

# Module 2: Recommended Exercises - Solution

TMA4268 Statistical Learning V2021

Michail Spitieris, Emma Skarstein, Stefanie Muff  
Department of Mathematical Sciences, NTNU

January 26, 2021

---

## Problem 1

a)

```
library(ISLR)
Auto = subset(Auto, select = -name)
# Auto$origin = factor(Auto$origin)
summary(Auto)
```

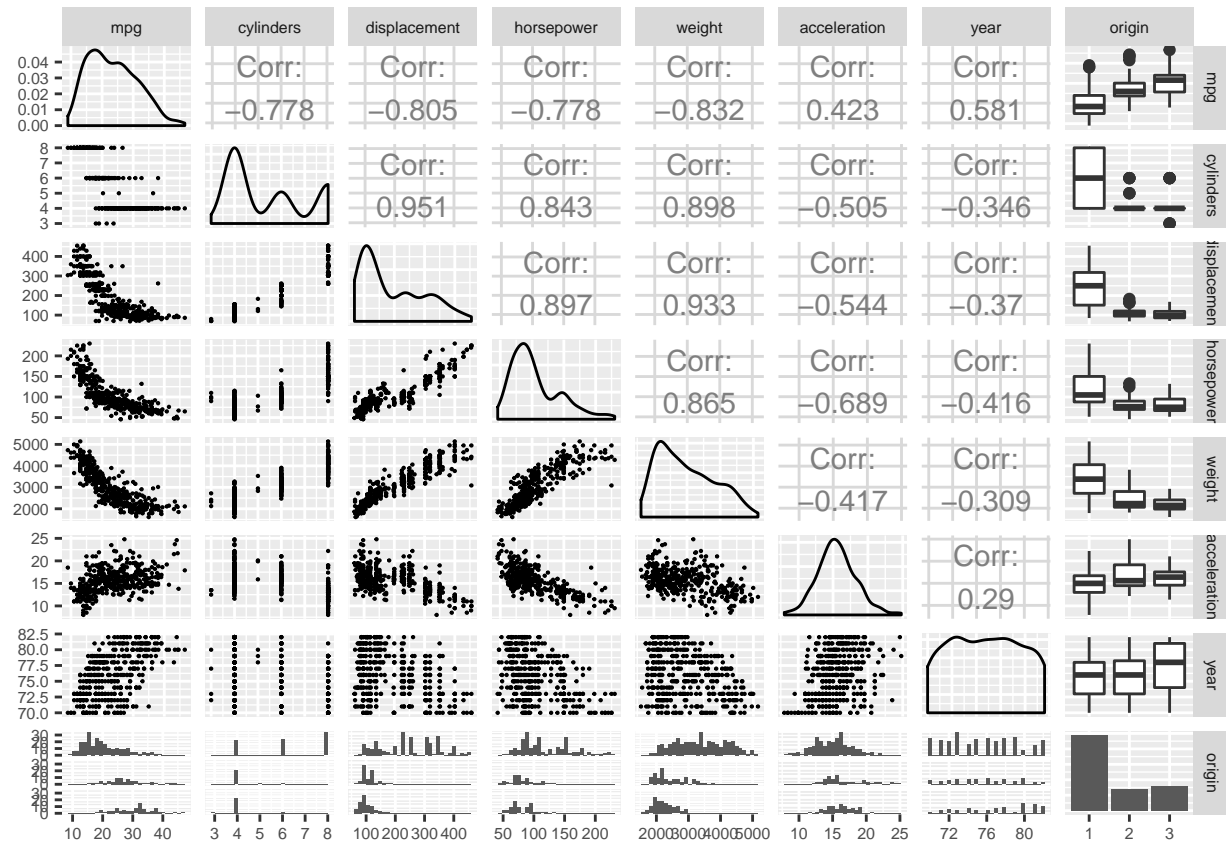
```
##      mpg      cylinders  displacement  horsepower
##  Min.   : 9.00   Min.   :3.000   Min.   : 68.0   Min.   : 46.0
## 1st Qu.:17.00   1st Qu.:4.000   1st Qu.:105.0   1st Qu.: 75.0
## Median :22.75   Median :4.000   Median :151.0   Median : 93.5
## Mean   :23.45   Mean   :5.472   Mean   :194.4   Mean   :104.5
## 3rd Qu.:29.00   3rd Qu.:8.000   3rd Qu.:275.8   3rd Qu.:126.0
## Max.   :46.60   Max.   :8.000   Max.   :455.0   Max.   :230.0
##      weight  acceleration      year      origin
##  Min.   :1613   Min.   : 8.00   Min.   :70.00   Min.   :1.000
## 1st Qu.:2225   1st Qu.:13.78   1st Qu.:73.00   1st Qu.:1.000
## Median :2804   Median :15.50   Median :76.00   Median :1.000
## Mean   :2978   Mean   :15.54   Mean   :75.98   Mean   :1.577
## 3rd Qu.:3615   3rd Qu.:17.02   3rd Qu.:79.00   3rd Qu.:2.000
## Max.   :5140   Max.   :24.80   Max.   :82.00   Max.   :3.000
```

```
str(Auto)
```

```
## 'data.frame':   392 obs. of  8 variables:
## $ mpg      : num  18 15 18 16 17 15 14 14 14 15 ...
## $ cylinders : num   8  8  8  8  8  8  8  8  8  8 ...
## $ displacement: num  307 350 318 304 302 429 454 440 455 390 ...
## $ horsepower  : num  130 165 150 150 140 198 220 215 225 190 ...
## $ weight      : num 3504 3693 3436 3433 3449 ...
## $ acceleration: num  12 11.5 11 12 10.5 10 9 8.5 10 8.5 ...
## $ year        : num  70 70 70 70 70 70 70 70 70 70 ...
## $ origin      : num   1  1  1  1  1  1  1  1  1  1 ...
```

```
Auto$origin = factor(Auto$origin)

library(GGally)
ggpairs(Auto, lower = list(continuous = wrap("points", size=0.1))) + # change points size
  theme(text = element_text(size = 7)) # change text size
```



