week1_ML

11/8/2020

- Answer anonymously, i.e., do not write your name to the answer sheet.
- Submit the answer via Moodle at latest on 8 November 2020 (Moodle submission will open during the week starting on 2 November).
- Your answer will be peer-reviewed by other students and you will review your answer and answers of 2 random peers during the week starting on 9 November.
- The assignment should be completed by one person, but discussions with others are encouraged. Your final solution must be your own.
- The language of the assignments is English.
- The submitted report should be in a single Portable Document Format (pdf) file.
- Answer to the problems in the correct order.
- Read the general instructions in Moodle before starting with the problems.
- Main source material: James et al., Chapters 1-3 and 5. Please feel to use other resources as well.
- Version history: v2: Task 4c clarified.

Problem 1

[5% points]

Objective: familiarity with command line tools

Unix command line tools are quite useful. Download the csv-file of Problem 2 below and do the following operations with the unix command line tools.

Tasks

a. Show the first and last lines by using `head` and `tail`.

First 10 Lines :

C:-Helsinki1> head npf_train.csv
"id","date","class4","partlybad","CO2168.mean","CO2168.std","CO2336.mean","CO2336.std","CO242.mean","CO242.std","CO2504.mean","CO2504.std","Glob.mear
1,"2000-02-
23","nonevent",FALSE,380.52811965812,0.802000657433635,380.371465517241,0.889550208090346,381.816206896552,1.29259271313084,380.29646551724
2,"2000-03-
25","Ib",FALSE,373.128684210526,1.0966171246884,372.98,1.04775009575653,373.701830065359,1.25919796837658,372.91,1.00416370022131,252.4803272
3,"2000-04-
06","Ib",FALSE,372.363293413174,0.626329374753749,372.245688622754,0.615803402208043,372.847245508982,0.647279445314092,372.193952095808,0.5
05 4,"2000-04-
11","nonevent",FALSE,381.437441860465,7.28115876422255,381.380404624277,7.23600159075259,381.926531791908,7.29437355610594,381.381156069364,
5,"2000-04-
23","II",FALSE,375.426310160428,3.26424632247952,375.436524064171,3.1108858820587,375.740215053764,3.27492421319194,375.337058823529,2.90378(
6,"2000-05-
29","nonevent",FALSE,375.497688888889,2.98129376145046,375.151244444444,2.16954534773375,377.234646017699,3.41240783135203,374.657466666666
7,"2000-06-
10","Ib",FALSE,369.19321888412,4.19445060612572,369.118931623932,3.72353392119065,369.439615384615,3.92151316670903,368.485384615385,2.55189
8,"2000-06-
15","Ib",FALSE,368.569446808511,1.90732598884253,368.706144067797,1.76957260268618,368.879787234043,2.19934944670586,368.707744680851,1.6567
9,"2000-06-
22","nonevent",FALSE,367.847076271186,7.41731002528405,367.422627118644,6.53578768997684,368.767627118644,7.85090700036963,366.909322033898,

Last 10 Lines:

C:-Helsinki1>tail npf_train.csv 421,"2009-01-
29","nonevent",FALSE,404.45231884058,1.76740516421467,404.423043478261,1.72719707001546,404.288695652174,1.86866963769248,404.288088235294,
422,"2008-07-
24","Ib",FALSE,375.610057142857,7.67355179321491,375.751771428571,7.55785233392911,377.547167630058,9.76082330452729,375.730914285714,7.4198
423,"2006-07-
01","Ib",FALSE,366.630307692308,3.18125370924253,366.826923076923,2.73498652802627,370.087743589744,7.51327567557679,366.44912371134,2.23267
05,0.0171863139859807,49.9076817792986,28.7821085386894,44.8470769230769,15.0105024459416,44.4532820512821,14.4965814490416,46.95367875647
424,"2004-01-
01","nonevent",FALSE,381.605185185185,0.540257593140693,381.267407407407,0.667007355296465,383.688148148148,1.44167365275019,380.6040740740
425,"2008-05-
06","Ib",FALSE,386.231158536585,1.53426523626221,386.530548780488,1.48592135514616,385.999506172839,1.53612070989395,386.634969325153,1.4524
426,"2004-01-
31","nonevent",FALSE,389.310533333333,1.38488061084928,389.320933333333,1.37243930928024,389.411333333333,1.37215684759957,389.308133333333
427,"2008-08-
20","nonevent",FALSE,380.048926174497,6.12786558900778,380.105533333333,5.6581274366601,381.673758389262,6.83540133823492,379.8884,5.4853047
428,"2009-04-
22","Ib",FALSE,392.901307692308,2.236589453881,392.738372093023,2.83058154103803,393.167054263566,2.19658840701776,392.839689922481,2.071451
429,"2006-12-
12","nonevent",FALSE,387.960192307692,0.40942566365253,388.0075,0.417033759859759,388.179411764706,0.463350458140268,387.972307692308,0.3221
05 430,"2009-11-
19","nonevent",FALSE,401.640476190476,2.05221157540603,400.852096774194,2.88418249183689,402.276031746032,2.13837181027435,399.988095238095

b. Count the rows by using `wc` .

C:-Helsinki1>wc npf_train.csv 431 431 755089 npf_train.csv

Therefore we have 431 rows in our file (430 without the header)

c. Change the file from csv (comma-separated values) format to tsv (tab-separated values) format by using `sed` or `tr` .

cat npf_train.csv | sed 's/,/g' > npf_train.tsv

If we apply head now we will get the variables separated with a tab (only copied the first line) :

C:-Helsinki1> head npf_train.tsv ==> npf_train.csv <== "id" "date" "class4" "partlybad" "CO2168.mean" "CO2168.std" "CO2336.mean" "CO2336.std" "CO242.mean" "CO242.std" "CO2504.mean" "CO2504.std" "Glob.mean" "Glob.std" "H2O168.mean" "H2O168.std" "H2O336.mean" "H2O336.std" "H2O42.mean" "H2O42.std" "H2O504.mean" "H2O504.std" "H2O672.mean" "H2O672.std" "H2O84.mean" "H2O84.std" "NET.mean" "NET.std" "NO168.mean" "NO168.std" "NO336.mean" "NO336.std" "NO42.mean" "NO42.std" "NO504.mean" "NO504.std" "NO672.mean" "NO672.std" "NO84.mean" "NO84.std" "NOx168.mean" "NOx168.std" "NOx336.mean" "NOx336.std" "NOx42.mean" "NOx42.std" "NOx504.mean" "NOx504.std" "NOx672.mean" "NOx672.std" "NOx84.mean" "NOx84.std" "O3168.mean" "O3168.std" "O342.mean" "O342.std" "O3504.mean" "O3504.std" "O3672.mean" "O3672.std" "O384.mean" "O384.std" "Pamb0.mean" "Pamb0.std" "PAR.mean" "PAR.std" "PTG.mean" "PTG.std" "RGlob.mean" "RGlob.std" "RHIRGA168.mean" "RHIRGA168.std" "RHIRGA336.mean" "RHIRGA336.std" "RHIRGA42.mean" "RHIRGA42.std" "RHIRGA504.mean" "RHIRGA504.std" "RHIRGA672.mean" "RHIRGA672.std

d. Separate the class variable (column "class4") by using `awk` and list the unique classes.

C:-Helsinki1>awk -F "","" '{print $3}' npf_train.csv | sort | uniq II Ia Ib class4 nonevent

Hint: Useful commands: `head` , `tail` , `sort` , `uniq` . You can find the instructions in any unix system by man command (e.g., `man head`), or by using Google.

Problem 2

[10% points]

Objective: familiarity with tools, basic description of the data set

This exercise relates to a data set about new particle formation (NPF) of which you will do your term project. Read the data description and download the dataset file `npf_train.csv` from the Moodle term project page.

The instructions below are in R, but you can do equivalent tasks with Python as well. Before reading the data into R, it can be viewed in Excel or a text editor.

Task a

Use the read.csv() function to read the data into R. Call the loaded data npf. Make sure that you have the directory set to the correct location for the data.

```
npf <- read.csv("npf_train.csv")
```

Task b

Look at the data using the `fix` function. In RStudio, instead of `fix` , you can use the nicer Data Viewer (https://support.rstudio.com/hc/en-us/articles/205175388-Using-the-Data-Viewer) `View` .

```
fix(npf)
```

You should notice that the second column is the date and first column is the id. We don't really want R to treat this as data. However, it may be handy to have the dates. Try the following commands:

```
rownames(npf) <- npf[,"date"]
fix(npf)
```

You should see that there is now a row.names column with the name of each day recorded. This means that R has given each row a name corresponding to the appropriate id. R will not try to perform calculations on the row names. However, we still need to eliminate the first column in the data where the id is stored. Try

```
npf <- npf[,-1]
fix(npf)
```

Now you should see that the first data column is date. Note that another column labeled row.names now appears before the data column. However, this is not a data column but rather the name that R is giving to each row.

Task c

i. Use the summary() function to produce a numerical summary of the variables in the data set. We notice that the variable "partlybad" is always false and thereofore useless. We opt to remove it as well.

```
summary(npf)
```

```
##      date              class4           partlybad        CO2168.mean   
##  Length:430         Length:430         Mode :logical   Min.   :360.5  
##  Class :character   Class :character   FALSE:430       1st Qu.:373.2  
##  Mode  :character   Mode  :character                   Median :380.5  
##                                                        Mean   :381.4  
##                                                        3rd Qu.:388.2  
##                                                        Max.   :421.5  
##    CO2168.std       CO2336.mean      CO2336.std       CO242.mean   
##  Min.   : 0.1645   Min.   :360.4   Min.   : 0.1492   Min.   :361.8  
##  1st Qu.: 0.8874   1st Qu.:373.3   1st Qu.: 0.8955   1st Qu.:374.5  
##  Median : 2.2818   Median :380.5   Median : 2.1879   Median :381.4  
##  Mean   : 3.2596   Mean   :381.4   Mean   : 3.0728   Mean   :382.4  
##  3rd Qu.: 4.6052   3rd Qu.:388.2   3rd Qu.: 4.2729   3rd Qu.:388.7  
##  Max.   :17.2848   Max.   :421.1   Max.   :15.9555   Max.   :422.6  
##    CO242.std        CO2504.mean      CO2504.std        Glob.mean     
##  Min.   : 0.1527   Min.   :360.0   Min.   : 0.1342   Min.   :  3.719  
##  1st Qu.: 1.1535   1st Qu.:373.3   1st Qu.: 0.8906   1st Qu.: 74.046  
##  Median : 2.6715   Median :380.5   Median : 2.0735   Median :200.062  
##  Mean   : 4.1952   Mean   :381.3   Mean   : 2.8687   Mean   :196.704  
##  3rd Qu.: 6.1548   3rd Qu.:388.2   3rd Qu.: 4.0826   3rd Qu.:313.597  
##  Max.   :40.3667   Max.   :419.9   Max.   :14.3424   Max.   :425.991  
##     Glob.std        H2O168.mean       H2O168.std       H2O336.mean    
##  Min.   :  1.998   Min.   : 0.8246   Min.   :0.01367   Min.   : 0.8285  
##  1st Qu.: 46.537   1st Qu.: 4.0293   1st Qu.:0.19046   1st Qu.: 4.0059  
##  Median :154.675   Median : 6.3176   Median :0.46029   Median : 6.2786  
##  Mean   :145.639   Mean   : 7.0840   Mean   :0.54496   Mean   : 7.0093  
##  3rd Qu.:233.168   3rd Qu.: 9.7301   3rd Qu.:0.79322   3rd Qu.: 9.5896  
##  Max.   :320.099   Max.   :18.6295   Max.   :3.05996   Max.   :18.5017  
##    H2O336.std        H2O42.mean        H2O42.std        H2O504.mean    
##  Min.   :0.01682   Min.   : 0.8006   Min.   :0.01689   Min.   : 0.8356  
##  1st Qu.:0.18669   1st Qu.: 4.0823   1st Qu.:0.19840   1st Qu.: 3.9892  
##  Median :0.45464   Median : 6.4746   Median :0.49102   Median : 6.3081  
##  Mean   :0.54215   Mean   : 7.2170   Mean   :0.55808   Mean   : 6.9676  
##  3rd Qu.:0.77506   3rd Qu.: 9.9973   3rd Qu.:0.79199   3rd Qu.: 9.4671  
##  Max.   :3.07436   Max.   :18.8547   Max.   :3.08742   Max.   :18.4412  
##    H2O504.std       H2O672.mean      H2O672.std        H2O84.mean     
##  Min.   :0.01713   Min.   : 0.864   Min.   :0.01851   Min.   : 0.8042  
##  1st Qu.:0.18734   1st Qu.: 3.948   1st Qu.:0.19262   1st Qu.: 4.0454  
##  Median :0.44834   Median : 6.250   Median :0.44970   Median : 6.3817  
##  Mean   :0.54082   Mean   : 6.937   Mean   :0.54252   Mean   : 7.1572  
##  3rd Qu.:0.78485   3rd Qu.: 9.408   3rd Qu.:0.78014   3rd Qu.: 9.8693  
##  Max.   :3.04878   Max.   :18.403   Max.   :3.06305   Max.   :18.7498  
##    H2O84.std          NET.mean         NET.std         NO168.mean      
##  Min.   :0.01701   Min.   :-59.71   Min.   :  1.74   Min.   :-0.01484  
##  1st Qu.:0.19049   1st Qu.: 44.31   1st Qu.: 43.91   1st Qu.: 0.01624  
##  Median :0.47055   Median :127.43   Median :133.59   Median : 0.03597  
##  Mean   :0.55186   Mean   :124.55   Mean   :128.35   Mean   : 0.09691  
##  3rd Qu.:0.79030   3rd Qu.:203.10   3rd Qu.:205.56   3rd Qu.: 0.08781  
##  Max.   :3.09960   Max.   :302.83   Max.   :271.79   Max.   : 5.11143  
##    NO168.std         NO336.mean          NO336.std         NO42.mean       
##  Min.   :0.02096   Min.   :-0.008052   Min.   :0.02442   Min.   :-0.01227  
##  1st Qu.:0.05077   1st Qu.: 0.018905   1st Qu.:0.05024   1st Qu.: 0.01493  
##  Median :0.06351   Median : 0.039923   Median :0.06429   Median : 0.03493  
##  Mean   :0.10245   Mean   : 0.101441   Mean   :0.10035   Mean   : 0.08250  
##  3rd Qu.:0.09631   3rd Qu.: 0.092411   3rd Qu.:0.09959   3rd Qu.: 0.07119  
##  Max.   :1.04660   Max.   : 5.200000   Max.   :1.11947   Max.   : 5.03329  
##     NO42.std         NO504.mean          NO504.std         NO672.mean      
##  Min.   :0.02213   Min.   :-0.02333   Min.   :0.02535   Min.   :-0.01231  
##  1st Qu.:0.04950   1st Qu.: 0.01823   1st Qu.:0.05067   1st Qu.: 0.01833  
##  Median :0.06329   Median : 0.03841   Median :0.06367   Median : 0.03825  
##  Mean   :0.11030   Mean   : 0.09901   Mean   :0.09740   Mean   : 0.09667  
##  3rd Qu.:0.09269   3rd Qu.: 0.09218   3rd Qu.:0.09614   3rd Qu.: 0.09055  
##  Max.   :1.79315   Max.   : 5.21594   Max.   :1.16840   Max.   : 5.18457  
##    NO672.std         NO84.mean           NO84.std         NOx168.mean     
##  Min.   :0.02537   Min.   :-0.02126   Min.   :0.02220   Min.   : 0.04289  
##  1st Qu.:0.05118   1st Qu.: 0.01344   1st Qu.:0.04876   1st Qu.: 0.49184  
##  Median :0.06319   Median : 0.03063   Median :0.06104   Median : 1.01360  
##  Mean   :0.09565   Mean   : 0.08419   Mean   :0.09821   Mean   : 1.51065  
##  3rd Qu.:0.09501   3rd Qu.: 0.07383   3rd Qu.:0.09038   3rd Qu.: 1.93862  
##  Max.   :1.20562   Max.   : 5.05143   Max.   :1.37139   Max.   :16.10108  
##    NOx168.std       NOx336.mean         NOx336.std       NOx42.mean      
##  Min.   :0.03606   Min.   : 0.02023   Min.   :0.0578   Min.   : 0.07533  
##  1st Qu.:0.20397   1st Qu.: 0.49398   1st Qu.:0.1947   1st Qu.: 0.51880  
##  Median :0.36246   Median : 1.02860   Median :0.3463   Median : 1.02928  
##  Mean   :0.51957   Mean   : 1.49787   Mean   :0.5084   Mean   : 1.52797  
##  3rd Qu.:0.64662   3rd Qu.: 1.91334   3rd Qu.:0.6196   3rd Qu.: 1.96479  
##  Max.   :5.14500   Max.   :15.99608   Max.   :5.5703   Max.   :16.05554  
##    NOx42.std          NOx504.mean         NOx504.std        NOx672.mean     
##  Min.   : 0.06466   Min.   : 0.05432   Min.   :0.05679   Min.   : 0.03304  
##  1st Qu.: 0.22179   1st Qu.: 0.48199   1st Qu.:0.19619   1st Qu.: 0.48134  
##  Median : 0.37851   Median : 1.01287   Median :0.34013   Median : 0.98863  
```

