Statistics 101C - Week 8 PCA

Shirong Xu

University of California, Los Angeles

shirong@stat.ucla.edu

November 18, 2024

Ridge regression

- Ridge regression uses an L_2-norm penalty, $\|\beta\|^2 = \sum_{j=1}^{p} \beta_j^2 = \beta^T \beta$,

$$\hat{\beta}_\lambda^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda \|\beta\|^2$$

Ridge regression

- Ridge regression uses an L_2-norm penalty, $\|\beta\|^2 = \sum_{j=1}^{p} \beta_j^2 = \beta^T \beta$,

$$\hat{\beta}_\lambda^{ridge} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda \|\beta\|^2$$

- The lasso uses an L_1-norm penalty, $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$,

$$\hat{\beta}^{lasso} = \underset{\beta}{\operatorname{argmin}} \, (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda \|\beta\|_1$$

Other extensions

- Group lasso: if the p variables are partitioned into J groups, and then it is desirable to include or exclude the whole group

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \sum_{j=1}^{J} \|\vec{\beta}_j\|_2,$$

where $\vec{\beta}_j$ is a coefficient vector for the j-th group

Other extensions

- Group lasso: if the p variables are partitioned into J groups, and then it is desirable to include or exclude the whole group

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda \sum_{j=1}^{J} \|\vec{\beta}_j\|_2,$$

  where $\vec{\beta}_j$ is a coefficient vector for the j-th group
- Elastic net:

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|^2 + \lambda_1\|\beta\|_1 + \lambda_2\|\beta\|_2^2$$

Other extensions

- Group lasso: if the p variables are partitioned into J groups, and then it is desirable to include or exclude the whole group

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{y}-\mathbf{X}\beta\|^2+\lambda\sum_{j=1}^{J}\|\vec{\beta}_j\|_2,$$

  where $\vec{\beta}_j$ is a coefficient vector for the j-th group
- Elastic net:

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{y}-\mathbf{X}\beta\|^2+\lambda_1\|\beta\|_1+\lambda_2\|\beta\|_2^2$$

- Fused lasso: penalize the difference between adjacent coef's

$$\min_{\beta} \ \frac{1}{2}\|\mathbf{y}-\mathbf{X}\beta\|^2+\lambda\sum_{j=2}^{p}\|\beta_j-\beta_{j-1}\|_1,$$

An example: Ridge and Lasso

A dataset with 322 observations of major league players on the following 20 variables.

- AtBat: Number of times at bat in 1986
- Hits: Number of hits in 1986
- HmRun: Number of home runs in 1986
- Runs: Number of runs in 1986
- ...
- Salary (response)

An example: Ridge and Lasso

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CR
-Alan Ashby	315	81	7	24	38	39	14	3449	835	69	321	4
-Alvin Davis	479	130	18	66	72	76	3	1624	457	63	224	2
-Andre Dawson	496	141	20	65	78	37	11	5628	1575	225	828	8
-Andres Galarraga	321	87	10	39	42	30	2	396	101	12	48	
-Alfredo Griffin	594	169	4	74	51	35	11	4408	1133	19	501	3
-Al Newman	185	37	1	23	8	21	2	214	42	1	30	

	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
-Alan Ashby	375	N	W	632	43	10	475.0	N
-Alvin Davis	263	A	W	880	82	14	480.0	A
-Andre Dawson	354	N	E	200	11	3	500.0	N
-Andres Galarraga	33	N	E	805	40	4	91.5	N
-Alfredo Griffin	194	A	W	282	421	25	750.0	A
-Al Newman	24	N	E	76	127	7	70.0	A

Implementation of Ridge Regression

```
# install.packages("glmnet")
library(glmnet)
# The set up.
x <- model.matrix(Salary~.,Hitters)[,-1]
y <- Hitters$Salary
```

- Package: glmnet
- Objective: use data of players to predict salary

Implementation of Ridge Regression

```
# Let's define some possible values for lambda
i.exp <- seq(10, -2, length = 100)
grid <- 10^i.exp # From very large to small.

# alpha = 0 implies ridge regression penalty.
ridge.mod <- glmnet(x, y, family = "gaussian", alpha = 0,
                    lambda = grid, standardize = TRUE)
```

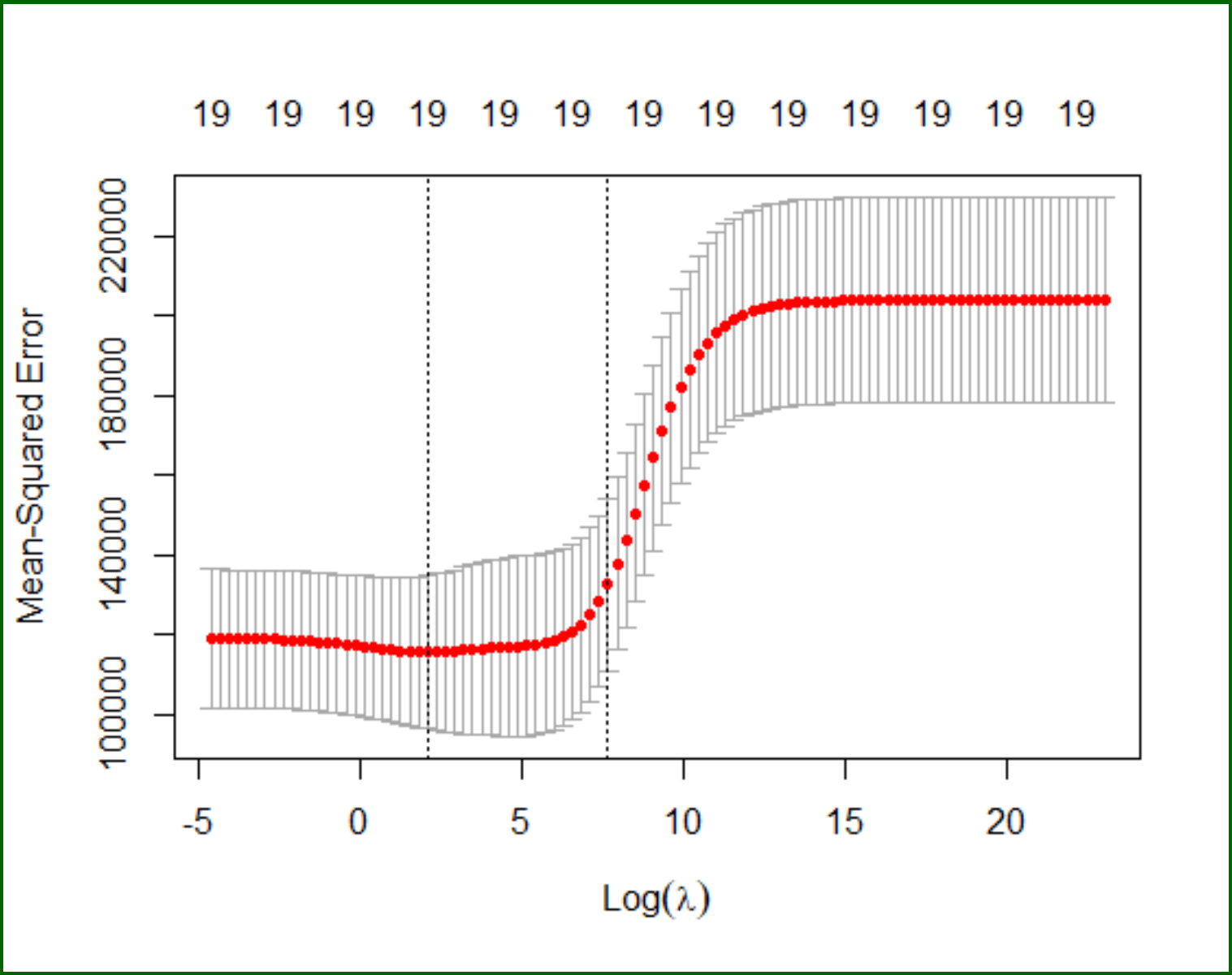
- Package: glmnet
- Objective: use data of players to predict salary
- Consider different λ: $10^{-2} \sim 10^{10}$

