IFN501 - System Modeling and Simulation

Session 4: Introduction to Statistics (Part 1)

Daniel Febrian Sengkey

Department of Electrical Engineering
Faculty of Engineering
Universitas Sam Ratulangi

Outline

Outline

Definition

Statistics

A group of methods used to collect, analyze, present, and interpret data and to make decision [1].

- Statistics is used to make *educated guesses*, decisions made using statistical methods[1].

Definition

Statistics

A group of methods used to collect, analyze, present, and interpret data and to make decision [1].

- Statistics is used to make *educated guesses*, decisions made using statistical methods[1].
- There are 3 components of statistics [2]:

Definition

Statistics

A group of methods used to collect, analyze, present, and interpret data and to make decision [1].

- Statistics is used to make *educated guesses*, decisions made using statistical methods[1].
- There are 3 components of statistics [2]:
  - How best can we collect data?

Definition

Statistics

A group of methods used to collect, analyze, present, and interpret data and to make decision [1].

- Statistics is used to make *educated guesses*, decisions made using statistical methods[1].
- There are 3 components of statistics [2]:
    - How best can we collect data?
    - How should it be analyzed?

Definition

Statistics

A group of methods used to collect, analyze, present, and interpret data and to make decision [1].

- Statistics is used to make *educated guesses*, decisions made using statistical methods[1].
- There are 3 components of statistics [2]:
  - How best can we collect data?
  - How should it be analyzed?
  - What can we infer from the analysis?

Types of Statistics [1]

- Two aspects:

Types of Statistics [1]

- Two aspects:
  - Theoretical statistics

Types of Statistics [1]

- Two aspects:
  - Theoretical statistics
  - Applied statistics

Types of Statistics [1]

- Two aspects:
  - Theoretical statistics
  - Applied statistics ($\leftarrow$ what discusses in this course)

Types of Statistics [1]

- Two aspects:
  - Theoretical statistics
  - Applied statistics ($\leftarrow$ what discusses in this course)
- Applied statistics falls into two areas:

Types of Statistics [1]

- Two aspects:
  - Theoretical statistics
  - Applied statistics ($\leftarrow$ what discusses in this course)
- Applied statistics falls into two areas:
  - **Descriptive statistics**: methods for organizing, displaying, and describing data by using tables, graphs, and summary measures.

Types of Statistics [1]

- Two aspects:
  - Theoretical statistics
  - Applied statistics (← what discusses in this course)
- Applied statistics falls into two areas:
  - **Descriptive statistics**: methods for organizing, displaying, and describing data by using tables, graphs, and summary measures.
  - **Inferential statistics**: methods that use sample results to help make decision or predictions about a population.

Outline

Usage

- Applied statistics is used broadly and in various fields, from medical, education, engineering, etc.

Usage

- Applied statistics is used broadly and in various fields, from medical, education, engineering, etc.
- If you often read reports or newspaper containing result of a survey, you are likely to find some statistics terms such as *degree of freedom*, *confidence interval*, or α value.

Usage

- Applied statistics is used broadly and in various fields, from medical, education, engineering, etc.
- If you often read reports or newspaper containing result of a survey, you are likely to find some statistics terms such as *degree of freedom*, *confidence interval*, or α value.
- Statistics is commonly used in research as a tools to proofing hypothesis before the researchers come to conclusion.

Usage Scenario

- Example 1:

Usage Scenario

- Example 1:
  - Suppose that you just developed a network protocol.

Usage Scenario

- Example 1:
  - Suppose that you just developed a network protocol.
  - Your hypothesis is that your recently-developed protocol will bring lower latency, lower bandwidth usage, and provide higher throughput.

Usage Scenario

- Example 1:
  - Suppose that you just developed a network protocol.
  - Your hypothesis is that your recently-developed protocol will bring lower latency, lower bandwidth usage, and provide higher throughput.
  - How will you prove your theory?

Usage Scenario

- Example 1:
  - Suppose that you just developed a network protocol.
  - Your hypothesis is that your recently-developed protocol will bring lower latency, lower bandwidth usage, and provide higher throughput.
  - How will you prove your theory?
- Example 2:

Usage Scenario

- Example 1:
  - Suppose that you just developed a network protocol.
  - Your hypothesis is that your recently-developed protocol will bring lower latency, lower bandwidth usage, and provide higher throughput.
  - How will you prove your theory?
- Example 2:
  - in your internship, you made a video as a publication media (ad) for the organization that you worked for.

Usage Scenario

- Example 1:
  - Suppose that you just developed a network protocol.
  - Your hypothesis is that your recently-developed protocol will bring lower latency, lower bandwidth usage, and provide higher throughput.
  - How will you prove your theory?
- Example 2:
  - in your internship, you made a video as a publication media (ad) for the organization that you worked for.
  - You stated that your video will bring higher interest to the products sold by the organization you worked for.

Usage Scenario

- Example 1:
  - Suppose that you just developed a network protocol.
  - Your hypothesis is that your recently-developed protocol will bring lower latency, lower bandwidth usage, and provide higher throughput.
  - How will you prove your theory?
- Example 2:
  - in your internship, you made a video as a publication media (ad) for the organization that you worked for.
  - You stated that your video will bring higher interest to the products sold by the organization you worked for.
  - How you prove that your video brings higher interest?

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.
- The experiment addresses the use of stents to assist patient recovery after cardiac events and reduce the risk of another heart attack or death.

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.
- The experiment addresses the use of stents to assist patient recovery after cardiac events and reduce the risk of another heart attack or death.
- The research question is:

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.
- The experiment addresses the use of stents to assist patient recovery after cardiac events and reduce the risk of another heart attack or death.
- The research question is:
  Does the use of stents reduce the risk of stroke?

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.
- The experiment addresses the use of stents to assist patient recovery after cardiac events and reduce the risk of another heart attack or death.
- The research question is:
  Does the use of stents reduce the risk of stroke?
- The researchers already collected data on 451 at-risk patients.

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.
- The experiment addresses the use of stents to assist patient recovery after cardiac events and reduce the risk of another heart attack or death.
- The research question is:
  Does the use of stents reduce the risk of stroke?
- The researchers already collected data on 451 at-risk patients.
- The volunteer patient was randomly assigned to one of two groups:

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.
- The experiment addresses the use of stents to assist patient recovery after cardiac events and reduce the risk of another heart attack or death.
- The research question is:
  Does the use of stents reduce the risk of stroke?
- The researchers already collected data on 451 at-risk patients.
- The volunteer patient was randomly assigned to one of two groups:
  - Treatment group: 224 patients

Case Study: Stents to Prevent Stroke [2]

- A classic challenge in statistics: evaluating the efficacy of a medical treatment.
- The experiment addresses the use of stents to assist patient recovery after cardiac events and reduce the risk of another heart attack or death.
- The research question is:
  Does the use of stents reduce the risk of stroke?
- The researchers already collected data on 451 at-risk patients.
- The volunteer patient was randomly assigned to one of two groups:
  - Treatment group: 224 patients
  - Control group: 227 patients

Case Study: Stents to Prevent Stroke [2]

Table 1 : Results for five patients from the stent study.

Patient	group	0-30 days	0-365 days
1	treatment	no event	no event
2	treatment	stroke	stroke
3	treatment	no event	no event
⋮	⋮	⋮	
450	control	no event	no event
451	control	no event	no event

- The effect of stents was studied at two time points: 30 days and 365 days after enrollment.

Case Study: Stents to Prevent Stroke [2]

Table 1 : Results for five patients from the stent study.

Patient	group	0-30 days	0-365 days
1	treatment	no event	no event
2	treatment	stroke	stroke
3	treatment	no event	no event
⋮	⋮	⋮	
450	control	no event	no event
451	control	no event	no event

- The effect of stents was studied at two time points: 30 days and 365 days after enrollment.
- Table 1 summarizes results of 5 patients.

Case Study: Stents to Prevent Stroke [2]

Table 1 : Results for five patients from the stent study.

Patient	group	0-30 days	0-365 days
1	treatment	no event	no event
2	treatment	stroke	stroke
3	treatment	no event	no event
⋮	⋮	⋮	
450	control	no event	no event
451	control	no event	no event

- The effect of stents was studied at two time points: 30 days and 365 days after enrollment.
- Table 1 summarizes results of 5 patients.
- The result is recorded as ”stroke” or ”no event” depending on the event at the related time period.

Case Study: Stents to Prevent Stroke [2]

Table 2 : Descriptive statistics for the stent study.

	0-30 days		0-365 days	
	stroke	no event	stroke	no event
treatment	33	191	45	179
control	13	214	28	199
Total	46	405	73	378

- Considering each patient data individually is a long and cumbersome path to answer the research question.

Case Study: Stents to Prevent Stroke [2]

Table 2 : Descriptive statistics for the stent study.

	0-30 days		0-365 days	
	stroke	no event	stroke	no event
treatment	33	191	45	179
control	13	214	28	199
Total	46	405	73	378

- Considering each patient data individually is a long and cumbersome path to answer the research question.
- By performing statistical analysis we can consider all of the data at once.

Case Study: Stents to Prevent Stroke [2]

Table 2 : Descriptive statistics for the stent study.

	0-30 days		0-365 days	
	stroke	no event	stroke	no event
treatment	33	191	45	179
control	13	214	28	199
Total	46	405	73	378

- Considering each patient data individually is a long and cumbersome path to answer the research question.
- By performing statistical analysis we can consider all of the data at once.
- Table 2 shows a summary of the raw data in a more compact way.

Case Study: Stents to Prevent Stroke [2]

Table 2 : Descriptive statistics for the stent study.

