R tutorial medicine

Aayush Kapri

2023-09-12

1 R and RStudio

1.1 R

R is a programming language which is highly relevant in medicine and health science due to its robust data analysis and statistical capabilities. Researchers in these fields can use R to analyze large datasets, conduct statistical modeling, and visualize medical data. It is open-source and has extensive libraries, which make it a powerful tool for advancing medical research and improving healthcare practices.

To install R, go to cran. And download the version that is appropriate to your device. It is recommended to install the latest version of R. If you already have R installed in your device, you can update your current version by installing the latest version. You do not need to unistall the previous version to update your previous version.

1.2 RStudio

RStudio is an integrated development environment (IDE) for the R programming language. RStudio provides a user-friendly and feature-rich interface for working with R, making it easier to write, debug, and run R code efficiently.

In order to install free version of RStudio, go to the download page. You need to download the version that is appropriate to your device.

1.3 RStudio Cloud

R programming can also to used in a cloud version of RStudio, making users to access R and their work from anywhere with a web browser. Visit Postit and it will have the exact interface as Rstudio where you can write codes and it will be stored in the cloud. This platform facilitates collaboration between researchers by enabling simultaneous workflows.

2 Operators, Variables and Data Types

2.1 Operators

Before proceeding further, let us introduce the operator that is most commonly used in R, i.e., '<-'. It is the assignment operator.

```r
# Suppose we have a patient's age and want to store it in a variable.
patient_age <- 35

# Now, you can use 'patient_age' for medical calculations or analysis.
# print() is used for printing in R. \n is used to break lines.
print(paste("The patient's age is:", patient_age, "years old"))
```

```
## [1] "The patient's age is: 35 years old"
```

Operators in R should be very familiar to researchers.

```r
# Example 1: Addition (+)
dose_1 <- 100
dose_2 <- 50
total_dose <- dose_1 + dose_2
print(paste("Total dose:", total_dose, "mg"))
```

```
## [1] "Total dose: 150 mg"
```

```r
# Example 2: Subtraction (-)
pre_treatment_bp <- 120
post_treatment_bp <- 110
delta_bp <- pre_treatment_bp - post_treatment_bp
print(paste("Change in blood pressure:", delta_bp, "mmHg"))
```

```
## [1] "Change in blood pressure: 10 mmHg"
```

```r
# Example 3: Multiplication (*)
weight_kg <- 70
height_m <- 1.75
bmi <- weight_kg / (height_m * height_m)
print(paste("BMI:", bmi))
```

```
## [1] "BMI: 22.8571428571429"
```

```r
# Example 4: Division (/)
total_beats <- 4500
time_in_minutes <- 60
heart_rate <- total_beats / time_in_minutes
print(paste("Heart rate:", heart_rate, "beats per minute"))
```

```
## [1] "Heart rate: 75 beats per minute"
```

```r
# Example 5: Exponentiation (^ or **)
base_dosage <- 10
weight_kg <- 65
drug_dosage <- base_dosage * (weight_kg^2)
print(paste("Drug dosage:", drug_dosage, "mg"))
```

```
## [1] "Drug dosage: 42250 mg"
```

2.2 Variables Types

Variables in R are named storage containers that hold data or values, allowing you to work with and manipulate information in your R scripts and programs. These variables can represent a wide range of data types, including numeric values, text, logical values, factors, and dates. You can create, assign, and modify variables to perform various data analysis and calculations in R.

```
# Numeric (for representing measurements):
# Example: Patient's Body Mass Index (BMI)
bmi_numeric <- 25.5
bmi_numeric
```

```
## [1] 25.5
```

```
# Integer (for whole numbers):
# Example: Patient's Age
age_integer <- 35L
print(paste("Age (integer):", age_integer))
```

```
## [1] "Age (integer): 35"
```

```
# Character (for textual data or patient names):
# Example: Patient's Name
patient_name_char <- "John Doe"
patient_name_char
```

```
## [1] "John Doe"
```

```
# Logical (for binary data, like whether a patient is a smoker or not):
# Example: Smoking Status
smoker_logical <- TRUE  # If the patient is a smoker
smoker_logical
```

```
## [1] TRUE
```

```
# Factor (for categorical data like blood types):
# Example: Blood Type
blood_type_factor <- factor("A", levels = c("A", "B", "AB", "O"))
blood_type_factor
```

```
## [1] A
## Levels: A B AB O
```

```
# Date (for recording dates of medical events):
# Example: Date of Admission
admission_date_date <- as.Date("2023-09-14")
admission_date_date
```

```
## [1] "2023-09-14"
```

2.3 Data Format

In R, there exists a diverse range of data types, including scalars, vectors (both numeric, character, and logical), matrices, data frames, lists, etc. We consider the most important data type to be the data frame, for which we provide a separate section. Examples of other data types are given below.

```
# Vectors (for storing a sequence of data):
# Example: Heart rate measurements over time.
heart_rate_vector <- c(72, 75, 80, 76, 82)
heart_rate_vector
```

```
## [1] 72 75 80 76 82
```

```
# Matrices (two-dimensional arrays with rows and columns):
# Example: Blood pressure measurements of patients over time.
patient_bp_matrix <- matrix(c(120, 130, 115, 122, 135, 118), nrow = 3, ncol = 2)
patient_bp_matrix
```

```
##      [,1] [,2]
## [1,]  120  122
## [2,]  130  135
## [3,]  115  118
```

```
# Lists (for storing heterogeneous data types):
# Example: A list of patient characteristics and test results.
patient_info_list <- list(
  Name = "John Doe",
  Age = 35,
  Diagnosis = "Hypertension",
  TestResults = c(120, 75, 160)
)
patient_info_list
```

```
## $Name
## [1] "John Doe"
##
## $Age
## [1] 35
##
## $Diagnosis
## [1] "Hypertension"
##
## $TestResults
## [1] 120  75 160
```

2.3.1 Data Frame Data frames in R are similar to tables of data, akin to spreadsheets in Excel. Unlike matrices, data frames can accommodate columns containing categorical, logical, numerical, and other types of data. When we need to import an external file, often in a tabular format, we typically load it as a data frame in R.

```r
# Create a data frame for patient information
patient_data <- data.frame(
  PatientID = c(1, 2, 3, 4, 5),
  Name = c("John Doe", "Alice Smith", "Bob Johnson", "Emily Davis", "Michael Wilson"),
  Age = c(45, 28, 35, 52, 40),
  Gender = c("Male", "Female", "Male", "Female", "Male"),
  Diagnosis = c("Hypertension", "Diabetes", "Asthma", "Heart Disease", "Allergies"),
  BloodPressure = c("120/80", "135/88", "122/78", "140/95", "118/72")
)

# Display the patient data frame
print(patient_data)
```

```
##   PatientID           Name Age Gender     Diagnosis BloodPressure
## 1         1       John Doe  45   Male  Hypertension        120/80
## 2         2    Alice Smith  28 Female      Diabetes        135/88
## 3         3    Bob Johnson  35   Male        Asthma        122/78
## 4         4    Emily Davis  52 Female Heart Disease        140/95
## 5         5 Michael Wilson  40   Male     Allergies        118/72
```

4 Packages and Libraries

Packages in R are used to extend the functionality of the programming ability of R. Packages allow programmers to perform specialized task such as data analysis, visualization, statistical modeling and more. Popular packages in R include `dplyr`and `ggplot2`. In order to install packages, run `install.packages("ggplot2")` at the command line. That will have R automatically download the package to your computer. Similarly, you can install other packages.

