

DATA SCIENCE AND VISUALIZATION

Course Code	21CS644	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	3:0:0:0	SEE Marks	50
Total Hours of Pedagogy	40	Total Marks	100
Credits	03	Exam Hours	03

Overfitting

- Sometimes the [machine learning](#) model performs well with the training data but does not perform well with the test data. It means the model is not able to predict the output when deals with unseen data by introducing noise in the output, and hence the model is called overfitted.

Overfitting

- **Overfitting** in model development refers to the phenomenon where a model learns not only the *underlying patterns in the training data but also captures noise, random fluctuations, or outliers that are specific to the training dataset*. This results in a model that performs *extremely well on the training data* but fails to generalize effectively to new, unseen data.
- When a model is overfitted, **it may exhibit poor performance** when applied to new data that was not part of the training dataset.
- Overfitted models tend to **be too complex**, capturing spurious relationships and details that are not reflective of the true underlying patterns in the data.
- Overfitting can lead to **misleading conclusions** and **inaccurate predictions** when deployed in real-world scenarios.

Strategies to Mitigate Overfitting and Improve Model Generalizability:

- **Simplify the Model:**
- **Regularization:**
- **Cross-Validation:**
- **Feature Selection**
- **Early Stopping**

Regularization

- Regularization is one of the most important concepts of machine learning. It is a technique to prevent the model from overfitting by **adding extra information** to it.

Library Assignment #1 [on or before 15/5/2024]

Student Details

- Name:
- USN:
- Sem:
- Date and Time Of Library Visit:
- Reference Details:

Questions:

1. Explain how Cross validation and regularization are used to avoid overfitting in Data Science With Examples.
2. Discuss different types of Regularization (like **L1/L2/Elastic Net/Dropout**) in Data Science with examples.

Module1 : Introduction to Data Science and Statistical Inference Needed:

- 1. Introduction** : What is Data Science? Big Data and Data Science hype – and getting past the hype, Why now? Datafication, Current landscape of perspectives, Skill sets.
- 2. Needed Statistical Inference**: Populations and samples, Statistical modelling, probability distributions, fitting a model.

Data Scientist Profile

The key elements that define the Data Scientist Profile are:

1. Computer science skills
2. Mathematical skills
3. Statistical skills
4. Machine learning skills
5. Domain expertise (subject matter knowledge)
6. Communication and presentation skills
7. Data visualization skills

What is Data Science?

- **Data science is an interdisciplinary** field that involves using scientific methods, processes, algorithms, and systems to extract knowledge and insights from structured and unstructured data. It combines techniques and theories from **statistics, mathematics, computer science, AI, information science, and domain-specific fields like business or healthcare** to analyze complex data sets.
- **The main goal of data science** is to turn raw data into actionable insights, predictions, and recommendations.

Various stages of data processing and analysis

1. **Data Collection:** Gathering data from various sources such as databases, APIs, sensors, or web scraping.
2. **Data Cleaning:** Preprocessing and preparing the data by handling missing values, removing duplicates, and transforming data into a suitable format for analysis.
3. **Exploratory Data Analysis (EDA):** Understanding the data through statistical summaries, visualizations, and identifying patterns or anomalies.
4. **Feature Engineering:** Creating new features or variables that can enhance the predictive power of machine learning models.
5. **Machine Learning:** Applying statistical models and algorithms to build predictive models and make data-driven decisions. This includes supervised learning (classification, regression), unsupervised learning (clustering, dimensionality reduction), and other techniques.
6. **Data Visualization:** Presenting findings and insights through charts, graphs, dashboards, and interactive visualizations.
7. **Deployment and Monitoring:** Integrating data science solutions into production systems and continuously monitoring model performance.

Model Questions

1. What is Data Science? **Explain the Venn Diagram** of data Science.
2. Explain in detail about BigData and Data Science Hype. Why Hype? How to get past this hype?
3. Explain the Data Science Profile. Explain the work of the Data Scientist in academia and industry.
4. “The Data Scientists play a major role in Academics and industry”. Justify.
5. Explain the concept of Datafication with example. Why it is required in the current scenario?
6. Discuss in detail about the current landscape of perspectives and skill sets.

Model Questions

7. There are varieties of types of data that a Data Scientist has to deal”, explain the statement with examples.
8. Explain the following concepts involved in building models:
 - Statistical modeling.
 - Probability distributions
 - Fitting and overfitting
9. What is Statistical Thinking? Discuss in details statistical inferences and also explain about population and Samples
10. What is Statistical Modelling? How do you build a model?
11. Explain in detail about probability distribution with examples and fitting a model. Write about over fitting.
12. Explain the following concepts with examples
 - Statistical inference
 - Population
 - Samples
 - Types of Data

End of Module1

Module2: Syllabus

- **Exploratory Data Analysis and the Data Science Process:** Basic tools (plots, graphs and summary statistics) of EDA, Philosophy of EDA, The Data Science Process.
- **Case Study:** Real Direct (online real estate firm).
- **Three Basic Machine Learning Algorithms:** Linear Regression, k-Nearest Neighbors (k- NN), k-means.

“Exploratory data analysis” is *an attitude, a state of flexibility, a willingness* to look for those things that we believe **are not there, as well as those we **believe to be there**.**

— John Tukey

What is Exploratory Data Analysis?

- EDA involves using basic tools like **plots, graphs, and summary statistics to systematically analyze data.**
- It includes:
 - Plotting distributions (e.g., box plots)
 - Examining time series data
 - Transforming variables
 - Exploring pairwise relationships (scatterplot matrices)
 - Computing summary statistics (mean, min, max, quartiles)
 - Identifying outliers

Define Exploratory Data Analysis?

- **Exploratory Data Analysis (EDA)** is a data analysis approach focused on summarizing the main characteristics of a dataset using **visual methods like plots and graphs and statistical summaries**.
- **Its objectives include**
 - Gaining insights,
 - Understanding data structure,
 - Detecting patterns and anomalies, and
 - Formulating hypotheses for further analysis.

Plots and Graphs

- Plots refer specifically to visual representations of data using graphical elements such as points, lines, bars, or other symbols to display relationships, distributions, or trends within the data.
- **Examples:** Scatter plots, line plots, bar plots, box plots, histograms, pie charts, heatmaps, etc.
- **Graphs**, in a broader sense, can refer to any visual representation of data or mathematical relationships. In some contexts, graphs specifically refer to mathematical structures represented by nodes (vertices) and edges (connections) that illustrate relationships between entities.
- **Examples:** Line graphs, network graphs (nodes and edges), flowcharts, tree diagrams, Venn diagrams, etc.

Statistical Summary

- A **statistical summary** is a concise description or summary of key characteristics or features of a dataset or a sample of data. It provides numerical measures that describe the central tendency, variability, and distribution of the data.
- Common statistical summaries include:
 1. **Measures of Central Tendency:**
 2. **Measures of Variability**
 3. **Summary of Distribution**

Measures of Central Tendency:

- **Mean:** Average value of the data points.
- **Median:** Middle value of the dataset when arranged in order.
- **Mode:** Most frequently occurring value(s) in the dataset.

Measures of Variability:

- **Range:** Difference between the maximum and minimum values.
- **Variance:** Average of the squared differences from the mean.
- **Standard Deviation:** Square root of the variance, indicating the spread of data around the mean.

Summary of Distribution:

- **Quartiles:** Values that divide the dataset into four equal parts.
- **Interquartile Range (IQR):** Range of values between the **first and third** quartiles.
- **Skewness and Kurtosis:** Measures of **asymmetry** and **tail heaviness** of the distribution.

Other Summary Statistics:

- **Sum:** Total sum of all data points.
- **Count:** Number of observations in the dataset.
- **Percentiles:** Values below which a certain percentage of data falls.

Mindset of EDA

- EDA is more than just tools—it's a mindset.
- Focuses on developing a deep relationship with the data.
- **Goal: Gain intuition** and understand the data's shape.
- Connect this understanding to the underlying data generation process.

Personal Exploration

- EDA is a personal exploration between the analyst and the data.
- No need to prove anything to others at this stage.
- It's about understanding the data without external validation.

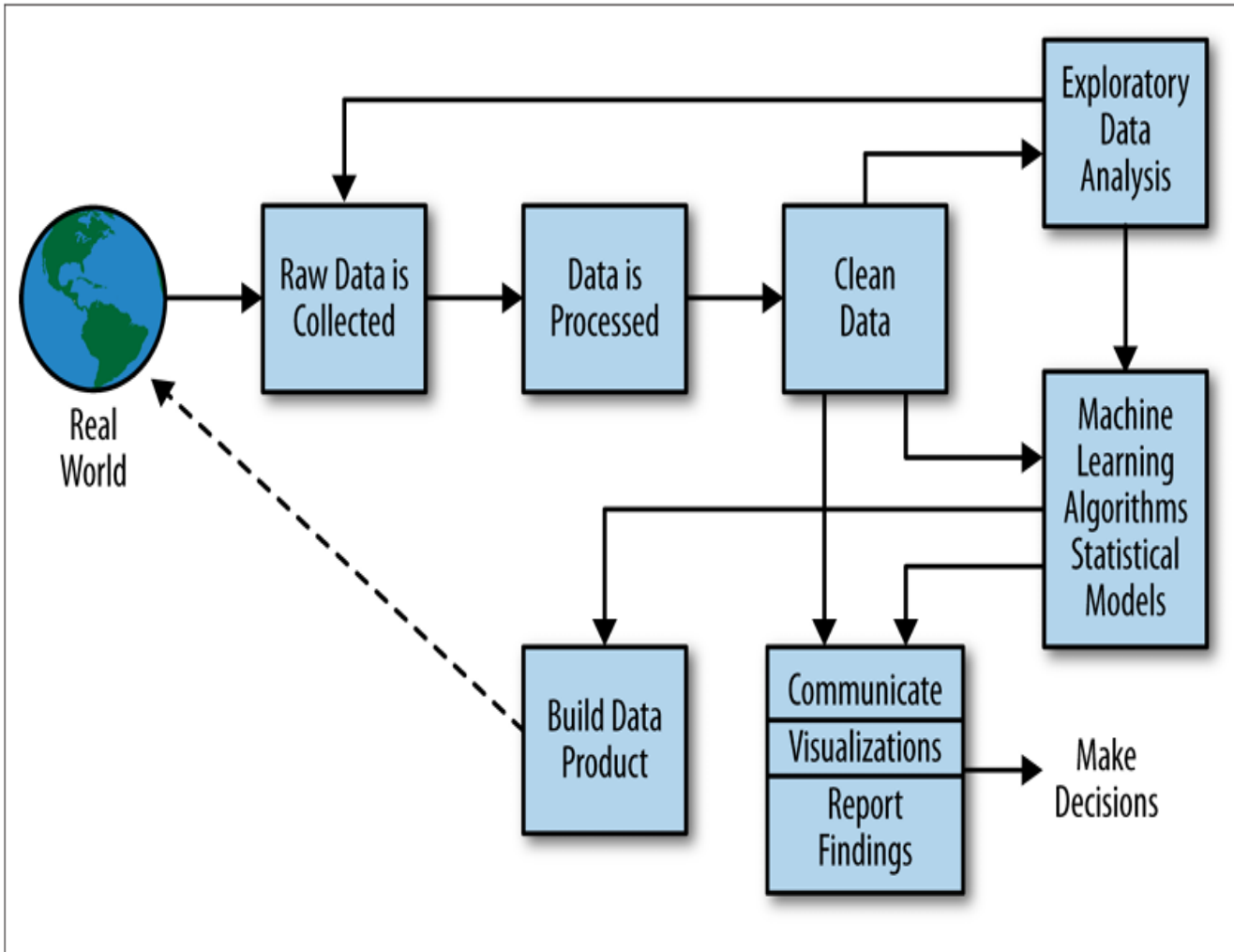
Philosophy of Exploratory Data Analysis

- **Quote:** *"Long before worrying about how to convince others, you first have to understand what's happening yourself."* - Andrew Gelman
- **Key Points:**
- Exploratory Data Analysis (EDA) is crucial for understanding the data before diving into advanced analysis.
- EDA helps **gain intuition, compare distributions, ensure data sanity, identify missing data, and detect outliers.**
- In the **context of log-generated data, EDA plays a vital role in debugging and ensuring accurate patterns.**

The Importance of EDA

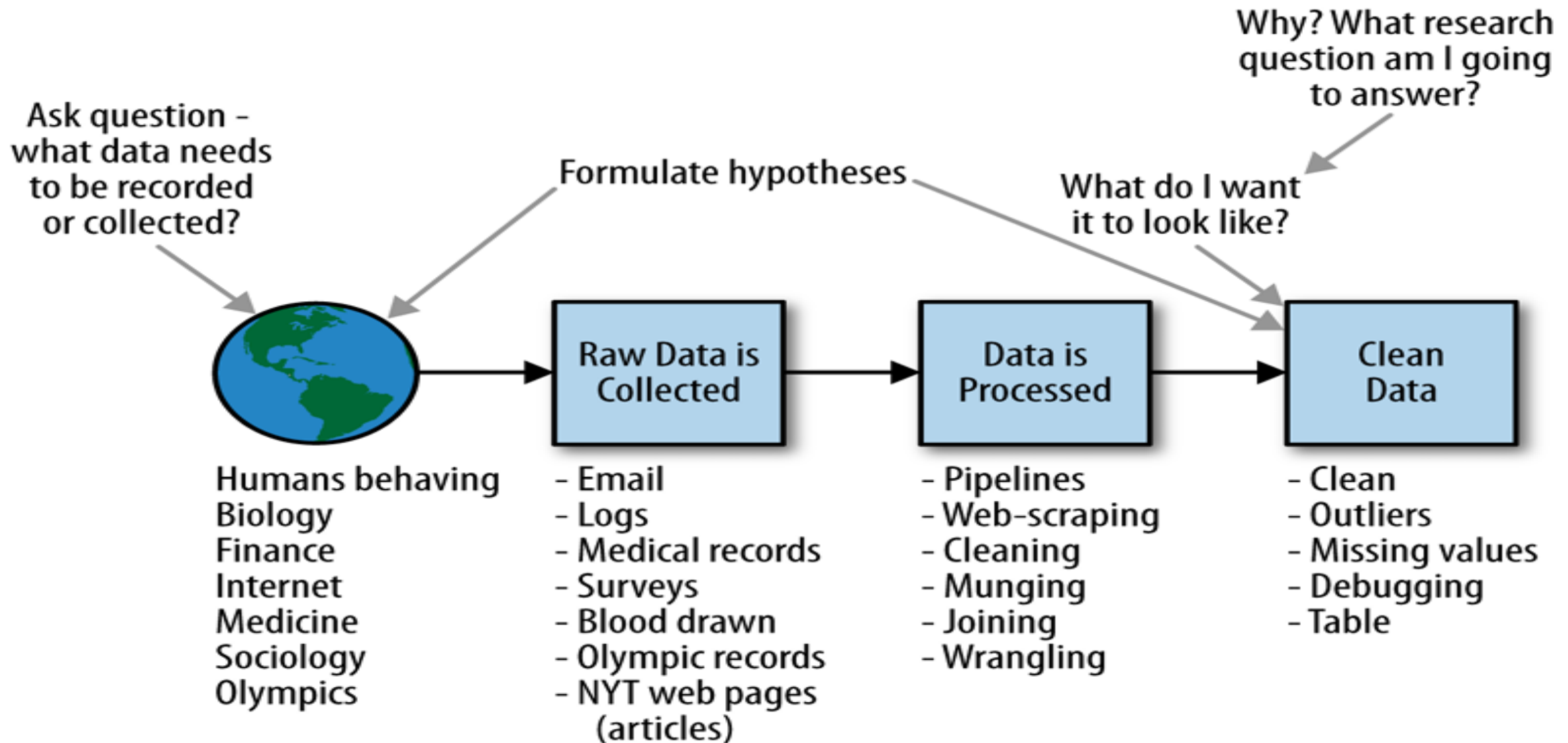
- EDA differs from data visualization used for **communicating findings** later in the analysis process.
- EDA focuses on **personal understanding rather** than external communication.
- It **empowers algorithm** development by informing decisions on metrics like "popularity" based on data behavior.
- EDA ensures a **more robust and informed** approach to data analysis.

