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

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Task a

```
# Python
amol = pd.read_csv("./p1.csv")
amol = amol.drop(columns=["id","SMILES","InChIKey"],axis=1)
amol
```

```
##       pSat_Pa  ChemPot_kJmol  ...  carbonylperoxyacid  nitroester
## 0    0.000010       4.269730  ...                   0           0
## 1    0.000003       5.474680  ...                   0           0
## 2    0.000146      16.510064  ...                   1           0
## 3    0.000010      10.097080  ...                   1           0
## 4    0.014830      14.133175  ...                   0           0
## ..        ...            ...  ...                 ...         ...
## 395  0.051990      14.422792  ...                   1           0
## 396  0.000599       2.973652  ...                   0           0
## 397  0.000013      13.627330  ...                   0           0
## 398  0.000375      11.332322  ...                   1           0
## 399  0.000025       8.847988  ...                   0           0
##
## [400 rows x 29 columns]
```

```
# R
amol = read.csv("./p1.csv")
amol = amol[,-(1:3)]
#amol
# View(amol)
```

Task b

```
# Python
amol[["pSat_Pa","NumOfConf","ChemPot_kJmol"]].describe()
```

```
##             pSat_Pa    NumOfConf  ChemPot_kJmol
## count  4.000000e+02   400.000000     400.000000
## mean   2.961988e+00   223.497500      12.434427
## std    3.353178e+01   190.915391       4.778872
## min    2.458865e-10     2.000000      -3.160050
## 25%    4.926186e-06    73.250000       9.722915
## 50%    1.210795e-04   172.500000      12.780677
## 75%    2.287152e-03   324.250000      15.658871
## max    5.628970e+02  1058.000000      28.095686
```

```r
# R
summary(amol[,c("pSat_Pa","NumOfConf","ChemPot_kJmol")])
```

```
##     pSat_Pa           NumOfConf       ChemPot_kJmol
##  Min.   :  0.0000   Min.   :   2.00   Min.   :-3.160
##  1st Qu.:  0.0000   1st Qu.:  73.25   1st Qu.: 9.723
##  Median :  0.0001   Median : 172.50   Median :12.781
##  Mean   :  2.9620   Mean   : 223.50   Mean   :12.434
##  3rd Qu.:  0.0023   3rd Qu.: 324.25   3rd Qu.:15.659
##  Max.   :562.8970   Max.   :1058.00   Max.   :28.096
```

Task c

```python
# Python
print(amol['ChemPot_kJmol'].mean(), amol['ChemPot_kJmol'].std())
```

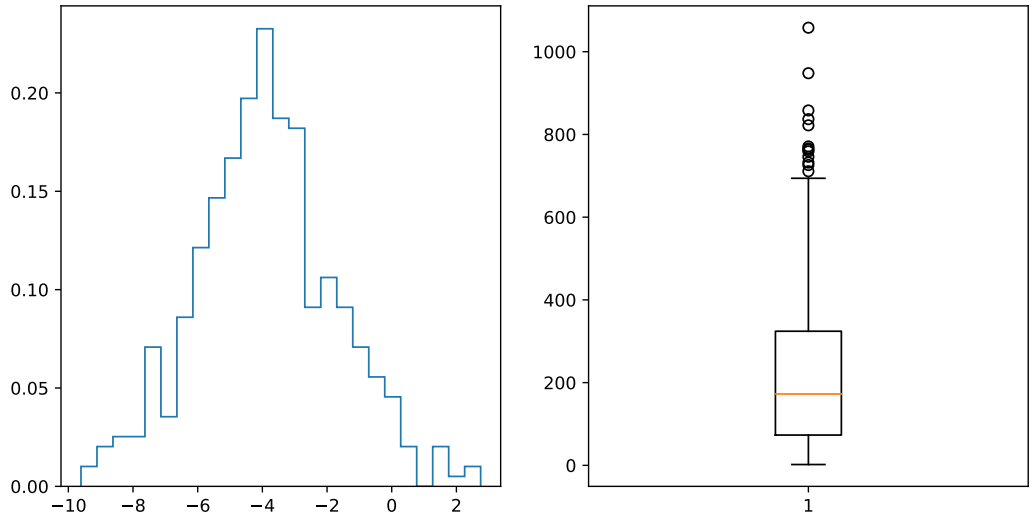
```
## 12.434427089600002 4.77887217784492
```

```r
# R
cat(mean(amol$ChemPot_kJmol), sd(amol$ChemPot_kJmol))
```

