DAT320: Compulsory assignment 1

Group 2: Hanna Lye Moum, Sunniva Strumse, Ingebrigt Kjæreng

2024-09-27

Task 1

Importing necessary variables

```
library(ggplot2)
library(dplyr)
```

1 a)

Load the data set and convert categorical variables to factors and all other variables to an appropriate type. Print a summary of the variables in the data.frame.

Loading the data

```
df <- read.csv(file = "gapminder.csv", header = TRUE, sep = ",")
```

Converting all variables to either factors, numeric or integers

```
# Convert categorical variables to factors
df[, c("country", "continent")] <- lapply(df[,c("country", "continent")], as.factor)

# Convert other variables to appropriate data types
df[, c("lifeExp", "gdpPercap")] <- lapply(df[,c("lifeExp", "gdpPercap")], as.numeric)
df[, c("X", "year", "pop")]     <- lapply(df[,c("X", "year", "pop")],     as.integer)
```

Summary of the variables in the data.frame:

```
#Summary
summary(df)
```

```
##        X                  country        continent        year     
##  Min.   :   1.0   Afghanistan:  12   Africa  :624   Min.   :1952  
##  1st Qu.: 426.8   Albania    :  12   Americas:300   1st Qu.:1966  
##  Median : 852.5   Algeria    :  12   Asia    :396   Median :1980  
##  Mean   : 852.5   Angola     :  12   Europe  :360   Mean   :1980  
##  3rd Qu.:1278.2   Argentina  :  12   Oceania : 24   3rd Qu.:1993  
##  Max.   :1704.0   Australia  :  12                  Max.   :2007  
##                   (Other)    :1632                                
##     lifeExp          pop             gdpPercap       
##  Min.   :23.60   Min.   :6.001e+04   Min.   :   241.2  
##  1st Qu.:48.20   1st Qu.:2.794e+06   1st Qu.:  1202.1  
##  Median :60.71   Median :7.024e+06   Median :  3531.8  
##  Mean   :59.47   Mean   :2.960e+07   Mean   :  7215.3  
##  3rd Qu.:70.85   3rd Qu.:1.959e+07   3rd Qu.:  9325.5
```

```
##  Max.   :82.60   Max.   :1.319e+09   Max.   :113523.1
##
```

```
# All countries
levels(df$country)
```

```
##   [1] "Afghanistan"              "Albania"
##   [3] "Algeria"                  "Angola"
##   [5] "Argentina"                "Australia"
##   [7] "Austria"                  "Bahrain"
##   [9] "Bangladesh"               "Belgium"
##  [11] "Benin"                    "Bolivia"
##  [13] "Bosnia and Herzegovina"   "Botswana"
##  [15] "Brazil"                   "Bulgaria"
##  [17] "Burkina Faso"             "Burundi"
##  [19] "Cambodia"                 "Cameroon"
##  [21] "Canada"                   "Central African Republic"
##  [23] "Chad"                     "Chile"
##  [25] "China"                    "Colombia"
##  [27] "Comoros"                  "Congo, Dem. Rep."
##  [29] "Congo, Rep."              "Costa Rica"
##  [31] "Cote d'Ivoire"            "Croatia"
##  [33] "Cuba"                     "Czech Republic"
##  [35] "Denmark"                  "Djibouti"
##  [37] "Dominican Republic"       "Ecuador"
##  [39] "Egypt"                    "El Salvador"
##  [41] "Equatorial Guinea"        "Eritrea"
##  [43] "Ethiopia"                 "Finland"
##  [45] "France"                   "Gabon"
##  [47] "Gambia"                   "Germany"
##  [49] "Ghana"                    "Greece"
##  [51] "Guatemala"                "Guinea"
##  [53] "Guinea-Bissau"            "Haiti"
##  [55] "Honduras"                 "Hong Kong, China"
##  [57] "Hungary"                  "Iceland"
##  [59] "India"                    "Indonesia"
##  [61] "Iran"                     "Iraq"
##  [63] "Ireland"                  "Israel"
##  [65] "Italy"                    "Jamaica"
##  [67] "Japan"                    "Jordan"
##  [69] "Kenya"                    "Korea, Dem. Rep."
##  [71] "Korea, Rep."              "Kuwait"
##  [73] "Lebanon"                  "Lesotho"
##  [75] "Liberia"                  "Libya"
##  [77] "Madagascar"               "Malawi"
##  [79] "Malaysia"                 "Mali"
##  [81] "Mauritania"               "Mauritius"
##  [83] "Mexico"                   "Mongolia"
##  [85] "Montenegro"               "Morocco"
##  [87] "Mozambique"               "Myanmar"
##  [89] "Namibia"                  "Nepal"
##  [91] "Netherlands"              "New Zealand"
##  [93] "Nicaragua"                "Niger"
##  [95] "Nigeria"                  "Norway"
##  [97] "Oman"                     "Pakistan"
```

```
##  [99] "Panama"                    "Paraguay"
## [101] "Peru"                      "Philippines"
## [103] "Poland"                    "Portugal"
## [105] "Puerto Rico"               "Reunion"
## [107] "Romania"                   "Rwanda"
## [109] "Sao Tome and Principe"     "Saudi Arabia"
## [111] "Senegal"                   "Serbia"
## [113] "Sierra Leone"              "Singapore"
## [115] "Slovak Republic"           "Slovenia"
## [117] "Somalia"                   "South Africa"
## [119] "Spain"                     "Sri Lanka"
## [121] "Sudan"                     "Swaziland"
## [123] "Sweden"                    "Switzerland"
## [125] "Syria"                     "Taiwan"
## [127] "Tanzania"                  "Thailand"
## [129] "Togo"                      "Trinidad and Tobago"
## [131] "Tunisia"                   "Turkey"
## [133] "Uganda"                    "United Kingdom"
## [135] "United States"             "Uruguay"
## [137] "Venezuela"                 "Vietnam"
## [139] "West Bank and Gaza"        "Yemen, Rep."
## [141] "Zambia"                    "Zimbabwe"
```

```
#All continents
levels(df$continent)
```

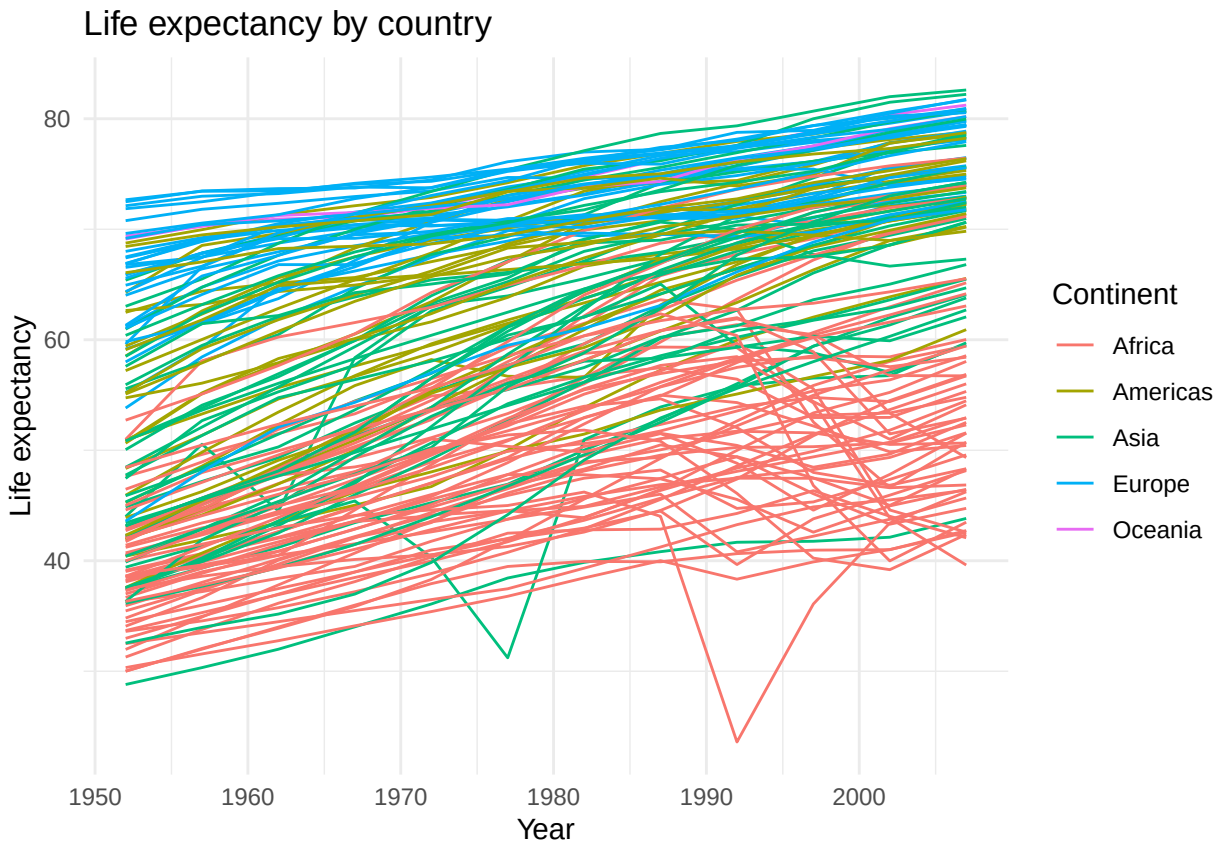
```
## [1] "Africa"   "Americas" "Asia"     "Europe"   "Oceania"
```

1 b)

Plot the time series of each country (Year versus lifeExp), colored by continent.

Plotting using ggplot2:

```
ggplot(data=df,
       mapping=aes(x     = year,
                   y     = lifeExp,
                   color = continent,
                   group = country)) +
       geom_line() +
       labs(title    = "Life expectancy by country",
              x     = "Year",
              y     = "Life expectancy",
              color = "Continent"
       ) +
       theme_minimal()
```



1 c)

Give a summary of the lifeExp for each continent and for each year. Compute minimum, median, mean, maximum, and standard deviations. Display this as a data.frame

---

Group data by continent and year, and summarise across life expectancy:

```
grp_df <- df %>% group_by(continent, year) %>%

            # summarise on grouped data across life expectancy
            summarise(across(lifeExp,
                             list(min    = min,
                                  median = median,
                                  mean   = mean,
                                  max    = max,
                                  sd     = sd)
                             )
                      ) %>%
            # convert tbl to data frame
            as.data.frame()
```

Display grouped data:

```
print(grp_df)
```

```
##    continent year lifeExp_min lifeExp_median lifeExp_mean lifeExp_max
## 1     Africa 1952      30.000        38.8330     39.13550      52.724
```

```
## 2     Africa 1957      31.570        40.5925     41.26635      58.089
## 3     Africa 1962      32.767        42.6305     43.31944      60.246
## 4     Africa 1967      34.113        44.6985     45.33454      61.557
## 5     Africa 1972      35.400        47.0315     47.45094      64.274
## 6     Africa 1977      36.788        49.2725     49.58042      67.064
## 7     Africa 1982      38.445        50.7560     51.59287      69.885
## 8     Africa 1987      39.906        51.6395     53.34479      71.913
## 9     Africa 1992      23.599        52.4290     53.62958      73.615
## 10    Africa 1997      36.087        52.7590     53.59827      74.772
## 11    Africa 2002      39.193        51.2355     53.32523      75.744
## 12    Africa 2007      39.613        52.9265     54.80604      76.442
## 13  Americas 1952      37.579        54.7450     53.27984      68.750
## 14  Americas 1957      40.696        56.0740     55.96028      69.960
## 15  Americas 1962      43.428        58.2990     58.39876      71.300
## 16  Americas 1967      45.032        60.5230     60.41092      72.130
## 17  Americas 1972      46.714        63.4410     62.39492      72.880
## 18  Americas 1977      49.923        66.3530     64.39156      74.210
## 19  Americas 1982      51.461        67.4050     66.22884      75.760
## 20  Americas 1987      53.636        69.4980     68.09072      76.860
## 21  Americas 1992      55.089        69.8620     69.56836      77.950
## 22  Americas 1997      56.671        72.1460     71.15048      78.610
## 23  Americas 2002      58.137        72.0470     72.42204      79.770
## 24  Americas 2007      60.916        72.8990     73.60812      80.653
## 25       Asia 1952      28.801        44.8690     46.31439      65.390
## 26       Asia 1957      30.332        48.2840     49.31854      67.840
## 27       Asia 1962      31.997        49.3250     51.56322      69.390
## 28       Asia 1967      34.020        53.6550     54.66364      71.430
## 29       Asia 1972      36.088        56.9500     57.31927      73.420
## 30       Asia 1977      31.220        60.7650     59.61056      75.380
## 31       Asia 1982      39.854        63.7390     62.61794      77.110
## 32       Asia 1987      40.822        66.2950     64.85118      78.670
## 33       Asia 1992      41.674        68.6900     66.53721      79.360
## 34       Asia 1997      41.763        70.2650     68.02052      80.690
## 35       Asia 2002      42.129        71.0280     69.23388      82.000
## 36       Asia 2007      43.828        72.3960     70.72848      82.603
## 37     Europe 1952      43.585        65.9000     64.40850      72.670
## 38     Europe 1957      48.079        67.6500     66.70307      73.470
## 39     Europe 1962      52.098        69.5250     68.53923      73.680
## 40     Europe 1967      54.336        70.6100     69.73760      74.160
## 41     Europe 1972      57.005        70.8850     70.77503      74.720
## 42     Europe 1977      59.507        72.3350     71.93777      76.110
## 43     Europe 1982      61.036        73.4900     72.80640      76.990
## 44     Europe 1987      63.108        74.8150     73.64217      77.410
## 45     Europe 1992      66.146        75.4510     74.44010      78.770
## 46     Europe 1997      68.835        76.1160     75.50517      79.390
## 47     Europe 2002      70.845        77.5365     76.70060      80.620
## 48     Europe 2007      71.777        78.6085     77.64860      81.757
## 49    Oceania 1952      69.120        69.2550     69.25500      69.390
## 50    Oceania 1957      70.260        70.2950     70.29500      70.330
## 51    Oceania 1962      70.930        71.0850     71.08500      71.240
## 52    Oceania 1967      71.100        71.3100     71.31000      71.520
## 53    Oceania 1972      71.890        71.9100     71.91000      71.930
## 54    Oceania 1977      72.220        72.8550     72.85500      73.490
## 55    Oceania 1982      73.840        74.2900     74.29000      74.740
```

```
## 56   Oceania 1987      74.320        75.3200     75.32000      76.320
## 57   Oceania 1992      76.330        76.9450     76.94500      77.560
## 58   Oceania 1997      77.550        78.1900     78.19000      78.830
## 59   Oceania 2002      79.110        79.7400     79.74000      80.370
## 60   Oceania 2007      80.204        80.7195     80.71950      81.235
##     lifeExp_sd
## 1   5.15158143
## 2   5.62012285
## 3   5.87536393
## 4   6.08267263
## 5   6.41625832
## 6   6.80819741
## 7   7.37594009
## 8   7.86408911
## 9   9.46107099
## 10  9.10338658
## 11  9.58649585
## 12  9.63078067
## 13  9.32608188
## 14  9.03319228
## 15  8.50354374
## 16  7.90917104
## 17  7.32301680
## 18  7.06949562
## 19  6.72083382
## 20  5.80192884
## 21  5.16710381
## 22  4.88758390
## 23  4.79970550
## 24  4.44094763
## 25  9.29175070
## 26  9.63542862
## 27  9.82063194
## 28  9.65096458
## 29  9.72270004
## 30 10.02219698
## 31  8.53522141
## 32  8.20379188
## 33  8.07554897
## 34  8.09117061
## 35  8.37459539
## 36  7.96372447
## 37  6.36108825
## 38  5.29580539
## 39  4.30249956
## 40  3.79972850
## 41  3.24057637
## 42  3.12102998
## 43  3.21826030
## 44  3.16968034
## 45  3.20978109
## 46  3.10467655
## 47  2.92217958
## 48  2.97981266
```

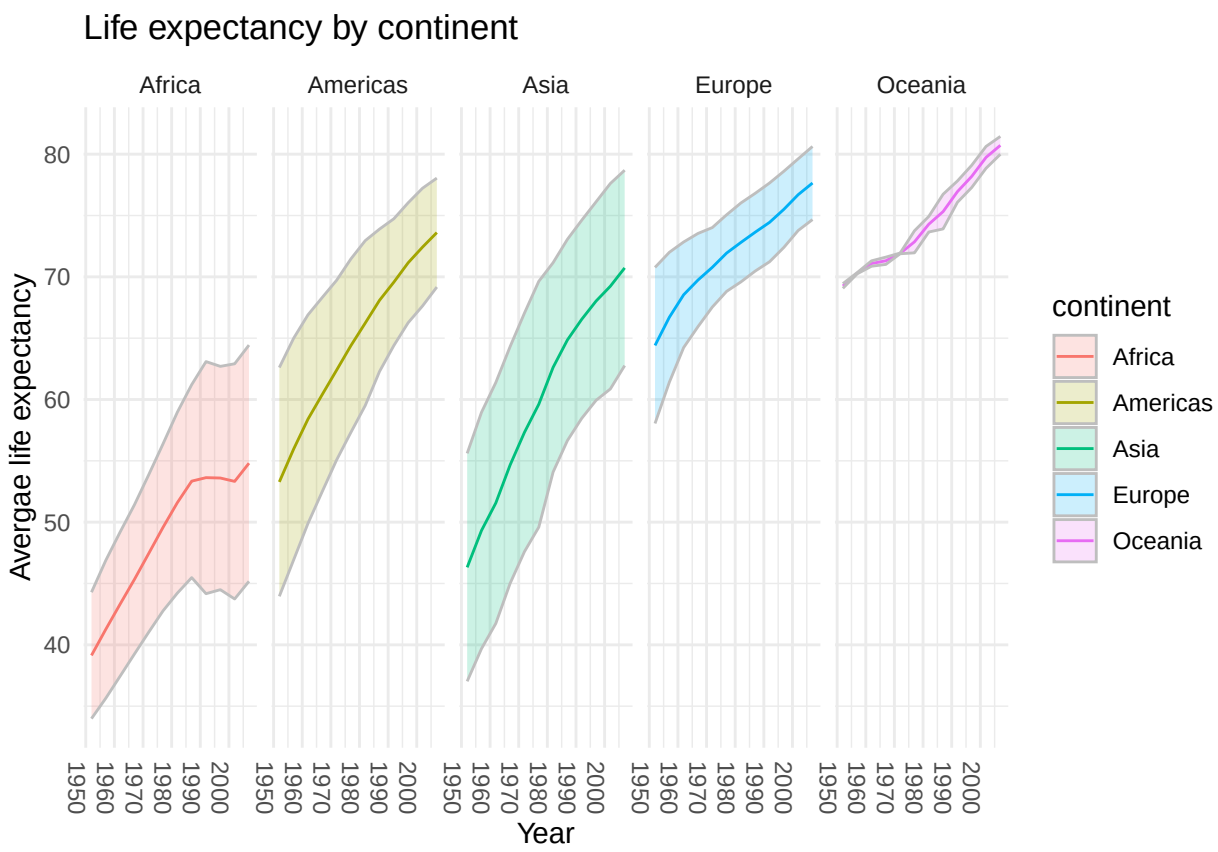
```
## 49  0.19091883
## 50  0.04949747
## 51  0.21920310
## 52  0.29698485
## 53  0.02828427
## 54  0.89802561
## 55  0.63639610
## 56  1.41421356
## 57  0.86974134
## 58  0.90509668
## 59  0.89095454
## 60  0.72902709
```

We observe that the life expectancy in general grows larger for each continent over the years. We will visualize the data for better interpretability in the next exercise.

