StatisticalModellingTechniques

```
library(readxl)
veri1 <- read_excel("C:/veri1.xls")
View(veri1)

library(dplyr)library(ggplot2)
library(corrplot)

Warning: package 'corrplot' was built under R version 4.3.3

corrplot 0.92 loaded

# Install the reshape2 package if it is not already installed
if (!requireNamespace("reshape2", quietly = TRUE)) {
  install.packages("reshape2")
}
library(reshape2)

head(veri1)

# A tibble: 6 × 9
  index name       date_of_birth       height weight spike block
position_number
  <dbl> <chr>      <dttm>               <dbl>  <dbl> <dbl> <dbl>
<dbl>
1     0 Angelina … 1998-04-13 00:00:00    193     80   320   305
3
2     1 Svetlana … 1996-05-15 00:00:00    182     71   295   284
1
3     2 Ekaterina… 1996-06-17 00:00:00    190     72   306   296
2
4     3 Kristina … 1997-06-17 00:00:00    176     62   288   278
6
5     4 Ekaterina… 1996-12-07 00:00:00    181     70   290   275
1
6     5 Victoria … 1996-03-17 00:00:00    186     67   306   297
3
# ℹ 1 more variable: country <dbl>

data <- na.omit(veri1)
```

```r
summary(veri1)
```

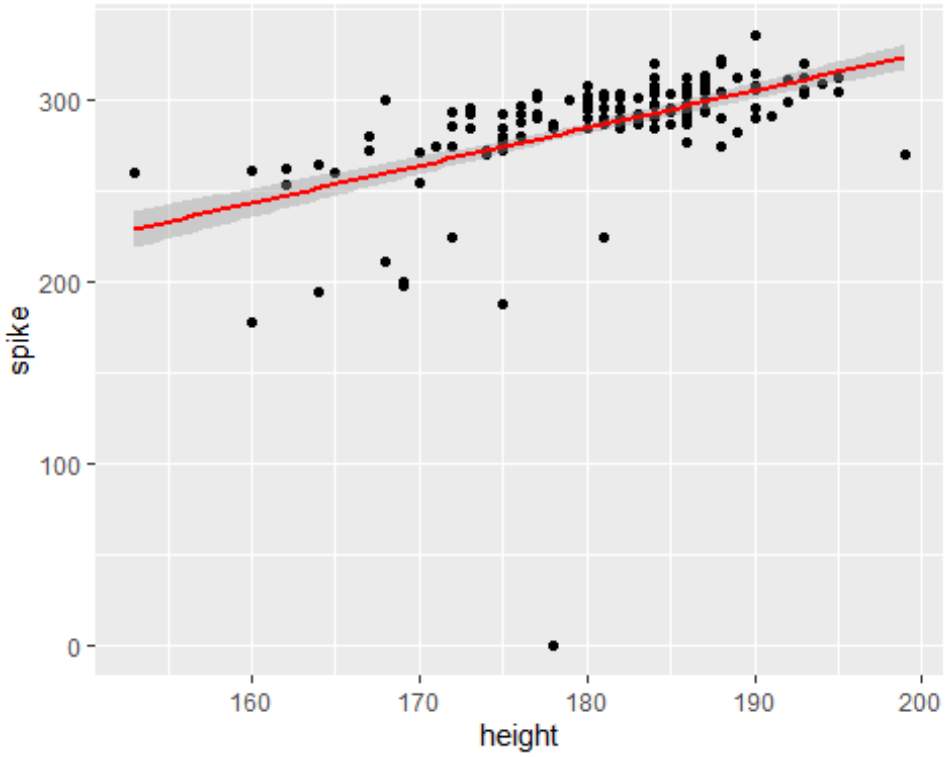
```
     index           name           date_of_birth                
 Min.   :  0.0   Length:432         Min.   :1996-01-03 00:00:00  
 1st Qu.:107.8   Class :character   1st Qu.:1996-05-26 18:00:00  
 Median :215.5   Mode  :character   Median :1997-01-11 12:00:00  
 Mean   :215.5                      Mean   :1997-03-27 10:20:00  
 3rd Qu.:323.2                      3rd Qu.:1997-07-30 00:00:00  
 Max.   :431.0                      Max.   :2000-11-25 00:00:00  
     height          weight          spike           block      
 Min.   :153.0   Min.   :52.00   Min.   :  0.0   Min.   :  0.0  
 1st Qu.:175.8   1st Qu.:63.75   1st Qu.:285.0   1st Qu.:273.5  
 Median :182.0   Median :69.50   Median :293.5   Median :283.0  
 Mean   :181.0   Mean   :68.74   Mean   :286.8   Mean   :275.5  
 3rd Qu.:187.0   3rd Qu.:73.00   3rd Qu.:304.0   3rd Qu.:292.0  
 Max.   :199.0   Max.   :87.00   Max.   :336.0   Max.   :310.0  
 position_number    country     
 Min.   :1.000   Min.   : 5.00  
 1st Qu.:2.000   1st Qu.:10.00  
 Median :2.000   Median :21.00  
 Mean   :2.757   Mean   :19.70  
 3rd Qu.:3.000   3rd Qu.:26.75  
 Max.   :6.000   Max.   :31.00  
```

```r
# Simple Linear Regression Model: spike ~ height + weight

model <- lm(spike ~ height + weight, data = veri1)

summary(model)
```

```

Call:
lm(formula = spike ~ height + weight, data = veri1)

Residuals:
     Min       1Q   Median       3Q      Max 
-276.511   -5.214    4.271   14.652   42.743 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -122.2271    33.2514  -3.676 0.000267 ***
height         2.6569     0.2299  11.555  < 2e-16 ***
weight        -1.0449     0.2572  -4.063 5.76e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 30.15 on 429 degrees of freedom
Multiple R-squared:  0.2619,    Adjusted R-squared:  0.2584 
F-statistic:  76.1 on 2 and 429 DF,  p-value: < 2.2e-16
```

