

Introduction to Data Science Topic-5

- Instructor: Professor Henry Horng-Shing Lu,
Institute of Statistics, National Yang Ming Chiao Tung University, Taiwan
Email: henryhslu@nycu.edu.tw
- WWW: <http://misg.stat.nctu.edu.tw/hslu/course/DataScience.htm>
- Classroom: ED B27 (新竹市大學路1001號工程四館B27教室)
- References:
M. A. Pathak, Beginning Data Science with R, 2014, Springer-Verlag.
K.-T. Tsai, Machine Learning for Knowledge Discovery with R: Methodologies for Modeling, Inference, and Prediction, 2021, Chapman and Hall/CRC.
- Evaluation: Homework: 70%, Term Project: 30%
- Office hours: By appointment

Course Outline

10 Topics and 10 Homeworks:

- **Introduction of Data Science**
- **Introduction of R and Python**
- **Cleaning Data into R and Python**
- **Data Visualization**
- **Exploratory Data Analysis**
- **Regression (Supervised Learning)**
- **Classification (Supervised Learning)**
- **Text Mining**
- **Clustering (Unsupervised Learning)**
- **Neural Network and Deep Learning**

Exploratory Data Analysis with R

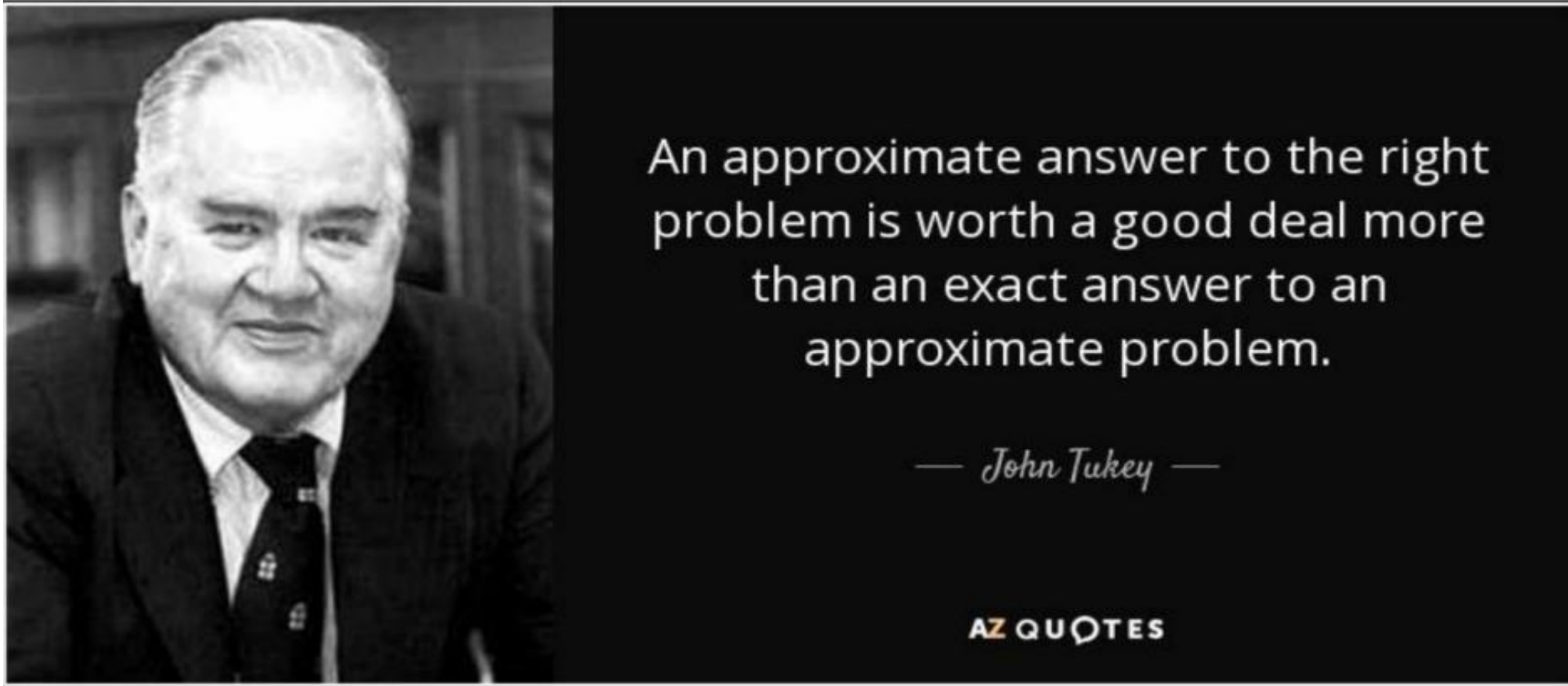
References:

Ch. 5, M. A. Pathak, Beginning Data Science with R, 2014, Springer-Verlag.

<https://www.kaggle.com/code/dongdongxzoez/python-topic-3>

<https://repository.kaust.edu.sa/handle/10754/680231>





John W. Tukey wrote the book *Exploratory Data Analysis* in 1977. Tukey held that too much emphasis in statistics was placed on statistical hypothesis testing (confirmatory data analysis); more emphasis needed to be placed on using data to suggest hypotheses to test.

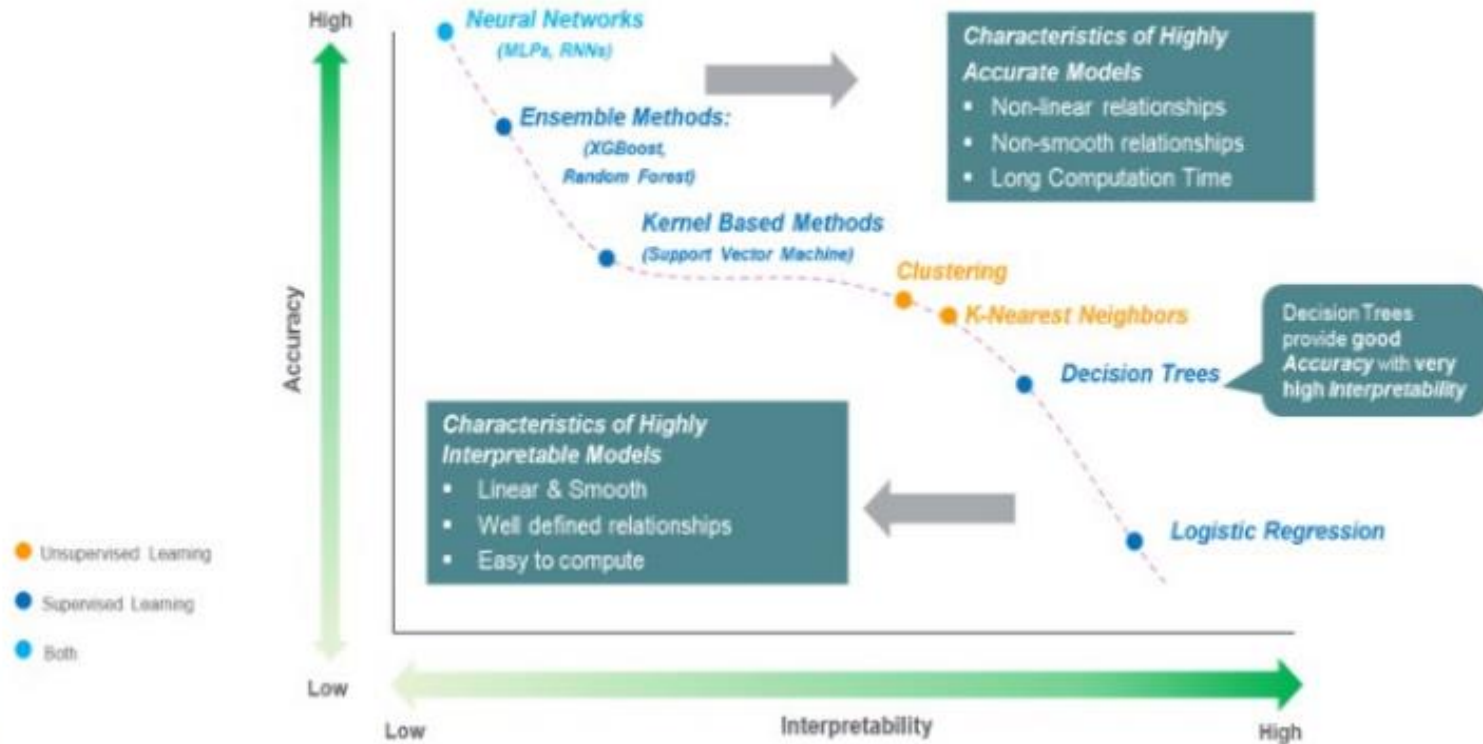
Exploratory Data Analysis

- **Summary Statistics**
- **Getting Sense of Data Distribution**
- **Putting it all together: Outlier Detection**

Exploratory Data Analysis (EDA)

In this lecture, we are going to discuss the importance of data understanding.

In my opinion, EDA is the most crucial part in analyzing data, before we get into the content, Let's ask ourselves a question, explanatory or precision?





Exploratory Data Analysis (EDA) is a critical step in the data analysis process, primarily in the field of statistics and data science. It involves the initial examination and exploration of a dataset to summarize its main characteristics, gain insights, and detect patterns, anomalies, or relationships within the data. EDA helps analysts and data scientists understand their data better before performing more advanced statistical analyses or building predictive models.



Key components and techniques of Exploratory Data Analysis include:

1. **Data Summarization:** This involves generating summary statistics, such as mean, median, variance, standard deviation, and quantiles, to describe the central tendency, spread, and distribution of the data.
2. **Data Visualization:** EDA often relies on visualizations like histograms, box plots, scatter plots, and bar charts to visually represent the data's distribution and relationships. Visualization can reveal trends, outliers, clusters, and other important patterns in the data.
3. **Handling Missing Data:** Identifying and addressing missing values is crucial during EDA. Understanding how much data is missing and determining appropriate strategies for handling missing values can impact the quality of subsequent analyses.
4. **Outlier Detection:** EDA helps in identifying outliers or extreme values that can significantly impact statistical analyses or machine learning models. Outliers may indicate errors in data collection or highlight unique phenomena.
5. **Data Transformation:** Transformations such as scaling, normalization, or encoding categorical variables may be necessary to prepare the data for further analysis. EDA helps determine which transformations are suitable.

6. **Correlation Analysis:** EDA explores relationships between variables, often using correlation coefficients or scatter plots. This can help identify potential predictors or dependencies.
7. **Pattern Recognition:** EDA can reveal underlying patterns, trends, or clusters within the data, which can inform subsequent analyses and modeling approaches.
8. **Hypothesis Generation:** EDA can inspire hypotheses about the data, which can be tested in later stages of analysis. It can also guide the selection of appropriate statistical tests or machine learning algorithms.
9. **Data Cleaning:** EDA often uncovers data quality issues, such as duplicate records or inconsistent values, which may need to be addressed before further analysis.

EDA is an iterative process, and its results can influence data preprocessing, feature engineering, and the choice of modeling techniques. By thoroughly exploring the data, analysts can make informed decisions about how to proceed with their analyses, improving the chances of deriving meaningful insights or building accurate predictive models.



Exploratory Data Analysis (EDA) is a flexible and iterative process, and the specific steps can vary depending on the dataset and analysis goals. However, a typical EDA process includes the following key steps:



1. **Collect the Data:**

- Obtain the dataset you intend to analyze, ensuring it's complete and properly formatted.

2. **Load and Inspect the Data:**

- Load the data into your analysis environment (e.g., Python, R, or a spreadsheet tool).
- Begin by inspecting the first few rows to get a sense of the data's structure.
- Check the data types of each column (numeric, categorical, date, etc.).
- Verify if there are any missing values and how they are represented.

3. **Summary Statistics:**

- Calculate summary statistics to describe the central tendency and dispersion of numeric variables. Common statistics include mean, median, standard deviation, and quartiles.
- For categorical variables, compute frequency counts and percentages for each category.

4. **Data Visualization:**

- Create a variety of visualizations to explore the data's distribution and relationships.

Common visualizations include:

- Histograms and density plots for numeric variables.
- Box plots to visualize the spread and central tendency.
- Bar charts for categorical variables.
- Scatter plots to examine relationships between variables.
- Heatmaps to visualize correlations between variables.

5. **Handling Missing Data:**

- Develop strategies to handle missing data, such as imputation or removal of rows/columns with missing values, based on the nature and extent of missingness.

6. **Outlier Detection:**

- Identify outliers in the data using visualization and statistical methods like the IQR (Interquartile Range) method or z-scores. Decide whether to remove, transform, or keep outliers based on domain knowledge and analysis goals.

