Summarising and Plotting Data in R

Summarising Data

Glenn Williams

University of Sunderland

2020-09-29 (updated: 2020-10-16)

Summarising Data

Individual data points are some **abstraction away from reality**, essentially removing some detail.

- Even with trial level data, we will **summarise responses** in terms of reaction times or correct responses.
- This removes a lot of the details of the real experience, but allows us to **see patterns**.
- Similarly, we can summarise data from many participants in a number of ways to help communicate a pattern or idea.

We can present summaries of data as a **plot**, or even as a **table of descriptive statistics**.

Descriptive Statistics

We can often summarise data from individual trials in terms of measures of **central tendency** and **dispersion**.

- **Central tendency**: What is the average score? This can be determined in a few ways.
    - **Mean**: used to describe data that are normally distributed. Can be misleading when outliers are present.
    - **Median**: used to describe data that are skewed in some way away from a normal distribution. Suppresses the impact of outliers.

```
nonskew <- c(10, 20, 15, 20, 22)
mean(nonskew)
```

```
## [1] 17.4
```

```
skew <- c(10, 20, 15, 20, 2000)
mean(skew)
```

```
## [1] 413
```

```
median(nonskew)
```

```
## [1] 20
```

```
median(skew)
```

```
## [1] 20
```

Descriptive Statistics

- **Dispersion**: What is the spread of scores (e.g. for individuals) around the average score?
    - **Standard Deviation**: the spread of the data from the sample mean. With a normal distribution approximately 1, 2, and 3 SDs capture approximately 68, 95, and 99.7% of the data.
    - **Interquartile Range**: Rank orders data into four equal parts; the first region – or quartile – (i.e. 25%), the median (i.e. 50%), and the middle part of the third quartile (i.e. 75%). Used for non-normal data.

```r
# low variability in scores = low SD
low_dispersion <- c(10, 12, 8, 9, 11, 10, 10.5, 9, 11.5)
sd(low_dispersion)
```

```
## [1] 1.293681
```

```r
# lots of variability in scores = high SD
high_dispersion <- c(10, 1000, 89, -400, 90, 880, 0)
sd(high_dispersion)
```

```
## [1] 508.6885
```

Why Make these in R?

- **Scales up easily to new data**: If you write this code to analyse data from 10 participants, you can instantly rerun it for millions.
- **Easier to spot mistakes** when you're writing the recipe.
- **Easier to fix mistakes** with a minor modification instead of repeating every step of the analysis by hand.
- **Repeatability**: If you do a task once in R, you can copy and paste code for a new study. This saves a lot of time in the long run.
- **Reproducibility**: Allows others (and future you) to check work and inspect methods at every step. You'll make science more reliable!

Summarise

Let's take a look at the inbuilt `starwars` data set in the `tidyverse` pacakage.

```
data(starwars) # load the data from within the package
starwars # print it
```

```
## # A tibble: 87 x 14
##   name   height  mass hair_color skin_color eye_color birth_year sex   genc
##   <chr>   <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr
## 1 Luke…     172    77 blond      fair       blue              19 male  masc
## 2 C-3PO     167    75 <NA>       gold       yellow           112 none  masc
## 3 R2-D2      96    32 <NA>       white, bl… red               33 none  masc
## # … with 84 more rows, and 5 more variables: homeworld <chr>, species <chr
## #   films <list>, vehicles <list>, starships <list>
```

There's a lot of data here! How might we summarise this?

Summarise

`summarise()` collapses across all observations in your data set to produce a single row of data.

- `summarise()` takes two main arguments:
    - 1: what is the **data** set you want to summarise?
    - 2: additional information specifying **how you want to summarise it**.

Let's summarise the mean height of the Star Wars characters:

```
summarise(starwars, mean_height = mean(height))
```

```
## # A tibble: 1 x 1
##   mean_height
##         <dbl>
## 1          NA
```

Oops, we got NA! Why? R by default produces NA for any summary of data containing NA values.