1 d)

For each continent, plot the average lifeExp across all countries per year (Year versus lifeExp) as a line plot. Add errorbars or ribbons indicating ± 1 standard deviation.

```r
ggplot(data=grp_df,
       mapping=aes(x=year,
                   y=lifeExp_mean,
                   color=continent,
                   group=continent)) +
      # create lineplot
      geom_line() +
      # add error bars
      geom_ribbon(aes(ymin = lifeExp_mean - lifeExp_sd,
                      ymax = lifeExp_mean + lifeExp_sd,
                      fill = continent),
                  alpha=0.2,   # transparency of sd
                  color='grey' # color of outline border
                  ) +
      labs(title = "Life expectancy by continent",
           x     = "Year",
           y     = "Avergae life expectancy",
           ) +
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 270, hjust = 1)) +
      # create grid
      facet_grid( .~ continent)
```



We observe that Africa has the lowest life expectancy over the years. Compared to the other continents, Africa has an interesting "flattening" and small reduction in life expectancy around years 1990-2000. There might be great explanations for this, but we will not make a guess here.

America has somewhat higher life expectancy than Asia, in addition to a smaller standard deviation. Oceania has the highest life expectancy, followed by Europe. Interestingly, Oceania has a very low standard deviation compared to all other continents. We speculate whether this might be explained by factors such as few countries being part of Oceania, or the effect of living more isolated from the rest of the world when on an island.

Task 2

Import necessary libraries:

```
library(ggplot2)     # for plotting
library(patchwork)   # creates a patchwork of all our plots
library(fastDummies) # for dummy encoding
library(lubridate)   # for date/time handling
library(dplyr)
library(caret)       # test-train splitting
```

2 a)

Import the file weatherHistory.csv. Look at the first few rows of the data. Discuss what types of variables you have. Which variables are categorical and which are numerical? Visualize the data (use bar plots and histograms with kernel density estimates). Discuss the data.

Read the dataset as a data.frame

```
df <- read.csv("weatherHistory.csv")
```

Display the first few rows of the data

```
head(df)
```

```
##                  Formatted.Date       Summary Precip.Type Temperature..C.
## 1 2006-04-01 00:00:00.000 +0200 Partly Cloudy        rain        9.472222
## 2 2006-04-01 01:00:00.000 +0200 Partly Cloudy        rain        9.355556
## 3 2006-04-01 02:00:00.000 +0200 Mostly Cloudy        rain        9.377778
## 4 2006-04-01 03:00:00.000 +0200 Partly Cloudy        rain        8.288889
## 5 2006-04-01 04:00:00.000 +0200 Mostly Cloudy        rain        8.755556
## 6 2006-04-01 05:00:00.000 +0200 Partly Cloudy        rain        9.222222
##   Apparent.Temperature..C. Humidity Wind.Speed..km.h. Wind.Bearing..degrees.
## 1                 7.388889     0.89           14.1197                    251
## 2                 7.227778     0.86           14.2646                    259
## 3                 9.377778     0.89            3.9284                    204
## 4                 5.944444     0.83           14.1036                    269
## 5                 6.977778     0.83           11.0446                    259
## 6                 7.111111     0.85           13.9587                    258
##   Visibility..km. Loud.Cover Pressure..millibars.
## 1         15.8263          0              1015.13
## 2         15.8263          0              1015.63
## 3         14.9569          0              1015.94
## 4         15.8263          0              1016.41
## 5         15.8263          0              1016.51
## 6         14.9569          0              1016.66
##                        Daily.Summary
## 1 Partly cloudy throughout the day.
## 2 Partly cloudy throughout the day.
## 3 Partly cloudy throughout the day.
## 4 Partly cloudy throughout the day.
## 5 Partly cloudy throughout the day.
## 6 Partly cloudy throughout the day.
```

Observing and editing variable classes

We observe dates, text and numbers. By investigating further we find that the column "Loud.Cover" consistently holds a value of 0, and does not actually contribute with any information. We believe this column is actually meant to be named "Cloud cover".

Let us see what type of class R has stored these values as:

```
lapply(df, class) # printing the class for each column in df
```

```
## $Formatted.Date
## [1] "character"
##
## $Summary
## [1] "character"
##
## $Precip.Type
## [1] "character"
##
## $Temperature..C.
## [1] "numeric"
##
## $Apparent.Temperature..C.
## [1] "numeric"
##
## $Humidity
## [1] "numeric"
##
## $Wind.Speed..km.h.
## [1] "numeric"
##
## $Wind.Bearing..degrees.
## [1] "numeric"
##
## $Visibility..km.
## [1] "numeric"
##
## $Loud.Cover
## [1] "numeric"
##
## $Pressure..millibars.
## [1] "numeric"
##
## $Daily.Summary
## [1] "character"
```

We want to transform all variables of type character (expect date) to categorical values, and make "Loud.Cover" and "Wind.Bearing..degrees." integers.

```
# Categorical
df[,c("Summary", "Precip.Type", "Daily.Summary")] <-
  lapply(df[,c("Summary", "Precip.Type", "Daily.Summary")], as.factor)

# Integers
df[,c("Wind.Bearing..degrees.", "Loud.Cover")] <-
  lapply(df[,c("Wind.Bearing..degrees.", "Loud.Cover")], as.integer)
```

Also, we choose to format the date to contain only the necessary information.

```
# Convert 'Formatted.Date' column to POSIXlt format with specified date and time format
df$Formatted.Date <- as.POSIXlt(df$Formatted.Date, format="%Y-%m-%d %H:%M:%OS %z")

# Reformat date to a simpler format (year-month-day hour:minute)
df$Formatted.Date <- format(df$Formatted.Date, format="%Y-%m-%d %H:%M")
```

Now we can produce a better summary of the data:

```
summary(df)
```

```
##  Formatted.Date                    Summary      Precip.Type  Temperature..C.
##  Length:96453       Partly Cloudy      :31733   null:  517   Min.   :-21.822
##  Class :character   Mostly Cloudy      :28094   rain:85224   1st Qu.:  4.689
##  Mode  :character   Overcast           :16597   snow:10712   Median : 12.000
##                     Clear              :10890                Mean   : 11.933
##                     Foggy              : 7148                3rd Qu.: 18.839
##                     Breezy and Overcast:  528                Max.   : 39.906
##                     (Other)            : 1463
##  Apparent.Temperature..C.    Humidity      Wind.Speed..km.h.
##  Min.   :-27.717          Min.   :0.0000   Min.   : 0.000
##  1st Qu.:  2.311          1st Qu.:0.6000   1st Qu.: 5.828
##  Median : 12.000          Median :0.7800   Median : 9.966
##  Mean   : 10.855          Mean   :0.7349   Mean   :10.811
##  3rd Qu.: 18.839          3rd Qu.:0.8900   3rd Qu.:14.136
##  Max.   : 39.344          Max.   :1.0000   Max.   :63.853
##
##  Wind.Bearing..degrees. Visibility..km.   Loud.Cover Pressure..millibars.
##  Min.   :  0.0          Min.   : 0.00   Min.   :0    Min.   :   0
##  1st Qu.:116.0          1st Qu.: 8.34   1st Qu.:0    1st Qu.:1012
##  Median :180.0          Median :10.05   Median :0    Median :1016
##  Mean   :187.5          Mean   :10.35   Mean   :0    Mean   :1003
##  3rd Qu.:290.0          3rd Qu.:14.81   3rd Qu.:0    3rd Qu.:1021
##  Max.   :359.0          Max.   :16.10   Max.   :0    Max.   :1046
##
##                                                 Daily.Summary
##  Mostly cloudy throughout the day.                     :20085
##  Partly cloudy throughout the day.                     : 9981
##  Partly cloudy until night.                            : 6169
##  Partly cloudy starting in the morning.                : 5184
##  Foggy in the morning.                                 : 4201
##  Foggy starting overnight continuing until morning.: 3576
##  (Other)                                               :47257
```

Visualizing the data

Create the histograms with kernel density estimate curve. Note that we first remove the rows where the pressure is recorded as 0. This is not a possible value, and has to be wrongful measurements.

```
sprintf("Number of rows with Pressure = 0 to delete: %s", sum(df$Pressure..millibars. == 0))
```

```
## [1] "Number of rows with Pressure = 0 to delete: 1288"
```

```
# Remove outliers/wrongful measurements of 0 pressure
df_filtered <- df[df$Pressure..millibars. != 0, ]
```

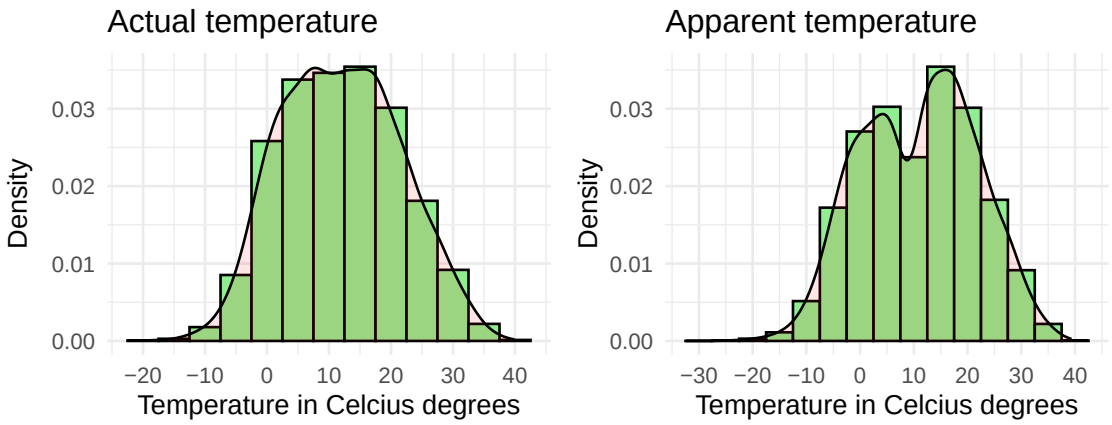
```r
# Creating framework for the histogram with KDE plot
hist_KDE_plot <- function(data=df_filtered, col,
                          hist_bw, density_bw,
                          title, xlab,
                          xmin, xmax, xstep,
                          ymin, ymax, ystep){
        ggplot(data, aes(x=col)) +
    # Create histogram with density on the y-axis
        geom_histogram(aes(y=after_stat(density)), # density (not frequency) on y axis
                       binwidth=hist_bw,  # set width of the bins
                       color="black",
                       fill="lightgreen") +
    # Create kernel density plot
        geom_density(alpha=0.1,
                     fill="red",
                     bw=density_bw) + # bw = smoothing bandwidth
    # Set title and axis-names
        labs(title=title,
             x = xlab,
             y = "Density") +
    # Set values for the span of the axes
        scale_x_continuous(breaks = seq(xmin, xmax, by = xstep)) +
        scale_y_continuous(breaks = seq(ymin, ymax, by = ystep)) +
        theme_minimal()
}

# Creating the different plots
tempC_plot <- hist_KDE_plot(col         = df_filtered$Temperature..C.,
                            hist_bw     = 5,
                            density_bw = 2,
                            title       = "Actual temperature",
                            xlab        = "Temperature in Celcius degrees",
                            xmin        = -20,
                            xmax        = 40,
                            xstep       = 10,
                            ymin        = 0,
                            ymax        = 0.04,
                            ystep       = 0.01)

appTempC_plot <- hist_KDE_plot(col      = df_filtered$Apparent.Temperature..C.,
                            hist_bw     = 5,
                            density_bw = 2,
                            title       = "Apparent temperature",
                            xlab        = "Temperature in Celcius degrees",
                            xmin        = -30,
                            xmax        = 40,
                            xstep       = 10,
                            ymin        = 0,
                            ymax        = 0.04,
                            ystep       = 0.01)

humid_plot <- hist_KDE_plot(col         = df_filtered$Humidity,
                            hist_bw     = 0.1,
```

```
                              density_bw = 0.05,
                              title      = "Humidity",
                              xlab       = "Humidity in %",
                              xmin       = 0,
                              xmax       = 1,
                              xstep      = 0.1,
                              ymin       = 0,
                              ymax       = 3,
                              ystep      = 0.5)

windSpeed_plot <- hist_KDE_plot(col     = df_filtered$Wind.Speed..km.h.,
                              hist_bw    = 5,
                              density_bw = 2,
                              title      = "Wind Speed",
                              xlab       = "km/h",
                              xmin       = 0,
                              xmax       = 45,
                              xstep      = 5,
                              ymin       = 0,
                              ymax       = 0.07,
                              ystep      = 0.01)

windDirection_plot <- hist_KDE_plot(col = df_filtered$Wind.Bearing..degrees.,
                              hist_bw    = 30,
                              density_bw = 10,
                              title      = "Wind Direction",
                              xlab       = "degrees from 0 (north)",
                              xmin       = 0,
                              xmax       = 400,
                              xstep      = 50,
                              ymin       = 0,
                              ymax       = 0.005,
                              ystep      = 0.001)

visibility_plot <- hist_KDE_plot(col    = df_filtered$Visibility..km.,
                              hist_bw    = 3,
                              density_bw = 1.5,
                              title      = "Visibility",
                              xlab       = "Visibility in km",
                              xmin       = 0,
                              xmax       = 17,
                              xstep      = 2,
                              ymin       = 0,
                              ymax       = 0.125,
                              ystep      = 0.025)

pressure_plot <- hist_KDE_plot(col      = df_filtered$Pressure..millibars.,
                              hist_bw    = 5,
                              density_bw = 2,
                              title      = "Pressure",
                              xlab       = "Pressure in millibars",
                              xmin       = 970,
                              xmax       = 1050,
```

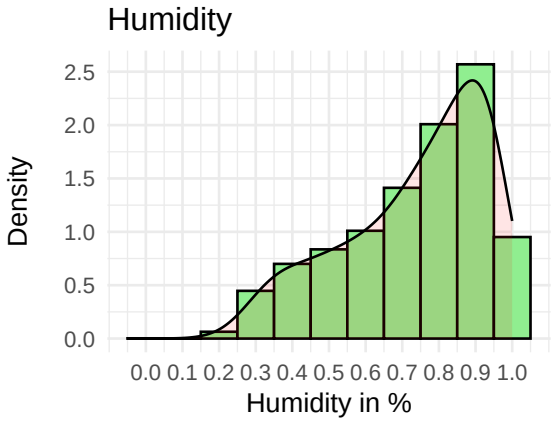
```
                              xstep      = 10,
                              ymin       = 0,
                              ymax       = 0.06,
                              ystep      = 0.01)
```

