Dimension reduction: Penalized regression

Statistical Methods in Bioinformatics

Claus Thorn Ekstrøm
UCPH Biostatistics
ekstrom@sund.ku.dk

Slides: biostatistics.dk/teaching/bioinformatics/

Quick refresher on GLMs

Multiple regression / linear models refresher

Model y as a function of predictors $x_1, \ldots, x_P$. For individual i

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_P x_{Pi} + \epsilon_i$$

or in matrix notation

$$Y = X\beta + \epsilon$$

The overall trend or mean effect is

$$E(Y) = X\beta$$

Interpretation of parameters

- Each coefficient, β_j, expresses the expected change in the response y when x_j changes one unit and the remaining explanatory variables are held fixed.
- Might be that we have a significant effect of x_1 in the univariate model, while its effect in the mult. lin. model is non-significant. Its effect is explained by the other explanatory variables.

Generalized linear models

Extends the linear more to other outcomes and distributions. Still focus on the **mean effect** but for the transformed data.

$$g(E[Y_i]) = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_P X_{Pi}$$

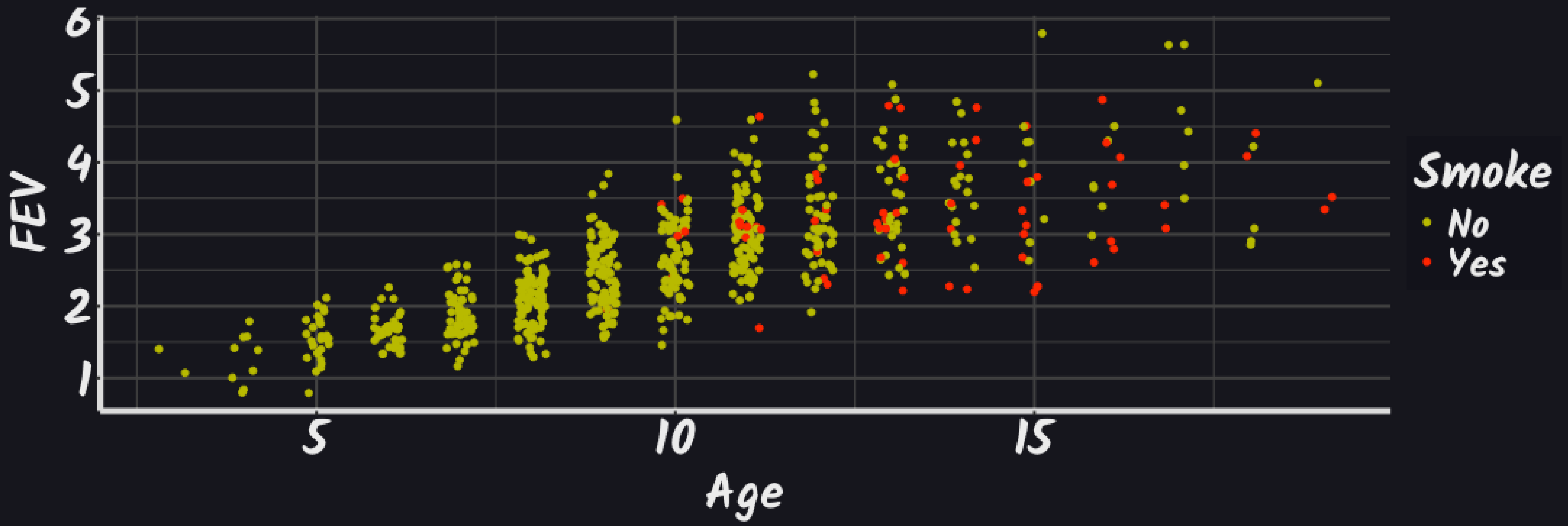
or equivalently

$$E[Y_i] = g^{-1}(\beta_0 + \beta_1 X_{1i} + \cdots + \beta_P X_{Pi})$$

where the **link function** g in principle could be any function that maps the population mean into the **linear predictor**.

Example: FEV in children

Effect of smoking on the pulmonary function of children. $N = 654$. Info on `age`, `height`, `sex`, `smoking`, and `fev` (forced expiratory volume).



```r
library("broom")
res <- lm(FEV ~ Age + Ht + I(Ht^2) + Gender + Smoke,
          data=fev)
tidy(res)
```

```
# A tibble: 6 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   6.83     1.50         4.54 6.68e- 6
2 Age           0.0700   0.00913      7.67 6.31e-14
3 Ht          -10.7      1.96        -5.46 6.85e- 8
4 I(Ht^2)       4.81     0.635        7.58 1.21e-13
5 GenderBoy     0.0939   0.0330       2.85 4.51e- 3
6 SmokeYes     -0.136    0.0573      -2.38 1.76e- 2
```

Each unit change in an x produces an **average** effect in y.

Example: Who survived the Titanic disaster?

