Laboratorio con R - 3

Metodi e Modelli per l'Inferenza Statistica - Ing. Matematica - a.a. 2018-19

31/05/2019

---

0. Librerie

```
library( MASS )
library( car )
## Warning: package 'car' was built under R version 3.4.4
## Loading required package: carData
## Warning: package 'carData' was built under R version 3.4.4
library( corrplot )
## Warning: package 'corrplot' was built under R version 3.4.4
## corrplot 0.84 loaded
library(ElemStatLearn)
## Warning: package 'ElemStatLearn' was built under R version 3.4.4
library( faraway )
## Warning: package 'faraway' was built under R version 3.4.4
##
## Attaching package: 'faraway'
## The following objects are masked from 'package:ElemStatLearn':
##
##     ozone, prostate
## The following objects are masked from 'package:car':
##
##     logit, vif
library( lars )
## Warning: package 'lars' was built under R version 3.4.4
## Loaded lars 1.2
library( Matrix )
## Warning: package 'Matrix' was built under R version 3.4.4
```

Reference:

Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, pp. 337-387). New York: Springer series in statistics.

Teoria: Multicollinearità e Ridge regression

Una delle ipotesi alla base del modello di regressione classico (OLS) è che la matrice delle variabili esplicative Z abbia rango pieno. Se il rango di questa matrice è inferiore a r+1 si ha che $|Z^T Z| = 0$ e non si può calcolare l'inversa di $Z^T Z$. Ne consegue che la stima dei coefficienti di regressione non può essere determinata univocamente. In questa circostanza si è in presenza di multicollinearità.

La multicollinearità fa esplodere la variabilità delle stime dei parametri di regressione. La Ridge Regresion può essere introdotta come metodo che, penalizzando nella verosimiglianza la norma dei regressori, ne controlla la magnitudo.

Si parla di multicollinearità quando si è in presenza di una forte dipendenza lineare tra due o più regressori.

La presenza di multicollinearità causa **problemi di stima**. Considerando il metodo dei minimi quadrati, se vi è multicollinearità, la matrice $Z^T Z$ risulta quasi singolare e quindi si ha un mal-condizionamento del sistema di equazioni che dovrebbero fornirci la stima dei parametri del modello; questi problemi si riflettono chiaramente anche nella scarsa affidabilità degli IC/test di significatività dei predittori. Chiaramente, ci sono anche problemi di previsione: risulta inutile fare previsione per un valore della variabile di risposta y, proprio per l'inaffdabilità dei valori di $\hat{\beta}$.

Collinearity detection Problemi di multicollinearità possono essere individuati con i seguenti metodi:

1. calcolando la **matrice di correlazione**;

2. calcolando il **numero di condizionamento** della matrice $Z^T Z$;

3. calcolandi i **VIF**. La varianza stimata del j-esimo coefficiente di regressione può essere scritta come:

$$Var(\beta_j) = \frac{S^2}{(n-1) \cdot S_j^2} \cdot \frac{1}{1 - R_j^2}$$

dove S^2 è la varianza dell'errore, S_j^2 è la varianza di x_j ed R_j^2 è il coefficiente di determinazione calcolato facendo la regressione di x_j sulle altre variabili esplicative x_i. La quantità:

$$VIF = \frac{1}{1 - R_j^2}$$

E' chiamata fattore di inflazione di varianza per β_j . I VIF sono utilizzati come misure di multicollinearità, perché la radice quadrata dei VIF indica di quanto l'intervallo di confidenza, costruito su ciascuno dei coefficienti di regressione β_j è più grande rispetto alla situazione di dati non correlati. In particolare, quindi, le variabili che risultano maggiormente sospette di provocare il fenomeno della multicollinearità sono quelle che presentano VIF più elevato.

Collinearity resolution

Per risolvere il problema della multicollinearità ci sono diverse strade che possono essere percorse:

1. l'esclusione dal modello delle variabili correlate ovvero di quelle per le quali la stima della varianza del coefficiente di regressione associato è elevata (**riduzione di modello**);

2. l'uso della **ridge regression**. L'uso della ridge regression consente di ottenere delle stime stabili dei coefficienti di regressione in presenza di multicollinearità con la matrice $Z^T Z$ assai prossima alla singolarità.

Lo stimatore di tipo ridge è definito come:

$$\hat{\beta}_R = [(Z^T Z) + \lambda \cdot I]^{-1} \cdot Z^T y$$

dove $\lambda \in (0, 1)$ è lo **shrinkage parameter**.

La scelta di questa costante viene effettuata in base all'intensità della multicollinearità esistente, cercando di garantire un opportuno bilanciamento tra la varianza e la distorsione dello stimatore. Un metodo esplorativo consiste nella costruzione di un grafico che rappresenti gli elementi del vettore $\hat{\beta}_R$ (sull'asse delle ordinate) in funzione di λ. Si ritiene che le curve di tale grafico, detto traccia della regressione ridge, tendano a stabilizzarsi in corrispondenza di valori accettabili di lambda.

3. l'uso della principal component regression (**PCR**): si estraggono le componenti principali dai regressori originali (queste nuove variabili sono per definizione tra loro ortogonali) e si fa regredire la variabile risposta su queste [Approfondimento e possibile RC];

4. l'aggiunta di nuove osservazioni che rendano la matrice Z a rango pieno (anche se questo rimedio non è sempre applicabile).

---

1. Collinearity detection: formaggio dataset

Si considerino i dati contenuti nel file `formaggio.txt`. Essi sono relativi alle concentrazioni di varie sostanze chimiche in 30 campioni di formaggio Cheddar, prodotto nella zona LaTrobe Valley dello stato Victoria in Australia.

Come variabile di risposta è stata considerata una misura soggettiva del gusto per ogni campione. E' noto, infatti, che man mano che il formaggio matura, hanno luogo diversi processi chimici che determinano il sapore del prodotto finale.

Le variabili prese in considerazione sono:

- **Taste** Punteggio soggettivo del test sul gusto, ottenuto combinando i punteggi dei diversi assaggiatori.
- **Acetic** Logaritmo naturale della concentrazione di acido acetico.
- **H2S** Logaritmo naturale della concentrazione di acido solfidrico.
- **Lactic** Concentrazione di acido lattico.

Eseguire un'analisi dei dati mediante regressione lineare multipla, individuando, se presenti collinearità fra i regressori.

Soluzione

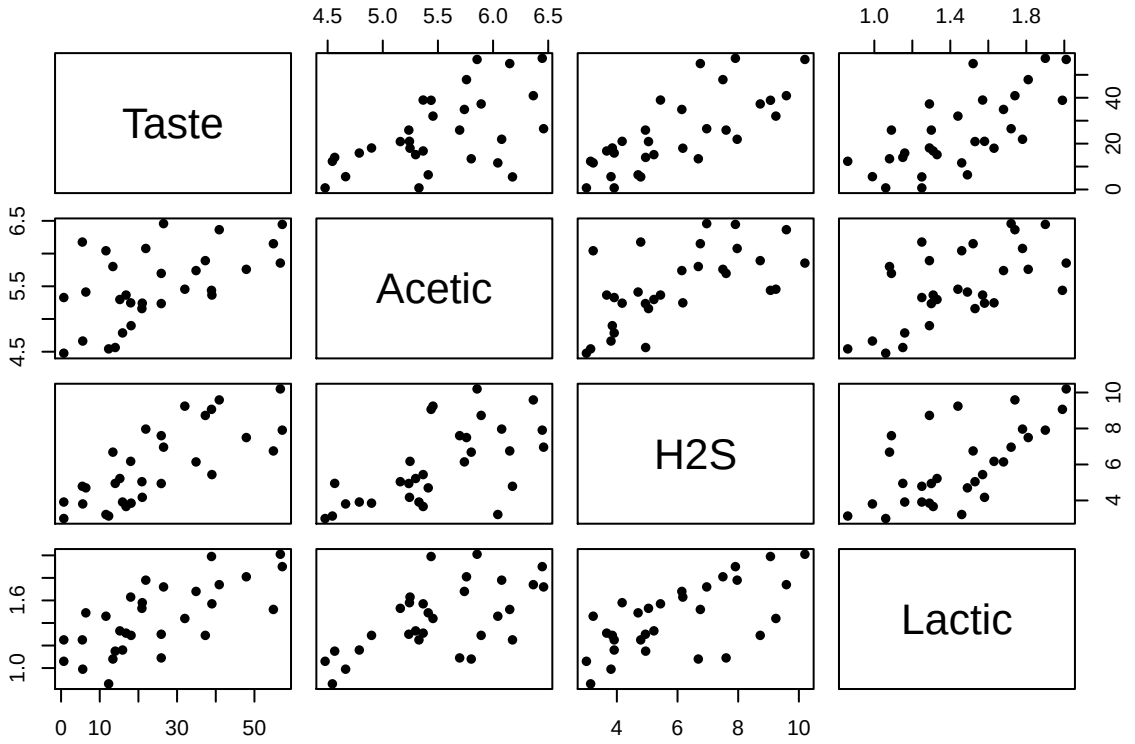
Importiamo i dati.

```
formaggio = read.table( "formaggio.txt", header = TRUE )

head( formaggio )
##   Taste Acetic   H2S Lactic
## 1  12.3  4.543 3.135   0.86
## 2  20.9  5.159 5.043   1.53
## 3  39.0  5.366 5.438   1.57
## 4  47.9  5.759 7.496   1.81
## 5   5.6  4.663 3.807   0.99
## 6  25.9  5.697 7.601   1.09

str( formaggio )
## 'data.frame':    30 obs. of  4 variables:
##  $ Taste : num  12.3 20.9 39 47.9 5.6 25.9 37.3 21.9 18.1 21 ...
##  $ Acetic: num  4.54 5.16 5.37 5.76 4.66 ...
##  $ H2S   : num  3.13 5.04 5.44 7.5 3.81 ...
##  $ Lactic: num  0.86 1.53 1.57 1.81 0.99 1.09 1.29 1.78 1.29 1.58 ...

names( formaggio )
## [1] "Taste"  "Acetic" "H2S"    "Lactic"
```

```
dim( formaggio )
## [1] 30  4

attach( formaggio )
```

Eseguiamo un'analisi esplorativa dei dati.

```
pairs( formaggio, pch = 16 )
```



Tutte le variabili sembrano piuttosto correlate (positivamente).

Verifichiamo quindi la presenza di **multicollinearità**.

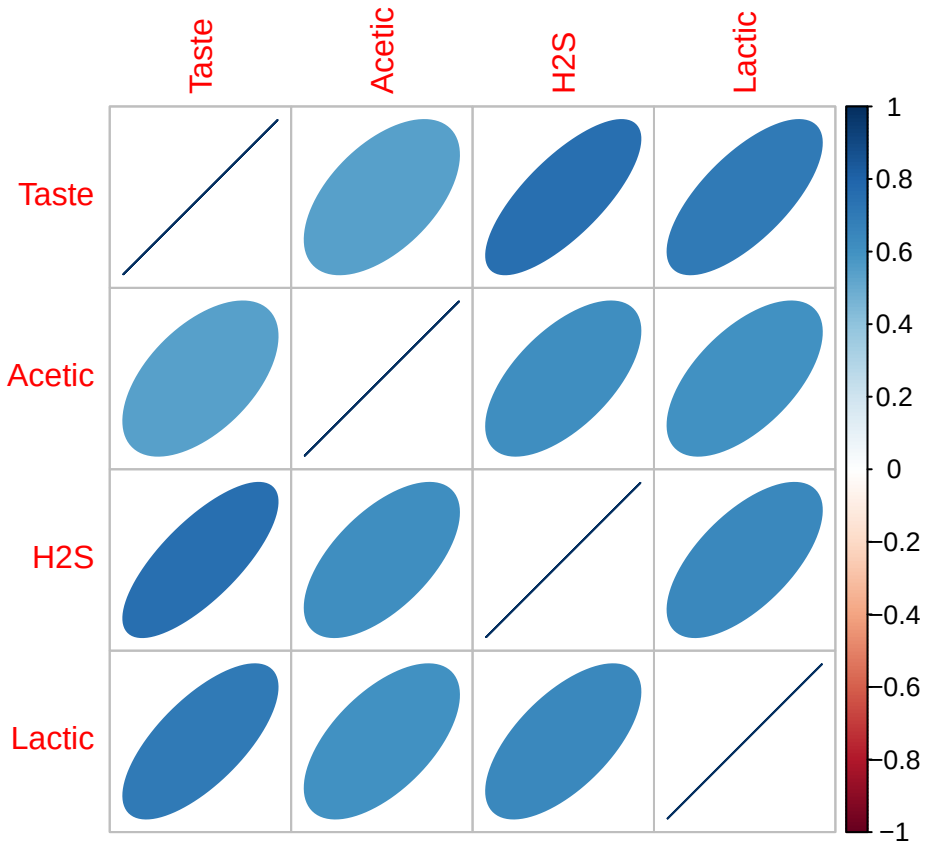
Il modo migliore per agire, in questo caso, è di vedere quali sono le variabili esplicative responsabili del fenomeno di multicollinearità ed escluderle dall'analisi; questo modo di procedere, dal punto di vista della spiegazione della variabile di risposta a partire delle variabili esplicative, porta ad una perdita di informazione molto piccola.

