Variable Selection

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

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.
- $X_1, X_2, \ldots X_p$: explanatory or independent variables or covariates.

Let us assume a multiple linear regression model:

- $Y=\beta_0+\beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2)$

  or equivalently

- $Y \sim N(\mu, \sigma^2), E(Y)=\mu= \beta_0+\beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$

Model expression when fitted to data:

- Y_i, X_i pairs of values for i=1,2, … , n
  - $Y_i= \beta_0+\beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2)$
  - $Y_i \sim N(\mu_i, \sigma^2), \mu_i= \beta_0+\beta_1 X_{i1} + \beta_2 X_{i2} + \ldots + \beta_p X_{ip}$

Multiple Linear Regression

- Ordinal Least Square (OLS) method for estimating the model coefficients β_0, $\beta_1,\dots,\beta_{p.}$

$$\text{minimize w.r.t. } \boldsymbol{\beta}: \quad SS = \sum_{i=1}^{n} (y_i - \boldsymbol{\beta}\mathbf{X}_i)^2$$

where $\boldsymbol{\beta} = (\beta_0,\beta_1,\dots,\beta_p)^T$ and

$\mathbf{X}_i = (1, X_{i1},\dots,X_{ip})$, $i = 1,\dots,n$.

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** $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_p) \in \{0, 1\}^p$, denoting which explanatory variables are present in the linear predictor.
- The γ_j, j=1,…,p, takes the value 1 if variable j is included in the model and 0 otherwise.
- Therefore the model with only the constant term can be represented by (0,…,0), the model with all the explanatory variables by (1,….1) and the model with only X_1 and X_p (for example) by (1,0,0,…,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 1.1259e+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 Procedures

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.

Stepwise Procedures

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.

Variable Selection with R

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≤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

Example 1: A simulated dataset for variable selection illustration

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

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

Example 1

Stepwise (from full)

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

```
Step:  AIC=7.13
y ~ X1 + X2 + X3 + X4 + X5 + X6 + X10

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

Backward

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

```
Step:  AIC=7.13
y ~ X1 + X2 + X3 + X4 + X5 + X6 + X10

       Df Sum of Sq    RSS     AIC
<none>               91.51   7.133
- X4    1      2.80  94.31   8.142
- X6    1      3.67  95.18   9.062
- X2    1     14.39 105.91  19.739
- X5    1     19.91 111.43  24.820
- X3    1     22.98 114.50  27.536
- X10   1     26.31 117.82  30.399
- X1    1    443.72 535.23 181.753
```

Example 1

Forward

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

```
Step:  AIC=7.13
y ~ X1 + X2 + X4 + X10 + X6 + X3 + X5

       Df Sum of Sq    RSS    AIC
<none>              91.515 7.1332
+ X8    1   1.61889 89.896 7.3484
+ X7    1   0.99180 90.523 8.0435
+ X12   1   0.32143 91.194 8.7813
+ X15   1   0.21962 91.295 8.8929
+ X14   1   0.11046 91.404 9.0124
+ X9    1   0.05046 91.464 9.0780
+ X13   1   0.02164 91.493 9.1096
+ X11   1   0.00001 91.515 9.1332
```

Example 1

Stepwise from null

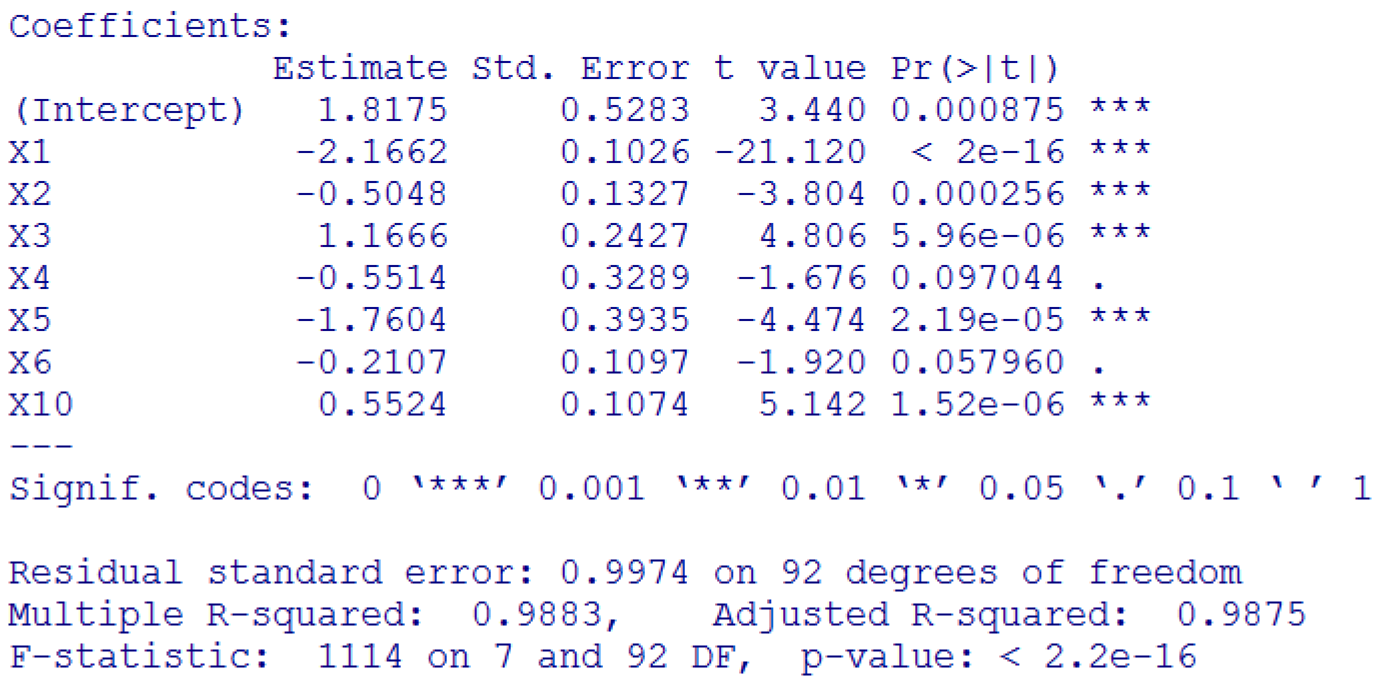
```
mfull<-lm(y~.,data=simex62)
mnull<-lm(y~1,data=simex62)
step(mnull,
     scope=list(lower=mnull,
         upper=mfull),
         direction=both')
```

```
Step:  AIC=7.13
y ~ X1 + X2 + X4 + X10 + X6 + X3 + X5

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

Example 1

Model selected by AIC

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

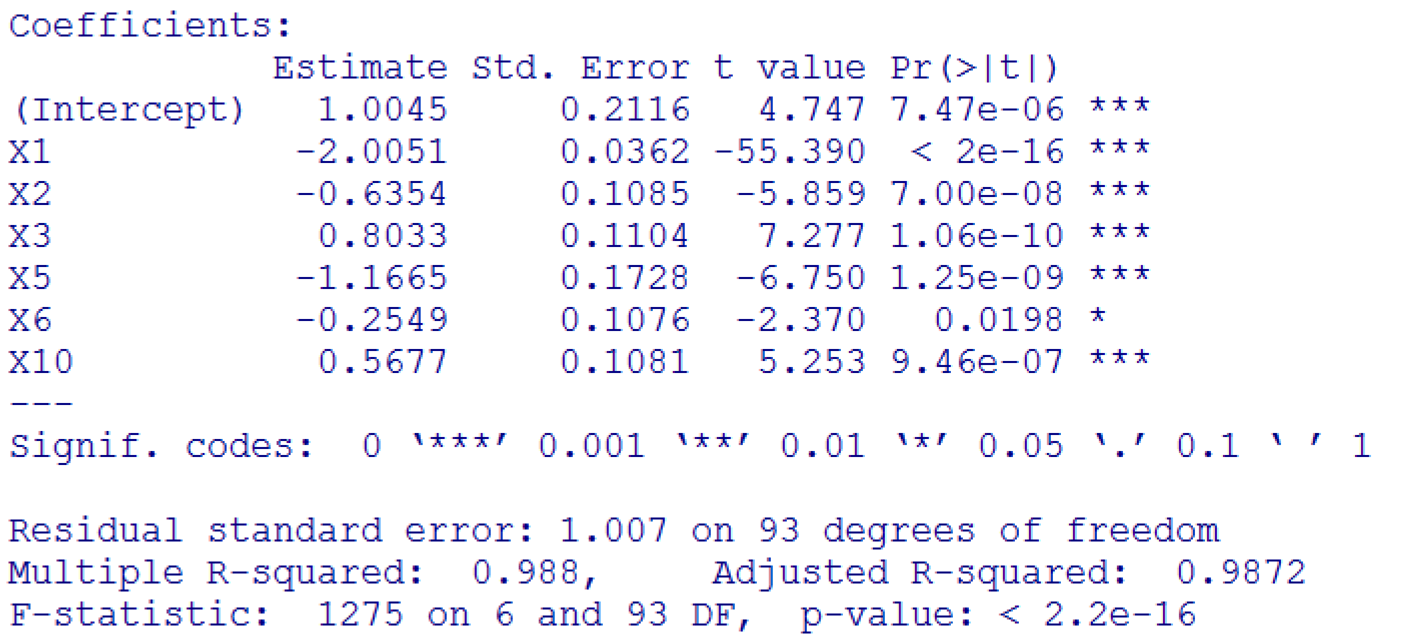


```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.8175     0.5283   3.440 0.000875 ***
X1           -2.1662     0.1026 -21.120  < 2e-16 ***
X2           -0.5048     0.1327  -3.804 0.000256 ***
X3            1.1666     0.2427   4.806 5.96e-06 ***
X4           -0.5514     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

```
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

```
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')
```

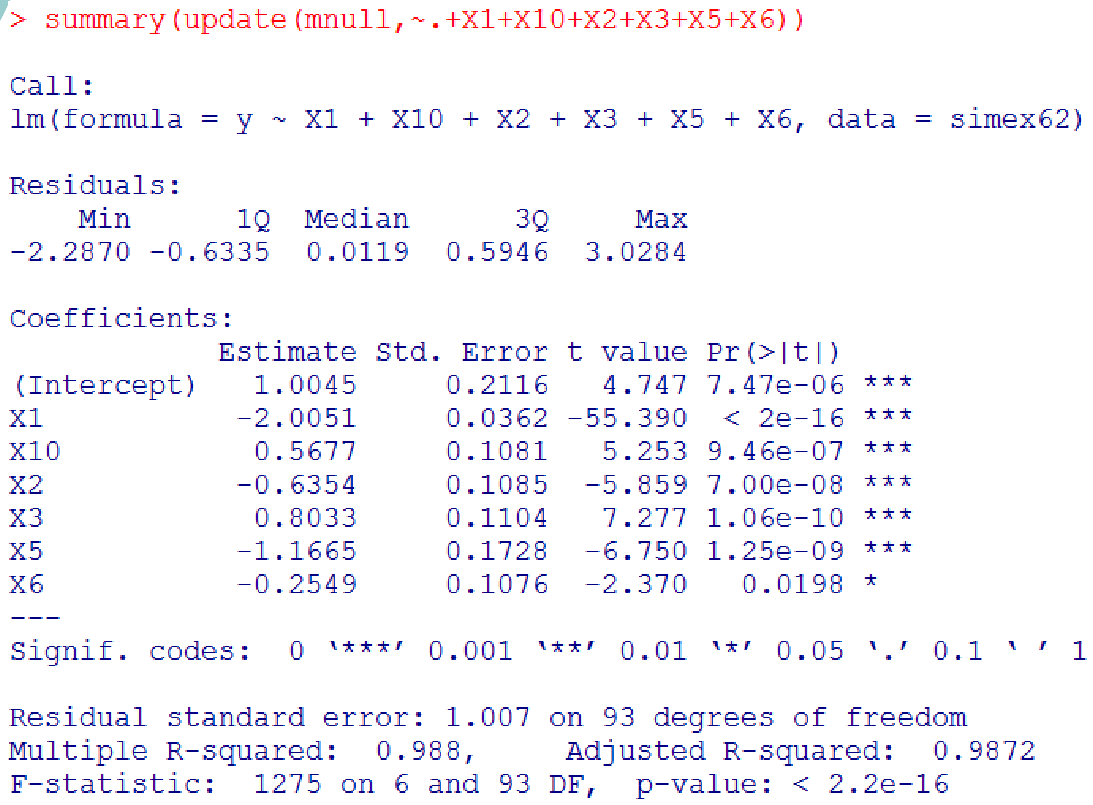
```
> add1(mnull, scope=mfull, test='F')
Single term additions

Model:
y ~ 1
       Df Sum of Sq    RSS    AIC   F value     Pr(>F)
<none>              7849.5 438.30
X1      1    7566.4  283.1 108.06 2619.3063 < 2.2e-16 ***
X2      1     420.0 7429.4 434.80    5.5407  0.020575 *
X3      1     665.1 7184.4 431.45    9.0723  0.003302 **
X4      1    2025.6 5823.8 410.45   34.0865 6.850e-08 ***
X5      1    3214.2 4635.2 387.63   67.9566 7.645e-13 ***
X6      1      84.9 7764.5 439.22    1.0718  0.303080
X7      1      50.3 7799.2 439.66    0.6317  0.428645
X8      1      97.3 7752.2 439.06    1.2299  0.270132
X9      1       8.4 7841.1 440.20    0.1050  0.746647
X10     1      10.4 7839.1 440.17    0.1299  0.719329
X11     1      55.4 7794.0 439.59    0.6971  0.405778
X12     1      55.3 7794.1 439.60    0.6956  0.406291
X13     1     230.5 7618.9 437.32    2.9651  0.088232 .
X14     1     137.8 7711.6 438.53    1.7517  0.188747
X15     1       3.9 7845.5 440.25    0.0491  0.825026
```

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

Model:
y ~ X1 + X10 + X2 + X3 + X5 + X6
       Df Sum of Sq    RSS     AIC F value   Pr(>F)
<none>              94.311  8.1424
X4      1   2.79568 91.515  7.1332  2.8105 0.09704 .
X7      1   0.54355 93.767  9.5643  0.5333 0.46707
X8      1   0.92854 93.382  9.1529  0.9148 0.34135
X9      1   0.01691 94.294 10.1244  0.0165 0.89808
X11     1   0.00041 94.310 10.1419  0.0004 0.98400
X12     1   0.19585 94.115  9.9345  0.1914 0.66274
X13     1   0.00508 94.306 10.1370  0.0050 0.94401
X14     1   0.02482 94.286 10.1160  0.0242 0.87667
X15     1   0.28497 94.026  9.8397  0.2788 0.59874
```

Example 1

Manual forward using F-tests and add1 function

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



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

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

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

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 ***
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

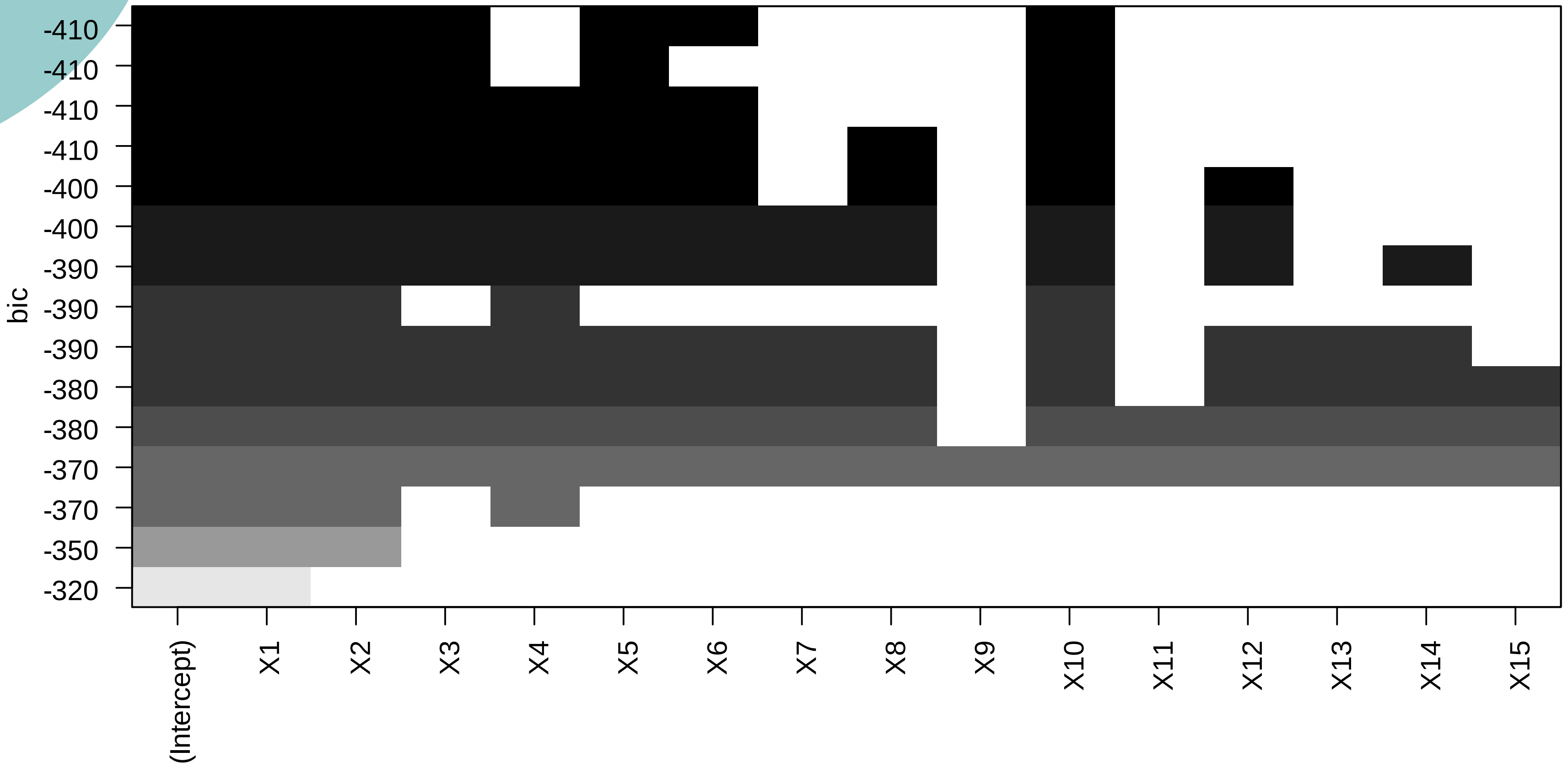
Several measures

```
> 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)
[1] 16.00000 19.90274
> 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

