Data Preparation Activities

1) There are several aspects of data that need to be assessed to increase the quality of the data and hence, improve the analysis results.
   a) Accuracy
   b) Completeness
   c) Consistency
   d) Reliability
   e) Timeliness

   These are some of the quality checks that can be done on the dataset before using the data. The quality checks can vary in each scenario, and it may not be necessary to have your data accurate, complete, consistent, unique, and up to date each time we are preparing the data.

   For e.g., when working with financial datasets, precision and accuracy of the data is of utmost importance for financial year-end reporting in organizations. Accuracy and completeness therefore take a higher precedence for quality checks over timeliness (since financial reporting involves using data from previous years). A small discrepancy in the financial data can account for significant business consequences like setting inaccurate sales projections.

   E-commerce projects, for e.g., rely on their data set to be relevant and consistent with their needs. Since the primary focus is centered around the customer behavior and their e-commerce platform, the priority is to acquire accurate and consistent data. Similarly, for analysis of historical trends accessibility and longevity of data takes precedence over other data quality attributes. Ease of retrieval and data storage for longer periods of time is prioritized so that a successful historical trend or analysis can be run on the data.

2) Handling missing values in tuples is a very common challenge when analyzing data or creating machine learning models. There are many ways to handle this problem some of which are elaborated below:
   f) **Median method**: If the tuples consist of only quantitative data and contain some missing values, the median method is one of the ways to replace the missing values. In this method, the non-null/non-blank fields are arranged in ascending order. The median is then computed, and all the missing values are replaced with the median values. This maintains the overall distribution of the dataset.
   g) **Deleting missing values**: In this method values that are missing in the dataset are removed from the dataset. This approach is used when the missing values in the dataset contribute to only a small percentage of the overall dataset.
   h) Mean and mode method: This approach is like the median method. Except, instead of replacing the missing values with the median, we can replace them with the mean or mode of the quantitative dataset. This method maintains the distribution of the dataset.
   i) **Constant value:** In some cases, we can fill the missing values with a constant value that is predefined. To use this approach, it is necessary that the blanks are meaningful and can be differentiated with the rest of the data.

3)

a) **Mean**: 29.963
Median: 25

```
> age <- c(13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 3
5, 35, 35, 36, 40, 45, 46, 52, 70)
> age
 [1] 13 15 16 16 19 20 20 21 22 22 25 25 25 25 30 33 33 35 35 35 35 36 40 45 46 52 70
> mean(age)
[1] 29.96296
> median(age)
[1] 25
```

b) **Mode**: 25 and 35
The dataset is **bimodal** since 1) it contains 2 modes and 2) the frequency distribution of this dataset has 2 peaks at 25 and 35.

```
> install.packages("DescTools")
```

```
> library(DescTools)
> Mode(age$age)
[1] 25 35
attr(,"freq")
[1] 4
```

c) To calculate the **midrange**: (minimum + maximum)/2 = 41.5

```
> mean(range(age))
[1] 41.5
```

d) The **first quartile** is the 1/4th (n +1) value = 7th value = 20.5
The **third quartile** is the 3/4th (n+1) value = 21st value = 35

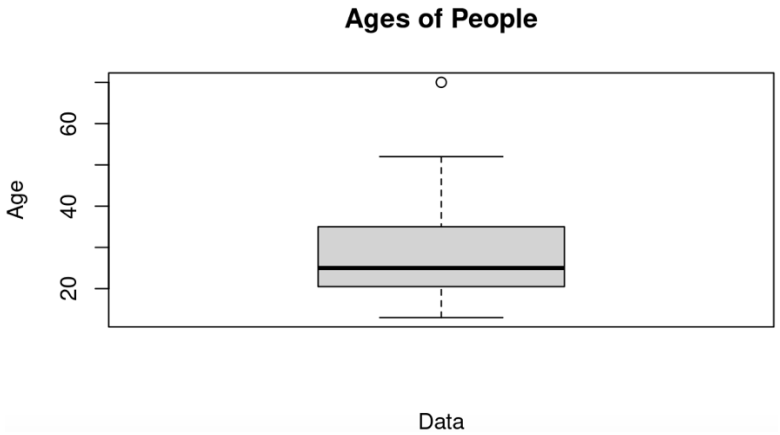
```
> Q1Q3<-quantile(age, probs = c(0,0.25,0.5,0.75,1))
> Q1Q3
  0%  25%  50%  75% 100%
13.0 20.5 25.0 35.0 70.0
```

