

Penalized regression

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



Today's topics

- Penalized regression
- Variants and extensions to penalized regression

Handle complicated modeling aspects with many potential predictors.

Multiple regression

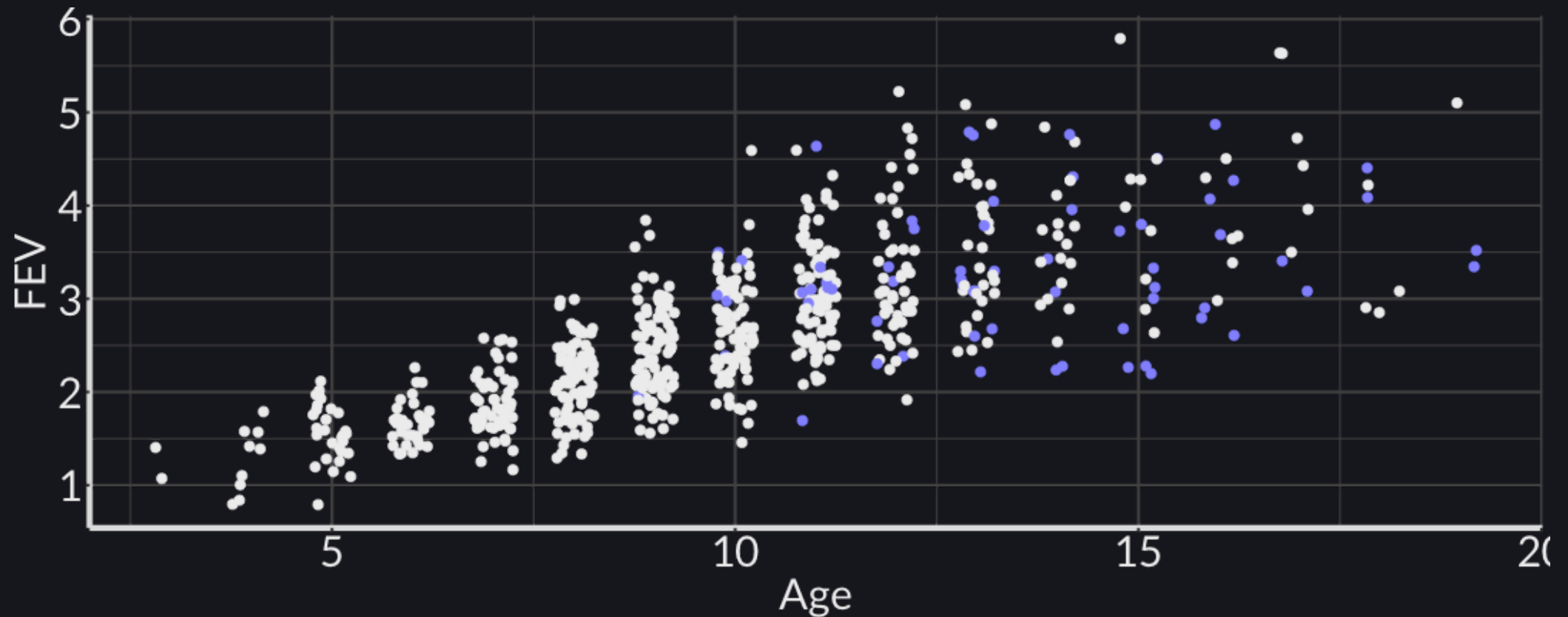
For the i 'th individual,

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

- Each coefficient, β_j , expresses the expected change in y when x_j changes one unit and the remaining explanatory variables are fixed.
- β_j measures the effect of x_j on the response taking the effect of the other variables into account.
- Might 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.

Example - FEV

654 children. Does smoking influence lung capacity?