4.1 tidyverse, survival and ggplot

The `tidyverse` is a library with collection of R packages designed for data science that share a common philosophy and grammar, making data manipulation and visualization in R more consistent and efficient. It includes popular packages like `ggplot2`, `dplyr`, `tidyr`, `readr`, and more.

```r
# Assuming lung_data is your dataset
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(survival)
```

```
## Warning: package 'survival' was built under R version 4.3.1
```

```r
#data function can display all available data sets in this package
data(package = "survival")
# We will use data set "lung

#the sex variable in a dataset is coded as 1 for male and 2 for female, and we want to convert it to "M
lung %>%
  mutate(sex = ifelse(sex == 1, "Male", "Female")) %>%
  head()
```

```
##   inst time status age  sex ph.ecog ph.karno pat.karno meal.cal wt.loss
## 1    3  306      2  74 Male       1       90       100     1175      NA
## 2    3  455      2  68 Male       0       90        90     1225      15
## 3    3 1010      1  56 Male       0       90        90       NA      15
## 4    5  210      2  57 Male       1       90        60     1150      11
## 5    1  883      2  60 Male       0      100        90       NA       0
## 6   12 1022      1  74 Male       1       50        80      513       0
```

4.2 Pipes

You may have noticed or notice the operator %>%. It is called the pipe operator. It is from the magittr package. Pipes are a powerful tool for enhancing the readability and expressiveness of your code. It allows you to write code in a way that flows from left to right, making it easier to understand. As %>% is a bit tedious to type, there exist shortcuts: shift-ctrl-m on a Mac, shift-ctrl-m on a Windows machine. Here is an example of code with and without pipe:

```r
# Load the necessary library
library(survival)

# Load datset
data(cancer, package = "survival")

# Without pipes
male_data <- subset(cancer, sex == 1)
mean_age <- mean(male_data$age)

# Display the result
print(mean_age)
```

```
## [1] 63.34058
```

```r
# Load the necessary library
library(survival)
library(dplyr)

# Load dataset
data(cancer, package = "survival")

# With pipes
mean_age <- cancer %>%
  filter(sex == 1) %>%
  summarise(mean_age = mean(age)) %>%
  pull(mean_age)
```

```
# Display the result
print(mean_age)
```

```
## [1] 63.34058
```

5 Importing data in r

In order to work with data in R, it is important to import data from external sources such as your computer. The external data is most commonly a csv file such as Excel workbook. When we import a external data, it is converted into `data.frame`.

```
# Use read.csv() to import the data
# <- read.csv("C:/Users/admin/Dropbox/Cooper Project/20230509_transition_matrix/PEMBRO.OS.csv")

# Display the first few rows of the data
#head(data)
```

R also allows us to work with pre-loaded data. R comes with several datasets that are preloaded for users to explore and analyze. These datasets are part of the base R installation, and you can access them without accessing your local directory. In fact, the lung dataset we imported above is a pre-loaded dataset.

```
#Survival in patients with advanced lung cancer from the North Central Cancer Treatment Group. Performa
library(survival)
#data function can display all available data sets in this package
data(package = "survival")
# We will use data set "lung" as an example
?data(lung)
```

```
## starting httpd help server ... done
```

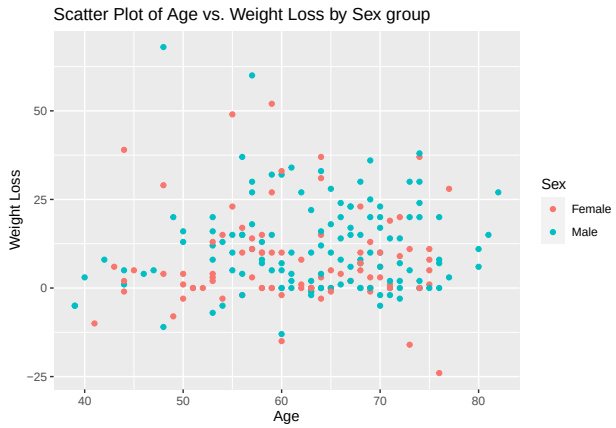
```
#we can call the data set directly after loading the survival library
head(lung)
```

```
##   inst time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
## 1    3  306      2  74   1       1       90       100     1175      NA
## 2    3  455      2  68   1       0       90        90     1225      15
## 3    3 1010      1  56   1       0       90        90       NA      15
## 4    5  210      2  57   1       1       90        60     1150      11
## 5    1  883      2  60   1       0      100        90       NA       0
## 6   12 1022      1  74   1       1       50        80      513       0
```

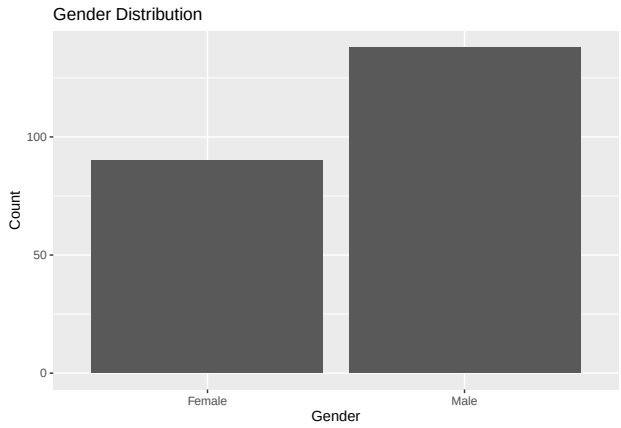
6 Graphing using R

`ggplot2` is a powerful and versatile data visualization package in R that facilitates the creation of a wide range of high-quality plots. With ggplot2, you can produce a diverse array of plots, including scatter plots, bar charts, line graphs, and more, all with a focus on customization. The package encourages a declarative style, where you specify what you want the plot to look like and let ggplot2 handle the details.

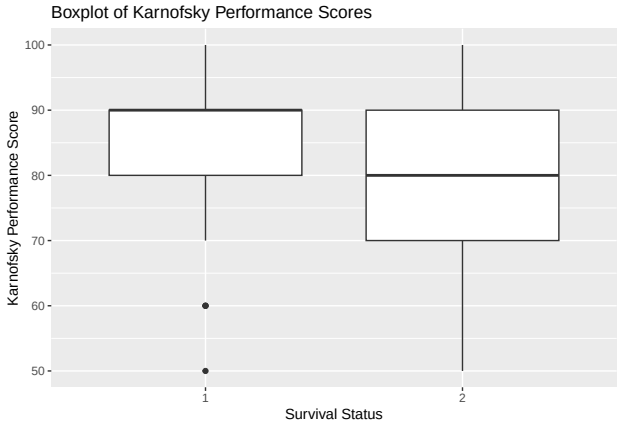
```r
library(ggplot2)

library(survival)
data(cancer, package = "survival")
lung <- as.data.frame(cancer)

lung <- lung %>%
  mutate(sex = ifelse(sex == 1, "Male", "Female"))

# 1. Scatter Plot of Age vs. Weight Loss
plot <-ggplot(lung, aes(x = age, y = wt.loss, color = factor(sex))) +
       geom_point() +
      labs(title = "Scatter Plot of Age vs. Weight Loss by Sex group",
       x = "Age",
       y = "Weight Loss",
       color = "Sex")
plot
```



