05 - Regression and Regularization

ml4econ, HUJI 2024

Itamar Caspi

June 1, 2024 (updated: 2024-06-10)

Packages and setup

Use the {pacman} package that automatically loads and installs packages if necessary:

```
if (!require("pacman")) install.packages("pacman")

pacman::p_load(
  tidyverse,   # for data wrangling and visualization
  knitr,       # for displaying nice tables
  broom,       # for tidying estimation output
  here,        # for referencing folders and files
  glmnet,      # for estimating lasso and ridge
  gamlr,       # for forward stepwise selection
  pls,         # for estimating PCR and PLS
  elasticnet,  # for estimating PCR and PLS
  ggfortify    # for enhancing data visualization
)
```

Set a theme for `ggplot` (Relevant only for the presentation)

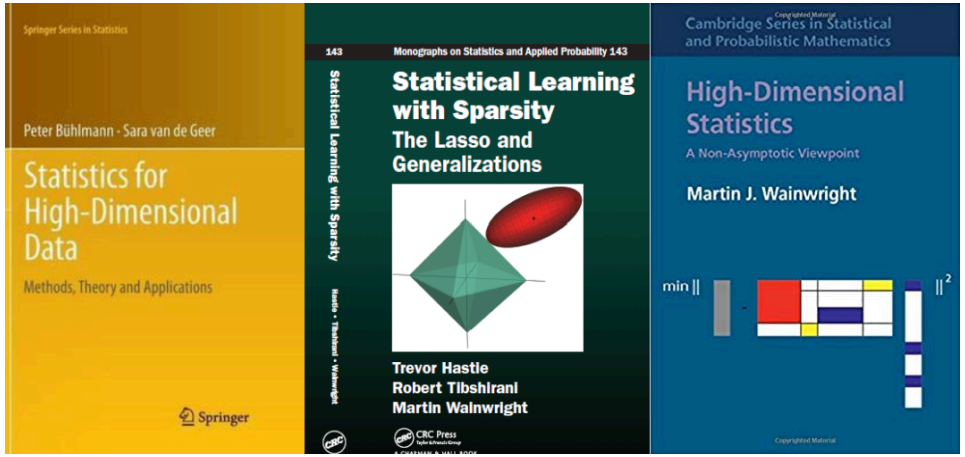
```
theme_set(theme_grey(20))
```

And set a seed for replication

```
set.seed(1203)
```

Resources on hige-dimensional statsitics

- *Statistical Learning with Sparsity - The Lasso and Generalizations* (Hastie, Tibshirani, and Wainwright), (PDF available online)
- *Statistics for High-Dimensional Data - Methods, Theory and Applications* (Buhlmann and van de Geer)
- *High Dimensional Statistics - A Non-Asymptotic Viewpoint* (Wainwright)



Outline

- Linear regression
- Penalized regression
- Subset selection
- Shrinkage
- Dimension Reduction

Linear Regression

Econometrics

In econometrics, we typically assume a "true" linear data generating process (DGP):

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i$$

where y_i is the outcome variable, $x_{ij}, \ldots, x_{ip}$ represent explanatory or control variables (including interactions, polynomials, etc.), and ε_i is the regression error.

Sample: $\{(x_1, \ldots, x_p, y_i)\}_{i=1}^{n}$

Estimation

Ordinary least squares minimizes:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2$$

The emphasis is on in-sample fit (minimizing residual sum of squares).

Typical workflow:

- Impose identifying assumptions (causal claims).
- Estimate $\beta_0, \ldots, \beta_p$ using the entire sample.
- Assume a random sample from a larger population.
- Perform hypothesis testing.

Supervised Learning

Consider the following linear data generating process (DGP):

$$y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + \varepsilon_i$$

where y_i is the predicted (response) variable, $x_{i1}, \ldots, x_{ip}$ are the features, and ε_i is the irreducible error.

- **Training set**: $\{(x_{1i}, \ldots, x_{ip}, y_i)\}_{i=1}^{n}$
- **Test set**: $\{(x_{1i}, \ldots, x_{ip}, y_i)\}_{i=n+1}^{m}$

> Typical assumptions: (1) independent observations; (2) stable DGP across both training and test sets.

Our objective is to predict unseen data, $\hat{y}_i$.

Dffierent objective, different approach

To illustrate how these two approaches (estimation vs. prediction) differ, consider the following data generating process[1]:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \qquad \varepsilon_i \sim N(0, \sigma^2)$$

where $\beta_0 = 0$ and $\beta_1 = 2$.

Next, suppose you get a sample of size n and use it to estimate the model and get (standard errors in parentheses):

$$\begin{underset{}{y_i} = \underset{(0.2)}{0.0} + \underset{(10.0)}{2.0} \times x_i + \hat{\varepsilon}_i}$$

Given a new unseen x^0 and the output above, how would you predict y^0?

[1] Adapted from Susan Athey's lecture.

Standard Errors and Prediction Accuracy

- OLS prioritizes unbiasedness first, followed by efficiency.
- OLS is unbiased, providing accurate predictions on average. But is that our goal?
- A noisy estimated coefficient (high standard error) leads to a variable prediction (high variability).
- The bias-variance trade-off comes into play.
- Prediction involves balancing bias and variance.

Note: In multivariate regression, the complexity increases due to the correlation structure of the x's.

Illustration: Browser Data

In this lecture, we will use Matt Taddy's **browser dataset** (available in our repo), which contains web browsing logs for 10,000 people. Taddy extracted a year's worth of browser logs for the 1,000 most heavily trafficked websites. Each browser in the sample spent at least $1 online during the same year.

The goal of estimation is to predict spending based on browser history:

$$\log(spend_i) = \beta_0 + \beta' visits_i + \varepsilon_i, \qquad \text{for } i = 1, \ldots, n$$

where $visits_i$ is a vector of site-visit percentages. This model can be used to segment expected user budget as a function of browser history.

Loading Data

Specifically, we will use a sample from the browser dataset, consisting of 250 websites and 1,000 users.

```
browser <- here("05-regression-regularization/data", "browser-all.csv") %>%
  read_csv()
```

The log spending by the first user and the fraction of times she visited the first 4 websites are as follows:

```
browser[6, 1:5]
```

```
## # A tibble: 1 x 5
##   log_spend `123greetings.com` `204.95.60.12` `2o7.net` `65.115.67.11`
##       <dbl>              <dbl>          <dbl>     <dbl>          <dbl>
## 1      7.74                  0         0.0483    0.0483         0.0322
```

NOTE: The design matrix in this case is sparse.

Transforming Data to Matrices

It is useful to transform the data into response and feature vectors/matrices:

```
browser_mat <- browser %>%
  as.matrix()

Y_browser <- browser_mat[, 1]     # response
X_browser <- browser_mat[, 2:201] # features
```

OLS Results

Estimate the model using `lm()`:

```
lm_fit <- lm(log_spend ~ ., data = browser)
```

Display the estimation output, sorted by p-values:

```
lm_fit %>%
  tidy() %>%
  arrange(p.value) %>%
  head(3) %>%
  kable(format = "html", digits = 2)
```

term	estimate	std.error	statistic	p.value
$bizrate.com - o01$	1.86	0.25	7.54	0
staples.com	1.33	0.22	5.91	0
victoriassecret.com	1.45	0.25	5.91	0

Model Performance

What is the training-set MSE?

```
lm_fit %>%
  augment() %>%
  summarise(mse = mean((log_spend - .fitted)^2)) %>%
  kable(format = "html", digits = 3)
```

mse
2.075

This is clearly an *underestimate* of the test set MSE.

Penalized Linear Regression

Estimation with Penalization

Penalized (or regularized) sum of squares solves:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \text{ subject to } R(\beta) \leq t$$

where $R(\cdot)$ is a penalty function that measures the **expressiveness** of the model.

As the number of features grows, linear models become *more* expressive.

Notation: Norms

Suppose $\boldsymbol{\beta}$ is a $p \times 1$ vector with a typical element β_i.

- The ℓ_0-norm is defined as $||\boldsymbol{\beta}||_0 = \sum_{j=1}^{p} \mathbf{1}_{\{\beta_j \neq 0\}}$, i.e., the number of non-zero elements in $\boldsymbol{\beta}$.
- The ℓ_1-norm is defined as $||\boldsymbol{\beta}||_1 = \sum_{j=1}^{p} |\beta_j|$.
- The ℓ_2-norm is defined as $||\boldsymbol{\beta}||_2 = \left(\sum_{j=1}^{p} |\beta_j|^2 \right)^{\frac{1}{2}}$, i.e., Euclidean norm.

Commonly Used Penalty Functions

It is often convenient to rewrite the regularization problem in the Lagrangian form:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda R(\beta)$$

Note: There is a one-to-one correspondence between λ and t.

Method	$R(\boldsymbol{\beta})$
OLS	0
Subset selection	$\|\boldsymbol{\beta}\|_0$
Lasso	$\|\boldsymbol{\beta}\|_1$
Ridge	$\|\boldsymbol{\beta}\|_2^2$
Elastic Net*	$\alpha\|\boldsymbol{\beta}\|_1 + (1-\alpha)\|\boldsymbol{\beta}\|_2^2$

[*] Elastic Net will not be covered in this lecture. It is essentially a combination of ridge and lasso.