Example - FEV

```
lm(FEV ~ Smoke + Age + Ht + Gender, data=fev) |> broom::tidy()
```

```
# A tibble: 5 × 5
```

	term	estimate	std.error	statistic	p.value
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	(Intercept)	-4.45	0.224	-19.9	3.53e-69
2	SmokeYes	-0.0898	0.0594	-1.51	1.31e- 1
3	Age	0.0661	0.00950	6.95	8.69e-12
4	Ht	4.10	0.188	21.8	1.89e-79
5	GenderBoy	0.156	0.0333	4.68	3.46e- 6

Generalized linear models

Expand the multiple regression model by applying a transformation with a suitable function. This gives a so-called *generalized* linear model

$$g(\mathbb{E}[Y_i | X_{1i}, \dots, X_{pi}]) = \beta_0 + \beta_1 X_{1i} + \dots + \beta_p X_{pi}$$

or equivalently

$$\mathbb{E}[Y_i | X_{1i}, \dots, X_{pi}] = g^{-1}(\beta_0 + \beta_1 X_{1i} + \dots + \beta_p X_{pi})$$

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

Generalized linear models

Equivalently

$$\mathbb{E}[Y_i | X_{1i}, \dots, X_{pi}] = g^{-1}(X\beta)$$

Only going to consider Gaussian data with identity link:

$$\mathbb{E}[Y_i | X_{1i}, \dots, X_{pi}] = X\beta$$

and

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

OLS estimation

From first year statistics the ordinary least squares estimator is

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

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

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

or "just"

$$\sum_i (y_i - X_i \beta)^2$$

Penalized regression

Let $Y = (Y_1, \dots, Y_n)$ be vector of outcomes in $\mathbb{R}$, and $X = (X_1, \dots, X_n)$ a set of M predictors for each observation.

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

The Lasso estimates β by minimizing the penalized LS function

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

so

$$\hat{\beta} = \arg \min_{\beta \in \mathbb{R}^M} Z_n(\beta)$$

Properties of regularized regression

Even for $M > n$ or $M \gg n$ 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: depression

- Outcome: depression (scale 0-40).
- Input: 384 normalized genes expression value for 40 persons

	y	x1	x2	x3	x4	x5	x6
[1,]	22	-0.63	-0.16	-0.57	-0.51	0.43	0.41
[2,]	15	0.18	-0.25	-0.14	1.34	-0.24	1.69
[3,]	21	-0.84	0.70	1.18	-0.21	1.06	1.59
[4,]	27	1.60	0.56	-1.52	-0.18	0.89	-0.33
[5,]	12	0.33	-0.69	0.59	-0.10	-0.62	-2.29

Example - 384 linear regressions

```
pvals <- pt(
  -abs(MESS::mfastLmCpp(y, x)$tstat)
  , df=38)

head(sort(pvals))
```

```
[1] 0.0003440182 0.0007709503 0.0013871463 0.0014807050
[5] 0.0018858497 0.0026072106
```

