Variable Selection

Δημήτρης Φουσκάκης,
Αναπληρωτής Καθηγητής,
ΣΕΜΦΕ, Ε.Μ.Π.

М.Д.Е.: Εφαρμοσμένες Μαθηματικές Επιστήμες
Μάθημα: Υπολογιστική Στατιστική και Στοχαστική Βελτιστοποίηση
Model Selection

- **What is Model Selection?**
  - Evaluation of performance of scientific scenarios.
  - Selection of the “best”.

  **“Best” Model?**
  - The “best” performed model is totally subjective.
    - It may not be possible to find a single model capturing the preferences of all relevant stakeholders in the visited problem.
  - Different procedures (or scientists) support different scientific theories.

  All Models are wrong, but some are useful: George, E.P. Box

- **Main Principles:** Goodness of fit vs. Parsimony.
Multiple Linear Regression

Let us assume that p+1 quantitative variables are available.

- **Y**: is the response or dependent variable.
- **X<sub>1</sub>, X<sub>2</sub>, ... X<sub>p</sub>**: explanatory or independent variables or covariates.

Let us assume a multiple linear regression model:

- **Y=β<sub>0</sub>+β<sub>1</sub> X<sub>1</sub> + β<sub>2</sub> X<sub>2</sub> + ... + β<sub>p</sub> X<sub>p</sub> +ε, ε~N( 0, σ<sup>2</sup> )**
  or equivalently
- **Y~N(μ , σ<sup>2</sup> ), E(Y)=μ= β<sub>0</sub>+β<sub>1</sub> X<sub>1</sub> + β<sub>2</sub> X<sub>2</sub> + ... + β<sub>p</sub> X<sub>p</sub>**

Model expression when fitted to data:

- **Y<sub>i</sub>, X<sub>i</sub> pairs of values for i=1,2, ... , n**
  - **Y<sub>i</sub>= β<sub>0</sub>+β<sub>1</sub> X<sub>i1</sub> + β<sub>2</sub> X<sub>i2</sub> + ... + β<sub>p</sub> X<sub>ip</sub> +ε<sub>i</sub>, ε<sub>i</sub>~N( 0, σ<sup>2</sup> )**
  - **Y<sub>i</sub>~N( μ<sub>i</sub>, σ<sup>2</sup> ), μ<sub>i</sub>= β<sub>0</sub>+β<sub>1</sub> X<sub>i1</sub> + β<sub>2</sub> X<sub>i2</sub> + ... + β<sub>p</sub> X<sub>ip</sub>**
Multiple Linear Regression

Ordinal Least Square (OLS) method for estimating the model coefficients $\beta_0$, $\beta_1$, ..., $\beta_p$.

\[
\text{minimize w.r.t. } \beta : \quad SS = \sum_{i=1}^{n} (y_i - \beta X_i)^2
\]

where $\beta = (\beta_0, \beta_1, ..., \beta_p)^T$ and $X_i = (1, X_{i1}, ..., X_{ip})$, $i = 1, ..., n.$
Variable Selection Problem

- **Problem**: Selection of covariates.
- Variable Selection Problem is a Model Selection Problem; we compare models with the same “structure” but with different covariates.
Variable Selection Problem

- The set of all possible models under consideration can be represented by a vector of binary indicators \( \gamma = (\gamma_1, \ldots, \gamma_p) \in \{0, 1\}^p \), denoting which explanatory variables are present in the linear predictor.
- Therefore the model with only the constant term can be represented by \((0,\ldots,0)\), the model with all the explanatory variables by \((1,\ldots,1)\) and the model with only \(X_1\) and \(X_p\) (for example) by \((1,0,0,\ldots,0,0,1)\).
- The number of all available models (size of model space) is \(2^p\). This can be enormous for even moderate values of \(p\); for example for \(p = 50\) we have \(1.1259\times10^{15}\) available models!
Variable Selection

- **Stepwise procedure:** Adding and removing explanatory variables based on a criterion.
- **Backward procedure:** Removing variables according to a criterion (usually starting from the full model).
- **Forward procedure:** Adding covariates based on a criterion (usually starting from the null/constant model).
- **Full enumeration:** For low number of covariates, we evaluate AIC or BIC for all models \((2^p)\) and select the optimal one.
Variable Selection

Criteria for variable selection:
1. Significance tests
2. AIC
3. BIC
4. Bayesian procedures – Bayes factors (e.g. BAS package)
5. Deviance information criterion (DIC in WinBUGS)

Other methods:
1. Ridge regression
2. Lasso and shrinkage methods
3. Bayesian variable selection and model search algorithms
Stepwise Procedure
Step by step procedure of adding and removing covariates:
- We start from a given model and in every step we check which variable to include (select the one with the min AIC, min BIC, min p-value).
- After the addition of the best variable, we check in all included if they should be removed.
- Each time we select the move according to the minimum or maximum value of a criterion (e.g. min AIC, BIC or p-value).
- We stop when no other move/improvement can be achieved.
- Usual starting models: the null/constant (with no covariates) or the full (with all covariates of the dataset).
Stepwise Procedures

**Backward procedure**

Step by step removal of insignificant covariates:

- We start from the full model and in every step we check which variable must be excluded (once at a time).
- We select the move/model which minimizes a criterion (min AIC or BIC, max p-value).
- We stop when no other covariates can be removed.
- Excluded covariates that may be significant at a step cannot be re-included in the model.
Stepwise Procedures

Forward procedure
Step by step addition of covariates:

✓ We start from the null model and in every step we check which covariates must be added in the model (once at a time).

✓ We select to move/add the covariate with the min AIC, min BIC or min p-value.

✓ We stop when we cannot add any other covariates.

✓ Less computational expensive than the backward and the stepwise methods since it fits less models.
Stepwise Procedures

- Best is step-wise because of double checking.
- Select as starting model the full for moderate $p < n$.
- When $p$ is large or $p > n$, then select as starting model the constant.
- All stepwise methods usually select sub-optimal models.
- Different procedures may end-up to different models.
- If $X$ (design matrix) is (nearly) orthogonal, then variable selection is easier => variable selection procedures will select the optimal model.
- If there are collinear covariates, then variable selection is more difficult => variable selection procedures may end-up to different good but sub-optimal models.
- For $p < 15$ perform full enumeration using the leaps or the BAS packages.
- For large $p$ or $p > n$, use lasso to remove all really bad covariates and continue in the reduced space.
Stepwise Procedures

Disadvantages of stepwise procedures (1)

- The final model is not guaranteed to be optimal in any specified sense since in every step we add or remove a covariate [and we may trap in a locally maximum model space area].
- The procedure yields to a single final model, although in practice there are often several equally good models [use instead some “good models” and (Bayesian) model averaging].
- It doesn’t take into account a researcher’s knowledge about the predictors.
- The p-values used should not be treated too literally. There is so much multiple testing occurring that its validity is dubious.
- The removal of less significant predictors tends to increase the significance of the remaining predictors. This effect leads one to overstate the importance of the remaining predictors.
Disadvantages of stepwise procedures (2)

The final model is not guaranteed to be optimal in any specified sense. Variables that are dropped can still be correlated with the response. It would be wrong to say that these variables are unrelated to the response, it’s just that they provide no additional explanatory effect beyond those variables already included in the model.

✓ Stepwise variable selection tends to pick models that are smaller than desirable for prediction purposes. To give a simple example, consider the simple regression with just one predictor variable. Suppose that the slope for this predictor is not quite statistically significant. We might not have enough evidence to say that it is related to $Y$ but still might be better to use it for predictive purpose.

✓ Therefore, for prediction purposes out-of-sample measures may be useful.
R functions for variable selection (default functions):

- **step**: Stepwise methods using AIC (default) or BIC.
- **add1, drop1**: Computes all the single terms in the scope argument that can be added to or dropped from the model, fit those models and compute a table of the changes in fit.
- **extractAIC, AIC**: Computes the (generalized) AIC.
- **logLik, deviance**: Computes the log-likelihood and the deviance measures.
- **update(formula)**: updates model formulae. This typically involves adding or dropping terms, but updates can be more general.

**MASS library**

- **stepAIC**: similar to step.
- **addterm**: similar to add1.
- **dropterm**: similar to drop1.
Variable Selection with R

R functions for variable selection (functions in other packages):

**Leaps library**
- **leaps**: exhaustive search for the best subsets of the variables, using an efficient branch-and-bound algorithm.
- **regsubsets**: Model selection by exhaustive search, forward or backward stepwise, or sequential replacement (more options than leaps).
- **plot.regsubsets**: Plots a table of models showing which variables are in each model. The models are ordered by the specified model selection statistic.
- **summary.regsubsets**: Table of models plotted using plot.regsubsets.