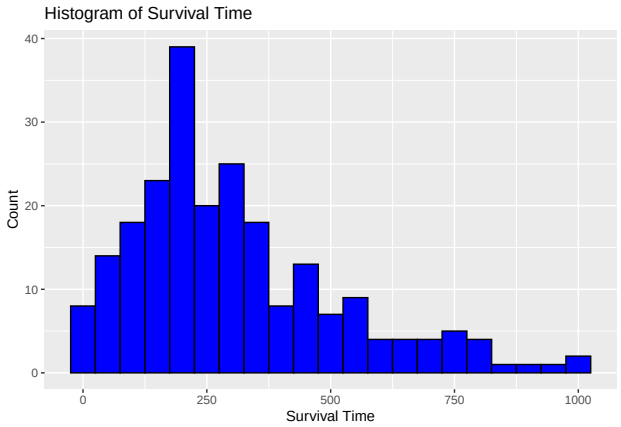

```r
# 2. Bar Chart of Gender Distribution
ggplot(lung, aes(x = factor(sex))) +
  geom_bar() +
  labs(title = "Gender Distribution",
       x = "Gender",
       y = "Count")
```



```r
# 3. Boxplot of Karnofsky Performance Scores
ggplot(lung, aes(x = factor(status), y = ph.karno)) +
  geom_boxplot() +
  labs(title = "Boxplot of Karnofsky Performance Scores",
       x = "Survival Status",
       y = "Karnofsky Performance Score")
```




```r
# 4. Histogram of Survival Time
ggplot(lung, aes(x = time)) +
  geom_histogram(binwidth = 50, fill = "blue", color = "black") +
  labs(title = "Histogram of Survival Time",
       x = "Survival Time",
       y = "Count")
```

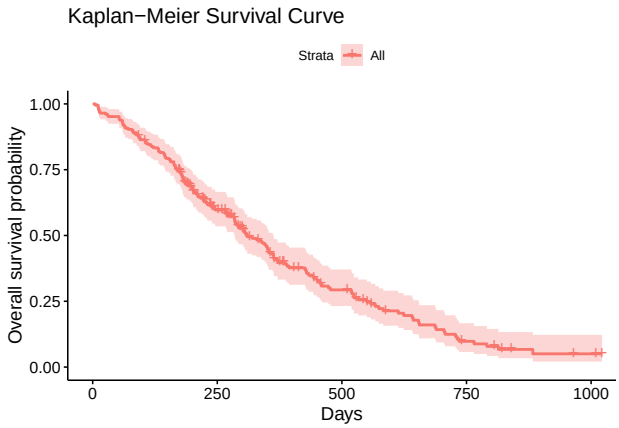



```r
# 5. Survival Plot

# # Load necessary libraries
library(survival)
library(survminer)
```

```
## Loading required package: ggpubr

##
## Attaching package: 'survminer'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     myeloma

## The following object is masked from 'package:survival':
##
##     myeloma
```

```r
# Fit a survival model
survfit_object <- survfit(Surv(time, status) ~ 1, data = cancer)

# Create survival plot using ggsurvplot
ggsurvplot(survfit_object, data = cancer,
           xlab = "Days", ylab = "Overall survival probability",
           title = "Kaplan-Meier Survival Curve")
```



7 Data Transformation

Data transformation is the process of cleaning, organizing, and transforming raw data into a format that is easier to access and analyze. It is also known as data summarizing or data wrangling. Data transformation is important because it ensures that data is of high quality and can be used to produce accurate and valuable insights. Data analysts typically spend the majority of their time on data wrangling, as it is a necessary step before any further analysis can be done. Some useful functions under data wrangling are below.

```r
library(ggplot2)

library(survival)
data(cancer, package = "survival")
lung <- as.data.frame(cancer)


# Selecting a subset of variables
lung_selected <- lung %>%
  select(time, status, age, sex)

# Displaying the first few rows of the updated dataset
head(lung_selected)
```

```
##   time status age sex
## 1  306      2  74   1
## 2  455      2  68   1
## 3 1010      1  56   1
## 4  210      2  57   1
## 5  883      2  60   1
## 6 1022      1  74   1
```

```
# Generating a new variable 'sex_category' based on the 'sex' variable
lung_mutated <- lung %>%
  mutate(sex_category = ifelse(sex == 1, "Male", "Female"))

# Displaying the first few rows of the updated dataset
head(lung_mutated)
```

```
##   inst time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
## 1    3  306      2  74   1       1       90       100     1175      NA
## 2    3  455      2  68   1       0       90        90     1225      15
## 3    3 1010      1  56   1       0       90        90       NA      15
## 4    5  210      2  57   1       1       90        60     1150      11
## 5    1  883      2  60   1       0      100        90       NA       0
## 6   12 1022      1  74   1       1       50        80      513       0
##   sex_category
## 1         Male
## 2         Male
## 3         Male
## 4         Male
## 5         Male
## 6         Male
```

```
# Sorting the dataset by age in descending order
lung_sorted <- lung %>%
  arrange(desc(age))

# Displaying the first few rows of the sorted dataset
head(lung_sorted)
```

```
##     inst time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
## 149   12   31      2  82   1       0      100        90      413      27
## 79     3   11      2  81   1       0       90        NA      731      15
## 113   10  283      2  80   1       1       80       100     1030       6
## 120   15  363      2  80   1       1       80        90      346      11
## 129   16  551      1  77   2       2       80        60      750      28
## 224    1  188      1  77   1       1       80        60       NA       3
```

```
# Filtering the dataset to include only females
lung_females <- lung %>%
  filter(sex == 2)

# Displaying the first few rows of the filtered dataset
head(lung_females)
```

```
##    inst time status age sex ph.ecog ph.karno pat.karno meal.cal wt.loss
## 7     7  310      2  68   2       2       70        60      384      10
## 8    11  361      2  71   2       2       60        80      538       1
## 12   16  654      2  68   2       2       70        70       NA      23
## 13   11  728      2  68   2       1       90        90       NA       5
## 19    1   61      2  56   2       2       60        60      238      10
## 22    6   81      2  49   2       0      100        70     1175      -8
```

```
# Grouping by sex and calculating mean age and median survival time
summary_stats <- lung %>%
  group_by(sex) %>%
  summarize(
    MeanAge = mean(age),
    MedianSurvival = median(time)
  )

# Displaying the summary statistics
print(summary_stats)
```

```
## # A tibble: 2 x 3
##     sex MeanAge MedianSurvival
##   <dbl>   <dbl>          <dbl>
## 1     1    63.3            224
## 2     2    61.1            292.
```

```
# Grouping by sex and using slice to keep the first row in each group
lung_sliced <- lung %>%
  group_by(sex) %>%
  slice(1)  # Keep the first row in each group

# Displaying the first few rows of the sliced dataset
head(lung_sliced)
```

```
## # A tibble: 2 x 10
## # Groups:   sex [2]
##    inst  time status   age   sex ph.ecog ph.karno pat.karno meal.cal wt.loss
##   <dbl> <dbl>  <dbl> <dbl> <dbl>   <dbl>    <dbl>     <dbl>    <dbl>   <dbl>
## 1     3   306      2    74     1       1       90       100     1175      NA
## 2     7   310      2    68     2       2       70        60      384      10
```

