x_2\ x_4$	$E(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_4 x_4$
$x_1\ x_3\ x_4$	$E(Y) = \beta_0 + \beta_1 x_1 + \beta_3 x_3 + \beta_4 x_4$
$x_2\ x_3\ x_4$	$E(Y) = \beta_0 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$
$x_1\ x_2\ x_3\ x_4$	$E(Y) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$

Which variables are important?

- An old Idea: try all possible subset models and pick the best one.
- Fit a subset of predictors to the linear regression model.
- Let S be the subset predictors, e.g., $S = \{x_1, x_3, x_7\}$, and $RSS = (\hat{y} - y)^2$. Mallow's C_p statistic is given by

$$C_p = \underbrace{\frac{\text{RSS}_S}{\sigma^2}}_{\text{model fit}} + \underbrace{2|S|}_{\text{model complexity}}$$

  where $|S|$ is the number of predictors in the set S
- We pick the model with the smallest C_p value.

Remarks on Best Subset Selection

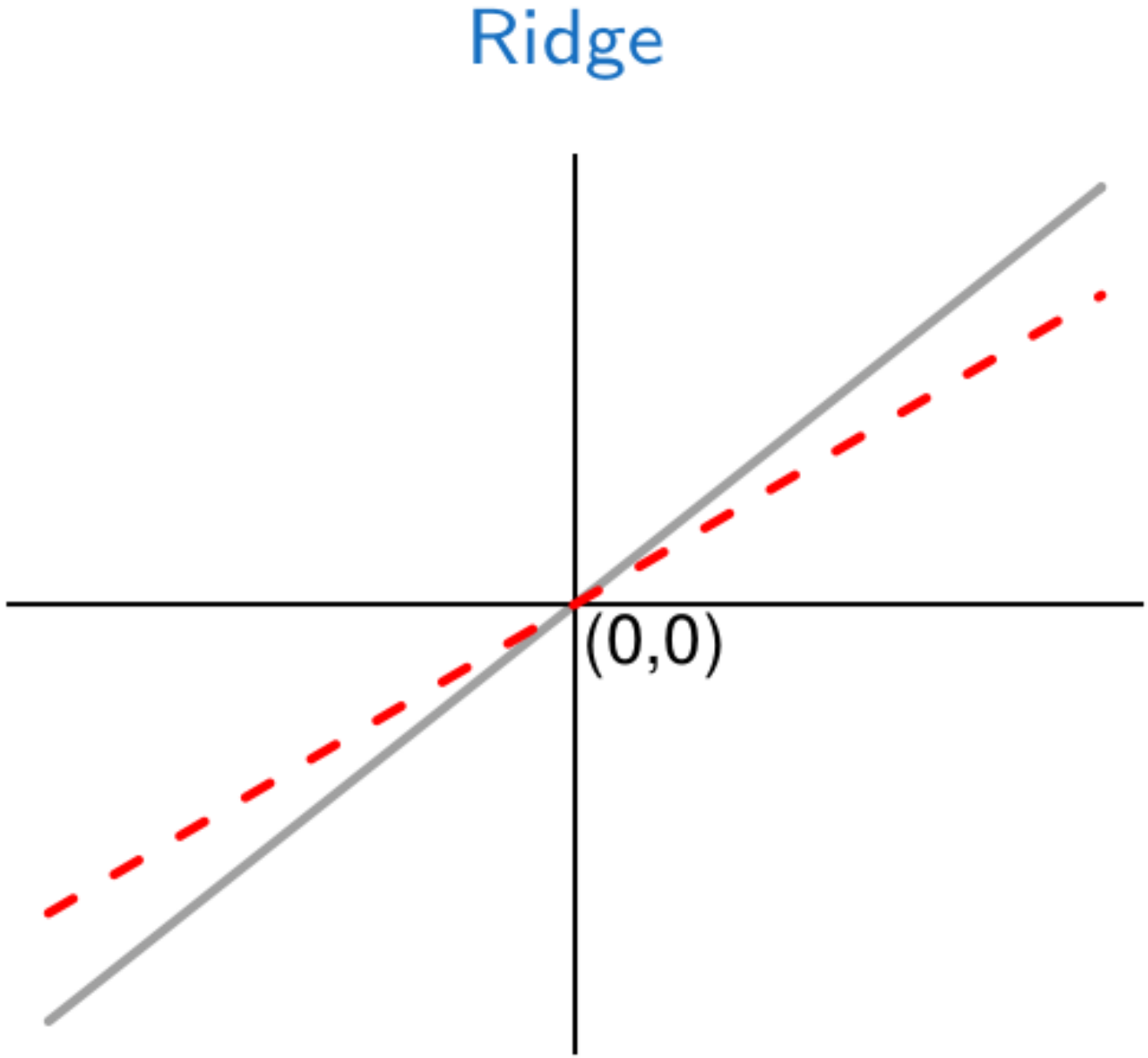
- Computing all possible subset models is a combinatorial optimization problem (NP hard)
- Instability in the selection process (Breiman, 1996)
- UPDATE: There has been recent work (2016 to present) in the statistics literature looking at efficient ways to solve this best-subset selection problem
  - Best Subset Selection via a Modern Optimization Lens (https://arxiv.org/pdf/1507.03133.pdf)
  - Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms (https://arxiv.org/pdf/1803.01454.pdf)
  - R package: https://github.com/hazimehh/L0Learn

Ridge Regression (Hoerl & Kennard 1970, Technometrics)

- $\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} ||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||^2 + \lambda||\boldsymbol{\beta}||_2^2$
- $||\boldsymbol{\beta}||_2^2 = \sum_{j=1}^{p} \beta_j^2$
- $\widehat{\boldsymbol{\beta}}_{Ridge} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \rightarrow$ exact solution
- $\widehat{\boldsymbol{\beta}}_{LS} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$
- Let $\mathbf{X}^\top\mathbf{X} = \mathbf{I}_{p\times p}$

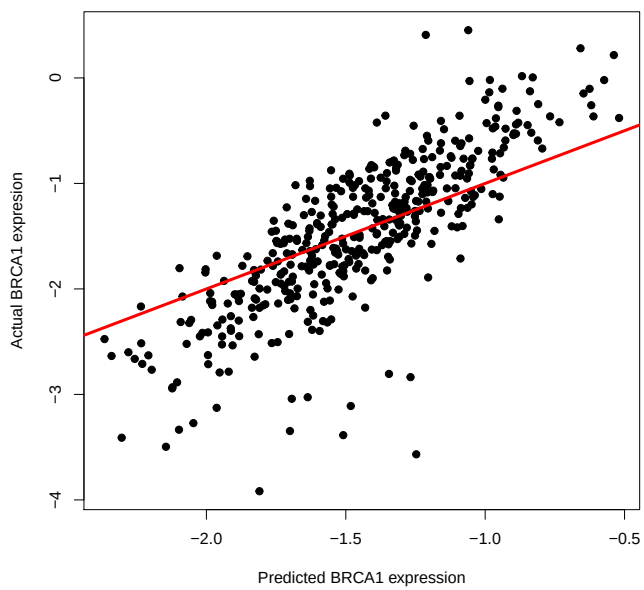
$$\hat{\beta}_{j(Ridge)} = \frac{\hat{\beta}_{j(LS)}}{1+\lambda}$$

Least squares vs. Ridge



Ridge Regression on TCGA Training Set

```r
set.seed(101) # for reproducibility
sample <- sample.int(n = nrow(TCGA$X), size = floor(.80*nrow(TCGA$X)), replace = F) # 80% training / 20% testing
X.train <- TCGA$X[sample, ]
X.test  <- TCGA$X[-sample, ]
y.train <- TCGA$y[sample]
y.test <- TCGA$y[-sample]

# fit ridge regression on training
library(glmnet)
fit.ridge <- cv.glmnet(x = X.train, y = y.train, alpha = 0, nfolds = 5, intercept = FALSE)
beta_hat_ridge <- coef(fit.ridge)
any(is.na(beta_hat_ridge))

## [1] FALSE

plot(predict(fit.ridge, newx = X.train), y.train, ylab = "Actual BRCA1 expresion",
           xlab = "Predicted BRCA1 expression", pch = 19)
abline(a=0,b=1, col = "red", lwd = 3)
```



Ridge Regression on TCGA Testing Set

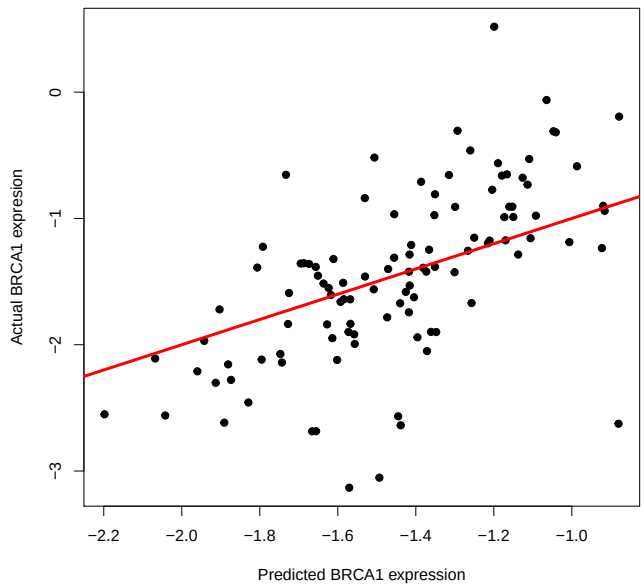
```
yhat.test <- predict(fit.ridge, newx = X.test)
(mse.ridge <- mean((yhat.test - y.test)^2)) # test set mean squared error
```

```
## [1] 0.302211
```

```
mse.lm
```

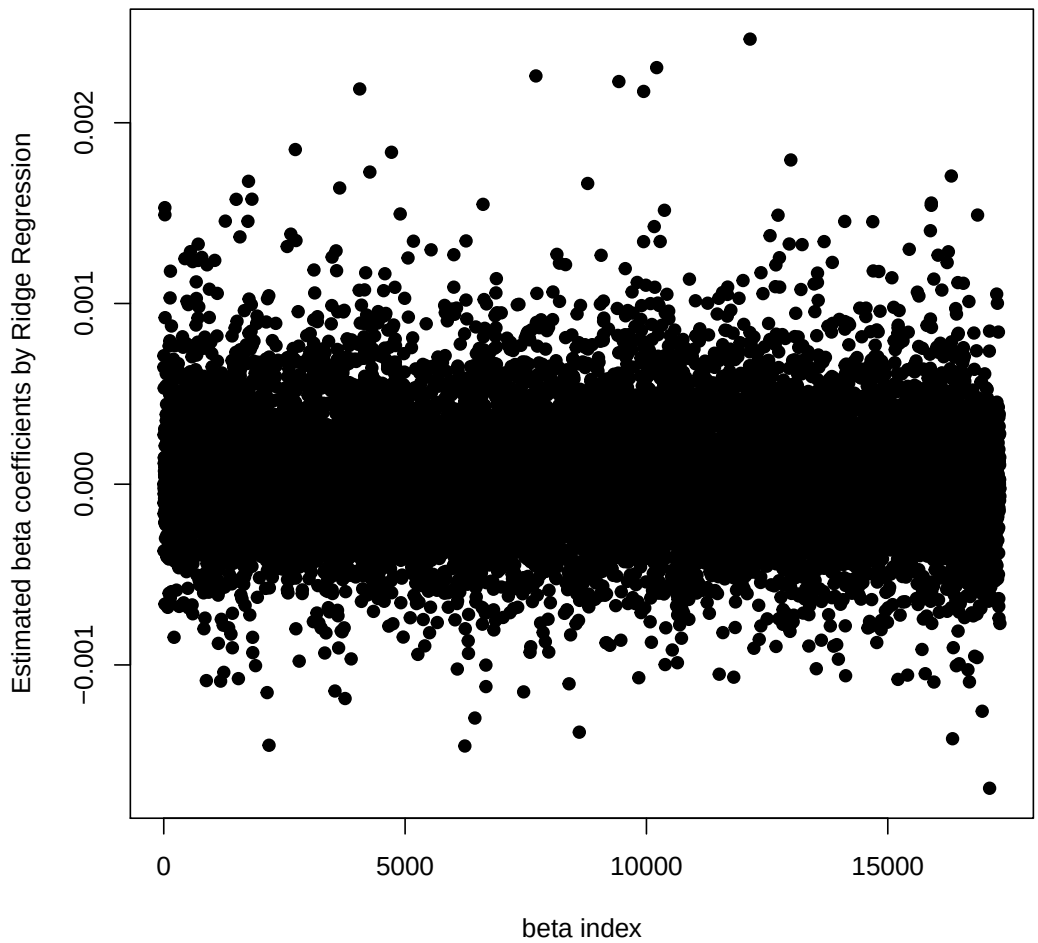
```
## [1] 123.895
```

```
plot(yhat.test, y.test, ylab = "Actual BRCA1 expresion", xlab = "Predicted BRCA1 expression", pch = 19)
abline(a=0,b=1, col = "red", lwd = 3)
```



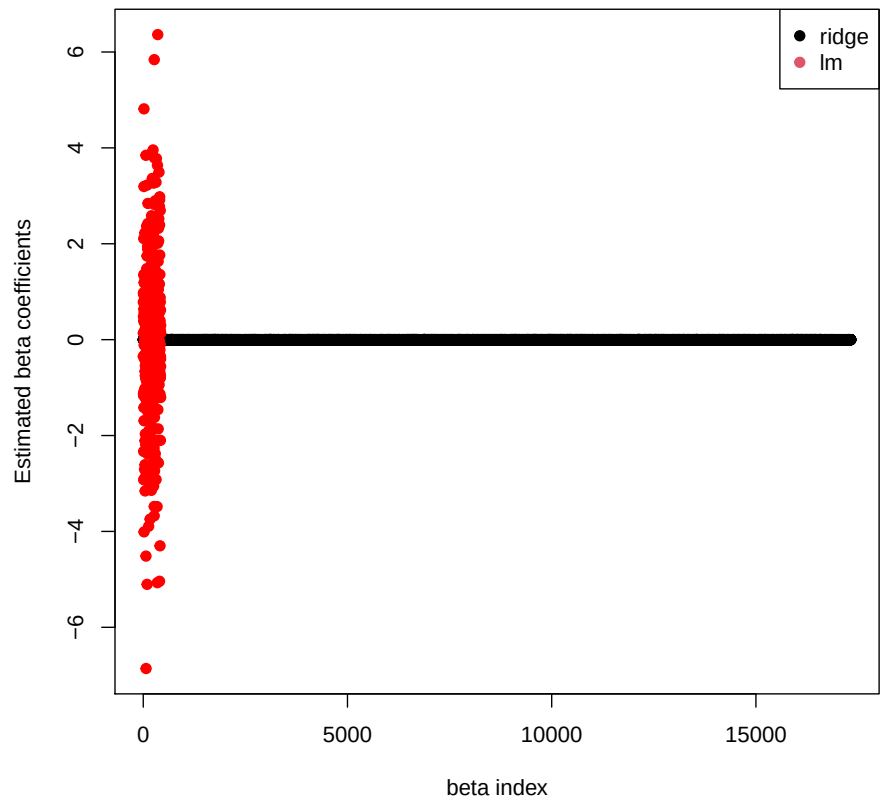
Estimated Regression Coefficients $\widehat{\beta}$

```r
plot(beta_hat_ridge, pch = 19, ylab = "Estimated beta coefficients by Ridge Regression", xlab = "beta index")
```



Estimated Regression Coefficients $\widehat{\beta}$

```
plot(beta_hat_ridge, pch = 19, ylab = "Estimated beta coefficients", xlab = "beta index",
        ylim = range(beta_hat_lm, na.rm = TRUE))
points(beta_hat_lm, pch = 19, col = "red")
legend("topright", legend =  c("ridge","lm"), col = 1:2, pch = 19)
```



Ridge Regression for Multi-Collinearity

```
set.seed(1234)
x1 <- rnorm(20)
x2 <- rnorm(20, mean=x1, sd=.01)
cor(x1,x2)

## [1] 0.9999717

y <- rnorm(20, mean=3+x1+x2) # true betas are 3, 1 and 1
(fit <- lm(y ~ x1 + x2))

##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Coefficients:
## (Intercept)           x1           x2
##       2.169       50.386      -48.784

sum(coef(fit)[-1])

## [1] 1.602687
```

- The strong correlation between results in extremely biased estimates of the regression parameters when using linear regression

Ridge Regression for Multi-Collinearity

```
set.seed(1234)
x1 <- rnorm(20)
x2 <- rnorm(20, mean=x1, sd=.01)
cor(x1,x2)

## [1] 0.9999717

y <- rnorm(20, mean=3+x1+x2) # true betas are 3, 1 and 1
ridge.fit <- cv.glmnet(x = cbind(x1,x2), y = y, alpha = 0)
coef(ridge.fit)

## 3 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 2.3719161
## x1          0.6390786
## x2          0.6362313
```

- When we introduce the added assumption that small coefficients are more likely than large ones by using a ridge penalty, however, this uncertainty is resolved

Correlations in High-Dimensional Data

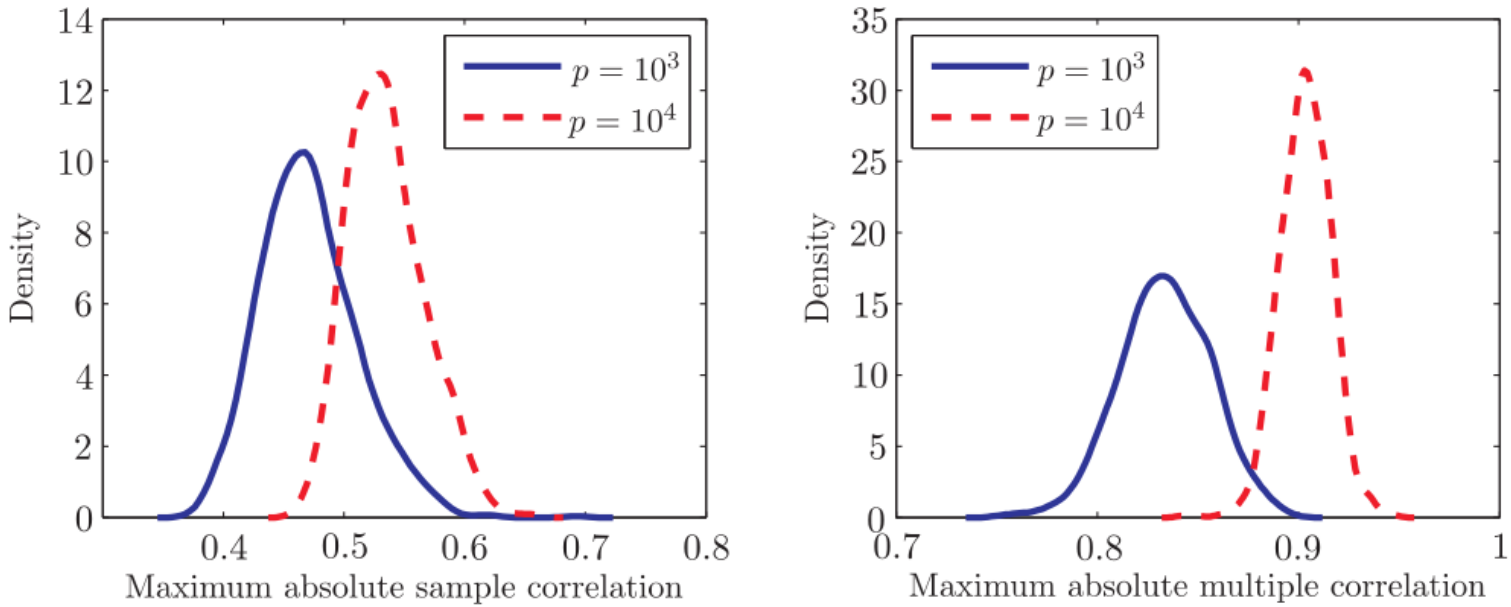


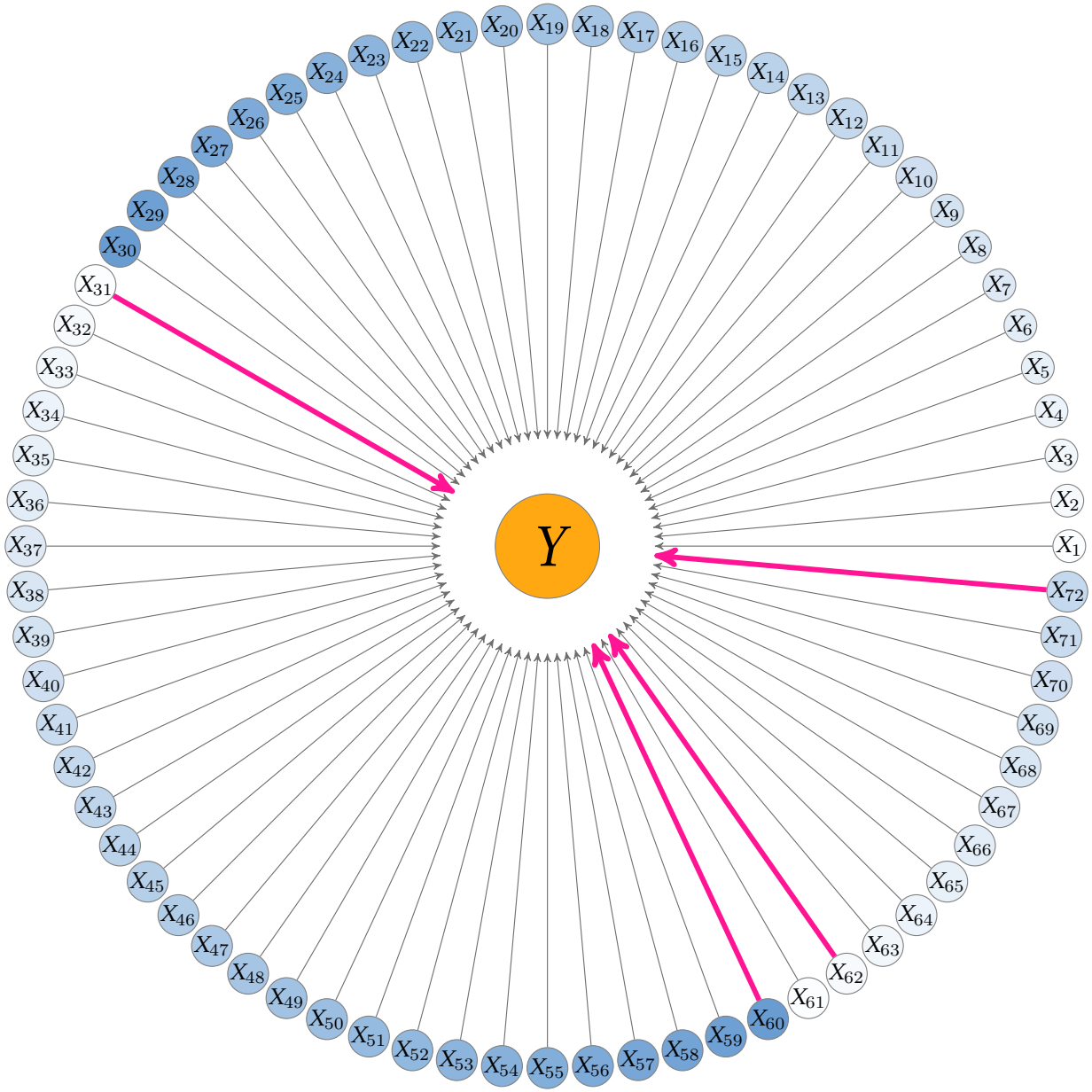
Figure 1. Distributions (left panel) of the maximum absolute sample correlation coefficient $\max_{2\le j\le p} |\mathrm{corr}(Z_1, Z_j)|$, and distributions (right panel) of the maximum absolute multiple correlation coefficient of Z_1 with 5 other variables ($\max_{|S|=5} |\mathrm{corr}(Z_1, \mathbf{Z}_S^T \hat{\boldsymbol{\beta}}_S)|$, where $\hat{\boldsymbol{\beta}}_S$ is the regression coefficient of Z_1 regressed on $\mathbf{Z}_S$, a subset of variables indexed by S and excluding Z_1), computed by the stepwise addition algorithm (the actual values are larger than what are presented here), when $n = 50$, $p = 1,000$ (solid curve) and $p = 10,000$ (dashed), based on 1,000 simulations.