**BAS library**
- **bas.lm**: for \( p \leq 15 \) fits all models and compares them using AIC/BIC and Bayesian measures. For larger spaces it uses adaptive sampling.
- **image.bma**: Creates an image of the models selected using BAS.
- **plot.bma**: Plot Diagnostics for an blm object.
Example 1: A simulated dataset for variable selection illustration

- n=100 data points.
- p=15 covariates.
- Data in simex62 (a data frame in R).

\[
X_j \sim N(0, 1) \text{ for } j = 2, \ldots, 15
\]
\[
X_1 = N(5 + X_2 + 2.1X_3 - 2.8X_4 - 3.6X_5 + 0.3X_6, 1)
\]
\[
Y \sim 2 - 2.2X_1 - 0.4X_2 + 1.2X_3 - 0.6X_4 - 1.9X_5 - 0.2X_6 + 0.6X_{10} + N(0, 1)
\]
Example 1

Stepwise (from full)

```r
mfull <- lm(y ~ ., data = simex62)
step(mfull, direction = 'both')
```

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum of Sq</th>
<th>RSS</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;none&gt;</td>
<td>91.51</td>
<td>7.133</td>
<td></td>
</tr>
<tr>
<td>+ X8</td>
<td>1</td>
<td>1.62</td>
<td>89.90</td>
</tr>
<tr>
<td>+ X7</td>
<td>1</td>
<td>0.99</td>
<td>90.52</td>
</tr>
<tr>
<td>- X4</td>
<td>1</td>
<td>2.80</td>
<td>94.31</td>
</tr>
<tr>
<td>+ X12</td>
<td>1</td>
<td>0.32</td>
<td>91.19</td>
</tr>
<tr>
<td>+ X15</td>
<td>1</td>
<td>0.22</td>
<td>91.30</td>
</tr>
<tr>
<td>+ X14</td>
<td>1</td>
<td>0.11</td>
<td>91.40</td>
</tr>
<tr>
<td>- X6</td>
<td>1</td>
<td>3.67</td>
<td>95.18</td>
</tr>
<tr>
<td>+ X9</td>
<td>1</td>
<td>0.05</td>
<td>91.46</td>
</tr>
<tr>
<td>+ X13</td>
<td>1</td>
<td>0.02</td>
<td>91.49</td>
</tr>
<tr>
<td>+ X11</td>
<td>1</td>
<td>0.00</td>
<td>91.51</td>
</tr>
<tr>
<td>- X2</td>
<td>1</td>
<td>14.39</td>
<td>105.91</td>
</tr>
<tr>
<td>- X5</td>
<td>1</td>
<td>19.91</td>
<td>111.43</td>
</tr>
<tr>
<td>- X3</td>
<td>1</td>
<td>22.98</td>
<td>114.50</td>
</tr>
<tr>
<td>- X10</td>
<td>1</td>
<td>26.31</td>
<td>117.82</td>
</tr>
<tr>
<td>- X1</td>
<td>1</td>
<td>443.72</td>
<td>535.23</td>
</tr>
</tbody>
</table>
```

Backward

```r
mfull <- lm(y ~ ., data = simex62)
step(mfull, direction = 'back')
```

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum of Sq</th>
<th>RSS</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;none&gt;</td>
<td>91.51</td>
<td>7.133</td>
<td></td>
</tr>
<tr>
<td>- X4</td>
<td>1</td>
<td>2.80</td>
<td>94.31</td>
</tr>
<tr>
<td>- X6</td>
<td>1</td>
<td>3.67</td>
<td>95.18</td>
</tr>
<tr>
<td>- X2</td>
<td>1</td>
<td>14.39</td>
<td>105.91</td>
</tr>
<tr>
<td>- X5</td>
<td>1</td>
<td>19.91</td>
<td>111.43</td>
</tr>
<tr>
<td>- X3</td>
<td>1</td>
<td>22.98</td>
<td>114.50</td>
</tr>
<tr>
<td>- X10</td>
<td>1</td>
<td>26.31</td>
<td>117.82</td>
</tr>
<tr>
<td>- X1</td>
<td>1</td>
<td>443.72</td>
<td>535.23</td>
</tr>
</tbody>
</table>
Example 1

Forward

\[ m_{\text{full}} <- \text{lm}(y \sim ., \text{data}=\text{simex62}) \]
\[ m_{\text{null}} <- \text{lm}(y \sim 1, \text{data}=\text{simex62}) \]
\[ \text{step}(m_{\text{null}}, \text{scope} = \text{list}(<\text{lower}=m_{\text{null}}, <\text{upper}=m_{\text{full}}), \text{direction} = '\text{forward}') \]

Step: AIC=7.13
\[ y \sim X_1 + X_2 + X_4 + X_{10} + X_6 + X_3 + X_5 \]

<table>
<thead>
<tr>
<th>Df</th>
<th>Sum of Sq</th>
<th>RSS</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;none&gt;</td>
<td>91.515</td>
<td>7.1332</td>
<td></td>
</tr>
<tr>
<td>+ X8</td>
<td>1</td>
<td>1.61889</td>
<td>89.896</td>
</tr>
<tr>
<td>+ X7</td>
<td>1</td>
<td>0.99180</td>
<td>90.523</td>
</tr>
<tr>
<td>+ X12</td>
<td>1</td>
<td>0.32143</td>
<td>91.194</td>
</tr>
<tr>
<td>+ X15</td>
<td>1</td>
<td>0.21962</td>
<td>91.295</td>
</tr>
<tr>
<td>+ X14</td>
<td>1</td>
<td>0.11046</td>
<td>91.404</td>
</tr>
<tr>
<td>+ X9</td>
<td>1</td>
<td>0.05046</td>
<td>91.464</td>
</tr>
<tr>
<td>+ X13</td>
<td>1</td>
<td>0.02164</td>
<td>91.493</td>
</tr>
<tr>
<td>+ X11</td>
<td>1</td>
<td>0.00001</td>
<td>91.515</td>
</tr>
</tbody>
</table>
Example 1

Stepwise from null

```r
cmpull<-lm(y~.,data=simex62)
mnull<-lm(y~1,data=simex62)
step(mnull,
    scope=list(lower=mnull,
                upper=mfull),
    direction=both')
```

<table>
<thead>
<tr>
<th>Step</th>
<th>Df</th>
<th>Sum of Sq</th>
<th>RSS</th>
<th>AIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;none&gt;</td>
<td>0</td>
<td>91.51</td>
<td>7.133</td>
<td></td>
</tr>
<tr>
<td>+ X8</td>
<td>1</td>
<td>1.62</td>
<td>89.90</td>
<td>7.348</td>
</tr>
<tr>
<td>+ X7</td>
<td>1</td>
<td>0.99</td>
<td>90.52</td>
<td>8.044</td>
</tr>
<tr>
<td>- X4</td>
<td>1</td>
<td>2.80</td>
<td>94.31</td>
<td>8.142</td>
</tr>
<tr>
<td>+ X12</td>
<td>1</td>
<td>0.32</td>
<td>91.19</td>
<td>8.781</td>
</tr>
<tr>
<td>+ X15</td>
<td>1</td>
<td>0.22</td>
<td>91.30</td>
<td>8.893</td>
</tr>
<tr>
<td>+ X14</td>
<td>1</td>
<td>0.11</td>
<td>91.40</td>
<td>9.012</td>
</tr>
<tr>
<td>- X6</td>
<td>1</td>
<td>3.67</td>
<td>95.18</td>
<td>9.062</td>
</tr>
<tr>
<td>+ X9</td>
<td>1</td>
<td>0.05</td>
<td>91.46</td>
<td>9.078</td>
</tr>
<tr>
<td>+ X13</td>
<td>1</td>
<td>0.02</td>
<td>91.49</td>
<td>9.110</td>
</tr>
<tr>
<td>+ X11</td>
<td>1</td>
<td>0.00</td>
<td>91.51</td>
<td>9.133</td>
</tr>
<tr>
<td>- X2</td>
<td>1</td>
<td>14.39</td>
<td>105.91</td>
<td>19.739</td>
</tr>
<tr>
<td>- X5</td>
<td>1</td>
<td>19.91</td>
<td>111.43</td>
<td>24.820</td>
</tr>
<tr>
<td>- X3</td>
<td>1</td>
<td>22.98</td>
<td>114.50</td>
<td>27.536</td>
</tr>
<tr>
<td>- X10</td>
<td>1</td>
<td>26.31</td>
<td>117.82</td>
<td>30.399</td>
</tr>
<tr>
<td>- X1</td>
<td>1</td>
<td>443.72</td>
<td>535.23</td>
<td>181.753</td>
</tr>
</tbody>
</table>
```
**Example 1**

Model selected by AIC

```r
summary( step( mfull, direction='both' ) )
```

|              | Estimate | Std. Error | t value | Pr(>|t|) |
|--------------|----------|------------|---------|----------|
| (Intercept)  | 1.8175   | 0.5283     | 3.440   | 0.000875 *** |
| X1           | -2.1662  | 0.1026     | -21.120 | < 2e-16 *** |
| X2           | -0.5048  | 0.1327     | -3.804  | 0.000256 *** |
| X3           | 1.1666   | 0.2427     | 4.806   | 5.96e-06 *** |
| X4           | -0.5514  | 0.3289     | -1.676  | 0.097044 .  |
| X5           | -1.7604  | 0.3935     | -4.474  | 2.19e-05 *** |
| X6           | -0.2107  | 0.1097     | -1.920  | 0.057960 .  |
| X10          | 0.5524   | 0.1074     | 5.142   | 1.52e-06 *** |

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9974 on 92 degrees of freedom
Multiple R-squared:  0.9883,    Adjusted R-squared:  0.9875
F-statistic:  1114 on 7 and 92 DF,  p-value:  < 2.2e-16
Example 1