b)

Correlation matrix for the Auto data set where we omit column 8:

```
cor(Auto[, -c(8)])
```

|              | mpg        | cylinders  | displacement | horsepower | weight     | acceleration | year       |
|--------------|------------|------------|--------------|------------|------------|--------------|------------|
| mpg          | 1.0000000  | -0.7776175 | -0.8051269   | -0.7784268 | -0.8322442 | 0.4233285    | 0.5805410  |
| cylinders    | -0.7776175 | 1.0000000  | 0.9508233    | 0.8429834  | 0.8975273  | -0.5046834   | -0.3456474 |
| displacement | -0.8051269 | 0.9508233  | 1.0000000    | 0.8972570  | 0.9329944  | -0.5438005   | -0.3698552 |
| horsepower   | -0.7784268 | 0.8429834  | 0.8972570    | 1.0000000  | 0.8645377  | -0.6891955   | -0.4163615 |
| weight       | -0.8322442 | 0.8975273  | 0.9329944    | 0.8645377  | 1.0000000  | -0.4168392   | -0.3091199 |
| acceleration | 0.4233285  | -0.5046834 | -0.5438005   | -0.6891955 | -0.4168392 | 1.0000000    | 0.2903161  |
| year         | 0.5805410  | -0.3456474 | -0.3698552   | -0.4163615 | -0.3091199 | 0.2903161    | 1.0000000  |

```
## year                0.2903161  1.0000000
```

c)

```
fit.lm = lm(mpg ~ ., data = Auto)
summary(fit.lm)

##
## Call:
## lm(formula = mpg ~ ., data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -9.0095 -2.0785 -0.0982  1.9856 13.3608
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.795e+01  4.677e+00  -3.839 0.000145 ***
## cylinders   -4.897e-01  3.212e-01  -1.524 0.128215
## displacement  2.398e-02  7.653e-03   3.133 0.001863 **
## horsepower  -1.818e-02  1.371e-02  -1.326 0.185488
## weight      -6.710e-03  6.551e-04 -10.243 < 2e-16 ***
## acceleration  7.910e-02  9.822e-02   0.805 0.421101
## year         7.770e-01  5.178e-02  15.005 < 2e-16 ***
## origin2       2.630e+00  5.664e-01   4.643 4.72e-06 ***
## origin3       2.853e+00  5.527e-01   5.162 3.93e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.307 on 383 degrees of freedom
## Multiple R-squared:  0.8242, Adjusted R-squared:  0.8205
## F-statistic: 224.5 on 8 and 383 DF,  p-value: < 2.2e-16

# kable(broom::tidy(fit.lm))
```