Implementation of Ridge Regression

```
# Select the best value for lambda using K-fold cross-validation.
set.seed(1234)
cv.output <- cv.glmnet(x, y, family = "gaussian",
                       alpha = 0,
                       lambda = grid,
                       standardize = TRUE,
                       nfolds = 10)
plot(cv.output)
plot.cv.glmnet
```

- cv.glmnet: will automatically use K-fold cross validation to choose the best λ with minimal validation error

K-fold CV in ridge regression



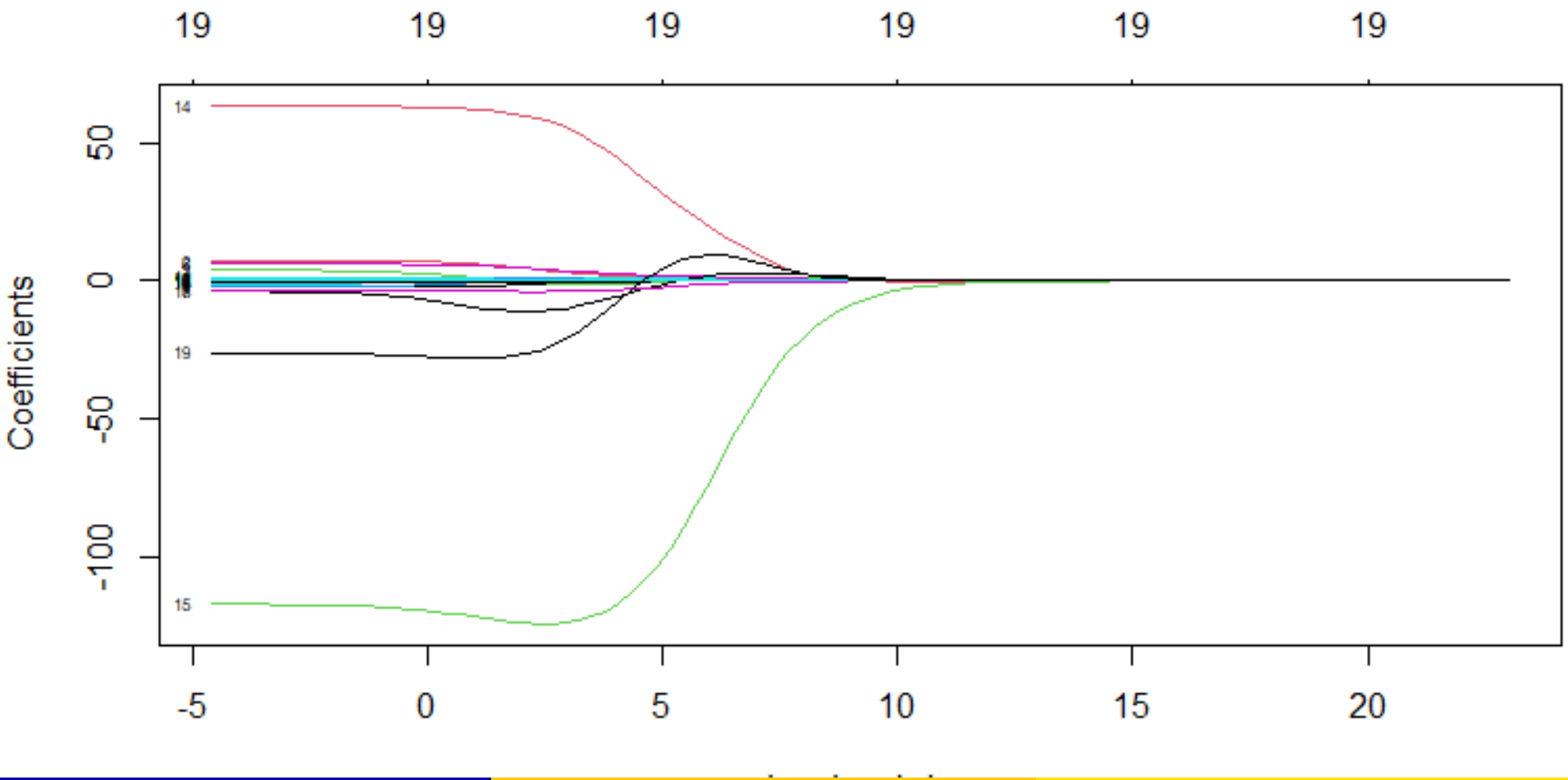
Implementation of Ridge Regression

```
# Retrieve the actual best value of lambda.
best.lambda.cv <- cv.output$lambda.min
best.lambda.cv
```

```
> best.lambda.cv
[1] 8.111308
```

Ridge regression: regularization path

plot(ridge.mod, xvar = "lambda", label = TRUE)