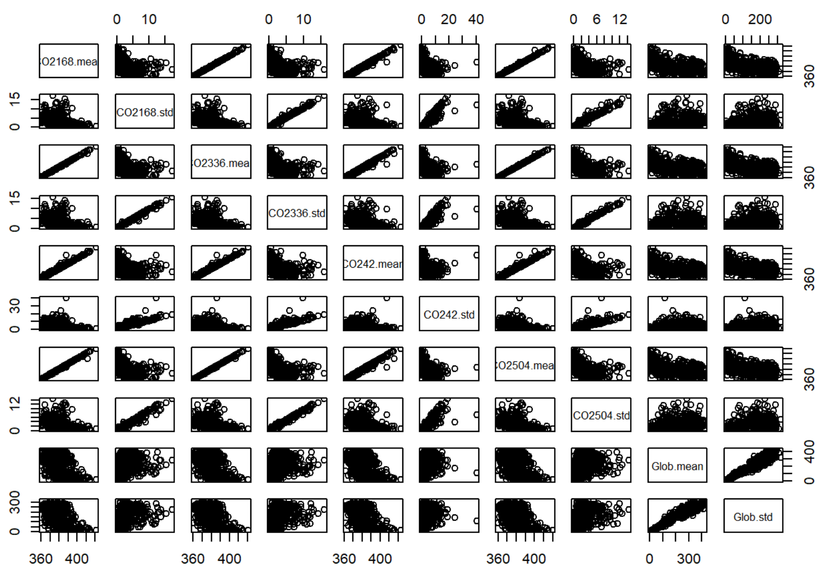
```
##  Mean   : 0.62245   Mean   : 1.47840   Mean   :0.51742   Mean   : 1.46693
##  3rd Qu.: 0.68409   3rd Qu.: 1.86923   3rd Qu.:0.63908   3rd Qu.: 1.90222
##  Max.   :11.31208   Max.   :16.19192   Max.   :5.76871   Max.   :16.09151
##    NOx672.std       NOx84.mean        NOx84.std        O3168.mean
##  Min.   :0.05122   Min.   : 0.07178   Min.   :0.05788   Min.   : 0.9544
##  1st Qu.:0.20630   1st Qu.: 0.49983   1st Qu.:0.20665   1st Qu.:26.6414
##  Median :0.33982   Median : 1.03106   Median :0.36150   Median :33.0944
##  Mean   :0.49949   Mean   : 1.51015   Mean   :0.53286   Mean   :33.1906
##  3rd Qu.:0.61895   3rd Qu.: 1.93005   3rd Qu.:0.65173   3rd Qu.:39.8784
##  Max.   :5.17625   Max.   :16.07575   Max.   :5.11127   Max.   :71.3321
##    O3168.std        O342.mean         O342.std         O3504.mean
##  Min.   : 0.2101   Min.   : 0.8273   Min.   : 0.2001   Min.   : 1.156
##  1st Qu.: 1.7522   1st Qu.:25.2315   1st Qu.: 1.9634   1st Qu.:28.100
##  Median : 3.3076   Median :31.7883   Median : 3.9185   Median :34.170
##  Mean   : 3.7427   Mean   :31.9879   Mean   : 4.2244   Mean   :34.135
##  3rd Qu.: 5.1082   3rd Qu.:38.7435   3rd Qu.: 5.9024   3rd Qu.:40.575
##  Max.   :12.4769   Max.   :69.9324   Max.   :12.3771   Max.   :72.510
##    O3504.std        O3672.mean        O3672.std         O384.mean
##  Min.   : 0.2249   Min.   : 0.8159   Min.   : 0.2252   Min.   : 0.8671
##  1st Qu.: 1.6531   1st Qu.:28.6850   1st Qu.: 1.6843   1st Qu.:25.8252
##  Median : 3.1024   Median :34.4689   Median : 3.0245   Median :32.4759
##  Mean   : 3.4762   Mean   :34.4586   Mean   : 3.3906   Mean   :32.5545
##  3rd Qu.: 4.7710   3rd Qu.:40.8428   3rd Qu.: 4.5269   3rd Qu.:39.1910
##  Max.   :12.4335   Max.   :72.9690   Max.   :12.3542   Max.   :70.5454
##     O384.std         Pamb0.mean       Pamb0.std          PAR.mean
##  Min.   : 0.09849   Min.   : 954.9   Min.   :0.06661   Min.   :  7.53
##  1st Qu.: 1.78944   1st Qu.: 984.7   1st Qu.:0.41527   1st Qu.:148.92
##  Median : 3.52155   Median : 991.9   Median :0.73789   Median :397.78
##  Mean   : 3.92492   Mean   : 991.4   Mean   :0.96126   Mean   :388.43
##  3rd Qu.: 5.51109   3rd Qu.: 998.7   3rd Qu.:1.27927   3rd Qu.:602.87
##  Max.   :12.38309   Max.   :1022.9   Max.   :5.79942   Max.   :825.89
##     PAR.std          PTG.mean            PTG.std          RGlob.mean
##  Min.   :  4.408   Min.   :-0.0077226   Min.   :0.000000   Min.   :-0.1367
##  1st Qu.: 95.378   1st Qu.:-0.0026694   1st Qu.:0.005002   1st Qu.:12.7102
##  Median :309.849   Median :-0.0001512   Median :0.008817   Median :28.9198
##  Mean   :288.375   Mean   : 0.0005468   Mean   :0.009649   Mean   :27.9264
##  3rd Qu.:466.141   3rd Qu.: 0.0010277   3rd Qu.:0.013380   3rd Qu.:42.2458
##  Max.   :613.025   Max.   : 0.1029073   Max.   :0.046563   Max.   :70.3992
##    RGlob.std       RHIRGA168.mean   RHIRGA168.std     RHIRGA336.mean
##  Min.   : 0.3671   Min.   : 27.71   Min.   : 0.1931   Min.   : 27.63
##  1st Qu.: 9.7500   1st Qu.: 51.75   1st Qu.: 3.3300   1st Qu.: 51.35
##  Median :21.6363   Median : 67.32   Median : 9.1339   Median : 68.00
##  Mean   :18.5868   Mean   : 68.03   Mean   : 8.4804   Mean   : 68.41
##  3rd Qu.:27.0281   3rd Qu.: 86.63   3rd Qu.:12.3753   3rd Qu.: 87.52
##  Max.   :37.3906   Max.   :106.02   Max.   :23.7104   Max.   :107.44
##  RHIRGA336.std     RHIRGA42.mean     RHIRGA42.std     RHIRGA504.mean
##  Min.   : 0.1949   Min.   : 29.66   Min.   : 0.2453   Min.   : 26.99
##  1st Qu.: 3.4477   1st Qu.: 52.99   1st Qu.: 2.7035   1st Qu.: 51.52
##  Median : 8.7077   Median : 67.94   Median : 9.8364   Median : 67.79
##  Mean   : 8.3424   Mean   : 68.65   Mean   : 8.9803   Mean   : 68.44
##  3rd Qu.:12.2265   3rd Qu.: 86.74   3rd Qu.:13.4855   3rd Qu.: 87.91
##  Max.   :24.1679   Max.   :103.13   Max.   :22.5871   Max.   :105.74
##  RHIRGA504.std     RHIRGA672.mean   RHIRGA672.std     RHIRGA84.mean
##  Min.   : 0.2377   Min.   : 26.70   Min.   : 0.1364   Min.   : 28.46
##  1st Qu.: 3.5330   1st Qu.: 51.82   1st Qu.: 3.4155   1st Qu.: 52.15
##  Median : 8.2929   Median : 68.38   Median : 8.0670   Median : 67.43
##  Mean   : 8.1769   Mean   : 69.12   Mean   : 8.1124   Mean   : 68.15
##  3rd Qu.:11.9507   3rd Qu.: 87.93   3rd Qu.:11.8755   3rd Qu.: 86.75
##  Max.   :24.5990   Max.   :106.31   Max.   :25.3959   Max.   :103.81
##   RHIRGA84.std      RPAR.mean         RPAR.std       SO2168.mean
##  Min.   : 0.244   Min.   : 0.000   Min.   : 0.000   Min.   :-0.02403
##  1st Qu.: 2.779   1st Qu.: 9.863   1st Qu.: 7.438   1st Qu.: 0.06199
##  Median : 9.613   Median :18.514   Median :14.144   Median : 0.12546
##  Mean   : 8.801   Mean   :18.995   Mean   :13.257   Mean   : 0.27825
##  3rd Qu.:13.001   3rd Qu.:25.311   3rd Qu.:17.060   3rd Qu.: 0.28617
##  Max.   :23.305   Max.   :88.191   Max.   :49.764   Max.   : 4.26351
##    SO2168.std        SWS.mean        SWS.std          T168.mean
##  Min.   :0.02423   Min.   :528.1   Min.   :  0.0000   Min.   :-24.778
##  1st Qu.:0.08065   1st Qu.:911.5   1st Qu.:  0.6999   1st Qu.: -1.264
##  Median :0.11904   Median :918.9   Median :  1.6287   Median :  8.157
##  Mean   :0.16702   Mean   :909.9   Mean   : 17.5475   Mean   :  6.511
##  3rd Qu.:0.18470   3rd Qu.:923.6   3rd Qu.: 16.5328   3rd Qu.: 14.626
##  Max.   :1.74204   Max.   :936.6   Max.   :190.6516   Max.   : 25.059
##     T168.std         T42.mean          T42.std          T504.mean
##  Min.   :0.03323   Min.   :-24.883   Min.   :0.02759   Min.   :-24.017
##  1st Qu.:0.83055   1st Qu.: -0.996   1st Qu.:0.83414   1st Qu.: -1.570
##  Median :1.87721   Median :  8.276   Median :2.04219   Median :  7.767
##  Mean   :1.86257   Mean   :  6.584   Mean   :2.05113   Mean   :  6.226
##  3rd Qu.:2.72364   3rd Qu.: 14.586   3rd Qu.:3.07854   3rd Qu.: 14.294
##  Max.   :5.24601   Max.   : 25.239   Max.   :5.53951   Max.   : 24.677
##     T504.std         T672.mean         T672.std          T84.mean
```

```
##  Min.   :0.02411   Min.   :-23.769   Min.   :0.03333   Min.   :-24.875
##  1st Qu.:0.81929   1st Qu.: -1.757   1st Qu.:0.79198   1st Qu.: -1.050
##  Median :1.70726   Median :  7.561   Median :1.66046   Median :  8.265
##  Mean   :1.71066   Mean   :  6.035   Mean   :1.64505   Mean   :  6.603
##  3rd Qu.:2.48396   3rd Qu.: 14.138   3rd Qu.:2.42862   3rd Qu.: 14.621
##  Max.   :5.16427   Max.   : 24.455   Max.   :5.12472   Max.   : 25.215
##     T84.std          UV_A.mean         UV_A.std          UV_B.mean
##  Min.   :0.03102   Min.   : 0.2958   Min.   : 0.1748   Min.   :0.005346
##  1st Qu.:0.87262   1st Qu.: 4.8984   1st Qu.: 2.8027   1st Qu.:0.146704
##  Median :2.00607   Median :11.8862   Median : 8.2665   Median :0.434569
##  Mean   :1.98711   Mean   :11.1052   Mean   : 7.8545   Mean   :0.458112
##  3rd Qu.:2.93527   3rd Qu.:17.2047   3rd Qu.:12.4985   3rd Qu.:0.723246
##  Max.   :5.42255   Max.   :22.5606   Max.   :16.8305   Max.   :1.242857
##     UV_B.std           CS.mean              CS.std
##  Min.   :0.003226   Min.   :0.0002273   Min.   :2.259e-05
##  1st Qu.:0.100064   1st Qu.:0.0016014   1st Qu.:2.902e-04
##  Median :0.373912   Median :0.0025608   Median :5.145e-04
##  Mean   :0.394200   Mean   :0.0031373   Mean   :7.017e-04
##  3rd Qu.:0.628708   3rd Qu.:0.0040417   3rd Qu.:8.710e-04
##  Max.   :1.074115   Max.   :0.0158369   Max.   :5.078e-03
```

```
npf <- npf[,-3]
```

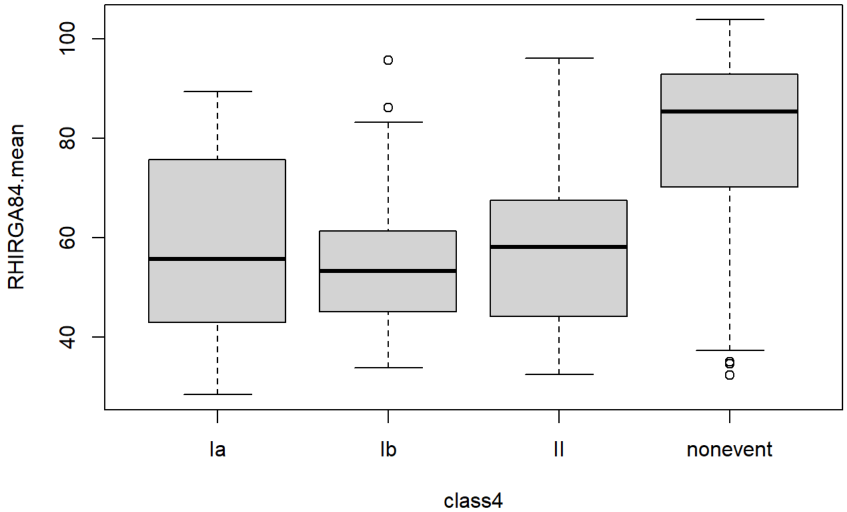
ii. Use the pairs() function to produce a scatterplot matrix of the first ten columns or variables of the data. Recall that you can reference the columns from 3 to 12 of a matrix A using A[,3:12].

```
pairs(npf[,3:12])
```



iii. Use the plot() function to produce side-by-side boxplots of event vs. nonevent days.

```
boxplot(RHIRGA84.mean ~ class4,npf)
```



iv. Create a new qualitative variable, called class2, which is “event” if there was a NPF event and “nonevent” otherwise.

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

Use the summary() function to see how many event days there are. Now use the plot() function to produce side-by-side boxplots of RHIRGA84.mean versus event.