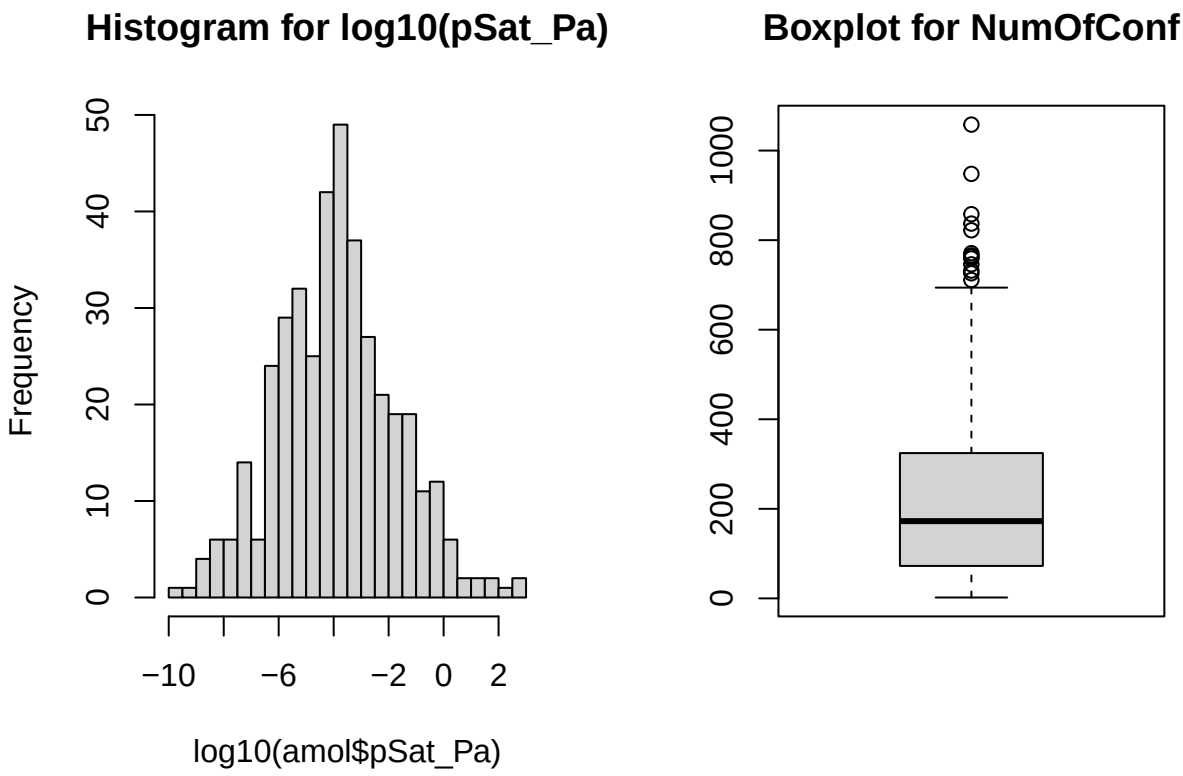
```
## 12.43443 4.778872
```

Task d

```python
# Python
fig, axes = plt.subplots(1,2,figsize=(10,5))
_ = axes[0].hist(np.log10(amol["pSat_Pa"].values), bins=25,histtype="step",density=True)
_ = axes[1].boxplot(amol["NumOfConf"].values, showcaps=True)
plt.show()
```