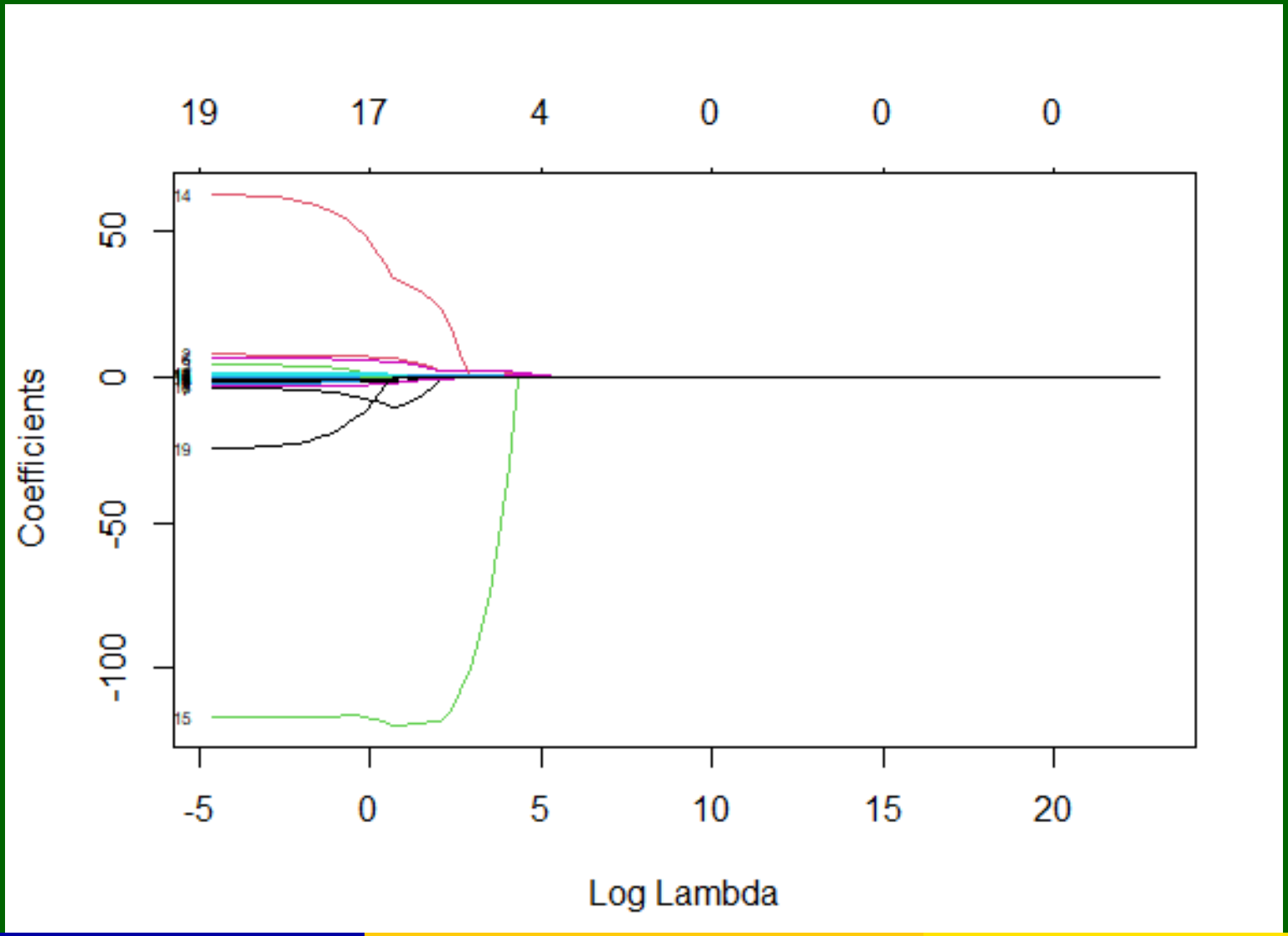
Lasso

```
# alpha = 1 implies Lasso penalty.
lasso.mod <- glmnet(x, y, family = "gaussian",
                    alpha = 1,
                    lambda = grid,
                    standardize = TRUE)
```

- Lasso: also use glmnet package

Lasso

plot(lasso.mod, xvar = "lambda", label = TRUE)

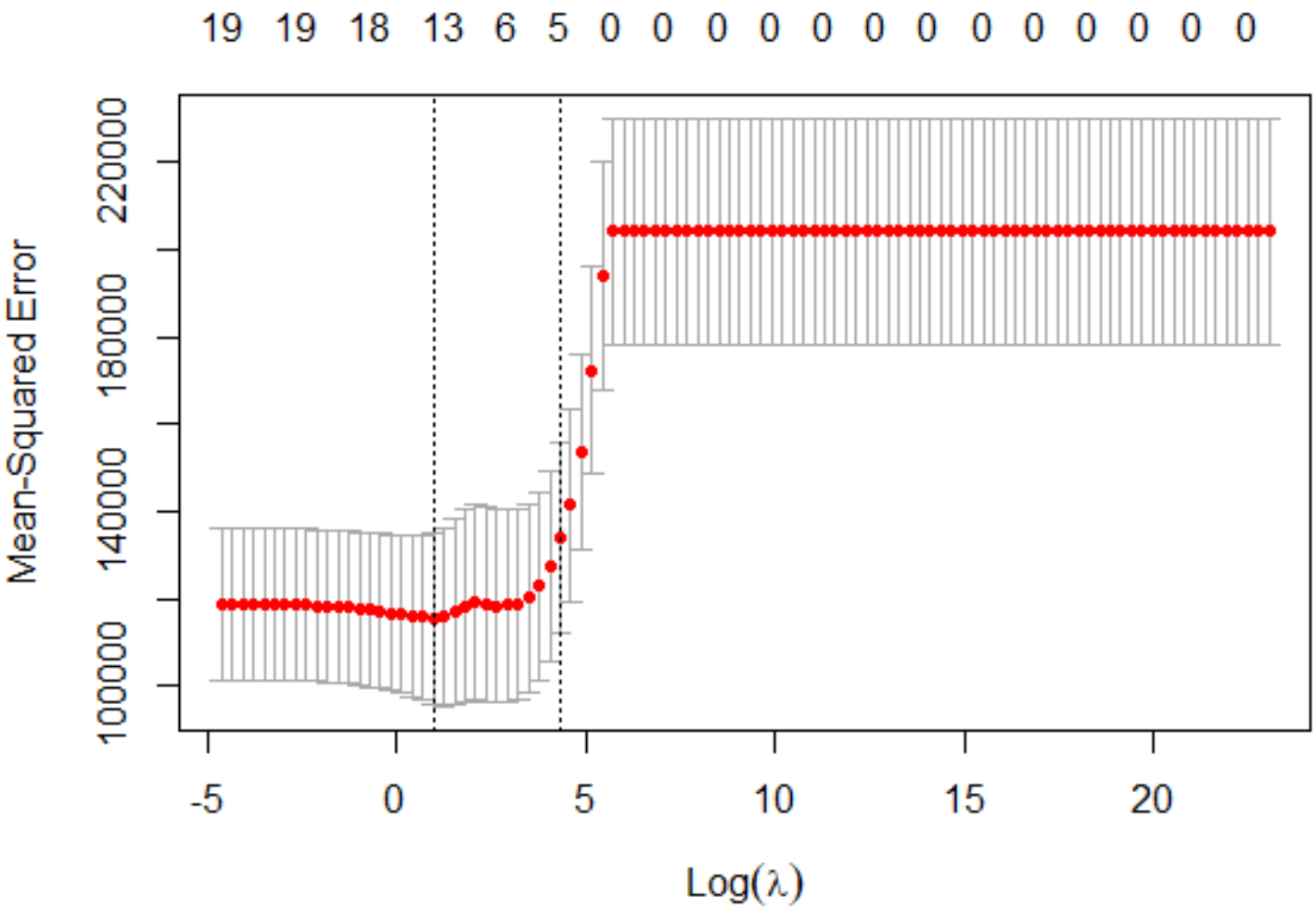


Lasso

```
# Plots of coefficients.
plot(lasso.mod, xvar = "lambda", label = TRUE)


# Select the best value for lambda using K-fold cross-val
set.seed(1234)
lasso.cv.output <- cv.glmnet(x, y, family = "gaussian",
                             alpha = 1,
                             lambda = grid,
                             standardize = TRUE,
                             nfolds = 10)
plot(lasso.cv.output)
```

K-fold CV in Lasso



Lasso

```
# Retrieve the actual best value of lambda.
lasso.best.lambda.cv <- lasso.cv.output$lambda.min
lasso.best.lambda.cv
# Show the output for the best model.
predict(lasso.mod, s = lasso.best.lambda.cv,
        type = "coefficients")
```

```
> lasso.best.lambda.cv
[1] 2.656088
```

Lasso

```
20 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept)  124.0894873
AtBat         -1.5600984
Hits           5.6931685
HmRun          .
Runs           .
RBI            .
Walks          4.7505395
Years         -9.5180241
CAtBat         .
CHits          .
CHmRun         0.5191611
CRuns          0.6604074
CRBI           0.3915415
CWalks        -0.5326868
LeagueN       32.1125493
DivisionW   -119.2583540
PutOuts        0.2726207
Assists        0.1748164
Errors        -2.0567432
NewLeagueN     .
```

Principal component analysis (PCA)

- PCA is a dimension reduction technique.
- Given $X = (X^1, \ldots, X^p)^T$ and $\Sigma = \mathrm{cov}(X)$, find $\{a_1, \ldots, a_p\}$ with $\|a_j\| = 1$ such that
  - $\mathrm{var}(a_j^T X) = a_j^T \Sigma a_j$ is as large as possible, and
  - $\mathrm{cov}(a_j^T X, a_l^T X) = a_j^T \Sigma a_l = 0$ when $j \neq l$
  - **goal**: finding p linear combinations of p raw predictors.