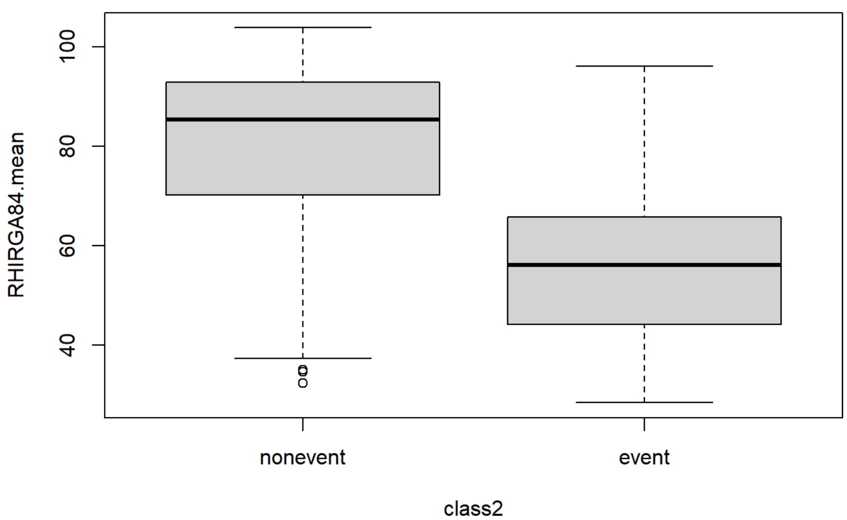
```
summary(npf)
```

```
##      date              class4           CO2168.mean      CO2168.std     
##  Length:430         Length:430         Min.   :360.5   Min.   : 0.1645  
##  Class :character   Class :character   1st Qu.:373.2   1st Qu.: 0.8874  
##  Mode  :character   Mode  :character   Median :380.5   Median : 2.2818  
##                                        Mean   :381.4   Mean   : 3.2596  
##                                        3rd Qu.:388.2   3rd Qu.: 4.6052  
##                                        Max.   :421.5   Max.   :17.2848  
##   CO2336.mean      CO2336.std        CO242.mean      CO242.std      
##  Min.   :360.4   Min.   : 0.1492   Min.   :361.8   Min.   : 0.1527  
##  1st Qu.:373.3   1st Qu.: 0.8955   1st Qu.:374.5   1st Qu.: 1.1535  
##  Median :380.5   Median : 2.1879   Median :381.4   Median : 2.6715  
##  Mean   :381.4   Mean   : 3.0728   Mean   :382.4   Mean   : 4.1952  
##  3rd Qu.:388.2   3rd Qu.: 4.2729   3rd Qu.:388.7   3rd Qu.: 6.1548  
##  Max.   :421.1   Max.   :15.9555   Max.   :422.6   Max.   :40.3667  
##   CO2504.mean      CO2504.std        Glob.mean          Glob.std      
##  Min.   :360.0   Min.   : 0.1342   Min.   :  3.719   Min.   :  1.998  
##  1st Qu.:373.3   1st Qu.: 0.8906   1st Qu.: 74.046   1st Qu.: 46.537  
##  Median :380.5   Median : 2.0735   Median :200.062   Median :154.675  
##  Mean   :381.3   Mean   : 2.8687   Mean   :196.704   Mean   :145.639  
##  3rd Qu.:388.2   3rd Qu.: 4.0826   3rd Qu.:313.597   3rd Qu.:233.168  
##  Max.   :419.9   Max.   :14.3424   Max.   :425.991   Max.   :320.099  
##   H2O168.mean       H2O168.std       H2O336.mean       H2O336.std     
##  Min.   : 0.8246   Min.   :0.01367   Min.   : 0.8285   Min.   :0.01682  
##  1st Qu.: 4.0293   1st Qu.:0.19046   1st Qu.: 4.0059   1st Qu.:0.18669  
##  Median : 6.3176   Median :0.46029   Median : 6.2786   Median :0.45464  
##  Mean   : 7.0840   Mean   :0.54496   Mean   : 7.0093   Mean   :0.54215  
##  3rd Qu.: 9.7301   3rd Qu.:0.79322   3rd Qu.: 9.5896   3rd Qu.:0.77506  
##  Max.   :18.6295   Max.   :3.05996   Max.   :18.5017   Max.   :3.07436  
##    H2O42.mean        H2O42.std        H2O504.mean       H2O504.std     
##  Min.   : 0.8006   Min.   :0.01689   Min.   : 0.8356   Min.   :0.01713  
##  1st Qu.: 4.0823   1st Qu.:0.19840   1st Qu.: 3.9892   1st Qu.:0.18734  
##  Median : 6.4746   Median :0.49102   Median : 6.3081   Median :0.44834  
##  Mean   : 7.2170   Mean   :0.55808   Mean   : 6.9676   Mean   :0.54082  
##  3rd Qu.: 9.9973   3rd Qu.:0.79199   3rd Qu.: 9.4671   3rd Qu.:0.78485  
##  Max.   :18.8547   Max.   :3.08742   Max.   :18.4412   Max.   :3.04878  
##   H2O672.mean      H2O672.std        H2O84.mean       H2O84.std      
##  Min.   : 0.864   Min.   :0.01851   Min.   : 0.8042   Min.   :0.01701  
##  1st Qu.: 3.948   1st Qu.:0.19262   1st Qu.: 4.0454   1st Qu.:0.19049  
##  Median : 6.250   Median :0.44970   Median : 6.3817   Median :0.47055  
##  Mean   : 6.937   Mean   :0.54252   Mean   : 7.1572   Mean   :0.55186  
##  3rd Qu.: 9.408   3rd Qu.:0.78014   3rd Qu.: 9.8693   3rd Qu.:0.79030  
##  Max.   :18.403   Max.   :3.06305   Max.   :18.7498   Max.   :3.09960  
##     NET.mean         NET.std         NO168.mean         NO168.std      
##  Min.   :-59.71   Min.   :  1.74   Min.   :-0.01484   Min.   :0.02096  
##  1st Qu.: 44.31   1st Qu.: 43.91   1st Qu.: 0.01624   1st Qu.:0.05077  
##  Median :127.43   Median :133.59   Median : 0.03597   Median :0.06351  
##  Mean   :124.55   Mean   :128.35   Mean   : 0.09691   Mean   :0.10245  
##  3rd Qu.:203.10   3rd Qu.:205.56   3rd Qu.: 0.08781   3rd Qu.:0.09631  
##  Max.   :302.83   Max.   :271.79   Max.   : 5.11143   Max.   :1.04660  
##    NO336.mean          NO336.std          NO42.mean           NO42.std      
##  Min.   :-0.008052   Min.   :0.02442   Min.   :-0.01227   Min.   :0.02213  
##  1st Qu.: 0.018905   1st Qu.:0.05024   1st Qu.: 0.01493   1st Qu.:0.04950  
##  Median : 0.039923   Median :0.06429   Median : 0.03493   Median :0.06329  
##  Mean   : 0.101441   Mean   :0.10035   Mean   : 0.08250   Mean   :0.11030  
##  3rd Qu.: 0.092411   3rd Qu.:0.09959   3rd Qu.: 0.07119   3rd Qu.:0.09269  
##  Max.   : 5.200000   Max.   :1.11947   Max.   : 5.03329   Max.   :1.79315  
##    NO504.mean         NO504.std         NO672.mean         NO672.std      
##  Min.   :-0.02333   Min.   :0.02535   Min.   :-0.01231   Min.   :0.02537  
##  1st Qu.: 0.01823   1st Qu.:0.05067   1st Qu.: 0.01833   1st Qu.:0.05118  
##  Median : 0.03841   Median :0.06367   Median : 0.03825   Median :0.06319  
##  Mean   : 0.09901   Mean   :0.09740   Mean   : 0.09667   Mean   :0.09565  
##  3rd Qu.: 0.09218   3rd Qu.:0.09614   3rd Qu.: 0.09055   3rd Qu.:0.09501  
##  Max.   : 5.21594   Max.   :1.16840   Max.   : 5.18457   Max.   :1.20562  
##    NO84.mean           NO84.std         NOx168.mean        NOx168.std     
##  Min.   :-0.02126   Min.   :0.02220   Min.   : 0.04289   Min.   :0.03606  
##  1st Qu.: 0.01344   1st Qu.:0.04876   1st Qu.: 0.49184   1st Qu.:0.20397  
##  Median : 0.03063   Median :0.06104   Median : 1.01360   Median :0.36246  
##  Mean   : 0.08419   Mean   :0.09821   Mean   : 1.51065   Mean   :0.51957  
##  3rd Qu.: 0.07383   3rd Qu.:0.09038   3rd Qu.: 1.93862   3rd Qu.:0.64662  
##  Max.   : 5.05143   Max.   :1.37139   Max.   :16.10108   Max.   :5.14500  
##   NOx336.mean        NOx336.std       NOx42.mean        NOx42.std      
##  Min.   : 0.02023   Min.   :0.0578   Min.   : 0.07533   Min.   : 0.06466  
##  1st Qu.: 0.49398   1st Qu.:0.1947   1st Qu.: 0.51880   1st Qu.: 0.22179  
##  Median : 1.02860   Median :0.3463   Median : 1.02928   Median : 0.37851  
##  Mean   : 1.49787   Mean   :0.5084   Mean   : 1.52797   Mean   : 0.62245  
##  3rd Qu.: 1.91334   3rd Qu.:0.6196   3rd Qu.: 1.96479   3rd Qu.: 0.68409  
##  Max.   :15.99608   Max.   :5.5703   Max.   :16.05554   Max.   :11.31208  
##   NOx504.mean        NOx504.std        NOx672.mean        NOx672.std     
##  Min.   : 0.05432   Min.   :0.05679   Min.   : 0.03304   Min.   :0.05122  
##  1st Qu.: 0.48199   1st Qu.:0.19619   1st Qu.: 0.48134   1st Qu.:0.20630  
##  Median : 1.01287   Median :0.34013   Median : 0.98863   Median :0.33982  
```

```
##  Mean   : 1.47840   Mean   :0.51742   Mean   : 1.46693   Mean   :0.49949  
##  3rd Qu.: 1.86923   3rd Qu.:0.63908   3rd Qu.: 1.90222   3rd Qu.:0.61895  
##  Max.   :16.19192   Max.   :5.76871   Max.   :16.09151   Max.   :5.17625  
##    NOx84.mean        NOx84.std        O3168.mean       O3168.std      
##  Min.   : 0.07178   Min.   :0.05788   Min.   : 0.9544   Min.   : 0.2101  
##  1st Qu.: 0.49983   1st Qu.:0.20665   1st Qu.:26.6414   1st Qu.: 1.7522  
##  Median : 1.03106   Median :0.36150   Median :33.0944   Median : 3.3076  
##  Mean   : 1.51015   Mean   :0.53286   Mean   :33.1906   Mean   : 3.7427  
##  3rd Qu.: 1.93005   3rd Qu.:0.65173   3rd Qu.:39.8784   3rd Qu.: 5.1082  
##  Max.   :16.07575   Max.   :5.11127   Max.   :71.3321   Max.   :12.4769  
##    O342.mean         O342.std        O3504.mean      O3504.std      
##  Min.   : 0.8273   Min.   : 0.2001   Min.   : 1.156   Min.   : 0.2249  
##  1st Qu.:25.2315   1st Qu.: 1.9634   1st Qu.:28.100   1st Qu.: 1.6531  
##  Median :31.7883   Median : 3.9185   Median :34.170   Median : 3.1024  
##  Mean   :31.9879   Mean   : 4.2244   Mean   :34.135   Mean   : 3.4762  
##  3rd Qu.:38.7435   3rd Qu.: 5.9024   3rd Qu.:40.575   3rd Qu.: 4.7710  
##  Max.   :69.9324   Max.   :12.3771   Max.   :72.510   Max.   :12.4335  
##    O3672.mean       O3672.std         O384.mean         O384.std       
##  Min.   : 0.8159   Min.   : 0.2252   Min.   : 0.8671   Min.   : 0.09849  
##  1st Qu.:28.6850   1st Qu.: 1.6843   1st Qu.:25.8252   1st Qu.: 1.78944  
##  Median :34.4689   Median : 3.0245   Median :32.4759   Median : 3.52155  
##  Mean   :34.4586   Mean   : 3.3906   Mean   :32.5545   Mean   : 3.92492  
##  3rd Qu.:40.8428   3rd Qu.: 4.5269   3rd Qu.:39.1910   3rd Qu.: 5.51109  
##  Max.   :72.9690   Max.   :12.3542   Max.   :70.5454   Max.   :12.38309  
##    Pamb0.mean       Pamb0.std          PAR.mean         PAR.std       
##  Min.   : 954.9   Min.   :0.06661   Min.   :  7.53   Min.   :  4.408  
##  1st Qu.: 984.7   1st Qu.:0.41527   1st Qu.:148.92   1st Qu.: 95.378  
##  Median : 991.9   Median :0.73789   Median :397.78   Median :309.849  
##  Mean   : 991.4   Mean   :0.96126   Mean   :388.43   Mean   :288.375  
##  3rd Qu.: 998.7   3rd Qu.:1.27927   3rd Qu.:602.87   3rd Qu.:466.141  
##  Max.   :1022.9   Max.   :5.79942   Max.   :825.89   Max.   :613.025  
##     PTG.mean             PTG.std           RGlob.mean        RGlob.std      
##  Min.   :-0.0077226   Min.   :0.000000   Min.   :-0.1367   Min.   : 0.3671  
##  1st Qu.:-0.0026694   1st Qu.:0.005002   1st Qu.:12.7102   1st Qu.: 9.7500  
##  Median :-0.0001512   Median :0.008817   Median :28.9198   Median :21.6363  
##  Mean   : 0.0005468   Mean   :0.009649   Mean   :27.9264   Mean   :18.5868  
##  3rd Qu.: 0.0010277   3rd Qu.:0.013380   3rd Qu.:42.2458   3rd Qu.:27.0281  
##  Max.   : 0.1029073   Max.   :0.046563   Max.   :70.3992   Max.   :37.3906  
##  RHIRGA168.mean   RHIRGA168.std     RHIRGA336.mean   RHIRGA336.std    
##  Min.   : 27.71   Min.   : 0.1931   Min.   : 27.63   Min.   : 0.1949  
##  1st Qu.: 51.75   1st Qu.: 3.3300   1st Qu.: 51.35   1st Qu.: 3.4477  
##  Median : 67.32   Median : 9.1339   Median : 68.00   Median : 8.7077  
##  Mean   : 68.03   Mean   : 8.4804   Mean   : 68.41   Mean   : 8.3424  
##  3rd Qu.: 86.63   3rd Qu.:12.3753   3rd Qu.: 87.52   3rd Qu.:12.2265  
##  Max.   :106.02   Max.   :23.7104   Max.   :107.44   Max.   :24.1679  
##  RHIRGA42.mean    RHIRGA42.std     RHIRGA504.mean   RHIRGA504.std    
##  Min.   : 29.66   Min.   : 0.2453   Min.   : 26.99   Min.   : 0.2377  
##  1st Qu.: 52.99   1st Qu.: 2.7035   1st Qu.: 51.52   1st Qu.: 3.5330  
##  Median : 67.94   Median : 9.8364   Median : 67.79   Median : 8.2929  
##  Mean   : 68.65   Mean   : 8.9803   Mean   : 68.44   Mean   : 8.1769  
##  3rd Qu.: 86.74   3rd Qu.:13.4855   3rd Qu.: 87.91   3rd Qu.:11.9507  
##  Max.   :103.13   Max.   :22.5871   Max.   :105.74   Max.   :24.5990  
##  RHIRGA672.mean   RHIRGA672.std     RHIRGA84.mean     RHIRGA84.std   
##  Min.   : 26.70   Min.   : 0.1364   Min.   : 28.46   Min.   : 0.244  
##  1st Qu.: 51.82   1st Qu.: 3.4155   1st Qu.: 52.15   1st Qu.: 2.779  
##  Median : 68.38   Median : 8.0670   Median : 67.43   Median : 9.613  
##  Mean   : 69.12   Mean   : 8.1124   Mean   : 68.15   Mean   : 8.801  
##  3rd Qu.: 87.93   3rd Qu.:11.8755   3rd Qu.: 86.75   3rd Qu.:13.001  
##  Max.   :106.31   Max.   :25.3959   Max.   :103.81   Max.   :23.305  
##    RPAR.mean         RPAR.std       SO2168.mean        SO2168.std     
##  Min.   : 0.000   Min.   : 0.000   Min.   :-0.02403   Min.   :0.02423  
##  1st Qu.: 9.863   1st Qu.: 7.438   1st Qu.: 0.06199   1st Qu.:0.08065  
##  Median :18.514   Median :14.144   Median : 0.12546   Median :0.11904  
##  Mean   :18.995   Mean   :13.257   Mean   : 0.27825   Mean   :0.16702  
##  3rd Qu.:25.311   3rd Qu.:17.060   3rd Qu.: 0.28617   3rd Qu.:0.18470  
##  Max.   :88.191   Max.   :49.764   Max.   : 4.26351   Max.   :1.74204  
##     SWS.mean        SWS.std           T168.mean          T168.std      
##  Min.   :528.1   Min.   :  0.0000   Min.   :-24.778   Min.   :0.03323  
##  1st Qu.:911.5   1st Qu.:  0.6999   1st Qu.: -1.264   1st Qu.:0.83055  
##  Median :918.9   Median :  1.6287   Median :  8.157   Median :1.87721  
##  Mean   :909.9   Mean   : 17.5475   Mean   :  6.511   Mean   :1.86257  
##  3rd Qu.:923.6   3rd Qu.: 16.5328   3rd Qu.: 14.626   3rd Qu.:2.72364  
##  Max.   :936.6   Max.   :190.6516   Max.   : 25.059   Max.   :5.24601  
##     T42.mean          T42.std          T504.mean          T504.std      
##  Min.   :-24.883   Min.   :0.02759   Min.   :-24.017   Min.   :0.02411  
##  1st Qu.: -0.996   1st Qu.:0.83414   1st Qu.: -1.570   1st Qu.:0.81929  
##  Median :  8.276   Median :2.04219   Median :  7.767   Median :1.70726  
##  Mean   :  6.584   Mean   :2.05113   Mean   :  6.226   Mean   :1.71066  
##  3rd Qu.: 14.586   3rd Qu.:3.07854   3rd Qu.: 14.294   3rd Qu.:2.48396  
##  Max.   : 25.239   Max.   :5.53951   Max.   : 24.677   Max.   :5.16427  
##    T672.mean          T672.std          T84.mean           T84.std      
```

```
##  Min.   :-23.769   Min.   :0.03333   Min.   :-24.875   Min.   :0.03102
##  1st Qu.: -1.757   1st Qu.:0.79198   1st Qu.: -1.050   1st Qu.:0.87262
##  Median :  7.561   Median :1.66046   Median :  8.265   Median :2.00607
##  Mean   :  6.035   Mean   :1.64505   Mean   :  6.603   Mean   :1.98711
##  3rd Qu.: 14.138   3rd Qu.:2.42862   3rd Qu.: 14.621   3rd Qu.:2.93527
##  Max.   : 24.455   Max.   :5.12472   Max.   : 25.215   Max.   :5.42255
##    UV_A.mean         UV_A.std        UV_B.mean          UV_B.std
##  Min.   : 0.2958   Min.   : 0.1748   Min.   :0.005346   Min.   :0.003226
##  1st Qu.: 4.8984   1st Qu.: 2.8027   1st Qu.:0.146704   1st Qu.:0.100064
##  Median :11.8862   Median : 8.2665   Median :0.434569   Median :0.373912
##  Mean   :11.1052   Mean   : 7.8545   Mean   :0.458112   Mean   :0.394200
##  3rd Qu.:17.2047   3rd Qu.:12.4985   3rd Qu.:0.723246   3rd Qu.:0.628708
##  Max.   :22.5606   Max.   :16.8305   Max.   :1.242857   Max.   :1.074115
##     CS.mean              CS.std                class2
##  Min.   :0.0002273   Min.   :2.259e-05   nonevent:215
##  1st Qu.:0.0016014   1st Qu.:2.902e-04   event   :215
##  Median :0.0025608   Median :5.145e-04
##  Mean   :0.0031373   Mean   :7.017e-04
##  3rd Qu.:0.0040417   3rd Qu.:8.710e-04
##  Max.   :0.0158369   Max.   :5.078e-03
```

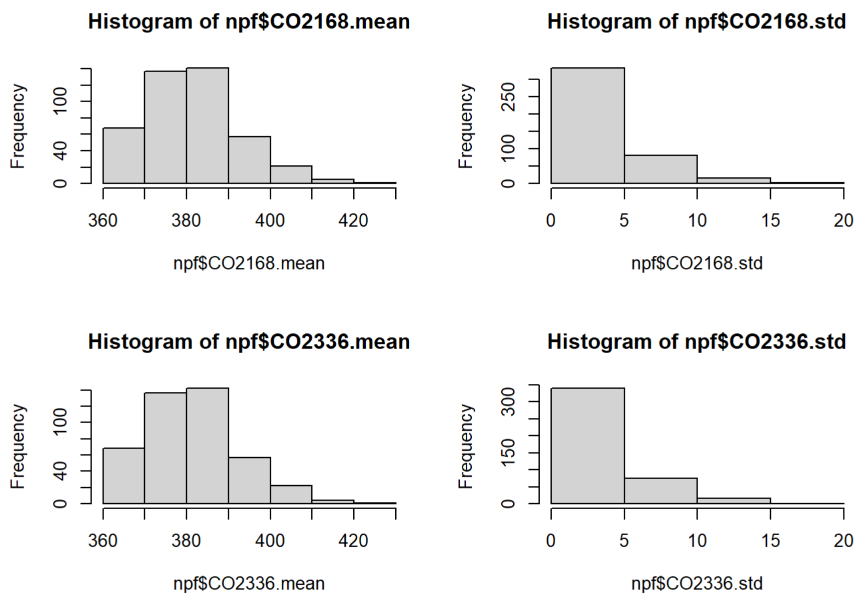
```
boxplot(RHIRGA84.mean ~ class2,npf)
```



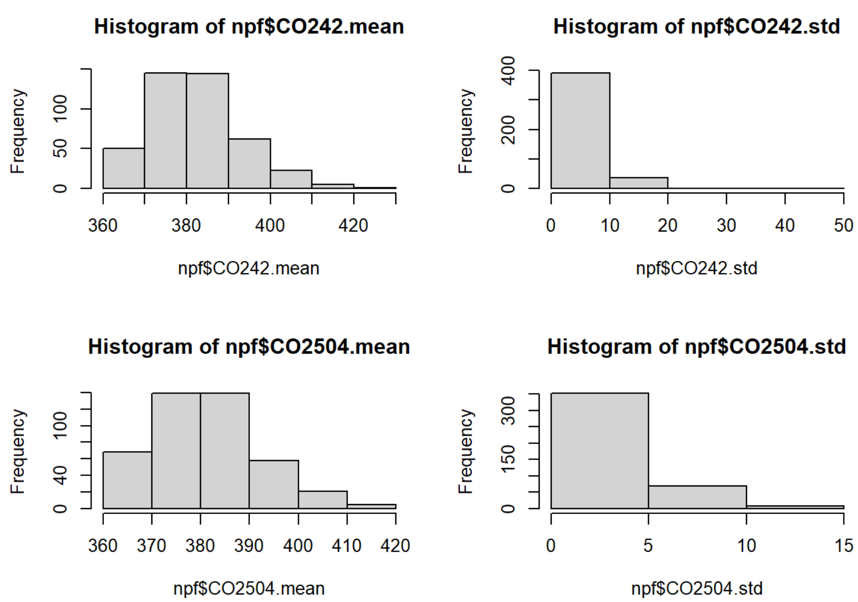
As we can see we have 215 event in our dataframe.

v. Use the hist() function to produce some histograms with differing numbers of bins for a few of the quantitative variables. You may find the command par(mfrow=c(2,2)) useful: it will divide the print window into four regions so that four plots can be made simultaneously. Modifying the arguments to this function will divide the screen in other ways.

