R Fundamentals

CFI™

What is R?

R is a free, open-source programming language for statistical computing and generating graphics.

Base-R

R is extensible. Its base capability is known as Base-R, but R possesses many different packages that extend this functionality

Install

R Studio

The most popular integrated development environment for programming with R is RStudio

Install

Tidyverse

The tidyverse consists of the most popular packages for data analysis

Install

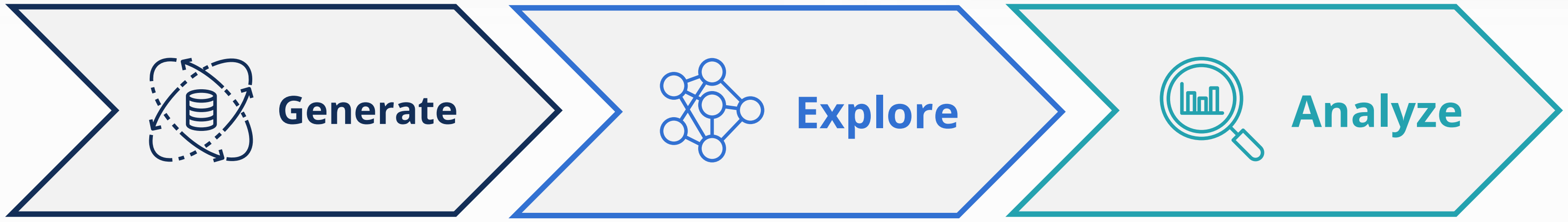
1 – base R

Chapter Introduction – base R

R is a popular programming language for Data Science and data analysis.

We will begin coding with R in the console of RStudio to:

- Use **operators**
- Define **variables**
- Group variables into **data structures**
- Use some built-in **functions** that are widely used for mathematical and statistical analysis
- Extend the capabilities of base R by **installing** and **loading packages**



2 – RStudio & Tidyverse

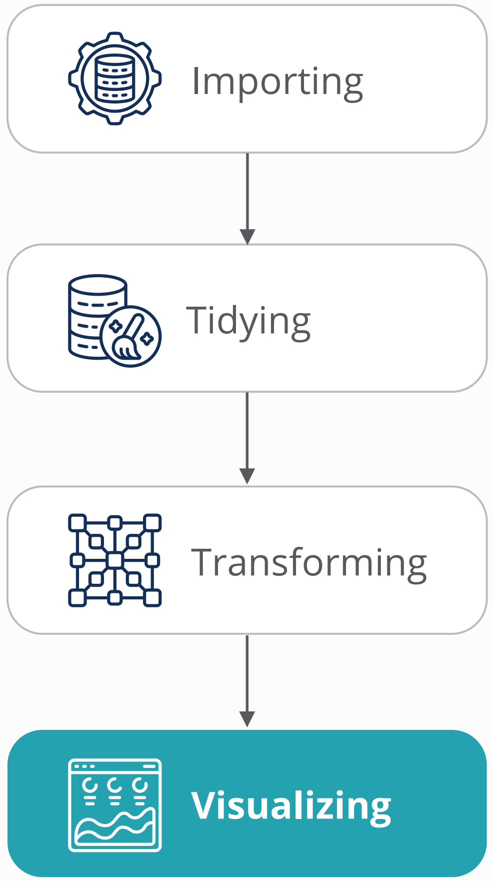
Chapter Introduction – RStudio & Tidyverse

RStudio

- RStudio is an integrated development environment for R
- Coding with R can be quicker, more efficient, and more convenient with features in RStudio
- This allows us to focus on what matters - analyzing our data

Tidyverse

- The tidyverse is a collection of R packages designed for data science
- All packages in the tidyverse share a consistent design philosophy, grammar, and data structures
- The tidyverse provides intuitive and readable functions that can be combined together across packages



What is the Tidyverse?

The tidyverse is a collection of R packages designed for data science.

