Solutions for Exercise Set 1

Note that the model answers contain more code than is required from the actual answers. The code is included for your reference (for the full source code, see the corresponding R markdown file). Some questions in the problem sheet are marked "optional": you should not reduce points if an answer to such a question is missing.

While we have tried to make the answers comprehensive and correct it is possible that there are mistakes and omissions; please tell us if you find anything!

Problem 1

Tasks a and b

We load the dataset, and then run the instructed commands.

Grading: It is enough to report that the instructed commands were run in tasks a-b.

Task c

i. We use `summary` to produce a numeric summary of the variables in the data set:

```
summary(npf[, 1:5])
```

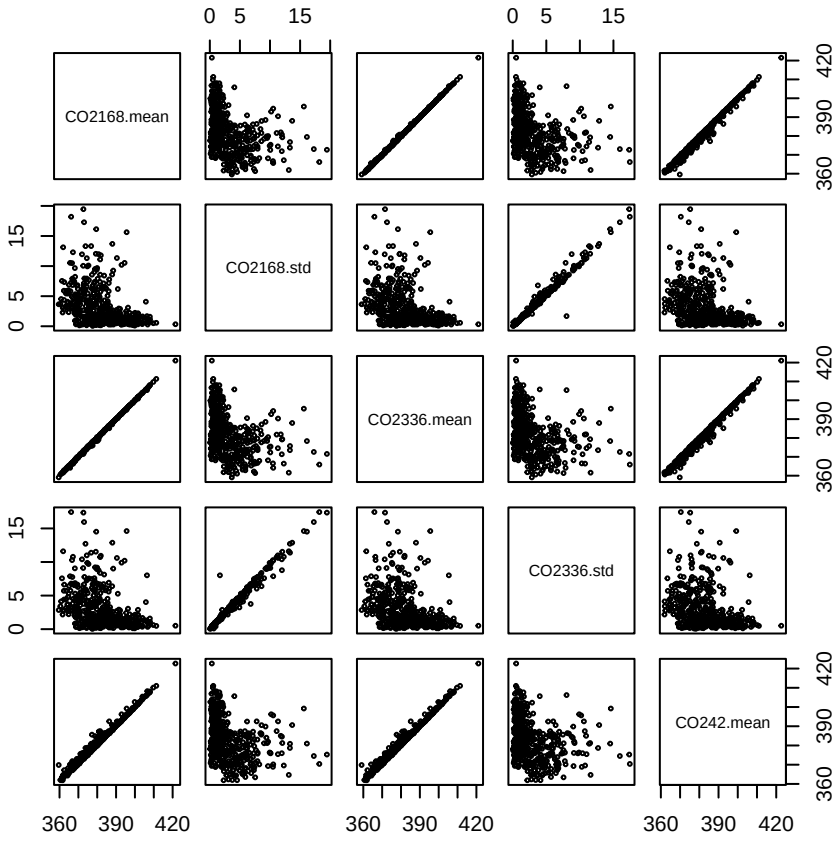
```
##     class4          partlybad        CO2168.mean      CO2168.std
##  Length:464         Mode :logical   Min.   :359.6   Min.   : 0.05397
##  Class :character   FALSE:464       1st Qu.:374.4   1st Qu.: 0.84564
##  Mode  :character                   Median :380.8   Median : 1.95273
##                                     Mean   :382.1   Mean   : 3.12997
##                                     3rd Qu.:389.0   3rd Qu.: 4.42806
##                                     Max.   :421.5   Max.   :19.46052
##   CO2336.mean
##  Min.   :359.1
##  1st Qu.:374.4
##  Median :380.7
##  Mean   :382.1
##  3rd Qu.:389.0
##  Max.   :421.1
```

Grading: It is sufficient to output few lines or no lines at all.

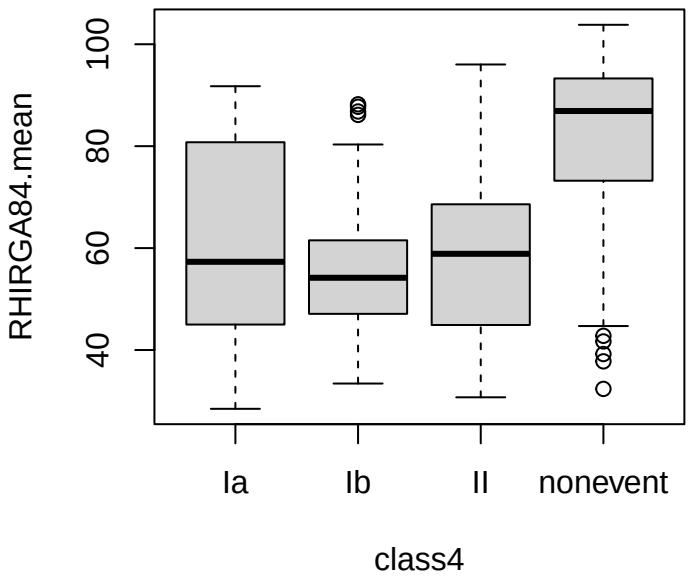
Column 2 (`partlybad`) is removed, as instructed:

```
npf <- npf[, -2]
```

ii. We next make a scatterplot matrix (or pairplot) of columns 2 to 6:

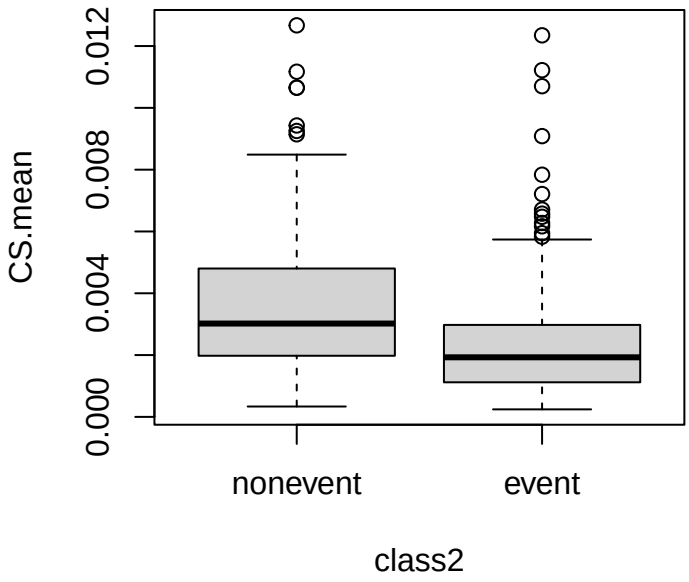


iii. Below we plot side-by-side boxplots of event vs nonevent days:

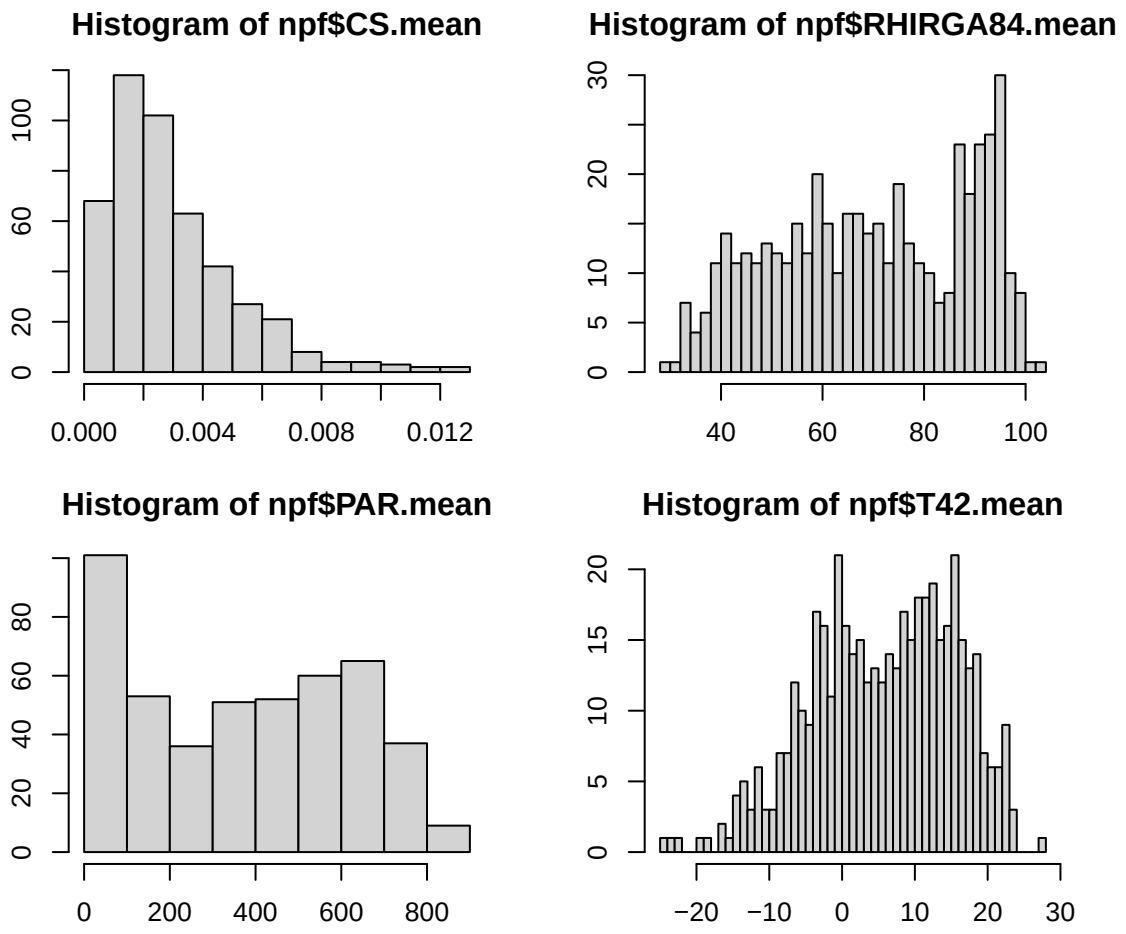


iv. We create a new class variable `class2`, as instructed, and then plot the requested boxplot:

```
npf$class2 <- factor("event", levels = c("nonevent", "event"))
npf$class2[npf$class4 == "nonevent"] <- "nonevent"
```



v. Below are histograms for some of the quantitative variables:

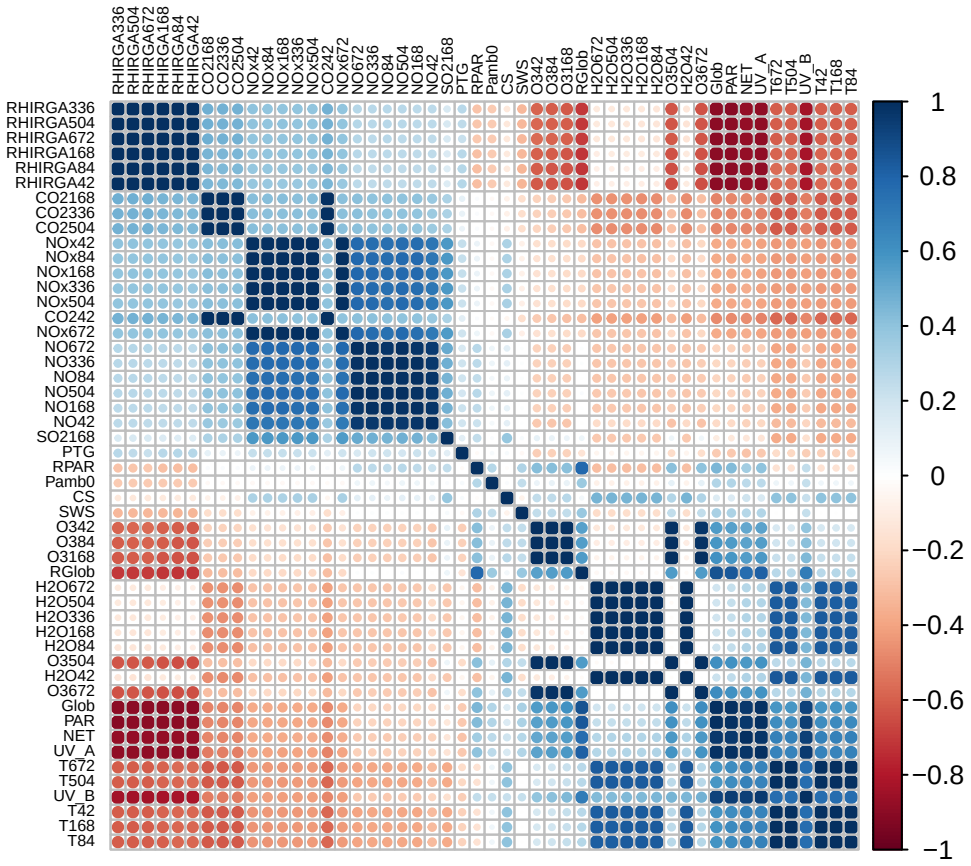


CS is condensation sink, RHIRGA84 is related to humidity, PAR is radiation, and T42 is temperature.

Grading: The student gets a point if there are histograms for multiple variables. The choice of (quantitative) variables is free.

vi. We continue exploring the data using a correlation plot of the mean values (variables ending with `.mean`):

```
library(corrplot)
# Calculate the correlation matrix
cm <- cor(npf[, endsWith(colnames(npf), ".mean")])
# Remove .mean from column and row names.
colnames(cm) <- rownames(cm) <- sapply(colnames(cm), function(s) gsub(".mean", "", s))
# Order variables by 1st principal component (PCA later in the course!)
corrplot(cm, order = "FPC", tl.cex = 0.5, tl.col = "black")
```



We notice that there are several mutually correlated variable blocks. For example, relative humidity measurements (`RHIRGA`) at different heights are quite correlated, as are other measurements of the same thing at different heights.

> **Grading:** The student gets full points from "explore the data and provide a brief summary" if at least one new aspect (not covered earlier) of the data is given (*not necessarily the correlation plot shown above*).

Grading

Points: max. 6 (a: 1, b: 1, c[i-ii]: 1, c[iii-iv]: 1, c[v]: 1, c[vi]: 1)

In this Problem, the student can use R or Python or something else, as long as the student provides an answer to the questions asked and the answer is understandable to others (e.g., if a boxplot was asked, a boxplot is produced). If a figure is requested, it should be shown, otherwise it is sufficient that the answer states that the requested task has been done (e.g., loading the dataset). The important thing is that the dataset is preprocessed as instructed (in a manner done in the programming environment) and the requested analysis is performed.

Problem 2

Tasks a

As an additional model we select LASSO regression. The results are shown in the following table:

```
##        train   test     cv  losocv
## dummy  3.114  3.089  3.109   3.113
## OLS    1.419  1.568  1.491   1.488
## SVR    3.106  3.088  3.109   3.114
## RF     0.545  1.482  1.440   1.468
## LASSO  1.458  1.633  1.516   1.548
```

Bonus observation: *The SVR results are surprisingly bad (comparable to the dummy model). This is because we haven't scaled the data. Some implementations of SVM:s automatically scale the data (such as the one in the* `e1071` *package for* `R`*). Lets try the SVR from* `sklearn` *again, but this time preprocessing the data to have zero mean and unit variance:*

```
##      train   test   cv  losocv
## SVR  1.312  1.726  1.6   1.622
```

As we can see, it is often a good idea to scale the variables before doing machine learning, since some models are sensitive to the scale of the variables. (This observation does not affect the grading)

Task b

RF overfits to the training data but it still has the smallest CV and test losses. The OLS linear regressor is suprisingly good on test data.

The train errors are smaller than the test errors (except for the dummy model, since it is so simple and the train and test sets are drawn from the same distribution). CV and test errors are roughly the same, as expected.

These regression models can be improved in several ways. For example, I could take only a subset of the features which might lead to a better loss on the training data (feature selection). The regression models also often include parameters that can be tuned, I could use the cross-validation loss to find variants of the models that would perform better.