ANALISI DI MULTICOLLINEARITA'

1. Matrice di correlazione;

```
cor_form = cor( formaggio )

cor_form
##            Taste    Acetic       H2S    Lactic
## Taste  1.0000000 0.5495393 0.7557523 0.7042362
## Acetic 0.5495393 1.0000000 0.6179559 0.6037826
```

```
## H2S    0.7557523 0.6179559 1.0000000 0.6448123
## Lactic 0.7042362 0.6037826 0.6448123 1.0000000

#library(corrplot)

corrplot( cor_form, method = 'ellipse' )
```



Il grafico della matrice di correlazione conferma l'intuizione che avevamo avuto osservando il grafico pairs, ovvero buon livello di correlazione positiva fra i regressori.

Domanda: potrei guardare la matrice di covarianza invece?

2. Calcolo del numero di condizionamento;

```
reg.formaggio = lm( Taste ~ Acetic + H2S + Lactic )

Z = model.matrix( reg.formaggio )

# Stima affidabile
# library( Matrix )
cond = condest( t( Z ) %*% Z )
cond  # >> 30 = > presenza di collinearità
## $est
## [1] 12885.76
##
## $v
## [1]  0.79524468 -0.15980539  0.02099422 -0.02395571
```

```
# Oppure, usando la definizione (ma la stima non è affidabile nel
# caso di matrice mal condizionata):
eigs  = eigen( t( Z ) %*% Z )$values
cond = sqrt( max( eigs ) / min( eigs ) )
```

Il fatto che il numero di condizionamento sia 12885.76, valore >> 30, conferma la forte presenza di collinearità fra i regressori.

N.B. Dato che la matrice è mal condizionata, la stima del numero di condizionamento manuale non è affidabile, è quindi consigliabile usare il comando `condest`.

3. Calcolo dei VIF;

```
reg_acetic = lm( Acetic ~  H2S + Lactic )

1/( 1-summary( reg_acetic )$r.squared )
## [1] 1.831589

#library( car )
vif( reg.formaggio )
##   Acetic      H2S   Lactic
## 1.831589 1.992200 1.937912

var_ac = var( Acetic )

S_res = summary( reg.formaggio )$sigma^2

var_beta_ac = S_res/( ( 30 - 1 )* var_ac ) * 1/( 1-summary( reg_acetic )$r.squared )
var_beta_ac
## [1] 19.88943

summary( reg.formaggio )$coef[ 2, 2 ]^2
## [1] 19.88943
```

Considerato che, solitamente, si ritengono responsabili del fenomeno di multicollinearità quelle variabile esplicative che producono valori dei VIF superiori a 10, i VIF in questo caso non sono informativi.

Vagliamo ora le possibili soluzioni al problema della multicollinearità:

1. Model selection;

Procediamo con un'automatica stepwise regression.

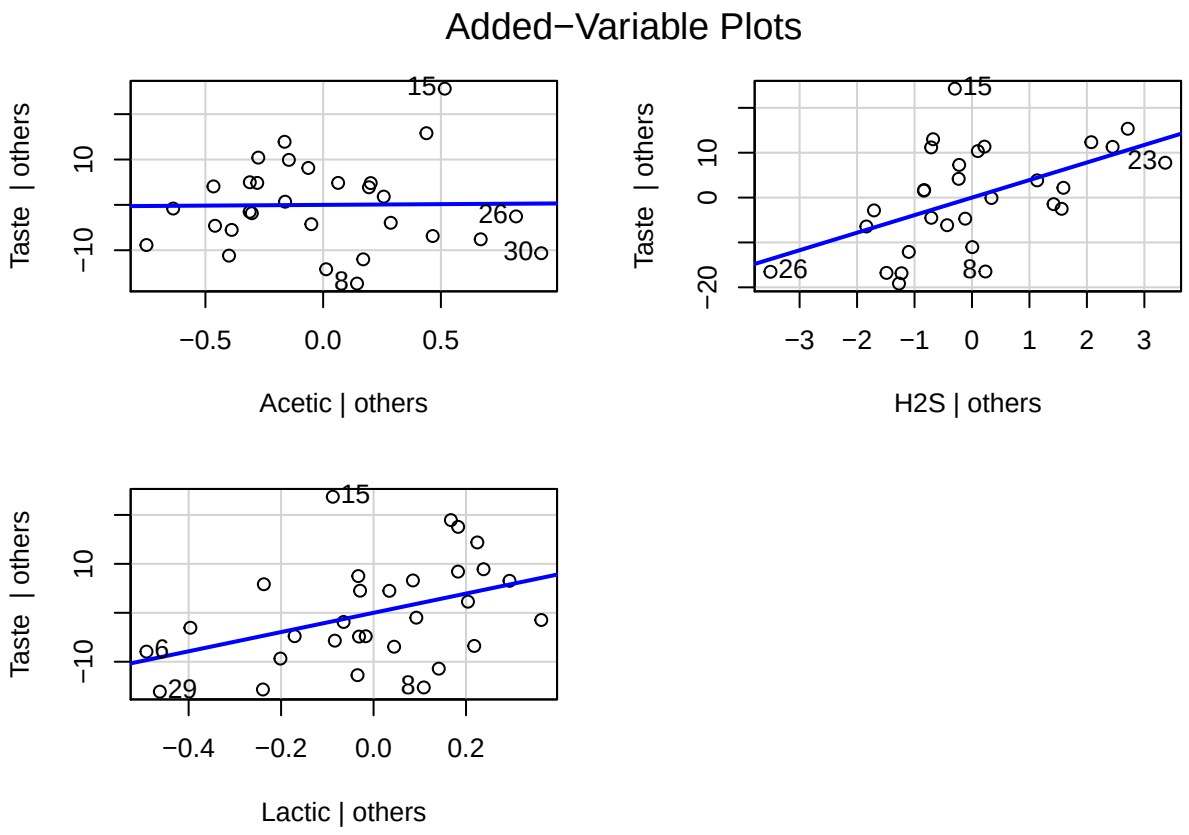
```
step( reg.formaggio, direction = "backward" )
## Start:  AIC=142.64
## Taste ~ Acetic + H2S + Lactic
##
##          Df Sum of Sq    RSS    AIC
## - Acetic  1      0.55 2669.0 140.65
## <none>                2668.4 142.64
## - Lactic  1    533.32 3201.7 146.11
## - H2S     1   1007.66 3676.1 150.25
##
## Step:  AIC=140.65
## Taste ~ H2S + Lactic
##
##          Df Sum of Sq    RSS    AIC
```

```
## <none>                2669.0 140.65
## - Lactic  1    617.18 3286.1 144.89
## - H2S     1   1193.52 3862.5 149.74
##
## Call:
## lm(formula = Taste ~ H2S + Lactic)
##
## Coefficients:
## (Intercept)          H2S       Lactic
##     -27.592        3.946       19.887
```

Sembra suggerire di rimuovere la variabile Acetic.

Per capire il motivo, proviamo a guardere l'added variable plot, che ci spiega l'effetto di ogni predittore, depurato della collinearità con gli altri, sulla variabilità di Y che non viene spiegata dagli altri (plot tra i residui di Y ~ tutti gli altri predittori vs. residui di X_i ~ tutti gli altri predittori).

```
ac_reg = summary( lm( Acetic ~ H2S + Lactic ) )
h2s_reg = summary( lm( H2S ~ Acetic + Lactic ) )
lac_reg = summary( lm( Lactic ~ H2S + Acetic ) )

#costruzione manuale grafici avPlots
#plot( ac_reg$residuals, reg.formaggio$residuals )
#plot( h2s_reg$residuals, reg.formaggio$residuals )
#plot( lac_reg$residuals, reg.formaggio$residuals )


avPlots( reg.formaggio )
```



Pare che Acetic sia la variabile con meno contenuto predittivo unico tra quelle disponibili (linea blu più vicina allo 0).

```
reg1.formaggio = lm( Taste ~ H2S + Lactic )

summary( reg1.formaggio )
##
## Call:
## lm(formula = Taste ~ H2S + Lactic)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.343  -6.530  -1.164   4.844  25.618
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -27.592      8.982  -3.072  0.00481 **
## H2S            3.946      1.136   3.475  0.00174 **
## Lactic        19.887      7.959   2.499  0.01885 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.942 on 27 degrees of freedom
## Multiple R-squared:  0.6517, Adjusted R-squared:  0.6259
## F-statistic: 25.26 on 2 and 27 DF,  p-value: 6.551e-07

anova( reg.formaggio, reg1.formaggio )
```

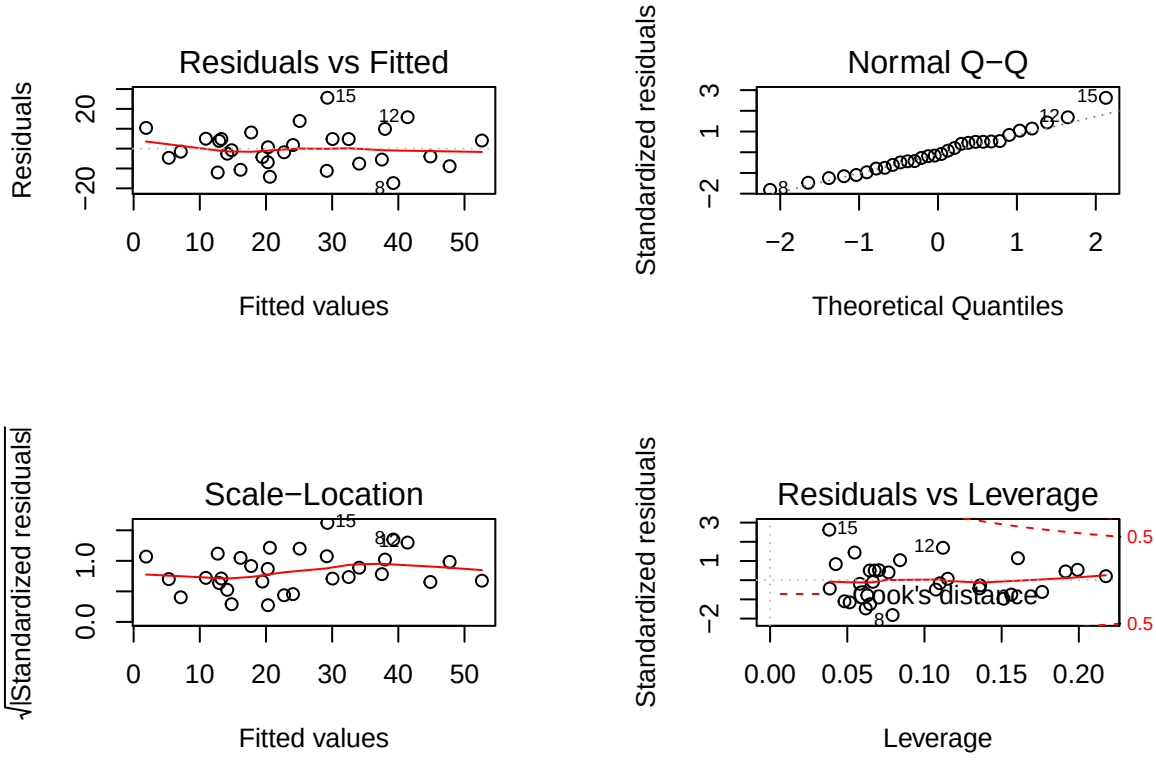
```
## Analysis of Variance Table
##
## Model 1: Taste ~ Acetic + H2S + Lactic
## Model 2: Taste ~ H2S + Lactic
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     26 2668.4
## 2     27 2669.0 -1  -0.55427 0.0054  0.942
```

Sulla riduzione del modello possiamo fare i seguenti commenti:

- R^2 ed R^2_{adj} non sono cambiati in modo sostanziale (rimangono accettabili sebbene non altissimi);
- La significatività complessiva del modello è aumentata;
- I regressori sono ora entrambi molto significativi;
- I gdl (o df) sono 27 come correttamente ci aspettiamo avendo 30 osservazioni e 2 covariate (oltre all'intercetta);
- F-test eseguito con comando `anova` conferma la riduzione del modello, ovvero non c'è differenza nel passare da un modello all'altro (p-value pari a 0.942, molto alto, quindi prediligo modello più semplice).

Verifica delle ipotesi

```
par( mfrow = c( 2, 2 ), mar = c( 5, 4, 5, 4 ) + 0.1 )
plot( reg1.formaggio )
```




```
shapiro.test( reg1.formaggio$residuals )
##
```

```
##  Shapiro-Wilk normality test
##
## data:  reg1.formaggio$residuals
## W = 0.97945, p-value = 0.8107
```

Le assunzioni di Normalità ed omoschedasticità sembrano rispettate.