Model selected by BIC

```r
summary( step( mfull, direction='both',k=log(100) ) )
```

Coefficients:

|            | Estimate | Std. Error | t value  | Pr(>|t|) |
|------------|----------|------------|----------|----------|
| (Intercept)| 1.0045   | 0.2116     | 4.747    | 7.47e-06 *** |
| X1         | -2.0051  | 0.0362     | -55.390  | < 2e-16 *** |
| X2         | -0.6354  | 0.1085     | -5.859   | 7.00e-08 *** |
| X3         | 0.8033   | 0.1104     | 7.277    | 1.06e-10 *** |
| X5         | -1.1665  | 0.1728     | -6.750   | 1.25e-09 *** |
| X6         | -0.2549  | 0.1076     | -2.370   | 0.0198 *  |
| X10        | 0.5677   | 0.1081     | 5.253    | 9.46e-07 *** |

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.007 on 93 degrees of freedom
Multiple R-squared:  0.988,  Adjusted R-squared:  0.9872
F-statistic: 1275 on 6 and 93 DF,  p-value: < 2.2e-16
### Example 1

**Manual forward using F-tests and add1 function**

```r
add1(mnull, scope=mfull, test='F')
add1(update(mnull, ~.+X1), scope=mfull, test='F')
add1(update(mnull, ~.+X1+X10), scope=mfull, test='F')
add1(update(mnull, ~.+X1+X10+X2), scope=mfull, test='F')
add1(update(mnull, ~.+X1+X10+X2+X3), scope=mfull, test='F')
add1(update(mnull, ~.+X1+X10+X2+X3+X5), scope=mfull, test='F')
add1(update(mnull, ~.+X1+X10+X2+X3+X5+X6), scope=mfull, test='F')
```

```r
> add1(mnull, scope=mfull, test='F')
SINGLE TERM ADDITIONS
Model: y ~ 1

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum of Sq</th>
<th>RSS</th>
<th>AIC</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>7849.5</td>
<td>438.30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>1</td>
<td>7566.4</td>
<td>283.1</td>
<td>100.06</td>
<td>2619.306</td>
<td>&lt;2.2e-16***</td>
</tr>
<tr>
<td>X2</td>
<td>1</td>
<td>420.0</td>
<td>434.80</td>
<td></td>
<td>5.5407</td>
<td>0.020575***</td>
</tr>
<tr>
<td>X3</td>
<td>1</td>
<td>665.1</td>
<td>7184.4</td>
<td>431.45</td>
<td>9.0723</td>
<td>0.003302**</td>
</tr>
<tr>
<td>X4</td>
<td>1</td>
<td>2025.6</td>
<td>5823.8</td>
<td>410.45</td>
<td>34.0865</td>
<td>6.850e-08***</td>
</tr>
<tr>
<td>X5</td>
<td>1</td>
<td>3214.2</td>
<td>4635.1</td>
<td>387.63</td>
<td>67.9566</td>
<td>7.645e-13***</td>
</tr>
<tr>
<td>X6</td>
<td>1</td>
<td>84.9</td>
<td>7764.5</td>
<td>439.22</td>
<td>1.0718</td>
<td>0.303080</td>
</tr>
<tr>
<td>X7</td>
<td>1</td>
<td>50.3</td>
<td>7799.2</td>
<td>439.66</td>
<td>0.6317</td>
<td>0.428645</td>
</tr>
<tr>
<td>X8</td>
<td>1</td>
<td>97.3</td>
<td>7752.2</td>
<td>439.06</td>
<td>1.2299</td>
<td>0.270132</td>
</tr>
<tr>
<td>X9</td>
<td>1</td>
<td>8.4</td>
<td>7841.1</td>
<td>440.20</td>
<td>0.1050</td>
<td>0.746647</td>
</tr>
<tr>
<td>X10</td>
<td>1</td>
<td>10.4</td>
<td>7839.1</td>
<td>440.17</td>
<td>0.1299</td>
<td>0.719329</td>
</tr>
<tr>
<td>X11</td>
<td>1</td>
<td>55.4</td>
<td>7794.0</td>
<td>439.59</td>
<td>0.6971</td>
<td>0.405778</td>
</tr>
<tr>
<td>X12</td>
<td>1</td>
<td>55.3</td>
<td>7794.1</td>
<td>439.60</td>
<td>0.6956</td>
<td>0.406291</td>
</tr>
<tr>
<td>X13</td>
<td>1</td>
<td>230.5</td>
<td>7618.9</td>
<td>437.32</td>
<td>2.9651</td>
<td>0.089232</td>
</tr>
<tr>
<td>X14</td>
<td>1</td>
<td>137.8</td>
<td>7711.6</td>
<td>438.53</td>
<td>1.7517</td>
<td>0.188747</td>
</tr>
<tr>
<td>X15</td>
<td>1</td>
<td>3.9</td>
<td>7845.5</td>
<td>440.25</td>
<td>0.0491</td>
<td>0.825026</td>
</tr>
</tbody>
</table>
```

> add1(update(mnull, ~.+X1+X10+X2+X3+X5+X6), scope=mfull, test='F')

---

**Single term additions**

```r
Model: y ~ X1 + X10 + X2 + X3 + X5 + X6

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum of Sq</th>
<th>RSS</th>
<th>AIC</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>94.311</td>
<td>94.311</td>
<td></td>
<td>9.4311</td>
<td>0.1424</td>
</tr>
<tr>
<td>X4</td>
<td>1</td>
<td>2.79568</td>
<td>91.515</td>
<td>7.1332</td>
<td>2.8105</td>
<td>0.09704</td>
</tr>
<tr>
<td>X7</td>
<td>1</td>
<td>0.54355</td>
<td>93.767</td>
<td>9.5643</td>
<td>0.5333</td>
<td>0.46707</td>
</tr>
<tr>
<td>X8</td>
<td>1</td>
<td>0.92854</td>
<td>93.382</td>
<td>9.1529</td>
<td>0.9148</td>
<td>0.34135</td>
</tr>
<tr>
<td>X9</td>
<td>1</td>
<td>0.01691</td>
<td>94.294</td>
<td>10.1244</td>
<td>0.0165</td>
<td>0.89808</td>
</tr>
<tr>
<td>X11</td>
<td>1</td>
<td>0.00041</td>
<td>94.310</td>
<td>10.1419</td>
<td>0.0004</td>
<td>0.98400</td>
</tr>
<tr>
<td>X12</td>
<td>1</td>
<td>0.19585</td>
<td>94.115</td>
<td>9.9345</td>
<td>0.1914</td>
<td>0.66274</td>
</tr>
<tr>
<td>X13</td>
<td>1</td>
<td>0.00508</td>
<td>94.306</td>
<td>10.1370</td>
<td>0.0050</td>
<td>0.94401</td>
</tr>
<tr>
<td>X14</td>
<td>1</td>
<td>0.02482</td>
<td>94.286</td>
<td>10.1160</td>
<td>0.0242</td>
<td>0.87667</td>
</tr>
<tr>
<td>X15</td>
<td>1</td>
<td>0.28497</td>
<td>94.026</td>
<td>9.8397</td>
<td>0.2788</td>
<td>0.59874</td>
</tr>
</tbody>
</table>
```
Example 1

Manual forward using F-tests and add1 function

```
summary(update(mnull,~.+X1+X10+X2+X3+X5+X6))
```

```
Call:
  lm(formula = y ~ X1 + X10 + X2 + X3 + X5 + X6, data = simex62)

Residuals:   # Residuals statistics
  Min       1Q   Median       3Q      Max
-2.2870   -0.6335    0.0119    0.5946    3.0284

Coefficients:  # Coefficient estimates
  Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.0045    0.2116   4.747 7.47e-06  ***
X1            -2.0051    0.0362  -55.390 < 2e-16  ***
X10           -0.5677    0.1081   -5.253 9.46e-07  ***
X2            -0.6354    0.1085  -5.859 7.00e-08  ***
X3             0.8033    0.1104    7.277 1.06e-10  ***
X5            -1.1665    0.1728  -6.750 1.25e-09  ***
X6            -0.2549    0.1076  -2.370  0.0198    *
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.007 on 93 degrees of freedom
Multiple R-squared:  0.988,    Adjusted R-squared:  0.9872
F-statistic: 1275 on 6 and 93 DF,  p-value: < 2.2e-16
```
Example 1

Manual backward using F-tests and drop1 function

```
drop1(mfull, test='F')
drop1(update(mfull, ~.-X9), test='F')
drop1(update(mfull, ~.-X9-X11), test='F')
drop1(update(mfull, ~.-X9-X11-X15), test='F')
drop1(update(mfull, ~.-X9-X11-X15-X13), test='F')
drop1(update(mfull, ~.-X9-X11-X15-X13-X14), test='F')
drop1(update(mfull, ~.-X9-X11-X15-X13-X14-X12), test='F')
drop1(update(mfull, ~.-X9-X11-X15-X13-X14-X12-X7), test='F')
drop1(update(mfull, ~.-X9-X11-X15-X13-X14-X12-X7-X8), test='F')
drop1(update(mfull, ~.-X9-X11-X15-X13-X14-X12-X7-X8-X4), test='F')
summary(update(mfull, ~.-X9-X11-X15-X13-X14-X12-X7-X8-X4))
```

- Selects the same model as BIC and forward with F-tests.
Example 1

Several measures

```r
> n<-100
> p<-15
> logLik(mfull)
'log Lik.' -135.8452 (df=17)
> -2*logLik(mfull)+2*(p+2)
'log Lik.' 305.6904 (df=17)
> AIC(mfull)
[1] 305.6904
> extractAIC(mnull)-extractAIC(mfull)
[1] -15.0000 418.4002
> AIC(mnull)-AIC(mfull)
[1] 418.4002
> extractAIC(mfull)
> n*log( summary(mfull)$s^2*(n-p-1)/n )+2*(p+1)
[1] 19.90274
> extractAIC(mnull)
[1] 1.0000 438.3029
> n*log( summary(mnull)$s^2*(n-1)/n )+2*1
[1] 438.3029
```
Example 1

Leaps: selects the best model in every dimension according to BIC

plot(regsubsets(y~., data=simex62, nvmax=15, nbest=1))
Example 1

Leaps: selects the 10 best models in every dimension according to BIC

plot(regsubsets(y~.,data=simex62, nvmax=15, nbest=10))
Example 1

BAS: Full enumeration of the model space using BIC.
Inclusion probability $\Rightarrow$ rescaled weight measure for including each term.