Task c

The stations perform differently and it is possible that a model trained on some set of stations does not work well on a new station. In order to "simulate" this generalizability, the authors used leave-on-station-out cross-validation (LOSOCV). They wanted to create models that would work also on new stations and would not overfit to the peculiarities of one station.

Grading

Points: max. 6 (a:2, b:2, c:2)

Problem 3

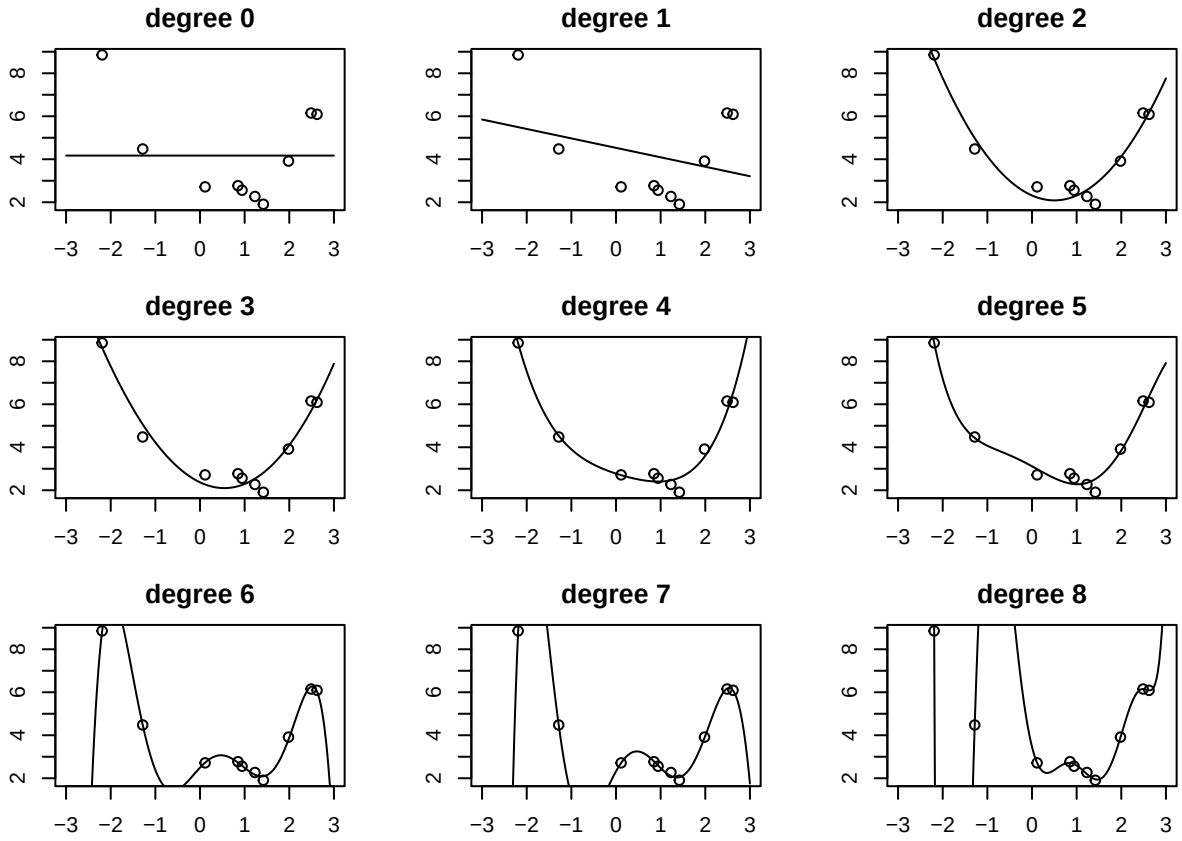
Task a

The function below generates the training, validation, and test dataset as instructed, first setting random seed so that we get the same results every time.

```
set.seed(42)

## the underlying function f(x)
f <- function(x) 2.0 - x + x^2

## make a dataset of n items
make_data <- function(n) {
    x <- runif(n, min = -3, max = 3)
    y <- f(x) + 0.4 * rnorm(n)
    data.frame(x = x, y = y)
}

data <- make_data(1020)
Str <- 1:10
Sva <- 11:20
Ste <- 21:1020
```

Task b

The plots for the polynomials are shown below.



The quantities required for the next tasks are shown in the following table. The rows correspond to polynomial degrees from 0 to 8, and the columns are named as follows:

- TRtr = training set loss for a model trained on the training set,
- VAtr = validation set loss for a model trained on training set,
- TEtr = test set loss for a model trained on training set,
- TEtrva = test set loss for a model trained on training+validation set, and
- CV = cross validation loss (on training+validation set).

Degree	TRtr	VAtr	TEtr	TEtrva	CV
0	4.512	6.659	11.716	11.263	6.364
1	4.089	7.128	8.876	9.935	9.634
2	0.219	0.294	0.246	0.214	0.352
3	0.217	0.283	0.290	0.275	0.561
4	0.119	0.625	0.969	0.224	2.028
5	0.097	0.573	4.895	1.039	5.812
6	0.008	3.417	213.297	0.881	2.979
7	0.005	6.863	1261.988	0.272	213.171
8	0.002	401.652	154266.871	11.224	7820.298

The losses on the training set are reported in column TRtr. We notice that these losses are always smaller for larger polynomial degrees.