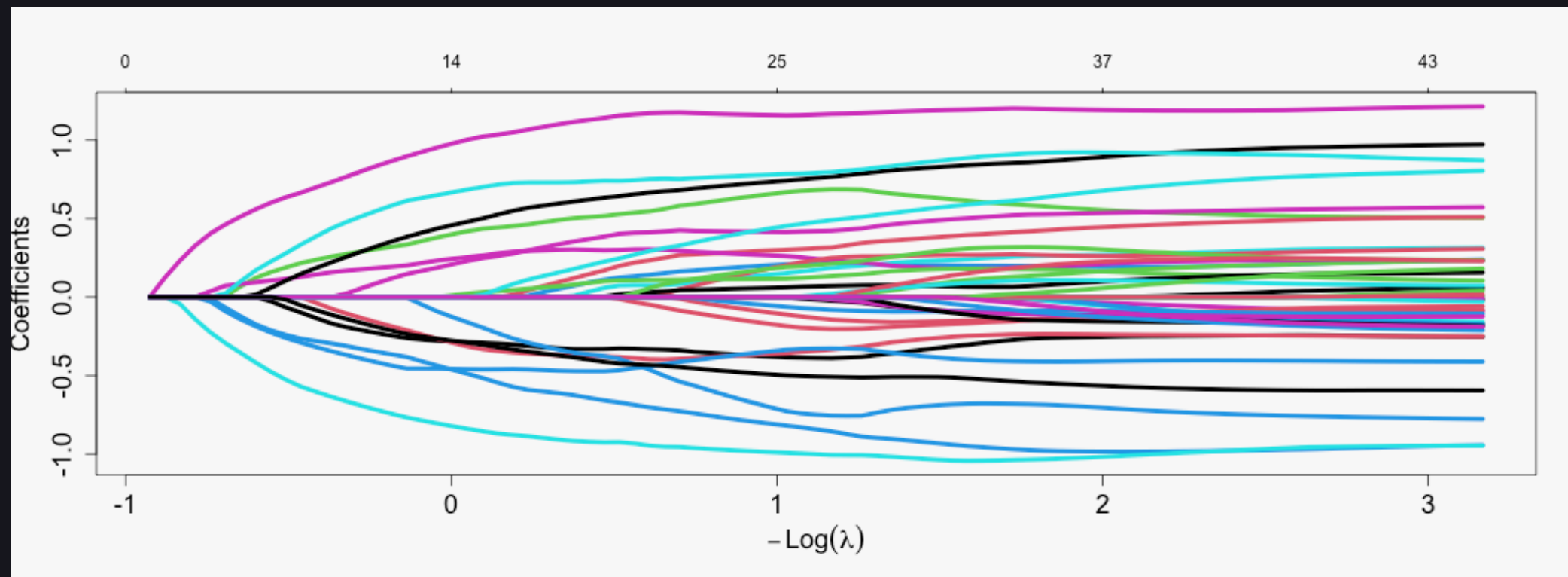
Bonferroni, Holm and FDR "kills" all but variable 264. Not significant.

```
384*head(sort(pvals))
```

```
[1] 0.1321030 0.2960449 0.5326642 0.5685907 0.7241663
[6] 1.0011689
```

Lasso

```
library("glmnet")  
res <- glmnet(x, y) ; plot(res, xvar="lambda", lwd=4, cex.axis=1.5, ce
```



Results

```
lambda <- exp(0.5) ; coef(res, s=lambda)
```

385 x 1 sparse Matrix of class "dgCMatrix"

s=1.648721

(Intercept) 1.687416e+01

V1 .

V2 .

V3 .

V4 .

V5 .

V6 .

V7 .

V8 .

V9 .

V10 .

V11 .

V12 .

```
nonzero <- which(coef(res, s=lambda) != 0) ; nonzero
```

```
[1] 1 37 40 43 112 116 146 194 229 265 297 381
```

```
cbind(nonzero, coef(res, s=lambda)[nonzero])
```

	nonzero	
[1,]	1	1.687416e+01
[2,]	37	-7.199668e-02
[3,]	40	-3.854751e-05
[4,]	43	1.933381e-01
[5,]	112	-2.462418e-01
[6,]	116	-5.349191e-01
[7,]	146	1.172788e-01
[8,]	194	1.117310e-01
[9,]	229	2.889143e-01
[10,]	265	6.411096e-01
[11,]	297	-2.668680e-01
[12,]	381	-2.295539e-02

Evaluating regularized regression results

How to evaluate the results from regularized regression?

- We get a list of parameters: $\hat{\beta}_{(1)}, \hat{\beta}_{(2)}, \hat{\beta}_{(3)}, \dots, 0, 0, \dots$
- They are all shrunk and biased towards 0.
- How do we know which of them are significant?

Classical approach: Test hypothesis that the j th predictor is significant

$$H_0 : \beta_j = 0$$

Potential problems with multiple testing, selection algorithm, bias, lack of small-sample test statistic, ...