7. **Feature Engineering:**

- Create new features or transform existing ones if necessary. For example, you may convert timestamps into meaningful time-related features or engineer interaction terms between variables.

8. **Correlation Analysis:**

- Calculate correlation coefficients (e.g., Pearson, Spearman) to quantify relationships between numeric variables.
- Visualize correlations using scatter plots or heatmaps to identify strong positive or negative associations.

9. **Pattern Recognition:**

- Look for patterns, trends, or clusters in the data. This might involve identifying seasonality, cyclical patterns, or groupings of data points.

10. **Hypothesis Generation:**

- Formulate hypotheses or research questions based on the insights gained from the data. These hypotheses can guide further analysis and modeling.

11. **Communication:**

- Prepare clear and concise summaries of your findings, often using visualizations, and share them with stakeholders or team members. Explain any interesting or important patterns discovered during EDA.

12. **Iterate:**

- EDA is an iterative process. You may need to revisit and refine earlier steps as new insights emerge or as you progress to more advanced analyses or modeling.

13. **Document:**

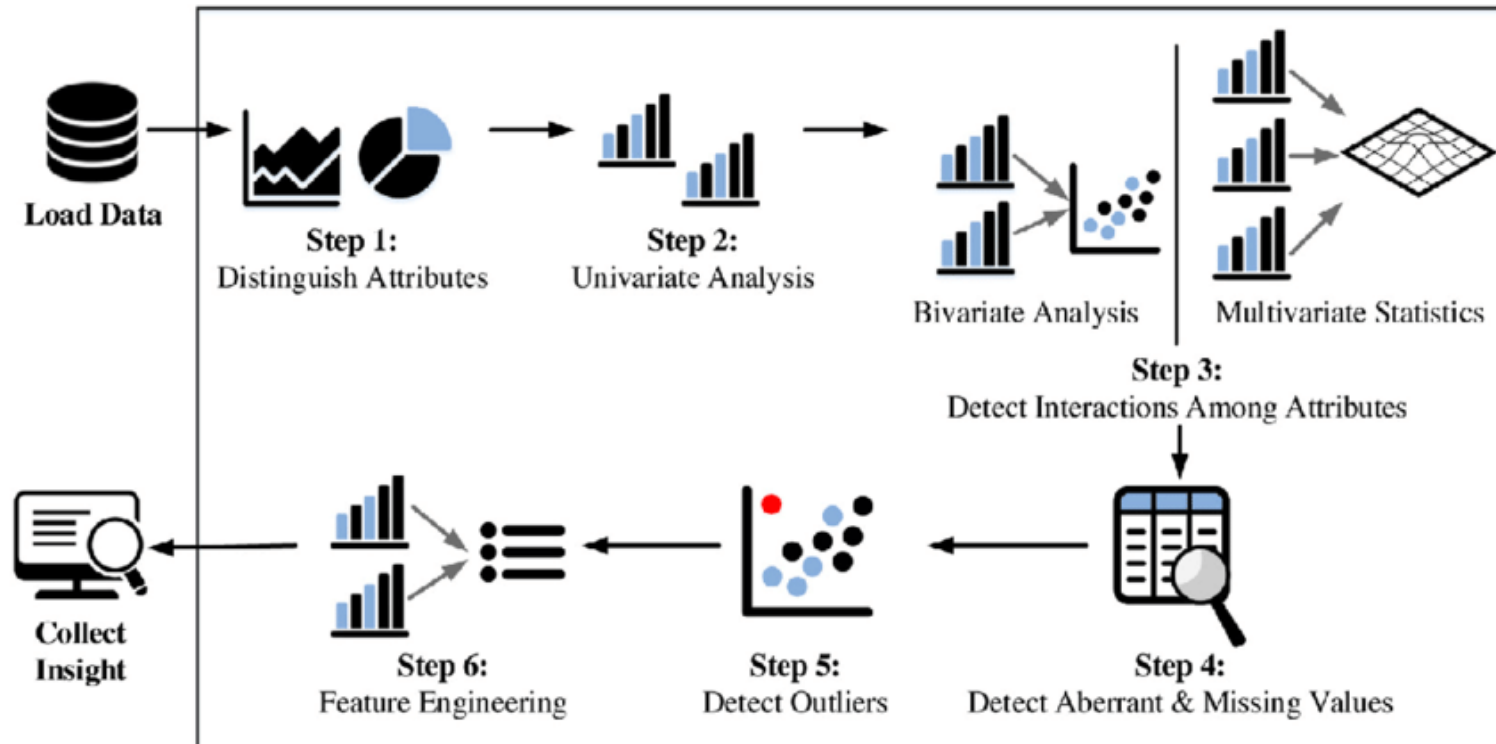
- Keep thorough documentation of the steps you've taken and the decisions made during EDA. This documentation is valuable for reproducibility and for providing context to others who may work with the data.

Remember that EDA is not a one-size-fits-all process, and its depth and complexity can vary depending on the dataset's nature and your analysis goals. The goal of EDA is to gain a deep understanding of the data, identify potential issues, and generate insights that inform subsequent data analysis and modeling tasks.

EDA

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, spot anomalies, test hypotheses and check assumptions with the help of summary statistics and graphical representations.

It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.



Summary Statistics – Data size

The dim()function outputs the number of rows and columns.

```
> data=read.csv("bike_buyers.csv")  
> dim(data)  
[1] 1000 13
```

We can only use the nrow()and ncol()function to find only the number of rows and columns.

```
> nrow(data)  
[1] 1000  
> ncol(data)  
[1] 13  
> |
```

Summarizing the Data

- The `summary()` functions give a brief summary for each column.

```

1000
> summary(data)

   ID          Marital.Status      Gender      Income      Children      Education      Occupation
Min.   :11000   Length:1000      Length:1000      Min.    : 10000      Min.    :0.00      Length:1000      Length:1000
1st Qu.:15291   Class :character      Class :character      1st Qu.: 30000      1st Qu.:0.00      Class :character      Class :character
Median :19744   Mode  :character      Mode  :character      Median : 60000      Median :2.00      Mode  :character      Mode  :character
Mean   :19966                                     Mean   : 56268      Mean   :1.91
3rd Qu.:24471                                     3rd Qu.: 70000      3rd Qu.:3.00
Max.   :29447                                     Max.   :170000      Max.   :5.00
NA's   :6                                           NA's    :8

   Home.Owner      Cars      Commute.Distance      Region      Age      Purchased.Bike
Length:1000      Min.    :0.000      Length:1000      Length:1000      Min.    :25.00      Length:1000
Class :character      1st Qu.:1.000      Class :character      Class :character      1st Qu.:35.00      Class :character
Mode  :character      Median :1.000      Mode  :character      Mode  :character      Median :43.00      Mode  :character
                        Mean   :1.455                                     Mean   :44.18
                        3rd Qu.:2.000                                     3rd Qu.:52.00
                        Max.   :4.000                                     Max.   :89.00
                        NA's    :9                                           NA's    :8

```

Summary Statistics – Summarizing the Data

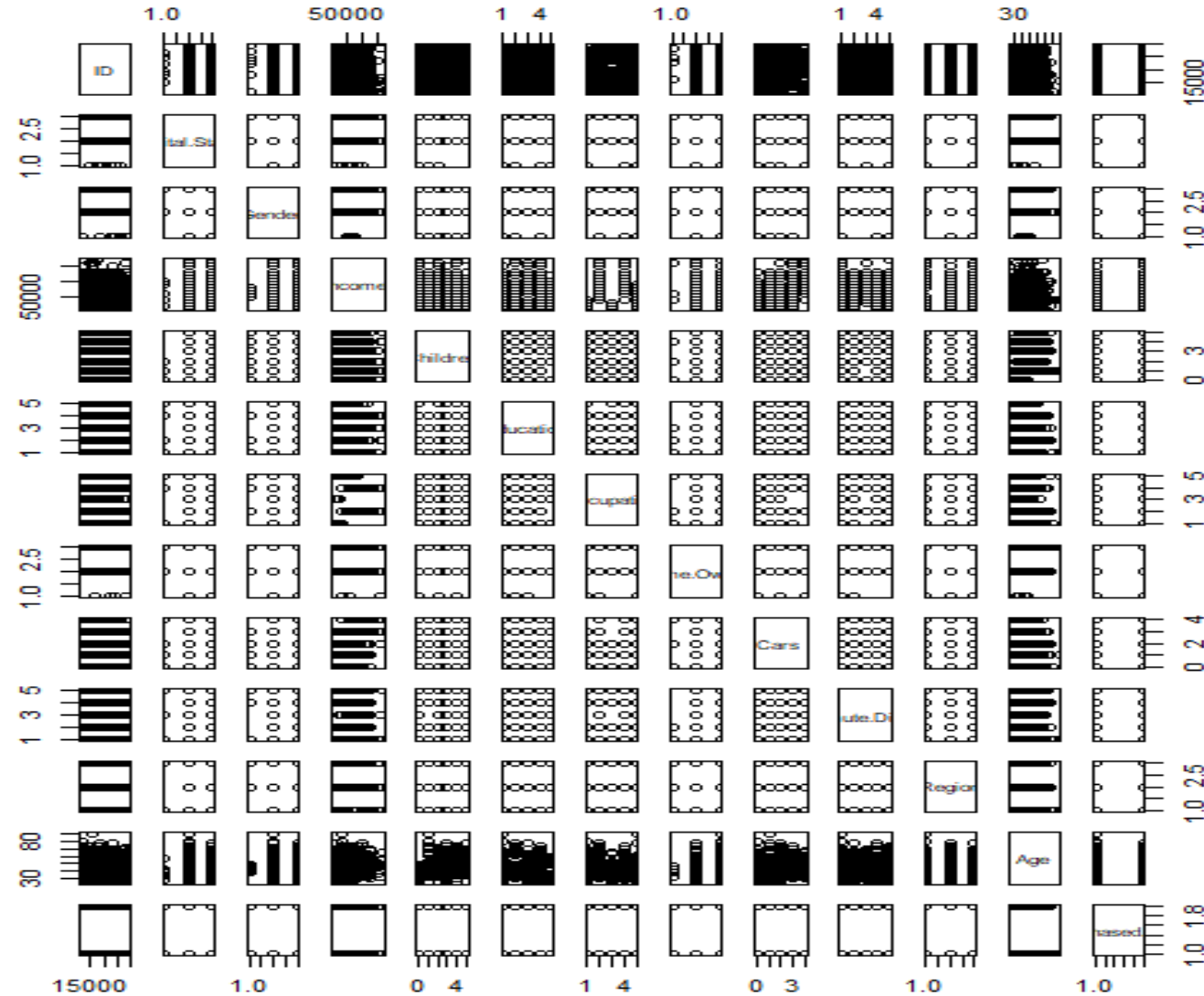
For numeric variables, the summary contains:

- Min.— the smallest value of the variable.
- 1st Qu. (Q1)— first quartile or 25th percentile
- Median— second quartile or 50th percentile
- Mean— Average value of the variable.
- 3rd Qu. (Q3)— third quartile or 75th percentile
- Max.— the largest value of the variable.