https://arxiv.org/abs/0910.1122

Remarks about Ridge Regression

- The major limitation of ridge regression is the fact that all of its coefficients are nonzero
- This poses two considerable problems for high-dimensional regression:
  - Solutions become very difficult to interpret
  - The computational burden becomes large
- It is desirable, then, to have models which allow for both shrinkage and selection; in other words, to retain the benefits of ridge regression while at the same time selecting a subset of important variables

Bet on Sparsity Principle



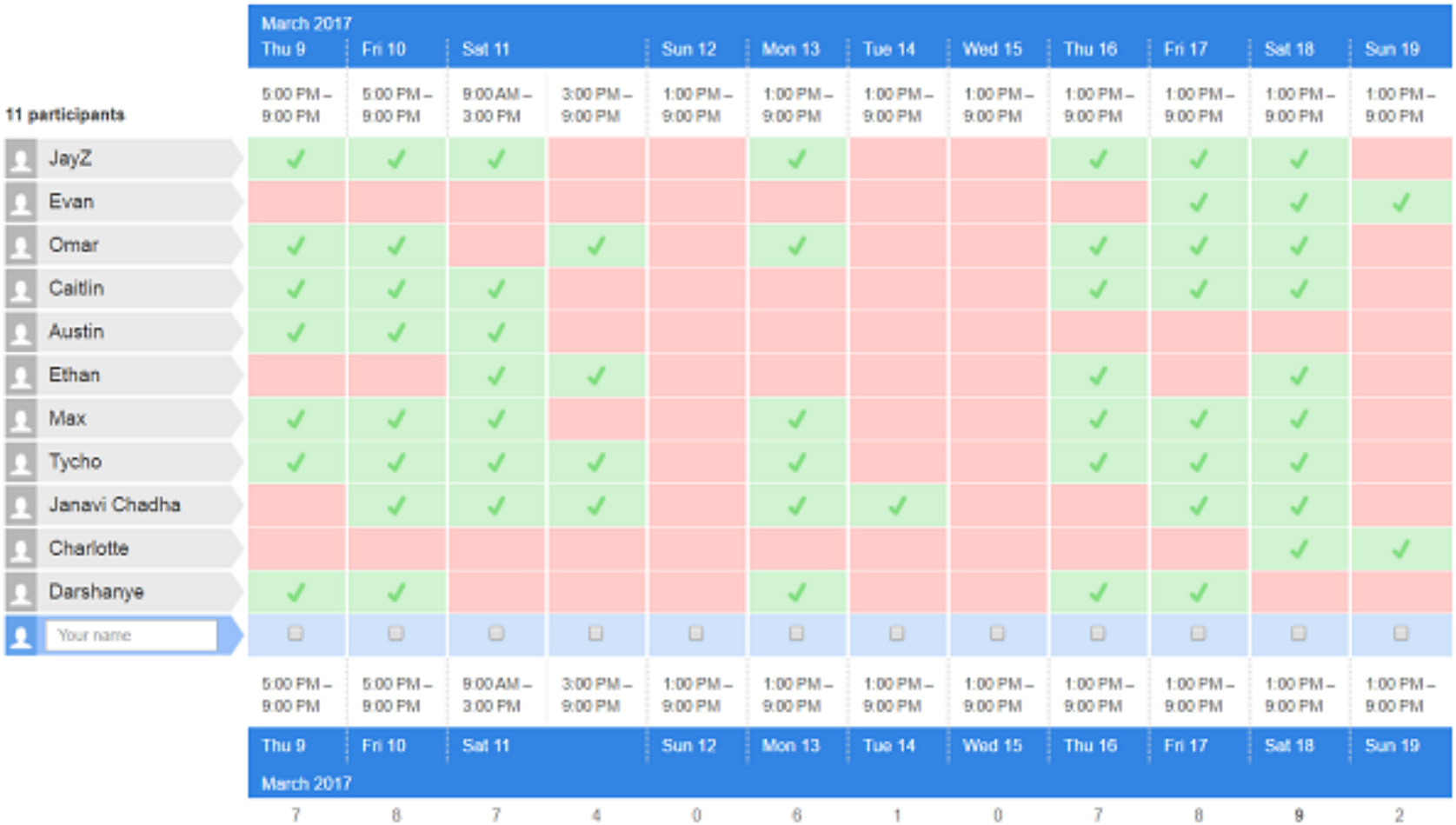
Bet on Sparsity Principle

Use a procedure that does well in sparse problems, since no procedure does well in dense problems.[1]

- We often don't have enough data to estimate so many parameters
- Even when we do, we might want to identify a **relatively small number of predictors** ($k < N$) that play an important role
- Faster computation, easier to understand, and stable predictions on new datasets.

[1] The elements of statistical learning. Springer series in statistics, 2001.

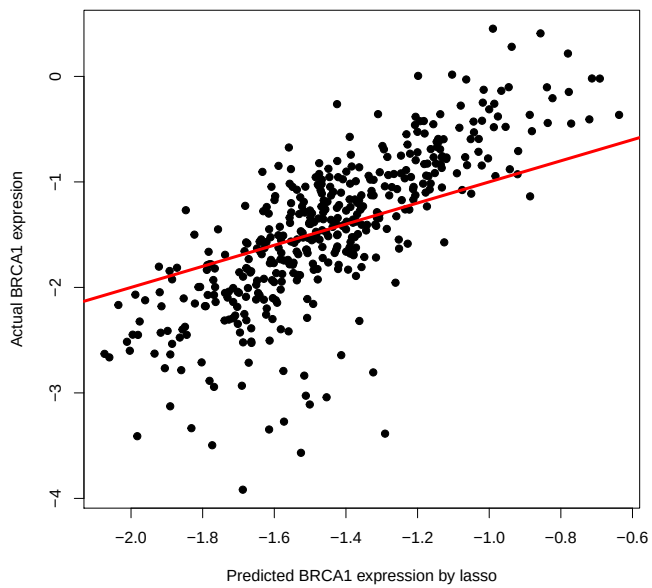
How would you schedule a meeting of 20 people?



Doctors Bet on Sparsity Also

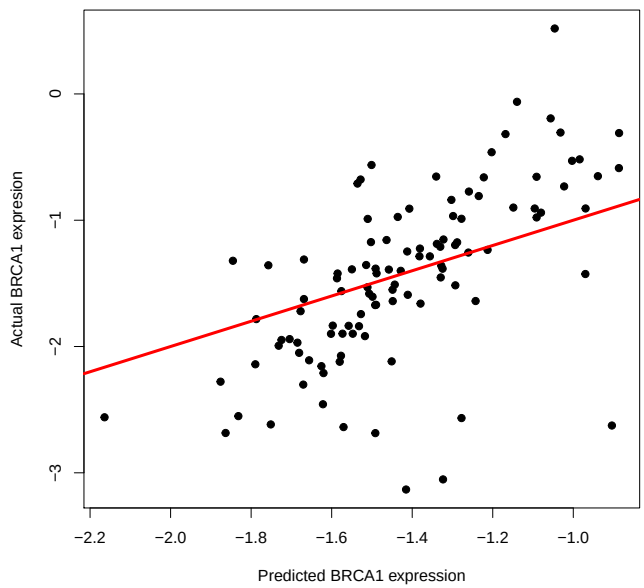
Lasso Regression on TCGA Training Set

```r
set.seed(101) # for reproducibility
sample <- sample.int(n = nrow(TCGA$X), size = floor(.80*nrow(TCGA$X)), replace = F) # 80% training / 20% testing
X.train <- TCGA$X[sample, ]
X.test  <- TCGA$X[-sample, ]
y.train <- TCGA$y[sample]
y.test <- TCGA$y[-sample]

# fit ridge regression on training
library(glmnet)
fit.lasso <- cv.glmnet(x = X.train, y = y.train, alpha = 1, nfolds = 5, intercept = FALSE)
beta_hat_lasso <- coef(fit.lasso)
plot(predict(fit.lasso, newx = X.train), y.train, ylab = "Actual BRCA1 expresion",
xlab = "Predicted BRCA1 expression by lasso", pch = 19)
abline(a=0,b=1, col = "red", lwd = 3)
```



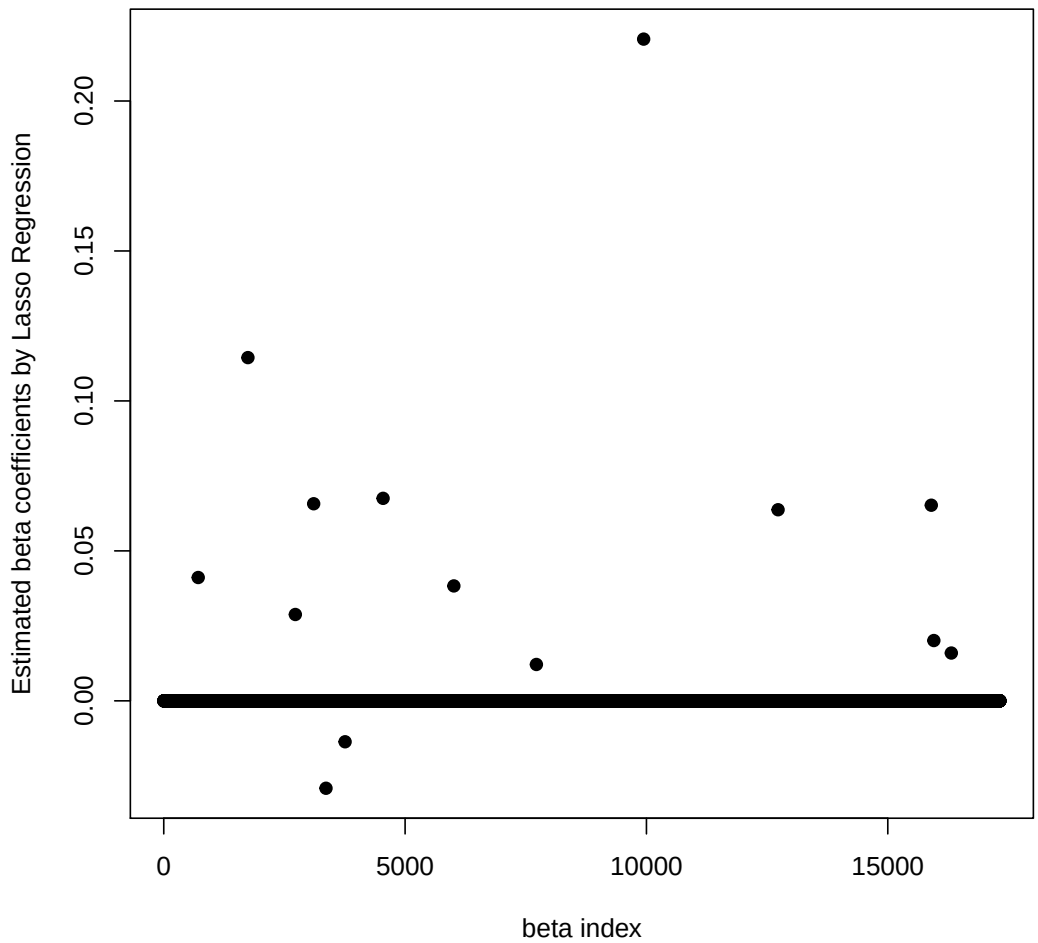
Lasso Regression on TCGA Testing Set

```
yhat.test <- predict(fit.lasso, newx = X.test)
(mse.lasso <- mean((yhat.test - y.test)^2)) # test set mean squared error

## [1] 0.3095205

mse.ridge ; mse.lm

## [1] 0.302211
## [1] 123.895

plot(yhat.test, y.test, ylab = "Actual BRCA1 expresion", xlab = "Predicted BRCA1 expression", pch = 19)
abline(a=0,b=1, col = "red", lwd = 3)
```



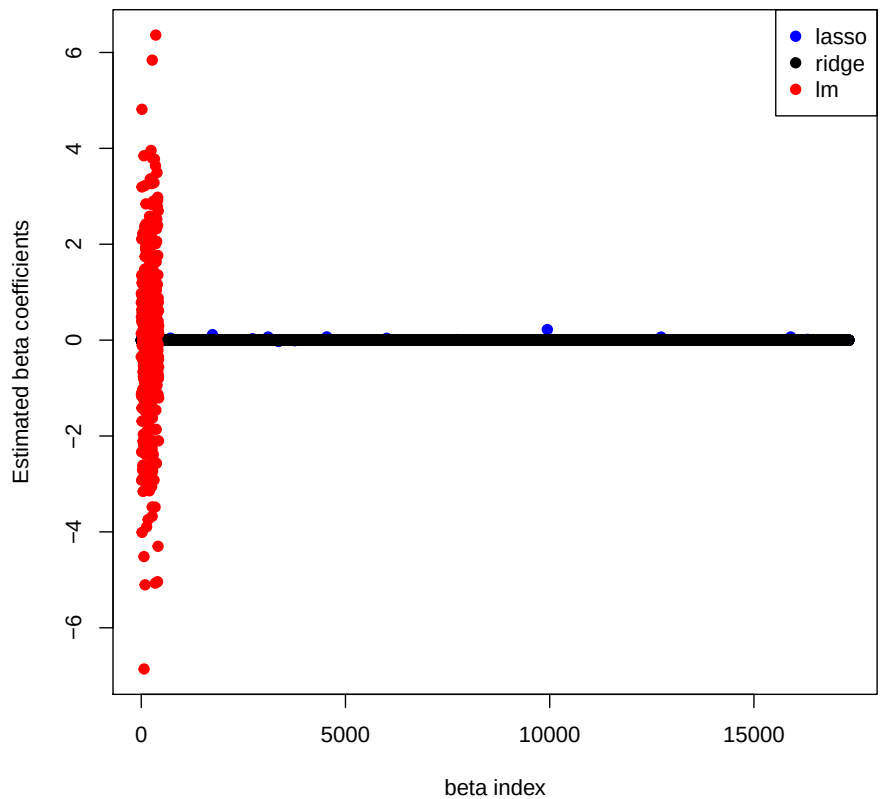
Estimated Regression Coefficients $\widehat{\beta}$

```
plot(beta_hat_lasso, pch = 19, ylab = "Estimated beta coefficients by Lasso Regression", xlab = "beta index")
```



Estimated Regression Coefficients $\widehat{\beta}$

```
plot(beta_hat_lasso, pch = 19, ylab = "Estimated beta coefficients", xlab = "beta index",
ylim = range(beta_hat_lm, na.rm = TRUE), col = "blue")
points(beta_hat_ridge, pch = 19, col = "black")
points(beta_hat_lm, pch = 19, col = "red")
legend("topright", legend =  c("lasso","ridge","lm"),
         col = c("blue","black","red"), pch = 19)
```



la**SS**o: Shrinkage and Selection

- Its name captures the essence of what the lasso penalty accomplishes
  - **Shrinkage**: Like ridge regression, the lasso penalizes large regression coefficients and shrinks estimates towards zero
  - **Selection**: Unlike ridge regression, the lasso produces sparse solutions: some coefficient estimates are exactly zero, effectively removing those predictors from the model
- Sparsity has two very attractive properties
  - Speed: Algorithms which take advantage of sparsity can scale up very efficiently, offering considerable computational advantages
  - Interpretability: In models with hundreds or thousands of predictors, sparsity offers a helpful simplification of the model by allowing us to focus only on the predictors with nonzero coefficient estimates

Bridge regression (Frank and Friedman, 1993)

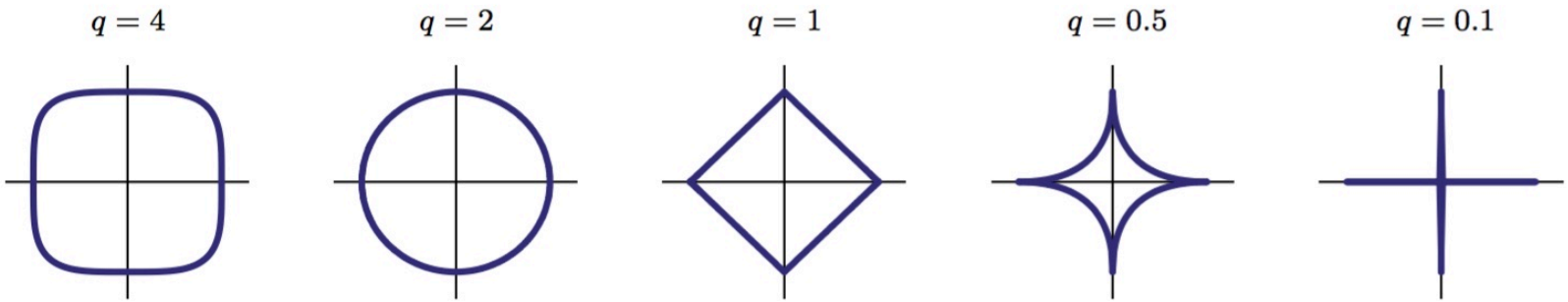
$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda\|\boldsymbol{\beta}\|_q \qquad 0 \leq q \leq 2.$$

Its constrained formulation

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2$$

$$\text{subject to } \|\boldsymbol{\beta}\|_q = \sum_{j=1}^{p} |\beta_j|^q \leq s$$

Bridge regression (Frank and Friedman, 1993)



Contours of equal value for the L_q penalty for difference values of q. For $q < 1$, the constraint region is **nonconvex**.