e) Min. 1st Qu. Median 3rd Qu. Max.
13.00 20.50 25.00 35.00 70.00

```
> fivenum(age)
[1] 13.0 20.5 25.0 35.0 70.0
```

f) **BoxPlot:**

```
> boxplot(age,data=age, main="Ages of People", xlab="Data", ylab="Age")
```



g) Differences between Quantile vs Quantile-Quantile plots:

Quantile	Quantile - Quantile
Quantile plots directly display the quantiles of a set of values. There is no comparison here.	Quantile-quantile plots allow us to compare the quantiles of two sets of numbers/ samples to check if they come from same population.
This is a basic and simple plot.	This kind of comparison is much more detailed than a simple comparison of means or medians.
The sample quantiles are plotted against the fraction of the sample they correspond to.	Plots quantiles of 1 dataset against quantiles of second data set.
Example: Plotting the score of a football team in one game.	Example: Compare tails or distribution shape.
Quantile plots directly display the quantiles of a set of values.	Requires normalized datasets. If the data is normally distributed, the points will fall on the 45-degree reference line. If the data is not normally distributed, the points will deviate from the reference line.

Part 2

1) The dataset we used is the Social Power NBA obtained from Kaggle. This dataset contains player performance scores, their salary, social media activity, and other seasonal data. The author of this data collected the data points from multiple sources, compiled, and used for data analysis. Sources include ESPN, Twitter, Wikipedia, and NBA.

This dataset answers, for example, the following questions: What is the worth of athletes in terms of salary in the NBA? How does player statistics affect their salary?
The dataset is saved in csv format. It is delimited with "," (commas). This file can be opened in r using read.csv() function.

2) Read the data using read.csv() and assigned it to data.df:

```
> data <- read.csv("nba_2017_players_stats_combined.csv")
> data <- data.frame(data)
> head(data)
  X Rk            PLAYER POSITION AGE   MP   FG  FGA
1 0  1 Russell Westbrook       PG  28 34.6 10.2 24.0
2 1  2      James Harden       PG  27 36.4  8.3 18.9
3 2  3     Isaiah Thomas       PG  27 33.8  9.0 19.4
4 3  4     Anthony Davis        C  23 36.1 10.3 20.3
5 4  5     DeMar DeRozan       SG  27 35.4  9.7 20.9
6 5  6  DeMarcus Cousins        C  26 34.2  9.0 19.9
```

Data cleansing: The first column X with the serial numbers was filtered out:

```
> data<-data[,2:37]
> head(data)
  Rk            PLAYER POSITION AGE   MP   FG  FGA   FG.
1  1 Russell Westbrook       PG  28 34.6 10.2 24.0 0.425
2  2      James Harden       PG  27 36.4  8.3 18.9 0.440
3  3     Isaiah Thomas       PG  27 33.8  9.0 19.4 0.463
```

Renamed the new first column 'Rk' to 'Player Number':

```
> names(data)[names(data) == "Rk"] <- "Player Number"
> names(data[1])
[1] "Player Number"
>
```

3) Our dataframe contains 446 rows and 36 total columns.

```
> nrow(data)
[1] 446
> ncol(data)
[1] 36
```

```
> colnames(data)
 [1] "Player Number" "PLAYER"        "POSITION"
 [4] "AGE"           "MP"            "FG"
 [7] "FGA"           "FG."           "X3P"
[10] "X3PA"          "X3P."          "X2P"
[13] "X2PA"          "X2P."          "eFG."
[16] "FT"            "FTA"           "FT."
[19] "ORB"           "DRB"           "TRB"
[22] "AST"           "STL"           "BLK"
[25] "TOV"           "PF"            "POINTS"
[28] "TEAM"          "GP"            "MPG"
[31] "ORPM"          "DRPM"          "RPM"
[34] "WINS_RPM"      "PIE"           "PACE"
>
```