	0-30 days		0-365 days	
	stroke	no event	stroke	no event
treatment	33	191	45	179
control	13	214	28	199
Total	46	405	73	378

- Considering each patient data individually is a long and cumbersome path to answer the research question.
- By performing statistical analysis we can consider all of the data at once.
- Table 2 shows a summary of the raw data in a more compact way.
- For instance, from it we can gather information that 33 patients in the treatment group had stroke in the 30 days period.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].

[1]A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 =$

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 =$

[1]A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
    - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.
  - Proportion who had a stroke in the control group: $28/227 =$

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.
  - Proportion who had a stroke in the control group: $28/227 = 0.12 =$

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.
  - Proportion who had a stroke in the control group: $28/227 = 0.12 = 12\%$.

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.
  - Proportion who had a stroke in the control group: $28/227 = 0.12 = 12\%$.
- The summary statistics are useful in looking for differences in the group: 8% of patients in the treatment group had a stroke.

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.
  - Proportion who had a stroke in the control group: $28/227 = 0.12 = 12\%$.
- The summary statistics are useful in looking for differences in the group: 8% of patients in the treatment group had a stroke.
- The information is important for two reasons:

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.
  - Proportion who had a stroke in the control group: $28/227 = 0.12 = 12\%$.
- The summary statistics are useful in looking for differences in the group: 8% of patients in the treatment group had a stroke.
- The information is important for two reasons:
  - The result is contrast to what doctors expected.

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- Based on the table we can compute **summary statistics**[1].
- A simple example for our stents experiment is the proportion of people who had a stroke in the treatment and control groups:
  - Proportion who had a stroke in the treatment group: $45/224 = 0.20 = 20\%$.
  - Proportion who had a stroke in the control group: $28/227 = 0.12 = 12\%$.
- The summary statistics are useful in looking for differences in the group: 8% of patients in the treatment group had a stroke.
- The information is important for two reasons:
  - The result is contrast to what doctors expected.
  - A statistical question: do the data show a real difference between the groups?

[1] A single number summarizing a large amount of data.

Case Study: Stents to Prevent Stroke [2]

- The second question leads to a bigger question:

Case Study: Stents to Prevent Stroke [2]

- The second question leads to a bigger question:
  Is the difference so large that we should reject the notion that it was due to chance?

Case Study: Stents to Prevent Stroke [2]

- The second question leads to a bigger question:
  Is the difference so large that we should reject the notion that it was due to chance?
- Consider this:

Case Study: Stents to Prevent Stroke [2]

- The second question leads to a bigger question:
  Is the difference so large that we should reject the notion that it was due to chance?
- Consider this:
  - The probability of getting the image of Garuda[2] if you flip a coin is 50%.

Case Study: Stents to Prevent Stroke [2]

- The second question leads to a bigger question:
  Is the difference so large that we should reject the notion that it was due to chance?
- Consider this:
  - The probability of getting the image of Garuda[2] if you flip a coin is 50%.

[2] An Indonesian context, instead of head or tail in American context.

Case Study: Stents to Prevent Stroke [2]

- The second question leads to a bigger question:
  Is the difference so large that we should reject the notion that it was due to chance?
- Consider this:
  - The probability of getting the image of Garuda[2] if you flip a coin is 50%.
  - However, when you flip a coin 100 times, you may not observe exactly 50 images of Garuda.

[2] An Indonesian context, instead of head or tail in American context.

Case Study: Stents to Prevent Stroke [2]

- The second question leads to a bigger question:
  Is the difference so large that we should reject the notion that it was due to chance?
- Consider this:
  - The probability of getting the image of Garuda[2] if you flip a coin is 50%.
  - However, when you flip a coin 100 times, you may not observe exactly 50 images of Garuda.
- The type of fluctuation as happened in the coin experiment is part of almost any type of data generating process.

[2] An Indonesian context, instead of head or tail in American context.

Case Study: Stents to Prevent Stroke [2]

- It is possible that the 8% difference in the stent study is due to natural variation.

Case Study: Stents to Prevent Stroke [2]

- It is possible that the 8% difference in the stent study is due to natural variation.
- The question in the previous slide can be answered with the statistical tools that we are going to discover in this course[3].

Case Study: Stents to Prevent Stroke [2]

- It is possible that the 8% difference in the stent study is due to natural variation.
- The question in the previous slide can be answered with the statistical tools that we are going to discover in this course[3].

---

[3]Meanwhile we can comprehend with the published analysis: there was compelling evidence of harm by stents in this study of stroke patients. Please be careful, do not generalize the results of this study to all patients and all stents.

Outline

Definitions [1]

- Element or Member: a specific subject or object (such as a person, firm, etc) about which the information is collected.

Definitions [1]

- Element or Member: a specific subject or object (such as a person, firm, etc) about which the information is collected.
- Variable: a characteristic under study that assumes different values for different elements.

Definitions [1]

- Element or Member: a specific subject or object (such as a person, firm, etc) about which the information is collected.
- Variable: a characteristic under study that assumes different values for different elements.
- Observation or measurement: the value of a variable for an element.

Definitions [1]

- Element or Member: a specific subject or object (such as a person, firm, etc) about which the information is collected.
- Variable: a characteristic under study that assumes different values for different elements.
- Observation or measurement: the value of a variable for an element.
- Data set: a collection of observations on one or more variables.

The `email50` Data set [2]

Table 3 : Four rows from the `email50` data matrix.

	spam	num_char	line_breaks	format	number
1	no	21,705	551	html	small
2	no	7,011	183	html	big
3	yes	631	28	text	none
⋮	⋮	⋮	⋮	⋮	⋮
50	no	15,829	242	html	small

- Table 3 shows a data set of 50 e-mails received during early 2012[4].

The email50 Data set [2]

Table 3 : Four rows from the email50 data matrix.

	spam	num_char	line_breaks	format	number
1	no	21,705	551	html	small
2	no	7,011	183	html	big
3	yes	631	28	text	none
⋮	⋮	⋮	⋮	⋮	⋮
50	no	15,829	242	html	small

- Table 3 shows a data set of 50 e-mails received during early 2012[4].

[4] The data in this table represent a data matrix, which is a common way to organize data

The email50 Data set [2]

Table 3 : Four rows from the email50 data matrix.

	spam	num_char	line_breaks	format	number
1	no	21,705	551	html	small
2	no	7,011	183	html	big
3	yes	631	28	text	none
⋮	⋮	⋮	⋮	⋮	⋮
50	no	15,829	242	html	small

- Table 3 shows a data set of 50 e-mails received during early 2012[4].
- The columns represent the variables.

[4] The data in this table represent a data matrix, which is a common way to organize data

The `email50` Data set [2]

Table 3 : Four rows from the `email50` data matrix.

	spam	num_char	line_breaks	format	number
1	no	21,705	551	html	small
2	no	7,011	183	html	big
3	yes	631	28	text	none
⋮	⋮	⋮	⋮	⋮	⋮
50	no	15,829	242	html	small

- Table 3 shows a data set of 50 e-mails received during early 2012[4].
- The columns represent the variables.
- Each row represents a single e-mail or a **case**, which is sometimes called as an observational unit.

[4] The data in this table represent a data matrix, which is a common way to organize data

The `email50` Data set [2]

Table 3 : Four rows from the `email50` data matrix.

	spam	num_char	line_breaks	format	number
1	no	21,705	551	html	small
2	no	7,011	183	html	big
3	yes	631	28	text	none
⋮	⋮	⋮	⋮	⋮	⋮
50	no	15,829	242	html	small

- Table 3 shows a data set of 50 e-mails received during early 2012[4].
- The columns represent the variables.
- Each row represents a single e-mail or a **case**, which is sometimes called as an observational unit.
- Each received e-mail (identified with its number in the first column) is an element according to the definition in [1].

[4]The data in this table represent a data matrix, which is a common way to organize data

The `email50` Data set [2]

Table 3 : Four rows from the `email50` data matrix.

	spam	num_char	line_breaks	format	number
1	no	21,705	551	html	small
2	no	7,011	183	html	big
3	yes	631	28	text	none
⋮	⋮	⋮	⋮	⋮	⋮
50	no	15,829	242	html	small

- Table 3 shows a data set of 50 e-mails received during early 2012[4].
- The columns represent the variables.
- Each row represents a single e-mail or a **case**, which is sometimes called as an observational unit.
- Each received e-mail (identified with its number in the first column) is an element according to the definition in [1].
- Each cell in an element-variable pair is an observation or measurement.

[4]The data in this table represent a data matrix, which is a common way to organize data

Types of Variables [1]

- Quantitative variable: a variable that can be measured numerically. The collected data are called quantitative data.

Types of Variables [1]

- Quantitative variable: a variable that can be measured numerically. The collected data are called quantitative data.
  - Discrete variable: a variable whose values are countable[5].

Types of Variables [1]

- Quantitative variable: a variable that can be measured numerically. The collected data are called quantitative data.
    - Discrete variable: a variable whose values are countable[5].

[5]Only assume certain values with no intermediate values

Types of Variables [1]

- Quantitative variable: a variable that can be measured numerically. The collected data are called quantitative data.
  - Discrete variable: a variable whose values are countable[5].
  - Continuous variable: a variable that can assume any numerical value over a certain interval(s).

[5] Only assume certain values with no intermediate values

Types of Variables [1]

- Quantitative variable: a variable that can be measured numerically. The collected data are called quantitative data.
  - Discrete variable: a variable whose values are countable[5].
  - Continuous variable: a variable that can assume any numerical value over a certain interval(s).
- Qualitative variable: a variable that cannot assume a numerical value but can be classified into two or more nonnumeric categories.

[5] Only assume certain values with no intermediate values

Cross-Section vs Time-Series Data [1]

- Cross-section data are the data collected on <u>different elements</u> at the same point or for the same period of time. (Example: patients and stroke events in the stents experiment).