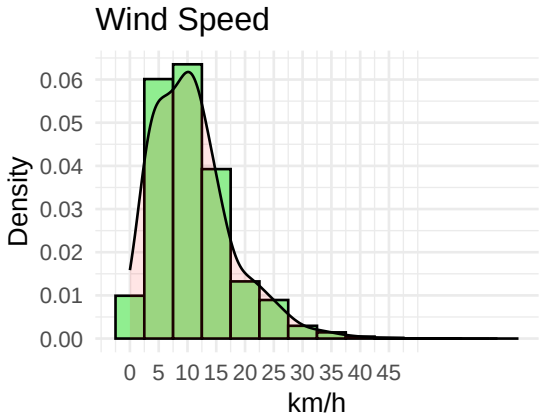
Displaying the histograms

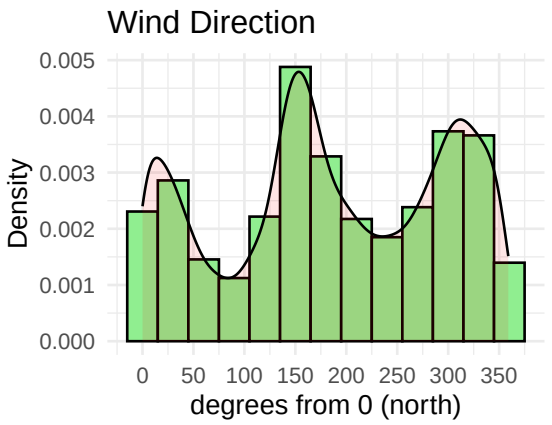
```
# Arrange the plots in a grid layout
combined_plot <- (tempC_plot + appTempC_plot) /
                 (humid_plot + windSpeed_plot) /
                 (windDirection_plot + visibility_plot) /
                 (pressure_plot + plot_spacer()) +
                 plot_layout(ncol=1)

print(combined_plot)
```

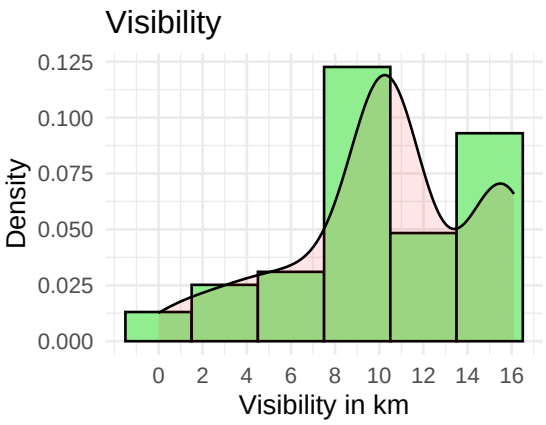

Actual temperature
Density
0.03
0.02
0.01
0.00
−20 −10 0 10 20 30 40
Temperature in Celcius degrees

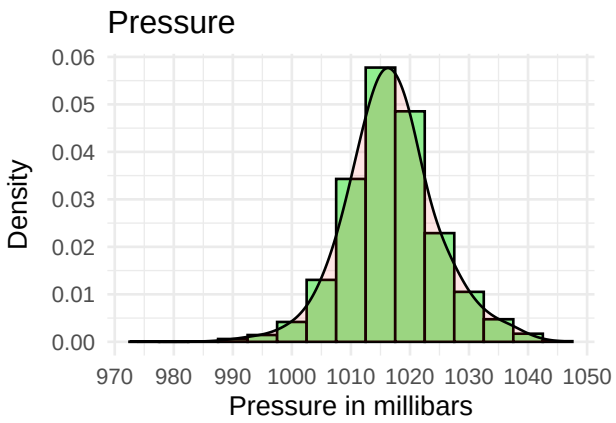
Apparent temperature
Density
0.03
0.02
0.01
0.00
−30 −20 −10 0 10 20 30 40
Temperature in Celcius degrees


Humidity
Density
2.5
2.0
1.5
1.0
0.5
0.0
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
Humidity in %


Wind Speed
Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
0 5 10 15 20 25 30 35 40 45
km/h

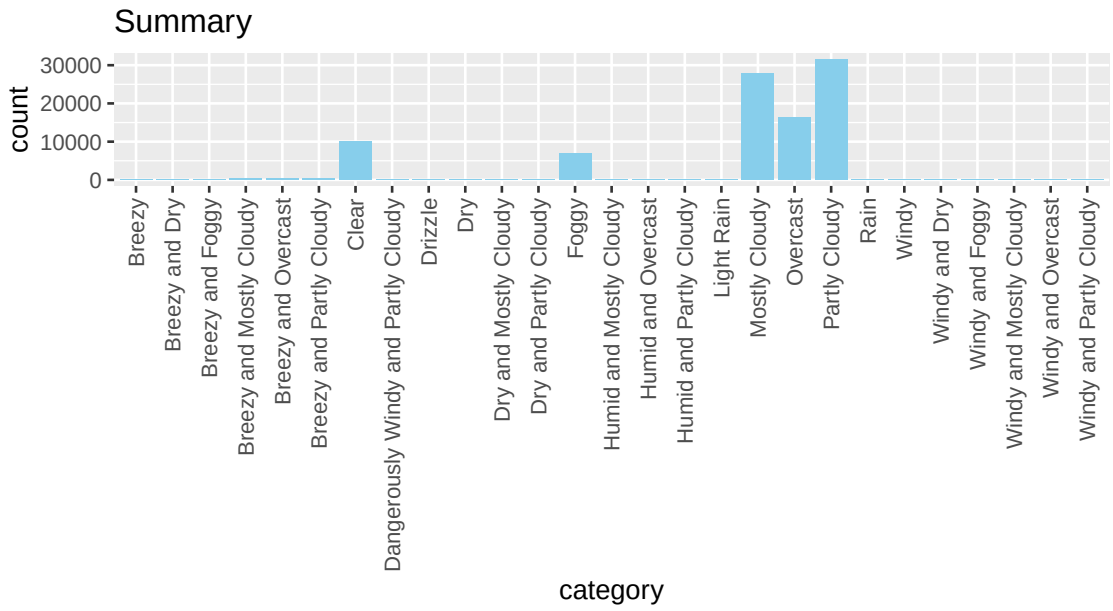

Wind Direction
Density
0.005
0.004
0.003
0.002
0.001
0.000
0 50 100 150 200 250 300 350
degrees from 0 (north)

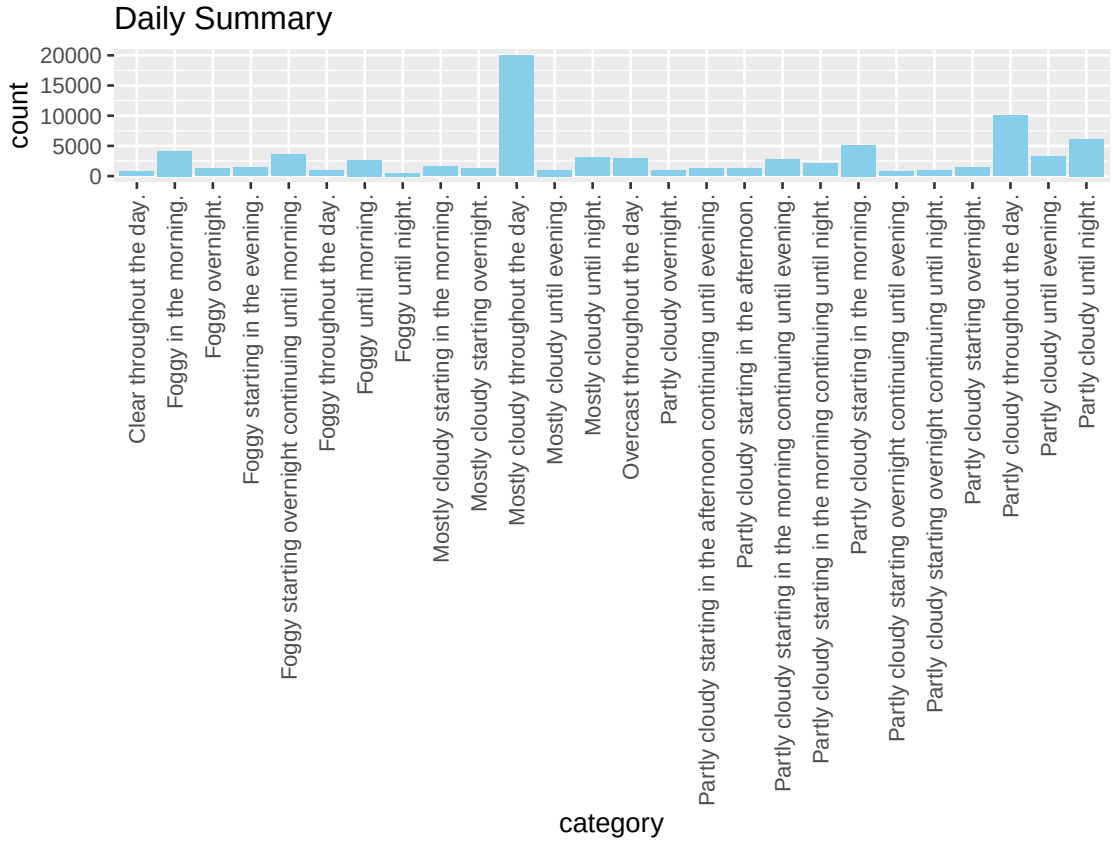

Visibility
Density
0.125
0.100
0.075
0.050
0.025
0.000
0 2 4 6 8 10 12 14 16
Visibility in km

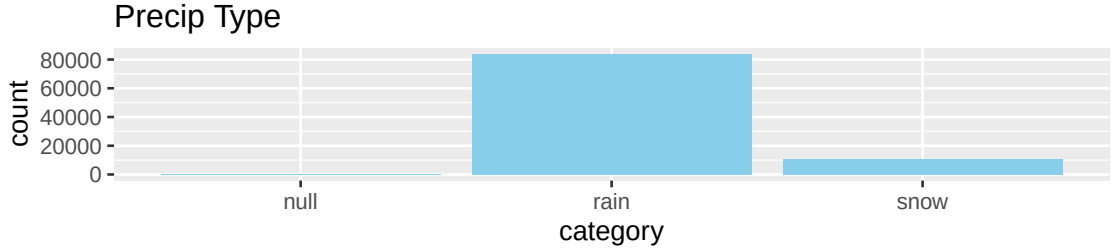

Pressure
Density
0.06
0.05
0.04
0.03
0.02
0.01
0.00
970 980 990 1000 1010 1020 1030 1040 1050
Pressure in millibars

Creating bar plots for categorical values. For "Daily.Summary" we only show categories with above 500 observations.

```
### Summary
summary_plot = ggplot(df_filtered, aes(x=Summary)) +
  geom_bar(fill = "skyblue") +
  labs(title  ="Summary",
            x = "category",
            y = "count") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) #adjust label angle


### Daily Summary
# length(unique(filtered_df$Daily.Summary)) = 214 levels

# count category observations
category_count <- table(df_filtered$Daily.Summary)
# find categories with observations above threshold
categories_above_threshold <- names(category_count[category_count>500])
# pick the correct observations
categories_to_plot <- df_filtered[df_filtered$Daily.Summary %in% categories_above_threshold,]

dailySummary_plot = ggplot(categories_to_plot, aes(x=Daily.Summary)) +
  geom_bar(fill = "skyblue") +
  labs(title  = "Daily Summary",
            x = "category",
            y = "count") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))


### Precip Type
precip_plot = ggplot(df_filtered, aes(x=Precip.Type)) +
  geom_bar(fill="skyblue") +
    labs(title = "Precip Type",
            x  = "category",
            y  = "count")


### Arrange the plots in a grid layout
combined_plot <- (summary_plot) /
                 (dailySummary_plot) /
                 (precip_plot) +
                plot_layout(ncol=1)

print(combined_plot)
```


Summary
count
30000
20000
10000
0
Breezy
Breezy and Dry
Breezy and Foggy
Breezy and Mostly Cloudy
Breezy and Overcast
Breezy and Partly Cloudy
Clear
Dangerously Windy and Partly Cloudy
Drizzle
Dry
Dry and Mostly Cloudy
Dry and Partly Cloudy
Foggy
Humid and Mostly Cloudy
Humid and Overcast
Humid and Partly Cloudy
Light Rain
Mostly Cloudy
Overcast
Partly Cloudy
Rain
Windy
Windy and Dry
Windy and Foggy
Windy and Mostly Cloudy
Windy and Overcast
Windy and Partly Cloudy
category


Daily Summary
count
20000
15000
10000
5000
0
Clear throughout the day.
Foggy in the morning.
Foggy overnight.
Foggy starting in the evening.
Foggy starting overnight continuing until morning.
Foggy throughout the day.
Foggy until morning.
Foggy until night.
Mostly cloudy starting in the morning.
Mostly cloudy starting overnight.
Mostly cloudy throughout the day.
Mostly cloudy until evening.
Mostly cloudy until night.
Overcast throughout the day.
Partly cloudy overnight.
Partly cloudy starting in the afternoon continuing until evening.
Partly cloudy starting in the afternoon.
Partly cloudy starting in the morning continuing until evening.
Partly cloudy starting in the morning continuing until night.
Partly cloudy starting in the morning.
Partly cloudy starting overnight continuing until evening.
Partly cloudy starting overnight continuing until night.
Partly cloudy starting overnight.
Partly cloudy throughout the day.
Partly cloudy until evening.
Partly cloudy until night.
category


Precip Type
count
80000
60000
40000
20000
0
null
rain
snow
category

Discussion: As mentioned we have unvalid measurements of pressure, and hence we choose to exclude this data. Also, the column "Loud.Cover" is consistently 0, and does not contribute with any information. We will remove this column later on in this exercise. We observe a tendency to a bimodal distribution in the apparent temperatures compared to the distribution for the actual temperatures, which seems to be normally distributed. As expected, the pressure also seems to follow a normal distribution. The wind distribution is right skewed, while the humidity has a left skewed distribution. The humidity also has interestigly high values. The wind direction has a multimodal distribution, and the visibility is relatively great on average.

For the categorical variables "Summary" and "Daily.Summary" we have quite many categories. The "Daily Summary" contains 214 levels, which is too many to obtain valuable information. In addition, the majority of the ledf_filvels contain relatively few observations, while only a few of the categories contain large amounts of observations. The same goes for "Summary" even though the number of categories are only 27. This uneven distribution between the categories is the reason for excluding these variables later on, when creating a linear regression model.

As for the precip type, we observe a majority of rain. It is unclear what the "null" values represents. This could represent missing information, but it could also represent dry days, without any sort of precipitation. However, the amount of rainy days does correspond greatly with the high level of humidity and relatively high temperatures.

2 b)

Perform preprocessing using dummy/onehot encoding and standard scaling. Divide the data into 75%-25% train-test split randomly.

We have already removed outliers of 0 pressure, so we continue with the following preprocessing:

1. Date formatting and converting date and time to numerical values using sinus and cosinus transformations.
2. Dummy encoding of selected features
3. Standard scaling
4. Train-test splitting

For simplicity, we start by renaming our dataframe

```
lr_df = df_filtered
```

Date formatting

We extract the different elements of the date into separate columns

```
# Format as a date
lr_df$Formatted.Date <- as.POSIXlt(lr_df$Formatted.Date, format="%Y-%m-%d %H:%M")
# Extract features and create new columns
lr_df$day   <- yday(lr_df$Formatted.Date) # day of year
lr_df$month <- month(lr_df$Formatted.Date)
lr_df$year  <- year(lr_df$Formatted.Date)
lr_df$hour  <- hour(lr_df$Formatted.Date)
```

We transform the day of year, month and hour to cyclical features using cosinus and sinus transformations. This way, january represented as number 1 and december represented as number 12, will no longer be "far" from each other when in reality they are closer than say, january and july. We keep the year as it is.

```
cycle <- function(col, div, sin=TRUE) {
  if (sin) {
    return(sin(2*pi * col/div))
  }
```

```
  else {
    return(cos(2*pi * col/div))
  }

}
# turn date and time into cyclical features
nMonths = 12
nHours = 24
nDays = 365 # ignore leap years

# Create new features
lr_df$hour_sin <- cycle(lr_df$hour, nHours)
lr_df$hour_cos <- cycle(lr_df$hour, nHours, FALSE)

lr_df$day_sin <- cycle(lr_df$day, nDays)
lr_df$day_cos <- cycle(lr_df$day, nDays, FALSE)

lr_df$month_sin <- cycle(lr_df$month, nMonths)
lr_df$month_cos <- cycle(lr_df$month, nMonths, FALSE)
```

Now, let's remove the redundant columns. Note, we also remove "Loud.Cover" as it contains no useful information, and "Daily.Summary" as it contains too many categories with uneven distribution to provide useful information.

```
# Remove columns
lr_df <- subset(lr_df, select = -c(day,
                                   month,
                                   hour,
                                   Formatted.Date,
                                   Daily.Summary,
                                   Loud.Cover))
```

Dummy-encoding

We perform dummy encoding on the columns "summary" and "precip type", and delete the original columns. We use dummy encoding rather than OneHot encoding to get fewer new features, and prevent issues arising from perfect correlation when doing linear regression later on.

```
# Dummy encoding
df_dummy <- dummy_cols(lr_df,
                       select_columns          = c("Precip.Type", "Summary"),
                       remove_first_dummy      = TRUE, # not onehot-encoding
                       remove_selected_columns = TRUE) # remove original columns
```

Scaling

Scaling all the numerical columns, except the target, Apparent Temperature.

```
# mutate: creates new variables on top of our existing ones
# across: enables mutating across multiple columns at once
df_scaled <- df_dummy %>%
  mutate(across(-c(starts_with("Precip"),
                   starts_with("Summary"),
                   Apparent.Temperature..C.),
                scale)
         )
```

Test-train splits

Create random test-train splits

```
tt_split <- createDataPartition(df_scaled$Apparent.Temperature..C.,
                                times=1, # number of partitions
                                p=0.75, # 75% training data
                                list = FALSE)
train <- df_scaled[tt_split,]
test <- df_scaled[-tt_split,]
```

2 c)

Train a linear regression model with all the predictors you find logical to include based on your analysis in A) and use ApparentTemperature (C) as the target. Calculate the metrics below and interpret the hypotheses tests for the model parameters.