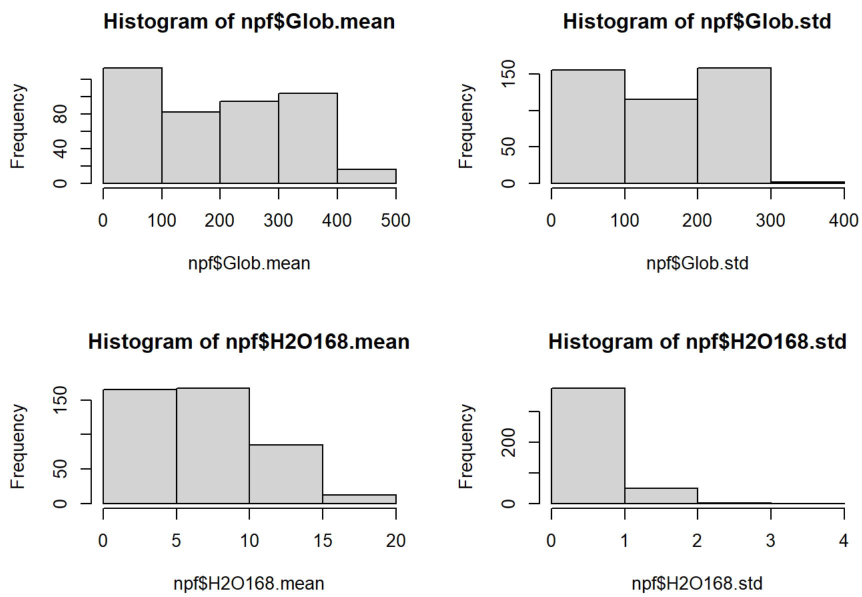
```
par(mfrow=c(2,2))
hist(npf$CO2168.mean,breaks=6)
hist(npf$CO2168.std,breaks=4)
hist(npf$CO2336.mean,breaks=6)
hist(npf$CO2336.std,breaks=4)
```




```
par(mfrow=c(2,2))
hist(npf$CO242.mean,breaks=6)
hist(npf$CO242.std,breaks=4)
hist(npf$CO2504.mean,breaks=6)
hist(npf$CO2504.std,breaks=4)
```




```
par(mfrow=c(2,2))
hist(npf$Glob.mean,breaks=6)
hist(npf$Glob.std,breaks=4)
hist(npf$H2O168.mean,breaks=6)
hist(npf$H2O168.std,breaks=4)
```



vi. Continue exploring the data, and provide a brief summary of what you discover.

```
str(npf)
```

```
## 'data.frame':    430 obs. of  103 variables:
##  $ date          : chr  "2000-02-23" "2000-03-25" "2000-04-06" "2000-04-11" ...
##  $ class4        : chr  "nonevent" "Ib" "Ib" "nonevent" ...
##  $ CO2168.mean   : num  381 373 372 381 375 ...
##  $ CO2168.std    : num  0.802 1.097 0.626 7.281 3.264 ...
##  $ CO2336.mean   : num  380 373 372 381 375 ...
##  $ CO2336.std    : num  0.89 1.048 0.616 7.236 3.111 ...
##  $ CO242.mean    : num  382 374 373 382 376 ...
##  $ CO242.std     : num  1.293 1.259 0.647 7.294 3.275 ...
##  $ CO2504.mean   : num  380 373 372 381 375 ...
##  $ CO2504.std    : num  0.969 1.004 0.596 7.208 2.904 ...
##  $ Glob.mean     : num  236.6 252.5 270 68.4 242.2 ...
##  $ Glob.std      : num  145.2 138.9 200.8 48.6 191 ...
##  $ H2O168.mean   : num  2.66 3.25 4.46 6.61 7.93 ...
##  $ H2O168.std    : num  0.319 0.3 0.368 0.508 0.326 ...
##  $ H2O336.mean   : num  2.7 3.23 4.42 6.57 7.88 ...
##  $ H2O336.std    : num  0.305 0.308 0.365 0.498 0.302 ...
##  $ H2O42.mean    : num  2.55 3.3 4.51 6.63 8.11 ...
##  $ H2O42.std     : num  0.383 0.29 0.361 0.518 0.368 ...
##  $ H2O504.mean   : num  2.69 3.23 4.41 6.58 7.86 ...
##  $ H2O504.std    : num  0.307 0.31 0.359 0.47 0.299 ...
##  $ H2O672.mean   : num  2.77 3.23 4.4 6.57 7.84 ...
##  $ H2O672.std    : num  0.366 0.308 0.358 0.454 0.296 ...
##  $ H2O84.mean    : num  2.61 3.27 4.46 6.62 8 ...
##  $ H2O84.std     : num  0.349 0.298 0.363 0.527 0.351 ...
##  $ NET.mean      : num  81.7 142.5 156.4 53.8 160.4 ...
##  $ NET.std       : num  109.2 115.9 173.2 44.5 149.7 ...
##  $ NO168.mean    : num  0.3193 0.0236 0.0309 0.7174 0.0689 ...
##  $ NO168.std     : num  0.1796 0.0403 0.0479 1.0466 0.1149 ...
##  $ NO336.mean    : num  0.3367 0.0281 0.03 0.7636 0.066 ...
##  $ NO336.std     : num  0.184 0.0421 0.0461 1.1195 0.1178 ...
##  $ NO42.mean     : num  0.2355 0.0253 0.0288 0.6014 0.044 ...
##  $ NO42.std      : num  0.1575 0.0454 0.0471 0.923 0.0823 ...
##  $ NO504.mean    : num  0.3325 0.0279 0.0249 0.8009 0.0709 ...
##  $ NO504.std     : num  0.1835 0.0468 0.0455 1.1684 0.1149 ...
##  $ NO672.mean    : num  0.2872 0.0305 0.0299 0.8224 0.0607 ...
##  $ NO672.std     : num  0.1726 0.0429 0.0524 1.2056 0.102 ...
##  $ NO84.mean     : num  0.2866 0.0254 0.0288 0.663 0.0556 ...
##  $ NO84.std      : num  0.1601 0.0943 0.0509 0.9876 0.1045 ...
##  $ NOx168.mean   : num  2.658 0.843 0.748 5.32 1.79 ...
##  $ NOx168.std    : num  0.672 0.16 0.208 3.781 0.55 ...
##  $ NOx336.mean   : num  2.65 0.83 0.736 5.339 1.795 ...
##  $ NOx336.std    : num  0.667 0.158 0.174 3.754 0.531 ...
##  $ NOx42.mean    : num  2.622 0.915 0.775 5.214 1.78 ...
##  $ NOx42.std     : num  0.646 0.574 0.254 3.688 0.559 ...
##  $ NOx504.mean   : num  2.642 0.825 0.705 5.385 1.822 ...
##  $ NOx504.std    : num  0.667 0.163 0.157 3.786 0.567 ...
##  $ NOx672.mean   : num  2.61 0.834 0.71 5.367 1.794 ...
##  $ NOx672.std    : num  0.685 0.175 0.153 3.754 0.531 ...
##  $ NOx84.mean    : num  2.645 0.864 0.771 5.273 1.796 ...
##  $ NOx84.std     : num  0.653 0.229 0.246 3.719 0.565 ...
##  $ O3168.mean    : num  32.6 48 46 25.8 44.4 ...
##  $ O3168.std     : num  0.698 3.039 1.893 11.283 8.022 ...
##  $ O342.mean     : num  31.3 47.6 45.7 25 42.9 ...
##  $ O342.std      : num  1.85 3.27 1.92 11.2 8.54 ...
##  $ O3504.mean    : num  32.9 48.1 46.1 26.1 46.2 ...
##  $ O3504.std     : num  0.892 2.781 1.895 11.229 6.354 ...
##  $ O3672.mean    : num  32.7 48.2 46.1 26.2 46.5 ...
##  $ O3672.std     : num  0.928 2.679 1.89 11.275 6.058 ...
##  $ O384.mean     : num  32.2 47.7 45.9 25.4 43.8 ...
##  $ O384.std      : num  0.938 3.193 1.944 11.25 8.359 ...
##  $ Pamb0.mean    : num  1007 993 987 991 998 ...
##  $ Pamb0.std     : num  0.217 0.281 2.777 0.27 2.499 ...
##  $ PAR.mean      : num  339 488 516 147 473 ...
##  $ PAR.std       : num  211 269 392 103 372 ...
##  $ PTG.mean      : num  0.000964 -0.00534 -0.00291 -0.000718 0.005027 ...
##  $ PTG.std       : num  0.00748 0.00878 0.0067 0.00493 0.02241 ...
##  $ RGlob.mean    : num  67.5 41.1 41 10.6 28.2 ...
##  $ RGlob.std     : num  30.53 21.32 29.48 6.54 22.32 ...
##  $ RHIRGA168.mean: num  95.1 59.7 65 90.9 60.9 ...
##  $ RHIRGA168.std : num  1.61 14.63 13.98 8.72 13.54 ...
##  $ RHIRGA336.mean: num  96.5 60.1 65.2 91 60.5 ...
##  $ RHIRGA336.std : num  2.38 14.37 14.18 8.51 12.68 ...
##  $ RHIRGA42.mean : num  92.2 59.5 64.8 89.3 61.4 ...
##  $ RHIRGA42.std  : num  1.78 15.59 13.83 8.52 14.62 ...
##  $ RHIRGA504.mean: num  96.8 60.9 65.9 91.9 60.3 ...
##  $ RHIRGA504.std : num  2.33 14.12 14.17 8.11 11.79 ...
##  $ RHIRGA672.mean: num  101.4 62.5 67.5 94 60.8 ...
##  $ RHIRGA672.std : num  4.57 14.22 14.65 7.92 10.48 ...
##  $ RHIRGA84.mean : num  93.3 59.2 64.3 89.8 60.5 ...
##  $ RHIRGA84.std  : num  1.98 15.26 13.82 8.9 13.97 ...
```

```
##  $ RPAR.mean     : num  84.5 32.4 32.9 11.9 17.8 ...
##  $ RPAR.std      : num  49.76 19.52 25.39 7.74 14.73 ...
##  $ SO2168.mean   : num  0.559 0.138 0.107 0.324 0.366 ...
##  $ SO2168.std    : num  0.375 0.115 0.123 0.227 0.324 ...
##  $ SWS.mean      : num  937 923 923 919 920 ...
##  $ SWS.std       : num  0.916 2.062 2.648 17.331 40.317 ...
##  $ T168.mean     : num  -10.27 -1.33 1.67 2.32 11.21 ...
##  $ T168.std      : num  1.575 1.947 1.943 0.374 2.933 ...
##  $ T42.mean      : num  -10.49 -1.04 1.89 2.61 11.42 ...
##  $ T42.std       : num  2.085 2.232 1.96 0.392 3.199 ...
##  $ T504.mean     : num  -10.35 -1.74 1.35 2.11 11.14 ...
##  $ T504.std      : num  1.347 1.748 1.91 0.338 2.455 ...
##  $ T672.mean     : num  -10.731 -2.096 0.992 1.753 10.94 ...
##  $ T672.std      : num  1.382 1.696 1.914 0.341 2.18 ...
##  $ T84.mean      : num  -10.28 -1.1 1.85 2.52 11.44 ...
##  $ T84.std       : num  1.87 2.09 1.955 0.414 3.049 ...
##  $ UV_A.mean     : num  8.36 12.91 14.29 4.95 13.09 ...
##  $ UV_A.std      : num  4.53 7.02 9.57 3.41 9.77 ...
##  $ UV_B.mean     : num  0.178 0.334 0.418 0.224 0.526 ...
##   [list output truncated]
```

-In our dataframe we have 430 observations and 103 variables from which only two aren't numerical. -The ScatterPlot done above with pairs() shows us that we have many correlated variables in our data. For example: . CO2168.mean and CO242 are correlated . CO2168.mean and CO2504 are correlated And many more …

-The boxplot of RHIRGA84.mean vs Classes shows that being not an event could increase the value of this variable (the 2 boxplots have a significatnt difference in their height). Thus, the class variable could affect the RHIRGA84.mean .

Problem 3

[10% points]

Objective: how to deal with probabilities, relevance of floating point arithmetics.

In machine learning we often have to deal with very small (or large) numbers. Take, for example, the probability density of the normal distribution given by $p(x) = \exp\left(-x^2/2\right)/\sqrt{2\pi}$ or in R, `p <- function(x) exp(-x^2/2)/sqrt(2*pi)` , which produces to very small numbers for any larger values of x.

Multiplication, division, and sums can easily lead to under or overflows. We can mitigate the problem by representing the probabilities as logarithms and then doing the multiplication, division, and summation using logarithmic values. I.e., instead of a probability p_i, where $i \in [n] = \{1, \ldots, n\}$, we use the natural logarithms $l_i = \log p_i$ to present the numbers. Denote the product of probabilities by $p_P = \prod_{i=1}^n p_i^{a_i}$ and sum by $p_S = \sum_{i=1}^n p_i$. Furthermore, denote $l_P = \log p_P$ and $l_S = \log p_S$.

Task a

Show the following equalities are true:

- $l_P = \sum_{i=1}^n a_i l_i$

$l_P = log(P_p) = log(\prod_{i=1}^n p_i^{a_i}) = \sum_{i=1}^n log(p_i^{a_i}) = \sum_{i=1}^n a_i log(p_i) = \sum_{i=1}^n a_i l_i$

- $l_S = \max_{j\in[n]} l_j + \log \sum_{i=1}^n e^{l_i - \max_{j\in[n]} l_j}$.

$\max_{j\in[n]} l_j + \log \sum_{i=1}^n e^{l_i - \max_{j\in[n]} l_j} = \max_{j\in[n]} l_j + \log \sum_{i=1}^n e^{l_i} * e^{-\max_{j\in[n]} l_j} =$
$\max_{j\in[n]} l_j + \log e^{-\max_{j\in[n]} l_j} * \sum_{i=1}^n e^{l_i} = \max_{j\in[n]} l_j + \log e^{-\max_{j\in[n]} l_j} + \log \sum_{i=1}^n e^{l_i} = \max_{j\in[n]} l_j - \max_{j\in[n]} l_j + \log \sum_{i=1}^n e^{l_i}$
$= \log \sum_{i=1}^n e^{l_i} = \log \sum_{i=1}^n e^{\log p_i} = \log \sum_{i=1}^n p_i = \log P_s = l_S$

Task b

Implement the product and sum operations (i.e, formulas for l_P and l_S) as R or Python functions. Compute the value of the following expression by without the log presentation and by using the log representation and the functions you implemented above: $p(100)/(p(100) + p(100.01))$, where $p(x) = \exp\left(-x^2/2\right)/\sqrt{2\pi}$.

You should notice that without the "log trick" you should obtain a NaN value because of floating points underflows, but with the log trick you should obtain a correct answer which in this case should be about 0.731. Operations like this could appear later, e.g., in Naive Bayes classifier or other machine learning computations.

```r
p = function(x) {
  exp(-x^2/2)/sqrt(2*pi)
}
p(100)/(p(100) + p(100.01))
```

```
## [1] NaN
```

Without using the log representation we get a NAN

```
li=function(x){
  tmp = - x * x / 2 - log(sqrt(2*pi))
  return(tmp)
}

LP = function(li){  #Vect_Proba contains the probabilities that need to be summed
  sum=0
  for (i in 1:length(li) ){
    sum=sum + li[i]
  }
  return(sum)
}

LS = function(li){
  max=max(li)
  sum=0
  for (i in 1:length(li) ){
    sum=sum + exp(li[i] - max)
  }
  sum= log(sum) + max
  return(sum)
}

answer = exp( li(100) - LS(c(li(100),li(100.01))) )

paste("The Answer is",answer)
```

```
## [1] "The Answer is 0.731068409113252"
```

Task c

Read through the R documentation of numerical characteristics of your computer by running R command `help(.Machine)` and `help(Inf)` and experiment with R.

- What do expressions giving very small and large numbers evaluate to, e.g., `exp(-1000)` and `exp(1000)` ?

exp(1000) = Inf exp(-1000) = 0

double.xmin the smallest non-zero normalized floating-point number, a power of the radix, i.e., double.base ^ double.min.exp. Normally 2.225074e-308.

Therefore the smallest number is evaluated as 2.225074e-308.

double.xmax the largest normalized floating-point number. Typically, it is equal to (1 - double.neg.eps) * double.base ^ double.max.exp, but on some machines it is only the second or third largest such number, being too small by 1 or 2 units in the last digit of the significand. Normally 1.797693e+308

The biggest number is evaluated to Normally 1.797693e+308.

- What is `1/Inf` and what does it evaluate to? 1/Inf = 0
- Let `x` be the smallest positive number such that the expression `1+x!=1` is true. What is `x` called and what is its numerical value in your computer?

double.eps
the smallest positive floating-point number x such that 1 + x != 1. It equals double.base ^ ulp.digits if either double.base is 2 or double.rounding is 0; otherwise, it is (double.base ^ double.ulp.digits) / 2. Normally 2.220446e-16.

Therefore in my computer, this x is equal to 2.220446e-16.

Problem 4

NOT DONE!

Problem 5

[20% points]

Objective: learning linear regression, concrete use of validation set, k-fold cross validation

In this problem we study linear regression on a synthetically generated dataset, with underlying function $f(x) = 1 + x - x^2/2$. The idea here is also to apply the theory in the previous problem into practice.

Task a

Create a *training set* of 20 pairs (x_i, y_i), where $i \in S_{tr} = [20] = \{1, \ldots, 20\}$, where x_i are sampled uniformly from interval $[-3, 3]$ (R function `runif`) and $y_i = f(x_i) + \epsilon_i$, where ϵ_i are i.i.d. normal random variables with zero mean and standard deviation of 0.4 (R function `rnorm`); this is your distribution F! Using the same procedure and distributions sample a *validation set* $S_{va} = \{21, \ldots, 40\}$ of 20 pairs and a *test set* $S_{te} = \{41, \ldots, 1040\}$ of 1000 pairs. In total, you should now therefore have 1040 pairs.

```r
f = function (x){
  return (1 + x - x*x/2)
}

MSE1 = function(y,ypred){
  sum=0
  for (i in 1:length(y)){
    sum=sum + ( y[i] - ypred[i] )^2
  }
  sum=sum/length(y)
  return (sum)
}



xtr=runif(20,-3,3)
xtr=sort(xtr)
ytr=f(xtr)
eps=rnorm(20,mean=0,sd=0.4)
ytr=ytr+eps
TrainData=data.frame(xtr,ytr)
names(TrainData)=c('x','y')
TRAIN=TrainData


xval=runif(20,-3,3)
xval=sort(xval)
yval=f(xval)
epsval=rnorm(20,mean=0,sd=0.4)
yval=yval+epsval
valData=data.frame(xval,yval)
names(valData)=c('x','y')

xte=runif(1000,-3,3)
xte=sort(xte)
yte=f(xte)
epste=rnorm(1000,mean=0,sd=0.4)
yte=yte+epste
testData=data.frame(xte,yte)
names(testData)=c('x','y')
```

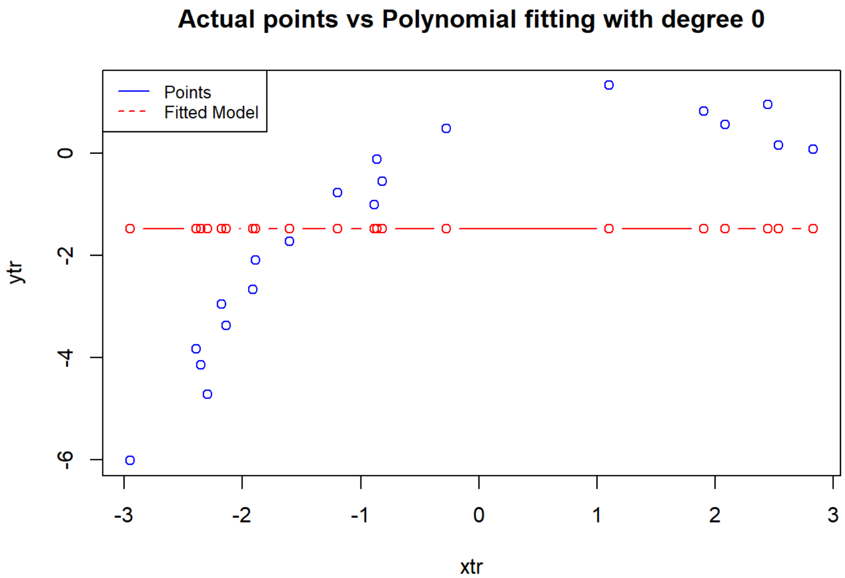
Task b

Lets choose $K = 11$ and let the learning algorithms to be OLS linear regression with polynomials of degree $k-1$, where $k \in [K]$. More specifically, fit polynomials $\hat{y} = \sum_{p=0}^{k-1} w_p x^p$ to the training set (of 20 data items) for different orders $k \in [K]$ by using ordinary least squares (OLS) regression. For each 11 values of k produce a plot showing the points (x_i, y_i) in the training set and the fitted polynomial in interval $[-3, 3]$. Calculate and report the means squared error (MSE) on training set, where $\mathrm{MSE}_{tr} = \sum_{i \in S_{tr}}^{n} (y_i - \hat{y}_i)^2 / n_{tr}$, and compare the MSE for the different orders of polynomials.

```r
mse=rep(0,11)
mseVAL=rep(0,11)
mseTesT=rep(0,11)

models <- list()

Train.fit <-lm(y~ 1,TrainData)
models[[1]] =  Train.fit
summary(Train.fit)
```