Best Subset Selection

Our Goal

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \text{ subject to } \|\boldsymbol{\beta}\|_0 \leq t$$

In words: Select the best model according to some statistical criteria, among all possible combinations of t features or fewer.

Best Subset Selection Algorithm

1. For $k = 0, 1, \ldots, p$

   1.1 Fit all models that contain exactly k predictors. If $k = 0$, the forecast is the unconditional mean.

   1.2 Pick the best (e.g., highest R^2) among these models, and denote it by $\mathcal{M}_k$.

2. Optimize over $\{\mathcal{M}_0, \ldots, \mathcal{M}_p\}$ using cross-validation (or other criteria)

Issues:

1. The algorithm is very slow: at each step, we deal with $\binom{p}{k}$ models ("N-P complete").
2. The prediction is highly unstable: the subsets of variables in $\mathcal{M}_{10}$ and $\mathcal{M}_{11}$ can be very different from each other, leading to high variance (the best subset of $\mathcal{M}_3$ need not include any of the variables in the best subset of $\mathcal{M}_2$).

Faster Subset Selection Algorithms

Instead of estimating all possible combinations, follow a particular path of models:

- Forward stepwise selection: Start simple and expand (feasible even if $p > n$)
- Backward stepwise selection: Start with the full model and drop features (not recommended)

Forward Stepwise Algorithm

1. Let $\mathcal{M}_0$ denote the null model, which contains just an intercept.

2. For $k = 0, \ldots, p-1$:

   2.1 Consider all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.

   2.2 Choose the best among these $p - k$ models and call it $\mathcal{M}_{k+1}$. Here, "best" is defined as having the highest R^2

3. Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_p$ using cross-validation.

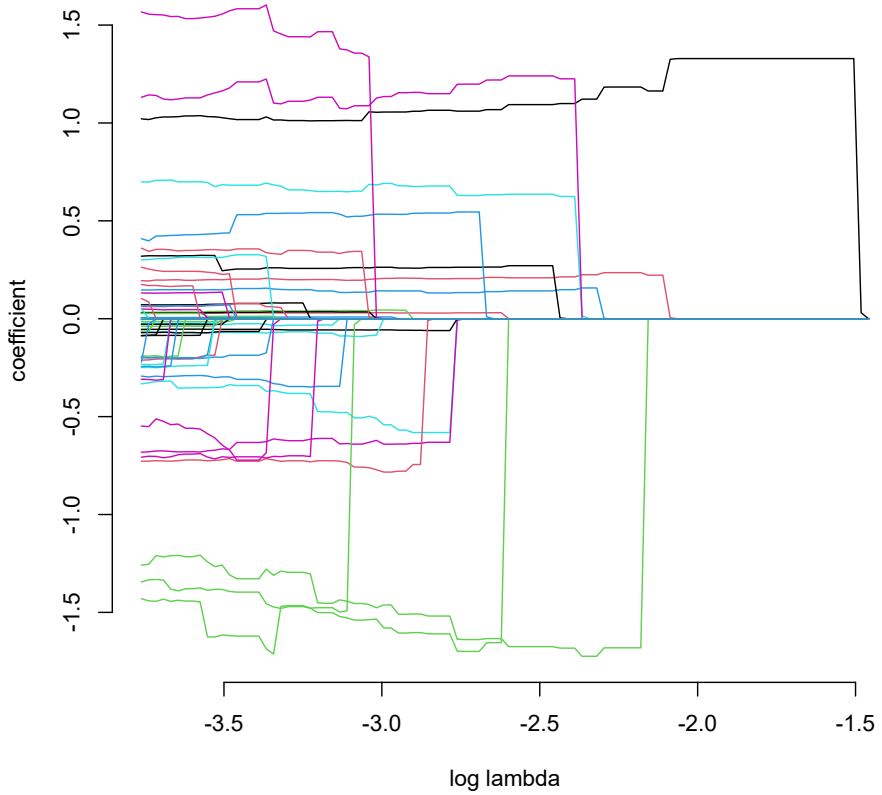
> This is our first example of a **greedy algorithm**: making the locally optimal selection at each stage with the intent of finding a global optimum.

Stepwise Using `gamlr`

{gamlr} is an R package that enables you, among other things, to estimate a forward stepwise regression model.

```
fit_step <- gamlr(X_browser, Y_browser, gamma=Inf, l
plot(fit_step, df=FALSE, select=FALSE)
```

The figure on the right shows the value of the coefficients along the forward stepwise selection path. Notice how jagged the solution paths are. This discontinuity is the cause for instability in subset selection algorithms.



Shrinkage

Prerequisite: Centering and Scaling

In what follows, we assume that each feature is centered and scaled to have mean zero and unit variance:

$$\frac{x_{ij} - \widehat{\mu}_i}{\widehat{\sigma}_i}, \qquad \text{for } j = 1, 2, \ldots, p$$

where $\widehat{\mu}_i$ and $\widehat{\sigma}_i$ are the estimated mean and standard deviation of x_i estimated over the *training* set.

NOTE: Centering and scaling are not important when using OLS. Can you think of why?

Ridge Regression

Ridge regression was introduced into the statistics literature by Hoerl and Kennard (1970).

The optimization problem:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

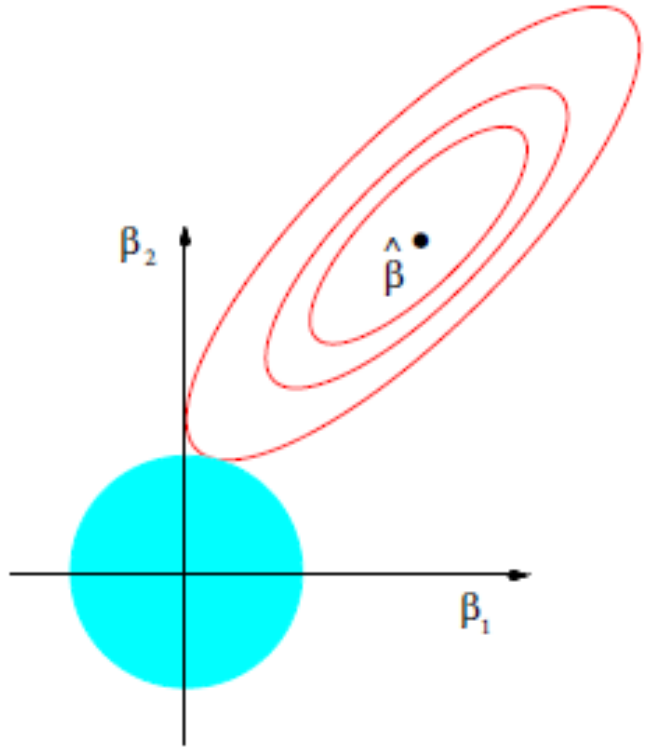
or in a budget constraint form:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 \textbf{ subject to } \|\boldsymbol{\beta}\|_2^2 \leq t$$

Ridge regression places a "budget constraint" on the sum of squared betas. This constraint incorporates the cost of being "too far away" from the null hypothesis of $\beta_j = 0$ (what if this assumption is wrong?).

Illistration of Ridge in 2-D

The solid blue area represents the constraint region, $\beta_1^2 + \beta_2^2 \leq t$, while the red ellipses are the contours of the RSS. $\widehat{\beta}$ is the OLS estimator.



Source: James et al. (2017)

Ridge Regression Solution

The problem in matrix notation:

$$\min_{\boldsymbol{\beta}} \ (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \boldsymbol{\beta}'\boldsymbol{\beta}$$

The ridge estimator is given by:

$$\widehat{\boldsymbol{\beta}}^{\mathrm{R}} = \left(\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}\right)^{-1}\mathbf{X}'\mathbf{y}$$

NOTE: We can have a solution even if $\mathbf{X}$ is not of full rank (e.g., due to multicollinearity) since $\mathbf{X}'\mathbf{X} + \lambda\mathbf{I}$ is non-singular (since $\lambda > 0$.)

Bayesian Interpretation of Ridge Regression

- Consider the regression model:

$$y_i \sim N\left(\mathbf{x}_i'\boldsymbol{\beta}, \sigma^2\right)$$

where we assume that σ is known.

- Suppose we put an independent prior on each β_j:

$$\beta_j \sim N\left(0, \tau^2\right)$$

- The posterior mean for $\boldsymbol{\beta}$ is:

$$\widehat{\boldsymbol{\beta}}_{\text{posterior}} = \left(\mathbf{X}'\mathbf{X} + \frac{\sigma^2}{\tau^2}\mathbf{1}\right)^{-1}\mathbf{X}'\mathbf{y}$$

- Therefore, $\lambda = \frac{\sigma^2}{\tau^2}$.