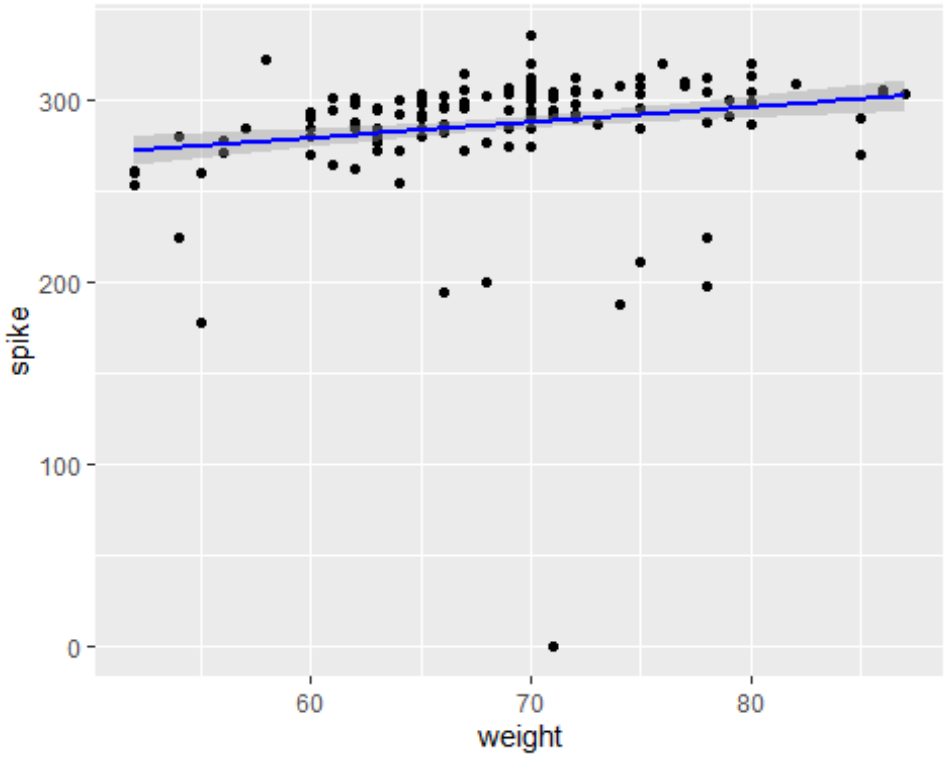
```
# Graphical representation of the model (for height)
ggplot(data, aes(x = height, y = spike)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red")
```

