Reproducible Analysis in R Studio

By

Dr. Blessing Ogbuokiri

(ogbuokiriblessing@gmail.com)

Content

- What is Reproducible Analysis
- Why is it important
- Advantages
- Making analysis reproducible
- Tools
- Why use knitr and rmarkdown
- Difference btw knitr and rmarkdown
- Markdown
- Writing Markdown file
- Literate programming
- Rmarkdown
- R Machine Learning Basics

What is Reproducible Analysis

- A data analysis is reproducible if all the information (data, files, etc.) required is available for someone else to re-do your entire analysis. This includes:
    - Data available
    - All code for cleaning raw data
    - All code and software (specific versions, packages) for analysis

Why Should Analysis Be Reproducible?

Two main reasons for Reproducible Analysis:

- For Science:
  - Standard to judge scientific claims
  - Reproducibility enhances replicability
  - Avoiding effort duplication & encouraging cumulative knowledge development

- For You:
  - Better work habits
  - Better teamwork
  - Changes are easier
  - Higher research impact

Some advantages of making your research reproducible are:

- You can (easily) figure out what you did six months from now.
- You can (easily) make adjustments to code or data, even early in the process, and re-run all analysis.
- When you're ready to publish, you can (easily) do a last double-check of your full analysis, from cleaning the raw data through generating figures and tables for the paper.
- You can pass along or share a project with others.
- You can give useful code examples to people who want to extend your research.

Some Steps To Making Analysis Reproducible

- All your raw data should be saved in the project directory. You should have clear documentation on the source of all this data.
- Scripts should be included with all the code used to clean this data into the dataset(s) used for final analyses and to create any figures and tables.
- You should include details on the versions of any software used in analysis (for R, this includes the version of R as well as versions of all packages used).
- If possible, there should be no “by hand” steps used in the analysis; instead, all steps should be done using code saved in scripts. For example, you should use a script to clean data, rather than cleaning it by hand in Excel. If any “non-scriptable” steps are unavoidable, you should document those steps very clearly .

Tools of Reproducible Analysis

Below are various tools of reproducible analysis:

- R
- Markup languages
- Rstudio
- Cloud storage & versioning
- Unix-like shell programs
- knitr and rmarkdown

We will concentrate on knitr and rmarkdown for reproducible analysis

Why Use knitr and rmarkdown for Reproducible Research

- Both *knitr* and *rmarkdown* can work with markup languages other than LaTeX including Markdown and HTML.
- *rmarkdown* can even output Microsoft Word documents.
- They can work with programming languages other than R.
- They highlight R code in presentation documents making it easier for your readers to follow.
- They give you better control over the inclusion of graphics and can cache code chunks, i.e. save the output for later.
- *knitr* and *rmarkdown* have broadly similar capabilities and syntax.
- They
- Both are literate programming tools that can produce presentation documents from multiple markup languages.

Difference Between knitr and rmarkdown

- Their main difference is that they take different approaches to creating presentation documents.
- knitr documents must be written using the markup language associated with the desired output. For example, with knitr, LaTeX must be used to create PDF output documents and Markdown or HTML must be used to create webpages.
- rmarkdown builds directly on knitr, the key difference being that it uses the straightforward Markdown markup language to generate PDF, HTML, and MS Word documents.
- rmarkdown is generally easier to use.

Markdown

- R Markdown files are mostly written using Markdown.

- To write R Markdown files, you need to understand what markup languages like Markdown are and how they work.

Examples of markup languages include:

- HTML (HyperText Markup Language)
- LaTex
- Markdown (a “lightweight” markup language)

Writing a Markdown File

- To write a file in Markdown, you'll need to learn the conventions for creating formatting. This table shows what you would need to write in a flat file for some common formatting choices:

Code	Rendering	Explanation
text	**text**	boldface
text	*text*	italicized
[text](www.google.com)	text	hyperlink
# text		first-level header
## text		second-level header

Some other simple things you can do in Markdown include:

- Lists (ordered or bulleted)
- Equations
- Tables
- Figures from file
- Block quotes
- Superscripts

For more Markdown conventions, see RStudio's R Markdown Reference Guide (link also available through "Help" in RStudio).

Literate programming

- Literate programming mixes code that can be executed with regular text.
- The files you create can then be rendered, to run any embedded code.
- The final output will have results from your code and the regular text.

Literate programming with knitr

- The knitr package can be used for literate programming in R.

- In essence, knitr allows you to write an R Markdown file that can be rendered into a pdf, Word, or HTML document.