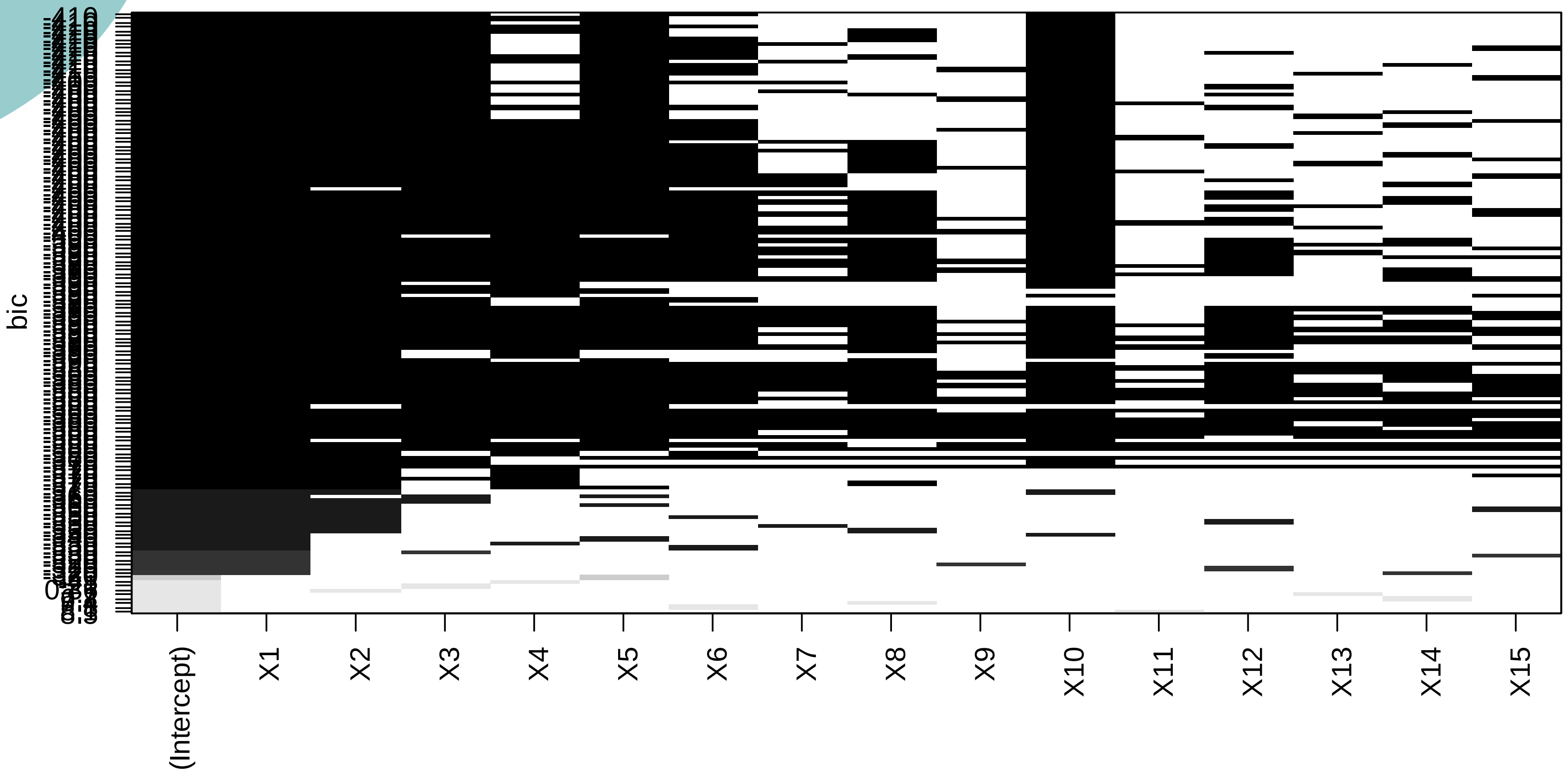
```
plot(regsubsets(y~.,data=simex62, nvmax=15, nbest=1))
```



Example 1

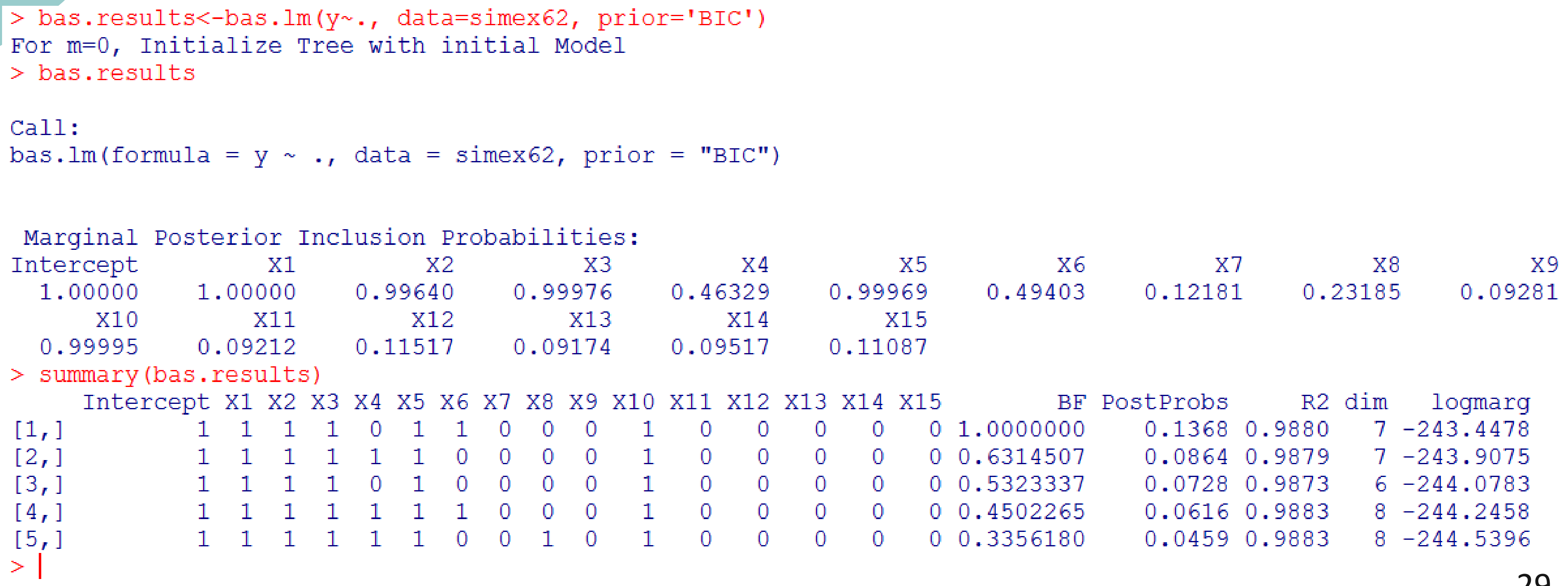
Leaps: selects the 10 best models in every dimension according to BIC

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



Example 1

- BAS: Full enumeration of the model space using BIC.
- Inclusion probability=> rescaled weight measure for including each term.
- Postprobs => posterior probability of each model.

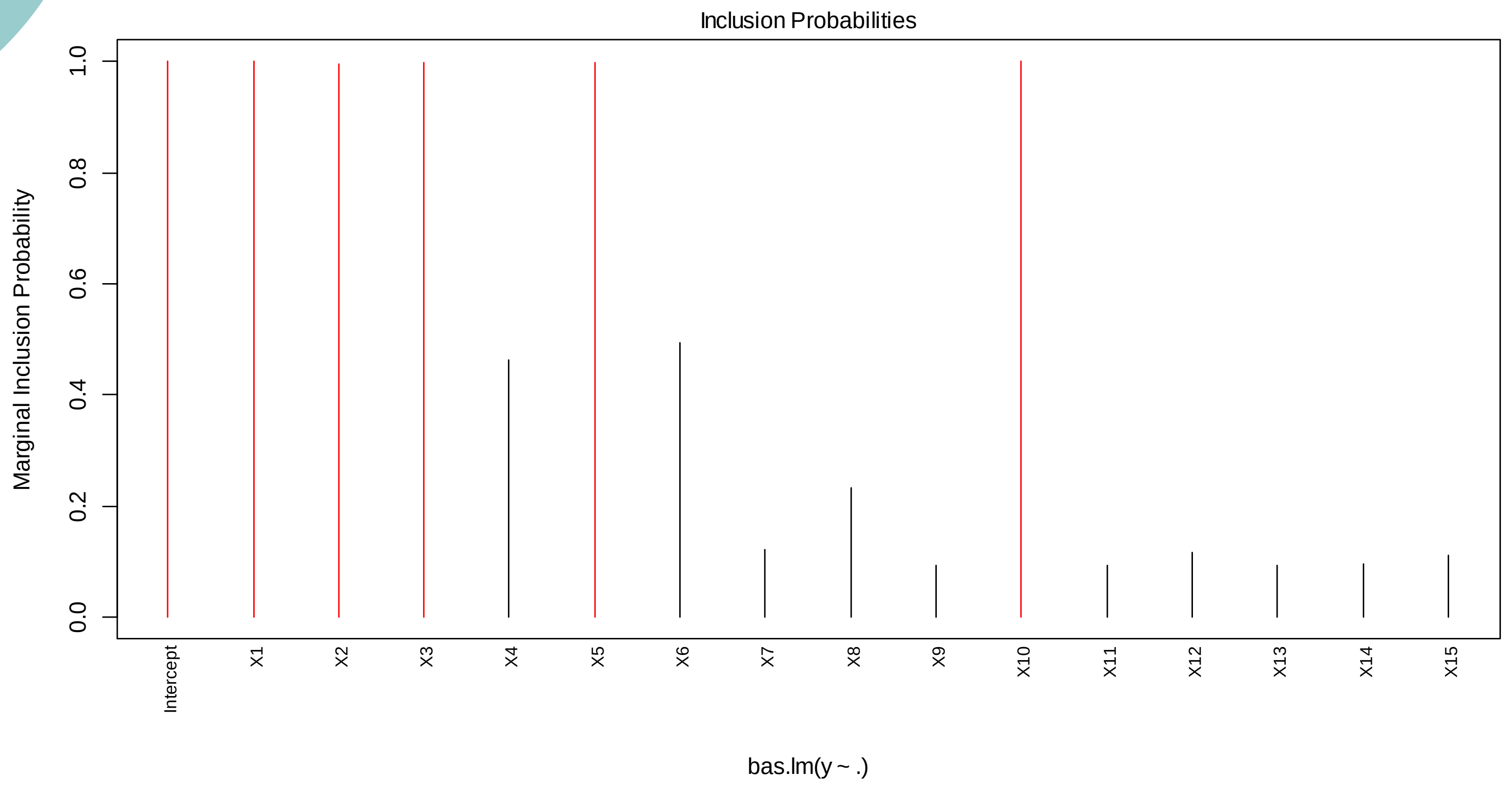


```
> bas.results<-bas.lm(y~., data=simex62, prior='BIC')
For m=0, Initialize Tree with initial Model
> bas.results

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


 Marginal Posterior Inclusion Probabilities:
Intercept         X1         X2         X3         X4         X5         X6         X7         X8         X9
  1.00000    1.00000    0.99640    0.99976    0.46329    0.99969    0.49403    0.12181    0.23185    0.09281
      X10        X11        X12        X13        X14        X15
  0.99995    0.09212    0.11517    0.09174    0.09517    0.11087
> 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  0  1  1  0  0  0   1   0   0   0   0   0 1.0000000    0.1368 0.9880   7 -243.4478
[2,]         1  1  1  1  1  1  0  0  0  0   1   0   0   0   0   0 0.6314507    0.0864 0.9879   7 -243.9075
[3,]         1  1  1  1  0  1  0  0  0  0   1   0   0   0   0   0 0.5323337    0.0728 0.9873   6 -244.0783
[4,]         1  1  1  1  1  1  1  0  0  0   1   0   0   0   0   0 0.4502265    0.0616 0.9883   8 -244.2458
[5,]         1  1  1  1  1  1  0  0  1  0   1   0   0   0   0   0 0.3356180    0.0459 0.9883   8 -244.5396
> |
```

Example 1

BAS: Posterior inclusion probabilities under BIC

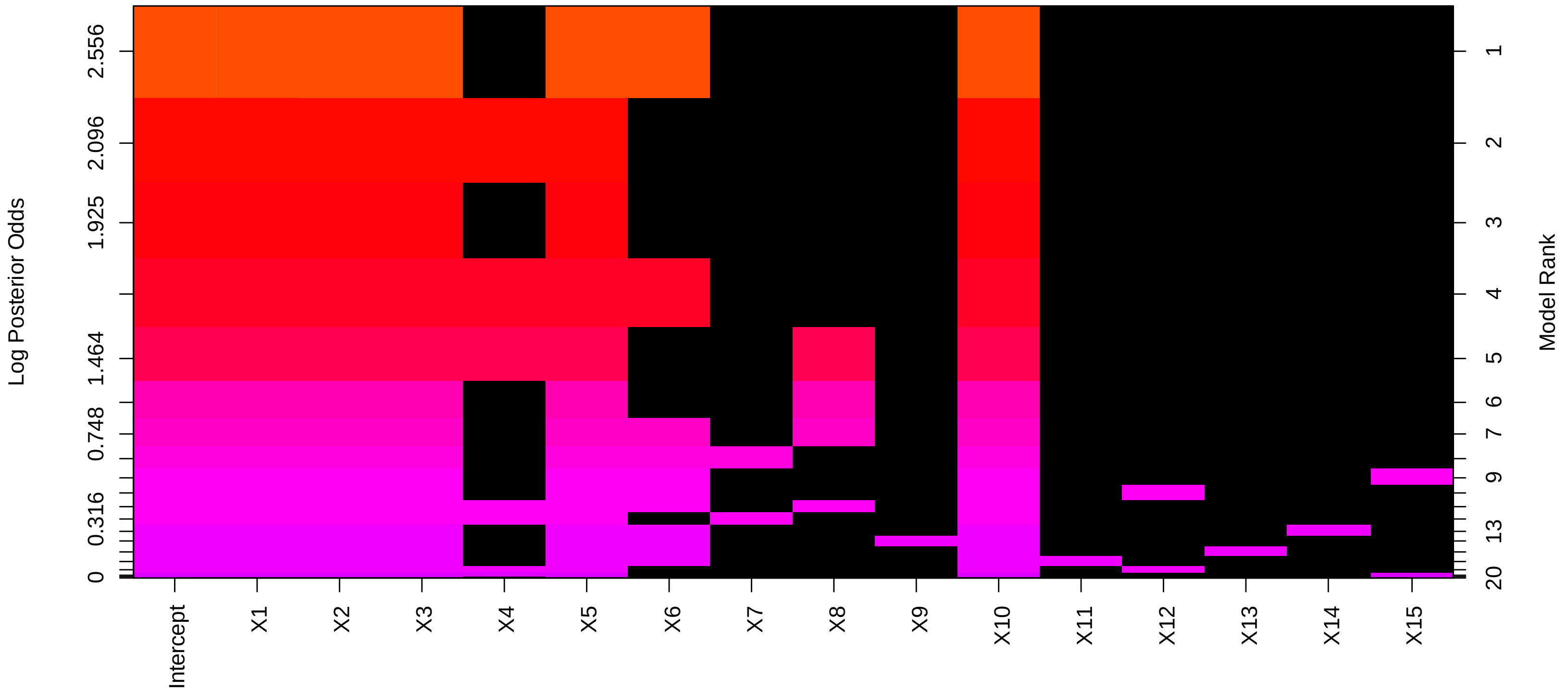
plot(bas.results)



Example 1

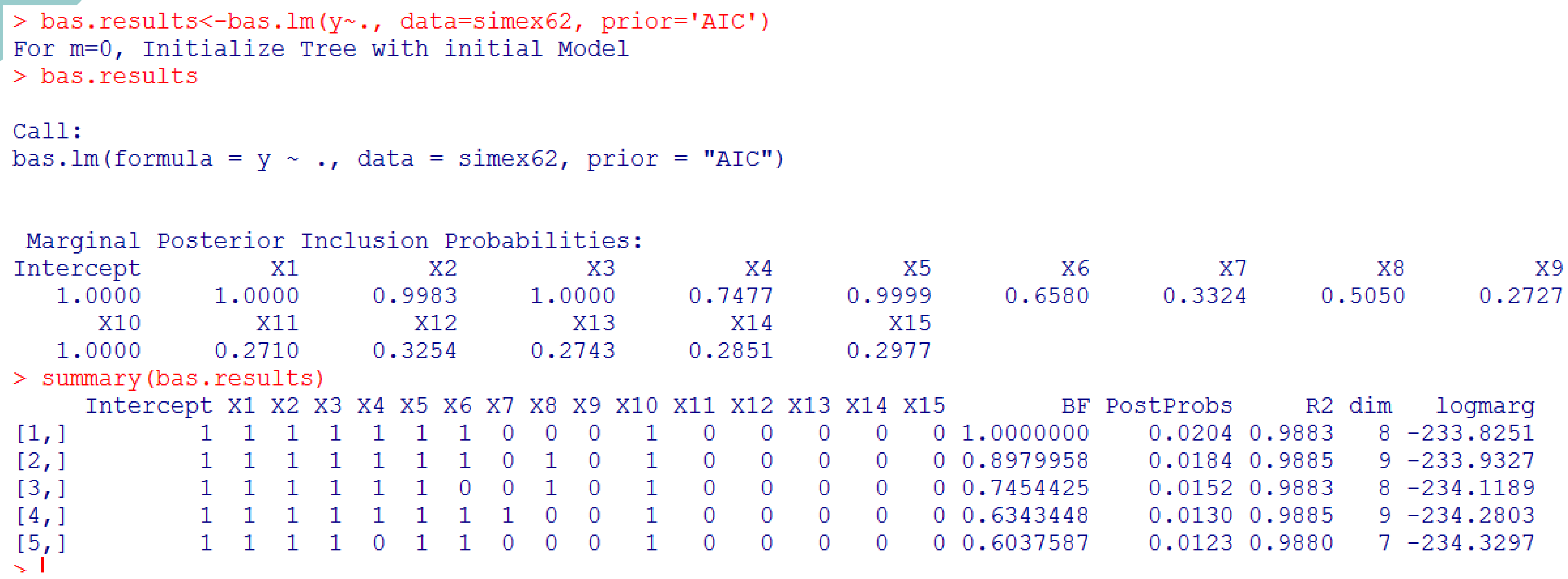
BAS: Posterior model probabilities of best 20 and included vars using BIC

