Prof. Dr. E. Cramer, Prof. Dr. M. Kateri,
L. Kaufmann, M.Sc., T. van Bentum, M.Sc. ,
A. Müllenmeister

RWTH Aachen, SS 2021
Release date: May 11th, 2021
Solution: May 18 th, 2021

Applied Data Analysis

R-Laboratory 5

Implementing Normal Linear Models – Linear Models Beyond Normality

Useful packages and functions:

- `model.matrix()`
- `qr()`
- `qr.solve()`
- `chol2inv()`
- `qt()`
- `rexp()`
- `density()`

Task 17

Let $\boldsymbol{y} = (y_1, \dots, y_n)'$ be a realization of the random sample $\boldsymbol{Y} = (Y_1, \dots, Y_n)'$. For $\boldsymbol{y}$, consider the linear model $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where $\boldsymbol{X}$ is a fixed $(n \times d)$-model (or design) matrix with $d < n$ and rank d, $\boldsymbol{\beta}$ is a $(d \times 1)$-vector of model parameters and $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 I_n)$ with $\sigma^2 > 0$.

Implement your own linear model R-function. This function should use a model formula and a `data.frame` object as input and should deliver the following output: least squares estimates $\hat{\boldsymbol{\beta}}$ of model parameters $\boldsymbol{\beta}$, $\widehat{\text{var}}(\hat{\boldsymbol{\beta}})$, the unbiased estimator $\hat{\sigma}^2$ of σ^2, and $R^2 = \frac{SSR}{SST}$.

Remark: Here we write $\|\boldsymbol{y} - y^*\|_2^2 = \|\hat{\boldsymbol{\mu}} - y^*\|_2^2 + \|\boldsymbol{y} - \hat{\boldsymbol{\mu}}\|_2^2$ as SST = SSR + SSE (total sum of squares = sum of squares due to the regression + sum of squared errors), where $y^* = \overline{y}$ for a model with intercept parameter and $y^* = 0$ for models without.

Hints:

(a) Use the function `model.matrix` to create the design matrix from the right hand side of the given formula.

(b) Let $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{R}$ be a $\boldsymbol{Q}\boldsymbol{R}$-decomposition of the matrix $\boldsymbol{X}$ into a upper triangular $(d \times d)$-matrix $\boldsymbol{R}$ and a matrix $\boldsymbol{Q}$ of the first d columns of an orthogonal $(n \times n)$-matrix. Use the fact that the least squares estimator is given by $\hat{\boldsymbol{\beta}} = \boldsymbol{R}^{-1}\boldsymbol{Q}'\boldsymbol{y}$ to implement a numerically more stable procedure than that using the classical representation $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$.

Test your function on the data set `Solar` and calculate 90%-confidence intervals for the differences $\alpha_{i+1} - \alpha_i$, $i = 1, \dots, 3$, of the factor levels $1, \dots, 4$ of `batch` treated as factor (cf. Task 15).

Task 18

Set the seed to 2020 and repeat the following procedure $n = 1000$ times for $N = 10,\ 20,\ 50$:

(a) Generate samples $x_{11}, \ldots, x_{1N}$ from a uniform distribution on $(0, 40)$.

(b) Generate samples $x_{21}, \ldots, x_{2N}$ from $\mathcal{N}(15, 10^2)$.

(c) Generate error values $\varepsilon_1, \ldots, \varepsilon_N$ from the distribution of the random variable ε, where

$$\frac{\varepsilon}{5} + 1 \sim \mathrm{Exp}(1)$$

and $\mathrm{Exp}(1)$ denotes the exponential distribution with parameter 1.

(d) Generate a sample $y_1, \ldots, y_N$ by setting

$$y_i = \beta_1 + \beta_2 x_{1i} + \beta_3 x_{2i}, \qquad i = 1, \ldots, N,$$

where $\beta_1 = 35$, $\beta_2 = 0.5$ and $\beta_3 = -0.1$.

(e) Estimate $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_3)$ using the least squares estimator $\hat{\boldsymbol{\beta}}$.

(f) Compute the norm-difference $\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|$ of the true parameter vector and its estimate.

Store the values of $\hat{\beta}_2$ and $\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|$ for all n generated data sets and all $N = 10,\ 20,\ 50$. Then illustrate some results using the following graphics.