Basics

- To open a new RMarkdown file, go to “File” -> “New File” ->
- “RMarkdown. . . ” -> for now, chose a “Document” in “HTML” format.
- This will open a new R Markdown file in RStudio. The file extension for RMarkdown files is “.Rmd”.
- The new file comes with some example code and text.
- You can run the file as-is to try out the example.
- You will ultimately delete this example code and text and replace it with your own.
- Once you “knit” the R Markdown file, R will render an HTML file with the output. This is automatically saved in the same directory where you saved your .Rmd file.
- Write everything besides R code using Markdown syntax.

Chunk syntax

- To include R code in an RMarkdown document, separate the code chunk
- using the following syntax:

```
---
title: “First Rmarkdown Example
author: "Dr Man B"
date: "11/05/2021"
output: html_document
---
```

```
```{r}
my_vec <- 1:10

```
```

Name Chunks

- You can specify a name for each chunk, if you'd like, by including it after
- "r" when you begin your chunk.
- For example, to give the name load_nepali to a code chunk that loads
- the nepali dataset, specify that name in the start of the code chunk:

````
```{r load_nepali}
library(faraway)
data(nepali)
```
````

Some tips:

- Chunk names must be unique across a document.
- Any chunks you don't name are given numbers by knitr.

Name Chunks

- You do not have to name each chunk. However, there are some advantages:

- It will be easier to find any errors.
- You can use the chunk labels in referencing for figure labels.
- You can reference chunks later by name.

Chunk Options

- You can add options when you start a chunk. Many of these options can be set as TRUE / FALSE and include:

Option Action	**Option Action**
echo	Print out the R code?
eval	Run the R code?
messages	Print out messages?
warnings	Print out warnings?
include	If FALSE, run code, but don't print code or results

Chunk Options

- Other chunk options take values other than TRUE / FALSE.
- Some you might want to include are:

Option	**Action**
results	How to print results (e.g., hide runs the code, but doesn't print the results)
fig.width	Width to print your figure, in inches (e.g., fig.width = 4)
fig.height	Height to print your figure

Chunk Options

- Add these options in the opening brackets and separate multiple ones with commas:

````
```{r messages = FALSE, echo = FALSE}
nepali[1, 1:3]
```
````

- We will go over other options later, once you've gotten the chance to try adding R code into RMarkdown files.

Global Options

- You can set “global” options at the beginning of the document. This will create new defaults for all of the chunks in the document.

- For example, if you want echo, warning, and message to be FALSE by default in all code chunks, you can run:

````
```{r global_options}
knitr::opts_chunk$set(echo = FALSE, message = FALSE,
warning = FALSE)
```
````

Global Options

Options that you set specifically for a chunk will take precedence over global options.

For example, running a document with:

````
```{r global_options}
knitr::opts_chunk$set(echo = FALSE, message = FALSE,
warning = FALSE)
```


```{r check_nepali, echo = TRUE}
head(nepali, 1)
```
````

would print the code for the check_nepali chunk.

Inline code

- You can also include R output directly in your text (“inline”) using backticks:

There are `r nrow(nepali)` observations in the nepali data set. The average age is `r mean(nepali$age, na.rm = TRUE)` months.

Once the file is rendered, this gives:

- There are 1000 observations in the nepali data set. The average age is 37.662 months.

Rmarkdown Example: Reading and Analysing a CSV file

Reading a CSV File

- Following is a simple example of read.csv() function to read a CSV file available in your current working directory using Rmarkdown


```
```{r}
data <- read.csv("input.csv")
print(data)

```
```

Rmarkdown Example: Analysing a CSV file

- By default the read.csv() function gives the output as a data frame. This can be easily checked as follows. Also we can check the number of columns and rows.

Create a data frame

```
```{r}
print(is.data.frame(data))

```
```

print the number of rows

```
```{r}
print(nrow(data))

```
```

prints the number of columns

```
```{r}
print(ncol(data))
```
```

Rmarkdown Example: Dataset Analysis

Get the max salary from data frame.

```{r}
sal <- max(data$salary)
print(sal)
```

Get all the people working in IT department

```{r}
retval <- subset( data, dept == "IT")
print(retval)
```

Get the details of the person with max salary
We can fetch rows meeting specific filter criteria similar to a SQL where clause.

```{r}
retval <- subset(data, salary == max(salary))
print(retval)

```

Get the persons in IT department whose salary is greater than 600

```{r}
info <- subset(data, salary > 600 & dept == "IT")
print(info)
```

