Statistical Computing Practical

Table of contents

Chapter 1

Statistical Computing Practical

The link to my Github repo : https://github.com/Mishka312/StatHonPrac1

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become er
```

Chapter 2

Question 1

There are 2 methods to present the rows containing NA values. The first is to use the apply() function across all rows of the airquality dataframe.

```
# Method 1
airquality_na_ <- airquality[apply(is.na(airquality), 1, any), ]
head(airquality_na_)
```

```
   Ozone Solar.R Wind Temp Month Day
5     NA      NA 14.3   56     5   5
6     28      NA 14.9   66     5   6
10    NA     194  8.6   69     5  10
11     7      NA  6.9   74     5  11
25    NA      66 16.6   57     5  25
26    NA     266 14.9   58     5  26
```

The second method to get the same information is to use a for loop. The range in the for loop is 1 to 6 so that only the first 6 rows are displayed. The official way would be to have the following range '1:nrow(airquality)'.

```
# Method 2
for (i in 1:nrow(airquality)) {
  if (sum(is.na(airquality[i, ])) > 0) {
    print(airquality[i, ])
  }
}
```

```
  Ozone Solar.R Wind Temp Month Day
5    NA      NA 14.3   56     5   5
```

```
  Ozone Solar.R Wind Temp Month Day
6    28      NA 14.9   66     5   6
   Ozone Solar.R Wind Temp Month Day
10    NA     194  8.6   69     5  10
   Ozone Solar.R Wind Temp Month Day
11     7      NA  6.9   74     5  11
   Ozone Solar.R Wind Temp Month Day
25    NA      66 16.6   57     5  25
   Ozone Solar.R Wind Temp Month Day
26    NA     266 14.9   58     5  26
   Ozone Solar.R Wind Temp Month Day
27    NA      NA    8   57     5  27
   Ozone Solar.R Wind Temp Month Day
32    NA     286  8.6   78     6   1
   Ozone Solar.R Wind Temp Month Day
33    NA     287  9.7   74     6   2
   Ozone Solar.R Wind Temp Month Day
34    NA     242 16.1   67     6   3
   Ozone Solar.R Wind Temp Month Day
35    NA     186  9.2   84     6   4
   Ozone Solar.R Wind Temp Month Day
36    NA     220  8.6   85     6   5
   Ozone Solar.R Wind Temp Month Day
37    NA     264 14.3   79     6   6
   Ozone Solar.R Wind Temp Month Day
39    NA     273  6.9   87     6   8
   Ozone Solar.R Wind Temp Month Day
42    NA     259 10.9   93     6  11
   Ozone Solar.R Wind Temp Month Day
43    NA     250  9.2   92     6  12
   Ozone Solar.R Wind Temp Month Day
45    NA     332 13.8   80     6  14
   Ozone Solar.R Wind Temp Month Day
46    NA     322 11.5   79     6  15
   Ozone Solar.R Wind Temp Month Day
52    NA     150  6.3   77     6  21
   Ozone Solar.R Wind Temp Month Day
53    NA      59  1.7   76     6  22
   Ozone Solar.R Wind Temp Month Day
54    NA      91  4.6   76     6  23
   Ozone Solar.R Wind Temp Month Day
55    NA     250  6.3   76     6  24
   Ozone Solar.R Wind Temp Month Day
56    NA     135    8   75     6  25
   Ozone Solar.R Wind Temp Month Day
57    NA     127    8   78     6  26
```