1. *RMSE (root mean squared error)*
2. *MAE (mean absolute error)*
3. *coefficient of determination (R2 score)*

Training a linear regression model; We choose to keep all the numerical values, but remove the dummy-encoded values for the "Summary"-groups. This is due to the same reason as excluding "Daily.Summary" earlier.

```
lm   <- lm(Apparent.Temperature..C. ~.,
           data = train %>% select(-starts_with("Summary") # remove summary dummy-columns
            ))
pred <- predict(lm,
                newdata = test %>% select(-starts_with("Summary")))
```

Calculating metrics

```
RMSE(pred, test$Apparent.Temperature..C.)
```

```
## [1] 1.056845
```

```
MAE(pred, test$Apparent.Temperature..C.)
```

```
## [1] 0.8270956
```

```
summary(lm)$r.squared
```

```
## [1] 0.9902131
```

The R-squared value indicated that 99% of the variation in the data is explained by the predictors. We also have a Mean Absolute Error of 0.8, meaning our absolute errors is less than 1 degree celcius off. This indicated a relatively good model.

Let's interpret the hypotheses tests for the model parameters. The hypotheses are:

- H0: beta = 0 (not a linear relationship between the target and the predictor)
- H1: beta!= 0 (a linear relationship between the target and predictor exists)

We observe the p-values:

```
summary(lm)
```

```
##
## Call:
## lm(formula = Apparent.Temperature..C. ~ ., data = train %>% select(-starts_with("Summary")))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1375 -0.7043 -0.1137  0.6323  5.2357
##
## Coefficients:
##                           Estimate Std. Error  t value Pr(>|t|)
## (Intercept)             11.296669   0.054251  208.230  < 2e-16 ***
## Temperature..C.         10.491381   0.011497  912.529  < 2e-16 ***
## Humidity                 0.198972   0.007043   28.251  < 2e-16 ***
## Wind.Speed..km.h.       -0.624294   0.004549 -137.228  < 2e-16 ***
## Wind.Bearing..degrees.   0.048700   0.004021   12.110  < 2e-16 ***
## Visibility..km.         -0.039261   0.005032   -7.803 6.14e-15 ***
## Pressure..millibars.     0.127198   0.004714   26.983  < 2e-16 ***
## year                    -0.006757   0.004048   -1.669 0.095068 .
## hour_sin                -0.014972   0.004098   -3.654 0.000259 ***
## hour_cos                -0.052666   0.005382   -9.786  < 2e-16 ***
## day_sin                 -0.016655   0.026312   -0.633 0.526740
## day_cos                 -0.301277   0.027461  -10.971  < 2e-16 ***
## month_sin                0.060171   0.026188    2.298 0.021583 *
## month_cos                0.045421   0.026296    1.727 0.084119 .
## Precip.Type_rain        -0.358696   0.054479   -6.584 4.61e-11 ***
## Precip.Type_snow        -0.904626   0.055935  -16.173  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.057 on 71358 degrees of freedom
## Multiple R-squared:  0.9902, Adjusted R-squared:  0.9902
## F-statistic: 4.813e+05 on 15 and 71358 DF,  p-value: < 2.2e-16
```

Not significant predictors have a p-value >0.05, which in our case are only the columns "year", "day_sin" and "month_cos". A possible interpretation could be that the global warming over the few years we are investigating are not significant enough to have an effect on our target variable. As for the cos/sin transformation of day and month: We assume that the cosinus variable are able to explain the yearly seasonal variances better than the sinus variable. As they are both representing seasonal variances over the year, the effect on the target explained by day_sin might correspond greatly to the effect already explained by month_sin, and vica verca for month_cos and day_cos. This model might therefore see day_sin and month_cos as "redundant". As opposed to this, hour_sin and hour_cos are explaining seasonal variances on a daily basis, explaining why both variables are considered significant.

We reject the null hypothesis for significant variables. The p-value for the overall model also indicates that we can reject the null hypothesis, telling us that we have a significant model.

Task 3

Import libraries

```
library(ggplot2)
library(dplyr)
```

3 a)

1) Linearity

Linear relationship between the independent variables(predictors) and the dependent variable(response). Lin.reg. models assume the relationship between predictors and the response is linear, and makes it straightforward to interpret the coefficients.

Each coefficient represents the change in the response variable for a one-unit change in the predictor.
If the relationship is not linear, estimated coefficients in least squares may not accurately represent the relationship between predictors and the response.

Diagnosed by: Residuals. Plotting the residuals against the fitted values or against each predictor. In a well-fitted linear model, residuals should be randomly scattered around 0 without any discernible pattern.

2) Homoscedasticity

The variance of the residuals is constant across all levels of the independent variables, shown by the spread across all levels of the independent variables. It should be the same for all predictors.

Diagnosed by: Using a residual vs.fitted plot, where the residuals are on the y-axis and the fitted values are on the x-axis. If the residuals show a random scatter with constant spread across the range of fitted values, homoscedasticity likely holds. If the plot shows a pattern, specifically we are looking for a fanning out or fanning in pattern, meaning that residuals are spreading out as predicted values change, then homoscedasticity likely does not hold.

Can be diagnosed with a Scale-Location plot. Homoscedasticity is indicated if the points are evenly distributed horizontally with no discernible pattern. A trend in the red line indicates non-constant variance.

Residuals vs. Leverage: If the spread of residuals changes with leverage, it can suggest heteroscedasticity.

3) Independence

Observations are independent of each other. The residuals for one observation should not be correlated with the residuals for another observation. This assumption is crucial because if observations are not independent, it can bias the regression estimates, affect standard errors and lead to incorrect statistical inferences.

It is likely to be violated in Time-Series Data, because observations here are often dependent as data points are recorded over time. In cases where data are clustered or grouped, observations within the same group may be more similar to each other than to observations from different groups.

Diagnosed with residual autocorrelation: how much a residual at time t depends on the residual at time t-1. Diagnosed by: plotting residuals. Plotting the residuals in the order of data collection(against time for time-series).

If the residuals show a pattern; trends, cycles or clustering, it suggests autocorrelation and thus a violation of the independence assumption.

4) Normality

Assumption that residuals are normally distributed.

Diagnosed by: Histogram of residuals. Visually check if they resemble a normal distribution. The histogram should appear symmetric, centered around 0 and bell-shaped.

Diagnosed by: QQ-Plot. If the residuals are normally distributed, the points on the QQ-plot will lie approximately on a straight diagonal line. If the points deviate from the line(e.g., forming an "S" shape), this suggests a violation of the normality assumption, such as skewness or heavy tails (kurtosis).

Diagnosed by: A Shapiro-Wilk test, checking whether the residuals follow a normal distribution. The Null-Hypothesis here is that the residuals are normally distributed. The Alternative Hypothesis is that the residuals are not normally distributed. A significant p-value, less than 0.05, indicates that the residuals are not normally distributed.

3 b)

Residuals vs. Fitted: Plots the residuals (errors): The distance of the observed points to the predicted values from the regression line, vs. the Fitted: The fitted line going through the points generated by the model. The coefficients of the terms in this line are called the fitted values. Fitted values are on the x-axis. Residuals are on the y-axis. Residuals vs. Fitted shows the distance between the residuals and the fitted line.

Normal QQ: Quantiles of the residuals plotted against the quantiles of a normal distribution. If the residuals are normally distributed, the points on the QQ-plot will lie approximately on a straight diagonal line. If the points deviate from the line, this suggests a violation of the normality assumption, such as skewness or heavy tails (kurtosis).

Scale-Location (or Spread-Location): Scale-Location plot, is transformation of the residual plot. It plots the square root of the standardized residuals on the y-axis against the fitted values on the x-axis.

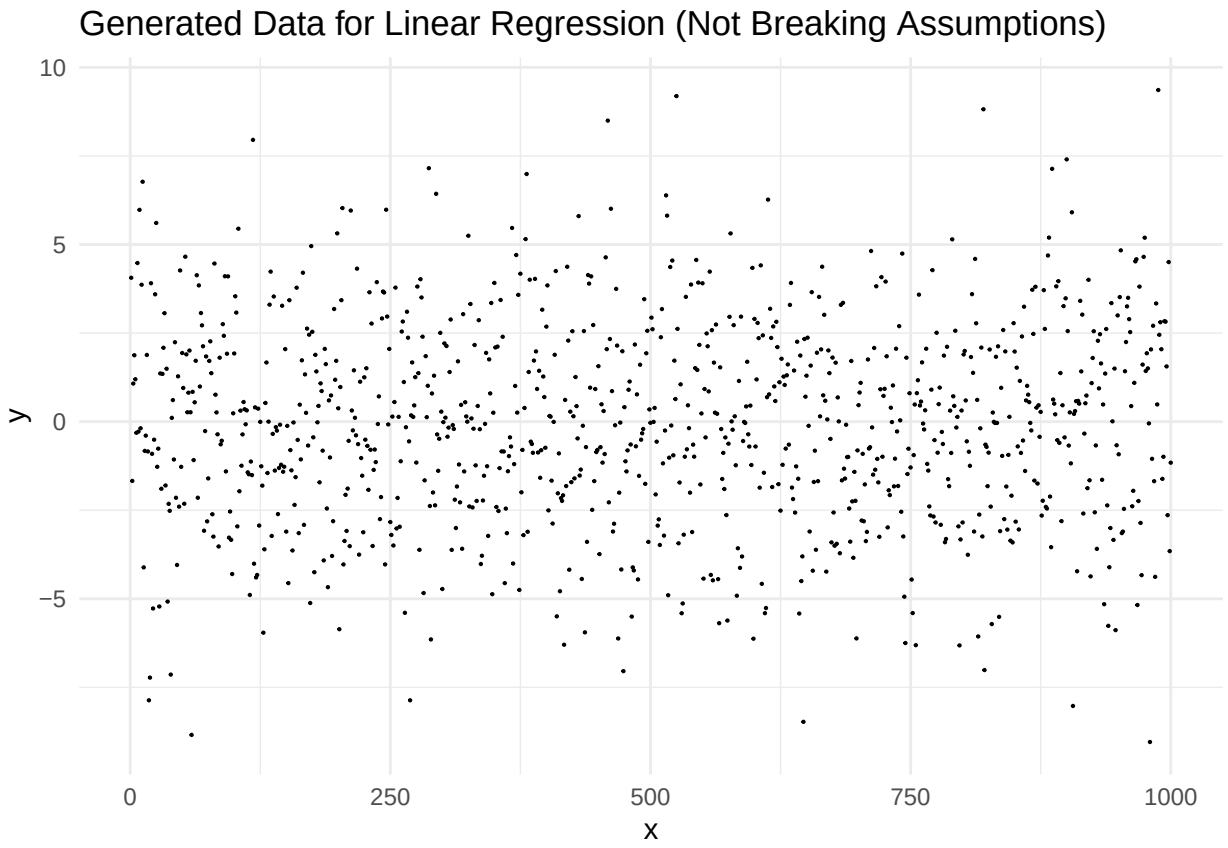
Residuals vs Leverage: The influence and quality of individual data points in a fitted model. Helps in identifying potential outliers, influential points, or cases where the model is poorly fitted. The vertical axis represents the residuals, showing how far off each observation is from the regression line. Leverage measures how far an observation's independent variable values (i.e. the predictor values) are from the mean of the predictor values. Points falling far outside the Cook's distance curves could be influential.

3 c)

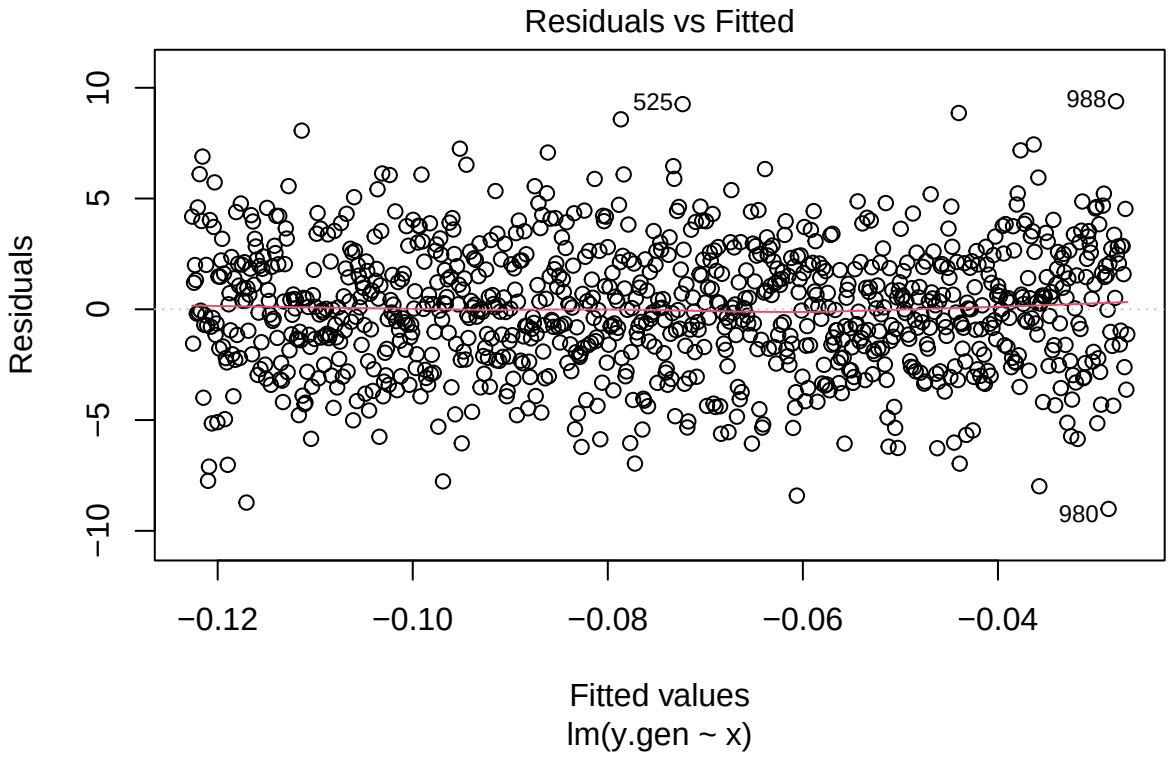
Generating data and creating a linear regression that: - Holds to all the assumptions

```
#A model that Holds to all the assumptions:
set.seed(42)
n <- 1000
x <- 1:n

constant <- 0
trend <- 0 #A trend will indicate nonconstant variance among the residuals,
#so for homoscedasticity to hold it is set to 0.
#If the variance of the residuals is not constant, the residuals will fan out
#or contract as the fitted values increase. So both trend<0 and trend>0
#might contribute to non-constant
#variance.
curve_magnitude <- 0 #If a relationship between variables is curved (non-linear)
#and a linear model is fitted, the linear model will be a poor fit.
#In this case, the linearity assumption of a linear regression is violated
#When curve_magnitude = 0, it suggests there is no curvature, and the relationship
#between X and Y is linear.
#This would fit the assumptions of a linear model.
#A curve magnitude > 0 would leave a non-linear coefficient among the residuals.
#Also, as the relationship becomes more non-linear, which can lead to unequal variance
```