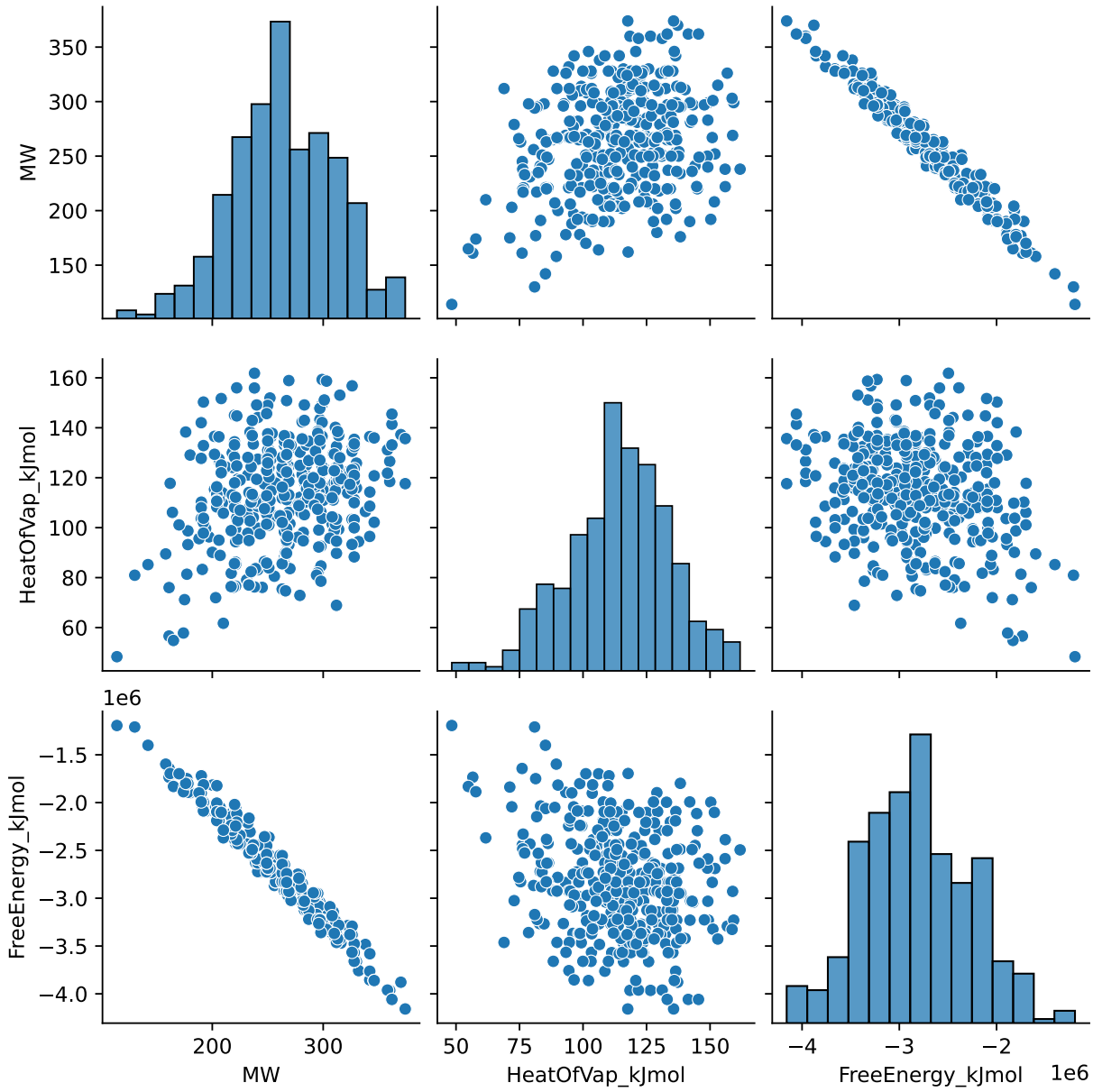

```r
# R
par(mfrow = c(1, 2))
hist(log10(amol$pSat_Pa), breaks = 20, main = "Histogram for log10(pSat_Pa)")
boxplot(amol$NumOfConf, main = "Boxplot for NumOfConf")
```



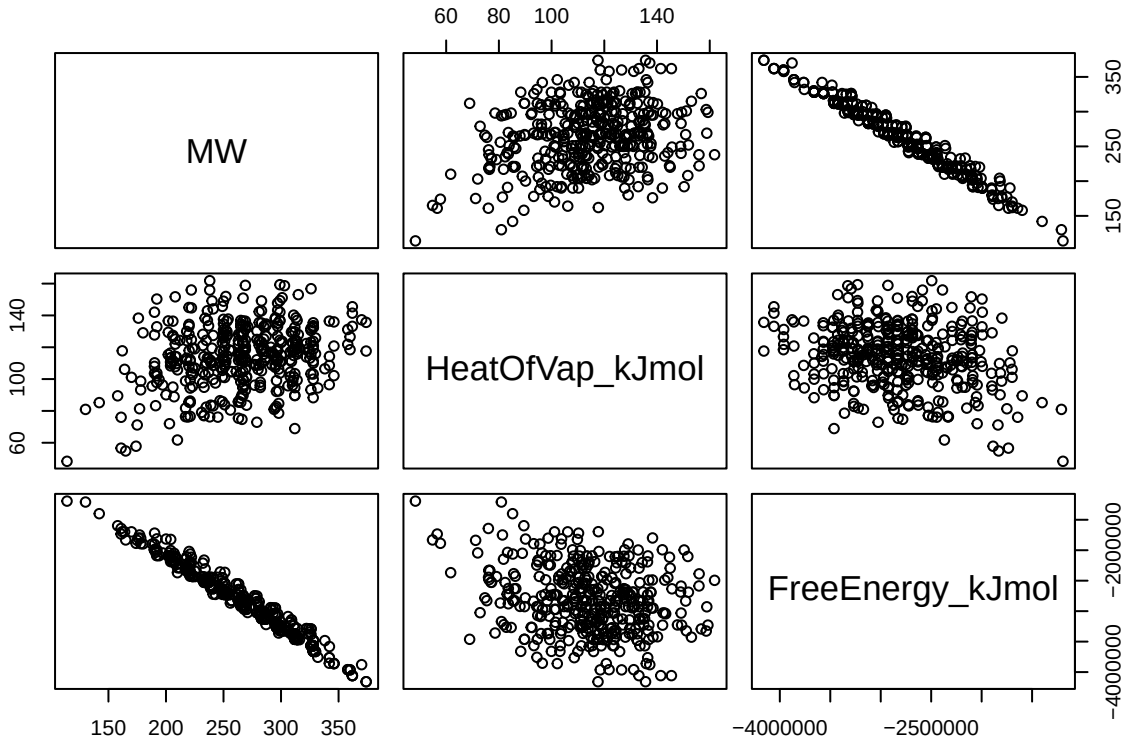
Task e

```python
# Python
sns.pairplot(amol[["MW","HeatOfVap_kJmol","FreeEnergy_kJmol"]])
```

```
## <seaborn.axisgrid.PairGrid object at 0x168cc6fd0>
```




```r
# R
pairs(amol[,c("MW","HeatOfVap_kJmol","FreeEnergy_kJmol")])
```