```
   Ozone Solar.R Wind Temp Month Day
58    NA      47 10.3   73     6  27
   Ozone Solar.R Wind Temp Month Day
59    NA      98 11.5   80     6  28
   Ozone Solar.R Wind Temp Month Day
60    NA      31 14.9   77     6  29
   Ozone Solar.R Wind Temp Month Day
61    NA     138    8   83     6  30
   Ozone Solar.R Wind Temp Month Day
65    NA     101 10.9   84     7   4
   Ozone Solar.R Wind Temp Month Day
72    NA     139  8.6   82     7  11
   Ozone Solar.R Wind Temp Month Day
75    NA     291 14.9   91     7  14
   Ozone Solar.R Wind Temp Month Day
83    NA     258  9.7   81     7  22
   Ozone Solar.R Wind Temp Month Day
84    NA     295 11.5   82     7  23
   Ozone Solar.R Wind Temp Month Day
96    78      NA  6.9   86     8   4
   Ozone Solar.R Wind Temp Month Day
97    35      NA  7.4   85     8   5
   Ozone Solar.R Wind Temp Month Day
98    66      NA  4.6   87     8   6
    Ozone Solar.R Wind Temp Month Day
102    NA     222  8.6   92     8  10
    Ozone Solar.R Wind Temp Month Day
103    NA     137 11.5   86     8  11
    Ozone Solar.R Wind Temp Month Day
107    NA      64 11.5   79     8  15
    Ozone Solar.R Wind Temp Month Day
115    NA     255 12.6   75     8  23
    Ozone Solar.R Wind Temp Month Day
119    NA     153  5.7   88     8  27
    Ozone Solar.R Wind Temp Month Day
150    NA     145 13.2   77     9  27
```

Chapter 3

Question 2

To calculate the average temperature and ozone levels, the NA values need to first be removed. Thereafter, the means are calculated. The average temperature and ozone levels are 77.87 and 42. 13 respectively.

```
airquality %>%
  drop_na(Temp, Ozone) %>%
  summarise(mean_Temp = mean(Temp),
            mean_Ozone = mean(Ozone))
```

```
  mean_Temp mean_Ozone
1  77.87069   42.12931
```

Chapter 4

Question 3

A linear regression model was fitted to the data manually and the beta values retrieved. These were -17.58 and 3.92 respectively.

```
X <- as.matrix(cbind(rep(1, nrow(cars)), cars[, 1]))
Y <- as.matrix(cars[, 2])


beta <- solve(t(X) %*% X) %*% t(X) %*% Y
beta
```

```
          [,1]
[1,] -17.579095
[2,]   3.932409
```

Chapter 5

Question 4

A linear model was once again fitted to the data, this time with R's built in lm() function. The output of this proves that manually fitting a regression model yields the same beta coefficients.

```
cars
```

```
   speed dist
1      4    2
2      4   10
3      7    4
4      7   22
5      8   16
6      9   10
7     10   18
8     10   26
9     10   34
10    11   17
11    11   28
12    12   14
13    12   20
14    12   24
15    12   28
16    13   26
17    13   34
18    13   34
19    13   46
20    14   26
21    14   36
22    14   60
```

```
23    14   80
24    15   20
25    15   26
26    15   54
27    16   32
28    16   40
29    17   32
30    17   40
31    17   50
32    18   42
33    18   56
34    18   76
35    18   84
36    19   36
37    19   46
38    19   68
39    20   32
40    20   48
41    20   52
42    20   56
43    20   64
44    22   66
45    23   54
46    24   70
47    24   92
48    24   93
49    24  120
50    25   85
```

```
cars_model <- lm(dist ~ speed, cars)
#cars_model <- lm(speed ~ dist, cars)
summary(cars_model)
```

```

Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max 
-29.069  -9.525  -2.272   9.215  43.201 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -17.5791     6.7584  -2.601   0.0123 *  
speed         3.9324     0.4155   9.464 1.49e-12 ***
```

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

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,    Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```

The beta coefficients output by the lm() function are the same as those produced using the matrix calculations.

Chapter 6

Prac 3: Tidyverse

The tidyverse packages and the flights data sets need to be imported.

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become er
```

Chapter 7

Below are 2 methods to display the flights dataset

```
head(flights)
```

```
# A tibble: 6 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
1  2013     1     1      517            515         2      830            819
2  2013     1     1      533            529         4      850            830
3  2013     1     1      542            540         2      923            850
4  2013     1     1      544            545        -1     1004           1022
5  2013     1     1      554            600        -6      812            837
6  2013     1     1      554            558        -4      740            728
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
#alternatively glimpse(flights) could be used.
```

Chapter 8

Reducing base r code using dplyr

The 10 lines of code below makes use of dplyr functions. Doing the equivalent using base R functions required approximately 18 lines of code, almost double the dplyr code amount.