Summarise

What should we do? We can make a new data set without NAs, or tell `mean()` to ignore them.

```
summarise(starwars, mean_height = mean(height, na.rm = TRUE))
```

```
## # A tibble: 1 x 1
##   mean_height
##         <dbl>
## 1        174.
```

We set the argument na.rm in `mean()` to TRUE, meaning "Should R remove NAs? TRUE (yes)".

Many Summaries

What if we want to get different types of summaries? Imagine we want the count, mean, and standard deviation of height?

We can tell R to generate these summaries on the data by first dropping any observations with missing values (NA) in height.

```
summarise(
  drop_na(starwars, height),
  n = n(),
  sd = sd(height),
  mean_height = mean(height)
)
```

```
## # A tibble: 1 x 3
##       n    sd mean_height
##   <int> <dbl>       <dbl>
## 1    81  34.8        174.
```

How might we interpret this? We have 81 measured heights of characters. The mean height is 174.36cm, while the standard deviation is 34.77.

Grouping

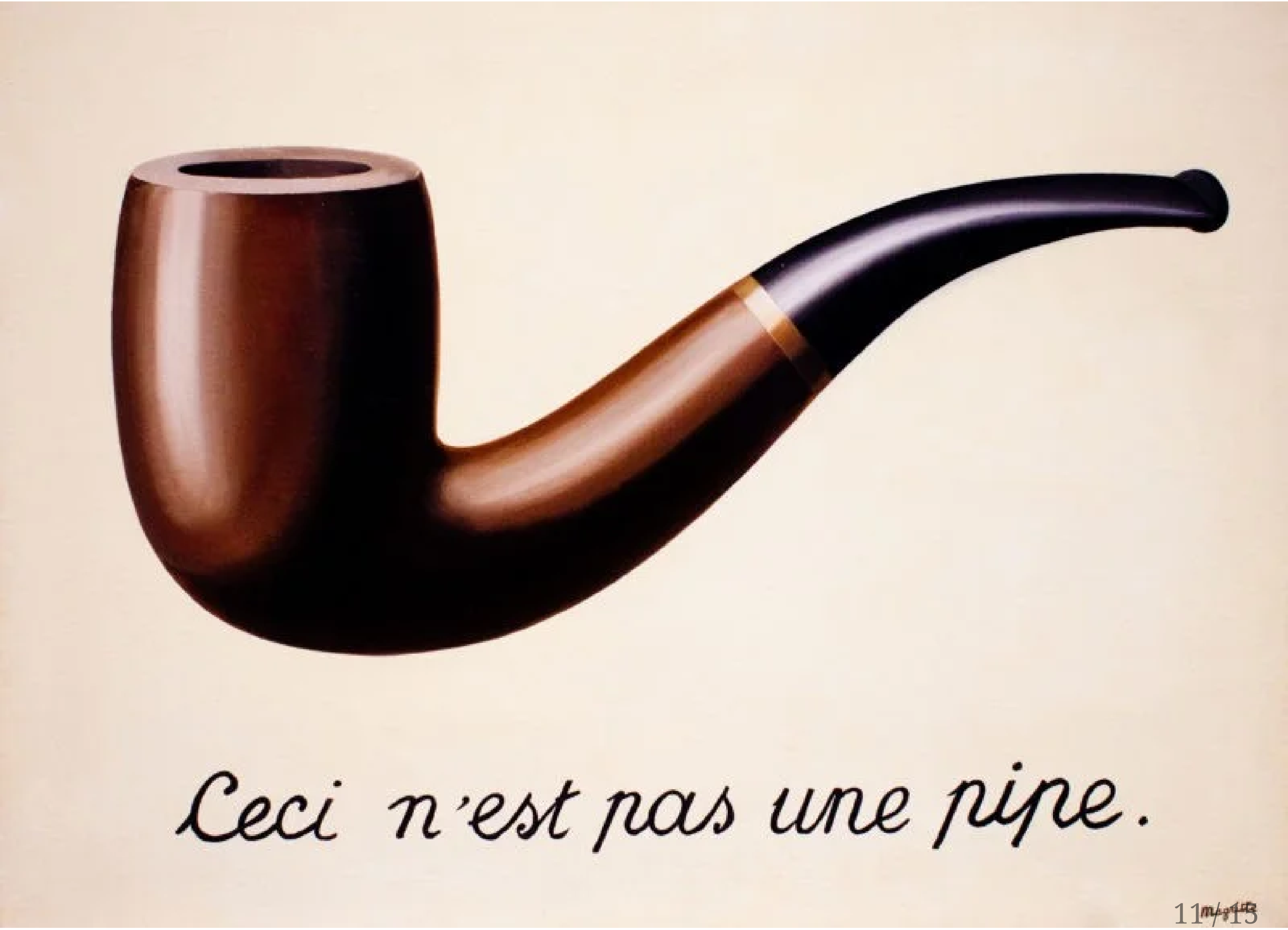
As we can see, the data set contains very many species of creates from the Star Wars universe. What if we want summaries of all of them?