	Player Number	PLAYER	POSITION	AGE	MP	FG	FGA
1	1	Russell Westbrook	PG	28	34.6	10.2	
2	2	James Harden	PG	27	36.4	8.3	
3	3	Isaiah Thomas	PG	27	33.8	9.0	
4	4	Anthony Davis	C	23	36.1	10.3	
5	5	DeMar DeRozan	SG	27	35.4	9.7	
6	6	DeMarcus Cousins	C	26	34.2	9.0	
7	7	Damian Lillard	PG	26	35.9	8.8	
8	8	LeBron James	SF	32	37.8	9.9	
9	9	Kawhi Leonard	SF	25	33.4	8.6	
10	10	Stephen Curry	PG	28	33.4	8.5	
11	11	Kyrie Irving	PG	24	35.1	9.3	
12	12	Kevin Durant	SF	28	33.4	8.9	
13	13	Karl-Anthony Towns	C	21	37.0	9.8	
14	14	Jimmy Butler	SF	27	37.0	7.5	

Showing 1 to 14 of 446 entries, 36 total columns

Each row in this data measures a single athlete's statistics (Field goals made, Wins Points scored etc) in the 2016-2017 NBA season.

4) Summary of “Age”, “MP” and “FG”, including minimum value, maximum value and mean value for each column.

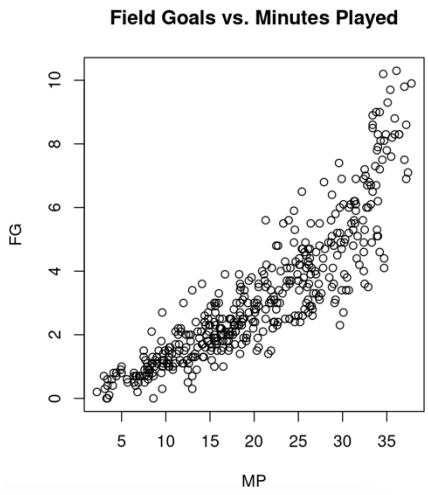
```
> part4 <- data[4:6]
> summarized <- summary(part4)
> summarized
      AGE             MP              FG        
 Min.   :19.00   Min.   : 2.20   Min.   : 0.000  
 1st Qu.:23.00   1st Qu.:13.43   1st Qu.: 1.600  
 Median :26.00   Median :19.90   Median : 2.700  
 Mean   :26.46   Mean   :20.36   Mean   : 3.219  
 3rd Qu.:29.00   3rd Qu.:27.38   3rd Qu.: 4.300  
 Max.   :40.00   Max.   :37.80   Max.   :10.300
```

To check for missing values, we checked the sum of missing values in each column.

```
> sum(is.na(part4[1]))
[1] 0
> sum(is.na(part4[2]))
[1] 0
> sum(is.na(part4[3]))
[1] 0
>
```

5) Below is a scatter plot (non-statistical geom) graph of Field Goals (FG) vs Minutes Played (MP).

```
> plot(data$MP, data$FG, main="Field Goals vs. Minutes Played", xlab = "MP", ylab="FG", pch=1)
```



The graph clearly shows a correlation between MP and FG. Players who played more minutes made a greater number of field goals. It seems that the two variables are in a direct relationship.

To plot a bar chart (statistical geom) an aggregation was used on Position and Points column.

```
> aggregate(data[27], by = data[3], FUN = mean)
  POSITION   POINTS
1        C 8.103333
2       PF 7.552128
3     PF-C 5.400000
4       PG 9.607527
5       SF 9.291026
6       SG 9.032222
> pts_per_pos <- aggregate(data[27], by = data[3], FUN = mean)
```

The PF-C value was found to be an outlier value; therefore, it was removed from the aggregation.