- Postprobs $\Rightarrow$ posterior probability of each model.

```r
> bas.results <- bas.lm(y ~ ., data = simex62, prior = 'BIC')
For m=0, Initialize Tree with initial Model
> bas.results

Call:
bas.lm(formula = y ~ ., data = simex62, prior = "BIC")

Marginal Posterior Inclusion Probabilities:

<table>
<thead>
<tr>
<th></th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
<th>X8</th>
<th>X9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1.000</td>
<td>1.000</td>
<td>0.996</td>
<td>0.999</td>
<td>0.463</td>
<td>0.999</td>
<td>0.494</td>
<td>0.122</td>
<td>0.232</td>
</tr>
<tr>
<td>X10</td>
<td>0.999</td>
<td>0.923</td>
<td>0.115</td>
<td>0.094</td>
<td>0.095</td>
<td>0.111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```r
> summary(bas.results)

Marginal Posterior Inclusion Probabilities:

<table>
<thead>
<tr>
<th></th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
<th>X8</th>
<th>X9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intercept</td>
<td>1.000</td>
<td>1.000</td>
<td>0.996</td>
<td>0.999</td>
<td>0.463</td>
<td>0.999</td>
<td>0.494</td>
<td>0.122</td>
<td>0.232</td>
</tr>
<tr>
<td>X10</td>
<td>0.999</td>
<td>0.923</td>
<td>0.115</td>
<td>0.094</td>
<td>0.095</td>
<td>0.111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```r
> |
```

Call:
bas.lm(formula = y ~ ., data = simex62, prior = "BIC")
```
Example 1

BAS: Posterior inclusion probabilities under BIC

plot(bas.results)
Example 1

**BAS**: Posterior model probabilities of best 20 and included vars using BIC

```
image(bas.results)
```
Example 1

BAS: Full enumeration of the model space using AIC
Inclusion probability => all are higher than BIC
Postprobs => quite small – AIC cannot separate between models

```r
> bas.results <- bas.lm(y ~ ., data = simex62, prior = 'AIC')
For m=0, Initialize Tree with initial Model
> bas.results

Call:
bas.lm(formula = y ~ ., data = simex62, prior = "AIC")

Marginal Posterior Inclusion Probabilities:

<table>
<thead>
<tr>
<th></th>
<th>Intercept</th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
<th>X7</th>
<th>X8</th>
<th>X9</th>
<th>X10</th>
<th>X11</th>
<th>X12</th>
<th>X13</th>
<th>X14</th>
<th>X15</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0000</td>
<td>1.0000</td>
<td>0.9983</td>
<td>1.0000</td>
<td>0.7477</td>
<td>0.9999</td>
<td>0.6580</td>
<td>0.3324</td>
<td>0.5050</td>
<td>0.2727</td>
<td>0.2710</td>
<td>0.3254</td>
<td>0.2743</td>
<td>0.2851</td>
<td>0.2977</td>
<td></td>
</tr>
</tbody>
</table>

> summary(bas.results)

                   Intercept  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10  X11  X12  X13  X14  X15   BF  PostProbs    R2  dim  logmarg
[1,]          1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 1.00000000 0.0204 0.9883 8 -233.8251
[2,]          1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 0.8979958 0.0184 0.9885 9 -233.9327
[3,]          1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0 0.7454425 0.0152 0.9883 8 -234.1189
[4,]          1 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0.6343448 0.0130 0.9885 9 -234.2803
[5,]          1 1 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0.6037587 0.0123 0.9880 7 -234.3297
```
Example 1

BAS: Posterior inclusion probabilities using AIC

plot(bas.results)
Example 1

**BAS:** Posterior model probabilities of best 20 and included vars using AIC

```
image(bas.results)
```
Multi-collinearity: is the (statistically) high linear relationship between one explanatory with (some of) the rest of the explanatories.

Collinearity: Is the perfect (deterministic) linear relationship between one explanatory with (some of) the rest of the explanatories.

- In the bibliography the two terms are frequently used inter-changeably.
Multi-Collinearity

**Side effects**

When one $X$ is a perfect linear combination of the rest $\Leftrightarrow$ the OLS estimates (or the MLEs) do not exist.

When one $X$ is multi-collinear to the rest:

- High standard errors of coefficients.
- Instability of estimators.
- Significant effects will appear as non-significant.
- Deterioration of the effects (even opposite signs of effects).
- Effects between multi-collinear variables will be inseparable and therefore we will not be able to estimate them.
Why multi-collinearity is a problem?

Logical explanation

- When 2 covariates are highly related => they carry similar information (since when we know the value of the one we can precisely predict the value of the other).
- Therefore, such variables are not adding any further information about the effect on Y when we add them sequentially.
- Similar is the case when a covariate is a linear function of more than one.
Multi-Collinearity

**Why multi-collinearity is a problem?**

**Explanation using interpretation of the parameters**

Let us assume the regression model:

\[ Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon \]

If \( X_2 = a + b X_1 \) (perfect linear relationship)

we cannot use the usual interpretation since changing \( X_1 \) has a result changes also in \( X_2 \).

Moreover

\[ Y = \beta_0 + \beta_1 X_1 + \beta_2 (a+bX_1) + \epsilon \]

\[ = (\beta_0 + a \beta_2) + (\beta_1 + \beta_2 b) X_1 + \epsilon \]

Which is the correct effect of \( X_1 \)?
Why multi-collinearity is a problem?

Mathematical explanation

\[ \hat{\beta} = (X^TX)^{-1}X^Ty \]

- \( \hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)^T \) is the vector of the OLS estimators (or MLEs) of dimension \((p+1)\times1\).
- \( X \) is the data or design matrix of dimension \(n \times (p+1)\). The first column refers to the constant term with all elements equal to one (1). Each of the rest columns refer to the data of each variable.
- \( y \) is a vector of dimension \(n \times 1\) with the values of the response variable.
Why multi-collinearity is a problem?

Mathematical explanation

\[ \hat{\beta} = (X^T X)^{-1} X^T y \]

- **Problem**: If a variable (i.e. a column of the data matrix \( X \)) is a linear combination of the rest the inverse \((X^T X)^{-1}\) does not exist.

- **In practice**: Rarely we will observe a perfect linear relationship. If a covariate is highly associated with the rest (i.e. we regress between them and we end up with a very high value of \( R^2 \)) then we have unstable estimates and high standard errors.
Multi-Collinearity

Diagnostics checks for multi-collinearity

- Pearson correlations (for identifying pairwise comparisons).
- $R^2$ for all the regressions between the covariates.
- Variance inflation factors $[ = 1/(1-R^2) ]$.
- Checking the eigenvalues of $X^TX$ and the conditional indexes.
Diagnostics checks for multi-collinearity

1. **Pearson correlations** [They show high linear association between two covariates but it will fail when more variables are involved in the linear combination e.g. for $X_1 = X_2 + X_3 + X_4$].

2. **Variance inflation factors**
   - $VIF_j = (1 - R_j^2)^{-1}$.
   - $R_j^2$ is the coefficient of determination obtained when we fit the regression model with response the covariate $X_j$ and covariates the rest of $X$s.
   - If $VIF_j > 10$  [$R_j^2 > 0.90$] then we have a potential collinearity problem.
Multi-Collinearity

Variance inflation factors

\[
\widehat{\text{Var}}(\hat{\beta}) = (X^T X)^{-1}\hat{\sigma}^2
\]

\[
\widehat{\text{Var}}(\hat{\beta}_j) = \frac{\hat{\sigma}^2}{(n - 1)S^2_{\hat{X}_j}} \times \frac{1}{1 - R^2_j}
\]

VIFs are also given by the diagonal of the inverse correlation matrix!

**VIF Interpretation:** The square root of the variance inflation factor tells you how much larger the standard error is, compared with what it would be if that variable were uncorrelated with the other predictor variables in the model.
Multi-Collinearity

Variance inflation factors in R: "vif" in "car"

