Summarising and Plotting Data in R

Summarising Data

Glenn Williams

University of Sunderland

2021-10-27 (updated: 2021-12-05)

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**, as a **table of descriptive statistics**, and make inferences and decisions based on summaries of the data by **modelling our data**.

Descriptive Statistics

We often summarise data 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.

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

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

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

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

```
median(noskew)
```

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

```
median(skew)
```

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

Descriptive Statistics

- **Dispersion**: The spread of scores (e.g. for individuals) around an average.
    - **Standard Deviation**: the spread of the data from the sample mean. With a normal distribution approximately 1, 2, and 3 *SD*s 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. There's a lot of data here! How might we summarise this?

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

```
## Rows: 87
## Columns: 14
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Or…
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 180, 2…
## $ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 84.0, 77.…
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", N…
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "light", "…
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue",…
## $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0, 57.0, …
## $ sex        <chr> "male", "none", "none", "male", "female", "male", "female",…
## $ gender     <chr> "masculine", "masculine", "masculine", "masculine", "femini…
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "T…
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Huma…
## $ films      <list> <"The Empire Strikes Back", "Revenge of the Sith", "Return…
## $ vehicles   <list> <"Snowspeeder", "Imperial Speeder Bike">, <>, <>, <>, "Imp…
## $ starships  <list> <"X-wing", "Imperial shuttle">, <>, <>, "TIE Advanced x1",…
```

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. (You can't average something that isn't there.)

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 also 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.
- **What's the SD mean**? With normally distributed data 95% of values fall within $\pm$ 1.96 SD of the mean!

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 specify our data but also give a column by which to **group the summaries**.

```
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
```

Getting Complicated

We can combine `tidyverse` functions together. We could filter before summarising the data, and even make multiple grouped summaries.

```
summarise(
  group_by(filter(starwars, mass < 400), species),
  mean_height = mean(height, na.rm = TRUE),
  sd_height = sd(height, na.rm = TRUE),
  n = n()
)
```

```
## # A tibble: 31 x 4
##   species  mean_height sd_height     n
##   <chr>          <dbl>     <dbl> <int>
## 1 Aleena            79        NA     1
## 2 Besalisk         198        NA     1
## 3 Cerean           198        NA     1
## # … with 28 more rows
```

Interim 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 combine tidyverse functions together to get the summaries we need.

Analysing Data

Some Background

Data analysis is surprisingly one of the easiest parts of working with R.

- Once your data is in the correct (long) format, analysis using any test is highly consistent.
- We rely on a formula interface like this:

```
DV ~ IV
```

- Our dependent variable/predicted variable goes to the left of the ~ (tilde), while our independent variables or predictors go to the right.
- After this we specify our data:

```
DV ~ IV, data
```

We then apply a function to our formula which is the name of our test. There's some minor options we can choose within tests, but that's pretty much it!

Correlations

The Data

Let's check out the **starwars** data set again. We'll use this for our tests.

```
starwars <- starwars %>% filter(mass < 500)
```

We will use the height and mass columns, looking at whether mass is associated with height.

```
## Rows: 58
## Columns: 14
## $ name       <chr> "Luke Skywalker", "C-3PO", "R2-D2", "Darth Vader", "Leia Or…
## $ height     <int> 172, 167, 96, 202, 150, 178, 165, 97, 183, 182, 188, 228, 1…
## $ mass       <dbl> 77.0, 75.0, 32.0, 136.0, 49.0, 120.0, 75.0, 32.0, 84.0, 77.…
## $ hair_color <chr> "blond", NA, NA, "none", "brown", "brown, grey", "brown", N…
## $ skin_color <chr> "fair", "gold", "white, blue", "white", "light", "light", "…
## $ eye_color  <chr> "blue", "yellow", "red", "yellow", "brown", "blue", "blue",…
## $ birth_year <dbl> 19.0, 112.0, 33.0, 41.9, 19.0, 52.0, 47.0, NA, 24.0, 57.0, …
## $ sex        <chr> "male", "none", "none", "male", "female", "male", "female",…
## $ gender     <chr> "masculine", "masculine", "masculine", "masculine", "femini…
## $ homeworld  <chr> "Tatooine", "Tatooine", "Naboo", "Tatooine", "Alderaan", "T…
## $ species    <chr> "Human", "Droid", "Droid", "Human", "Human", "Human", "Huma…
```