We can add an argument to our data in `summarise()` called `group_by()`.

Within `group_by()`, we again specify our data, but also give a column by which to separate the summaries.

```r
summarise(
  group_by(starwars, species),
  mean_height = mean(height, na.rm = TRUE)
)
```

```
## # A tibble: 38 x 2
##   species  mean_height
##   <chr>          <dbl>
## 1 Aleena            79
## 2 Besalisk         198
## 3 Cerean           198
## # … with 35 more rows
```


Ceci n'est pas une pipe.

The Pipe!

So far, I've avoided showing you some R magic so you get the principles of functions. But R has a way to make code a lot more readable: the pipe!

The pipe in R (`%>%`) passes data from one function to another.

```
starwars %>%
  group_by(species) %>%
  summarise(mean_height = mean(height, na.rm = TRUE))
```

```
## # A tibble: 38 x 2
##   species  mean_height
##   <chr>          <dbl>
## 1 Aleena            79
## 2 Besalisk         198
## 3 Cerean           198
## # … with 35 more rows
```

You can now read left to right: 1. take my data **and then** 2. group by species **and then** 3. summarise, creating the column mean height from the mean of the height column (removing NAs of course).

Adding Complexity

We can **add as many groups as we like** to group by, and as many summaries as we like to summary. For example:

```
starwars %>%
  drop_na(height) %>%
  group_by(sex, gender) %>%
  summarise(
    n = n(),
    sd = sd(height),
    mean_height = mean(height)
  )
```

```
## # A tibble: 6 x 5
## # Groups:   sex [5]
##   sex            gender        n    sd mean_height
##   <chr>          <chr>     <int> <dbl>       <dbl>
## 1 female         feminine     15 15.3         169.
## 2 hermaphroditic masculine     1 NA           175
## 3 male           masculine    57 36.0         179.
## 4 none           feminine      1 NA            96
## 5 none           masculine     4 52.0         140
## 6 <NA>           <NA>          3  2.89        181.
```

All Tidyverse Functions Work with the Pipe!

Let's keep only humans over 70kg in mass, and calculate their height and weight.

```r
starwars %>%
  filter(species == "Human" & mass > 70) %>%
  drop_na(height) %>%
  summarise(
    n = n(),
    mean_height = mean(height),
    sd_height = sd(height)
  )
```

```
## # A tibble: 1 x 3
##       n mean_height sd_height
##   <int>       <dbl>     <dbl>
## 1    20        182.      8.99
```

We could even add `select()` and `mutate()` in here if we wanted.

Recap

We've learned...

- Why we summarise data and recapped on basic **descriptive statistics**.
- **Why we might use R** to summarise our data.
- How to pass **optional arguments** to functions (e.g. na.rm in `mean()` and `sd()`).
- How to make **separate summaries per group** using `group_by()`.
- How to **pipe** data from one function to the next.
- How to **combine tidy functions** like `filter()` and `summarise()` in one pipeline.