- $q = 0$, $\|\boldsymbol{\beta}\|_0 = \sum_{j=1}^{p} |\beta_j|^0 = \sum_{j=1}^{p} I(\beta_j \neq 0)$
- $q = 1$, $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$ convex

Background on the Lasso

- Predictors x_{ij}, $j = 1, \ldots, p$ and outcome values y_i for the ith observation, $i = 1, \ldots, n$
- Assume x_{ij} are standardized so that $\sum_i x_{ij}/n = 0$ and $\sum_i x_{ij}^2 = 1$. The lasso[1] solves

$$\widehat{\boldsymbol{\beta}}^{lasso} = \arg\min_{\beta} \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

$$\text{subject to } \sum_{j=1}^{p} |\beta_j| \leq s, \qquad s > 0$$

- Equivalently, the Lagrange version of the problem, for $\lambda > 0$

$$\widehat{\boldsymbol{\beta}}^{lasso} = \arg\min_{\beta} \frac{1}{2} \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

[1] Tibshirani. JRSSB (1996)

Inspection of the Lasso Solution

- Consider a single predictor setting based on the observed data $\{(x_i, y_i)\}_{i=1}^n$. The problem then is to solve

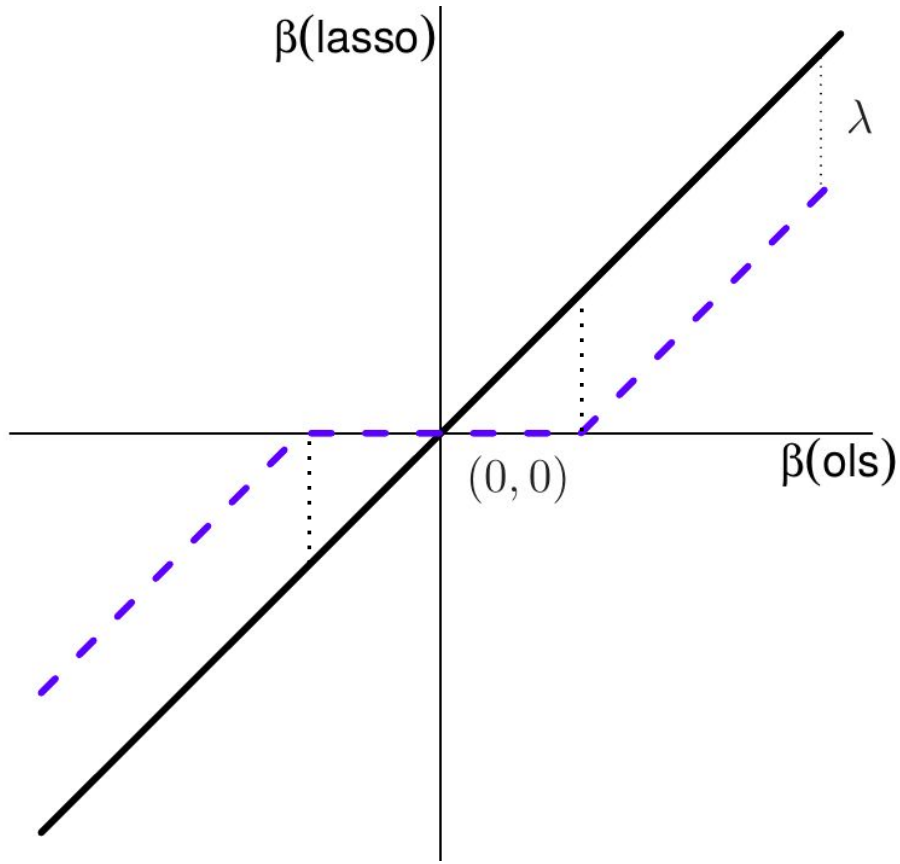
$$\widehat{\beta}^{lasso} = \arg\min_{\beta} \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i\beta)^2 + \lambda|\beta| \tag{1}$$

- With a **standardized** predictor, the lasso solution (1) is a **soft-thresholded** version of the least-squares (LS) estimate $\widehat{\beta}^{LS}$

$$\begin{aligned}
\widehat{\beta}^{lasso} &= S_\lambda\left(\widehat{\beta}^{LS}\right) = \text{sign}\left(\widehat{\beta}^{LS}\right)\left(|\widehat{\beta}^{LS}| - \lambda\right)_+ \\
&= \begin{cases} \widehat{\beta}^{LS} - \lambda, & \widehat{\beta}^{LS} > \lambda \\ 0 & |\widehat{\beta}^{LS}| \leq \lambda \\ \widehat{\beta}^{LS} + \lambda & \widehat{\beta}^{LS} \leq -\lambda \end{cases}
\end{aligned}$$

Inspection of the Lasso Solution

- When the data are standardized, the lasso solution **shrinks the LS estimate toward zero** by the amount λ



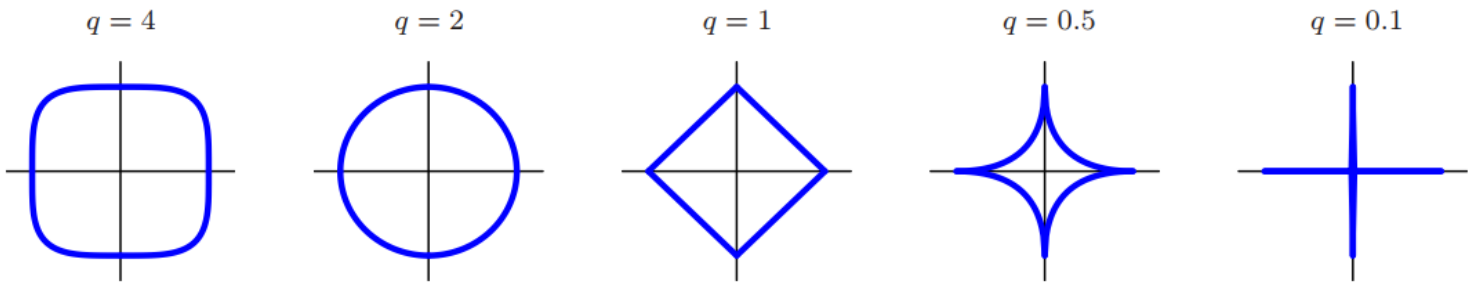
[1] Hastie et al. Statistical learning with sparsity: the lasso and generalizations

Why the ℓ_1 norm?

- For $q \geq 0$, evaluate the criteria

$$\widetilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|^q \right\}$$

- Why do we use the ℓ_1 and not $q = 2$ (Ridge) or any other norm ℓ_q?

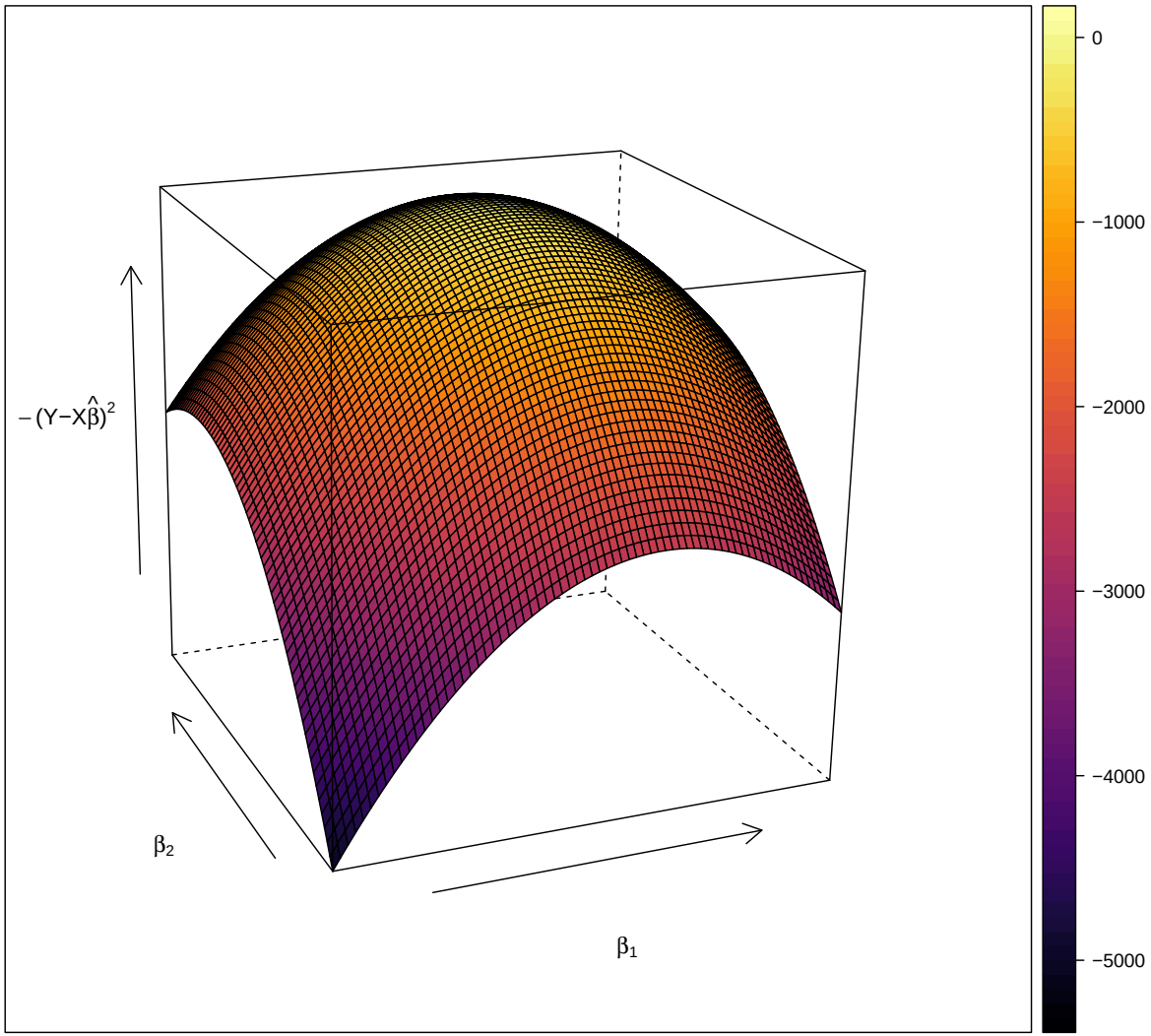


- $q = 1$ is the smallest value which gives sparse solutions AND is **convex** $\rightarrow$ scales well to high-dimensions
- For $q < 1$ the constrained region is **not-convex**

Least-squares regression surface

- Consider the following model with two predictors ($\mathbf{y}$ is centered)

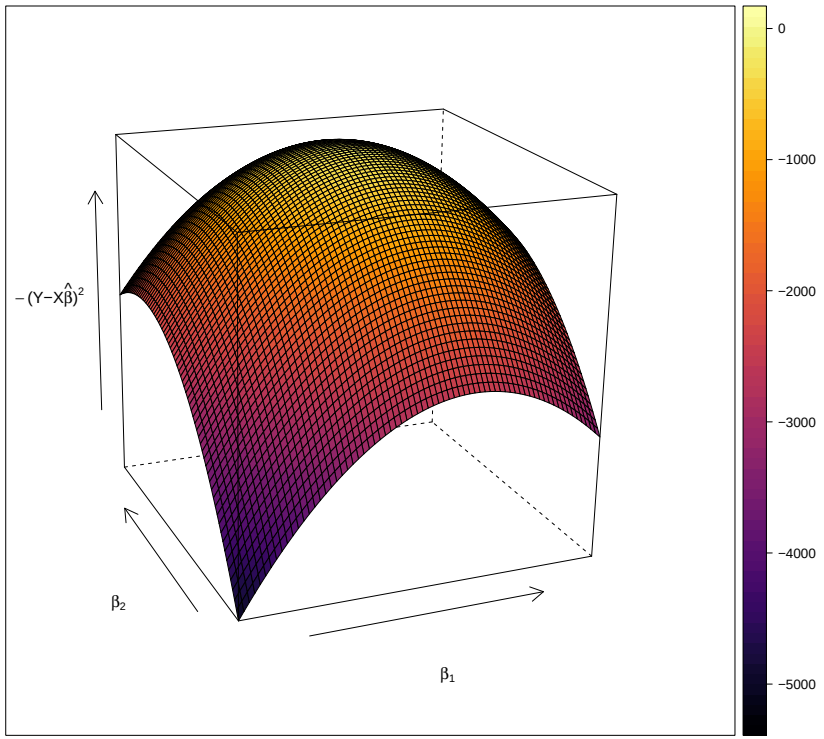
$$\mathbf{y} = \beta_1 \mathbf{x}_1 + \beta_2 \mathbf{x}_2 + \boldsymbol{\varepsilon}$$

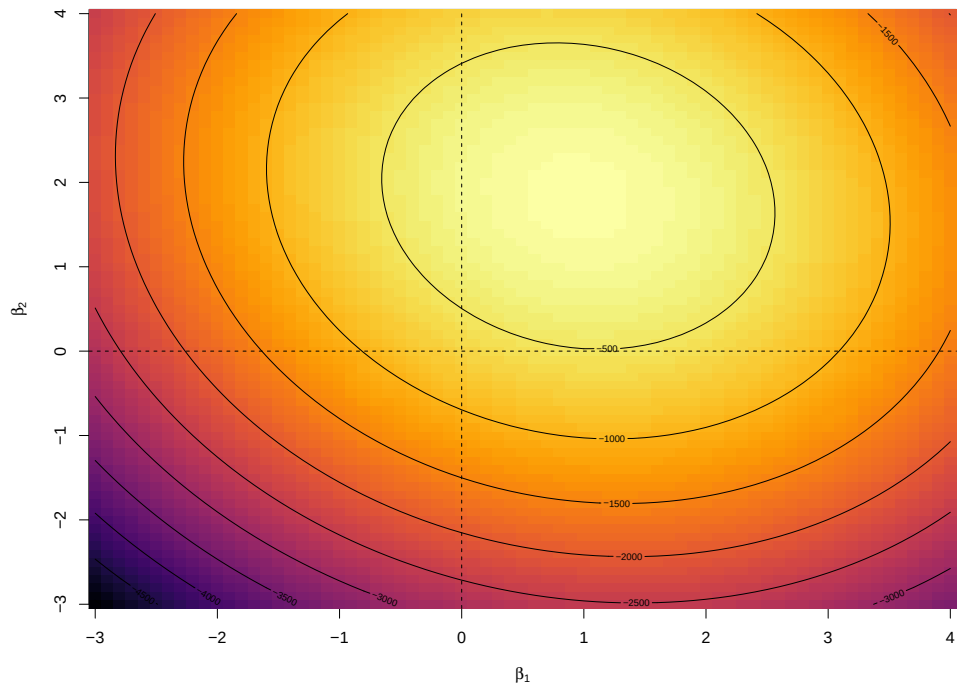


code to generate previous plot

```
pacman::p_load(viridis,fields,lattice,latex2exp,plotrix)

set.seed(12345)
b0 <- 0
b1 <- 1
b2 <- 2
X <- cbind(1,replicate(2, rnorm(100)))
y <- X %*% matrix(c(b0,b1,b2)) + sqrt(2)*rnorm(100)

# Define function for RSS
MyRss <- function(beta0, beta1) {
b <- c(0, beta0, beta1)
rss <- crossprod(y - X %*% b)
return(rss)
}

b0 <- seq(-3, 4, by=0.1)
b1 <- seq(-3, 4, by = 0.1)
z <- outer(b0, b1, function(x,y) mapply(MyRss, x, y))

wireframe(-z,drape = TRUE, colorkey = TRUE, screen = list(z = 20, x = -70, y = 3),
xlab = TeX("$\\beta_1$"), ylab = TeX("$\\beta_2$"),
zlab = TeX("$-(Y-X\\hat{\\beta})^2$"), col.regions = viridis::inferno(100))
```

Contours of the least-squares regression surface

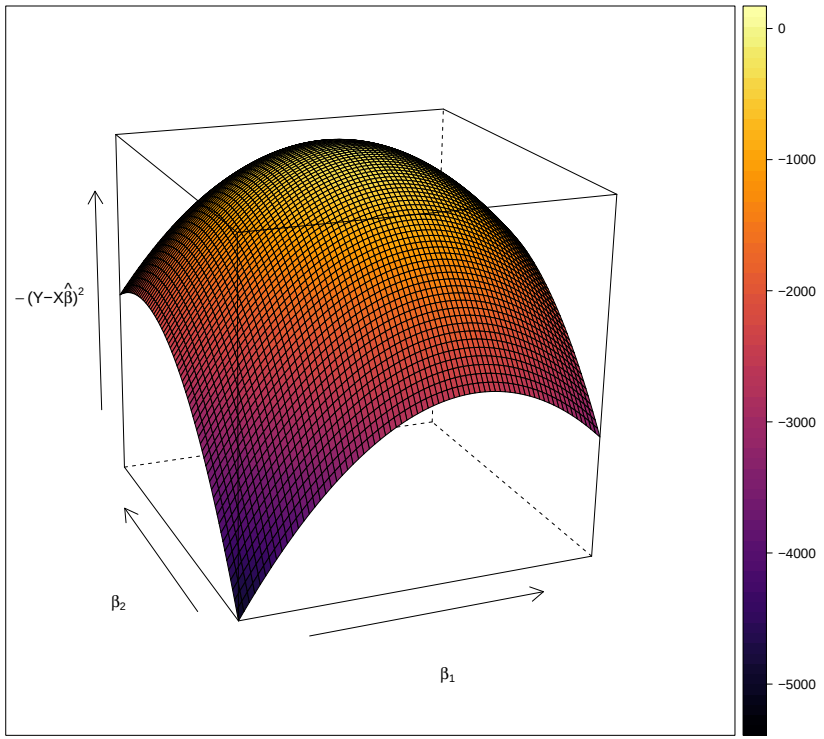


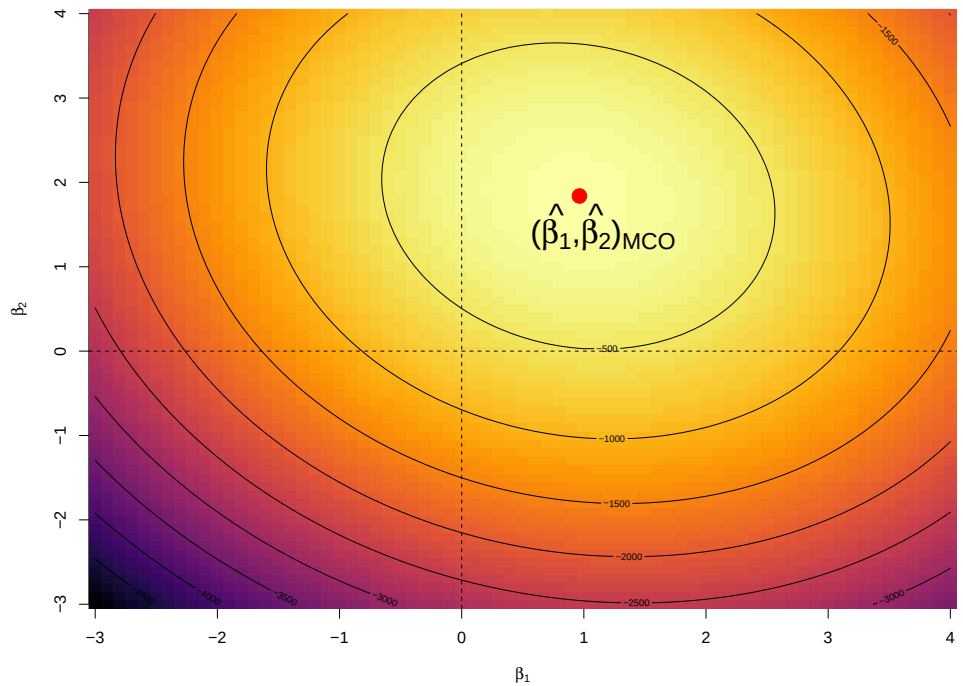


code to generate previous plot

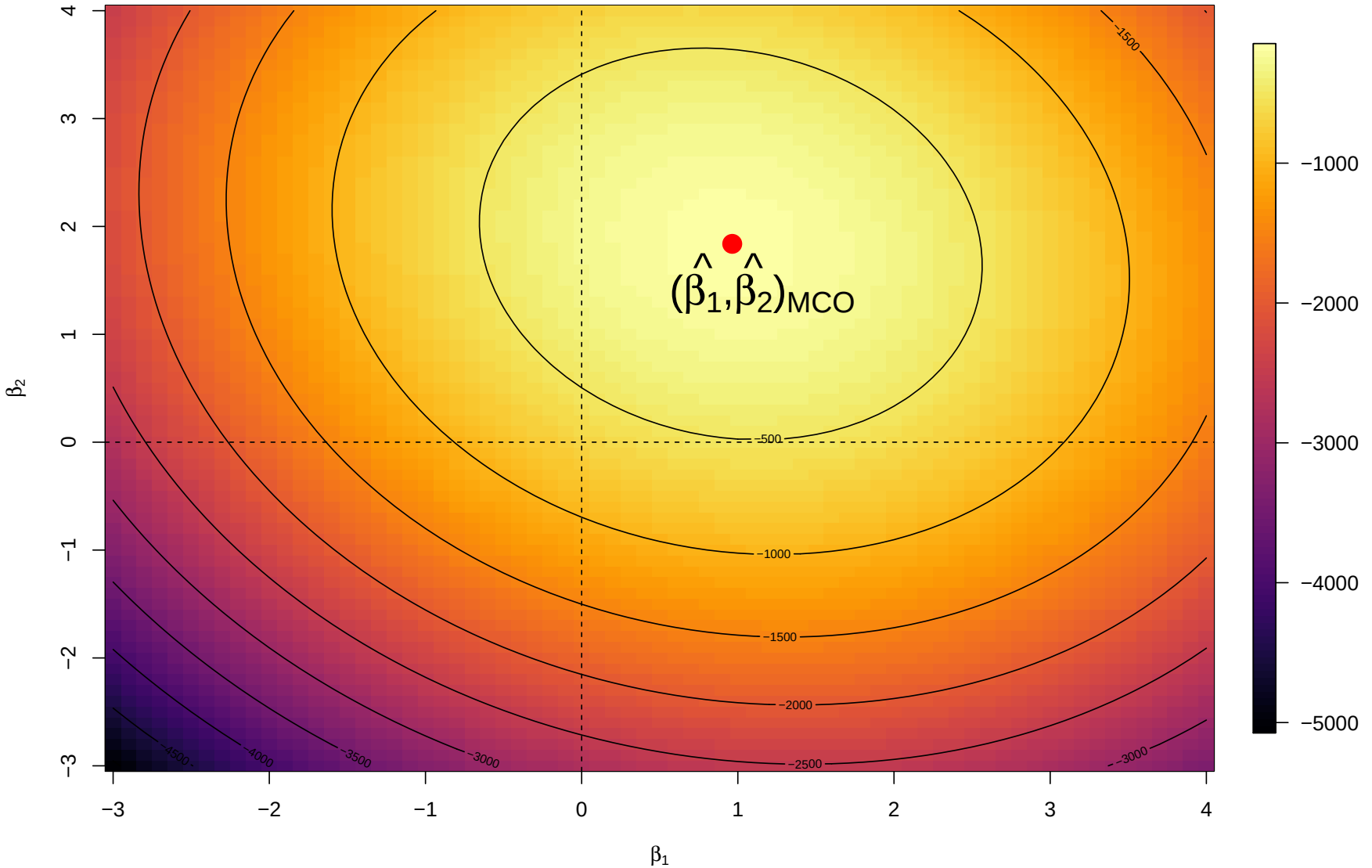
```
pacman::p_load(viridis,fields,lattice,latex2exp,plotrix)

set.seed(12345)
b0 <- 0
b1 <- 1
b2 <- 2
X <- cbind(1,replicate(2, rnorm(100)))
y <- X %*% matrix(c(b0,b1,b2)) + sqrt(2)*rnorm(100)

# Define function for RSS
MyRss <- function(beta0, beta1) {
b <- c(0, beta0, beta1)
rss <- crossprod(y - X %*% b)
return(rss)
}

b0 <- seq(-3, 4, by=0.1)
b1 <- seq(-3, 4, by = 0.1)
z <- outer(b0, b1, function(x,y) mapply(MyRss, x, y))

fields::image.plot(x = b0, y = b1, z = -z,xlab = TeX("$\\beta_1$"), ylab = TeX("$\\beta_2$"),
col = viridis::inferno(100))
contour(x = b0, y = b1, z = -z,xlab = TeX("$\\beta_1$"), ylab = TeX("$\\beta_2$"),
nlevels = 10, add=TRUE)
abline(v = 0, lty=2)
abline(h = 0, lty=2)
```