Correlation

- Here, we aren't predicting any one variable from the other, so both variables go to the right of the tilde.
- We add multiple variables with a `+`.
- We choose the type of correlation we want (e.g. Pearson, Spearman) with the method.

```
cor.test(~ height + mass, starwars, method = "pearson")
```

```
## 
##     Pearson's product-moment correlation
## 
## data:  height and mass
## t = 8.7853, df = 56, p-value = 4.018e-12
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6260700 0.8520232
## sample estimates:
##       cor 
## 0.7612612
```

Tests of Difference

- We'll use some different data here on out.
- Let's assume this data looks at giving people a placebo or drug, and tests the effect of that drug at two different time points.
- We care about improvements in reaction times.

```r
mixed_data <- read_csv(here("data", "mixed_factorial.csv"))
head(mixed_data)
```

```
## # A tibble: 6 x 4
##   id    drug    time          rt
##   <chr> <chr>   <chr>      <dbl>
## 1 S001  control daylater    431.
## 2 S001  control monthlater  421.
## 3 S002  control daylater    372.
## 4 S002  control monthlater  350.
## 5 S003  control daylater    393.
## 6 S003  control monthlater  368.
```

t-tests

One-sample t-test

- We have only one variable here, so we don't even need a formula.
- We compare the mean of this variable against a specified baseline mean (here 400).

```
t.test(mixed_data$rt, mu = 400)
```

```
## 
##      One Sample t-test
## 
## data:  mixed_data$rt
## t = -9.5311, df = 479, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 400
## 95 percent confidence interval:
##  376.3267 384.4193
## sample estimates:
## mean of x 
##   380.373
```

t-tests

Independent-samples t-test

- Do reaction times vary depending on the drug given to participants?
- We test reaction times predicted by drug, with a regular t-test where variances are assumed to be equal (`var.equal = TRUE`).

```
t.test(rt ~ drug, mixed_data, var.equal = TRUE)
```

```
## 
## 	Two Sample t-test
## 
## data:  rt by drug
## t = 3.732, df = 478, p-value = 0.0002128
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   7.18135 23.15251
## sample estimates:
##   mean in group control mean in group treatment 
##                387.9564                372.7895
```

t-tests

Paired t-test

- Do reaction times vary over time (i.e. practice)?
- We test reaction times predicted by the time of testing. This is a paired test (`paired = TRUE`) and a regular t-test where variances are assumed to be equal (`var.equal = TRUE`).

```
t.test(rt ~ time, mixed_data, paired = TRUE, var.equal = TRUE)
```

```
##
##      Paired t-test
##
## data:  rt by time
## t = 18.348, df = 239, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  49.10644 60.91970
## sample estimates:
## mean of the differences
##                55.01307
```

One-way ANOVA

Between-subjects

- What if we had **more than two groups** for the drug condition? We use an ANOVA.
- We simply change the test function to `aov()` (**A**nalysis **O**f **V**ariance)
- We need to summarise the model results here to get a regular ANOVA output.

```
summary(aov(rt ~ drug, mixed_data))
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## drug          1  27604   27604   13.93 0.000213 ***
## Residuals   478 947379    1982
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

One-way ANOVA

Within-subjects

- What if we have more than two groups and a **within-subjects design**?
- We do the same as before, but need to add an **Error term** to the formula. This states that we adjust our errors to account for the fact scores in each group belong to the same participant (i.e. **id** in our data).

```
summary(aov(rt ~ time + Error(id), mixed_data))
```

```
##
## Error: id
##            Df Sum Sq Mean Sq F value Pr(>F)
## Residuals 239 353969    1481
##
## Error: Within
##            Df Sum Sq Mean Sq F value Pr(>F)
## time        1 363173  363173   336.6 <2e-16 ***
## Residuals 239 257842    1079
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Two-Way ANOVA