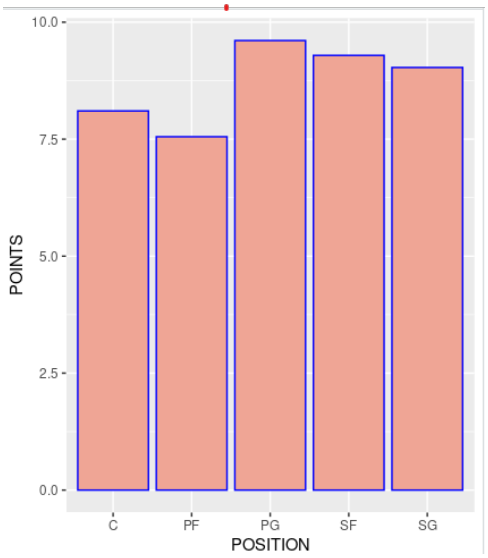
For the final part of this question about "good graphics", we could not find the Chapter 4 reference in the recommended reading sources. Hence, we've skipped it as mentioned on the discussion board.

```
> pts_per_pos[-3,]
  POSITION   POINTS
1        C 8.103333
2       PF 7.552128
4       PG 9.607527
5       SF 9.291026
6       SG 9.032222
```

Alternatively, this value could have been renamed to "PF" or "C", however, since it was 1 value out of the 400+ rows we decided to remove it.

This aggregation was then used to plot the bar chart using the following command:

```
> ggplot(pts_per_pos) + geom_bar(aes(x=POSITION, y = POINTS), s
tat = "identity", color = "blue", fill ="#EFA595")
```



The bar chart depicts the Points scored by players (numerical) in each of the 5 positions (categorical).

The PGs (Point Guards) averaged the most points (almost 10 Points Per Game) in the 2016-17 season. Whereas the PFs (Power Forwards) averaged the lowest points (7.6 Points Per Game).

This chart helps us determine the player positions that contribute the most to a team's success in terms of scoring.

6) To find missing values, we checked the sum of missing values in all the columns.

```
> sum(is.na(data[,4:36]))
[1] 39
```

```
> colSums(is.na(data))
PLAYER NUMBER        PLAYER      POSITION           AGE
            0             0             0             0
           MP            FG           FGA           FG.
            0             0             0             0
          X3P          X3PA          X3P.          X2P
            0             0            31             0
         X2PA          X2P.          eFG.           FT
            0             0             0             0
          FTA           FT.           ORB           DRB
            0             8             0             0
          TRB           AST           STL           BLK
            0             0             0             0
          TOV            PF        POINTS          TEAM
            0             0             0             0
           GP           MPG          ORPM          DRPM
            0             0             0             0
          RPM      WINS_RPM           PIE          PACE
            0             0             0             0
            W
            0
```

7) We used the following code to replace all missing numerical values in X3P% and FT% columns with the column mean.

```
> mean(data$X3P., na.rm= TRUE)
[1] 0.2995542
> data$X3P.[is.na(data$X3P.)]<-mean(data$X3P., na.rm = TRUE)
> sum(is.na(data$X3P.))
[1] 0
```

```
> mean(data$FT., na.rm=TRUE)
[1] 0.7404817
> data$FT.[is.na(data$FT.)] <- mean(data$FT., na.rm=TRUE)
> sum(is.na(data$FT.))
[1] 0
```

For categorical data, our dataset did not have any missing values. Therefore, for the column “Team” we created some missing values.

```
> rows.to.missing <- sample(row.names(data[28]),10)
> rows.to.missing
 [1] "88"  "217" "149" "45"  "90"  "30"  "407" "436" "391" "402"
> data[rows.to.missing,]$TEAM <- NA
> colSums(is.na(data[28]))
TEAM
  10
```