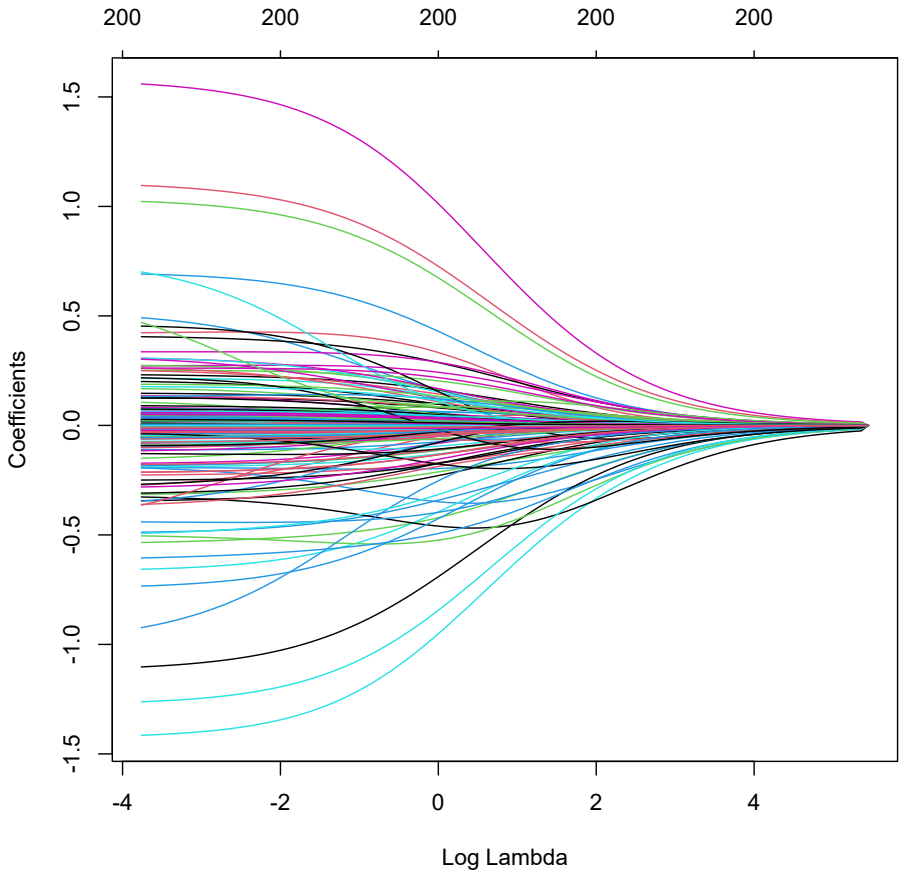
Ridge Regression using `glmnet`

{glmnet} enables you to estimate ridge regression, along with its path for λ.

(Note that we need to set `alpha` to 0.)

```
fit_ridge <- glmnet(
  x = X_browser,
  y = Y_browser,
  alpha = 0
)
plot(fit_ridge, xvar = "lambda")
```

The figure on the right shows the ridge *regularization path*, i.e., the values of the (standardized) coefficients for different values of (log) λ.



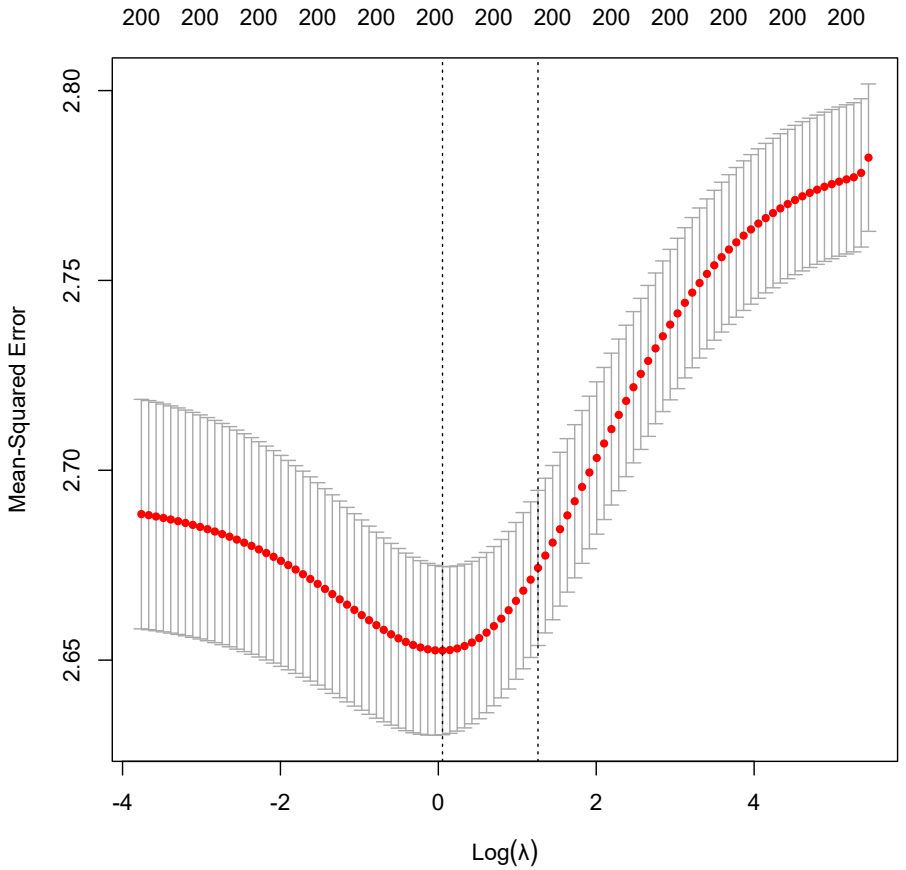
Tuning λ

The `cv.glmnet()` function streamlines cross-validation for ridge regression.

You can modify the default number of folds (10) by adjusting the `nfolds` parameter.

```
cv_ridge <- cv.glmnet(x = X_browser, y = Y_browser,
plot(cv_ridge)
```

- *Left* dotted vertical line: Optimal λ with minimum MSE
- *Right* dotted vertical line: Largest λ with MSE not exceeding one SE from the minimum



Lasso Regression

Lasso (Least Absolute Shrinkage and Selection Operator) was introduced by Tibshirani (1996). The optimization problem is:

$$\min_{\beta_0,\beta} \sum_{i=1}^{N} \left(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \|\boldsymbol{\beta}\|_1$$

Lasso imposes a "budget constraint" on the sum of *absolute* β values.

In contrast to ridge regression, the lasso penalty is linear (shifting from 1 to 2 is equivalent to shifting from 101 to 102).

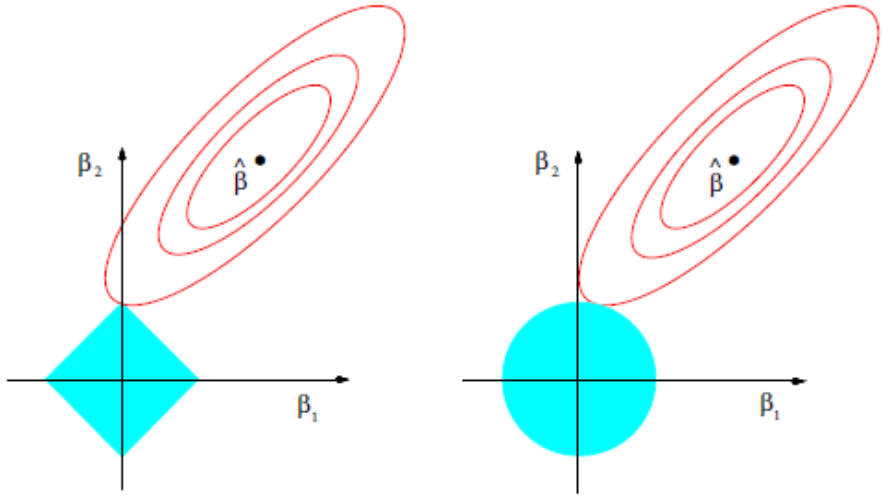
A significant advantage of lasso is that it performs model selection by zeroing out most of the β values in the model (resulting in a *sparse* solution).

Any penalty involving the ℓ_1 norm exhibits this behavior.

Lasso vs. ridge

Contours of error and constraint functions for lasso (left) and ridge (right):

- The solid blue areas represent constraint regions: $\beta_1^2 + \beta_2^2 \leq t$ for ridge and $|\beta_1| + |\beta_2| \leq t$ for lasso.
- The red ellipses depict the contours of the residual sum of squares (RSS).
- $\widehat{\beta}$ denotes the ordinary least squares (OLS) estimator.

