Lab 3

Yubo Rasmussen

## Lab 3

Today we are going to explore “Factors in R” and “Linear Regression”.

## Factors

First load the following libraries:

library(forcats)

Warning: package 'forcats' was built under R version 4.2.3

library(ggplot2)

Warning: package 'ggplot2' was built under R version 4.2.3

library(tidyverse)

Warning: package 'tidyverse' was built under R version 4.2.3

Warning: package 'tibble' was built under R version 4.2.3

Warning: package 'tidyr' was built under R version 4.2.3

Warning: package 'readr' was built under R version 4.2.3

Warning: package 'purrr' was built under R version 4.2.3

Warning: package 'dplyr' was built under R version 4.2.3

Warning: package 'stringr' was built under R version 4.2.3

Warning: package 'lubridate' was built under R version 4.2.3

── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
✔ dplyr 1.1.4 ✔ stringr 1.5.1  
✔ lubridate 1.9.3 ✔ tibble 3.2.1  
✔ purrr 1.0.2 ✔ tidyr 1.3.1  
✔ readr 2.1.4   
── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
✖ dplyr::filter() masks stats::filter()  
✖ dplyr::lag() masks stats::lag()  
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

### Sorting a vector

Here is a vector with the names of some months, which we want to sort, but can’t use the sort function.

# Vector with the names of some months:  
x1 <- c("Dec", "Apr", "Jan", "Mar")  
# We want to sort this vector but sort()  
# will not produce the desired outcome in this case  
sort(x1)

[1] "Apr" "Dec" "Jan" "Mar"

Instead, we need to do this:

# Step 1:  
month\_levels <- c(  
 "Jan", "Feb", "Mar", "Apr", "May", "Jun",  
 "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"  
)  
# Step 2:  
y1 <- factor(x1, levels = month\_levels)  
y1

[1] Dec Apr Jan Mar  
Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

# Now sort will produce the outcome we want:  
sort(y1)

[1] Jan Mar Apr Dec  
Levels: Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

# The function levels() provides access the levls of a factor.  
levels(y1)

[1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"

### Star Wars Example

Star Wars Data: A summary of the frequency of each species in the star wars universe.

First attach the star wars dataframe from dplyr

attach(starwars)

Suppose that we want to a summary of the frequency of each species in the star wars universe.

# Filter out the rows with NA values  
# groub by their species and count their quantities   
starwars %>% filter(!is.na(species)) %>%   
 group\_by(species) %>%   
 summarise(n = n()) %>% head(20)

# A tibble: 20 × 2  
 species n  
 <chr> <int>  
 1 Aleena 1  
 2 Besalisk 1  
 3 Cerean 1  
 4 Chagrian 1  
 5 Clawdite 1  
 6 Droid 6  
 7 Dug 1  
 8 Ewok 1  
 9 Geonosian 1  
10 Gungan 3  
11 Human 35  
12 Hutt 1  
13 Iktotchi 1  
14 Kaleesh 1  
15 Kaminoan 2  
16 Kel Dor 1  
17 Mirialan 2  
18 Mon Calamari 1  
19 Muun 1  
20 Nautolan 1

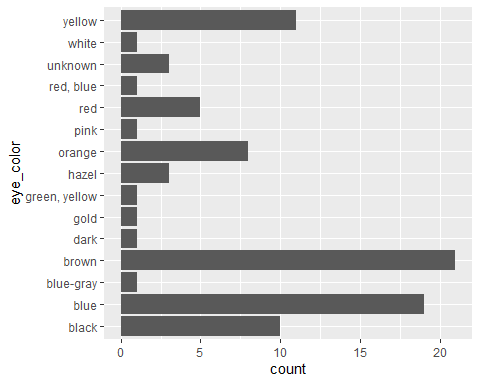
We observe that several species have very low frequency so it would make sense to lump them together using the fct\_lump() function and create a new group called “other” which, for example, will include all the species with frequency less than 3.