We then ran some analysis on the column to understand the nature of the missing data.

```
> player_per_team<-count(data, TEAM)
> player_per_team[order(player_per_team$n, decreasing = TRUE), ]
          TEAM  n
14         DAL 16
53          SA 16
7          BOS 15
21          GS 15
28         LAC 15
56        UTAH 15
8          CHA 14
17         DEN 14
20         DET 14
34         MIN 14
```

We analyzed the distribution of players across teams. We also calculated the weighted mean of the dataset.

```
> weighted.mean(player_per_team$n)
[1] 7.689655
```

Since the players were scattered across the teams, and our missing values contributed to only 2% of the entire dataset, we decided to populate them with a random sample.

```
> #Take a random sample of Teams
> random_sample<-sample(row.names(data),10)
> #Populate random sample with missing values
> data$TEAM[is.na(data$TEAM)] <-data[random_sample,]$TEAM
> data[random_sample,]$TEAM
 [1] "DAL"     "CLE"     "NO"      "LAL"     "MIN"     "DEN/POR"
 [7] "LAC"     "CHA"     "MIA"     "WSH"
```

8) To normalize the data, we used the column “PACE” to bring the values to a single scale to compare them better.
These were the results:

```
> library(caret)
Loading required package: lattice
> process <- preProcess(as.data.frame(data$PACE), method=c("ran
ge"))
> process
Created from 446 samples and 1 variables

Pre-processing:
  - ignored (0)
  - re-scaling to [0, 1] (1)

> View(process)
> norm_scale <- predict(process, as.data.frame(data$PACE))
> norm_scale
     data$PACE
1   0.68434238
2   0.71231733
3   0.58121086
4   0.59582463
```

	data$PACE
1	0.6843424
2	0.7123173
3	0.5812109
4	0.5958246
5	0.4914405
6	0.4672234
7	0.5745303
8	0.5202505
9	0.4121086
10	0.8000000
11	0.5511482
12	0.7427975
13	0.4668058

9) We created a dummy variable of the position column to bifurcate player positions.

The dummy variable gave the following results:

```
> dummy <- model.matrix(~ 0 + POSITION, data = data)
> dummy
```

	POSITIONC	POSITIONPF	POSITIONPF-C	POSITIONPG	POSITIONSF
1	0	0	0	1	
2	0	0	0	1	
3	0	0	0	1	
4	1	0	0	0	
5	0	0	0	0	
6	1	0	0	0	
7	0	0	0	1	
8	0	0	0	0	
9	0	0	0	0	
10	0	0	0	1	
11	0	0	0	1	

Analysis of the variable showed that the Position column was split into 6 dummy variables, one for each position. Only 1 of these columns will be “1” for each row.

10) Data preparation operations:

- **Random sampling**:

```
> #Sampling training data with 50% of the total
> train.rows <- sample(rownames(data), dim(data)[1]*0.5)
> #Sampling validation data with 30% of the total
> valid.rows <- sample(setdiff(rownames(data), train.rows),dim(data)[1]*0.3)
> #testing data with 20% of the total
> test.rows <- setdiff(rownames(data), union(train.rows, valid.rows))
> #Populating samples
> train.data <- data[train.rows, ]
> valid.data <- data[valid.rows, ]
> test.data <- data[test.rows, ]
>
```

The above code was used to create sample data, 50% for training, 30% for validation and 20% for training.

Below snippet shows a random sample of 90 rows used to populate the testing data.

	PLAYER NUMBER	PLAYER	POSITION	AGE	MP	FG	FGA	FG.	X3P
3	3	Isaiah Thomas	PG	27	33.8	9.0	19.4	0.463	
12	12	Kevin Durant	SF	28	33.4	8.9	16.5	0.537	
16	16	Andrew Wiggins	SF	21	37.2	8.6	19.1	0.452	
20	21	Giannis Antetokounmpo	SF	22	35.6	8.2	15.7	0.521	
22	23	Kyle Lowry	PG	30	37.4	7.1	15.3	0.464	
32	33	Jabari Parker	PF	21	33.9	7.8	16.0	0.490	
33	34	Marc Gasol	C	32	34.2	7.2	15.7	0.459	
34	35	Harrison Barnes	PF	24	35.5	7.6	16.2	0.468	
35	36	Kevin Love	PF	28	31.4	6.2	14.5	0.427	
36	37	Zach LaVine	SG	21	37.2	6.9	15.1	0.459	
51	52	Avery Bradley	SG	26	33.4	6.5	14.1	0.463	
54	55	Victor Oladipo	SG	24	33.2	6.1	13.9	0.442	