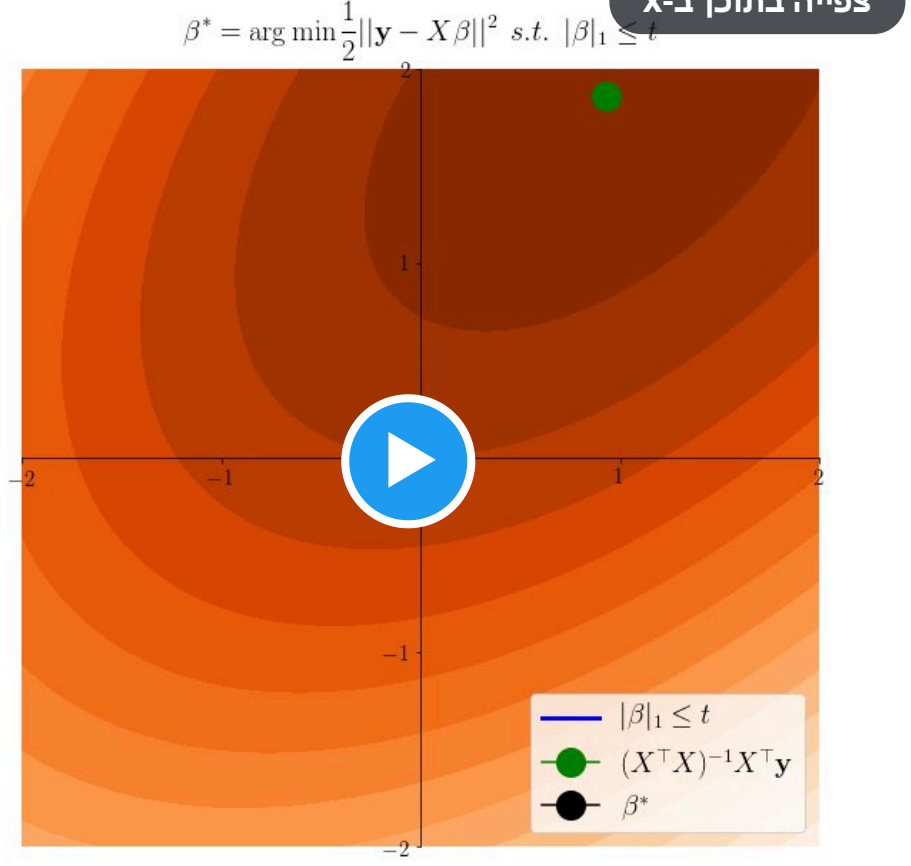


Source: James et al. (2017)

Pierre Ablin
עקוב · @PierreAblin

Illustration of the Lasso and its path in 2D: for t small enough, the solution is sparse!

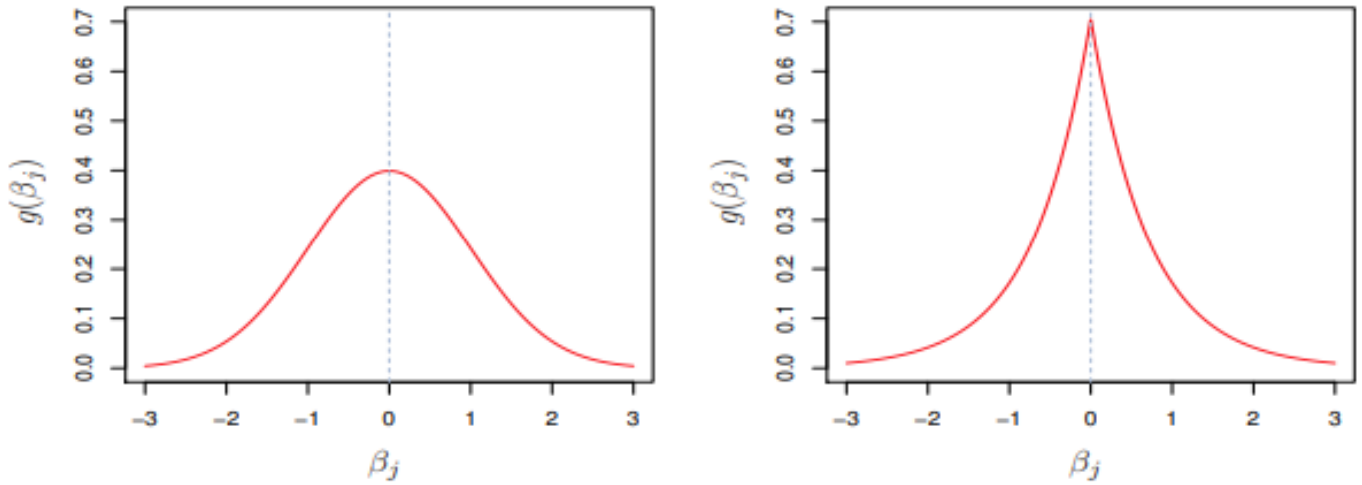
צפייה בתוכן ב-X



Bayesian Interpretation of Lasso

Under lasso, the prior distribution of $\boldsymbol{\beta}$ is the double exponential (Laplace) distribution with density $\frac{1}{2\tau}\exp(-|\boldsymbol{\beta}|/\tau)$, where $\tau = \frac{1}{\lambda}$. The posterior mode corresponds to the lasso solution.

- Left: Normal prior (ridge)
- Right: Laplace prior (lasso)

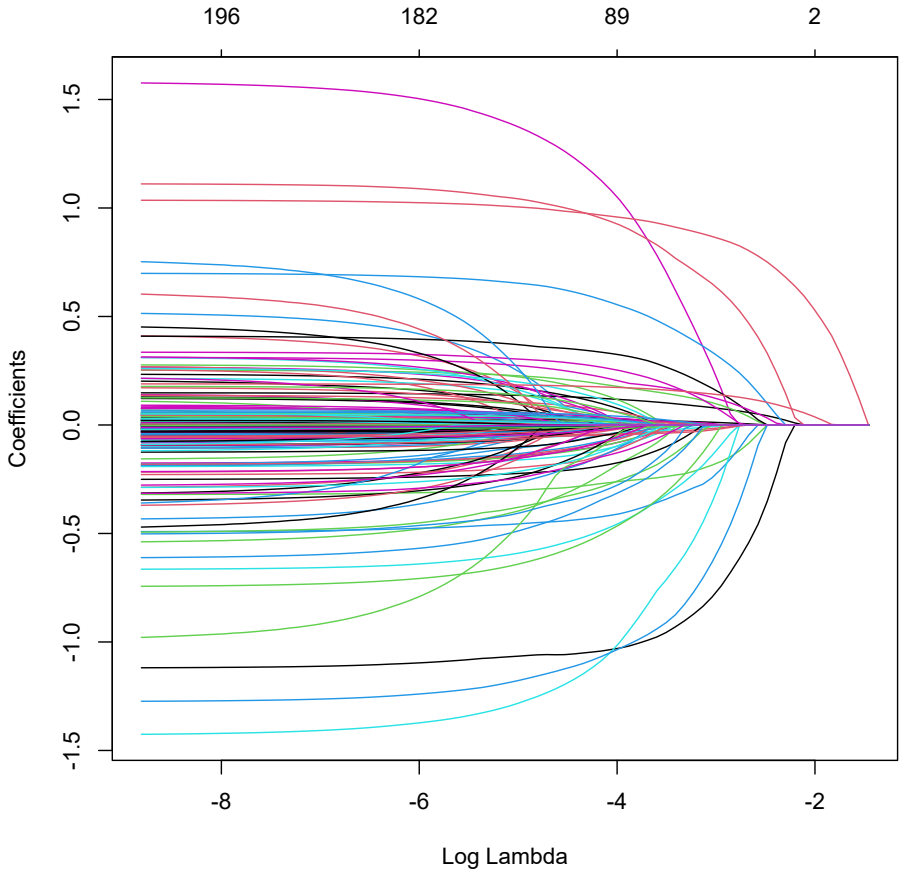


Source: James et al. (2017)

Estimating Lasso with `glmnet`

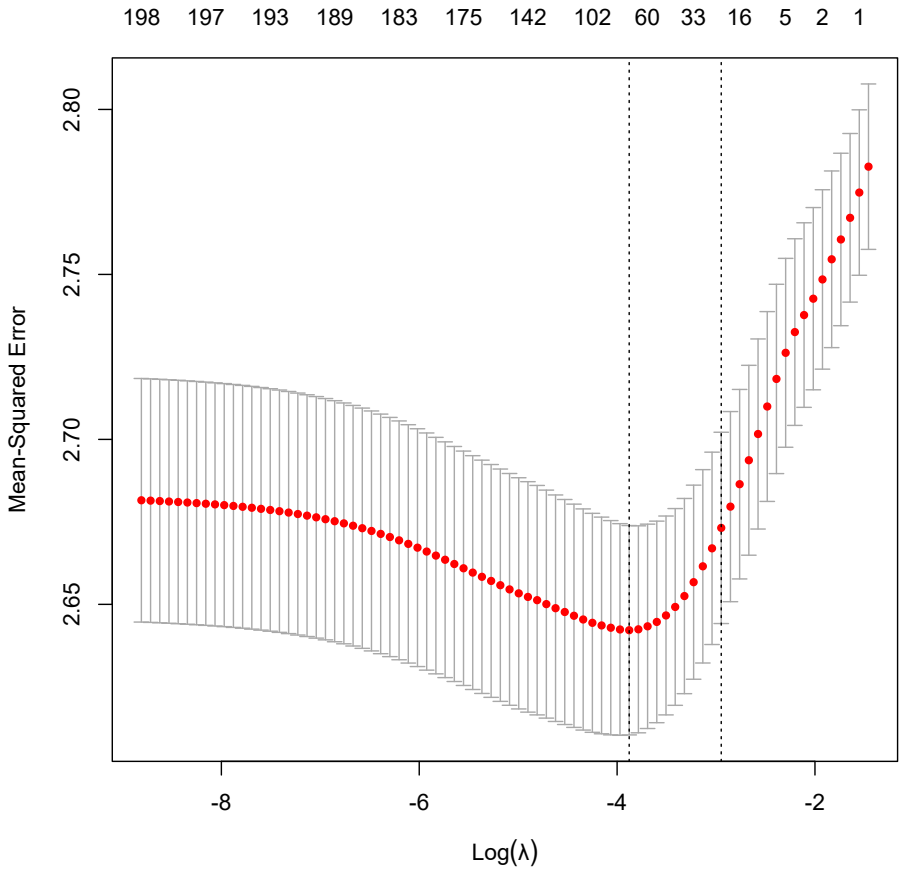
Use the `glmnet()` function with `alpha = 1` (default setting) to estimate the entire lasso regularization path.

```
fit_lasso <- glmnet(
  x = X_browser,
  y = Y_browser,
  alpha = 1
)
plot(fit_lasso, xvar = "lambda")
```