The Data Science Process



1. Real World
2. Raw Data is Collected
3. Data is Processed
4. Clean Data
5. Exploratory Data Analysis
6. Machine Learning Algorithms/Statistical Models
7. Communicate Visualizations/Report Findings
8. Build Data Product
9. Make Decisions

A Data Scientist's Role in This Process



Connection to the Scientific Method

We can think of the data science process as an extension of or variation of the scientific method:

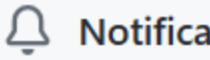
- 1. Ask a question.**
- 2. Do background research.**
- 3. Construct a hypothesis.**
- 4. Test your hypothesis by doing an experiment.**
- 5. Analyze your data and draw a conclusion.**
- 6. Communicate your results**

Exploratory Data Analysis Example

- There are 31 datasets named **nyt1.csv**, **nyt2.csv**, ..., **nyt31.csv**, which you can find here:
- **https://github.com/oreillymedia/doing_data_science**



oreillymedia / doing_data_science Public



<> Code

Issues 6

Pull requests 2

Actions

Projects

Wiki

Security

Insights

master ▾



1 Branch 0 Tags

Go to file






<> Code ▾

 kristenORM Update README.md

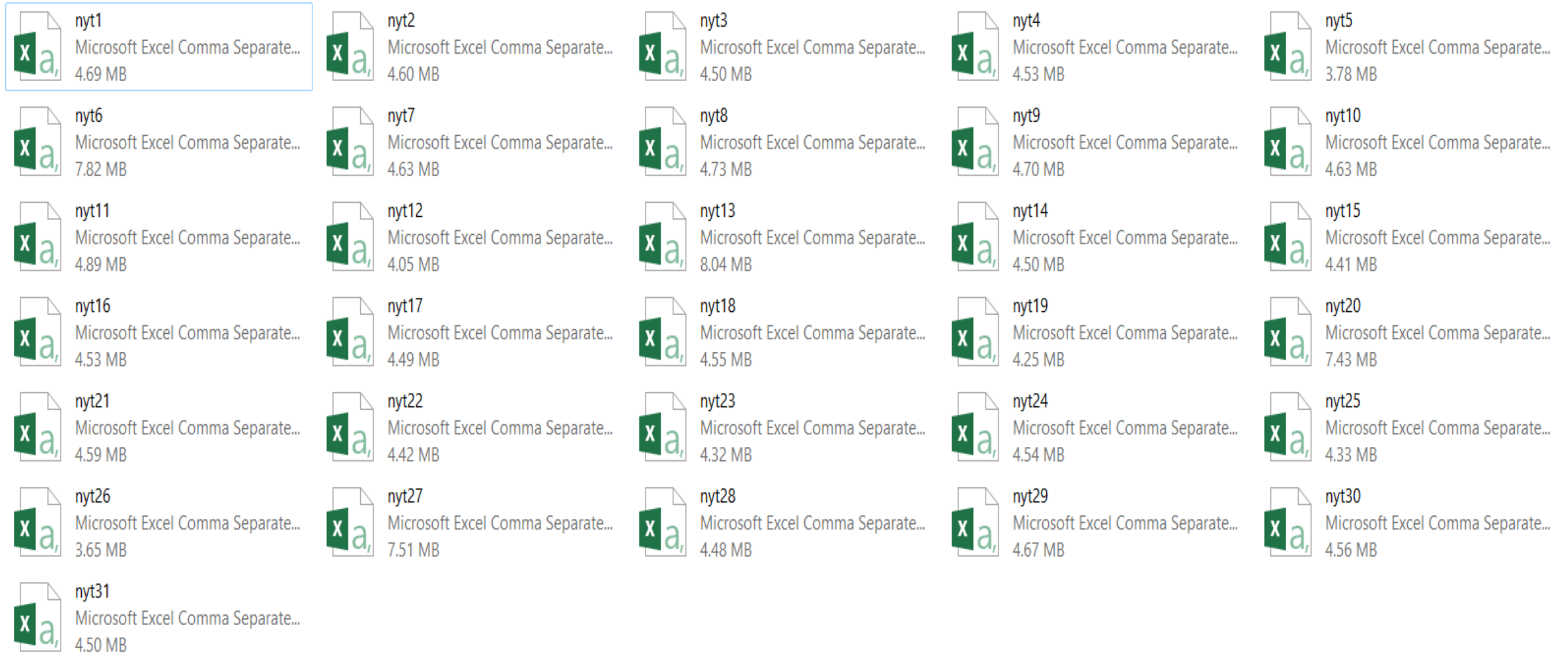
609d9dd · 9 years ago 5 Commits

 README.md	Update README.md	9 years ago
 dds_datasets.zip	adding zip file	11 years ago

 README

 dds_ch5_binary-class-dataset	11-05-2024 07:29	Text Document	2,873 KB
 .DS_Store	11-05-2024 07:29	DS_STORE File	7 KB
 dds_ch2_rollingsales	11-05-2024 07:29	File folder	
 dds_ch2_nyt 	11-05-2024 07:29	File folder	

31 CSV Files: nyt1.csv, nyt2.csv,...,nyt31.csv



Download and Install R: <https://www.r-project.org/>



[\[Home\]](#)

Download

[CRAN](#)

R Project

[About R](#)

[Logo](#)

[Contributors](#)

[What's New?](#)

[Reporting Bugs](#)

[Conferences](#)

[Search](#)

[Get Involved: Mailing Lists](#)

[Get Involved: Contributing](#)

[Developer Pages](#)

[R Blog](#)

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).

If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

News

- We are deeply sorry to announce that our friend and colleague Friedrich (Fritz) Leisch has died. [Read our tribute to Fritz here](#).
- **R version 4.4.0 (Puppy Cup)** has been released on 2024-04-24.
- **R version 4.3.3 (Angel Food Cake)** (wrap-up of 4.3.x) was released on 2024-02-29.
- **Registration for useR! 2024** has opened with early bird deadline March 31 2024.
- You can support the R Foundation with a renewable subscription as a [supporting member](#).

<https://mirror.kamp.de/cran/>

Greece

<https://ftp.cc.uoc.gr/mirrors/CRAN/>

Iceland

<https://cran.hafro.is/>

India

<https://cran.icts.res.in/>

<https://mirror.niser.ac.in/cran/>

Indonesia

<https://cran.usk.ac.id/>

Iran

<https://cran.um.ac.ir/>

KAMP Netzwerkdienste GmbH

University of Crete

Marine Research Institute

International Centre for Theoretical Sciences

National Institute of Science Education and Research (NISER)

Universitas Syiah Kuala

Ferdowsi University of Mashhad

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#) ([Debian](#), [Fedora/Redhat](#), [Ubuntu](#))
- [Download R for macOS](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

R-4.4.0 for Windows

[Download R-4.4.0 for Windows](#) (82 megabytes, 64 bit)

[README on the Windows binary distribution](#)

[New features in this version](#)

This build requires UCRT, which is part of Windows since Windows 10 and Windows Server 2016. On older systems, UCRT has to be installed manually from [here](#).

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)



```
R version 4.4.0 (2024-04-24 ucrt) -- "Puppy Cup"  
Copyright (C) 2024 The R Foundation for Statistical Computing  
Platform: x86_64-w64-mingw32/x64
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.
```

```
  Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.
```

```
> |
```

```
data1 <- read.csv("D://DVSDATA// 31Files// nyt1.csv")
```




```
R version 4.4.0 (2024-04-24 ucrt) -- "Puppy Cup"  
Copyright (C) 2024 The R Foundation for Statistical Computing  
Platform: x86_64-w64-mingw32/x64
```

```
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.
```

```
  Natural language support but running in an English locale
```

```
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.
```

```
> data1 <- read.csv("D://DVSDATA//31Files//nyt1.csv")  
> |
```

Clipboard: Paste, Cut, Copy, Format Painter

Font: Calibri, 11, Bold, Italic, Underline, Text Color, Background Color

Alignment: Wrap Text, Merge & Center, Left, Right, Center, Justify

Number: General, Percent, Decimals, Thousands Separator

	A	B	C	D	E
1	Age	Gender	Impressions	Clicks	Signed_In
2	36	0	3	0	1
3	73	1	3	0	1
4	30	0	3	0	1
5	49	1	3	0	1
6	47	1	11	0	1
7	47	0	11	1	1
8	0	0	7	1	0
9	46	0	5	0	1
10	16	0	3	0	1
11	52	0	4	0	1
12	0	0	8	1	0
13	21	0	3	0	1
14	0	0	4	0	0
15	57	0	6	0	1
16	31	0	5	0	1
17	0	0	6	0	0

```
> data1 <- read.csv("D://DVSDATA//31Files//nyt1.csv")
```

```
> head(data1)
```

	Age	Gender	Impressions	Clicks	Signed_In
1	36	0	3	0	1
2	73	1	3	0	1
3	30	0	3	0	1
4	49	1	3	0	1
5	47	1	11	0	1
6	47	0	11	1	1

```
> |
```

```
> data1$agecat <-cut(data1$Age,c(-Inf,0,18,24,34,44,54,64,Inf))
```

```
> head(data1)
```

	Age	Gender	Impressions	Clicks	Signed_In	agecat
1	36	0	3	0	1	(34,44]
2	73	1	3	0	1	(64, Inf]
3	30	0	3	0	1	(24,34]
4	49	1	3	0	1	(44,54]
5	47	1	11	0	1	(44,54]
6	47	0	11	1	1	(44,54]

```
> |
```

```
> # view
```

```
> summary(data1)
```

Age	Gender	Impressions	Clicks
Min. : 0.00	Min. :0.000	Min. : 0.000	Min. :0.00000
1st Qu.: 0.00	1st Qu.:0.000	1st Qu.: 3.000	1st Qu.:0.00000
Median : 31.00	Median :0.000	Median : 5.000	Median :0.00000
Mean : 29.48	Mean :0.367	Mean : 5.007	Mean :0.09259
3rd Qu.: 48.00	3rd Qu.:1.000	3rd Qu.: 6.000	3rd Qu.:0.00000
Max. :108.00	Max. :1.000	Max. :20.000	Max. :4.00000

Signed_In	agecat
Min. :0.0000	(-Inf,0] :137106
1st Qu.:0.0000	(34,44] : 70860
Median :1.0000	(44,54] : 64288
Mean :0.7009	(24,34] : 58174
3rd Qu.:1.0000	(54,64] : 44738
Max. :1.0000	(18,24] : 35270
	(Other) : 48005

```
> |
```

brackets

```
install.packages("doBy")
```

```
library("doBy")
```

```
siterange <- function(x){c(length(x), min(x), mean(x), max(x))}
```

```
summaryBy(Age~agecat, data =data1, FUN=siterange)
```



```
> # brackets
```

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

```
Warning in install.packages("doBy") :
```

```
'lib = "C:/Program Files/R/R-4.4.0/library"' is not writable
```

```
--- Please select a CRAN mirror for use in this session ---
```

```
also installing the dependencies 'crayon', 'fs', 'pkgbuild', 'rprojroot', 'diff$
```

```
trying URL 'https://cran.icts.res.in/bin/windows/contrib/4.4/crayon_1.5.2.zip'
```

```
Content type 'application/zip' length 164120 bytes (160 KB)
```

```
downloaded 160 KB
```

```
trying URL 'https://cran.icts.res.in/bin/windows/contrib/4.4/fs_1.6.4.zip'
```

```
Content type 'application/zip' length 413207 bytes (403 KB)
```

```
downloaded 403 KB
```

```
trying URL 'https://cran.icts.res.in/bin/windows/contrib/4.4/pkgbuild_1.4.4.zip'
```

```
Content type 'application/zip' length 204134 bytes (199 KB)
```

```
downloaded 199 KB
```

```
> library("doBy")
> siterange <- function(x){c(length(x), min(x), mean(x), max(x))}
> summaryBy(Age~agecat, data =data1, FUN=siterange)
  agecat Age.FUN1 Age.FUN2 Age.FUN3 Age.FUN4
1 (-Inf,0]   137106      0  0.00000      0
2  (0,18]    19252      7 16.03350     18
3  (18,24]   35270     19 21.26904     24
4  (24,34]   58174     25 29.50335     34
5  (34,44]   70860     35 39.49468     44
6  (44,54]   64288     45 49.49258     54
7  (54,64]   44738     55 59.49819     64
8 (64, Inf]   28753     65 72.98870    108
> |
```

```
# so only signed in users have ages and genders
summaryBy(Gender+Signed_In+Impressions+Clicks~agecat,
          data =data1)
```

```
> # so only signed in users have ages and genders
```

```
> summaryBy(Gender+Signed_In+Impressions+Clicks~agecat,data
+ =data1)
```

	agecat	Gender.mean	Signed_In.mean	Impressions.mean	Clicks.mean
1	(-Inf, 0]	0.0000000	0	4.999657	0.14207985
2	(0, 18]	0.6421151	1	4.998961	0.13105132
3	(18, 24]	0.5338531	1	5.006635	0.04845478
4	(24, 34]	0.5321621	1	4.993829	0.05048647
5	(34, 44]	0.5316963	1	5.021507	0.05167937
6	(44, 54]	0.5289790	1	5.010406	0.05027377
7	(54, 64]	0.5361885	1	5.022308	0.10183736
8	(64, Inf]	0.3632664	1	5.012347	0.15128856

```
> |
```

Drawing Plots

- **Install Package:** `install.packages("ggplot2")`
- **Load Library:** `library(ggplot2)`
- **Draw Histogram :** `ggplot(data1, aes(x=Impressions, fill=agecat))
+geom_histogram(binwidth=1)`
- **Draw Box Plot:** `ggplot(data1, aes(x=agecat, y=Impressions,
fill=agecat))+geom_boxplot()`

Install Package: `install.packages("ggplot2")`

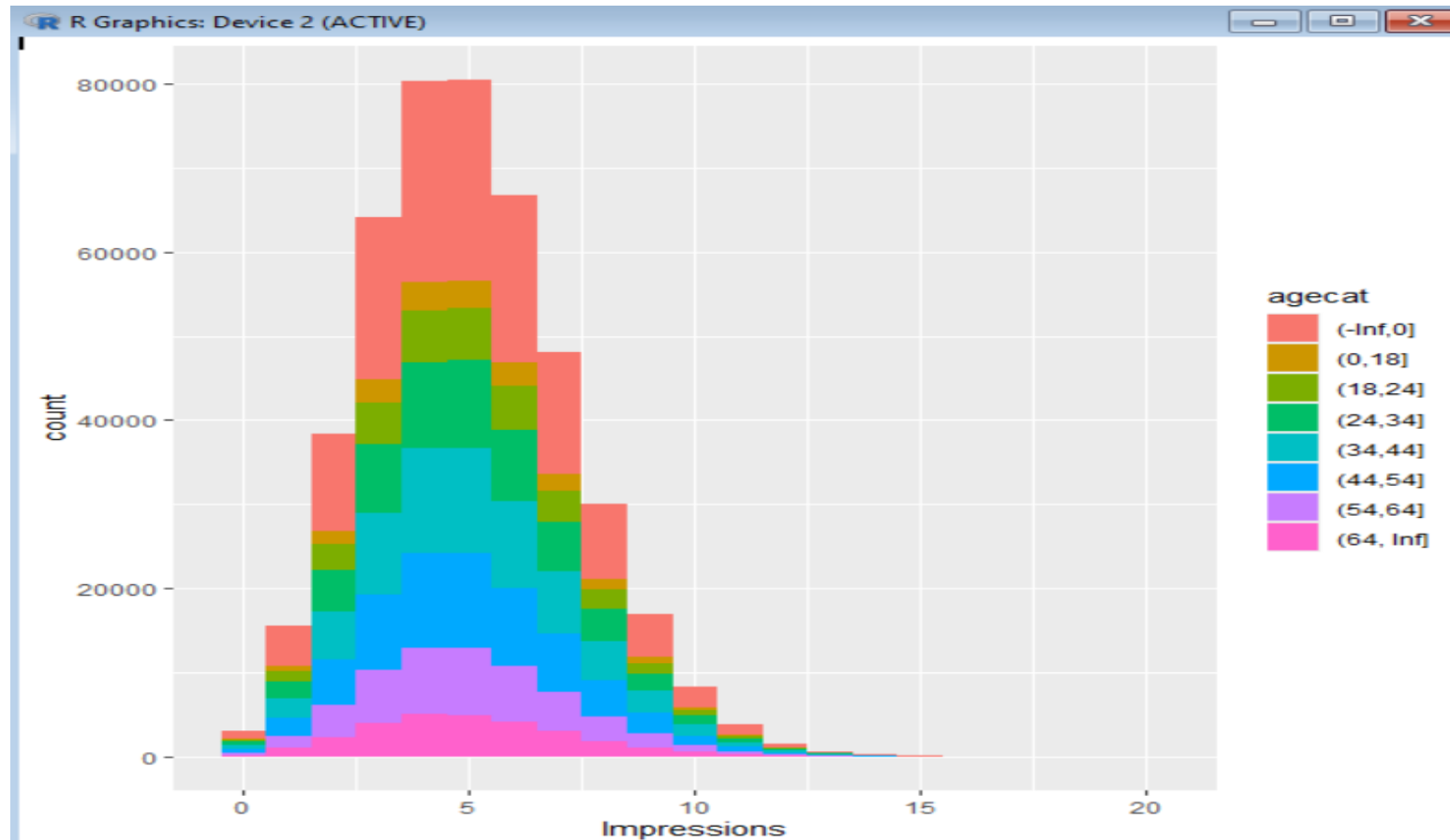
```
> # plot
> install.packages("ggplot2")
Installing package into 'C:/Users/proft/AppData/Local/R/win-library$
(as 'lib' is unspecified)
trying URL 'https://cran.icts.res.in/bin/windows/contrib/4.4/ggplot$
Content type 'application/zip' length 5011490 bytes (4.8 MB)
downloaded 4.8 MB

package 'ggplot2' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
  C:\Users\proft\AppData\Local\Temp\RtmpaAfXim\downloaded_pac$
```

Draw Histogram : `ggplot(data1, aes(x=Impressions, fill=agecat)) + geom_histogram(binwidth=1)`

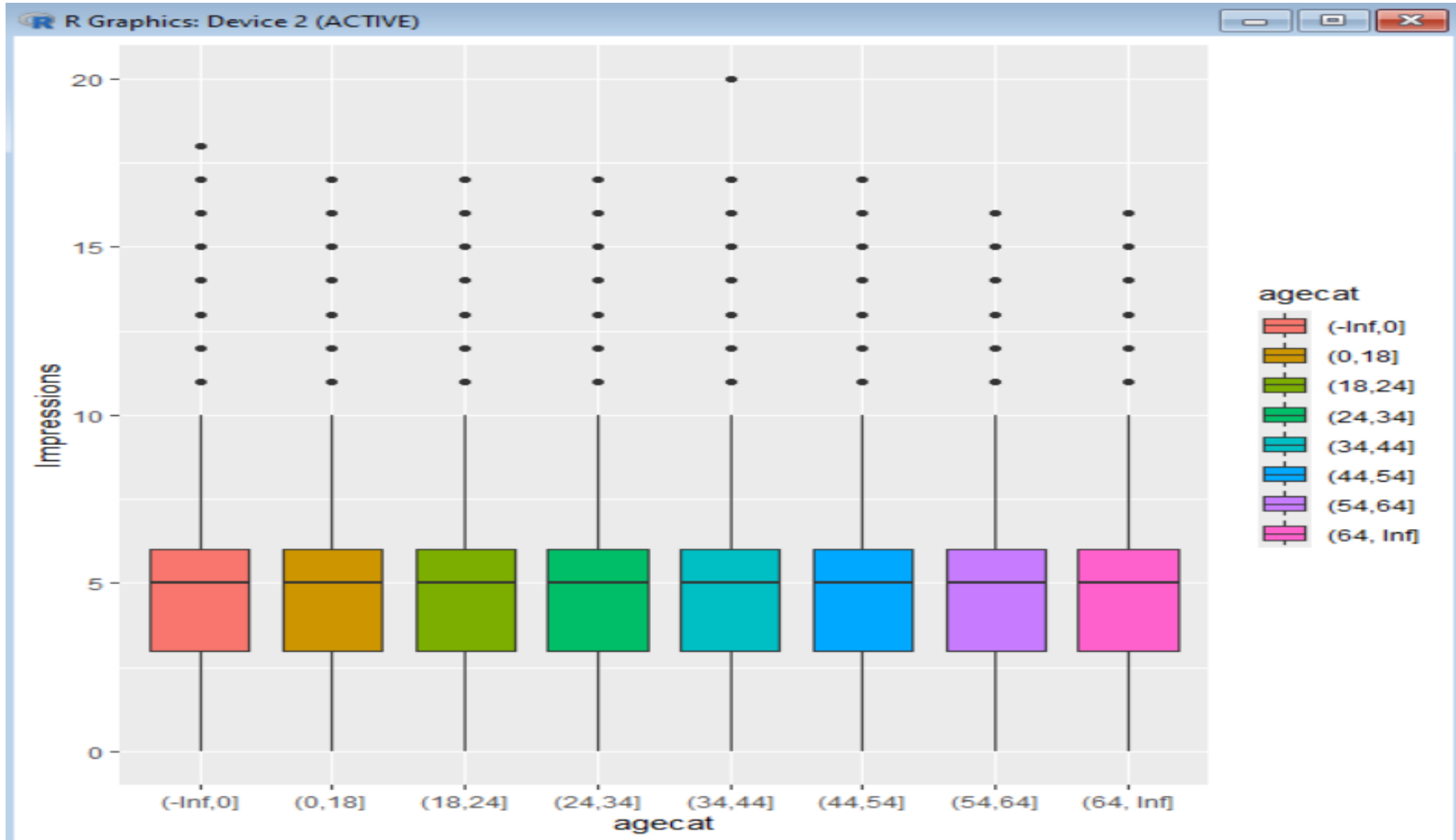
```
> library(ggplot2)
> ggplot(data1, aes(x=Impressions, fill=agecat)) + geom_histogram(binwidth=1)
> |
```