$N = 2201$ individuals on the Titanic. Who survived? Info on `Class` (1st, 2nd, 3rd, Crew), `Sex`, `Age` group, and survival status.

$$\log\left(\frac{P(Y=1)}{1-P(Y=1)}\right) = X\beta = \beta_0 + \beta_1 x_1 + \cdots + \beta_5 x_5$$

and then

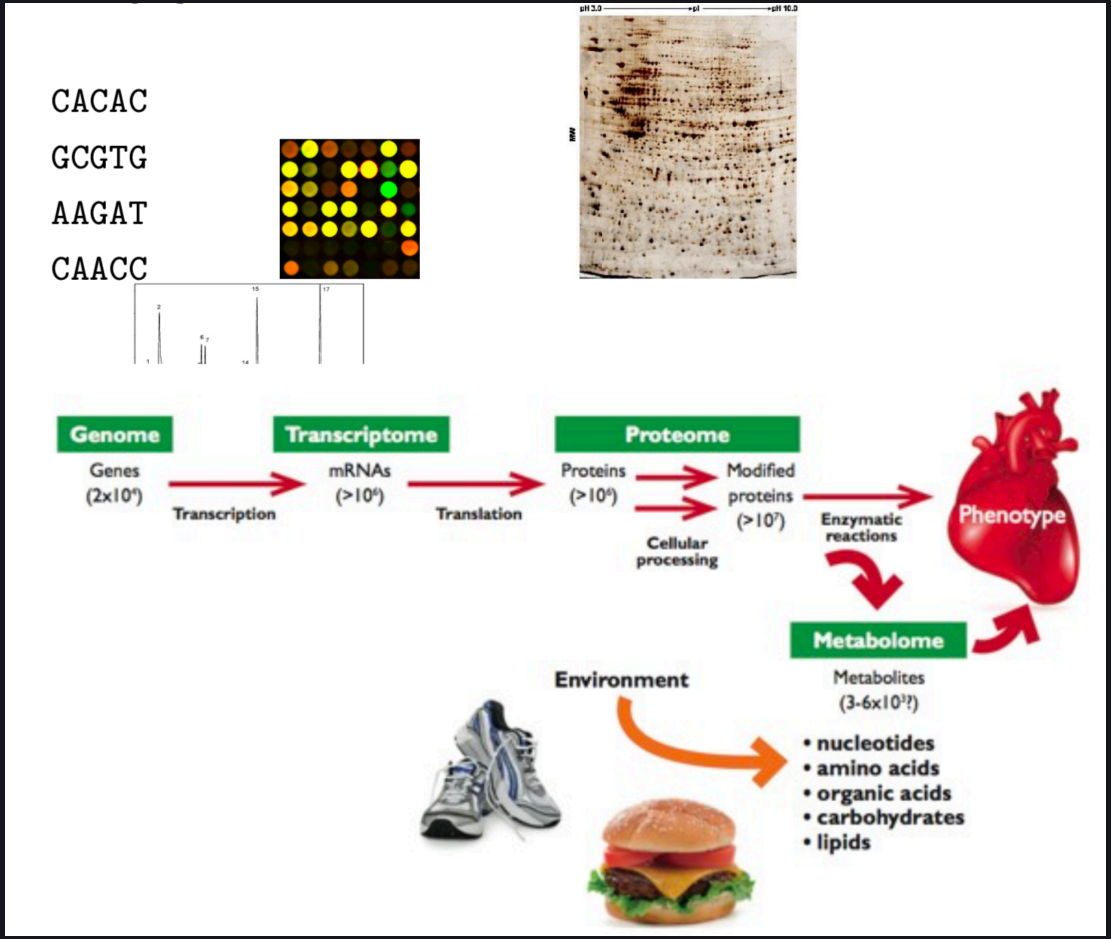
$$P(Y=1) = \frac{\exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_5 x_5)}{1+\exp(\beta_0 + \beta_1 x_1 + \cdots + \beta_5 x_5)}$$

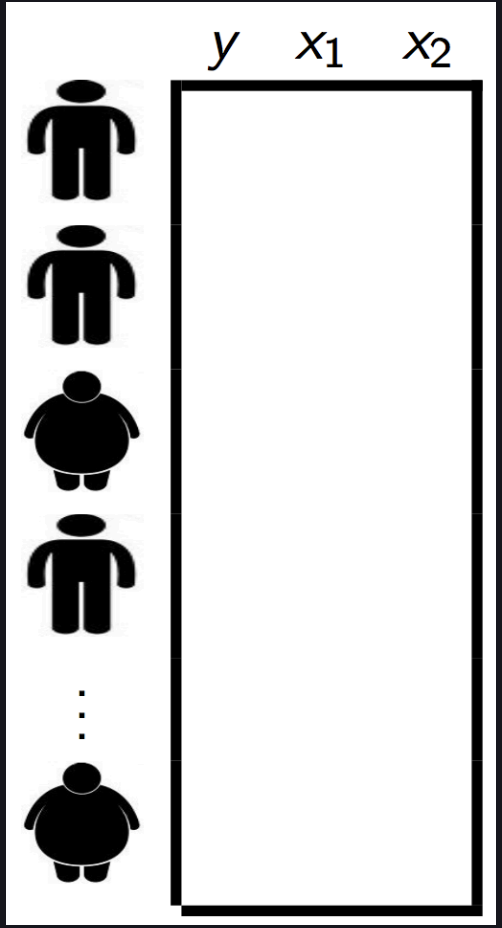
```r
res <- glm(Survived ~ Class + Sex + Age,
           data=titanic, family=binomial)
tidy(res)
```