Contours of the least-squares regression surface

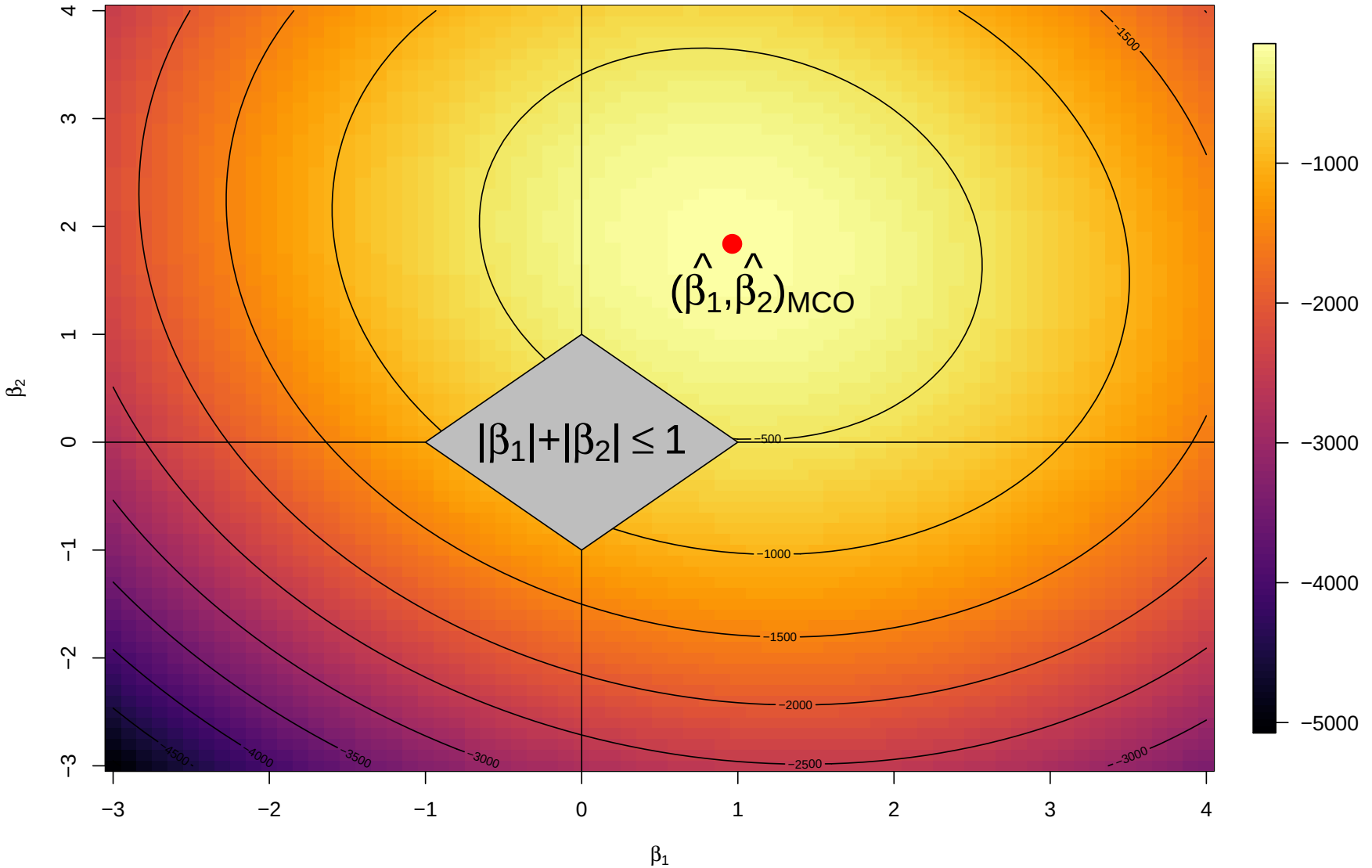




Contours of the least-squares regression surface



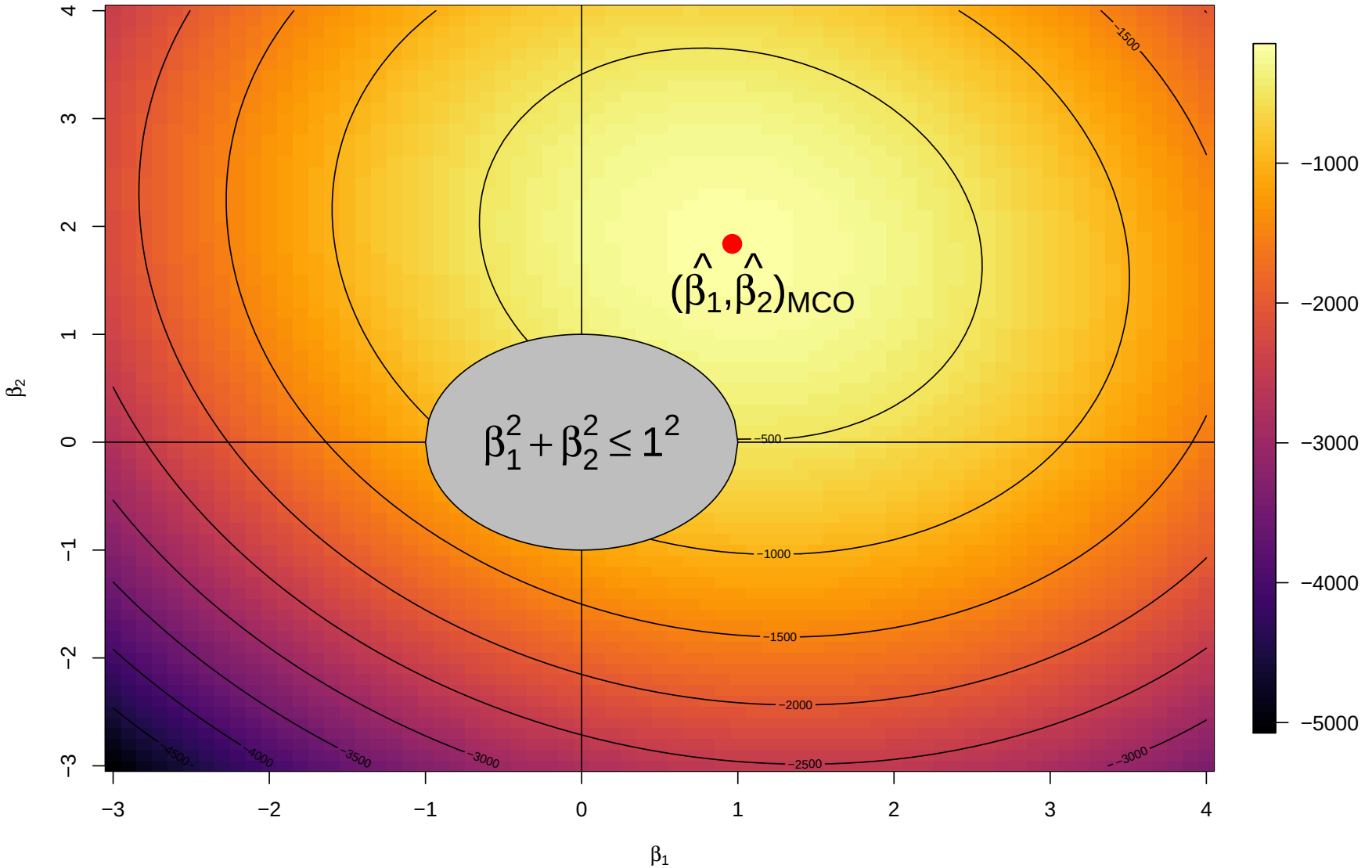
Constraint region of the lasso



code to generate previous plot

```
fields::image.plot(x = b0, y = b1, z = -z,xlab = TeX("$\\beta_1$"),
ylab = TeX("$\\beta_2$"),
col = viridis::inferno(100))
contour(x = b0, y = b1, z = -z,xlab = TeX("$\\beta_1$"), ylab = TeX("$\\beta_2$"),
nlevels = 10, add=TRUE)
points(x = lm.fit(x = X, y = y)$coef[2], y = lm.fit(x = X, y = y)$coef[3],
pch = 19, cex=2, col = "red")
text(x = lm.fit(x = X, y = y)$coef[2]*1.2,
y = lm.fit(x = X, y = y)$coef[3]*0.80,
labels = TeX("$(\\hat{\\beta_1},\\hat{\\beta_2})_{LS}$"),
cex = 2)
abline(v = 0)
abline(h = 0)

conditions <- function(x,y) {
c1 <- (abs(x) + abs(y)) <= 1
return(c1)}

zz <- expand.grid(x=b0,y=b1)
zz <- zz[conditions(zz$x,zz$y),]

polygon(c(zz$x[which.min(zz$x)],zz$x[which.max(zz$y)],
zz$x[which.max(zz$x)], zz$x[which.min(zz$y)]),
c(zz$y[which.min(zz$x)],zz$y[which.max(zz$y)],
zz$y[which.max(zz$x)], zz$y[which.min(zz$y)]),
col = "grey")
text(x = 0, y= 0,
labels = TeX("$|\\beta_1|+|\\beta_2| \\leq 1$"), cex = 2)
```

Constraint region of the ridge



code to generate previous plot

```
fields::image.plot(x = b0, y = b1, z = -z,xlab = TeX("$\\beta_1$"),
ylab = TeX("$\\beta_2$"),
col = viridis::inferno(100))
contour(x = b0, y = b1, z = -z,xlab = TeX("$\\beta_1$"), ylab = TeX("$\\beta_2$"),
nlevels = 10, add=TRUE)
points(x = lm.fit(x = X, y = y)$coef[2], y = lm.fit(x = X, y = y)$coef[3],
pch = 19, cex=2, col = "red")
text(x = lm.fit(x = X, y = y)$coef[2]*1.2,
y = lm.fit(x = X, y = y)$coef[3]*0.80,
labels = TeX("$(\\hat{\\beta_1},\\hat{\\beta_2})_{LS}$"), cex = 2)
abline(v = 0)
abline(h = 0)

beta2 <- function(x,r=1) {
y <- sqrt(r^2 - x^2)
return(y)}

xseq <- seq(-1,1, length.out = 100)
polygon(cbind(c(xseq, rev(xseq)),c(beta2(x=xseq), -beta2(x=xseq))), col = "grey")
text(x = 0, y= 0,
labels = TeX("$\\beta_1^2+\\beta_2^2 \\leq 1^2$"), cex = 2)
```

Lasso vs. ridge

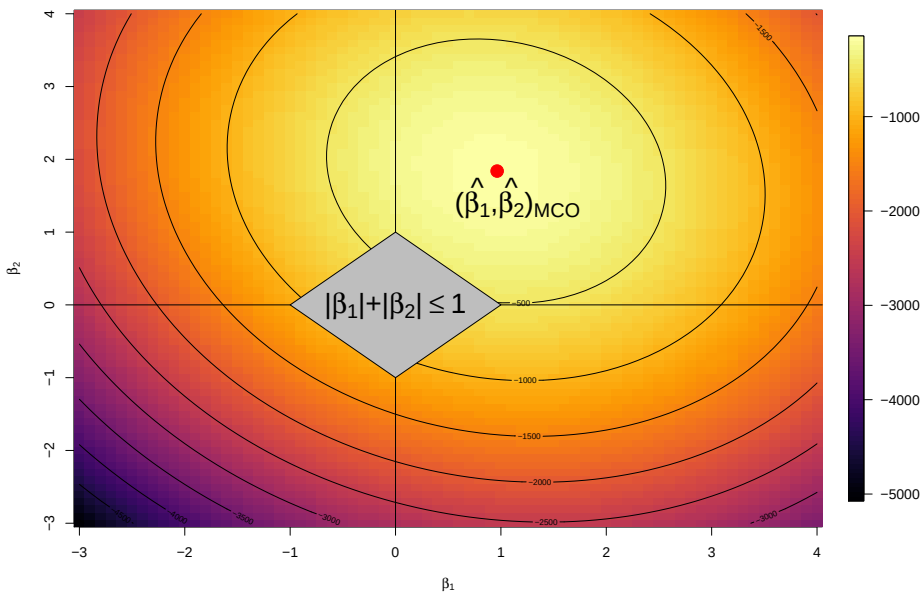


Figure: `lasso`

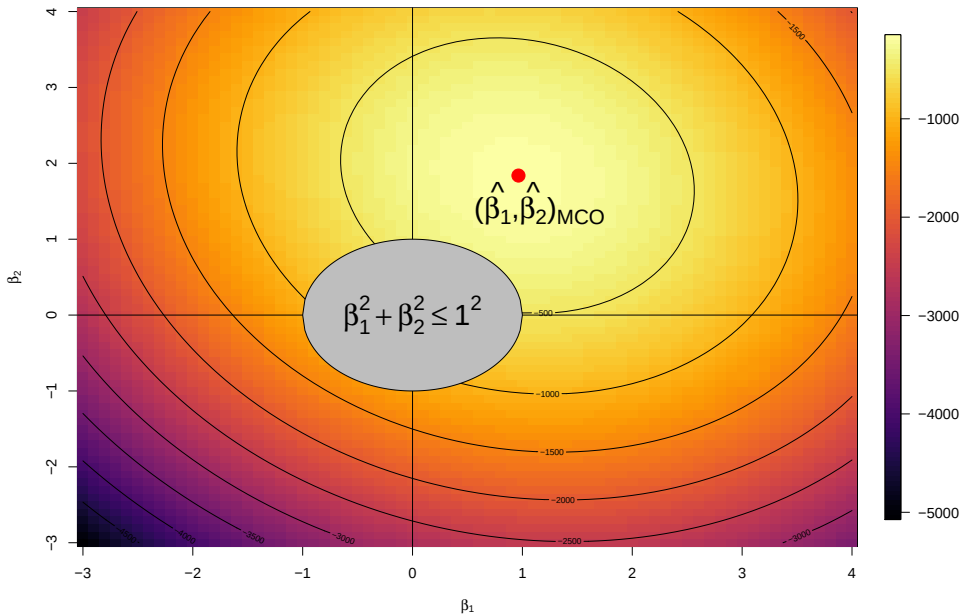
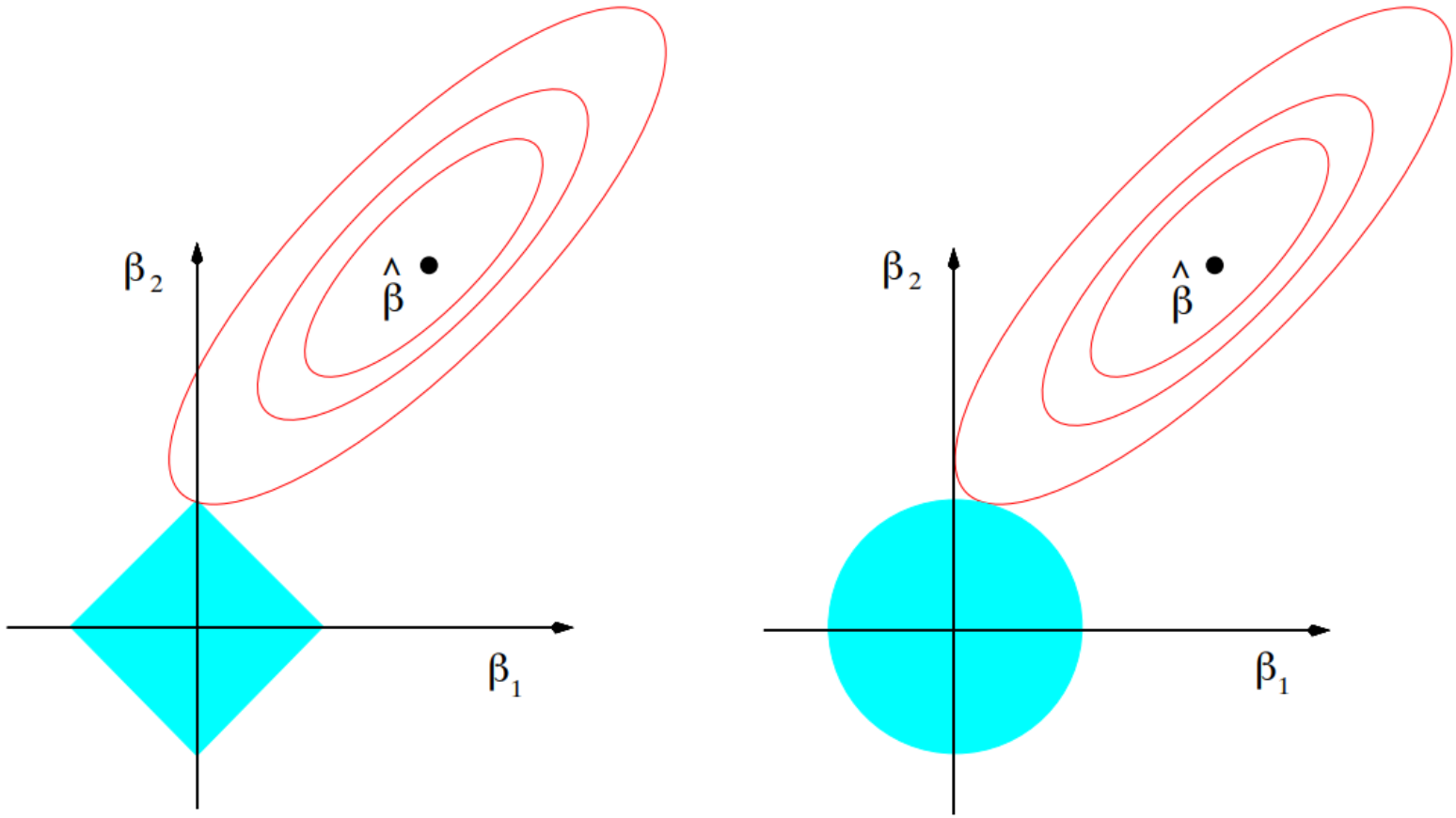


Figure: `ridge`

Classic version of the previous figure



Elements of Statistical Learning

Score functions and penalized score functions

- In classical statistical theory, the derivative of the log-likelihood function $\mathcal{L}(\theta)$ is called the score function, and maximum likelihood estimators are found by setting this derivative equal to zero, thus yielding the likelihood equations (or score equations):

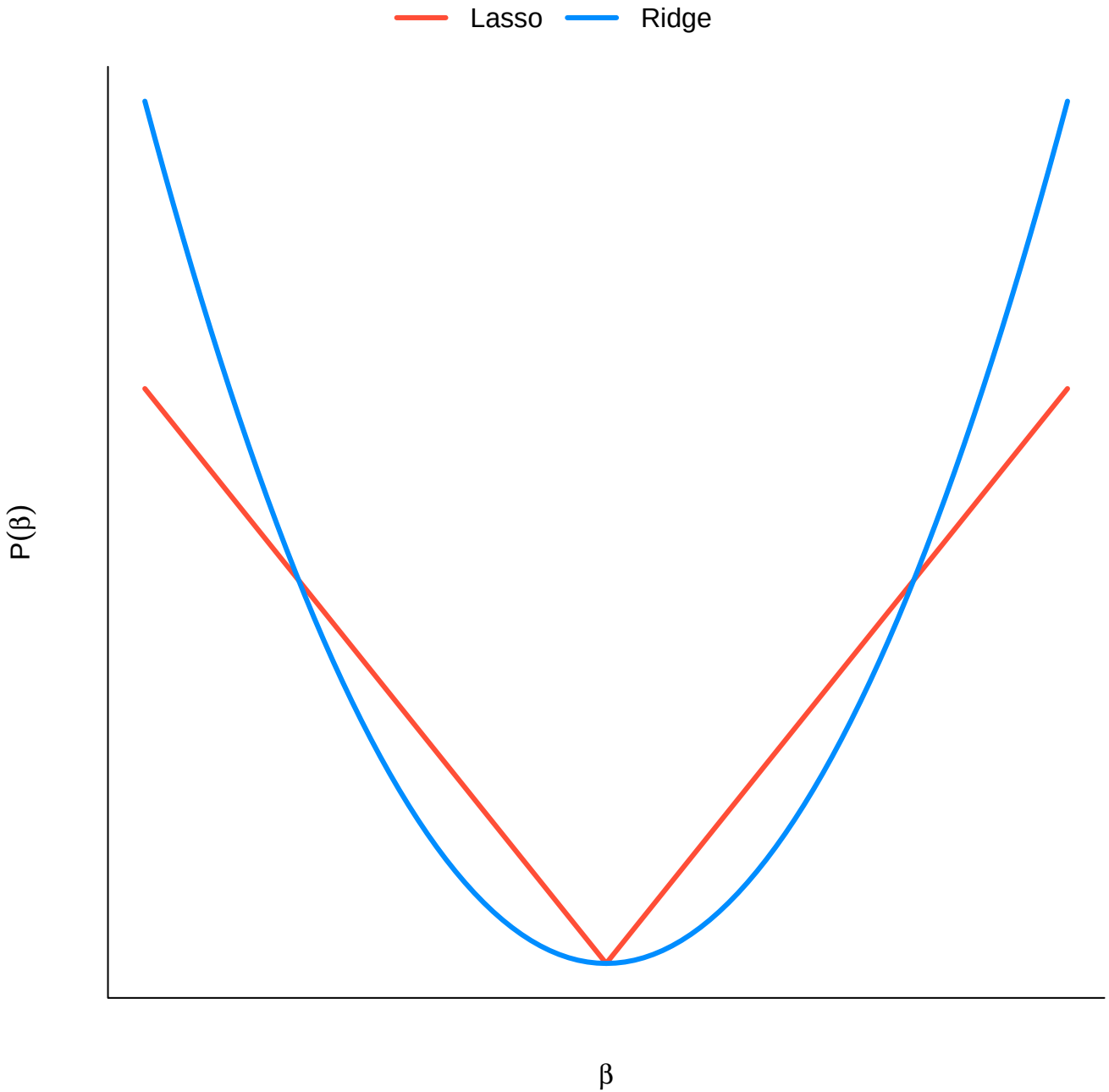
$$0 = \frac{\partial}{\partial \theta} \mathcal{L}(\theta)$$