Box Blot

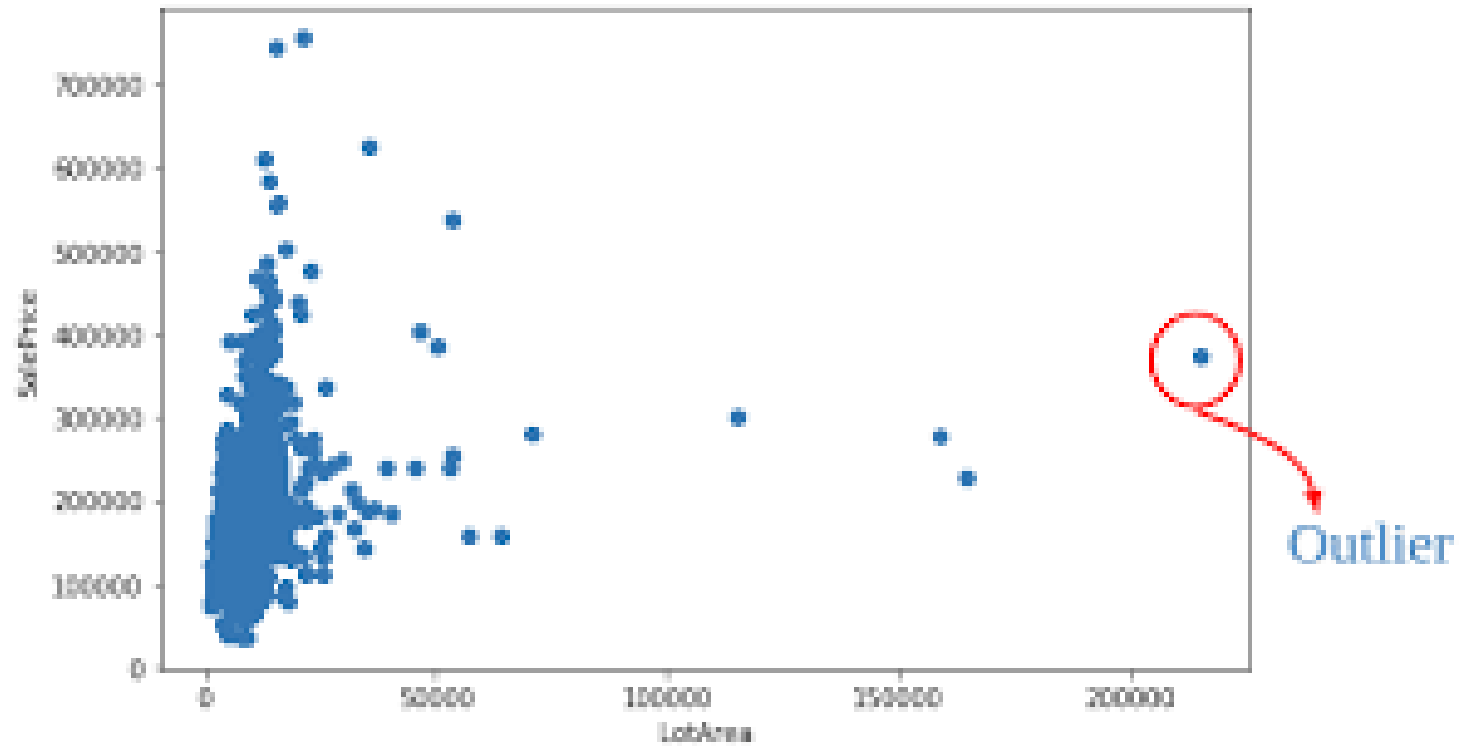
- A box plot, also known as a box-and-whisker plot, is a graphical representation of a dataset that displays its distribution and central tendency. It is particularly useful for visualizing the spread and skewness of the data, as well as identifying outliers.
- Here are the main components of a box plot:
- **Box:** The box itself represents the interquartile range (IQR), which is the range between the first quartile (Q1) and the third quartile (Q3). This range contains the middle 50% of the data.
- **Median Line:** A line inside the box marks the median (Q2) of the dataset.
- **Whiskers:** Lines extending from the box to the smallest and largest values within 1.5 times the IQR from Q1 and Q3, respectively. These lines show the range of the data excluding outliers.
- **Outliers:** Data points that fall outside of the whiskers. They are typically plotted as individual points beyond the whiskers.

Draw Box Plot: `ggplot(data1, aes(x=agecat, y=Impressions, fill=agecat))+geom_boxplot()`



Outliers

- In data science, outliers are data points that significantly deviate from the rest of the dataset. These anomalies can occur due to variability in the data or errors in the data collection process. Identifying and handling outliers is crucial because they can significantly affect the results of data analysis and statistical modeling.



Example of Outliers

- Consider a dataset representing the ages of students in a university class:
- **Ages=[18,19,20,21,22,23,24,25,26,27,40]**
- In this dataset, most students' ages range from **18 to 27 years**. However, there is one student aged 40, which is significantly higher than the rest. This age of 40 is an outlier.

Thought Experiment: How Would You Simulate Chaos?

[Simulating Chaos: Exploring Order from Disorder]

- Most data problems start out with a certain amount of dirty data, ill defined questions, and urgency. As data scientists we are, in a sense, attempting to create order from chaos.
- **Lorenzian Water Wheel: Ferris wheel-like apparatus** with rotating water buckets, exhibiting chaotic behavior influenced by molecular interactions.
- **Digital Chaos Simulations:** Binder and Jensen's paper explores **chaos using finite-state machines in digital simulations**.
- **Interdisciplinary Program:** M.I.T., Harvard, and Tufts collaborated on a **chaos simulation involving a humanitarian crisis** scenario in Chad-Sudan border region.
- **Creating Order in Startups:** Gascoigne's essay discusses strategies for **managing chaos within startup environments** to establish order amidst uncertainty and complexity.

Exploring Chaos and Simulation in Data Science

- **Simulation serves** as a **vital tool in data science**, aiding in understanding model behavior and debugging code
- **Data scientists** navigate **organizational chaos**, striving to impose order and learn from chaotic experiences.

Case Study: RealDirect

https://www.realdirect.co.za



[HOME](#) [SERVICES](#) [THE TEAM](#) [CONTACT](#)



Services

Property Valuations | Business Valuations | Surveying | Accounting



Consultants and valuers in the disposal and acquisition of real estate



Property Insurance valuations



Property portfolio audits



Liquidations valuations



Feasibility studies for township developments



Rental reviews and optimisation

Case Study: RealDirect

- **CEO** : Doug Perlson, Expert in real estate law, startups, and online advertising.
- **Vision** : To *leverage comprehensive real estate data to enhance the process of buying and selling homes.*
- **Existing Status** : Typically, homeowners sell their properties approximately *every seven years* with assistance from **brokers** and relying on **current data**.
- **Challenges** : The **broker system** and **data quality**.

Case Study: RealDirect

- **RealDirect** collects information by assembling a **team of licensed real estate agents** who **collaborate and share knowledge** through a **seller interface with data-driven tips for selling homes** and **utilizing interaction data to offer real-time recommendations** for next steps.
- Brokers are **trained to become proficient** in using **information-gathering tools**.
- RealDirect is **actively developing real-time data feeds**, covering crucial aspects such as **home search initiation, initial offers, duration between offer and closure, and online search patterns**.

How Does RealDirect Make Money?

- Offering a **subscription to Registered sellers**—about **\$395 a month**—to access the **selling tools**.
- Reduced Real Estate Agents commission, from **2.5% or 3%** to **2%**.
- Website provide statuses for each person on site: **active, offer made, offer rejected, showing, in contract, etc**. Based on status, **different actions** are suggested by the software
- **Important factors** that buyers prioritize was included in website, such as proximity to **parks, subway stations, and schools, along with comparisons of apartment prices per square foot** within the same building or block.

How Does RealDirect Make Money?

- Ultimately, access to accurate and timely information benefits both *buyers and sellers, assuming transparency and honesty in the transaction process.*

Exercise: RealDirect Data Strategy

- You have been hired as chief data scientist at **realdirect.com**, and report directly to the CEO.
- The company (hypothetically) does not yet have its data plan in place.
- It's looking to you to come up with a **data strategy**.
- Here are a couple ways you could begin to approach this problem:

Approaches

- 1. Explore all Current Details of Existing Website**
- 2. Get Auxiliary Data**
- 3. Load, Clean and Conduct Exploratory Data Analysis**
- 4. Summarize the out come of EDA**
- 5. Can you think of any other people you should talk to?**
- 6. Acquire Domain Knowledge**
- 7. Work on Best Practices for Data Strategy**

1. Explore all Current Details of Existing Website

Explore its **existing website**, thinking about how buyers and sellers would navigate through it, and how the website is **structured/organized**.






Try to understand the existing business model, and think about how analysis of **RealDirect** user-behavior data could be used to inform decision-making and product development.

Come up with a list of research questions you think could be answered by data:

- What data would you advise the engineers log and what would your ideal datasets look like?
- How would data be used for reporting and monitoring product usage?
- How would data be built back into the product/website?

2. Get Auxiliary Data

- Because there is no data yet for you to analyze (typical in a start up when its still building its product), you should get some auxiliary data to **help gain intuition about this market.**
- For example, go to https://github.com/oreillymedia/doing_data_science. Click on **Rolling Sales Update (after the fifth paragraph)**. You can use any or all of the datasets here.

 rollingsales_bronx	14-05-2024 05:32	Microsoft Excel 97...	1,508 KB
 rollingsales_brooklyn	14-05-2024 05:32	Microsoft Excel 97...	6,454 KB
 rollingsales_manhattan	14-05-2024 05:32	Microsoft Excel 97...	7,193 KB
 rollingsales_queens	14-05-2024 05:32	Microsoft Excel 97...	6,582 KB
 rollingsales_statenisland	14-05-2024 05:32	Microsoft Excel 97...	1,814 KB

Clipboard Font Alignment Number Styles Cells Editing

Paste

Arial 10 A A

B I U

Wrap Text

General

Conditional Formatting

Format as Table

Cell Styles

Insert

Delete

Format

Sort & Filter

Find & Select

	A	B	C	D	E	F	G	H	I
1	Bronx Rolling Sales File. All Sales From August 2012 - August 2013.								
2	Sales File as of 08/30/2013 Coop Sales Files as of 09/18/2013								
3	Neighborhood Name 09/06/13, Descriptive Data is as of 06/01/13								
4	Building Class Category is based on Building Class at Time of Sale.								
5	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESEN	BLOCK	LOT	EASEMENT	BUILDING CLASS AT PRESEN	ADDRESS
6	2	BATHGATE	01 ONE FAMILY HOMES	1	3028	25		A5	412 EAST 179TH STREET
7	2	BATHGATE	01 ONE FAMILY HOMES	1	3039	28		A1	2329 WASHINGTON AVENUE
8	2	BATHGATE	01 ONE FAMILY HOMES	1	3046	39		A1	2075 BATHGATE AVENUE
9	2	BATHGATE	01 ONE FAMILY HOMES	1	3046	52		A1	2047 BATHGATE AVENUE
10	2	BATHGATE	02 TWO FAMILY HOMES	1	2900	61		S2	406 EAST TREMONT AVENUE
11	2	BATHGATE	02 TWO FAMILY HOMES	1	2912	158		B1	505 EAST 171ST STREET
12	2	BATHGATE	02 TWO FAMILY HOMES	1	2929	117		B1	3860 3 AVENUE
13	2	BATHGATE	02 TWO FAMILY HOMES	1	3030	60		B3	4469 PARK AVENUE
14	2	BATHGATE	02 TWO FAMILY HOMES	1	3035	27		B1	454 EAST 179 STREET
15	2	BATHGATE	02 TWO FAMILY HOMES	1	3039	65		B2	465 EAST 185 STREET
16	2	BATHGATE	02 TWO FAMILY HOMES	1	3040	5		S2	4654-4656 PARK AVENUE
17	2	BATHGATE	02 TWO FAMILY HOMES	1	3046	54		B2	2043 BATHGATE AVENUE
18	2	BATHGATE	02 TWO FAMILY HOMES	1	3050	85		B1	2241 BATHGATE AVENUE
19	2	BATHGATE	02 TWO FAMILY HOMES	1	3052	37		S2	4557 3 AVENUE

File

Home

Insert

Page Layout

Formulas

Data

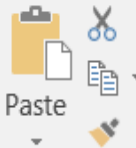
Review

View

Tell me what you want to do...

Sign in

Share



Arial 12

A A



Wrap Text

General

B I U



A



Merge & Center

%

.00

.00



Conditional Formatting



Format as Table



Cell Styles



Insert



Delete



Format



Sort & Filter



Find & Select



Find & Select



Find & Select



Find & Select



Find & Select

Clipboard

Font

Alignment

Number

Styles

Cells

Editing

A

B

C

D

E

F

G

H

I

1 **Staten Island Rolling Sales File. All Sales From August 2012 - August 2013.**

2 Sales File as of 08/30/2013 Coop Sales Files as of 09/18/2013

3 Neighborhood Name 09/06/13, Descriptive Data is as of 06/01/13

4 Building Class Category is based on Building Class at Time of Sale.

	BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESEN	BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESEN	ADDRESS
5	5	ANNADALE	01 ONE FAMILY HOMES	1	5395	32		A1	541 SYCAMORE STREET
6	5	ANNADALE	01 ONE FAMILY HOMES	1	5401	10		A2	16 JANSEN STREET
7	5	ANNADALE	01 ONE FAMILY HOMES	1	5401	38		A1	27 WEAVER STREET
8	5	ANNADALE	01 ONE FAMILY HOMES	1	5407	11		A1	24 ELMBANK STREET
9	5	ANNADALE	01 ONE FAMILY HOMES	1	5425	39		A1	23 SANDGAP STREET
10	5	ANNADALE	01 ONE FAMILY HOMES	1	6205	16		A5	93 EAGAN AVENUE
11	5	ANNADALE	01 ONE FAMILY HOMES	1	6205	55		A5	36 SEGUINE PLACE
12	5	ANNADALE	01 ONE FAMILY HOMES	1	6205	126		A5	20 MAY PLACE
13	5	ANNADALE	01 ONE FAMILY HOMES	1	6211	20		A5	9 EAGAN AVENUE
14	5	ANNADALE	01 ONE FAMILY HOMES	1	6212	28		A1	96 JEANNETTE AVENUE
15	5	ANNADALE	01 ONE FAMILY HOMES	1	6212	44		A5	50 LUCY LOOP
16	5	ANNADALE	01 ONE FAMILY HOMES	1	6212	85		A1	1086 ARDEN AVENUE
17	5	ANNADALE	01 ONE FAMILY HOMES	1	6216	32		A1	8 EAGAN AVENUE
18	5	ANNADALE	01 ONE FAMILY HOMES	1	6217	32		A2	10 FABIAN STREET
19	5	ANNADALE	01 ONE FAMILY HOMES	1	6217	32		A2	10 FABIAN STREET

Staten Island



3. Load, Clean and Conduct Exploratory Data Analysis

- **First challenge:** load in and clean up the data.
- **Next**, conduct exploratory data analysis in order to find out where there are **outliers or missing values**, decide how you will treat them, make sure the dates are formatted correctly, make sure values you think are numerical are being treated as such, etc.
- **Once the data is in good shape**, conduct exploratory data analysis to visualize and make comparisons
 - (i) across neighborhoods, and
 - (ii) across time.

3. Summarize the outcome of EDA

- Summarize your findings in a brief report aimed at the CEO

4. Can you think of any other people you should talk to?

- Being the “**data scientist**” often involves speaking to people **who aren’t also data scientists**, so it would be ideal to have a **set of communication strategies** for getting to the information you need about the data.

5. Acquire Domain Knowledge

- Most of you are not “**domain experts**” in real estate or online businesses.
- Step out of your comfort zone and figure out how to “**collect data**”
- Master the domain specific vocabulary. Sometimes “**domain experts**” have their own set of vocabulary.
 - Did Doug use vocabulary specific to his domain that you didn’t understand (“**comps,**” “**open houses,**” “**CPC**”)?

6. Work on Best Practices for Data Strategy

- **Doug** mentioned the company didn't necessarily have a data strategy. There is **no industry standard for creating one**.
- As you work through this assignment, think about whether there is a **set of best practices you would recommend** with respect to developing a data strategy for an online business, or in your own domain.