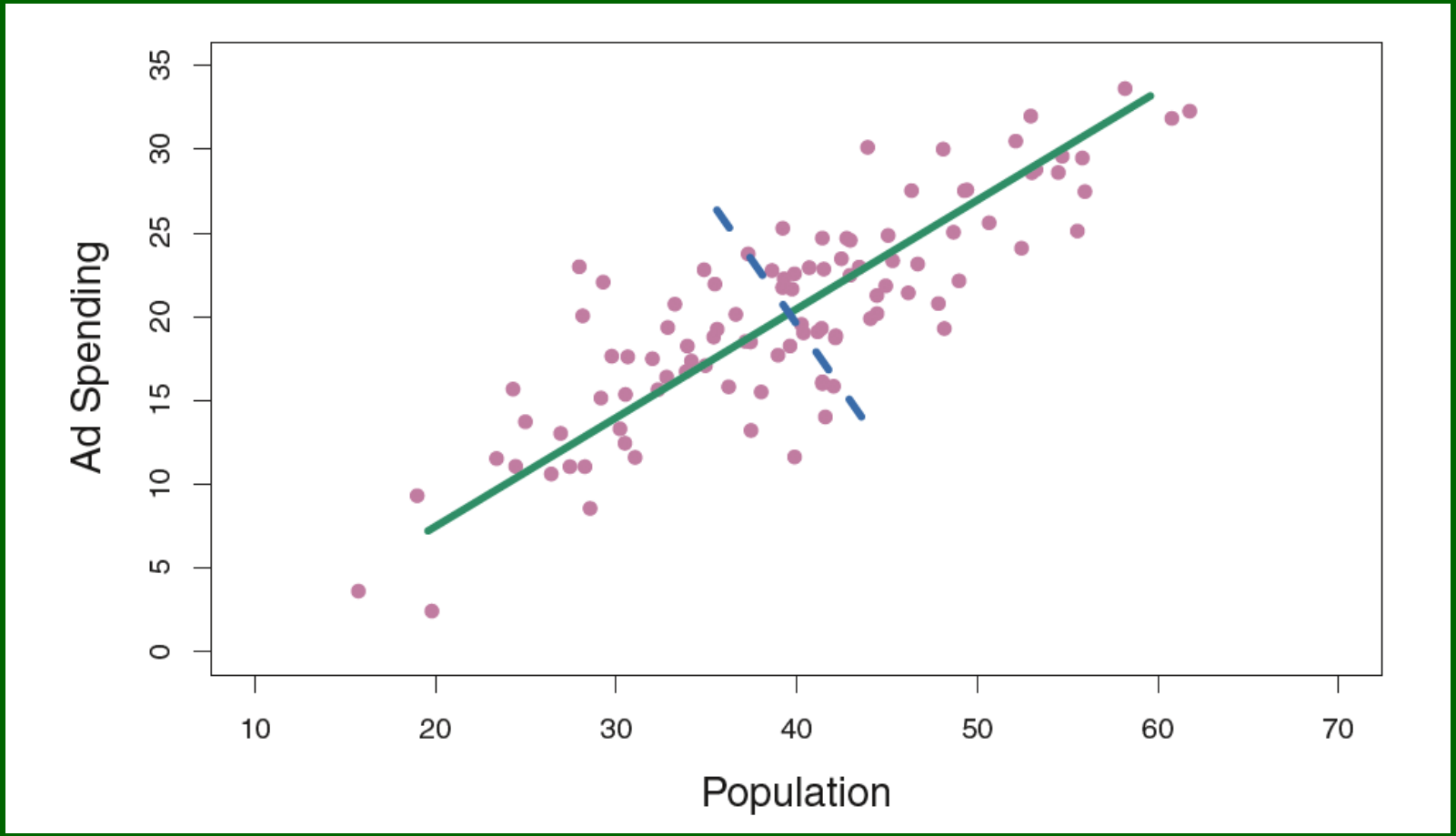
Principal component analysis (PCA)

- PCA is a dimension reduction technique.
- Given $X = (X^1, \ldots, X^p)^T$ and $\Sigma = \mathrm{cov}(X)$, find $\{a_1, \ldots, a_p\}$ with $\|a_j\| = 1$ such that
  - $\mathrm{var}(a_j^T X) = a_j^T \Sigma a_j$ is as large as possible, and
  - $\mathrm{cov}(a_j^T X, a_l^T X) = a_j^T \Sigma a_l = 0$ when $j \neq l$
  - **goal**: finding p linear combinations of p raw predictors.

In general,

- First, find $a_1 = \mathrm{argmax}_a \, a^T \Sigma a$ subject to $\|a\| = 1$
- Then find $a_k = \mathrm{argmax}_a \, a^T \Sigma a$ subject to $\|a\| = 1$ and $a^T \Sigma a_j = 0$ for $j = 1, \ldots, k-1$

An example



PCA and eigen-decomposition

Assume the eigenvalues of Σ is $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$, and the associated eigenvectors are $e_1, \ldots, e_p$, then

- $a_j = e_j$ and the j-th PC is $U_j = e_j^T X$
- $\mathrm{var}(U_j) = e_j^T \Sigma e_j = \lambda_j$
- $\mathrm{cov}(U_j, U_l) = e_j^T \Sigma e_l = 0$

PCA and eigen-decomposition

Assume the eigenvalues of Σ is $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_p \geq 0$, and the associated eigenvectors are $e_1, \ldots, e_p$, then

- $a_j = e_j$ and the j-th PC is $U_j = e_j^T X$
- $\mathrm{var}(U_j) = e_j^T \Sigma e_j = \lambda_j$
- $\mathrm{cov}(U_j, U_l) = e_j^T \Sigma e_l = 0$

To reduce dimension, set $0 < \alpha < 1$ and choose $k \ll p$ such that

$$\frac{\lambda_1 + \ldots + \lambda_k}{\lambda_1 + \ldots + \lambda_p} \geq \alpha,$$

and then work on the feature space spanned by the first k PC's

Principal components regression (PCR)

- Let S be the sample covariance matrix and $\mathbf{q}^j; j = 1, \ldots, J$ be the PC loadings of S
- PCR computes the derived input columns $\mathbf{z}^j = \mathbf{X}\,\mathbf{q}^j$ (sample principal components), and then regresses $\mathbf{y}$ on $\mathbf{z}^1, \ldots, \mathbf{z}^J$