Cross-Section vs Time-Series Data [1]

- Cross-section data are the data collected on different elements at the same point or for the same period of time. (Example: patients and stroke events in the stents experiment).
- Time-series data are the data collected on the same element for the same variable at different points in time or for different periods of time. (Example: number of accepted students accepted at the Faculty of Engineering from year 2000 until 2016.

Data Sources [1]

- Internal sources: company’s personnel files, accounting records, etc.

Data Sources [1]

- Internal sources: company's personnel files, accounting records, etc.
- External sources: government data, openly available dataset, data from other institutions.

Data Sources [1]

- Internal sources: company's personnel files, accounting records, etc.
- External sources: government data, openly available dataset, data from other institutions.
- Surveys

Data Sources [1]

- Internal sources: company's personnel files, accounting records, etc.
- External sources: government data, openly available dataset, data from other institutions.
- Surveys
- Experiments

Outline

Population and Sample [1]

- **Population**: all elements whose characteristics are being studied.

Population and Sample [1]

- **Population**: all elements whose characteristics are being studied.
- **Target population**: the population that being studied.

Population and Sample [1]

- **Population**: all elements whose characteristics are being studied.
- **Target population**: the population that being studied.
- **Sample**: a portion of the population selected for study.

Experiments [2]

- **Experiments**: studies where the researchers assign treatments to cases.

Experiments [2]

- **Experiments**: studies where the researchers assign treatments to cases.
- **Randomized Experiments**: experiments that include randomization.

Experiments [2]

- **Experiments**: studies where the researchers assign treatments to cases.
- **Randomized Experiments**: experiments that include randomization.
- There are four principles of randomized experiments:

Experiments [2]

- **Experiments**: studies where the researchers assign treatments to cases.
- **Randomized Experiments**: experiments that include randomization.
- There are four principles of randomized experiments:
    - Controlling

Experiments [2]

- **Experiments**: studies where the researchers assign treatments to cases.
- **Randomized Experiments**: experiments that include randomization.
- There are four principles of randomized experiments:
  - Controlling
  - Randomization

Experiments [2]

- **Experiments**: studies where the researchers assign treatments to cases.
- **Randomized Experiments**: experiments that include randomization.
- There are four principles of randomized experiments:
  - Controlling
  - Randomization
  - Replication

Experiments [2]

- **Experiments**: studies where the researchers assign treatments to cases.
- **Randomized Experiments**: experiments that include randomization.
- There are four principles of randomized experiments:
  - Controlling
  - Randomization
  - Replication
  - Blocking

Outline

GNU R

- R is an integrated suite of software for data manipulation, calculation and graphical display [3].

GNU R

- R is an integrated suite of software for data manipulation, calculation and graphical display [3].
- Although it seems that R commonly used for statistics, R has broader capabilities [3].

GNU R

- R is an integrated suite of software for data manipulation, calculation and graphical display [3].
- Although it seems that R commonly used for statistics, R has broader capabilities [3].
- R is an open source solution for data analysis, and it has many features to recommend as mentioned in [4].

Data Structures in R [4]

Data structures in R are:

- Vector: one-dimensional array that can hold numeric data, character data, or logical data. All data must be of same mode

Data Structures in R [4]

Data structures in R are:

- Vector: one-dimensional array that can hold numeric data, character data, or logical data. All data must be of same mode
- Matrix: a two-dimensional array. Same with vector, all data have the same mode

Data Structures in R [4]

Data structures in R are:

- Vector: one-dimensional array that can hold numeric data, character data, or logical data. All data must be of same mode
- Matrix: a two-dimensional array. Same with vector, all data have the same mode
- Array: similar with matrix, but with more dimensions.

Data Structures in R [4]

Data structures in R are:

- Vector: one-dimensional array that can hold numeric data, character data, or logical data. All data must be of same mode
- Matrix: a two-dimensional array. Same with vector, all data have the same mode
- Array: similar with matrix, but with more dimensions.
- Data frame: similar with matrix, but each of its column can have different modes.

Data Structures in R [4]

Data structures in R are:

- Vector: one-dimensional array that can hold numeric data, character data, or logical data. All data must be of same mode
- Matrix: a two-dimensional array. Same with vector, all data have the same mode
- Array: similar with matrix, but with more dimensions.
- Data frame: similar with matrix, but each of its column can have different modes.
- List: the most complex data types in R. It can contain several objects with different type in each object.

Outline

Frequently Used Functions
c()

- **C**oncatenate several values/objects.

Frequently Used Functions

c()

- **C**oncatenate several values/objects.
- Mostly used to create a vector.

Frequently Used Functions

c()

- **C**oncatenate several values/objects.
- Mostly used to create a vector.
- ```
  a <- c(1, 2, 3, 4, 5)
  b <- c("u", "n", "s", "r", "a", "t")
  c <- c(TRUE, FALSE, FALSE, F, F, T)

  a

  ## [1] 1 2 3 4 5

  b

  ## [1] "u" "n" "s" "r" "a" "t"

  c

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

Frequently Used Functions

c()

- Scalar is a vector with single-element.

Frequently Used Functions

c()

- Scalar is a vector with single-element.
- ```r
  d <- 1
  e <- TRUE
  f <- "unsrat"

  d
  ```
  ```
  ## [1] 1
  ```
  ```r
  e
  ```
  ```
  ## [1] TRUE
  ```
  ```r
  f
  ```
  ```
  ## [1] "unsrat"
  ```
  ```

- We can create sequence of values from `a:b`

Frequently Used Functions

Sequence

- We can create sequence of values from `a:b`
- ```r
  1:10
  ```

  ```
  ##  [1]  1  2  3  4  5  6  7  8  9 10
  ```

  ```r
  10:1
  ```

  ```
  ##  [1] 10  9  8  7  6  5  4  3  2  1
  ```
  ```

Frequently Used Functions

Sequence

- We can create sequence of values from `a:b`
- ```
  1:10

  ##  [1]  1  2  3  4  5  6  7  8  9 10

  10:1

  ##  [1] 10  9  8  7  6  5  4  3  2  1
  ```
- As usual, we can assign it to a variable
  ```

Frequently Used Functions

Sequence

- We can create sequence of values from `a:b`

- ```r
  1:10
  ```
  ```
  ##  [1]  1  2  3  4  5  6  7  8  9 10
  ```
  ```r
  10:1
  ```
  ```
  ##  [1] 10  9  8  7  6  5  4  3  2  1
  ```

- As usual, we can assign it to a variable

- ```r
  g <- 1:10
  h <- 10:1


  g
  ```
  ```
  ##  [1]  1  2  3  4  5  6  7  8  9 10
  ```
  ```r
  h
  ```
  ```
  ##  [1] 10  9  8  7  6  5  4  3  2  1
  ```

- To create sequence with defined step, we can use the function `seq(from = a, to = b, by = c)`

- To create sequence with defined step, we can use the function `seq(from = a, to = b, by = c)`

-
```r
seq(0, 10, 2)
```

```
## [1]  0  2  4  6  8 10
```

```r
seq(10, 0, -2.5)
```

```
## [1] 10.0  7.5  5.0  2.5  0.0
```

```r
seq(1, 10, 0.5)
```

```
##  [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5
##  [ reached getOption("max.print") -- omitted 9 entries ]
```

Outline

Loading Data into R

- R could import data from various sources[2]:

Loading Data into R

- R could import data from various sources[2]:
  - Direct keyboard input

Loading Data into R

- R could import data from various sources[2]:
  - Direct keyboard input
  - Statistical packages: SAS, SPSS, Stata

Loading Data into R

- R could import data from various sources[2]:
  - Direct keyboard input
  - Statistical packages: SAS, SPSS, Stata
  - Text files: ASCII, XML, Webscraping, CSV

Loading Data into R

- R could import data from various sources[2]:
  - Direct keyboard input
  - Statistical packages: SAS, SPSS, Stata
  - Text files: ASCII, XML, Webscraping, CSV
  - DBMS: SQL, MySQL, Oracle, MS Access

Loading Data into R

- R could import data from various sources[2]:
  - Direct keyboard input
  - Statistical packages: SAS, SPSS, Stata
  - Text files: ASCII, XML, Webscraping, CSV
  - DBMS: SQL, MySQL, Oracle, MS Access
- Beside those formats, R could save an entire workspace into a `.RData` file.

Outline

- Comma Separated Value (CSV) is a famous format to keep data in a structured text file.

- Comma Separated Value (CSV) is a famous format to keep data in a structured text file.
- The cells are separated by either comma (,) or sometimes a semicolon (;).

Loading Data into R
CSV

- Comma Separated Value (CSV) is a famous format to keep data in a structured text file.
- The cells are separated by either comma (,) or sometimes a semicolon (;).
- For example, this is the content of the `IFC6503-A-2016.csv` which already available to you.

Loading Data into R
CSV

- Comma Separated Value (CSV) is a famous format to keep data in a structured text file.
- The cells are separated by either comma (,) or sometimes a semicolon (;).
- For example, this is the content of the `IFC6503-A-2016.csv` which already available to you.

```
"No","Nilai_Akhir","Grade"
1,0,"E"
2,0,"E"
3,82.95,"A"
4,71.76,"B"
5,56.55,"C"
```

Loading Data into R

CSV

- To read .csv file, use the `read.csv()` function.

```
read.csv("IFC6503-A-2016.csv")
```

```
##    No Nilai_Akhir Grade
## 1   1         0.0     E
## 2   2         0.0     E
## 3   3        83.0     A
## 4   4        71.8     B
## 5   5        56.5     C
## 6   6        86.8     A
## 7   7        96.7     A
## 8   8        89.2     A
##  [ reached getOption("max.print") -- omitted 34 rows ]
```

- To read .csv file, use the `read.csv()` function.

```
read.csv("IFC6503-A-2016.csv")

##    No Nilai_Akhir Grade
## 1   1         0.0     E
## 2   2         0.0     E
## 3   3        83.0     A
## 4   4        71.8     B
## 5   5        56.5     C
## 6   6        86.8     A
## 7   7        96.7     A
## 8   8        89.2     A
##  [ reached getOption("max.print") -- omitted 34 rows ]
```

- The contents are shown but not saved.