8 Data Summary

```
# Install and load the required packages
# Load necessary libraries
library(survival)
library(dplyr)
library(summarytools)
```

```
## Warning: package 'summarytools' was built under R version 4.3.1
```

```
##
## Attaching package: 'summarytools'

## The following object is masked from 'package:tibble':
##
##     view
```

```
library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2022). stargazer: Well-Formatted Regression and Summary Statistics Tables.

##  R package version 5.2.3. https://CRAN.R-project.org/package=stargazer
```

```
#library(compareGroups)

# Assuming your dataset is named 'lung'
data(cancer, package = "survival")
lung <- as.data.frame(cancer)

# Use dfsummary to get a summary of the dataset
dfSummary(lung)
```

```
## Data Frame Summary
## lung
## Dimensions: 228 x 10
## Duplicates: 0
##
## ------------------------------------------------------------------------------------------------------------------
## No   Variable    Stats / Values              Freqs (% of Valid)    Graph                 Valid      N
## ---- ----------- --------------------------- --------------------- --------------------- ---------- -
## 1    inst        Mean (sd) : 11.1 (8.3)      18 distinct values    :   .                 227        1
##      [numeric]   min < med < max:                                  :   :                 (99.6%)
##                  1 < 11 < 33                                       :   :
##                  IQR (CV) : 13 (0.7)                               : : :   :
##                                                                    : : : : : . .
##
## 2    time        Mean (sd) : 305.2 (210.6)   186 distinct values     : .                 228        0
##      [numeric]   min < med < max:                                    : :                 (100.0%)
##                  5 < 255.5 < 1022                                  : : : .
##                  IQR (CV) : 229.8 (0.7)                            : : : : . .
##                                                                    : : : : : : : . . .
##
## 3    status      Min  : 1                    1 :  63 (27.6%)       IIIII                 228        0
##      [numeric]   Mean : 1.7                  2 : 165 (72.4%)       IIIIIIIIIIIII         (100.0%)
##                  Max  : 2
##
## 4    age         Mean (sd) : 62.4 (9.1)      42 distinct values              :         228        0
##      [numeric]   min < med < max:                                      . : : : .       (100.0%)
##                  39 < 63 < 82                                      : : : : : .
```

```
##                  IQR (CV) : 13 (0.1)                                  . : : : : : : :
##                                                                     . : : : : : : : : : .
##
## 5    sex         Min  : 1                  1 : 138 (60.5%)       IIIIIIIIIIII          228        0
##      [numeric]   Mean : 1.4                2 :  90 (39.5%)       IIIIIII               (100.0%)
##                  Max  : 2
##
## 6    ph.ecog     Mean (sd) : 1 (0.7)       0 :  63 (27.8%)       IIIII                 227        1
##      [numeric]   min < med < max:          1 : 113 (49.8%)       IIIIIIIII             (99.6%)
##                  0 < 1 < 3                 2 :  50 (22.0%)       IIII
##                  IQR (CV) : 1 (0.8)        3 :   1 ( 0.4%)
##
## 7    ph.karno    Mean (sd) : 81.9 (12.3)   50 :  6 ( 2.6%)                             227        1
##      [numeric]   min < med < max:          60 : 19 ( 8.4%)       I                     (99.6%)
##                  50 < 80 < 100             70 : 32 (14.1%)       II
##                  IQR (CV) : 15 (0.2)       80 : 67 (29.5%)       IIIII
##                                            90 : 74 (32.6%)       IIIIII
##                                            100 : 29 (12.8%)      II
##
## 8    pat.karno   Mean (sd) : 80 (14.6)     30 :  2 ( 0.9%)                             225        3
##      [numeric]   min < med < max:          40 :  2 ( 0.9%)                             (98.7%)
##                  30 < 80 < 100             50 :  4 ( 1.8%)
##                  IQR (CV) : 20 (0.2)       60 : 30 (13.3%)       II
##                                            70 : 41 (18.2%)       III
##                                            80 : 51 (22.7%)       IIII
##                                            90 : 60 (26.7%)       IIIII
##                                            100 : 35 (15.6%)      III
##
## 9    meal.cal    Mean (sd) : 928.8 (402.2) 60 distinct values          :                 181        4
##      [numeric]   min < med < max:                                      :                 (79.4%)
##                  96 < 975 < 2600                                 .   : :
##                  IQR (CV) : 515 (0.4)                           : : : :
##                                                              : : : : : .       .
##
## 10   wt.loss     Mean (sd) : 9.8 (13.1)    53 distinct values        :                   214        1
##      [numeric]   min < med < max:                                    : :                 (93.9%)
##                  -24 < 7 < 68                                    : : .
##                  IQR (CV) : 15.8 (1.3)                           : : : .
##                                                               . : : : : .
## ------------------------------------------------------------------------------------------------------
```

```
# Use stargazer to get a summary of the dataset
stargazer(lung, type = "text")
```

```
##
## ========================================
## Statistic  N   Mean   St. Dev. Min  Max
## ----------------------------------------
## inst      227 11.088   8.303    1   33
## time      228 305.232 210.646   5  1,022
## status    228  1.724   0.448    1    2
## age       228 62.447   9.073   39   82
## sex       228  1.395   0.490    1    2
## ph.ecog   227  0.952   0.718    0    3
```

```
## ph.karno  227 81.938   12.328  50   100
## pat.karno 225 79.956   14.623  30   100
## meal.cal  181 928.779 402.175  96  2,600
## wt.loss   214  9.832   13.140  -24  68
## ----------------------------------------
```

```r
# Use compareGroups to get a significance of the variables in dataset with each other
#compareGroups(sex ~ ., data = lung, method = 4)
```