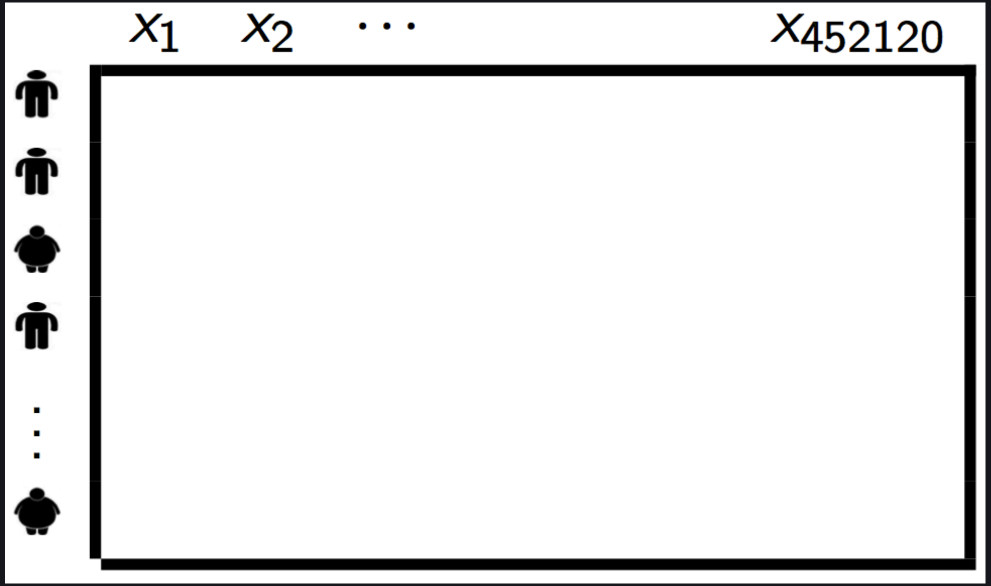
```
# A tibble: 6 × 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)    0.685     0.273      2.51 1.21e- 2
2 Class2nd      -1.02      0.196     -5.19 2.05e- 7
3 Class3rd      -1.78      0.172    -10.4  3.69e-25
4 ClassCrew     -0.858     0.157     -5.45 5.00e- 8
5 SexFemale      2.42      0.140     17.2  1.43e-66
6 AgeAdult      -1.06      0.244     -4.35 1.36e- 5
```

Estimates are log odds ratios. Use `exp()` to transform back to odds ratios.

The joy of big data




y
x_1
x_2


x_1
x_2
$\cdots$
x_{452120}

Penalized regression

Penalized regression models

Recall the generalized linear model

$$g(E[Y_i]) = \beta_0 + \beta_1 X_{1i} + \cdots + \beta_P X_{Pi} = X\beta$$

or equivalently

$$E[Y_i] = g^{-1}(\beta_0 + \beta_1 X_{1i} + \cdots + \beta_P X_{Pi}) = g^{-1}(X\beta)$$

where the function g is in principle any function that maps the population mean into the **linear predictor**. g is called the **link function**

Estimator in linear regression

The ordinary least squares estimator is

$$\hat{\beta} = (X^t X)^{-1} X^t y$$

Written as least squares, so $\hat{\beta}$ minimizes

$$(y - X\hat{\beta})^t (y - X\hat{\beta}) = \|y - X\beta\|_2^2$$

What do we do then P is large?

Penalized regression - the lasso

Assume linear mean effect: $Y = X\beta + \epsilon$

The **Lasso** estimates β by minimizing the penalized LS function

$$\begin{aligned} Z_n(\beta) &= \frac{1}{N}(Y - X\beta)^t(Y - X\beta) + \lambda_N\|\beta\| \\ &= \|(Y - X\beta)\|_2^2 + \lambda_N\|\beta\|_1 \end{aligned}$$

so $\hat{\beta} = \text{arg min}_{\beta \in R^P}\ Z_n(\beta)$

Properties of the lasso

Even for $P > N$ or $P \gg N$ regularized regression methods can:

- **select** a sparse model
- lead to accurate prediction

Limitations of lasso type regularization:

- not consistent in variable selection
- non-standard limiting distribution
- no oracle property
- multiple testing problem

Example: Biopsy Data on Breast Cancer Patients

Biopsies of breast tumours for 699 patients up to 15 July 1992; each of nine attributes has been scored on a scale of 1 to 10, and the outcome is also known.

Predictors: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei (16 values are missing), bland chromatin, normal nucleoli, mitoses.