- i. The F-statistic in the very last line of the summary output shows that it is extremely unlikely that we would see such a result if none of the variables had any explanatory power – the  $p$ -value is  $< 2.2e - 16$ ! So we can be sure that the set of predictors is explaining the response to a quite large degree. This is also confirmed by both a quite high  $R^2$  and  $R^2_{adjust}$  (both around 0.82).
- ii. Yes, there is very strong evidence for a relationship between the weight of a car and the response. The  $p$ -value is extremely small. The interpretation is that, if a car weights 1000 kg more, we have a decrease of  $1000 \cdot (-6.710e - 03) = -6.71$  miles per gallon (mpg), that is, the car can drive 6.71 miles less far per gallon of fuel.
- iii. The coefficient  $\beta_{year} = 0.77$  coefficient suggests that a new model has higher mpg compared to an older one. A one year newer model thus should, in average, be able to drive 0.77 miles longer per gallon fuel compared to the older car.

d)

Remember that if we want to test whether a factor variable with more than two levels should be removed from the model, we actually have to test wheter all slope coefficients that are associated with the respective binary dummy variables are zero simultaneously. Look again at equation (3.30) in the course book (p.85).

Consequently, we have to test here whether  $\beta_{\text{origin}2} = \beta_{\text{origin}3} = 0$  at the same time, and for this we need the  $F$ -test, which is implemented in the `anova()` function in R:

```
anova(fit.lm)

## Analysis of Variance Table
##
## Response: mpg
##
```

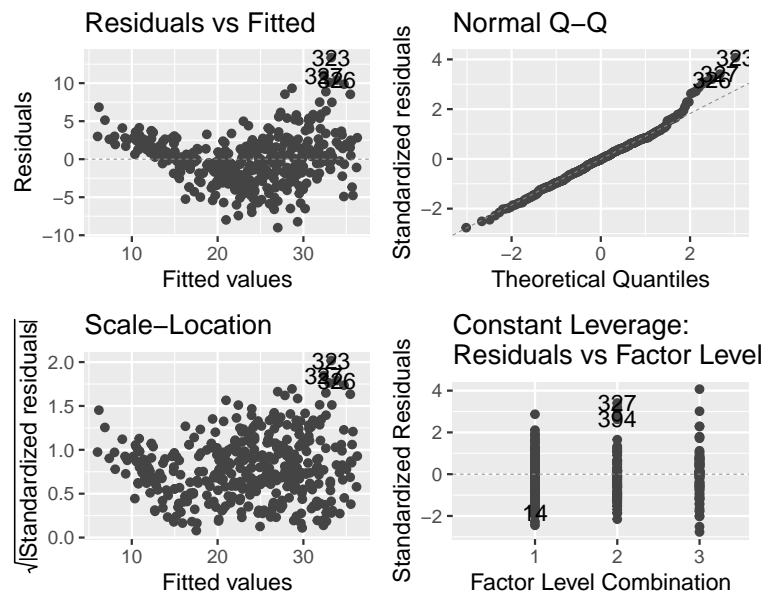
|              | Df  | Sum Sq  | Mean Sq | F value   | Pr(>F)        |
|--------------|-----|---------|---------|-----------|---------------|
| cylinders    | 1   | 14403.1 | 14403.1 | 1317.3788 | < 2.2e-16 *** |
| displacement | 1   | 1073.3  | 1073.3  | 98.1735   | < 2.2e-16 *** |
| horsepower   | 1   | 403.4   | 403.4   | 36.8977   | 3.004e-09 *** |
| weight       | 1   | 975.7   | 975.7   | 89.2447   | < 2.2e-16 *** |
| acceleration | 1   | 1.0     | 1.0     | 0.0884    | 0.7664        |
| year         | 1   | 2419.1  | 2419.1  | 221.2650  | < 2.2e-16 *** |
| origin       | 2   | 356.0   | 178.0   | 16.2787   | 1.639e-07 *** |
| Residuals    | 383 | 4187.4  | 10.9    |           |               |