8 Functions

In R, a function is a reusable and self-contained block of code that performs a specific task or computation. Functions take input parameters, perform operations, and return results. They facilitate code organization, modularity, and reusability, allowing users to encapsulate logic and efficiently apply it to different data or scenarios. Functions in R are defined using the function keyword and can be called with specified arguments to execute their tasks.

```r
# 'cancer' is your lung cancer dataset

# Function to rescale a variable to the interval [0, 1]
rescale01 <- function(x) {
  rng <- range(x, na.rm = TRUE)
  (x - rng[1]) / (rng[2] - rng[1])
}

# Example usage: Rescale the 'age' variable from the lung cancer dataset
rescaled_age <- rescale01(cancer$age)

# Display the original and rescaled values
data.frame(original_age = cancer$age, rescaled_age = rescaled_age)
```

```
##     original_age rescaled_age
## 1             74   0.81395349
## 2             68   0.67441860
## 3             56   0.39534884
## 4             57   0.41860465
## 5             60   0.48837209
## 6             74   0.81395349
## 7             68   0.67441860
## 8             71   0.74418605
## 9             53   0.32558140
## 10            61   0.51162791
## 11            57   0.41860465
## 12            68   0.67441860
## 13            68   0.67441860
## 14            60   0.48837209
## 15            57   0.41860465
## 16            67   0.65116279
## 17            70   0.72093023
## 18            63   0.55813953
## 19            56   0.39534884
## 20            57   0.41860465
```

```
## 21          67   0.65116279
## 22          49   0.23255814
## 23          50   0.25581395
## 24          58   0.44186047
## 25          72   0.76744186
## 26          70   0.72093023
## 27          60   0.48837209
## 28          70   0.72093023
## 29          53   0.32558140
## 30          74   0.81395349
## 31          69   0.69767442
## 32          73   0.79069767
## 33          48   0.20930233
## 34          60   0.48837209
## 35          61   0.51162791
## 36          62   0.53488372
## 37          65   0.60465116
## 38          66   0.62790698
## 39          74   0.81395349
## 40          64   0.58139535
## 41          70   0.72093023
## 42          73   0.79069767
## 43          59   0.46511628
## 44          60   0.48837209
## 45          68   0.67441860
## 46          76   0.86046512
## 47          74   0.81395349
## 48          63   0.55813953
## 49          74   0.81395349
## 50          50   0.25581395
## 51          72   0.76744186
## 52          63   0.55813953
## 53          68   0.67441860
## 54          58   0.44186047
## 55          59   0.46511628
## 56          62   0.53488372
## 57          65   0.60465116
## 58          57   0.41860465
## 59          58   0.44186047
## 60          64   0.58139535
## 61          75   0.83720930
## 62          48   0.20930233
## 63          73   0.79069767
## 64          65   0.60465116
## 65          69   0.69767442
## 66          68   0.67441860
## 67          67   0.65116279
## 68          64   0.58139535
## 69          68   0.67441860
## 70          67   0.65116279
## 71          63   0.55813953
## 72          48   0.20930233
## 73          74   0.81395349
## 74          40   0.02325581
```

```
## 75            53   0.32558140
## 76            71   0.74418605
## 77            51   0.27906977
## 78            56   0.39534884
## 79            81   0.97674419
## 80            73   0.79069767
## 81            59   0.46511628
## 82            55   0.37209302
## 83            42   0.06976744
## 84            44   0.11627907
## 85            44   0.11627907
## 86            71   0.74418605
## 87            62   0.53488372
## 88            61   0.51162791
## 89            44   0.11627907
## 90            72   0.76744186
## 91            63   0.55813953
## 92            70   0.72093023
## 93            66   0.62790698
## 94            57   0.41860465
## 95            69   0.69767442
## 96            72   0.76744186
## 97            69   0.69767442
## 98            71   0.74418605
## 99            64   0.58139535
## 100           70   0.72093023
## 101           58   0.44186047
## 102           69   0.69767442
## 103           56   0.39534884
## 104           63   0.55813953
## 105           59   0.46511628
## 106           66   0.62790698
## 107           54   0.34883721
## 108           67   0.65116279
## 109           55   0.37209302
## 110           75   0.83720930
## 111           69   0.69767442
## 112           44   0.11627907
## 113           80   0.95348837
## 114           75   0.83720930
## 115           54   0.34883721
## 116           76   0.86046512
## 117           49   0.23255814
## 118           68   0.67441860
## 119           66   0.62790698
## 120           80   0.95348837
## 121           75   0.83720930
## 122           60   0.48837209
## 123           69   0.69767442
## 124           72   0.76744186
## 125           70   0.72093023
## 126           66   0.62790698
## 127           50   0.25581395
## 128           64   0.58139535
```

```
## 129          77   0.88372093
## 130          48   0.20930233
## 131          59   0.46511628
## 132          53   0.32558140
## 133          47   0.18604651
## 134          55   0.37209302
## 135          67   0.65116279
## 136          74   0.81395349
## 137          58   0.44186047
## 138          56   0.39534884
## 139          54   0.34883721
## 140          56   0.39534884
## 141          73   0.79069767
## 142          74   0.81395349
## 143          76   0.86046512
## 144          65   0.60465116
## 145          57   0.41860465
## 146          53   0.32558140
## 147          71   0.74418605
## 148          54   0.34883721
## 149          82   1.00000000
## 150          59   0.46511628
## 151          70   0.72093023
## 152          60   0.48837209
## 153          62   0.53488372
## 154          53   0.32558140
## 155          55   0.37209302
## 156          69   0.69767442
## 157          68   0.67441860
## 158          62   0.53488372
## 159          63   0.55813953
## 160          56   0.39534884
## 161          62   0.53488372
## 162          44   0.11627907
## 163          69   0.69767442
## 164          63   0.55813953
## 165          64   0.58139535
## 166          57   0.41860465
## 167          60   0.48837209
## 168          46   0.16279070
## 169          61   0.51162791
## 170          65   0.60465116
## 171          61   0.51162791
## 172          58   0.44186047
## 173          56   0.39534884
## 174          43   0.09302326
## 175          53   0.32558140
## 176          59   0.46511628
## 177          56   0.39534884
## 178          55   0.37209302
## 179          53   0.32558140
## 180          74   0.81395349
## 181          60   0.48837209
## 182          39   0.00000000
```

```
## 183          66   0.62790698
## 184          65   0.60465116
## 185          51   0.27906977
## 186          45   0.13953488
## 187          72   0.76744186
## 188          58   0.44186047
## 189          64   0.58139535
## 190          53   0.32558140
## 191          72   0.76744186
## 192          52   0.30232558
## 193          50   0.25581395
## 194          64   0.58139535
## 195          71   0.74418605
## 196          70   0.72093023
## 197          63   0.55813953
## 198          64   0.58139535
## 199          52   0.30232558
## 200          60   0.48837209
## 201          64   0.58139535
## 202          73   0.79069767
## 203          63   0.55813953
## 204          50   0.25581395
## 205          63   0.55813953
## 206          62   0.53488372
## 207          55   0.37209302
## 208          50   0.25581395
## 209          69   0.69767442
## 210          59   0.46511628
## 211          60   0.48837209
## 212          67   0.65116279
## 213          69   0.69767442
## 214          64   0.58139535
## 215          65   0.60465116
## 216          65   0.60465116
## 217          41   0.04651163
## 218          76   0.86046512
## 219          70   0.72093023
## 220          57   0.41860465
## 221          67   0.65116279
## 222          71   0.74418605
## 223          76   0.86046512
## 224          77   0.88372093
## 225          39   0.00000000
## 226          75   0.83720930
## 227          66   0.62790698
## 228          58   0.44186047
```

9 Hypothesis testing

```
# Load necessary libraries
library(survival)
library(dplyr)
```

```r
# Assuming your dataset is named 'lung'
data(cancer, package = "survival")
lung <- as.data.frame(cancer)

# Display summary statistics
summary(lung$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   39.00   56.00   63.00   62.45   69.00   82.00
```

```r
summary(lung$time)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     5.0   166.8   255.5   305.2   396.5  1022.0
```

```r
# Shapiro-Wilk Normality Test
# Interpretation: The Shapiro-Wilk test assesses whether the age variable follows a normal distribution
shapiro_test_result <- shapiro.test(lung$age)
print(shapiro_test_result)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  lung$age
## W = 0.98173, p-value = 0.004829
```