Task c

The validation and test losses are reported in columns VAtr and TEtr, respectively. The test set losses are smallest for the degree 2 polynomial, which is consistent with the fact that data is created by a degree 2 polynomial plus noise. Due to the quite small validation set, the degree 3 polynomial sometimes has a smaller loss than the degree 2 polynomial.

Task d

The 5-fold CV losses are shown in column CV. The loss is smallest for the degree 2 polynomial, which is the degree we should choose if we would trust the CV result.

The moral of the story is that the losses behave roughly as expected, but because of finite sample size and variance in the estimator the relative ordering of the various losses varies a bit, at least if the losses are close enough.

Task e

The degree $\kappa = 2$ polynomial has the smallest loss on the validation set. We would expect that E[TRtr]<E[VAtr]<E[TEtr]. However, this expectation does not always hold, due to variance in the estimators (e.g., the validation set has only 10 items).

When we train a new regressor on training+validation set the error of the test set (column TEtrva) decreases for the regressor trained on the training set only (column TEtr), as expected.

As pointed out in Task d, the 2nd degree polynomial should be chosen according to the cross-validation. We would train the "final" model by using the combined training and validation set.

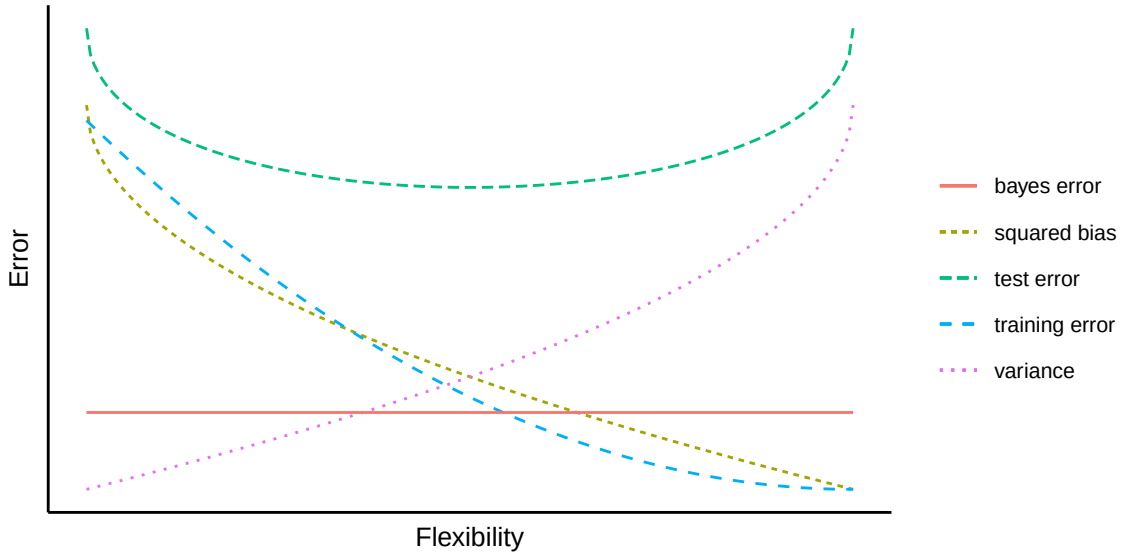
Grading

Points: max. 7 (a: 1, b: 1, c: 2, d: 2, e: 1)

The answer should get full points if the asked for numbers from the table have been produced, as well as figures requested, and there additionally an answer to direct questions asked.

Problem 4

Task a



The figure shows models with low flexibility on the left and with large flexibility on the right. The curve shapes are explained as follows:

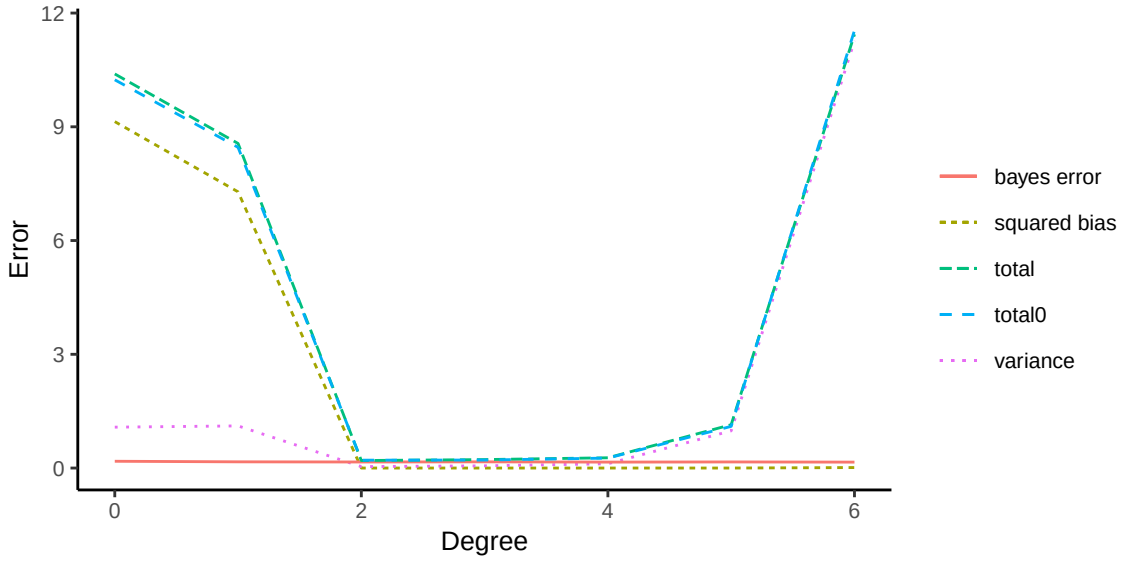
- The Bayes error is constant, because it does not depend on the model. It gives a lower bound for the generalization error of any model for a particular task.
- The squared bias is large for inflexible models and small for flexible models. The opposite is true for variance.
- The test error for regressors with MSE loss is the sum of the Bayes error, squared bias, and variance. It is large for small flexibility (under-fitting) and large flexibility (over-fitting), having the minimum in between.
- The training error is usually always smaller than the error in the test set. It tends to decrease as model flexibility grows, going close to zero for very flexible models.