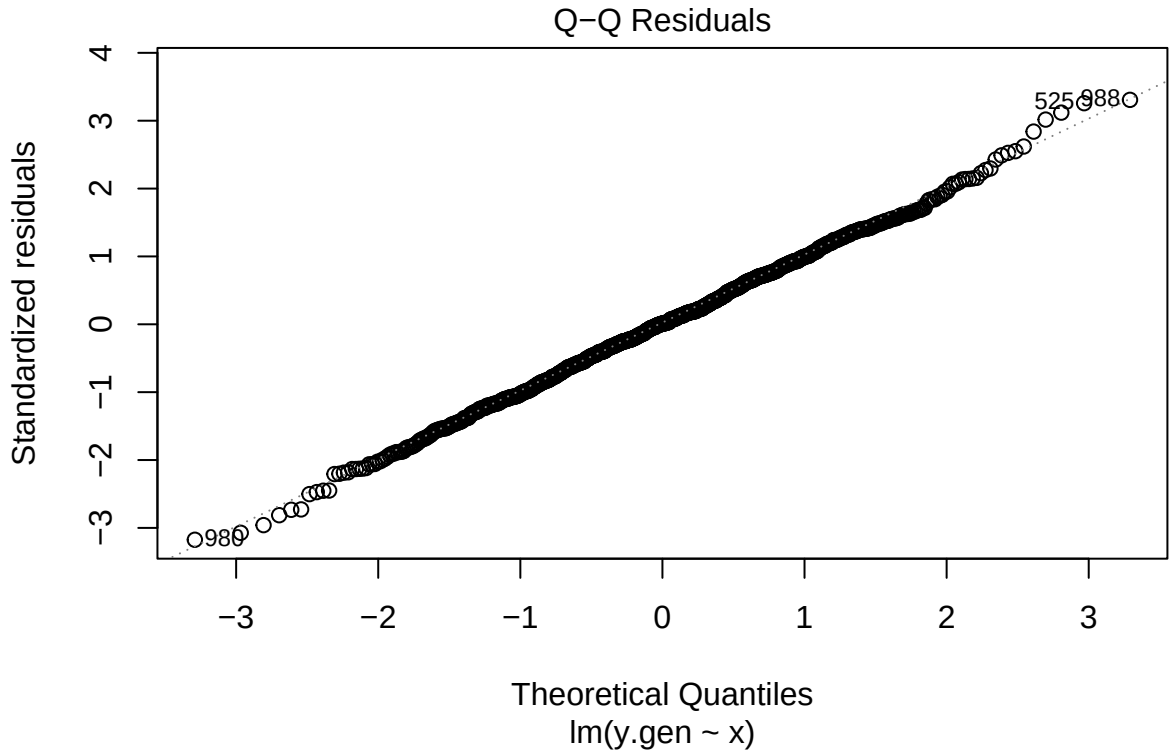
```r
#in the residuals across the range of x. So it can also contribute to heteroscedasticity.
curve_period <- 100
curve_shift <- 0 #There is no shift in the curve, essential for a linear model.
#If the QQ-plot is not a straight line (has a shift), normality
#assumption will break. So it is essential for this to hold.
normal_noise_magnitude <- 1 #The magnitude of the noise in the normal distribution.
#Too much noise may make it harder to interpret accurate results, and
#will not accurately represent normally distributed residuals.
#So this is essential for not breaking the normality assumption.
norm_noise_periode <- 10000 #If residuals are not independent, we may see patterns
#over time, such as cycles or periodicity. This can be violated by adding periodic noise
#components, and setting this parameter to a small value can add periodic noise components.
#So if this is low, independence can break.
shift_norm_noise <- 500
non_normal_noise_magnitude <- 0 #Adding this (setting a value >0) here would add a
#non-normal component to the residuals, breaking the normality assumption.
#Increasing it can add more variability, so it could also
#contribute to heteroscedasticity.
non_norm_noise_periode <- 10000 #Setting this to a small value can also add periodic noise components.
shift_non_norm_noise <- 500.

y.gen <- constant +
  trend * x +
  curve_magnitude * sin((x/curve_period + curve_shift) * pi) +
  normal_noise_magnitude * cos((x/norm_noise_periode + shift_norm_noise/norm_noise_periode) * pi) * rno
  non_normal_noise_magnitude * cos((x/non_norm_noise_periode + shift_non_norm_noise/non_norm_noise_peri

# Use ggplot2 for the scatter plot
p <- ggplot(data = data.frame(x = x, y = y.gen), aes(x = x, y = y)) +
  geom_point(size = 0.1) +
  labs(title = "Generated Data for Linear Regression (Not Breaking Assumptions)", y = "y") +
  theme_minimal()

print(p)
```



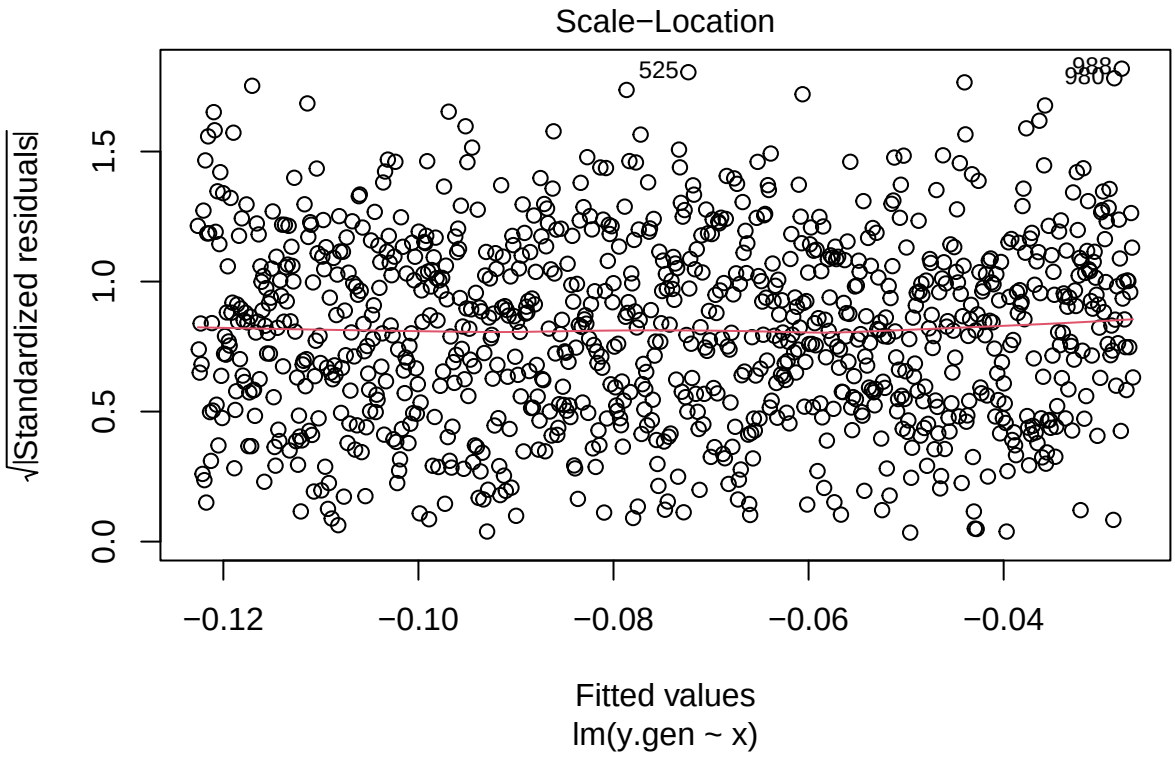

```r
#Diagnostics of model that Holds to all assumptions:
lm.gen <- lm(y.gen ~ x)

# Plot diagnostics
plot(lm.gen, which = 1)  # Residuals vs. Fitted (check linearity, homoscedasticity)
```



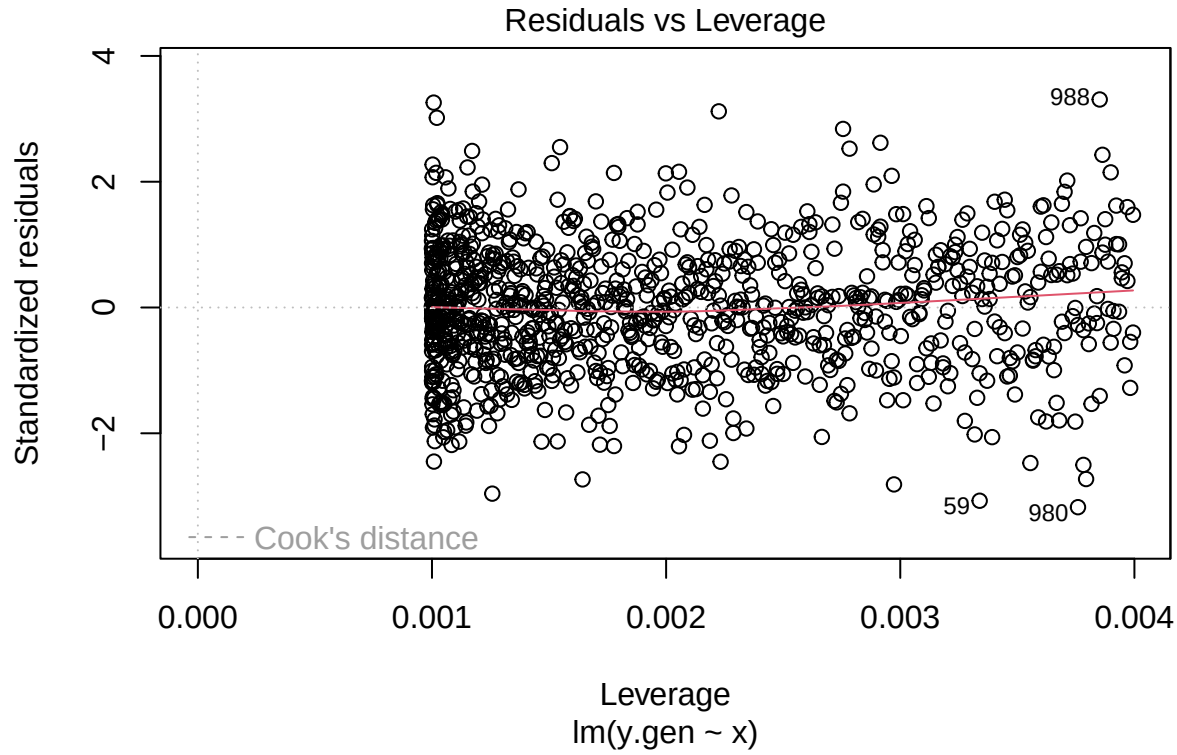

```r
plot(lm.gen, which = 2)  # Q-Q plot (check normality)
```




```r
plot(lm.gen, which = 3)  # Scale-Location (check homoscedasticity)
```

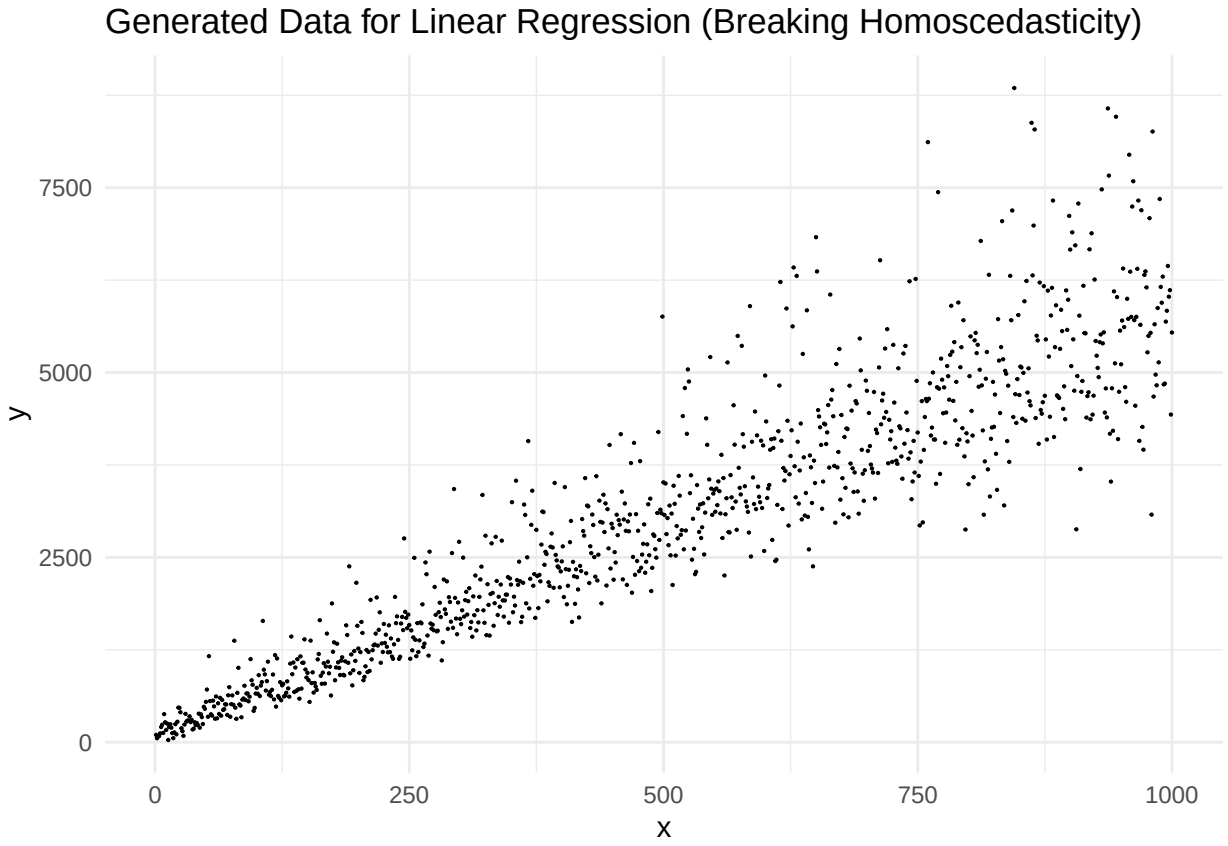



```r
plot(lm.gen, which = 5)  # Residuals vs. Leverage (check influence)
```

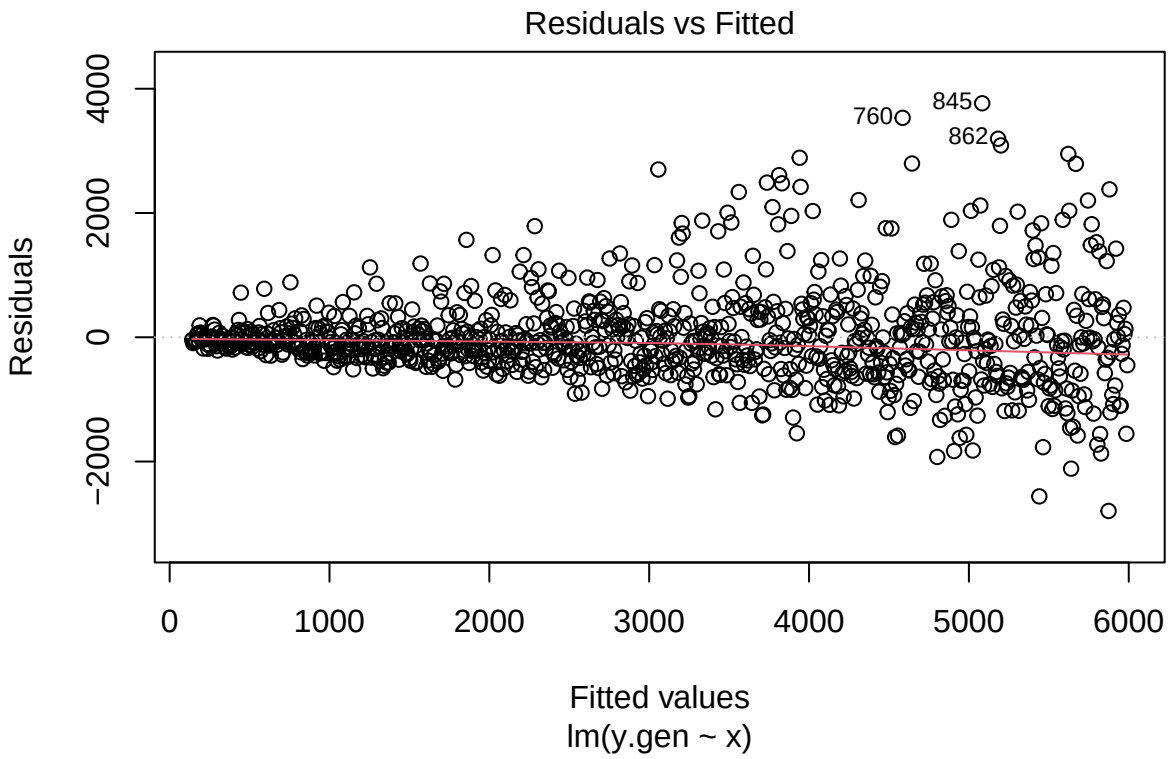



```r
#A model breaking the Homoscedasticity assumption:
set.seed(42)
n <- 1000
x <- 1:n
```

```r
constant <- 0
trend <- 5 #A trend will indicate nonconstant variance among the residuals,
#so for homoscedasticity to hold it is set to 0.
#If the variance of the residuals is not constant, the residuals will fan out
#or contract as the fitted values increase. So both trend<0 and trend>0 will give non-constant
#variance.
curve_magnitude <- 20 #Increased to create variability
curve_period <- 100
curve_shift <- 0
normal_noise_magnitude <- 15 + 0.2*x #Adjusting the noise to vary
#with x to get heteroscedasticity by getting a "fanning in" pattern
norm_noise_periode <- 10000
shift_norm_noise <- 100
non_normal_noise_magnitude <- 20 + 0.2*x #Adjust the noise to vary
#with x to get heteroscedasticity by getting a "fanning in" pattern
non_norm_noise_periode <- 10000
shift_non_norm_noise <- 100.

y.gen <- constant +
  trend * x +
  curve_magnitude * sin((x/curve_period + curve_shift) * pi) +
  normal_noise_magnitude * cos((x/norm_noise_periode + shift_norm_noise/norm_noise_periode) * pi) * rno
  non_normal_noise_magnitude * cos((x/non_norm_noise_periode + shift_non_norm_noise/non_norm_noise_peri

p <- ggplot(data = data.frame(x = x, y = y.gen), aes(x = x, y = y)) +
  geom_point(size = 0.1) +
  labs(title = "Generated Data for Linear Regression (Breaking Homoscedasticity)", y = "y") +
  theme_minimal()

print(p)
```