# The fct\_lump() function for classifying species with   
#frequency less than 3 into a new group called "other":  
starwars %>%  
 filter(!is.na(species)) %>%  
 mutate(species = fct\_lump(species, n = 3)) %>%  
 group\_by(species) %>%   
 summarise(n = n())

# A tibble: 4 × 2  
 species n  
 <fct> <int>  
1 Droid 6  
2 Gungan 3  
3 Human 35  
4 Other 39

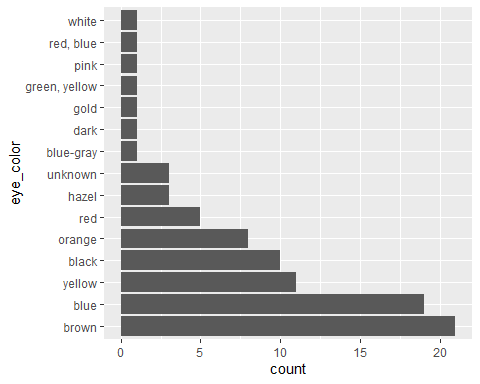
Suppose now, that we want to visualize the frequency of eye colors of all the characters in the star wars universe.

# Visualize the frequency of eye colors of   
# all the characters in the star wars universe:  
ggplot(starwars,aes(y = eye\_color)) + geom\_bar()



Also, we can order the bars in the previous chart according to their frequency using the fct\_infreq() function:

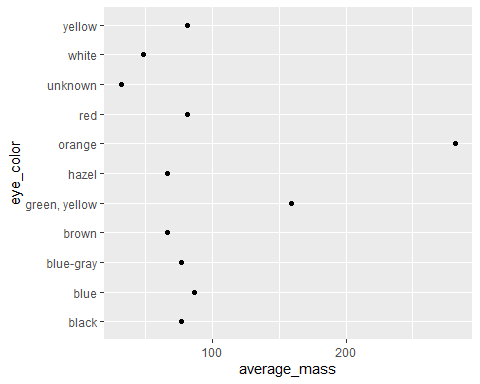
# Order the bars according to their frequency:  
starwars %>%  
 mutate(eye\_color = fct\_infreq(eye\_color)) %>%  
 ggplot(aes(y = eye\_color)) + geom\_bar()



Finally, we may want to visualize the summary statistics of a certain variable based on a specific categorical feature.

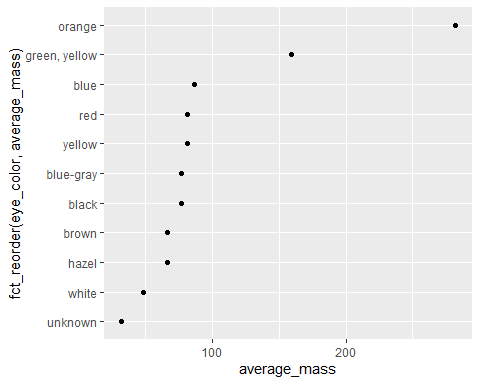
For example, it is interesting to see how average mass of the species in the star wars universe varies across different eye colors.

# Average mass of star wars specicies based on different eye\_colors:  
starwars %>% filter(!is.na(mass)) %>%   
 group\_by(eye\_color) %>%   
 summarise(average\_mass = mean(mass)) %>%   
 ggplot(aes(x = average\_mass,y= eye\_color)) + geom\_point() # creates scatter plots



Finally, we can reorder factor levels by sorting along another variable using the fct\_reorder() function. As an example we will reorder the levels of the variable eye color accoridng to the variable average mass:

# Reorder the levels of the variable eye\_color  
# accoridng to the variable average\_mass:  
starwars %>% filter(!is.na(mass)) %>%   
 group\_by(eye\_color) %>%   
 summarise(average\_mass = mean(mass)) %>%   
 ggplot(aes(x = average\_mass,y= fct\_reorder(eye\_color,average\_mass))) + geom\_point()



## Linear Regression

Linear Regression is a supervised machine learning algorithm which is used to predict the value of an outcome variable, or response, , based on one or more input features, or explanatory, variables .

Suppose that is the th observation of the dependent variable and is th observation of the th independent variable, for and Then, the basic model for multiple linear regression is and the response variable .

where is an i.i.d. random variable.

A multiple linear regression can be fitted with the lm() function in R. In our numerical analysis, we will demonstrate linear regression in a simple and easy to understand fashion by using the built-in data set “cars” that comes with R by default.