Debiasing

```
selected <- nonzero[-1]-1 ; selected
```

```
[1] 36 39 42 111 115 145 193 228 264 296 380
```

```
broom::tidy(lm(y ~ x[,selected]))
```

```
# A tibble: 12 × 5
```

	term <chr>	estimate <dbl>	std.error <dbl>	statistic <dbl>	p.value <dbl>
1	(Intercept)	16.9	0.343	49.3	9.16e-29
2	x[, selected]1	-0.874	0.353	-2.47	1.98e- 2
3	x[, selected]2	-0.830	0.300	-2.77	9.94e- 3
4	x[, selected]3	0.674	0.402	1.68	1.05e- 1
5	x[, selected]4	-0.595	0.435	-1.37	1.82e- 1
6	x[, selected]5	-1.29	0.362	-3.57	1.30e- 3
7	x[, selected]6	0.455	0.400	1.14	2.66e- 1

Selective Inference

Compute p -values and CIs for the lasso estimate, at a fixed value of the tuning parameter λ .

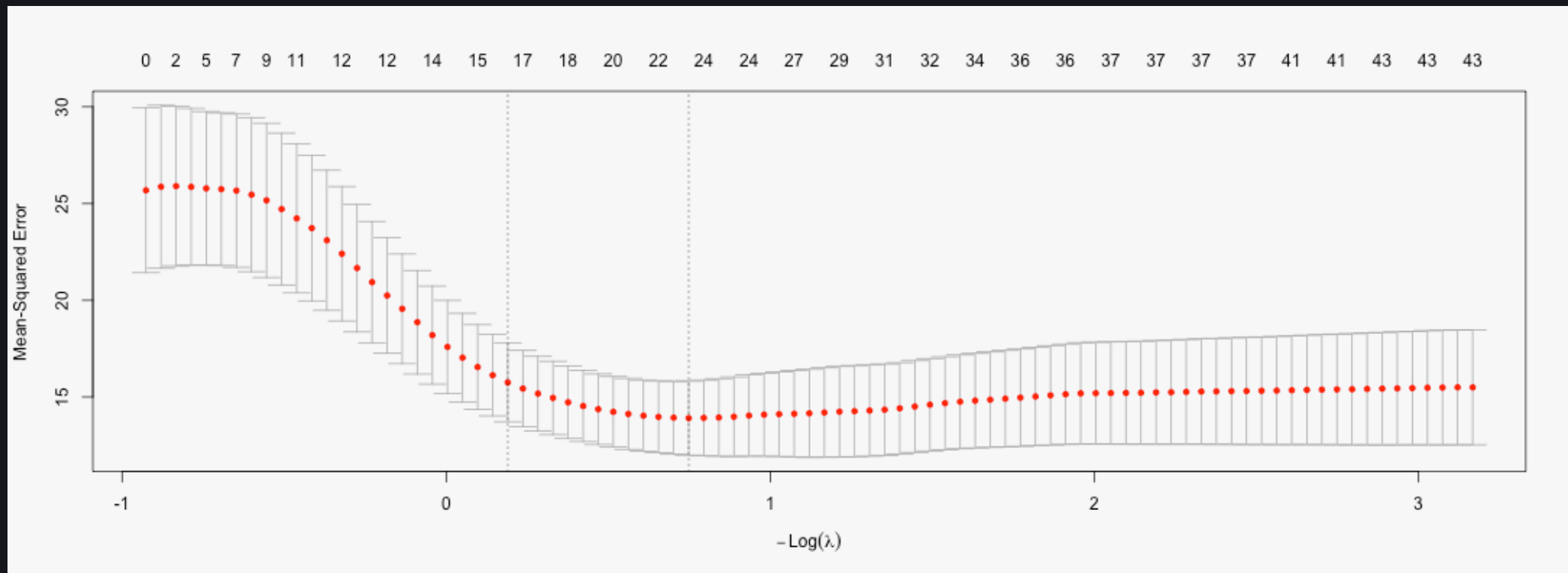
```
library("selectiveInference")
lambda <- 2
n <- 40
beta <- coef(res, s=lambda/n,
              exact=TRUE, x=x, y=y)[-1]
fixedLassoInf(x, y, beta, lambda, sigma=sigma)
```

Caution: quite persnickety

Exercises!