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




```
# Graphical representation of the model (for weight)

ggplot(data, aes(x = weight, y = spike)) +
  geom_point() +
  geom_smooth(method = "lm", col = "blue")
```

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




```r
# Model summary
model_summary <- summary(model)

# Obtaining standard errors of coefficients
standard_errors <- model_summary$coefficients[, "Std. Error"]
# Print standard errors
print(standard_errors)
```

```
(Intercept)      height      weight
 33.2514339   0.2299314   0.2571781
```

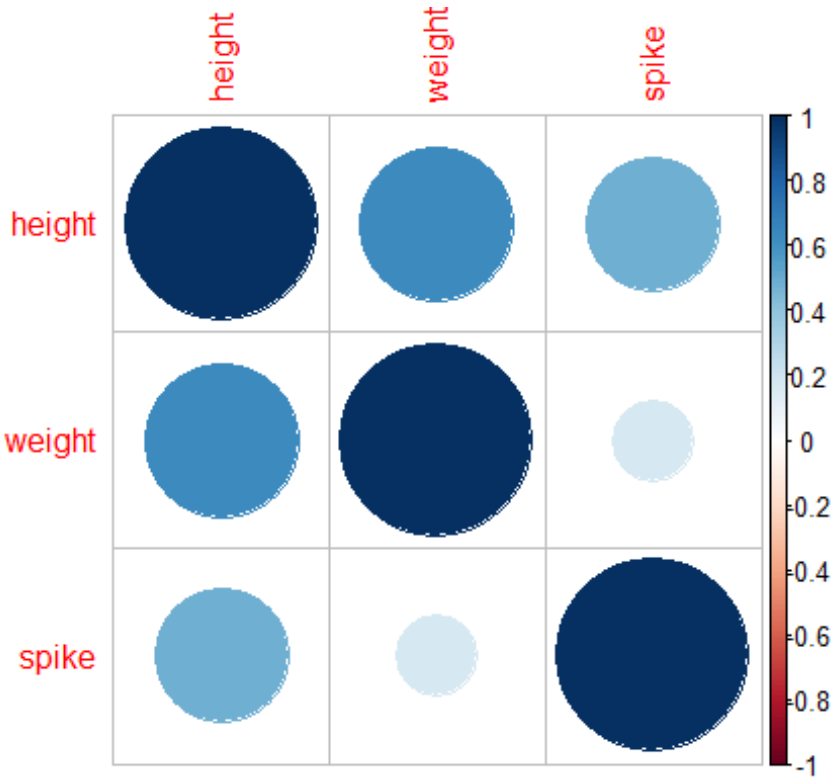
```r
# Obtaining R^2 value
r_squared <- model_summary$r.squared
print(r_squared)
```

```
[1] 0.2618823
```

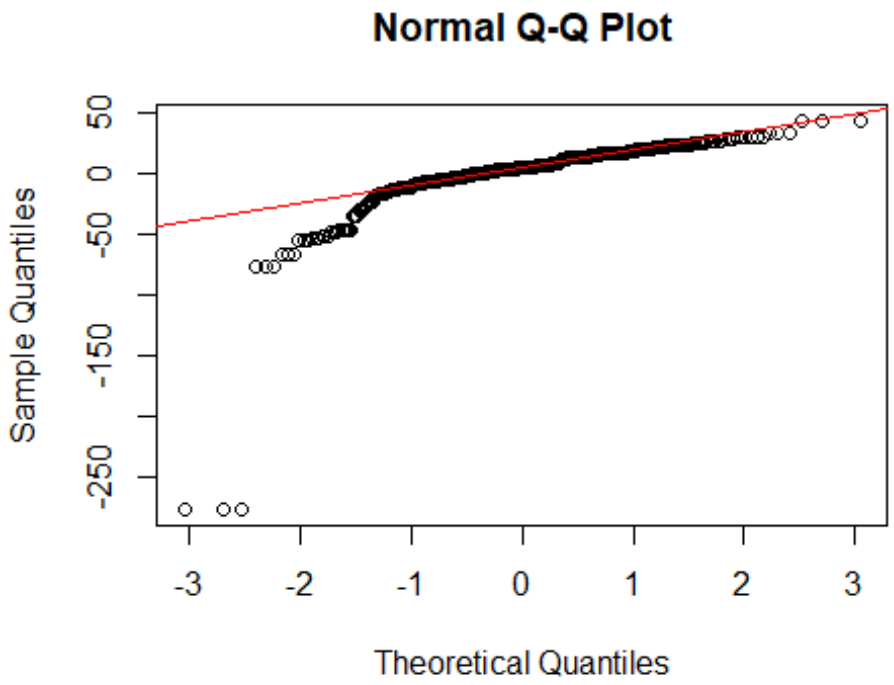
```r
# Calculate the correlation matrix
correlation_matrix <- cor(veri1[, c("height", "weight", "spike")]) # Seçili
sütunlar için
print(correlation_matrix)
```

```
          height    weight     spike
height 1.0000000 0.6392886 0.4831985
weight 0.6392886 1.0000000 0.1793112
spike  0.4831985 0.1793112 1.0000000
```

```r
corrplot(correlation_matrix, method = "circle")
```