The code below yields the average and standard deviation of distance for each carrier. Furthermore, the mean distance is arranged ascendingly. This shows that carrier YV has the lowest average distance.

```
# A tibble: 16 x 3
   carrier mean_distance sd_distance
   <chr>           <dbl>       <dbl>
 1 YV               229          0
 2 9E               476.       334.
 3 EV               522.       294.
 4 US               536.       553.
 5 MQ               566.       223.
 6 FL               691.       142.
 7 OO               733         NA
 8 WN               942.       496.
 9 B6              1062.       681.
10 DL              1220.       644.
11 AA              1350.       626.
12 UA              1462.       778.
13 F9              1620          0
14 AS              2402          0
15 VX              2495.        98.2
16 HA              4983          0
```

```r
dist_tbl2 <- flights %>%
  filter(month == 1) %>%
  group_by(carrier) %>%
  summarise(mean_distance = mean(distance, na.rm = T),
            sd_distance = sd(distance, na.rm = T)) %>%
  arrange((mean_distance))
```

Chapter 9

Reasons for perculiar standard deviation entries

Some values were NA and 0. This can be explained by looking at the number and types of entries that this calculation is based on.

Carrier OO has a NA value for its standard deviation since it there is only one value. R, by default, considers the 0 sd of a single element.

```
flights %>%                         #### sd = NA
  filter(month == 1,
         carrier == "OO") %>%
  select(distance) %>%
  count()
```

```
# A tibble: 1 x 1
      n
  <int>
1     1
```

The other carriers (AS, F9, HA and YV) have more than one entry. However, all entries are of the same value. Hence, The standard deviation is 0.

```
flights %>%
  filter(month == 1,
         carrier == "AS") %>%  #### sd = 0
  select(distance) %>%
  distinct()
```

```
# A tibble: 1 x 1
  distance
     <dbl>
1     2402
```

```
flights %>%
  filter(month == 1,
         carrier == "F9") %>%
  select(distance) %>%
  distinct()
```

```
# A tibble: 1 x 1
  distance
     <dbl>
1     1620
```

```
flights %>%
  filter(month == 1,
         carrier == "HA") %>%
  select(distance) %>%
  distinct()
```

```
# A tibble: 1 x 1
  distance
     <dbl>
1     4983
```

```
flights %>%
  filter(month == 1,
         carrier == "YV") %>%
  select(distance) %>%
  distinct()
```

```
# A tibble: 1 x 1
  distance
     <dbl>
1      229
```

Chapter 10

Carriers along columns

To have carriers along the columns, the pivot_wider function is required. Although this may be helpful in some scenarios, there are an excessive amount of NA values embedded within the new, tranformed data frame.

```r
flights %>%
  pivot_wider(names_from = carrier, values_from = flight)
```

```
# A tibble: 336,776 x 33
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      517            515         2      830            819
 2  2013     1     1      533            529         4      850            830
 3  2013     1     1      542            540         2      923            850
 4  2013     1     1      544            545        -1     1004           1022
 5  2013     1     1      554            600        -6      812            837
 6  2013     1     1      554            558        -4      740            728
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# i 336,766 more rows
# i 25 more variables: arr_delay <dbl>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>,
#   time_hour <dttm>, UA <int>, AA <int>, B6 <int>, DL <int>, EV <int>,
#   MQ <int>, US <int>, WN <int>, VX <int>, FL <int>, AS <int>, `9E` <int>,
#   F9 <int>, HA <int>, YV <int>, OO <int>
```

Chapter 11

Proportion of flights

The proportion of flights that experienced a departure delay but not an arrival delay can be calculated by first filtering for flights where dep_delay > 0 and arr_delay < 1. Thereafter, this number of flights should be divided by the total number of flights. Approximately 10.5% of flights recover from their departure delay.

```
del_flight <- flights %>%
  filter(dep_delay > 0,
         arr_delay <= 0) %>%
  count()

prop_flights <- del_flight/nrow(flights)
prop_flights
```

```
          n
1 0.1052391
```