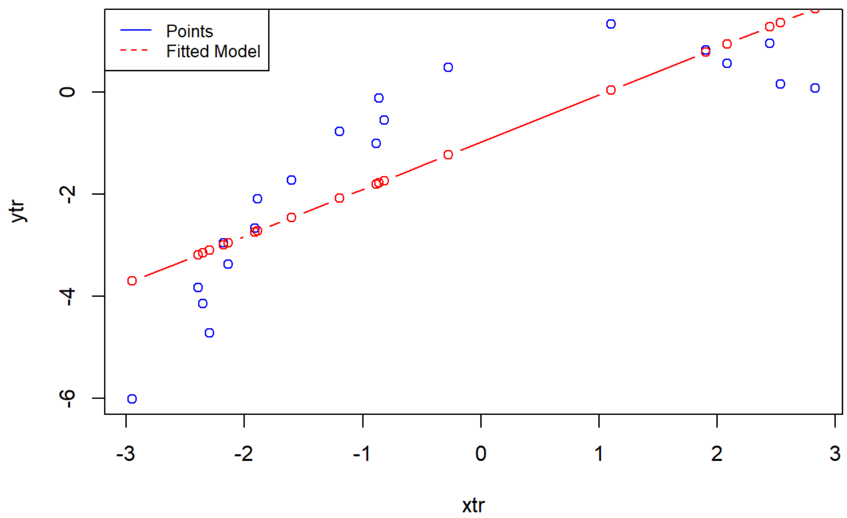
```
##
## Call:
## lm(formula = y ~ 1, data = TrainData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5353 -1.5791  0.5873  1.7191  2.8058
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.4844     0.4782  -3.104  0.00584 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.139 on 19 degrees of freedom
```

```
y_pred_TR=predict(Train.fit,TrainData)

plot(xtr,ytr,col='blue' , main=paste("Actual points vs Polynomial fitting with degree",0) )
lines(sort(xtr), fitted(Train.fit)[order(xtr)], col='red', type='b')
legend("topleft", legend=c("Points", "Fitted Model"),
       col=c("blue", "Red"), lty=1:2, cex=0.8)
```

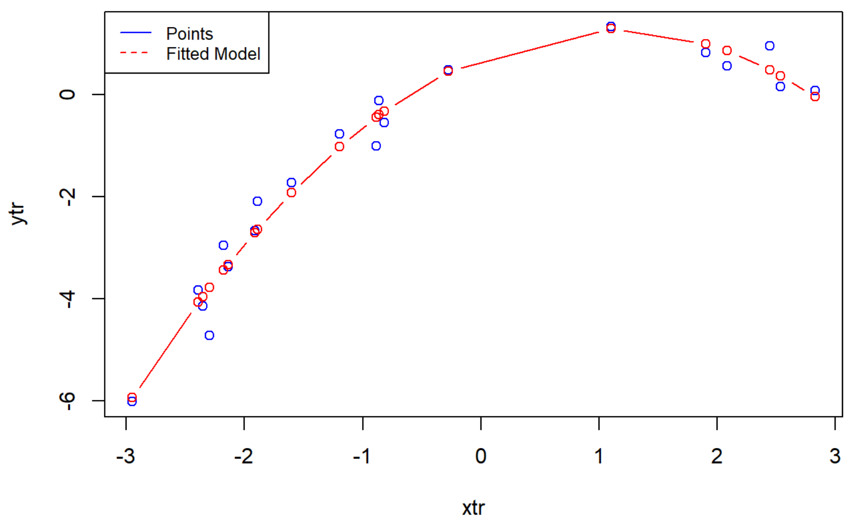



```
mse[1] = MSE1(ytr,y_pred_TR)


y_predTest=predict(Train.fit,testData)
y_predValidation=predict(Train.fit,valData)


mseVAL[1] = MSE1(yval,y_predValidation)
mseTesT[1] = MSE1(yte,y_predTest)

fitWithK=function(k){
  tmp=lm(y~ poly(x,k,raw=TRUE),TrainData)
}

for (k in 2:11) {
Train.fit <-fitWithK(k-1)
models[[k]] =  Train.fit
summary(Train.fit)
y_pred_TR=predict(Train.fit,TrainData)


plot(xtr,ytr,col='blue' , main=paste("Actual points vs Polynomial fitting with degree",k-1) )
lines(sort(xtr), fitted(Train.fit)[order(xtr)], col='red', type='b')
legend("topleft", legend=c("Points", "Fitted Model"),
       col=c("blue", "Red"), lty=1:2, cex=0.8)
mse[k] = MSE1(ytr,y_pred_TR)


y_predTest=predict(Train.fit,testData)
y_predValidation=predict(Train.fit,valData)


mseVAL[k] = MSE1(yval,y_predValidation)
mseTesT[k] = MSE1(yte,y_predTest)
}
```

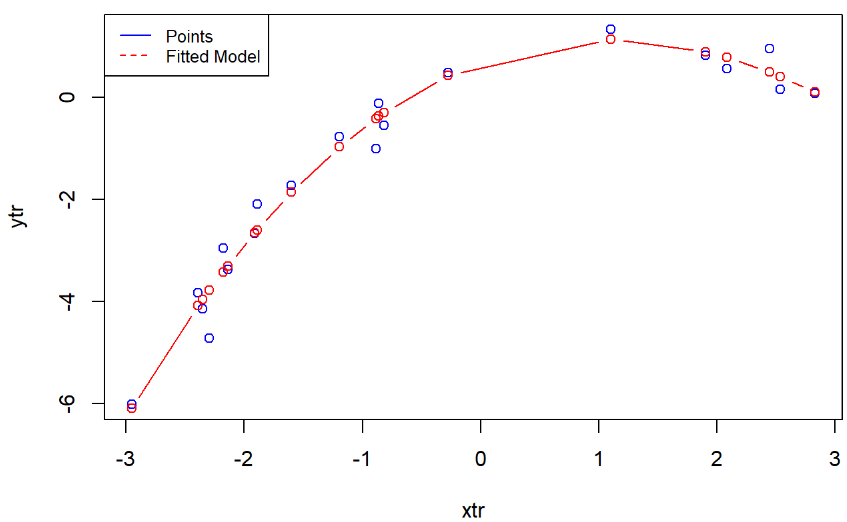
Actual points vs Polynomial fitting with degree 1

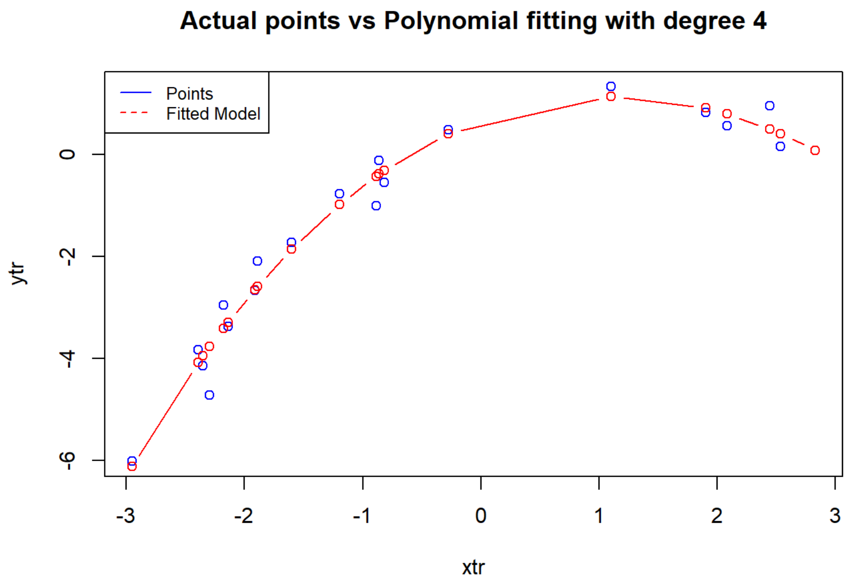


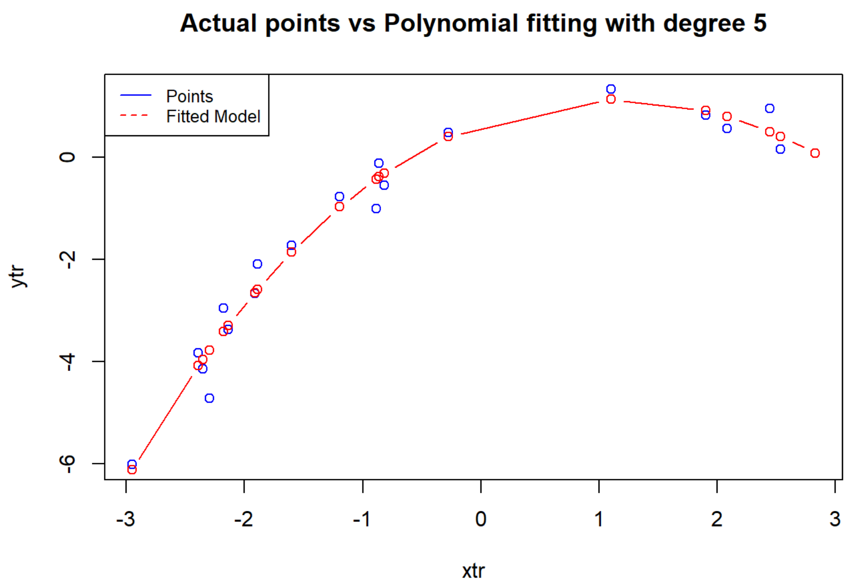
Actual points vs Polynomial fitting with degree 2

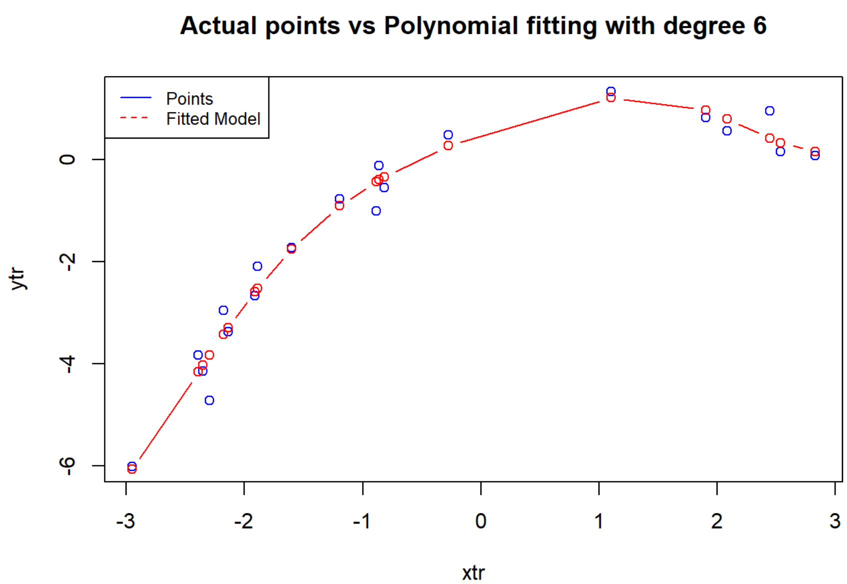


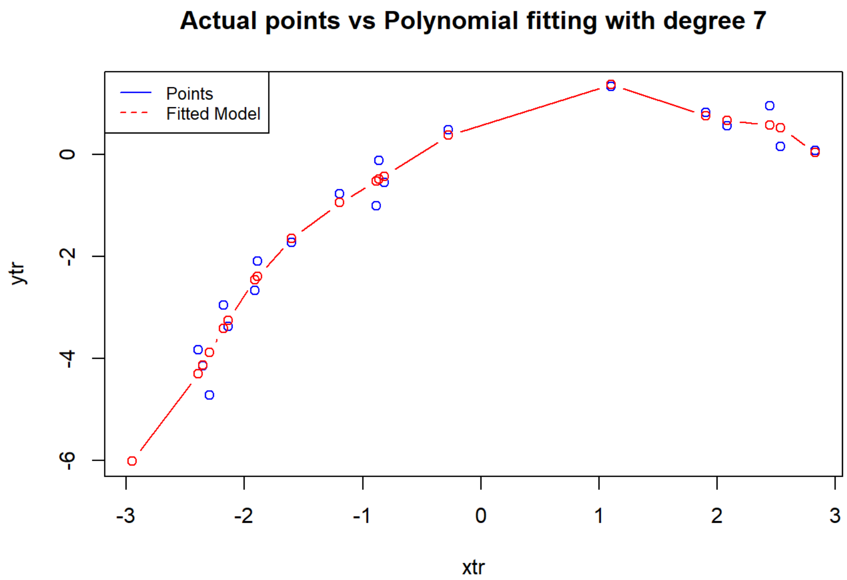
Actual points vs Polynomial fitting with degree 3

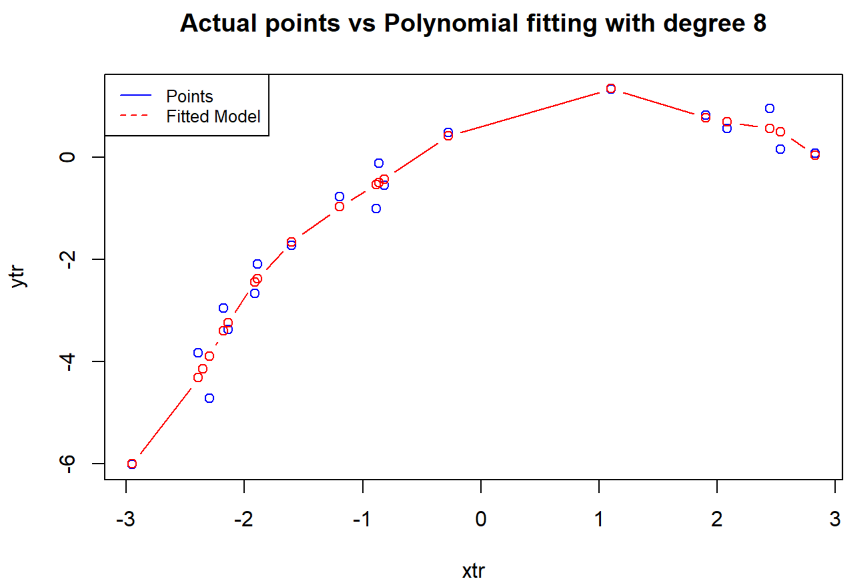


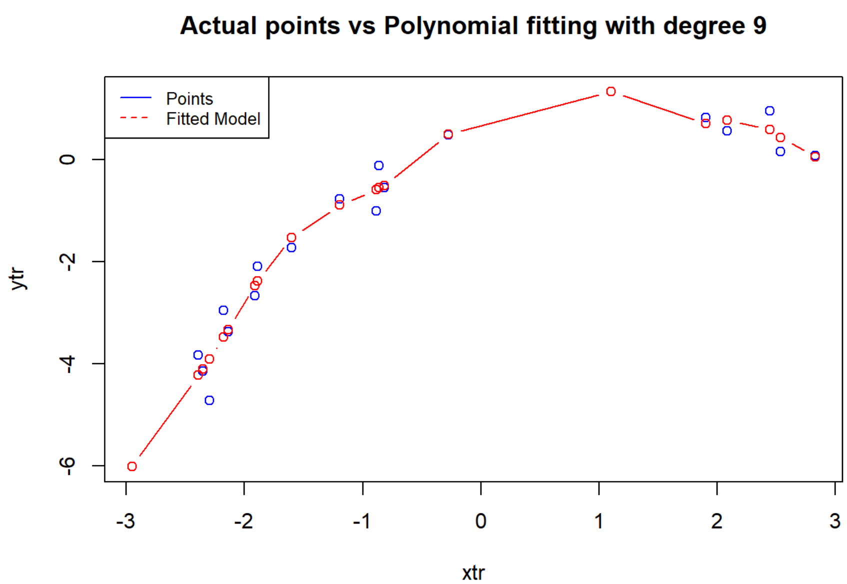

Actual points vs Polynomial fitting with degree 4
Points
Fitted Model
ytr
xtr
0
-2
-4
-6
-3
-2
-1
0
1
2
3

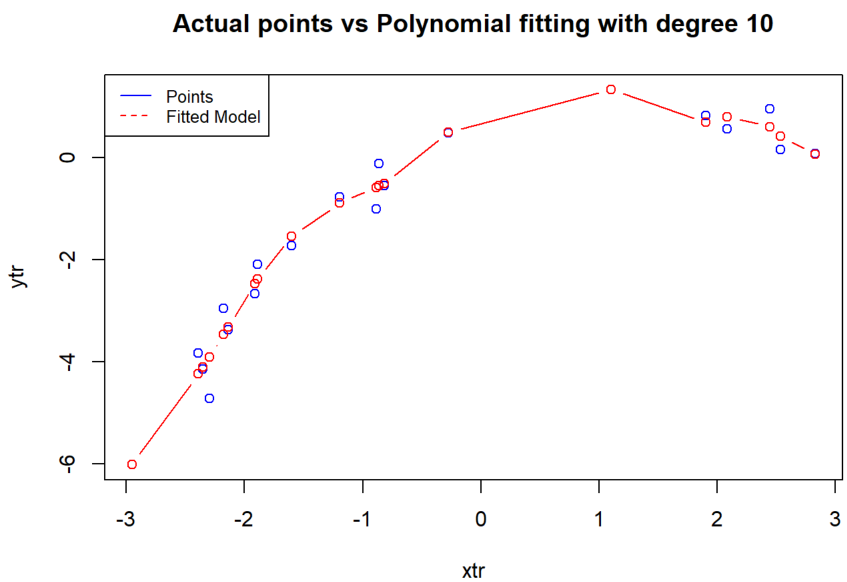

Actual points vs Polynomial fitting with degree 5
Points
Fitted Model
ytr
xtr
0
-2
-4
-6
-3
-2
-1
0
1
2
3


Actual points vs Polynomial fitting with degree 6
Points
Fitted Model
ytr
xtr
0
-2
-4
-6
-3
-2
-1
0
1
2
3

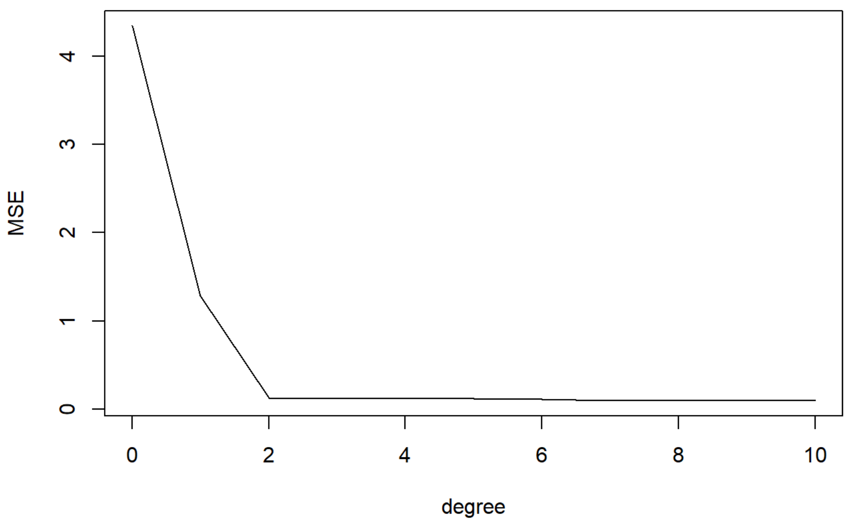

Actual points vs Polynomial fitting with degree 7
Points
Fitted Model
ytr
xtr
0
-2
-4
-6
-3
-2
-1
0
1
2
3


Actual points vs Polynomial fitting with degree 8
Points
Fitted Model
ytr
xtr
0
-2
-4
-6
-3
-2
-1
0
1
2
3