Task b

(i) In the table below, the rows correspond to polynomial degrees and the columns to the squared Bayes error (`sigma2`), squared bias (`bias2`), variance (`variance`), and their sum (`total`) as well as the MSE (`total0`). All values were computed by sampling 1000 datasets, fitting a polynomial to them, and observing how the prediction $\hat{f}(0)$ and $x = 0$ changes compared to the test data point at $(0, y)$.

degree	sigma2	bias2	variance	total	total0
0	0.1785	9.1368	1.0787	10.3939	10.2391
1	0.1639	7.2847	1.1116	8.5603	8.4574
2	0.1601	0.0000	0.0378	0.1979	0.2073
3	0.1641	0.0000	0.0587	0.2228	0.2115
4	0.1570	0.0001	0.1134	0.2706	0.2622
5	0.1611	0.0004	0.9815	1.1431	1.0997
6	0.1565	0.0125	11.2698	11.4387	11.5403

We also plot the terms from the table:



(ii) We can verify Eq. (2.7) of James et al. by observing that the columns `total` and `total0` are roughly equal.

(iii) The bias-variance decomposition behaves as expected. Bayes error is roughly constant (it does not depend on the regression model used), squared bias decreases, and variance increases as flexibility (here polynomial degree) increases.
The small inconsistencies are due to to the fact that we estimated the components by sampling 1000 datasets. The inconsistencies should be smaller if we would sample more datasets.

Grading

Points: max. 7 (a: 3, b: 4)

Problem 5

Tasks a

Since all test data points are drawn from the same population, we have $E\left[\left(\overline{y}_1 - \hat{\beta}^T \overline{x}_1\right)^2\right] = \ldots = E\left[\left(\overline{y}_m - \hat{\beta}^T \overline{x}_m\right)^2\right]$. It follows that

$$E\left[L_{test}\right] = E\left[\frac{1}{m}\sum\nolimits_{i=1}^{m}\left(\overline{y}_i - \hat{\beta}^T \overline{x}_i\right)^2\right] = \frac{1}{m}\sum\nolimits_{i=1}^{m} E\left[\left(\overline{y}_i - \hat{\beta}^T \overline{x}_i\right)^2\right] = E\left[\left(\overline{y}_1 - \hat{\beta}^T \overline{x}_1\right)^2\right],$$

where we have used the linearity of the expectation.

This proves the claim of the Task a.

Task b

The generalisation error for OLS regression is defined as the expected squared loss for a new data point not used in training. Therefore, $L = E\left[\left(\overline{y}_1 - \hat{\beta}^T \overline{x}_1\right)^2\right] = E\left[L_{test}\right]$ is the generalisation error by definition. Since the expectation equals the estimated quantity the estimator is unbiased.

Notice that, as discussed in the problem statement, there are confusingly at least two different ways to define the expectation E and the generalisation error: (i) take the training data to be fixed and sample the test data, and (ii) sample over both training and test data. See Bengio et al.[1], Section 2.1, and Nadeau et al.[2] for discussion. Tasks a and b would work with either definition; in Task c we use the second definition.

Task c

It follows from Task a above that $E\left[L'_{test}\right] = E\left[L_{test}\right]$ as well, where

$$L'_{test} = \frac{1}{n}\sum\nolimits_{i=1}^{n}\left(\overline{y}'_i - \hat{\beta}^T \overline{x}'_i\right)^2,$$

and $(\overline{x}'_1, \overline{y}'_1), \ldots, (\overline{x}'_n, \overline{y}'_n)$ have been drawn randomly from the same population. In other words, the value of m does not really matter here.

We can use this observation prove our "theorem".

Theorem: $E\left[L_{train}\right] \leq E\left[L_{test}\right]$.

Proof: By Task a and the observation above, the claim is equivalent to $E\left[L_{train}\right] \leq E\left[L'_{test}\right]$. Recall that ordinary least squares (OLS) linear regression finds $\hat{\beta}$ such that the loss given by L_{train} is minimised, after which

$$L_{train} = \min_\beta\left(\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(y_i - \beta^T x_i\right)^2\right).$$

The following equation obviously holds for any $\hat{\beta}$ (random variable or not), because the expression inside the expectation is always non-negative:

$$E\left[\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(\overline{y}'_i - \hat{\beta}^T \overline{x}'_i\right)^2 - \min_\beta\left(\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(\overline{y}'_i - \beta^T \overline{x}'_i\right)^2\right)\right] \geq 0$$