Grading

Points: max. 6 (Tasks a-c: 1 point each, Tasks d-e: 3 points total)

It is enough to report that the instructed commands were run in Tasks a-b. The important thing is that the data set is preprocessed as instructed (in a manner done in the programming environment) and the requested analysis is performed.

Problem 2

Task a

The requested table is shown below.

Degree	Train	Validation	Test	TestTRVA	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

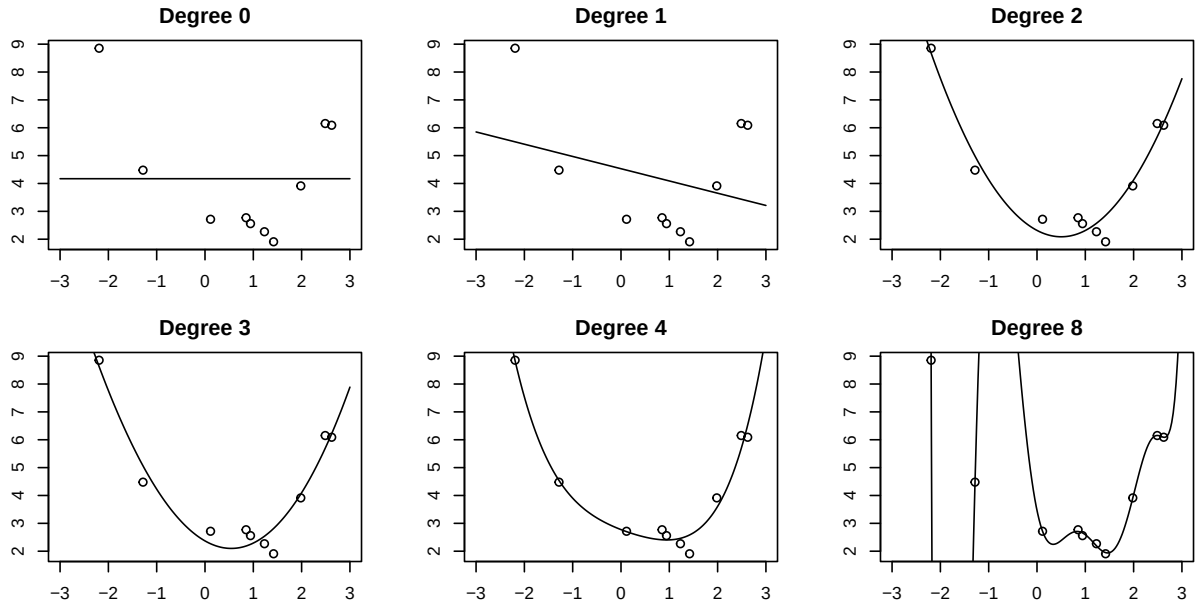
We choose the polynomial order according to the `CV` loss, which is minimized for the 2nd degree polynomial. We would train the "final" model by using the combined training and validation set.

Some things to notice:

- `Train` decreases for larger polynomial degrees.
- `Test` is smallest for the degree 2 polynomial, which is consistent with the fact that data is created by a degree 2 polynomial plus noise (which was not known to you).
- `TestTRVA` is lower than `Test`, as expected, since we train the regressor on more data.
- `Validation` is smallest for the degree 3 polynomial, due to the quite small validation set.
  - We would expect that E[Train] < E[Validation] < E[Test]. However, this expectation does not always hold, due to variance in the estimators (e.g., the validation set has only 10 items).
- `CV` is smallest for the degree 2 polynomial, which is the degree we should choose if we would trust the CV result.

Task b

The plots for the polynomials are shown below.



Task c

The requested table is shown below. As an additional model we selected a regression tree (`RT`).