Showing 1 to 12 of 90 entries, 37 total columns

- **Data cleaning** (column deletion):
  Columns which were not useful to our analysis were removed. These were columns 29-36 which included advanced player statistics which were not useful for our analysis.

```
> data_final <- data[-c(29:36)]
> data final
```

- **Merge:**
  For our analysis, we merged the player statistics table with the player salary table. To complete this, we analyzed that both the tables have a primary key column “Player Number”. This was used to merge the two tables by using the merge() function.

```
> players_by_salary <- merge(data_final, salary, by = intersect(names(data_fina
l), names(salary))
+ )
> View(players_by_salary)
```

```
> players_by_salary[c(2,1,31)]
    PLAYER NUMBER                  PLAYER SALARY_MILLIONS
1             328            Aaron Brooks            2.70
2             101            Aaron Gordon            4.35
3             482          Aaron Harrison            0.87
4             381           Adreian Payne            2.02
5              78              Al Horford           26.54
6             201             Al Jefferson           10.31
7             185         Al-Farouq Aminu            7.68
```

The dataset can now be used to predict player salaries with respect to their statistics.

11) The only other categorical variable (“POINTS”) was used for binning.

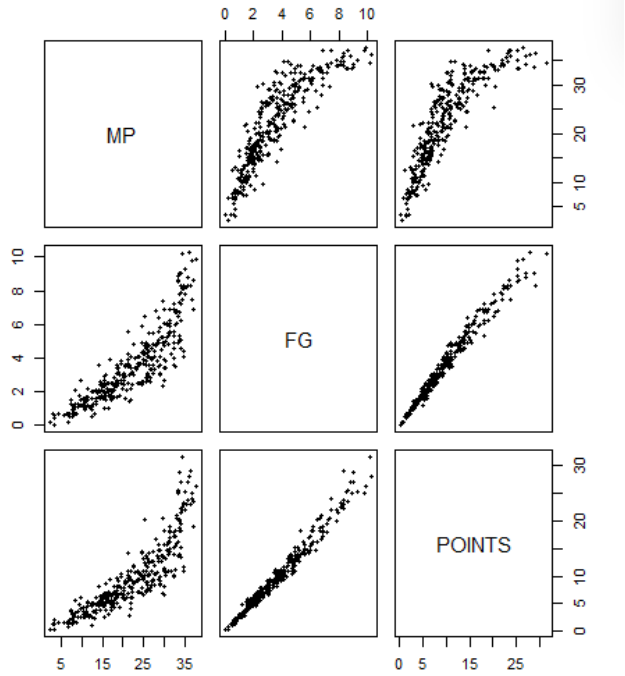
```
> breaks <- c(0, 10, 20, 30, 40, 50, 60 ,70)
```

```
> breaks <- c(0, 5, 10, 15, 20, 25, 30, 35)
> discretized_column <- cut(players_by_salary$POINTS, breaks = breaks, lab
els = c("Bin 1", "Bin 2", "Bin 3", "Bin 4","Bin 5","Bin 6","Bin 7"))
>
> df <- data.frame(numeric_column = players_by_salary$POINTS, discretized_
column)
>
> df
    numeric_column discretized_column
1             31.6              Bin 7
2             25.3              Bin 6
3             12.8              Bin 3
4             12.7              Bin 3
5             12.7              Bin 3
6             12.4              Bin 3
7             12.3              Bin 3
8             12.0              Bin 3
```

The above code was used to discretize the numeric column of “Wins” into bins. Data was discretized into 7 bins.