Rmarkdown Example: Reporting

Get the people who joined on or after 2014

```{r}
retval <- subset(data, as.Date(start_date) >
as.Date("2014-01-01"))
print(retval)
```

```{r}
#Following is the description of the parameters used
#v is a vector containing the numeric values.
#type takes the value "p" to draw only the points, "l" to draw
only the lines and "o" to draw both points and lines.
#xlab is the label for x axis.
#ylab is the label for y axis.
#main is the Title of the chart.
#col is used to give colors to both the points and lines.

# Create the data for the chart.
v <- c(7,12,28,3,41)

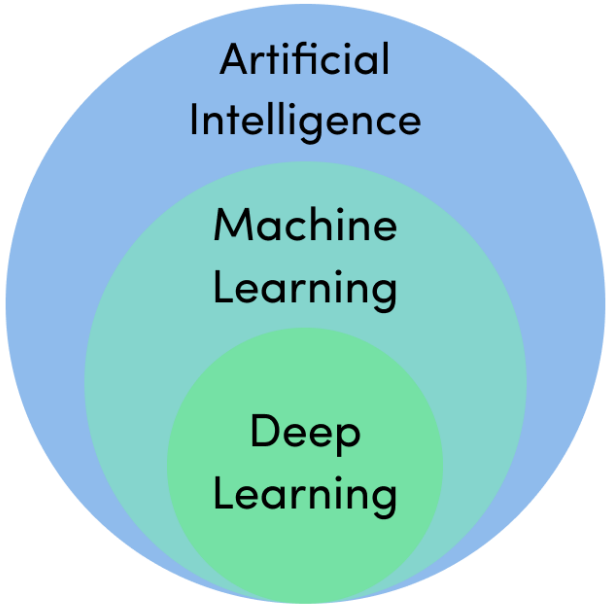
# Give the chart file a name.
#png(file = "line_chart_label_colored.jpg")

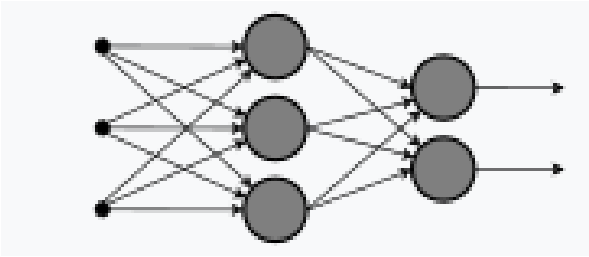
# Plot the bar chart.
plot(v,type = "o", col = "red", xlab = "Month", ylab = "Rain fall",
main = "Rain fall chart")
```

R Machine Learning

- Machine learning is an application of AI that includes algorithms that parse data, learn from that data, and then apply what they've learned to make informed decisions.

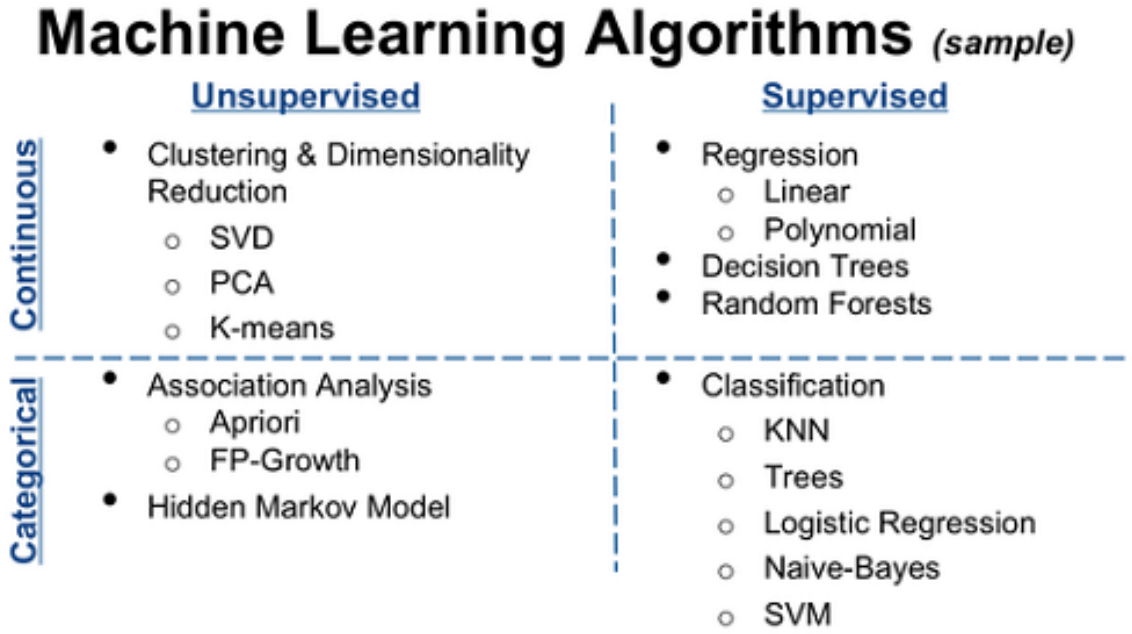
- Deep learning is a subfield of machine learning that structures algorithms in layers to create an "artificial neural network" that can learn and make intelligent decisions on its own.