```r
> mfull <- lm(y~., data=simex62)
> library(car)
> vif(mfull)

    X1     X2     X3     X4     X5     X6     X7     X8     X9     X10     X11     X12     X13     X14     X15
X1 26.87  1.80  8.02  9.25 15.96  1.36  1.27  1.37  1.13  1.19  1.26  1.22  1.14  1.25
X2  1.80  8.02  9.25 15.96  1.36  1.27  1.37  1.13  1.19  1.26  1.22  1.14  1.25
X3  8.02  9.25 15.96  1.36  1.27  1.37  1.13  1.19  1.26  1.22  1.14  1.25
X4  9.25 15.96  1.36  1.27  1.37  1.13  1.19  1.26  1.22  1.14  1.25
X5 15.96  1.36  1.27  1.37  1.13  1.19  1.26  1.22  1.14  1.25

> round(vif(mfull), 1)

   X1   X2  X3  X4  X5   X6   X7  X8  X9  X10  X11  X12  X13  X14  X15
X1 26.9 1.8  8.0 9.2 16.0 1.4  1.3 1.4 1.3  1.1  1.2  1.3  1.2
X2  1.8  8.0  9.2 16.0 1.4  1.3 1.4 1.3  1.1  1.2  1.3  1.2
X3  8.0  9.2  16.0 1.4  1.3 1.4 1.3  1.1  1.2  1.3  1.2
X4  9.2 16.0  1.4  1.3 1.4 1.3  1.1  1.2  1.3  1.2
X5 16.0 1.4  1.3 1.4  1.1  1.2  1.3  1.2
```
Multi-Collinearity

Condition indexes

- Calculate the eigenvalues of $\mathbf{X}^{T}\mathbf{X}$.
- Eigenvalues close to zero indicate a problem.
- Condition Index
  
  \[ = \text{Square root of } \frac{\text{MAX(eigenvalues)}}{\text{eigenvalues}}.\]
- If $CI_j > 30$ $\Leftrightarrow$ Serious collinearity problem.
- If $CI_j > 15$ $\Leftrightarrow$ possible collinearity problem.
- For small eigenvalues, high values of eigenvectors indicate variables that participate in linear combinations.
Multi-Collinearity

Condition indexes using “colldiag” in “perturb” package

```r
> X <- model.matrix(mfull)
> v <- sqrt(eigen(t(X) %*% X)$value)
> max(v)/v
[1] 1.000000 5.124575 5.451744 5.687871 5.886674 6.131369
> colldiag(mfull, scale=F)$cond
  cond.index
1   1.000000
2   5.124575
3   5.451744
4   5.687871
5   5.886674
6   6.131369
7   6.351875
8   6.465027
9   7.286807
10  7.321037
11  8.030066
12  8.499335
13  8.902555
14 10.304457
15 11.109056
16 54.203670
```

One linear combination
Multi-Collinearity

Variance-decomposition proportions

- Is the proportion of $\text{Var}(\beta_j)$ explained by the corresponding component.
- If a large condition index is associated with two or more variables with large variance decomposition proportions, these variables may be causing collinearity problems. Belsley et al suggest that a large proportion is 50 percent or more.

2004=> 2nd edition
Multi-Collinearity

Variance-decomposition proportions using “colldiag” in “perturb” package

```r
> round(colldiag(mfull, scale=F)$pi, 2)
  intercept  X1  X2  X3  X4  X5  X6  X7  X8  X9  X10  X11  X12  X13  X14  X15
[1,] 0.00  0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
[2,] 0.00  0.00 0.01 0.03 0.00 0.00 0.03 0.04 0.02 0.02 0.00 0.00 0.01 0.12 0.00
[3,] 0.00  0.00 0.00 0.02 0.00 0.00 0.01 0.03 0.02 0.11 0.00 0.02 0.13 0.03 0.00
[4,] 0.00  0.00 0.11 0.01 0.00 0.00 0.04 0.07 0.01 0.01 0.07 0.00 0.02 0.00 0.04
[5,] 0.00  0.00 0.01 0.00 0.00 0.00 0.09 0.10 0.06 0.02 0.03 0.02 0.13 0.00 0.04
[6,] 0.00  0.00 0.03 0.00 0.00 0.01 0.04 0.13 0.04 0.07 0.06 0.06 0.02 0.00 0.01
[7,] 0.00  0.00 0.00 0.00 0.00 0.00 0.08 0.10 0.06 0.00 0.02 0.04 0.02 0.05 0.15
[8,] 0.00  0.00 0.13 0.01 0.00 0.00 0.05 0.01 0.01 0.03 0.12 0.13 0.01 0.00 0.04
[9,] 0.00  0.00 0.03 0.02 0.03 0.01 0.02 0.02 0.10 0.07 0.01 0.02 0.00 0.08 0.03
[10,] 0.00  0.00 0.01 0.00 0.00 0.00 0.09 0.06 0.01 0.11 0.07 0.00 0.00 0.06 0.00
[11,] 0.00  0.00 0.13 0.00 0.00 0.00 0.01 0.01 0.03 0.21 0.00 0.01 0.01 0.08 0.01
[12,] 0.00  0.00 0.06 0.03 0.01 0.00 0.09 0.03 0.04 0.01 0.11 0.00 0.00 0.00 0.01
[13,] 0.00  0.00 0.00 0.00 0.03 0.00 0.15 0.08 0.02 0.13 0.03 0.48 0.14 0.12 0.03
[14,] 0.00  0.01 0.09 0.00 0.02 0.00 0.18 0.26 0.41 0.28 0.03 0.01 0.15 0.19 0.12
[15,] 0.98  0.97 0.38 0.88 0.89 0.95 0.07 0.00 0.06 0.01 0.02 0.00 0.01 0.01 0.03
[16,] 0.98  0.97 0.38 0.88 0.89 0.95 0.07 0.00 0.06 0.01 0.02 0.00 0.01 0.01 0.03
```
6 Multi-Collinearity

How to deal with the collinearity problem

1. Careful design of the experiment
   - Not random X but based on experimental design.
   - The aim is to achieve a nearly orthogonal X (or at least far away from being ill conditioned).
   - Difficult to be implemented (and expensive).

2. Removal of one of the collinear variables
   - Identify the biggest VIF and remove the corresponding covariate.
   - We try to have a model with CI<15 (or at least CI<30).

3. Use of orthogonal transformations (Principal Components) of X.
   - The interpretation of the model is difficult.

Note: In most cases the Stepwise methods will solve the problem by removing one of the collinear covariates.
Ridge Regression is a technique for analyzing multiple regression data that suffer from multicollinearity.

- It shrinks coefficients towards zero (esp. not important ones).
- It is not a variable selection method but it can simplify variable selection.
- It lead to other more efficient shrinkage methods that perform full shrinkage to zero and indirectly variable selection (e.g. LASSO).
- It can be implemented to fit even models on large $p$ – small $n$ datasets.
Ridge Regression

When multi-collinearity occurs

=> least squares estimates are unbiased

=> but their variances are large so they may be far from the true value.

By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.

It is hoped that the net effect will be to give estimates that are more reliable.
Ridge Regression

We start by standardizing all covariates. Hence $X \Rightarrow Z$ (matrix of standardized covariates)

$$\text{minimize} \sum_{i=1}^{n} (y_i - \beta^T z_i)^2 \text{ s.t. } \sum_{j=1}^{p} \beta_j^2 \leq t$$

$$\Leftrightarrow \text{minimize} \ (y - Z\beta)^T (y - Z\beta) \text{ s.t. } \sum_{j=1}^{p} \beta_j^2 \leq t$$

When including an intercept term in the regression, we leave this coefficient unpenalized. If we centered the columns of $X$ then $\beta_0 = \text{mean}(y)$. 
Penalized sum of squares

Using non-linear programming, the above constrained optimization problem is equivalent minimizing the following penalized version of the (residual) sum of squares (RSS)

\[
PRSS(\beta)_{\ell_2} = \sum_{i=1}^{n} (y_i - z_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2
\]

\[
= (y - Z\beta)^\top (y - Z\beta) + \lambda \|\beta\|_2^2
\]
The ellipses correspond to the contours of RSS: the inner ellipse has smaller RSS, and RSS is minimized at OLS estimates. For $p = 2$ the constraint in Ridge corresponds to a circle:

$$\beta_1^2 + \beta_2^2 \leq t$$

We are trying to minimize the ellipse size and circle simultaneously in the ridge regression. The ridge estimate is given by the point at which the ellipse and the circle touch.
Ridge Regression

- There is a trade-off between the penalty term and RSS.
- Maybe a large $\beta$ would give you a better RSS but then it will push the penalty term higher.
- This is why you might actually prefer smaller $\beta$'s with worse RSS. From an optimization perspective, the penalty term is equivalent to a constraint on the $\beta$'s. The function is still the RSS but now you constrain the norm of the $\beta_j$'s to be smaller than some constant $t$.
- There is a correspondence between $\lambda$ and $t$. The larger the $\lambda$ is, the more you prefer the $\beta_j$'s close to zero. In the extreme case when $\lambda=0$, then you would simply be doing a normal linear regression. And the other extreme as $\lambda$ approaches infinity, you set all the $\beta$'s to zero.
Ridge Regression

The ridge solution

Minimizing the penalized RSS, provides us the ridge solution in closed form given by

$$\hat{\beta}_\lambda^{\text{ridge}} = (Z^T Z + \lambda I_p)^{-1} Z^T y$$

which usually has better prediction error than MLEs or OLS estimators.

For $\lambda > 0$, a solution exists even if the original $X^T X$ is not invertible giving us solutions in cases with

- co-linear regressors
- $p > n$
The data augmentation interpretation of the ridge sol.

\[ \hat{\beta}_{\lambda}^{\text{ridge}} = (Z^T Z + \lambda I_p)^{-1} Z^T y \]

Is like considering \( p \) additional data points with zero values for the response and \( X = \text{diag}(\lambda^{1/2}) \) as the data matrix for the additional explanatory variables since the penalized residual sum of squares can be written as

\[ \text{PRSS}(\beta)_{\ell_2} = \sum_{i=1}^{n} (y_i - z_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \]

\[ = \sum_{i=1}^{n} (y_i - z_i^T \beta)^2 + \sum_{j=1}^{p} (0 - \sqrt{\lambda} \beta_j)^2 \]
The ridge estimators are biased since

$$\hat{\beta}_{\lambda}^{\text{ridge}} = \left( I_p + \lambda R^{-1} \right) \hat{\beta}^{\text{ls}}$$

where

$$R = Z^T Z$$

$$\mathbb{E}(\hat{\beta}_{\lambda}^{\text{ridge}}) = \mathbb{E}\left\{ \left( I_p + \lambda R^{-1} \right) \hat{\beta}^{\text{ls}} \right\}$$

$$= \left( I_p + \lambda R^{-1} \right) \beta$$

which means that the ridge estimators are biased for any $\lambda > 0$.
Main Problem: The selection of $\lambda$

- For each $\lambda$, we have a solution of coefficients.
- These are indexed in a single line-plot.
- Hence, the $\lambda$’s trace out a path of solutions (a path for each coefficient depicted by one line for each covariate).
- $\lambda$ is the shrinkage parameter.
- $\lambda$ controls the size of the coefficients.
- $\lambda$ controls the amount of regularization.
- As $\lambda = 0$, we obtain the least squares solutions.
- As $\lambda \uparrow \infty$, we have $\beta^{\text{ridge}} = 0$ (intercept-only model).
Ridge Regression

An example using \texttt{lm.ridge} in MASS package

\texttt{\lambda=0 if no value is specified => provides the OLS estimators and model}

\begin{verbatim}
> ridgel <- lm.ridge( y~., data=simex62 )
> ridgel$coef
   X1         X2         X3         X4         X5
-10.678119176 -0.483937934  1.368944834 -0.610336100 -1.842312223
   X6         X7         X8         X9        X10
 -0.146090230 -0.050276093  0.139990980 -0.006823618  0.501533721
   X11        X12        X13        X14        X15
 -0.011910288  0.070926782 -0.038102733 -0.043846311  0.035371795
\end{verbatim}

The above provides the ridge estimators using standardized covariates.

The intercept is not included here; since we have centered the covariates it is equal to mean(y).

Here, \( \lambda=0 \) so these are the usual OLS for standardized covariates.
Ridge Regression

An example using \texttt{lm.ridge} in \texttt{MASS} package