Chapter 12

Using the flights and airlines datasets together

To identify routes, one considers the origin and destination. Then count the total number of unique airlines (carriers) use this route, and filter for routes that support more than one carrier.

```
##origin and destination
airlines
```

```
# A tibble: 16 x 2
   carrier name
   <chr>   <chr>
 1 9E      Endeavor Air Inc.
 2 AA      American Airlines Inc.
 3 AS      Alaska Airlines Inc.
 4 B6      JetBlue Airways
 5 DL      Delta Air Lines Inc.
 6 EV      ExpressJet Airlines Inc.
 7 F9      Frontier Airlines Inc.
 8 FL      AirTran Airways Corporation
 9 HA      Hawaiian Airlines Inc.
10 MQ      Envoy Air
11 OO      SkyWest Airlines Inc.
12 UA      United Air Lines Inc.
13 US      US Airways Inc.
14 VX      Virgin America
15 WN      Southwest Airlines Co.
16 YV      Mesa Airlines Inc.
```

```
flights
```

```
# A tibble: 336,776 x 19
    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
 1  2013     1     1      517            515         2      830            819
 2  2013     1     1      533            529         4      850            830
 3  2013     1     1      542            540         2      923            850
 4  2013     1     1      544            545        -1     1004           1022
 5  2013     1     1      554            600        -6      812            837
 6  2013     1     1      554            558        -4      740            728
 7  2013     1     1      555            600        -5      913            854
 8  2013     1     1      557            600        -3      709            723
 9  2013     1     1      557            600        -3      838            846
10  2013     1     1      558            600        -2      753            745
# i 336,766 more rows
# i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
#   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
#   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
common_routes <- flights %>%
  group_by(origin, dest, carrier) %>%
  summarise(flight_count = n(), .groups = 'drop') %>%
  group_by(origin, dest) %>%
  summarise(unique_carriers = n_distinct(carrier), .groups = 'drop') %>%
  filter(unique_carriers > 1)

common_routes
```

```
# A tibble: 128 x 3
   origin dest  unique_carriers
   <chr>  <chr>           <int>
 1 EWR    ATL                 4
 2 EWR    AUS                 2
 3 EWR    BDL                 2
 4 EWR    BNA                 2
 5 EWR    BOS                 3
 6 EWR    BWI                 2
 7 EWR    CHS                 2
 8 EWR    CLE                 2
 9 EWR    CLT                 3
10 EWR    CVG                 2
# i 118 more rows
```

The carrier_routes output shows the average delay per carrier per route. the full airline names are also included by using the left_join() function.

```
carrier_routes <- flights %>%
  right_join(common_routes, by = c("origin", "dest")) %>%
  group_by(origin, dest, carrier) %>%
  summarise(mean_arr_del = mean(arr_delay, na.rm = T), .groups = "keep") %>%
  drop_na() %>%
  ungroup() %>%
  left_join(airlines, by = "carrier")

carrier_routes
```

```
# A tibble: 342 x 5
   origin dest  carrier mean_arr_del name
   <chr>  <chr> <chr>          <dbl> <chr>
 1 EWR    ATL   9E             -6.25 Endeavor Air Inc.
 2 EWR    ATL   DL             10.0  Delta Air Lines Inc.
 3 EWR    ATL   EV             19.5  ExpressJet Airlines Inc.
 4 EWR    ATL   UA             10.5  United Air Lines Inc.
 5 EWR    AUS   UA              4.28 United Air Lines Inc.
 6 EWR    AUS   WN            -11.2  Southwest Airlines Co.
 7 EWR    BDL   EV              6.78 ExpressJet Airlines Inc.
 8 EWR    BDL   UA             22.6  United Air Lines Inc.
 9 EWR    BNA   EV             17.7  ExpressJet Airlines Inc.
10 EWR    BNA   WN             -2.13 Southwest Airlines Co.
# i 332 more rows
```