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The  $p$ -value associated with `origin` is now the one we can look at. Clearly,  $p$  is very small, thus the origin of the car has an influence on the response `mpg`.

e)

```
library(ggfortify)
autoplot(fit.lm, smooth.colour = NA)
```

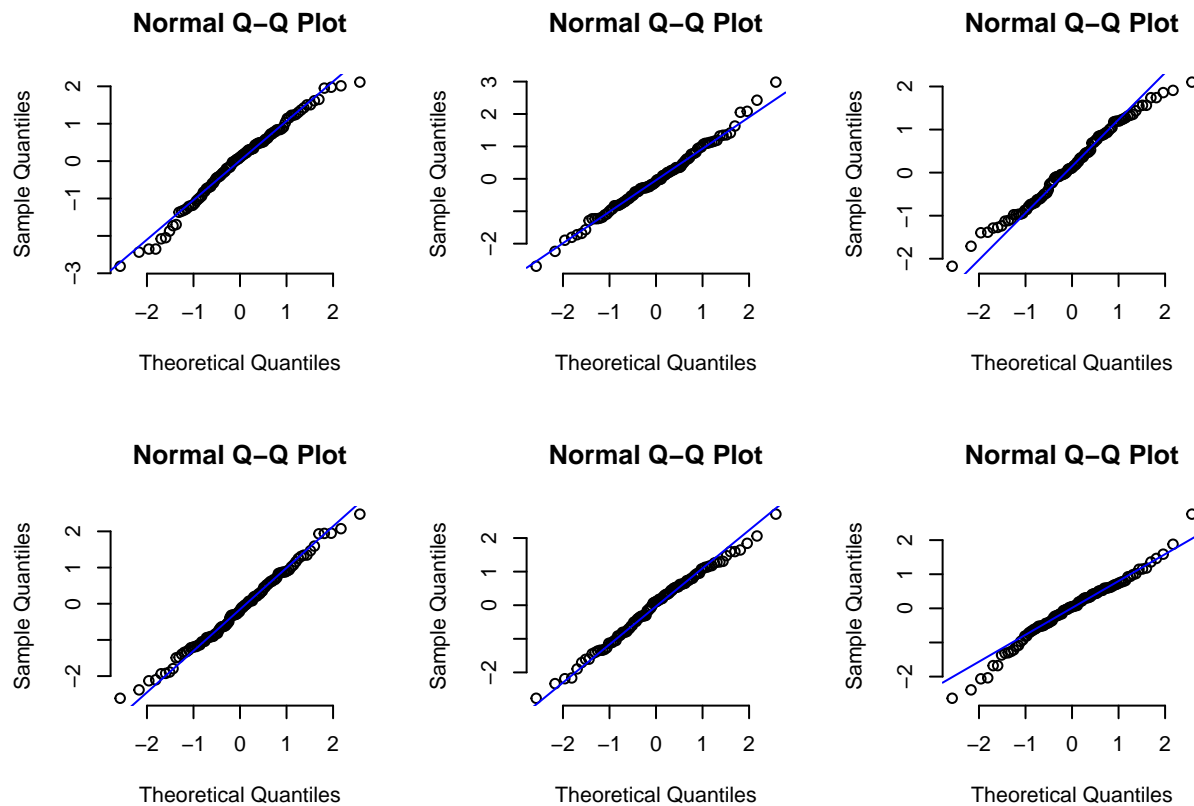


- In the residual vs fitted plot (the so-called Tukey-Anscombe plot) there is evidence of non-linearity.
- Observation 14 has an unusually high leverage. This does not necessarily need to be a problem, but it would be wise to double-check that this observation is not an outlier.