These statistics are useful to get a sense of the data distribution for a variable:
its range and centrality

Plot the Summary of the Data

plot(data)




```

{r}
library(psych)
describe(data) #describe() can compute the statistics of all numerical variables in data:

```

```
> describe(data)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
ID	1	1000	19965.99	5347.33	19744.0	19925.80	6848.13	11000	29447	18447	0.05	-1.19	169.10
Marital.Status*	2	1000	2.45	0.51	2.0	2.45	0.00	1	3	2	0.04	-1.62	0.02
Gender*	3	1000	2.49	0.52	2.5	2.50	0.74	1	3	2	-0.19	-1.43	0.02
Income	4	994	56267.61	31067.82	60000.0	53680.90	29652.00	10000	170000	160000	0.75	0.50	985.41
Children	5	992	1.91	1.63	2.0	1.79	1.48	0	5	5	0.39	-1.02	0.05
Education*	6	1000	2.63	1.35	3.0	2.57	1.48	1	5	4	0.13	-1.36	0.04
Occupation*	7	1000	3.26	1.45	4.0	3.32	1.48	1	5	4	-0.30	-1.33	0.05
Home.Owner*	8	1000	2.68	0.48	3.0	2.73	0.00	1	3	2	-0.87	-0.95	0.02
Cars	9	991	1.46	1.12	1.0	1.37	1.48	0	4	4	0.42	-0.41	0.04
Commute.Distance*	10	1000	2.64	1.56	2.0	2.56	1.48	1	5	4	0.33	-1.46	0.05
Region*	11	1000	1.89	0.69	2.0	1.86	0.00	1	3	2	0.15	-0.93	0.02
Age	12	992	44.18	11.36	43.0	43.48	11.86	25	89	64	0.52	-0.27	0.36
Purchased.Bike*	13	1000	1.48	0.50	1.0	1.48	0.00	1	2	1	0.08	-2.00	0.02

```
>
```

Summary of The Describe () Function

3. Explore the Output:

The ``describe()`` function generates various statistics and information about your data frame. Some of the information included in the output typically consists of:

- **Means and Medians:** It provides the mean, median (50th percentile), and other percentiles for numeric variables.
- **Standard Deviations:** It includes the standard deviation and interquartile range for numeric variables.
- **Frequency Counts:** For categorical variables, it shows the number of unique values and the most frequent value.
- **Missing Values:** It indicates the number of missing values for each variable.
- **Skewness and Kurtosis:** It computes skewness and kurtosis statistics for numeric variables, which describe the shape of the distribution.
- **Correlation Matrix:** If requested, it can also provide a correlation matrix for numeric variables.

4. Customize the Output:

You can customize the output by passing additional arguments to the ``describe()`` function. For example, you can specify whether to include correlations, set the number of digits for rounding, and more. Refer to the package documentation for details on customization options.

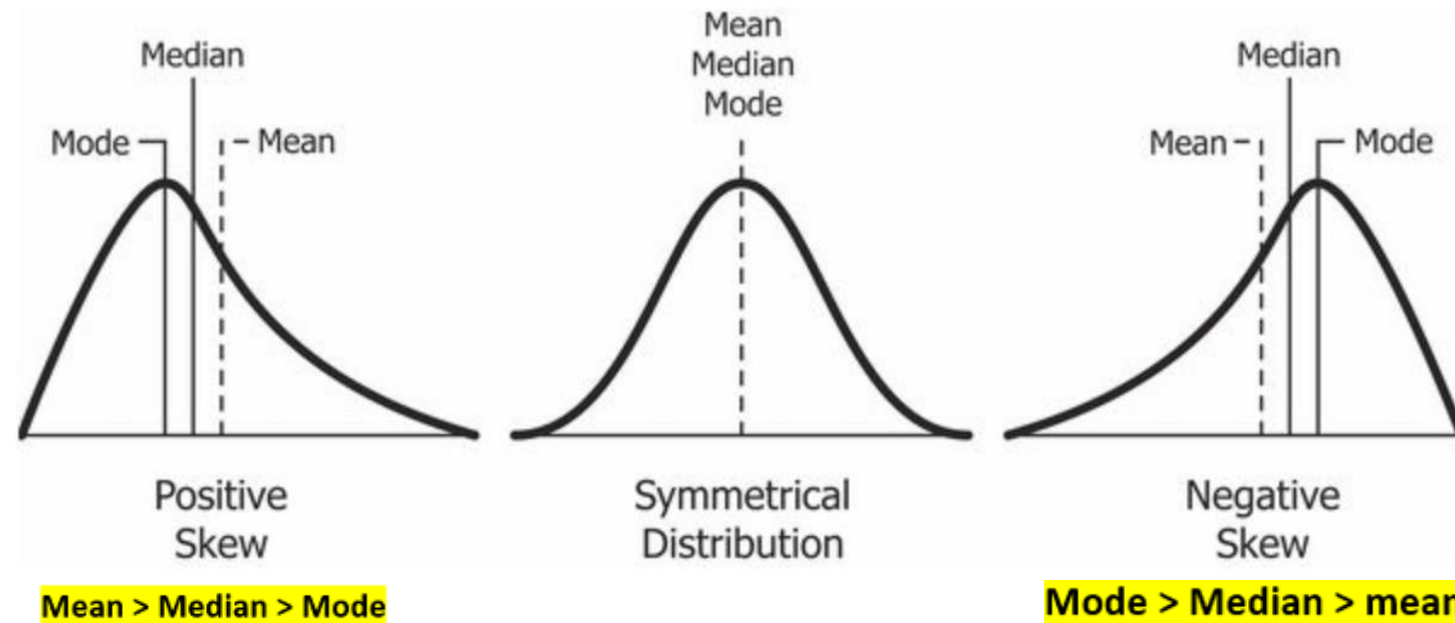
Remember that the ``psych`` package is just one option for data exploration and description in R. Depending on your specific needs, you may also consider using other packages like ``summarytools``, ``Hmisc``, or ``dplyr`` for similar tasks.

Skewness

- In probability theory and statistics, **skewness** is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive, zero, negative, or undefined.

There are two main types of skewness:

- 1. Positive Skew (Right Skew):** In a positively skewed distribution, the tail extends to the right, indicating that there are a few unusually large values or outliers on the right side of the distribution. The bulk of the data is concentrated on the left side.
- 2. Negative Skew (Left Skew):** In a negatively skewed distribution, the tail extends to the left, indicating that there are a few unusually small values or outliers on the left side of the distribution. The bulk of the data is concentrated on the right side.



Skewness

- If the skewness is between -0.5 & and 0.5, the data are nearly symmetrical.
- If the skewness is between -1 & and -0.5 (negative skewed) or between 0.5 & and 1 (positive skewed), the data are slightly skewed.
- If the skewness is lower than -1 (negative skewed) or greater than 1 (positive skewed), the data are extremely skewed.

Deal with Positively and Negatively skewed data as follows.

Positively skewed data:

Common transformations of this data include **square root, cube root, and log.**

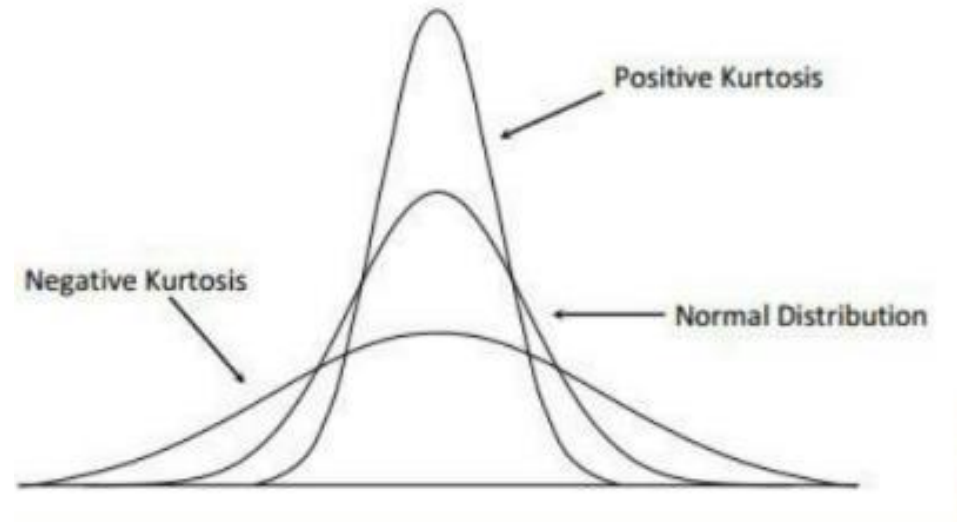
Negatively skewed data:

Common transformations include **square, cube root, and logarithmic.**

Measuring Data Symmetry Using Kurtosis

- Kurtosis is a measure of whether the data are heavy-tailed or light-tailed relative to a normal distribution.

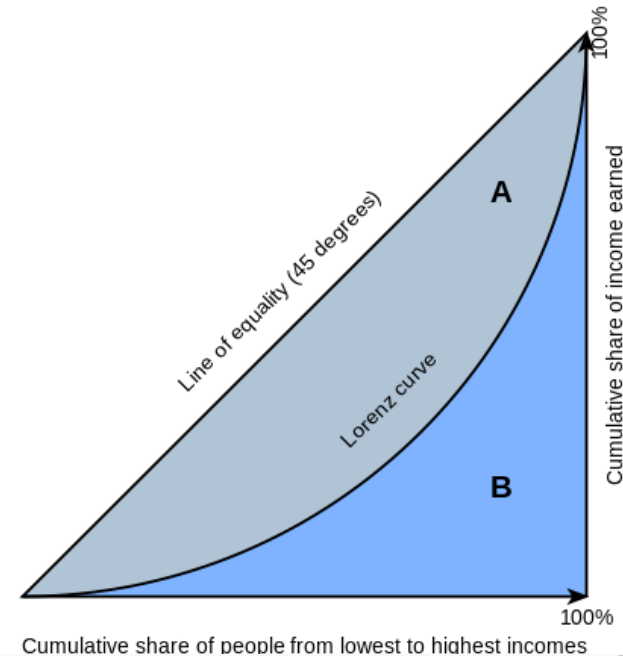
There are three types of kurtosis: mesokurtic (**Normal Kurtosis**), leptokurtic (**Negative Kurtosis**), and platykurtic (**Positive Kurtosis**).



Gini coefficient

- The Gini coefficient measures the inequality among values of a frequency. A Gini coefficient of zero expresses perfect equality, where all values are the same. A Gini coefficient of 1 expresses maximal inequality among values.

$$Gini = \frac{\sum_{i=1}^n \sum_{j=1}^n |x_i - x_j|}{2n \sum_{i=1}^n x_i}$$



The Gini coefficient is equal to the area marked A

divided by the total area of A and B, i.e. $Gini = \frac{A}{A+B}$.

The axes run from 0 to 1, so A and B form a triangle of area $\frac{1}{2}$ and $Gini = 2A = 1 - 2B$.




The Gini coefficient, also known as the Gini index or Gini ratio, is a statistical measure often used in economics and income distribution analysis to quantify inequality within a dataset. While it is not typically a primary tool in standard Exploratory Data Analysis (EDA), it can provide valuable insights into the distribution of a variable and its role in understanding disparities within the data. Here's how the Gini coefficient can be used in EDA:



1. **Income or Wealth Distribution Analysis:** In EDA related to income, wealth, or other financial data, the Gini coefficient can help assess the degree of inequality in the distribution of income or wealth among individuals or groups. A high Gini coefficient indicates greater income or wealth inequality, while a low coefficient suggests more equitable distribution.
2. **Consumer Behavior and Market Segmentation:** In EDA for market research and consumer behavior studies, the Gini coefficient can be used to examine the concentration of purchases or consumption among different groups or products. It can help identify key segments that contribute disproportionately to sales or consumption.
3. **Resource Allocation:** When dealing with resources allocation, such as budget allocation in organizations, understanding the distribution of resources among departments or projects is crucial. The Gini coefficient can highlight areas of resource concentration or disparities.
4. **Customer Segmentation and Retention:** In EDA for customer data, the Gini coefficient can reveal the concentration of spending or loyalty among different customer segments. This information can guide marketing and customer retention strategies.
5. **Healthcare and Disease Distribution:** In healthcare EDA, the Gini coefficient can be applied to understand the distribution of diseases or health outcomes among populations. It can help identify areas or groups with disproportionate health burdens.
6. **Educational Attainment:** When analyzing educational data, the Gini coefficient can be used to assess disparities in educational attainment among different demographic groups, such as gender, ethnicity, or socioeconomic status.

7. **Social Sciences:** In social science research, the Gini coefficient can be employed to study various aspects of inequality, such as access to services, quality of life, or political power distribution.
8. **Insurance and Risk Assessment:** In the insurance industry, the Gini coefficient can help assess the distribution of risks or insurance claims among policyholders. It aids in understanding risk concentration.
9. **Credit Scoring:** In financial EDA, the Gini coefficient can be used in credit scoring models to evaluate the discriminatory power of variables in predicting creditworthiness.

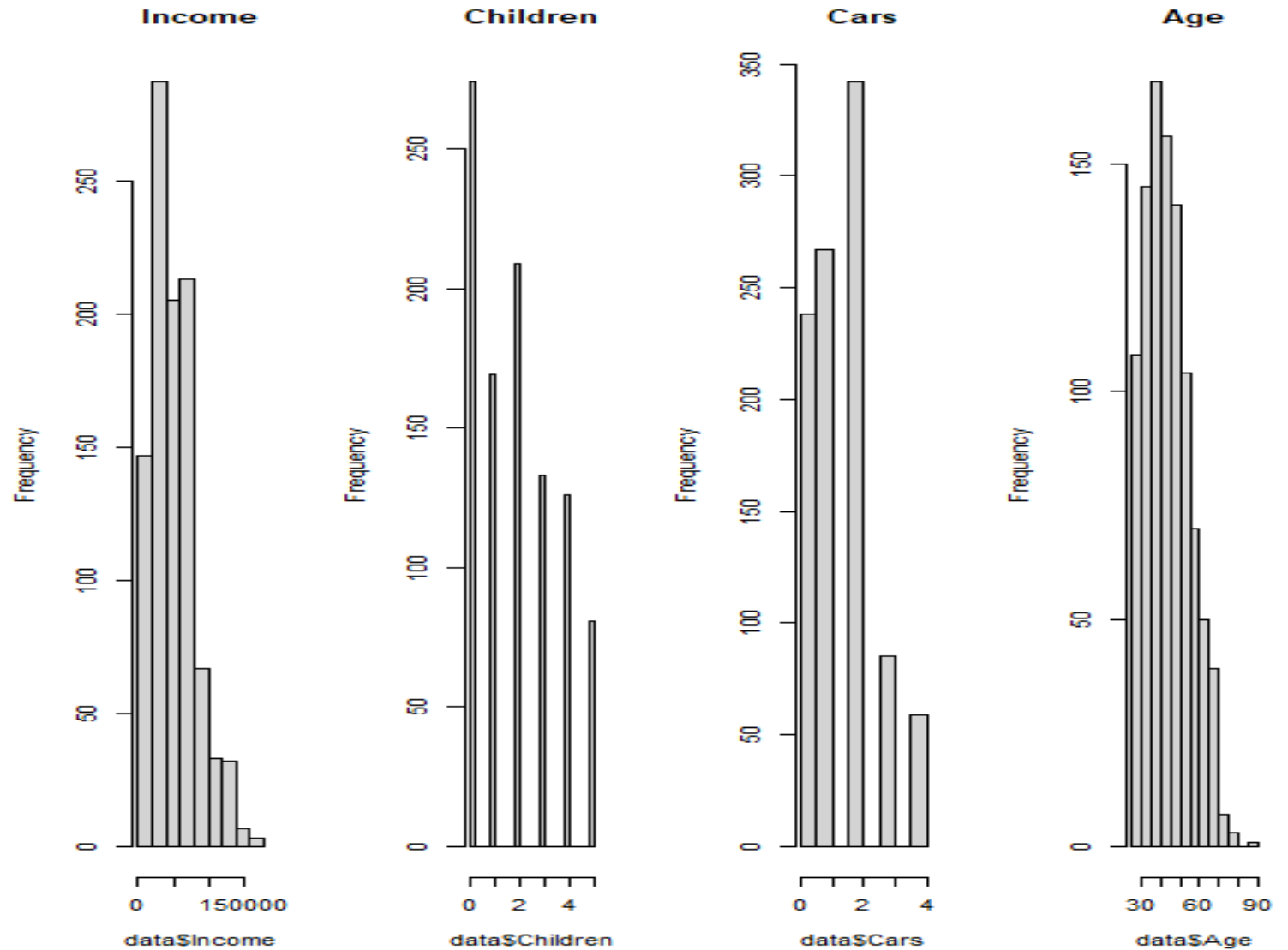
While the Gini coefficient is valuable for analyzing inequality, it is important to use it in conjunction with other EDA techniques to gain a comprehensive understanding of the dataset. Additionally, the interpretation of the Gini coefficient depends on the context and the specific application, so it should be used thoughtfully and alongside other relevant measures and visualizations.

 Regene

Histogram With Missing Values

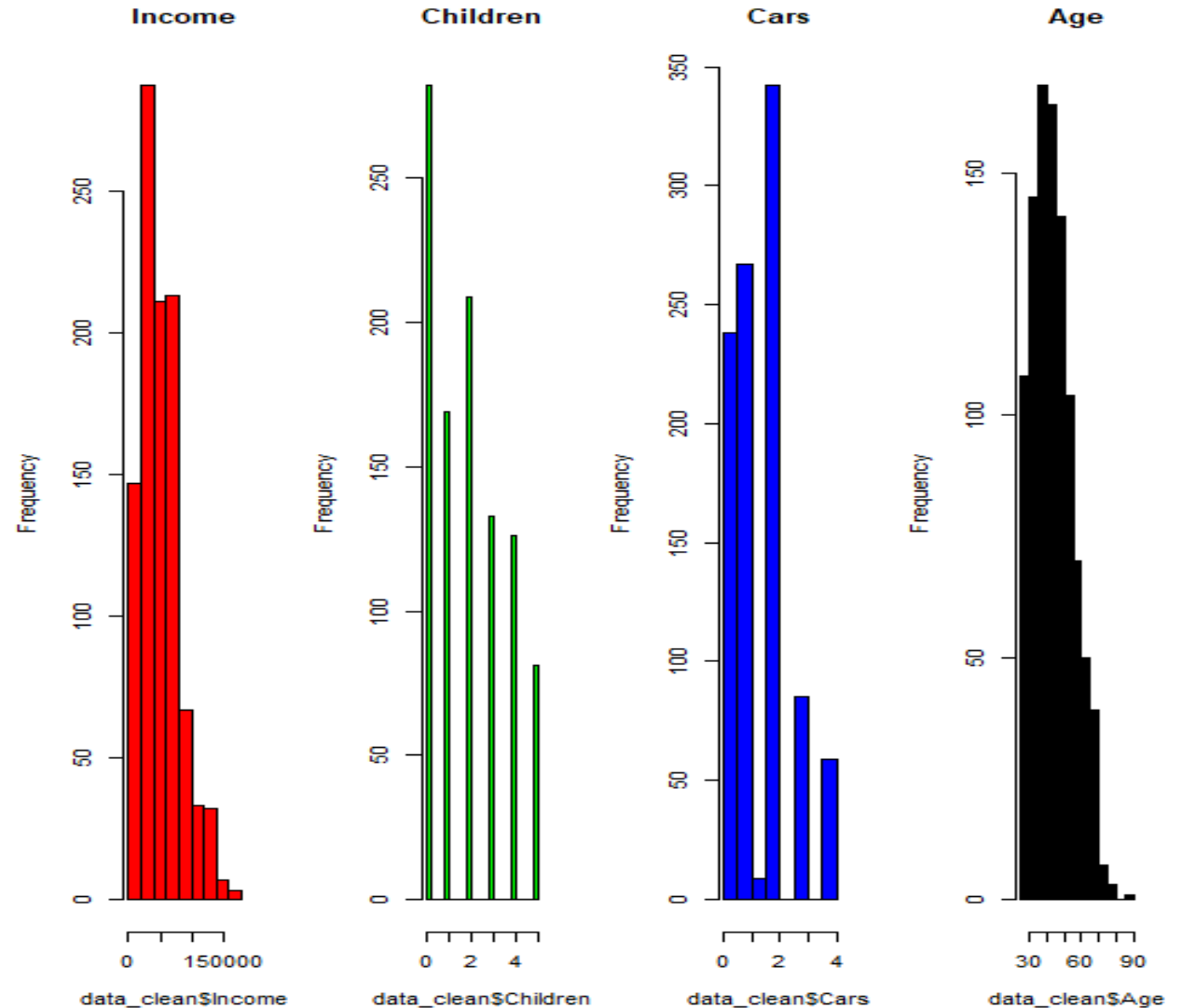
```
```{r}
par(mfrow = c(1, 4))

hist(data$Income,breaks = "sturges",main = "Income")|
hist(data$Children, breaks = 20,main = "Children")
hist(data$Cars,breaks = "scott",main = "Cars")
hist(data$Age,breaks = "fd",main = "Age")
```
```



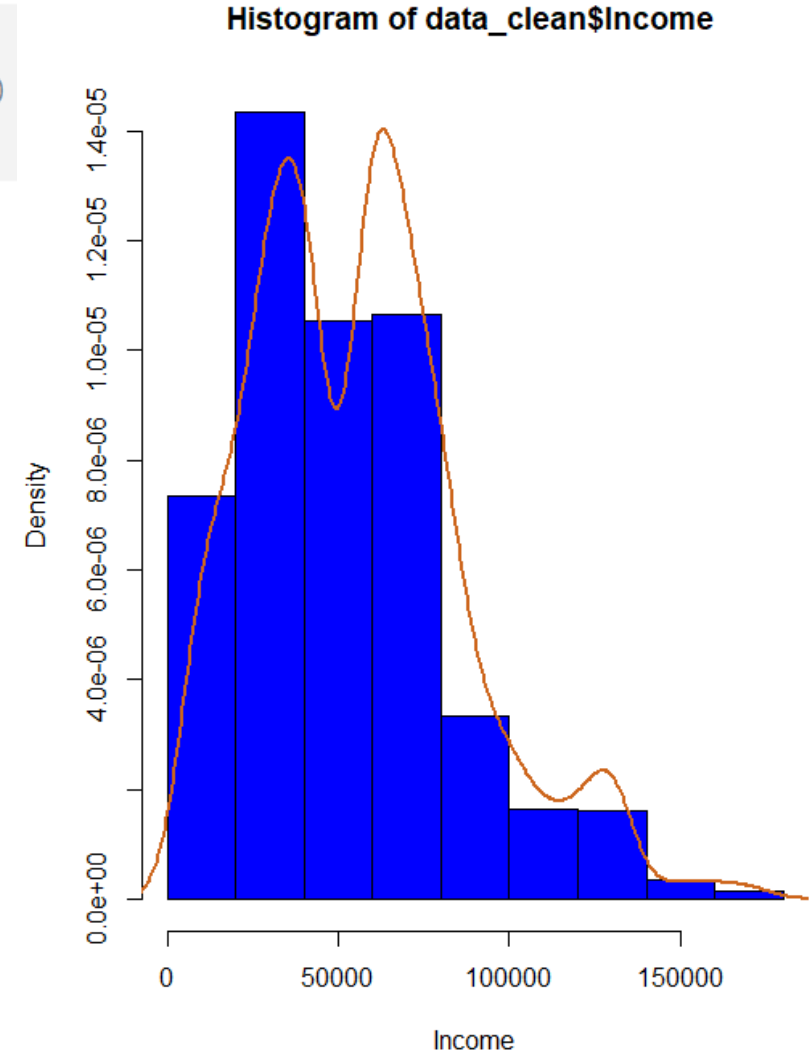
Histogram Without Missing Values