- Extending this idea to penalized likelihoods involves taking the derivatives of objective functions of the form:

$$\mathbf{Q}(\theta) = \underbrace{\mathcal{L}(\theta)}_{\text{likelihood}} + \underbrace{P(\theta)}_{\text{penalty}}$$

yielding the penalized score function

Ridge vs. Lasso penalty



Penalized likelihood equations

- For ridge regression, the penalized likelihood is everywhere differentiable, and the extension to penalized score equations is straightforward

$$\widehat{\boldsymbol{\beta}}^{ridge} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\beta||_2^2$$

- For the lasso, the penalized likelihood is not differentiable - specifically, not differentiable at zero - and *subdifferentials* are needed to characterize them

$$\widehat{\boldsymbol{\beta}}^{lasso} = \arg\min_{\boldsymbol{\beta}} \mathbf{Q}(\theta) = \arg\min_{\beta} \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\beta||_1$$

- Letting $\partial\mathbf{Q}(\theta)$ denote the subdifferential of $\mathbf{Q}$, penalized likelihood equations are:

$$0 \in \partial\mathbf{Q}(\theta)$$

---

http://myweb.uiowa.edu/pbreheny/7240/s19/notes/2-13.pdf

Karush-Kuhn-Tucker (KKT) Conditions

- In the optimization literature, the resulting equations are known as the Karush-Kuhn-Tucker (KKT) conditions
- For convex optimization problems such as the lasso, the KKT conditions are both necessary and sufficient to characterize the solution
- The idea is simple: to solve for $\widehat{\boldsymbol{\beta}}^{lasso}$, we simply replace the derivative with the subderivative and the likelihood with the penalized likelihood

Subdifferential for $|x|$

The subdifferential for $f(x) = |x|$ is:

$$\partial|x| = \begin{cases} -1 & \text{if } x < 0 \\ [-1, 1] & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

KKT conditions for the lasso

- $$\widehat{\boldsymbol{\beta}}^{lasso} = \arg\min_{\boldsymbol{\beta}} \mathbf{Q}(\theta) = \arg\min_{\beta} \frac{1}{2}||\mathbf{y} - \mathbf{X}\boldsymbol{\beta}||_2^2 + \lambda||\beta||_1$$

- Result: $\widehat{\boldsymbol{\beta}}^{lasso}$ minimizes the lasso objective function if and only if it satisfies the KKT conditions:

$$\frac{1}{n}\mathbf{x}_j^\top(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}) = \lambda \text{sign}(\widehat{\beta}_j) \qquad \widehat{\beta}_j \neq 0$$

$$\frac{1}{n}|\mathbf{x}_j^\top(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})| \leq \lambda \qquad \widehat{\beta}_j = 0$$

- In other words, the correlation between a predictor and the residuals, $\mathbf{x}_j^\top(\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}})/n$, must exceed a certain minimum threshold λ before it is included in the model

- When this correlation is below λ, $\widehat{\beta}_j = 0$

Some remarks

- If we set

$$\lambda = \lambda_{\max} \equiv \max_{1 \le j \le p} \left| \mathbf{x}_j^T \mathbf{y} \right| / n$$

  then $\widehat{\boldsymbol{\beta}} = 0$ satisfies the KKT conditions

- That is, for any $\lambda \ge \lambda_{\max}$, we have $\widehat{\boldsymbol{\beta}}(\lambda) = 0$

- On the other hand, if we set $\lambda = 0$, the KKT conditions are simple the normal equations for OLS

$$\frac{1}{n} \mathbf{x}_j^\top (\mathbf{y} - \mathbf{X}\widehat{\boldsymbol{\beta}}) = 0 \cdot \text{sign}(\widehat{\beta}_j) \qquad \widehat{\beta}_j \neq 0$$

- Thus, the coefficient path for the lasso starts at $\lambda_{\max}$ and continues until $\lambda = 0$ if $\mathbf{X}$ is full rank; otherwise the solution will fail to be unique for λ values below some point $\lambda_{\min}$

Recall the Lasso Solution in the Orthonormal Design

- When the design matrix $\mathbf{X}$ is orthonormal, i.e., $n^{-1}\mathbf{X}^\top\mathbf{X} = \mathbf{I}$, the lasso estimate is a **soft-thresholded** version of the least-squares (LS) estimate $\widehat{\beta}^{LS}$

$$\begin{aligned}\widehat{\beta}^{lasso} &= S_\lambda\left(\widehat{\beta}^{LS}\right) = \text{sign}\left(\widehat{\beta}^{LS}\right)\left(|\widehat{\beta}^{LS}| - \lambda\right)_+ \\ &= \begin{cases}\widehat{\beta}^{LS} - \lambda, & \widehat{\beta}^{LS} > \lambda \\ 0 & |\widehat{\beta}^{LS}| \leq \lambda \\ \widehat{\beta}^{LS} + \lambda & \widehat{\beta}^{LS} \leq -\lambda\end{cases}\end{aligned}$$

- where $\widehat{\beta}^{LS} = \mathbf{x}_j^\top\mathbf{y}/n$

Probability that $\hat{\beta}_j = 0$

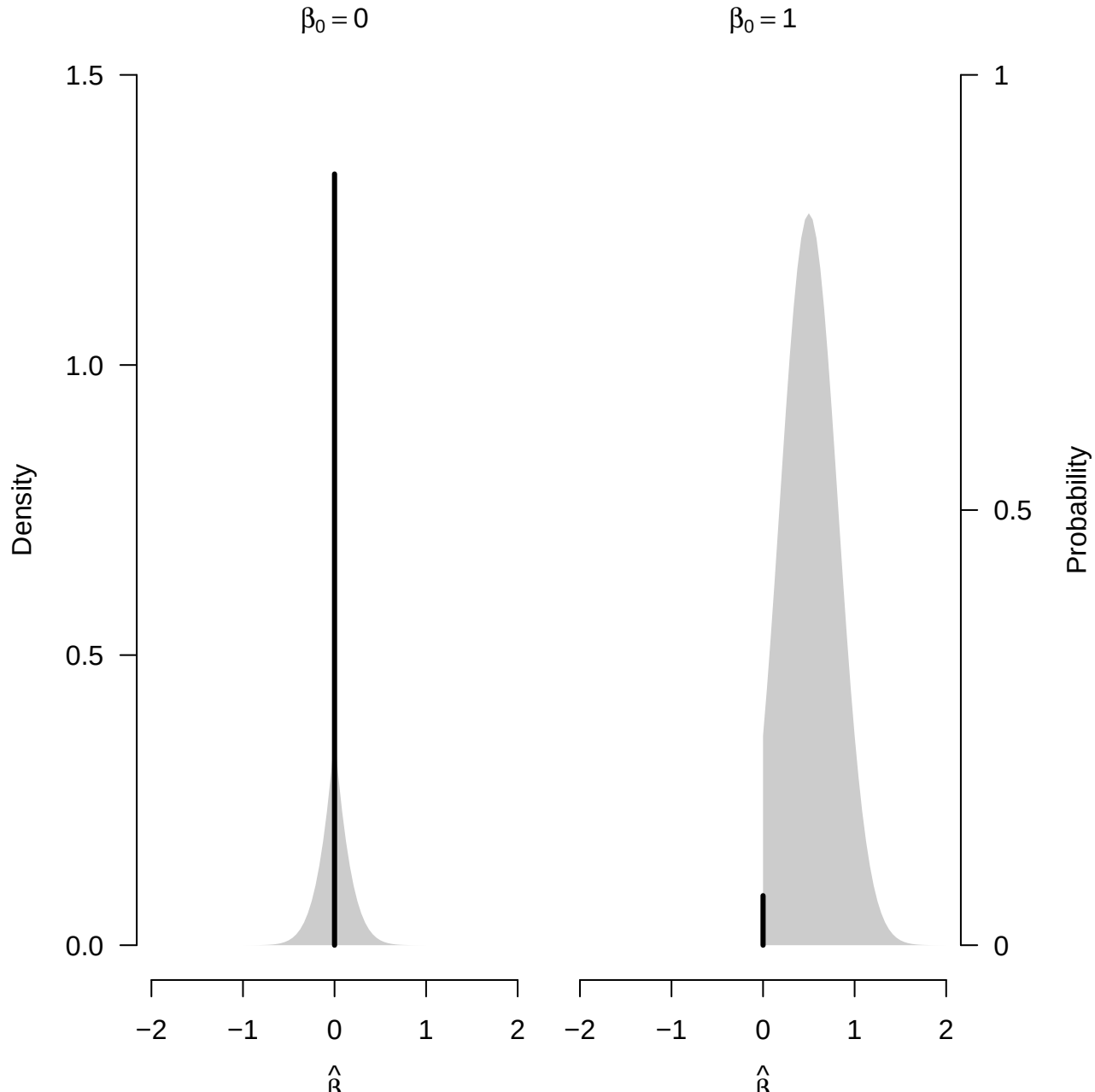
- With soft thresholding, it is clear that the lasso has a positive probability of yielding an estimate of exactly 0 - in other words, of producing a sparse solution
- Specifically, the probability of dropping $\mathbf{x}_j$ from the model is $\mathbb{P}\left(\left|\beta_j^{LS}\right| \leq \lambda\right)$
- Under the assumption that $\epsilon_i \overset{\perp\!\!\!\perp}{\sim} \mathrm{N}\left(0, \sigma^2\right)$, we have $\beta_j^{LS} \sim \mathcal{N}(\beta, \sigma^2/n)$ and

$$\mathbb{P}\left(\widehat{\beta}_j(\lambda) = 0\right) = \Phi\left(\frac{\lambda - \beta}{\sigma/\sqrt{n}}\right) - \Phi\left(\frac{-\lambda - \beta}{\sigma/\sqrt{n}}\right)$$

where Φ is the Gaussian CDF

Sampling Distribution

For $\sigma = 1$, $n = 10$, and $\lambda = 1/2$:



Why standard inference is invalid?

- This sampling distribution is very different from that of a classical MLE:
  - The distribution is mixed: a portion is continuously distributed, but there is also a point mass at zero
  - The continuous portion is not normally distributed
  - The distribution is asymmetric (unless $\beta = 0$)
  - The distribution is not centered at the true value of β

Algorithms for the lasso

- The KKT conditions only allow us to check a solution
- They do not necessarily help us to find the solution in the first place

Coordinate descent[1]

- The idea behind coordinate descent is, simply, to optimize a target function with respect to a single parameter at a time, iteratively cycling through all parameters until convergence is reached

- Coordinate descent is particularly suitable for problems, like the lasso, that have a simple closed form solution in a single dimension but lack one in higher dimensions

[1] Fu (1998), Friedman et al. (2007), Wu and Lange (2008)

Coordinate descent

- Let us consider minimizing $\mathbf{Q}$ with respect to β_j, while temporarily treating the other regression coefficients $\boldsymbol{\beta}_{-j}$ as fixed:

$$\mathbf{Q}(\beta_j|\boldsymbol{\beta}_{-j}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y_i - \sum_{k\neq j} x_{ij}\beta_k - x_{ij}\beta_j\right)^2 + \lambda|\beta_j| + \lambda\sum_{k\neq j}|\beta_k|$$

$$\widetilde{\beta}_j = \arg\min_{\beta_j} \mathbf{Q}(\beta_j|\boldsymbol{\beta}_{-j}) = S_\lambda(\tilde{z}_j) = \begin{cases} \tilde{z}_j - \lambda, & \tilde{z}_j > \lambda \\ 0 & |\tilde{z}_j| \leq \lambda \\ \tilde{z}_j + \lambda & \tilde{z}_j < -\lambda \end{cases}$$

- $\tilde{r}_{ij} = y_i - \sum_{k\neq j} x_{ik}\widetilde{\beta}_k \qquad \tilde{z}_j = n^{-1}\sum_{i=1}^{n} x_{ij}\tilde{r}_{ij}$

- $\{\tilde{r}_{ij}\}_{i=1}^{n}$ are the partial residuals with respect to the j^{th} predictor, and $\tilde{z}_j$ OLS estimator based on $\{\tilde{r}_{ij}, x_{ij}\}_{i=1}^{n}$

Convergence

- Numerical analysis of optimization problems of the form

$$\mathbf{Q}(\theta) = \mathcal{L}(\theta) + P(\theta)$$

  has shown that coordinate descent algorithms converge to a solution of the penalized likelihood equations provided that:
  - the function $\mathcal{L}(\boldsymbol{\beta})$ is differentiable and
  - the penalty function $P_\lambda(\boldsymbol{\beta})$ is separable $\rightarrow P_\lambda(\boldsymbol{\beta}) = \sum_j P_\lambda(\beta_j)$

- Lasso-penalized linear regression satisfies both of these criteria

Convergence

- Furthermore, because the lasso objective is a convex function, the sequence of the objective functions $\left\{ Q\left(\widetilde{\boldsymbol{\beta}}^{(s)}\right) \right\}$ converges to the global minimum
- However, because the lasso objective is not strictly convex, there may be multiple solutions
- In such situations, coordinate descent will converge to one of those solutions, but which solution it converges to is essentially arbitrary, as it depends on the order of the features

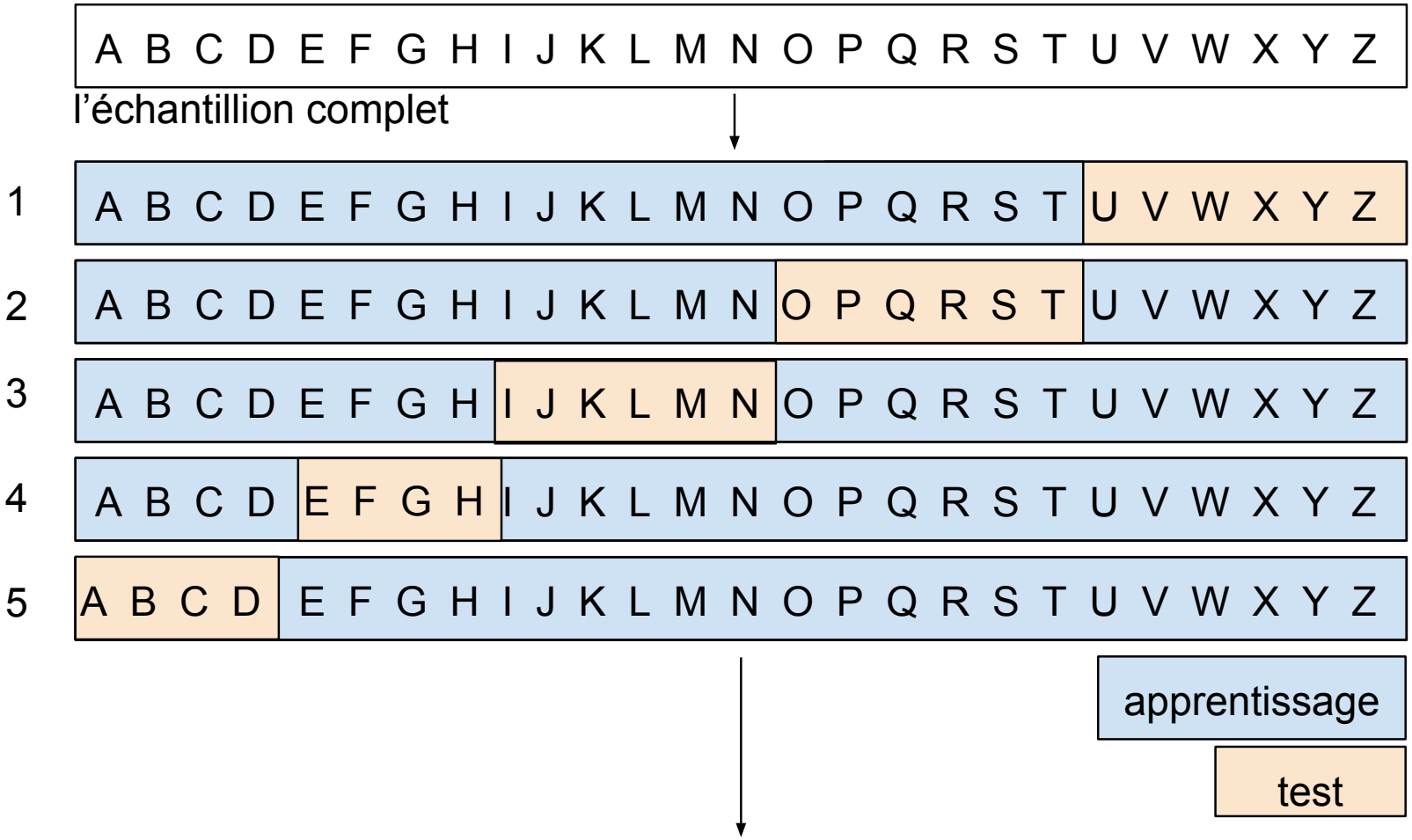
Coordinate descent, pathwise optimization, warm starts

- We are typically interested in determining $\widehat{\boldsymbol{\beta}}^{Lasso}$ for a range of values of λ, thereby obtaining the coefficient path
- In applying the coordinate descent algorithm to determine the lasso path, an efficient strategy is to compute solutions for decreasing values of λ, starting at $\lambda_{\max} = \max_{1 \leq j \leq p} \left| \mathbf{x}_j^T \mathbf{y} \right| / n$, the point at which all coefficients are 0
- Warm starts $\rightarrow$ By continuing along a decreasing grid of λ values, we can use the solutions $\widehat{\boldsymbol{\beta}}(\lambda_k)$ as initial values when solving for $\widehat{\boldsymbol{\beta}}(\lambda_{k+1})$

Sample Splitting

- As we have discussed, using the observed agreement between fitted values and the data is too optimistic; we require independent data to test predictive accuracy
- One solution we showed earlier, known as sample splitting, is to split the data set into two fractions, a training set and test set, using one portion to estimate $\widehat{\boldsymbol{\beta}}$ (i.e., *train* the model) and the other to evaluate how well $\mathbf{X}_{\mathbf{test}}\widehat{\boldsymbol{\beta}}$ predicts the observations in the second portion (i.e., *test* the model)
- The problem with this solution is that we rarely have so much data that we can freely part with half of it solely for the purpose of choosing λ

Cross-Validation



$$CV(\alpha) = \frac{1}{5}\sum_{v=1}^{5} MSE_v^{(test)}$$

Cross-validation: Details

1. Specify a grid of regularization parameter values $\Lambda = \{\lambda_1, \ldots, \lambda_K\}$
2. Divide the data into V roughly equal parts $D_1, \ldots, D_V$
3. For each $v = 1, \ldots, V$, compute the lasso solution path using the observations in $\{D_u, u \neq v\}$
4. For each $\lambda \in \Lambda$, compute the mean squared prediction error

$$\text{MSPE}_v(\lambda) = \frac{1}{n_v} \sum_{i \in D_v} \left\{ y_i - \mathbf{x}_i^T \widehat{\boldsymbol{\beta}}_{-v}(\lambda) \right\}^2$$

where n_v is the number of observations in D_v, $\widehat{\boldsymbol{\beta}}_{-v}$ are the estimated regression coefficients trained on the observations in $\{D_u, u \neq v\}$, as well as

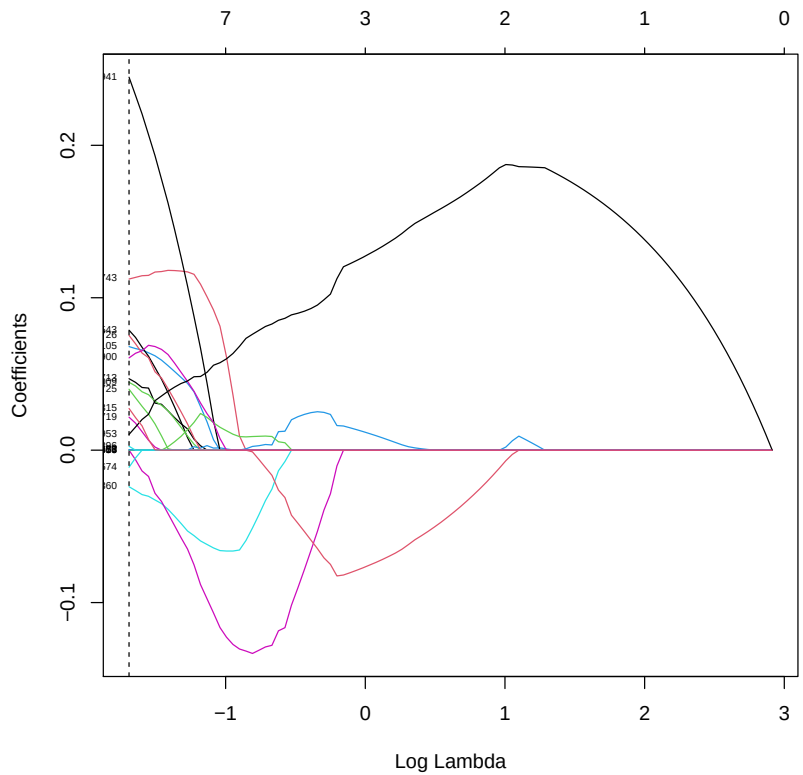
$$\text{CV}(\lambda) = \frac{1}{V} \sum_{v=1}^{V} \text{MSPE}_v(\lambda)$$