Response variable :

* mpg (miles/gallon)

Explanatory Variables :

* cyl (Number of cylinders),
* hp (gross horsepower) and
* wt (weight).

data(mtcars) # load the data from R  
help(mtcars) # see the online documentation for data description

starting httpd help server ... done

head(mtcars) # view the first few rows of the entire data set.

mpg cyl disp hp drat wt qsec vs am gear carb  
Mazda RX4 21.0 6 160 110 3.90 2.620 16.46 0 1 4 4  
Mazda RX4 Wag 21.0 6 160 110 3.90 2.875 17.02 0 1 4 4  
Datsun 710 22.8 4 108 93 3.85 2.320 18.61 1 1 4 1  
Hornet 4 Drive 21.4 6 258 110 3.08 3.215 19.44 1 0 3 1  
Hornet Sportabout 18.7 8 360 175 3.15 3.440 17.02 0 0 3 2  
Valiant 18.1 6 225 105 2.76 3.460 20.22 1 0 3 1

We will investigate the relationship between the response mpg and features: cyl, hp and wt.

# Fit a multivariate linear regession model:  
LM <- lm(mpg ~ cyl + hp + wt, data = mtcars)  
  
# Finally, we’ll get the summary output using the summary() function:  
summary(LM)

Call:  
lm(formula = mpg ~ cyl + hp + wt, data = mtcars)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-3.9290 -1.5598 -0.5311 1.1850 5.8986   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) 38.75179 1.78686 21.687 < 2e-16 \*\*\*  
cyl -0.94162 0.55092 -1.709 0.098480 .   
hp -0.01804 0.01188 -1.519 0.140015   
wt -3.16697 0.74058 -4.276 0.000199 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 2.512 on 28 degrees of freedom  
Multiple R-squared: 0.8431, Adjusted R-squared: 0.8263   
F-statistic: 50.17 on 3 and 28 DF, p-value: 2.184e-11

The estimated regression coefficients , are shown in the ‘Estimate’ column.

The fitted linear model is:

Look at the significance codes. What do they tell us?

Copy and paste the results into ChatGPT and ask it to interpret it for us. Do you agree with what it says?

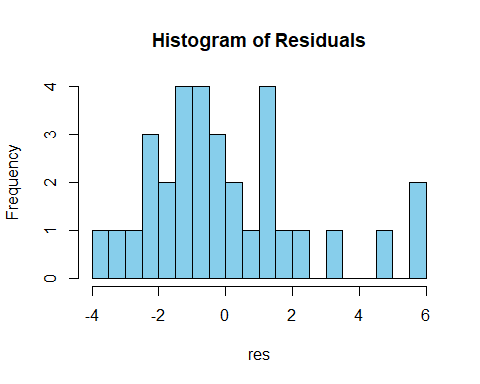
**“Heavier cars with more cylinders and higher horsepower generally use more fuel. But in this model, weight is the key factor driving mpg differences.”**

### Residuals

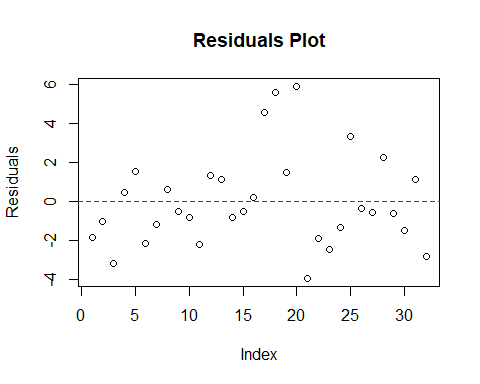
A good way to test the quality of the fit of the model is via a graphical investigation of the residuals , which are the differences between the real values and the predicted values .

To access the residuals of a linear model, use residuals() function:

# Extract residuals   
res <- residuals(LM)   
  
# Histogram and simple plot of residuals  
hist(res, breaks = 20, col = "skyblue", main = "Histogram of Residuals")



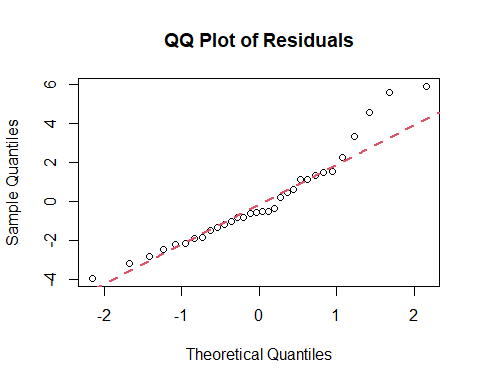
plot(res, main = "Residuals Plot", ylab = "Residuals")  
abline(h = 0, col = "red", lty = 2)



# Summary  
summary(res)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
-3.9290 -1.5598 -0.5311 0.0000 1.1850 5.8986

# QQ plot  
qqnorm(res, main = "QQ Plot of Residuals")  
qqline(res, col = 2, lwd = 2, lty = 2)



The pattern of residuals shown in plot should exhibit a random pattern (fluctuate around 0).