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

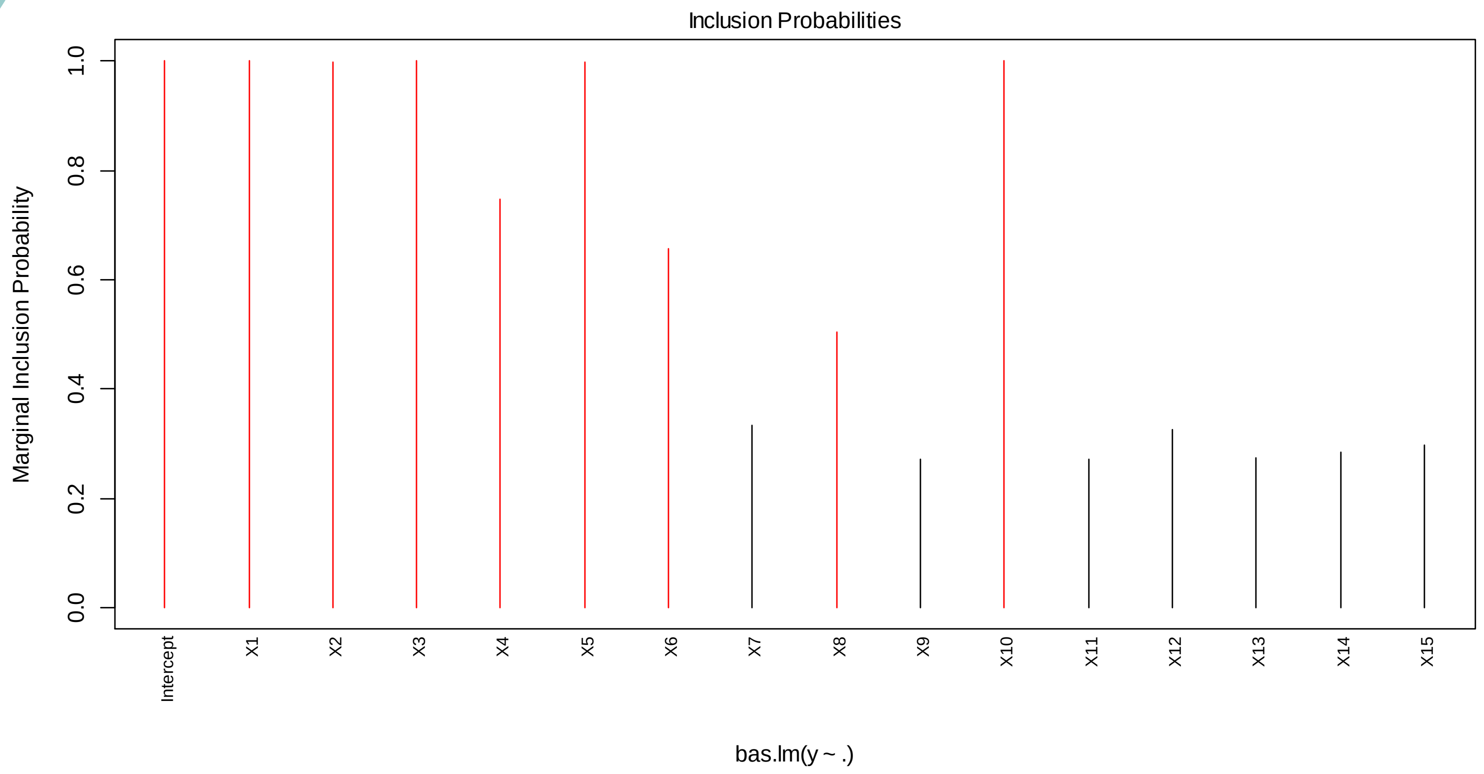


```
> bas.results<-bas.lm(y~., data=simex62, prior='AIC')
For m=0, Initialize Tree with initial Model
> bas.results

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


 Marginal Posterior Inclusion Probabilities:
Intercept         X1         X2         X3         X4         X5         X6         X7         X8         X9
   1.0000     1.0000     0.9983     1.0000     0.7477     0.9999     0.6580     0.3324     0.5050     0.2727
      X10        X11        X12        X13        X14        X15
   1.0000     0.2710     0.3254     0.2743     0.2851     0.2977
> 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 1.0000000    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  0  0  1  0   1   0   0   0   0   0 0.7454425    0.0152 0.9883   8 -234.1189
[4,]         1  1  1  1  1  1  1  1  0  0   1   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

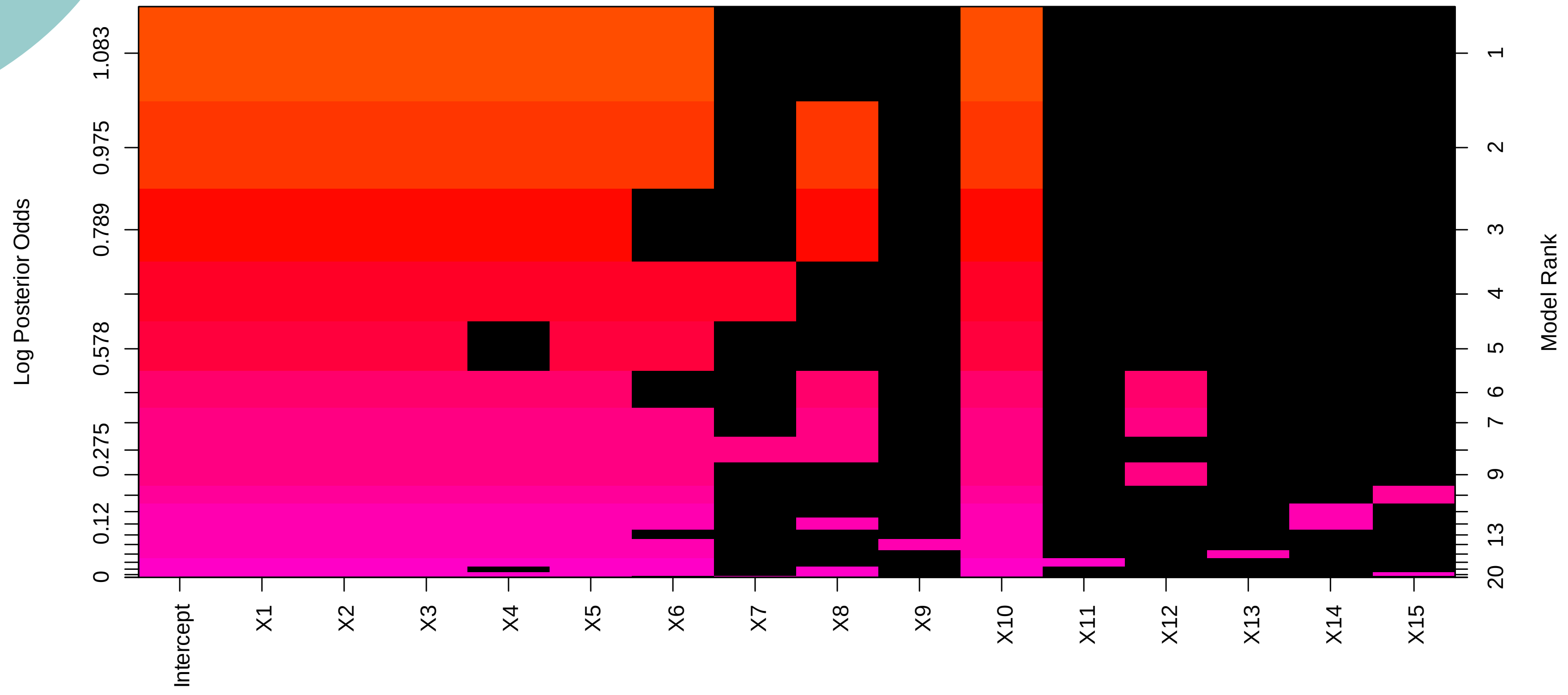
plot(bas.results)



Example 1

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

image(bas.results)



Multi-Collinearity

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
⬄ 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.

Multi-Collinearity

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

$$Y= \beta_0+\beta_1 X_1 + \beta_2 X_2 +\varepsilon$$

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

$$Y= \beta_0+\beta_1 X_1 + \beta_2 (a+bX_1) +\varepsilon$$

$$= (\beta_0 +a \beta_2) + (\beta_1 +\beta_2 b)X_1 +\varepsilon$$

Which is the correct effect of X_1?

Multi-Collinearity

Why multi-collinearity is a problem?

Mathematical explanation

$$\hat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- $\hat{\beta} = (\hat{\beta}_0, \hat{\beta}_1, \ldots, \hat{\beta}_p)^T$ is the vector of the OLS estimators (or MLEs) of dimension (p+1)x1.
- **X** is the data or design matrix of dimension n×(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×1 with the values of the response variable.

Multi-Collinearity

Why multi-collinearity is a problem?

Mathematical explanation

$$\widehat{\beta} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

- **Problem**: If a variable (i.e. a column of the data matrix **X**) is a linear combination of the rest the inverse $(\mathbf{X}^T\mathbf{X})^{-1}$ 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.

Multi-Collinearity

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 Xs.
   - ✓ If $VIF_j > 10$ $[R_j^2 > 0.90]$ then we have a potential collinearity problem.

Multi-Collinearity

Variance inflation factors

$$\widehat{Var}(\widehat{\beta}) = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\widehat{\sigma}^2$$

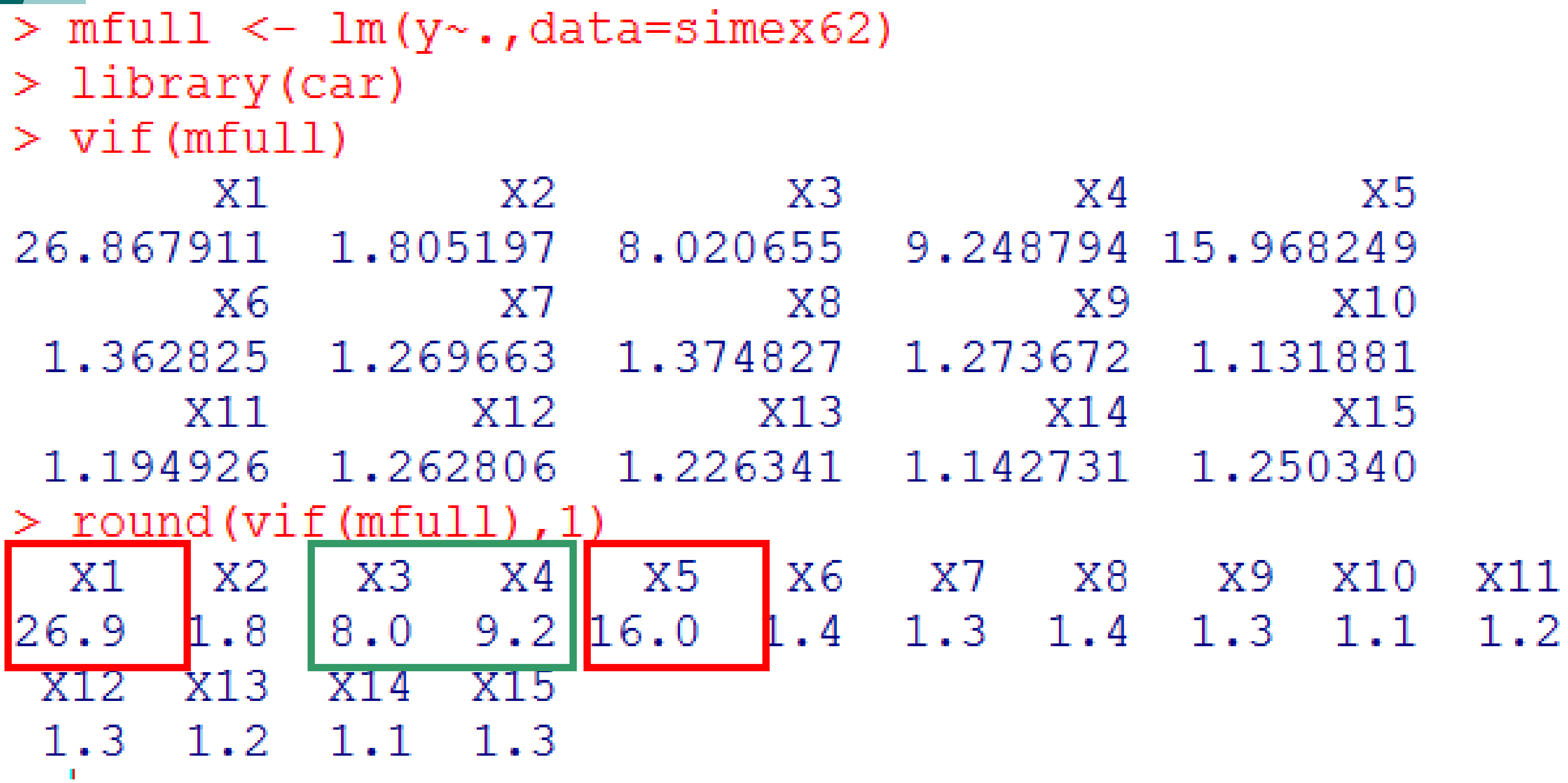
$$\widehat{Var}(\widehat{\beta}_j) = \frac{\widehat{\sigma}^2}{(n-1)S_{X_j}^2} \times \frac{1}{1-R_j^2}$$

VIFs are is 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”



```
> mfull <- lm(y~.,data=simex62)
> library(car)
> vif(mfull)
       x1        x2        x3        x4        x5 
26.867911  1.805197  8.020655  9.248794 15.968249 
       x6        x7        x8        x9       x10 
 1.362825  1.269663  1.374827  1.273672  1.131881 
      x11       x12       x13       x14       x15 
 1.194926  1.262806  1.226341  1.142731  1.250340 
> round(vif(mfull),1)
  x1   x2   x3   x4   x5   x6   x7   x8   x9  x10  x11 
26.9  1.8  8.0  9.2 16.0  1.4  1.3  1.4  1.3  1.1  1.2 
 x12  x13  x14  x15 
 1.3  1.2  1.1  1.3 
```

Multi-Collinearity

Condition indexes

- Calculate the eigenvalues of $\mathbf{X}^T\mathbf{X}$.
- Eigenvalues close to zero indicate a problem.
- Condition Index

  = Square root of MAX(eigenvalues)/ 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

```
> X<-model.matrix(mfull)
> v<-sqrt(eigen(t(X)%*%X)$value)
> max(v)/v
 [1]  1.000000  5.124575  5.451744  5.687871  5.886674  6.131369
 [7]  6.351875  6.465027  7.286807  7.321037  8.030066  8.499335
[13]  8.902555 10.304457 11.109056 54.203670
> colldiag(mfull, scale=F)$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

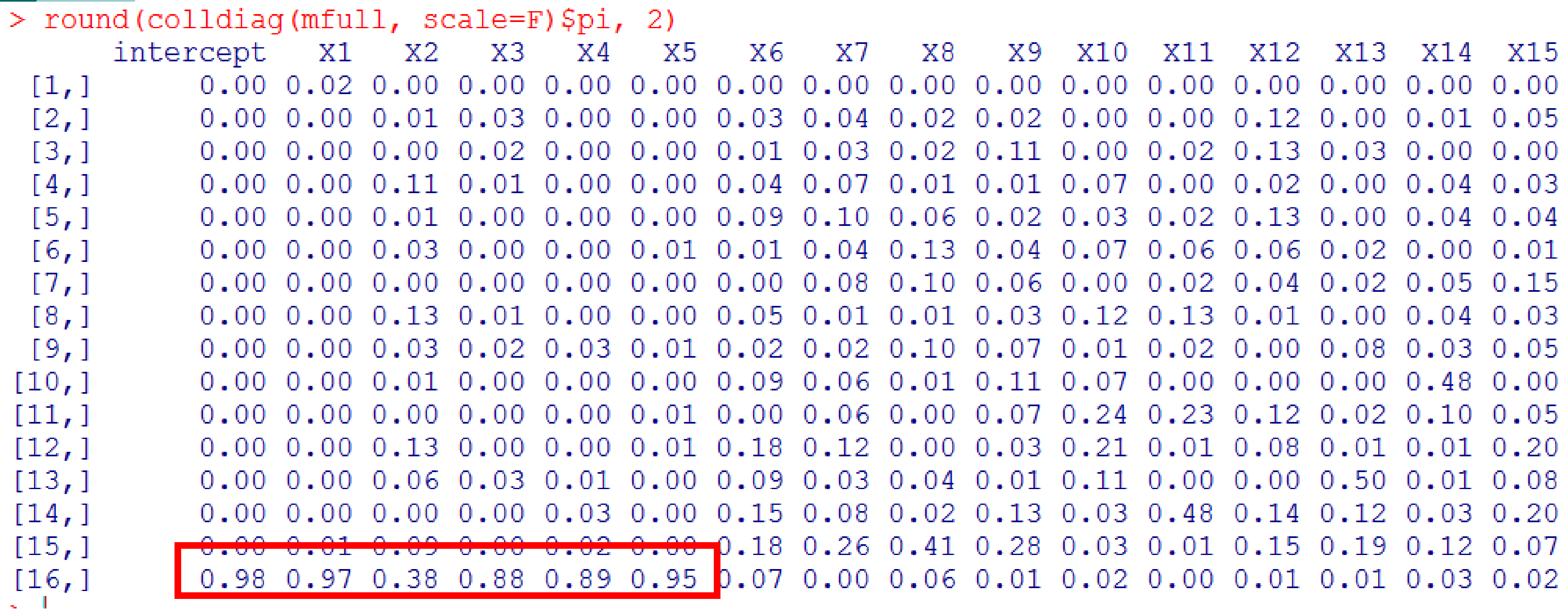
- Is the proportion of Var(β_j) explained by the corresponding component.
- If a large condition index is associated with two or more variables with *large* variance decomposition proportions, these variables may be causing collinearity problems. Belsley et al suggest that a *large* proportion is 50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

Reference: D. Belsley, E. Kuh, and R. Welsch (1980). Regression Diagnostics. Wiley.

2004=> 2nd edition

Multi-Collinearity

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



```
> 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 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 0.12 0.00 0.01 0.05
 [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 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 0.03
 [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 0.04
 [6,]      0.00 0.00 0.03 0.00 0.00 0.01 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 0.00 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 0.03
 [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 0.05
[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 0.00 0.48 0.00
[11,]      0.00 0.00 0.00 0.00 0.00 0.01 0.00 0.06 0.00 0.07 0.24 0.23 0.12 0.02 0.10 0.05
[12,]      0.00 0.00 0.13 0.00 0.00 0.01 0.18 0.12 0.00 0.03 0.21 0.01 0.08 0.01 0.01 0.20
[13,]      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 0.50 0.01 0.08
[14,]      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 0.20
[15,]      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 0.07
[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 0.02
```

6Multi-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

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

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

Ridge Regression

We start by standardizing all covariates.

Hence X => Z (matrix of standardized covariates)

$$\text{minimize} \sum_{i=1}^{n} (y_i - \beta^\top \mathbf{z}_i)^2 \text{ s.t. } \sum_{j=1}^{p} \beta_j^2 \le t$$

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

When including an intercept term in the regression, we leave this coefficient unpenalized. If we centered the columns of X then β_0 = mean(y).