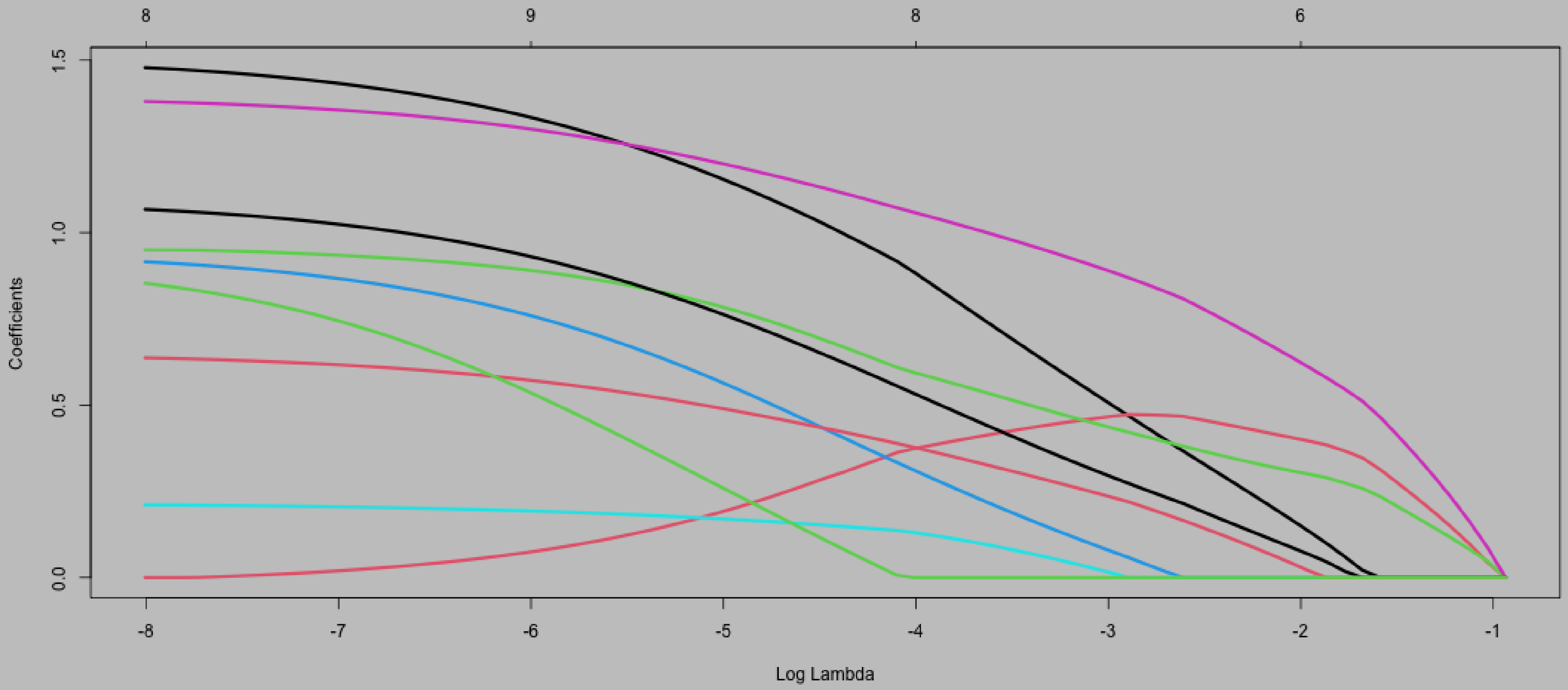
Output is "benign" or "malignant".

Example: The biopsy data

```r
library("MASS")  # Get data
data(biopsy)
# Remove missing variables
ccb <- complete.cases(biopsy)
predictors <- as.matrix(scale(biopsy[ccb,2:10]))
y <- as.numeric(biopsy$class[ccb])-1

library("glmnet")
res <- glmnet(predictors, y, family="binomial")
```

```
plot(res, lwd=3, xvar="lambda")
```



Quick digression: Cross-validation

How do we choose the penalty λ?

- Want it large to reduce the number of predictors
- Want it small to have as good a models as possible