f)

```
set.seed(2332)
n = 100

par(mfrow = c(2, 3))
for (i in 1:6) {
  sim = rnorm(n)
  qqnorm(sim, pch = 1, frame = FALSE)
  qqline(sim, col = "blue", lwd = 1)
}
```



g)

```
fit.lm1 = lm(mpg ~ displacement + weight + year * origin, data = Auto)
summary(fit.lm1)
```

```
##
## Call:
## lm(formula = mpg ~ displacement + weight + year * origin, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -8.7710 -2.0204 -0.0207  1.7045 13.0017
##
## Coefficients:
```

```
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.117e+00  5.259e+00  -0.973 0.331220
## displacement  4.803e-03  5.032e-03   0.955 0.340420
## weight      -6.685e-03  5.543e-04 -12.060 < 2e-16 ***
## year         6.152e-01  6.614e-02   9.302 < 2e-16 ***
## origin2      -3.735e+01  1.026e+01  -3.642 0.000307 ***
## origin3      -2.532e+01  9.441e+00  -2.682 0.007631 **
## year:origin2  5.187e-01  1.342e-01   3.865 0.000130 ***
## year:origin3  3.564e-01  1.213e-01   2.937 0.003514 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.257 on 384 degrees of freedom
## Multiple R-squared:  0.829, Adjusted R-squared:  0.8259
## F-statistic: 265.9 on 7 and 384 DF, p-value: < 2.2e-16
```

Again, if we want to find evidence whether the interaction term between year and origin is needed, we are actually testing whether two slope coefficients are zero at the same time ( $\beta_{\text{year:origin1}} = \beta_{\text{year:origin2}} = 0$ ). Consequently, we need the  $F$ -test, which is implemented in the `anova()` function:

```
anova(fit.lm1)

## Analysis of Variance Table
##
## Response: mpg
##           Df Sum Sq Mean Sq  F value    Pr(>F)
## displacement  1 15440.2 15440.2 1455.6706 < 2.2e-16 ***
## weight        1  1208.5  1208.5  113.9364 < 2.2e-16 ***
## year          1  2601.4  2601.4  245.2550 < 2.2e-16 ***
## origin        2   296.9   148.5   13.9977 1.356e-06 ***
## year:origin   2   198.9    99.5    9.3764 0.0001057 ***
## Residuals    384  4073.1    10.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is very strong evidence that the year-effect depends on the origin of the car, as can be seen by the  $F$ -test that is given by the anova table ( $p = 0.0001057$ ). For European (2) and Japanese (3) cars, it seems that the fuel consumption (mpg) has a steeper slope for year:  $\beta_{\text{year}} = 0.615$  for the reference category 1 (American), whereas  $\beta_{\text{year}} = 0.615 + 0.519$  and  $\beta_{\text{year}} = 0.615 + 0.356$  for category 2 (European) and 3 (Japanese), respectively. This means that the fuel consumption is reduced faster by cars produced outside America (note that mpg is “miles per gallon”, thus a higher value means that the car consumes less fuel, as it can drive further per gallon).

Note: For a full understanding of interaction terms, you really do need both the `summary()` and the `anova()` tables.

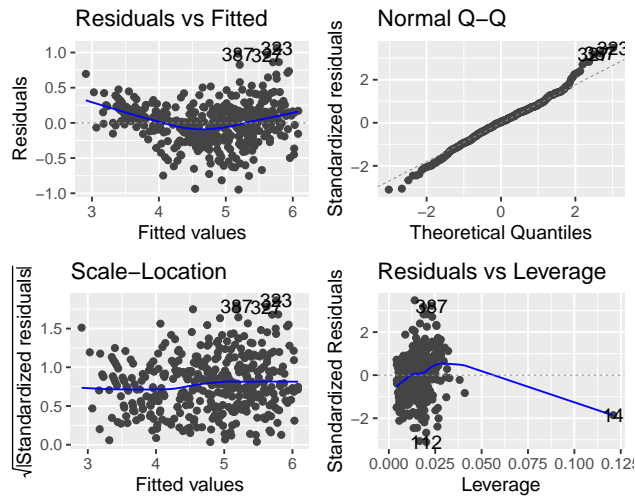
## h)

We try three different transformations and look at the residual plots:

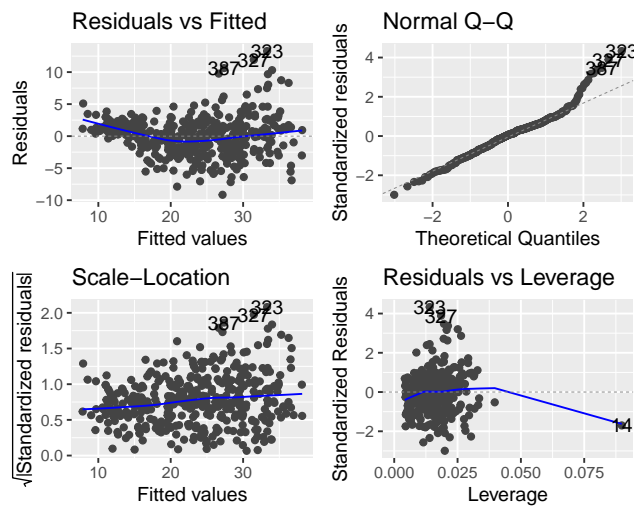
```
# try 3 predictor transformations

