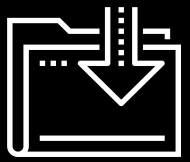
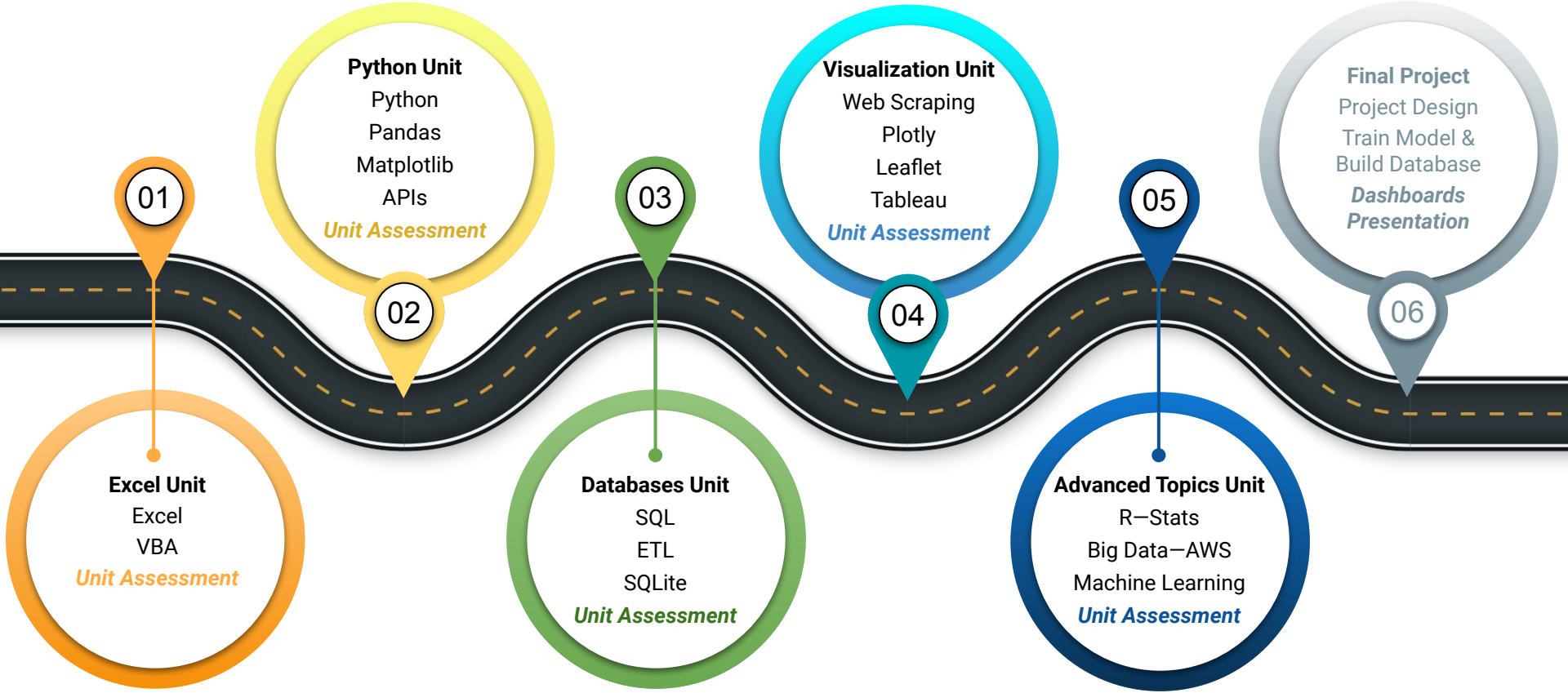
Fundamentals of R

Data Boot Camp

Lesson 15.1



The Big Picture



Quick Tip for Success:

You'll have many opportunities this week to dig into documentation while solving problems. This is pretty on par with what your day-to-day experience will be like when learning new tools on the job.