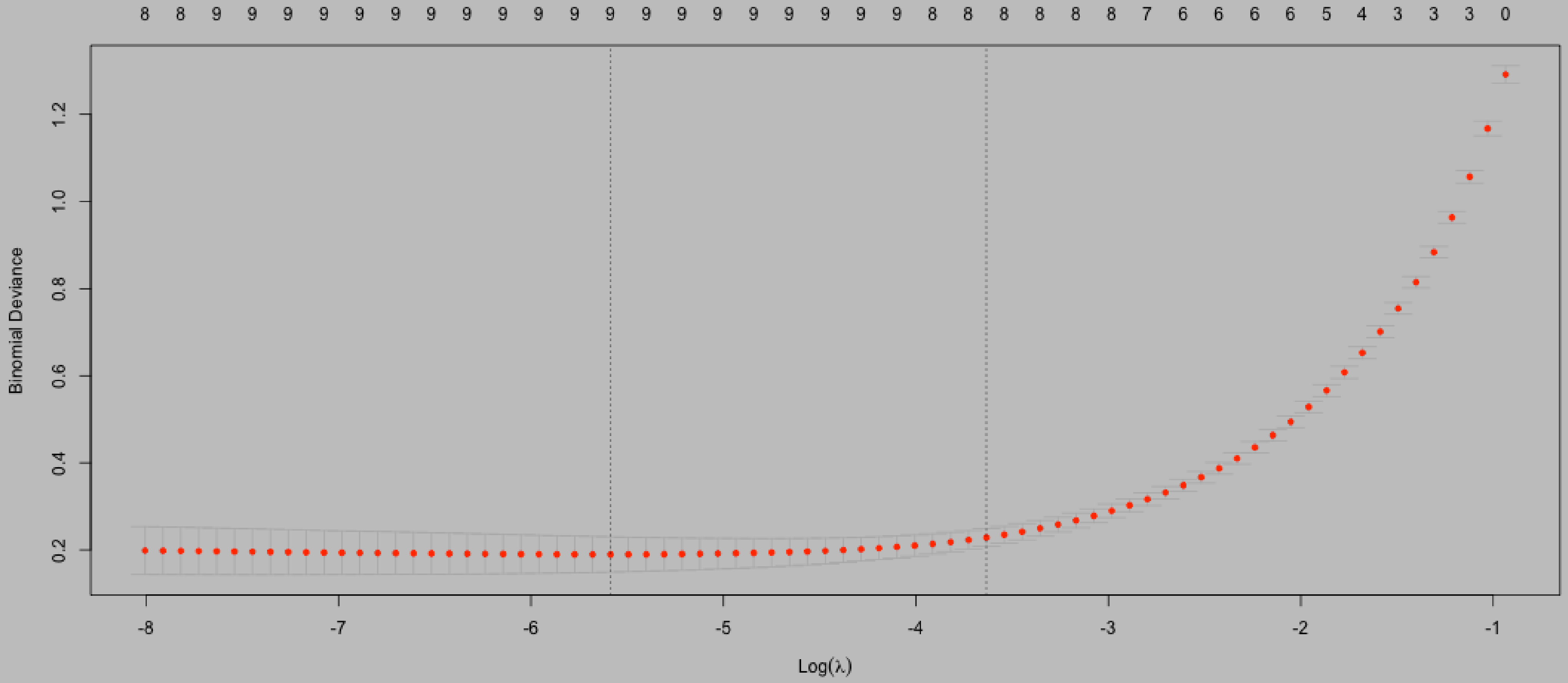
Split and test

Example: The biopsy data

```r
coef(res, s=.3)
```

```
10 x 1 sparse Matrix of class "dgCMatrix"
                    s1
(Intercept) -0.6425224
V1           .        
V2           0.1394930
V3           0.1102769
V4           .        
V5           .        
V6           0.2221975
V7           .        
V8           .        
V9           .        
```

```r
res2 <- cv.glmnet(predictors, y, family="binomial")
plot(res2, lwd=3, cex=1.8)
```



Example: The biopsy data

```r
coef(res, s=exp(-3.3))
```

```
10 x 1 sparse Matrix of class "dgCMatrix"
                     s1
(Intercept) -0.93594535
V1           0.61727404
V2           0.44331361
V3           0.48275683
V4           0.14338234
V5           0.05694629
V6           0.94445772
V7           0.36293816
V8           0.28012376
V9           .
```

Delassoing

We *know* the lasso estimates are shrunk towards zero.

What if we want optimized prediction?

Ad hoc approach: delassoing

1. Use lasso for predictor selection
2. Refit model with only selected predictors to obtain unbiased estimates

```r
predictors <- scale(predictors)
res <- glm(y ~ predictors, family="binomial")
tidy(res)
```

```
# A tibble: 10 × 5
   term         estimate std.error statistic   p.value
   <chr>           <dbl>     <dbl>     <dbl>     <dbl>
 1 (Intercept)   -1.09       0.323   -3.38   0.000714
 2 predictorsV1   1.51       0.401    3.77   0.000165
 3 predictorsV2  -0.0192     0.641   -0.0300 0.976
 4 predictorsV3   0.964      0.689    1.40   0.162
 5 predictorsV4   0.947      0.354    2.68   0.00740
 6 predictorsV5   0.215      0.348    0.617  0.537
 7 predictorsV6   1.40       0.342    4.08   0.0000447
 8 predictorsV7   1.10       0.420    2.61   0.00907
 9 predictorsV8   0.650      0.345    1.89   0.0591
10 predictorsV9   0.927      0.570    1.63   0.104
```