```
```{r}
par(mfrow = c(1, 4))
hist(data_clean$Income,breaks = "sturges",main = "Income", col="red")
hist(data_clean$Children, breaks = 20,main = "Children",col="green")
hist(data_clean$Cars,breaks = "scott",main = "Cars",col="blue")
hist(data_clean$Age,breaks = "fd",main = "Age",col="black")
```
```



Density and Histogram plots

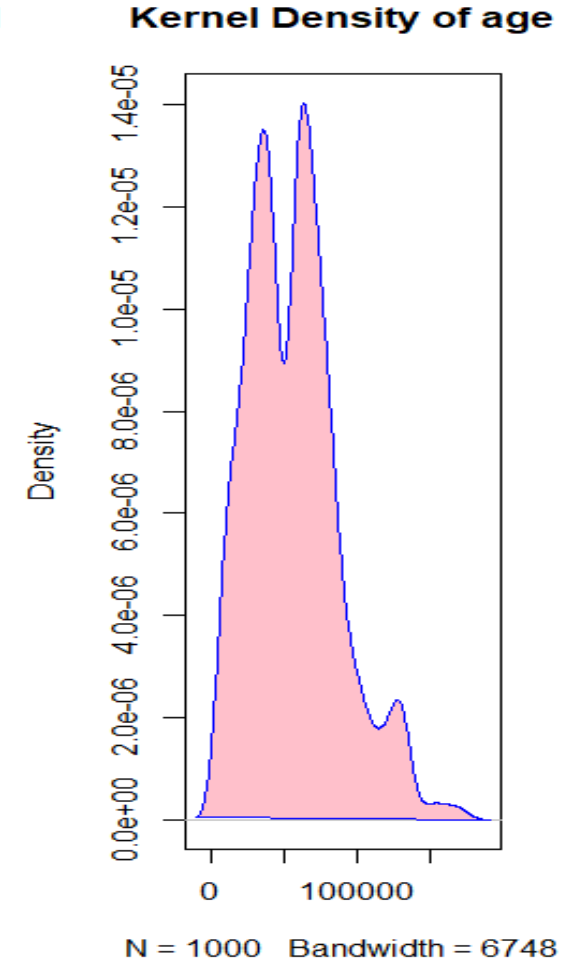
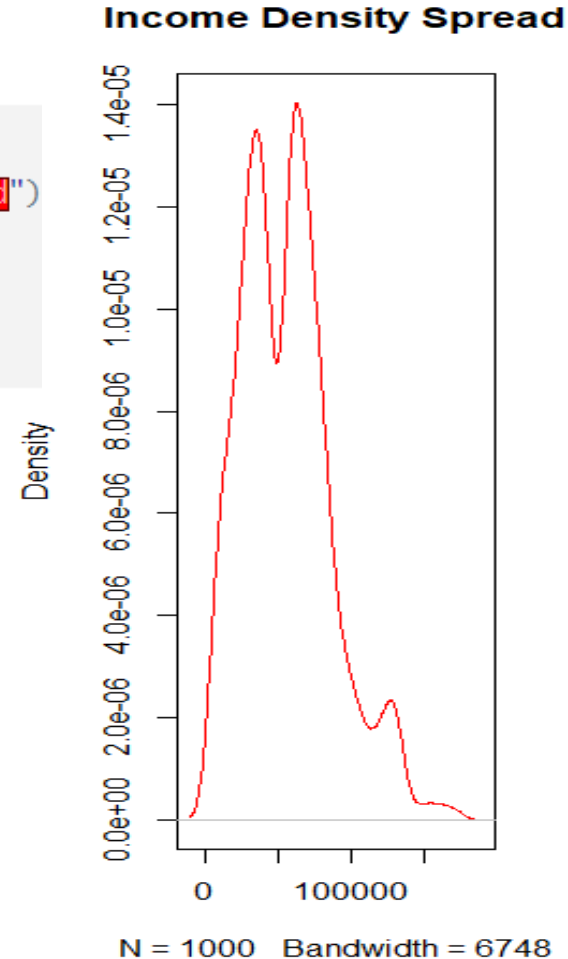
```
```{r}
par(mfrow = c(1, 1))
hist(data_clean$Income, col="blue",border="black",prob = TRUE,xlab = "Income",main = "Histogram of data_clean$Income ")
lines(density(data_clean$Income),lwd = 2,col = "chocolate3")
```
```



Density Plot

- A density plot is a representation of the distribution of a numeric variable. It uses a kernel density estimate to show the probability density function of the variable (see more). It is a smoothed version of the histogram and is used in the same concept.