Sample R code

- Here's some **sample R** code that takes the Brooklyn housing data in the preceding exercise, and cleans and explores it a bit.
- [Size of Data Set: Rows: 23374, Columns: 21]

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

```
>library(readxl)
```

```
>bk <- read_excel("D://DVSDATA//rollingsales_brooklyn.xls", sheet = 1)
```

```
>head(bk)
```

```
>summary(bk)
```

head(bk)

```
> head(bk)
# A tibble: 6 × 21
  BOROUGH NEIGHBORHOOD BUILDING CLASS CATEG...1 `TAX CLASS AT PRESENT` BLOCK LOT
  <dbl> <chr>          <chr>          <chr>          <dbl> <dbl>
1     3 <NA>          15 CONDOS - 2-10 UNI... <NA>      814 1103
2     3 <NA>          15 CONDOS - 2-10 UNI... <NA>      814 1105
3     3 <NA>          15 CONDOS - 2-10 UNI... <NA>     1967 1401
4     3 <NA>          15 CONDOS - 2-10 UNI... <NA>     1967 1402
5     3 <NA>          15 CONDOS - 2-10 UNI... <NA>     1967 1403
6     3 <NA>          15 CONDOS - 2-10 UNI... <NA>     1967 1404
# i abbreviated name: 1`BUILDING CLASS CATEGORY`
# i 15 more variables: `EASE-MENT` <lgl>, `BUILDING CLASS AT PRESENT` <chr>,
# ADDRESS <chr>, `APART\nMENT\nNUMBER` <chr>, `ZIP CODE` <dbl>,
# `RESIDENTIAL UNITS` <dbl>, `COMMERCIAL UNITS` <dbl>, `TOTAL UNITS` <dbl>,
# `LAND SQUARE FEET` <dbl>, `GROSS SQUARE FEET` <dbl>, `YEAR BUILT` <dbl>,
# `TAX CLASS AT TIME OF SALE` <dbl>, `BUILDING CLASS AT TIME OF SALE` <chr>,
# `SALE\nPRICE` <dbl>, `SALE DATE` <dtm>
```

summary (bk)

> summary (bk)

BOROUGH	NEIGHBORHOOD	BUILDING CLASS CATEGORY	TAX CLASS AT PRESENT
Min. :3	Length:23373	Length:23373	Length:23373
1st Qu.:3	Class :character	Class :character	Class :character
Median :3	Mode :character	Mode :character	Mode :character
Mean :3			
3rd Qu.:3			
Max. :3			

BLOCK	LOT	EASE-MENT	BUILDING CLASS AT PRESENT
Min. : 20	Min. : 1.0	Mode:logical	Length:23373
1st Qu.:1638	1st Qu.: 22.0	NA's:23373	Class :character
Median :3839	Median : 48.0		Mode :character
Mean :3984	Mean : 305.4		
3rd Qu.:6259	3rd Qu.: 142.0		
Max. :8955	Max. :9039.0		

ADDRESS	APART\MENT\NUMBER	ZIP CODE	RESIDENTIAL UNITS
Length:23373	Length:23373	Min. : 0	Min. : 0.000
Class :character	Class :character	1st Qu.:11209	1st Qu.: 1.000
Mode :character	Mode :character	Median :11218	Median : 1.000
		Mean :11211	Mean : 2.156
		3rd Qu.:11230	3rd Qu.: 2.000
		Max. :11416	Max. :509.000

To print Missing Values

```
na_count <- sum(is.na(bk$SALEPRICE))  
print(na_count)
```

Output: 0

List all column names : names(bk)

```
> bk <- read_excel("D://DVSDATA//rollingsales_brooklyn.xls", sheet = 1)
> names(bk)
 [1] "BOROUGH"                "NEIGHBORHOOD"
 [3] "BUILDING CLASS CATEGORY" "TAX CLASS AT PRESENT"
 [5] "BLOCK"                  "LOT"
 [7] "EASE-MENT"              "BUILDINGCLASSAT PRESENT"
 [9] "ADDRESS"                "APARTMENTNUMBER"
[11] "ZIPCODE"                "RESIDENTIALUNITS"
[13] "COMMERCIALUNITS"       "TOTALUNITS"
[15] "LANDSQUAREFEET"        "GROSSSQUAREFEET"
[17] "YEARBUILT"              "TAXCLASSATTIMEOFSALE"
[19] "BUILDINGCLASSATTIMEOFSALE" "SALEPRICE"
[21] "SALEDATE"
```


clean/format the data with regular expressions

```
bk$gross.sqft <- as.numeric(gsub("[^[:digit:]]", "", bk$GROSSSSQUAREFEET))
```

- Access the **GROSSSSQUAREFEET** column:
- Use **gsub** to remove non-digit characters
- **as.numeric(...)** converts the resulting cleaned string, which now contains only digits, to a numeric type.
- Assign the numeric value to a new column **gross.sqft**

Example

Suppose **bk\$GROSSSSQUAREFEET** has the following values:

"1,000"

"2,500"

"3,750sqft"

The **gsub("[^[:digit:]]", "", bk\$GROSSSSQUAREFEET)** operation would transform these values to:

"1000"

"2500"

"3750"

```
> print(bk$gross.sqft)
```

```
[1] 0 0 0 0 0 0 0 0 0
[10] 0 0 0 0 0 0 0 0 0
[19] 0 0 0 0 0 0 0 1492 1724
[28] 2132 1704 2640 3304 2000 1800 1281 2346 2835
[37] 1562 1328 2500 2070 4071 5107 1392 1392 1392
[46] 1392 1860 1860 1392 1479 992 1320 2700 2700
[55] 2700 3160 2224 2242 2090 1622 1935 2340 1401
[64] 2048 2745 3240 4140 2512 1200 1584 2521 2062
[73] 2890 3340 3340 3340 2544 1984 2904 2845 2845
[82] 2845 1771 2115 3240 3240 2772 2018 2160 2160
[91] 3195 3195 2448 2953 2953 4400 2365 2365 1513
[100] 2910 3254 2000 1802 1812 2560 2194 2194 2194
[109] 4105 1635 1785 2275 2620 1940 1953 1762 2490
[118] 1934 3420 2352 1995 2760 2802 2348 1674 2060
[127] 2520 1756 1998 1729 1928 1928 1782 1782 1860
[136] 2340 2160 1980 1980 2160 2160 1980 1980 1776
[145] 1645 1928 1610 1840 1720 1652 1652 1652 1584
[154] 1869 1869 1869 1869 1673 1320 1993 1584 1584
[163] 1734 1734 5300 3060 2328 3240 3826 2648 4480
```

Similarly

- `bk$land.sqft <- as.numeric(gsub("[^[:digit:]]", "", bk$LANDSQUAREFEET))`
- `bk$sale.date <- as.Date(bk$SALEDATE)`
- `bk$year.built <- as.numeric(as.character(bk$YEARBUILT))`

Print and Check

- `print(bk$land.sqft)`
- `print(bk$sale.date)`
- `print(bk$year.built)`

```
print(bk$land.sqft)
```

```
[1] 0 0 0 0 0 0 0 0 0 0
[10] 0 0 0 0 0 0 0 0 0 0
[19] 0 0 0 0 0 0 0 0 2058 4833
[28] 2417 3867 1600 2707 2417 1172 1440 2513 5810
[37] 1933 1990 1713 2223 5800 7747 2511 1648 1648
[46] 1649 2062 2062 1861 1878 1372 1600 2320 2320
[55] 2320 1948 1547 2383 2500 2103 3867 2167 2140
[64] 1800 3178 1933 2610 2417 985 2417 2223 1456
[73] 5000 3853 3853 3853 2900 4833 9667 4833 4833
[82] 4833 2127 1933 2344 2344 1933 1925 1993 1933
[91] 2083 2083 2167 2167 2167 2025 1558 1558 1933
[100] 1820 5518 2900 2449 2000 1933 3867 3867 3867
[109] 3867 2147 1933 1904 2020 2331 2058 1916 1906
```

```
print(head(bk$sale.date,n= 100))
```

```
[1] "2013-07-09" "2013-07-12" "2013-04-25" "2013-04-25" "2013-04-25"  
[6] "2013-04-25" "2013-07-25" "2013-07-25" "2012-11-19" "2013-04-22"  
[11] "2012-11-12" "2013-02-08" "2012-11-13" "2012-11-13" "2012-12-07"  
[16] "2013-02-15" "2013-01-09" "2013-01-07" "2012-11-07" "2013-06-25"  
[21] "2013-07-03" "2013-06-19" "2013-06-03" "2013-01-16" "2013-03-18"  
[26] "2013-06-06" "2012-12-18" "2012-08-24" "2013-06-18" "2012-12-14"  
[31] "2012-11-29" "2012-11-14" "2013-06-03" "2013-03-11" "2013-06-06"  
[36] "2012-09-27" "2013-05-07" "2012-09-25" "2013-05-31" "2012-10-22"  
[41] "2012-11-29" "2013-07-25" "2013-01-03" "2013-02-21" "2013-01-11"  
[46] "2012-12-12" "2013-03-22" "2013-03-01" "2013-07-24" "2012-11-16"  
[51] "2013-03-18" "2013-01-24" "2012-11-07" "2013-03-22" "2012-08-30"  
[56] "2013-07-17" "2012-10-20" "2013-02-20" "2013-05-10" "2012-11-13"  
[61] "2012-10-12" "2013-06-19" "2013-06-17" "2013-03-21" "2013-01-02"  
[66] "2013-07-03" "2013-06-24" "2013-04-03" "2013-06-28" "2013-04-02"  
[71] "2013-05-01" "2013-04-23" "2013-04-05" "2013-06-18" "2013-05-17"  
[76] "2013-01-15" "2013-03-01" "2012-08-13" "2012-12-28" "2012-09-19"  
[81] "2012-09-19" "2012-09-19" "2012-11-29" "2012-08-24" "2012-11-20"  
[86] "2012-09-27" "2013-06-25" "2012-11-16" "2013-03-26" "2013-01-14"  
[91] "2012-12-20" "2012-12-19" "2012-09-28" "2012-08-15" "2012-08-15"  
[96] "2013-02-13" "2013-01-25" "2013-01-25" "2012-08-17" "2013-01-15"
```



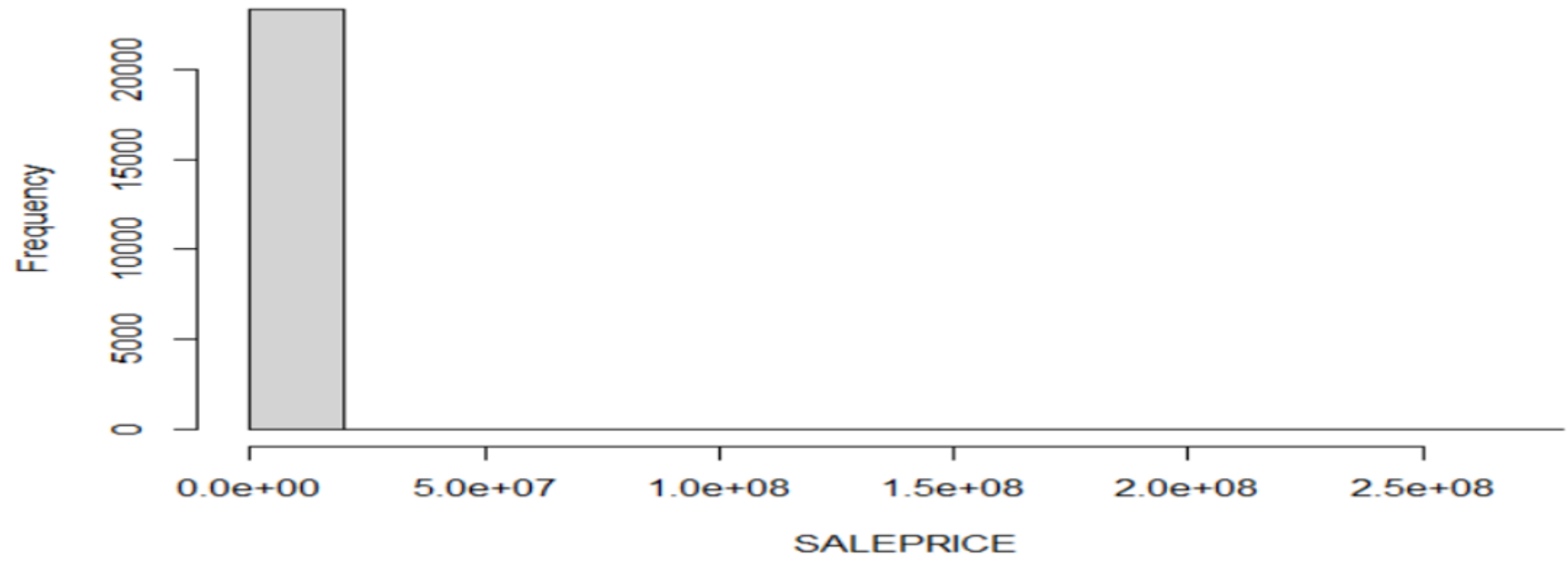
```
> print(head(bk$year.built,n=100))
```

```
[1] 0 0 0 0 0 0 0 0 0 2011 2011 2011 2011 2011 2011 2011
[16] 2011 2011 2011 2011 2011 0 0 0 0 0 1930 1930 1930 1899 1925
[31] 1960 1935 1920 1915 1935 1950 1920 1925 1930 1955 1925 1945 1950 1945 1945
[46] 1940 1940 1940 1940 1945 1950 1940 1930 1930 1930 1930 1930 1930 1930 1930
[61] 1950 1920 1960 1910 1950 1901 1950 1905 1930 1930 1930 1940 1935 1940 1940
[76] 1940 1930 1899 1899 1899 1899 1899 1950 1920 1940 1940 1935 1955 1960 1960
[91] 1930 1930 1925 1960 1960 1925 1965 1965 1910 1992
```

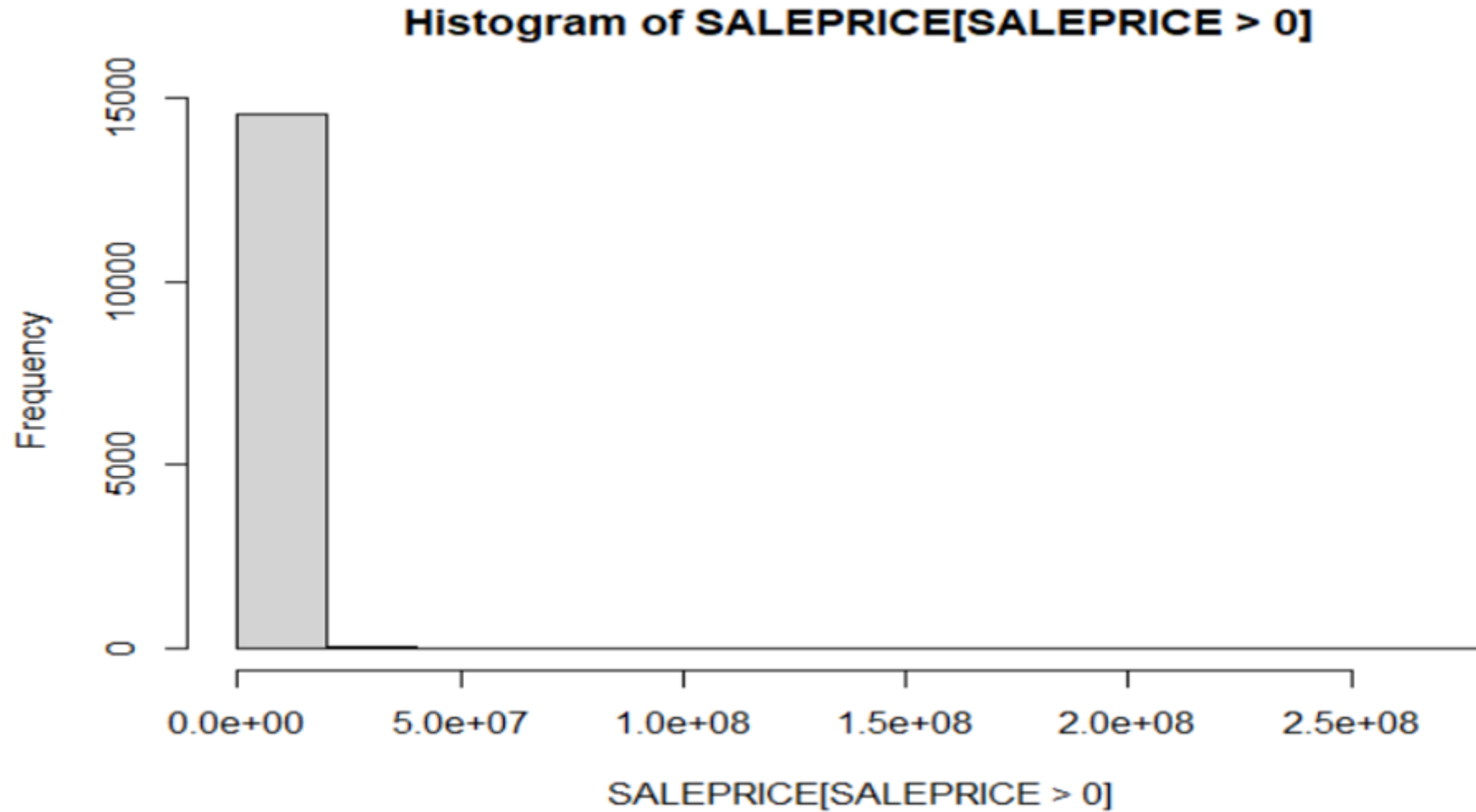

Exploratory Data Analysis

```
attach(bk)
hist(SALEPRICE)
```

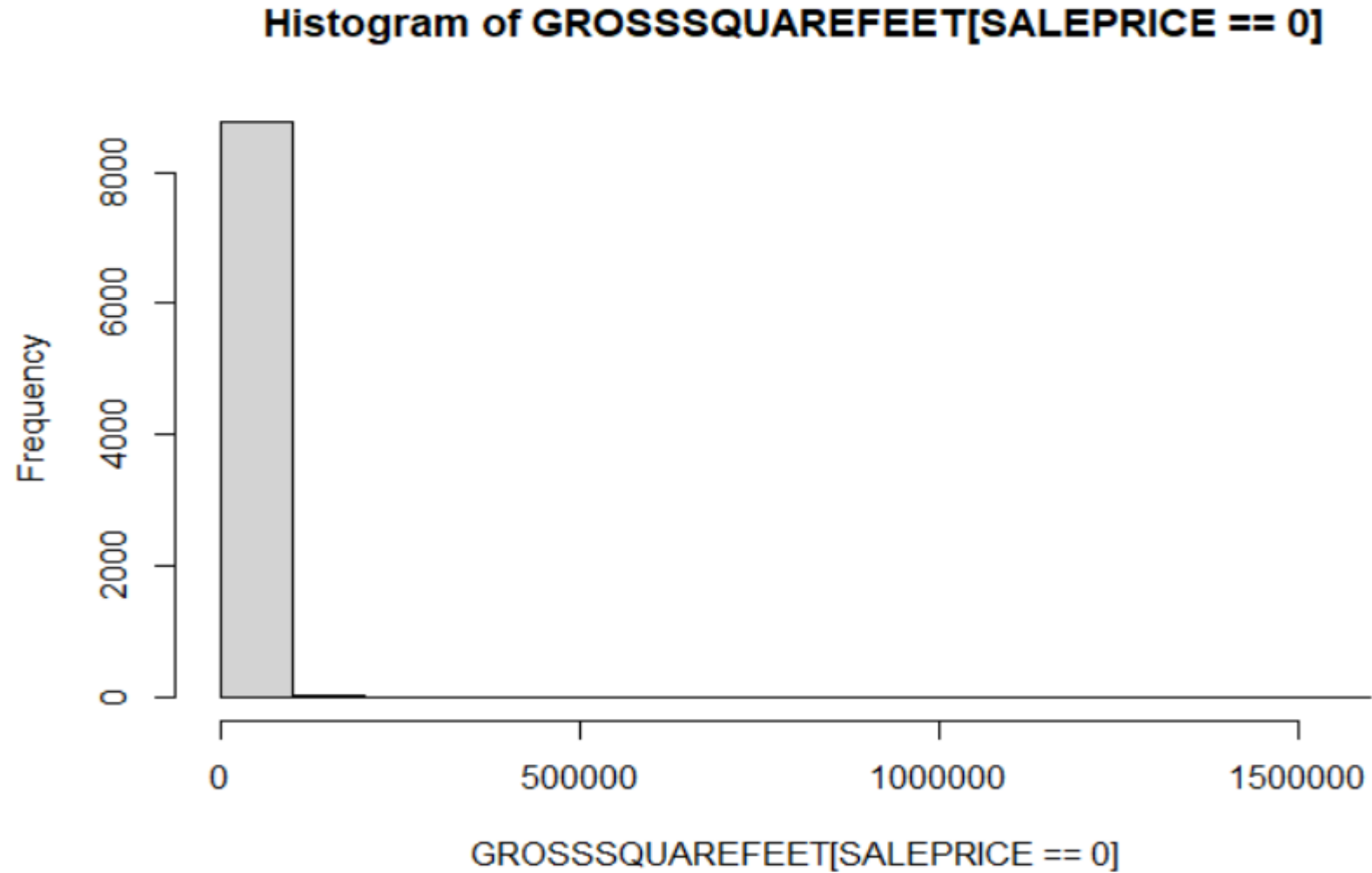
Histogram of SALEPRICE



```
hist(SALEPRICE[SALEPRICE>0])
```



hist(GROSSSQUAREFEET[SALEPRICE==0])