Tuning λ

```
cv_lasso <- cv.glmnet(x = X_browser, y = Y_browser,
plot(cv_lasso, xvar = "lambda")
```



Which features were selected?

Using `s = lambda.min`:

```
#coef(cv_lasso, s = "lambda.min") %>%
 # tidy() %>%
  #as_tibble()
```

Using `s = lamda.1se`:

```
#coef(cv_lasso, s = "lambda.1se") %>%
 # tidy() %>%
  #as_tibble()
```

Understanding Shrinkage

Assume that $n = p$ and $\mathbf{X}$ is an $n \times p$ diagonal matrix with 1's on the diagonal. We are estimating a regression without an intercept.

Under these assumptions, OLS finds $\beta_1, \ldots, \beta_p$ that minimize $\sum_{j=1}^{p} \left(y_j - \beta_j\right)^2$. The solution is $\hat{\beta}_j^{\text{OLS}} = y_j$ (i.e., a perfect fit, $R^2 = 1$).

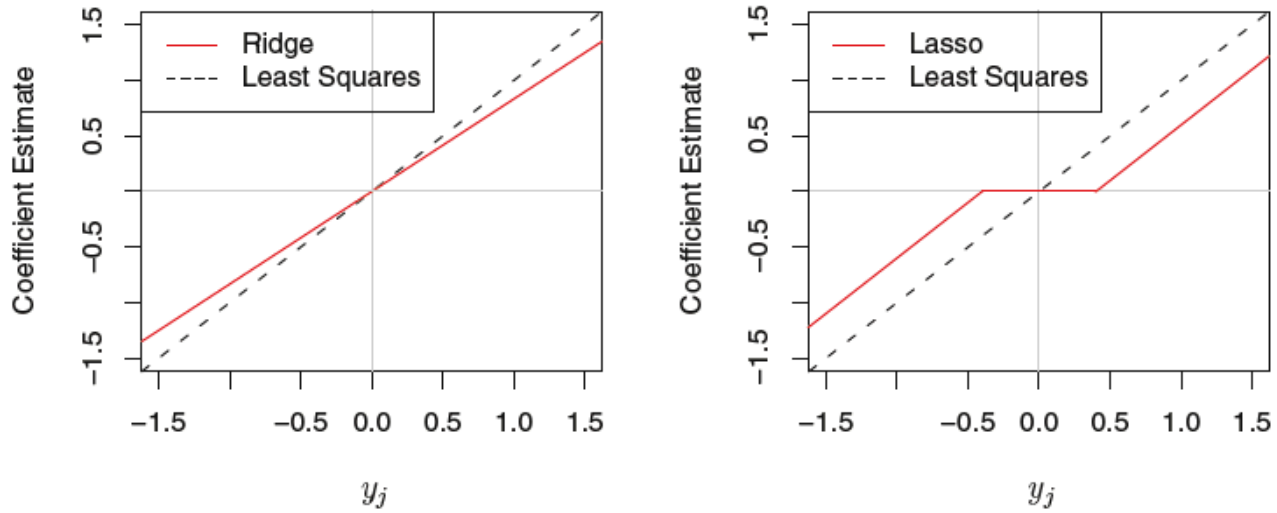
Ridge and lasso estimates take the following forms:

$$\hat{\beta}_j^{Ridge} = \frac{\hat{\beta}^{\text{OLS}}}{(1+\lambda)}, \qquad \hat{\beta}_j^{Lasso} = \begin{cases} \hat{\beta}^{\text{OLS}} - \lambda/2 & \text{if } \hat{\beta}^{\text{OLS}} > \lambda/2 \\ \hat{\beta}^{\text{OLS}} + \lambda/2 & \text{if } \hat{\beta}^{\text{OLS}} < -\lambda/2 \\ 0 & \text{if } \left|\hat{\beta}^{\text{OLS}}\right| \leq \lambda/2 \end{cases}$$

Best-subset selection eliminates all variables with coefficients smaller than the t^{th} largest.

Shrinkage and Implications for Econometric Analysis

Ridge regression shrinks coefficients proportionally, while lasso regression shrinks them by a similar amount, setting sufficiently small coefficients to zero. This type of shrinkage in lasso is referred to as "soft-thresholding."



Source: James et al. (2017)

Main takeaway: Ridge and lasso coefficients should not be directly used in subsequent econometric analysis, unless you adopt a Bayesian approach.

Dimensionality Reduction

Alternative Approach

Main Idea: Instead of selecting and/or shrinking $\boldsymbol{\beta}$, aim to find M linear combinations of the x variables, where $M < p$, and use them to predict y.

Motivating Example

Consider the following regression model:

$$y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i$$

Next, define a new variable, f_i, as a linear combination of x_{i1}, x_{i2}:

$$f_i = \lambda_1 x_{i1} + \lambda_2 x_{i2}$$

for some constants λ_1 and λ_2. This "factor" represents a lower-dimension version of the feature space (one variable instead of two).

According to the dimension reduction approach, it is possible that

$$\hat{y}_i = \hat{\alpha} f_i$$

(estimated by, for example, OLS) could provide better predictions.

Relation to Penalized Regression

Since:

$$\hat{y}_i = \hat{\alpha} f_i = \hat{\alpha}\lambda_1 x_{i1} + \hat{\alpha}\lambda_2 x_{i2},$$

dimension reduction can be viewed as a method that puts a constraint on the estimated β's, i.e.,

$$y_i = \beta_1^* x_{i1} + \beta_2^* x_{i2} + \varepsilon_i$$

where

$$\beta_1^* = \alpha\lambda_1 \qquad \text{and} \qquad \beta_2^* = \alpha\lambda_2$$

The main challenges involved are:

1. How to choose (estimate) λ_1, λ_2?
2. How many factors to use? (In this case, since $p = 2$, M can be at most 2.)

General Case

- Let $f_1, f_2, \ldots, f_p$ represent p linear combinations of the features $x_1, x_2, \ldots, x_p$, such that:

$$f_{im} = \lambda_{m1} x_{1i} + \ldots + \lambda_{mp} x_{ip}, \qquad \text{for } m = 1, \ldots, p$$

- The linear model is given by:

$$y_i = \alpha_0 + \sum_{m=1}^{M} \alpha_m f_{im} + \varepsilon_i, \qquad \text{for } i = 1, \ldots, N$$

  where $M < p$, and the model is estimated by OLS.

- The main challenge lies in estimating and selecting the number of factors in a way that improves prediction accuracy (i.e., reducing variance compared to OLS).

Principal components regression (PCR)

Let $\mathbf{x} = (x_1, \ldots, x_p)$ and $\boldsymbol{\lambda}_1 = (\lambda_{k1}, \ldots, \lambda_{kp})$ denote a vector of features and scalars, respectively. Principal component analysis (PCA) amounts to:

(1) Find a linear combinations $\boldsymbol{\lambda}_1$ of the elements of $\mathbf{x}$ that maximizes its variance.