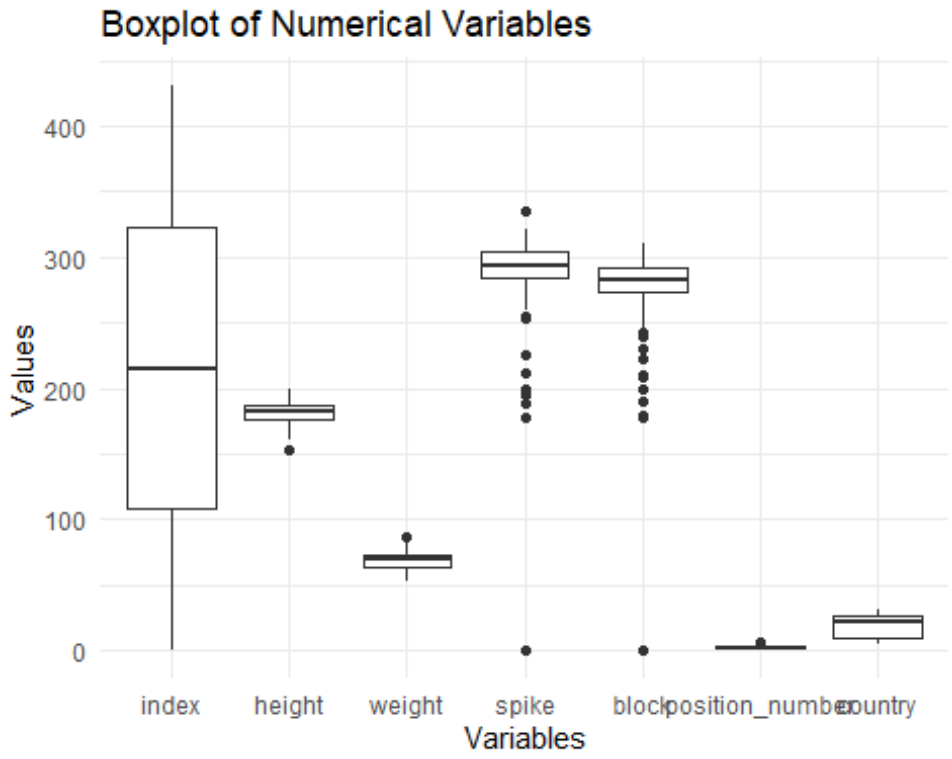

```r
# Checking whether the regression residuals have a normal distribution

# Q-Q chart
qqnorm(model$residuals)
qqline(model$residuals, col = "red")
```




```
# Corrected section for detection of outliers
for(column in colnames(veri1)) {
  if(is.numeric(veri1[[column]])) {
    Q1 <- quantile(veri1[[column]], 0.25, na.rm = TRUE)
    Q3 <- quantile(veri1[[column]], 0.75, na.rm = TRUE)
    IQR <- Q3 - Q1
    lower_bound <- Q1 - 1.5 * IQR
    upper_bound <- Q3 + 1.5 * IQR
    outliers <- sum(veri1[[column]] < lower_bound | veri1[[column]] >
upper_bound, na.rm = TRUE)
    cat(column, "için aykırı değer sayısı:", outliers, "\n")
  }
}
```

```
index için aykırı değer sayısı: 0
height için aykırı değer sayısı: 3
weight için aykırı değer sayısı: 3
spike için aykırı değer sayısı: 36
block için aykırı değer sayısı: 39
position_number için aykırı değer sayısı: 42
country için aykırı değer sayısı: 0
long_veri1 <- melt(veri1, id.vars = NULL)

Warning: attributes are not identical across measure variables; they will be
dropped
```

```r
numerics <- veri1 %>% select_if(is.numeric)

long_numerics <- melt(numerics, id.vars = NULL)

ggplot(long_numerics, aes(x = variable, y = value)) +
  geom_boxplot() +
  theme_minimal() +
  labs(title = "Boxplot of Numerical Variables", x = "Variables", y =
"Values")
```