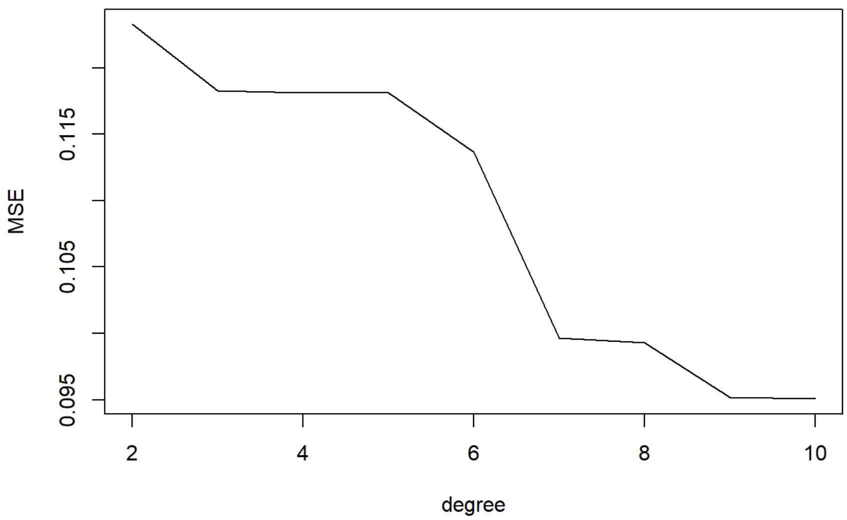

Actual points vs Polynomial fitting with degree 9
Points
Fitted Model
ytr
xtr
0
-2
-4
-6
-3
-2
-1
0
1
2
3




```
plot(0:10,mse,type='l', xlab='degree',ylab='MSE')
```



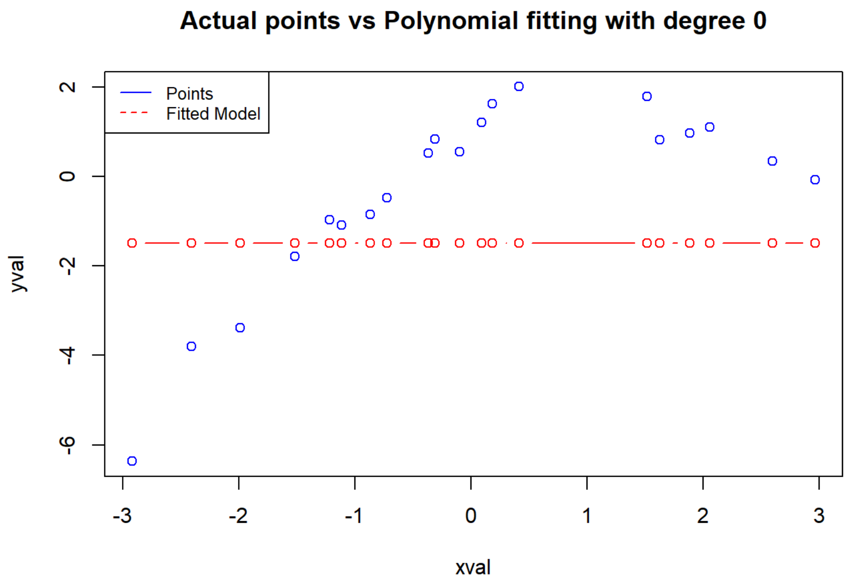

```
plot(2:10,mse[3:11],type='l', xlab='degree',ylab='MSE')
```



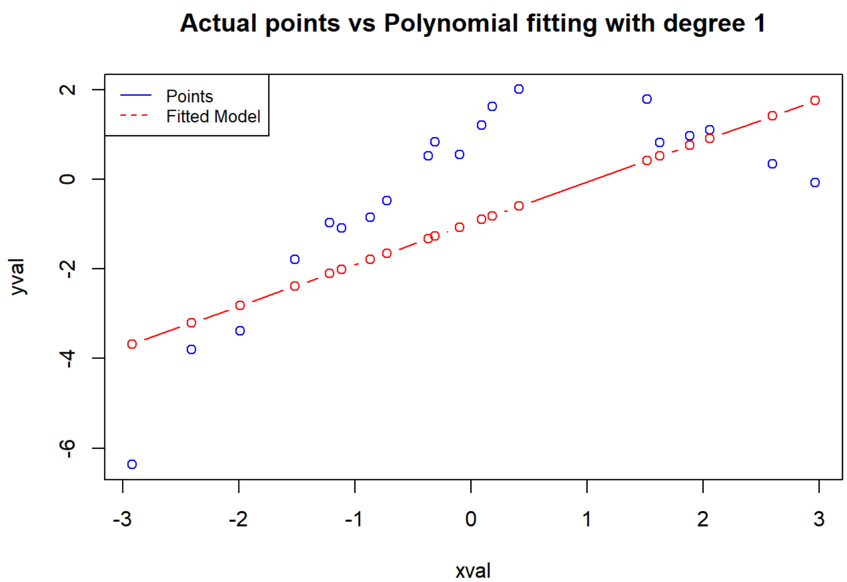
Task c

Use the 11 polynomials you found above and compute and report the MSE of the polynomials on the validation and test sets as well.

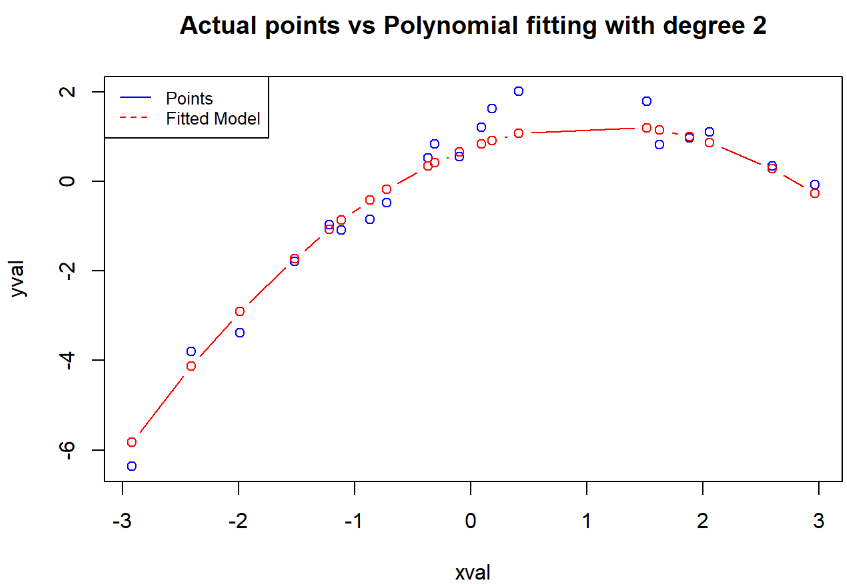
```
for (i in 0:10){
plot(xval,yval,col='blue' , main=paste("Actual points vs Polynomial fitting with degree",i) )
lines(xval, predict(models[[i+1]],valData), col='red', type='b')
legend("topleft", legend=c("Points", "Fitted Model"),
       col=c("blue", "Red"), lty=1:2, cex=0.8)
}
```



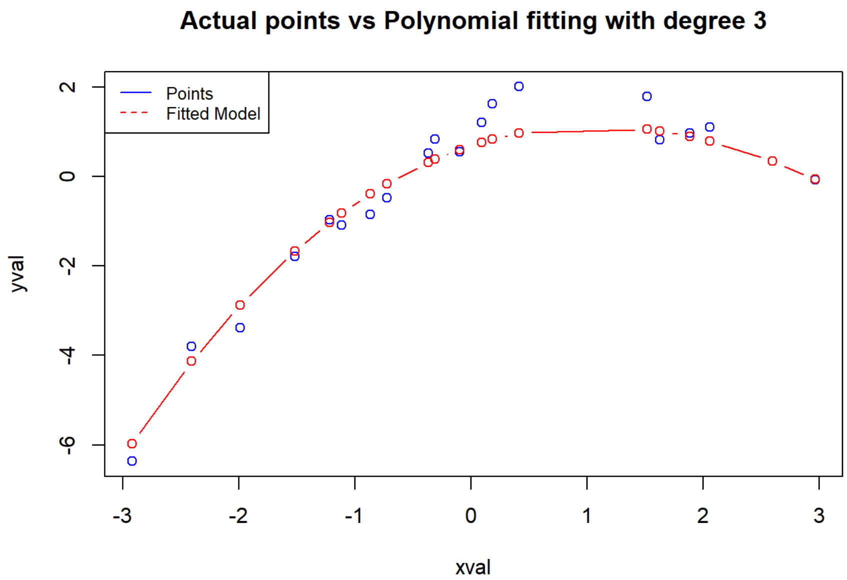
Actual points vs Polynomial fitting with degree 0
Points
Fitted Model
yval
xval
2
0
-2
-4
-6
-3
-2
-1
0
1
2
3

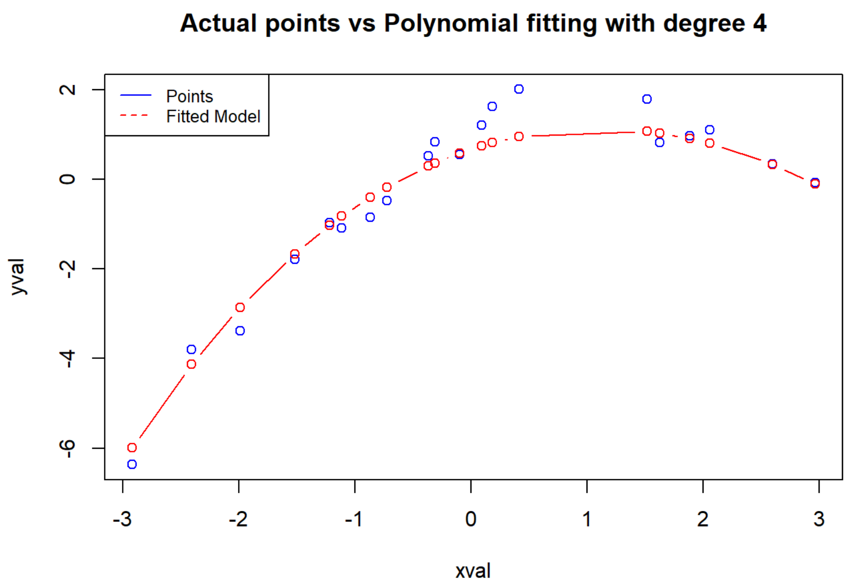


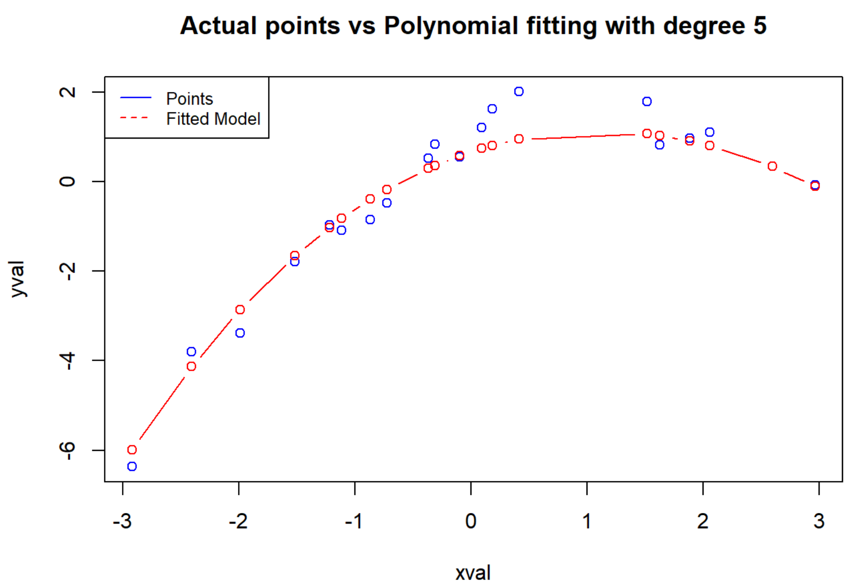
Actual points vs Polynomial fitting with degree 1
Points
Fitted Model
yval
xval
2
0
-2
-4
-6
-3
-2
-1
0
1
2
3



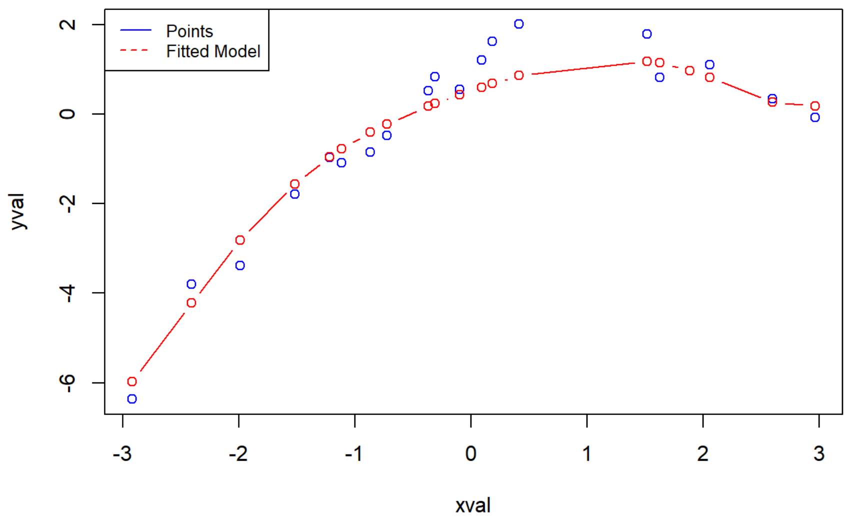
Actual points vs Polynomial fitting with degree 2
Points
Fitted Model
yval
xval
2
0
-2
-4
-6
-3
-2
-1
0
1
2
3


Actual points vs Polynomial fitting with degree 3
Points
Fitted Model
yval
xval
-6
-4
-2
0
2
-3
-2
-1
0
1
2
3

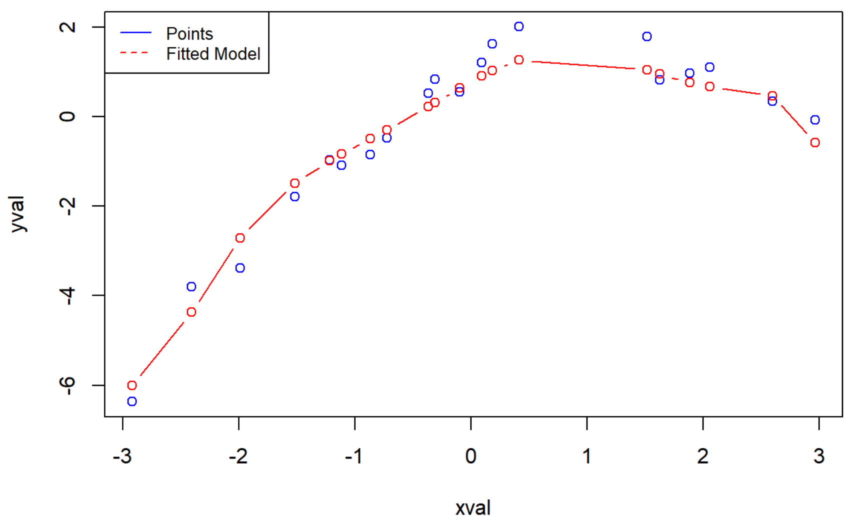

Actual points vs Polynomial fitting with degree 4
Points
Fitted Model
yval
xval
-6
-4
-2
0
2
-3
-2
-1
0
1
2
3


Actual points vs Polynomial fitting with degree 5
Points
Fitted Model
yval
xval
-6
-4
-2
0
2
-3
-2
-1
0
1
2
3

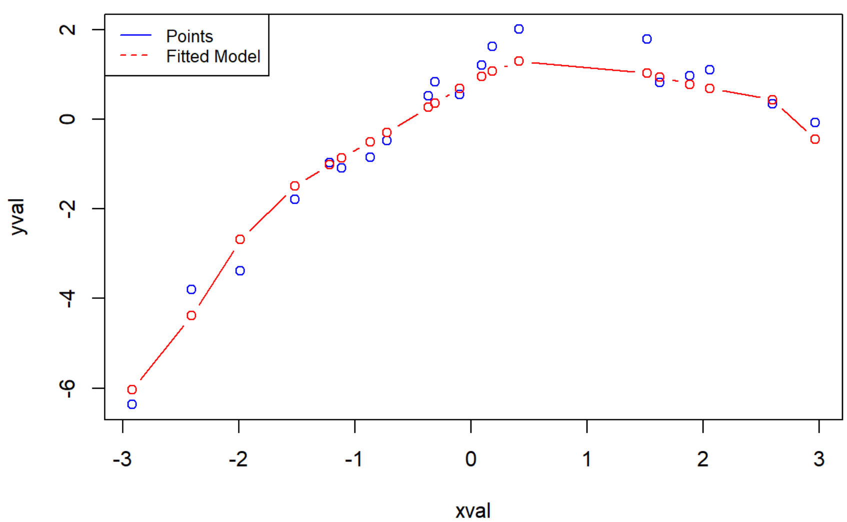
Actual points vs Polynomial fitting with degree 6

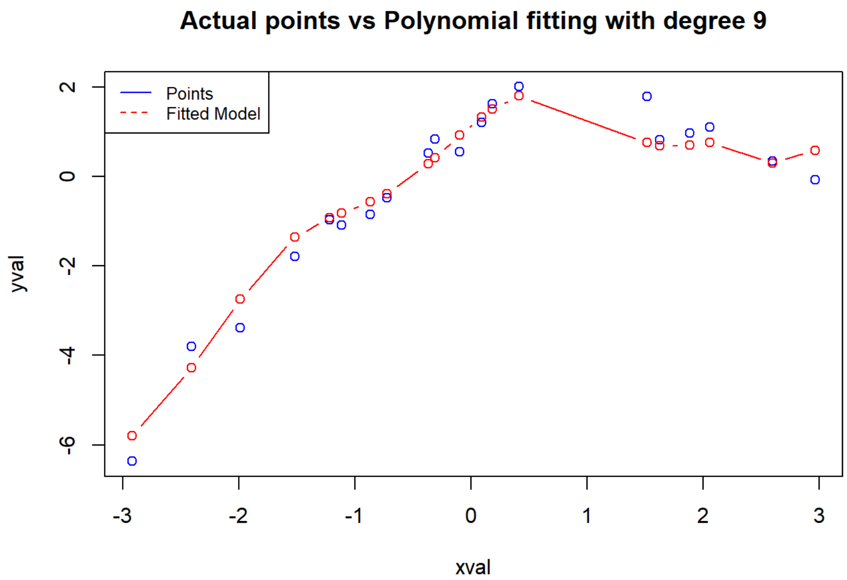


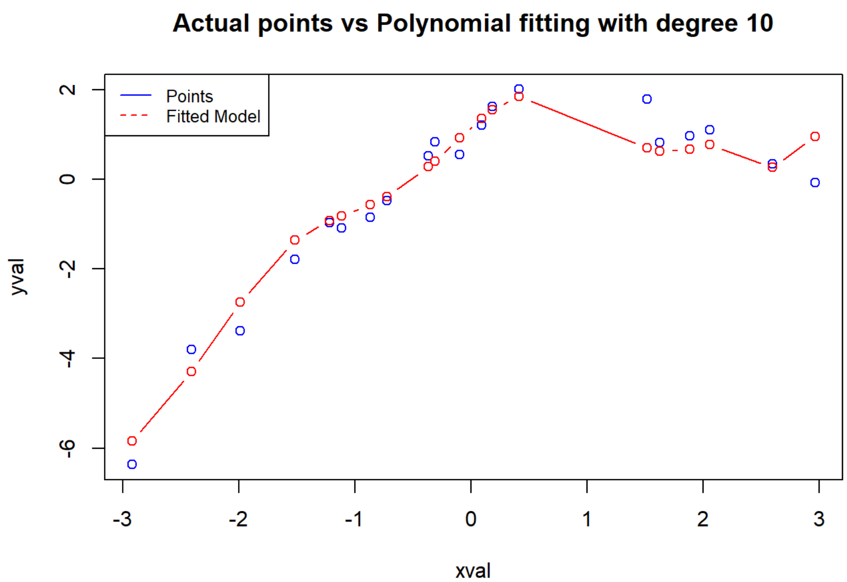
Actual points vs Polynomial fitting with degree 7