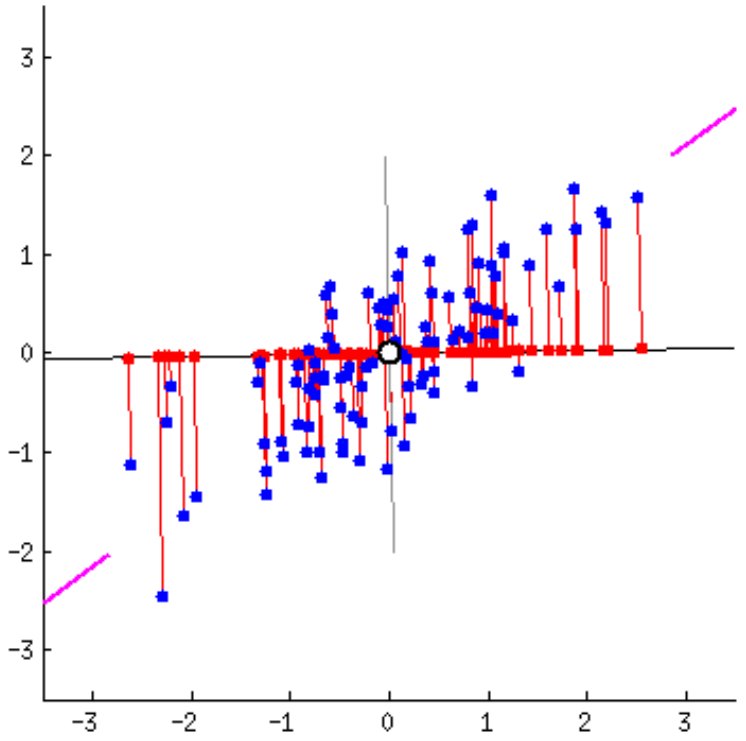
(2) Next, find a linear combination $\boldsymbol{\lambda}_2$ of the elements of $\mathbf{x}$, uncorrelated with $\boldsymbol{\lambda}_1'\mathbf{x}$ having maximum variance.

$\vdots$

(p) Finally, find a linear combination $\boldsymbol{\lambda}_p$ of the elements of $\mathbf{x}$, uncorrelated with $\boldsymbol{\lambda}_1'\mathbf{x}, \ldots, \boldsymbol{\lambda}_{p-1}'\mathbf{x}$ having maximum variance.

> $\boldsymbol{\lambda}_k'\mathbf{x}$ is the k^{th} **principal component** (PC), and $\boldsymbol{\lambda}_k$ is the vector of coefficients or **loadings** for the k^{th} PC.

Illustration in 2-D



Source

Illustration: NBC TV Pilots Data

- To illustrate, we will use Matt Taddy's NBC pilots dataset, which includes survey responses from focus groups on TV show pilots, as well as the first year of ratings results.
- Our objective: Predict viewer interest based on pilot surveys, thus helping studios make better programming decisions.
- The survey data includes 6241 views and 20 questions for 40 shows.
- PE is our outcome of interest. It measures viewer engagement with the show, which is reported on a 0 (no attention) to 100 (fully engaged) scale.

Read shows Dataset

shows includes data on 40 TV shows.

```
shows <- here("05-regression-regularization/data", "nbc_showdetails.csv") %>%
  read_csv()

head(shows)
```

```
## # A tibble: 6 x 6
##   Show                                Network    PE   GRP Genre           Duration
##   <chr>                               <chr>   <dbl> <dbl> <chr>              <dbl>
## 1 Living with Ed                      HGTV     54   151   Reality               30
## 2 Monarch Cove                        LIFE     64.6 376.  Drama/Adventure       60
## 3 Top Chef                            BRAVO    78.6 808.  Reality               60
## 4 Iron Chef America                   FOOD     62.6  17.3 Reality               30
## 5 Trading Spaces: All Stars           TLC      56    44.1 Reality               60
## 6 Lisa Williams: Life Among the Dead LIFE     56.2 383.  Reality               60
```

Read `survey` results dataset

The `survey` dataset includes 6241 views and 20 questions for 40 shows.

```
survey <- here("05-regression-regularization/data", "nbc_pilotsurvey.csv") %>%
  read_csv()

head(survey)
```

```
## # A tibble: 6 x 22
##   Viewer Show  Q1_Attentive Q1_Excited Q1_Happy Q1_Engaged Q1_Curious Q1_Motivated Q1_Comforted Q1_Annoyed Q1_Indifferent
##    <dbl> <chr>        <dbl>      <dbl>    <dbl>      <dbl>      <dbl>        <dbl>        <dbl>      <dbl>           <dbl
## 1     71 Iron~            3          4        4          3          5            4            3          2               2
## 2     71 Trad~            4          4        3          4          5            2            3          3               2
## 3     71 Hous~            4          4        4          5          5            3            3          2               1
## 4     71 What~            4          3        3          3          4            2            2          4               2
## 5     71 Amer~            4          4        3          4          4            4            3          3               4
## 6     73 Next             2          4        2          4          2            3            3          3               3
## # i 11 more variables: Q2_Relatable <dbl>, Q2_Funny <dbl>, Q2_Confusing <dbl>, Q2_Predictable <dbl>,
## #   Q2_Entertaining <dbl>, Q2_Fantasy <dbl>, Q2_Original <dbl>, Q2_Believable <dbl>, Q2_Boring <dbl>, Q2_Dramatic <dbl>,
## #   Q2_Suspenseful <dbl>
```

Aggregating Survey Data

We will now aggregate viewer answers by show using the `mean` function as the aggregating function.

```
survey_mean <- survey %>%
  select(-Viewer) %>%
  group_by(Show) %>%
  summarise_all(list(mean))

head(survey_mean)
```

```
## # A tibble: 6 x 21
##   Show         Q1_Attentive Q1_Excited Q1_Happy Q1_Engaged Q1_Curious Q1_Motivated Q1_Comforted Q1_Annoyed Q1_Indifferent
##   <chr>               <dbl>      <dbl>    <dbl>      <dbl>      <dbl>        <dbl>        <dbl>      <dbl>           <dbl>
## 1 30 Rock              3.69       3.47     3.65       3.74       3.58         2.96         2.97       2.12            2.50
## 2 America's N~         3.84       3.69     3.58       3.84       3.71         3.32         2.95       2.42            2.47
## 3 American Ch~         3.67       3.50     3.46       3.58       3.71         3.01         2.86       2.25            2.43
## 4 Bones                4.12       3.71     3.42       4.06       4.15         3.29         2.95       1.72            2.11
## 5 CSI: Crime ~         4.04       3.70     3.44       3.98       4.05         3.35         2.95       2.09            2.20
## 6 Close to Ho~         3.80       3.32     3.16       3.83       3.73         3.18         2.83       1.93            2.17
## # i 11 more variables: Q2_Relatable <dbl>, Q2_Funny <dbl>, Q2_Confusing <dbl>, Q2_Predictable <dbl>,
## #   Q2_Entertaining <dbl>, Q2_Fantasy <dbl>, Q2_Original <dbl>, Q2_Believable <dbl>, Q2_Boring <dbl>, Q2_Dramatic <dbl>,
## #   Q2_Suspenseful <dbl>
```

Joining shows and survey into a Single Tibble

We will now generate a tibble that holds PE and the mean survey answers, as well as Genre and Show.

```
df_join <- shows %>%
  left_join(survey_mean) %>%
  select(PE, starts_with("Q"), Genre, Show)

head(df_join)
```

```
## # A tibble: 6 x 23
##      PE Q1_Attentive Q1_Excited Q1_Happy Q1_Engaged Q1_Curious Q1_Motivated Q1_Comforted Q1_Annoyed Q1_Indifferent
##   <dbl>        <dbl>      <dbl>    <dbl>      <dbl>      <dbl>        <dbl>        <dbl>      <dbl>          <dbl>
## 1  54           3.89       3.78     3.93       3.87       3.80         3.61         3.57       2.61           3.12
## 2  64.6         4.05       3.86     3.83       3.88       4            3.94         3.91       2.97           3.15
## 3  78.6         3.85       3.60     3.63       3.77       3.86         3.20         3          2.19           2.40
## 4  62.6         3.91       3.69     3.61       3.85       3.94         3.33         3.05       2.12           2.44
## 5  56           3.81       3.54     3.51       3.78       3.91         3.29         2.80       2.10           2.43
## 6  56.2         3.72       3.65     3.55       3.66       3.76         3.57         3.38       2.20           2.58
## # i 13 more variables: Q2_Relatable <dbl>, Q2_Funny <dbl>, Q2_Confusing <dbl>, Q2_Predictable <dbl>,
## #   Q2_Entertaining <dbl>, Q2_Fantasy <dbl>, Q2_Original <dbl>, Q2_Believable <dbl>, Q2_Boring <dbl>, Q2_Dramatic <dbl>,
## #   Q2_Suspenseful <dbl>, Genre <chr>, Show <chr>
```

Setting up the Data for PCR

The features matrix, i.e., the mean survey answers, is given by:

```
X <- df_join %>%
  select(starts_with("Q")) %>%
  as.matrix()
```

Principal components based on `X` (scaled):

```
PC <- X %>%
  prcomp(scale. = TRUE) %>%
  predict()
```

where we've used the `prcomp()` and `predict()` function to extract the PCs. Setting `scale.=TRUE` makes sure that `X` is scaled prior to running PCA.

Response variable (`PE`):

```
Y <- df_join %>%
  select(PE) %>%
  as.matrix()
```

Understanding Principal Components

To better understand PCs, we can examine the loadings matrix (a.k.a, the "rotation matrix.")