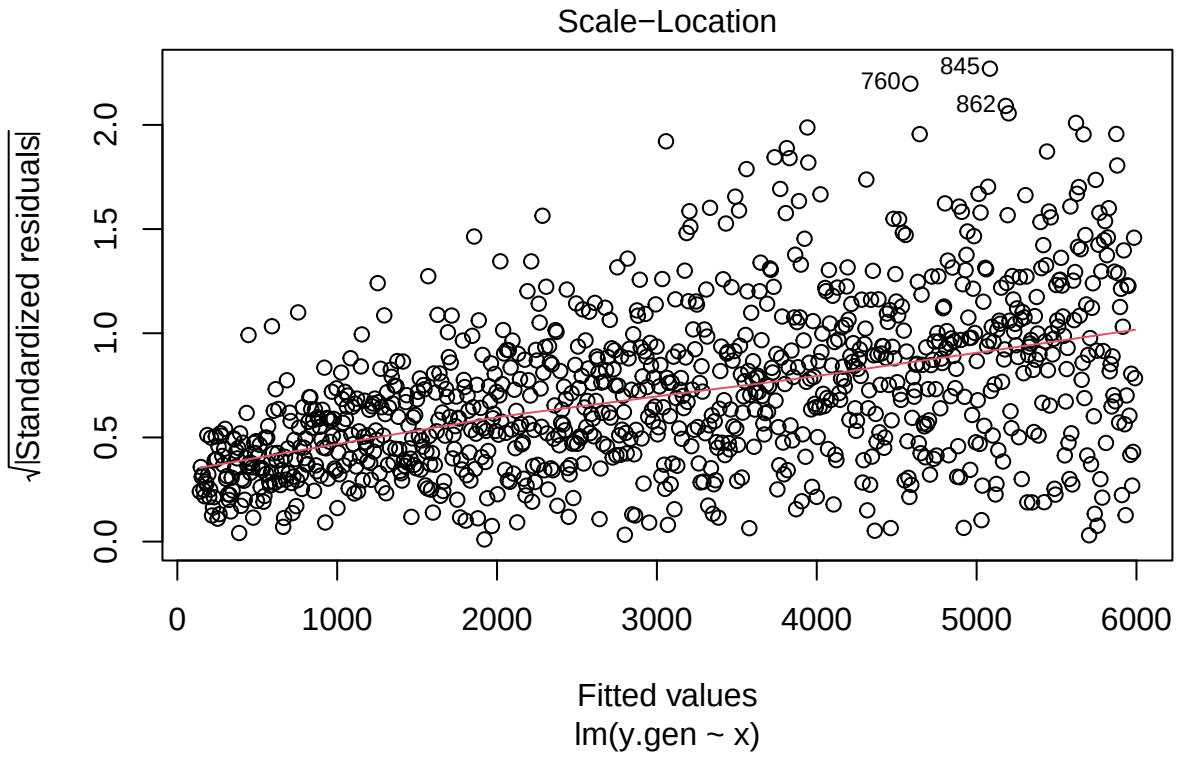



```r
#Diagnostics of model:
lm.gen <- lm(y.gen ~ x)

# Plot diagnostics
plot(lm.gen, which = 1)  # Residuals vs. Fitted (check linearity, homoscedasticity)
```



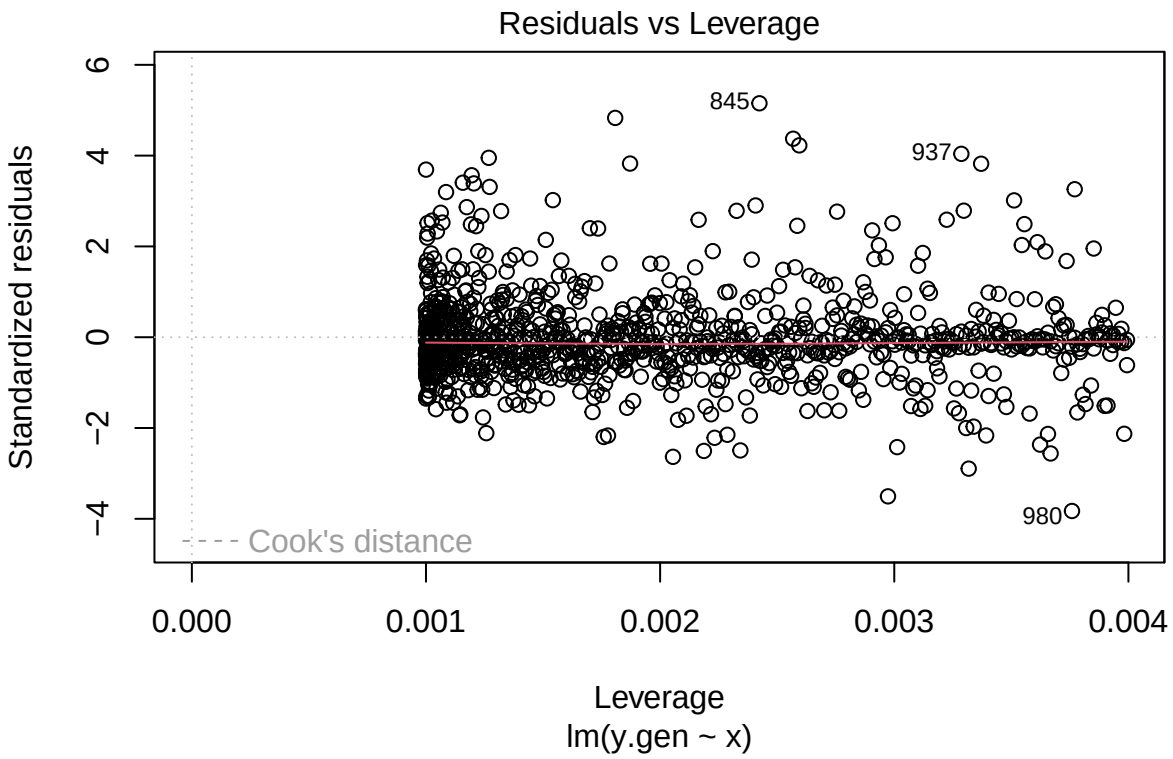

```
#Now, there are more anomalies and larger group of residuals further away from the mean,
#which is no longer around the residual mid-values.:

#If the residuals show a random scatter with constant
#spread across the range of fitted values, homoscedasticity likely holds.
#If the plot shows a pattern, homoscedasticity likely does not hold.

#Here, there is not a random scatter with constant spread across the range of fitted values:
#Spread is not constant. The variance of the residuals increases as fitted values increase, meaning the
```

```
plot(lm.gen, which = 3)  # Scale-Location (check homoscedasticity)
```




```
#Homoscedasticity is indicated if the points are evenly distributed
#horizontally with no discernible pattern. A trend indicates non-constant
#variance.

#The scale-location plot has more scatter and more anomalies further as x increases, and the red line s
```

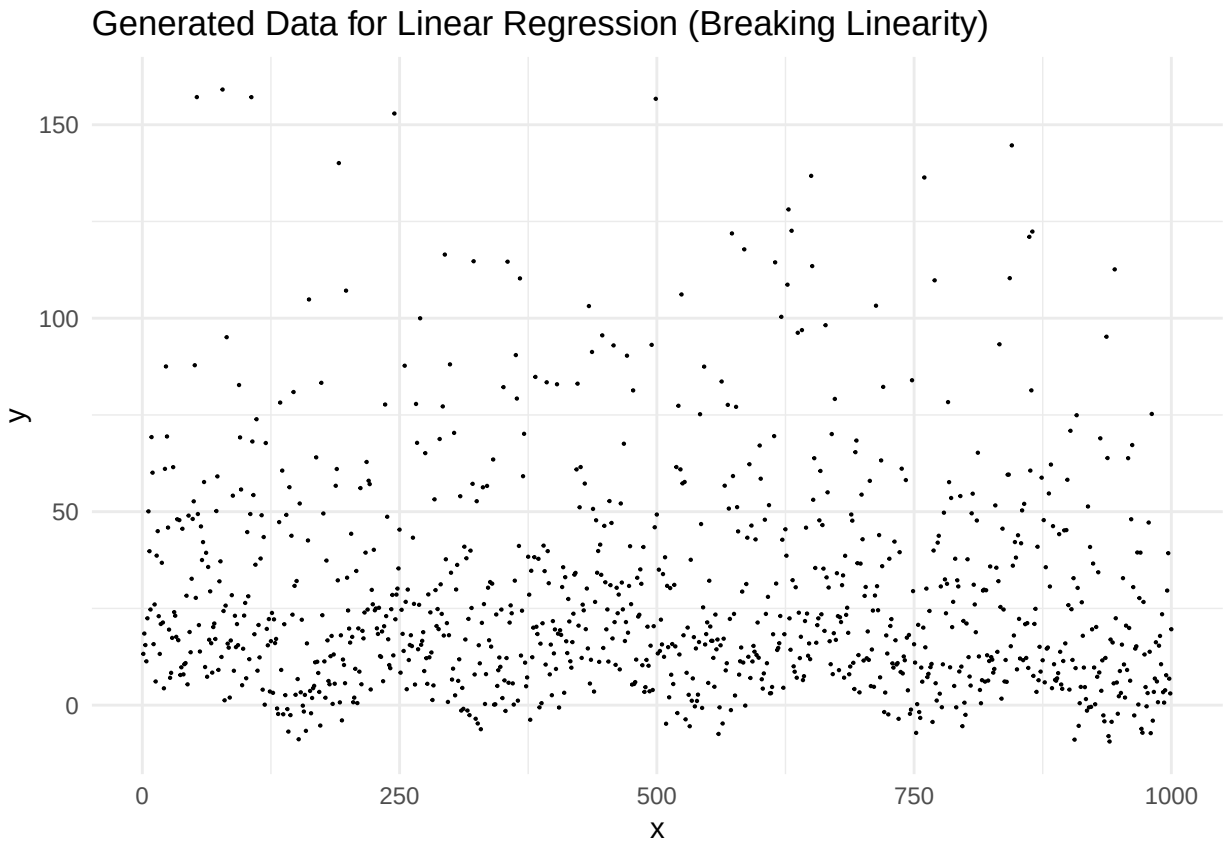
```
plot(lm.gen, which = 5)  # Residuals vs. Leverage (check influence)
```



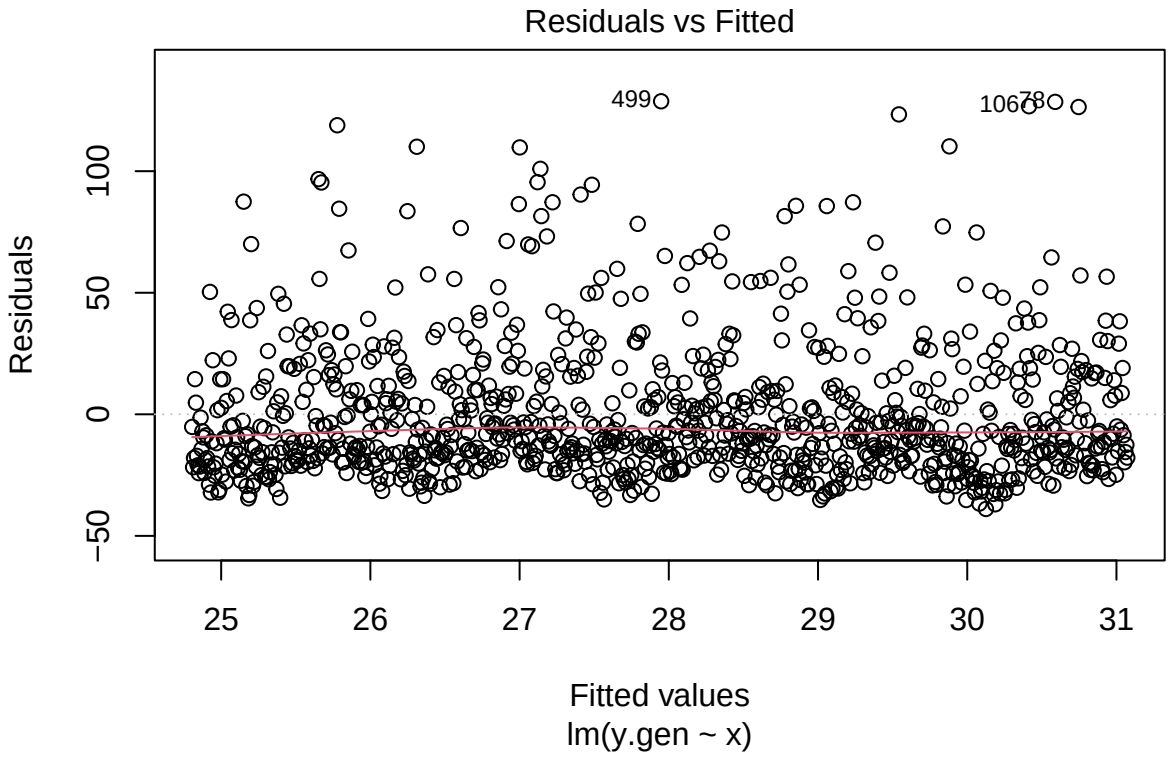

```
# If the spread of residuals changes with leverage, it suggests heteroscedasticity. The spread seems to
```

```
#Here, A model that Breaks the Linearity assumption:
set.seed(42)
n <- 1000
x <- 1:n

constant <- 0
trend <- 0
curve_magnitude <- 6 #Increased to create non-linearity, also among the residuals
curve_period <- 100
curve_shift <- 4  # A linear model should not have a curve shift.
normal_noise_magnitude <- 1
norm_noise_periode <- 10000
shift_norm_noise <- 500
non_normal_noise_magnitude <- 6 #Increased to create variability
non_norm_noise_periode <- 10000
shift_non_norm_noise <- 500.

y.gen <- constant +
  trend * x +
  curve_magnitude * sin((x/curve_period + curve_shift) * pi) +
  normal_noise_magnitude * cos((x/norm_noise_periode + shift_norm_noise/norm_noise_periode) * pi) * rno
  non_normal_noise_magnitude * cos((x/non_norm_noise_periode + shift_non_norm_noise/non_norm_noise_peri

p <- ggplot(data = data.frame(x = x, y = y.gen), aes(x = x, y = y)) +
  geom_point(size = 0.1) +
  labs(title = "Generated Data for Linear Regression (Breaking Linearity)", y = "y") +
  theme_minimal()
```

```r
print(p)
```



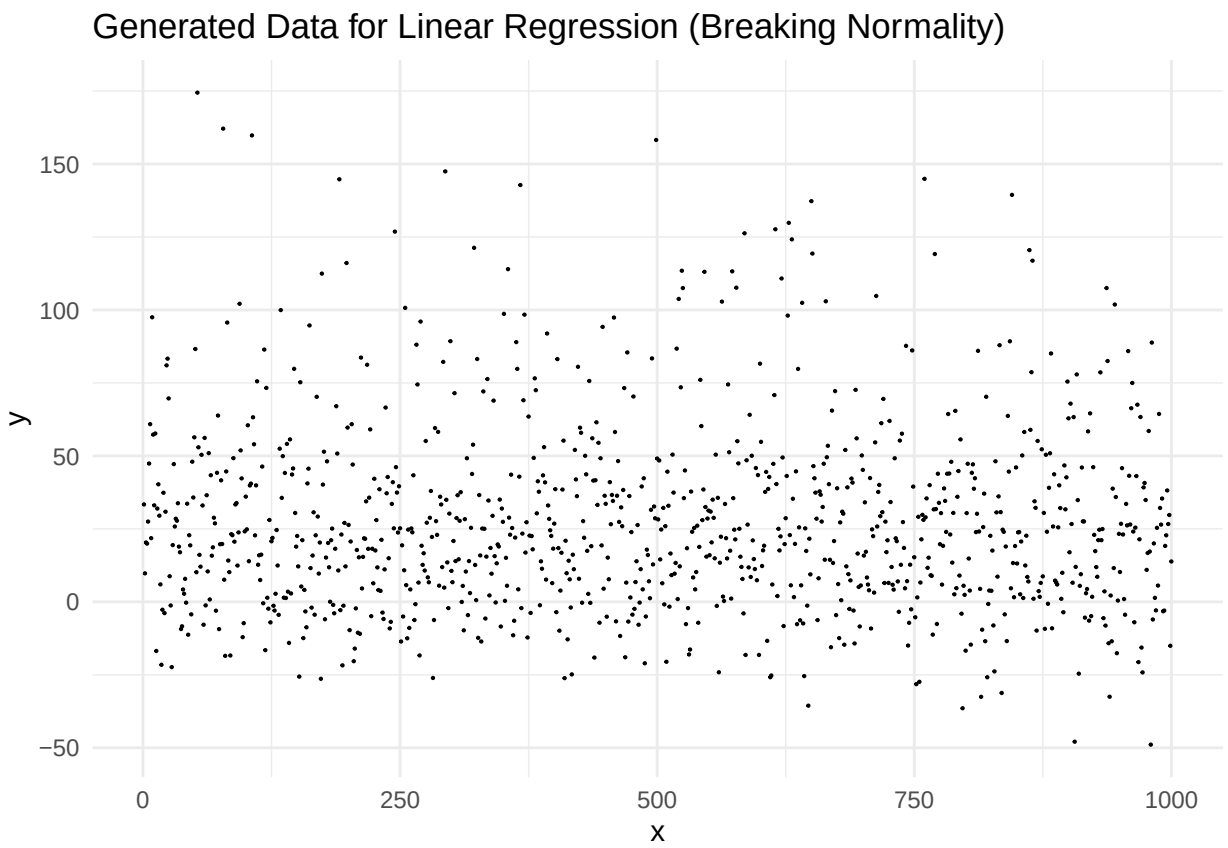

```r
#Diagnostics of model:
lm.gen <- lm(y.gen ~ x)

# Plot diagnostics
plot(lm.gen, which = 1)  # Residuals vs. Fitted (check linearity, homoscedasticity)
```