	Train	Test	CV
Dummy	3.003	3.250	3.011
OLS	1.405	1.546	1.500
RF	0.580	1.490	1.428
SVM	0.997	1.615	1.604
RT	1.470	1.831	1.685

1. The best regressor is RF because it has the smallest CV and test losses (even if it overfits to the training data). The simple baseline OLS is also surprisingly good on test data.
2. `Train` errors are smaller than `Test` errors. `CV` and `Test` errors are roughly the same, as expected. An exception is the dummy model, since it is so simple and the train and test sets are drawn from the same distribution.
3. The regressors can be improved in several ways, such as:

- Feature selection. Try to use only a subset of the features which might lead to a better loss on the training data.
- Parameter tuning. The regression models 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 (Python)

```
##        train   test     cv
## dummy  3.003  3.250  3.002
## OLS    1.411  1.549  1.480
## SVR    3.001  3.302  3.011
## RF     0.529  1.538  1.417
## LASSO  1.491  1.620  1.535
```

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
## SVR  1.384  1.796  1.698
```

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)

Grading

Points: max. 8 (a: 3, b: 2, c: 3)

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.

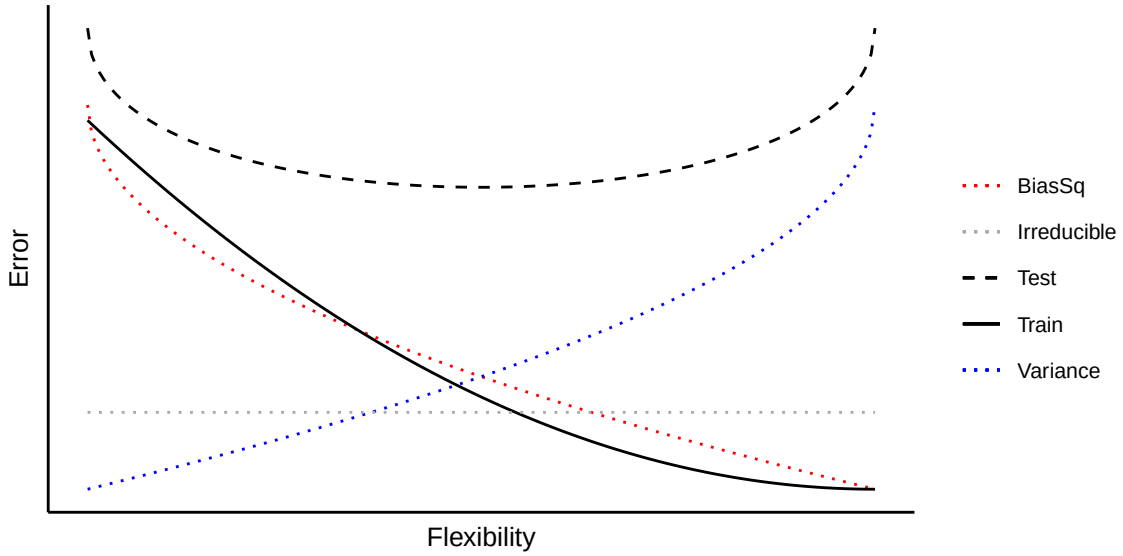
Problem 3

Task a

The curve shapes are explained as follows:

- The training error is usually smaller than the test error for any flexibility. It tends to decrease as model flexibility grows, going close to zero for very flexible models.
- The test error is large for small flexibility (under-fitting) and large flexibility (over-fitting), having the minimum in between. For regressors with MSE loss, the test error is the sum of the irreducible error, squared bias, and variance.
- The irreducible (or 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.

A sketch of the curves is shown in the figure below (low flexibility on the left and large flexibility on the right).

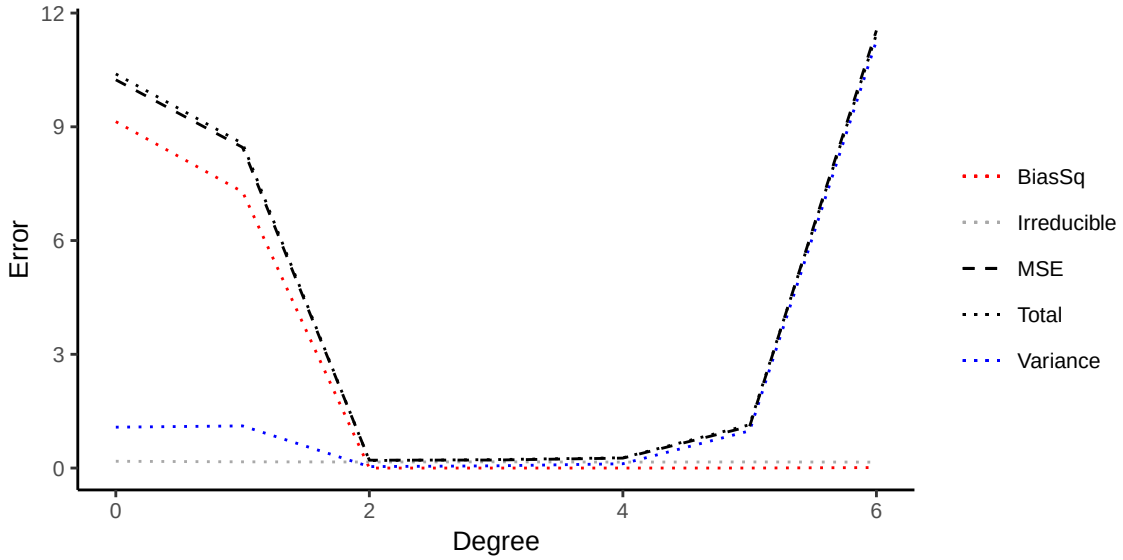


Task b

The requested table is shown below.

Degree	Irreducible	BiasSq	Variance	Total	MSE
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:



The bias-variance decomposition behaves as expected.

- The irreducible error is roughly constant (it does not depend on the regression model used),
- The squared bias decreases as flexibility (here polynomial degree) increases.
- The variance increases as flexibility (here polynomial degree) increases.

- We can verify Eq. (2.7) of James et al. by observing that the columns `Total` and `MSE` are roughly equal.

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. 6 (a: 2, b: 4)

Problem 4

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. However, for simplicity, in the following proof, the second definition is used.

Task 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.

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 to 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

Since $\min_\beta \left(\frac{1}{n} \sum_{i=1}^{n} \left(\overline{y}_i' - \beta^T \overline{x}_i' \right)^2 \right)$ and $\min_\beta \left(\frac{1}{n} \sum_{i=1}^{n} \left(y_i - \beta^T x_i \right)^2 \right)$ have the same law, their expectations are equal. So

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

And as $E\left[L_{test}'\right] = E\left[L_{test}\right]$, we derive

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

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 $= E\left[L_{train}\right] - L = E\left[L_{train}\right] - E\left[L_{test}\right]$. 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 5

The dataset in this problem is the famous Anscombe's quartet.

Task a

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 `x` has a p-value less than α, or equivalently, by observing that zero is not in the 95% confidence interval for `x`, given by `Estimate` $\pm$ `2 * Std. Error`. The small p-value gives some indication that an increase in x is associated with an increase or decrease in y, but it is not a "safe conclusion" because we are not sure if assumptions of linear regression hold for a given dataset.