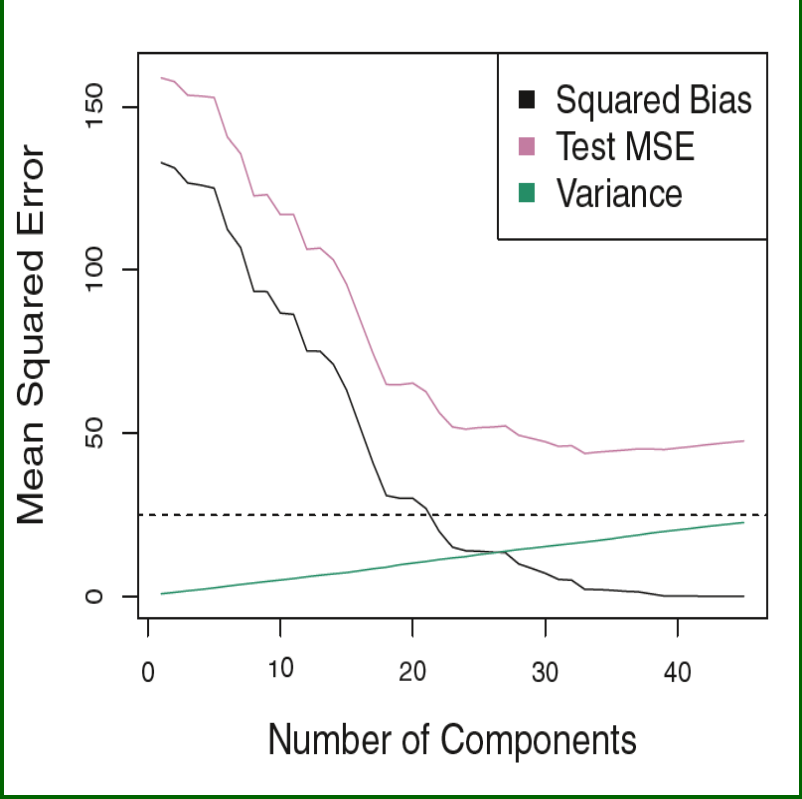
Principal components regression (PCR)

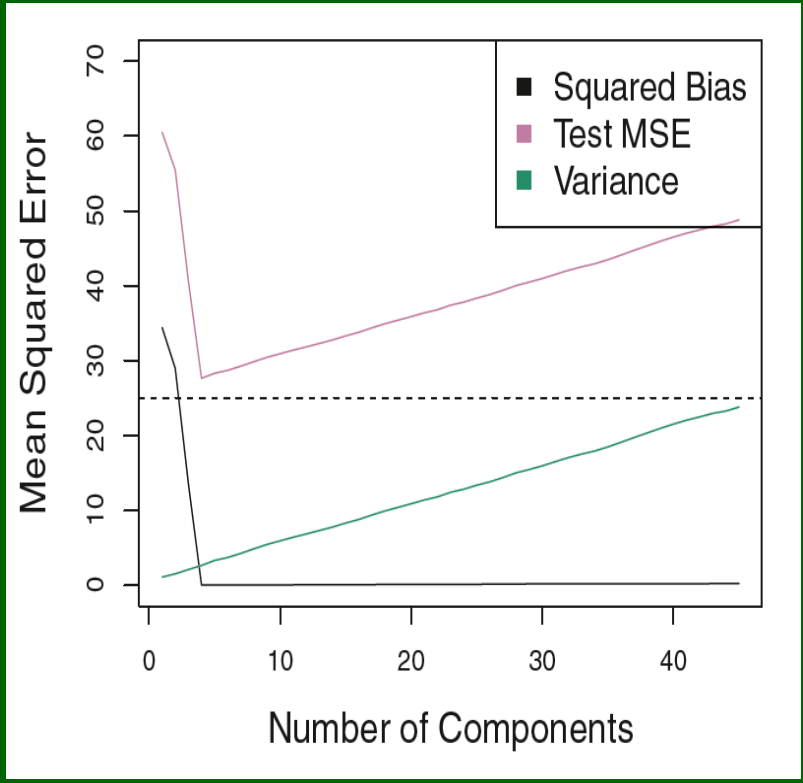
- Let S be the sample covariance matrix and $\mathbf{q}^j; j = 1, \ldots, J$ be the PC loadings of S
- PCR computes the derived input columns $\mathbf{z}^j = \mathbf{X}\,\mathbf{q}^j$ (sample principal components), and then regresses $\mathbf{y}$ on $\mathbf{z}^1, \ldots, \mathbf{z}^J$
- Since the $\mathbf{z}^j$'s are orthogonal, this regression is just a sum of univariate regressions,

$$\hat{\mathbf{y}}^{pcr} = \bar{y} + \sum_{j=1}^{J} \tilde{\gamma}_j \mathbf{z}^j,$$

where $\tilde{\gamma}_j$ is the correlation coefficient of $\mathbf{y}$ on $\mathbf{z}^j$

Some simulated examples with PCR





Some remarks on PCR

- PCR works well when the first few principal components are sufficient, and capture most of the variation in the predictors and the relationship with the response
- PCR does not produce variable selection, as all predictors are included in each principal component
- The number of principal components is typically chosen by cross-validation
- When performing PCR, it is generally recommended to first standardize each predictor

Principal Component Analysis

Objective:

1. Create a new coordinate system in which the new variables are independent of one another.

2. Reduce the dimensions of the problem from p predictors to M new variables, where $M << p$

Setup

We consider a $n \times p$ matrix of predictors:

$$\boldsymbol{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{pmatrix}$$

- Here we have n observation with p predictors.
- We assume that the predictor columns are centered.

Goal in PCA

- We wish to find new predictors $\boldsymbol{Z}_m, m = 1, \ldots, M$

$$\boldsymbol{Z}_m = \boldsymbol{X}\boldsymbol{e}_m = e_{m,1} \begin{pmatrix} x_{1,1} \\ x_{1,1} \\ \vdots \\ x_{n,1} \end{pmatrix} + e_{m,2} \begin{pmatrix} x_{1,2} \\ x_{1,2} \\ \vdots \\ x_{n,2} \end{pmatrix} + \cdots + e_{m,p} \begin{pmatrix} x_{1,p} \\ x_{1,p} \\ \vdots \\ x_{n,p} \end{pmatrix}$$

  where $\boldsymbol{e}_m = (e_{m,1}, e_{m,2}, \ldots, e_{m,p})^T$
- $\boldsymbol{Z}_m$ is a linear combination of the predictor columns. They are called **principal components**.
- $\boldsymbol{e}_m$'s are called **loadings** or **principal component directions**

Constraints on loadings

We search for $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_M$ such that

1. Orthogonal to each other. In other words, if $i \neq j$

$$\boldsymbol{e}_i^T \boldsymbol{e}_j = 0.$$

Constraints on loadings

We search for $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_M$ such that

1. Orthogonal to each other. In other words, if $i \neq j$

$$\boldsymbol{e}_i^T \boldsymbol{e}_j = 0.$$

2. On the same scale. They have a fixed L_2 norm

$$\|\boldsymbol{e}_m\|_2 = 1, m = 1, \ldots, M$$

Constraints on loadings

We search for $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_M$ such that