```
```{r}
par(mfrow = c(1, 2))
plot(density(data_clean$Income), main='Income Density Spread', col="red")
#Kernel density for age
d <- density(data_clean$Income)
plot(d, main="Kernel Density of age")
polygon(d, border="blue", col="pink")
```
```



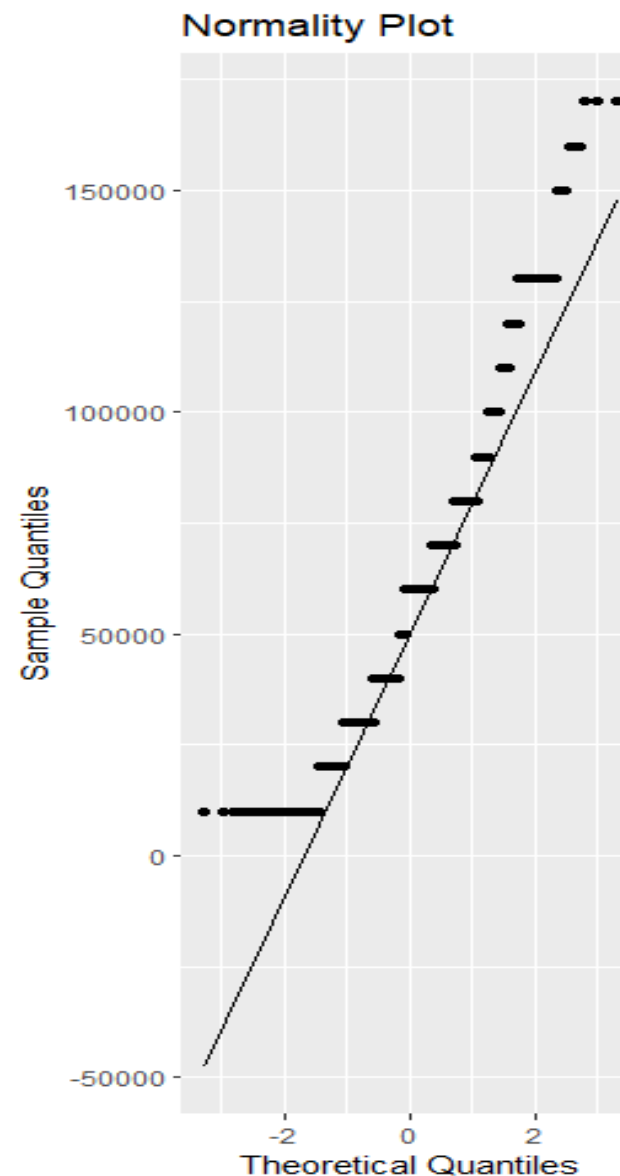
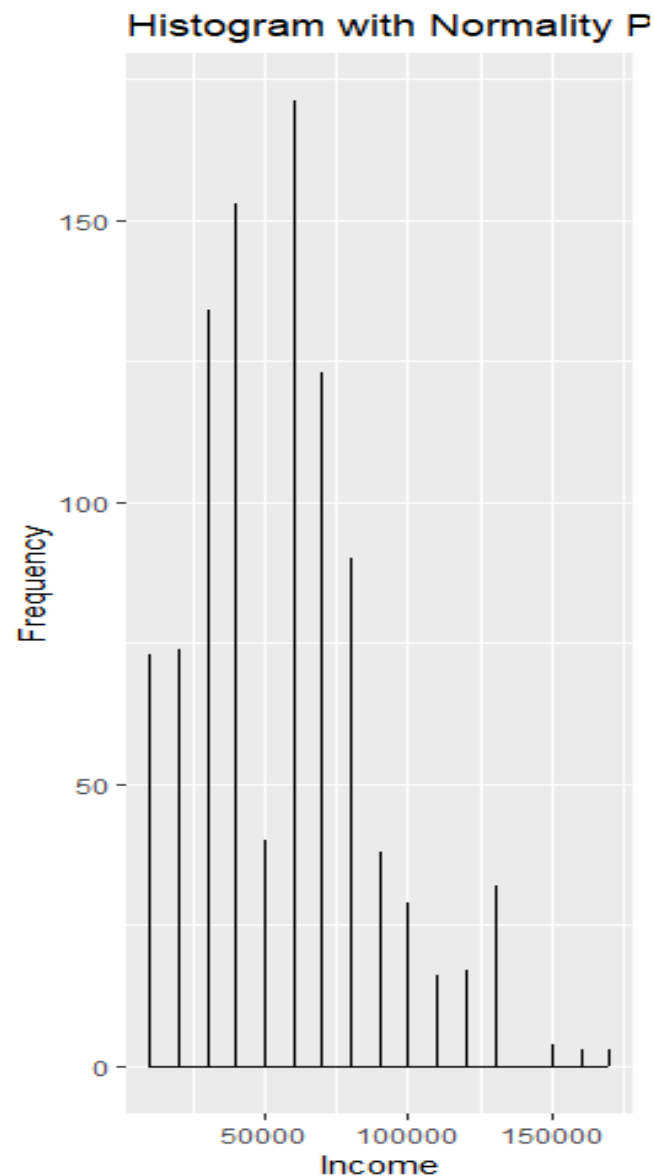
```

```{r}
Create a histogram
hist_plot <- ggplot(data_clean, aes(x = Income)) +
 geom_histogram(binwidth = 0.5, fill = "blue", color = "black") +
 labs(x = "Income", y = "Frequency", title = "Histogram with Normality Plot")

Create a normality plot (Q-Q plot)
qq_plot <- ggplot(data_clean, aes(sample = Income)) +
 stat_qq() +
 stat_qq_line() +
 labs(x = "Theoretical Quantiles", y = "Sample Quantiles", title = "Normality Plot")

Arrange the two plots side by side
library(gridExtra)
grid.arrange(hist_plot, qq_plot, ncol = 2)
```

```

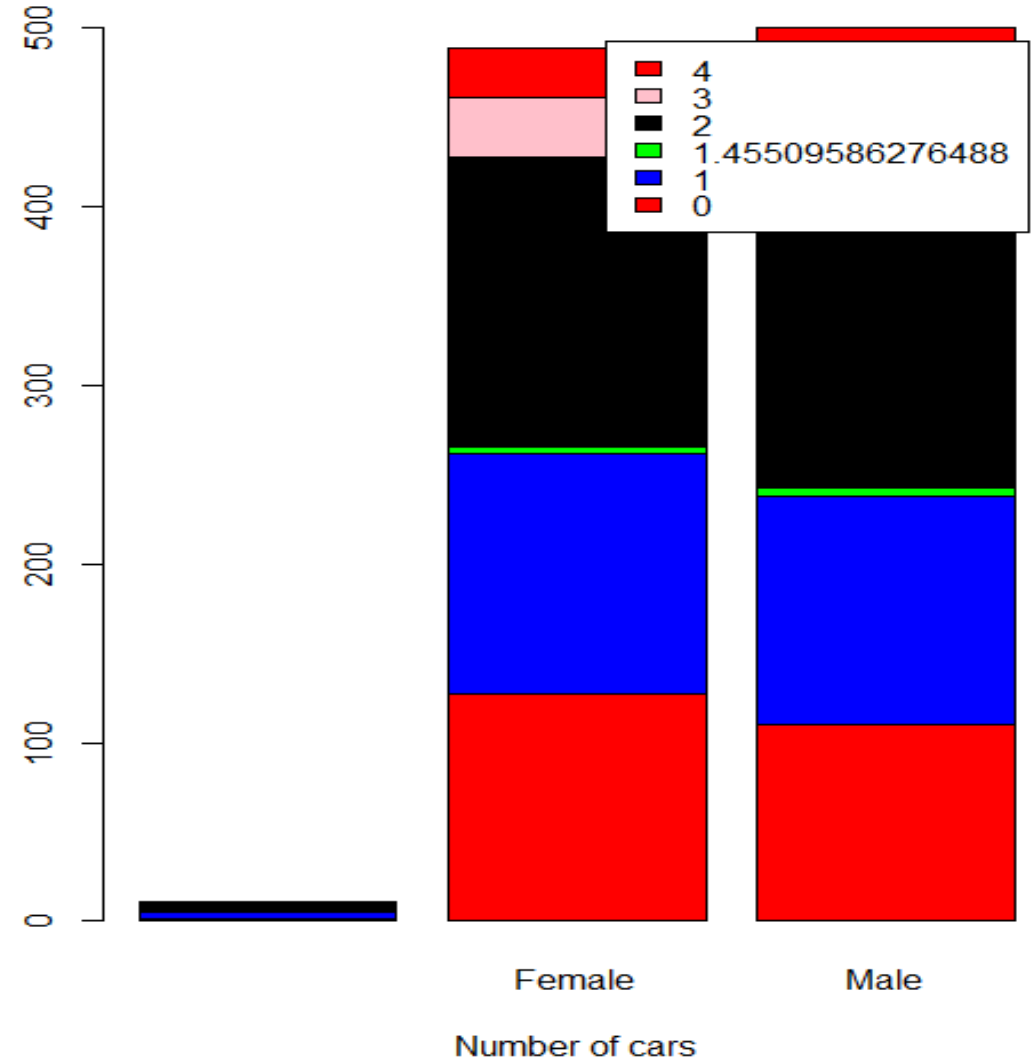


Bar Plot