Quantile-Quantile (QQ) plots) let you check that the data meet the assumption of normality. In particular, Q-Q plots take your sample data, sort it in ascending order, and then plot them versus the quantiles calculated from a theoretical distribution, in this case the normal distribution. If the residuals are from a normal population, they should fall close to the straight line.

### Goodness-of-fit

We can also look at and (adjusted ) from summary(fitted LM). Their formulas are:

* is defined as the proportion of the total variability explained by the regression model.
* One problem with this is that it invariably increases as you add more explanatory variables to your model. This encourages adding extra useless features.
* The adjusted helps to mitigate this issue by penalizing additional variables.

## Case Study: A Motor Third-party Liability Policies

Load the data from the previous lab our data-set on MTPL claim counts.

library(dplyr)  
dat <- read.csv("freMTPL2freq.csv") %>%  
 # Set data types for variables  
 dplyr::mutate(across(c(Area, VehPower, VehGas, VehBrand, Region), as.factor),  
 across(c(Density, VehAge, DrivAge, BonusMalus), as.numeric)) %>%  
 na.omit()

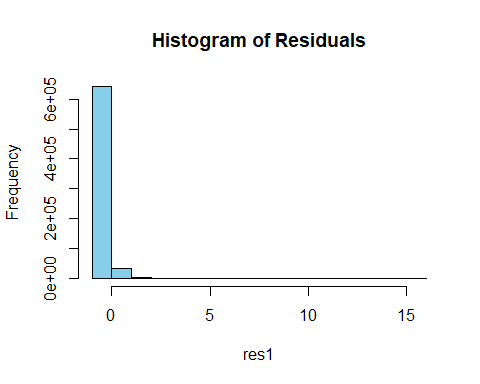
1. Fit a linear regression model with ClaimNb as the response variable.

# Fit a multivariate linear regession model:  
model1 <- lm(ClaimNb ~ Area + VehPower + VehGas + VehBrand + Region  
 + Density + VehAge + DrivAge + BonusMalus, data = dat)  
summary(model1)

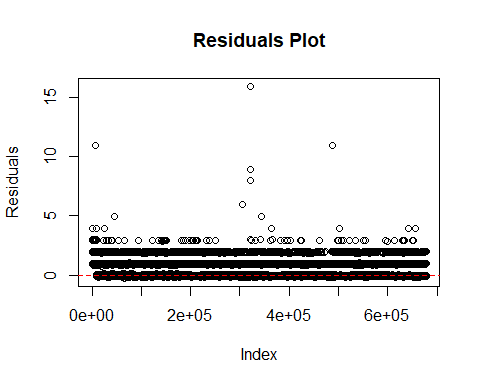
Call:  
lm(formula = ClaimNb ~ Area + VehPower + VehGas + VehBrand +   
 Region + Density + VehAge + DrivAge + BonusMalus, data = dat)  
  