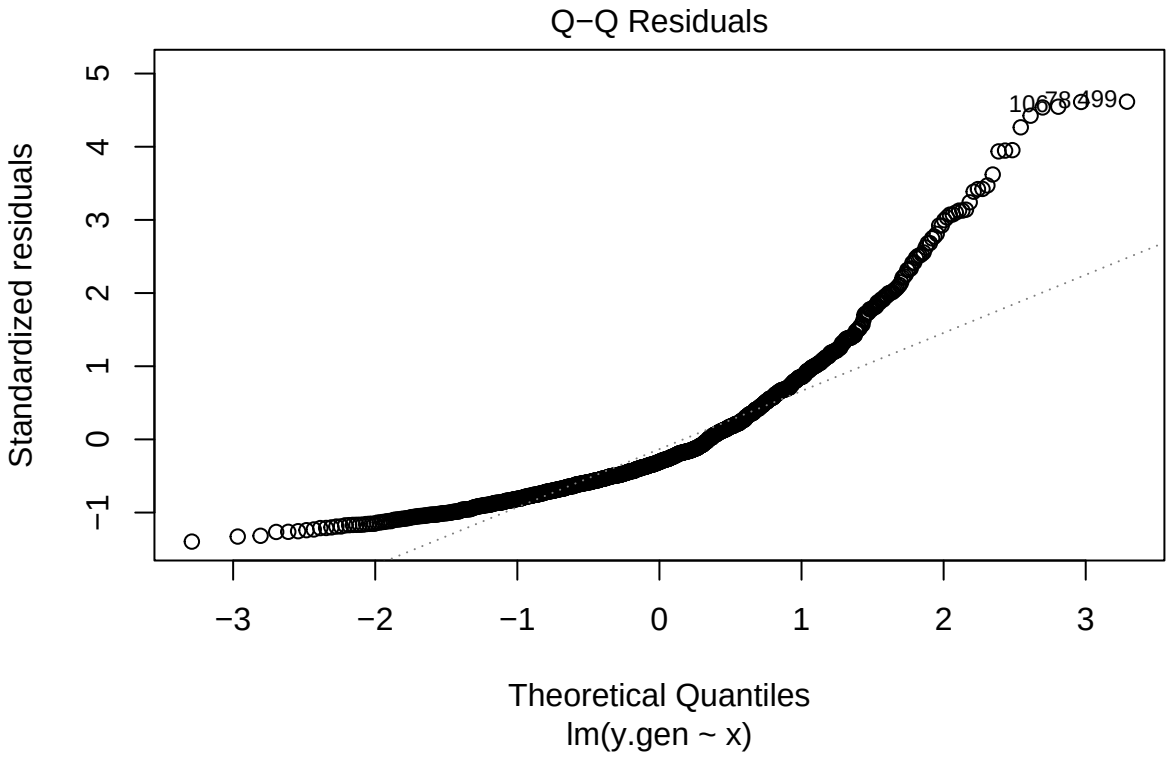

```
#Linearity:
##Diagnosed by: Residuals. Plotting the residuals against the fitted values
#or against each predictor. In a well-fitting linear model, residuals
#should be randomly scattered around 0 without any discernible pattern.

#More anomalies and larger group of residuals further away from the mean,
#which are no longer randomly scattered around 0.
```

```
#A model breaking the Normality assumption:
set.seed(42)
n <- 1000
x <- 1:n

constant <- 0
trend <- 0
curve_magnitude <- 0
curve_period <- 100
curve_shift <- 5  #There is no shift in the curve, essential for a linear model.
#If the QQ-plot is not a straight line (has a shift), normality
#assumption will break. So it is essential for this to hold.
normal_noise_magnitude <- 6 #The magnitude of the noise in the normal distribution.
#Too much noise may make it harder to interpret accurate results, and
#will not accurately represent normally distributed residuals.
#So this is essential for not breaking the normality assumption.
norm_noise_periode <- 10000
shift_norm_noise <- 500
non_normal_noise_magnitude <- 6 #Setting this non-zero would add a
#non-normal component to the residuals, breaking the normality assumption.
non_norm_noise_periode <- 10000
shift_non_norm_noise <- 500.
```

```r
y.gen <- constant +
  trend * x +
  curve_magnitude * sin((x/curve_period + curve_shift) * pi) +
  normal_noise_magnitude * cos((x/norm_noise_periode + shift_norm_noise/norm_noise_periode) * pi) * rno
  non_normal_noise_magnitude * cos((x/non_norm_noise_periode + shift_non_norm_noise/non_norm_noise_perio

p <- ggplot(data = data.frame(x = x, y = y.gen), aes(x = x, y = y)) +
  geom_point(size = 0.1) +
  labs(title = "Generated Data for Linear Regression (Breaking Normality)", y = "y") +
  theme_minimal()

print(p)
```



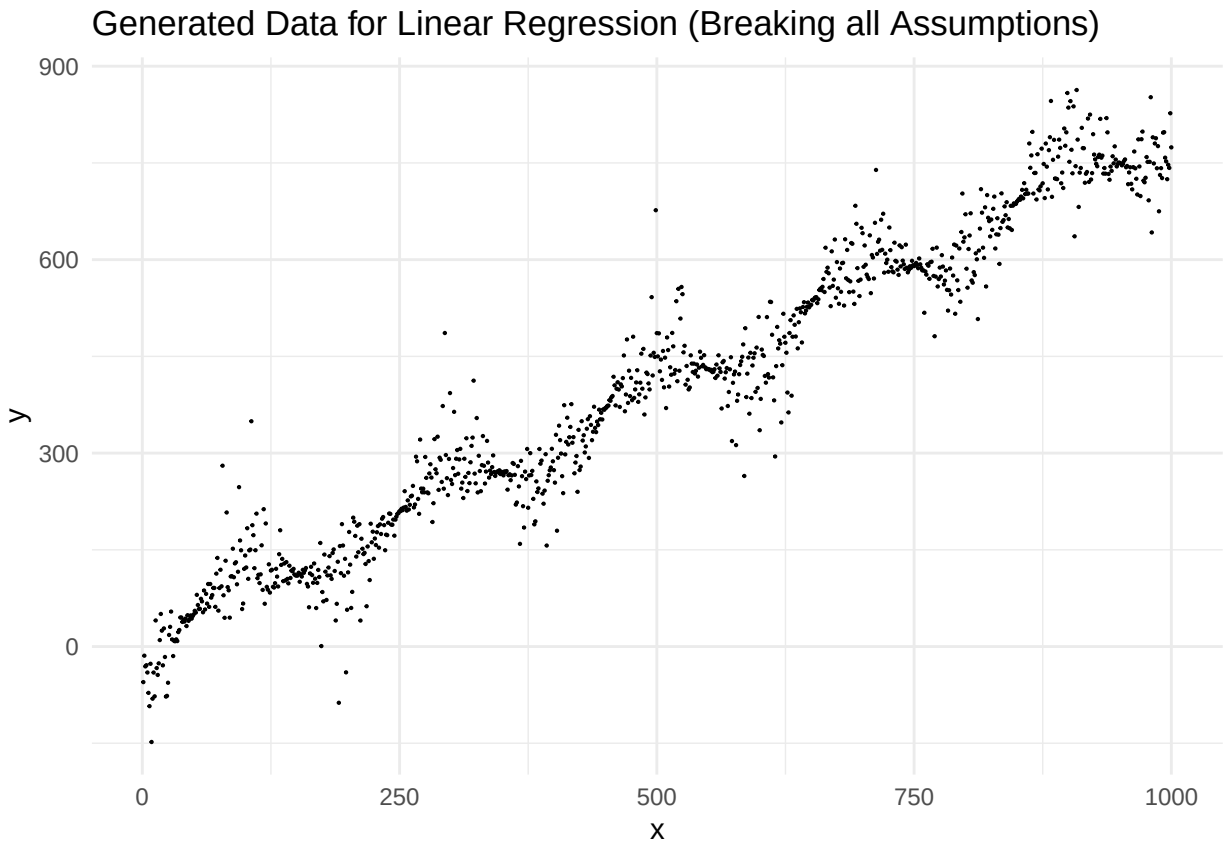

```r
#If the residuals are normally distributed, the
#points on the QQ-plot will lie approximately on a straight diagonal line.
#If the points deviate symmetrically from the line(e.g., forming an "S" shape),
#this suggests a violation of the normality assumption, such as
#skewness or heavy tails(kurtosis).

plot(lm.gen, which = 2)  # Q-Q plot
```



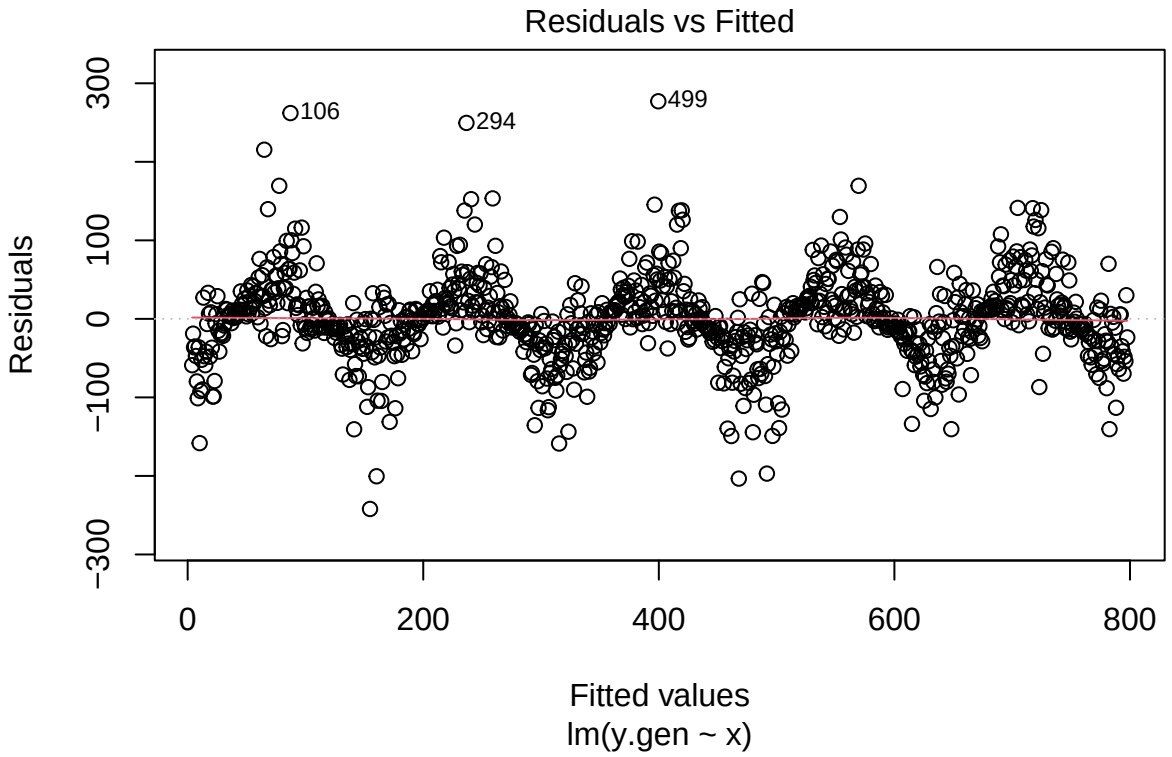

```
#Breaks normality as the line forms an S-shape, with kurtosis.
```

```
#A model breaking all assumptions including independence:
library(ggplot2)
set.seed(42)
n <- 1000
x <- 1:n

constant <- 0
trend <- 0.8 #A trend will indicate nonconstant variance among the residuals,
#so for homoscedasticity to hold it is set to 0.
#If the variance of the residuals is not constant, the residuals will fan out
#or contract as the fitted values increase. So both trend<0 and trend>0
#might contribute to non-constant
#variance.
curve_magnitude <- 10 #If a relationship between variables is curved (non-linear)
#and a linear model is fitted, the linear model will be a poor fit.
#In this case, the linearity assumption of a linear regression is violated
#When curve_magnitude = 0, it suggests there is no curvature, and the relationship
#between X and Y is linear.
#This would fit the assumptions of a linear model.
#A curve magnitude > 0 would leave a non-linear coefficient among the residuals.
#Also, as the relationship becomes more non-linear, which can lead to unequal variance
#in the residuals across the range of x. So it can also contribute to heteroscedasticity.
curve_period <- 100
curve_shift <- 10 #There is no shift in the curve, essential for a linear model.
#If the QQ-plot is not a straight line (has a shift), normality
#assumption will break. So it is essential for this to hold.
normal_noise_magnitude <- 10 #The magnitude of the noise in the normal distribution.
#Too much noise may make it harder to interpret accurate results, and
```

```r
#will not accurately represent normally distributed residuals.
#So this is essential for not breaking the normality assumption.
norm_noise_periode <- 100 #If residuals are not independent, we may see patterns
#over time, such as cycles or periodicity. This can be violated by adding periodic noise
#components, and setting this parameter to a small value can add periodic noise components.
#So if this is low, independence can break.
shift_norm_noise <- 500
non_normal_noise_magnitude <- 10 #Setting a value >0 here would add a
#non-normal component to the residuals, breaking the normality assumption.
#Increasing it can add more variability, so it could also
#contribute to heteroscedasticity.
non_norm_noise_periode <- 100 #Setting this to a small value can also add periodic noise
#components. So if this is low, independence can break.
shift_non_norm_noise <- 500.

y.gen <- constant +
  trend * x +
  curve_magnitude * sin((x/curve_period + curve_shift) * pi) +
  normal_noise_magnitude * cos((x/norm_noise_periode + shift_norm_noise/norm_noise_periode) * pi) * rno
  non_normal_noise_magnitude * cos((x/non_norm_noise_periode + shift_non_norm_noise/non_norm_noise_peri

# Use ggplot2 for the scatter plot
p <- ggplot(data = data.frame(x = x, y = y.gen), aes(x = x, y = y)) +
  geom_point(size = 0.1) +
  labs(title = "Generated Data for Linear Regression (Breaking all Assumptions)", y = "y") +
  theme_minimal()

print(p)
```

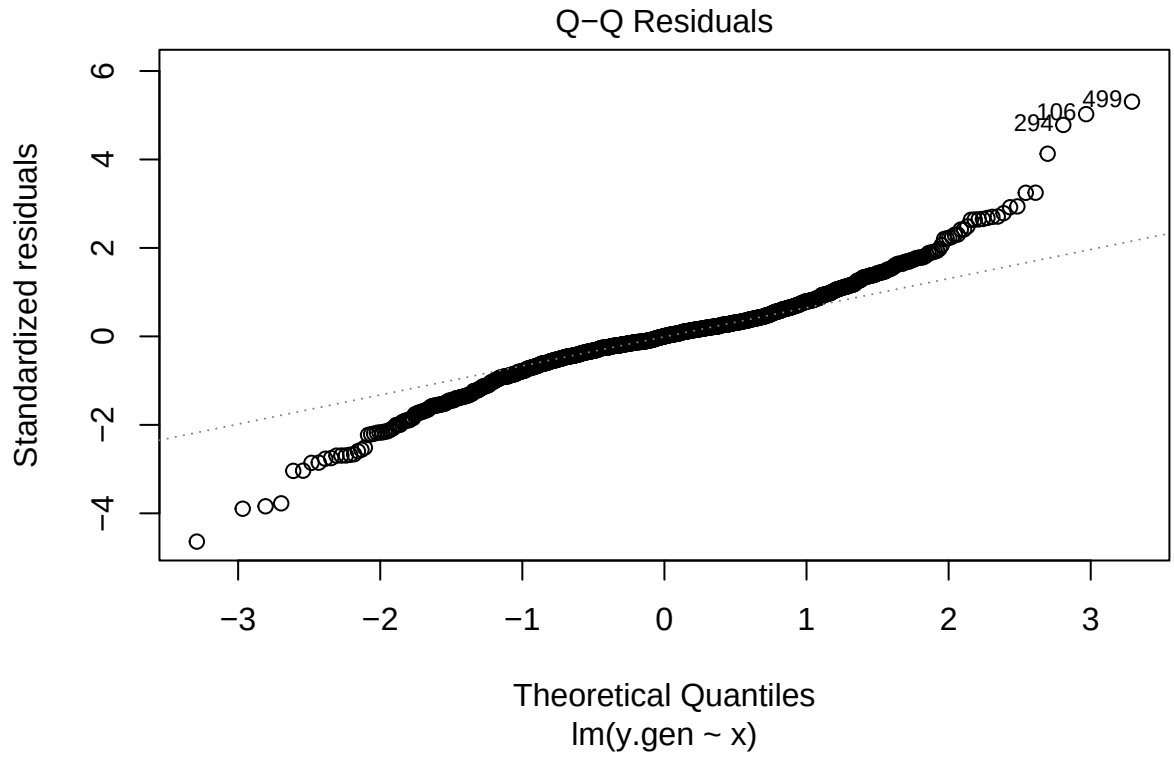



```r
#Diagnostics of model:
lm.gen <- lm(y.gen ~ x)

# Plot diagnostics
plot(lm.gen, which = 1)  # Residuals vs. Fitted (check linearity, homoscedasticity)
```