```
**Bar plot**
```{r}
par(mfrow = c(1, 1))
Create a contingency table of counts
counts <- table(data_clean$Cars, data_clean$Gender)

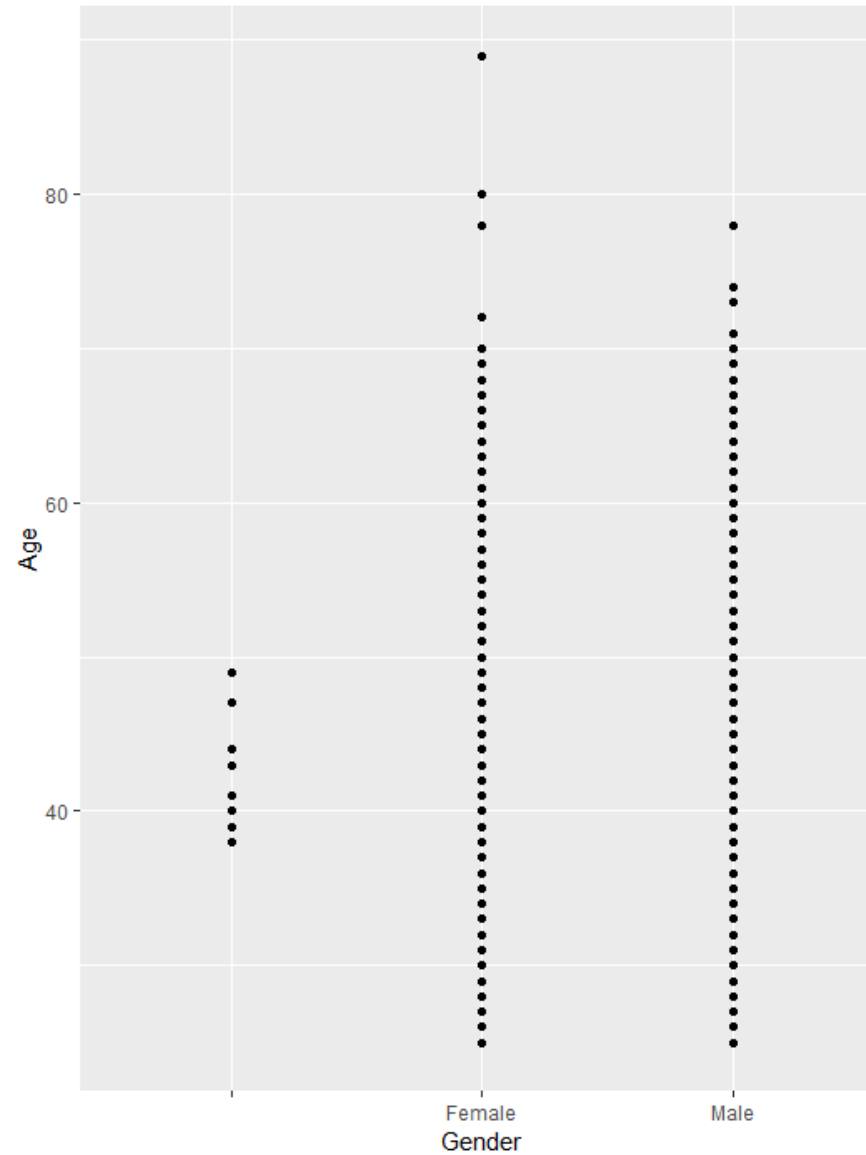
Define a vector of colors for the bars (you can specify your own colors)
bar_colors <- c("red", "blue", "green", "black", "pink")

Create the barplot with different colors
barplot(counts, main = '',
 xlab = "Number of cars",
 legend = rownames(counts),
 col = bar_colors)
```
```



Exploring ggplot library

```
{r}  
library(ggplot2)  
ggplot(data_clean,  
       aes(y = Age, x = Gender)) +  
  geom_point()
```



Scatter plot

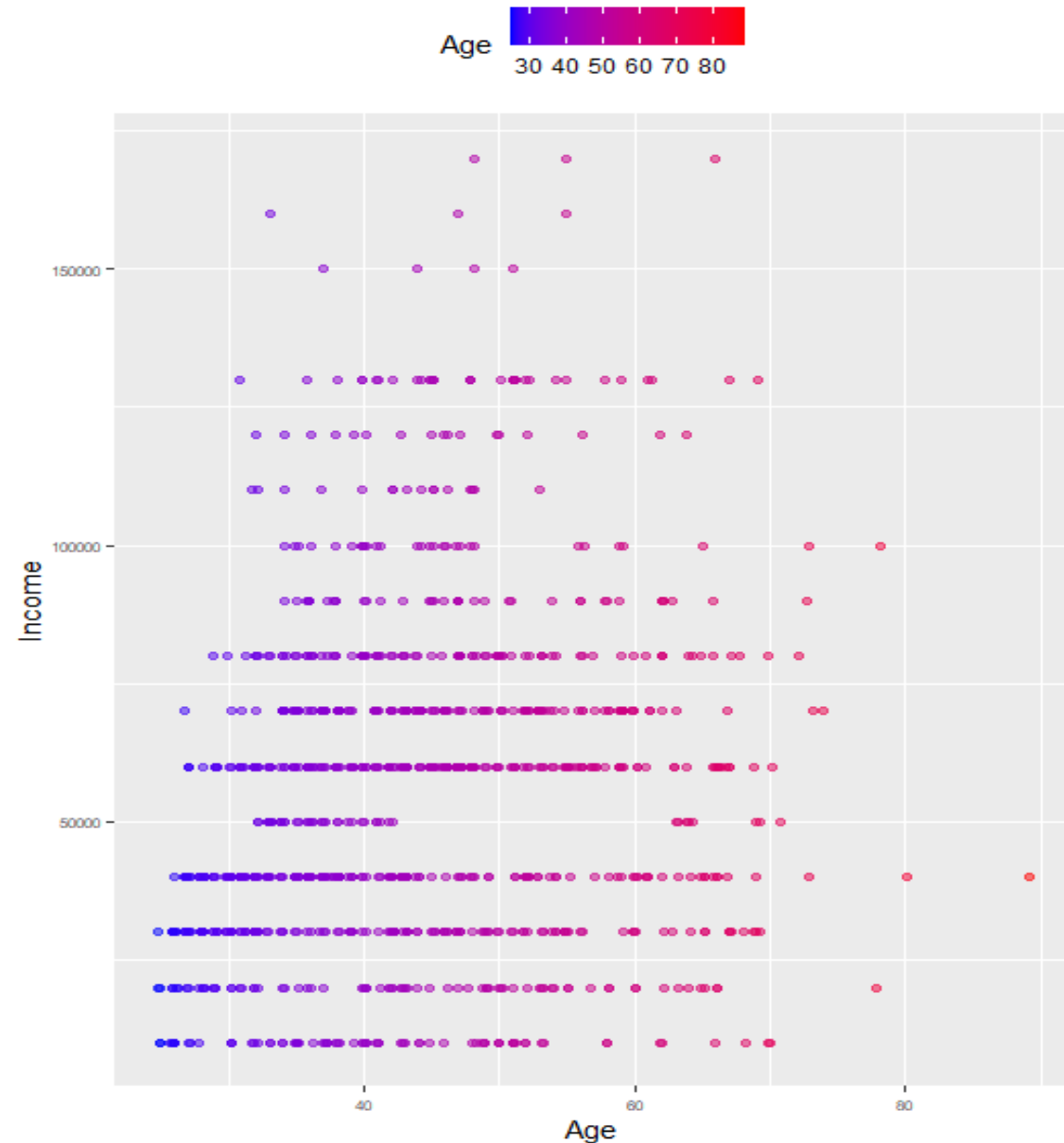
```
```{r}
Load the ggplot2 package if not already loaded
library(ggplot2)

Create the scatterplot
p3 <- ggplot(data_clean, aes(x = Age, y = Income, color = Age)) +
 geom_point(alpha = 0.5, size = 1.5, position = position_jitter(width = 0.25, height = 0)) +

Customize the appearance
labs(x = "Age", y = "Income") + # Add axis labels
scale_color_gradient(low = "blue", high = "red", name = "Age") + # Customize color scale

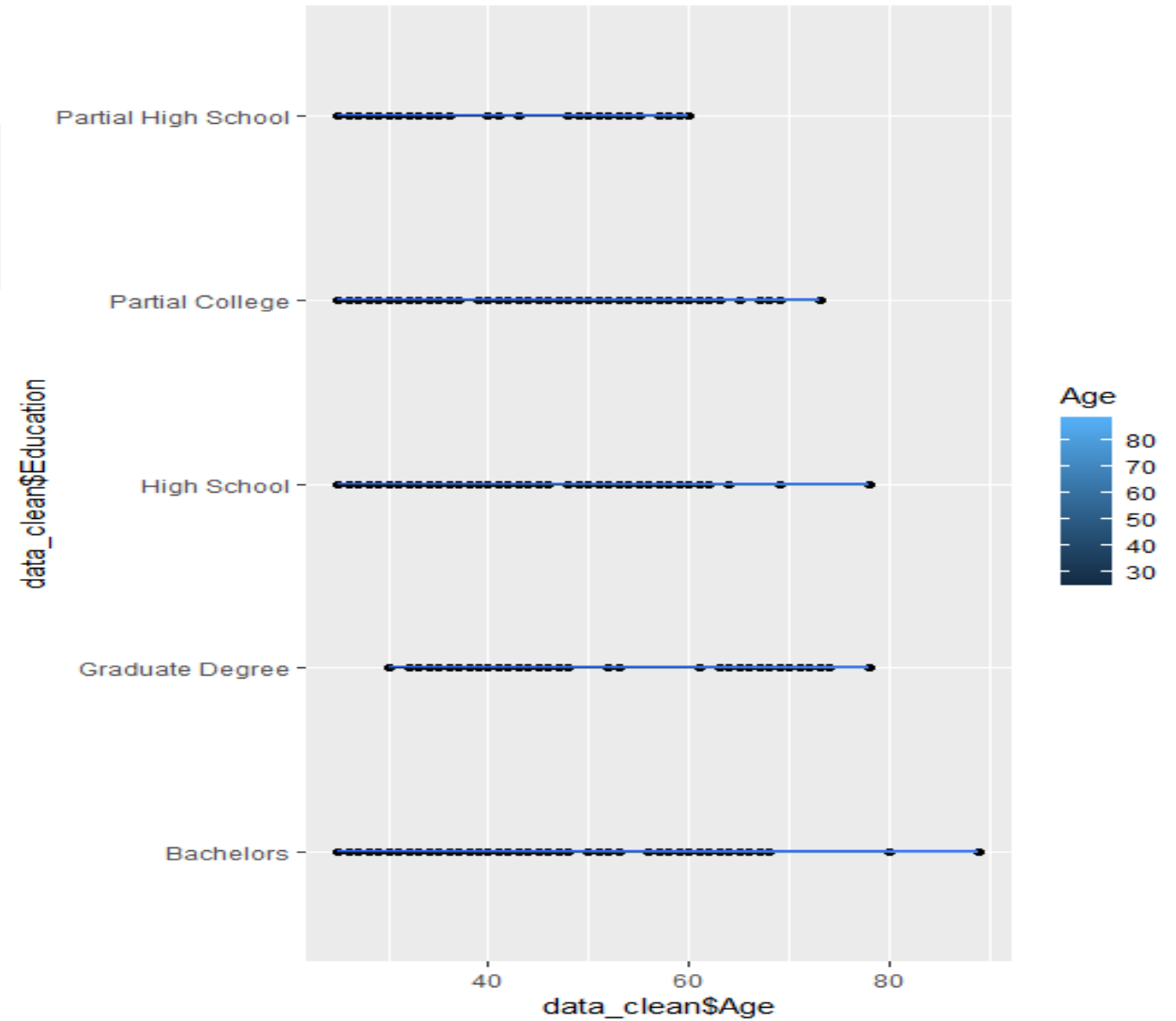
Set theme options
theme(legend.position = "top", axis.text = element_text(size = 6))

Print the plot
print(p3)
```
```



Trend Plot