Cross-validation: Details

1. $\hat{\lambda}$ is taken to be the value that minimizes $\text{CV}(\lambda)$ and $\widehat{\boldsymbol{\beta}} \equiv \widehat{\boldsymbol{\beta}}(\hat{\lambda})$ the estimator of the regression coefficients
2. Note that
   - $\text{MSPE}_v(\lambda)$ is the mean squared prediction error for the model based on the training data $\{D_u, u \neq v\}$ in predicting the response variables in D_v
   - $\text{CV}(\lambda)$ is an estimate of the expected mean squared prediction error
3. Regardless of the number of cross-validation folds, each observation in the data appears exactly once in a test set

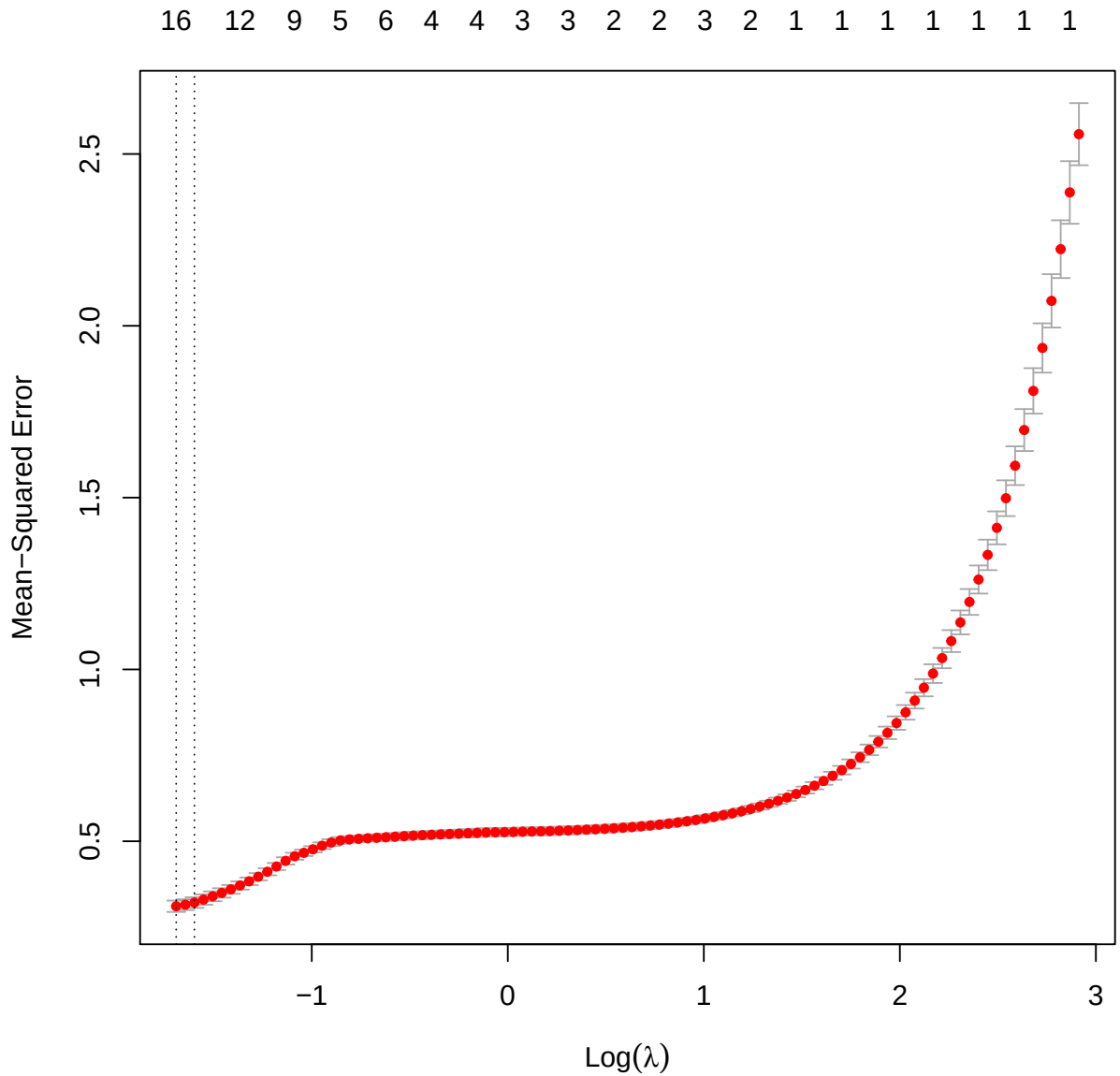
Lasso Solution Path on TCGA

```r
set.seed(101) # for reproducibility
sample <- sample.int(n = nrow(TCGA$X), size = floor(.80*nrow(TCGA$X)), replace = F) # 80% training / 20% testing
X.train <- TCGA$X[sample, ]
X.test  <- TCGA$X[-sample, ]
y.train <- TCGA$y[sample]
y.test <- TCGA$y[-sample]

# fit ridge regression on training
library(glmnet)
cvfit <- cv.glmnet(x = X.train, y = y.train, alpha = 1, nfolds = 5, intercept = FALSE)
fit <- cvfit$glmnet.fit
plot(fit, xvar = "lambda", label = TRUE)
abline(v = log(cvfit$lambda.min), lty = 2)
```



Lasso Cross-Validation on TCGA

```
plot(cvfit)
```



Motivating Dataset

	ID	Response	Gene1	Gene2	Gene3	Gene4	Gene5	Gene6
1	2610781	−1.255	1	2	0	0	0	1
2	4114347	−0.339	1	2	0	2	0	1
3	4399930	−0.6	1	2	1	1	0	1
4	2081319	0.809	1	2	0	1	0	2
5	1347380	0.279	2	2	0	0	0	0
6	3262449	−0.421	2	2	0	1	0	1
7	4870063	−0.454	2	2	0	0	0	2
8	1141212	1.383	2	2	1	1	1	0
9	2997954	−2.29	1	2	0	0	0	1
10	5805218	2.289	1	2	0	1	1	1

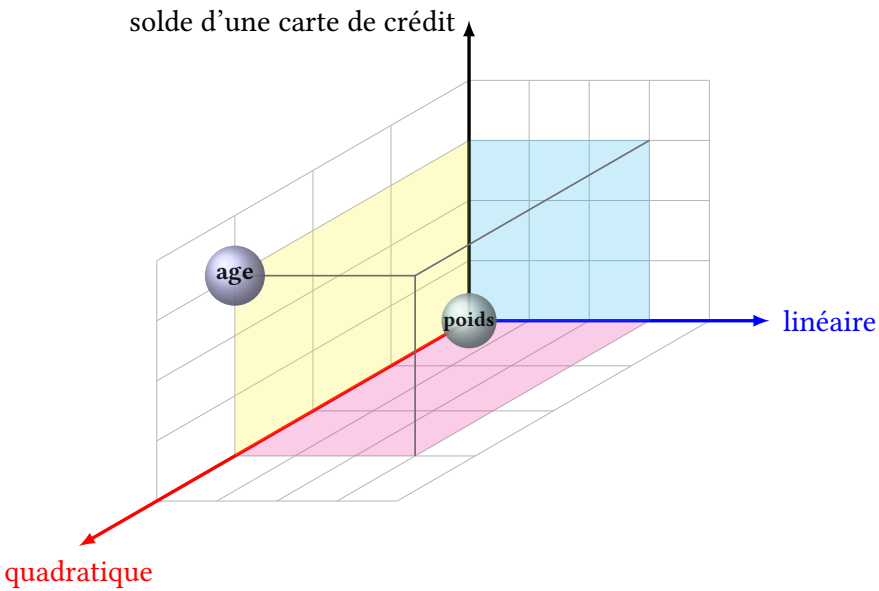
Groups of Predictors Affect the Response

	ID	Response	Gene1	Gene2	Gene3	Gene4	Gene5	Gene6
1	2610781	−1.255	1	2	0	0	0	1
2	4114347	−0.339	1	2	0	2	0	1
3	4399930	−0.6	1	2	1	1	0	1
4	2081319	0.809	1	2	0	1	0	2
5	1347380	0.279	2	2	0	0	0	0
6	3262449	−0.421	2	2	0	1	0	1
7	4870063	−0.454	2	2	0	0	0	2
8	1141212	1.383	2	2	1	1	1	0
9	2997954	−2.29	1	2	0	0	0	1
10	5805218	2.289	1	2	0	1	1	1

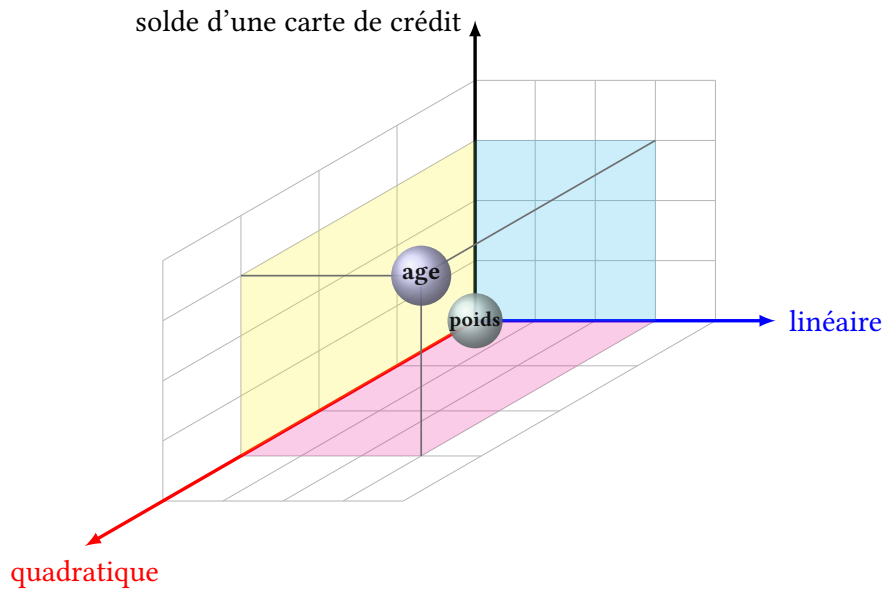
Group lasso for Categorical variables and Basis expansions

Useful for groups of variables (factor with > 2 categories, Age, Age^2).
Group lasso estimator is:

$$\min_{(\beta_0,\boldsymbol{\beta})} \frac{1}{2} \|\mathbf{y} - \beta_0 - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda \sum_{k=1}^{K} \sqrt{p_k} \|\boldsymbol{\beta}^{(k)}\|_2 \qquad p_k - \text{taille de group}$$



(a) Lasso



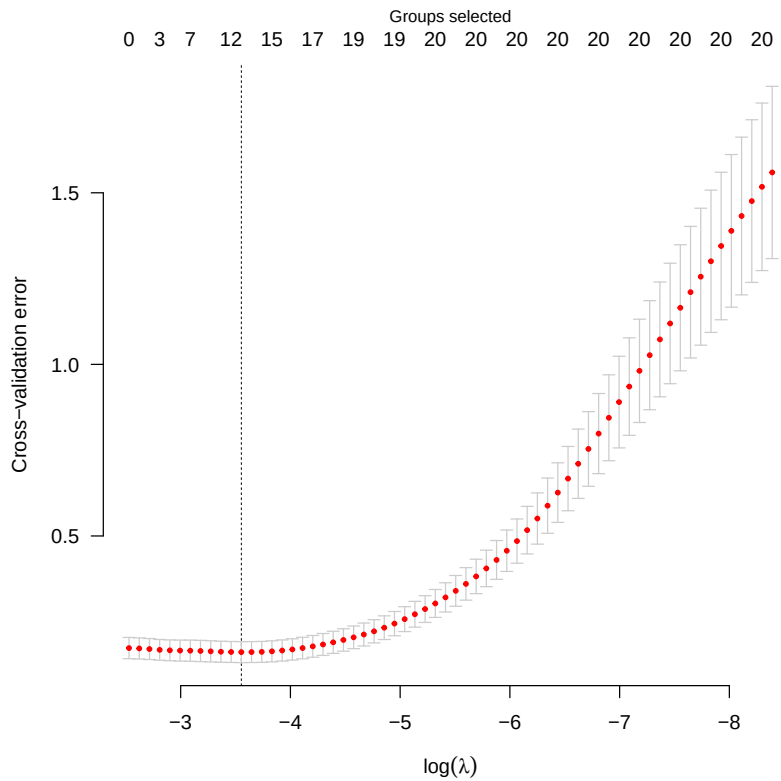
(b) Groupe lasso

Motivating Dataset: Gene Expression in Ovarian Cancer

- We now turn our attention to a second case study, involving gene expression changes in ovarian cancer, which brings up some issues we have not encountered previously
- The current standard treatment for ovarian cancer consists of surgery, followed by either carboplatin and paclitaxel or carboplatin alone
- This approach, however, is not effective for all patients
- The goal of this study was to identify genes and pathways associated with drug response
- To identify such genes, the investigators implanted ovarian cell lines into adult mice and allowed the tumors to grow for 2 months, at which point one of three treatments (carboplatin, carboplatin + paclitaxel, or control) was administered to each mouse
- At various time points ranging from 0 to 14 days following the initiation of treatment, the mice were sacrificed, at which point the investigators measured the size of the tumor as well as gene expression in the cancerous tissue

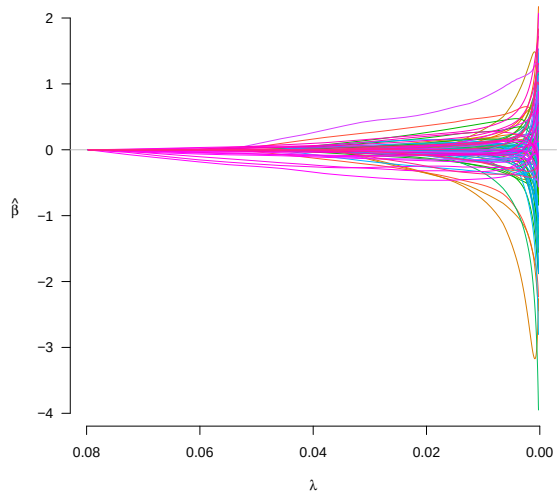
Group Lasso on Ovarian Cancer Dataset

```
library(mcgillHDA)
library(grpreg)
data("ovarian") #help(ovarian)
dim(ovarian$X)

## [1]   101 34694

X <- ovarian$X[,1:100] # take a subset of genes for compuational ease
groups <- rep(1:20,each=5) # put predictors into 20 non-overlapping groups
cvfit <- cv.grpreg(X=X,y=ovarian$y,group=groups, penalty = "grLasso", family = "gaussian")
plot(cvfit)
```



Group Lasso on Ovarian Cancer Dataset

```
plot(cvfit$fit)
```




```
summary(cvfit)
```

```
## grLasso-penalized linear regression with n=101, p=100
## At minimum cross-validation error (lambda=0.0286):
## -------------------------------------------------
##   Nonzero coefficients: 65
##   Nonzero groups: 13
##   Cross-validation error of 0.16
##   Maximum R-squared: 0.04
##   Maximum signal-to-noise ratio: 0.04
##   Scale estimate (sigma) at lambda.min: 0.402
```

Group Lasso on Ovarian Cancer Dataset

```
coef(cvfit)
```

```
##   (Intercept)         1-Dec         1-Mar        10-Mar        11-Mar
##  4.7150488015 -0.2364852512 -0.0240206564  0.0564242574  0.0515562538
##         2-Mar         3-Mar         4-Mar         5-Mar         6-Mar
##  0.2733450991 -0.0036730532  0.0373591631  0.0260141786 -0.0166263015
##         7-Mar           7A5         8-Mar         9-Mar          A1BG
## -0.0176897785  0.0085055976 -0.1608452850  0.0008361223  0.0569335160
##          A1CF         A26C3         A2BP1         A2LD1          A2M
## -0.1786026712  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##         A2ML1       A3GALT2        A4GALT         A4GNT          AAA1
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##          AAAS          AACS         AACSL         AADAC       AADACL1
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##       AADACL2       AADACL3       AADACL4         AADAT         AAGAB
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##          AAK1          AAMP         AANAT          AARS         AARS2
##  0.0000000000  0.1616359373  0.0522873498 -0.0926770214  0.1114519063
##        AARSD1         AASDH      AASDHPPT          AASS          AATF
## -0.0670354661  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##          AATK          ABAT         ABCA1        ABCA10        ABCA11
##  0.0000000000  0.0250005051 -0.0014849154  0.0079381718 -0.0217100878
##        ABCA12        ABCA13         ABCA2         ABCA3         ABCA4
##  0.0078423325  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##         ABCA5         ABCA6         ABCA7         ABCA8         ABCA9
##  0.0000000000  0.0000000000  0.0000000000  0.0000000000  0.0000000000
##         ABCB1        ABCB10        ABCB11         ABCB4         ABCB5
##  0.0000000000 -0.0474704267 -0.1918117104 -0.0598936858  0.1235449097
##         ABCB6         ABCB7         ABCB8         ABCB9         ABCC1
##  0.0754368749 -0.0007046247 -0.0021167986 -0.0046599570 -0.0012858981
##        ABCC10        ABCC11        ABCC12        ABCC13         ABCC2
##  0.0030977484 -0.0064286966 -0.0174967520 -0.0176222671  0.0158219190
##         ABCC3         ABCC4         ABCC5         ABCC6       ABCC6P1
##  0.0012623373 -0.0302924552 -0.0238386431  0.4515607326 -0.1293526014
##       ABCC6P2         ABCC8         ABCC9         ABCD1         ABCD2
## -0.0035310168 -0.0990251765  0.0171818366 -0.1404601554 -0.4270296644
##         ABCD3         ABCD4         ABCE1         ABCF1         ABCF2
## -0.0028367857 -0.2881639310 -0.0853203051  0.0224865614 -0.2651333438
##         ABCF3         ABCG1         ABCG2         ABCG4         ABCG5
##  0.0559535546 -0.0620337993 -0.0320294920  0.1346346248  0.1698735868
```

Group Lasso Model

- Assume the predictors in $\mathbf{X} \in \mathbb{R}^{n \times p}$ belong to K **non-overlapping groups** with **pre-defined** group membership and cardinality p_k
- Let $\boldsymbol{\beta}_{(k)}$ to denote the segment of $\boldsymbol{\beta}$ corresponding to group k
- We consider the group lasso penalized estimator

$$\min_{\boldsymbol{\beta}} L(\boldsymbol{\beta}|\mathbf{D}) + \lambda \sum_{k=1}^{K} w_k \|\boldsymbol{\beta}_{(k)}\|_2, \tag{2}$$

- where

$$L(\boldsymbol{\beta} \mid \mathbf{D}) = \frac{1}{2} \left[\mathbf{Y} - \widehat{\mathbf{Y}}\right]^\top \mathbf{W} \left[\mathbf{Y} - \widehat{\mathbf{Y}}\right] \tag{3}$$

  $\widehat{\mathbf{Y}} = \sum_{j=1}^{p} \beta_j X_j$, $\mathbf{D}$ is the working data $\{\mathbf{Y}, \mathbf{X}\}$, and $\mathbf{W}_{n \times n}$ is an observation weight matrix

Groupwise Descent: Exploiting Sparsity Structure

Minimize the objective function

$$\frac{1}{2}\left[\mathbf{Y}-\widehat{\mathbf{Y}}\right]^{\top}\mathbf{W}\left[\mathbf{Y}-\widehat{\mathbf{Y}}\right]+\lambda\sum_{k=1}^{K}w_k\|\boldsymbol{\beta}^{(k)}\|_2$$

During each sub-iteration only optimize $\boldsymbol{\beta}^{(k)}$. Set $\boldsymbol{\beta}^{(k')}=\widetilde{\boldsymbol{\beta}}^{(k')}$ for $k'\neq k$ at their current value.