Auto$sqrtmpg <- sqrt(Auto$mpg)

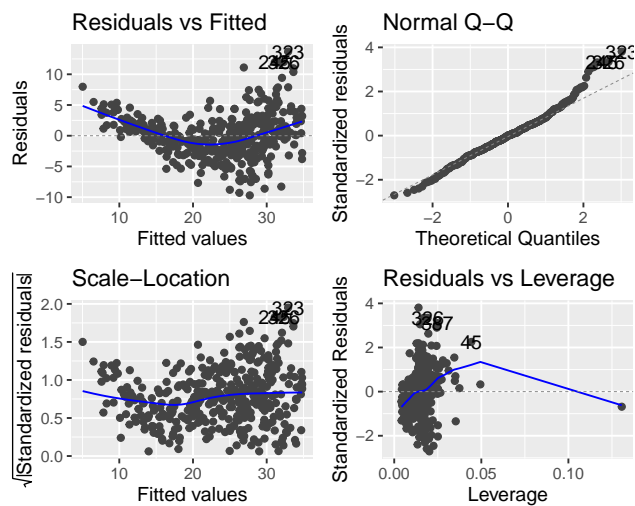
fit.lm3 = lm(sqrtmpg ~ displacement + weight + year + origin, data = Auto)
autoplot(fit.lm3)
```



```
fit.lm4 = lm(mpg ~ displacement + I(log(weight)) + year + origin, data = Auto)
autoplot(fit.lm4)
```



```
fit.lm5 = lm(mpg ~ displacement + I(weight^2) + year + origin, data = Auto)
autoplot(fit.lm5)
```



## Problem 2

a)

$$E(\hat{\beta}) = E((X^T X)^{-1} X^T Y) = (X^T X)^{-1} X^T E(Y) = (X^T X)^{-1} X^T E(X\beta + \varepsilon) \quad (1)$$

$$= (X^T X)^{-1} X^T (X\beta + 0) = (X^T X)^{-1} (X^T X)\beta = I\beta = \beta \quad (2)$$

$$Cov(\hat{\beta}) = Cov((X^T X)^{-1} X^T Y) = (X^T X)^{-1} X^T Cov(Y) ((X^T X)^{-1} X^T)^T \quad (3)$$

$$= (X^T X)^{-1} X^T \sigma^2 I ((X^T X)^{-1} X^T)^T \quad (4)$$

$$= \sigma^2 (X^T X)^{-1} \quad (5)$$

$$(6)$$

We need to assume that  $Y$  is multivariate normal. As  $\hat{\beta}$  is a linear transformation of a multivariate normal vector  $Y$ ,  $\hat{\beta}$  is also multivariate normal.

All components of a multivariate normal vector are themselves univariate normal. This means that  $\hat{\beta}_j$  is normally distributed with expected value given by the  $\beta_j$  and the variance given by the  $j$ 'th diagonal element of  $\sigma^2 (X^T X)^{-1}$ .

b)

Fix covariates  $X$  and repeat the following precedure many times ( $nsim = 1000$ )

- collect  $Y$
- create CI using  $\hat{\beta}$  and  $\hat{\sigma}$

95 % of the times the CI contains the true  $\beta$ .