Chapter 13

Identifying inconsistencies

By using the unique function, all unique entries in the data frame are displayed. This brings light to mispellings and other typographical and data-related errors. The output shows that there are 3 unique entries for gender. One, ‘femal’ is a typing error.
Disease status has 3 unique entries, however, 2 are equivalent. These are “healthy” and “Healthy”. They are considered unique since R is case sensitive.

```
erroneous_df %>%
  apply(2, unique)
```

```
$id
 [1] "id_1"  "id_2"  "id_3"  "id_4"  "id_5"  "id_6"  "id_7"  "id_8"  "id_9"
[10] "id_10" "id_11" "id_12" "id_13" "id_14" "id_15" "id_16" "id_17" "id_18"
[19] "id_19" "id_20" "id_21" "id_22" "id_23" "id_24" "id_25" "id_26" "id_27"
[28] "id_28" "id_29" "id_30" "id_31" "id_32" "id_33" "id_34" "id_35" "id_36"
[37] "id_37" "id_38" "id_39" "id_40" "id_41" "id_42" "id_43" "id_44" "id_45"
[46] "id_46" "id_47" "id_48" "id_49" "id_50"

$age
 [1] "50" "34" "70" "33" "22" "61" "69" "73" "62" "56" "71" "44" "45" "46" "24"
[16] "76" "47" "28" "48" "54" "27" "26" "38" "55" "36" "58" "72" "31" "51" "64"
[31] "60" "29" "42" "79"

$gender
[1] "male"   "female" "femal"

$height
 [1] "174.4" "197.7" "174.1" "194.5" NA      "180.4" "170.5" "157.4" "196.8"
[10] "165.1" "153.0" "197.4" "186.0" "157.1" "177.5" "179.3" "170.2" "182.4"
```

```
[19] "165.4" "161.0" "168.5" "199.2" "157.7" "154.6" "184.5" "181.0" "194.6"
[28] "183.6" "186.9" "176.1" "183.0" "191.1" "189.3" "199.0" "172.0" "165.6"
[37] "150.5" "159.2" "192.1" "161.6" "162.0" "153.8" "162.3" "186.6" "192.4"
[46] "174.9"

$weight
 [1] "69.4" "62.3" "55.6" "69.5" "78.6" "60.8" "72.2" "60.9" "75.1" "67.7"
[11] "82.5" "68.7" "67.8" "76.7" "87.0" "61.1" "70.6" "63.3" "81.5" "59.2"
[21] "93.2" "87.3" "83.4" "80.9" "68.6" "76.5" "93.7" "79.1" "92.0" "65.6"
[31] "85.4" "79.7" "74.1" "78.2" "95.7" "95.1" "63.7" "66.1" "99.3" "81.0"
[41] "96.9" "73.3" "70.3" "83.0" "57.6" "61.9" "98.1"

$blood_type
[1] "O"  "A"  "B"  "AB"

$disease_status
[1] "diseased" "healthy"  "Healthy"

$cholesterol
 [1] "228" "223" "213" "198" "166" "151" "195" "199" "189" "196" "221" "156"
[13] "185" "230" "234" "174" "236" "235" "180" "165" "220" "160" "153" "250"
[25] "184" "242" "212" "179" "224" "233" "181" "214" "248" "191" "162" "203"
[37] "173" "187" "164" "247"

$glucose
 [1] " 96" " 78" "101" "119" "103" " 91" " 86" NA    " 77" " 80" "115" " 85"
[13] " 88" "109" " 71" " 90" " 94" " 87" "113" " 93" " 97" "118" " 99" "108"
[25] " 89" "116" " 79" " 84" " 75" " 81" "106" " 82" " 76" "120"

$smoker
[1] "yes" "no"

$exercise
[1] "occasional" "regular"    "none"

$income
 [1] "84820" "81547" "22588" "72490" "74533" "25338" "41469" "57315" "63629"
[10] "88662" "62615" "56261" "58499" "82232" "77584" "77275" "38468" "54510"
[19] "91326" "78611" "31402" "29586" "21441" "58269" "84173" "88295" "37940"
[28] "43750" "69750" "92356" "82518" "91455" "68866" "51178" "68275" "27689"
[37] "35418" "81318" "62405" "86851" "25654" "47553" "74474" "51409" "22607"
[46] "55360" "96351" "21516" "41927" "55810"

$education
[1] "master"     "bachelor"   "PhD"        "highschool"
```

```
$region
[1] "North" "South" "West"  "East"

$marital_status
[1] "divorced" "single"   "married"  "widowed"
```