Loading Data into R

CSV

- Assign the output of `read.csv()` to a variable so we can access it later.

```
IFC6503.A.2016 <- read.csv("IFC6503-A-2016.csv")
```

Loading Data into R

CSV

- Assign the output of `read.csv()` to a variable so we can access it later.

```
IFC6503.A.2016 <- read.csv("IFC6503-A-2016.csv")
```

- The dot (.) which means access a method/attribute in Object Oriented Programming has no meaning in R. So it is safe to use it in a variable name.

- Assign the output of `read.csv()` to a variable so we can access it later.

```
IFC6503.A.2016 <- read.csv("IFC6503-A-2016.csv")
```

- The dot (.) which means access a method/attribute in Object Oriented Programming has no meaning in R. So it is safe to use it in a variable name.
- The `IFC6503.A.2016` is now available in the workspace.

- Assign the output of `read.csv()` to a variable so we can access it later.

```
IFC6503.A.2016 <- read.csv("IFC6503-A-2016.csv")
```

- The dot (.) which means access a method/attribute in Object Oriented Programming has no meaning in R. So it is safe to use it in a variable name.
- The `IFC6503.A.2016` is now available in the workspace.
- The objects in the active workspace are saved in the computer's memory.

Loading Data into R

CSV

- Assign the output of `read.csv()` to a variable so we can access it later.

```
IFC6503.A.2016 <- read.csv("IFC6503-A-2016.csv")
```

- The dot (.) which means access a method/attribute in Object Oriented Programming has no meaning in R. So it is safe to use it in a variable name.
- The `IFC6503.A.2016` is now available in the workspace.
- The objects in the active workspace are saved in the computer's memory.
- To list all the objects in the workspace, use the `ls()` function.

```
ls()

## [1] "IFC6503.A.2016"
```

Loading Data into R

CSV

- To check the content of `IFC6503.A.2016` variable, enter its name and press enter.

```
IFC6503.A.2016
```

```
##    No Nilai_Akhir Grade
## 1   1         0.0     E
## 2   2         0.0     E
## 3   3        83.0     A
## 4   4        71.8     B
## 5   5        56.5     C
## 6   6        86.8     A
##  [ reached getOption("max.print") -- omitted 36 rows ]
```

Outline

Loading Data into R

- RData is a format used to save an entire R workspace.

Loading Data into R

- RData is a format used to save an entire R workspace.
- R workspace is the collection of all objects that are available.

Loading Data into R

- RData is a format used to save an entire R workspace.
- R workspace is the collection of all objects that are available.
- These objects are located in the computer's memory.

Loading Data into R

- RData is a format used to save an entire R workspace.
- R workspace is the collection of all objects that are available.
- These objects are located in the computer's memory.
- Therefore the number of loaded objects depends on the size of memory and the size of each object itself.

Loading Data into R

- To load an RData into workspace, use the `load()` function.

Loading Data into R

- To load an RData into workspace, use the `load()` function.
- Remember to pass the parameter (file location/file name) as a string.

```
load("ripv2-nRouters-experiment-omnetpp.RData")
```

Loading Data into R

- To load an RData into workspace, use the `load()` function.
- Remember to pass the parameter (file location/file name) as a string.

```
load("ripv2-nRouters-experiment-omnetpp.RData")
```

- As the `load()` function loads a workspace, there is no need for variable assignment.

```
ls()

## [1] "allData"        "IFC6503.A.2016"
```

Loading Data into R

- Check the mode of an object by using `mode()` function.

```
mode(IFC6503.A.2016)

## [1] "list"
```

Loading Data into R

- Check the mode of an object by using `mode()` function.

```
mode(IFC6503.A.2016)

## [1] "list"
```

- Similarly, to check the class of an object we can use the `class()` function.

```
class(IFC6503.A.2016)

## [1] "data.frame"
```

Loading Data into R

- Check the mode of an object by using `mode()` function.

```
mode(IFC6503.A.2016)

## [1] "list"
```

- Similarly, to check the class of an object we can use the `class()` function.

```
class(IFC6503.A.2016)

## [1] "data.frame"
```

- To see the structure of an object, use the `str()` function.

```
str(IFC6503.A.2016)

## 'data.frame': 42 obs. of  3 variables:
##  $ No         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Nilai_Akhir: num  0 0 83 71.8 56.5 ...
##  $ Grade      : Factor w/ 6 levels "A","B","B+","C",..: 6
```

Outline

Learning Sources

- You can start exploring GNU R here:

Learning Sources

- You can start exploring GNU R here:
  - Installation: `https://cran.r-project.org/doc/manuals/r-release/R-admin.html`

Learning Sources

- You can start exploring GNU R here:
  - Installation: `https://cran.r-project.org/doc/manuals/r-release/R-admin.html`
  - Introduction to R: `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`

Learning Sources

- You can start exploring GNU R here:
  - Installation: `https://cran.r-project.org/doc/manuals/r-release/R-admin.html`
  - Introduction to R: `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`
  - Some recommended tutorials:

Learning Sources

- You can start exploring GNU R here:
  - Installation: `https://cran.r-project.org/doc/manuals/r-release/R-admin.html`
  - Introduction to R: `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`
  - Some recommended tutorials:
    - `www.r-tutor.com`

Learning Sources

- You can start exploring GNU R here:
  - Installation: `https://cran.r-project.org/doc/manuals/r-release/R-admin.html`
  - Introduction to R: `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`
  - Some recommended tutorials:
    - `www.r-tutor.com`
    - R Section at TutorialsPoint

Learning Sources

- You can start exploring GNU R here:
  - Installation: `https://cran.r-project.org/doc/manuals/r-release/R-admin.html`
  - Introduction to R: `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`
  - Some recommended tutorials:
    - `www.r-tutor.com`
    - R Section at TutorialsPoint
    - `http://www.r-bloggers.com/`

Learning Sources

- You can start exploring GNU R here:
  - Installation: `https://cran.r-project.org/doc/manuals/r-release/R-admin.html`
  - Introduction to R: `https://cran.r-project.org/doc/manuals/r-release/R-intro.html`
  - Some recommended tutorials:
    - `www.r-tutor.com`
    - R Section at TutorialsPoint
    - `http://www.r-bloggers.com/`
  - A powerful IDE for R: RStudio

Outline

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
    1.1 What are the *mode* and *class* of the recently assigned variable?

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
   1.1 What are the *mode* and *class* of the recently assigned variable?
   1.2 Observe the *structure* of the recently assigned variable!

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
   1.1 What are the *mode* and *class* of the recently assigned variable?
   1.2 Observe the *structure* of the recently assigned variable!
2. Take the contents of `Nilai_Akhir`[6] column and assign them to a new variable!

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
   1.1 What are the *mode* and *class* of the recently assigned variable?
   1.2 Observe the *structure* of the recently assigned variable!
2. Take the contents of `Nilai_Akhir`[6] column and assign them to a new variable!

[6] final score

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
   1.1 What are the *mode* and *class* of the recently assigned variable?
   1.2 Observe the *structure* of the recently assigned variable!
2. Take the contents of `Nilai_Akhir`[6] column and assign them to a new variable!
   2.1 What are the *mode* and *class* of the recently assigned variable?

[6] final score

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
   1.1 What are the *mode* and *class* of the recently assigned variable?
   1.2 Observe the *structure* of the recently assigned variable!
2. Take the contents of `Nilai_Akhir`[6] column and assign them to a new variable!
   2.1 What are the *mode* and *class* of the recently assigned variable?
   2.2 Observe the *structure* of the recently assigned variable!

[6] final score

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
   1.1 What are the *mode* and *class* of the recently assigned variable?
   1.2 Observe the *structure* of the recently assigned variable!
2. Take the contents of `Nilai_Akhir`[6] column and assign them to a new variable!
   2.1 What are the *mode* and *class* of the recently assigned variable?
   2.2 Observe the *structure* of the recently assigned variable!
   2.3 How many numbers[7] are stored in the recently assigned variable?

[6] final score

Exercise 1

1. Read the IFC6503-A-2016.csv into your R workspace and assign its contents to a variable!
   1.1 What are the *mode* and *class* of the recently assigned variable?
   1.2 Observe the *structure* of the recently assigned variable!
2. Take the contents of `Nilai_Akhir`[6] column and assign them to a new variable!
   2.1 What are the *mode* and *class* of the recently assigned variable?
   2.2 Observe the *structure* of the recently assigned variable!
   2.3 How many numbers[7] are stored in the recently assigned variable?

[6] final score
[7] vector length

Exercise 1

3. Take all the scores ≥ 55 from `Nilai_Akhir` column and assign them to a new variable!

Exercise 1

3. Take all the scores ≥ 55 from `Nilai_Akhir` column and assign them to a new variable!
4. Take all the ”A” grades from `Grade` column and assign them to a new variable!

Exercise 1

3. Take all the scores ≥ 55 from `Nilai_Akhir` column and assign them to a new variable!
4. Take all the ”A” grades from `Grade` column and assign them to a new variable!
5. To get the ”A” grade, a student must achieve final score ≥ 80. Does this dataset comply to this rule?

Outline

Raw Data

- Raw data: data recorded in the sequence in which they are collected and before they are processed or ranked [1].

Raw Data

- Raw data: data recorded in the sequence in which they are collected and before they are processed or ranked [1].
- Example of raw data are the final scores of the students in a course.