Ridge Regression

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)

$$
\begin{aligned}
PRSS(\beta)_{\ell_2} &= \sum_{i=1}^{n}(y_i - \mathbf{z}_i^\top \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \\
&= (\mathbf{y} - \mathbf{Z}\beta)^\top(\mathbf{y} - \mathbf{Z}\beta) + \lambda||\beta||_2^2
\end{aligned}
$$

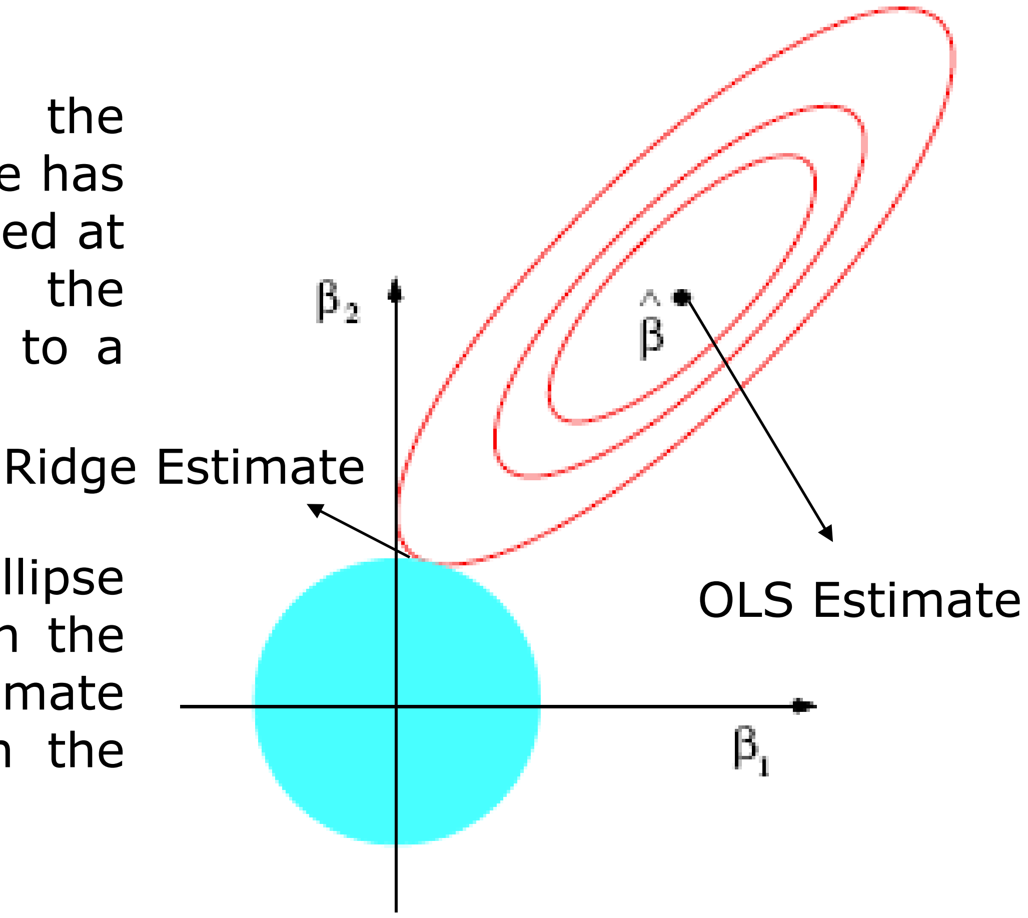
Ridge Regression

Ridge Regression

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 β would give you a better RSS but then it will push the penalty term higher.
- This is why you might actually prefer smaller β's with worse RSS. From an optimization perspective, the penalty term is equivalent to a constraint on the β's. The function is still the RSS but now you constrain the norm of the β_j's to be smaller than some constant t.
- There is a correspondence between λ and t. The larger the λ is, the more you prefer the β_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 λ approaches infinity, you set all the β's to zero.

Ridge Regression

The ridge solution

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

$$\hat{\beta}_{\lambda}^{\text{ridge}} = (\mathbf{Z}^\top \mathbf{Z} + \lambda \mathbf{I}_p)^{-1} \mathbf{Z}^\top \mathbf{y}$$

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

Ridge Regression

The data augmentation interpretation of the ridge sol.

$$\hat{\beta}_{\lambda}^{\text{ridge}} = (\mathbf{Z}^{\top}\mathbf{Z} + \lambda \mathbf{I}_p)^{-1}\mathbf{Z}^{\top}\mathbf{y}$$

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

$$\begin{aligned} PRSS(\beta)_{\ell_2} &= \sum_{i=1}^{n}(y_i - \mathbf{z}_i^{\top}\beta)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 \\ &= \sum_{i=1}^{n}(y_i - \mathbf{z}_i^{\top}\beta)^2 + \sum_{j=1}^{p}(0 - \sqrt{\lambda}\beta_j)^2 \end{aligned}$$

Ridge Regression

The ridge estimators are biased since

$$\hat{\beta}_\lambda^{\text{ridge}} = (\mathbf{I}_p + \lambda\mathbf{R}^{-1})\hat{\beta}^{\text{ls}}$$

where $\mathbf{R} = \mathbf{Z}^\top\mathbf{Z}$

$$\begin{aligned}\mathbb{E}(\hat{\beta}_\lambda^{\text{ridge}}) &= \mathbb{E}\{(\mathbf{I}_p + \lambda\mathbf{R}^{-1})\hat{\beta}^{\text{ls}}\} \\ &= (\mathbf{I}_p + \lambda\mathbf{R}^{-1})\beta \\ &\overset{(\text{if } \lambda\neq 0)}{\neq} \beta.\end{aligned}$$

Which means that the ridge estimators are biased for any λ>0

Ridge Regression

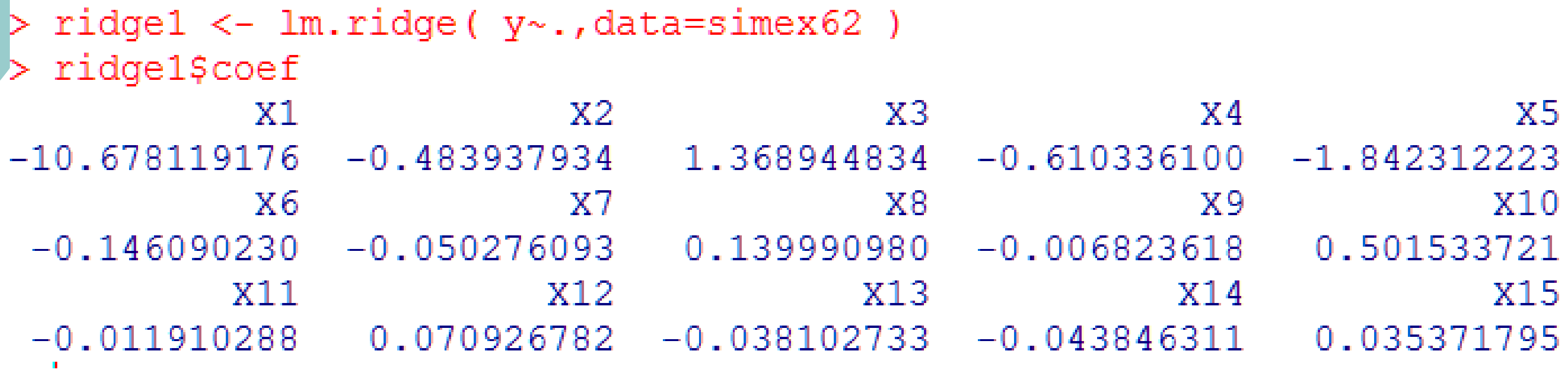
Main Problem: The selection of λ

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

Ridge Regression

An example using lm.ridge in MASS package

λ=0 if no value is specified => provides the OLS estimators and model

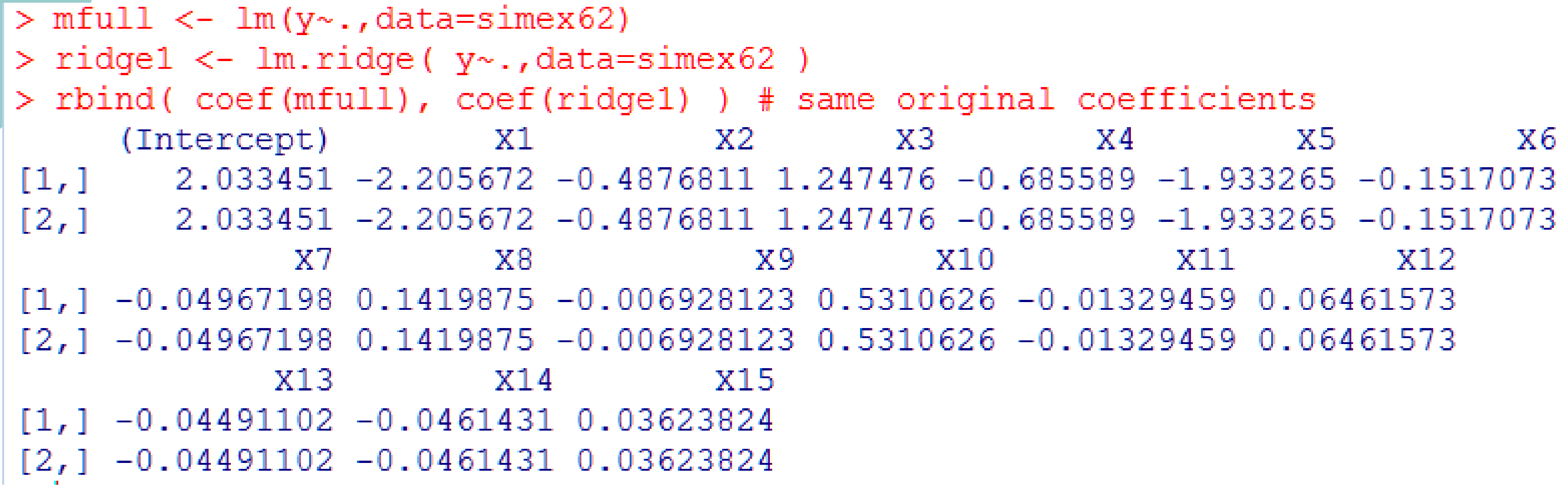


```
> ridge1 <- lm.ridge( y~.,data=simex62 )
> ridge1$coef
         x1           x2           x3           x4           x5
-10.678119176 -0.483937934  1.368944834 -0.610336100 -1.842312223
         x6           x7           x8           x9          x10
 -0.146090230 -0.050276093  0.139990980 -0.006823618  0.501533721
        x11          x12          x13          x14          x15
 -0.011910288  0.070926782 -0.038102733 -0.043846311  0.035371795
```

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, λ=0 so these are the usual OLS for standardized covariates.

Ridge Regression

An example using lm.ridge in MASS package

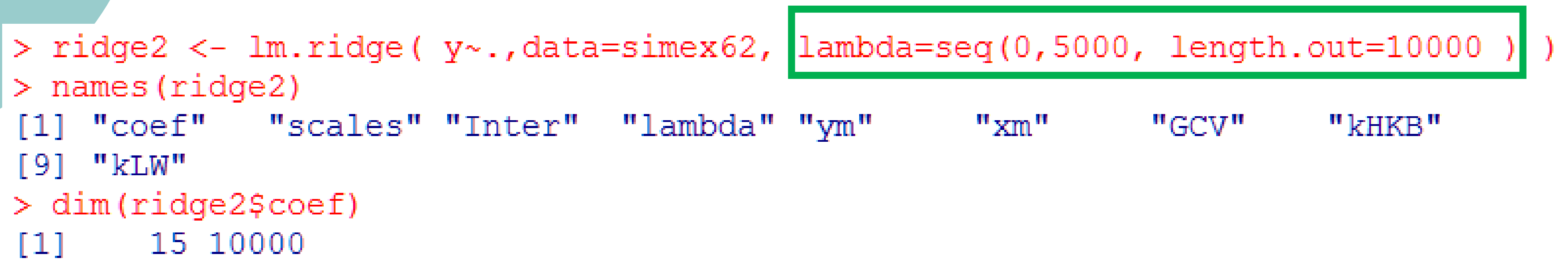


```
> mfull <- lm(y~.,data=simex62)
> ridge1 <- lm.ridge( y~.,data=simex62 )
> rbind( coef(mfull), coef(ridge1) ) # same original coefficients
     (Intercept)        X1         X2       X3        X4        X5         X6
[1,]    2.033451 -2.205672 -0.4876811 1.247476 -0.685589 -1.933265 -0.1517073
[2,]    2.033451 -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
```

We use coef(ridge1) to obtain the coefficients for the original data.
Here, λ=0 so these are the usual OLS for the original data

Ridge Regression

An example using lm.ridge in MASS package

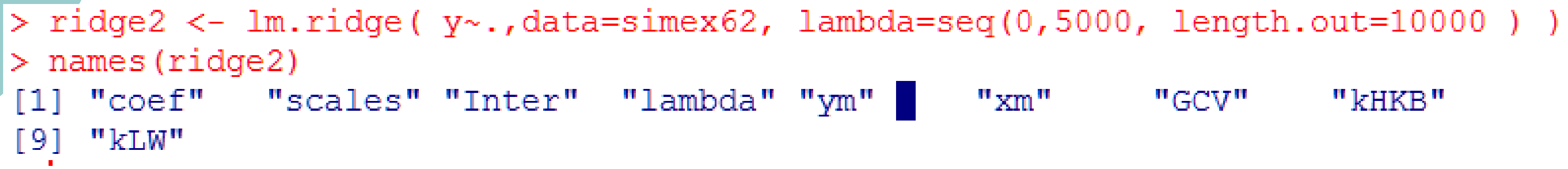


```
> ridge2 <- lm.ridge( y~.,data=simex62, lambda=seq(0,5000, length.out=10000 ) )
> names(ridge2)
[1] "coef"    "scales" "Inter"   "lambda" "ym"      "xm"      "GCV"     "kHKB"
[9] "kLW"
> dim(ridge2$coef)
[1]    15 10000
```

Coef : The coefficients are in a matrix of dimension p x length(lambda).
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



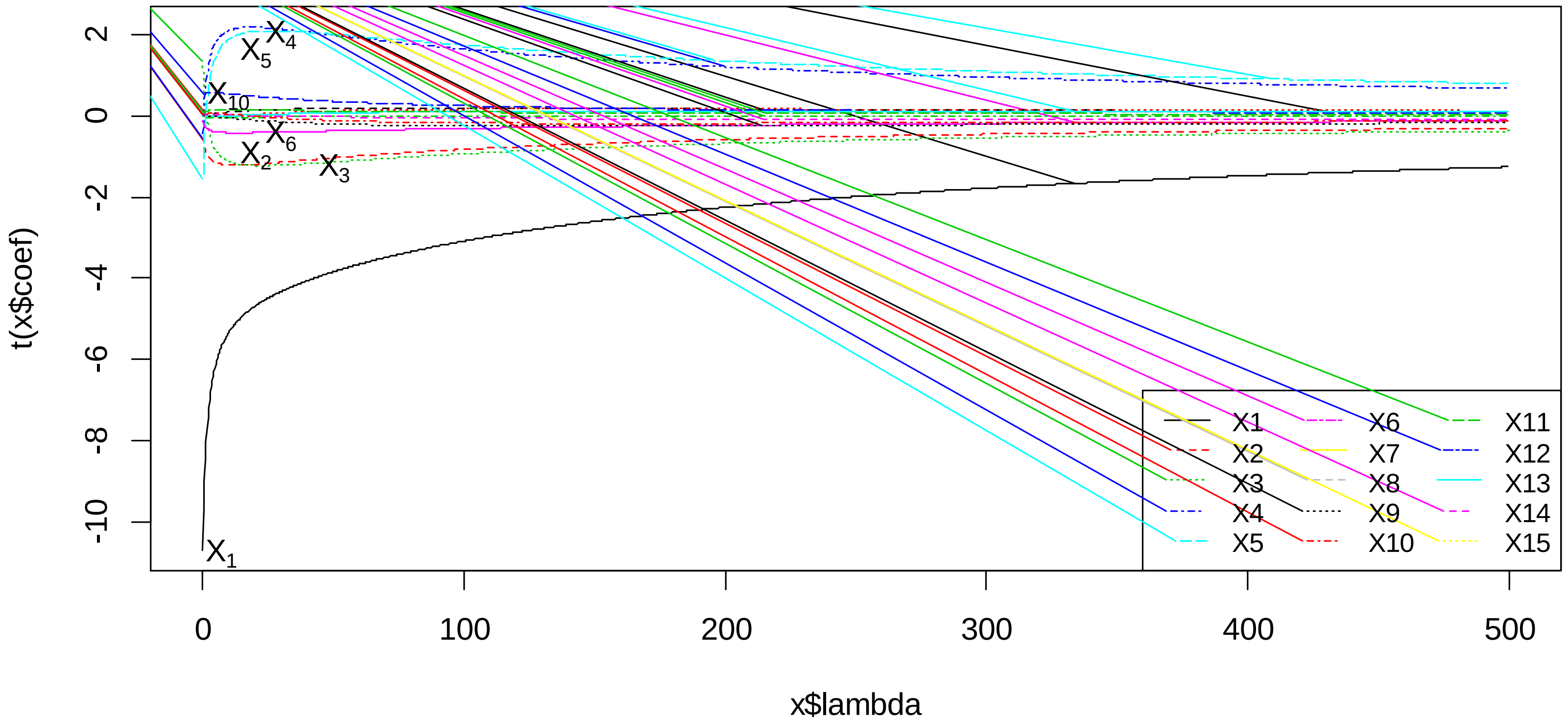
```
> ridge2 <- lm.ridge( y~.,data=simex62, lambda=seq(0,5000, length.out=10000 ) )
> names(ridge2)
[1] "coef"    "scales" "Inter"  "lambda" "ym"     "xm"      "GCV"     "kHKB"
[9] "kLW"
```