How to choose the penalty?

```
res <- cv.glmnet(x, y) # 10-fold CV  
plot(res, lwd=2)
```



```
res
```

```
Call:  cv.glmnet(x = x, y = y)
```

```
Measure: Mean-Squared Error
```

	Lambda	Index	Measure	SE	Nonzero
min	0.4733	37	13.91	1.913	24
1se	0.8270	25	15.75	2.034	16

Penalized regression - Ridge regression

How about using another penalty function:

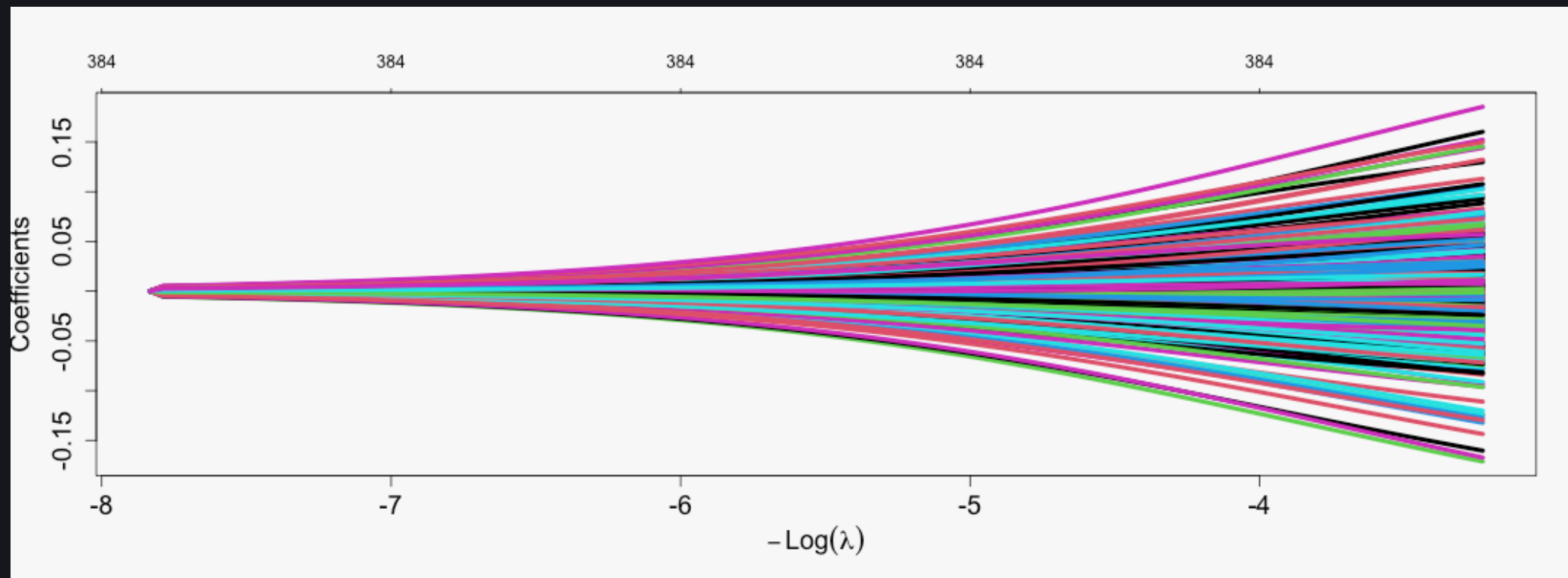
The *Ridge regression* estimates β by minimizing the penalized LS function

$$Z_n(\beta) = \frac{1}{n}(Y - X\beta)^t(Y - X\beta) + \xi_n \|\beta\|^2$$

Ridge regression proceeds by adding a small value, to the diagonal elements of the correlation matrix of the parameters.