12) Scatter plot matrix was plotted using following code:

```
> pairs(players_by_salary[,c(5,6,27)], pch = 20)
```

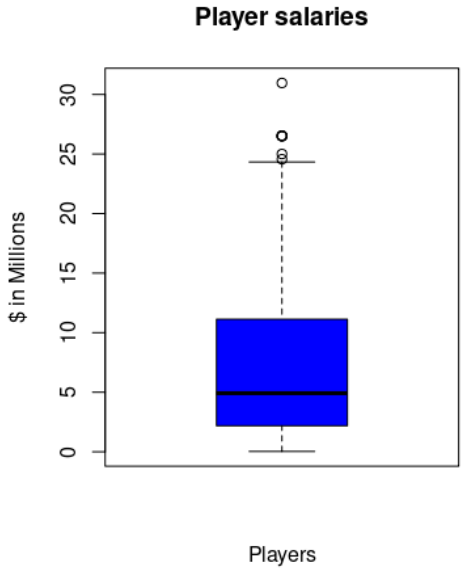


This scatter plot matrix plots Minutes Played, Field Goals and Points. The plot shows a direct relation between points and Field Goals made, as should be the case. A higher number of field goals reflects a greater number of points.

On the other hand, Minutes played has a direct relationship with both Field goals and Minutes. However, the curve starts to flatten out at > 5 FG and > 20 points.

13) To analyze players' salary, we created a **boxplot** of the salary in millions column.

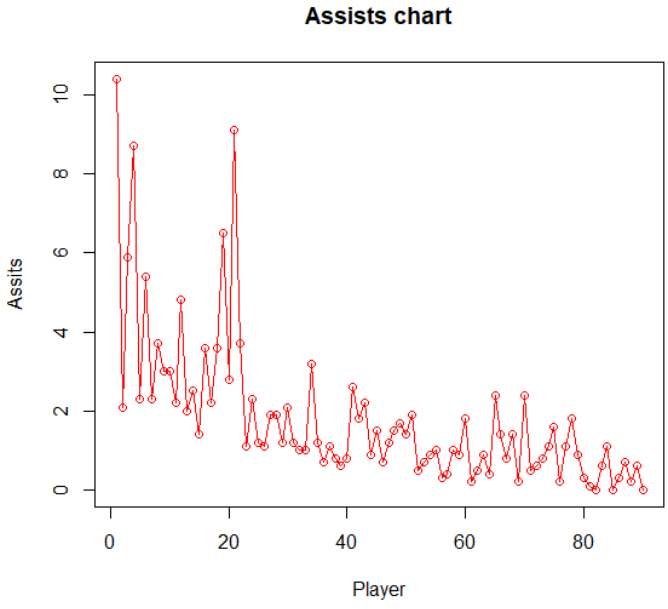
```
> boxplot(players_by_salary$SALARY_MILLIONS , main ="Player salaries", xlab="Players",
ylab="$ in Millions", col="blue")
```



We identified some outliers in the salary column. The mean salary was $5 million, while the highest was above $30 million.

We then used our random samples created in the previous sections to plot a line chart.

```
> plot(test.data$AST,type = "o", col = "red", xlab = "Player", ylab
= "Assits",
+      main = "Assists chart")
```



This line chart shows the number of players and their assists averages. It can be concluded that in this random sample, 7 assists are averaged by only 3 players only.

Whereas more than 100 players average less than 1 assists.

Part 3

Q1. Write two paragraphs discussing a number of error detection and error repairing approaches.

Error detection is the pivotal initial step in the process of cleaning dirty data and improving data quality. It encompasses a series of critical decisions, beginning with the determination of what to detect, how to detect it, and where to carry out the detection, often utilizing the business intelligence layer (BI). Two significant error types are addressed in this context: first, integrity constraints (IC) and denial constraints (pertaining to data quality rules), and second, duplicates that violate these integrity constraints. An illustration of an integrity constraint involves imposing a minimum length requirement, such as 50 characters, for responses in a survey question, thereby preventing errors like excessively brief or empty responses. Conversely, a duplicate error might occur when qualitative data is manually entered into a system, inadvertently leading to the replication of certain data entries. The subsequent step involves determining the methodology for error detection, with two primary approaches: automation and human intervention. Automation entails the utilization of algorithms to detect violations, such as functional dependency breaches, while human involvement may be required for manually identifying duplicates. Lastly, the decision of where to detect errors arises, primarily occurring either in the source database or within the data processing pipeline. While most errors are identified at the source, others can only be detected further downstream in the data processing pipeline. For instance, constraints related to 'discounted partner sales' can only be enforced after aggregating partner types, product prices, and discount tiers (Chu et al, 2016).