Residuals:  
 Min 1Q Median 3Q Max   
-0.2875 -0.0634 -0.0501 -0.0376 15.9713   
  
Coefficients:  
 Estimate Std. Error t value Pr(>|t|)   
(Intercept) -5.510e-02 2.673e-03 -20.613 < 2e-16 \*\*\*  
AreaB 2.024e-03 1.160e-03 1.744 0.081073 .   
AreaC 2.908e-03 9.637e-04 3.018 0.002546 \*\*   
AreaD 6.439e-03 1.053e-03 6.114 9.69e-10 \*\*\*  
AreaE 6.679e-03 1.424e-03 4.690 2.73e-06 \*\*\*  
AreaF 7.993e-03 5.138e-03 1.556 0.119790   
VehPower5 9.556e-03 1.006e-03 9.500 < 2e-16 \*\*\*  
VehPower6 1.018e-02 9.884e-04 10.298 < 2e-16 \*\*\*  
VehPower7 5.351e-03 9.644e-04 5.548 2.89e-08 \*\*\*  
VehPower8 -3.251e-03 1.350e-03 -2.407 0.016064 \*   
VehPower9 1.250e-02 1.569e-03 7.969 1.60e-15 \*\*\*  
VehPower10 1.199e-02 1.555e-03 7.709 1.27e-14 \*\*\*  
VehPower11 5.785e-03 1.933e-03 2.993 0.002765 \*\*   
VehPower12 6.470e-04 2.793e-03 0.232 0.816814   
VehPower13 6.252e-04 4.384e-03 0.143 0.886588   
VehPower14 4.704e-03 5.026e-03 0.936 0.349320   
VehPower15 -7.194e-03 4.500e-03 -1.599 0.109871   
VehGasRegular 6.147e-03 6.344e-04 9.690 < 2e-16 \*\*\*  
VehBrandB10 -6.285e-04 1.971e-03 -0.319 0.749829   
VehBrandB11 2.423e-03 2.218e-03 1.092 0.274787   
VehBrandB12 -3.051e-03 9.870e-04 -3.091 0.001994 \*\*   
VehBrandB13 1.189e-03 2.269e-03 0.524 0.600323   
VehBrandB14 -5.925e-03 3.824e-03 -1.549 0.121351   
VehBrandB2 -5.178e-05 8.483e-04 -0.061 0.951327   
VehBrandB3 -1.687e-03 1.208e-03 -1.397 0.162462   
VehBrandB4 -2.033e-03 1.634e-03 -1.244 0.213485   
VehBrandB5 3.673e-03 1.430e-03 2.569 0.010193 \*   
VehBrandB6 -1.969e-03 1.547e-03 -1.273 0.203119   
RegionR21 3.494e-03 4.499e-03 0.777 0.437444   
RegionR22 3.736e-03 2.902e-03 1.287 0.198023   
RegionR23 -1.403e-02 2.784e-03 -5.041 4.64e-07 \*\*\*  
RegionR24 1.224e-02 1.370e-03 8.934 < 2e-16 \*\*\*  
RegionR25 1.009e-02 2.570e-03 3.925 8.67e-05 \*\*\*  
RegionR26 -7.681e-04 2.612e-03 -0.294 0.768675   
RegionR31 -8.527e-03 1.824e-03 -4.676 2.93e-06 \*\*\*  
RegionR41 -3.025e-03 2.400e-03 -1.261 0.207420   
RegionR42 7.176e-03 5.218e-03 1.375 0.169068   
RegionR43 -6.295e-03 6.675e-03 -0.943 0.345607   
RegionR52 5.118e-03 1.673e-03 3.060 0.002215 \*\*   
RegionR53 1.748e-02 1.669e-03 10.473 < 2e-16 \*\*\*  
RegionR54 4.093e-03 2.106e-03 1.943 0.051957 .   
RegionR72 -6.402e-03 1.754e-03 -3.649 0.000263 \*\*\*  
RegionR73 -9.127e-03 2.140e-03 -4.264 2.01e-05 \*\*\*  
RegionR74 1.317e-02 3.743e-03 3.519 0.000433 \*\*\*  
RegionR82 9.153e-03 1.375e-03 6.657 2.81e-11 \*\*\*  
RegionR83 -1.259e-02 3.484e-03 -3.613 0.000303 \*\*\*  
RegionR91 -6.955e-03 1.710e-03 -4.068 4.75e-05 \*\*\*  
RegionR93 -2.324e-03 1.383e-03 -1.680 0.092882 .   
RegionR94 2.300e-03 3.751e-03 0.613 0.539735   
Density 5.980e-08 2.152e-07 0.278 0.781131   
VehAge -1.493e-03 5.938e-05 -25.147 < 2e-16 \*\*\*  
DrivAge 7.533e-04 2.379e-05 31.657 < 2e-16 \*\*\*  
BonusMalus 1.137e-03 2.152e-05 52.836 < 2e-16 \*\*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
  
Residual standard error: 0.2393 on 677960 degrees of freedom  
Multiple R-squared: 0.00663, Adjusted R-squared: 0.006554   
F-statistic: 87.02 on 52 and 677960 DF, p-value: < 2.2e-16

1. Perform a residual analysis of our fitted model. You would expect things to look strange.

# Extract residuals   
res1 <- residuals(model1)   
  
# Histogram and simple plot of residuals  
hist(res1, breaks = 20, col = "skyblue", main = "Histogram of Residuals")



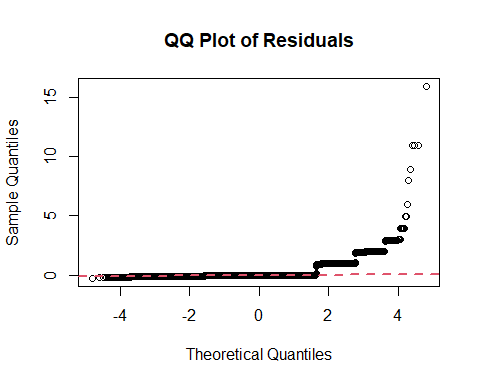
plot(res1, main = "Residuals Plot", ylab = "Residuals")  
abline(h = 0, col = "red", lty = 2)