Raw Data

- Raw data: data recorded in the sequence in which they are collected and before they are processed or ranked [1].
- Example of raw data are the final scores of the students in a course.
- Recall the example dataset `IFC6503.A.2016`, current sequences of either final scores and grade are raw data.

```
IFC6503.A.2016$Nilai_Akhir
```

```
##  [1]  0.0  0.0 83.0 71.8 56.5 86.8 96.7 89.2 48.7 72.0  5
## [12] 84.8 96.7 78.3 83.0 80.5 85.8 56.9 80.2 78.1 90.9 51
## [23] 66.2 93.0 82.6 52.6 84.0 76.1 48.1 88.0 87.3 61.0 18
## [34] 68.2 70.6 69.0 65.7 93.6 92.1  5.0 62.6 42.1
```

Outline

Outline

Frequency Distributions [1]

- Frequency distribution exhibits how the frequencies are distributed over various categories.

Frequency Distributions [1]

- Frequency distribution exhibits how the frequencies are distributed over various categories.
- For example, `IFC6503.A.2016` dataset contains grades of all course participants.

Frequency Distributions [1]

- Frequency distribution exhibits how the frequencies are distributed over various categories.
- For example, `IFC6503.A.2016` dataset contains grades of all course participants.
- In this case grade is the category of the data.

Frequency Distributions [1]

Table 4 : Grade frequencies of course IFC 6503 Class A 2016

Grade	Number of Students
A	18
B+	3
B	3
C+	6
C	7
D	0
E	5

- Table 4 shows the frequency distribution of grades from the `IFC6503.A.2016` dataset.

Frequency Distributions [1]

Table 4 : Grade frequencies of course IFC 6503 Class A 2016

Grade	Number of Students
A	18
B+	3
B	3
C+	6
C	7
D	0
E	5

- Table 4 shows the frequency distribution of grades from the `IFC6503.A.2016` dataset.
  - Grade is the variable.

Frequency Distributions [1]

Table 4 : Grade frequencies of course IFC 6503 Class A 2016

Grade	Number of Students
A	18
B+	3
B	3
C+	6
C	7
D	0
E	5

- Table 4 shows the frequency distribution of grades from the `IFC6503.A.2016` dataset.
  - Grade is the variable.
  - Number of Students is the frequency column.

Frequency Distributions [1]

Table 4 : Grade frequencies of course IFC 6503 Class A 2016

Grade	Number of Students
A	18
B+	3
B	3
C+	6
C	7
D	0
E	5

- Table 4 shows the frequency distribution of grades from the `IFC6503.A.2016` dataset.
  - Grade is the variable.
  - Number of Students is the frequency column.
  - Each grade is a category.

Frequency Distributions [1]

Table 4 : Grade frequencies of course IFC 6503 Class A 2016

Grade	Number of Students
A	18
B+	3
B	3
C+	6
C	7
D	0
E	5

- Table 4 shows the frequency distribution of grades from the `IFC6503.A.2016` dataset.
  - Grade is the variable.
  - Number of Students is the frequency column.
  - Each grade is a category.
  - Each number in the frequency column is the frequency of the category left to it.

Frequency Distributions [1]

Table 5 : Frequency distributions of grade on course IFC 6503 Class A 2016 in tally marks

Grade	Frequency
A	𝍸 𝍸 𝍸 𝍷𝍷𝍷
B+	𝍷𝍷𝍷
B	𝍷𝍷𝍷
C+	𝍸 𝍷
C	𝍸 𝍷𝍷
D	
E	𝍸

Outline

Relative Frequency and Percentage Distributions

$$\text{Relative frequency of a category} = \frac{\text{Frequency of that category}}{\text{Sum of all frequencies}} \quad (1)$$

Relative Frequency and Percentage Distributions

$$\text{Relative frequency of a category} = \frac{\text{Frequency of that category}}{\text{Sum of all frequencies}} \quad (1)$$

$$\text{Percentage} = (\text{Relative frequency}) \cdot 100 \quad (2)$$

Relative Frequency and Percentage Distributions

Example

Table 6 : Relative frequency and percentage distributions of students grade in course IFC 6503 Class A 2016.

Grade	Relative Frequency	Percentage
A	18/42 = 0.43	42.86
B+	3/42 = 0.07	7.14
B	3/42 = 0.07	7.14
C+	6/42 = 0.14	14.29
C	7/42 = 0.17	16.67
D	0/42 = 0	0
E	5/42 = 0.12	11.9

Outline

Graphical Presentation of Qualitative Data [1]

- **Bar Graph**: a graph made of bars whose heights represent the *frequencies* of respective categories.

Graphical Presentation of Qualitative Data [1]

- **Bar Graph**: a graph made of bars whose heights represent the *frequencies* of respective categories.
- **Pie Chart**: a circle divided into portions that represent the relative frequencies or percentages of a population or a sample belonging to different categories.

- To create a bar graph using R, first prepare frequency distributions table by using `table()` function.

[8] Using the default graph. There are several advanced graphing function such as Lattice and ggplot2.

- To create a bar graph using R, first prepare frequency distributions table by using `table()` function.

```
grade.freq <- table(IFC6503.A.2016$Grade)
```

[8] Using the default graph. There are several advanced graphing function such as Lattice and ggplot2.

- To create a bar graph using R, first prepare frequency distributions table by using `table()` function.

```
grade.freq <- table(IFC6503.A.2016$Grade)
```

- The table was assigned to `grade.freq` variable. As usual, we can check the content by entering the variable name and press [Enter].

[8] Using the default graph. There are several advanced graphing function such as Lattice and ggplot2.

- To create a bar graph using R, first prepare frequency distributions table by using `table()` function.

```
grade.freq <- table(IFC6503.A.2016$Grade)
```

- The table was assigned to `grade.freq` variable. As usual, we can check the content by entering the variable name and press [Enter].

```
grade.freq

##
##  A  B B+  C C+  E
## 18  3  3  7  6  5
```

[8] Using the default graph. There are several advanced graphing function such as Lattice and ggplot2.

- To create a bar graph using R, first prepare frequency distributions table by using `table()` function.

```
grade.freq <- table(IFC6503.A.2016$Grade)
```

- The table was assigned to `grade.freq` variable. As usual, we can check the content by entering the variable name and press [Enter].

```
grade.freq

##
##  A  B B+  C C+  E
## 18  3  3  7  6  5
```

- To create standard[8] bar graph, we use the `barplot()` function.

[8]Using the default graph. There are several advanced graphing function such as Lattice and ggplot2.

- To create a bar graph using R, first prepare frequency distributions table by using `table()` function.

```
grade.freq <- table(IFC6503.A.2016$Grade)
```

- The table was assigned to `grade.freq` variable. As usual, we can check the content by entering the variable name and press [Enter].

```
grade.freq

##
##  A  B B+  C C+  E
## 18  3  3  7  6  5
```

- To create standard[8] bar graph, we use the `barplot()` function.

```
barplot(grade.freq)
```

[8]Using the default graph. There are several advanced graphing function such as Lattice and ggplot2.

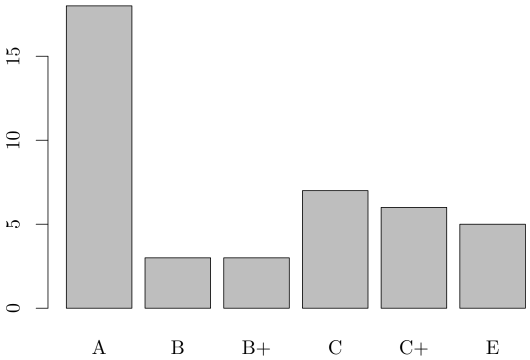

Figure 1 : Bar graph for frequency distributions of Table 5.

- Since coloring makes the graph more readable, it is better to add some colors to our graph.

[9] Search RColorBrewer for more options.

- Since coloring makes the graph more readable, it is better to add some colors to our graph.
- The simple[9] coloring can be achieved by using `col=` parameter.

[9] Search RColorBrewer for more options.

- Since coloring makes the graph more readable, it is better to add some colors to our graph.
- The simple[9] coloring can be achieved by using `col=` parameter.

```
col = c("red", "green", "blue")
```

[9]Search RColorBrewer for more options.

- Since coloring makes the graph more readable, it is better to add some colors to our graph.
- The simple[9] coloring can be achieved by using `col=` parameter.

```
col = c("red", "green", "blue")
```

- We need a color for each bar in the bar graph.

[9] Search RColorBrewer for more options.

- Since coloring makes the graph more readable, it is better to add some colors to our graph.
- The simple[9] coloring can be achieved by using `col=` parameter.

```
col = c("red", "green", "blue")
```

- We need a color for each bar in the bar graph.
- We can assign the color names to a vector of strings

[9]Search RColorBrewer for more options.

- Since coloring makes the graph more readable, it is better to add some colors to our graph.
- The simple[9] coloring can be achieved by using `col=` parameter.

```
col = c("red", "green", "blue")
```

- We need a color for each bar in the bar graph.
- We can assign the color names to a vector of strings

```
dist.freq.colors <- c("red", "yellow", "green", "violet", "pink",
    "orange", "cyan")
```

[9] Search RColorBrewer for more options.

Graphical Presentation of Qualitative Data

Bar Graph

- Since coloring makes the graph more readable, it is better to add some colors to our graph.
- The simple[9] coloring can be achieved by using `col=` parameter.

```
col = c("red", "green", "blue")
```

- We need a color for each bar in the bar graph.
- We can assign the color names to a vector of strings

```
dist.freq.colors <- c("red", "yellow", "green", "violet", "pink",
    "orange", "cyan")
```

- then use the `col=` parameter

```
barplot(grade.freq, col = dist.freq.colors)
```

[9]Search RColorBrewer for more options.