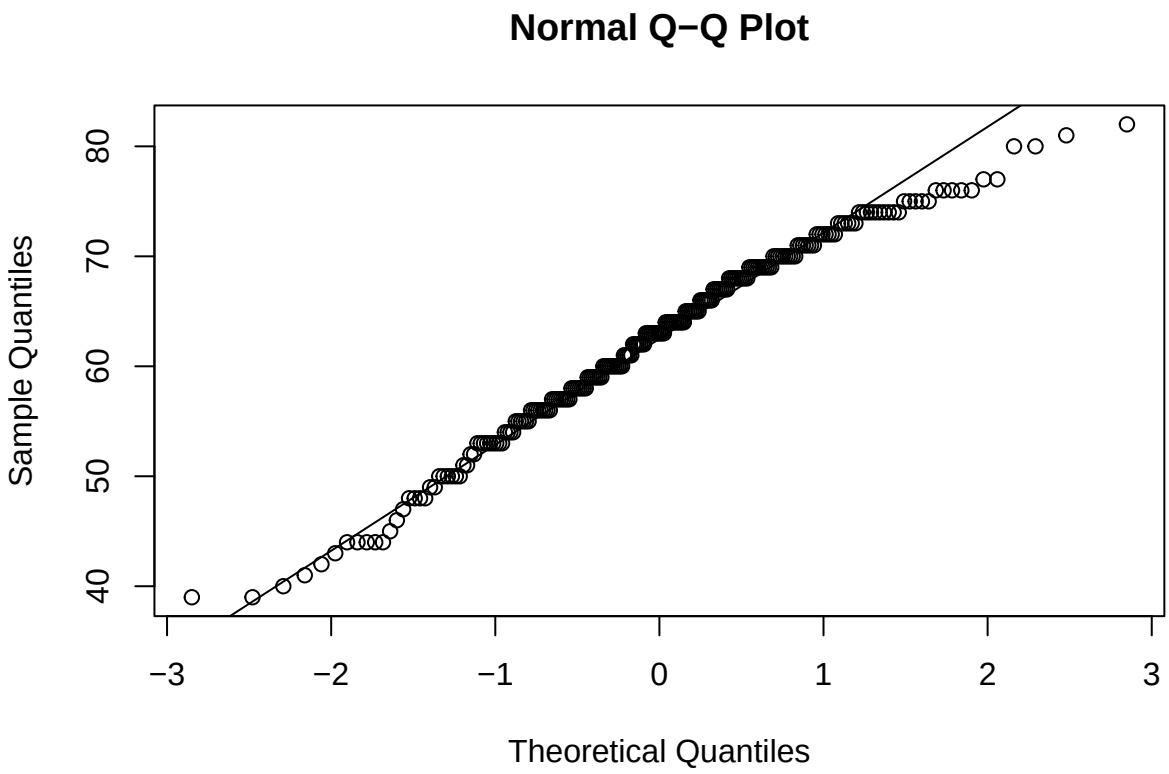
```r
# Kolmogorov-Smirnov Test for Normality
# Interpretation: The Kolmogorov-Smirnov test checks if the time variable is normally distributed. Usef
ks_test_result <- ks.test(lung$time, "pnorm")
```

```
## Warning in ks.test.default(lung$time, "pnorm"): ties should not be present for
## the Kolmogorov-Smirnov test
```

```r
print(ks_test_result)
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  lung$time
## D = 1, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```r
# Q-Q Plot for Age
# Interpretation: The Q-Q plot visually assesses whether the distribution of age follows a normal distr
qqnorm(lung$age)
qqline(lung$age)
```




```
# Hypothesis Testing - T-Test
# Interpretation: The t-test compares the mean age between different sexes. Useful for assessing differ
# T-Test
# H0: There is no difference in mean age between sexes.
# H1: There is a significant difference in mean age between sexes
t_test_result <- t.test(age ~ sex, data = lung)
print(t_test_result)
```

```
##
##  Welch Two Sample t-test
##
## data:  age by sex
## t = 1.8632, df = 194.72, p-value = 0.06394
## alternative hypothesis: true difference in means between group 1 and group 2 is not equal to 0
## 95 percent confidence interval:
##  -0.1324347  4.6580386
## sample estimates:
## mean in group 1 mean in group 2
##        63.34058        61.07778
```

```
# Hypothesis Testing - Wilcoxon Rank Sum Test
# Interpretation: The Wilcoxon Rank Sum Test evaluates if the distribution of ages differs between sexes
# Wilcoxon Rank Sum Test
# H0: There is no difference in the distribution of age between sexes.
# H1: There is a significant difference in the distribution of age between sexes.
```

```r
wilcox_test_result <- wilcox.test(age ~ sex, data = lung)
print(wilcox_test_result)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  age by sex
## W = 7136.5, p-value = 0.05701
## alternative hypothesis: true location shift is not equal to 0
```

```r
# Chi-squared Test for Independence - Gender vs. Survival Status
# Interpretation: The Chi-squared test assesses the independence between gender and survival status. It
# H0: There is no association between gender and survival status.
# H1: There is a significant association between gender and survival status.
contingency_table_gender_status <- table(lung$sex, lung$status)
contingency_table_gender_status
```

```
##
##       1   2
##   1  26 112
##   2  37  53
```

```r
chi_squared_result_gender_status <- chisq.test(contingency_table_gender_status)
print(chi_squared_result_gender_status)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  contingency_table_gender_status
## X-squared = 12.42, df = 1, p-value = 0.0004247
```

```r
#The Fisher's Exact Test is a statistical test used to determine if there are nonrandom associations be

# Example using Fisher's Exact Test
# Assuming you have a 2x2 contingency table named 'my_table'
my_table <- matrix(c(10, 5, 8, 15), nrow = 2)
rownames(my_table) <- c("Group 1", "Group 2")
colnames(my_table) <- c("Category A", "Category B")

# Perform Fisher's Exact Test
fisher_test_result <- fisher.test(my_table)
print(fisher_test_result)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  my_table
## p-value = 0.09597
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   0.7905252 18.7989457
```

```
## sample estimates:
## odds ratio
##   3.612716
```

10 Regression

```
# Remove rows with NA values
lung_no_na <- na.omit(lung)


# Create a correlation matrix for numeric variables
correlation_matrix <- cor(lung_no_na)

# Display the correlation matrix
print(correlation_matrix)
```

```
##                 inst        time      status         age          sex
## inst       1.00000000  0.02462876 -0.12719801  0.04859452  0.084362910
## time       0.02462876  1.00000000 -0.16217237 -0.07854153  0.114149616
## status    -0.12719801 -0.16217237  1.00000000  0.15911933 -0.218780030
## age        0.04859452 -0.07854153  0.15911933  1.00000000 -0.125280356
## sex        0.08436291  0.11414962 -0.21878003 -0.12528036  1.000000000
## ph.ecog    0.05947203 -0.19116847  0.23805821  0.30865378 -0.005363288
## ph.karno  -0.02252266  0.09487913 -0.16127595 -0.32261297 -0.019623924
## pat.karno  0.04147893  0.17505701 -0.18542442 -0.23989736  0.071014942
## meal.cal   0.09869124  0.07467151  0.02483564 -0.23958240 -0.171044801
## wt.loss   -0.17485406  0.03342528  0.04868879  0.04286056 -0.169892775
##                ph.ecog    ph.karno   pat.karno    meal.cal     wt.loss
## inst       0.059472025 -0.02252266  0.04147893  0.09869124 -0.17485406
## time      -0.191168469  0.09487913  0.17505701  0.07467151  0.03342528
## status     0.238058213 -0.16127595 -0.18542442  0.02483564  0.04868879
## age        0.308653782 -0.32261297 -0.23989736 -0.23958240  0.04286056
## sex       -0.005363288 -0.01962392  0.07101494 -0.17104480 -0.16989278
## ph.ecog    1.000000000 -0.82269739 -0.54719617 -0.10563039  0.17125740
## ph.karno  -0.822697393  1.00000000  0.53502749  0.05385409 -0.12524032
## pat.karno -0.547196168  0.53502749  1.00000000  0.17465190 -0.18213953
## meal.cal  -0.105630385  0.05385409  0.17465190  1.00000000 -0.11134425
## wt.loss    0.171257402 -0.12524032 -0.18213953 -0.11134425  1.00000000
```

```
# Making sex variable binary
lung_no_na$sex <- ifelse(lung_no_na$sex == 1, 1, 0)


# Fit a Linear Regression (OLS) - Predicting Weight loss based on age and sex
ols_model <- lm(wt.loss ~ age + sex, data = lung_no_na)

# Display the summary of the OLS
summary(ols_model)
```

```
##
## Call:
```

```
## lm(formula = wt.loss ~ age + sex, data = lung_no_na)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -31.318  -7.761  -3.331   5.335  56.988
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.89806    7.07971   0.692   0.4900
## age          0.03184    0.11263   0.283   0.7778
## sex          4.58576    2.12746   2.156   0.0326 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.26 on 164 degrees of freedom
## Multiple R-squared:  0.02934,    Adjusted R-squared:  0.0175
## F-statistic: 2.478 on 2 and 164 DF,  p-value: 0.08702
```

```
# Fit a Generalized Linear Model (GLM) - Predicting weight loss based on age and sex with gaussian erro
glm_model_gau <- glm(wt.loss ~ age + sex, data = lung_no_na, family = gaussian)

# Display the summary of the GLM
summary(glm_model_gau)
```

```
##
## Call:
## glm(formula = wt.loss ~ age + sex, family = gaussian, data = lung_no_na)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.89806    7.07971   0.692   0.4900
## age          0.03184    0.11263   0.283   0.7778
## sex          4.58576    2.12746   2.156   0.0326 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 175.8545)
##
##     Null deviance: 29712  on 166  degrees of freedom
## Residual deviance: 28840  on 164  degrees of freedom
## AIC: 1342.2
##
## Number of Fisher Scoring iterations: 2
```

```
# Making censoring status variable binary
lung_no_na$status <- ifelse(lung_no_na$status == 1, 1, 0)

# Fit a Generalized Linear Model (GLM) - Predicting weight loss based on age and sex with binomial erro
glm_model_bin <- glm(status ~ age + sex, data = lung_no_na, family = binomial(link = "logit"))

# Display the summary of the GLM
summary(glm_model_bin)
```

```
##
```

```
## Call:
## glm(formula = status ~ age + sex, family = binomial(link = "logit"),
##     data = lung_no_na)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.66095    1.19566   1.389  0.16479
## age         -0.03353    0.01926  -1.741  0.08175 .
## sex         -0.92189    0.35730  -2.580  0.00987 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 198.50  on 166  degrees of freedom
## Residual deviance: 187.59  on 164  degrees of freedom
## AIC: 193.59
##
## Number of Fisher Scoring iterations: 4
```

```
# Calculate 95% confidence intervals for coefficients
conf_intervals <- confint(glm_model_bin, level = 0.95)
```

```
## Waiting for profiling to be done...
```

```
# Interpretation of coefficients and odds ratios for logit
coef_summary <- summary(glm_model_bin)

# Display coefficients, odds ratios, and 95% confidence intervals
result <- cbind(
  Estimate = coef_summary$coefficients[, 1],
  Odds_Ratio = exp(coef_summary$coefficients[, 1]),
  Lower_CI = conf_intervals[, 1],
  Upper_CI = conf_intervals[, 2]
)

# Print the results
print(result)
```

```
##               Estimate Odds_Ratio    Lower_CI    Upper_CI
## (Intercept)  1.66095304  5.2643256 -0.68463620  4.03081409
## age         -0.03352885  0.9670270 -0.07182838  0.00411253
## sex         -0.92188768  0.3977675 -1.63055416 -0.22473077
```