Introduces bias but reduces variation.

```
res <- glmnet(x, y, alpha=0)  
plot(res, xvar="lambda", lwd=4, cex.axis=1.5, cex.lab=1.5)
```



Penalized regression - elastic net

How about using yet another penalty function:

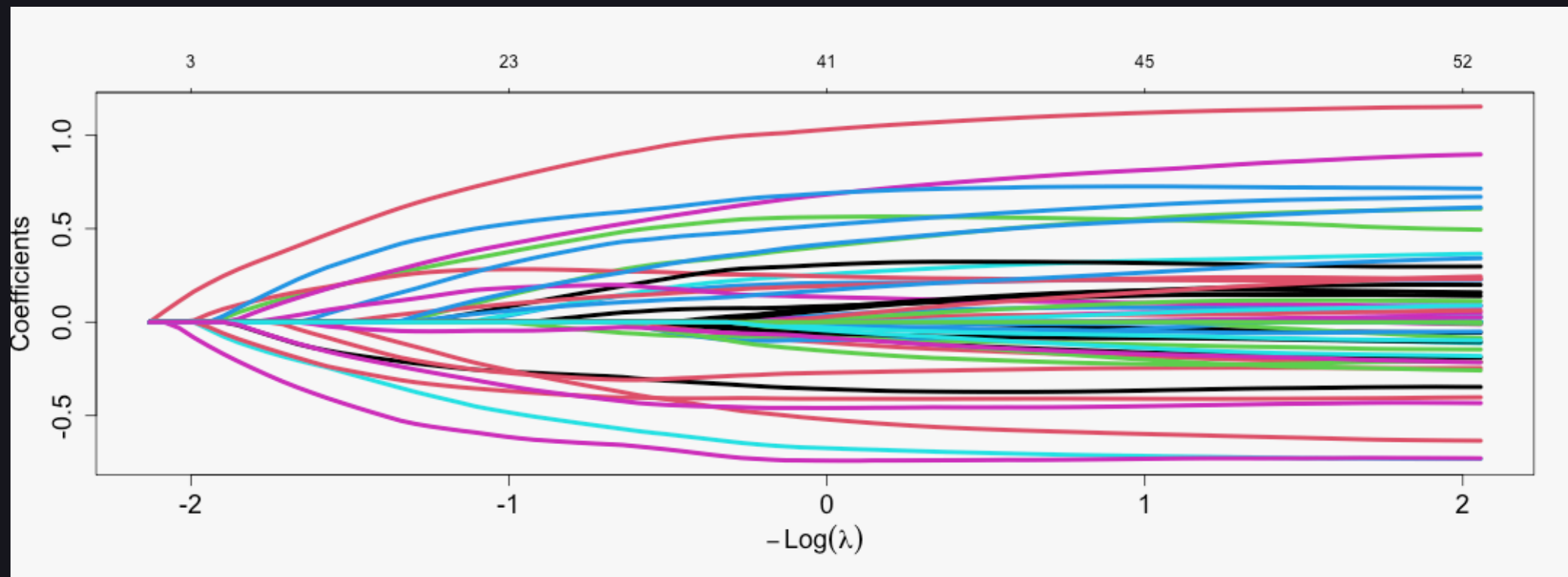
The elastic net regression estimates β by minimizing the penalized LS function

$$Z_n(\beta) = \frac{1}{n}(Y - X\beta)^t(Y - X\beta) + \lambda_n\|\beta\| + \xi_n\|\beta\|^2$$

has Lasso and ridge regression as special cases.

Estimated using an iterative two-step procedure.


```
res <- glmnet(x, y, alpha=.3)
plot(res, xvar="lambda", lwd=4, cex.axis=1.5, cex.lab=1.5)
```



Exercises!