```
## D1
## Call:
## lm(formula = y ~ x, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.921 -0.456 -0.041  0.709  1.839
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.00       1.12     2.7    0.026
## x               0.50       0.12     4.2    0.002
##
## Residual standard error: 1.2 on 9 degrees of freedom
## Multiple R-squared:  0.67,   Adjusted R-squared:  0.63
## F-statistic:  18 on 1 and 9 DF,  p-value: 0.0022
##
## D2
## Call:
## lm(formula = y ~ x, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -1.90  -0.76   0.13   0.95   1.27
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.00       1.13     2.7    0.026
## x               0.50       0.12     4.2    0.002
##
## Residual standard error: 1.2 on 9 degrees of freedom
## Multiple R-squared:  0.67,   Adjusted R-squared:  0.63
## F-statistic:  18 on 1 and 9 DF,  p-value: 0.0022
##
## D3
## Call:
## lm(formula = y ~ x, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -1.16  -0.61  -0.23   0.15   3.24
##
```

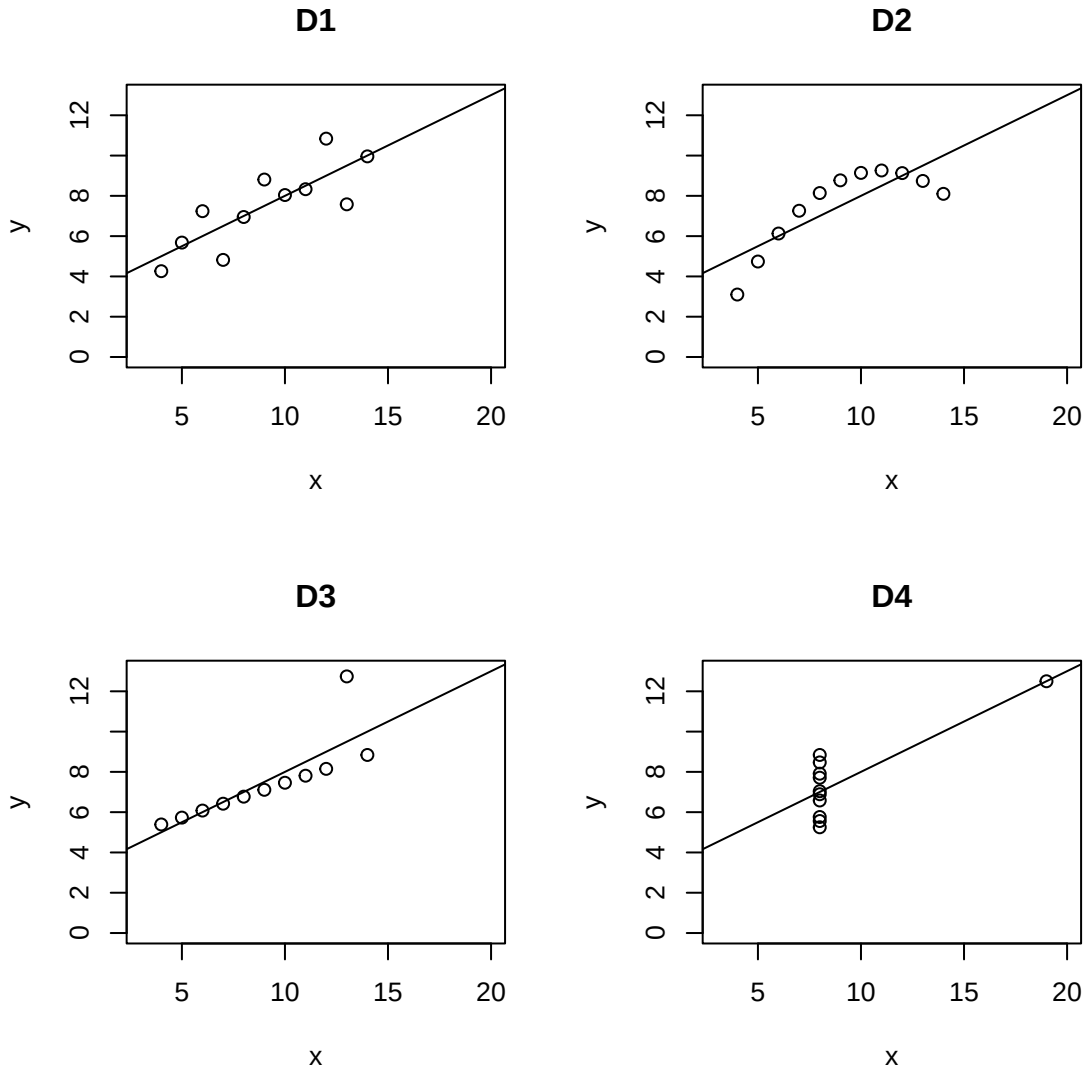
```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.00       1.12     2.7    0.026
## x               0.50       0.12     4.2    0.002
##
## Residual standard error: 1.2 on 9 degrees of freedom
## Multiple R-squared:  0.67,   Adjusted R-squared:  0.63
## F-statistic:  18 on 1 and 9 DF,  p-value: 0.0022
##
## D4
## Call:
## lm(formula = y ~ x, data = d)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -1.75  -0.83   0.00   0.81   1.84
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)     3.00       1.12     2.7    0.026
## x               0.50       0.12     4.2    0.002
##
## Residual standard error: 1.2 on 9 degrees of freedom
## Multiple R-squared:  0.67,   Adjusted R-squared:  0.63
## F-statistic:  18 on 1 and 9 DF,  p-value: 0.0022
```