---

2. Ridge regression

A macroeconomic data set which provides a well-known example for a highly collinear regression. A data frame with 7 economical variables, observed yearly from 1947 to 1962 (n = 16).

- **GDP.deflator**: GDP implicit price deflator (1954 = 100)
- **GDP**: Gross National Product.
- **Unemployed**: number of unemployed.
- **ArmedForce**: number of people in the armed forces.
- **Population**: 'noninstitutionalized' population $\geq$ 14 years of age.
- **Year**: the year (time).
- **Employed**: number of people employed.

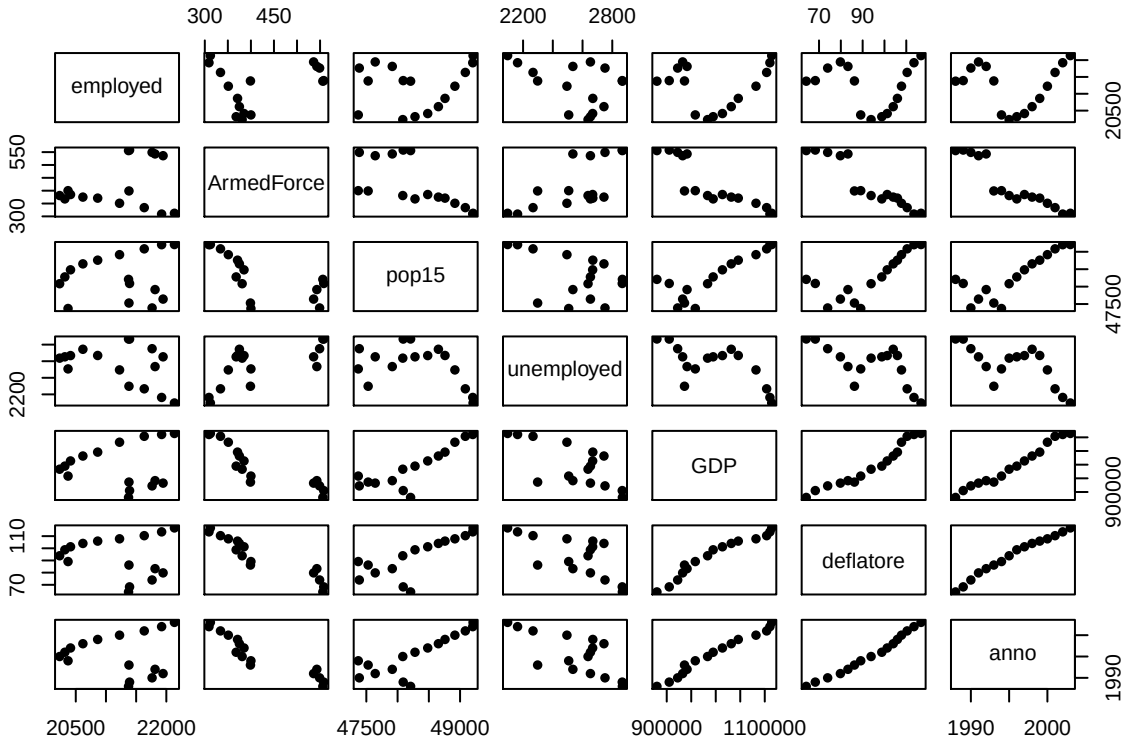
Import and explore the data. Fit the best linear model for predicting the number of employed people (*Employed*).

Solution

```
library( faraway )

#data( longley )
longley = read.table("LONGLEYdata.txt", header=T)

#help( longley )

#str( longley )
```

First of all, we should explore the data (graphically).

```
pairs( longley, pch = 16 )
```



There is a clear linear dependence between: GDP, GDP.deflator, year and Employed.

Let's fit the complete model for predicting the number of people employed.

```
g = lm( employed ~ ., longley )
summary( g )
##
## Call:
## lm(formula = employed ~ ., data = longley)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -341.69 -155.15   26.76  137.76  278.21
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.705e+05  6.077e+05  -0.445 0.666789
## ArmedForce   1.077e+01  2.620e+00   4.110 0.002635 **
## pop15        1.713e-01  2.925e-01   0.586 0.572365
## unemployed  -2.458e+00  4.984e-01  -4.933 0.000811 ***
## GDP          5.042e-03  9.265e-03   0.544 0.599509
## deflatore   -3.897e+01  4.817e+01  -0.809 0.439329
## anno         1.422e+02  3.131e+02   0.454 0.660413
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 249.7 on 9 degrees of freedom
```

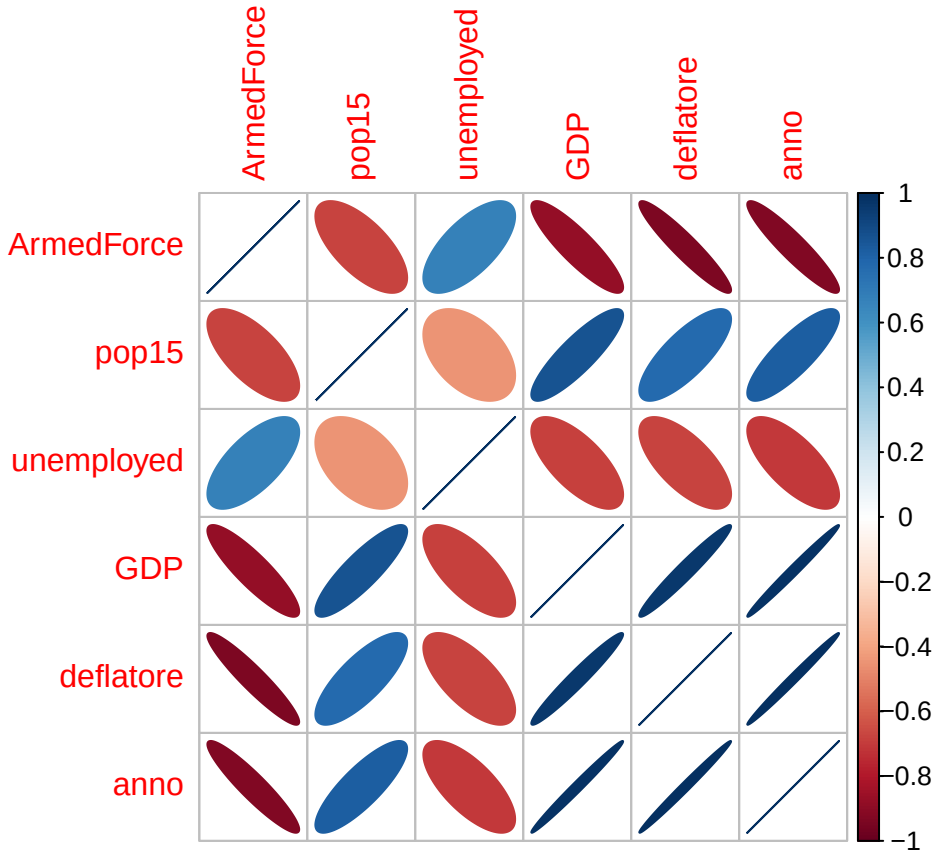
```
## Multiple R-squared:  0.9123, Adjusted R-squared:  0.8538
## F-statistic:  15.6 on 6 and 9 DF,  p-value: 0.0002698
```

It seems a very good model: p-value 0.0002698, Adjusted R-squared 0.8538. However, there are some non-significant covariates (GDP.deflator, Population, GDP, anno). This can be due to the linear dependence that we already detected from pairs plot.

We can investigate possible collinearity among variables in the following ways:

1. Checking the **correlation matrix** first (we round to 3 digits for convenience).

```
corm = round( cor( longley [ , -1 ] ), 3 )
corm
##             ArmedForce  pop15 unemployed    GDP deflatore   anno
## ArmedForce       1.000 -0.672      0.672 -0.879    -0.937 -0.927
## pop15           -0.672  1.000     -0.447  0.863     0.776  0.828
## unemployed       0.672 -0.447      1.000 -0.690    -0.676 -0.704
## GDP             -0.879  0.863     -0.690  1.000     0.968  0.989
## deflatore       -0.937  0.776     -0.676  0.968     1.000  0.991
## anno            -0.927  0.828     -0.704  0.989     0.991  1.000

corrplot( corm, method = 'ellipse' )
```



We immediately see that there is high correlation among: GDP.deflator, GDP, Population and Year.

N.B We have already inferred this result from pairs plot.

2. Checking the **condition number** of $Z^T Z$ through *eigen decomposition*.

```
P = ncol( longley )
N = nrow( longley )
Z = model.matrix(g)

# Stima affidabile
cond = condest( t( Z ) %*% Z )
cond
## $est
## [1] 9.985066e+19
##
## $v
## [1]  9.994145e-01 -2.446539e-06  2.331228e-07 -4.133861e-07  1.239245e-08
## [6]  6.760915e-05 -5.147955e-04

# Oppure, usando la definizione
#(ma la stima NON è affidabile nel caso di matrice mal condizionata):
eigs  = eigen( t( Z ) %*% Z )$values
cond = sqrt( max( eigs ) / min( eigs ) )

cond
## [1] 9747732854
```

The condition number is very high, which confirms that there is high collinearity among variables.

3. VIF.

```
library( car )
vif( g )
## ArmedForce      pop15 unemployed        GDP  deflatore       anno
##  14.266720   8.199176   3.349282 123.203277 149.948808 534.533470
```

There's definitely a lot of variance inflation. For example, we can interpret $\sqrt{123.2} = 11.1$ as telling us that the standard error for GDP is about 11 times larger than it would have been without collinearity.

We can *fix* collinearity problems as follows:

1. Reducing the model.

```
step( g )
## Start:  AIC=181.44
## employed ~ ArmedForce + pop15 + unemployed + GDP + deflatore +
##     anno
##
##              Df Sum of Sq     RSS    AIC
## - anno        1     12864  573979 179.80
## - GDP         1     18466  579580 179.96
## - pop15       1     21399  582514 180.04
## - deflatore   1     40813  601928 180.56
## <none>                     561115 181.44
## - ArmedForce  1   1053372 1614487 196.35
## - unemployed  1   1516861 2077976 200.39
##
## Step:  AIC=179.8
## employed ~ ArmedForce + pop15 + unemployed + GDP + deflatore
##
##              Df Sum of Sq     RSS    AIC
## - deflatore   1     41234  615212 178.91
```

```
## - pop15       1     54568  628547 179.26
## <none>                     573979 179.80
## - GDP         1    149363  723342 181.50
## - ArmedForce  1   1361872 1935851 197.25
## - unemployed  1   2240389 2814367 203.24
##
## Step:  AIC=178.91
## employed ~ ArmedForce + pop15 + unemployed + GDP
##
##              Df Sum of Sq     RSS    AIC
## <none>                      615212 178.91
## - pop15       1    118215  733428 179.73
## - GDP         1    150569  765781 180.42
## - unemployed  1   2623919 3239132 203.49
## - ArmedForce  1   3270537 3885750 206.40
##
## Call:
## lm(formula = employed ~ ArmedForce + pop15 + unemployed + GDP,
##     data = longley)
##
## Coefficients:
## (Intercept)   ArmedForce        pop15   unemployed          GDP
##   3.055e+03    1.131e+01    3.201e-01   -2.672e+00    4.779e-03
```

```
mod_red = lm( employed ~ ArmedForce + pop15 + unemployed + GDP, longley )

summary( mod_red )
```

```
##
## Call:
## lm(formula = employed ~ ArmedForce + pop15 + unemployed + GDP,
##     data = longley)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -447.47 -141.43   46.01  139.04  273.29
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.055e+03  8.143e+03   0.375    0.715
## ArmedForce   1.131e+01  1.479e+00   7.647 1.00e-05 ***
## pop15        3.201e-01  2.202e-01   1.454    0.174
## unemployed  -2.672e+00  3.900e-01  -6.850 2.77e-05 ***
## GDP          4.779e-03  2.913e-03   1.641    0.129
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 236.5 on 11 degrees of freedom
## Multiple R-squared:  0.9038, Adjusted R-squared:  0.8689
## F-statistic: 25.85 on 4 and 11 DF,  p-value: 1.522e-05
```