The adaptive lasso

The *adaptive lasso* uses weighted coefficient penalties to obtain better recoverage of the model (the oracle property)

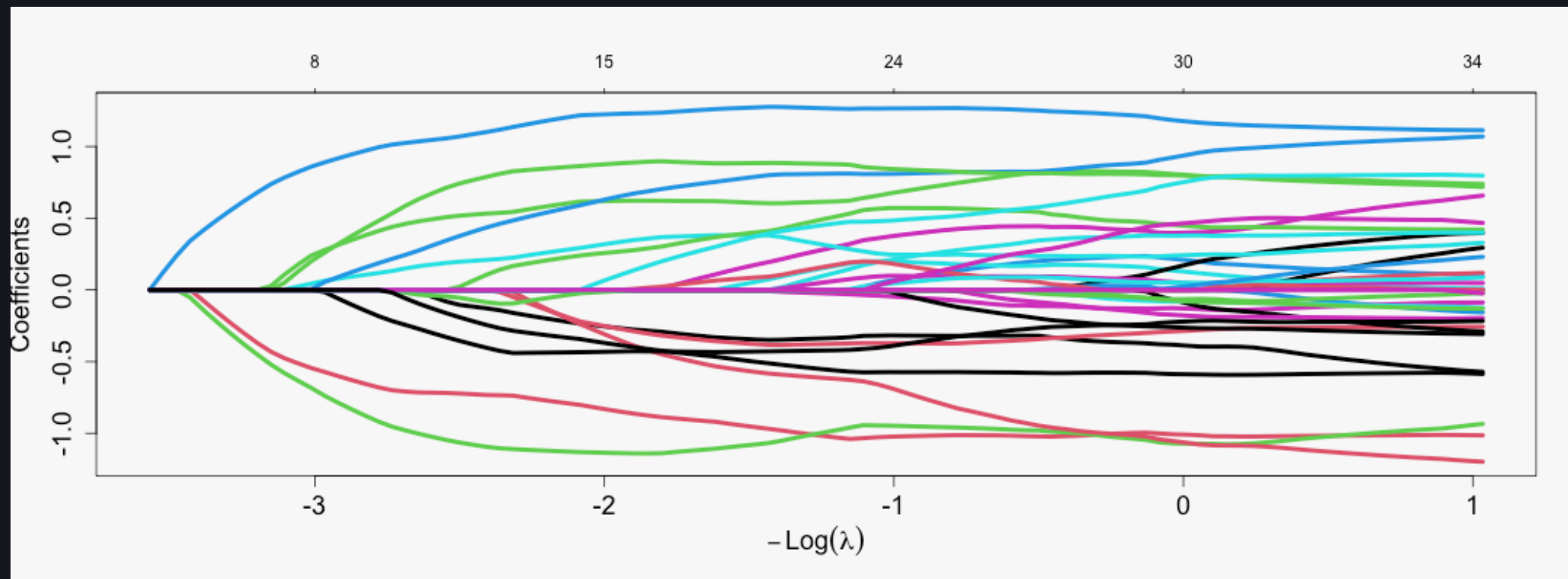
$$Z_n(\beta) = \frac{1}{n}(Y - X\beta)^t(Y - X\beta) + \lambda_n \|\omega\beta\|$$

with $\omega_j = 1/|\hat{\beta}_j|$.