```r
> mfull <- \texttt{lm(y~.,data=simex62)}
> ridge1 <- \texttt{lm.ridge( y~.,data=simex62 )}
> \texttt{rbind( coef(mfull), coef(ridge1) ) } # same original coefficients

\begin{verbatim}
             (Intercept)      X1        X2       X3        X4       X5       X6
[1,]  2.0334510 -2.205672 -0.4876811  1.247476 -0.685589 -1.933265 -0.1517073
[2,]  2.0334510 -2.205672 -0.4876811  1.247476 -0.685589 -1.933265 -0.1517073
         X7       X8        X9       X10      X11      X12
[1,] -0.04967198  0.1419875 -0.006928123  0.5310626 -0.01329459  0.06461573
[2,] -0.04967198  0.1419875 -0.006928123  0.5310626 -0.01329459  0.06461573
         X13      X14      X15
[1,] -0.04491102 -0.0461431  0.03623824
[2,] -0.04491102 -0.0461431  0.03623824
\end{verbatim}
```

We use \texttt{coef(ridge1)} to obtain the coefficients for the original data.
Here, $\lambda = 0$ so these are the usual OLS for the original data.
Ridge Regression

An example using \texttt{lm.ridge} in \texttt{MASS} package

\begin{verbatim}
> ridge2 <- lm.ridge( y~.,data=simex62, lambda=seq(0,5000, length.out=10000 ) )
> names(ridge2)
[1] "coef"  "scales"  "Inter"  "lambda"  "ym"  "xm"  "GCV"  "kHKB"
[9] "kLW"
> dim(ridge2$coef)
[1] 15 10000
\end{verbatim}

\textbf{Coef :} The coefficients are in a matrix of dimension $p \times \text{length(}\lambda\text{)}$.
Each column corresponds to a set of ridge solution for a single value of $\lambda$.
Each row corresponds to the path of a covariate.
Ridge Regression

An example using `lm.ridge` in MASS package

```r
> ridge2 <- lm.ridge( y~., data=simex62, lambda=seq(0,5000, length.out=10000) )
> names(ridge2)
[1] "coef"  "scales"  "Inter"  "lambda"  "ym"  "xm"  "GCV"  "kHKB"
[9] "kLW"
```

- **scales**: square root of the (biased) variance of X used for the standardization.
- **Inter**: whether the intercept was included in the model (1=yes, 0=no).
  - **lambda**: values of \( \lambda \) used.
  - **ym**, **xm**: means of y and Xs respectively.
- **GCV**: Generalized cross validation (vector, one for each fitted model).
- **kHKB**: k solution according to Hoerl, Kannard a & Baldwin (1975, Comm. Stats).
- **kLW**: k solution according to Lawless & Wang (1976, Comm. Stats).
Ridge Regression

The regularization plot

```r
ridge2 <- lm.ridge( y~., data=simex62, lambda=seq(0,500, length.out=1500 ) )
plot(ridge2)
legend('bottomright', legend=paste('X',1:15, sep=''), ncol=3, col=1:15, lty=1:15, cex=0.8)
```
Ridge Regression

The effective degrees of freedom

In OLS regression:

\[ \hat{y} = X (X^T X)^{-1} X^T y = H y \]

Hence the hat matrix is defined as \( H = X (X^T X)^{-1} X^T \)
and the number of estimated parameters is given by
the rank of the hat matrix (and of the trace
because \( H \) is idempotent) i.e.

\[ p' = rank(H) = trace(H) \]

so \( p' \) are the number of degrees of freedom used by
the model to estimate the parameters
The effective degrees of freedom

In ridge regression:

\[ \hat{y}^{ridge} = Z \left( Z^T Z + \lambda I_p \right)^{-1} Z^T y = H^{ridge} y \]

Hence the hat matrix is defined as \( H^{ridge} = Z \left( Z^T Z + \lambda I_p \right)^{-1} Z^T \).

In analogy to OLS, the number of effectively estimated parameters (effective degrees of freedom) is given by the rank of the hat matrix i.e.

\[ df_\lambda = rank(H^{ridge}) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda} \]

where \( d_j^2 \) are the eigenvalues of matrix \( X^TX \).
Ridge Regression

The regularization plot using the effective degrees of freedom
Ridge Regression

The regularization plot using the effective degrees of freedom: \textit{The R-code}

\begin{verbatim}
l<-seq(0,10000, length.out=10000 )
ridge2 <- lm.ridge( y~.,data=simex62, lambda=l )
n0<-length(l)
df <- numeric(n0)
p<-15
for (i in 1:n0){
  Z <- scale( simex62[,,-1] )
  A <- solve( t(Z)%*%Z + l[i]*diag(p) )
  B <- Z %*% A %*% t(Z)
  df[i] <- sum( diag( B ) )
}
plot(df, ridge2$coef[1,], ylim=range(ridge2$coef))
plot(df, ridge2$coef[1,], ylim=range(ridge2$coef), type='l')
for (j in 2:15) lines(df, ridge2$coef[j,], col=j)
\end{verbatim}
Ridge Regression

The regularization plots using the “genridge” library
Ridge Regression

The regularization plots using the “genridge” library

The R-code

```r
l<-seq(0,1000, length.out=100 )
library(genridge)
r1<-ridge(y~.,data=simex62, lambda=l)
par(mfrow=c(1,2),cex=0.7)
traceplot(r1)
traceplot(r1, X='df')
```
Ridge Regression

Tuning $\lambda$

- We monitor all values by indexing each solution is indexed vs. $\lambda$ (more on this later).
- We use the effective degrees of freedom.
- We use AIC and/or BIC to select $\lambda$ and covariates.
- We use k-fold cross-validation to tune $\lambda$ by selecting the value with the minimum (out-of-sample) prediction error.
Ridge Regression

Selection of $\lambda$ using AIC, BIC and effective dfs

Select $\lambda$ which minimize the AIC or BIC

$$\text{AIC} = n \log(\text{RSS}) + 2df$$
$$\text{BIC} = n \log(\text{RSS}) + df \log(n)$$

Where $df$ is the effective degrees of freedom
Ridge Regression

Plots of AIC and BIC

AIC vs. lambda

BIC vs. lambda

> ridge2$lambda[ AIC==min(AIC) ]
[1] 0.01464646

> ridge2$lambda[ BIC==min(BIC) ]
[1] 0.03433434
Ridge Regression