First, we can apply `prcomp()` to the feature matrix and store the output in the `pca` object:

```
pca <- X %>% prcomp(scale. = TRUE)
```

Next, we can access the rotation matrix using `$rotation`. In the example below, we'll look at the first two components:

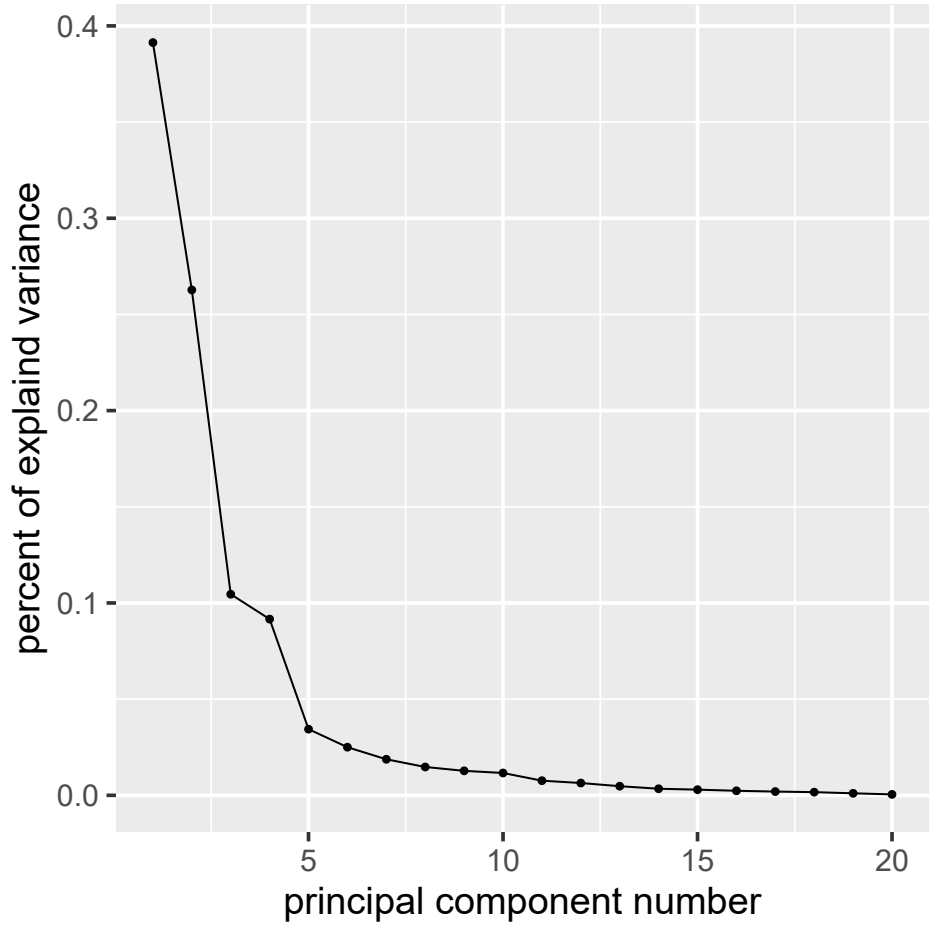
```
pca$rotation[, 1:2] %>% round(1)
```

```
##                    PC1  PC2
## Q1_Attentive      -0.3  0.0
## Q1_Excited        -0.3 -0.1
## Q1_Happy          -0.1 -0.2
## Q1_Engaged        -0.3  0.0
## Q1_Curious        -0.3  0.0
## Q1_Motivated      -0.2 -0.3
## Q1_Comforted      -0.1 -0.4
## Q1_Annoyed         0.2 -0.3
## Q1_Indifferent     0.2 -0.4
## Q2_Relatable      -0.1 -0.3
## Q2_Funny           0.1 -0.2
## Q2_Confusing      -0.1 -0.3
## Q2_Predictable     0.2 -0.3
## Q2_Entertaining   -0.3  0.1
## Q2_Fantasy        -0.1 -0.2
## Q2_Original       -0.3 -0.1
## Q2_Believable     -0.1 -0.1
## Q2_Boring          0.2 -0.4
## Q2_Dramatic       -0.2  0.0
## Q2_Suspenseful    -0.3  0.0
```

Explained Variance (Scree Plot)

And here is the cumulative variance explained by each PC ("scree plot")

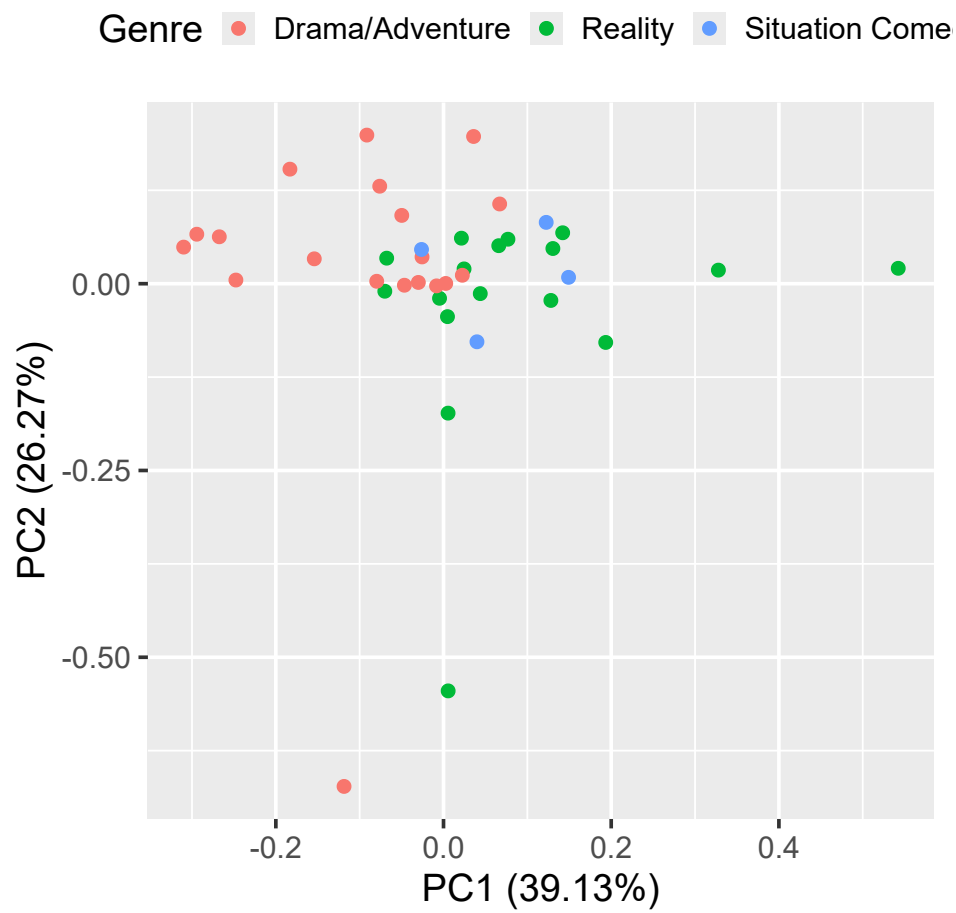
```
pca %>%
  tidy("pcs") %>%
  ggplot(aes(PC, percent)) +
  geom_line() +
  geom_point() +
  labs(
    y = "percent of explaind variance",
    x = "principal component number"
  )
```



Plot data with PCs on axis

The code below generates a scatter plot of the survey data using the first two principal components as the x and y axis.

```
pca %>%
  autoplot(
    data = df_join,
    colour = "Genre",
    size = 3
  ) +
  theme(legend.position = "top")
```



Notes about PCR, Ridge, and Lasso

Here are some important notes to keep in mind when working with PCR, Ridge, and Lasso:

- PCR is an unsupervised method that extracts PCs without using y. While it is possible to extract PCs first and then proceed to cross-validation, this may result in overfitting and biased estimates (why?).
- PCR assumes that the data is *dense*, whereas Lasso assumes that the data is *sparse*.
- Ridge regression can be interpreted as an algorithm that shrinks the PCs of x that have small variance, while PCR leaves M PCs untouched and removes the rest.*

[*] For more information, please see Section 3.4.1 in Hastie et al. (2009).

Sparse PCA (SPCA)

Sparse principal component analysis (SPCA) is a technique that employs lasso-type penalties to produce components with many loadings equal to zero (Zou, Hastie, and Tibshirani, 2006).

To estimate SPCA, we can use the `spca()` function from the `{elasticnet}` package, as shown below:

```
spc <- spca(
  x = X,
  K = 2,
  type = "predictor",
  sparse = "varnum",
  para = c(4, 4)
)
```

In the code above, `type = "predictor"` indicates that the input is a feature matrix (rather than, for example, a covariance matrix), and the number of PCs K is limited to the first two.

Sparseness in the loadings is imposed by `sparse = "varnum"`, which limits the number of non-zero loading parameters, and `para = c(4, 4)` means that each component will have four non-zero loadings.