```
**Trend Plot**  
```{r}  
ggplot(data_clean, aes(x=data_clean$Age, y=data_clean$Education)) +
 geom_point() +
 geom_smooth(method=lm, level=0.99) + geom_line(aes(color = Age))
```
```





A trend plot, also known as a time series plot or time plot, is a type of data visualization used to display data points over time. It is commonly used in various fields such as economics, finance, business, and environmental science to examine and analyze patterns, trends, and changes in data that are collected or observed over a series of time intervals. Trend plots are especially useful for identifying temporal trends, seasonal patterns, and anomalies in time-dependent data.



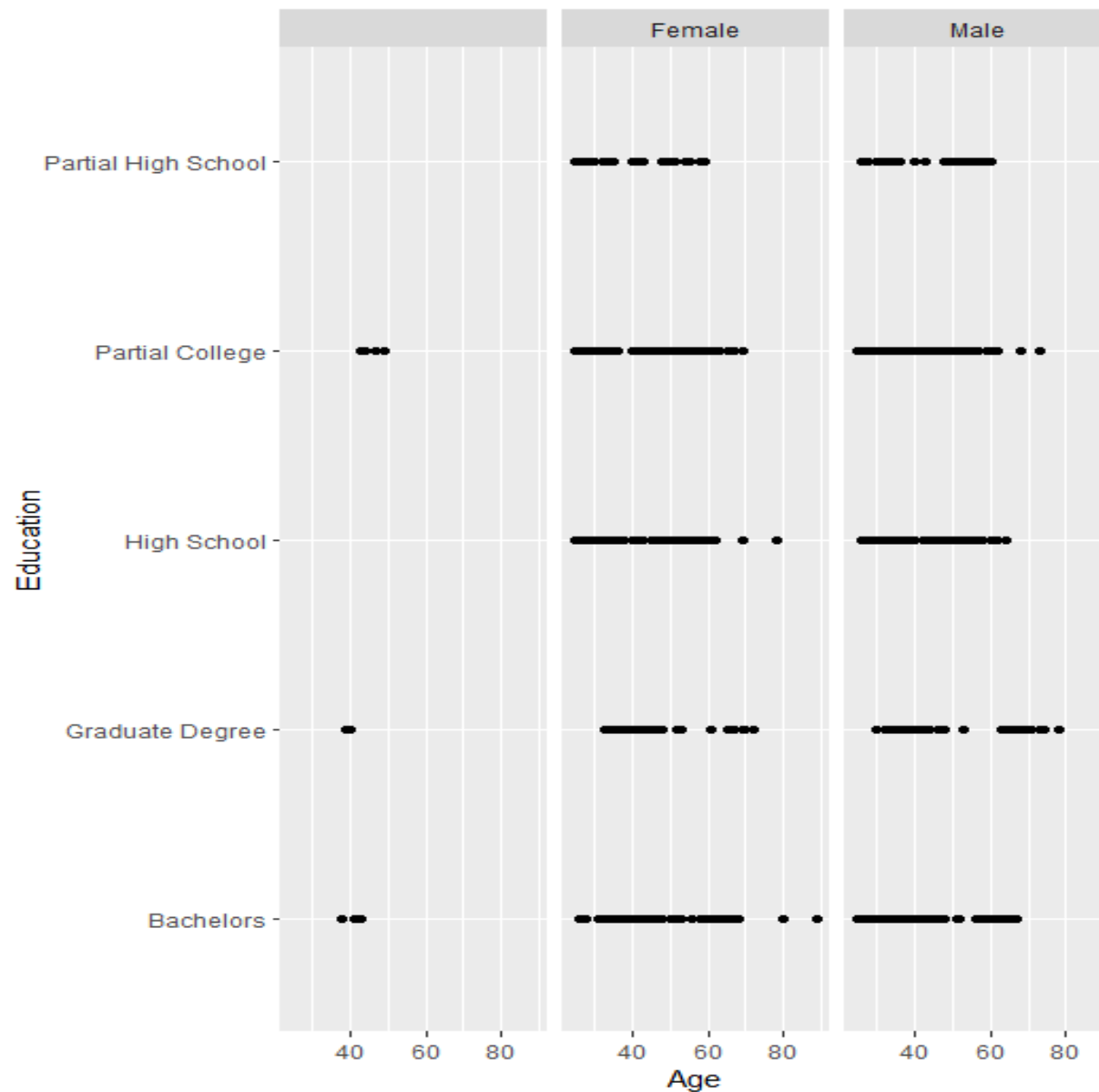
Key characteristics of trend plots include:

1. **Time on the x-axis:** The horizontal axis of a trend plot represents time, typically in chronological order. Time can be displayed in various units, such as years, months, days, or even hours, depending on the granularity of the data.
2. **Dependent variable on the y-axis:** The vertical axis of a trend plot represents the variable of interest that changes over time. This variable could be numeric, such as stock prices, temperature, or sales figures, or categorical, such as the status of a project (e.g., "in progress," "completed").
3. **Data points or lines:** Each data point on the plot corresponds to an observation or measurement taken at a specific time point. In some cases, data points may be connected with lines to visualize trends more clearly.
4. **Trend identification:** Trend plots help visualize trends in data, which can be upward (increasing), downward (decreasing), or flat (constant) trends. They can also reveal seasonal or cyclical patterns.
5. **Anomalies and outliers:** Trend plots can highlight unusual data points or outliers that do not follow the overall trend, allowing for further investigation.
6. **Data labeling:** Depending on the complexity of the data and the plot, data points or lines may be labeled with additional information, such as data values, labels, or annotations.

Faceting Plot

****Faceted Plot****

```
```{r}
Create a faceted plot
ggplot(data_clean, aes(x = Age, y = Education)) +
 geom_point() + # Add points to the plot
 labs(x = "Age", y = "Education") + # Add axis labels
 facet_wrap(~ Gender) # Facet by the 'Gender' variable
```
```





Faceting, also known as small multiples or trellis plots, is a data visualization technique used in statistical graphics to display multiple subsets or facets of a dataset as separate plots within the same visualization. Faceting allows you to break down your data into smaller, more manageable segments and create a grid or matrix of plots, each representing a different subset of the data. This technique is especially useful for exploring and comparing patterns and relationships within these subsets.



Key characteristics of faceting include:

1. **Subplots:** Faceting generates a grid of subplots, where each subplot corresponds to a unique subset of the data based on one or more categorical variables.
2. **Categorical Variables:** Faceting relies on one or more categorical variables to define the subsets. For example, you can facet data by grouping it based on categories like "gender," "product type," or "time period."
3. **Consistency:** The individual subplots share the same scales for their axes, making it easier to compare patterns across the facets.
4. **Facet Labels:** You can label the facets to indicate which category or condition each subplot represents.
5. **Flexible Layouts:** Faceting allows for flexible arrangements of subplots, including rows, columns, and grids, depending on your specific requirements.

Faceting is a powerful tool for data exploration and visualization because it enables you to uncover insights and patterns within different segments of your data simultaneously. It's commonly used in packages like `ggplot2` in R, `seaborn` in Python, and similar data visualization libraries, making it easy to create faceted plots to gain a deeper understanding

🔄 Regenerate

Box Plot

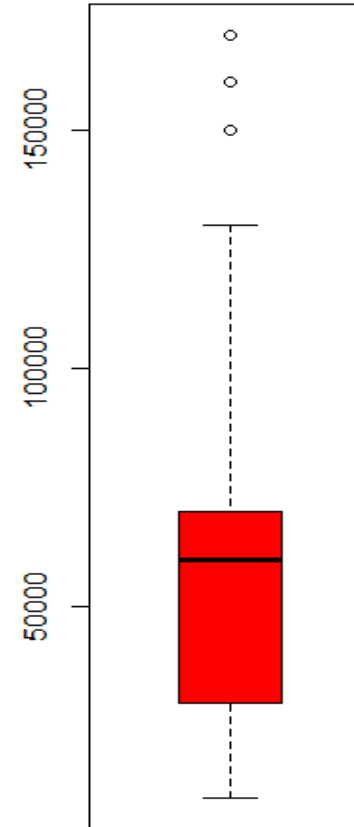
****Box Plot****

```
```{r}
par(mfrow = c(1, 2))
Create a multi-panel layout
par(mfrow = c(1, 2)) # 1 row and 2 columns for two plots side by side

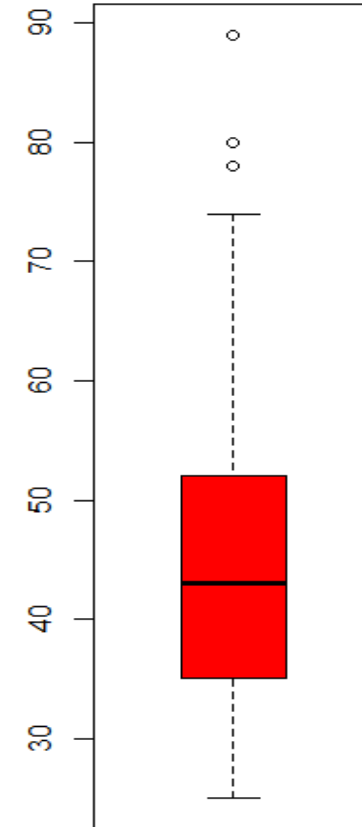
Create the first box plot for "Income"
boxplot(data_clean$Income, main = 'Boxplot of Income', col = 'red')

Create the second box plot for "Age"
boxplot(data_clean$Age, main = 'Boxplot of Age', col = 'red')|
```
```

Boxplot of Income



Boxplot of Age



Summary Statistics – Variable Correlation

```
cor(census[, 4:13])
```

```
> cor(census[, 4:13])
```

| | TotalPop | Men | Women | Hispanic | White | Black | Native | Asian | Pacific | Citizen |
|----------|-------------|-------------|-------------|--------------|-------------|-------------|-------------|-------------|--------------|-------------|
| TotalPop | 1.00000000 | 0.99987719 | 0.99988606 | 0.113882208 | -0.18737043 | 0.07529568 | -0.04603813 | 0.44787393 | 0.027103846 | 0.99637817 |
| Men | 0.99987719 | 1.00000000 | 0.99952669 | 0.114286493 | -0.18647769 | 0.07280682 | -0.04581787 | 0.44908456 | 0.028143374 | 0.99581922 |
| Women | 0.99988606 | 0.99952669 | 1.00000000 | 0.113466341 | -0.18818684 | 0.07767554 | -0.04623960 | 0.44660379 | 0.026096244 | 0.99668517 |
| Hispanic | 0.11388221 | 0.11428649 | 0.11346634 | 1.000000000 | -0.72495945 | -0.14454020 | -0.05611909 | 0.04424438 | -0.002645406 | 0.10409166 |
| White | -0.18737043 | -0.18647769 | -0.18818684 | -0.724959445 | 1.000000000 | -0.46618504 | -0.23156503 | -0.19934713 | -0.075080135 | -0.18313319 |
| Black | 0.07529568 | 0.07280682 | 0.07767554 | -0.144540197 | -0.46618504 | 1.000000000 | -0.09713960 | 0.01991644 | -0.035384600 | 0.08113776 |
| Native | -0.04603813 | -0.04581787 | -0.04623960 | -0.056119087 | -0.23156503 | -0.09713960 | 1.000000000 | -0.00266330 | 0.033927186 | -0.05010819 |
| Asian | 0.44787393 | 0.44908456 | 0.44660379 | 0.044244382 | -0.19934713 | 0.01991644 | -0.00266330 | 1.000000000 | 0.353934597 | 0.45673294 |
| Pacific | 0.02710385 | 0.02814337 | 0.02609624 | -0.002645406 | -0.07508014 | -0.03538460 | 0.03392719 | 0.35393460 | 1.000000000 | 0.02780608 |
| Citizen | 0.99637817 | 0.99581922 | 0.99668517 | 0.104091659 | -0.18313319 | 0.08113776 | -0.05010819 | 0.45673294 | 0.027806082 | 1.000000000 |

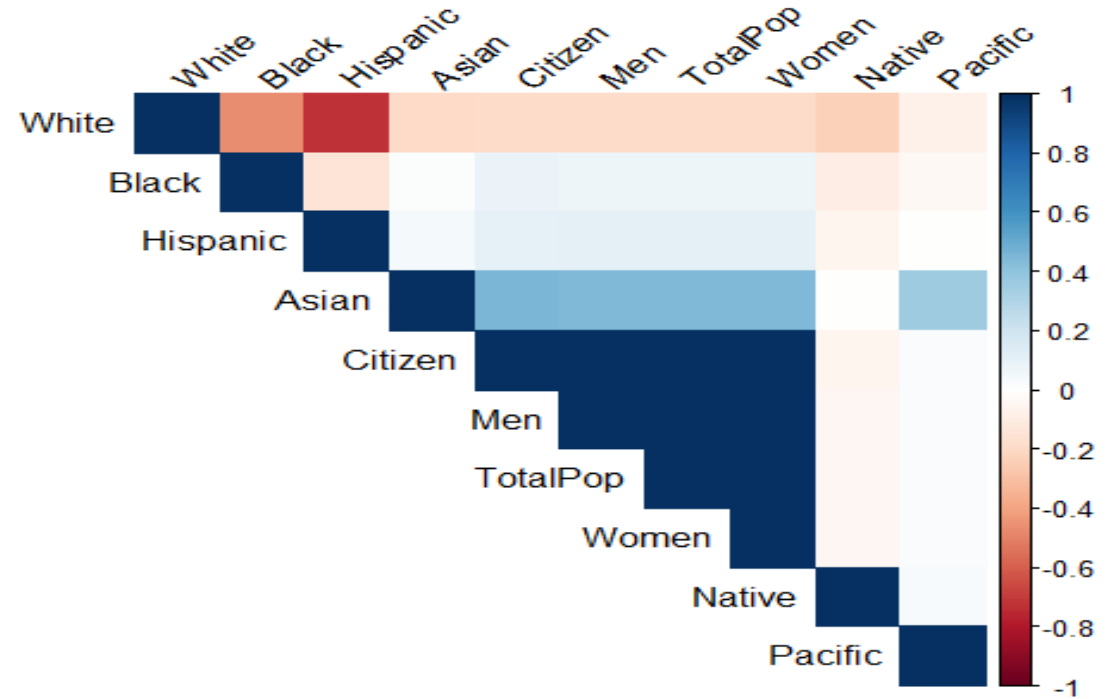
```
> pairs(census[, 4:13])
```

Summary Statistics – Variable Correlation