(i) Create boxplots of the $\|\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}\|$ for the three values of N.

(ii) Create plots of the estimated densities of $\hat{\beta}_2$ for the three values of N and add the curve of the $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$ density, where $\hat{\mu}, \hat{\sigma}^2$ are the mean value and the standard deviation of the sample of $\hat{\beta}_2$ values for the current N.

What do you observe?

```
#############
#
# Task 17
#
#############

my.lm<-function(formula,data){
  y=data[,as.character(formula[[2]])]
  X=model.matrix(formula,data = data)
  param.names=colnames(X)

  model.with.intercep=any(colnames(X)=="(Intercept)")
  X=matrix(X,ncol=ncol(X))
  p.columns=ncol(X)
  n.observations=nrow(X)
  degrees.of.freedom=n.observations-p.columns

  qr.decomp.X=qr(X)
  beta.hat=qr.solve(qr.decomp.X,y)
  beta.hat=matrix(beta.hat)
  mu.hat=X%*%beta.hat
  residuals=y-mu.hat
  SSE=sum((residuals)**2)
  y.star=ifelse(model.with.intercep,mean(y),0)
  TSS=sum((y-y.star)**2)
  sigma.hat.2=SSE/degrees.of.freedom
  R.2=(TSS-SSE)/TSS
  beta.hat.cov=sigma.hat.2 * chol2inv(qr.decomp.X$qr)

  rownames(beta.hat)=param.names
  rownames(beta.hat.cov)=param.names
  colnames(beta.hat.cov)=param.names

  erg=list(beta.hat=beta.hat,beta.hat.cov=beta.hat.cov,R.2=R.2,sigma.hat.2=sigma.hat.2)
}


load("Solar.RData")
model<-my.lm(Pmax~factor(batch)-1,data=solar)
print(model)
print(lm(Pmax~factor(batch)-1,data=solar))

degrees.of.freedom=length(solar$Pmax)-length(model$beta.hat)

t_quantile=qt(0.95,degrees.of.freedom)

printf <- function(...) cat(sprintf(...))

for(i in 1:3){
  contrast=matrix(rep(0,length(model$beta.hat)),nrow = 1)
  contrast[1,i]=-1
  contrast[1,i+1]=1
  CI.center=contrast%*%model$beta.hat
  CI.step.from.center=t_quantile*sqrt(contrast%*%model$beta.hat.cov%*%t(contrast))
  printf("CI for factor level difference of levels %d and %d: [%2.4f,%2.4f]\n",i,i+1,CI.center-CI.step.from.center,CI.center+CI.step.from.center)
}
```

```
#############
#
# Task 18
#
#############

set.seed(2020)
n = 1000
sigma = 5
N_vec = c(10,20,50)
beta = c(35,.5,-.1)
estim.true.norm.diff = matrix(NA,n,length(N_vec))
beta2.estims = matrix(NA,n,length(N_vec))
for (j in seq_along(N_vec)){
  N = N_vec[j]
  for(i in 1:n){
    # (a)
    x1 = runif(N,0,40)

    # (b)
    x2 = rnorm(N,15,10)

    # (c)
    eps = (rexp(N)-1)*sigma

    # (d)
    mu = beta[1]+beta[2]*x1+beta[2]*x2
    y = mu + eps

    # (e)
    lm.fit = lm(y~x1+x2)
    beta.hat = lm.fit$coefficients
    beta2.estims[i,j] = beta.hat[2]

    # (f)
    estim.true.norm.diff[i,j] = sqrt(sum((beta-beta.hat)^2))
  }
}

# (i)
boxplot(estim.true.norm.diff)

# (ii)
plot(density(beta2.estims[,1]))
curve(dnorm(x,mean=mean(beta2.estims[,1]),sd=sd(beta2.estims[,1])),add=TRUE,col="red")

plot(density(beta2.estims[,2]))
curve(dnorm(x,mean=mean(beta2.estims[,2]),sd=sd(beta2.estims[,2])),add=TRUE,col="red")

plot(density(beta2.estims[,3]))
curve(dnorm(x,mean=mean(beta2.estims[,3]),sd=sd(beta2.estims[,3])),add=TRUE,col="red")
```