Module 15

This Week: Fundamentals of R

This Week: Fundamentals of R

By the end of this week, you'll know how to:

- Load, clean up, and reshape datasets using tidyverse in R
- Visualize datasets with basic plots, such as line, bar, and scatter plots, using ggplot2
- Generate and interpret more complex plots, such as boxplots and heatmaps, using ggplot2
- Plot and identify the distribution characteristics of a given dataset
- Understand and apply the null and alternative hypothesis tests for a given problem
- Implement simple linear regression and multiple linear regression models, as well as a chi-squared test for a given dataset
- Implement one-sample and two-sample t-tests, and analysis of variance (ANOVA) models for a given dataset

This Week's Challenge

Using the skills learned throughout the week, students will use linear regression to predict vehicle MPG, create summary statistics on each manufacturing lot, and perform t-tests to determine if all manufacturing lots, and each lot, are statistically different from the mean population.

Career Connection

How will you use this module's content in your career?

Module 15

How to Succeed This Week

Quick Tip for Success:

Remember that R has a very deep well of knowledge, and we'll only just be scratching the surface. If you enjoy this module, think of it as the starting point for your future studies.

Module 15

Today’s Agenda

Today's Agenda

By completing today's activities, you'll learn the following skills:

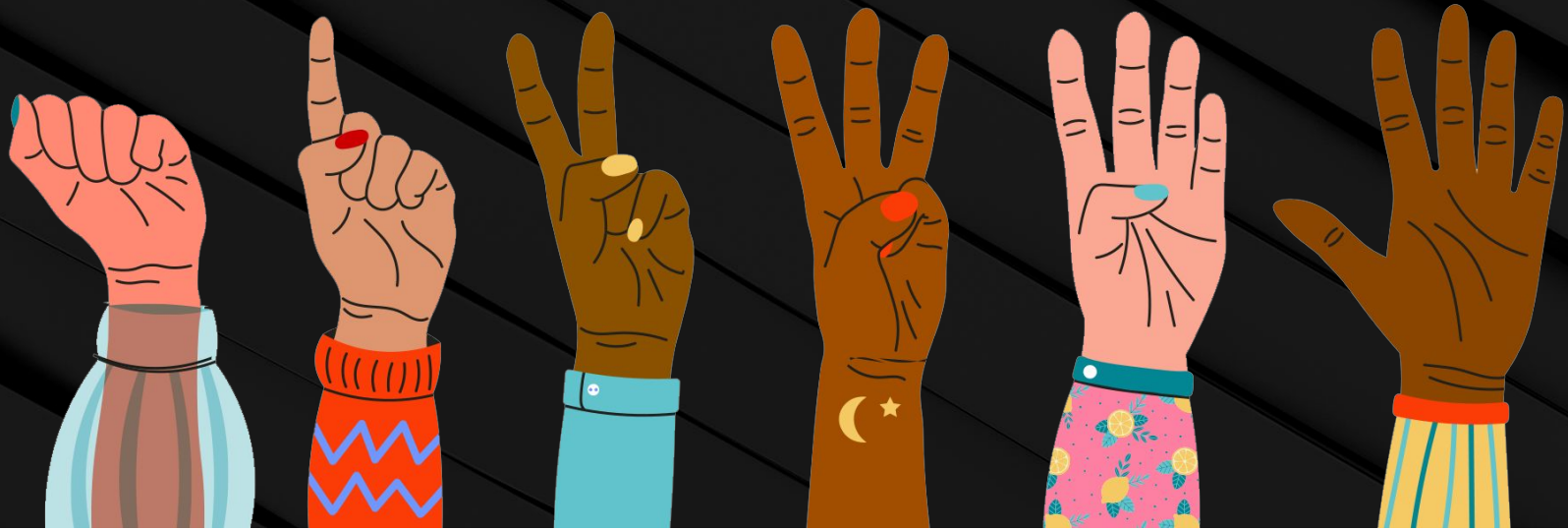
01 Fundamental R programming

02 Creating and manipulating data in tibbles

Make sure you've downloaded any relevant class files!

FIST TO FIVE:

How comfortable do you feel with this topic?



Introduction to R

is a language used for data analysis, statistics, and machine learning; it is also widely used in academia.

Introduction to R

Whether Python or R is better is up for debate.