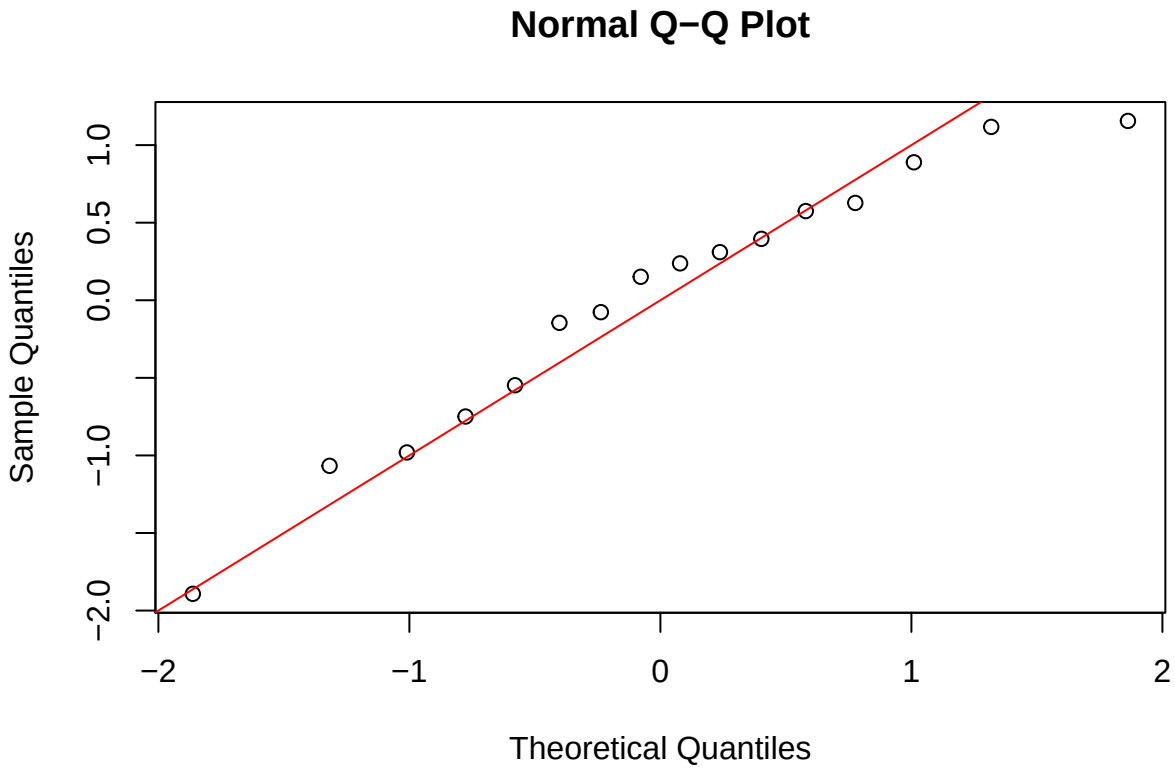
```
vif( mod_red )
```

```
## ArmedForce      pop15 unemployed        GDP
##   5.067703   5.180528   2.287084  13.576344
```

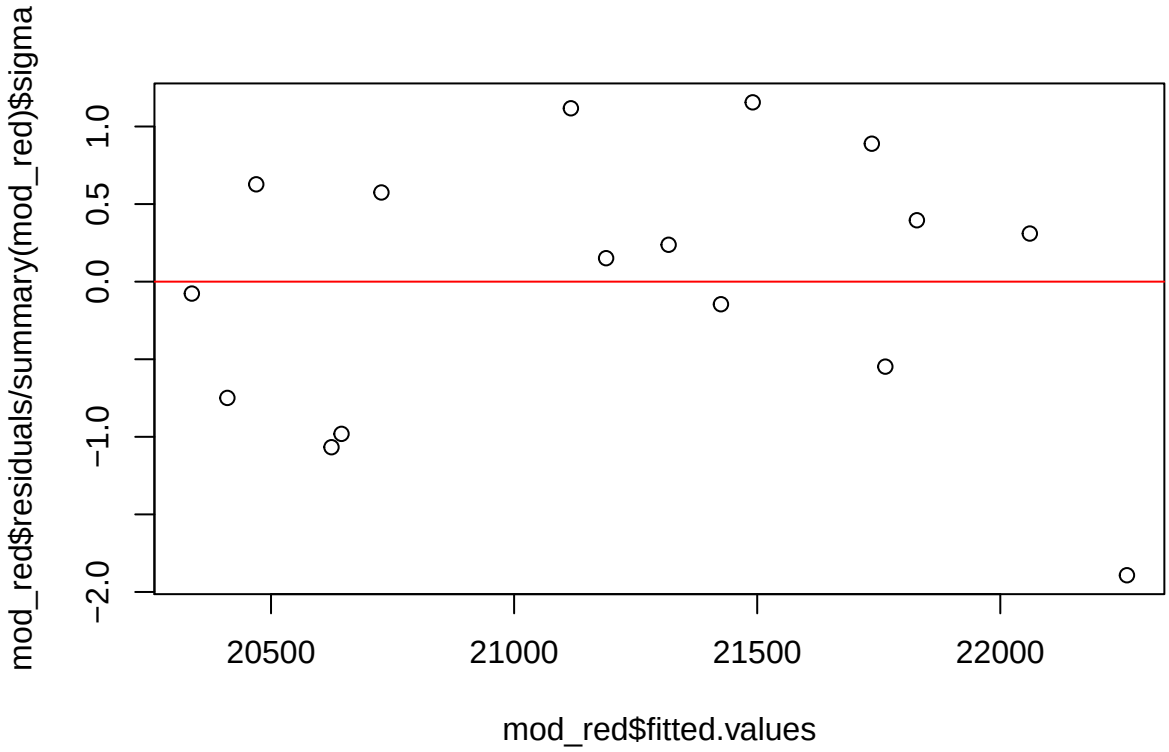
```
cor( model.matrix( mod_red ) [ , -1 ] )
##             ArmedForce      pop15 unemployed        GDP
## ArmedForce   1.0000000 -0.6722391  0.6719497 -0.8792445
## pop15       -0.6722391  1.0000000 -0.4465790  0.8626140
## unemployed   0.6719497 -0.4465790  1.0000000 -0.6896239
## GDP         -0.8792445  0.8626140 -0.6896239  1.0000000
```

Finally, let's chceck the normality and homoskedasticity hypothesis for the reduced model.

```
qqnorm( mod_red$residuals/summary( mod_red )$sigma )
abline( 0, 1, col='red' )
```




```
shapiro.test( mod_red$residuals/summary( mod_red )$sigma )
##
##  Shapiro-Wilk normality test
##
## data:  mod_red$residuals/summary(mod_red)$sigma
## W = 0.95398, p-value = 0.5553


plot( mod_red$fitted.values, mod_red$residuals/summary( mod_red )$sigma )
abline( h = 0, col='red' )
```



Hypotheses are respected, but we have to carefully interpret this output because we have very few data (16).

2. RIDGE REGRESSION: regularizing the OLS (Ordinary Least Squares) problem by *penalising* the magnitude of β term. This corresponds to finding a *biased estimator* of the β vector instead of the unbiased, traditional OLS-based one. In practice, by allowing the estimator to have some bias, we are able to get a lower MSE by reducing the amount of variance of the original, zero-bias but high-variance OLS β estimator.

```
library( MASS )
gr = lm.ridge( employed ~ ., longley, lambda = seq( 0, 1.5, 0.01 ) )

names( gr )
## [1] "coef"   "scales" "Inter"  "lambda" "ym"     "xm"     "GCV"    "kHKB"
## [9] "kLW"
gr$lambda
##   [1] 0.00 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10 0.11 0.12 0.13
##  [15] 0.14 0.15 0.16 0.17 0.18 0.19 0.20 0.21 0.22 0.23 0.24 0.25 0.26 0.27
##  [29] 0.28 0.29 0.30 0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.40 0.41
##  [43] 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.50 0.51 0.52 0.53 0.54 0.55
##  [57] 0.56 0.57 0.58 0.59 0.60 0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69
##  [71] 0.70 0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.80 0.81 0.82 0.83
##  [85] 0.84 0.85 0.86 0.87 0.88 0.89 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97
##  [99] 0.98 0.99 1.00 1.01 1.02 1.03 1.04 1.05 1.06 1.07 1.08 1.09 1.10 1.11
## [113] 1.12 1.13 1.14 1.15 1.16 1.17 1.18 1.19 1.20 1.21 1.22 1.23 1.24 1.25
## [127] 1.26 1.27 1.28 1.29 1.30 1.31 1.32 1.33 1.34 1.35 1.36 1.37 1.38 1.39
## [141] 1.40 1.41 1.42 1.43 1.44 1.45 1.46 1.47 1.48 1.49 1.50
gr$coef
##                 0.00      0.01      0.02      0.03      0.04      0.05
```

```
## ArmedForce  969.1546  950.9332  940.6227  933.5234  927.9981  923.3357
## pop15       104.7189  120.6686  130.7325  138.1627  144.1269  149.1563
## unemployed -563.4934 -571.8876 -576.4711 -579.3887 -581.4058 -582.8673
## GDP         377.0803  436.2093  460.5191  470.7687  474.2008  473.9673
## deflatore  -618.4613 -517.3833 -460.1973 -422.0088 -393.9586 -372.0710
## anno        655.5658  460.6365  358.6532  295.7842  253.0389  222.0071
##                 0.06      0.07      0.08      0.09      0.10      0.11
## ArmedForce  919.1829  915.3475  911.7178  908.2263  904.8302  901.5020
## pop15       153.5322  157.4208  160.9282  164.1266  167.0678  169.7903
## unemployed -583.9536 -584.7696 -585.3812 -585.8324 -586.1541 -586.3687
## GDP         471.6392  468.0742  463.7715  459.0356  454.0588  448.9651
## deflatore  -354.2776 -339.3847 -326.6481 -315.5752 -305.8239 -297.1475
## anno        198.3921  179.7701  164.6703  152.1483  141.5704  132.4952
##                 0.12      0.13      0.14      0.15      0.16       0.17
## ArmedForce  898.2238  894.9836  891.7733  888.5877  885.4231  882.27722
## pop15       172.3236  174.6910  176.9114  179.0003  180.9705  182.83299
## unemployed -586.4934 -586.5411 -586.5223 -586.4454 -586.3172 -586.14358
## GDP         443.8363  438.7265  433.6718  428.6957  423.8137  419.03529
## deflatore  -289.3618 -282.3262 -275.9304 -270.0865 -264.7230 -259.78116
## anno        124.6063  117.6704  111.5121  105.9972  101.0205   96.49931
##                 0.18       0.19      0.20       0.21       0.22       0.23
## ArmedForce  879.1487  876.03654  872.9404  869.85990  866.79518  863.74630
## pop15       184.5971  186.27096  187.8616  189.37510  190.81705  192.19229
## unemployed -585.9292 -585.67835 -585.3944 -585.08049 -584.73927 -584.37309
## GDP         414.3659  409.80808  405.3626  401.02865  396.80454  392.68790
## deflatore  -255.2121 -250.97467 -247.0337 -243.35915 -239.92506 -236.70884
## anno         92.3670   88.56966   85.0631   81.81074   78.78205   75.95133
##                 0.24       0.25       0.26       0.27      0.28
## ArmedForce  860.71343  857.69683  854.69673  851.71339  848.7471
## pop15       193.50517  194.75962  195.95916  197.10702  198.2061
## unemployed -583.98404 -583.57395 -583.14449 -582.69712 -582.2332
## GDP         388.67595  384.76560  380.95363  377.23676  373.6117
## deflatore  -233.69070 -230.85323 -228.18106 -225.66050 -223.2793
## anno         73.29683   70.79996   68.44476   66.21746   64.1061
##                 0.29       0.30       0.31      0.32       0.33      0.34
## ArmedForce  845.79805  842.86651  839.95267  837.0567  834.17883  831.3191
## pop15       199.25911  200.26847  201.23642  202.1651  203.05628  203.9119
## unemployed -581.75390 -581.26034 -580.75352 -580.2343 -579.70363 -579.1622
## GDP         370.07505  366.62373  363.25456  359.9645  356.75061  353.6101
## deflatore  -221.02669 -218.89271 -216.86856 -214.9462 -213.11847 -211.3787
## anno         62.10023   60.19069   58.36943   56.6293   54.96397   53.3678
##                 0.35       0.36       0.37       0.38       0.39
## ArmedForce  828.47772  825.65472  822.85020  820.06422  817.29680
## pop15       204.73349  205.52266  206.28081  207.00929  207.70935
## unemployed -578.61064 -578.04972 -577.47999 -576.90201 -576.31630
## GDP         350.54021  347.53840  344.60215  341.72911  338.91700
## deflatore  -209.72085 -208.13949 -206.62958 -205.18650 -203.80602
## anno         51.83574   50.36325   48.94625   47.58106   46.26435
##                 0.40       0.41       0.42       0.43       0.44
## ArmedForce  814.54796  811.81772  809.10606  806.41294  803.73834
## pop15       208.38215  209.02880  209.65034  210.24774  210.82193
## unemployed -575.72333 -575.12356 -574.51740 -573.90524 -573.28745
## GDP         336.16366  333.46701  330.82510  328.23603  325.69800
```

```
## deflatore  -202.48423 -201.21755 -200.00264 -198.83645 -197.71611
## anno         44.99307   43.76449   42.57607   41.42551   40.31071
##                 0.45       0.46       0.47       0.48       0.49
## ArmedForce  801.08220  798.44446  795.82504  793.22388  790.64087
## pop15       211.37378  211.90411  212.41369  212.90327  213.37356
## unemployed -572.66436 -572.03630 -571.40356 -570.76643 -570.12518
## GDP         323.20931  320.76832  318.37346  316.02324  313.71623
## deflatore  -196.63899 -195.60264 -194.60478 -193.64328 -192.71615
## anno         39.22972   38.18075   37.16216   36.17242   35.21012
##                 0.50       0.51       0.52       0.53       0.54
## ArmedForce  788.07592  785.52894  782.9998  780.48841  777.99463
## pop15       213.82521  214.25886  214.6751  215.07454  215.45770
## unemployed -569.48006 -568.83129 -568.1791 -567.52374 -566.86535
## GDP         311.45108  309.22646  307.0411  304.89394  302.78369
## deflatore  -191.82156 -190.95778 -190.1232 -189.31627 -188.53562
## anno         34.27395   33.36269   32.4752   31.61044   30.76741
##                 0.55       0.56       0.57       0.58       0.59       0.60
## ArmedForce  775.51834  773.0594  770.61774  768.19317  765.78556  763.3948
## pop15       215.82511  216.1773  216.51466  216.83774  217.14695  217.4427
## unemployed -566.20416 -565.5403 -564.87403 -564.20543 -563.53468 -562.8619
## GDP         300.70931  298.6697  296.66400  294.69110  292.75012  290.8402
## deflatore  -187.77991 -187.0479 -186.33837 -185.65028 -184.98256 -184.3342
## anno         29.94518   29.1429   28.35975   27.59497   26.84785   26.1177
##                0.61       0.62       0.63       0.64       0.65       0.66
## ArmedForce  761.0207  758.66316  756.32203  753.9972  751.68842  749.39564
## pop15       217.7254  217.99544  218.25319  218.4990  218.73322  218.95618
## unemployed -562.1873 -561.51096 -560.83300 -560.1535 -559.47272 -558.79064
## GDP         288.9604  287.11003  285.28821  283.4942  281.72731  279.98679
## deflatore  -183.7044 -183.09221 -182.49680 -181.9174 -181.35332 -180.80383
## anno         25.4039   24.70583   24.02295   23.3547   22.70058   22.06011
##                 0.67       0.68       0.69       0.70       0.71
## ArmedForce  747.11869  744.85742  742.61169  740.38134  738.16624
## pop15       219.16819  219.36956  219.56059  219.74156  219.91274
## unemployed -558.10739 -557.42308 -556.73780 -556.05164 -555.36470
## GDP         278.27199  276.58225  274.91695  273.27548  271.65727
## deflatore  -180.26830 -179.74611 -179.23666 -178.73941 -178.25383
## anno         21.43283   20.81831   20.21613   19.62591   19.04727
##                 0.72       0.73       0.74       0.75       0.76
## ArmedForce  735.96624  733.78119  731.61095  729.4554  727.31432
## pop15       220.07441  220.22680  220.37018  220.5048  220.63082
## unemployed -554.67705 -553.98877 -553.29994 -552.6106 -551.92091
## GDP         270.06174  268.48837  266.93662  265.4060  263.89598
## deflatore  -177.77942 -177.31571 -176.86225 -176.4186 -175.98439
## anno         18.47985   17.92331   17.37733   16.8416   16.31583
##                 0.77       0.78       0.79       0.80       0.81
## ArmedForce  725.18765  723.07522  720.97688  718.89251  716.82195
## pop15       220.74854  220.85815  220.95985  221.05385  221.14034
## unemployed -551.23086 -550.54052 -549.84997 -549.15926 -548.46845
## GDP         262.40613  260.93598  259.48510  258.05304  256.63940
## deflatore  -175.55920 -175.14267 -174.73445 -174.33422 -173.94165
## anno         15.79972   15.29301   14.79544   14.30675   13.82672
##                 0.82       0.83       0.84       0.85       0.86
## ArmedForce  714.76509  712.72176  710.69186  708.67523  706.67174
```