```
```{r}
Load the corrplot library
library(corrplot)

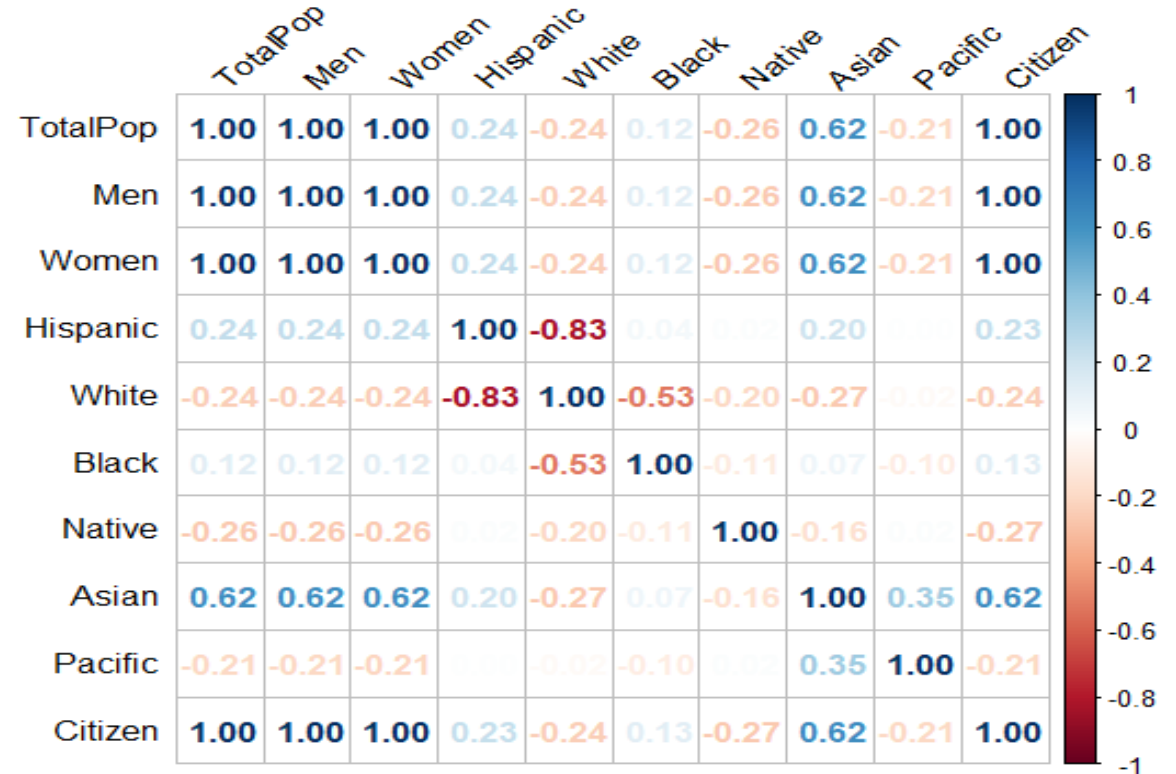
par(mfrow = c(1, 1))
Create a correlation matrix (example data)
Replace this with your own correlation matrix or data frame
cor_matrix <- cor(census[, 4:13])

Create the correlation plot
corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
 tl.col = "black", tl.srt = 45)
```
```



Summary Statistics – Variable Correlation

```
```{r}
Example correlation matrix (replace with your own data)
cor_matrix <- cor(cor(census[, 4:13]))
Create the correlation plot with numeric values
corrplot(cor_matrix, method = "number", type = "full",
 tl.col = "black", tl.srt = 45)
```
```

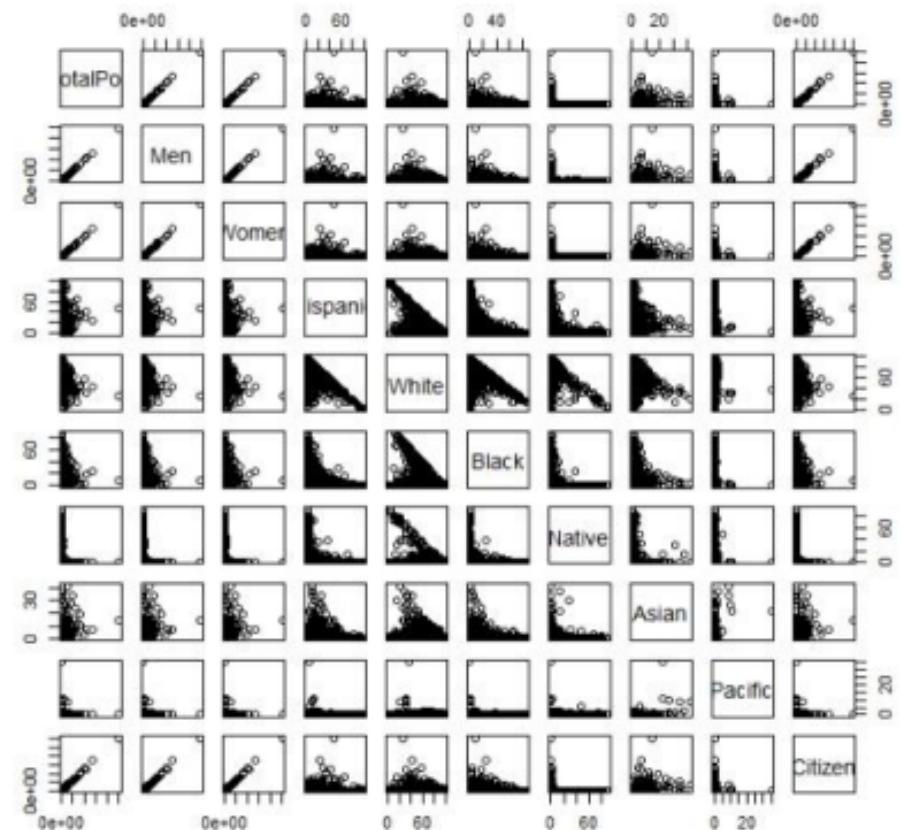


`cor()` function computes the correlation of a pair of variable.

```
> cor(census$TotalPop, census$Men)
[1] 0.9998772
```

We can use the `cor()` function to obtain the pairwise correlation between a set of numeric variables

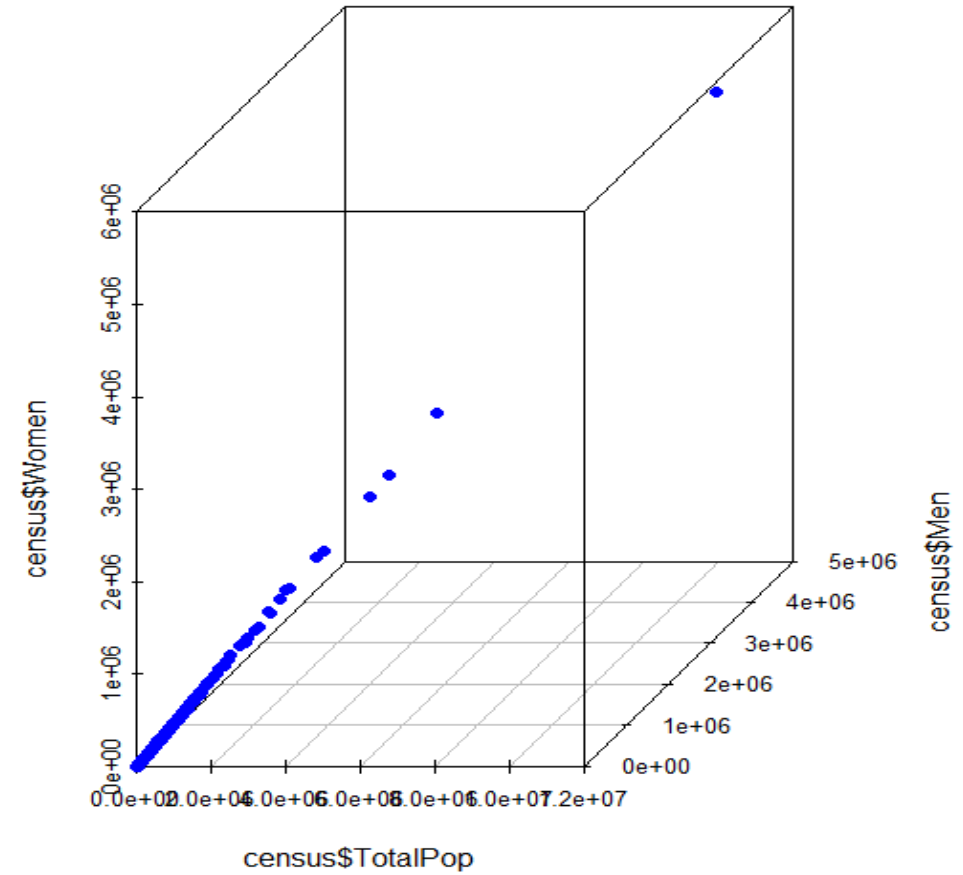
```
> cor(census[, 4:13])
> pairs(census[, 4:13])
```



3D scatter plot

```
```{r}
#install.packages("scatterplot3d")
Load the scatterplot3d library
library(scatterplot3d)

Create a 3D scatter plot
scatterplot3d(census[, 4], census[, 5], census[, 6], color = "blue", pch = 16,
 xlab = "census$TotalPop", ylab = "census$Men", zlab = "census$Women")
```
```



What is data normalization?

KS

What is data normalization?



Data normalization is a preprocessing technique used in data analysis and machine learning to transform data into a common scale or distribution. The goal of data normalization is to make different features or variables comparable and to improve the performance of machine learning algorithms. Here's why data normalization is important and how it's typically done:



1. **Improving Model Performance:** Many machine learning algorithms are sensitive to the scale of input features. Features with larger scales can dominate those with smaller scales, leading to suboptimal model performance. Data normalization can mitigate this issue.
2. **Ensuring Convergence:** Some optimization algorithms used in machine learning, such as gradient descent, converge faster and more reliably when the data is on a similar scale.
3. **Interpretability:** Normalized data can be easier to interpret because the scales of different variables are consistent.

Common methods of data normalization include:

- **Min-Max Scaling (Normalization):** This method scales data to a specific range, typically between 0 and 1. It is achieved by subtracting the minimum value of the feature from each data point and then dividing by the range (the difference between the maximum and minimum values).
- **Z-Score Standardization:** This method transforms data to have a mean of 0 and a standard deviation of 1. It is achieved by subtracting the mean of the feature from each data point and then dividing by the standard deviation.
- **Robust Scaling:** This method is similar to Min-Max scaling but uses the interquartile range (IQR) instead of the range, making it less sensitive to outliers.
- **Log Transformation:** When data is highly skewed or follows a power-law distribution, a log transformation can be applied to make it more normally distributed.
- **Box-Cox Transformation:** This is a family of power transformations that can be applied to data with various distributions to make it more Gaussian.

The choice of normalization method depends on the nature of the data and the specific requirements of the analysis or machine learning task. Proper data normalization can significantly improve the performance and reliability of models and analyses.

Need of Normalization

- Normalization is generally required when we are dealing with attributes on a different scale, otherwise, it may lead to a dilution in effectiveness of an important equally important attribute (on lower scale) because of other attribute having values on larger scale. In simple words, when multiple attributes are there but attributes have values on different scales, this may lead to poor data models while performing data mining operations. So they are normalized to bring all the attributes on the same scale.

Example:

| person_name | Salary | Year_of_experience | Expected Position Level |
|-------------|--------|--------------------|-------------------------|
| Aman | 100000 | 10 | 2 |
| Abhinav | 78000 | 7 | 4 |
| Ashutosh | 32000 | 5 | 8 |
| Dishi | 55000 | 6 | 7 |
| Abhishek | 92000 | 8 | 3 |
| Avantika | 120000 | 15 | 1 |
| Ayushi | 65750 | 7 | 5 |

The attributes salary and year_of_experience are on different scale and hence attribute salary can take high priority over attribute year_of_experience in the model.

Homework 5 (submitted to e3.nycu.edu.tw before Oct 25, 2023)

- Use some data sets with missing data to conduct Exploratory Data Analysis (EDA) using R, Python, and suitable computer packages.
- Explain what you find and why you choose these EDA methods.
- Detect if there are any outliers in this dataset.
- If there is any outlier in this dataset, how would you deal with it and why? Give your point of view what you found in this dataset.
- Discuss possible problems you plan to investigate for future studies

Possible sources of open datasets:

- UCI Machine Learning Repository
(<https://archive.ics.uci.edu/ml/datasets.php>)
- Kaggle Datasets (<https://www.kaggle.com/datasets>)