Graphical Presentation of Qualitative Data

Bar Graph

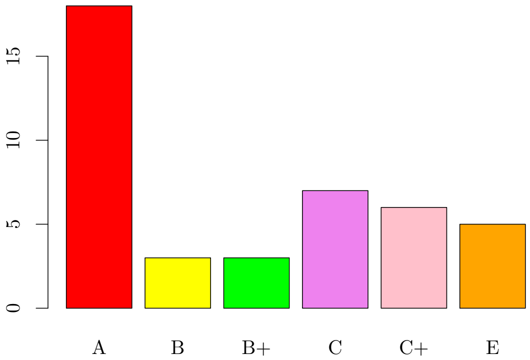

Figure 2 : Bar graph with colors.

- To create a pie chart, we need to prepare the relative frequency distributions table first.

- To create a pie chart, we need to prepare the relative frequency distributions table first.
- To have it, we use the frequency distributions table and apply mathematics operation on it to achieve the percentage.

- To create a pie chart, we need to prepare the relative frequency distributions table first.
- To have it, we use the frequency distributions table and apply mathematics operation on it to achieve the percentage.

```
grade.pct <- grade.freq/nrow(IFC6503.A.2016)
grade.pct
```

```
##
##     A     B    B+     C    C+     E
## 0.429 0.071 0.071 0.167 0.143 0.119
```

- To create a pie chart, we need to prepare the relative frequency distributions table first.
- To have it, we use the frequency distributions table and apply mathematics operation on it to achieve the percentage.

```
grade.pct <- grade.freq/nrow(IFC6503.A.2016)
grade.pct
```

```
##
##     A     B    B+     C    C+     E
## 0.429 0.071 0.071 0.167 0.143 0.119
```

- As you may already aware, the data was not sorted as usual.

- To sort the data as we need, modify the `levels` sequence[10].

- To sort the data as we need, modify the `levels` sequence[10].

[10] Levels and Factor are attributes in R. Read more by entering `?levels`, and `?factor`

- To sort the data as we need, modify the `levels` sequence[10].

```
IFC6503.A.2016$Grade <- factor(
       as.character(IFC6503.A.2016$Grade),
       levels = c("A", "B+", "B", "C+", "C", "D", "E")
       )
grade.freq <-  table(IFC6503.A.2016$Grade)
grade.pct <- grade.freq/nrow(IFC6503.A.2016)
grade.pct
```

```
##
##     A    B+     B    C+     C     D     E
## 0.429 0.071 0.071 0.143 0.167 0.000 0.119
```

[10] Levels and Factor are attributes in R. Read more by entering `?levels`, and `?factor`

- To sort the data as we need, modify the `levels` sequence[10].

```
IFC6503.A.2016$Grade <- factor(
        as.character(IFC6503.A.2016$Grade),
        levels = c("A", "B+", "B", "C+", "C", "D", "E")
        )
grade.freq <-  table(IFC6503.A.2016$Grade)
grade.pct <- grade.freq/nrow(IFC6503.A.2016)
grade.pct

##
##     A    B+     B    C+     C     D     E
## 0.429 0.071 0.071 0.143 0.167 0.000 0.119
```

- To create a pie chart, use the `pie()` function.

```
pie(grade.pct, col = dist.freq.colors)
```

[10] Levels and Factor are attributes in R. Read more by entering `?levels`, and `?factor`

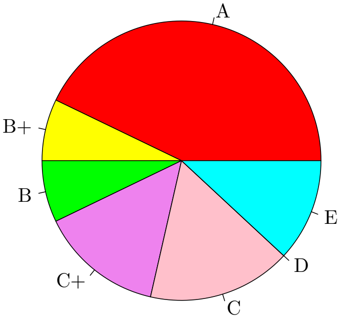

Figure 3 : Pie chart of relative frequency from `IFC6503.A.2016` dataset.

Outline

Outline

Frequency Distributions [1]

- Frequency distribution for quantitative data lists all the classes and the number of values that belong to each class.

Frequency Distributions [1]

- Frequency distribution for quantitative data lists all the classes and the number of values that belong to each class.
- Grouped data are the data that presented in the form of a frequency distribution.

Frequency Distributions [1]

- Frequency distribution for quantitative data lists all the classes and the number of values that belong to each class.
- Grouped data are the data that presented in the form of a frequency distribution.
- To construct frequency distribution table for quantitative data, we need three major decisions:

Frequency Distributions [1]

- Frequency distribution for quantitative data lists all the classes and the number of values that belong to each class.
- Grouped data are the data that presented in the form of a frequency distribution.
- To construct frequency distribution table for quantitative data, we need three major decisions:
  - Number of classes

Frequency Distributions [1]

- Frequency distribution for quantitative data lists all the classes and the number of values that belong to each class.
- Grouped data are the data that presented in the form of a frequency distribution.
- To construct frequency distribution table for quantitative data, we need three major decisions:
  - Number of classes
  - Class width

Frequency Distributions [1]

- Frequency distribution for quantitative data lists all the classes and the number of values that belong to each class.
- Grouped data are the data that presented in the form of a frequency distribution.
- To construct frequency distribution table for quantitative data, we need three major decisions:
  - Number of classes
  - Class width
  - Lower limit of the first class or the starting point.

To create the frequency distribution in R, we need several steps.

1. Put the data in a numeric vector[11].

To create the frequency distribution in R, we need several steps.

1. Put the data in a numeric vector[11].

[11] This step only to simplify further codes, hence can be omitted.

To create the frequency distribution in R, we need several steps.

1. Put the data in a numeric vector[11].

```
final.scores <- IFC6503.A.2016$Nilai_Akhir
```

[11] This step only to simplify further codes, hence can be omitted.

To create the frequency distribution in R, we need several steps.

1. Put the data in a numeric vector[11].

```
final.scores <- IFC6503.A.2016$Nilai_Akhir
```

2. Prepare another vector that contains the *breaks*.

[11] This step only to simplify further codes, hence can be omitted.

To create the frequency distribution in R, we need several steps.

1. Put the data in a numeric vector[11].

```
final.scores <- IFC6503.A.2016$Nilai_Akhir
```

2. Prepare another vector that contains the *breaks*.
3. We use the `seq()` function. The number passed to `by=` parameter represents *class width*.

[11] This step only to simplify further codes, hence can be omitted.

To create the frequency distribution in R, we need several steps.

1. Put the data in a numeric vector[11].

```
final.scores <- IFC6503.A.2016$Nilai_Akhir
```

2. Prepare another vector that contains the *breaks*.
3. We use the `seq()` function. The number passed to `by=` parameter represents *class width*.
4. Suppose we choose 10 as the class width, and as we know that the score span from 0 to 100 therefore:

```
breaks <- seq(0, 100, by = 10)
breaks

##  [1]   0  10  20  30  40  50  60  70  80  90 100
```

[11] This step only to simplify further codes, hence can be omitted.

5. Use the `cut()` function to divide the scores into several ranges as defined by `breaks`.

5. Use the `cut()` function to divide the scores into several ranges as defined by `breaks`.

```
final.scores.cut <- cut(final.scores, breaks, right = FALSE)
```

5. Use the `cut()` function to divide the scores into several ranges as defined by `breaks`.

```
final.scores.cut <- cut(final.scores, breaks, right = FALSE)
```

6. Use the `table()` function to construct the frequency table.

5. Use the `cut()` function to divide the scores into several ranges as defined by `breaks`.

```
final.scores.cut <- cut(final.scores, breaks, right = FALSE)
```

6. Use the `table()` function to construct the frequency table.

```
final.scores.freq <- table(final.scores.cut)
```

5. Use the `cut()` function to divide the scores into several ranges as defined by `breaks`.

```
final.scores.cut <- cut(final.scores, breaks, right = FALSE)
```

6. Use the `table()` function to construct the frequency table.

```
final.scores.freq <- table(final.scores.cut)
```

7. The result is

5. Use the `cut()` function to divide the scores into several ranges as defined by `breaks`.

```
final.scores.cut <- cut(final.scores, breaks, right = FALSE)
```

6. Use the `table()` function to construct the frequency table.

```
final.scores.freq <- table(final.scores.cut)
```

7. The result is

```
final.scores.freq

## final.scores.cut
##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##        4        1        0        0        3        4
##  [60,70)  [70,80)  [80,90) [90,100)
##        6        6       12        6
```

5. As described earlier, we can have the relative frequency table by dividing the frequency with number of data.

5. As described earlier, we can have the relative frequency table by dividing the frequency with number of data.

```
final.scores.relfreq <- final.scores.freq/length(final.scores)
final.scores.relfreq
```

```
## final.scores.cut
##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##    0.095    0.024    0.000    0.000    0.071    0.095
##  [60,70)  [70,80)  [80,90) [90,100)
##    0.143    0.143    0.286    0.143
```

5. As described earlier, we can have the relative frequency table by dividing the frequency with number of data.

```
final.scores.relfreq <- final.scores.freq/length(final.scores)
final.scores.relfreq
```

```
## final.scores.cut
##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##    0.095    0.024    0.000    0.000    0.071    0.095
##  [60,70)  [70,80)  [80,90) [90,100)
##    0.143    0.143    0.286    0.143
```

6. Then the percentage

5. As described earlier, we can have the relative frequency table by dividing the frequency with number of data.

```
final.scores.relfreq <- final.scores.freq/length(final.scores)
final.scores.relfreq

## final.scores.cut
##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##    0.095    0.024    0.000    0.000    0.071    0.095
##  [60,70)  [70,80)  [80,90) [90,100)
##    0.143    0.143    0.286    0.143
```

6. Then the percentage

```
final.scores.pct <- final.scores.relfreq * 100
final.scores.pct

## final.scores.cut
##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##      9.5      2.4      0.0      0.0      7.1      9.5
##  [60,70)  [70,80)  [80,90) [90,100)
##     14.3     14.3     28.6     14.3
```

Outline

Graphing Grouped Data [1]

- **Histogram**: a graph in which classes are marked on the horizontal axis and the frequencies, relative frequencies, or percentages are marked on the vertical axis.

