Primary functions for data analysis:

- **read_csv()** : imports text files in CSV format into the R environment.
  - ex: `read_csv("path/filename.csv")`
- **filter()** : select only certian rows from a dataset.
  - ex: `filter(text_column=="value")` or `filter(number_column > 0)`
- **group_by()** : group the rows in a dataset based on the values in one or more columns.
  - ex: `group_by(column)` or `group_by(column_a, column_b, …)`
- **summarise()** : apply summary functions (see below) to the whole dataset, or to groups if `group_by()` is used
  - ex: `summarise(total = sum(number_column)` or `summarise(average = mean(number_column)`
- **arrange()** : sort results based on one or more columns; ascending is the default, use `desc()` for descending.
  - ex: `arrange(column)` or `arrange(desc(column))`
- **count()** : group the data based on a column or columns and apply the n() summary function to those groups.
  - ex: `count(text_column)`
- **mutate()** : create a new column in a dataset
  - ex: `mutate(new_column = number_column1 + number_column2)`

Summary functions for analysis, used in `summarise()`

- **sum()** : sum a numbers column. If NAs are present will return NA, unless `na.rm=T` is used.
  - ex: `summarise(total = sum(number_column, na.rm=T))`
- **mean()** : average a numbers column. If NAs are present will return NA, unless `na.rm=T` is used.
- **median()** : get the median of a numbers column. If NAs are present will return NA, unless `na.rm=T` is used.
- **n()** : count rows or observations (no arguments in the parentheses).
  - ex: `summarise(num = n())`
- **range()** : get the highest and lowest values in a column. This is best used with `reframe()` instead of `summarise()` :
  - ex: `reframe(range(column))`

Supplementary functions for manipulating data:

- **clean_names()** : from the `janitor` package, standardizes column names in a dataset to all lowercase, and replaces non-letter characters and spaces to underscores (_).
  - ex: `tibble <- clean_names(tibble)`
- **is.na()** : used within `filter()` to find `NA` s (NULLs).
  - find nulls: `filter(is.na(column))`
  - exclude nulls: `filter(!is.na(column))`
- **as.numeric()** : converts strings to numbers; will coerce non-numeric values into NA.
- **as.character()** : converts non-string values to strings.