1. **Loading and Checking Data:**
    - We used a tool called **readxl** to get data from an Excel file. After loading, we looked at the data quickly using something called **View**.
2. **Preparing the Data:**
    - We removed any missing information with a step called **na.omit**. This makes sure our analysis is accurate.
3. **Making the Regression Model:**
    - We used a function called **lm** to see how two things we measured (**height** and **weight**) can predict another thing (**spike**). After making the model, we checked a summary to understand how well it works.
4. **Evaluating the Model:**
    - We looked at some numbers from the model's summary (like standard errors of coefficients and R^2 value) and a correlation matrix to understand the relationship between the things we measured.
5. **Checking for Outliers:**
    - We used a method called IQR to find any unusual values in each thing we measured. This helps us identify any data that looks strange.
6. **Checking if Residuals are Normally Distributed:**
    - We checked if the leftovers (residuals) from our model fit a normal distribution by drawing a type of graph called Q-Q plot and by looking at outliers. This checks if our model's predictions are reliable.

This process goes through basic steps to make and check a regression model with data ready for analysis. Each step is important for making sure the analysis is done right.

1- Explain
vignettes in R, basic symbols in R,NULL, NA, NaN in programming - Missing values, censored and truncated variables ,kernel density,apply function and its types in R

1. vignettes in R

Vignettes are long-form documentation used to introduce R packages. They explain how to use a package, its functions, and include examples from real-world applications. You can access a package's vignettes with the vignette() function. For example, to see the vignettes for the ggplot2 package, you use the command vignette(package = "ggplot2").

2. Basic Symbols in R

In the R programming language, there are some basic symbols used to represent various operations. For example:

• +, -, *, / : It is used for addition, subtraction, multiplication and division operations respectively.

• <- or =:It is used for value assignment.

• #:It is used to comment a line.

• ==, !=, <, >, <=, >=:They are used as comparison operators.

- &, |: Used for logical "and" and "or" operations.
- " %*% " : For matrix multiplication.

3. NULL, NA, NaN: Missing Values, Censored, and Truncated Variables

- **NULL**: Indicates that an object does not exist. For instance, it is used to denote that a variable has not yet been assigned any value.
- **NA (Not Available)**: Shows that a data point is missing. It is commonly used to represent missing values in data.
- **NaN (Not a Number)**: Represents a result that is not numeric, typically emerging in situations where a calculation is undefined (such as division by zero).

4. Kernel Density

Kernel density estimation is a method used to guess the probability density function of a dataset. This method places kernels around data points and then sums these kernels to make a smooth curve. Kernel density estimation provides a visual understanding of the shape of data distribution and is a non-parametric method.

5. Apply Functions and Their Types

In R, the apply function and its variations (like lapply, sapply, mapply) are used to apply a function over a data structure:

- **apply**: Applies a function across rows or columns of matrices or arrays.
- **lapply**: Applies a function to each element of a list and returns the results as a list.
- **sapply**: Works like lapply but tries to simplify the result into a vector or matrix if possible.
- **mapply**: Applies a function to multiple lists or vectors, working on corresponding elements of each.

These functions are used to make data analysis and manipulation operations vectorized, meaning they can handle multiple data points without the need for explicit loops.