1. Orthogonal to each other. In other words, if $i \neq j$

$$\boldsymbol{e}_i^T \boldsymbol{e}_j = 0.$$

2. On the same scale. They have a fixed L_2 norm

$$\|\boldsymbol{e}_m\|_2 = 1, m = 1, \ldots, M$$

3. Each $\boldsymbol{Z}_m = \boldsymbol{X}\boldsymbol{e}_m$ should contain high-quality information of $\boldsymbol{X}$. More specifically, should capture a significant portion of the variability of $\boldsymbol{X}$

How do we estimate $e_m, m = 1, \ldots, M$

We start from the sample covariance matrix

$$\boldsymbol{S} = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X} \Rightarrow \Sigma$$

1. We then consider the eigen-decomposition

$$\boldsymbol{S} = \boldsymbol{E}\Lambda\boldsymbol{E}^T$$

2. Λ is a diagonal matrix including the ordered eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \lambda_p \geq 0$$

2. $\boldsymbol{E}$ is a matrix of orthogonal matrix (eigen vectors)

$$\boldsymbol{E} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_p]$$

Some facts

1. The principal components are

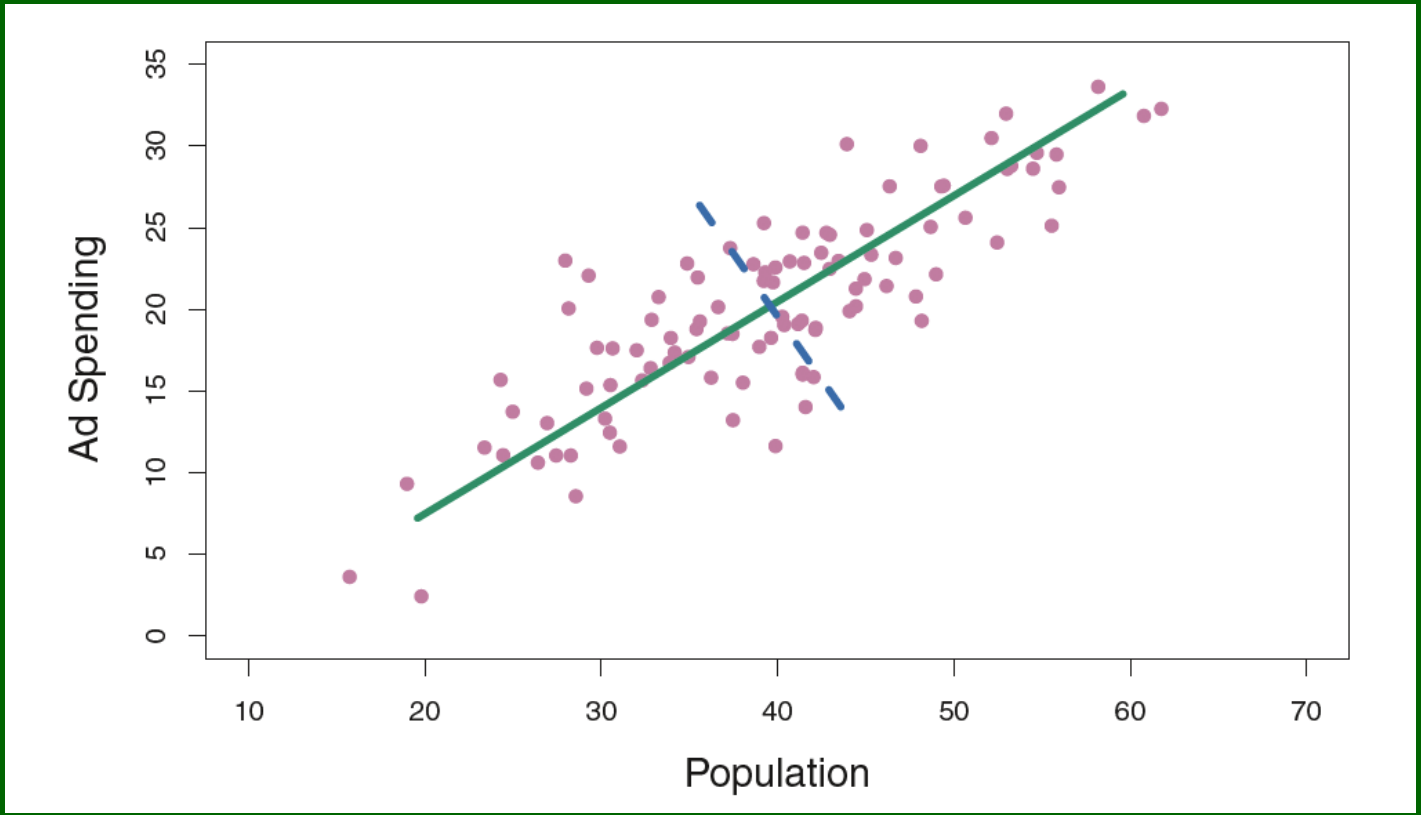
$$\boldsymbol{Z}_m = \boldsymbol{X}\boldsymbol{e}_m,$$

   where $\boldsymbol{e}_m$ is the eigenvector of sample covariance matrix $\boldsymbol{S}$
2. $\boldsymbol{e}_m$ define a new coordinate system
3. The total variance of predictors is

$$\sum_{i=1}^{p} Var(X_i) = \sum_{j=1}^{p} \lambda_j,$$

   where X_j is the j-th predictor.
4. The variance of $\boldsymbol{Z}_m$ is λ_m

- The first principal component points in the direction of the axis that has the most variation.
- The second PC projects onto the axis that is (a) orthogonal to the first and (b) maximizes the variation in that direction.



A Remark on Scaling

- If all predictor columns are both centered and scaled to have a sample variance equal to 1, then

$$C = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X}$$

  is the correlation matrix
- The total variance of the predictors is $\sum_{i=1}^{p} Var(X_i) = p$
- Proportion of variability explained by $\boldsymbol{Z}_m$ is

$$\lambda_m/p$$

- **Note**: The principal components derived from $\boldsymbol{C}$ are different from those of $\boldsymbol{S}$

Discussion on PCA

1. Principal components are constructed without the response. They can be also used for unsupervised learning.

Discussion on PCA

1. Principal components are constructed without the response. They can be also used for unsupervised learning.

2. Reduces dimension but not the number of predictors. Fewer dimensions means less flexibility.

Discussion on PCA

1. Principal components are constructed without the response. They can be also used for unsupervised learning.
2. Reduces dimension but not the number of predictors. Fewer dimensions means less flexibility.
3. Difficult to interpret compared to LASSO.