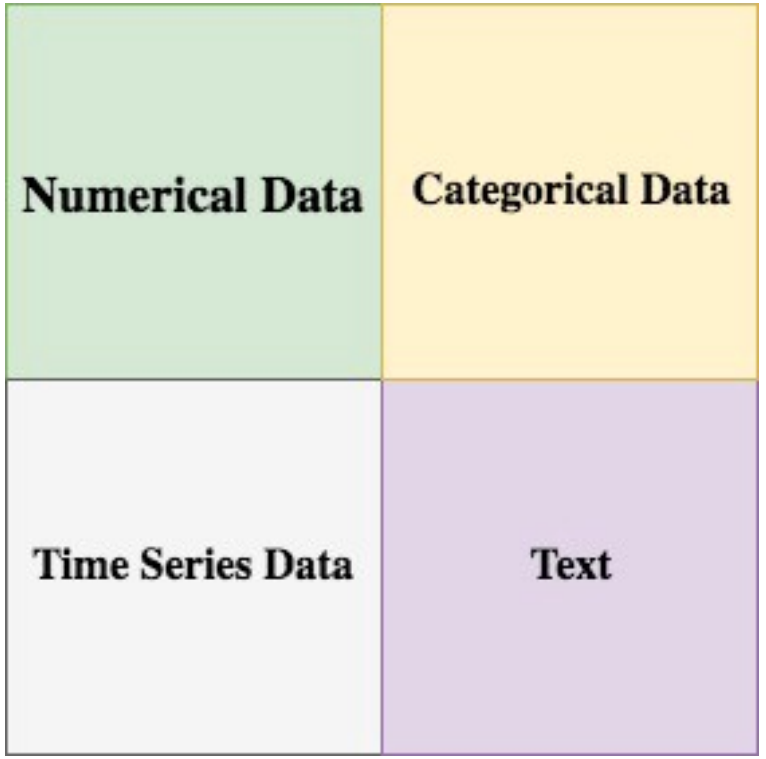
Types of Learning

- **Supervised learning** is a machine learning approach that's defined by its use of labeled datasets. These datasets are designed to train or "supervise" algorithms into classifying data or predicting outcomes accurately. Using labeled inputs and outputs, the model can measure its accuracy and learn over time.
- **Unsupervised learning** uses machine learning algorithms to analyze and cluster unlabeled data sets. These algorithms discover hidden patterns in data without the need for human intervention (hence, they are "unsupervised").
- Note that there are also **semi-supervised learning** approaches that use labelled data to inform unsupervised learning on the unlabelled data to identify and annotate new classes in the dataset



Data Types for Machine Learning

- **Numerical Data:** Numerical data is any data where data points are exact numbers. Can be Continuous or Discrete
- **Categorical Data:** represents characteristics, such as a hockey player's position, team, hometown.
- **Time Series Data:** a sequence of numbers collected at regular intervals over some period of time. like a date or a timestamp
- **Text:** Text data is basically just words



Data Normalization

- Normalization is a technique often applied as part of data preparation for machine learning.
- The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values.
- For machine learning, every dataset does not require normalization. It is required only when features have different ranges.

Types of data for different machine learning tasks

- Regression
- Classification
- Clustering
- Dimensionality reduction
- Reinforcement learning (Ranking)

Regression

Type of Data	ML Technique	Algorithms	Application
Numerical and Categorical Data	Regression	• Simple linear regression • Multiple linear regression • Polynomial linear regression • Decision tree • Radom forest • Support Vector machine (SVM)	• Risk assessment • Score prediction • Market forecasting • Whether forecasting • Housing and product price prediction • Digital personal Assistance (Siri and YouTube) • For better understanding, see below an image.