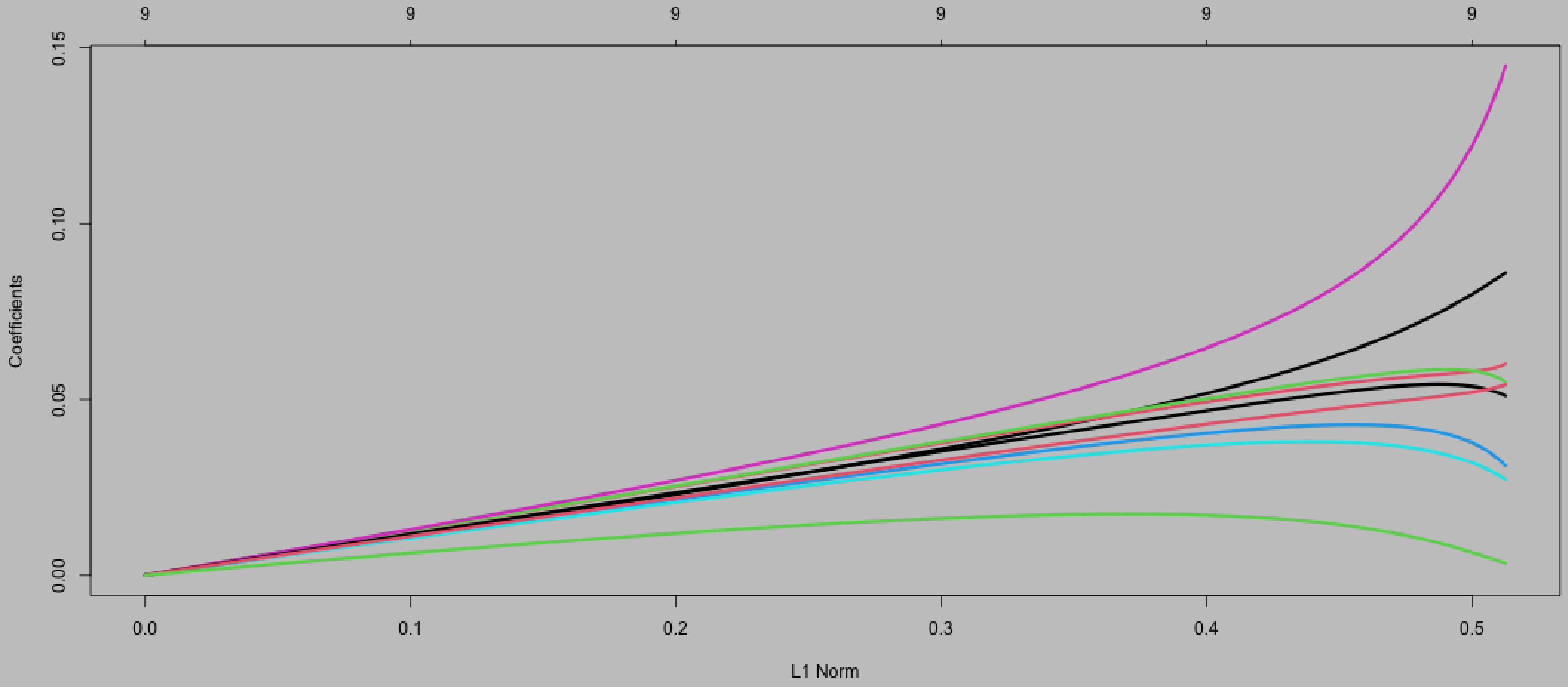
Other penalties

The **Ridge regression** estimates β by minimizing the penalized LS function

$$Z_n(\beta) = \|(Y - X\beta)\|_2^2 + \lambda_N \|\beta\|_1^2$$

this corresponds toadding a small value to the diagonal elements of the correlation matrix of the parameters. Introduces bias but reduces variation. No selection but all predictors scaled down.

```r
res <- glmnet(predictors, y, alpha = 0)
plot(res, lwd=3)
```