SPCA results

Here are the loadings of the first two sparse PCs based on the `spca()` algorithm:

```
spc$loadings %>% round(1)
```

We can now proceeds to PCR by tuning `K` and or `para` using cross-validation (not in this lecture.)

```
##                    PC1  PC2
## Q1_Attentive       0.0  0.0
## Q1_Excited         0.0  0.0
## Q1_Happy           0.0  0.0
## Q1_Engaged         0.0  0.0
## Q1_Curious         0.0  0.0
## Q1_Motivated      -0.5  0.0
## Q1_Comforted      -0.1  0.0
## Q1_Annoyed         0.0 -0.4
## Q1_Indifferent     0.0 -0.7
## Q2_Relatable       0.0  0.0
## Q2_Funny           0.0  0.0
## Q2_Confusing       0.0  0.0
## Q2_Predictable     0.0 -0.6
## Q2_Entertaining    0.0  0.0
## Q2_Fantasy        -0.2  0.0
## Q2_Original        0.0  0.0
## Q2_Believable      0.0  0.0
## Q2_Boring          0.0  0.0
## Q2_Dramatic        0.0  0.0
## Q2_Suspenseful    -0.8  0.0
```

PCR-lasso

Taddy (2019) proposes an alternative approach to PCR, where instead of tuning the number of principal components, we tune their number using lasso:

$$\min_{\alpha_0,\alpha} \sum_{i=1}^{N} \left(y_i - \alpha_0 - \sum_{j=1}^{p} \hat{f}_{ij} \alpha_j \right)^2 + \lambda \|\boldsymbol{\alpha}\|_1$$

Tuning each Method using Cross-Validation

In this example, we will compare three methods for predicting y:

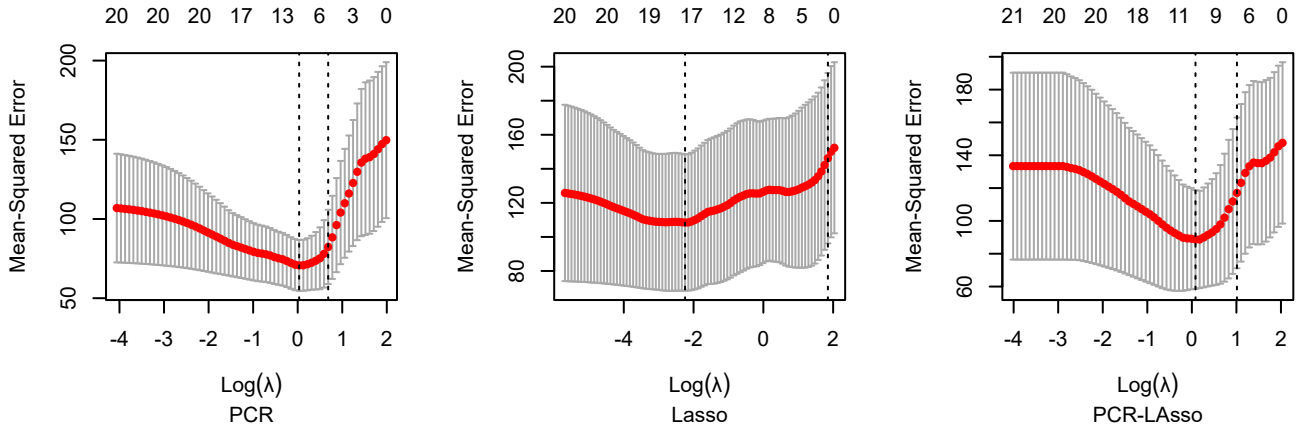
1. PCR-lasso for predicting y on f.
2. Lasso for predicting y on x.
3. PCR-lasso for predicting y on both x and f.

```
lasso_pcr <- cv.glmnet(x = PC, y = Y, nfold = 20)
lasso_x <- cv.glmnet(x = X , y = Y, nfold = 20)
lasso_pcrx <- cv.glmnet(x = cbind(X, PC) , y = Y, nfold = 20)
```

Note that the preferred method depends on the specific application.

Plot cross-validation results

```
par(mfrow=c(1,3))
plot(lasso_pcr, sub = "PCR")
plot(lasso_x, sub = "Lasso")
plot(lasso_pcrx, sub = "PCR-LAsso")
```



Partial Least Squares (PLS)

- Unlike PCR, which extracts PCs in an unsupervised way and doesn't use information on y during the process, PLS extracts factors in a supervised way.
- In theory, PLS should have an advantage over PCR, as it may identify features that are highly correlated with y that PCR would discard.
- However, in practice, there is usually little difference in terms of prediction performance between PLS and PCR in most applications.

The PLS algorithm

(1) Regress y onto each x_j independently, i.e.,

$$y_i = \phi_{1j} x_{i,j} + \varepsilon_{ij}, \qquad \text{for } j = 1, \ldots, p$$

and store $\phi_{11}, \ldots, \phi_{1p}$.

(2) Use the estimated vector of coefficients $\hat{\boldsymbol{\phi}}_1 = (\phi_{11}, \ldots, \phi_{1p})$ to construct a new component v_{i1}, which is a linear combination of the xs:

$$v_{i1} = \hat{\boldsymbol{\phi}}_1{}' \mathbf{x}_i$$

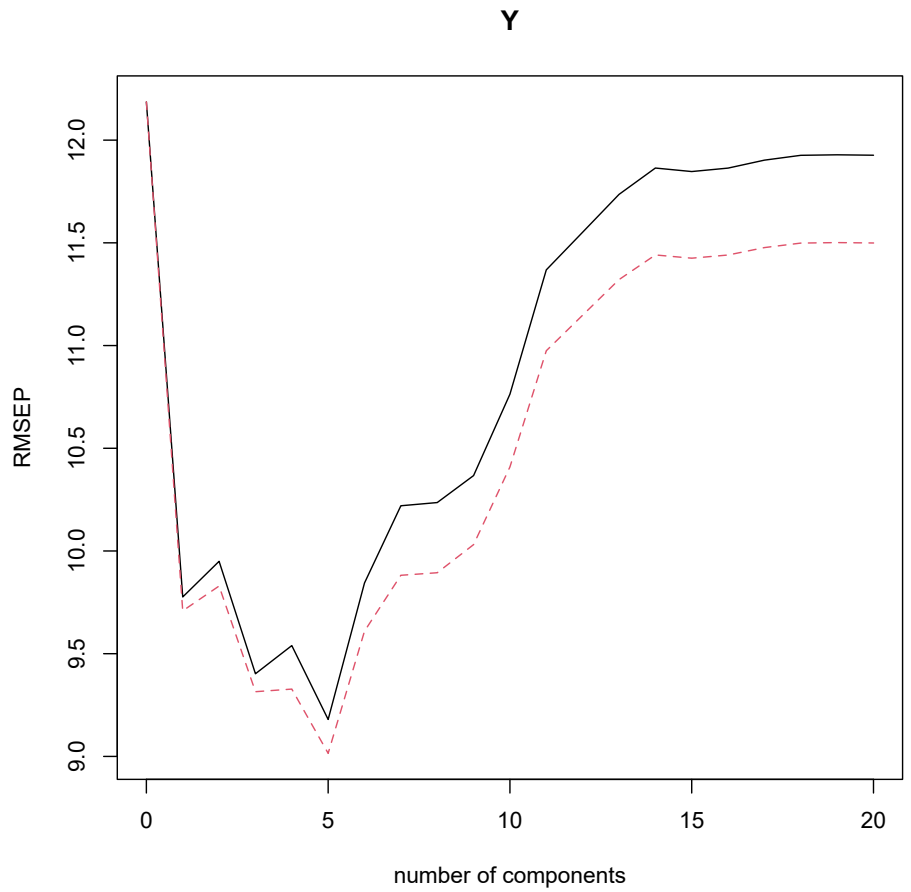
(3) Define the residuals from (1) as a new set of features, and proceed to (2) and estimate the second component, $v_{i2} = \hat{\boldsymbol{\phi}}_2{}' \mathbf{x}_i$.

(4) Continue until you reach the minimum of p or n.

Run PLS with the `pls` Package

We're now ready to estimate PLS using, wait for it..., the {pls} package!

```
cv_pls <- plsr(
  Y ~ X,
  scale = TRUE,
  validation = "CV",
  segments = 10
)
validationplot(cv_pls)
```



Summary

"Despite its simplicity, the linear model has distinct advantages in terms of its interpretability and often shows good predictive performance." (Hastie and Tibshirani)

- Subset selection is slow, unstable, and infeasible when p is large.
- Ridge regression is fast, works when $p > n$, provides good predictions, but is hard to interpret (keeps all features in).
- Lasso is also fast, works when $p > n$, provides good predictions, and performs variable selection.
- However, ridge and lasso shrink the parameters of the model and cannot be used out-of-the-box for econometric analysis.
- Despite their advantages, linear models require a lot of bookkeeping (as well as memory and computing power) when it comes to explicitly adding non-linearities, interactions, and other complex features.

`slides::end()`

Source code

References

Hoerl, A. E., and Kennard, R. W. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1), 55-67.

Jolliffe, I. T. (2010). *Principal Component Analysis*. Springer.

Taddy, M. 2019. *Business Data Science: Combining Machine Learning and Economics to Optimize, Automate, and Accelerate Business Decisions* . McGraw-Hill Education. Kindle Edition.

Tibshirani, R. 1996. Regression shrinkage and selection via the lasso. *J . Roy. Statist. Soc. B*, 58(1), 267-288.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. _Journal of computational and graphical statistic_s, 15(2), 265-286.