Classification

Type of Data	ML Technique	Algorithms	Application
Time Series Data and Data	Classification	• Logistic regression • KNN (k-nearest Neighbor) • Naive-Base • Discriminant Analysis • SVM (support vector machine) • Decision tree • Neural network	• Image classificatio n • Email spam detection • Fraudulent detection

Clustering

Type of Data	ML Technique	Algorithms	Application
Text, Time Series, Categorical and numerical	Clustering	• Singular-value Decomposition (SVD) • Hidden Markov model • K-means • Gaussian Mixture • Neural networks	• Recommender system • City planning • Targeted marketing • Customer segmentation

Dimensionality Reduction

Type of Data	ML Technique	Algorithms	Application
Numerical, Text, continuous, categorical	Dimensionality Reduction	• Principal component analysis (PCA) • Linear Discriminant Analysis (LDA) • Generalized Discriminant Analysis (GDA)	• Text mining, • Face recognition • Bigdata visualization • Structure discovery • Image recognition

Reinforcement Learning (Ranking)

Type of Data	ML Technique	Algorithms	Application
Text, images	Reinforcement Learning (Ranking)	• Q – Learning • State – Action -Reward- State- Action (SARSA) • Deep Q Network (DQN) • Deep Deterministic Policy Gradient (DDPG)	• RL can be used in space exploration and Navigation (Rover) • RL can be used in Motor control • RL can be used in Sequence learning for visual captioning • RL can be used for better Decision-making tasks.

R Machine Learning Step-by-Step

- Load the dataset.
- Summarise the dataset.
- Visualise the dataset.
- Evaluate some algorithms.
- Make some predictions.

R Machine Learning Simple Example

- Calculate the height of a child given the age.
- We assume use the age dataset.
- We use simple linear regression

```{r}
# 1. READ IN THE DATA
data1 <- read.csv('age.csv')
print(data1)
```

R Machine Learning Plot and Correlation check

Plot a graph of Height vs age

```` {r}
plot(height~age, data=data1)

```

```{r}
library(corrgram)
# correlation check
corrgram(data1, lower.panel=panel.shade,
upper.panel=panel.cor)
```

# Split Dataset (Train/Test)

Divide the dateset into two, 70% Training Set and 30% test set

```{r}
library(caTools)
# 3. TRAIN/TEST SPLIT
set.seed(42)

sampleSplit <- sample.split(Y=data1$height, SplitRatio=0.7)
trainSet <- subset(x=data1, sampleSplit==TRUE)
testSet <- subset(x=data1, sampleSplit==FALSE)
trainSet
testSet
```

Train the model using trainSet data

```{r}
# 4. TRAIN THE MODEL
model <- lm(formula=height ~ ., data=trainSet)
summary(model)
```

# Visualise the Residual

# Visualise the Residual

The difference between the observed value of the dependent variable (y) and the predicted value (ŷ) is called the residual (e). Each data point has one residual.

Residual = Observed value - Predicted value

Both the sum and the mean of the residuals are equal to zero

A residual plot is a graph that shows the residuals on the vertical axis and the independent variable on the horizontal axis. If the points in a residual plot are randomly dispersed around the horizontal axis, a linear regression model is appropriate for the data; otherwise, a nonlinear model is more appropriate.

```{r}
# visualize residual
library(ggplot2)
modelResiduals <- as.data.frame(residuals(model))
ggplot(modelResiduals, aes(residuals(model))) +
  geom_histogram(fill='deepskyblue', color='black')
```

# Make Prediction

# Make Prediction
```{r}

# 5. MAKE PREDICTIONS
preds <- predict(model, testSet)
Preds

```

# Evaluate Prediction

# Evaluate the prediction model
Calculate the mean squared error (MSE). The MSE of an estimator (of a procedure for estimating an unobserved quantity) measures the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated

Calculate the Root mean square error (RMSE) or root mean square deviation.  RMSE is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance.

```{r}
# 6. EVALUATE PREDICTIONS
modelEval <- cbind(testSet$age, testSet$height, preds)
colnames(modelEval) <- c('age', 'Actual', 'Predicted')
modelEval <- as.data.frame(modelEval)
head(modelEval)

mse <- mean((modelEval$Actual - modelEval$Predicted)^2)
mse

rmse <- sqrt(mse)
rmse
```

# R Machine Learning More Examples

- Calculate the weight of a fish using the length, height and width. The following uses Fish market dataset, it is free and released under the GPL2 license. This dataset includes a number of species of fish and for each fish some measurements such as weight and height are recorded.
- Using the iris dataset available in R or the UC Irvine Machine Learning Repository build a machine learning model that can predict any of the flowers below:


**Iris Versicolor** **Iris Setosa** **Iris Virginica**

# Conclusion

- Your knowledge of Rmarkdown programming is your sure way to collaborative research.
- Do not forget that machine learning has made life easier.

# Questions