The following output shows that all columns are of equal length.

```r
erroneous_df %>%
  apply(2, length)
```

```
            id            age         gender         height         weight
            50             50             50             50             50
    blood_type disease_status    cholesterol        glucose         smoker
            50             50             50             50             50
      exercise         income      education         region marital_status
            50             50             50             50             50
```

The following output shows that all rows are of equal length.

```r
erroneous_df %>%
  apply(1, length)
```

```
 [1] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
[26] 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
```

The following code gives the indices of all the NA values.

```r
NA_positions <- which(is.na(erroneous_df), arr.ind = TRUE)
```

Chapter 14

Prac 2: Lowess algorithm

Chapter 15

Generating simulated data

```
set.seed(1)
x <- seq(1, 100, 1)

e <- rnorm(100, mean = 0, sd = 0.2)

y <- sin(x / 10) + e

f = 2/3

dists <- dist(x)
dists_m <- as.matrix(dists)

y_hat <- numeric(length(x))
```

```
#eval=FALSE
k <- ceiling(f*length(x))
w <- numeric(k)


for (i in 1:length(x)) {

  ifelse(k%%2==0,
         neighbs_dist <- dists_m[i, c(floor(i - (k)/2): ceiling(i + k- (k)/2))],
         neighbs_dist <- dists_m[i, c(floor(i - (k+1)/2): ceiling(i + k- (k+1)/2))])

  ifelse(k%%2==0,
         x_neighbs <- c(floor(i - (k)/2): ceiling(i + k- (k)/2)),
         x_neighbs <- c(floor(i - (k+1)/2): ceiling(i + k- (k+1)/2)))
```

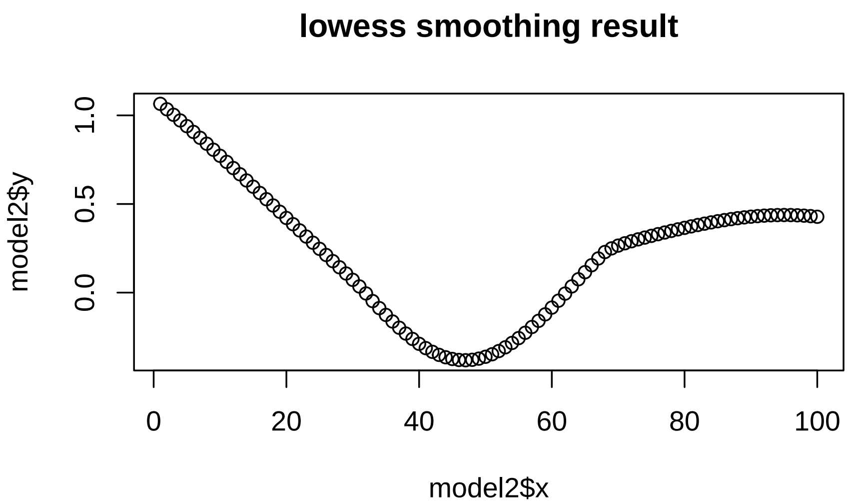
```
  y_neighbs <- x[c(x_neighbs)]

  x_and_dist <- cbind(x_neighbs, neighbs_dist)

  for (j in i:length(neighbs_dist)) {
    W <- diag(((abs(neighbs_dist)/max(neighbs_dist))**3)**3)

  }

  betas <- solve(t(x_neighbs) %*% W %*% (x_neighbs)) %*% t(x_neighbs) %*% W %*% (y_neig

  y_hat[i] <- betas[1] +betas[2] * x[i]


}
```

Chapter 16

Implementing the Lowess algorithm

```
#eval = FALSE

customLowess <- function(x, y, f) {
  k <- ceiling(f*length(x))

  distances <- numeric(length(x) -1)
  closes <- numeric(k)

  for (i in length(x)) {
    distances[i] <- x[i] - x[-i]
    sort(distances, decreasing = F)
    closest <- distances[1:k]
  }

}
```

```
model2 <- lowess(x, y, iter = 0)
plot(model2$x, model2$y, main = "lowess smoothing result")
```


lowess smoothing result
model2$y
1.0
0.5
0.0
0
20
40
60
80
100
model2$x