Application of PCA

```
library(ISLR)
attach(Hitters)
# Remove rows with missing observations.
Hitters <- na.omit(Hitters)
# The set up.
x <- model.matrix(Salary~.,Hitters)[,-1]
y <- Hitters$Salary
hit.pc <- princomp(x[,c(1,2,3)], cor = TRUE)
summary(hit.pc)
```

- Use the first 3 predictors of Hitters dataset for PCA
- cor=TRUE: correlation matrix is considered

Application of PCA

```
> summary(hit.pc)
Importance of components:
                          Comp.1    Comp.2     Comp.3
Standard deviation     1.5454768 0.7588974 0.18861662
Proportion of Variance 0.7961662 0.1919751 0.01185874
Cumulative Proportion  0.7961662 0.9881413 1.00000000
```

- The standard deviation of three components: $\sqrt{\lambda_1} = 1.5454$, $\sqrt{\lambda_2} = 0.7589$, and $\sqrt{\lambda_3} = 0.1886$

Application of PCA

```
> summary(hit.pc)
Importance of components:
                          Comp.1    Comp.2     Comp.3
Standard deviation     1.5454768 0.7588974 0.18861662
Proportion of Variance 0.7961662 0.1919751 0.01185874
Cumulative Proportion  0.7961662 0.9881413 1.00000000
```

- The standard deviation of three components: $\sqrt{\lambda_1} = 1.5454$, $\sqrt{\lambda_2} = 0.7589$, and $\sqrt{\lambda_3} = 0.1886$
- The sum of variance is 3:

$$1.5454^2 + 0.7589^2 + 0.1886^2 \approx 3$$

Application of PCA

```
> summary(hit.pc)
Importance of components:
                          Comp.1    Comp.2     Comp.3
Standard deviation     1.5454768 0.7588974 0.18861662
Proportion of Variance 0.7961662 0.1919751 0.01185874
Cumulative Proportion  0.7961662 0.9881413 1.00000000
```

- The standard deviation of three components: $\sqrt{\lambda_1} = 1.5454$, $\sqrt{\lambda_2} = 0.7589$, and $\sqrt{\lambda_3} = 0.1886$
- The sum of variance is 3:

$$1.5454^2 + 0.7589^2 + 0.1886^2 \approx 3$$

- The proportion of variance of first component $1.5454^2/3 \approx 79.6\%$

Application of PCA

```
> hit.pc$loadings

Loadings:
       Comp.1 Comp.2 Comp.3
AtBat   0.621  0.322  0.714
Hits    0.616  0.362 -0.699
HmRun   0.484 -0.875
```

- We have three new predictors:

$$Z_1 = 0.621 \times AtBat + 0.616 \times Hits + 0.484 \times HmRun$$
$$Z_2 = 0.322 \times AtBat + 0.362 \times Hits - 0.875 \times HmRun$$
$$Z_3 = 0.714 \times AtBat - 0.699 \times Hits + 0 \times HmRun$$

Application of PCA

```
> hit.pc$scores
                          Comp.1       Comp.2       Comp.3
-Alan Ashby          -0.99746408  0.052440762 -8.759047e-05
-Alvin Davis          0.97505642 -0.295173453  2.595080e-03
-Andre Dawson         1.30813889 -0.369670788 -9.165479e-02
-Andres Galarraga    -0.72391711 -0.186425403 -7.313030e-02
-Alfredo Griffin      1.21956889  1.671391010 -2.347811e-03
-Al Newman           -2.48106801  0.014372531  6.972394e-02
-Argenis Salazar     -1.56631926  0.651434541  6.259634e-02
-Andres Thomas       -1.01902166  0.170054269  4.177864e-02
-Andre Thornton       0.07011752 -0.671432811  2.168106e-01
-Alan Trammell        1.93941571 -0.154129096  4.715970e-03
-Alex Trevino        -2.02420562 -0.120023147 -1.051769e-01
```

- New predictors for observations

Implementation of Principal Component Regression

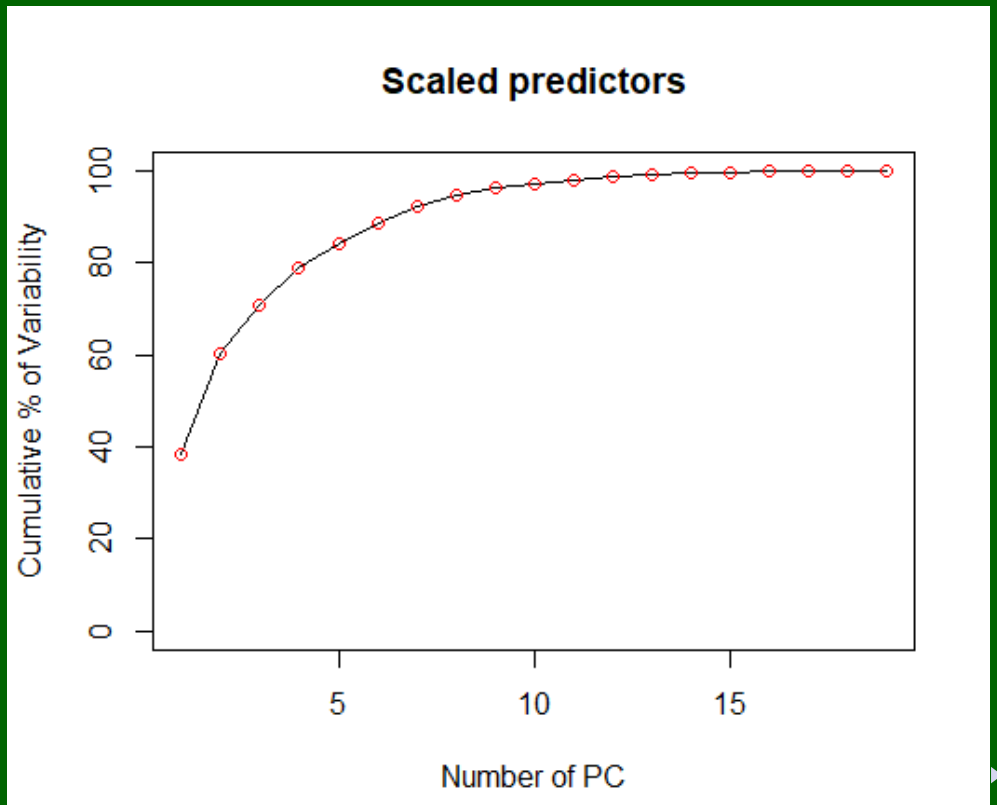
- library(pls)
  - pcr function to conduct Principal Component Regression.
- pcr(y x, scale = TRUE, validation = "CV")
  - To select the number of components using K-fold CV.
- pcr(y x, scale = TRUE, ncomp = M
  - To fit PCR using M components only.
- summary, validationplot.

```
X <- model.matrix(Salary~.,Hitters)[,-1]
hit.pc.sc <- princomp(X, cor = TRUE) # Scaled predictors.

# Percentage of variability explained by the Principal Components.
summary(hit.pc.sc)
```

```
Importance of components:
                          Comp.1    Comp.2    Comp.3    Comp.4     Comp.5
Standard deviation     2.6980929 2.0371069 1.4249239 1.2476293 0.99932745
Proportion of Variance 0.3831424 0.2184108 0.1068636 0.0819252 0.05256081
Cumulative Proportion  0.3831424 0.6015532 0.7084167 0.7903419 0.84290275
                           Comp.6     Comp.7     Comp.8     Comp.9     Comp.10
Standard deviation     0.90854598 0.83026536 0.71626082 0.50073259 0.429903631
Proportion of Variance 0.04344504 0.03628108 0.02700156 0.01319648 0.009727217
Cumulative Proportion  0.88634779 0.92262888 0.94963043 0.96282691 0.972554130
                          Comp.11     Comp.12     Comp.13     Comp.14     Comp.1
Standard deviation     0.370465704 0.357043070 0.309170596 0.247056329 0.22798243
Proportion of Variance 0.007223413 0.006709461 0.005030866 0.003212465 0.00273557
Cumulative Proportion  0.979777542 0.986487003 0.991517869 0.994730334 0.99746591
                          Comp.16      Comp.17      Comp.18      Comp.19
Standard deviation     0.167348055 0.1187122439 0.0697309207 3.445714e-02
Proportion of Variance 0.001473967 0.0007417156 0.0002559159 6.248919e-05
Cumulative Proportion  0.998939879 0.9996815949 0.9999375108 1.000000e+00
```

```r
plot(hit.pc.sc, main = 'Scree Plot on Scaled Predictors')
pc.var.sc <- (hit.pc.sc$sdev)^2
prop.var.sc <- (pc.var.sc/(sum(pc.var.sc)))*100
plot(1:length(prop.var.sc), cumsum(prop.var.sc),
     type = 'l', xlab = 'Number of PC',
        ylab = 'Cumulative % of Variability',
     ylim = c(0,100), xlim = c(1,19),
     main = 'Scaled predictors')
points(1:length(prop.var.sc), cumsum(prop.var.sc), col = 'red')
```