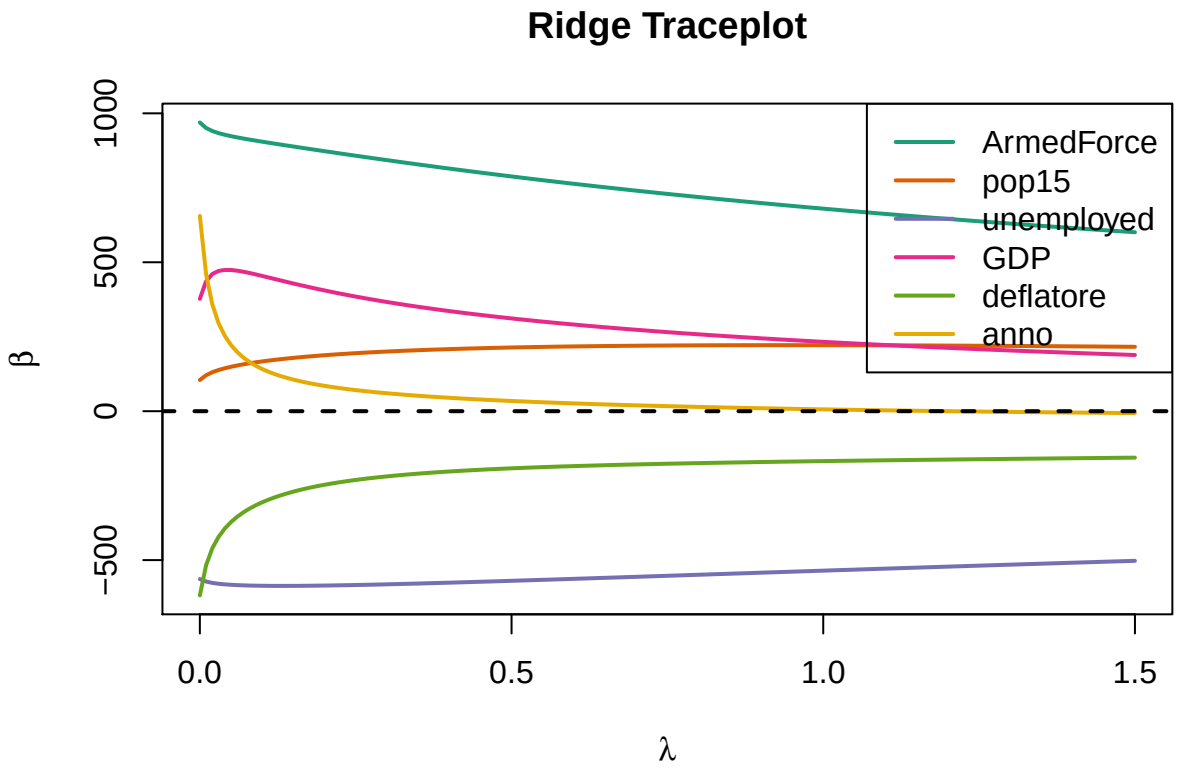
```
## pop15       221.21952  221.29156  221.35665  221.41495  221.46664
## unemployed -547.77760 -547.08675 -546.39596 -545.70528 -545.01474
## GDP         255.24378  253.86579  252.50506  251.16121  249.83390
## deflatore  -173.55644 -173.17831 -172.80698 -172.44218 -172.08368
## anno         13.35511   12.89169   12.43627   11.98862   11.54857
##                  0.87       0.88       0.89       0.90       0.91
## ArmedForce  704.68128  702.70369  700.73887  698.786675  696.846984
## pop15       221.51187  221.55081  221.58361  221.610411  221.631363
## unemployed -544.32440 -543.63430 -542.94448 -542.254985 -541.565842
## GDP         248.52278  247.22753  245.94781  244.683318  243.433750
## deflatore  -171.73123 -171.38461 -171.04359 -170.707964 -170.377542
## anno         11.11591   10.69047   10.27207    9.860537    9.455717
##                  0.92       0.93       0.94       0.95       0.96
## ArmedForce  694.919671  693.004612  691.101687  689.210774  687.331754
## pop15       221.646606  221.656276  221.660505  221.659422  221.653152
## unemployed -540.877095 -540.188776 -539.500922 -538.813564 -538.126733
## GDP         242.198809  240.978209  239.771671  238.578924  237.399706
## deflatore  -170.052130 -169.731548 -169.415620 -169.104181 -168.797071
## anno          9.057449    8.665581    8.279965    7.900459    7.526927
##                  0.97       0.98       0.99       1.00       1.01
## ArmedForce  685.464508  683.608920  681.764874  679.932256  678.110954
## pop15       221.641819  221.625539  221.604429  221.578601  221.548165
## unemployed -537.440461 -536.754775 -536.069705 -535.385277 -534.701518
## GDP         236.233760  235.080837  233.940695  232.813099  231.697819
## deflatore  -168.494138 -168.195237 -167.900227 -167.608975 -167.321351
## anno          7.159235    6.797256    6.440863    6.089938    5.744364
##                  1.02       1.03       1.04       1.05       1.06
## ArmedForce  676.300854  674.501849  672.713827  670.936683  669.170309
## pop15       221.513227  221.473890  221.430257  221.382424  221.330490
## unemployed -534.018451 -533.336103 -532.654495 -531.973650 -531.293590
## GDP         230.594631  229.503317  228.423667  227.355473  226.298535
## deflatore  -167.037233 -166.756502 -166.479046 -166.204755 -165.933526
## anno          5.404026    5.068816    4.738626    4.413355    4.092901
##                  1.07       1.08       1.09       1.10       1.11
## ArmedForce  667.414600  665.669453  663.934765  662.210434  660.496361
## pop15       221.274546  221.214686  221.150996  221.083565  221.012476
## unemployed -530.614336 -529.935907 -529.258322 -528.581601 -527.905762
## GDP         225.252655  224.217643  223.193312  222.179481  221.175971
## deflatore  -165.665257 -165.399854 -165.137222 -164.877275 -164.619926
## anno          3.777167    3.466059    3.159484    2.857355    2.559583
##                  1.12       1.13       1.14       1.15       1.16
## ArmedForce  658.792446  657.098593  655.414703  653.74068  652.076438
## pop15       220.937813  220.859656  220.778083  220.69317  220.604997
## unemployed -527.230821 -526.556794 -525.883699 -525.21155 -524.540362
## GDP         220.182611  219.199230  218.225663  217.26175  216.307335
## deflatore  -164.365094 -164.112699 -163.862667 -163.61492 -163.369401
## anno          2.266084    1.976777    1.691582    1.41042    1.133217
##                  1.17        1.18        1.19        1.20
## ArmedForce  650.4218744  648.7769011  647.1414271  645.51536281
## pop15       220.5136314  220.4191466  220.3216125  220.22109693
## unemployed -523.8701495 -523.2009260 -522.5327047 -521.86549836
## GDP         215.3622616  214.4263814  213.4995475  212.58161665
## deflatore  -163.1260290 -162.8847447 -162.6454852 -162.40819064
```

```
## anno          0.8598986    0.5903929    0.3246302    0.06254245
##                    1.21         1.22         1.23         1.24        1.25
## ArmedForce  643.8986197  642.2911105  640.6927486  639.1034489  637.523127
## pop15       220.1176666  220.0113864  219.9023199  219.7905291  219.676075
## unemployed -521.1993191 -520.5341787 -519.8700883 -519.2070590 -518.545101
## GDP         211.6724489  210.7719074  209.8798583  208.9961708  208.120717
## deflatore  -162.1728032 -161.9392672 -161.7075292 -161.4775375 -161.249243
## anno         -0.1959368   -0.4508721   -0.7023268   -0.9503622   -1.195039
##                    1.26         1.27         1.28         1.29        1.30
## ArmedForce  635.951700  634.389086  632.835202  631.289970  629.753311
## pop15       219.559016  219.439411  219.317316  219.192786  219.065877
## unemployed -517.884224 -517.224439 -516.565754 -515.908178 -515.251719
## GDP         207.253372  206.394013  205.542520  204.698776  203.862666
## deflatore  -161.022596 -160.797553 -160.574068 -160.352098 -160.131602
## anno         -1.436414   -1.674547   -1.909492   -2.141303   -2.370035
##                    1.31         1.32         1.33         1.34        1.35
## ArmedForce  628.22514  626.705395  625.193985  623.690841  622.195887
## pop15       218.93664  218.805127  218.671390  218.535478  218.397438
## unemployed -514.59639 -513.942187 -513.289130 -512.637220 -511.986466
## GDP         203.03408  202.212903  201.399031  200.592359  199.792782
## deflatore  -159.91254 -159.694874 -159.478567 -159.263584 -159.049888
## anno         -2.59574   -2.818467   -3.038266   -3.255187   -3.469277
##                    1.36         1.37         1.38         1.39        1.40
## ArmedForce  620.709050  619.230257  617.759436  616.296517  614.841429
## pop15       218.257320  218.115168  217.971029  217.824947  217.676965
## unemployed -511.336873 -510.688449 -510.041198 -509.395128 -508.750243
## GDP         199.000199  198.214511  197.435622  196.663435  195.897858
## deflatore  -158.837448 -158.626231 -158.416206 -158.207344 -157.999615
## anno         -3.680581   -3.889145   -4.095014   -4.298231   -4.498838
##                    1.41         1.42         1.43         1.44        1.45
## ArmedForce  613.394103  611.954472  610.522466  609.098021  607.681069
## pop15       217.527126  217.375472  217.222044  217.066881  216.910023
## unemployed -508.106548 -507.464050 -506.822751 -506.182658 -505.543773
## GDP         195.138799  194.386169  193.639879  192.899843  192.165977
## deflatore  -157.792992 -157.587447 -157.382955 -157.179490 -156.977029
## anno         -4.696877   -4.892388   -5.085412   -5.275987   -5.464151
##                    1.46         1.47         1.48         1.49        1.50
## ArmedForce  606.271545  604.869386  603.474527  602.086907  600.706461
## pop15       216.751508  216.591374  216.429657  216.266393  216.101618
## unemployed -504.906102 -504.269649 -503.634416 -503.000407 -502.367626
## GDP         191.438198  190.716424  190.000576  189.290575  188.586345
## deflatore  -156.775547 -156.575023 -156.375434 -156.176759 -155.978978
## anno         -5.649941   -5.833395   -6.014547   -6.193433   -6.370088
```