1. Initialization: $\widetilde{\boldsymbol{\beta}}$
2. Cyclic groupwise descent: for $k=1,2,\ldots,K$, update $\boldsymbol{\beta}^{(k)}$ by minimizing the objective function

$$\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new})\leftarrow\arg\min_{\boldsymbol{\beta}^{(k)}}L(\boldsymbol{\beta}\mid\mathbf{D})+\lambda w_k\|\boldsymbol{\beta}^{(k)}\|_2$$

3. Repeat (2) till convergence.

Quadratic Majorization Condition

$$\arg\min_{\boldsymbol{\beta}^{(k)}} \frac{1}{2}\left[\mathbf{Y}-\widehat{\mathbf{Y}}\right]^{\top}\mathbf{W}\left[\mathbf{Y}-\widehat{\mathbf{Y}}\right]+\lambda\sum_{k=1}^{K}w_k\|\boldsymbol{\beta}^{(k)}\|_2 \tag{4}$$

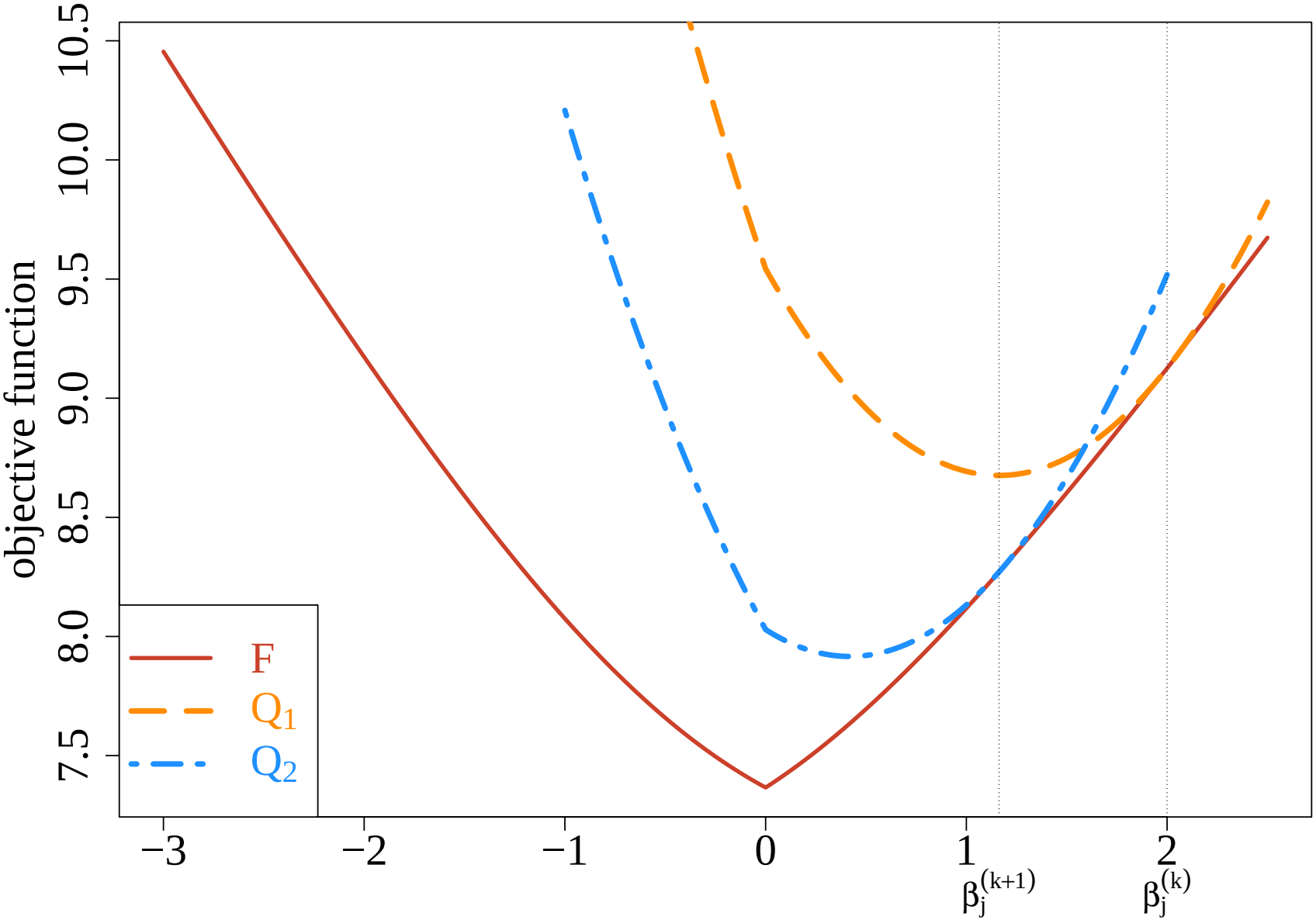
- Unfortunately, there is no closed form solution to (4)
- However, the loss function $L(\boldsymbol{\beta}|\mathbf{D})$ satisfies the quadratic majorization (QM) condition[1], since there exists
  - a $p \times p$ matrix $\mathbf{H} = \mathbf{X}^{\top}\mathbf{W}\mathbf{X}$, and
  - $\nabla L(\boldsymbol{\beta}|\mathbf{D}) = -\left(Y-\hat{Y}\right)^{\top}\mathbf{W}\mathbf{X}$

  which may only depend on the data $\mathbf{D}$, such that for all $\boldsymbol{\beta}, \boldsymbol{\beta}^*$,

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \leq L(\boldsymbol{\beta}^* \mid \mathbf{D}) + (\boldsymbol{\beta}-\boldsymbol{\beta}^*)^{\intercal}\nabla L(\boldsymbol{\beta}^*|\mathbf{D}) + \frac{1}{2}(\boldsymbol{\beta}-\boldsymbol{\beta}^*)^{\intercal}\mathbf{H}(\boldsymbol{\beta}-\boldsymbol{\beta}^*)$$

[1] Yang and Zou. Statistical Computing (2014)

Generalized Coordinate Descent (GCD)



Groupwise Majorization Descent

- Update $\boldsymbol{\beta}$ in a groupwise fashion

$$\boldsymbol{\beta} - \widetilde{\boldsymbol{\beta}} = (\underbrace{0, \ldots, 0}_{k-1}, \boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)}, \underbrace{0, \ldots, 0}_{K-k})$$

- Only need to compute the majorization function on group level

$$L(\boldsymbol{\beta} \mid \mathbf{D}) \leq L(\widetilde{\boldsymbol{\beta}} \mid \mathbf{D}) - (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal} U^{(k)} + \frac{1}{2}\gamma_k (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})^{\intercal} (\boldsymbol{\beta}^{(k)} - \widetilde{\boldsymbol{\beta}}^{(k)})$$

$$U^{(k)} = \frac{\partial}{\partial \boldsymbol{\beta}_{(k)}} L(\boldsymbol{\beta} \mid \mathbf{D}) = -\left(Y - \hat{Y}\right)^{\top} \mathbf{W} \mathbf{X}_{(k)}$$

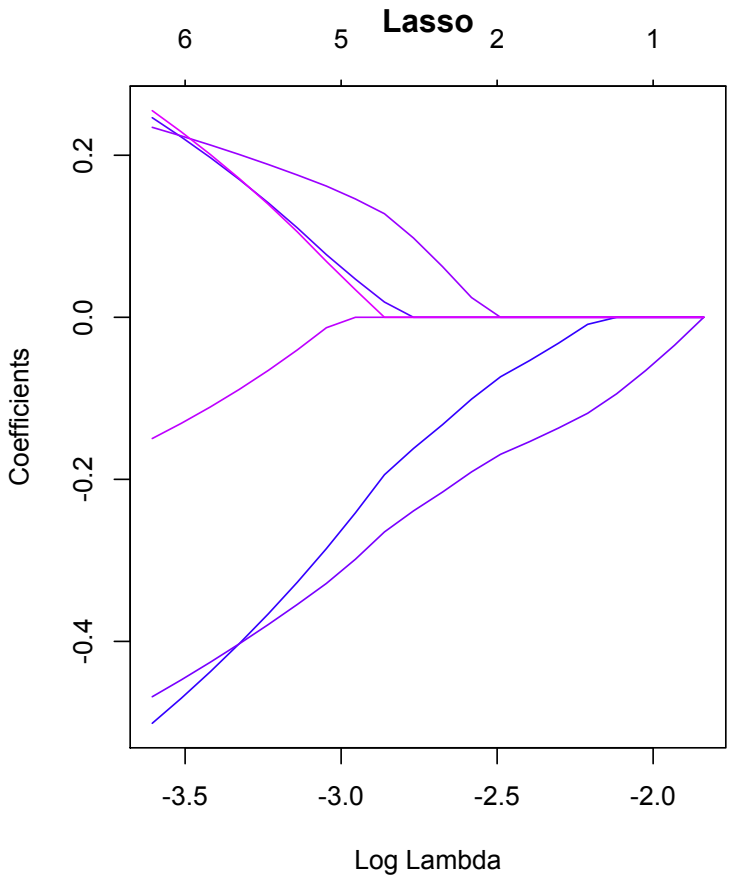
$$\mathbf{H}^{(k)} = \frac{\partial^2}{\partial \boldsymbol{\beta}_{(k)} \partial \boldsymbol{\beta}_{(k)}^{\top}} L(\boldsymbol{\beta} \mid \mathbf{D}) = \mathbf{X}_{(k)}^{\top} \mathbf{W} \mathbf{X}_{(k)}$$

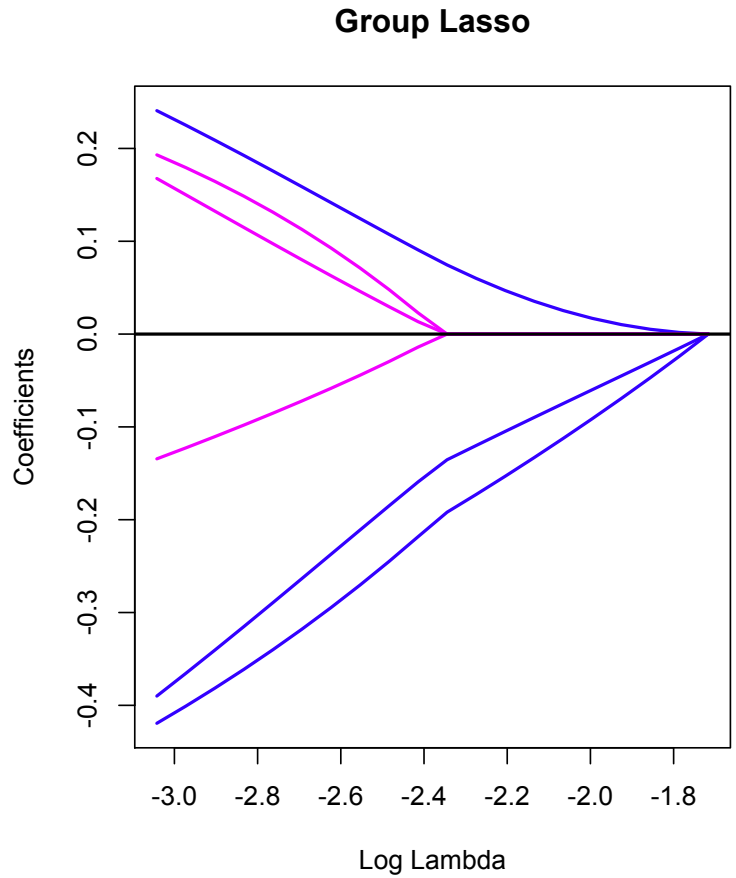
- $\gamma_k = \text{eigen}_{\max}(\mathbf{H}^{(k)})$
- Update $\widetilde{\boldsymbol{\beta}}^{(k)}$ with a fast operation:

$$\widetilde{\boldsymbol{\beta}}^{(k)}(\text{new}) = \frac{1}{\gamma_k}\left(U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)}\right)\left(1 - \frac{\lambda w_k}{\| U^{(k)} + \gamma_k \widetilde{\boldsymbol{\beta}}^{(k)} \|_2}\right)_+$$

Lasso vs. Group Lasso

- Logistic regression with group lasso: $n = 50$, $p = 6$.
- Group lasso: specify $(\beta_1, \beta_2, \beta_3)$, $(\beta_4, \beta_5, \beta_6)$. Variable selection at the group level.
- Solution path: view $\boldsymbol{\beta}$ as function of λ.





Generalizations of the Lasso Penalty

Generalized penalties arise in a wide variety of settings:

- **Adaptive lasso:** a lasso with the Oracle property.
- **Elastic net:** handle highly correlated features. e.g. genes.
- **SCAD and MCP:** non-convex penalties with the Oracle property.
- **Multitask lasso:** handle between-tasks sparsity while allowing within-task sparsity.

Adaptive Lasso

The adaptive lasso estimator

$$\widehat{\boldsymbol{\beta}}^{\text{alasso}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_n \sum_{j=1}^{p} \hat{w}_j |\beta_j|, \tag{5}$$

where $\hat{w}_j = \frac{1}{|\hat{\beta}_j|^\gamma}$ for some $\gamma > 0$ and a $\sqrt{n}$-consistent estimator $\hat{\beta}_j$ of β_j.

- For Lasso, if an irrelevant variable is highly correlated with variables in the true model, the lasso may fail to distinguish it from the true variables even with large n.
- As $n \to \infty$, the weights corresponding to insignificant variables tend to infinity, while the weights corresponding to significant variables converge to a finite constant.
- Zou (2006) showed that, under certain regularity conditions, the adaptive lasso has the **oracle property**.

Adaptive Lasso: Gene Expression in Ovarian Cancer

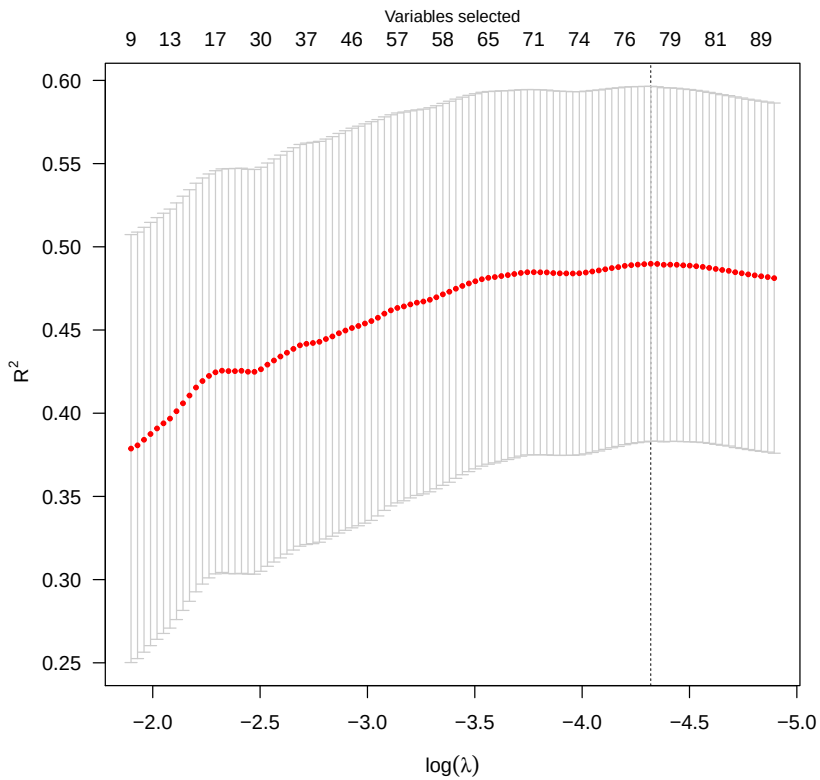
- Our goal is to assess the relationship between gene expression and tumor growth
- However, it is important to adjust for treatment group and time of collection in analyzing these data, both of which have significant effects on tumor size
- The lasso model is easily extended to allow for such an analysis

Coefficient-specific λ values

- Up to this point, we have kept λ the same across all variables, but all of our derivations can be easily modified to allow variable j to have its own regularization parameter, λ_j
- This is achieved via the `penalty.factor` argument in `glmnet` or `ncvreg`
- This argument allows one to modify the penalty applied to individual covariates through the use of a weighting factor: $\lambda_j = \lambda w_j$, where w_j is the multiplicative factor applied to term j
- The idea here is that w_j scales the baseline regularization factor λ up or down for certain covariates
- By assigning $w_j = 0$ for the treatment group and time of collection variables, we can include them in the model as unpenalized covariates
- The rationale for penalizing the gene expression variables is that we expect most genes to have no effect on relative tumor volume, but it does not make sense to extend that assumption to treatment group and time of collection

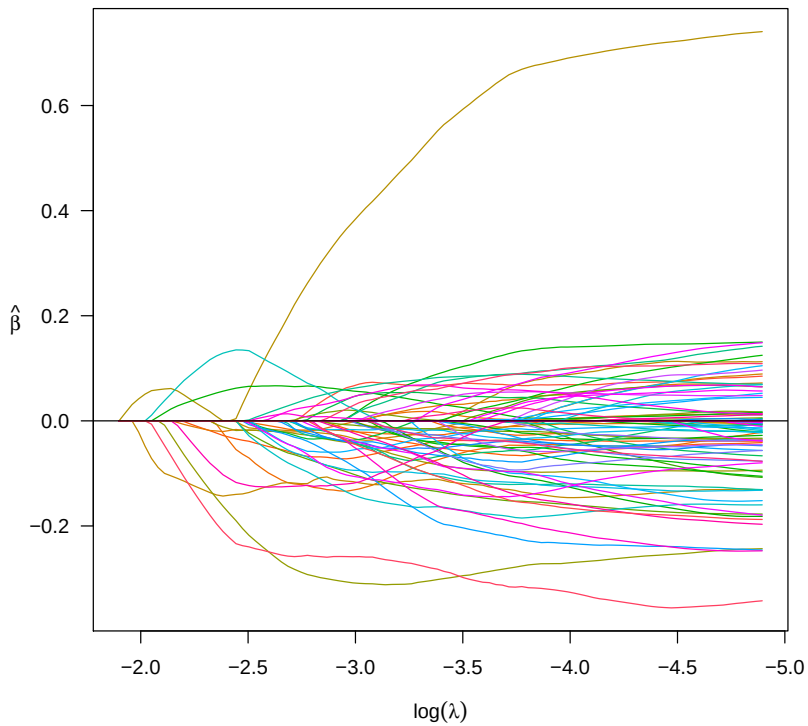
Adaptive Lasso on Ovarian Cancer Dataset

```
library(mcgillHDA); library(ncvreg); library(splines)
data("ovarian")
sDay <- ns(ovarian$sData$Day, df=2)
X0 <- model.matrix(~ Treatment*sDay, ovarian$sData)[,-1]
multiplier <- rep(0:1, c(ncol(X0), ncol(ovarian$X)))
XX <- cbind(X0, ovarian$X)
cvfit <- cv.ncvreg(XX, ovarian$y, penalty.factor=multiplier, penalty='lasso')
plot(cvfit, type='rsq')
```



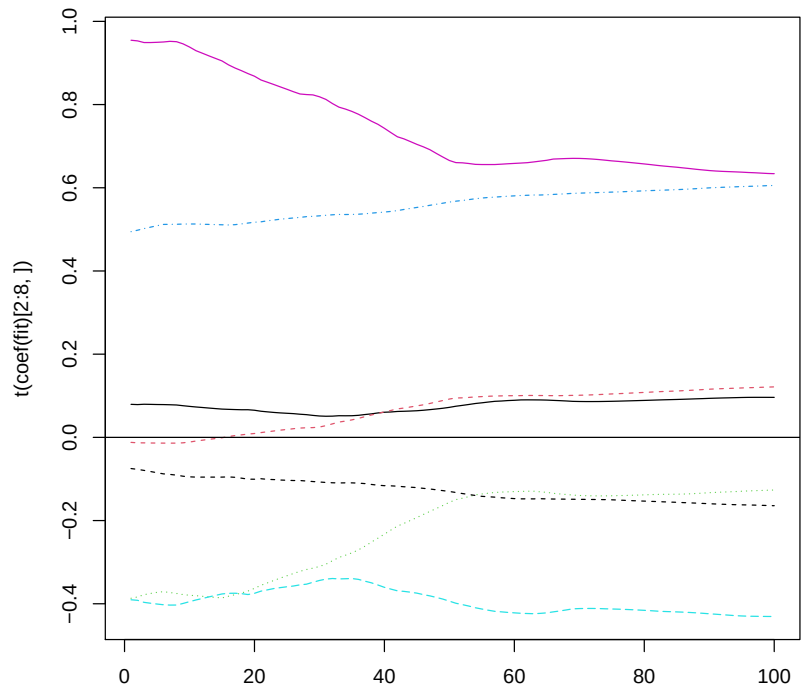
Adaptive Lasso on Ovarian Cancer Dataset

```
fit <- cvfit$fit
plot(fit, log=TRUE)
```



Unpenalized Covariates are always non-zero

```
matplot(t(coef(fit)[2:8,]), type="l")
abline(h=0)
```



Elastic Net

The **elastic net** (Zou and Hastie, 2005) solves the convex program

$$\min_{\boldsymbol{\beta}} \frac{1}{2}\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \left[\frac{1}{2}(1-\alpha)\|\boldsymbol{\beta}\|_2^2 + \alpha\|\boldsymbol{\beta}\|_1\right]$$

where $\alpha \in [0, 1]$ is a parameter. The penalty applied to an individual coefficient (disregarding the regularization weight $\lambda > 0$) is given by

$$\frac{1}{2}(1-\alpha)\beta_j^2 + \alpha|\beta_j|.$$

- The coefficients are selected approximately together in their groups.
- The coefficients approximately share their values equally.

