JW_Data_Pipelines

James Waterford

2023-03-07

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

There are two main ways to run a code:

Nested Code

This method involves running mulitple codes in a single line.

```
numbers <- 1:300
mean(numbers)
```

```
## [1] 150.5
```

```
sqrt(mean(numbers))
```

```
## [1] 12.26784
```

Sequential Code

This method generates intermediate variables to perform statistics on.

```
numbers <- -300:456
mn <- mean(numbers)
sqrt(mn)
```

```
## [1] 8.831761
```

```
library(readr)
surveys <- read_csv("197-raw_storage/surveys.csv")
```

```
## Rows: 35549 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): species_id, sex
## dbl (7): record_id, month, day, year, plot_id, hindfoot_length, weight
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
species_data <- read.csv( "197-raw_storage/species.csv")
plots_data <- read.csv("197-raw_storage/plots.csv")
```

```
## # A tibble: 6 x 4
##    year month   day species_id
##   <dbl> <dbl> <dbl> <chr>
## 1  1977     7    16 NL
## 2  1977     7    16 NL
## 3  1977     7    16 DM
## 4  1977     7    16 DM
## 5  1977     7    16 DM
## 6  1977     7    16 PF
```

```
## # A tibble: 6 x 4
##    year species_id weight weight_kg
##   <dbl> <chr>       <dbl>     <dbl>
## 1  1977 PF              4      4000
## 2  1981 PF              4      4000
## 3  1981 PF              4      4000
## 4  1982 PF              4      4000
## 5  1982 PF              4      4000
## 6  1983 RM              4      4000
```

Pipe

Pipes can be implemented in R with the `dplyr` package, and the `margittr` package.

The orginal symbol of the pipe is `%>%`. However, we can also use `|>` for the same effect.

The purpose of this pipe is to eliminate or reduce the need of intermediate variables.

```
library(magrittr)
1:300 |> mean() |> sqrt() -> mean_square
```

When we use a pipeline, we don't need to plug in the variable name every time. This was a good practice run, but let's load some real data now.

Let's calculate the median year of surveys.

```
library(readr)
surveys <- read_csv("197-raw_storage/surveys.csv")
```

```
## Rows: 35549 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): species_id, sex
## dbl (7): record_id, month, day, year, plot_id, hindfoot_length, weight
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
surveys$year |> median()
```

```
## [1] 1990
```

Let's try calculating the mean of the weight. Because there are NAs in our weight column, we'll need to remove these.

```
surveys$weight |> mean(na.rm=TRUE)
```

```
## [1] 42.67243
```

Data Manipulation Practice

Sometimes it is much easier to run keep editing a data set, until it matches your intentions.

```
surveys2 <- select(surveys, year, species_id, weight) |>
  mutate(weight_kg = weight/1000) |>
  filter(!is.na(weight_kg)) |>
  select(year, species_id, weight_kg)

str(surveys2)
```

```
## tibble [32,283 x 3] (S3: tbl_df/tbl/data.frame)
##  $ year      : num [1:32283] 1977 1977 1977 1977 1977 ...
##  $ species_id: chr [1:32283] "DM" "DM" "DM" "DM" ...
##  $ weight_kg : num [1:32283] 0.04 0.048 0.029 0.046 0.036 0.052 0.008 0.022 0.035 0.007 ...
```

```
# surveys[ , c(1,3)]
# surveys[ , c("year", "weight_kg")]
```

Let's try one more example

The following code is written using intermediate variables. It obtains the data for "DS" in the "species_id" column, sorted by year, with only the year and weight columns. Write the same code to get the same output but using pipes instead.

```
ds_data <- filter(surveys, species_id == "DS", !is.na(weight)) ds_data_by_year <- arrange(ds_data, year) ds_weight_by_year <- select(ds_data_by_year, year, weight)
```

```r
filter(surveys, species_id == "DS", !is.na(weight)) |>
  arrange(year) |>
  select(year, weight) -> ds_data_by_year
  head(ds_data_by_year)
```

```
## # A tibble: 6 x 2
##    year weight
##   <dbl>  <dbl>
## 1  1977    117
## 2  1977    121
## 3  1977    115
## 4  1977    120
## 5  1977    118
## 6  1977    126
```