Introduction to R

R offers compelling features:

Piping and easy-to-use plotting

Faster computation speed

Specialized statistical packages

Great visualization libraries

Installation Check

Suggested Time:

5 minutes

Basics of R

In R, like Python, we can assign values to variables without specifying the data type. However, unlike Python, the left-pointing arrow `<-` is used in R to assign the value on the right to the variable on the left.

```
a <- 3
b <- 3.1415
c <- "This is a string"
d <- "Yet another string"
e <- TRUE
f <- FALSE
g <- T
h <- F
```

Semantically, it is probably more accurate than the equals sign. The equals sign can and will be used in certain cases, as we will see. For simple assignment operations, however, the "assignment operator," `<-`, is preferred.

Basics of R

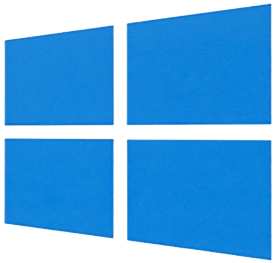
The keyboard shortcut for the assignment operator is:

Mac

`Option-Hyphen`

PC

`Alt-Hyphen`



Basics of R

Like Python lists, an R vector can hold multiple items; however, unlike Python lists, a **vector** must hold items of the same type:

```
disney_characters <- c("mickey", "minnie", "donald", "goofy")

presidents <- c("washington", "adams", "jefferson")

numbers_vector <- c(1, 3, 5, 7, 9, 11)
```

Even a single item can be a vector.

This next point is extremely important!

Basics of R

R data structures are indexed at one.
Python and JavaScript arrays are indexed at zero.

In this example, presidents[1] returns the first item from the vector, `"washington"`

```
disney_characters <- c("mickey", "minnie", "donald", "goofy")

presidents <- c("washington", "adams", "jefferson")

numbers_vector <- c(1, 3, 5, 7, 9, 11)
```

Basics of R

R data structures are indexed at one.
Python and JavaScript arrays are indexed at zero.

In Python or JavaScript, `"adams"` would be returned.

```
disney_characters <- c("mickey", "minnie", "donald", "goofy")

presidents <- c("washington", "adams", "jefferson")

numbers_vector <- c(1, 3, 5, 7, 9, 11)
```

Basics of R

Vectors are created using the `c()`, or concatenate, function.

We can combine two vectors into a single vector with the same operation:

```
combined_vector <- c(disney_characters, presidents)
```

Basics of R

A `for` loop in R is similar to what we've seen in Python and JavaScript:

```
for (x in combined_vector){
    print(x)
}
```

Basics of R

We can also create a vector of integers using the colon operator `(:)` and the `length` function. We can even perform operations on them en masse:

```
numeric_vector <- 1:length(combined_vector)

squared_vector <- numeric_vector**2
```

Basics of R

An `if` statement works much the same way in R as it does in Python:

```
for (prez in presidents){
    if (nchar(prez) > 5){
        next
    }
    else {
      print(prez)
    }
}
```

Basics of R

`nchar()` returns the number of characters in a string. `next` stops the current loop iteration and starts a new iteration from the beginning.

```
for (prez in presidents){
    if (nchar(prez) > 5){
        next
    }
    else {
      print(prez)
    }
}
```

Basics of R

R vectors can contain only a single data type, but a list in R can contain multiple data types.

```
.
   random_list <- list("movies"=c("Star Wars", "Titanic", "Avatar"),
                                    "states"=c("California", "Oklahoma", "Texas", "Virginia"),
                                    "coins"=c("penny", "dime", "nickel", "quarter"),
                                    "first_presidents"=presidents,
                                    "nums"=c(1,2,3,4,5),
                                    "bools"=c(T,F,T,T,T,F)
                                    )
```

Basics of R

We can use bracket notation to access an item in a list:

```
random_list["states"]
```

Basics of R

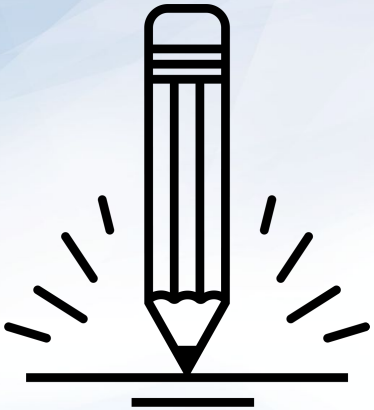
We can also use a dollar sign to accomplish the same task:

```
random_list$coins
```

Basics of R

We can verify that `random_list` is indeed a list with `typeof()`:

```
typeof(random_list)
```



Activity: We R in Junior High Again

In this activity, you will practice the basics of R syntax. You will use R to create vectors, for loops, and if-else statements; to identify substrings of strings; and to create functions to generate daily attendance sheets and locker combinations.