- ✓ **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 λ 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

```
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:

$$\widehat{\mathbf{y}} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} = \mathbf{H}\mathbf{y}$$

Hence the hat matrix is defined as $\mathbf{H} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$

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.

$$p' = rank(\mathbf{H}) = trace(\mathbf{H})$$

so p′ are the number of degrees of freedom used by the model to estimate the parameters

Ridge Regression

The effective degrees of freedom

In ridge regression:

$$\widehat{y}^{ridge} = \mathbf{Z}\left(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}_p\right)^{-1}\mathbf{Z}^T\mathbf{y} = \mathbf{H}^{ridge}\mathbf{y}$$

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

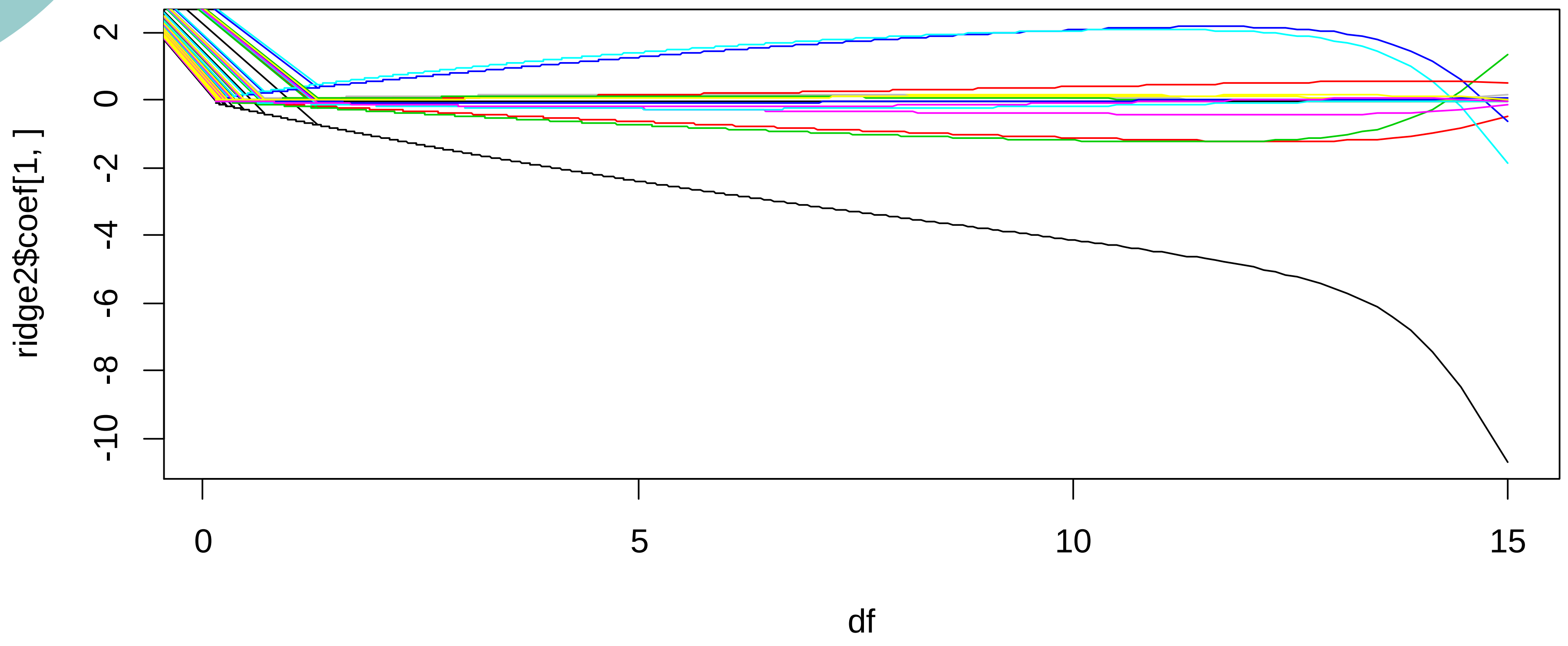
In analogy to OLS, the number of effectively estimated parameters (effective degrees of freedom) is given by the rank of the hat matrix i.e.

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

where d_j^2 are the eigenvalues of matrix X^TX

Ridge Regression

The regularization plot using the effective degrees of freedom



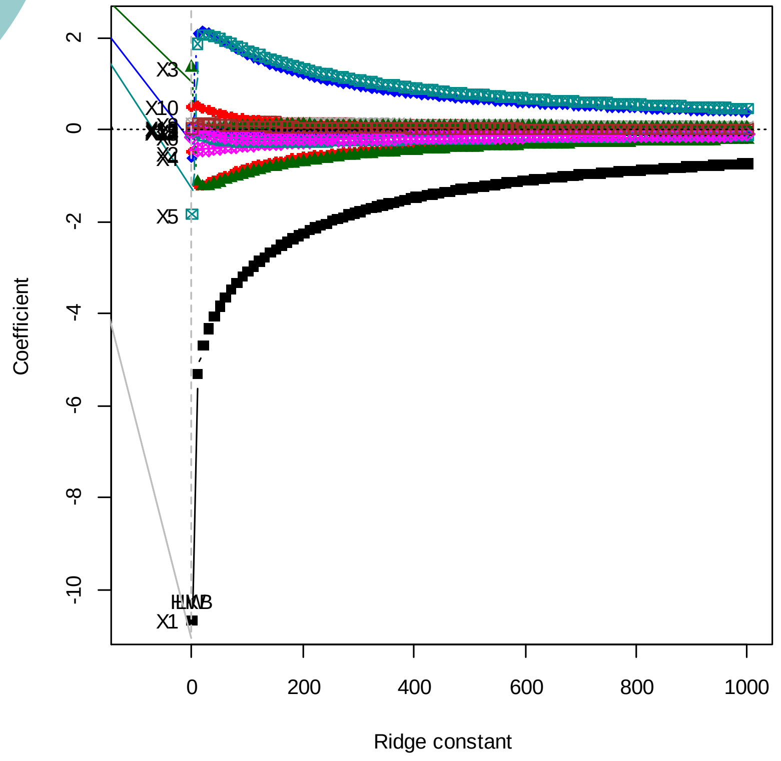
Ridge Regression

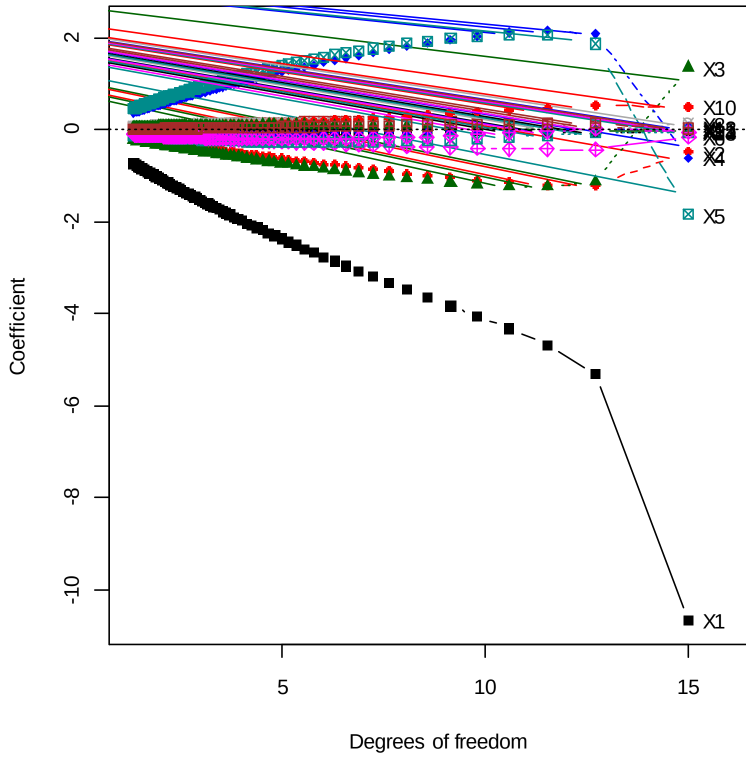
The regularization plot using the effective degrees of freedom: *The R-code*

```
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)
```

Ridge Regression

The regularization plots using the "genridge" library





Ridge Regression

The regularization plots using the "genridge" library

The R-code

```
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 λ

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

Ridge Regression

Selection of λ using AIC, BIC and effective dfs

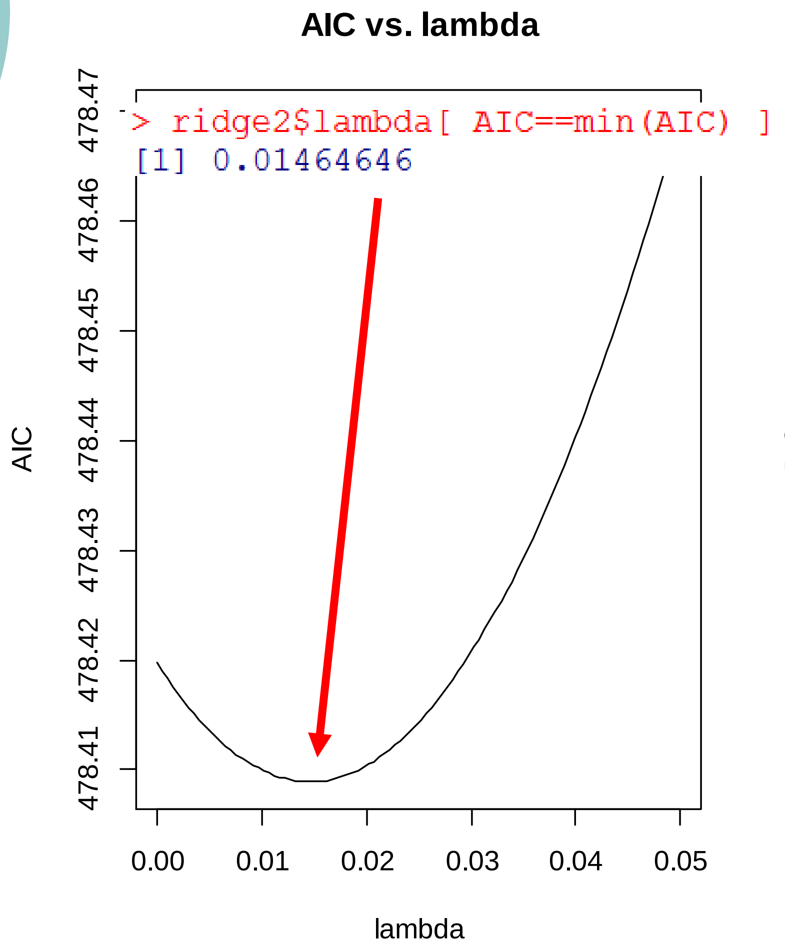
Select λ which minimize the AIC or BIC

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

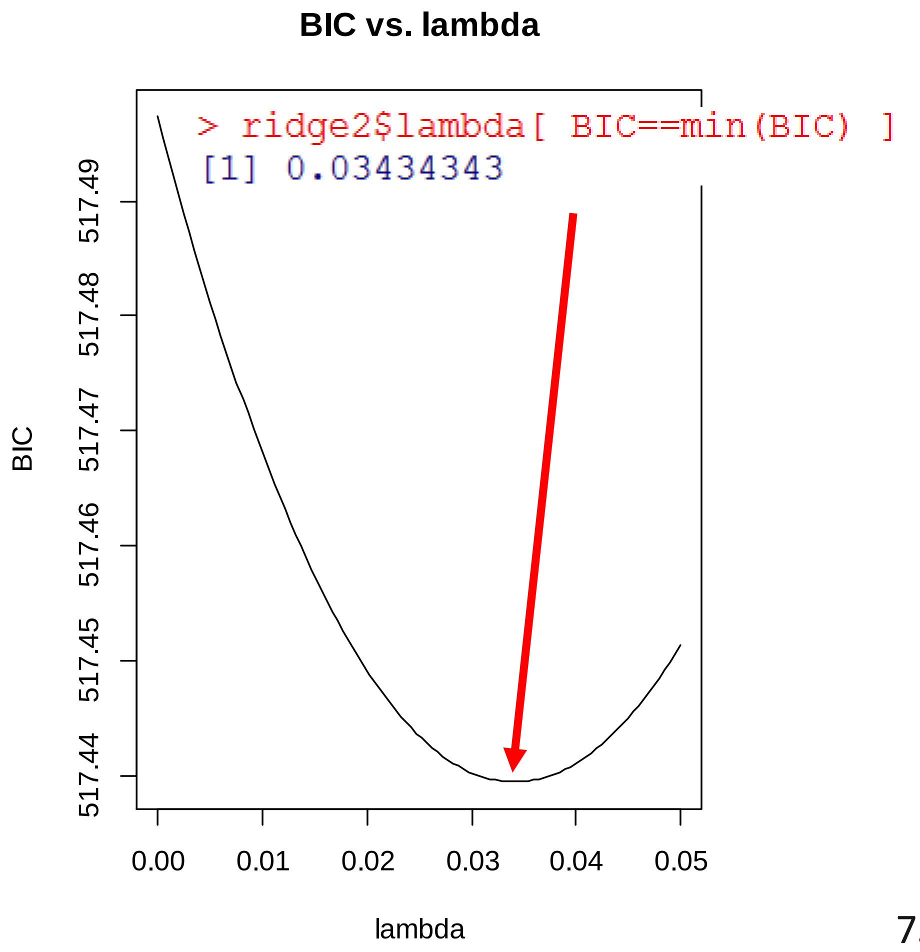
Where df is the effective degrees of freedom

Ridge Regression

Plots of AIC and BIC



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



```
> ridge2$lambda[ BIC==min(BIC) ]
[1] 0.03434343
```

Ridge Regression

```
#-----------------------------------------------------------------------------
#-----------------------------------------------------------------------------
#--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 λ

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

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

$\hat{\sigma}^2, \hat{\beta}$ 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")

Ridge Regression

How to select λ

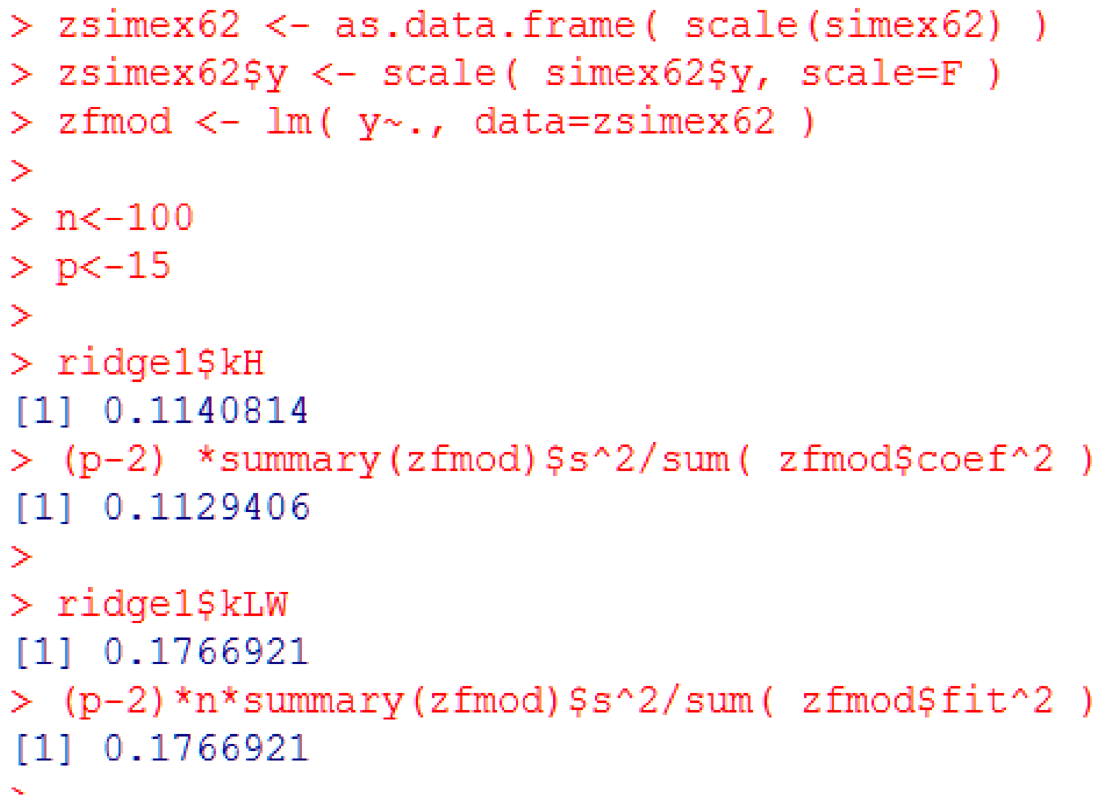
Lawless & Wang (1976, Comm.Stats) proposed a slightly modified estimator of $\lambda = k_{LW}$ given by

$$k_{LW} = \frac{p\widehat{\sigma}^2}{\widehat{\beta}^T(\mathbf{X}^T\mathbf{X})\widehat{\beta}} = \frac{p\widehat{\sigma}^2}{\widehat{\mathbf{y}}^T\widehat{\mathbf{y}}}$$

Ridge Regression

How to select λ

The criteria in R are slightly modified