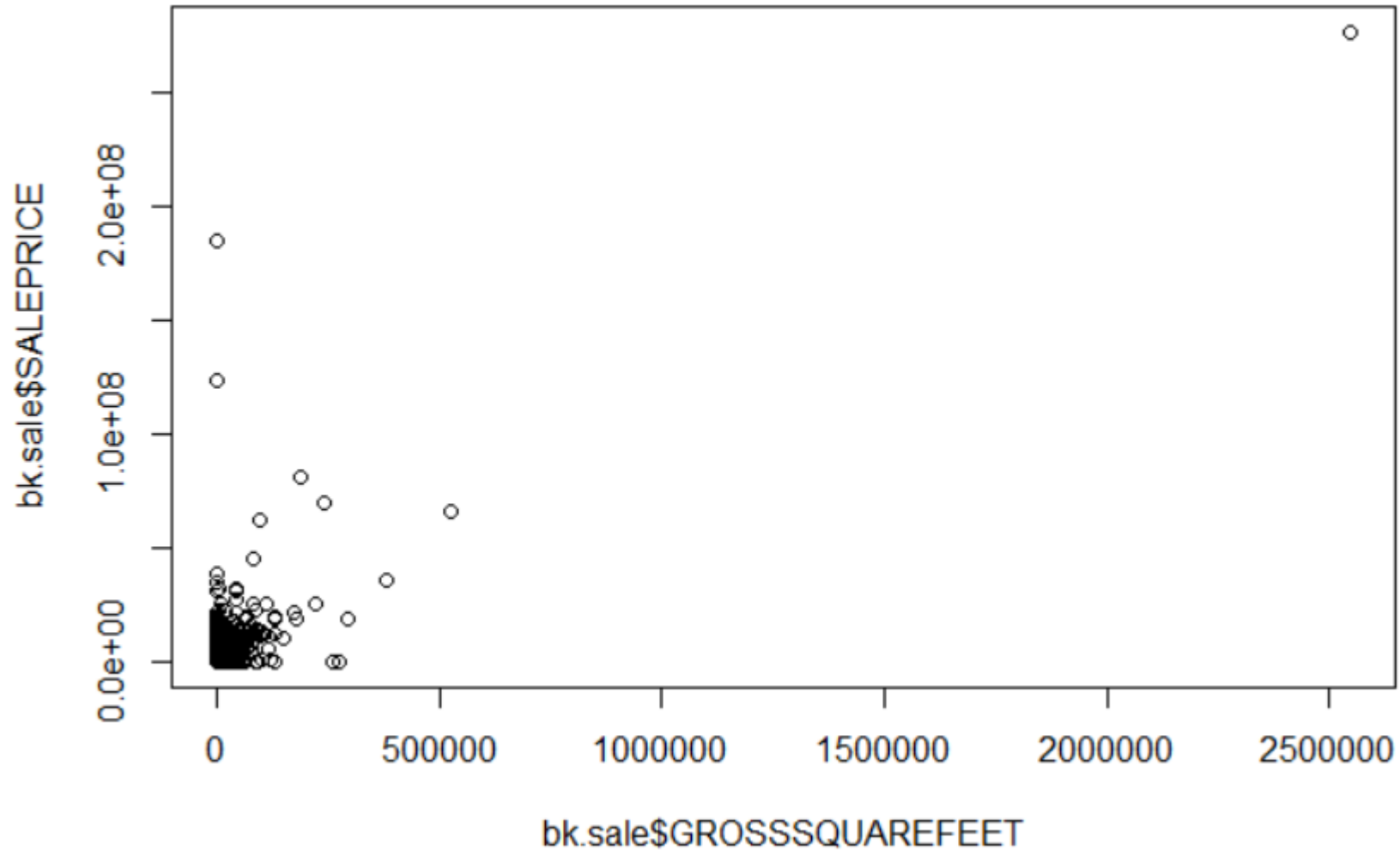
Complete Code

- `hist(SALEPRICE)`
- `hist(SALEPRICE[SALEPRICE>0])`
- `hist(GROSSSQAREFEET[SALEPRICE==0])`
- `detach(bk)`

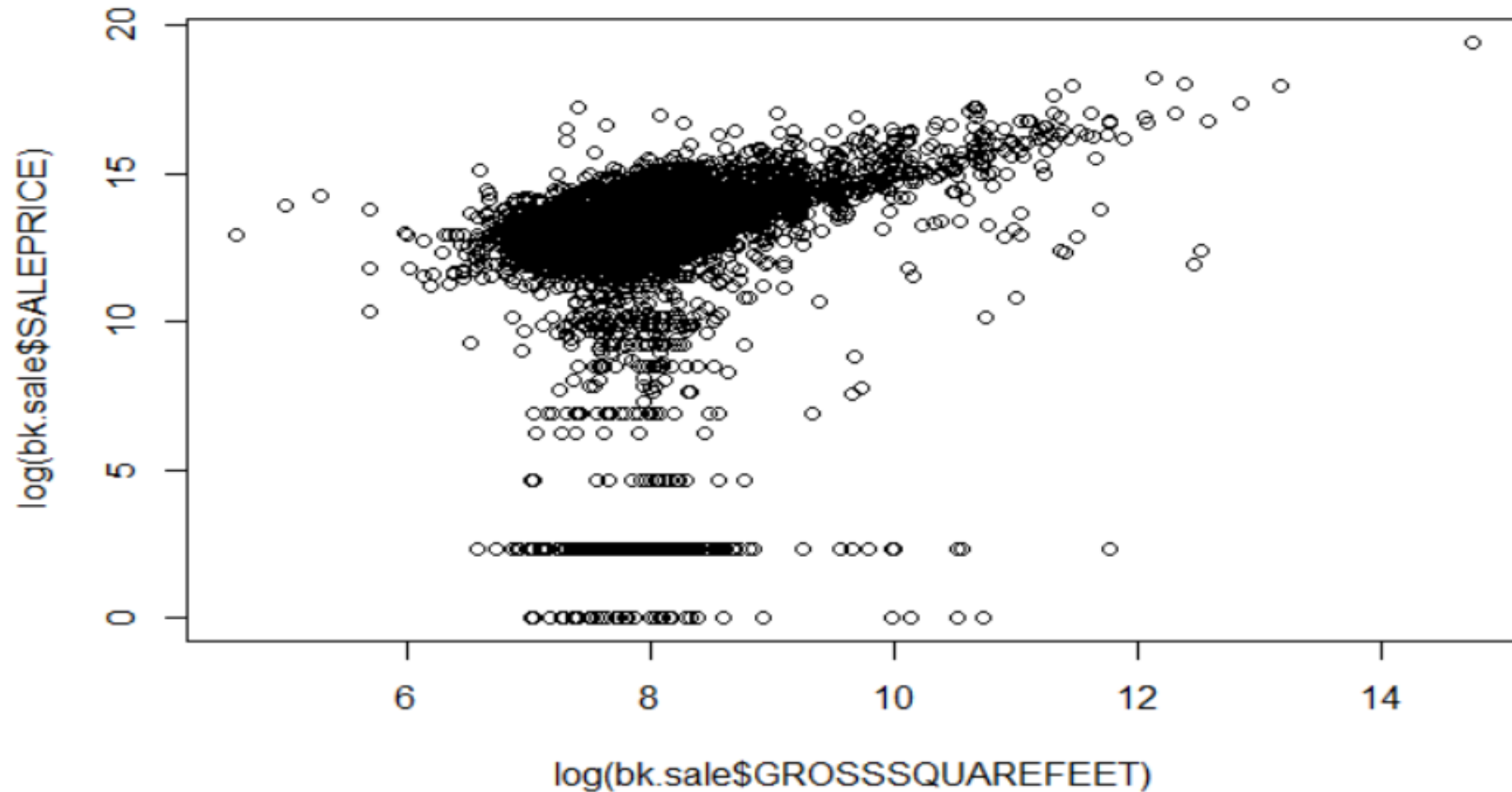
```
## keep only the actual sales
```

```
bk.sale <- bk[bk$SALEPRICE!=0,]
```

```
plot(bk.sale$GROSSSSQUAREFEET,bk.sale$SALEPRICE)
```



```
plot(log(bk.sale$GROSSSSQUAREFEET),log(bk.sale$SALEPRICE))
```



Algorithms

- An algorithm is a **procedure or set of steps or rules** to accomplish a task.
- Algorithms are one of the **fundamental concepts in, or building blocks of**, computer science:
 - the basis of the **design of elegant and efficient code, data preparation and processing, and software engineering.**
- Some of the basic types of tasks that algorithms can solve are ***sorting, searching, and graph-based computational problems.***
- **Efficiency is measured in terms of memory and computational time**, which matters especially when you're dealing with massive amounts of data and building consumer facing products.

Characteristics of an Algorithm

1. **(Definiteness) Well-Defined Instructions** : Clear and unambiguous steps.
2. **Input**: Specific data or parameters provided to the algorithm.
3. **Output** : One or more results produced by the algorithm.
4. **Finiteness** : Terminates after a finite number of steps.
5. **Effectiveness** : Each step can be performed in a finite amount of time using basic computational operations.

Additional Characteristics (Optional)

- **Deterministic:** Most algorithms are deterministic, meaning that given the **same input, they will always produce the same output**. However, some algorithms can be **non-deterministic or probabilistic**, incorporating randomness in their steps.
- **General-purpose:** Algorithms can often be designed *to solve a broad class of problems*, making them versatile and reusable for different applications with similar characteristics.

ML Algorithms

- In the context of machine learning, **an algorithm is a method or a process used to build models from data.**
- **Key characteristics of algorithms include:**
 1. **Procedural:** They specify a sequence of steps to achieve a specific outcome.
 2. **General-purpose:** They can often be applied to different types of data and problems.
 3. **Deterministic:** Given the same input, an algorithm will always produce the same output.
- **Examples in Machine Learning:** Decision Trees, Linear Regression, Support Vector Machines, Gradient Descent, K-Means Clustering.

ML Model

- A **model** is the **output or the representation generated after an algorithm processes data**.
- In machine learning, a **model is the mathematical representation of the learned patterns or relationships within the data**.
- The model is used to **make predictions or decisions** without being **explicitly programmed** to perform the task.

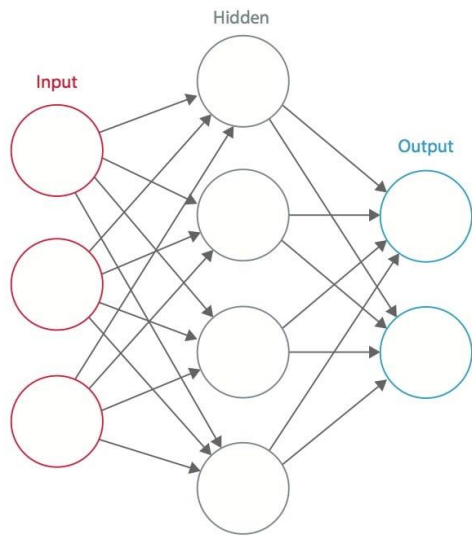
Key characteristics of models include:

- **Predictive:** They can make predictions or decisions based on new data.
- **Learned:** They are built using algorithms that learn from historical data.
- **Specific to Data:** Models are specific to the data they are trained on; different datasets typically produce different models.

Examples in Machine Learning: A trained neural network for image recognition, a regression equation from **linear regression**, a **decision tree classifier**.

Machine Learning Models Examples

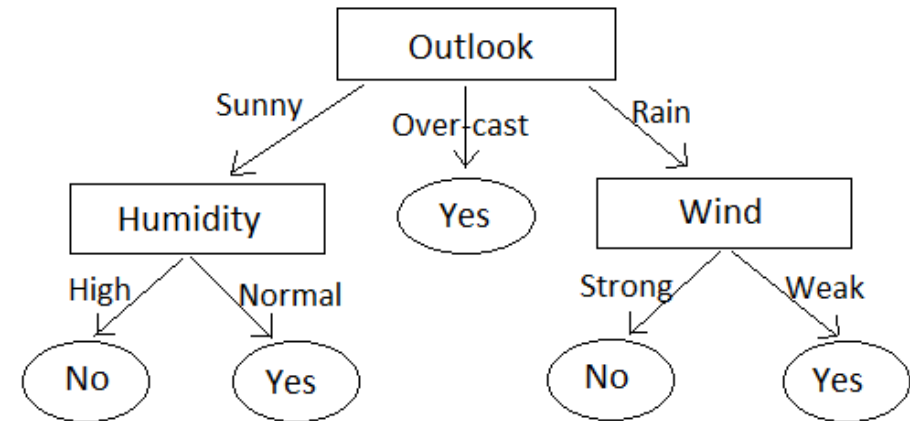
- A **trained neural network** for image recognition,
- A regression equation from **linear regression**,
- A **decision tree classifier**.



$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \varepsilon$$

Labels for the equation components:

- Dependent Variable (Response Variable) points to Y .
- Independent Variables (Predictors) points to X_1 and X_2 .
- Y intercept points to β_0 .
- Slope Coefficient points to β_1 and β_2 .
- Error Term points to ε .



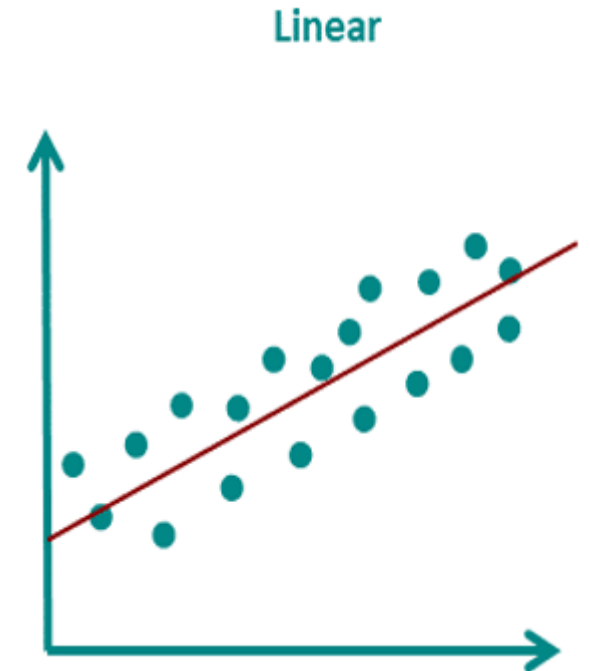
Summary

- An **algorithm** is the method or procedure used to learn from data.
- A **model** is the result or the representation produced by running an algorithm on data.

In essence, algorithms are the **recipes or instructions**, and models are the **final dishes** prepared following those recipes.

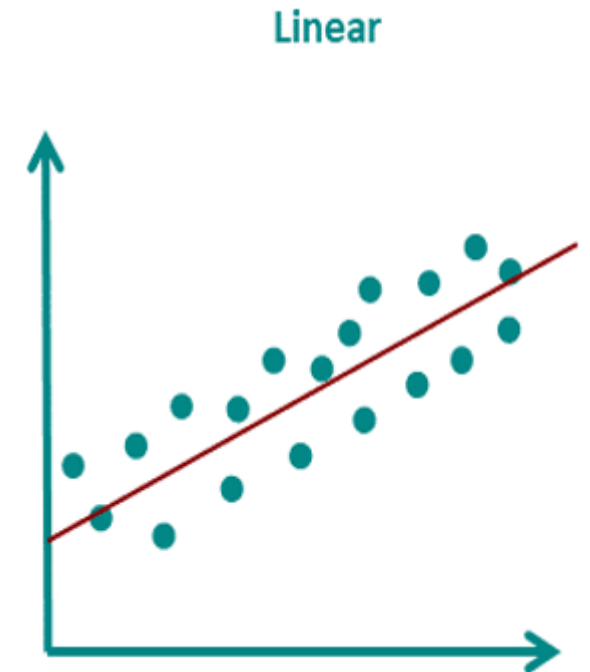
Linear Regression Algorithm:

1. Collect Data
 2. Create A Formula ($y = mx+c$ or $y = m_1x_1+m_2x_2+---+C$)
 3. Find the **best fitting Line** to a set of data points
(Calculate coefficients i.e slope and intercept)
- This algorithm defines the **steps to fit a line to a set of data points by minimizing the sum of the squared differences between the observed and predicted values.**



Linear Regression Model

- **Linear Regression Model:** After running the linear regression algorithm on a specific dataset, you get a model represented by the equation $y=mx+b$, where m is the slope (coefficient) and b is the intercept. This model can then be used to predict y for new values of x .



Example : Linear Regression

- Imagine you're a teacher trying to predict how well your students will do on their final exam based on how many hours they study.
- You have data from past students showing how many hours they studied and what their scores were.
- The **linear regression algorithm** is like a recipe or set of steps you follow to find a formula that best predicts the exam scores based on study hours.