```r
#----------------------------------------------------------------------------------
#----------------------------------------------------------------------------------
#--Computation of BIC and AIC
#----------------------------------------------------------------------------------
n<-nrow(simex62)
l<-seq(0,0.05, length.out=100 )
ridge2 <- lm.ridge( y~.,data=simex62, lambda=l )
n0<-length(l)
df <- numeric(n0)
AIC <- numeric(n0)
BIC <- numeric(n0)
p<-15
y<-scale(simex62$y, scale=F)
for (i in 1:n0){
  Z <- scale( simex62[,,-1] )
  A <- solve( t(Z)%*%Z + l[i]*diag(p) )
  B <- Z %*% A %*% t(Z)
  yhat<-B%*%y
  RSS <- sum( (y-yhat)^2 )
  df[i] <- sum( diag( B ) )
  AIC[i]<-n*log(RSS)+df[i]*2
  BIC[i]<-n*log(RSS)+df[i]*log(n)
}
par(mfrow=c(1,2))
plot(l,AIC, type='l', xlab='lambda', ylab='AIC', main='AIC vs. lambda')
plot(l,BIC, type='l', xlab='lambda', ylab='BIC', main='BIC vs. lambda')
ridge2$lambda[ AIC==min(AIC) ]
ridge2$lambda[ BIC==min(BIC) ]
```
Ridge Regression

How to select $\lambda$

$\lambda = k_{HKB}$ Proposed by Hoerl, Kennard & Baldwin (1975):

$$k_{HKB} = \frac{p\hat{\sigma}^2}{\hat{\beta}^T \hat{\beta}}$$

$\hat{\sigma}^2, \hat{\beta}$ estimated from ordinary least squares (OLS).

Cure & De Iorio (2012) use a slightly different criterion based on the r-first principal components; this is also used in R package “ridge” (function “linearRidge”)

How to select $\lambda$

Lawless & Wang (1976, Comm. Stats) proposed a slightly modified estimator of $\lambda = k_{LW}$ given by

$$k_{LW} = \frac{p\hat{\sigma}^2}{\hat{\beta}^T (X^T X) \hat{\beta}} = \frac{p\hat{\sigma}^2}{\hat{y}^T \hat{y}}$$
Ridge Regression

How to select $\lambda$

The criteria in R are slightly modified

```r
> zsimex62 <- as.data.frame( scale(simex62) )
> zsimex62$y <- scale( simex62$y, scale=F )
> zfmod <- lm( y~., data=zsimex62 )
>
> n<-100
> p<-15
>
> ridge1$kH
[1] 0.1140814
> (p-2) *summary(zfmod)$s^2/sum( zfmod$coef^2 )
[1] 0.1129406
>
> ridge1$kLW
[1] 0.1766921
> (p-2)*n*summary(zfmod)$s^2/sum( zfmod$fit^2 )
[1] 0.1766921
```
How to select $\lambda$ using cross-validation

Split the data into two fractions:
- Training sample => used for estimation
- Test sample => used for testing the predictive ability of the model

Problems:
- Not a lot of data.
- How to split them? (different splits provide different solutions)
- What size shall we use for training and testing?
How to select $\lambda$ using K-fold cross-validation

- Split the data to $K$ parts (called folds)
- Fit the data to $K-1$ folds
- Test the data to the remaining fold
- Repeat this for all possible test folds
- Report average prediction error
- USUALLY 10-fold CV or 5-Fold
- Also the $n$-fold CV => leave-one-out CV – CV(1)
Ridge Regression

Mean Square error for $T_k$ fold of size $n_k$

$$MSE(T_k) = \frac{1}{n_k} \sum_{i \in T_k} (y_i - \hat{y}_{i,-k})^2$$

$i \in T_k$ : denotes the indexes of all data that lie in $T_k$ fold

$\hat{y}_{i,-k}$ : stands for the predicted value of $y_i$ using the data of all folds except the $k$-th.

Select $\lambda$ with the minimum AMSE or ARMSE

$$AMSE = \frac{1}{K} \sum_{k=1}^{K} MSE(T_k)$$  $$ARMSE = \frac{1}{K} \sqrt{\sum_{k=1}^{K} MSE(T_k)}$$
Ridge Regression

**Mean Square error for CV(1) & GCV**

\[
MSE_{CV(1)} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_{i,-i})^2 = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{y_i - \hat{y}_i}{1 - h_i} \right)^2
\]

The generalized CV is approximately equal to the MSE obtained using CV(1), but much easier to compute

\[
MSE_{CV(1)} \approx GCV = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{y_i - \hat{y}_i}{1 - \frac{Tr(H)}{n}} \right)^2
\]
Ridge Regression

```r
ridge2 <- lm.ridge( y~., data=simex62, lambda=seq(0,0.05, length.out=1000) )
plot(ridge2$lambda, ridge2$GCV, type='l')

> ridge2$lambda[ ridge2$GCV==min(ridge2$GCV) ]
[1] 0.01736737
```
Ridge Regression

K-fold CV using “ridge.cv” in “parcor”

library(parcor); y<-simex62$y; x<-model.matrix(mfull)
ridge.cv(as.matrix(x[,,-1]),y, plot.it=T,
  lambda=seq(0.001,0.25,length.out=10000), k=5)

There seems to be large variability on the selection of k-folds and the corresponding λ but all of them are quite small.
Ridge Regression

Summary of proposed $\lambda$

```r
> lambdas <- numeric()
> lambda[1] <- ridge2$lambda[ AIC==min(AIC) ]
> lambda[2] <- ridge2$lambda[ BIC==min(BIC) ]
> lambda[3] <- ridge2$lambda[ ridge2$GCV==min(ridge2$GCV) ]
> lambda[4] <- ridge2$kHKB
> lambda[5] <- ridge2$kLW

> names(lambda)<-c('AIC', 'BIC', 'GCV', 'kHKB', 'kLW')
> lambda

<table>
<thead>
<tr>
<th></th>
<th>AIC</th>
<th>BIC</th>
<th>GCV</th>
<th>kHKB</th>
<th>kLW</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>0.01464646</td>
<td>0.03434343</td>
<td>0.01717172</td>
<td>0.11408143</td>
<td>0.17669206</td>
</tr>
</tbody>
</table>
```
The least absolute shrinkage and selection operator

Although ridge regression is not directly used in practice, it generated a whole new area of research by considering different penalties.

The most popular approach is the LASSO based on the $\ell_1$ penalization.


- Web of Science: 5063 citations [8/12/2014]
- Scholar google: 11720 citations [8/12/2014]
The least absolute shrinkage and selection operator

Although ridge regression is not directly used in practice, it generated a whole new area of research by considering different penalties. The most popular approach is the LASSO based on the $\ell_1$ penalization.

$$\minimize (y - Z\beta)\top (y - Z\beta) \text{ s.t. } \sum_{i=1}^{p} |\beta_i| \leq t$$

$$\Leftrightarrow \minimize \left\{ (y - Z\beta)\top (y - Z\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$
The least absolute shrinkage and selection operator

LASSO

RIDGEO
The ellipses correspond to the contours of RSS: the inner ellipse has smaller RSS, and RSS is minimized at OLS estimates. For $p = 2$ the constraint in LASSO corresponds to a diamond: 

$$|\beta_1| + |\beta_2| \leq t$$

We are trying to minimize the ellipse size and circle simultaneously in the ridge regression. The ridge estimate is given by the point at which the ellipse and the circle touch.

As $p$ increases, the multidimensional diamond has an increasing number of corners, and so it is highly likely that some coefficients will be set equal to zero. Hence, the lasso performs shrinkage and (effectively) variable selection.
Lasso and ridge regression both put penalties on $\beta$. More generally, penalties of the form

$$\lambda \sum_{j=1}^{p} |\beta_j|^q \leq t$$

may be considered, for $q \geq 0$. Ridge regression and the Lasso correspond to $q=2$ and $q=1$, respectively. When $X_j$ is weakly related with $Y$, the lasso pulls $\beta_j$ to zero faster than ridge regression.

- **Elastic Net** combines the two ideas; you're looking to find the $\beta$ that minimizes:

$$\begin{align*}
(y - Z\beta)^T (y - Z\beta) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} |\beta_j|^2
\end{align*}$$


Tuning $\lambda$ or $t$

- Again, we have a tuning parameter $\lambda$ that controls the amount of regularization.
- One-to-one correspondence with the threshold $t$ implemented on the $\ell_1$.

- If we set $t$ equal to then we obtain no shrinkage and hence the OLS are returned.

- We have a path of solutions indexed by $\lambda$ or $t$ or by the shrinkage factor $s = \|\beta\|_1 / \max |\beta|_1$.
LASSO

• In regression, you're looking to find the \( \beta \) that minimizes:

\[
(y - Z\beta)^T (y - Z\beta)
\]

• In LASSO, you're looking to find the \( \beta \) that minimizes:

\[
(y - Z\beta)^T (y - Z\beta) + \lambda \sum_{j=1}^{p} |\beta_j|
\]

• So when \( \lambda = 0 \) there is no penalization and you have the OLS solution; this is

\[
\max \sum_{j=1}^{p} |\beta_j| = \max |\beta|_1
\]

• As the penalization parameter \( \lambda \) increases, \( \sum_{j=1}^{p} |\beta_j| \)

is pulled towards zero, with the less important parameters pulled to zero earlier.

• Therefore the shrinkage factor \( s \) presents the ratio of the sum of the absolute current estimate over the sum of the absolute OLS estimates and takes values in \([0,1]\); when is equal to 1 there is no penalization and we have the OLS solution and when is equal to 0 all the \( \beta_j \)s are equal to zero.
Lasso performs also variable selection

- Large enough $\lambda$ (or small enough $t$ or $s$) will set some coefficients exactly equal to 0!
- So the LASSO will perform variable selection for us!
- Nevertheless, solutions proposed also by k-fold CV (we will discuss this later on) suggest that LASSO suggests over-fitted models.
Lasso performs also variable selection

Screening

SUGGESTION:
change name to least angle shrinkage and screening operator!

See for details in
- Bullman and Mandozi, 2013, *Comp. Stats*

Still extremely useful when $p$ is large (even $p >> n$) => it will clear all irrelevant variables very fast.
Computing the lasso solution

Lasso solution has no closed form (unlike ridge regression).

**Original implementation**: involves quadratic programming techniques from convex optimization.

**More popular implementation**: the least-angle regression (LARS) by Efron, Hastie & Tibshirani (2004). Annals of Stats. [Citations WOS: 1913; Scopus: 2319; Scholar: 4544 on 8/12/2014]

- lars package in R implements the LASSO.
- LARS computes the LASSO path efficiently.
- Other alternatives are also available.
LASSO

Implementation of LASSO

Steps:
1. Run Lasso for a variety of values.
2. Plot the regularization paths.
3. Implement k-fold regularization.
4. Estimate the coefficients using $\lambda$ with minimum CV-MSE.
Implementing LASSO using the "lars" package

Steps 1: Run Lasso for a variety of values

> library(lars)
> X<-model.matrix(mfull)[,-1]
> lassol <- lars( X, simex62$y )
> lassol

Call:
  lars(x = X, y = simex62$y)
R-squared:  0.989
Sequence of LASSO moves:
  X1  X2  X4  X10  X6  X15  X3  X5  X8  X12  X7  X9  X4  X13  X4  X14  X11
  Var  1  2  4  10  6  15  3  5  8  12  7  9  -4  13  4  14  11
  Step  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17

Sequence of actions - variables added or excluded in each value of \( \lambda \)
Implementing LASSO using the "lars" package

Steps 2: Plot the regularization paths

> plot(lasso1)
Implementing LASSO using the "lars" package

Steps 2: Plot the regularization paths