Mixed

```
summary(aov(rt ~ time * drug + Error(id), mixed_data))
```

```
##
## Error: id
##            Df Sum Sq Mean Sq F value   Pr(>F)
## drug        1  27604   27604   20.13 1.13e-05 ***
## Residuals 238 326365    1371
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Error: Within
##            Df Sum Sq Mean Sq F value Pr(>F)
## time        1 363173  363173   806.3 <2e-16 ***
## time:drug   1 150636  150636   334.4 <2e-16 ***
## Residuals 238 107206     450
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Throw Away the Alphabet Soup

All of the statistical tests you know (e.g. *t*-tests, ANOVA, chi-square) are just extensions of the **general linear model**. This is the most important thing you can learn to use in statistics.

Learn the mean and *variance* of some measurement by using an additive combination of other measurements.

- The **geocentric model of applied statistics**: used wisely, can be useful. But we shouldn't read too much into the numbers produced. They're almost certainly wrong because we can't (and shouldn't) model all sources of variance.
- Predict a **linear relationship** between one or more variable(s) and a continuous (e.g. scale) dependent variable.
- Predictor variables can be continuous or categorical.

Linear Regression

Takes the general form:

$$Y = \alpha + \beta X + e$$

- **Outcome** Y = intercept + (slope $\times$ X) + residual error
- **Residuals** e = distance of observed values from predicted values
- *Note*: We do not fit a perfect model, hence the error term. This is a good thing, otherwise we are probably **overfitting** to our data; relying too much on our observed sample to draw infferences.

Linear Regression

Takes the general form:

$$Y = \alpha + \beta X + e$$

- The **intercept**, α, is usually the point on the y-axis at the lowest value of X (usually 0).
- The **slope**, β, corresponds to how much Y increases by for every increment in X.
- The **error**, e, corresponds to a constant by which to add to our estimates accounting for additional variation from other sources that we do not model.

Linear Regression

Fit the model predicting height from weight from the starwars data.

$$Y = \alpha + \beta X + e$$

$$height = intercept + slope \times mass + error$$

```
starlm <- lm(height ~ mass, starwars)
summary(starlm)
```

```
##
## Call:
## lm(formula = height ~ mass, data = starwars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.369  -6.816   2.042  13.851  44.719
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 103.5133     8.5937  12.045  < 2e-16 ***
## mass          0.9327     0.1062   8.785 4.02e-12 ***
## ---
```

Comparing tests we know...

Correlation

```r
broom::tidy(cor.test(~ height + mass, starwars, method = "pearson"))
```

```
## # A tibble: 1 x 8
##   estimate statistic  p.value parameter conf.low conf.high method    alternative
##      <dbl>     <dbl>    <dbl>     <int>    <dbl>     <dbl> <chr>     <chr>
## 1    0.761      8.79 4.02e-12        56    0.626     0.852 Pearson'… two.sided
```

```r
broom::tidy(lm(height ~ mass, starwars, method = "pearson"))
```

```
## Warning in lm(height ~ mass, starwars, method = "pearson"): method = 'pearson'
## is not supported. Using 'qr'
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  104.        8.59       12.0  3.53e-17
## 2 mass           0.933     0.106       8.79 4.02e-12
```

Comparing tests we know...

t-tests

```
broom::tidy(t.test(rt ~ drug, mixed_data, var.equal = TRUE))
```

```
## # A tibble: 1 x 10
##   estimate estimate1 estimate2 statistic  p.value parameter conf.low conf.high
##      <dbl>     <dbl>     <dbl>     <dbl>    <dbl>     <dbl>    <dbl>     <dbl>
## 1     15.2      388.      373.      3.73 0.000213       478     7.18      23.2
## # … with 2 more variables: method <chr>, alternative <chr>
```

```
broom::tidy(summary(lm(rt ~ drug, mixed_data)))
```

```
## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)      388.       2.87    135.   0
## 2 drugtreatment    -15.2      4.06     -3.73 0.000213
```

- *t* statistics match exactly.

Comparing tests we know...

ANOVA

```
summary(aov(rt ~ drug, mixed_data))
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)    
## drug          1  27604   27604   13.93 0.000213 ***
## Residuals   478 947379    1982                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
broom::tidy(lm(rt ~ drug, mixed_data))
```

```
## # A tibble: 2 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)      388.       2.87    135.   0       
## 2 drugtreatment    -15.2      4.06     -3.73 0.000213
```

- *t* to *F* is just *t* squared. So, 3.732 squared = 13.93...

Bye!

Effect sizes are easily handled by the {effectsize} package. Super-easy ANOVAs are done using the {afex} package.