An simulated example

- Two independent "hidden" factors $\mathbf{z}_1$ and $\mathbf{z}_2$

$$\mathbf{z}_1 \sim U(0, 20), \qquad \mathbf{z}_2 \sim U(0, 20)$$

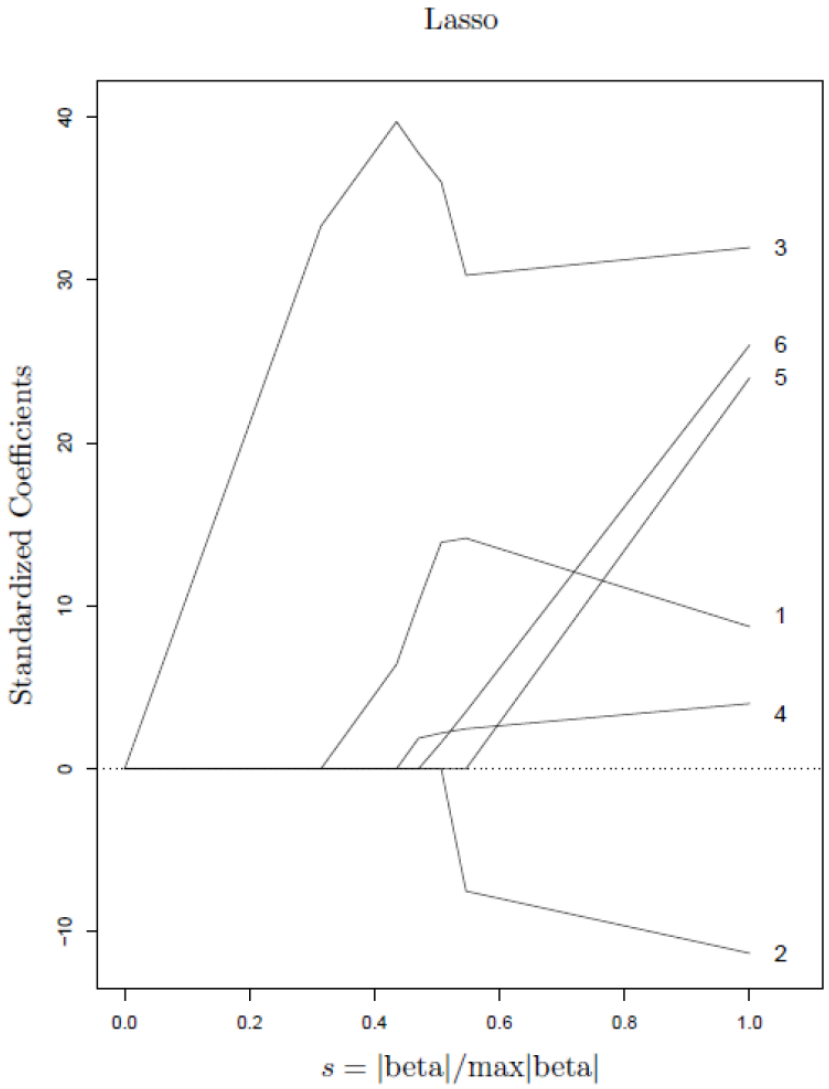
- Generate the response vector $\mathbf{y} = \mathbf{z}_1 + 0.1 \cdot \mathbf{z}_2 + N(0, 1)$
- Suppose only observe predictors

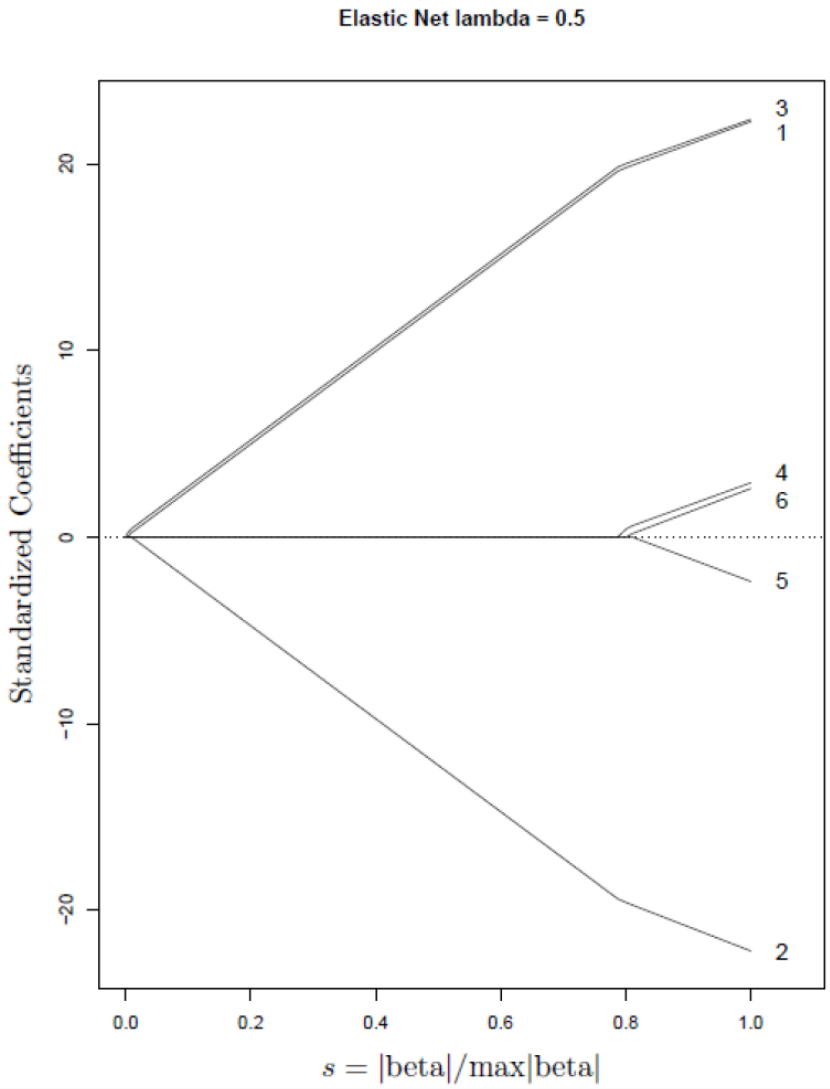
$$\mathbf{x}_1 = \mathbf{z}_1 + \epsilon_1, \quad \mathbf{x}_2 = \mathbf{z}_1 + \epsilon_2, \quad \mathbf{x}_3 = \mathbf{z}_1 + \epsilon_3$$

$$\mathbf{x}_4 = \mathbf{z}_2 + \epsilon_4, \quad \mathbf{x}_5 = \mathbf{z}_2 + \epsilon_5, \quad \mathbf{x}_6 = \mathbf{z}_2 + \epsilon_6$$

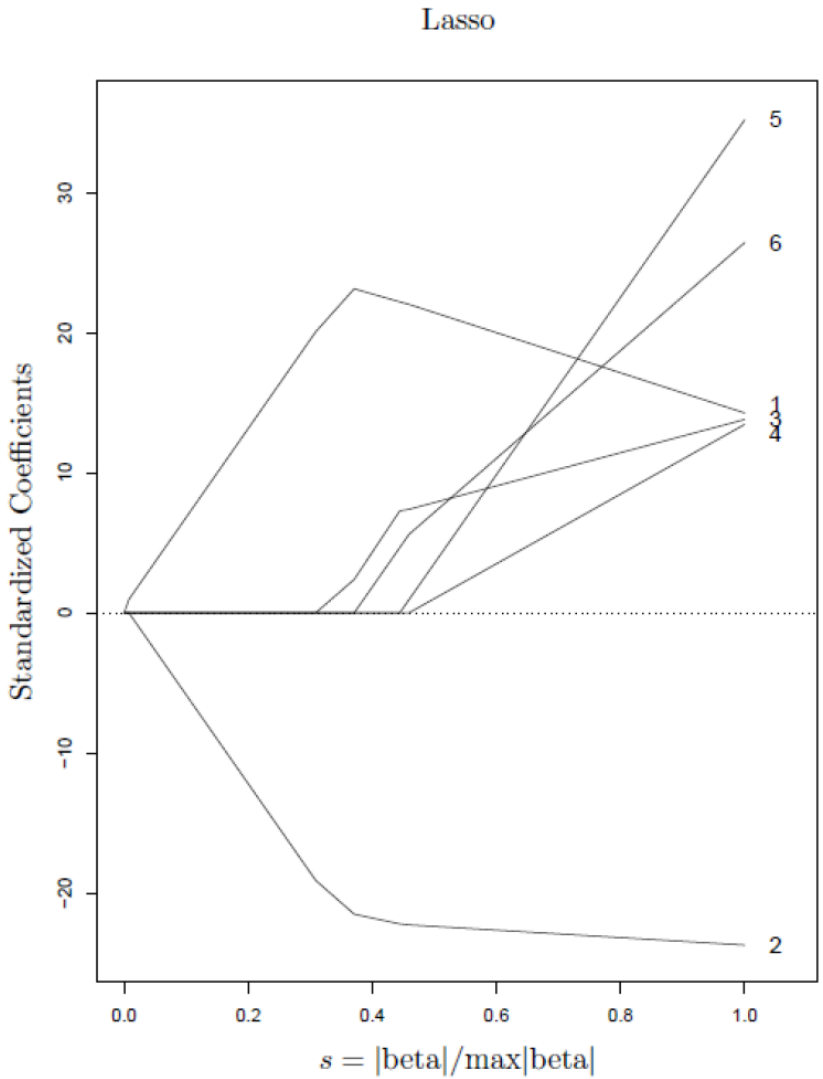
- Fit the model on $(\mathbf{X}, \mathbf{y})$
- An "oracle" would identify $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ (the $\mathbf{z}_1$ group) as the most important variables.

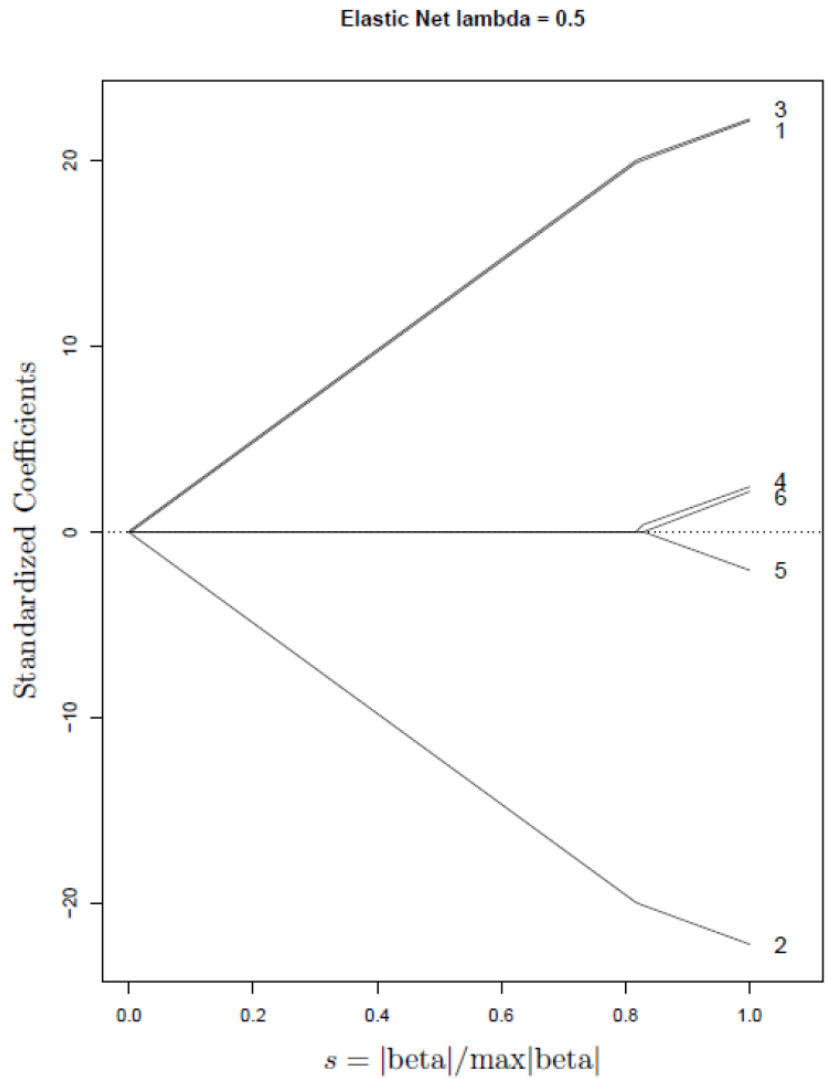
Simulation 1



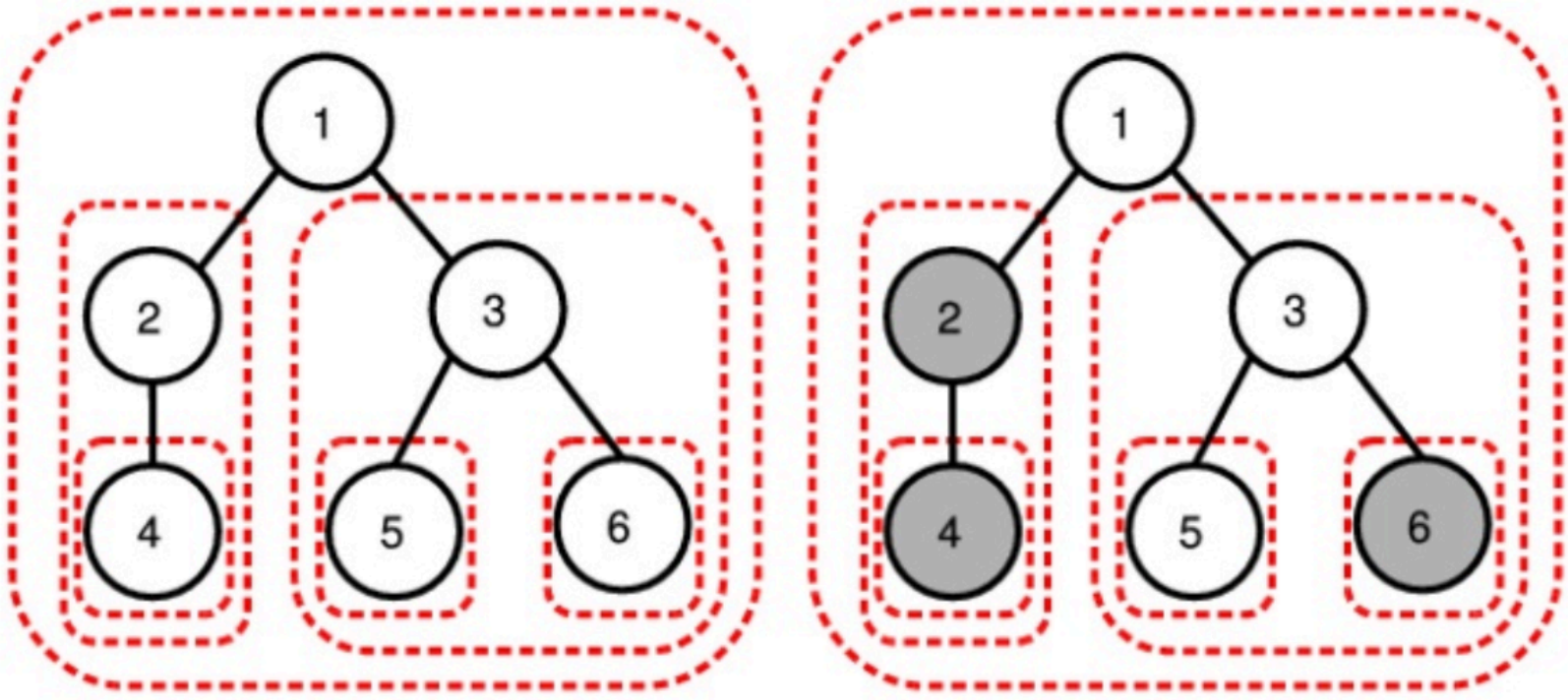


Simulation 2





Hierarchical Group Lasso



A node can be active only if its **ancestors are active**.
The selected patterns are **rooted subtrees.**

Optimization via efficient proximal methods (same cost as ℓ_1)

(Jenatton, Mairal, Obozinski, and Bach 2010)

Multitask Lasso

Suppose that we have K regression tasks

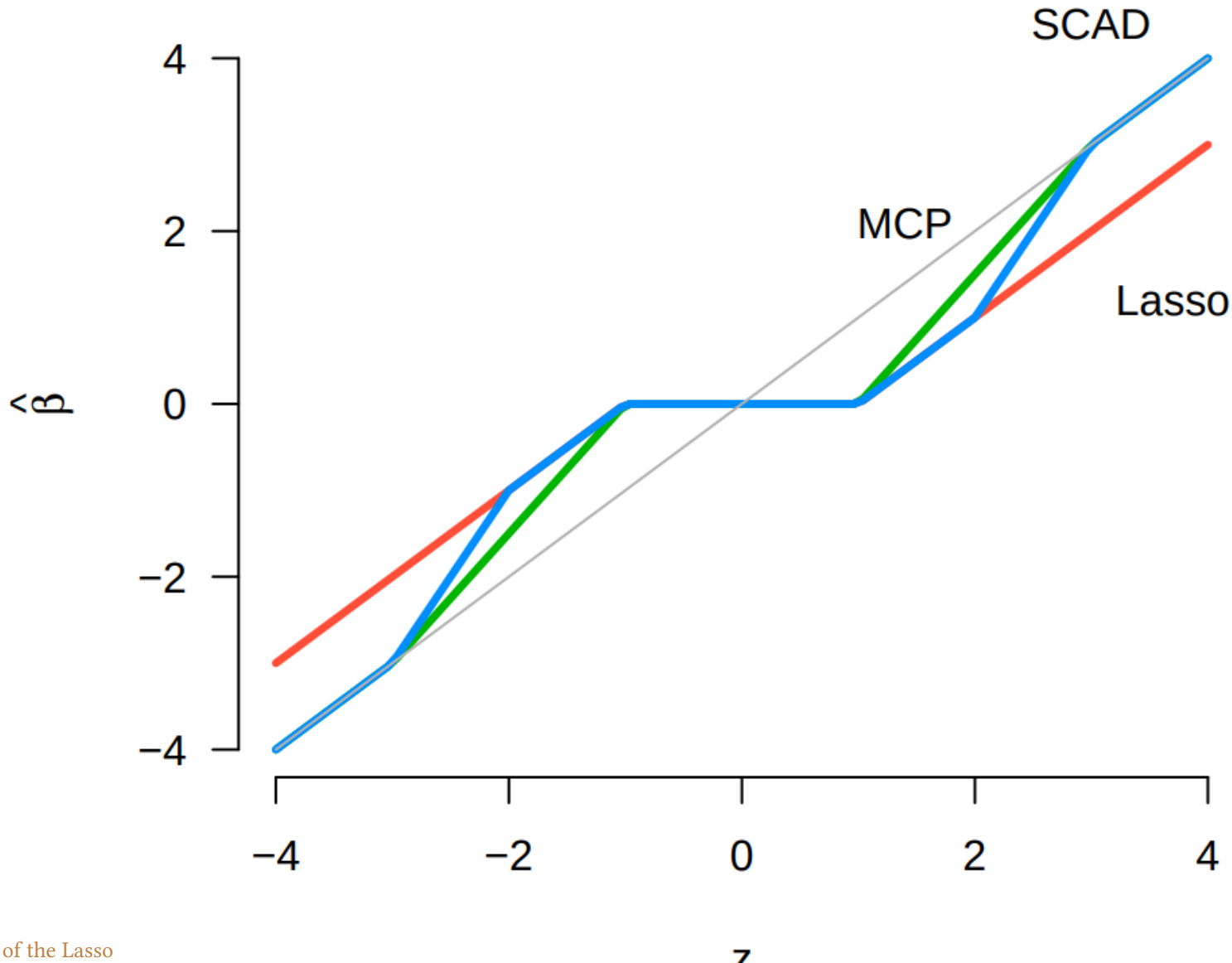
$$Y^{(k)} = \mathbf{X}^{(k)}\boldsymbol{\beta}^{(k)} + \epsilon^{(k)}, \qquad k = 1, \ldots, K.$$

- The k-th task has n_k observations for $k = 1, \ldots, K$
- $Y^{(k)} = (y_1^{(k)}, \ldots, y_{n_k}^{(k)})^\intercal$, $X_j^{(k)} = (x_{1j}^{(k)}, \ldots, x_{n_k j}^{(k)})^\top$
- $\mathbf{X}^{(k)} = (X_1^{(k)}, \ldots, X_p^{(k)})$ be the $n_k \times p$ design matrix for task k
- $\boldsymbol{\beta}^{(k)} = (\beta_1^{(k)}, \cdots, \beta_p^{(k)})^\top$ and $\boldsymbol{\beta}_j = (\beta_j^{(1)}, \cdots, \beta_j^{(K)})^\top$
- find commonly shared relevant covariates and retains the ability to recover covariates unique to individual data sources.

$$\min_{\boldsymbol{\beta}} \frac{1}{2} \sum_{k=1}^{K} \left\| Y^{(k)} - \mathbf{X}^{(k)}\boldsymbol{\beta}^{(k)} \right\|^2 + \lambda P_\alpha(\boldsymbol{\beta}),$$

$$P_\alpha(\boldsymbol{\beta}) = \sum_{j=1}^{p} w_j \left[(1-\alpha)||\boldsymbol{\beta}_j||_q + \alpha||\boldsymbol{\beta}_j||_1 \right]$$

SCAD (Fan et Li, JASA, 2001), MCP (Zhang, Ann. Stat., 2010)



Discussion

- Variable selection is an active area of research
- Few inference tools exist
- Robust software has been developed, but more scalable algorithms and implementations are needed

References

- Fan, J. and Li, R., 2001. Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American statistical Association, 96(456), pp.1348-1360.
- Tibshirani, R., 1996. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), pp.267-288.
- Friedman, J., Hastie, T., Höfling, H. and Tibshirani, R., 2007. Pathwise coordinate optimization. The annals of applied statistics, 1(2), pp.302-332.
- Buhlmann, P. & van de Geer, S. (2011), Statistics for High-Dimensional Data, Springer.
- Breheny, P. BIOS 7240 class notes (accessed March 15, 2019).
- Tibshirani, R. A Closer Look at Sparse Regression (accessed March 15, 2019).
- Gaillard, P. and Rudi, A. Introduction to Machine Learning (accessed March 15, 2019).
- Hastie, T., Tibshirani, R. & Friedman, J. (2009), The Elements of Statistical Learning; Data Mining, Inference and Prediction, Springer. Second edition.
- Hastie, T., Tibshirani, R. & Wainwright, M. (2015), Statistical Learning with Sparsity: the Lasso and Generalizations, Chapman & Hall.

Session Info

```
R version 4.0.2 (2020-06-22)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Pop!_OS 20.10

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.10.so

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] mcgillHDA_0.1.0 hdrm_0.3.7      ncvreg_3.13.0   glmnet_4.1-1
[5] Matrix_1.2-18   knitr_1.33

loaded via a namespace (and not attached):
 [1] lattice_0.20-41  codetools_0.2-16 gglasso_1.5      digest_0.6.27
 [5] foreach_1.5.1    grid_4.0.2       pacman_0.5.1     magrittr_2.0.1
 [9] evaluate_0.14    highr_0.9        stringi_1.6.2    grpreg_3.3.1
[13] splines_4.0.2    iterators_1.0.13 tools_4.0.2      stringr_1.4.0
[17] xfun_0.24        survival_3.2-3   compiler_4.0.2   shape_1.4.6
```