Visualization of ridge regression: trace plots.

```
library(RColorBrewer)
mycol = brewer.pal( 6, 'Dark2' )

# Per gli appassionati, c'è pure il pacchetto wesanderson!
#library(wesanderson)
#mycol = wes_palette( 'Darjeeling', 6, type = 'continuous' )

matplot( gr$lambda, t( gr$coef ), type = "l", xlab = expression( lambda ),
```

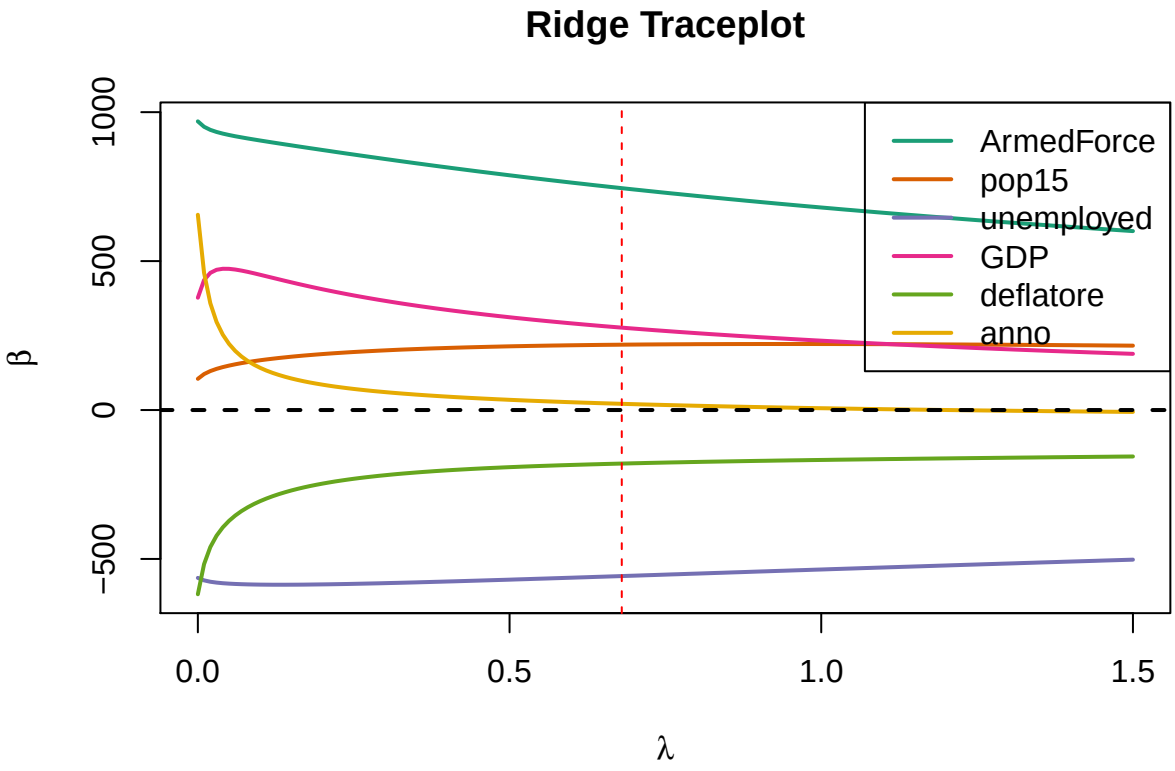
```
         ylab = expression( hat( beta ) ),
         main = "Ridge Traceplot", col = mycol, lwd = 2, lty = 1 )
legend( 'topright', rownames( gr$coef ), col = mycol, lwd = 2, lty = 1 )
abline( h = 0, lwd = 2, lty = 2 )
```



The optimal λ is the value with which all $\hat{\beta}_s$ are stabilized.

Various automatic selection for lambda are possible.

```
select( gr )
## modified HKB estimator is 0.1122239
## modified L-W estimator is 0.6835546
## smallest value of GCV  at 0.23

matplot( gr$lambda, t( gr$coef ), type = "l", xlab = expression( lambda ),
         ylab = expression( hat( beta ) ),
         main = "Ridge Traceplot", col = mycol, lwd = 2, lty = 1 )
legend( 'topright', rownames( gr$coef ), col = mycol, lwd = 2, lty = 1 )
abline( h = 0, lwd = 2, lty = 2 )
abline( v = 0.68, lty = 2, col = 'red' )
```




```
# se si vuole un altro valore di lambda diverso
# da quelli suggeriti dalle procedure automatiche.
gr$coef[ , gr$lam == 0.68 ]
## ArmedForce      pop15 unemployed        GDP  deflatore       anno
##  744.85742  219.36956 -557.42308  276.58225 -179.74611   20.81831
```

The best λ seems to be 0.68.

We now compare the estimates obtained with OLS ($\lambda = 0$) with the ones obtained with the ridge regression.

```
gr$coef[ , 1 ]
## ArmedForce      pop15 unemployed        GDP  deflatore       anno
##   969.1546   104.7189  -563.4934   377.0803  -618.4613   655.5658

abs( ( gr$coef[ , gr$lam == 0.68 ] - gr$coef[ , 1 ] ) / gr$coef[ , 1 ] )
## ArmedForce      pop15 unemployed        GDP  deflatore       anno
## 0.23143591 1.09484201 0.01077259 0.26651626 0.70936565 0.96824375
```

Approfondimento PCR

Volendo percorre la strada della PCR estraiamo le componenti principali (CP)22 delle variabili esplicative. Si ricorda che CP sono delle combinazioni lineari dei predittori, tra loro sono ortogonali e sono correlate con le variabili originarie. Ciascuna CP spiega una quota della varianza delle variabili originarie. Per comodità calcoliamo le CP standardizzando i regressori e operiamo sulla matrice di correlazione:

```
cp = princomp( longley[ ,-1 ], cor = T )
summary( cp )
## Importance of components:
```

```
##                            Comp.1     Comp.2     Comp.3     Comp.4
## Standard deviation     2.2477187 0.75933047 0.53190082 0.27301419
## Proportion of Variance 0.8420399 0.09609713 0.04715308 0.01242279
## Cumulative Proportion  0.8420399 0.93813701 0.98529009 0.99771288
##                             Comp.5      Comp.6
## Standard deviation     0.111174650 0.036917962
## Proportion of Variance 0.002059967 0.000227156
## Cumulative Proportion  0.999772844 1.000000000
```

La prima CP estratta spiega da sola più dell'84% della variabilità dei regressori, mentre le prime due quasi il 94%.

Vediamo ora l'identificazione delle CP esaminando le correlazioni con le variabili originali:

```
#coeff della proiezione delle variabili originarie
#sul sistema di riferimento delle componenti principali
cp$scores
##        Comp.1      Comp.2      Comp.3      Comp.4       Comp.5
## 1   3.3526574  0.80362788 -0.71318913 -0.55741707 -0.001670696
## 2   3.0751177  0.75492040 -0.54116452 -0.17098514 -0.088614339
## 3   2.9409083 -0.26513679  0.05737036  0.45365480 -0.187976148
## 4   2.2670978 -0.34033074 -0.18947722  0.39560268  0.039364031
## 5   1.7172047 -0.45447569 -0.66492003  0.40355785  0.232827990
## 6   0.7972521 -1.79099075  0.21133090 -0.42694498  0.075662100
## 7   0.8985713 -1.15133181  0.84519967  0.02405744 -0.114776382
## 8   0.1925238  0.01446301  0.63937375 -0.21450305 -0.030219259
## 9  -0.2603859  0.28193363  0.66932784 -0.19755442  0.094837191
## 10 -0.5595025  0.59280389  0.44620659  0.02328945  0.113422667
## 11 -0.8763394  1.05610198  0.55987852  0.09340203  0.066698667
## 12 -1.3011597  0.92861566  0.37129049  0.14246587  0.062364279
## 13 -2.1024357  0.52143818  0.00637250  0.11820524 -0.136761821
## 14 -2.9093194 -0.08462291 -0.47168357  0.02066286 -0.144537598
## 15 -3.4692982 -0.33348219 -0.57004895 -0.12387303 -0.057455875
## 16 -3.7628923 -0.53353376 -0.65586721  0.01637947  0.076835192
##          Comp.6
## 1   0.025861822
## 2   0.008310234
## 3   0.010989386
## 4  -0.039485806
## 5  -0.013981832
## 6  -0.002352473
## 7   0.058664637
## 8  -0.040946632
## 9  -0.052185718
## 10 -0.012866893
## 11  0.012726199
## 12  0.063658253
## 13 -0.013211209
## 14 -0.050174003
## 15 -0.017892087
## 16  0.062886122
cp$loadings
##
## Loadings:
##            Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
```

```
## ArmedForce  0.414        -0.574  0.669  0.189
## pop15      -0.373  0.572 -0.586 -0.398  0.171
## unemployed  0.334  0.802  0.478  0.102
## GDP        -0.438  0.120         0.429 -0.703 -0.332
## deflatore  -0.436         0.290  0.370  0.659 -0.392
## anno       -0.443         0.102  0.250         0.853
##
##                Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6
## SS loadings     1.000  1.000  1.000  1.000  1.000  1.000
## Proportion Var  0.167  0.167  0.167  0.167  0.167  0.167
## Cumulative Var  0.167  0.333  0.500  0.667  0.833  1.000
```

la prima CP è correlata in modo abbastanza forte con tutti i regressori, con alcuni positivamente (Armed.force e unemployed) e con la maggior parte negativamente; la seconda CP è correlata positivamente con pop15 e unemployed.

Stimiamo la regressione multipla della variabile risposta sulle CP:

```
fmcp = lm( longley$employed ~ cp$scores )
summary( fmcp )
##
## Call:
## lm(formula = longley$employed ~ cp$scores)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -341.69 -155.15   26.76  137.76  278.21
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)     21212.62      62.42 339.821  < 2e-16 ***
## cp$scoresComp.1   -11.49      27.77  -0.414   0.6887
## cp$scoresComp.2  -237.19      82.21  -2.885   0.0180 *
## cp$scoresComp.3 -1026.01     117.36  -8.743 1.08e-05 ***
## cp$scoresComp.4   645.71     228.64   2.824   0.0199 *
## cp$scoresComp.5  -413.19     561.49  -0.736   0.4805
## cp$scoresComp.6   724.88    1690.86   0.429   0.6782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 249.7 on 9 degrees of freedom
## Multiple R-squared:  0.9123, Adjusted R-squared:  0.8538
## F-statistic:  15.6 on 6 and 9 DF,  p-value: 0.0002698
```

Da cui emerge un legame statisticamente significativo tra la risposta employed e le CP 2, 3 e 4.