All packages in the tidyverse share a consistent design philosophy, grammar, and data structures

```
install.packages(“tidyverse”)
```

Installs the tidyverse metapackage

```
library(tidyverse)
```

Loads the tidyverse in the current session

Import

readr **readxl**

Transform

dplyr

Clean

tidyr **purrr**

Visualize

ggplot2

Tibbles

%>% Pipes

Cheat Sheets

3 – Import & Tidy Data

Chapter Introduction – Import & Tidy Data

We will use the **readr**, **readxl**, and **tidyr** packages to import and tidy data

Import Data

1. RStudio interface to import a delimited file
2. Replicate the functionality with readr
3. Import Excel data with readxl

Tidy Data

1. Separate and combine columns
2. Drop and replace NA values
3. Pivot and unpivot columns
4. Fill missing values in columns

We will use the **purrr** package to perform multiple operations with functional programming

Import → Tidy → **Analyze**

Chapter Review – Import & Tidy Data

In this chapter:

Import different types of data into RStudio

Tidy common issues with our data

Apply a function to multiple items

4 – Transform & Analyze Data

Chapter Introduction – Transform & Analyze Data

We will use the **dplyr** package to transform and analyze data

dplyr

Transform Data

1. Selecting Columns
2. Filtering Rows
3. Create New Columns
4. Group and Aggregate Data
5. Join Data

Advanced Functions

1. Tidy Selection
2. Perform operations on multiple columns/rows

Transform → Manipulate → **Analyze**

CFI™

Chapter Review – Transform & Analyze Data

In this chapter, we reviewed **common data transformations** using **dplyr verbs**.

Select and create columns

Filter and aggregate rows

Arrange tibbles and data structures

Summarise data

5 – Visualize Data

Chapter Introduction – Visualize Data

We will use the **ggplot2** package to visualize data

Gradually build plots

Start with the basics then continue to advanced arguments

Visualize → Save → **Share**

ggplot2 - Basics

To visualize data with **ggplot2**, we need an understanding of the **grammar of graphics**.

The grammar underlying plots:

- Makes it easier **to update individual elements**
- Provides a **framework** to think about plots

Plot Area	Definition	Syntax
Data	Select the data to visualize	`ggplot(data = dataset)`
Aesthetics	Map columns from data to plot attributes	`ggplot(aes(x = column1, y = column2))` `ggplot(data = dataset, aes(x = column1, y = column2))` FULL SYNTAX

Geoms	Define the type of plot	`ggplot() + geom_scatter()`
		`ggplot(data = dataset, aes(x = column1, y = column2))` FULL SYNTAX
		`+ geom_scatter()`
		`ggplot(data = dataset, aes(x = column1, y = column2))` FULL SYNTAX
		`+ geom_line()`

ggplot2 - Continued

We can start to add on some other **functions** in ggplot2 to further define the appearance of our plots.

Plot Area		Definition	Syntax
Data	Facet	Create multiple plots, or subplots	`facet_wrap()` `facet_grid()`
	Theme	Control the overall appearance of a plot	`theme_bw()` `theme_minimal()`
Aesthetics	Labels	Add description to a plot	`labs(title = “Plot Title”)`
	Stats	Define the type of plot appearance, similar to geoms()	`geom_bar(stat = “identity”)`
Geoms	Scales	Change the plot axis	`scale_y_continuous()`

```
ggplot(data = dataset,
       aes(x = column1, y = column2)
) +
       geom_scatter(stat = “identity”) +
       facet_wrap() +
       theme_something() +
       labs(title = “Plot title”) +
       scale_y_continuous()
```

Chapter Review – Visualize Data

In this chapter, we reviewed the basics of creating plots using the **grammar of graphics** with **ggplot2**.

Create basic plots

Layer multiple geoms()

Change the appearance of a plot with labels and themes

Control appearance of plot axis and geoms() with stats and scales