PART 2: OPERATORS, MISSINGNESS AND TIDY DATA PRACTICES WITH TIDYVERSE

Justine Béchard & Jan Eckardt

GAPS - R Workshop 2024

Basic Statistics - Mean and Median

Mean: Refers to the average value of a distribution...Think of this as you would about calculating your average grades in high school, for example

```
> mean(iris$Sepal.Length)
[1] 5.843333
```

Median: Refers to the 50th percentile of a distribution, meaning the value that is right in the middle when values are ordered from smallest to largest

```
> median(iris$Sepal.Length)
[1] 5.8
```

Arithmetric Operators

We can also run basic mathematic operations in R...

```
> # Addition...
>
> 4 + 4
[1] 8
>
> # ...subtraction...
>
> 4 - 4
[1] 0
>
> # ...multiplication...
>
> 4 * 4
[1] 16
>
> # ...and division
>
> 4 / 4
[1] 1
```

...which even works with variables (e.g., conversion to inches)

```
> iris$Sepal.Length * 0.393701
  [1] 2.007875 1.929135 1.850395
 [12] 1.889765 1.889765 1.692914
 [23] 1.811025 2.007875 1.889765
 [34] 2.165355 1.929135 1.968505
 [45] 2.007875 1.889765 2.007875
 [56] 2.244096 2.480316 1.929135
 [67] 2.204726 2.283466 2.440946
```

Relational Operators

There are also several “relational” operators in R. These help us “tell” R how it should evaluate the relationship between our objects or values:

Equal-to: “==”

```
> iris[iris$Species == "setosa",]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
```

Not-equal-to: “!=”

```
> iris[iris$Species != "setosa",]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
51           7.0         3.2          4.7         1.4 versicolor
52           6.4         3.2          4.5         1.5 versicolor
53           6.9         3.1          4.9         1.5 versicolor
```

Relational Operators

Greater-than: “>”
Smaller-than: “<”

```
> iris[iris$Sepal.Length > 5.4,]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
15           5.8         4.0          1.2         0.2     setosa
16           5.7         4.4          1.5         0.4     setosa
19           5.7         3.8          1.7         0.3     setosa
```

Greater-or-equal: “>=”
Smaller-or-equal: ”<=”

```
> iris[iris$Sepal.Length >= 5.4,]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
6            5.4         3.9          1.7         0.4     setosa
11           5.4         3.7          1.5         0.2     setosa
15           5.8         4.0          1.2         0.2     setosa
```

Relational Operators

Or: "|"

```
> iris[iris$Species == "setosa" | iris$Species == "virginica", ]
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
1            5.1         3.5          1.4         0.2    setosa
2            4.9         3.0          1.4         0.2    setosa
3            4.7         3.2          1.3         0.2    setosa
```

And: "&"

```
> iris[iris$Species == "virginica" & iris$Sepal.Width > mean(iris$Sepal.Width), ]
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
101          6.3         3.3          6.0         2.5 virginica
110          7.2         3.6          6.1         2.5 virginica
111          6.5         3.2          5.1         2.0 virginica
```

Relational Operators

Just FYI: Behind the scenes, R essentially looks at whether the relationship between the values or objects matches the relational operator....

.... and returns either a TRUE or FALSE logical value. We will get back to this later

Assignment Operator "<-”

This one’s important - If you want to save or alter anything to work with later in your code, this operator is essential!

```
> my_favorite_university <- "Western"
```

You can get the operator by using a shortcut if you’re pressed for time

“option” and “-” on Mac; “Alt” and “-” on PC

...or just type it out

Logical/Boolean Values

As stated before, R evaluates relationships using TRUE and FALSE statements

This can also be a useful property for exploring and working with our data

Logical/Boolean Values

Many functions output logical values, i.e., TRUE/FALSE…

```
> any(iris[iris$Species == "setosa", ]$Sepal.Length > mean(iris$Sepal.Length))
[1] FALSE
```

But TRUE or FALSE statements are also output when using relational operators to compare values or objects without employing a specific function

```
> mean(iris$Sepal.Length) > mean(iris$Sepal.Widt
[1] TRUE
```

Handling Missing Data

NULL: Represents the absence of value or an undefined object

NA: Stands for “Not Available”. It corresponds to missing or undefined data in a data frame or a vector.

NaN: Means “Not a Number”. It is a type of NA for undefined mathematical operations (e.g. 0/0).

A Brief Introduction to Tidyverse

It is easier to work with data once it is in a certain format.

- **Distinct variables for each column**. For example, a dataset with student grades would contain columns such as “student_id”, “exam_score”, “date”.
- **Each observation has its own row**.
- If an observation has multiple time points, each unit of them is considered its own row and time itself becomes another variable in the dataset.

A Brief Introduction to Tidyverse

- **ggplot2**: Plotting for publishable visualizations with a consistent syntax.
- **dplyr**: Data manipulation package with functions for filtering, selecting and transforming data.
- **tidyr**: Allows to clean and organize data.
- **readr**: Reading from many different types of files into R.
- **purrr**: Functions for iterative operations like manipulating lists.
- **stringr**: Working with strings/character data (manipulating text)
- **forcats**: Working with factor variables (categorical data).
- **tibble**: Display the data in more readable format.

The Pipe Operator %>%

A key feature when using **dplyr**. It turns nested code into sequential code. This allows you to chain together multiple functions in a step-by-step process.

Each line in a piped sequence takes the most recent form of an object and applies the next transformation to it.

Tidy Data Practices

select(): Select which columns/variables you want to keep in your dataset.

filter(): Subset rows based on logical conditions.

arrange(): Can help you sort the data based on the values in a specific column.

mutate(): Used to create new variables or columns. You can use it to transform existing columns or add new ones based on some calculations.

groub_by(): Grouping data by one or many variables. This allows us to perform operations within those groups.

summarize(): Create new summary variables (usually after group_by()).

Data Visualisation with ggplot2

Aesthetic (aes): Element of a plot tied directly to a variable (representing the variation in the data with some visuals). This includes mapping variables to axes, color, sizes, shapes and more.

Geometry (geom_): Determines the form of a plot. Each geom_ represents a specific type of visual representation such as scatter plots, lines, bars, etc.

Themes: This allows you to customize the appearance of your plots (axes appearance, text style, legend customization, and more).

Basic Setup for ggplot2

(1) ggplot() line. This is the initial setup of your plot code on the first line.

- Input: A data frame or tibble (can be piped in).
- Output: A blank canvas, at least, not without a geom.

Basic Setup for ggplot2

(2) geom_...() line.

- This line adds the geometry, specifying the king of plot you are making. This is also usually where aesthetic (aes()) must be called to call the data to visual elements.

Basic Setup for ggplot2

(3) Any other details follow with various ggplot2 function lines

- labs() allows personalized labels and a title
- Predefined (theme_minimal(), theme_classic(), ...) or custom themes