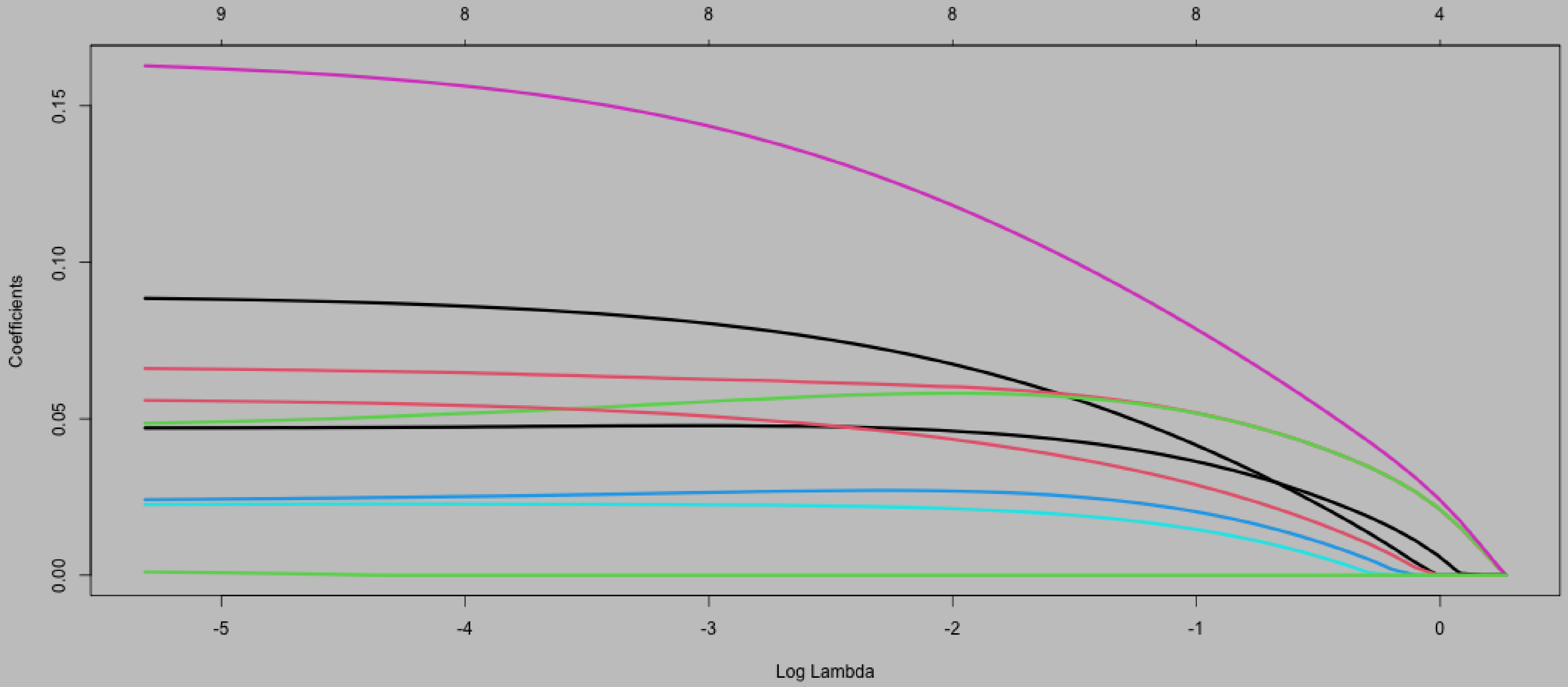


Elastic net

Combine the sparsity from lasso and the flexibility of ridge regression.

$$Z_n(\beta) = \|(Y - X\beta)\|_2^2 + \alpha\lambda_N\|\beta\|_1 + (1-\alpha)\lambda_N\|\beta\|_1^2$$

```r
res <- glmnet(predictors, y, alpha = .3)
plot(res, lwd=3, xvar="lambda")
```



Regularization and statistical inference

```r
library("selectiveInference")
res <- glmnet(predictors, y, family="binomial")
lambda <- exp(-4.5)
beta <- coef(res, x=predictors, y=y, s=lambda/683, exact=T)
res <- fixedLassoInf(predictors, y, beta, lambda,
                     family="binomial", alpha=0.05)
```

```
Warning in fixedLogitLassoInf(x, y, beta, lambda, alpha
= alpha, type = type, : Solution beta does not satisfy
the KKT conditions (to within specified tolerances)
```

```r
res
```

```

Call:
fixedLassoInf(x = predictors, y = y, beta = beta, lambda = lambda,
    family = "binomial", alpha = 0.05)

Testing results at lambda = 0.011, with alpha = 0.050

 Var   Coef Z-score P-value LowConfPt UpConfPt
   1  1.509   3.770   0.596    -9.749    2.005
   2 -0.019  -0.030   0.989     0.857      Inf
   3  0.964   1.399   0.962      -Inf    0.570
   4  0.947   2.680   0.808   -17.165    1.237
   5  0.215   0.617   0.863   -10.079    0.635
   6  1.396   4.084   0.000     0.743    3.114
   7  1.095   2.611   0.749   -14.329    1.537
   8  0.650   1.889   0.805   -12.461    1.008
   9  0.927   1.629   0.705   -10.952    1.724
 LowTailArea UpTailArea
       0.025      0.024
```

Adaptive lasso

Lasso *hates* paying the penalty. Collinearity typically results in only one "representative" *Adaptive lasso* addresses this (and the lacking oracle property) by considering

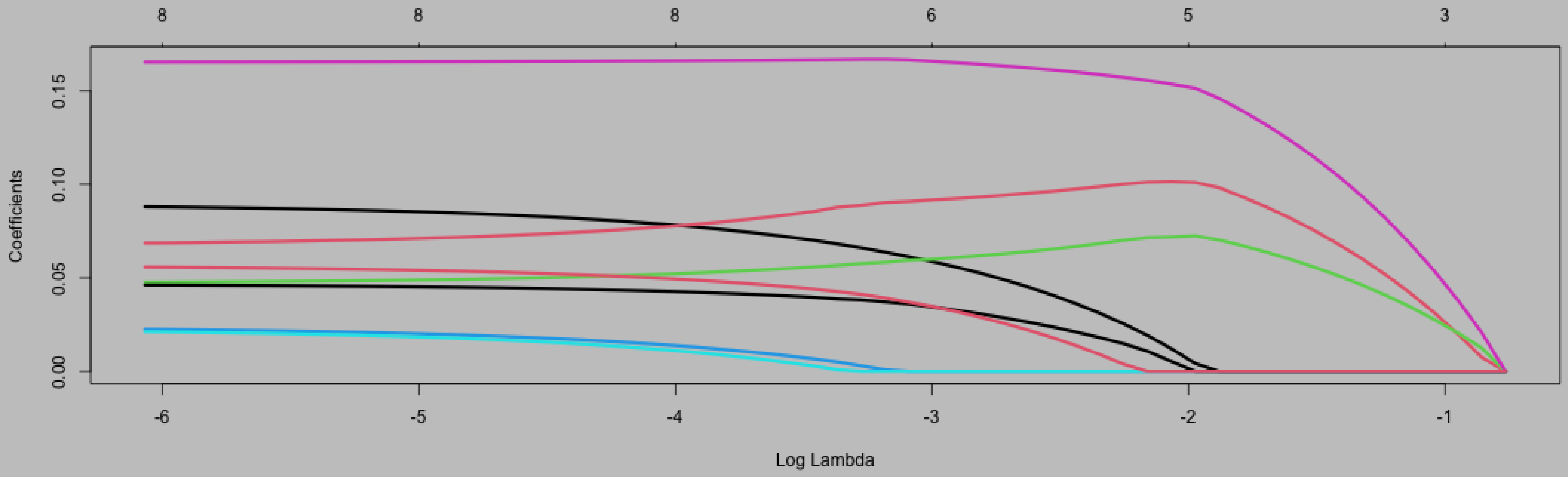
$$Z_n(\beta) = \|(Y - X\beta)\|_2^2 + \lambda_N \sum_{j=1}^{P} \widehat{\omega}_j |\beta|$$

with

$$\widehat{\omega}_j = \frac{1}{|\widehat{\beta_j}^{\text{OLS}}|}$$

Adaptive lasso

```
library("MESS")
weights <- adaptive.weights(predictors, y, weight.method = "univariate
res <- glmnet(predictors, y, penalty.factor = weights$weights)
plot(res, lwd=3, xvar="lambda")
```