```
> 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
```

Ridge Regression

How to select λ 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

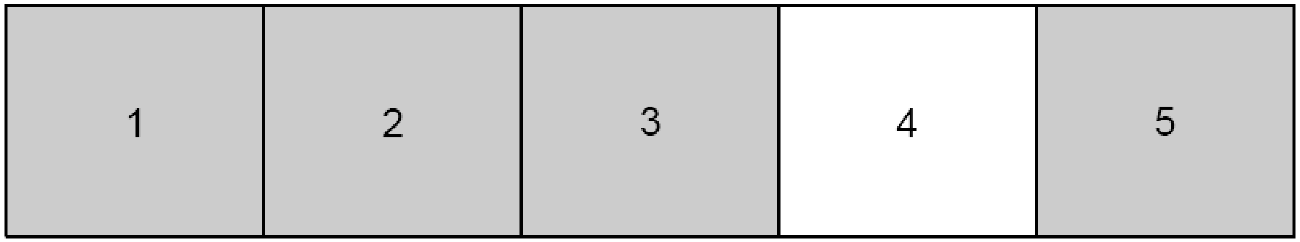
Problems:

- *Not a lot of data.*
- *How to split them? (different splits provide different solutions)*
- What size shall we use for training and testing?

Ridge Regression

How to select λ 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*

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

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

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

Select λ with the minimum AMSE or ARMSE

$$AMSE = \overline{MSE} = \frac{1}{K} \sum_{k=1}^{K} MSE(T_k) \qquad ARMSE = \overline{RMSE} = \frac{1}{K} \sqrt{\sum_{k=1}^{K} MSE(T_k)}$$

Ridge Regression

Mean Square error for CV(1) & GCV

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

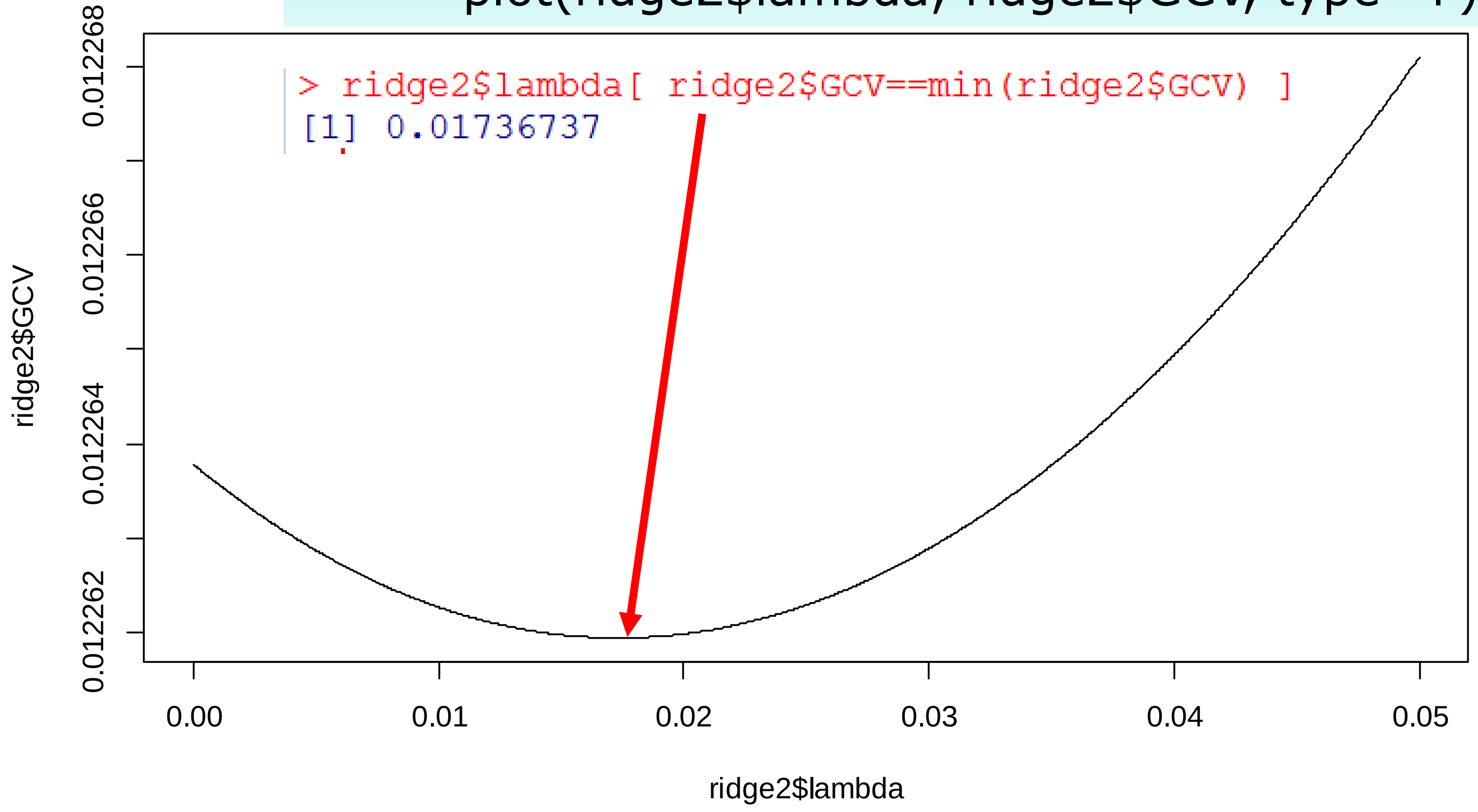
The generalized CV is approximately equal to the MSE obtained using CV(1), but much easier to compute

$$MSE_{CV(1)} \approx GCV = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{y}_i}{1 - \frac{Tr(H)}{n}}\right)^2$$

Ridge Regression

GCV in R

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

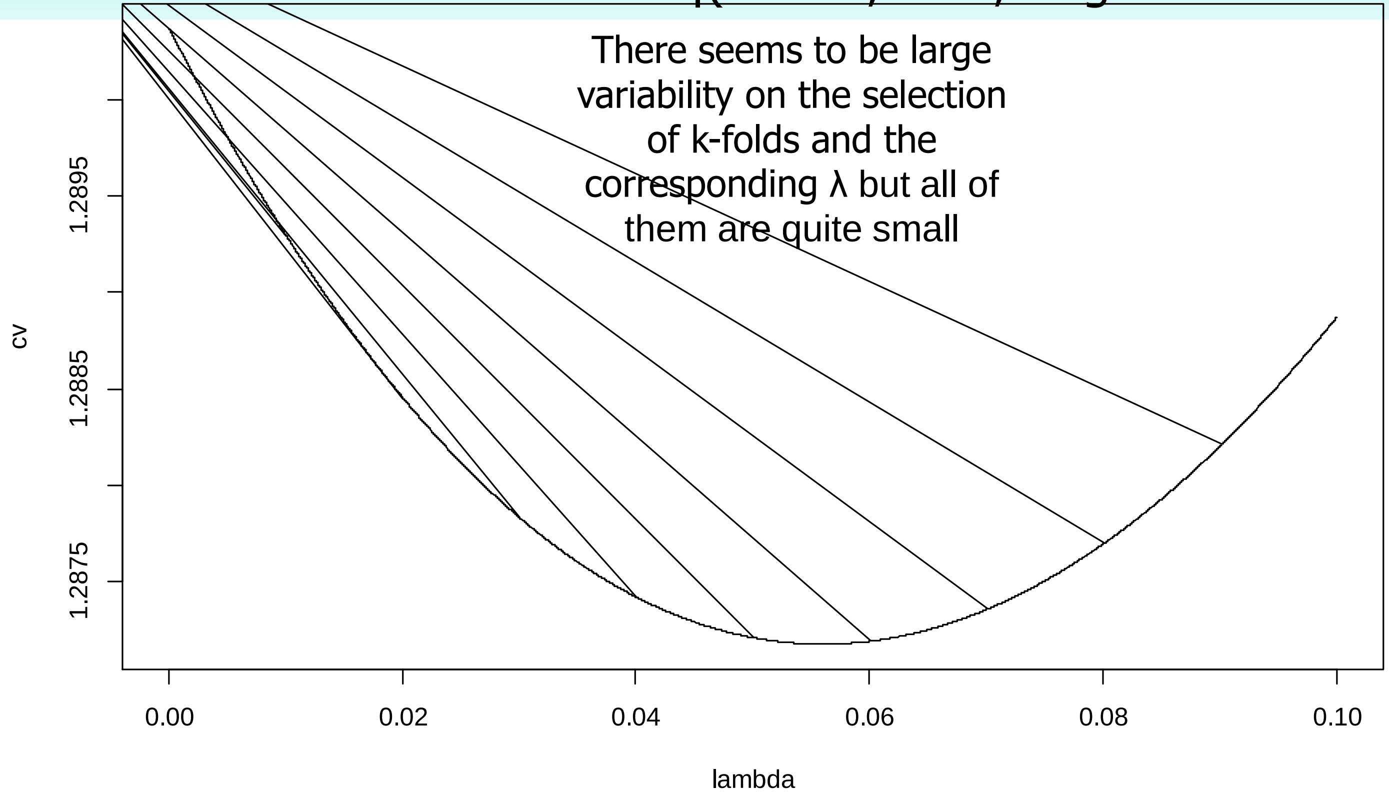


```
> ridge2$lambda[ ridge2$GCV==min(ridge2$GCV) ]
[1] 0.01736737
```

Ridge Regression

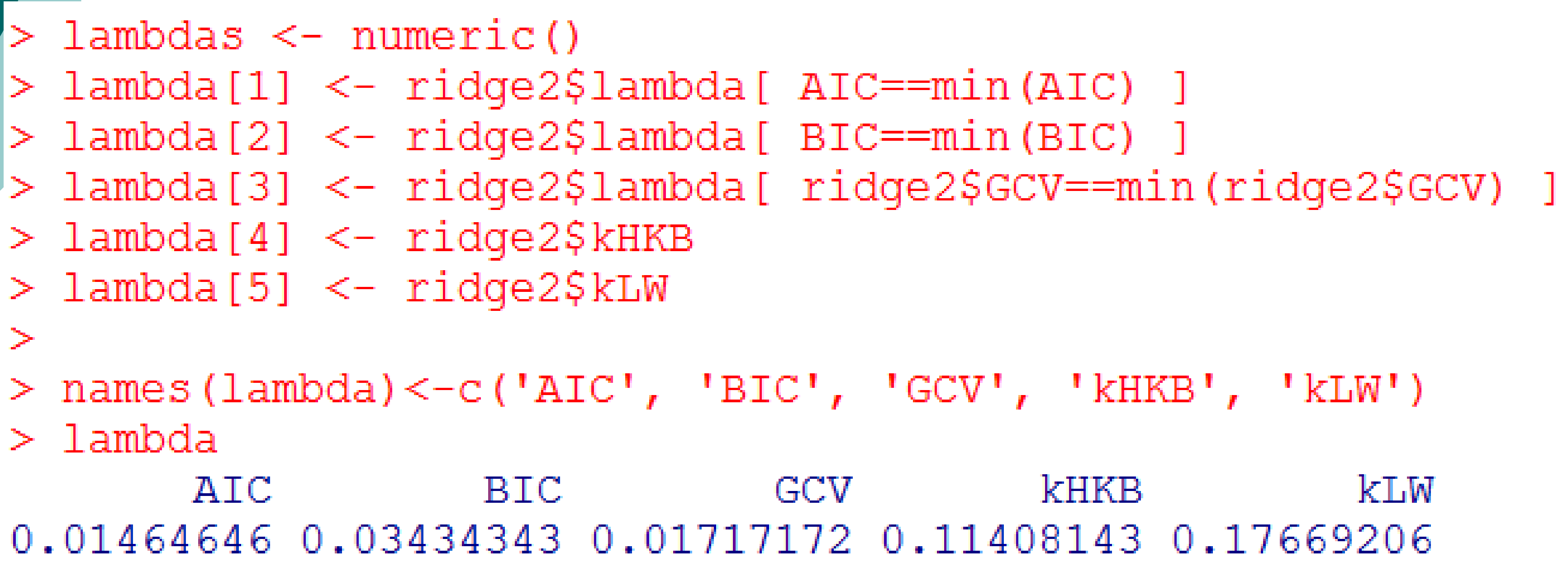
K-fold CV using “ridge.cv” in “parcor”

```
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)
```



Ridge Regression

Summary of proposed λ



```
> 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
       AIC        BIC        GCV       kHKB        kLW
0.01464646 0.03434343 0.01717172 0.11408143 0.17669206
```

LASSO

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 ℓ_1 penalization.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B*, *58*(1), 267–288.

- Web of Science: 5063 citations [8/12/2014]
- Scholar google: 11720 citations [8/12/2014]

LASSO

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

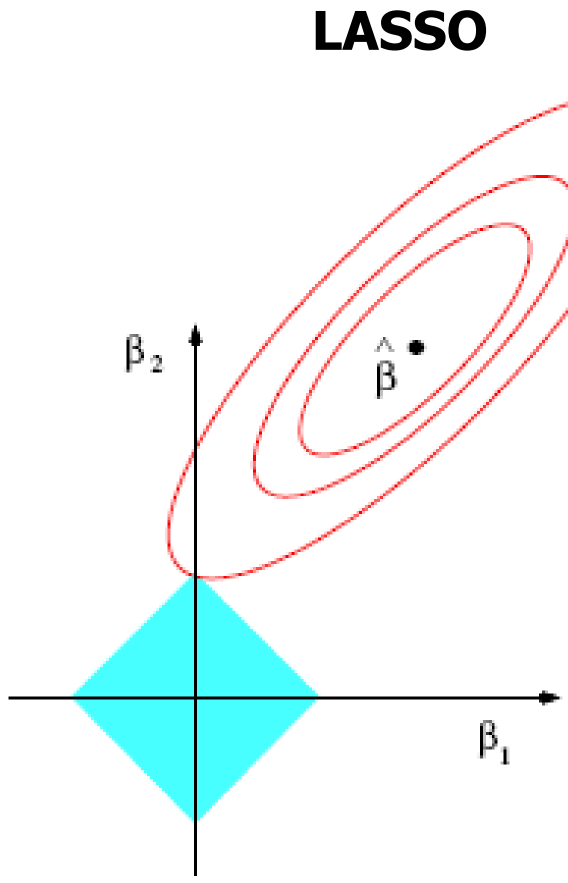
The most popular approach is the LASSO based on the ℓ_1 penalization.

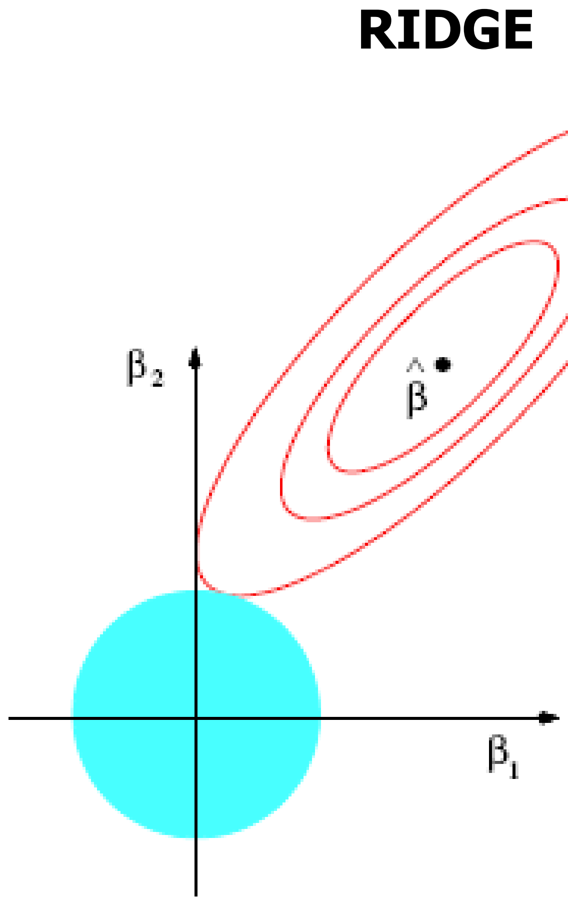
$$\text{minimize } (\mathbf{y} - \mathbf{Z}\beta)^\top (\mathbf{y} - \mathbf{Z}\beta) \text{ s.t. } \sum_{i=1}^{p} |\beta_j| \leq t$$

$$\Leftrightarrow \quad \text{minimize} \left\{ \mathbf{y} - \mathbf{Z}\beta)^\top (\mathbf{y} - \mathbf{Z}\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

LASSO

The least absolute shrinkage and selection operator

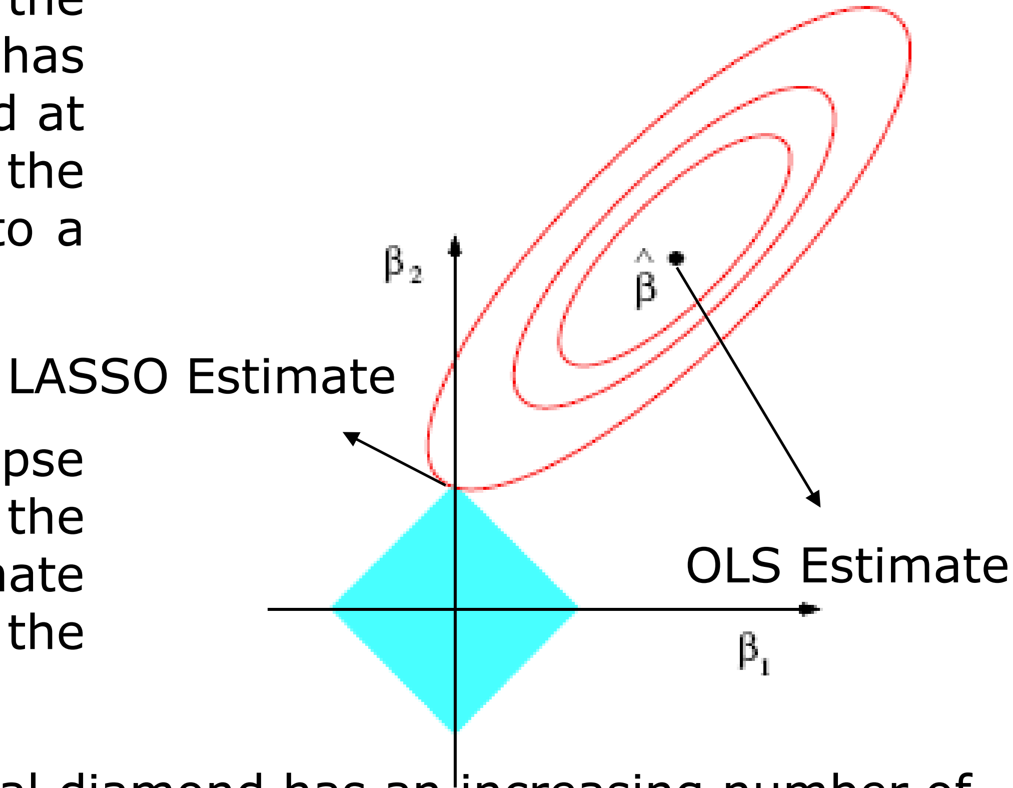




LASSO

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

Lasso and ridge regression both put penalties on β. More generally, penalties of the form

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

may be considered, for q≥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 β_j to zero faster than ridge regression.

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

$$(\mathbf{y} - Z\boldsymbol{\beta})^T (\mathbf{y} - Z\boldsymbol{\beta}) + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} |\beta_j|^2$$

LASSO

Tuning λ or t

- *Again, we have a tuning parameter λ that controls the amount of regularization.*
- *One-to-one correspondence with the threshold t implemented on the ℓ_1 .*
- *If we set t equal to*

$$t_0 = \sum_{j=1}^{p} \left|\hat{\beta}_j\right| = \max \sum_{j=1}^{p} \left|\beta_j\right| = \max \left|\boldsymbol{\beta}\right|_1$$

then we obtain no shrinkage and hence the OLS are returned.

- *We have a path of solutions indexed by λ or t or by the shrinkage factor* $s = \left|\boldsymbol{\beta}\right|_1 / \max \left|\boldsymbol{\beta}\right|_1$.

LASSO

- In regression, you're looking to find the **β** that minimizes:

$$(\mathbf{y}-\mathrm{Z}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}-\mathrm{Z}\boldsymbol{\beta})$$

- In LASSO, you're looking to find the **β** that minimizes:

$$(\mathbf{y}-\mathrm{Z}\boldsymbol{\beta})^{\mathrm{T}}(\mathbf{y}-\mathrm{Z}\boldsymbol{\beta}) \ +\lambda\sum_{j=1}^{p}\left|\beta_j\right|$$

- So when λ = 0 there is no penalization and you have the OLS solution; this is

$$\max\sum_{j=1}^{p}\left|\beta_j\right| = \max\left|\boldsymbol{\beta}\right|_1$$

- As the penalization parameter λ increases, $\sum_{j=1}^{p}\left|\beta_j\right|$

  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 β_js are equal to zero.

LASSO

Lasso performs also variable selection

- *Large enough λ (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

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*
- Bühlmann, P. and van de Geer, S. (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications.* Springer.

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

LASSO

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

Implemention of LASSO

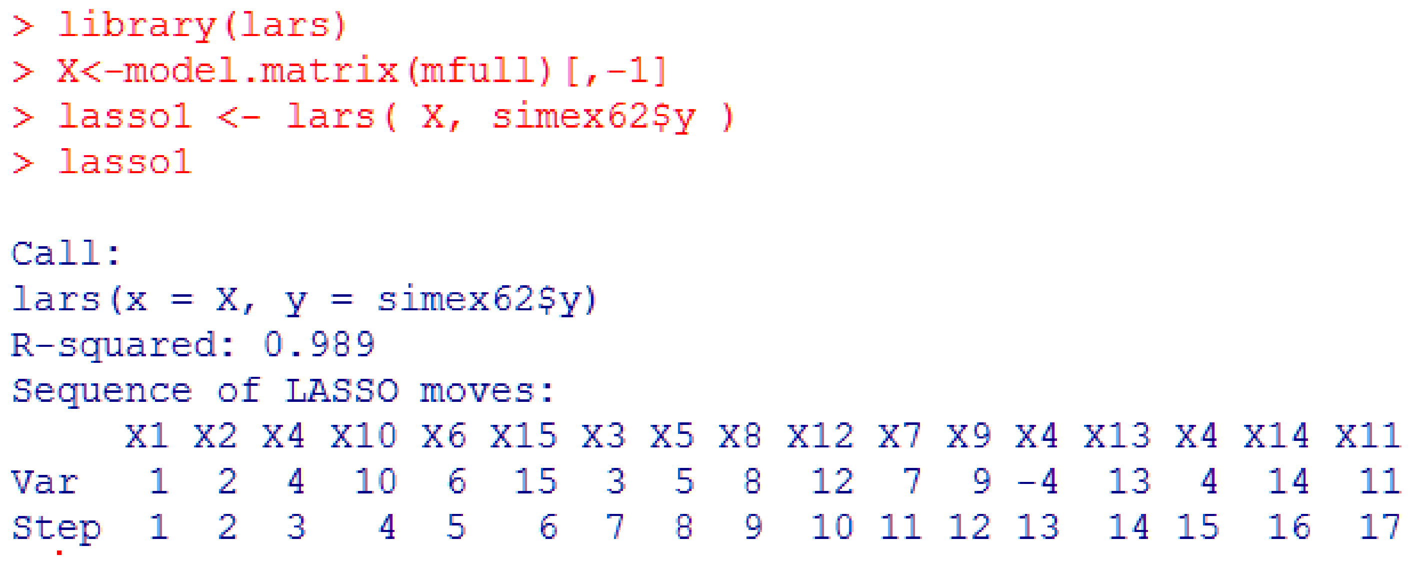
Steps:

1. *Run Lasso for a variety of values.*
2. *Plot the regularization paths.*
3. *Implement k-fold regularization.*
4. *Estimate the coefficients using λ with minimum CV-MSE.*

LASSO

Implementing LASSO using the “lars” package

Steps 1: Run Lasso for a variety of values