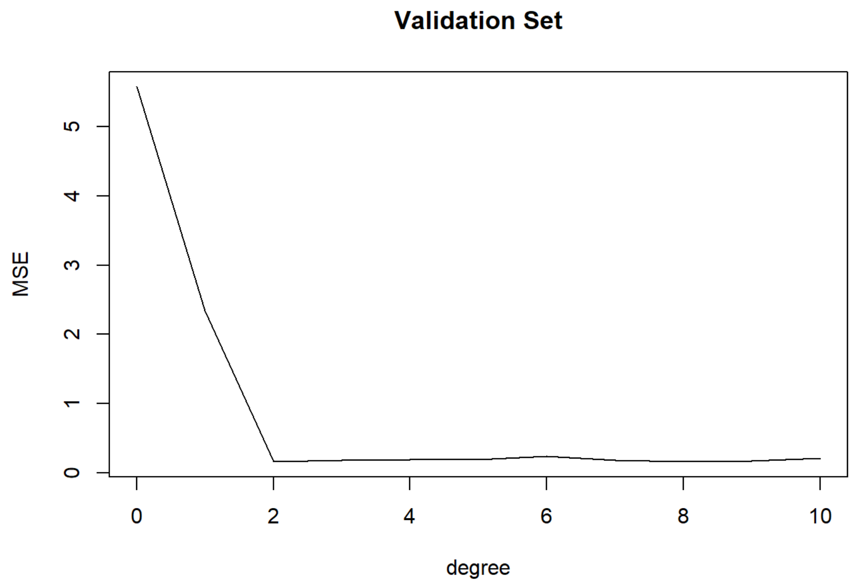
Actual points vs Polynomial fitting with degree 8



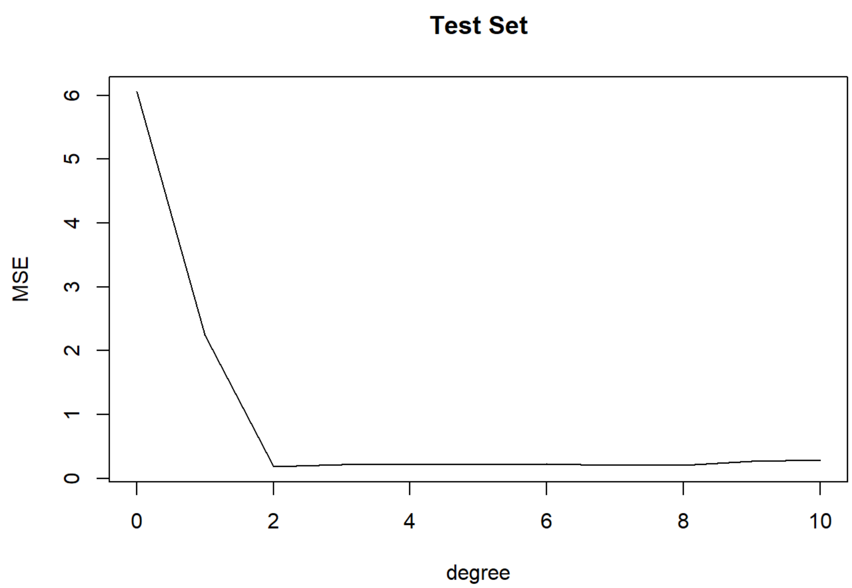





```
plot(0:10,mseVAL,type='l', xlab='degree',ylab='MSE',main="Validation Set")
```




```
plot(0:10,mseTesT,type='l', xlab='degree',ylab='MSE',main='Test Set')
```




```
degreeee=0:10
print(cbind(degreeee,mseTesT,mseVAL))
```

```
##       degreeee   mseTesT    mseVAL
##  [1,]        0 6.0542487 5.5732107
##  [2,]        1 2.2489892 2.3409374
##  [3,]        2 0.1831293 0.1627213
##  [4,]        3 0.2130996 0.1883265
##  [5,]        4 0.2135916 0.1904991
##  [6,]        5 0.2147782 0.1926776
##  [7,]        6 0.2307646 0.2383713
##  [8,]        7 0.2018030 0.1831591
##  [9,]        8 0.2027353 0.1692618
## [10,]        9 0.2665243 0.1736791
## [11,]       10 0.2916267 0.2110148
```

Task d

Now, choose the polynomial degree κ that has the smallest MSE on the validation set. How do the MSE on the training, validation, and test sets compare?

Train a new regressor of degree κ on the combined training and validation set and report the MSE on the test set.

```
ArgMin_mseVal=which.min(mseVAL)
paste("the polynomial degree with the smallest MSE on the validation set is ",ArgMin_mseVal-1)
```

```
## [1] "the polynomial degree with the smallest MSE on the validation set is  2"
```

```
NewData = rbind(TrainData, valData)
Train.fit <-lm(y~ poly(x,ArgMin_mseVal,raw=TRUE),TrainData)
summary(Train.fit)
```

```
##
## Call:
## lm(formula = y ~ poly(x, ArgMin_mseVal, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.94308 -0.19011  0.02827  0.21079  0.50382
##
## Coefficients:
##                                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)                          0.68986    0.18176   3.796  0.00159 **
## poly(x, ArgMin_mseVal, raw = TRUE)1  0.85514    0.14254   5.999 1.85e-05 ***
## poly(x, ArgMin_mseVal, raw = TRUE)2 -0.43169    0.03708 -11.643 3.18e-09 ***
## poly(x, ArgMin_mseVal, raw = TRUE)3  0.01962    0.02389   0.822  0.42342
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3845 on 16 degrees of freedom
## Multiple R-squared:  0.9728, Adjusted R-squared:  0.9677
## F-statistic: 190.6 on 3 and 16 DF,  p-value: 9.954e-13
```

```
y_pred_TR=predict(Train.fit,testData)
print(MSE1(testData$y,y_pred_TR))
```

```
##         1
## 0.2130996
```

Task e

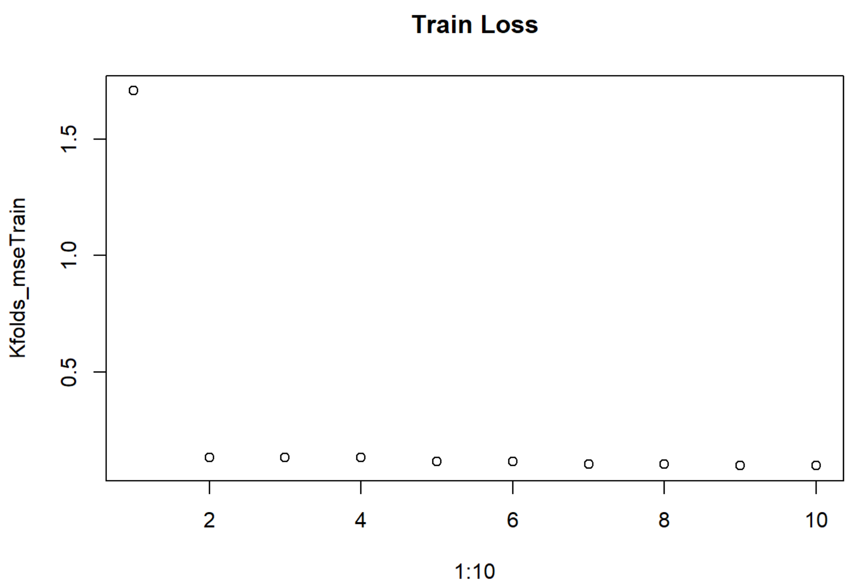
Now, instead of using simple training-validation set split combine the training and validation sets and use 10-fold cross validation to select a model based on the training+validation set (see James et al., Sec. 5.1.3). Report the cross-validation loss (Eq. (5.3) of James et al.) for different polynomial orders as well as the losses on the test set. How do the losses compare with the your previous results? Based on your cross-validation result, which polynomial degree should you choose?

```
library(caret)
```

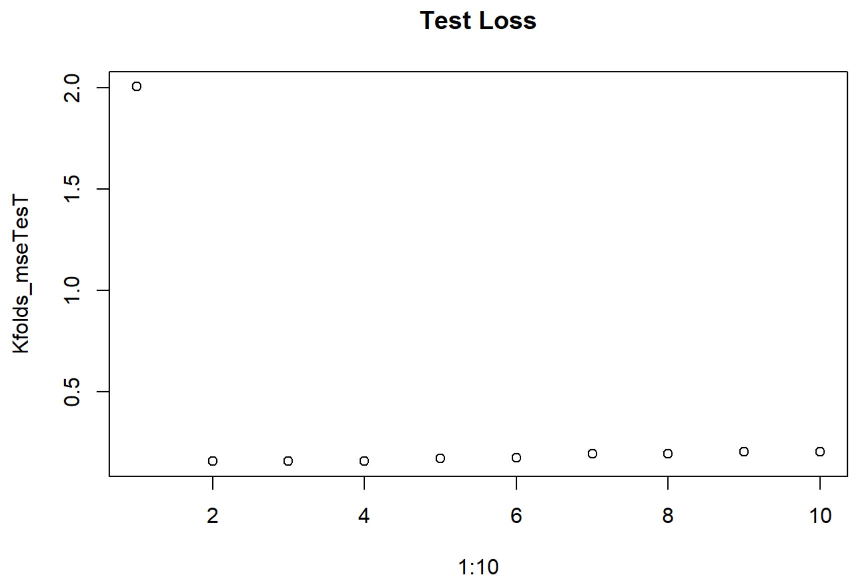
```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
# Define training control
set.seed(123)
Kfolds_mseTesT=rep(0,10)
Kfolds_mseTrain=rep(0,10)

modelsCV<- list()

train.control <- trainControl(method = "cv", number = 10)
modelsCV[[1]] <- train(y ~ poly(x,1,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[2]] <- train(y ~ poly(x,2,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[3]] <- train(y ~ poly(x,3,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[4]] <- train(y ~ poly(x,4,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[5]] <- train(y ~ poly(x,5,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[6]] <- train(y ~ poly(x,6,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[7]] <- train(y ~ poly(x,7,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[8]] <- train(y ~ poly(x,8,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[9]] <- train(y ~ poly(x,9,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)
modelsCV[[10]] <- train(y ~ poly(x,10,raw=TRUE),   method = "lm",data = NewData,   trControl = train.control)

for (k in 1:10) {
  y_pred_Te=predict(modelsCV[[k]],testData)
  Kfolds_mseTesT[k] = MSE1(yte,y_pred_Te)

  y_pred_Tr=predict(modelsCV[[k]],NewData)
  Kfolds_mseTrain[k] = MSE1(NewData$y,y_pred_Tr)


}
plot(1:10,Kfolds_mseTrain,main = "Train Loss")
```




```
plot(1:10,Kfolds_mseTesT,main = "Test Loss")
```




```
ArgMin = which.min(Kfolds_mseTesT)

paste("I should use k =",ArgMin)
```

```
## [1] "I should use k = 3"
```

Read about the bias-variance trade-off for k-fold cross-validation. (James et al., Section 5.1.4).

Problem 6

[20% points]

Learning objectives: bias and variance and model flexibility

Read Section 2.2 of James et al.

Consider the bias variance decomposition in the context of model selection.

Task a

Provide a sketch of typical (squared) bias, variance, training error, test error, and Bayes (or irreducible) error curves, on a single plot, as we go from less flexible statistical learning methods towards more flexible approaches. The x-axis should represent the amount of flexibility in the method, and the y-axis should represent the values for each curve. There should be five curves. Make sure to label each one.

Explain why each of the five curves has the shape displayed.

** I am taking a non linear f **

Kindly find the plot on the last page of the PDF (had some troubles uploading it to r-Markdown. Therefore I attached it at the end)

In general, and as statistical fundamentals, the train mse curve decreases with flexibility going up. Because, the more our curve is flexible, the more it will fit the training points and be closer to them.

However, for the test MSE this is not the case. Because, it is true that our curve fits better the training point with more flexibility, but in this case, at some point of flexibility, this curve will fit less the original function f (which doesn't concider the noise factor). Therefore, the test curve always go down at the beginning because we need a minimum of flexibility to fit the line, but at some point it will go up because we are fiting better the points but going far from f. Thus, it has a U shape.

The Bayes classifier produces the lowest possible test error rate. That's why i plotted it just under the test curve.

The bias curve, usually in theory decreases with flexibility.

The variance is always under the testMse and has a similar shape.

Task b

Test the bias variance tradeoff in practice with the data generation process described in Problem 5, at point $x = 0$. Generate 1000 new training sets of 20 pairs and train a polynomial regressor $g(x)$ on each of the training sets. For each of the 1000 training sets generate additionally a test data point at $x = 0$, i.e., $(0, y_0)$, where $y_0 = f(0) + \epsilon$ and ϵ is a random variable with zero mean and variance of $\sigma^2 = 0.4^2$.

According to Eq. (2.7) of James et al., the expected (expectation over your 1000 data sets, denoted by E_D) squared loss at $x = 0$, or $E_D[(y_0 - g(0))^2]$, can then be decomposed as a sum of irreducible error (here σ^2), the bias term $(E_D[g(0)] - f(0))^2$, and the variance term $E_D[(g(0) - E_D[g(0)])^2]$. Compute and plot these 4 terms (squared loss, irreducible error, bias term, variance term) at $x = 0$ as a function of polynomial degrees from 0 to 10 by using your 1000 data sets (i.e., you should end up with 4 curves). Check that the terms sum to squared loss for different polynomial degrees, i.e., verify Eq. (2.7) of James et al. Do the terms behave as you would expect from the discussion in the task a above?

```
get_sim_data = function(f, sample_size = 20) {
  x = runif(n = sample_size, min = 0, max = 1)
  y = f(x) + rnorm(n = sample_size, mean = 0, sd = 0.4)
  return(data.frame(x, y))
}


set.seed(1)
n_sims = 1000
n_models = 11
x0 = 0
predictions = matrix(0, nrow = n_sims, ncol = n_models)
sim_data = get_sim_data(f, sample_size = 100)
#plot(y ~ x, data = sim_data)

for (i in 1:n_sims) {

  sim_data = get_sim_data(f, sample_size = 100)

  fit_0 = lm(y ~ 1, data = sim_data)
  fit_1 = lm(y ~ poly(x, degree = 1), data = sim_data)
  fit_2 = lm(y ~ poly(x, degree = 2), data = sim_data)
  fit_3 = lm(y ~ poly(x, degree = 3), data = sim_data)
  fit_4 = lm(y ~ poly(x, degree = 4), data = sim_data)
  fit_5 = lm(y ~ poly(x, degree = 5), data = sim_data)
  fit_6 = lm(y ~ poly(x, degree = 6), data = sim_data)
  fit_7 = lm(y ~ poly(x, degree = 7), data = sim_data)
  fit_8 = lm(y ~ poly(x, degree = 8), data = sim_data)
  fit_9 = lm(y ~ poly(x, degree = 9), data = sim_data)
  fit_10 = lm(y ~ poly(x, degree = 10), data = sim_data)



  #lines(grid, predict(fit_1, newdata = data.frame(x = grid)), col = "red", lwd = 1)
  # lines(grid, predict(fit_2, newdata = data.frame(x = grid)), col = "blue", lwd = 1)
  # lines(grid, predict(fit_3, newdata = data.frame(x = grid)), col = "green", lwd = 1)
  #lines(grid, predict(fit_4, newdata = data.frame(x = grid)), col = "orange", lwd = 1)

  predictions[i, ] = c(
    predict(fit_0, newdata = data.frame(x = x0)),
    predict(fit_1, newdata = data.frame(x = x0)),
    predict(fit_2, newdata = data.frame(x = x0)),
    predict(fit_3, newdata = data.frame(x = x0)),
    predict(fit_4, newdata = data.frame(x = x0)),
    predict(fit_5, newdata = data.frame(x = x0)),
    predict(fit_6, newdata = data.frame(x = x0)),
    predict(fit_7, newdata = data.frame(x = x0)),
    predict(fit_8, newdata = data.frame(x = x0)),
    predict(fit_9, newdata = data.frame(x = x0)),
    predict(fit_10, newdata = data.frame(x = x0))
  )
}

#points(x0, f(x0), col = "black", pch = "x", cex = 2)
```

```
eps = rnorm(n = n_sims, mean = 0, sd = 0.3)
y0 = f(x0) + eps


get_bias = function(estimate, truth) {
  mean(estimate) - truth
}

get_mse = function(estimate, truth) {
  mean((estimate - truth) ^ 2)
}

bias = apply(predictions, 2, get_bias, f(x0))
variance = apply(predictions, 2, var)
mse = apply(predictions, 2, get_mse, y0)
```

```
bias ^ 2 + variance + var(eps)
```

```
##  [1] 0.2092248 0.1095828 0.1115440 0.1262197 0.1440481 0.1675969 0.2128235
##  [8] 0.2789553 0.4688126 0.8820765 3.2020813
```

```
mse
```

```
##  [1] 0.2104588 0.1084957 0.1100000 0.1256905 0.1451578 0.1699605 0.2197656
##  [8] 0.2773299 0.4629781 0.8675319 3.2242982
```

As we can see the vectors have approximatly the same values. Therefore the formula is verified.

Problem 7

[15% points]

Objective: properties of estimators

Lets continue with the linear regression and the data you created in Problem 5. So far, we have tried only to estimate the loss (with the purpose of choosing the best model). In machine learning it is also important to be able to estimate model parameters. For example, in the linear regression the parameters would be the regression coefficients, i.e., w_p in Problem 5 above.

Lets consider the simplest case of 0-degree polynomial ($k = 1$ in Problem 5) where you want to fit the constant line $\hat{y} = w_0$ to the data. In other words, lets first just compute mean of the data $\hat{y} = w_0 = \sum_{i \in S_{tr}} y_i / n_{tr}$.

Task a

What is the t-statistics (James et al., Sec. 3.1.2) and the corresponding 95% confidence intervals for the mean? Can you conclude that the true mean of the data is non-zero, by using the 20 data points in the training set of Problem 5 alone?

```
Deg0 = lm(y ~ 1,data = TRAIN)
summary(Deg0)
```

```
## 
## Call:
## lm(formula = y ~ 1, data = TRAIN)
## 
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -4.5353 -1.5791  0.5873  1.7191  2.8058 
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)   
## (Intercept)  -1.4844     0.4782  -3.104  0.00584 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 2.139 on 19 degrees of freedom
```

The p-value is bigger than 0.05. Therefore this model is bad and we can’t reject H0.

The corresponding IC for the mean is :

```
 confint(Deg0,Level=0.95)
```

```
##                 2.5 %    97.5 %
## (Intercept) -2.485251 -0.483539
```

I can’t conclude that the true mean of the data is non-zero because 0 is included in the IC calculated above.