Error repairing is equally vital, as it entails taking corrective actions to ensure data accuracy and facilitate informed business decision-making. Error repair strategies involve three essential steps: determining what to repair, how to repair it, and where to implement the repair. When identifying the repair target, three distinct approaches are applied. First, some techniques rely on existing integrity constraints and solely update data to rectify errors. Second, they may place trust in the data itself while relaxing constraints. Third, both data and constraints may undergo modifications. Techniques that exclusively modify data are further categorized based on error types, often focusing on addressing a single type of data-based error at a time. The decision on how to repair data can be divided into two broad categories: fully automated methods and those dependent on human involvement. An example of an automated approach involves minimizing costs by reducing the disparity between the original and modified databases. Human-based approaches may involve verification, suggesting fixes, or training machine learning models to automate repair decisions. Finally, the consideration of where to repair data encompasses either making in-situ adjustments to the database or constructing models that elucidate appropriate repairs. While most techniques opt for in-situ database repair, some may result in database destruction. Model-based approaches involve these models responding to queries, such as sampling duplicates from a range of potential repairs (Chu et al, 2016).

Q2. Write two paragraphs about the need of data cleaning for machine learning approaches. You need to provide two references that you used.

Data cleaning is a critical pre-processing step in the field of machine learning, essential for ensuring the accuracy and reliability of model predictions. Raw data collected from various sources often contain errors, inconsistencies, missing values, and outliers, which can significantly impact the performance of machine learning algorithms. A recent study emphasizes that “high-quality datasets are essential for developing machine learning models. For example, outliers in training dataset can cause instability or non-convergence in ensemble learning. Incomplete, inconsistent, and missing data can lead to drastic degradation in prediction” (Gudivada et al., 2017). As such, the presence of noisy, erroneous, or incomplete data can significantly compromise the performance of machine learning models. Data cleaning techniques are instrumental in addressing these issues by identifying and rectifying anomalies, inconsistencies, and missing values, ensuring that the underlying data is robust and reliable for modeling. This underlines the continued need for data cleaning as an indispensable step in the machine learning pipeline.

Furthermore, data cleaning plays a crucial role in addressing ethical and fairness concerns in machine learning. Recent research by Chouldechova and Roth (2018) emphasizes the importance of ensuring that training data is thoroughly cleaned and curated to mitigate biases. For instance, in the medical field, biased data fed into machine learning models can lead to biased results and unreliable insights which can have adverse effects on patients, doctors and damage hospital reputation. Biased data can perpetuate and amplify societal inequalities when used to train machine learning models, resulting in unfair or discriminatory outcomes. Data cleaning techniques can help detect and rectify biases in the data, fostering more equitable and just machine learning approaches. In the context of evolving ethical considerations, data cleaning takes on added significance, not just as a technical necessity but as a means to uphold ethical standards in machine learning practices.

References

- Chouldechova, A., & Roth, A. (2020). A snapshot of the frontiers of fairness in machine learning. *Communications of the ACM*, *63*(5), 82–89. https://doi.org/10.1145/3376898
- Chu, X., Ilyas, I. F., Krishnan, S., & Wang, J. (2016). Data Cleaning: Overview and Emerging Challenges. *Proceedings of the 2016 International Conference on Management of Data*, 2201–2206. https://doi.org/10.1145/2882903.2912574
- Gudivada, V., Apon, A., & Ding, J. (2017). Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. *International Journal on Advances in Software*, *10*(1), 1-20.