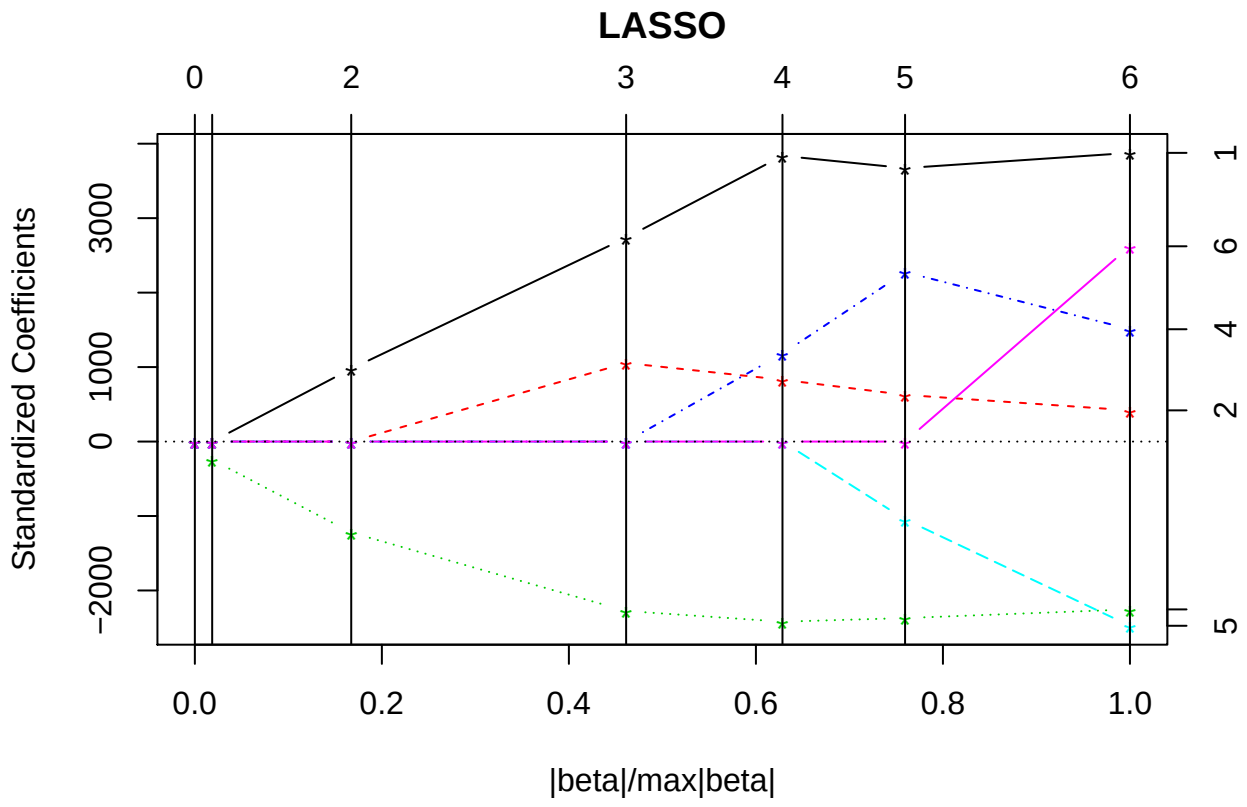
Stimiamo il modello solo con questi regressori:

```
fmcp2 = lm( longley$employed ~ cp$scores[ ,2 ] + cp$scores[ ,3 ] + cp$scores[ ,4 ] )

summary( fmcp2 )
##
## Call:
## lm(formula = longley$employed ~ cp$scores[, 2] + cp$scores[,
##     3] + cp$scores[, 4])
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -450.22 -117.41   26.91  162.19  254.42
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    21212.62      56.69 374.196  < 2e-16 ***
## cp$scores[, 2]  -237.19      74.66  -3.177  0.00796 **
## cp$scores[, 3] -1026.01     106.58  -9.627 5.39e-07 ***
## cp$scores[, 4]   645.71     207.64   3.110  0.00902 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 226.8 on 12 degrees of freedom
## Multiple R-squared:  0.9036, Adjusted R-squared:  0.8795
## F-statistic: 37.48 on 3 and 12 DF,  p-value: 2.258e-06
```

Approfondimento LASSO Regression

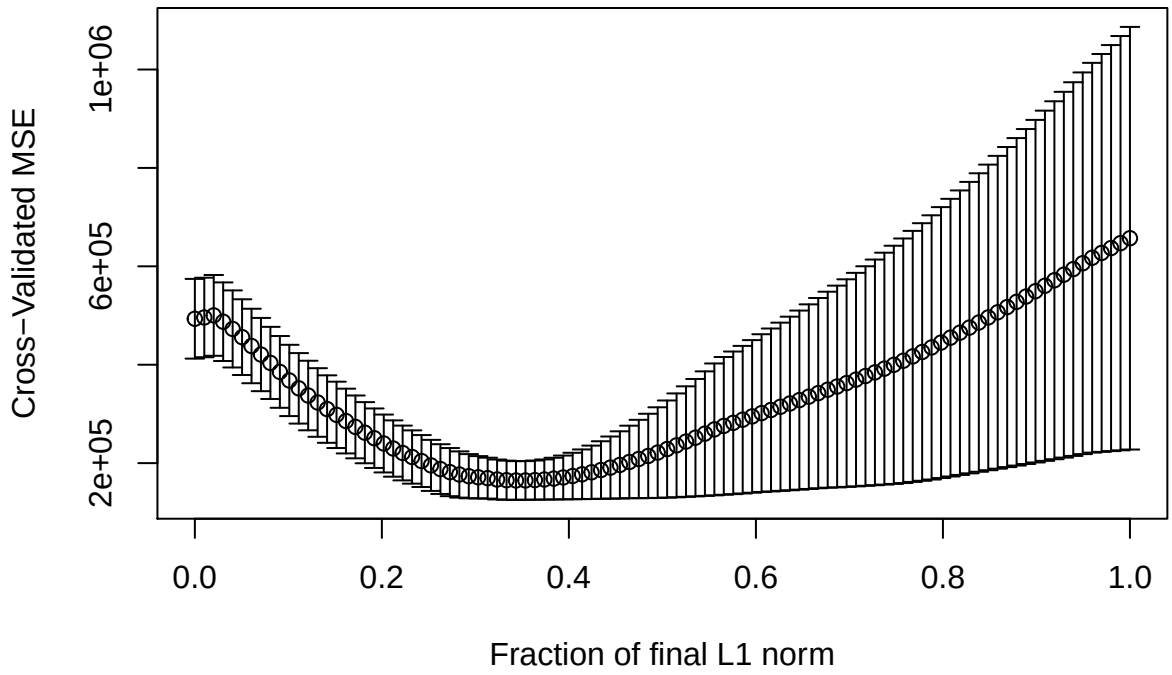
```
#library(lars)
longley.lasso = lars( as.matrix( longley[ ,-1 ] ), longley$employed,
                      type = "lasso", trace = FALSE )



plot( longley.lasso, plottype = 'coefficients' )
```



```
G = cv.lars( as.matrix( longley[ ,-1 ] ), longley$employed, type = "lasso", trace = TRUE, K = 10 )
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Lasso Step 7 :    Variable 4     dropped
## LARS Step 8 :     Variable 4     added
## Computing residuals, RSS etc .....
##
##  CV Fold 1
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Computing residuals, RSS etc .....
##
##  CV Fold 2
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Lasso Step 7 :    Variable 2     dropped
## LARS Step 8 :     Variable 2     added
## Computing residuals, RSS etc .....
##
##  CV Fold 3
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Computing residuals, RSS etc .....
##
##  CV Fold 4
##
```

```
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Computing residuals, RSS etc .....
##
##  CV Fold 5
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## Lasso Step 6 :    Variable 2     dropped
## LARS Step 7 :     Variable 6     added
## LARS Step 8 :     Variable 2     added
## Computing residuals, RSS etc .....
##
##  CV Fold 6
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Computing residuals, RSS etc .....
##
##  CV Fold 7
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Computing residuals, RSS etc .....
##
##  CV Fold 8
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 3     added
```

```
## LARS Step 2 :     Variable 1     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Lasso Step 7 :    Variable 4     dropped
## LARS Step 8 :     Variable 4     added
## Computing residuals, RSS etc .....
##
##  CV Fold 9
##
## LASSO sequence
## Computing X'X .....
## LARS Step 1 :     Variable 1     added
## LARS Step 2 :     Variable 3     added
## LARS Step 3 :     Variable 2     added
## LARS Step 4 :     Variable 4     added
## LARS Step 5 :     Variable 5     added
## LARS Step 6 :     Variable 6     added
## Computing residuals, RSS etc .....
##
##  CV Fold 10
```




```
coef( longley.lasso )
##      ArmedForce     pop15 unemployed         GDP deflatore     anno
## [1,]    0.00000 0.0000000  0.0000000 0.000000000   0.00000   0.0000
```

```
## [2,]    0.00000 0.0000000 -0.2659467 0.000000000   0.00000   0.0000
## [3,]    2.71642 0.0000000 -1.3325281 0.000000000   0.00000   0.0000
## [4,]    7.60238 0.4313253 -2.4814604 0.000000000   0.00000   0.0000
## [5,]   10.65864 0.3396009 -2.6383048 0.003942121   0.00000   0.0000
## [6,]   10.19913 0.2564128 -2.5852689 0.007606092 -16.47462   0.0000
## [7,]   10.76747 0.1713461 -2.4581399 0.005042031 -38.97406 142.2122
```

Per trovare l'ottimo, indaghiamo la CV MSE minore. Vediamo che il min CV MSE è tra 0.4 e 0.6, che corrisponde a selezionare 3 covariate, in particolare: ArmedForce, pop15, unemployed.

Ex. per casa: Prostate Cancer Data

Data to examine the correlation between the level of prostate-specific antigen and a number of clinical measures in men who were about to receive a radical prostatectomy.

A data frame with 97 observations on the following 10 variables.

- **lcavol**: log cancer volume;
- **lweight**: log prostate weight;
- **age**: in years;
- **lbph**: log of the amount of benign prostatic hyperplasia;
- **svi**: seminal vesicle invasion;
- **lcp**:log of capsular penetration;
- **gleason**:a numeric vector;
- **pgg45**:percent of Gleason score 4 or 5;
- **lpsa**:response;
- **train**:a logical vector.

The last column indicates which 67 observations were used as the "training set" and which 30 as the test set, as described on page 48 in the book.

Source *Stamey, T., Kabalin, J., McNeal, J., Johnstone, I., Freiha, F., Redwine, E. and Yang, N (1989) Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate II. Radical prostatectomy treted patients, Journall of Urology 16: 1076?1083.*

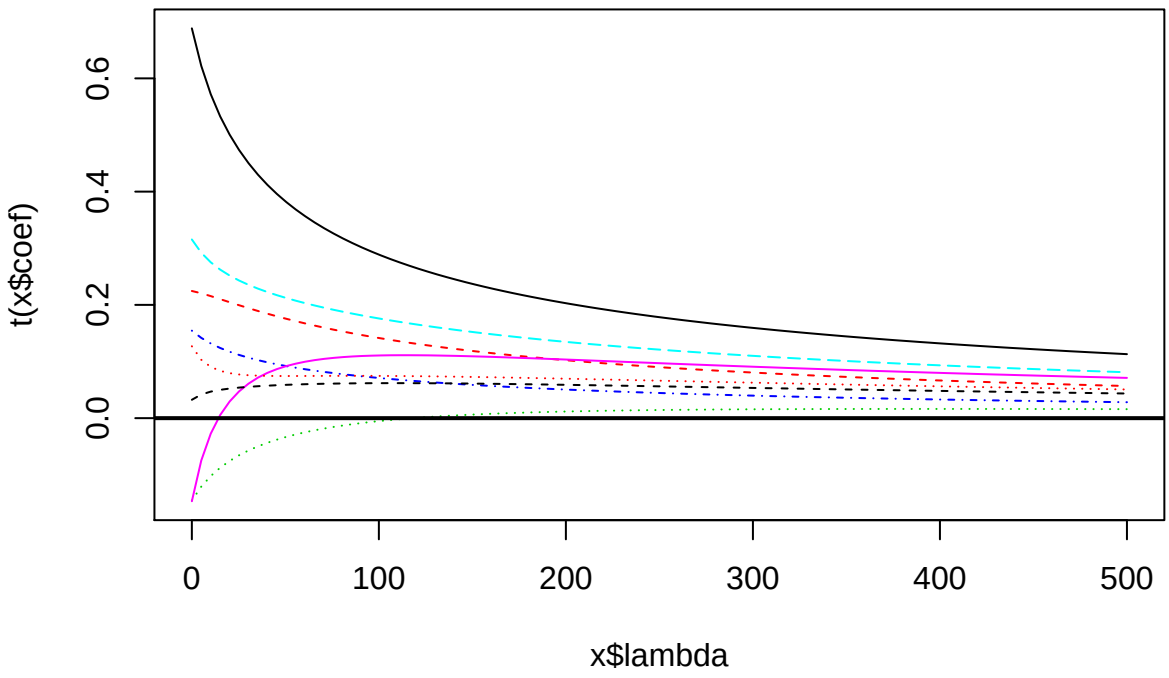
Investigate collinearity among predictors and fit a ridge and LASSO regression on data.