```
# CI for beta_j
beta0 = 1
beta1 = 3
true_beta = c(beta0, beta1) # vector of model coefficients
true_sd = 1 # choosing true sd
X = runif(100, 0, 1)
Xmat = model.matrix(~X, data = data.frame(X)) # Design Matrix

ci_int = ci_x = 0 # Counts how many times the true value is within the confidence interval
nsim = 1000
for (i in 1:nsim) {
  y = rnorm(n = 100, mean = Xmat %*% true_beta, sd = rep(true_sd, 100))
  mod = lm(y ~ x, data = data.frame(y = y, x = X))
  ci = confint(mod)
  ci_int[i] = ifelse(true_beta[1] >= ci[1, 1] && true_beta[1] <= ci[1, 2],
    1, 0)
  ci_x[i] = ifelse(true_beta[2] >= ci[2, 1] && true_beta[2] <= ci[2, 2], 1,
    0)
}

c(mean(ci_int), mean(ci_x))
```



```
## [1] 0.955 0.947
```

c)

We apply the same idea from b), but now we fix  $X$  and  $x_0$  and

- collect  $Y$
- create PI using  $\hat{\beta}$  and  $\hat{\sigma}$
- simulate  $Y_0$

95 % of the times the PI contains  $Y_0$ .

```
# PI for Y_0
beta0 = 1
beta1 = 3
true_beta = c(beta0, beta1) # vector of model coefficients
true_sd = 1 # choosing true sd
X = runif(100, 0, 1)
Xmat = model.matrix(~X, data = data.frame(X)) # Design Matrix

x0 = c(1, 0.4)

# simulating and fitting models many times
pi_y0 = 0
nsim = 1000
for (i in 1:nsim) {
  y = rnorm(n = 100, mean = Xmat %*% true_beta, sd = rep(true_sd, 100))
  mod = lm(y ~ x, data = data.frame(y = y, x = X))
  y0 = rnorm(n = 1, mean = x0 %*% true_beta, sd = true_sd)
  pi = predict(mod, newdata = data.frame(x = x0[2]), interval = "predict")[2:3]
  pi_y0[i] = ifelse(y0 >= pi[1] && y0 <= pi[2], 1, 0)
}

mean(pi_y0)

## [1] 0.945
```

d)

95 % CI for  $\mathbf{x}_0^T \beta$ : Same idea as for  $\beta_j$ . Use that  $\mathbf{x}_0^T \hat{\beta} \sim N(\mathbf{x}_0^T \beta, \mathbf{x}_0^T \text{Var}(\hat{\beta}) \mathbf{x}_0)$  and do as for  $\beta_j$ . Note that  $\mathbf{x}_0$  is a vector. The connection between CI for  $\beta$ ,  $\mathbf{x}_0^T \beta$  and PI for  $Y$  at  $\mathbf{x}_0$ : The first is CI for a parameter, the second is CI for the expected regression line in the point  $\mathbf{x}_0$  (when you only have one covariate, this may be more intuitive), and the last is the PI for the response  $Y_0$ . The difference between the two latter is that  $Y$  are the observations, and  $\mathbf{x}_0^T \beta$  is the expected value of the observations and hence a function of the model parameters (NOT an observation).

e)

We have a model on the form  $Y = X\beta + \varepsilon$  where  $\varepsilon$  is the error. The error of the model is unknown and unobserved, but we can estimate it by what we call the residuals. The residuals are given by the difference

between the true response and the predicted value

$$\hat{\varepsilon} = Y - \hat{Y} = (I - \underbrace{X(X^T X)^{-1} X^T}_{=H})Y,$$

where  $H$  is called the “hat matrix”.

Properties of raw residuals: Normally distributed with mean 0 and covariance  $Cov(\hat{\varepsilon}) = \sigma^2(I - H)$ . This means that the residuals may have different variance (depending on  $X$ ) and may also be correlated.

In a model check, we want to check that our errors are independent, homoscedastic (same variance for each observation) and not dependent on the covariates. As we don’t know the true error, we use the residuals as predictors, but as mentioned, the residuals may have different variances and may be correlated. This is why we don’t want to use the raw residuals for model check.

To amend our problem we need to try to fix the residuals so that they at least have equal variances. We do that by working with standardized or studentized residuals.