By linearity of expectation we can rewrite the above as;

$$E\left[\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(\overline{y}'_i - \hat{\beta}^T \overline{x}'_i\right)^2\right] - E\left[\min_\beta\left(\frac{1}{n}\sum\nolimits_{i=1}^{n}\left(\overline{y}'_i - \beta^T \overline{x}'_i\right)^2\right)\right] \geq 0$$

[1] Bengio, Y., Grandvalet, Y., 2004. No unbiased estimator of the variance of K-fold cross-validation. J. Mach. Learn. Res. 5, 1089-1105. https://www.jmlr.org/papers/v5/grandvalet04a.html

[2] Nadeau, C., Bengio, Y., 2003. Inference for the Generalization Error. Mach. Learn. 52, 239-281. https://doi.org/10.1023/A:1024068626366

The latter term is $E[L_{train}]$, as explained earlier, and the first term is $E[L'_{test}] = E[L_{test}]$, or

$$E[L_{test}] - E[L_{train}] \geq 0,$$

which proves the theorem.

Task d

Theorem 2 shows that the expected loss on the training data cannot be larger than the expected loss on the test data. In machine learning, we usually want to estimate the *generalization loss* L.

Consider the case where we use the loss on training data L_{train} as an estimator of the generalisation loss L. The difference between the expectation of the estimator and the ground truth is called the *bias* of the estimator, which can here be written as bias $= E[L_{train}] - L = E[L_{train}] - E[L_{test}]$. For finite training data (n finite) and OLS linear regression, this bias is always negative, which means that we tend to underestimate the generalisation loss and that OLS linear regression tends to overfit to the training data.

Grading

You should give full or almost full points *if* the idea is correct, even though the proofs would be "shaky". However, please point out any ambiguities in your review.

Points: max. 6 (a: 2, b: 1, c: 2, d: 1)

Problem 6

Task a

The t-statistic is the estimator (here of the mean) $\hat{\mu}$ divided by its standard error. Recall that the standard error for the estimator of the mean $\hat{\mu} = \sum_{i=1}^{n} y_i/n$ is $\mathrm{sd}(\hat{\mu}) = \sigma/\sqrt{n}$, where σ is the standard deviation of y_i. We compute the t-statistic as follows:

```
co2 <- read.csv("co2lite.csv")
```

```
## 50 values of y
yy <- co2$FCO2
sdhatmu <- sd(yy) / sqrt(length(yy))
tstat <- mean(yy) / sdhatmu
tstat95 <- c(tstat - 1.96, tstat + 1.96)
y95 <- tstat95 * sdhatmu
```

The t-statistic is t =-2.328, and the 95% confidence interval is given by $t \pm 1.96 \times \mathrm{sd}(\hat{\mu})$ which corresponds to the interval [-2.71,-0.232]. Because zero is outside the interval, we can reject the null hypothesis that the mean is zero at the level $\alpha = 0.05$.

We can also do the "real t test" in which the above calculations are automated:

```
t.test(yy)
```

```
##
##  One Sample t-test
##
## data:  yy
## t = -2.3277, df = 49, p-value = 0.0241
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -2.7410832 -0.2010703
## sample estimates:
## mean of x
## -1.471077
```

We get the same value for the t-statistic. The 95% confidence interval again does not contain zero.

Note: *Notice that the 95% confidence interval given by* `t.test` *is slightly wider than what we obtained above. The reason is that a mean of normally distributed variables follows Student's t-distribution which has heavier tails than the normal distribution. The 95% confidence interval for the t-distribution with 19 degrees of freedom (here, the size of data set minus one) is given by* ± 2.093 *standard deviations, instead of* ± 1.96 *standard deviations, as for the normal distribution. The* ± 1.96 *standard deviations, however, gives the correct 95% confidence intervals for the Student's t-distribution at the limit of infinitely large data, since at the limit* $n \to \infty$ *Student's t-distribution becomes a normal distribution. The choice* ± 1.96 *vs.* ± 2.093 *has little practical significance here, because both numbers are "approximately 2" and the underlying assumptions, such as normality of the random variables being averaged, are satisfied only approximately in practice anyway.*

> **Grading:** In this problem it is sufficient to use "approximately 2" to compute the 95% confidence interval for the t-statistic.

Task b

We use R's built-in summary function to obtain the required values, shown below. We can reject the null hypothesis (at level $\alpha = 0.05$) that the slope is zero, either by observing the coefficient for `Net` has a p-value less than α, or equivalently, by observing that zero is not in the 95% confidence interval for `Net`, given by `Estimate` $\pm$ `2 * Std. Error`.

```r
m <- lm(FCO2 ~ Net, co2)
summary(m)
```

```
##
## Call:
## lm(formula = FCO2 ~ Net, data = co2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0230 -1.5375 -0.1224  1.4328  6.9674
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.114285   0.440184   0.260    0.796
## Net         -0.025743   0.002986  -8.623 2.57e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.828 on 48 degrees of freedom
## Multiple R-squared:  0.6077, Adjusted R-squared:  0.5995
## F-statistic: 74.35 on 1 and 48 DF,  p-value: 2.566e-11
```