Solution

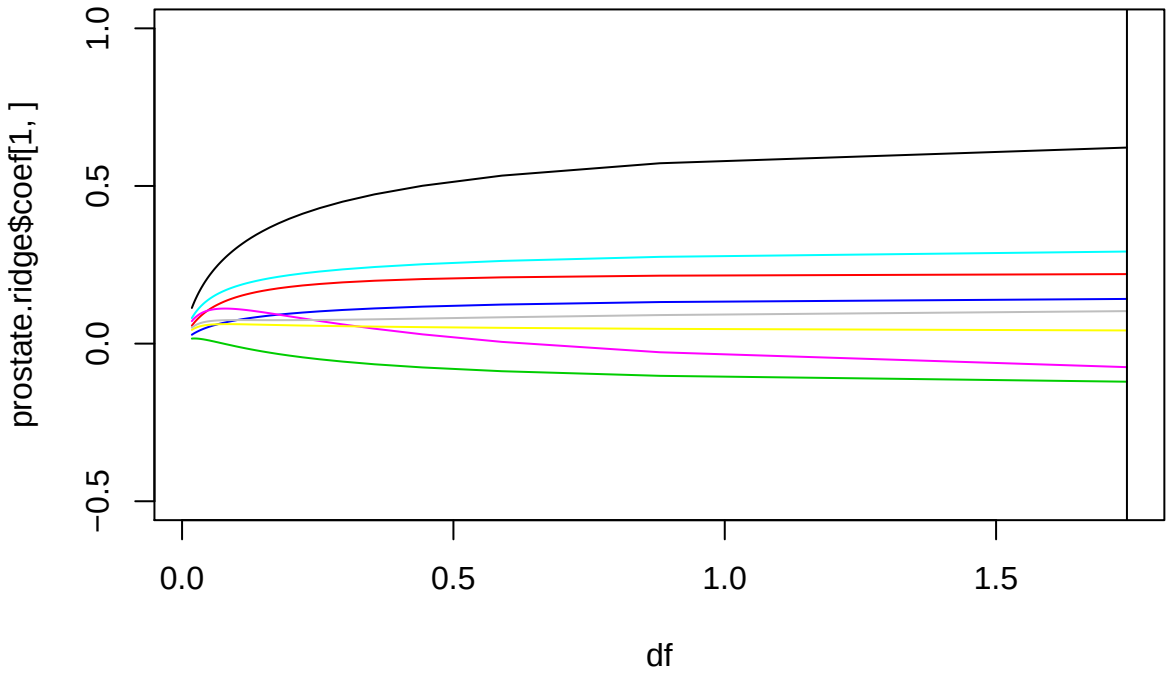
```
data( prostate )
prostate[ 1:15, ]
##        lcavol lweight age      lbph svi      lcp gleason pgg45     lpsa
## 1  -0.5798185  2.7695  50 -1.386294   0 -1.38629       6     0 -0.43078
## 2  -0.9942523  3.3196  58 -1.386294   0 -1.38629       6     0 -0.16252
## 3  -0.5108256  2.6912  74 -1.386294   0 -1.38629       7    20 -0.16252
## 4  -1.2039728  3.2828  58 -1.386294   0 -1.38629       6     0 -0.16252
## 5   0.7514161  3.4324  62 -1.386294   0 -1.38629       6     0  0.37156
```

```
## 6  -1.0498221  3.2288  50 -1.386294   0 -1.38629       6     0  0.76547
## 7   0.7371641  3.4735  64  0.615186   0 -1.38629       6     0  0.76547
## 8   0.6931472  3.5395  58  1.536867   0 -1.38629       6     0  0.85442
## 9  -0.7765288  3.5395  47 -1.386294   0 -1.38629       6     0  1.04732
## 10  0.2231436  3.2445  63 -1.386294   0 -1.38629       6     0  1.04732
## 11  0.2546422  3.6041  65 -1.386294   0 -1.38629       6     0  1.26695
## 12 -1.3470736  3.5987  63  1.266948   0 -1.38629       6     0  1.26695
## 13  1.6134299  3.0229  63 -1.386294   0 -0.59784       7    30  1.26695
## 14  1.4770487  2.9982  67 -1.386294   0 -1.38629       7     5  1.34807
## 15  1.2059708  3.4420  57 -1.386294   0 -0.43078       7     5  1.39872
```

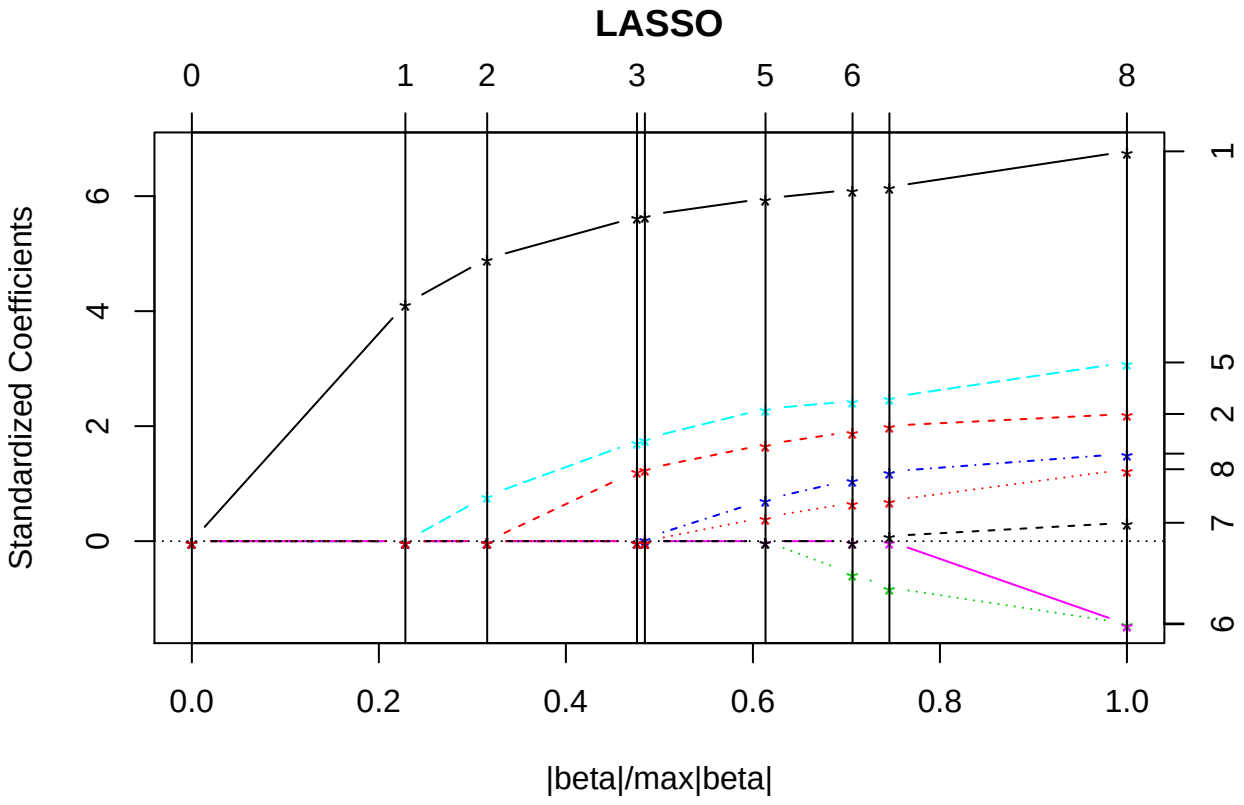
```
# Ridge Regression

X = as.matrix( prostate[ ,1:8 ] )
XtX = solve( t( X ) %*% X )

autoval = eigen(XtX)
autoval$val   # differenze di anche 3 ordini di grandezza
## [1] 1.235614e-01 4.940922e-02 2.245990e-02 1.543245e-02 5.693695e-03
## [6] 4.741489e-03 1.615325e-05 2.087323e-06

num.cond = sqrt(max(autoval$val)/min(autoval$val))
num.cond
## [1] 243.3025

H = hat( X )

lambda = seq( 0, 500, len = 100 )


# Gradi di libertà equivalenti

df = rep( 0, length( lambda ) )

for ( i in 1 : length( lambda ) ){

  v = diag( H )/( diag( H ) + lambda[ i ] )
    df[ i ] = sum( v )

}


prostate.ridge = lm.ridge( prostate[ , 9 ] ~ as.matrix( prostate[ , 1:8 ] ),lambda = lambda )

# Shrinkage dei coeff
plot( prostate.ridge )
abline( h=0, lwd = 2 )
```



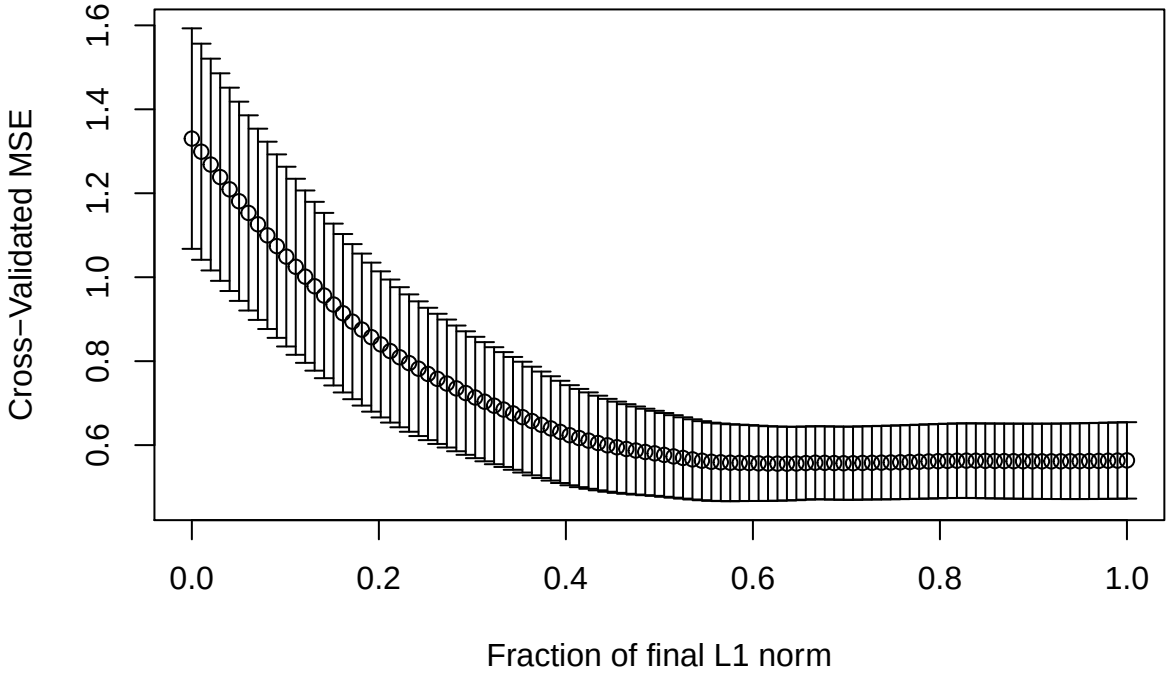

```
# Coeff vs gradi di libert? equivalenti
plot( df, prostate.ridge$coef[1,],type='l',ylim=c(-0.5,1))

for (j in 2:8){

  points( df, prostate.ridge$coef[ j, ], col = j, type = 'l' )

}

# Scelta del parametro
select( prostate.ridge )
## modified HKB estimator is 4.256152
## modified L-W estimator is 3.487311
## smallest value of GCV  at 5.050505

abline( v = df[ which.min( prostate.ridge$GCV ) ] )

l = prostate.ridge$kHKB
v = diag( H )/( diag( H ) + l )
dfl = sum( v )
abline( v = dfl, col = 2 )
```




```
###############################################################################

# Lasso Regression

prostate.lasso = lars( as.matrix( prostate[ , 1:8 ] ), as.matrix( prostate[ ,9 ] ),
                       type = "lasso", trace = FALSE )


plot( prostate.lasso )
```




```
# Scelta del parametro
G = cv.lars( as.matrix( prostate[ ,1:8 ] ), as.matrix( prostate[ ,9 ] ),
             type = "lasso", trace = FALSE, K = 10 )
```




```
coef( prostate.lasso )
##          lcavol   lweight          age        lbph       svi        lcp
##  [1,] 0.0000000 0.0000000  0.000000000 0.000000000 0.0000000  0.0000000
##  [2,] 0.3573071 0.0000000  0.000000000 0.000000000 0.0000000  0.0000000
##  [3,] 0.4257203 0.0000000  0.000000000 0.000000000 0.1947693  0.0000000
##  [4,] 0.4881368 0.2520802  0.000000000 0.000000000 0.4284335  0.0000000
##  [5,] 0.4905852 0.2576089  0.000000000 0.003588721 0.4397184  0.0000000
##  [6,] 0.5161125 0.3456512  0.000000000 0.050958368 0.5673424  0.0000000
##  [7,] 0.5293694 0.3920498 -0.007809141 0.075452800 0.6005655  0.0000000
##  [8,] 0.5333556 0.4126074 -0.011058519 0.085221900 0.6160889  0.0000000
##  [9,] 0.5870218 0.4544674 -0.019637176 0.107054031 0.7661573 -0.1054743
##          gleason       pgg45
##  [1,] 0.00000000 0.000000000
##  [2,] 0.00000000 0.000000000
##  [3,] 0.00000000 0.000000000
##  [4,] 0.00000000 0.000000000
##  [5,] 0.00000000 0.000000000
##  [6,] 0.00000000 0.001507659
##  [7,] 0.00000000 0.002410603
##  [8,] 0.01253885 0.002552384
##  [9,] 0.04514160 0.004525231
```

Il CV MSE minore si realizza per $\lambda \geq 0.6$. Scegliamo quindi il modello più semplice che ha 5 predittori: lcavol, lweight, lbph, svi, lcp, pgg45.