Group lasso

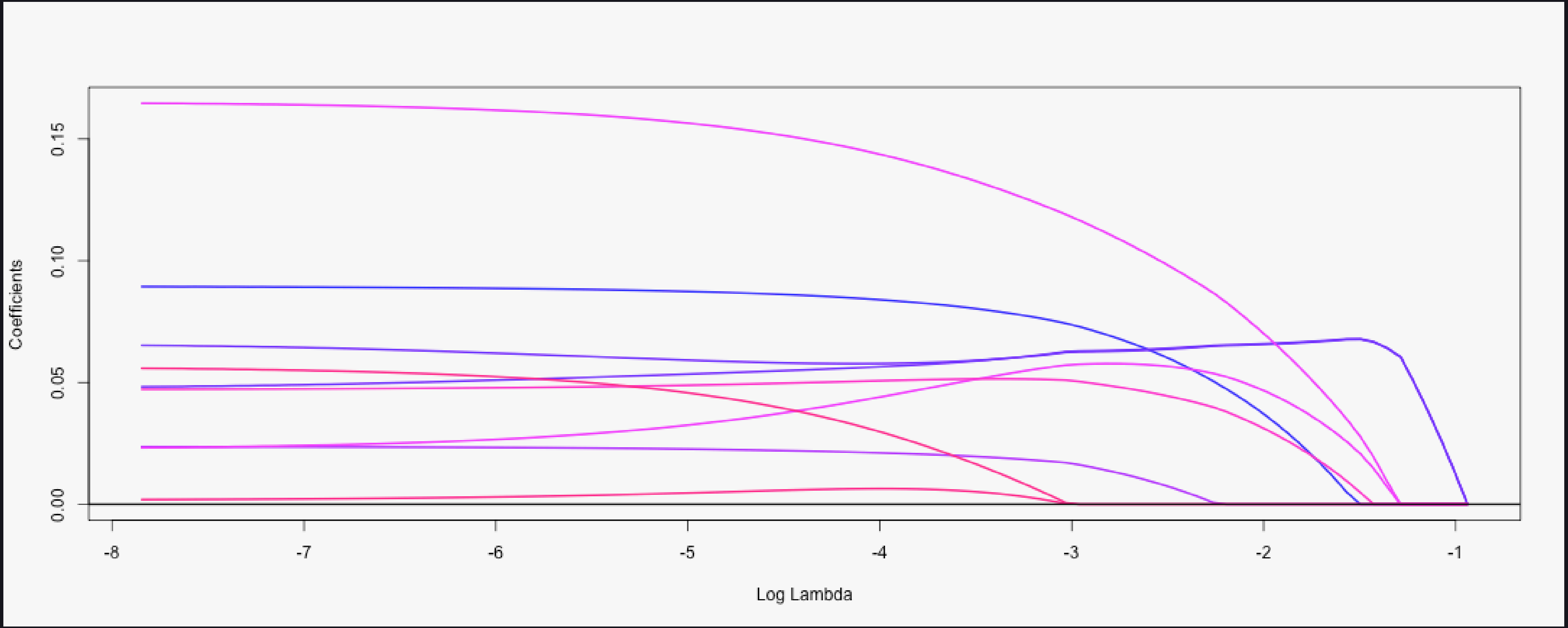
The *group lasso* considers blocks of parameters and places the penalty on each block - either all parameters are non-zero or all zero.

$$Z_n(\beta) = \frac{1}{n}(Y - X\beta)^t(Y - X\beta) + \lambda_n \sum_j \|\theta_j\|$$

where θ_j represents a group of p_j regression coefficients.

Use `grpreg` package with a factor to represent groups (of columns)

```r
library("gglasso")
# Individual groups - sequential
v.group <- c(1, rep(2, 2), 3, rep(4, 2), 5, rep(6, 2))
res <- gglasso(predictors, y, group = v.group, loss="ls") ; plot(res)
```



Exercises!

Highly adaptive lasso

Until now: all our predictors have had a linear relationship to the outcome (by choice).

Idea: any function can be approximated (from data) by a linear combination of spline basis functions.

If we can remove the linearity assumption we can get more flexible (so better) predictions.

Can be used as a smoother - even in low dimensions

Highly adaptive lasso

```r
library(hal9001)
hal_fit <- fit_hal(predictors, y)
preds_hal <- predict(object = hal_fit, new_data = genes)
# MSPE
```

Arguments:

- `max_degree` determines the order of interactions
- `smoothness_orders` - order of smoothness of the regression function. 1 = piecewise linear
- `num_knots` Number of knots for the spline basis functions