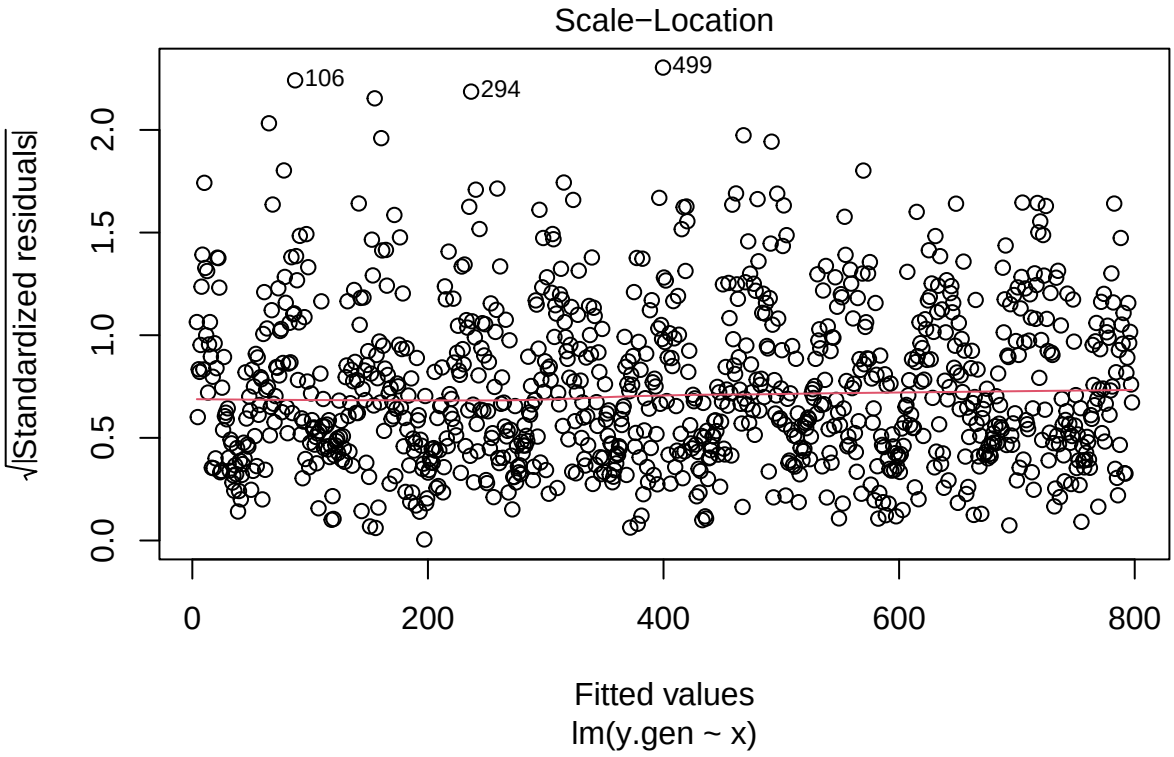

```
#Residuals are fairly placed around 0, but they are not randomly scattered as
#there are some big outliers and anomalies, breaking linearity.

#If the residuals show a random scatter with constant
#spread across the range of fitted values, homoscedasticity likely holds.
#If the plot shows a pattern, homoscedasticity likely does not hold.

#There is a clear jigsaw pattern in the residuals, breaking homoscedasticity.
```

```
#If the residuals are normally distributed, the
#points on the QQ-plot will lie approximately on a straight diagonal line.
#If the points deviate symmetrically from the line(e.g., forming an "S" shape),
#this suggests a violation of the normality assumption, such as
#skewness or heavy tails(kurtosis).

plot(lm.gen, which = 2)  # Q-Q plot
```




```
#Breaks normality as the line is skewed.
```

```
plot(lm.gen, which = 3)  # Scale-Location (check homoscedasticity)
```




```
#Homoscedasticity is indicated if the points are evenly distributed
#horizontally with no discernible pattern. A trend indicates non-constant
```

```
#variance.

#The scale-location plot has less scatter and there are more anomalies further away from the mean, so t
```

Exercise 4

Task a

The concept of correlation

By standard definition, correlation measures the degree of association between two continuous variables, represented by the correlation coefficient ρ. In this assignment, the Pearson correlation coefficient will be used. For Pearson correlation, the correlation coefficient ranges from -1 to 1 and indicates the linear relationship's direction and strength. There are general guidelines for interpreting the values of the correlation coefficient, and for two continuous variables X and Y, the following applies to Pearson correlation ("Pearson's Correlation Coefficient," n.d.):

- $|\rho_{XY}| \leq 0.29$: Weak correlation
- $|\rho_{XY}| = 0.30$ to 0.49: Moderate correlation
- $|\rho_{XY}| \geq 0.50$: Strong correlation

Hence, $\rho_{XY} = 1$ indicates a strong positive correlation, ρ_{XY} = -1 indicates a strong negative correlation, and $\rho_{XY} = 0$ indicates no correlation. (Note that r is used instead of ρ when analyzing sample data, using Pearson correlation.)

Impact of scaling

According to the given formulas, $\rho_{W_1W_2}$ can be written as follows:

$$\rho_{W_1W_2} = \frac{\text{cov}(\alpha_1 X, \alpha_2 Y)}{\sqrt{\text{var}(\alpha_1 X)\text{var}(\alpha_2 Y)}}$$
$$\rho_{W_1W_2} = \text{sgn}(\alpha_1\alpha_2)\rho_{XY}$$

As the formulas indicate, scaling the different variables X and Y might change the sign of $\rho_{W_1W_2}$, but not the magnitude. The sign depends on the product $\alpha_1\alpha_2$, and $\text{sgn}(\alpha_1\alpha_2)$ can flip from positive to negative or negative to positive, based on the signs of α_1 and α_2. Hence, scaling of the different variables X and Y may affect the direction of the correlation, but not the magnitude.

Task b

By standard definition, partial correlation measures the degree of association between two continuous variables, while controlling for the influence of one or more other continuous variables. These variables are called confounding variables and are associated with either the independent variable, the dependent variable, or both. Usually, the confounding variable will influence both the dependent and the independent variable. Like regular correlation, partial correlation is represented by the correlation coefficient ρ (r is used for sample data), which ranges from -1 to 1. The interpretation of ρ is the same for partial correlation as it is for regular correlation. In the formula provided, Z acts as a confounding variable, while X and Y serve as the main variables. If regular correlation suggests that X and Y are correlated, partial correlation can be used to determine whether this correlation is driven by the confounding variable Z. Thus, using the regular correlation coefficient may produce misleading results, and partial correlation addresses this issue by removing the influence of such confounding variables. The formula for partial correlation is:

$$\rho_{XY|Z} = \frac{\rho_{XY} - \rho_{XZ}\rho_{ZY}}{\sqrt{(1-\rho_{XZ}^2)(1-\rho_{ZY}^2)}}$$

The formula computes the correlation between X and Y after controlling for the effect of Z. The numerator adjusts the correlation between X and Y by subtracting the influence of Z on both variables. The denominator

ensures that $\rho_{XY|Z}$ is properly scaled (remains within the bounds of a correlation coefficient) and adjusts for the amount of correlation between X and Y that cannot be explained by Z.

Examples of scenarios to investigate partial correlations

Scenario 1 - Causes of Lung Cancer as the dependent variable: When studying the causes of cancer, one might find a correlation between alcohol consumption and lung cancer. However, this relation might be confounded by smoking, as smoking is associated with alcohol consumption and is a well-known risk factor for lung cancer. In this case, smoking serves as a confounder that could disrupt the relationship between alcohol and lung cancer. By setting x = alcohol consumption, y = lung cancer, and z = number of cigarettes per day, one can calculate the partial correlation which adjusts for the effect of smoking on both variables.

Scenario 2 - Exam Performance as the dependent variable: When analyzing the factors that influence exam performance, intelligence may act as a confounding variable. Intelligence could influence both study hours and exam performance, thereby affecting the true relationship between the two. By setting x = study hours, y = exam performance, and z = intelligence (e.g. measured by IQ), one can use the given formula to calculate the partial correlation which adjusts for intelligence's influence on both variables.

Scenario 3 - Heart health as the dependent variable: In a study investigating the relationship between heart health and exercise, age could act as a confounder because it affects both exercise habits and heart health. By setting x = heart health, y = hours of exercise and z = age, one can use the given formula to calculate. This way, one can better isolate the effect of exercise on heart health.

Task c

The property from part (a) indicates that scaling variables X and Y by constants does not affect the magnitude of the correlation, only the sign. To investigate whether this property holds for partial correlation, one can apply the scaling factors α_1 for X, α_2 for Y, and α_3 for Z from part (a) in the equation given in part (b), see equation *.

$$(*) \quad \rho_{XY|Z} = \frac{\rho_{XY} - \rho_{XZ}\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}}$$

Adjusting the formula given in (*) to account for these scaling factors yields the following:

$$\rho_{(\alpha_1 X)(\alpha_2 Y)} = \operatorname{sgn}(\alpha_1\alpha_2)\rho_{XY} \tag{1}$$
$$\rho_{(\alpha_1 X)(\alpha_3 Z)} = \operatorname{sgn}(\alpha_1\alpha_3)\rho_{XZ} \tag{2}$$
$$\rho_{(\alpha_2 Y)(\alpha_3 Z)} = \operatorname{sgn}(\alpha_2\alpha_3)\rho_{ZY} \tag{3}$$

After scaling, the partial correlation formula becomes:

$$
\begin{aligned}
\rho_{(\alpha_1 X)(\alpha_2 Y)|(\alpha_3 Z)} &= \frac{\operatorname{sgn}(\alpha_1\alpha_2)\rho_{XY} - \operatorname{sgn}(\alpha_1\alpha_3)\rho_{XZ}\cdot\operatorname{sgn}(\alpha_2\alpha_3)\rho_{ZY}}{\sqrt{1-(\operatorname{sgn}(\alpha_1\alpha_3)\rho_{XZ})^2}\sqrt{1-(\operatorname{sgn}(\alpha_2\alpha_3)\rho_{ZY})^2}} \\
&= \frac{\operatorname{sgn}(\alpha_1\alpha_2)\rho_{XY} - \operatorname{sgn}(\alpha_1)\cdot\operatorname{sgn}(\alpha_3)\rho_{XZ}\cdot\operatorname{sgn}(\alpha_2)\cdot\operatorname{sgn}(\alpha_3)\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}} \\
&= \frac{\operatorname{sgn}(\alpha_1\alpha_2)\rho_{XY} - \operatorname{sgn}(\alpha_3)\cdot\operatorname{sgn}(\alpha_3)\cdot\operatorname{sgn}(\alpha_1)\rho_{XZ}\cdot\operatorname{sgn}(\alpha_2)\cdot\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}} \\
&= \frac{\operatorname{sgn}(\alpha_1\alpha_2)\rho_{XY} - 1\cdot\operatorname{sgn}(\alpha_1)\cdot\rho_{XZ}\cdot\operatorname{sgn}(\alpha_2)\cdot\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}} \\
&= \frac{\operatorname{sgn}(\alpha_1\alpha_2)\rho_{XY} - \operatorname{sgn}(\alpha_1)\cdot\rho_{XZ}\cdot\operatorname{sgn}(\alpha_2)\cdot\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}} \\
&= \frac{\operatorname{sgn}(\alpha_1\alpha_2)\rho_{XY} - \operatorname{sgn}(\alpha_1\alpha_2)\cdot\rho_{XZ}\cdot\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}} \\
&= \frac{\operatorname{sgn}(\alpha_1\alpha_2)(\rho_{XY} - \rho_{XZ}\cdot\rho_{ZY})}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}}
\end{aligned}
$$

Which can be simplified to:

$$
\rho_{(\alpha_1 X)(\alpha_2 Y)|(\alpha_3 Z)} = \operatorname{sgn}(\alpha_1\alpha_2)\cdot\frac{\rho_{XY} - \rho_{XZ}\cdot\rho_{ZY}}{\sqrt{1-\rho_{XZ}^2}\sqrt{1-\rho_{ZY}^2}}
$$

Hence, the principle from task (a) also applies to partial correlation. The formula for partial correlation is based on correlations ρ_{XY}, ρ_{XZ}, ρ_{ZY}, which are unaffected by scaling. Therefore, scaling X or Y in the presence of a third variable Z might change the sign, but not the magnitude of the partial correlation.

Task d

```r
# Load necessary packages
library(dplyr)
library(ppcor)
```

```r
# Reading the data from the weatherHistory csv-file
data <- read.csv("weatherHistory.csv")

# Selecting the required features: Temperature (C), Apparent Temperature (C), Humidity
selected_data <-
data %>% dplyr::select(Temperature..C., Apparent.Temperature..C., Humidity)

# Compute the pairwise Pearson correlation of each pair of features
pairwise_correlation <- cor(selected_data)

# Display the pairwise Pearson correlation
cat("Pairwise Pearson correlation")
```

```
## Pairwise Pearson correlation
```

```r
print(pairwise_correlation)
```

```
##                           Temperature..C. Apparent.Temperature..C.   Humidity
## Temperature..C.                 1.0000000                 0.9926286 -0.6322547
## Apparent.Temperature..C.        0.9926286                 1.0000000 -0.6025710
```

```
## Humidity                      -0.6322547               -0.6025710  1.0000000
```

```r
# Compute partial Pearson correlation

# Extract the necessary columns
temperature <- selected_data[, "Temperature..C."]
apparent_temp <- selected_data[, "Apparent.Temperature..C."]
humidity <- selected_data[, "Humidity"]

# Create a Data Frame with the features
data_to_analyze <- data.frame(temperature, apparent_temp, humidity)

# Compute partial Pearson correlations
partial_corr_results <- pcor(data_to_analyze)

# Extract partial Pearson correlations from the result
partial_corr_temp_app_hum <- partial_corr_results$estimate["temperature", "apparent_temp"]
partial_corr_temp_hum_app <- partial_corr_results$estimate["temperature", "humidity"]
partial_corr_app_hum_temp <- partial_corr_results$estimate["apparent_temp", "humidity"]

# Store the results in a list
ppcor_partial_corr_results <- list(
  'Partial Pearson Correlation (Temp & App Temp | Humidity)' = partial_corr_temp_app_hum,
  'Partial Pearson Correlation (Temp & Humidity | App Temp)' = partial_corr_temp_hum_app,
  'Partial Pearson Correlation (App Temp & Humidity | Temp)' = partial_corr_app_hum_temp
)

# Display the results
cat("Partial Pearson Correlation")
```

```
## Partial Pearson Correlation
```

```r
print(ppcor_partial_corr_results)
```

```
## $`Partial Pearson Correlation (Temp & App Temp | Humidity)`
## [1] 0.9892298
##
## $`Partial Pearson Correlation (Temp & Humidity | App Temp)`
## [1] -0.3528185
##
## $`Partial Pearson Correlation (App Temp & Humidity | Temp)`
## [1] 0.2664916
```

The relationship between the variables: Do the variables have any confounding effect on each other?

When z = Humidity:

Both the pairwise correlation and the partial correlation between the temperature and the apparent temperature are strong. Hence, the humidity variable does not confound this relationship significantly.

When z = Apparent temperature:

Both the pairwise and the partial correlations are negative. However, the absolute value of the pairwise correlation is higher than the absolute value of the partial correlation. The pairwise correlation can be classified as strong negative, while the partial correlation is weak negative, indicating that the apparent temperature variable has a confounding effect.

When z = Temperature:

The pairwise correlation shows a strong negative relationship, while the partial correlation shows a weak positive relationship. The change from a strong negative pairwise correlation to a weak positive partial correlation indicates that the temperature variable is confounding the relationship between the humidity and the apparent temperature.

References

Pearson's correlation coefficient: A comprehensive overview. (n.d.). In *Statistics Solutions*. Retrieved September 27, 2024, from https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/pearsons-correlation-coefficient/