Task b

Study the coefficients $w_0, w_1, \ldots$ of the polynomials of different degree (from Problem 5) with R (or, e.g., with corresponding Python SciPy functions) as in Sec. 3.6.2-3 of James et al. Are the estimated coefficients and confidence limits consistent with what you know about the data generating process, i.e., that the data has been generated by using a degree-2 polynomial $1 + x - x^2/2$ plus random noise?

```
for (i in 1:11) {
print(summary(models[[i]]))
#print(coef(models[[i]]))
print(confint(models[[i]]))
}
```

```
##
## Call:
## lm(formula = y ~ 1, data = TrainData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5353 -1.5791  0.5873  1.7191  2.8058
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -1.4844     0.4782  -3.104  0.00584 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.139 on 19 degrees of freedom
##
##                 2.5 %    97.5 %
## (Intercept) -2.485251 -0.483539
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.31080 -0.72893  0.04203  0.88759  1.71873
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -0.9837     0.2779  -3.540  0.00234 **
## poly(x, k, raw = TRUE)   0.9233     0.1410   6.546 3.75e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.195 on 18 degrees of freedom
## Multiple R-squared:  0.7042, Adjusted R-squared:  0.6878
## F-statistic: 42.85 on 1 and 18 DF,  p-value: 3.755e-06
##
##                             2.5 %     97.5 %
## (Intercept)            -1.5676344 -0.3998254
## poly(x, k, raw = TRUE)  0.6270098  1.2196777
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.93805 -0.19060  0.03377  0.24491  0.55240
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.75270    0.16329    4.61  0.00025 ***
## poly(x, k, raw = TRUE)1  0.96611    0.04508   21.43 9.64e-14 ***
## poly(x, k, raw = TRUE)2 -0.44126    0.03486  -12.66 4.43e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3808 on 17 degrees of freedom
## Multiple R-squared:  0.9716, Adjusted R-squared:  0.9683
## F-statistic: 291.1 on 2 and 17 DF,  p-value: 7.076e-14
##
##                             2.5 %     97.5 %
## (Intercept)              0.408190  1.0972172
## poly(x, k, raw = TRUE)1  0.871004  1.0612184
## poly(x, k, raw = TRUE)2 -0.514813 -0.3677146
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.94308 -0.19011  0.02827  0.21079  0.50382
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.68986    0.18176   3.796  0.00159 **
## poly(x, k, raw = TRUE)1  0.85514    0.14254   5.999 1.85e-05 ***
## poly(x, k, raw = TRUE)2 -0.43169    0.03708 -11.643 3.18e-09 ***
## poly(x, k, raw = TRUE)3  0.01962    0.02389   0.822  0.42342
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.3845 on 16 degrees of freedom
## Multiple R-squared:  0.9728, Adjusted R-squared:  0.9677
## F-statistic: 190.6 on 3 and 16 DF,  p-value: 9.954e-13
##
##                               2.5 %      97.5 %
## (Intercept)              0.30455775  1.07516697
## poly(x, k, raw = TRUE)1  0.55297033  1.15731390
## poly(x, k, raw = TRUE)2 -0.51029182 -0.35309393
## poly(x, k, raw = TRUE)3 -0.03101637  0.07026594
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.95092 -0.19802  0.03359  0.21341  0.49431
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.671720   0.230602   2.913  0.01071 *
## poly(x, k, raw = TRUE)1  0.858319   0.148987   5.761 3.76e-05 ***
## poly(x, k, raw = TRUE)2 -0.416222   0.120591  -3.452  0.00356 **
## poly(x, k, raw = TRUE)3  0.019144   0.024912   0.768  0.45415
## poly(x, k, raw = TRUE)4 -0.001948   0.014397  -0.135  0.89418
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3969 on 15 degrees of freedom
## Multiple R-squared:  0.9728, Adjusted R-squared:  0.9656
## F-statistic: 134.2 on 4 and 15 DF,  p-value: 1.504e-11
##
##                               2.5 %      97.5 %
## (Intercept)              0.18020461  1.16323591
## poly(x, k, raw = TRUE)1  0.54076030  1.17587723
## poly(x, k, raw = TRUE)2 -0.67325603 -0.15918705
## poly(x, k, raw = TRUE)3 -0.03395506  0.07224228
## poly(x, k, raw = TRUE)4 -0.03263498  0.02873939
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.94890 -0.19679  0.03725  0.21191  0.49404
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.6669215  0.2862601   2.330   0.0353 *
## poly(x, k, raw = TRUE)1  0.8501813  0.3091808   2.750   0.0157 *
## poly(x, k, raw = TRUE)2 -0.4137420  0.1491558  -2.774   0.0149 *
## poly(x, k, raw = TRUE)3  0.0226425  0.1180742   0.192   0.8507
## poly(x, k, raw = TRUE)4 -0.0022149  0.0173038  -0.128   0.9000
## poly(x, k, raw = TRUE)5 -0.0003259  0.0107339  -0.030   0.9762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4108 on 14 degrees of freedom
## Multiple R-squared:  0.9728, Adjusted R-squared:  0.9631
## F-statistic: 100.2 on 5 and 14 DF,  p-value: 1.887e-10
##
##                               2.5 %      97.5 %
## (Intercept)              0.05295468  1.28088835
## poly(x, k, raw = TRUE)1  0.18705446  1.51330814
## poly(x, k, raw = TRUE)2 -0.73364938 -0.09383461
## poly(x, k, raw = TRUE)3 -0.23060147  0.27588655
## poly(x, k, raw = TRUE)4 -0.03932790  0.03489819
## poly(x, k, raw = TRUE)5 -0.02334790  0.02269600
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.89441 -0.14691 -0.02083  0.22877  0.53335
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.521853   0.355054   1.470   0.1654
## poly(x, k, raw = TRUE)1  0.882917   0.318034   2.776   0.0157 *
## poly(x, k, raw = TRUE)2 -0.159925   0.386053  -0.414   0.6854
## poly(x, k, raw = TRUE)3  0.006251   0.122357   0.051   0.9600
```

```
## poly(x, k, raw = TRUE)4 -0.077582   0.106856  -0.726   0.4807
## poly(x, k, raw = TRUE)5  0.001584   0.011248   0.141   0.8902
## poly(x, k, raw = TRUE)6  0.005770   0.008069   0.715   0.4872
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4182 on 13 degrees of freedom
## Multiple R-squared:  0.9738, Adjusted R-squared:  0.9618
## F-statistic: 80.65 on 6 and 13 DF,  p-value: 1.579e-09
##
##                               2.5 %     97.5 %
## (Intercept)             -0.24519460 1.28890048
## poly(x, k, raw = TRUE)1  0.19584669 1.56998774
## poly(x, k, raw = TRUE)2 -0.99394121 0.67409208
## poly(x, k, raw = TRUE)3 -0.25808457 0.27058645
## poly(x, k, raw = TRUE)4 -0.30843108 0.15326741
## poly(x, k, raw = TRUE)5 -0.02271589 0.02588343
## poly(x, k, raw = TRUE)6 -0.01166148 0.02320120
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83463 -0.12110 -0.00684  0.20487  0.46799
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.778516   0.398368   1.954   0.0744 .
## poly(x, k, raw = TRUE)1  1.412224   0.511715   2.760   0.0173 *
## poly(x, k, raw = TRUE)2 -0.398454   0.418568  -0.952   0.3599
## poly(x, k, raw = TRUE)3 -0.473516   0.387860  -1.221   0.2456
## poly(x, k, raw = TRUE)4 -0.018065   0.113751  -0.159   0.8765
## poly(x, k, raw = TRUE)5  0.114494   0.087549   1.308   0.2155
## poly(x, k, raw = TRUE)6  0.001225   0.008605   0.142   0.8891
## poly(x, k, raw = TRUE)7 -0.007625   0.005866  -1.300   0.2180
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4075 on 12 degrees of freedom
## Multiple R-squared:  0.9771, Adjusted R-squared:  0.9637
## F-statistic: 73.04 on 7 and 12 DF,  p-value: 6.899e-09
##
##                               2.5 %      97.5 %
## (Intercept)             -0.08945352 1.646484742
## poly(x, k, raw = TRUE)1  0.29729370 2.527154927
## poly(x, k, raw = TRUE)2 -1.31043554 0.513526658
## poly(x, k, raw = TRUE)3 -1.31859117 0.371558717
## poly(x, k, raw = TRUE)4 -0.26590718 0.229777124
## poly(x, k, raw = TRUE)5 -0.07625895 0.305246290
## poly(x, k, raw = TRUE)6 -0.01752365 0.019973920
## poly(x, k, raw = TRUE)7 -0.02040502 0.005155375
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.83016 -0.13207 -0.00189  0.21860  0.48995
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.8317875  0.4968543   1.674   0.1223
## poly(x, k, raw = TRUE)1  1.4076561  0.5340556   2.636   0.0232 *
## poly(x, k, raw = TRUE)2 -0.5368818  0.8321176  -0.645   0.5320
## poly(x, k, raw = TRUE)3 -0.4721844  0.4044632  -1.167   0.2677
## poly(x, k, raw = TRUE)4  0.0576857  0.4054367   0.142   0.8894
## poly(x, k, raw = TRUE)5  0.1143166  0.0912881   1.252   0.2364
## poly(x, k, raw = TRUE)6 -0.0129169  0.0729347  -0.177   0.8626
## poly(x, k, raw = TRUE)7 -0.0075888  0.0061187  -1.240   0.2407
## poly(x, k, raw = TRUE)8  0.0008351  0.0042741   0.195   0.8487
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4249 on 11 degrees of freedom
## Multiple R-squared:  0.9771, Adjusted R-squared:  0.9605
## F-statistic: 58.79 on 8 and 11 DF,  p-value: 6.137e-08
##
##                                2.5 %      97.5 %
## (Intercept)             -0.261781540 1.925356526
## poly(x, k, raw = TRUE)1  0.232207659 2.583104628
```

```
## poly(x, k, raw = TRUE)2 -2.368360325 1.294596759
## poly(x, k, raw = TRUE)3 -1.362401834 0.418032980
## poly(x, k, raw = TRUE)4 -0.834674455 0.950045829
## poly(x, k, raw = TRUE)5 -0.086607059 0.315240265
## poly(x, k, raw = TRUE)6 -0.173445021 0.147611180
## poly(x, k, raw = TRUE)7 -0.021055923 0.005878237
## poly(x, k, raw = TRUE)8 -0.008572166 0.010242355
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.8123 -0.1931 -0.0078  0.1620  0.5243
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.139238   0.691903   1.647    0.131
## poly(x, k, raw = TRUE)1  2.217342   1.347497   1.646    0.131
## poly(x, k, raw = TRUE)2 -0.903352   1.020014  -0.886    0.397
## poly(x, k, raw = TRUE)3 -1.633964   1.814282  -0.901    0.389
## poly(x, k, raw = TRUE)4  0.177447   0.454381   0.391    0.704
## poly(x, k, raw = TRUE)5  0.591858   0.731973   0.809    0.438
## poly(x, k, raw = TRUE)6 -0.027978   0.078313  -0.357    0.728
## poly(x, k, raw = TRUE)7 -0.082481   0.114022  -0.723    0.486
## poly(x, k, raw = TRUE)8  0.001561   0.004525   0.345    0.737
## poly(x, k, raw = TRUE)9  0.003947   0.006000   0.658    0.526
##
## Residual standard error: 0.4363 on 10 degrees of freedom
## Multiple R-squared:  0.9781, Adjusted R-squared:  0.9584
## F-statistic: 49.61 on 9 and 10 DF,  p-value: 4.041e-07
##
##                                2.5 %     97.5 %
## (Intercept)             -0.402417462 2.68089304
## poly(x, k, raw = TRUE)1 -0.785067710 5.21975162
## poly(x, k, raw = TRUE)2 -3.176083587 1.36938015
## poly(x, k, raw = TRUE)3 -5.676436834 2.40850811
## poly(x, k, raw = TRUE)4 -0.834976930 1.18987176
## poly(x, k, raw = TRUE)5 -1.039079584 2.22279643
## poly(x, k, raw = TRUE)6 -0.202469894 0.14651399
## poly(x, k, raw = TRUE)7 -0.336538240 0.17157636
## poly(x, k, raw = TRUE)8 -0.008522239 0.01164354
## poly(x, k, raw = TRUE)9 -0.009421275 0.01731457
##
## Call:
## lm(formula = y ~ poly(x, k, raw = TRUE), data = TrainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81634 -0.18501 -0.00675  0.17954  0.51525
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1.1507244  0.7389833   1.557    0.154
## poly(x, k, raw = TRUE)1   2.3183717  1.7756792   1.306    0.224
## poly(x, k, raw = TRUE)2  -0.8120844  1.4433326  -0.563    0.587
## poly(x, k, raw = TRUE)3  -1.7696953  2.3889128  -0.741    0.478
## poly(x, k, raw = TRUE)4   0.0452713  1.4752035   0.031    0.976
## poly(x, k, raw = TRUE)5   0.6443082  0.9493759   0.679    0.514
## poly(x, k, raw = TRUE)6   0.0246866  0.5620622   0.044    0.966
## poly(x, k, raw = TRUE)7  -0.0903179  0.1458632  -0.619    0.551
## poly(x, k, raw = TRUE)8  -0.0064678  0.0848890  -0.076    0.941
## poly(x, k, raw = TRUE)9   0.0043520  0.0076331   0.570    0.583
## poly(x, k, raw = TRUE)10  0.0004146  0.0043766   0.095    0.927
##
## Residual standard error: 0.4596 on 9 degrees of freedom
## Multiple R-squared:  0.9781, Adjusted R-squared:  0.9538
## F-statistic: 40.23 on 10 and 9 DF,  p-value: 2.997e-06
##
##                                 2.5 %     97.5 %
## (Intercept)              -0.520971923 2.82242065
## poly(x, k, raw = TRUE)1  -1.698493657 6.33523702
## poly(x, k, raw = TRUE)2  -4.077129616 2.45296080
## poly(x, k, raw = TRUE)3  -7.173791441 3.63440082
## poly(x, k, raw = TRUE)4  -3.291870836 3.38241352
## poly(x, k, raw = TRUE)5  -1.503329338 2.79194581
## poly(x, k, raw = TRUE)6  -1.246786451 1.29615958
## poly(x, k, raw = TRUE)7  -0.420283333 0.23964761
## poly(x, k, raw = TRUE)8  -0.198500116 0.18556460
## poly(x, k, raw = TRUE)9  -0.012915359 0.02161932
## poly(x, k, raw = TRUE)10 -0.009486089 0.01031524
```

As we can see for the model with a degree = 2 , all variables (x, x^2) are significant with a verry small pvalue. While in sum other models, we can see that many variables (degrees of x) aren't significant. If you look on the model with a degree = 10, you can see that none of the variables are significant. They all have a p-val>0.05 and the slope isn't included in its IC.

This was expected, since the model with degree two should be the better one since the initial funtion f is a degree 2 function. ## Problem 8

[5% points]

Objectives: self-reflection, giving feedback of the course

Tasks

- Write a learning diary of the topics of lectures 1-4 and this exercise set.

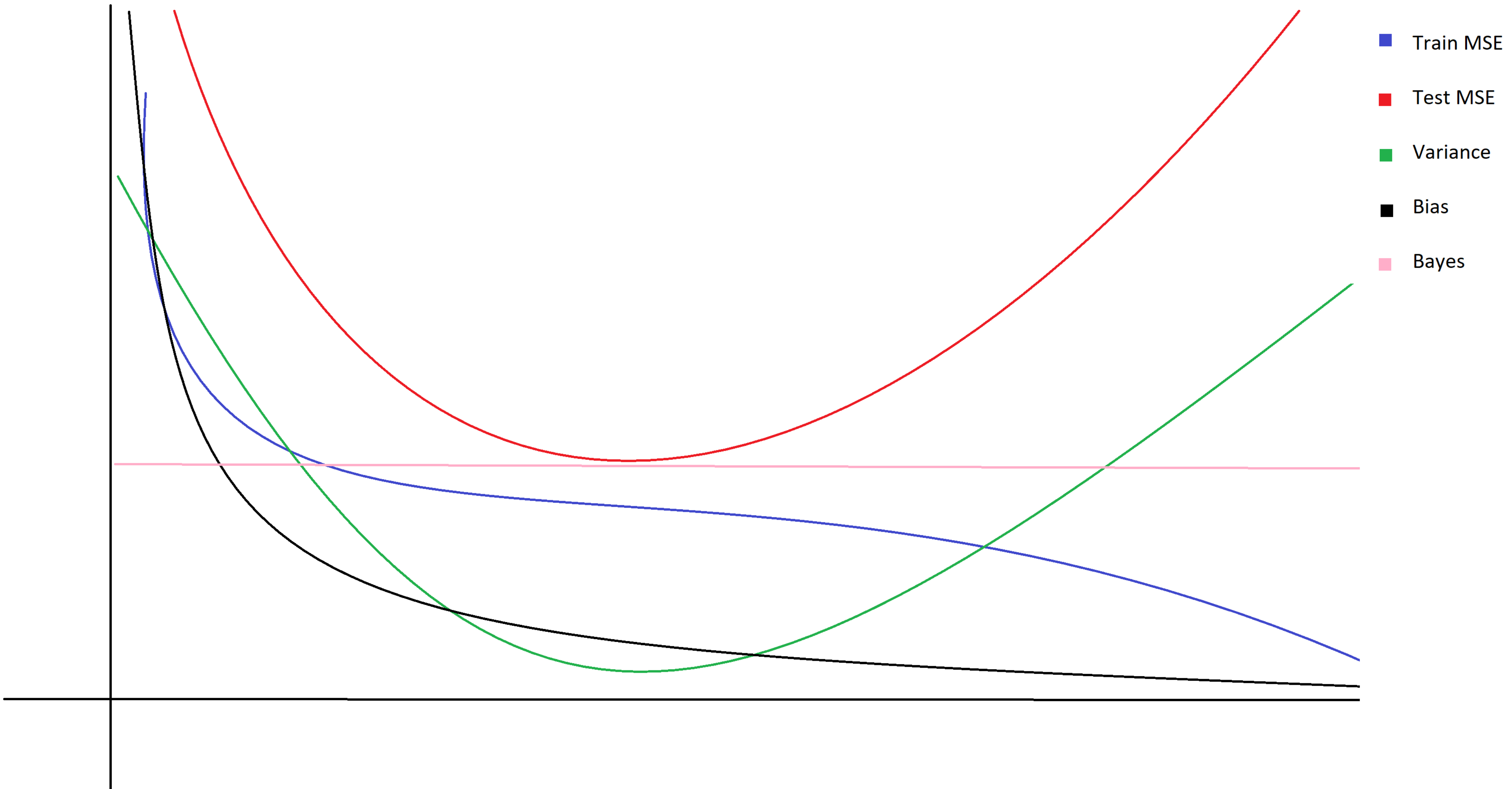
During these two weeks, I learned the fundamentals of Machine learning. I was introduced to the supervised, unsupervised learning, reinforcement and others. I also saw how these methods are applied in real life and how they are improving our life from different size.

In some other courses, I usually used some ML techniques without really knowing the theoretical part behind them. And that's what is different in this course.

In this exercise set, I leaned the most, the importance of applying the loss function on the test data and not the train data. I also learned how to escape from over and under fitting and what are the reason of getting such cases.Maybe before this set of exercises I thought that the more our curve is flexible the better results we will have. This was proven wrong in this Homework. I also learned how to choose between models based on polynomial fitting with different degrees of x and the importance of the valdiation set and how could it be used.

Otherwise I also saw again some fundamentals of probabilities.

The only thing that I regret is that this homework was so long and took with me so long in order to be accomplished. So maybe dividing the homework or making them lighter could be a good point to think about. I also learned how to apply what i learned in Rstudio and how to do a r-Markdown.


Train MSE
Test MSE
Variance
Bias
Bayes