Suggested Time:
15 minutes

Let's Review

Instructor Demonstration

Vectors and Pipe

Basics of R

A vector in R can be paired up with another vector using the `names()` function.

```
# Assign names to a vector
# Assign months to precipitation as names
names(precipitation) <- months

# Display precipitation
print(precipitation)
```

Basics of R

When we display `precipitation`, the output console returns each month and its average rainfall in a column.

```
# Display precipitation
print(precipitation)
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
3.9 2.9 4.1 3.9 4.5 3.5 4.5 4.1 4.0 3.4 3.8 3.6
```

Basics of R

Accessing a single member of `precipitation` is similar to retrieving Series data from a single column in Pandas using `df['column']`.

```r
# Access a single member of precipitation by its name
mar_precipitation <- precipitation["Mar"]
print(mar_precipitation)
```

Basics of R

The R `summary()` function will show a statistical summary, just like the `describe()` function does for Pandas.

We can store the results of `summary()` in a vector and access features of the summary.

```
# Display summary data of precipitation
summary(precipitation)


# Store the results in a vector.
precipitation_summary <- summary(precipitation)
```

Basics of R

We can also use the familiar square brackets to index elements in a vector.

```
# Access features of a summary
precipitation_summary["Min."]
precipitation_summary["Mean"]
```

Basics of R

You can access the value using double brackets.

```
# Use double brackets to access only the value
precipitation_summary[["Max."]]
```

Instructor Demonstration

File-Structure Navigation

Additional Commands

To create a directory called "data_science"	`dir.create("data_science")`
To create a file	`file.create("my_first.R")`
To determine whether a file exists	`file.exists()`
To obtain additional info on a file	`file.info()`
To rename a file	`file.rename(file1, file2)`
To copy a file	`file.copy()`

Data Frames in R

Instructor Demonstration

Data Frames in R

Tibbles in R are similar to Pandas DataFrames: data are organized by rows and columns, which allow operations for computation and data-wrangling.

Data Frames in R

Tibbles are not available in standard R, but they are enabled by `tidyverse` and are generally superior to R's standard data frame.

The `data()` function takes the data source, `diamonds`, and the plotting package, `ggplot2`, as arguments.

```
### Load dependency and sample data set
library(tidyverse)
data(diamonds, package='ggplot2')
```

Data Frames in R

Unlike other languages that we have encountered, R allows the use of periods in regular variable names.

The variable, `total.volume2`, does not refer to a `volume2` property of the `total` object as it would in JavaScript. Instead, it is equivalent to `total_volume2`.

```
total.volume2 <- mutate(diamonds, total.volume=(x*y*z))
```

Data Frames in R

With the `mutate()` function, we can add a new column to the tibble.

```
### Operate over a column, temporarily add a new column
```{r}
total_volume <- mutate(diamonds, x * y * z)
total_volume
```
```

cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>	x * y * z <dbl>
Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	38.20203
Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	34.50586
Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	38.07688
Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	46.72458
Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	51.91725
Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48	38.69395
Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47	38.83087
Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53	42.32108
Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49	36.42521
Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39	38.71800

1–10 of 53,940 rows | 2–11 of 11 columns Previous 1 2 3 4 5 6 … 100 Next

```
### Name the new column, save the results to another tibble
```{r}
# In R, variables can contain periods
total.volume2 <- mutate(diamonds, total.volume=(x*y*z))
total.volume2
```
```

cut <ord>	color <ord>	clarity <ord>	depth <dbl>	table <dbl>	price <int>	x <dbl>	y <dbl>	z <dbl>	total.volume <dbl>
Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43	38.20203
Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31	34.50586
Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31	38.07688
Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63	46.72458
Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75	51.91725
Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48	38.69395
Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47	38.83087
Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53	42.32108
Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49	36.42521
Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39	38.71800

Questions?

Time to Code

Back to School

Suggested Time:

25 minutes