Can use

- LS estimates when $p < N$
- Univariate estimates when $p \gg N$

```
univariate_coefs <- coef(MESS::mfastLmCpp(y, x))  
res <- glmnet(x, y, penalty.factor = 1/abs(univariate_coefs))  
plot(res, xvar="lambda", lwd=4, cex.axis=1.5, cex.lab=1.5)
```



Group lasso

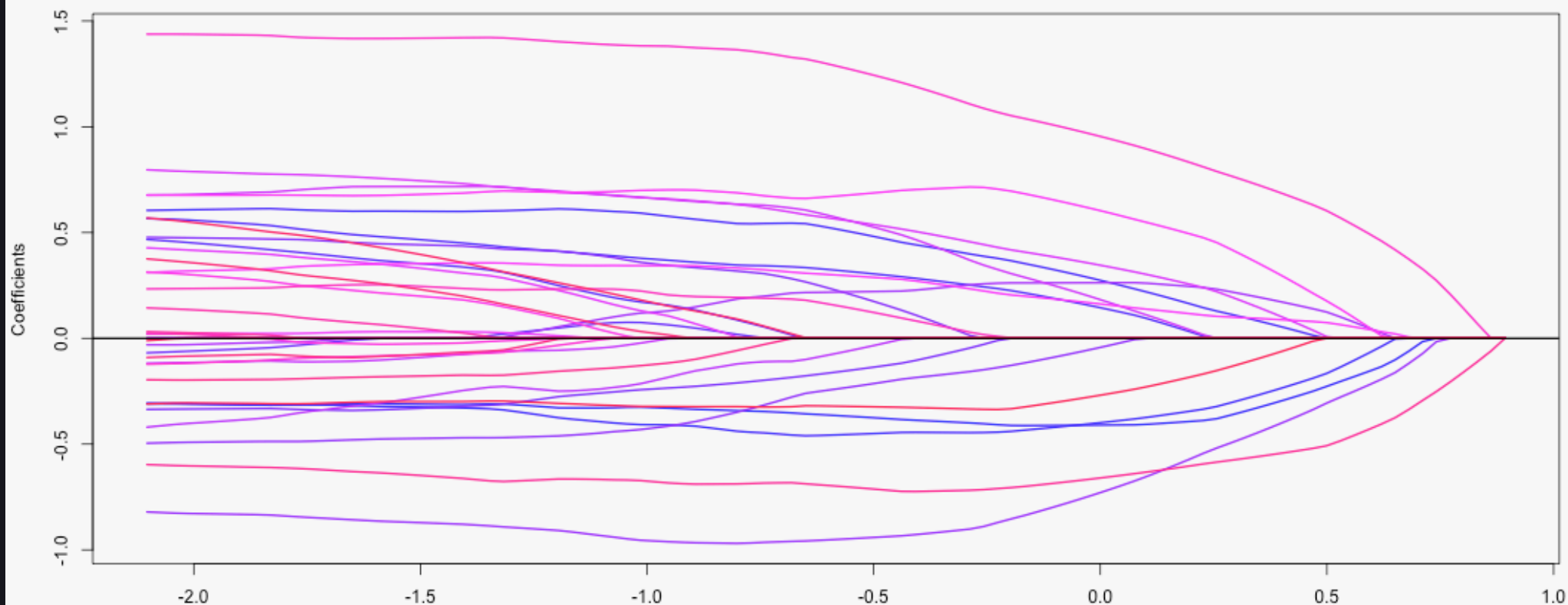
The *group lasso* considers blocks of parameters and places the penalty on each block - either all parameters are non-zero or all zero.

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

where θ_j represents a group of p_j regression coefficients.

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

```
library("gglasso")  
# Individual groups - sequential  
v.group <- c(rep(1, 10), rep(2, 8), rep(3, 6), seq_len(360) + 3)  
res <- gglasso(x, y, group = v.group, loss="ls") ; plot(res)
```



Exercises!

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

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

with

$$\hat{\omega}_j = \frac{1}{|\hat{\beta}_j^{\text{OLS}}|}$$

Adaptive lasso

```
library("MESS")  
weights <- adaptive.weights(x, y, weight.method = "univariate")
```

Warning in adaptive.weights(x, y, weight.method =
"univariate"): using univariate weight method since $p > n$

```
res <- glmnet(x, y, penalty.factor = weights$weights)
```

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

```
library(hal9001)
hal_fit <- fit_hal(x, 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