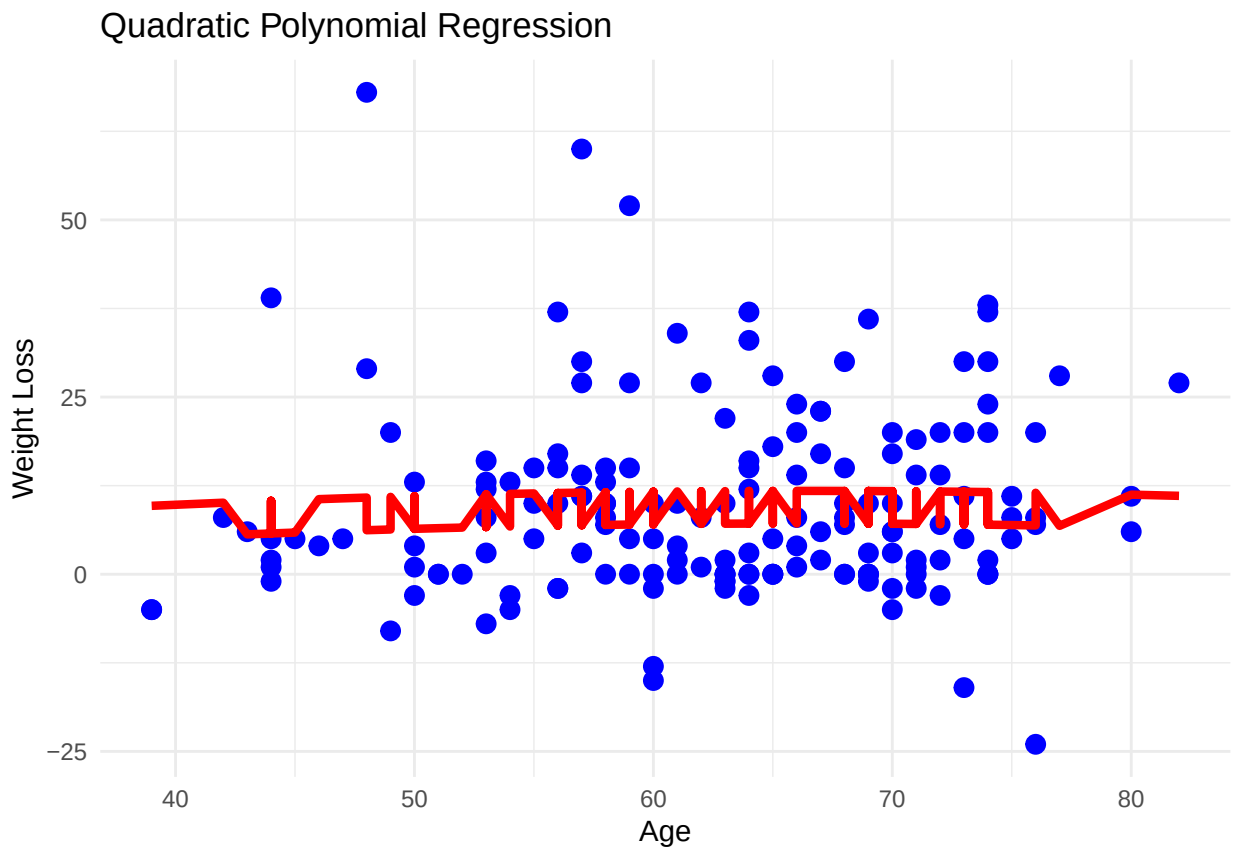
```
# Fit a Quadratic Polynomial Regression  - Predicting Weight loss based on age and sex
poly_model <- lm(wt.loss ~ poly(age, degree = 2) + sex, data=lung_no_na)

# Display the summary of the poly model
summary(poly_model)
```

```
##
## Call:
```

```
## lm(formula = wt.loss ~ poly(age, degree = 2) + sex, data = lung_no_na)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.890  -7.909  -3.492   5.261  57.185
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)                6.880      1.671   4.118 6.06e-05 ***
## poly(age, degree = 2)1     3.766     13.404   0.281   0.7791
## poly(age, degree = 2)2    -3.669     13.303  -0.276   0.7831
## sex                        4.602      2.134   2.156   0.0325 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.3 on 163 degrees of freedom
## Multiple R-squared:  0.02979,    Adjusted R-squared:  0.01193
## F-statistic: 1.668 on 3 and 163 DF,  p-value: 0.1759
```

```
# Create a data frame for plotting
plot_data <- data.frame(age = lung_no_na$age, wt.loss = lung_no_na$wt.loss, fitted = predict(poly_model)

# Create the scatter plot with a fitted curve using ggplot2
ggplot(plot_data, aes(x = age, y = wt.loss)) +
  geom_point(color = "blue", size = 3) +
  geom_line(aes(y = fitted), color = "red", size = 1.5) +
  labs(x = "Age", y = "Weight Loss") +
  ggtitle("Quadratic Polynomial Regression") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
#load spline package
library(splines)

# Fit a linear regression model with cubic spline for 'age'
spline_model <- lm(wt.loss ~ ns(age, df = 5) + sex, data = lung_no_na)

# Display the summary of the spline model
summary(spline_model)
```

```
##
## Call:
## lm(formula = wt.loss ~ ns(age, df = 5) + sex, data = lung_no_na)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.328  -8.689  -2.776   5.502  55.882
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -1.1818     6.5106  -0.182   0.8562
## ns(age, df = 5)1   6.8330     6.4021   1.067   0.2874
## ns(age, df = 5)2   7.3556     8.0603   0.913   0.3628
## ns(age, df = 5)3   0.8836     6.0306   0.147   0.8837
## ns(age, df = 5)4  23.4899    15.8435   1.483   0.1401
## ns(age, df = 5)5   8.2606     8.7688   0.942   0.3476
## sex                4.8710     2.1537   2.262   0.0251 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.31 on 160 degrees of freedom
## Multiple R-squared:  0.04665,    Adjusted R-squared:  0.0109
## F-statistic: 1.305 on 6 and 160 DF,  p-value: 0.2579
```

```
# Create a data frame for plotting
# plot_data <- data.frame(age = seq(min(lung_no_na$age), max(lung_no_na$age), length.out = 100))
#
# # Create a data frame for plotting
# plot_data <- data.frame(age = seq(min(lung_no_na$age), max(lung_no_na$age), length.out = 100))
# plot_data$fitted_spline <- predict(spline_model, newdata = data.frame(age = plot_data$age))
#
# # Create the spline regression plot using ggplot2
# ggplot(lung_no_na, aes(x = age, y = wt.loss)) +
#   geom_point(color = "blue", size = 3) +
#   geom_line(data = plot_data, aes(x = age, y = fitted_spline), color = "green", size = 1.5) +
#   labs(x = "Age", y = "Weight Loss") +
#   ggtitle("Cubic Spline Regression") +
#   theme_minimal()



# Assuming 'cut_point' is the cutoff point for the regression discontinuity
cut_point <- 50

# Create a variable indicating whether the observation is above or below the cutoff
lung_no_na$above_cutoff <- ifelse(lung_no_na$age >= cut_point, 1, 0)

# Fit a regression discontinuity model
rd_model <- lm(wt.loss ~ above_cutoff + sex, data = lung_no_na)

#Display the summary
summary(rd_model)
```

```
##
## Call:
## lm(formula = wt.loss ~ above_cutoff + sex, data = lung_no_na)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -30.681  -8.681  -3.375   5.625  55.134
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)     8.172      3.521   2.321   0.0215 *
## above_cutoff   -1.492      3.488  -0.428   0.6695
## sex             4.694      2.111   2.223   0.0276 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.26 on 164 degrees of freedom
## Multiple R-squared:  0.02995,    Adjusted R-squared:  0.01812
```

```
## F-statistic: 2.531 on 2 and 164 DF,  p-value: 0.08266
```

```
# Create a data frame for plotting
plot_data <- data.frame(
  age = lung_no_na$age,
  wt.loss = lung_no_na$wt.loss,
  above_cutoff = lung_no_na$above_cutoff,
  fitted_values = predict(rd_model)
)


# Create the first ggplot graph
plot1 <- ggplot(plot_data, aes(x = age, y = fitted_values, color = factor(above_cutoff))) +
  geom_point(aes(y = wt.loss), size = 3, alpha = 0.5) +
  geom_smooth(method = 'lm') +
  labs(x = "Age", y = "Fitted Values") +
  ggtitle("Above Cutoff") +
  theme_minimal()

# Create the second ggplot graph
plot2 <- ggplot(plot_data, aes(x = age, y = wt.loss)) +
  geom_point(aes(y = wt.loss), size = 3, alpha = 0.5) +
  geom_smooth(method = 'lm') +
  labs(x = "Age", y = "Fitted Values") +
  ggtitle("Below Cutoff") +
  theme_minimal()

# Arrange the plots side by side
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.3.2
```