Graphing Grouped Data [1]

- **Histogram**: a graph in which classes are marked on the horizontal axis and the frequencies, relative frequencies, or percentages are marked on the vertical axis.
- Depending on the data, histogram can show:

Graphing Grouped Data [1]

- **Histogram**: a graph in which classes are marked on the horizontal axis and the frequencies, relative frequencies, or percentages are marked on the vertical axis.
- Depending on the data, histogram can show:
  - Frequency

Graphing Grouped Data [1]

- **Histogram**: a graph in which classes are marked on the horizontal axis and the frequencies, relative frequencies, or percentages are marked on the vertical axis.
- Depending on the data, histogram can show:
  - Frequency
  - Relative frequency

Graphing Grouped Data [1]

- **Histogram**: a graph in which classes are marked on the horizontal axis and the frequencies, relative frequencies, or percentages are marked on the vertical axis.
- Depending on the data, histogram can show:
  - Frequency
  - Relative frequency
  - Percentage

Graphing Grouped Data [1]

- **Histogram**: a graph in which classes are marked on the horizontal axis and the frequencies, relative frequencies, or percentages are marked on the vertical axis.
- Depending on the data, histogram can show:
  - Frequency
  - Relative frequency
  - Percentage
- **Polygon**: a graph formed by joining the midpoints of the tops of successive bars in a histogram with straight lines.

Graphing Grouped Data [1]

- **Histogram**: a graph in which classes are marked on the horizontal axis and the frequencies, relative frequencies, or percentages are marked on the vertical axis.
- Depending on the data, histogram can show:
  - Frequency
  - Relative frequency
  - Percentage
- **Polygon**: a graph formed by joining the midpoints of the tops of successive bars in a histogram with straight lines.
- In this course we only discuss histogram.

- To create a histogram we use the `hist()` function.

- To create a histogram we use the `hist()` function.

```
hist(final.scores)
```

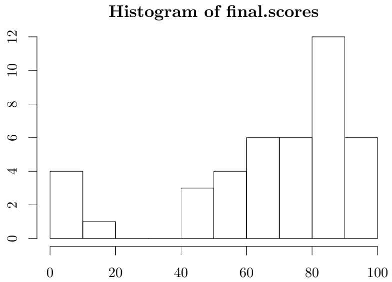

Figure 4 : Default histogram in R. The data was taken from the `IFC6503.A.2016` dataset.

- Figure 4 shows the default histogram in R, a frequency histogram.

- Figure 4 shows the default histogram in R, a frequency histogram.
- We can modify the histogram by passing some parameters.

- Figure 4 shows the default histogram in R, a frequency histogram.
- We can modify the histogram by passing some parameters.
- The following code will suppress the title, change the label of the x-axis, and coloring the bars.

```r
hist(
        final.scores,
        main=NULL,
        xlab='Final Scores',
        color='violet'
)
```

- Figure 4 shows the default histogram in R, a frequency histogram.
- We can modify the histogram by passing some parameters.
- The following code will suppress the title, change the label of the x-axis, and coloring the bars.

```
hist(
        final.scores,
        main=NULL,
        xlab='Final Scores',
        color='violet'
)
```

- The result is shown in Figure 5.

Graphing Grouped Data

Histogram in R

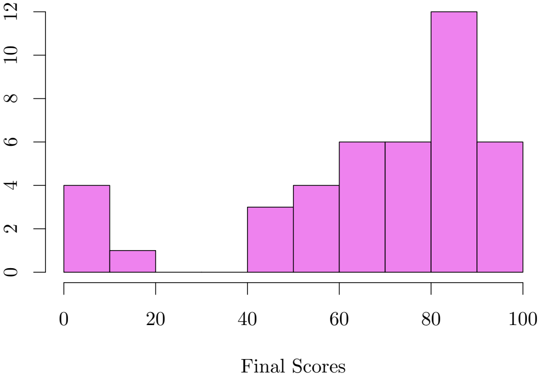

Figure 5 : Final scores histogram with modified options.

When `plot=FALSE`, `hist()` will print the computed histogram.

When `plot=FALSE`, `hist()` will print the computed histogram.

```
hist(final.scores, plot = F)

## $breaks
##  [1]   0  10  20  30  40  50  60  70  80  90 100
##
## $counts
##  [1]  4  1  0  0  3  4  6  6 12  6
##
## $density
##  [1] 0.0095 0.0024 0.0000 0.0000 0.0071 0.0095 0.0143 0.0143
##  [9] 0.0286 0.0143
##
## $mids
##  [1]  5 15 25 35 45 55 65 75 85 95
##
## $xname
## [1] "final.scores"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
```

When we pass `freq = F`, `hist()` will produce a density histogram.

When we pass `freq = F`, `hist()` will produce a density histogram.

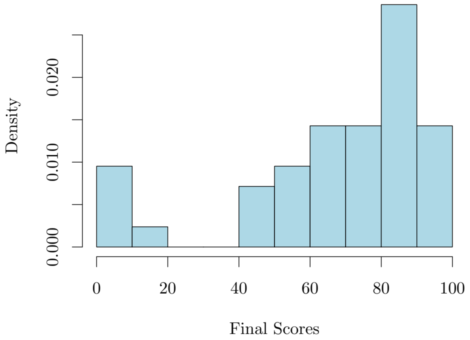

Figure 6 : Final scores density histogram.

Outline

Outline

Cumulative Frequency Distributions

- Cumulative frequency distribution gives the total number of values that fall below the upper boundary of each class [1].

Cumulative Frequency Distributions

- Cumulative frequency distribution gives the total number of values that fall below the upper boundary of each class [1].
- To produce cumulative frequency distribution table in R we apply the `cumsum()` function to the frequency distribution table.

Cumulative Frequency Distributions

- Cumulative frequency distribution gives the total number of values that fall below the upper boundary of each class [1].
- To produce cumulative frequency distribution table in R we apply the `cumsum()` function to the frequency distribution table.
- Recall our examples, the cumulative frequency distribution for the students score in course IFC 6503 can produced by

Cumulative Frequency Distributions

- Cumulative frequency distribution gives the total number of values that fall below the upper boundary of each class [1].
- To produce cumulative frequency distribution table in R we apply the `cumsum()` function to the frequency distribution table.
- Recall our examples, the cumulative frequency distribution for the students score in course IFC 6503 can produced by

```
cumsum(final.scores.freq)

##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##        4        5        5        5        8       12
##  [60,70)  [70,80)  [80,90) [90,100)
##       18       24       36       42
```

Cumulative Frequency Distributions [1]

- As usual, we can assign the function output to variable for further processing

Cumulative Frequency Distributions [1]

- As usual, we can assign the function output to variable for further processing

```r
final.scores.cumsum <- cumsum(final.scores.freq)
```

Cumulative Frequency Distributions [1]

- As usual, we can assign the function output to variable for further processing

```
final.scores.cumsum <- cumsum(final.scores.freq)
```

- To get the result in column format, we apply the `cbind()` function.

Cumulative Frequency Distributions [1]

- As usual, we can assign the function output to variable for further processing

```r
final.scores.cumsum <- cumsum(final.scores.freq)
```

- To get the result in column format, we apply the `cbind()` function.

```r
cbind(final.scores.cumsum)
```

```
##          final.scores.cumsum
## [0,10)                     4
## [10,20)                    5
## [20,30)                    5
## [30,40)                    5
## [40,50)                    8
## [50,60)                   12
## [60,70)                   18
## [70,80)                   24
## [80,90)                   36
## [90,100)                  42
```

Outline

Cumulative Frequency Graph

- Ogive is a plot of cumulative frequencies [1].

Cumulative Frequency Graph

- Ogive is a plot of cumulative frequencies [1].
- Ogive is drawn by joining with straight lines the dots marked above the upper boundaries of classes at heights equal to the cumulative frequencies of respective class [1].

Cumulative Frequency Graph

- Ogive is a plot of cumulative frequencies [1].
- Ogive is drawn by joining with straight lines the dots marked above the upper boundaries of classes at heights equal to the cumulative frequencies of respective class [1].
- To create an ogive in R, there are several steps needed:

Cumulative Frequency Graph

- Ogive is a plot of cumulative frequencies [1].
- Ogive is drawn by joining with straight lines the dots marked above the upper boundaries of classes at heights equal to the cumulative frequencies of respective class [1].
- To create an ogive in R, there are several steps needed:
    1. Prepare the cumulative frequency table and the breaks.

Cumulative Frequency Graph

- Ogive is a plot of cumulative frequencies [1].
- Ogive is drawn by joining with straight lines the dots marked above the upper boundaries of classes at heights equal to the cumulative frequencies of respective class [1].
- To create an ogive in R, there are several steps needed:
  1. Prepare the cumulative frequency table and the breaks.
  2. Add a starting 0 element to the cumulative frequency table.

Cumulative Frequency Graph

- Ogive is a plot of cumulative frequencies [1].
- Ogive is drawn by joining with straight lines the dots marked above the upper boundaries of classes at heights equal to the cumulative frequencies of respective class [1].
- To create an ogive in R, there are several steps needed:
  1. Prepare the cumulative frequency table and the breaks.
  2. Add a starting 0 element to the cumulative frequency table.
  3. Plot the points by matching the breaks and the cumulative frequency table that has been added with 0.