Steps of the Linear Regression Algorithm

- 1. Collect Data:** You have a list of study hours and corresponding exam scores for your past students.
- 2. Create a Formula:** You assume there's a straight-line relationship (linear) between study hours and exam scores. The formula looks like:
 - **Score = $\beta_0 + \beta_1 \times \text{Hours}$ $\Rightarrow y = c + m x$**
 - Here, β_0 is the starting point (**intercept**), and β_1 is how much the score increases for each additional hour of study (**slope**).
- 3. Find the Best Line:** The algorithm helps you find the best values for β_0 and β_1 that make the formula fit your data as closely as possible.
 - This is done by minimizing the differences between the actual scores and the scores predicted by your formula.

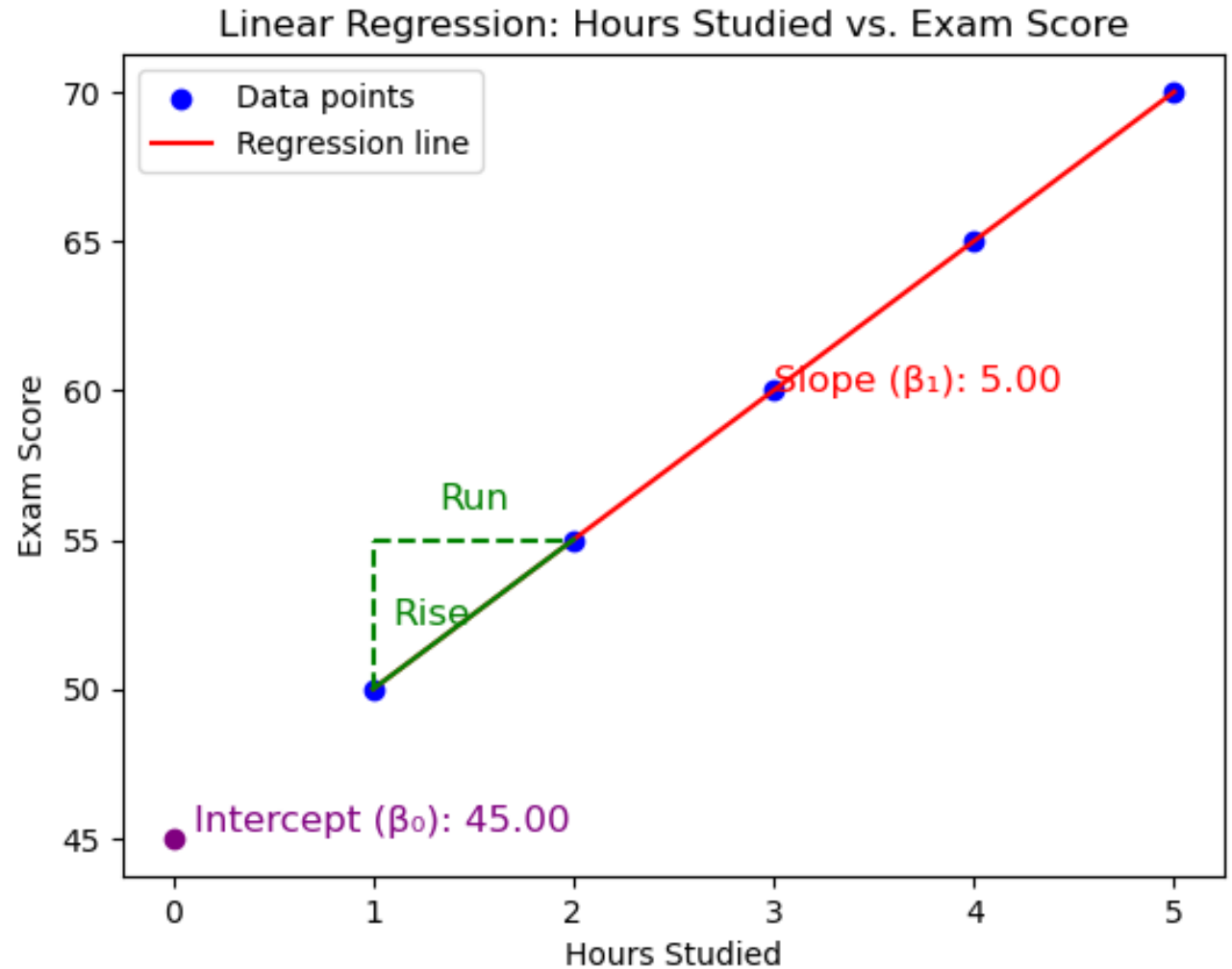
Linear Regression Model

- Once you've used the algorithm to find the best β_0 and β_1 , you get a specific formula.
- **Score = $\beta_0 + \beta_1 \times \text{Hours}$**
- This formula is your **linear regression model**.

Example: Let's say you collected the following data from past students:

Hours Studied	Exam Score
1	50
2	55
3	60
4	65
5	70

Model: $\text{Score} = 45 + 5 \times \text{Hours}$



Types/Classes of Data Science Algorithms

1. Data Engineering Algorithms

- Data munging, preparation, and processing algorithms, such as sorting, MapReduce, or Pregel.

2. Optimization algorithms

3. Machine learning algorithms.

Machine Learning Algorithms

- **Machine learning algorithms** are largely used to **predict, classify, or cluster**.
- **Overlaps with statistical modeling.**
- **Statistical Modeling:** Originates from statistics departments.
- **Machine Learning Algorithms:** Rooted in **computer science departments**.
- Some techniques belong to both fields:
 - **Example: Linear Regression (found in both machine learning and statistics)**

Key Differences and Cultural Approaches

Parameters	Statisticians	Machine Learning Software Engineers
Interpreting Parameters	Real-world interpretations	Focus on predictive power
Confidence Intervals	Use confidence intervals and posterior distributions	Often lack this notion
Explicit Assumptions	Make explicit assumptions about data distributions	Often implicit assumptions
Cultural Approaches	Investigate uncertainty, never 100% confident	Focus on building predictive models , emphasize rapid iteration and fixing issues

Josh Wills' quote

Data scientist (noun):

Person who is better at statistics than any software engineer and better at software engineering than any statistician.

Three Basic Algorithms

1. Linear Regression
2. KNN
3. K Means

1. Linear Regression Algorithm

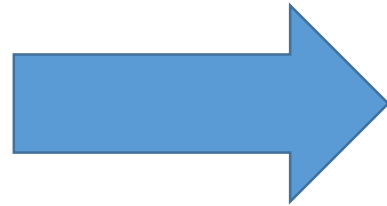
- 1. Collect Data:** Gather data for the dependent and independent variables.
- 2. Visualize Data:** Plot the data to see if there is an apparent linear relationship.
- 3. Fit the Model:** Use the least squares method to find the best-fitting line.
- 4. Evaluate the Model:** Check how well the line fits the data using metrics like Mean Squared Error (MSE).
- 5. Make Predictions:** Use the fitted model to make predictions on new data.

Example1:

- Suppose you **run a social networking** site that charges a monthly subscription **fee of \$25**, and that this is your only source of revenue.
- Each month you collect data and count your number of users and total revenue. You've done this daily over the **course of two years**, recording it all in a spreadsheet. You could express this data as a series of points. Here are the first four:
- $S = \{(x, y) = (1, 25), (10, 250), (100, 2500), (200, 5000)\}$

Equation to represent the relationship of x and y

x	y
1	25
10	250
100	2500
200	5000



$$y=25x$$

Deduce this mentally and identify the following:

- There's a linear pattern.
- The coefficient relating x and y is 25.
- The relationship seems deterministic.

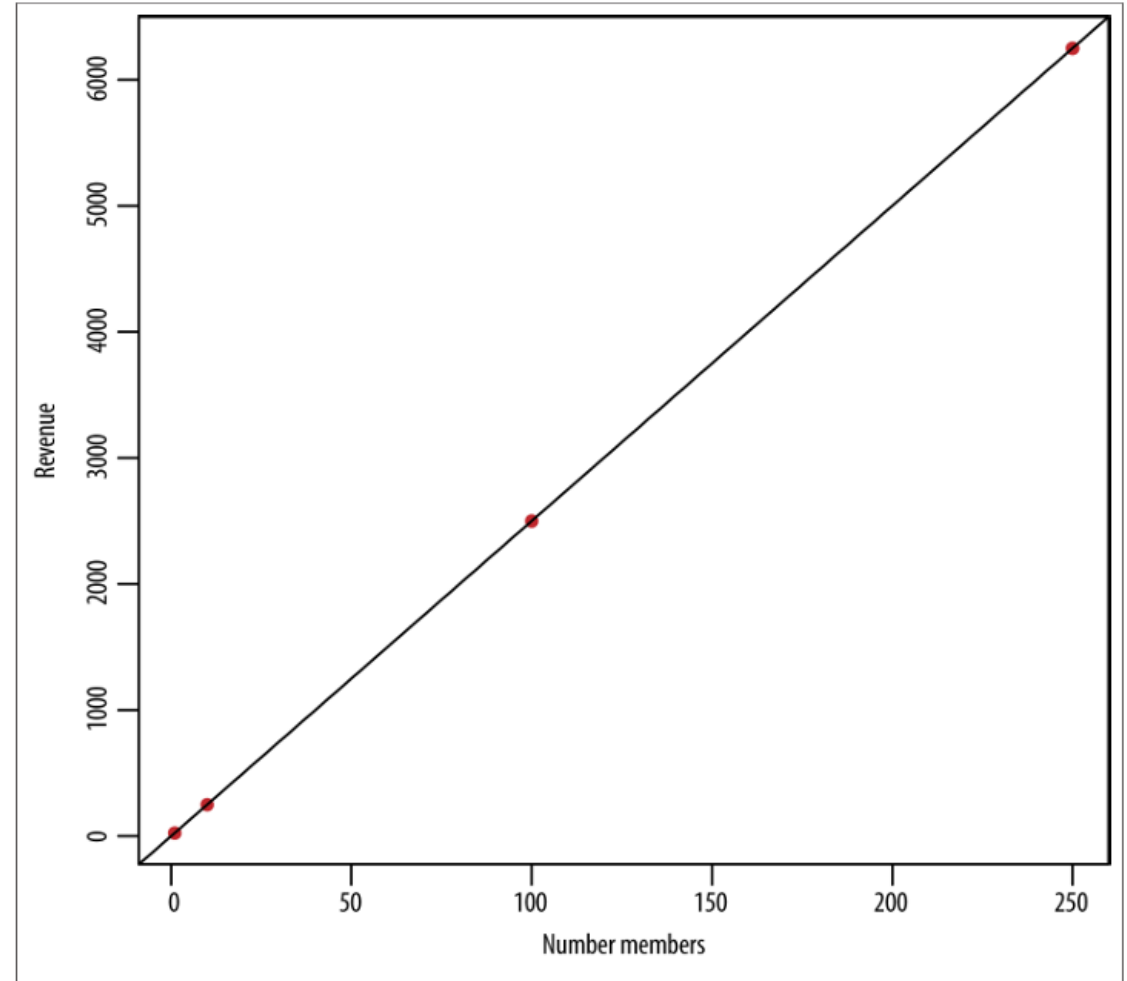


Figure 3-1. An obvious linear pattern

Example2: Sample Data and Plot:

new_friends	time_spent (seconds)
7	276
3	43
4	82
6	136
10	417
9	269

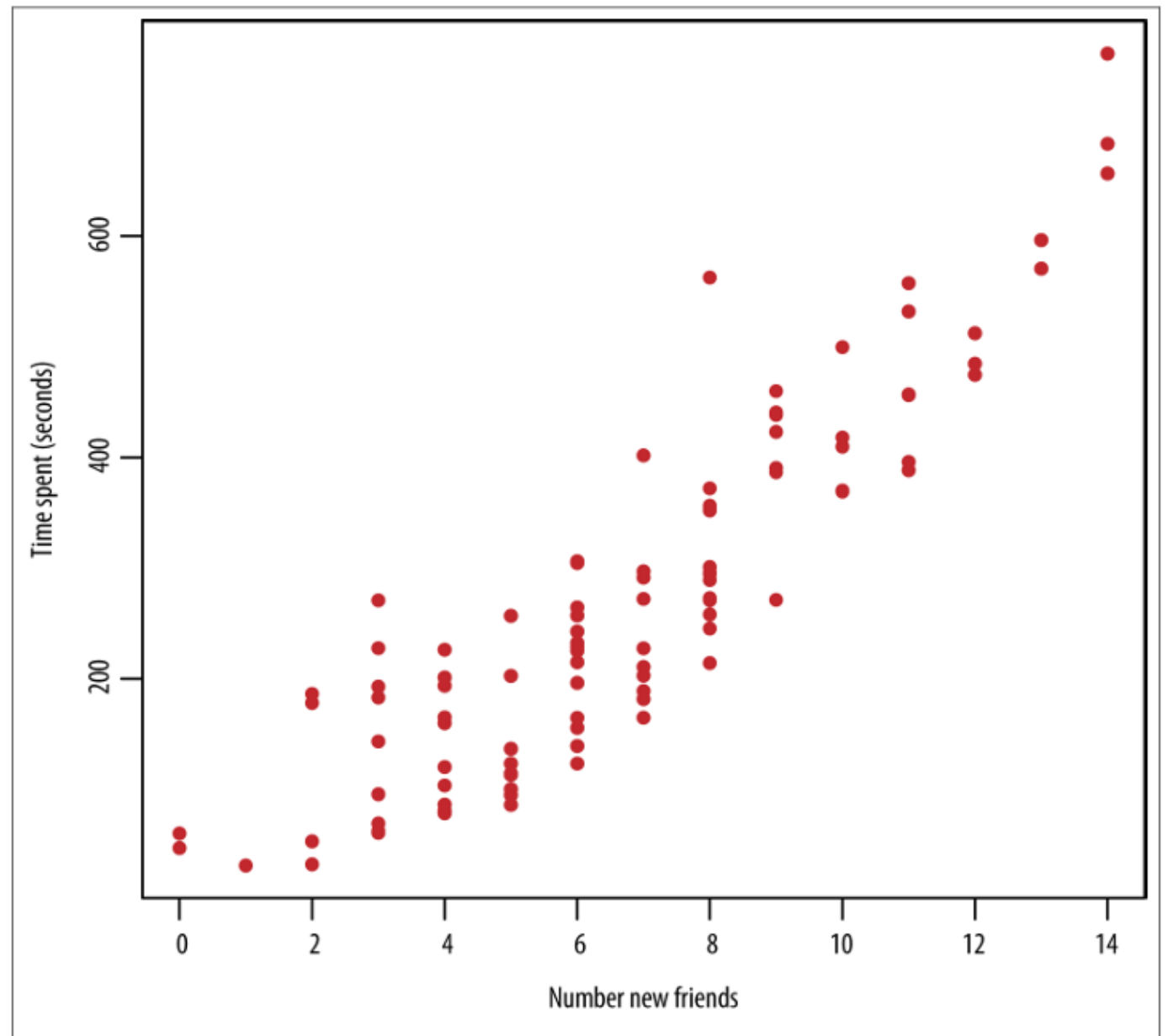


Figure 3-2. Looking kind of linear

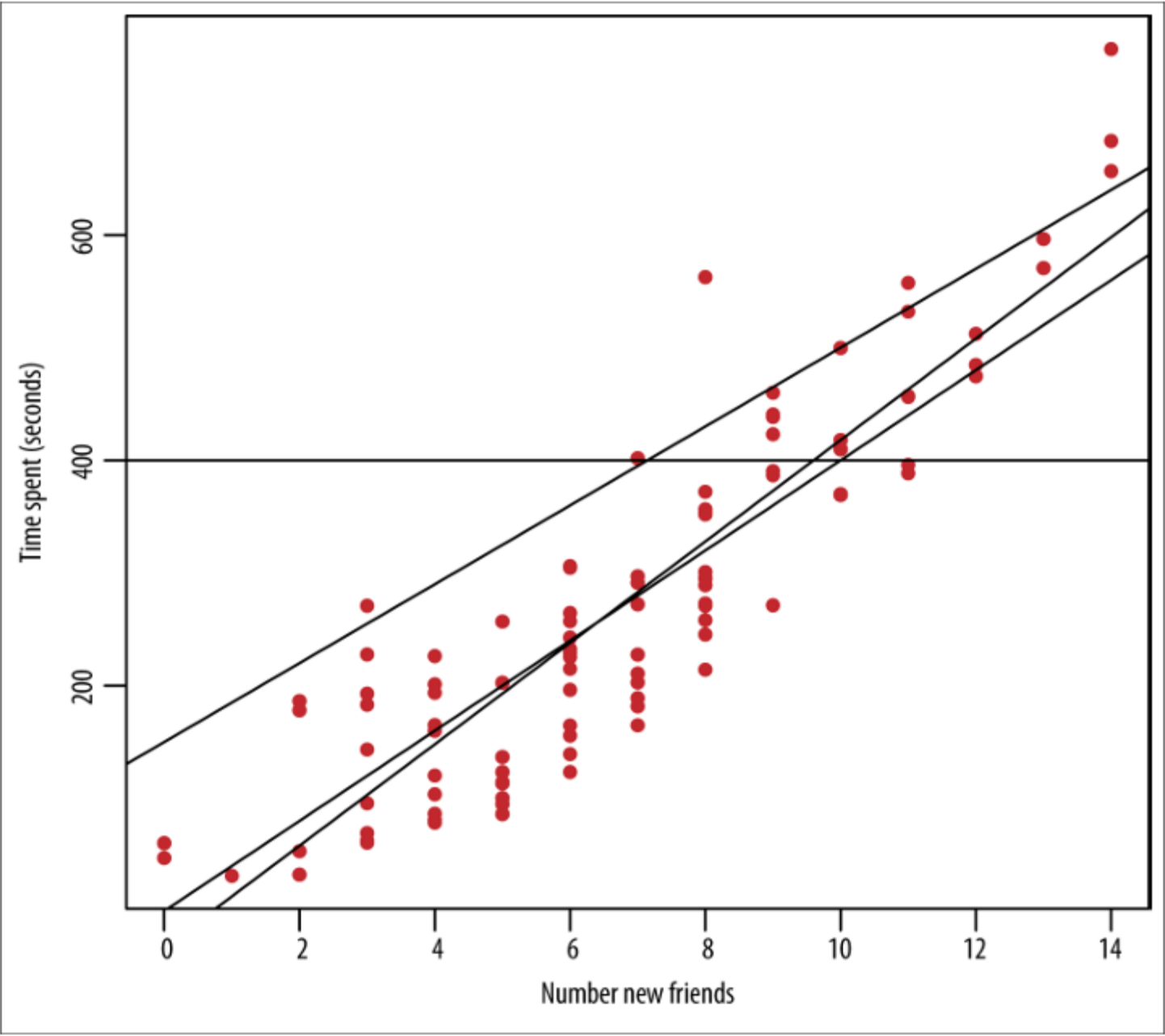


Figure 3-3. Which line is the best fit?