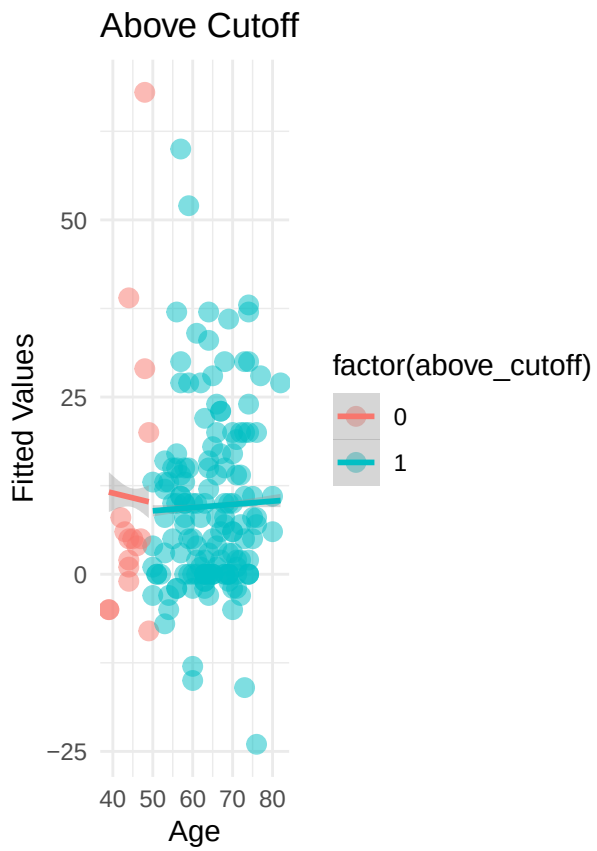
```
##
## Attaching package: 'gridExtra'
```

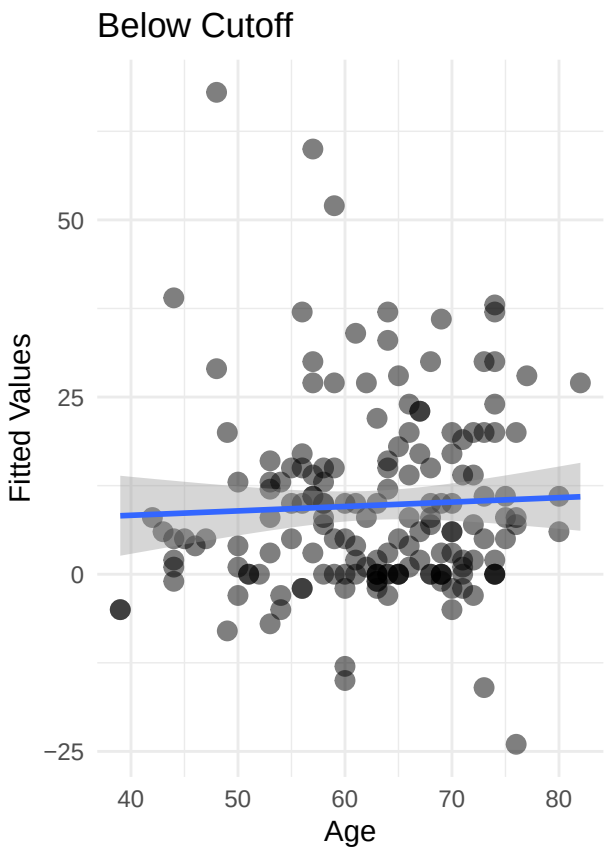
```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(plot1, plot2, ncol = 2)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## `geom_smooth()` using formula = 'y ~ x'
```






```
# Load the 'quantreg' package
# install.packages("quantreg")
library(quantreg)
```

```
## Warning: package 'quantreg' was built under R version 4.3.1
```

```
## Loading required package: SparseM
##
## Attaching package: 'SparseM'
##
## The following object is masked from 'package:base':
##
##     backsolve
##
##
## Attaching package: 'quantreg'
##
## The following object is masked from 'package:survival':
##
##     untangle.specials
```

```
# Fit quantile regression model (e.g., median, 0.5 quantile)
quantile_model <- rq(wt.loss ~ age + sex, data = lung_no_na, tau = 0.5)
```

```r
# Display the summary of the quantile regression model
summary(quantile_model)
```

```
## Warning in rq.fit.br(x, y, tau = tau, ci = TRUE, ...): Solution may be
## nonunique
```

```
##
## Call: rq(formula = wt.loss ~ age + sex, tau = 0.5, data = lung_no_na)
##
## tau: [1] 0.5
##
## Coefficients:
##             coefficients lower bd  upper bd
## (Intercept)  -2.00000    -12.25563  12.25680
## age           0.09091     -0.16556   0.28955
## sex           5.18182      1.80983   7.69876
```