```
> library(lars)
> X<-model.matrix(mfull)[,-1]
> lasso1 <- lars( X, simex62$y )
> lasso1

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
```

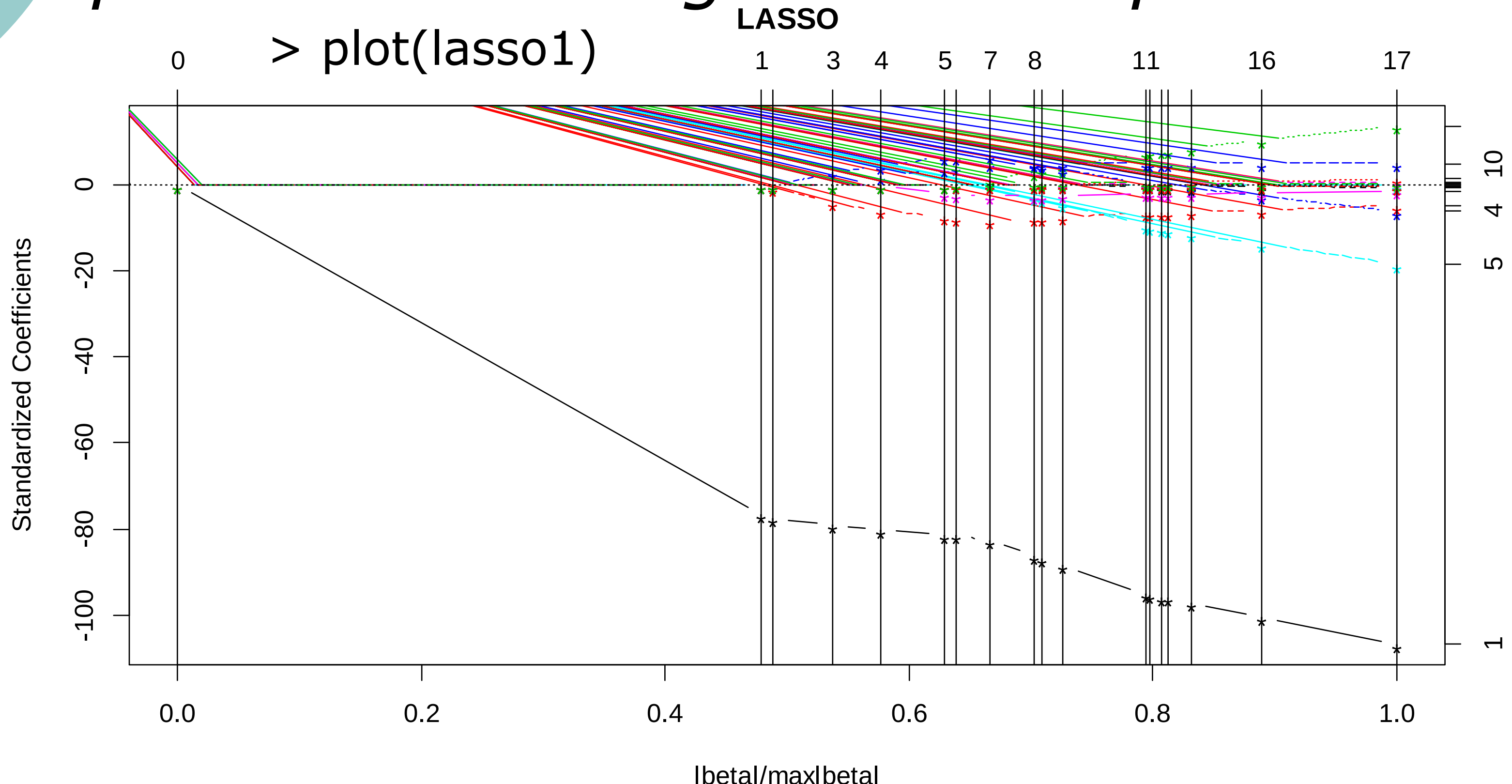
Sequence of actions – variables added or excluded in each value of λ

LASSO

Implementing LASSO using the "lars" package

Steps 2: Plot the regularization paths

> plot(lasso1)

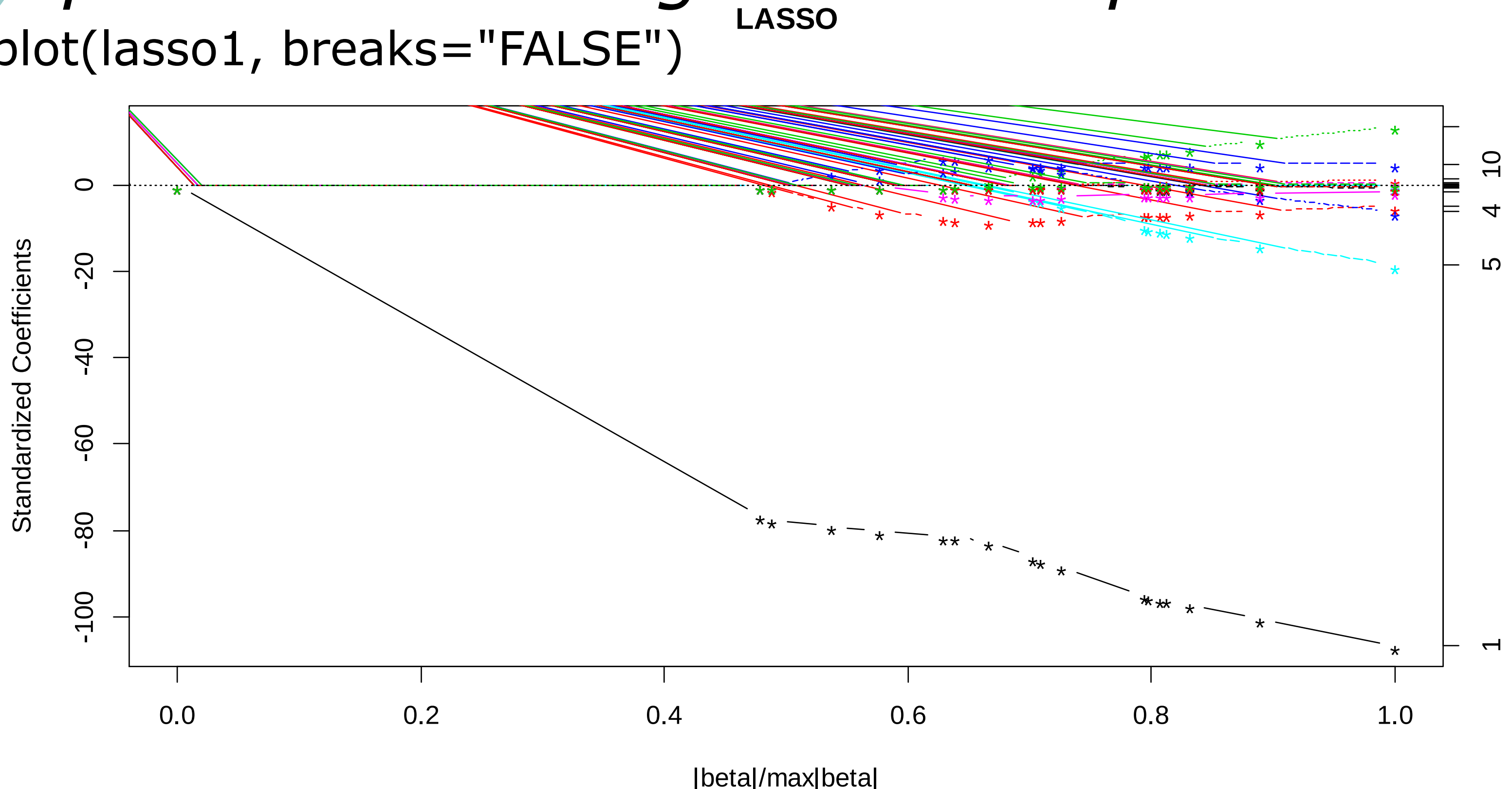


LASSO

Implementing LASSO using the “lars” package

Steps 2: Plot the regularization paths

```
> plot(lasso1, breaks="FALSE")
```

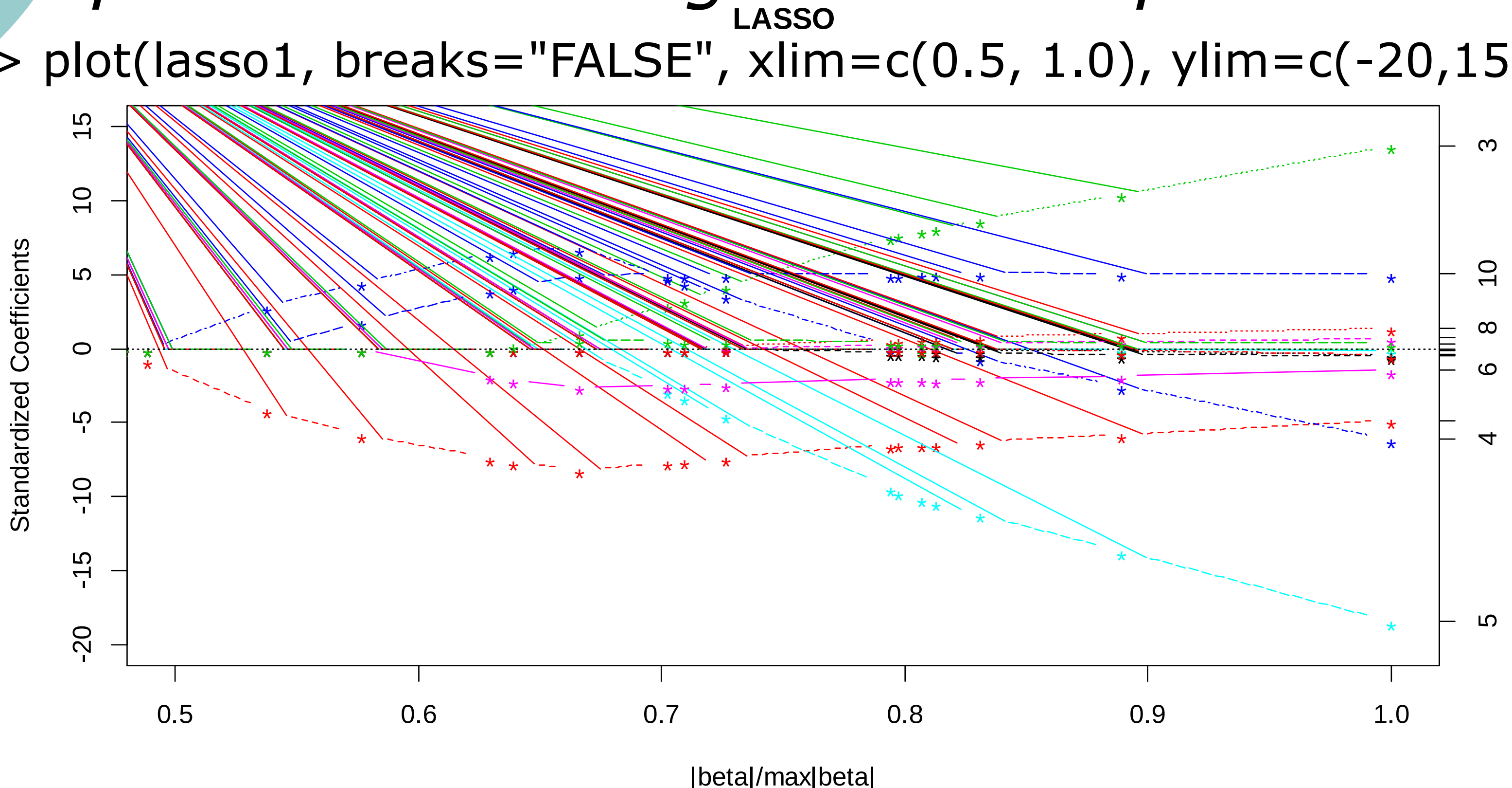


LASSO

Implementing LASSO using the "lars" package

Steps 2: Plot the regularization paths

```
> plot(lasso1, breaks="FALSE", xlim=c(0.5, 1.0), ylim=c(-20,15))
```



LASSO

Implementing LASSO using the “lars” package

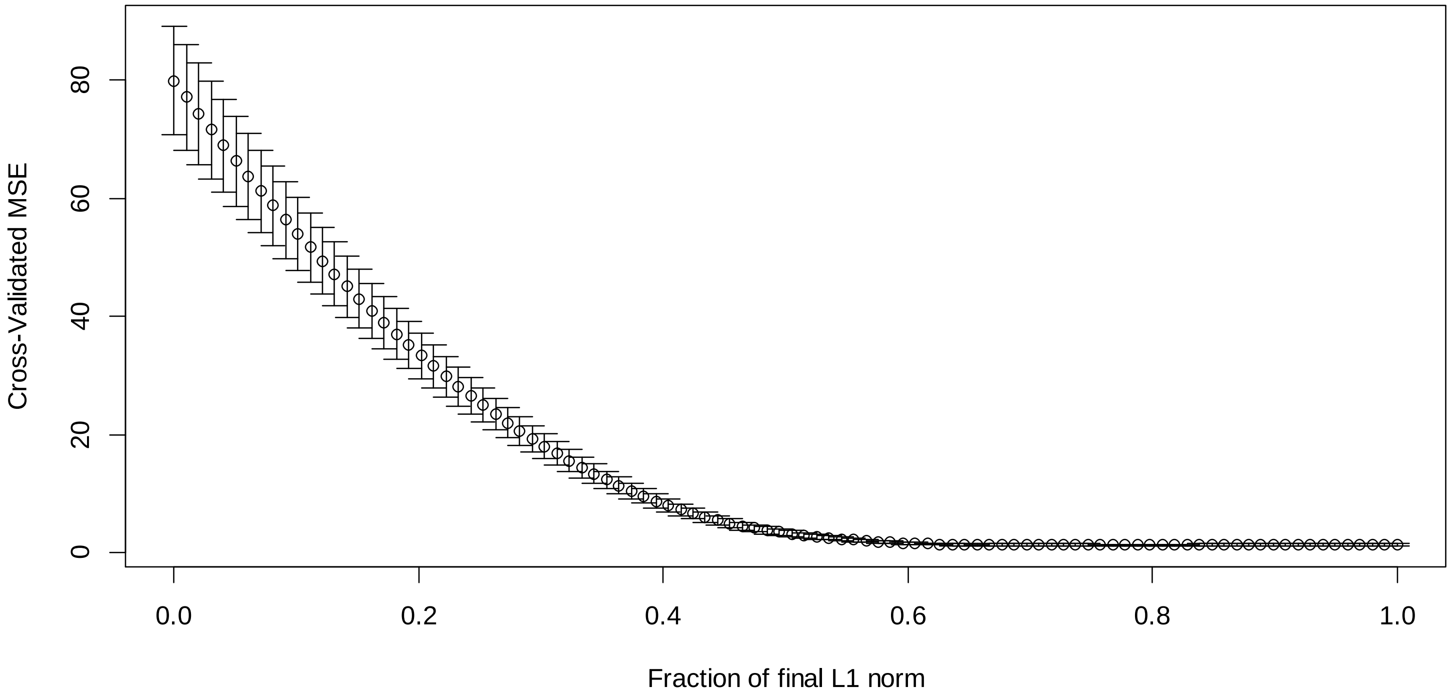
Steps 3-4: Implement 10-fold CV and select s

```
>  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( lasso1, 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
```

LASSO

Implementing LASSO using the "lars" package

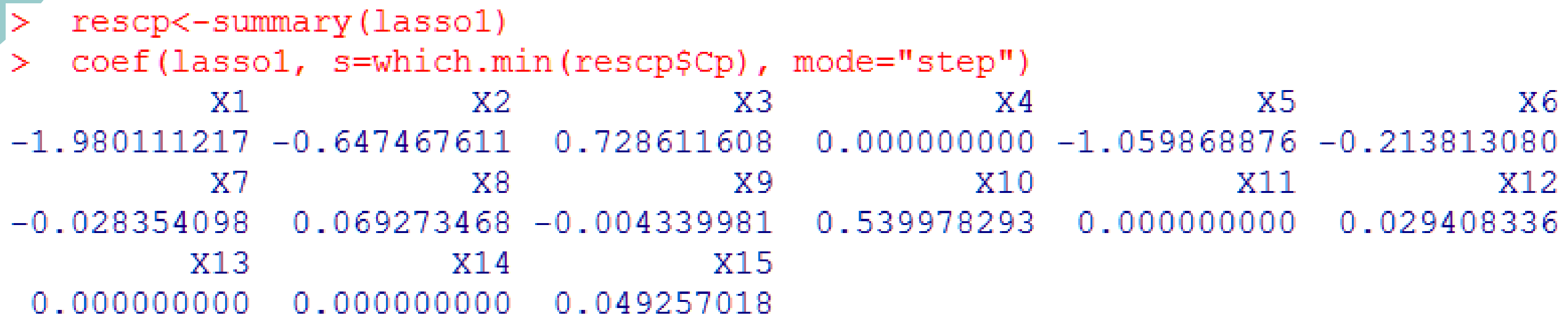
Steps 3-4: Implement 10-fold CV and select s



LASSO

Implementing LASSO using the “lars” package

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