Task c

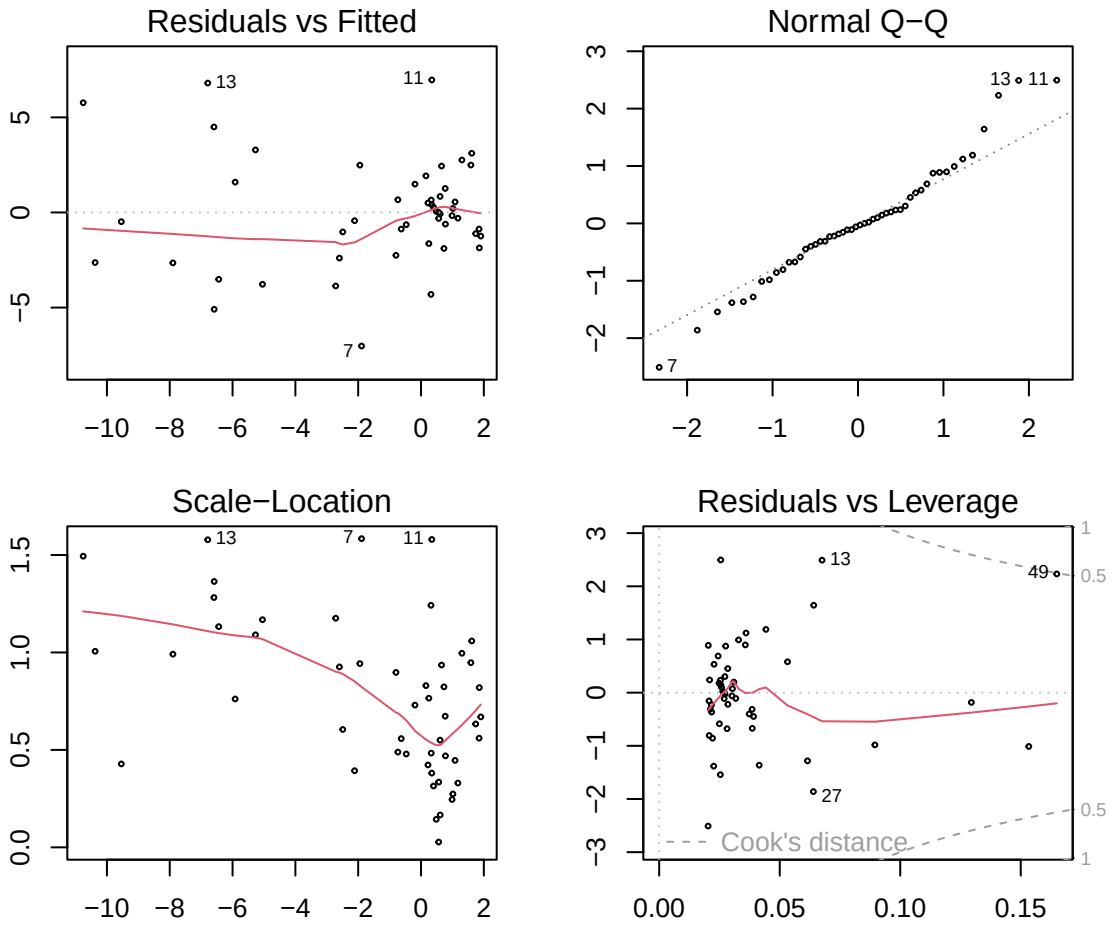
James et al. Sect. 3.3.3 lists 6 potential problems for linear regression:

1. Non-linearity of the response-predictor relationship
2. Correlation of error terms
3. Non-constant variance of error terms
4. Outliers
5. High-leverage points
6. Collinearity

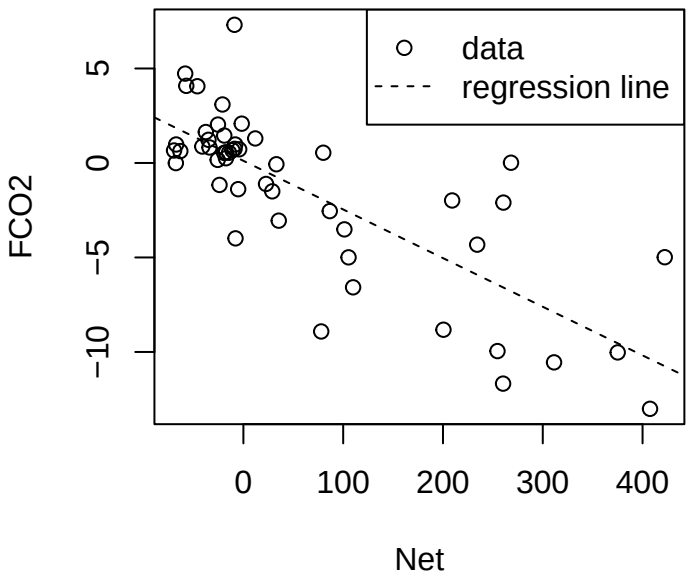
Since we don't know the true data-generating mechanism, it is difficult to know anything with certainty, but if we look at the diagnostic plots below, we can note the following:

- the data appears quite linear with respect to response
- the error terms seem relatively uncorrelated, as the residuals vs. fitted has little structure
- the error term variance appears to be roughly constant, with the exception of some outlier points (shown in the scale-location plot).
- collinearity of the predictor variables cannot be an issue, because we have only one predictor variable.

```r
par(mfrow = c(2, 2), mar = rep(2.3, 4))
plot(m, cex = 0.4)
```



The above can also be seen from the plot of the regression line, which is possible to do because we have only one covariate here:



Grading

Points: max. 6 (a: 2, b: 2, c: 2)

Problem 7

Give full points if the answer contains some sentences that are about the topic of the question.

Grading

Points: max. 2