Which Line is the best fit?

- **Assuming a Linear Relationship:**
 - Start with the assumption: $y = \beta_0 + \beta_1 x$,
 - β_0 and β_1 represent the intercept and slope, respectively
- **Estimate the parameters (β_0 and β_1)** using observed data pairs:
 - $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$.

Linear model in matrix notation

- Express the linear model in **matrix notation**: $y=x\cdot\beta$.
- x is the matrix of input variables, and β is the parameter vector.

x	y
2	5
3	7
5	11

$$\begin{pmatrix} 5 \\ 7 \\ 11 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 5 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$$

In this example:

- $y_1=5$, $y_2=7$, and $y_3=11$ are the observed values of the dependent variable.
- $x_1=2$, $x_2=3$, and $x_3=5$ are the values of the independent variable.
- β_0 is the intercept and β_1 is the slope of the linear

Fitting the model

- By applying least squares estimation, Linear regression seeks to find the line that minimizes the **sum of the squares** of the vertical distances between the **approximated or predicted** y_i^{\wedge} s and the **observed** y_i s.
- To find this line, you define the "residual sum of squares" (RSS), denoted as:
- **$RSS(\beta) = \sum_i (y_i - \beta x_i)^2$**
 - where i ranges over the various data points.

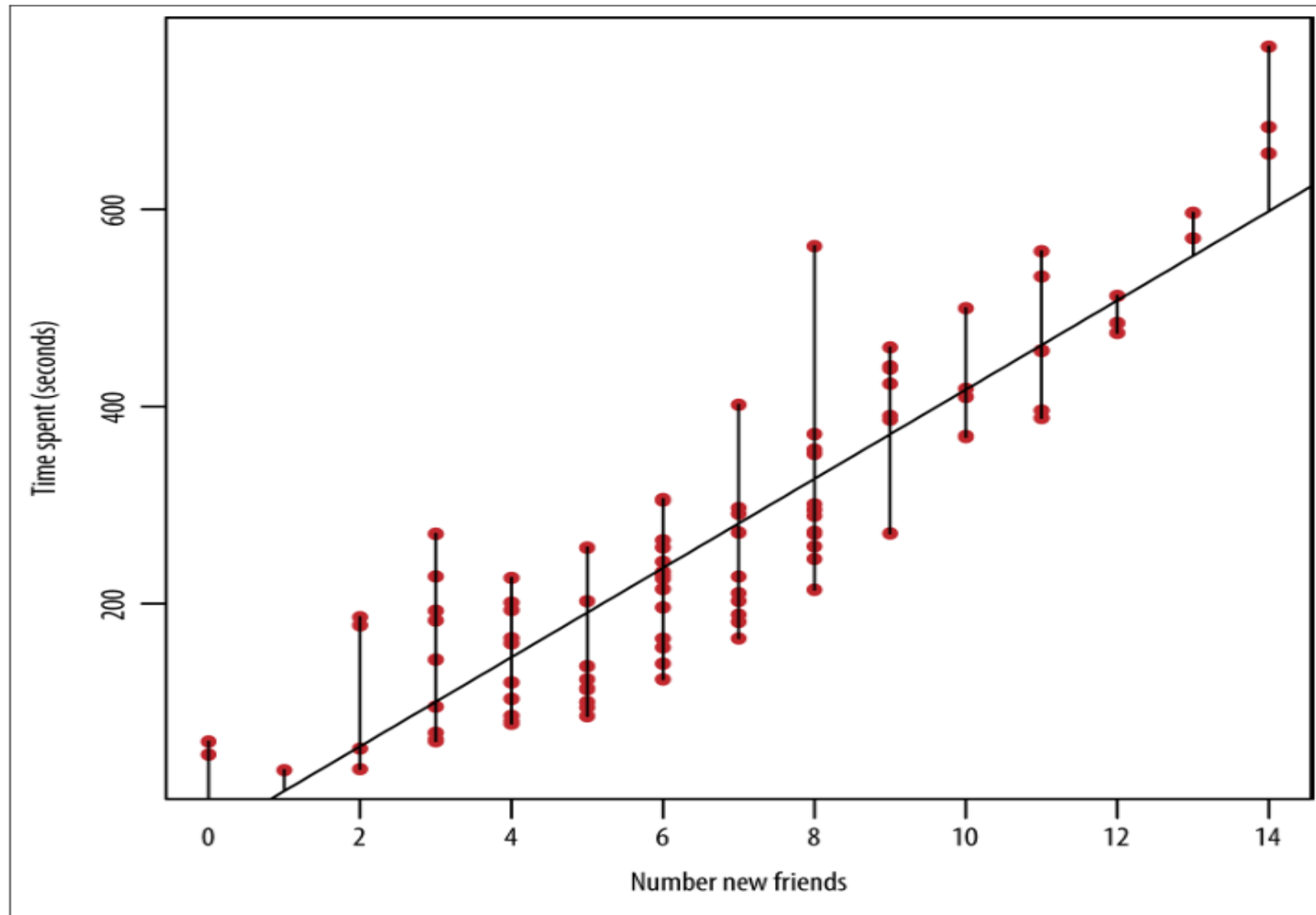


Figure 3-4. The line closest to all the points

Evaluation metrics

- **R-squared (R^2)** : R^2 measures the proportion of the variance in the **dependent variable that is predictable** from the **independent variable(s)**.
- R^2 ranges from **0 to 1**. A higher R^2 value indicates a better fit of the model to the data.

Formula:

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

- $\sum_i (y_i - \hat{y}_i)^2$: Sum of squared residuals (errors between observed and predicted values).
- $\sum_i (y_i - \bar{y})^2$: Total sum of squares (variance of the observed values from their mean).

Evaluation metrics

- **p-values:** The p-value tests the null hypothesis that a coefficient (β) is equal to zero (no effect).
- A low p-value (**< 0.05**) indicates that you can reject the **null hypothesis**, meaning the coefficient is significantly different from zero.

```
summary (model)
```

```
Call:
```

```
lm(formula = y ~ x)
```

```
Residuals:
```

```
      Min       1Q   Median       3Q      Max
-121.17  -52.63   -9.72   41.54  356.27
```

```
Coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  -32.083     16.623   -1.93  0.0565 .
x              45.918      2.141   21.45 <2e-16 ***
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 77.47 on 98 degrees of freedom
```

```
Multiple R-squared: 0.8244,    Adjusted R-squared: 0.8226
```

```
F-statistic: 460 on 1 and 98 DF, p-value: < 2.2e-16
```



Example R Code for the Sample Data

new_friends	time_spent (seconds)
7	276
3	43
4	82
6	136
10	417
9	269

Example of Code in R

```
# Example dataset  
x <- c(7, 3, 4, 6, 10, 9)  
y <- c(276, 43, 82, 136, 417, 269)
```

```
# Fit the linear model
```

```
model <- lm(y ~ x)
```

```
# Print the model summary
```

```
summary(model)
```

Output:

Call:

```
lm(formula = y ~ x)
```

Residuals:

```
 1      2      3      4      5      6  
47.547 11.507  1.267 -43.213 40.827 -57.933
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-116.227	54.792	-2.121	0.10120
x	49.240	7.868	6.258	0.00332

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 '' 1

Residual standard error: 48.18 on 4 degrees of freedom

Multiple R-squared: 0.9073, **Adjusted R-squared:** 0.8842

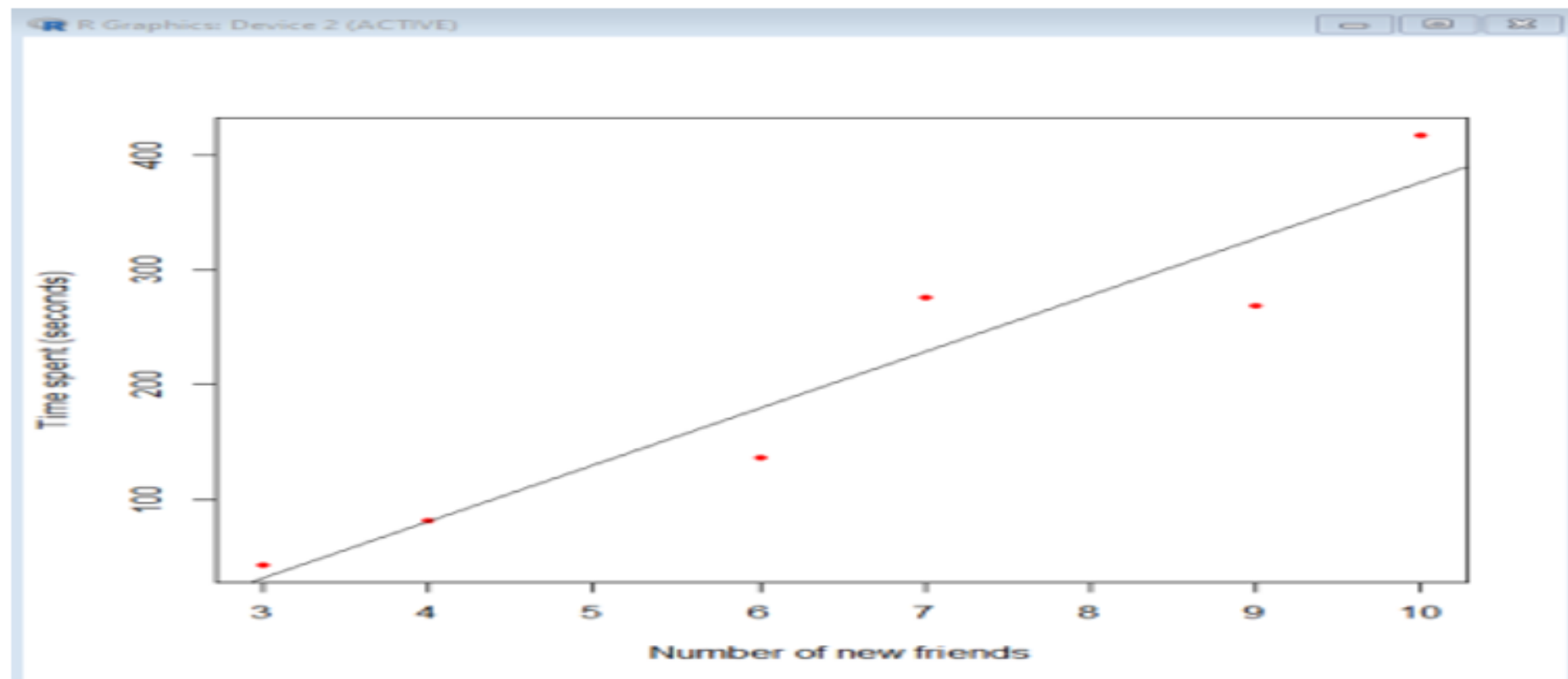
F-statistic: 39.17 on 1 and 4 DF, **p-value:** 0.003325

Plot the data and the fitted line

```
plot(x, y, pch=20, col="red", xlab="Number of new friends", ylab="Time spent  
(seconds)")
```

```
abline(model)
```

Output:



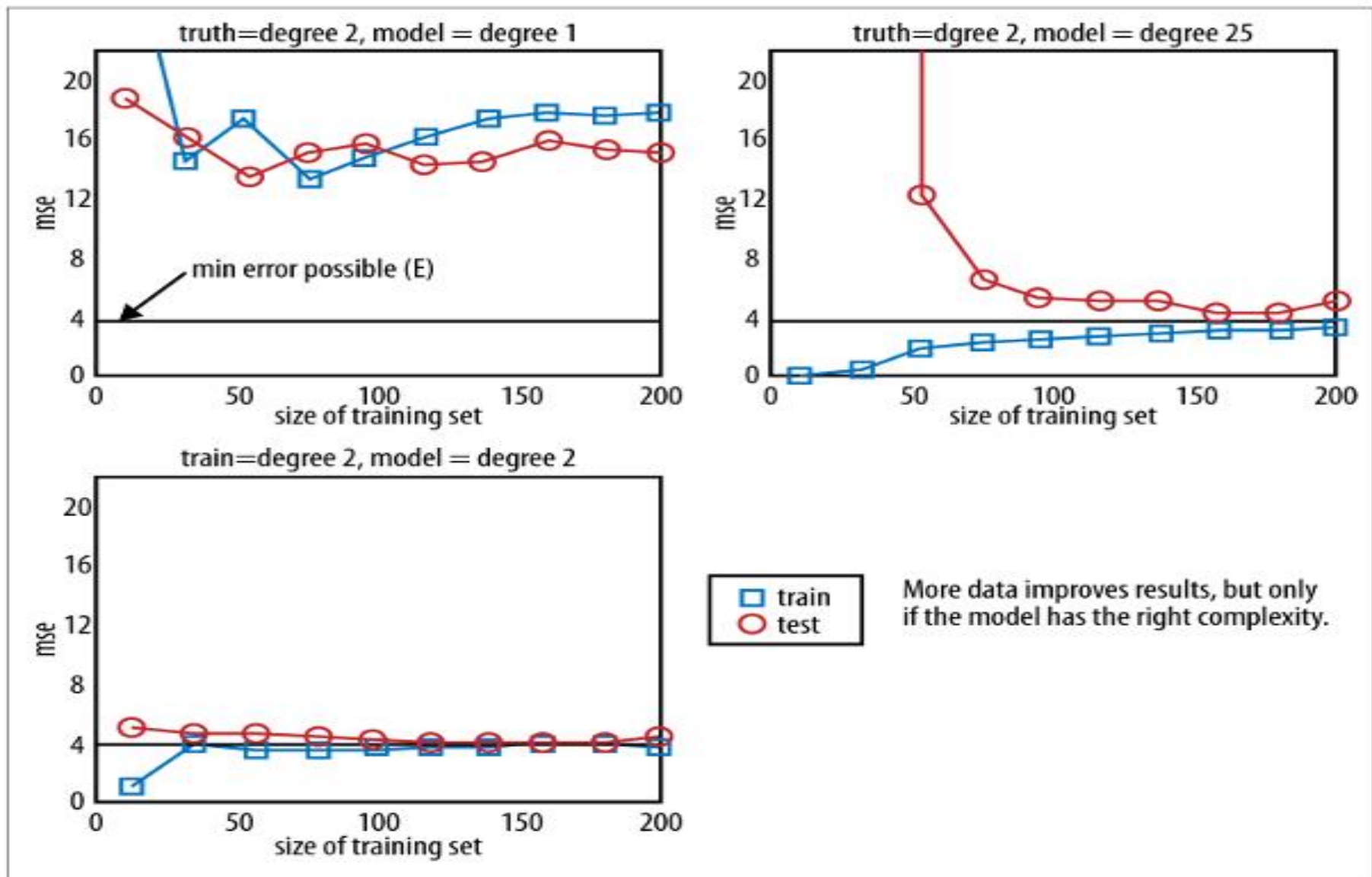


Figure 3-6. Comparing mean squared error in training and test set, taken from a slide of Professor Nando de Freitas; here, the ground truth is known because it came from a dataset with data simulated from a known distribution

2. KNN Algorithm

- The K-Nearest Neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that can be used for both classification and regression tasks.
- It works based on the principle that similar data points (neighbors) are likely to have similar outcomes.

2. KNN Algorithm

1. **Data Preparation:** Collect and prepare the dataset, ensuring it's in a suitable format for analysis.
2. **Choosing K:** Select the number of neighbors (K) to consider for making predictions.
3. **Distance Calculation:** For a given data point that needs to be classified or predicted, calculate the distance between this point and all other points in the dataset. Common distance metrics include Euclidean, Manhattan, and Minkowski distances.
4. **Identify Neighbors:** Identify the K closest data points (neighbors) to the data point in question.
5. **Voting/Prediction:**
 - **Classification:** The class with the majority among the K neighbors is assigned to the data point.
 - **Regression:** The average value of the K neighbors is assigned to the data point.

Key Points:

- **Distance Metrics:** Euclidean distance is most common, but other metrics can be used depending on the problem.
- **Choosing K:** The value of K can significantly impact the algorithm's performance. A common approach is to use cross-validation to find the optimal K.
- **Data Scaling:** KNN is sensitive to the scale of the data. It's important to normalize or standardize the data before applying KNN.

Example Data Set:

X1	X2	Label
1	2	A
2	3	A
3	3	B
6	5	B
7	8	B
8	8	A

We want to predict the label of a new point (4, 4) using KNN with $K=3$.

Step-by-Step Example:

We want to predict the label of a new point (4, 4) using KNN with K=3

1. **Choose K:** We choose K=3.

2. **Calculate Distances:** Compute the Euclidean distance from (4, 4) to all points in the dataset.

- Distance to (1, 2): $\sqrt{(4 - 1)^2 + (4 - 2)^2} = \sqrt{9 + 4} = \sqrt{13} \approx 3.61$
- Distance to (2, 3): $\sqrt{(4 - 2)^2 + (4 - 3)^2} = \sqrt{4 + 1} = \sqrt{5} \approx 2.24$
- Distance to (3, 3): $\sqrt{(4 - 3)^2 + (4 - 3)^2} = \sqrt{1 + 1} = \sqrt{2} \approx 1.41$
- Distance to (6, 5): $\sqrt{(4 - 6)^2 + (4 - 5)^2} = \sqrt{4 + 1} = \sqrt{5} \approx 2.24$
- Distance to (7, 8): $\sqrt{(4 - 7)^2 + (4 - 8)^2} = \sqrt{9 + 16} = \sqrt{25} = 5$
- Distance to (8, 8): $\sqrt{(4 - 8)^2 + (4 - 8)^2} = \sqrt{16 + 16} = \sqrt{32} \approx 5.66$

Step-by-Step Example:

3. **Identify Neighbors:** The 3 closest points to (4, 4) are:

- (3, 3) with distance ≈ 1.41 (Label B)
- (2, 3) with distance ≈ 2.24 (Label A)
- (6, 5) with distance ≈ 2.24 (Label B)

4. **Voting/Prediction:**

- Labels of the 3 closest points: B, A, B
- Majority label is B

Thus, the predicted label for the new point (4, 4) is **B**.

Implementation in Python:

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier

# Example dataset
X = np.array([[1, 2], [2, 3], [3, 3], [6, 5], [7, 8], [8, 8]])
y = np.array(['A', 'A', 'B', 'B', 'B', 'A'])

# New point to classify
new_point = np.array([[4, 4]])


# Create KNN classifier with K=3
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)

# Predict the label for the new point
prediction = knn.predict(new_point)
print(f"The predicted label for point {new_point[0]} is {prediction[0]}")
```

Implementation in Python:

Output:

CSS

 Copy code

```
The predicted label for point [4 4] is B
```

Implementation in R Program

```
# Install and load the required package
install.packages("class")
library(class)

# Example dataset
data <- data.frame(
  X1 = c(1, 2, 3, 6, 7, 8),
  X2 = c(2, 3, 3, 5, 8, 8),
  Label = factor(c('A', 'A', 'B', 'B', 'B', 'A'))
)

# Extract features and labels
features <- data[, 1:2]
labels <- data$Label

# Set the number of neighbors
k <- 3

# Apply KNN
predicted_label <- knn(train = features, test = new_point, cl = labels, k = k)
print(paste("The predicted label for point (4, 4) is", predicted_label))
```

KNN Applications

- **1. Image Classification**
- **2. Recommendation Systems**
- **3. Medical Diagnosis**
- **4. Fraud Detection**
- **5. Customer Segmentation**

Distance/ Similarity Metrics

- When working with various types of data, understanding and defining similarity or distance between data points is crucial.
- Here we discuss several common similarity and distance metrics, each with its unique definition of "closeness."
- These metrics are used in different contexts and types of data to determine how similar or different data points are.