```r
> plot(lasso1, breaks="FALSE")
```
Implementing LASSO using the "lars" package

Steps 2: Plot the regularization paths

```r
> plot(lasso1, breaks="FALSE", xlim=c(0.5, 1.0), ylim=c(-20,15))
```
Implementing LASSO using the “lars” package

Steps 3-4: Implement 10-fold CV and select s

```r
> res.cv <- cv.lars(X, simex62$y) # default model='fraction'
> lambda <- res.cv$index
> cv <- res.cv$cv
> mincv.s <- lambda[cv==min(cv)]
> coef(lassol, s=mincv.s, mode='fraction')

X1        X2        X3        X4        X5
-1.981141891 -0.647421308  0.731101103  0.000000000 -1.064833575
X6        X7        X8        X9        X10
-0.214042018 -0.028820012  0.070394537 -0.004542092  0.540497599
X11       X12       X13       X14       X15
 0.000000000  0.030101516 -0.001087987  0.000000000  0.049345451

> mincv.s
[1] 0.8080808
```
Implementing LASSO using the "lars" package

Steps 3-4: Implement 10-fold CV and select $s$
Implementing LASSO using the “lars” package

Steps 3-4: Select \( \lambda \) (or s) using Mallows Cp

> rescp<-summary(lassol)
> coef(lassol, s=which.min(rescp$Cp), mode="step")

<table>
<thead>
<tr>
<th></th>
<th>X1</th>
<th>X2</th>
<th>X3</th>
<th>X4</th>
<th>X5</th>
<th>X6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-1.980111217</td>
<td>-0.647467611</td>
<td>0.728611608</td>
<td>0.000000000</td>
<td>-1.059868876</td>
<td>-0.213813080</td>
</tr>
<tr>
<td></td>
<td>X7</td>
<td>X8</td>
<td>X9</td>
<td>X10</td>
<td>X11</td>
<td>X12</td>
</tr>
<tr>
<td></td>
<td>-0.028354098</td>
<td>0.069273468</td>
<td>-0.004339981</td>
<td>0.539978293</td>
<td>0.000000000</td>
<td>0.029408336</td>
</tr>
<tr>
<td></td>
<td>X13</td>
<td>X14</td>
<td>X15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.000000000</td>
<td>0.000000000</td>
<td>0.049257018</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Mallows (1973, *Technometrics*) \( C_p \), is used to assess the fit of a regression model.
- Is equal to

\[
C_{pm} = \frac{RSS_m}{\hat{\sigma}^2_{full}} - (n - 2p_m)
\]

- It is equivalent to AIC in normal regression models
- It is approximately equal to the MSE from the leave-one-out CV
Implementing LASSO using the “lars” package

Steps 3-4: Select $\lambda$ (or $s$) using Mallows Cp

plot(lasso1, xvar='n', plottype='Cp')
plot(lasso1, xvar='n', plottype='Cp', ylim=c(12,20), xlim=c(0.7,1))
Implementing LASSO using the “lars” package

Steps 3-4: Select $\lambda$ (or $s$) using Mallows Cp

```r
> # finding the $s$ corresponding to the optimal Cp
> blasso <- coef(lasso1, s=which.min(rescp$Cp), mode="step")
> bols <- coef(mfull)[-1]
> # use std coef
> zblasso <- coef(lasso1, s=which.min(rescp$Cp), mode="step") * apply(X,2,sd)
> zbols <- coef(mfull)[-1] * apply(X,2,sd)
> s <- sum( abs( zblasso ) )/sum( abs( zbols ) )
> s
[1] 0.8070412
```
Implementing LASSO using the “glmnet” package

- Glmnet package is more friendly
- Wider selection of functions
- Directly suggests min lambda and lambda with equivalent CV-MSE but supporting more parsimonious models
- Better plots
- Can be implemented for normal models
Implementing LASSO using the “glmnet” package

```
library(glmnet)
lasso2 = glmnet(X, simex62$y)
plot(lasso2, label=T)
```
Implementing LASSO using the "glmnet" package

plot(lasso2, xvar='lambda', label=T)
Implementing LASSO using the “glmnet” package

```r
> lasso3 <- cv.glmnet(X, simex62$y)
> lasso3$lambda.min
[1] 0.02056748
> lasso3$lambda.1se
[1] 0.1322092
> plot(lasso3)
```

Lambda.min = is the one with the minimum CV-MSE

Lambda.1se = largest value of lambda such that error is within 1 standard error of the minimum [more parsimonious]
Implementing LASSO using the "glmnet"

```r
> blasso3 <- coef(lasso3, s = "lambda.min")
> blasso3
16 x 1 sparse Matrix of class "dgCMatrix"
   1
 (Intercept) 0.868534480
X1      -1.978492963
X2      -0.648047223
X3       0.725104150
X4       .
X5      -1.053018513
X6      -0.213750082
X7       -0.027791640
X8       0.068381266
X9     -0.004655329
X10      0.539495160
X11      .
X12      0.029192426
X13      .
X14      .
X15      0.049742166
> zblasso <- blasso3[-1] * apply(X, 2, sd)
> zbols <- coef(mfull)[-1] * apply(X, 2, sd)
> s <- sum(abs(zblasso))/sum(abs(zbols))
> s
[1] 0.8058524
```

Coefficients for lambda min
With the minimum CV-MSE
These coefficients are the effects for original (unstandardized) variables

We multiply with the sds in order to obtain the effects for the standardized variables

S=0.805
Implementing LASSO using the “glmnet”

```r
> blasso3 <- coef(lasso3)  # blasso3 <- coef(lasso3, s = "lambda.1se")
> blasso3
16 x 1 sparse Matrix of class "dgCMatrix"

1
(Intercept) -0.596370022  
X1  -1.686700461  
X2  -0.774738777  
X3   0.009628299  
X4   0.750225831  
X5  
X6  -0.223093864  
X7  
X8  
X9  
X10 0.457502129  
X11  
X12  
X13  
X14  
X15 0.028836594
```

**Coefficients for lambda 1se**

With distance of 1 se from the minimum CV-MSE

```r
> zblasso <- blasso3[-1] * apply(X, 2, sd)
> zbols <- coef(mfull)[-1] * apply(X, 2, sd)
> s <- sum(abs(zblasso)) / sum(abs(zbols))
> s
[1] 0.6418527
```

S=0.64