# Summary  
summary(res1)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
-0.28749 -0.06338 -0.05009 0.00000 -0.03764 15.97132

# QQ plot  
qqnorm(res1, main = "QQ Plot of Residuals")  
qqline(res1, col = 2, lwd = 2, lty = 2)



1. Now let us assume that claim counts follow a Poisson distribution. This will allow us to account for exposure. Fit a GLM using gamlss. Use only 10% of the data as otherwise running the regression model over the entire dataset will take too long.

library(gamlss)

Warning: package 'gamlss' was built under R version 4.2.3

Loading required package: splines

Loading required package: gamlss.data

Attaching package: 'gamlss.data'

The following object is masked from 'starwars':  
  
 species

The following object is masked from 'package:datasets':  
  
 sleep

Loading required package: gamlss.dist

Loading required package: MASS

Attaching package: 'MASS'

The following object is masked from 'package:dplyr':  
  
 select

Loading required package: nlme

Attaching package: 'nlme'

The following object is masked from 'package:dplyr':  
  
 collapse

Loading required package: parallel

\*\*\*\*\*\*\*\*\*\* GAMLSS Version 5.4-12 \*\*\*\*\*\*\*\*\*\*

For more on GAMLSS look at https://www.gamlss.com/

Type gamlssNews() to see new features/changes/bug fixes.

subdat <- dat[sample(seq\_len(nrow(dat)), size = 0.1 \* nrow(dat)), ]  
# With Egamlss# With Exposure as offset  
model2 <- gamlss(ClaimNb ~ Area + VehPower + VehGas + VehBrand + Region  
 + Density + VehAge + DrivAge + BonusMalus + offset(log(Exposure)),  
 data = subdat, family = PO)

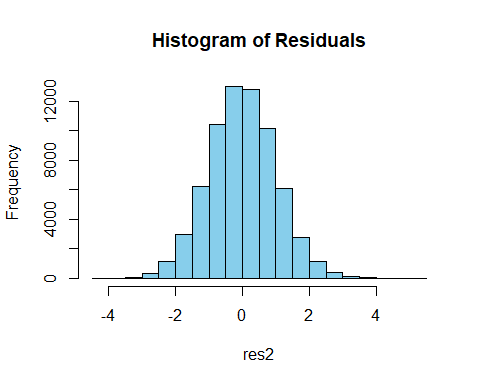
GAMLSS-RS iteration 1: Global Deviance = 27768.01   
GAMLSS-RS iteration 2: Global Deviance = 27768.01

Warning in runif(length(y), aval, bval): NAs produced

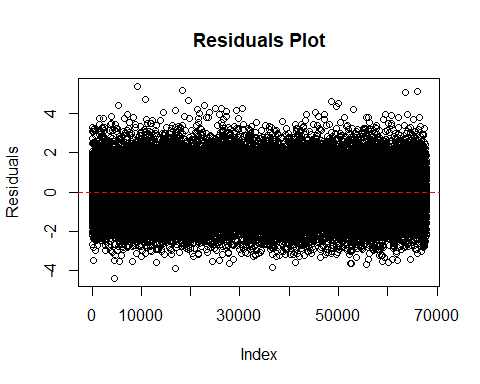
#summary(model2) - takes a long time to run

1. Perform residual analysis on this. Which model do you think is better, the Poisson or Linear Regression one?

# Extract residuals (default in gamlss = normalized quantile residuals)  
res2 <- residuals(model2) # replace with your model object  
  
# Histogram and simple plot of residuals  
hist(res2, breaks = 20, col = "skyblue", main = "Histogram of Residuals")



plot(res2, main = "Residuals Plot", ylab = "Residuals")  
abline(h = 0, col = "red", lty = 2)



# Summary  
summary(res2)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
-4.377129 -0.680450 -0.010939 0.001143 0.672490 5.378027 1

# QQ plot  
qqnorm(res2, main = "QQ Plot of Residuals")  
qqline(res2, col = 2, lwd = 2, lty = 2)