Cumulative Frequency Graph

- Ogive is a plot of cumulative frequencies [1].
- Ogive is drawn by joining with straight lines the dots marked above the upper boundaries of classes at heights equal to the cumulative frequencies of respective class [1].
- To create an ogive in R, there are several steps needed:
    1. Prepare the cumulative frequency table and the breaks.
    2. Add a starting 0 element to the cumulative frequency table.
    3. Plot the points by matching the breaks and the cumulative frequency table that has been added with 0.
    4. Plot the lines above the previous plot.

Cumulative Frequency Graph

Ogive in R

1. Prepare the cumulative frequency graph: here we reuse the `final.scores.cumsum`.

```
final.scores.cumsum
```

```
##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##        4        5        5        5        8       12
##  [60,70)  [70,80)  [80,90) [90,100)
##       18       24       36       42
```

Cumulative Frequency Graph

Ogive in R

1. Prepare the cumulative frequency graph: here we reuse the `final.scores.cumsum`.

```
final.scores.cumsum

##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##        4        5        5        5        8       12
##  [60,70)  [70,80)  [80,90) [90,100)
##       18       24       36       42
```

2. Prepare the breaks: we already have a variable that contains the breaks. Reuse it.

```
breaks

##  [1]   0  10  20  30  40  50  60  70  80  90 100
```

Cumulative Frequency Graph

Ogive in R

1. Prepare the cumulative frequency graph: here we reuse the `final.scores.cumsum`.

```
final.scores.cumsum

##   [0,10)  [10,20)  [20,30)  [30,40)  [40,50)  [50,60)
##        4        5        5        5        8       12
##  [60,70)  [70,80)  [80,90) [90,100)
##       18       24       36       42
```

2. Prepare the breaks: we already have a variable that contains the breaks. Reuse it.

```
breaks

##  [1]   0  10  20  30  40  50  60  70  80  90 100
```

3. Add 0 to the beginning of the `final.scores.cumsum`.

```
cumfreq0 <- c(0, final.scores.cumsum)
cumfreq0

##            [0,10)  [10,20)  [20,30)  [30,40)  [40,50)
##        0        4        5        5        5        8
##  [50,60)  [60,70)  [70,80)  [80,90) [90,100)
##       12       18       24       36       42
```

4. Plot the points

```
plot(breaks, cumfreq0, xlab = "Scores", ylab = "Cumulative Frequency",
    xaxs = "i", yaxs = "i")
```

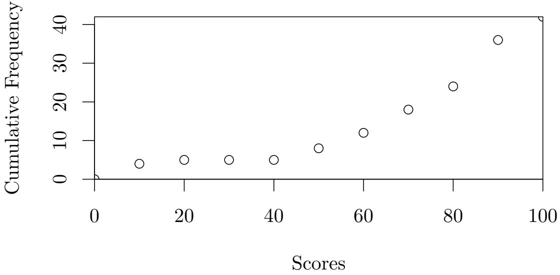

Figure 7 : Plotting the points.

5. Add the line

```
lines(breaks, cumfreq0)
```

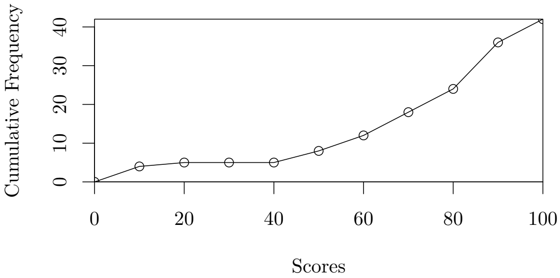

Figure 8 : Adding lines.

Outline

Stem-and-Leaf Displays

- Steam-and-leaf display is another technique to present quantitative data in condensed form [1].

Stem-and-Leaf Displays

- Steam-and-leaf display is another technique to present quantitative data in condensed form [1].
- In a stem-and-leaf display, each value is divided into two portions–a stem and a leaf [1].

Stem-and-Leaf Displays

- Steam-and-leaf display is another technique to present quantitative data in condensed form [1].
- In a stem-and-leaf display, each value is divided into two portions–a stem and a leaf [1].
- The leaves for each stem are shown separately in a display [1].

Stem-and-Leaf Displays

- Steam-and-leaf display is another technique to present quantitative data in condensed form [1].
- In a stem-and-leaf display, each value is divided into two portions–a stem and a leaf [1].
- The leaves for each stem are shown separately in a display [1].
- To create a stem-and-leaf display in R, we use the `stem()` function in R.

Stem-and-Leaf Displays

- Steam-and-leaf display is another technique to present quantitative data in condensed form [1].
- In a stem-and-leaf display, each value is divided into two portions–a stem and a leaf [1].
- The leaves for each stem are shown separately in a display [1].
- To create a stem-and-leaf display in R, we use the `stem()` function in R.
- The passed parameter is a numeric vector.

Stem-and-Leaf Displays

- Steam-and-leaf display is another technique to present quantitative data in condensed form [1].
- In a stem-and-leaf display, each value is divided into two portions–a stem and a leaf [1].
- The leaves for each stem are shown separately in a display [1].
- To create a stem-and-leaf display in R, we use the `stem()` function in R.
- The passed parameter is a numeric vector.
- In our case, since we already have the `final.scores` vector:

```
stem(final.scores)
```

Stem-and-Leaf Displays

- Steam-and-leaf display is another technique to present quantitative data in condensed form [1].
- In a stem-and-leaf display, each value is divided into two portions–a stem and a leaf [1].
- The leaves for each stem are shown separately in a display [1].
- To create a stem-and-leaf display in R, we use the `stem()` function in R.
- The passed parameter is a numeric vector.
- In our case, since we already have the `final.scores` vector:

```r
stem(final.scores)
```

- The result is shown on the next slide.

Stem-and-Leaf Displays

```
stem(final.scores)

##
##   The decimal point is 1 digit(s) to the right of the |
##
##   0 | 0055
##   1 | 8
##   2 |
##   3 |
##   4 | 289
##   5 | 1377
##   6 | 136689
##   7 | 122688
##   8 | 01333456778 9
##   9 | 123477
```

Outline

Exercise 2

- Load the `IFC6510.csv`.

Exercise 2

- Load the `IFC6510.csv`.
- Construct qualitative frequency distribution table from the Grades data.

Exercise 2

- Load the `IFC6510.csv`.
- Construct qualitative frequency distribution table from the Grades data.
- Construct qualitative relative frequency and percentage distribution tables from the Grades data.

Exercise 2

- Load the `IFC6510.csv`.
- Construct qualitative frequency distribution table from the Grades data.
- Construct qualitative relative frequency and percentage distribution tables from the Grades data.
- Build a bar graph and a pie chart for the Grades data.

Exercise 2

- Load the `IFC6510.csv`.
- Construct qualitative frequency distribution table from the Grades data.
- Construct qualitative relative frequency and percentage distribution tables from the Grades data.
- Build a bar graph and a pie chart for the Grades data.
- Construct quantitative frequency distribution table from the Grades data.

Exercise 2

- Load the `IFC6510.csv`.
- Construct qualitative frequency distribution table from the Grades data.
- Construct qualitative relative frequency and percentage distribution tables from the Grades data.
- Build a bar graph and a pie chart for the Grades data.
- Construct quantitative frequency distribution table from the Grades data.
- Construct quantitative relative frequency and percentage distribution tables from the Grades data.

Exercise 2

- Load the `IFC6510.csv`.
- Construct qualitative frequency distribution table from the Grades data.
- Construct qualitative relative frequency and percentage distribution tables from the Grades data.
- Build a bar graph and a pie chart for the Grades data.
- Construct quantitative frequency distribution table from the Grades data.
- Construct quantitative relative frequency and percentage distribution tables from the Grades data.
- Build a histogram for the `Nilai_Akhir` data.

Exercise 2

- Load the `IFC6510.csv`.
- Construct qualitative frequency distribution table from the Grades data.
- Construct qualitative relative frequency and percentage distribution tables from the Grades data.
- Build a bar graph and a pie chart for the Grades data.
- Construct quantitative frequency distribution table from the Grades data.
- Construct quantitative relative frequency and percentage distribution tables from the Grades data.
- Build a histogram for the `Nilai_Akhir` data.
- Build a cumulative frequency distribution table and graph for the `Nilai_Akhir` data.

Exercise 2

- While keeping the `IFC6510` dataset, load the `IFC6503.A.2016` dataset.

Exercise 2

- While keeping the `IFC6510` dataset, load the `IFC6503.A.2016` dataset.
- Build a single bar chart for the Grade data from both dataset.

Exercise 2

- While keeping the `IFC6510` dataset, load the `IFC6503.A.2016` dataset.
- Build a single bar chart for the Grade data from both dataset.
- Build a single histogram for the scores data from both dataset.

Next...

- Descriptive Statistics

Next...

- Descriptive Statistics
- Random Numbers

Next...

- Descriptive Statistics
- Random Numbers
- Preparation, read:

Next...

- Descriptive Statistics
- Random Numbers
- Preparation, read:
  - Chapters 3-8 from [1].

Next...

- Descriptive Statistics
- Random Numbers
- Preparation, read:
  - Chapters 3-8 from [1].
  - Chapters 2 and 3 from [2].

Outline

References I

[1] P. S. Mann, Introductory Statistics, 7th ed. NJ: John Wiley & Sons, Inc., 2010.

[2] D. Diez, C. Barr, and M. Çetinkaya-Rundel, OpenIntro Statistics. OpenIntro, Incorporated, 2015.

[3] W. Venables, D. Smith, and R Core Team. (2015, Dec.) An Introduction to R, Notes on R: A Programming Environment for Data Analysis and Graphics. Accessed on 20 February 2016. [Online]. Available: https://cran.r-project.org/doc/manuals/r-release/R-intro.html

[4] R. I. Kabacoff, R in Action: Data Analysis and Graphics with R, 2nd ed. NY: Manning Publications Co., 2015.