1. Euclidean Distance

- Euclidean distance is the straight-line distance between two points in a plane. It is commonly used for real-valued attributes that can be plotted in multidimensional space.

Formula:

$$d(x, y) = \sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Example:

Consider two points in 2D space, $x = (1, 2)$ and $y = (4, 6)$. The Euclidean distance between these points is:

$$d(x, y) = \sqrt{(1 - 4)^2 + (2 - 6)^2} = \sqrt{9 + 16} = \sqrt{25} = 5$$

2. Cosine Similarity

- Cosine similarity measures the cosine of the angle between two non-zero vectors of an inner product space. It is used to determine how similar two vectors are, yielding a value between -1 (exact opposite) and 1 (exactly the same), with 0 meaning no correlation.

Formula:

$$\cos(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$

Example:

For vectors $x = (1, 0, -1)$ and $y = (-1, 0, 1)$:

$$\cos(x, y) = \frac{(1 \cdot -1) + (0 \cdot 0) + (-1 \cdot 1)}{\sqrt{1^2 + 0^2 + (-1)^2} \cdot \sqrt{(-1)^2 + 0^2 + 1^2}} = \frac{-1 - 1}{\sqrt{2} \cdot \sqrt{2}} = \frac{-2}{2} = -1$$

3. Jaccard Distance/Similarity

- Jaccard similarity measures the **similarity between finite sets**. It is defined as the size of the **intersection divided by the size of the union of the sets**.

Formula:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Example:

For sets $A = \{1, 2, 3\}$ and $B = \{2, 3, 4\}$:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{2}{4} = 0.5$$

4. Mahalanobis Distance

- Mahalanobis distance is a measure of the distance between a **point and a distribution**. It accounts for the **correlations** of the data set and is scale-invariant.

Formula:

$$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$$

where S is the covariance matrix of the data.

Example:

Consider vectors $x = (1, 2)$ and $y = (2, 3)$, and covariance matrix $S = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}$:

$$d(x, y) = \sqrt{(1 - 2, 2 - 3) \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 - 2 \\ 2 - 3 \end{bmatrix}}$$

5. Hamming Distance

- Hamming distance measures the number of positions at which the corresponding symbols differ. It is used for strings of the same length.
- **Example:** For strings "**olive**" and "**ocean**": The Hamming distance is **4** (differences at positions 2, 3, 4, and 5).

6. Manhattan Distance

- Manhattan distance (also known as L1 distance) measures the sum of the absolute differences of their Cartesian coordinates. It is like a taxi driving on a grid of streets.

Formula:

$$d(x, y) = \sum_{i=1}^k |x_i - y_i|$$

Example:

For points $x = (1, 2)$ and $y = (4, 6)$:

$$d(x, y) = |1 - 4| + |2 - 6| = 3 + 4 = 7$$

Custom Distance Metrics

- Custom distance metrics are often created to handle specific needs in datasets that contain mixed types of data (e.g., numerical and categorical).

Scenario

- You are working with a movie dataset that includes the following attributes for each movie:
 - **Budget** (in millions of dollars) - Numerical
 - **Number of actors** - Numerical
 - **Genre** – Categorical
- You need to define a custom distance metric that appropriately handles these mixed data types.

Example Movies

Let's consider two movies:

- **Movie A:**

Budget = \$50 million, Number of Actors = 10, Genre = Action

- **Movie B:**

Budget = \$80 million, Number of Actors = 12, Genre = Comedy

Step 1: Normalize the Numerical Attributes

- Normalization scales the numerical values to a common range, usually $[0, 1]$, to ensure fair contribution to the overall distance.
- For simplicity, assume the following normalization (*based on min-max normalization*):
- **Budget:** min = \$0 million, max = \$100 million
- **Number of Actors:** min = 0, max = 20

Normalized values:

Movie A:

- **Normalized Budget** = $(50 - 0) / (100 - 0) = 0.5$
- **Normalized Number of Actors** = $(10 - 0) / (20 - 0) = 0.5$

Movie B:

- **Normalized Budget** = $(80 - 0) / (100 - 0) = 0.8$
- **Normalized Number of Actors** = $(12 - 0) / (20 - 0) = 0.6$

Step 2: Calculate Normalized Euclidean Distance for Numerical Attributes

$$d_{\text{num}} = \sqrt{(0.5 - 0.8)^2 + (0.5 - 0.6)^2} = \sqrt{0.09 + 0.01} = \sqrt{0.1} \approx 0.316$$

Step 3: Calculate Categorical Distance for Genre

- Since the genres are different (**Action vs. Comedy**), assign a distance of 10:

$$d_{\text{cat}} = 10$$

Step 4: Combine the Distances

- Combine the distances from numerical and categorical attributes. Since the categorical distance is significantly larger, it will dominate unless we scale it appropriately.
- For the purpose of this example, assume we directly add them:
- $d_{\text{total}=d_{\text{num}}+d_{\text{cat}}} = 0.316+10 = 10.316$

Custom Distance Metric Formula

In general, you can define your custom distance metric as follows:

$$d_{\text{custom}}(x, y) = w_{\text{num}} \cdot d_{\text{num}}(x, y) + w_{\text{cat}} \cdot d_{\text{cat}}(x, y)$$

Where:

- $d_{\text{num}}(x, y)$ is the normalized Euclidean distance for numerical attributes.
- $d_{\text{cat}}(x, y)$ is the categorical distance.
- w_{num} and w_{cat} are weights that balance the contributions of numerical and categorical distances.

Adjusting these weights allows fine-tuning the distance metric to best fit the specific needs of your data and application.

Training and Test Sets

- When you are working with a machine learning algorithm, the general process involves two main phases: training and testing.
- Here's a simple explanation of each phase:

Training Phase

- **Purpose:** To create and train a model using known data.
- **Process:**
 - You use a portion of your data, **called the training set**, where the outcomes (or labels) are already known.
 - The **model learns** from this data. For example, in a dataset of people with their ages, incomes, and credit scores labeled as "high" or "low," the model learns the patterns and relationships between these attributes.
 - In the **k-NN (k-Nearest Neighbors) algorithm**, the training phase involves simply reading in the data with the known labels (e.g., "high" or "low" credit).

Testing Phase

- **Purpose:** To evaluate how well the trained model performs on new, unseen data.
- **Process:**
 - You use a different portion of your data, called the **test set**, which was not used during training.
 - In the test set, you pretend that you don't know the outcomes (or labels). The model makes predictions based on what it learned during training.
 - You then compare the model's predictions to the actual known outcomes in the test set to see how accurate the model is.

Example with Steps

- Consider you have a dataset with 1,000 rows, each containing information about age, income, and credit score (either "high" or "low").
- Here's how you could split this data into training and test sets and use them:

1. Load the Data:

```
> head(data)
```

```
  age income credit
```

```
1 69    79  low
```

```
2 66    17  low
```

```
3 49    26  low
```

```
4 49    71  low
```

```
5 58    57  high
```

```
6 44    79  high
```

2. Define the Number of Rows and Sampling Rate:

- `n.points <- 1000` # Total number of rows in the dataset
- `sampling.rate <- 0.8` # 80% for training, 20% for testing

3. Calculate the Number of Test Set Labels:

- `num.test.set.labels <- n.points * (1 - sampling.rate)` # 20% of the data

4. Randomly Sample Training Set Rows:

```
training <- sample(1:n.points, sampling.rate * n.points, replace=FALSE)
```

5. Create Training Set:

- `train <- subset(data[training,], select = c(age, income))`
- `cl <- data$credit[training] # Labels for the training set`

6. Define Test Set Rows and Create Test Set:

- `testing <- setdiff(1:n.points, training)`
- `test <- subset(data[testing,], select = c(age, income))`
- `true.labels <- data$credit[testing] # True labels for the test set`

How It Works

- **Training Set:** 80% of the data (randomly selected) used to train the model. This set includes the "**credit**" labels.
- **Test Set:** The remaining 20% of the data used to test the model. This set does not provide the "**credit**" labels to the model during testing but keeps them for evaluating the model's performance.
- By comparing the model's predictions on the test set with the true labels (which are known but not used during prediction), you can assess how well your model is likely to perform on new, unseen data.

Picking an Evaluation Metric:

- **Customizing Evaluation Metrics**
- **Sensitivity and Specificity**
 - Sensitivity (True Positive Rate)
 - Specificity (True Negative Rate)
- **Precision and Recall**
 - Precision
 - Recall
- **Accuracy and Misclassification Rate**

1. Sensitivity (True Positive Rate or Recall):

$$\text{Sensitivity} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Sensitivity measures the proportion of actual positives that are correctly identified.

2. Specificity (True Negative Rate):

$$\text{Specificity} = \frac{\text{True Negatives (TN)}}{\text{True Negatives (TN)} + \text{False Positives (FP)}}$$

Specificity measures the proportion of actual negatives that are correctly identified.

3. Precision (Positive Predictive Value):

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}}$$

Precision measures the proportion of positive identifications that are actually correct.

4. Recall:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}}$$

Recall is another term for sensitivity, measuring the proportion of actual positives that are correctly identified.

5. **Accuracy:**

$$\text{Accuracy} = \frac{\text{True Positives (TP)} + \text{True Negatives (TN)}}{\text{Total Population}}$$

Accuracy measures the proportion of true results (both true positives and true negatives) among the total number of cases examined.

6. **Misclassification Rate:**

$$\text{Misclassification Rate} = \frac{\text{False Positives (FP)} + \text{False Negatives (FN)}}{\text{Total Population}}$$

Misclassification rate is the proportion of incorrect predictions (both false positives and false negatives) among the total number of cases examined.

Alternatively, misclassification rate can be expressed as:

$$\text{Misclassification Rate} = 1 - \text{Accuracy}$$

Summary of the Metrics

- **Sensitivity (Recall):** Focuses on correctly identifying positive cases.
- **Specificity:** Focuses on correctly identifying negative cases.
- **Precision:** Focuses on the accuracy of positive predictions.
- **Recall:** Focuses on the ability to find all relevant positive cases.
- **Accuracy:** Overall correctness of the model.
- **Misclassification Rate:** Overall incorrectness of the model.

Modeling Assumptions in k-NN Algorithm

- **Feature Space and Distance:** The data exists in a feature space where a notion of "distance" between data points is meaningful. This distance is crucial for determining the neighbors of a point.
- **Labeled Training Data:** The training data has **labels or classifications** for two or more classes. This labeling is necessary for the algorithm to make predictions about new, unlabeled data points.
- **Choosing the Number of Neighbors (k):** You must select the number of neighbors (k) to use in the algorithm. The choice of k can significantly impact the performance of the model.
- **Association Between Features and Labels:** There is an implicit assumption that the **observed features and the labels are somehow related**.

3.K Means Algorithm

- The K-Means algorithm is a popular clustering technique used to partition a dataset into K clusters.
- The goal is to minimize the variance within each cluster, thereby ensuring that data points within a cluster are as similar as possible.

Steps of K-Means Algorithm

1. Initialize Centroids:

- Randomly select K **initial centroids** from the dataset. These centroids will represent the initial cluster centers.

2. Assign Data Points to Clusters:

- For each data point, calculate the **Euclidean distance** to each centroid.
- Assign each data point to the cluster with the nearest centroid.

3. Update Centroids:

- For each cluster, calculate the new centroid by taking the mean of all data points assigned to that cluster.

4. Repeat:

- Repeat steps 2 and 3 until the centroids no longer change significantly, or for a fixed number of iterations.

5. Convergence:

- The algorithm converges when the assignments of data points to clusters no longer change.

Example Dataset

- Let's consider a simple 2D dataset for illustration:

Data Point	X	Y
A	1	2
B	1	4
C	3	4
D	5	2
E	5	4

Step1: Initialize Centroids

- Suppose we choose $K=2$ (we want to partition the data into 2 clusters).
- Randomly select two initial centroids. For simplicity, let's choose points **A (1, 2)** and **D (5, 2)**.

Step2: Assign Data Points to Clusters

- Calculate distances and assign points to the nearest centroid.

Data Point	Distance to (1, 2)	Distance to (5, 2)	Assigned Cluster
A (1, 2)	0	4	1
B (1, 4)	2	4.47	1
C (3, 4)	2.83	3.61	1
D (5, 2)	4	0	2
E (5, 4)	4.47	2	2

3. Update Centroids: Calculate the new centroids for each cluster.

Cluster 1: (A, B, C)

- New Centroid = $((1+1+3)/3, (2+4+4)/3) = (1.67, 3.33)$

Cluster 2: (D, E)

- New Centroid = $((5+5)/2, (2+4)/2) = (5, 3)$

Step 4: Repeat

- Reassign data points to the new centroids and update centroids again if necessary.

Data Point	Distance to (1.67, 3.33)	Distance to (5, 3)	Assigned Cluster
A (1, 2)	1.86	4.12	1
B (1, 4)	1.05	4.12	1
C (3, 4)	1.67	2.24	1
D (5, 2)	3.61	1	2
E (5, 4)	3.81	1	2

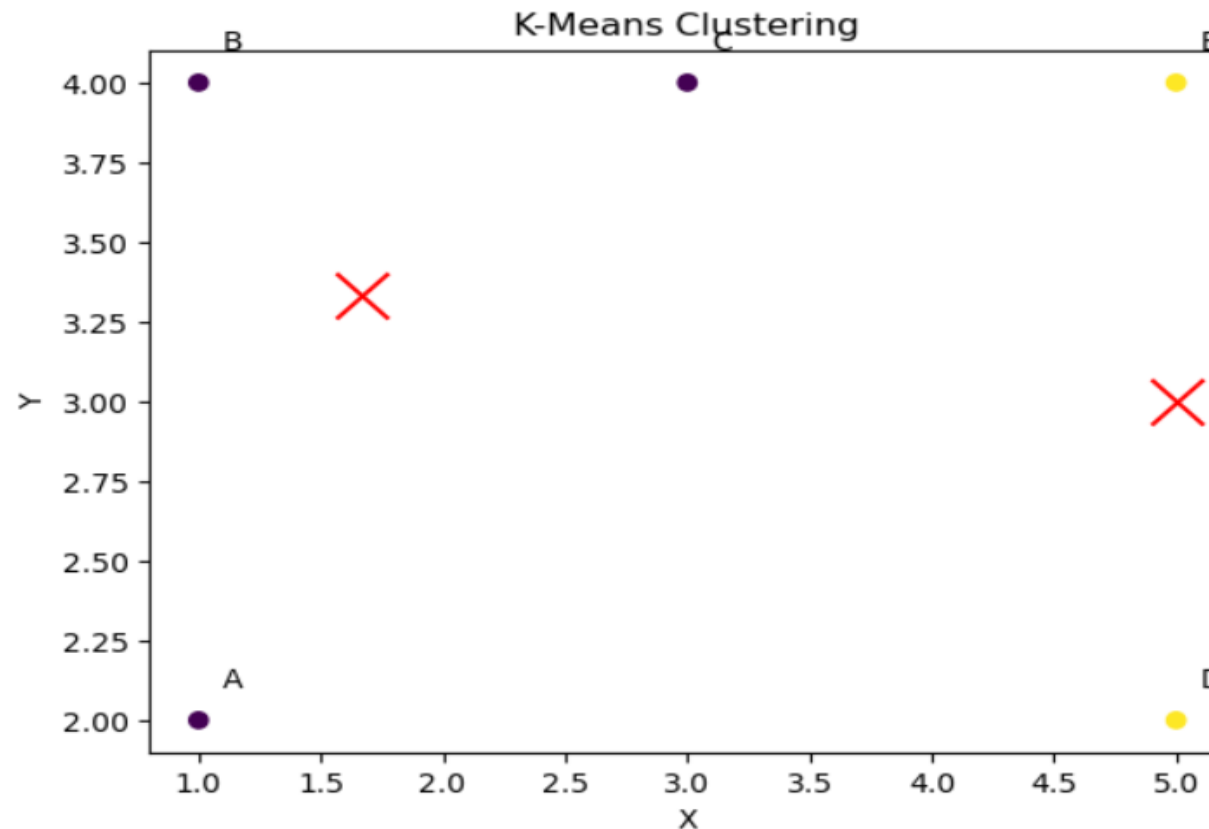
New centroids remain the same as above: Cluster 1 (1.67, 3.33), Cluster 2 (5, 3)

Step5: Convergence

- The algorithm converges when the centroids no longer change.
- In this example, the clusters have stabilized after the second iteration.

Step6: Visual Representation

- The final clusters can be visualized as follows:
- **Cluster 1:** Points A (1, 2), B (1, 4), C (3, 4) with centroid (1.67, 3.33)
- **Cluster 2:** Points D (5, 2), E (5, 4) with centroid (5, 3)



k-means has some known issues

- **Choosing k is more an art than a science**, although there are bounds: $1 \leq k \leq n$, where n is number of data points.
- **There are convergence issues**—the solution can fail to exist, if the algorithm falls into a loop, for example, and keeps going back and forth between two possible solutions, or in other words, there isn't a single unique solution.
- **Interpretability can be a problem**—sometimes the answer isn't at all useful. Indeed that's often the biggest problem.

Exercise: Basic Machine Learning Algorithms

- **Consider the NYC (Manhattan) Housing dataset and perform the following using R program:**
 1. Analyze sales using regression with any predictors you feel are relevant. Justify why regression was appropriate to use.
 2. Visualize the coefficients and fitted model.
 3. Predict the neighborhood using a k-NN classifier. Be sure to with hold a subset of the data for testing. Find the variables and the k that give you the lowest prediction error.
 4. Report and visualize your findings.
 5. Describe any decisions that could be made or actions that could be taken from this analysis.