Task b

The data and their regression lines are shown below.

We notice that the regression lines are the same, although the data sets look very different. The data sets also have the same average x, average y, standard deviation of x, standard deviation of y, correlation between x and y, and R squared value.

The conclusion here is that you should always plot the data and the model. Don't blindly trust loss values or p-values (or even averages).



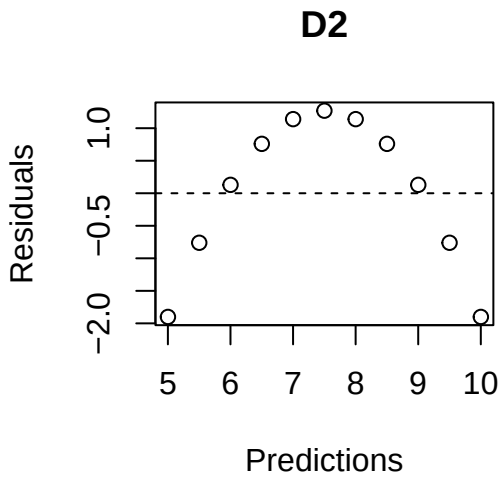
Task c

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

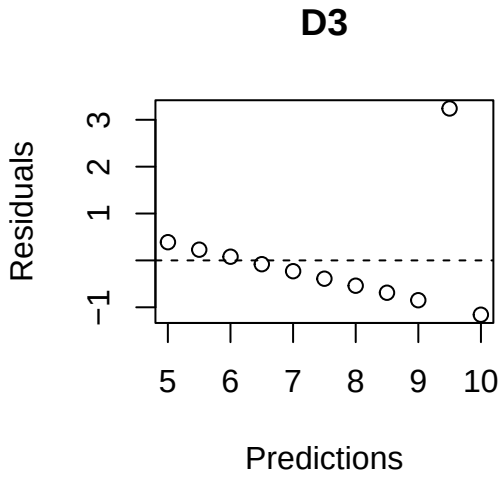
1. Non-linearity of the response-predictor relationship
2. Correlation of error terms
3. Non-constant variance of error terms
4. Outliers
5. High-leverage points
6. Collinearity

Out of these problems, the following apply to each data set:

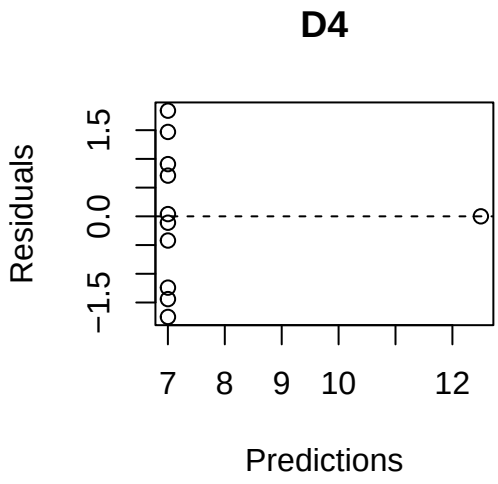
- D2: `y` is a *non-linear* function of `x`, as we can see from the scatterplot above, or from a residual plot:



- D3: There is an *outlier* point evident from the scatterplot above, or from the residual plot below. Robust regression would ignore the outlier.



- D4: There is a *high-leverage* point (outlier in x) evident from the scatterplot above, or from the residual plot below. If we delete the point, the estimates change a lot.



Grading

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

Problem 6

Task a

We follow here ISLR_v2 lab section Section 5.3.4. First, we make a function that fits a linear regression model to the data for a given subset of data points and outputs the regression coefficients.