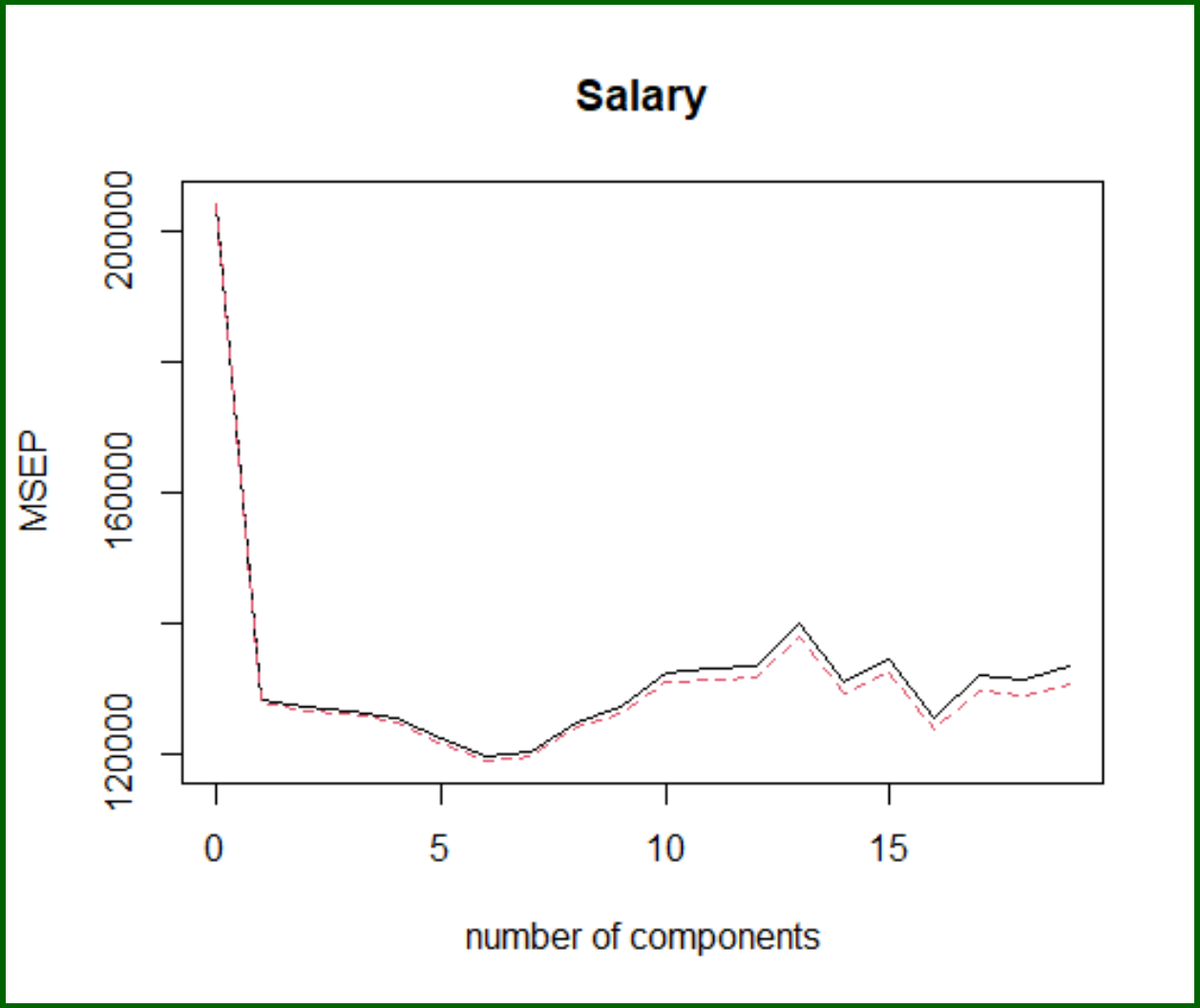
```
library(pls)
set.seed(123)
pcr.fit <- pcr(Salary~., data = Hitters,
               scale = TRUE,
               validation = "CV")
```

- scale = TRUE: scale each variable to have unit variance
- validation = "CV": use cross-validation to determined the number of components

```
> summary(pcr.fit)
Data:   X dimension: 263 19
        Y dimension: 263 1
Fit method: svdpc
Number of components considered: 19

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
CV             452    358.4    356.6    355.8    354.2    349.9    346.0    347.1
adjCV          452    357.8    356.0    355.1    353.4    349.0    345.1    346.1
       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
CV       353.4    357.0     364.0     364.8     365.2     374.2     362.2     366.9
adjCV    352.1    355.5     361.9     362.7     363.1     371.7     359.6     364.0
       16 comps  17 comps  18 comps  19 comps
CV        354.5     363.5     362.6     365.3
adjCV     351.9     360.2     359.2     361.7

TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
X         38.31    60.16    70.84    79.03    84.29    88.63    92.26    94.96
Salary    40.63    41.58    42.17    43.22    44.90    46.48    46.69    46.75
        9 comps  10 comps  11 comps  12 comps  13 comps  14 comps  15 comps
X         96.28     97.26     97.98     98.65     99.15     99.47     99.75
Salary    46.86     47.76     47.82     47.85     48.10     50.40     50.55
        16 comps  17 comps  18 comps  19 comps
X          99.89     99.97     99.99    100.00
Salary     53.01     53.85     54.61     54.61
```



- 6 Component: will have the smallest CV error.

```
pcr.fit.red <- pcr(Salary~., data = Hitters,
                   scale = TRUE, ncomp = 6)
summary(pcr.fit.red)
```

```
pcr.fit.red <- pcr(Salary~., data = Hitters,
                   scale = TRUE, ncomp = 6)
summary(pcr.fit.red)
```

```
> summary(pcr.fit.red)
Data:   X dimension: 263 19
        Y dimension: 263 1
Fit method: svdpc
Number of components considered: 6
TRAINING: % variance explained
        1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
X         38.31    60.16    70.84    79.03    84.29    88.63
Salary    40.63    41.58    42.17    43.22    44.90    46.48
```