```
> rescp<-summary(lasso1)
> coef(lasso1, s=which.min(rescp$Cp), mode="step")
          x1           x2           x3           x4           x5           x6 
-1.980111217 -0.647467611  0.728611608  0.000000000 -1.059868876 -0.213813080 
          x7           x8           x9          x10          x11          x12 
-0.028354098  0.069273468 -0.004339981  0.539978293  0.000000000  0.029408336 
         x13          x14          x15 
 0.000000000  0.000000000  0.049257018 
```

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

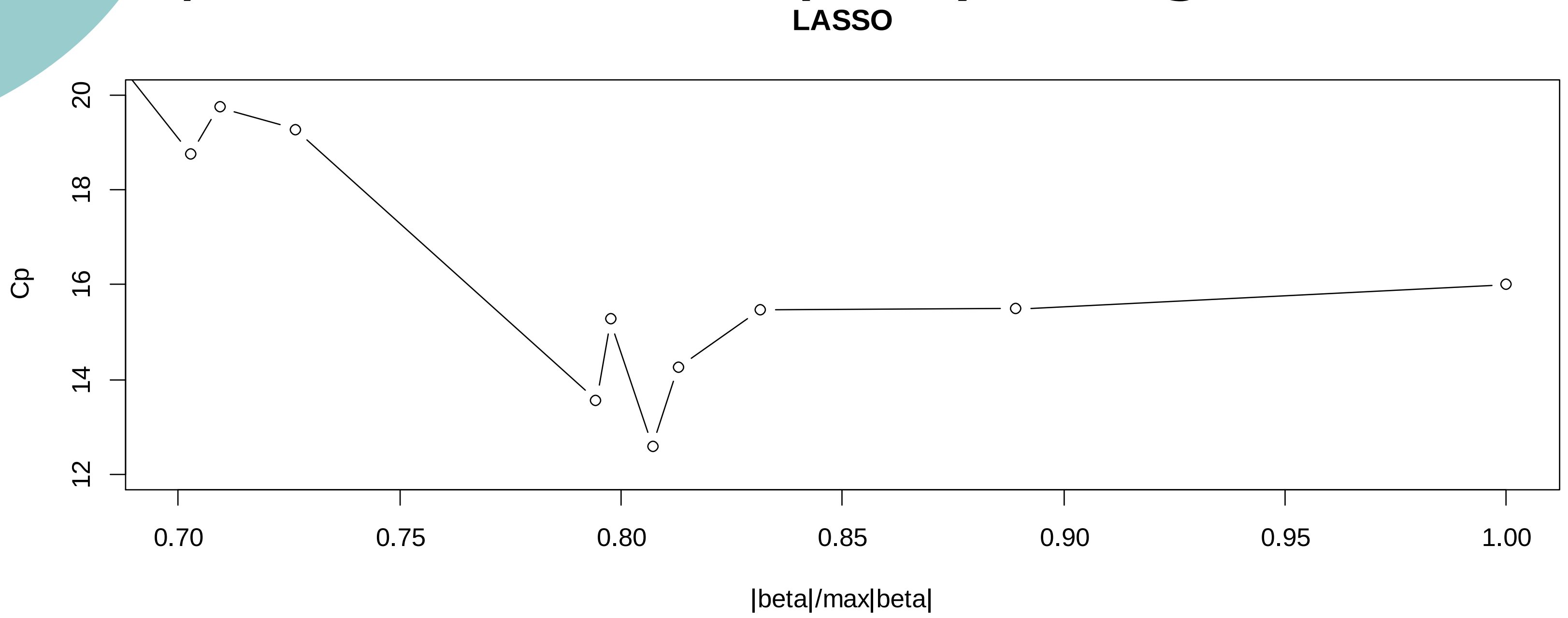
$$C_{p_m} = \frac{RSS_m}{\widehat{\sigma}^2_{full}} - (n - 2p_m)$$

- It is equivalent to AIC in normal regression models
- It is approximately equal to the MSE from the leave-one-out CV

LASSO

Implementing LASSO using the "lars" package

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

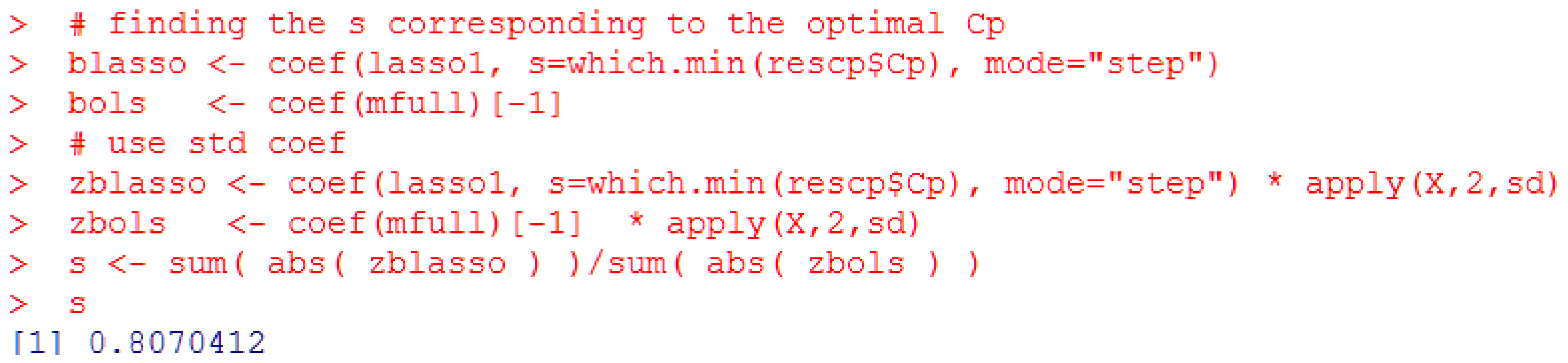


```
plot(lasso1, xvar='n', plottype='Cp')
plot(lasso1, xvar='n', plottype='Cp', ylim=c(12,20), xlim=c(0.7,1))
```

LASSO

Implementing LASSO using the “lars” package

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



```
> # 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
```

LASSO

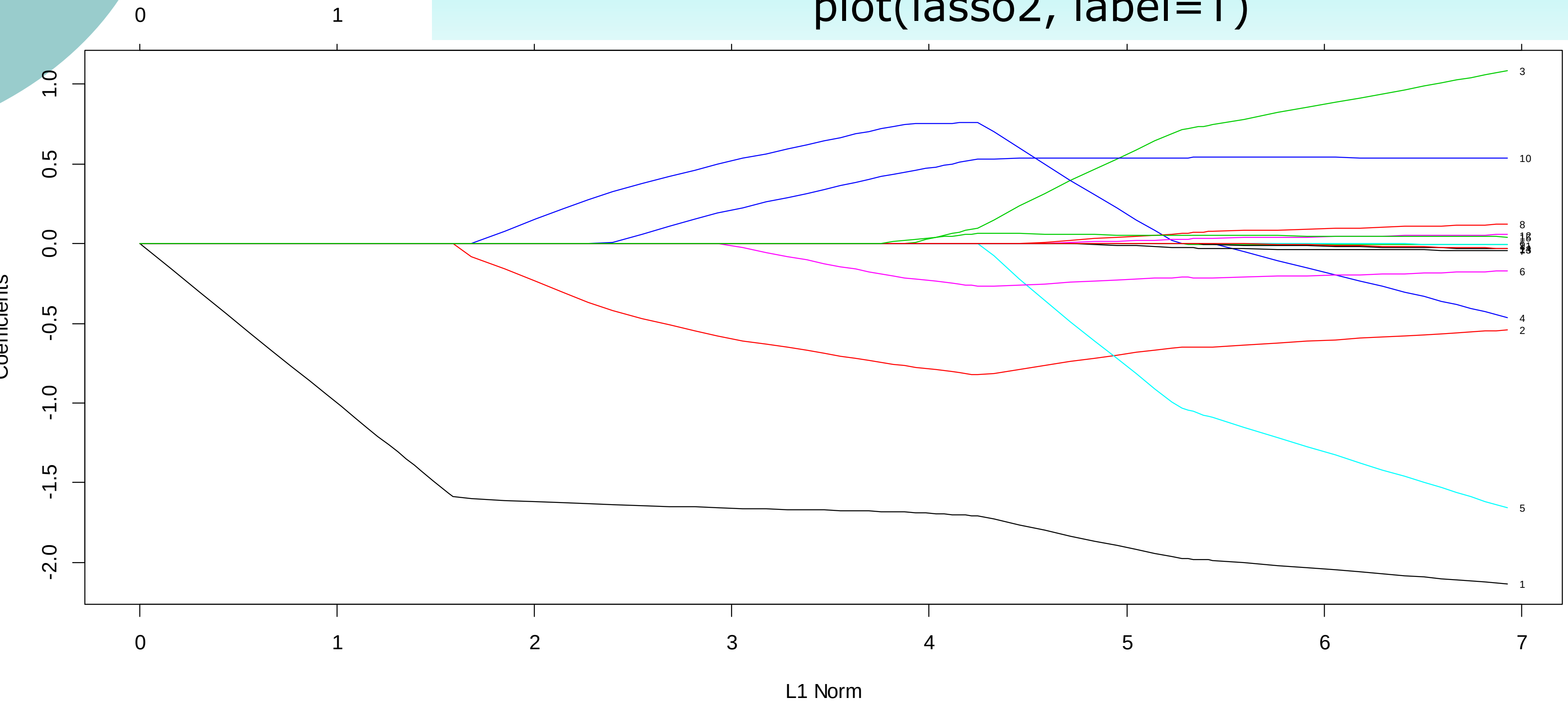
Implementing LASSO using the “glmnet” package

- *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*

LASSO

Implementing LASSO using the "glmnet" package

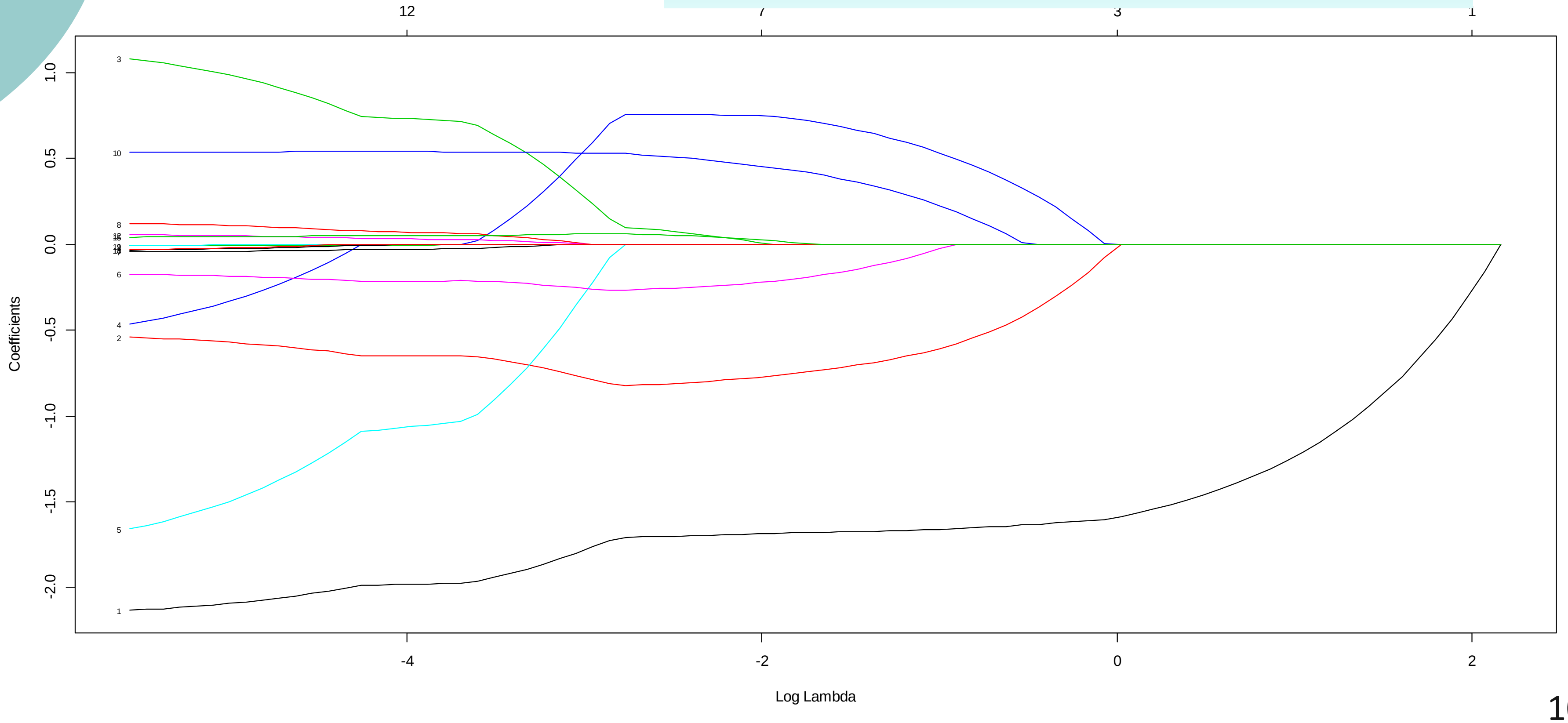
```
library(glmnet)
lasso2 = glmnet(X, simex62$y )
plot(lasso2, label=T)
```



LASSO

Implementing LASSO using the "glmnet" package

```
plot(lasso2, xvar='lambda',
label=T)
```

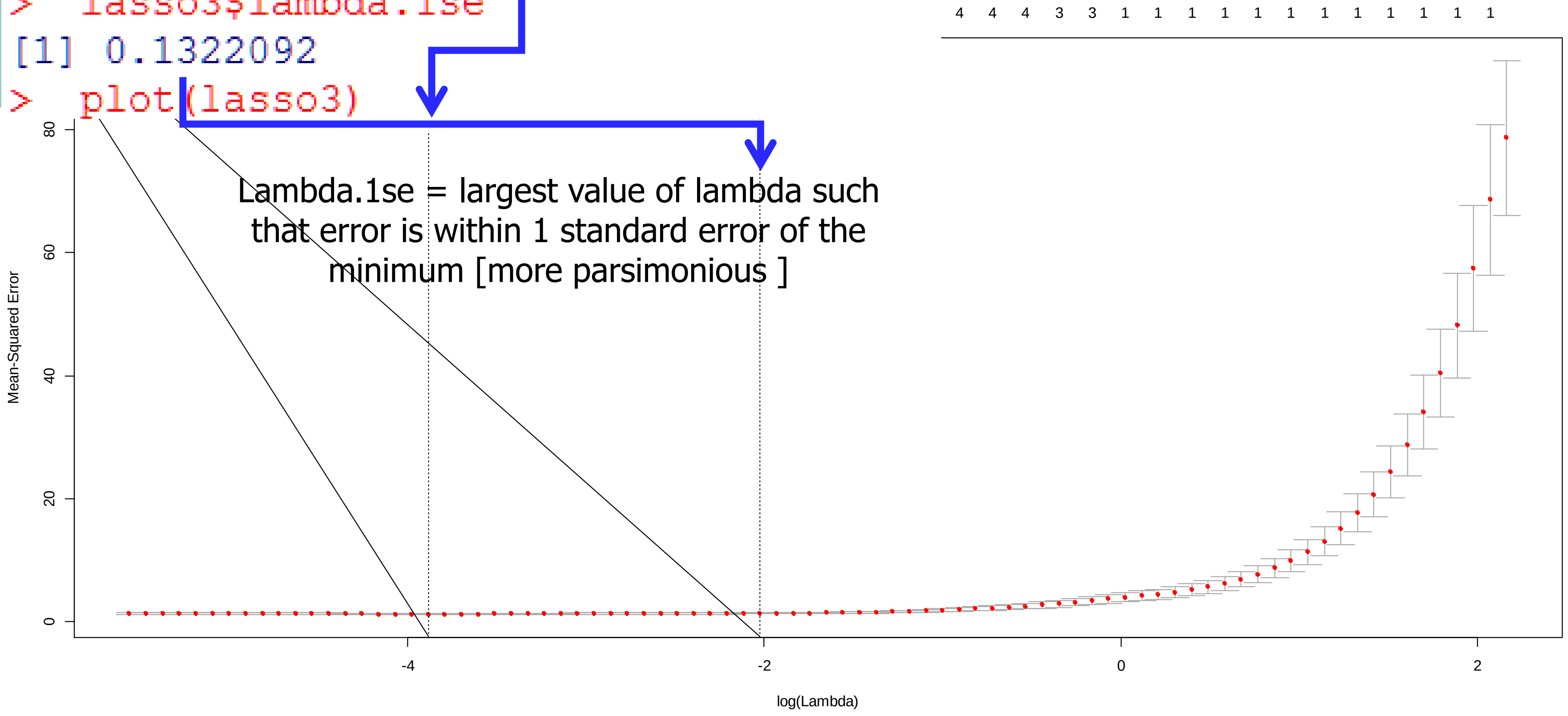


LASSO

Implementing LASSO using the "glmnet"

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

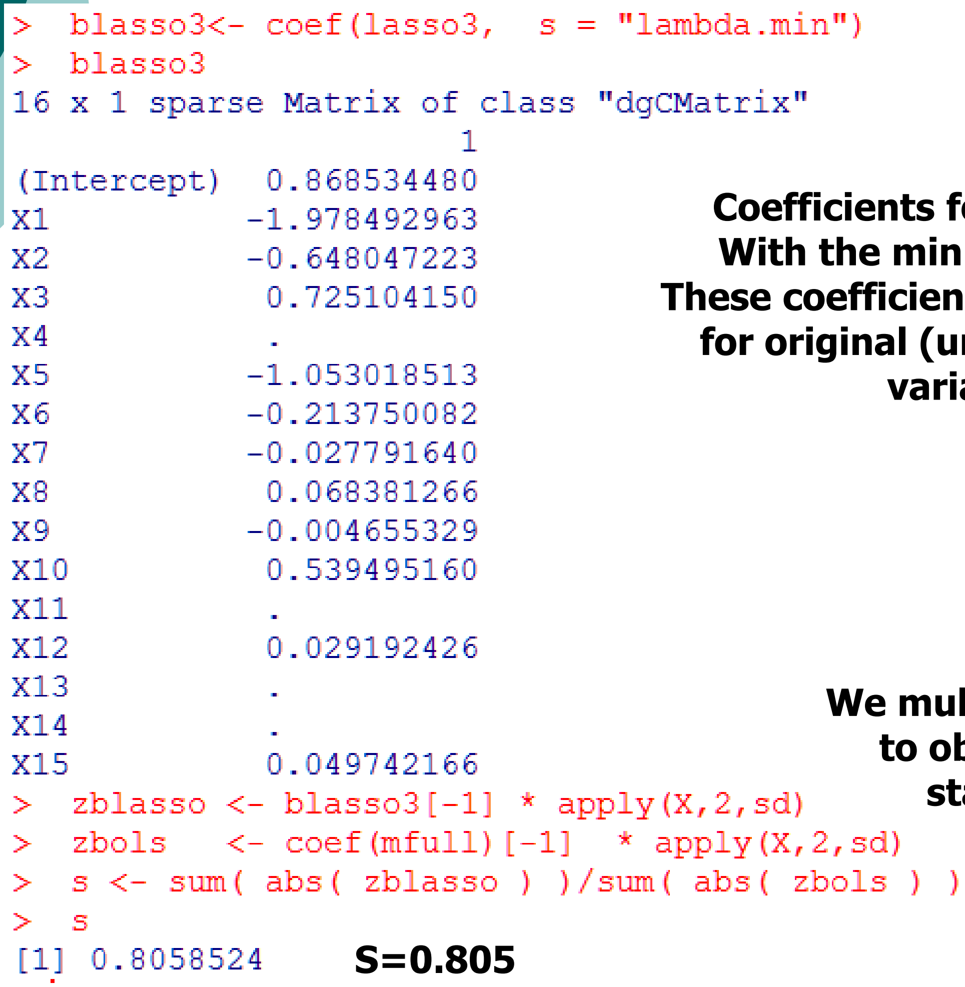
Lambda.min= is the one with the minimum CV-MSE



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

LASSO

Implementing LASSO using the “glmnet”



```
> 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
```

S=0.805

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

LASSO

Implementing LASSO using the “glmnet”

```
>  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
>  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
```

Coefficients for lambda 1se
With distance of 1 se from the minimum CV-MSE

S=0.64