```
## Use data from the previous problem
## From Sect. 5.3 of ISLR_v2:
boot.fn <- function(d, index)
  coef(lm(y ~ x, data=d, subset=index))
boot.fn(data[[2]], 1:nrow(data[[2]]))
```

```
## (Intercept)           x
##    3.000909    0.500000
```

Bootstrap samples new data set by replacement and fits a linear model. R function `boot` does this automatically, even though implementing it all by yourself would not be too complicated.

```
library(boot)
set.seed(42)
boot(data[[2]], boot.fn, 1000)
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data[[2]], statistic = boot.fn, R = 1000)
##
##
## Bootstrap Statistics :
##     original      bias    std. error
## t1* 3.000909 0.107966624   1.4875855
## t2* 0.500000 0.003024308   0.1602044
```

For dataset 2 the standard deviation for intercept (1.48 vs. 1.12) and slope (0.16 vs 0.12) are larger than in the previous problem. The difference is due to the fact that the t-test used in the previous problem involves several assumptions (normality, homoscedasticity, etc.) which may be violated in dataset 2. Bootstrap makes fewer distributional assumptions and, hence, in this case probably the bootstrap estimate is more trustworthy and robust. Notice that you should be careful in interpreting the regression coefficients for dataset 2 because our modelling assumptions seem to be wrong: the dataset looks like a curved line and it is doesn't seem that the underlying model is a straight line plus uncorrelated homoscedastic noise (our modelling assumption).

Taks b

The algorithm creates many bootstrap datasets from dataset 2. To create a bootstrap dataset, it draws n (x, y)-samples with replacement from dataset 2. For each bootstrap dataset, it fits a linear model and obtains the corresponding values for intercept and slope. The bootstrap standard deviations for the intercept and slope are the standard deviations of the intercept and slope, respectively, of the lines fitted to the bootstrapped datasets.

Task c

A bootstrap sample consists of n items drawn out of $\{1, \dots, n\}$ with replacement.

Let's look at one item $j \in \{1, \dots, n\}$. The item j is not in a draw with a probability of $1 - 1/n$. Since draws are independent the item j is not in any of the n draws with a probability $(1-1/n)^n$. At the limit $n \to \infty$ the

item j is therefore missing from the bootstrap sample with a probability of $\lim_{n\to\infty}(1-1/n)^n = 1/e \approx 0.368$. Or equivalently, the item j is in the bootstrap sample with a probability of $1 - 1/e \approx 0.632$.

Therefore roughly a third of the items is missing from a bootstrap sample!

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