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 × 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 × 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. R Studio includes a shortcut for the pipe: cmd + shft + m

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 × 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)

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 × 2  
## year weight  
## <dbl> <dbl>  
## 1 1977 117  
## 2 1977 121  
## 3 1977 115  
## 4 1977 120  
## 5 1977 118  
## 6 1977 126

## What if I want to pipe to an argument other than the first argument?

str(surveys)

## spc\_tbl\_ [35,549 × 9] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ record\_id : num [1:35549] 1 2 3 4 5 6 7 8 9 10 ...  
## $ month : num [1:35549] 7 7 7 7 7 7 7 7 7 7 ...  
## $ day : num [1:35549] 16 16 16 16 16 16 16 16 16 16 ...  
## $ year : num [1:35549] 1977 1977 1977 1977 1977 ...  
## $ plot\_id : num [1:35549] 2 3 2 7 3 1 2 1 1 6 ...  
## $ species\_id : chr [1:35549] "NL" "NL" "DM" "DM" ...  
## $ sex : chr [1:35549] "M" "M" "F" "M" ...  
## $ hindfoot\_length: num [1:35549] 32 33 37 36 35 14 NA 37 34 20 ...  
## $ weight : num [1:35549] NA NA NA NA NA NA NA NA NA NA ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. record\_id = col\_double(),  
## .. month = col\_double(),  
## .. day = col\_double(),  
## .. year = col\_double(),  
## .. plot\_id = col\_double(),  
## .. species\_id = col\_character(),  
## .. sex = col\_character(),  
## .. hindfoot\_length = col\_double(),  
## .. weight = col\_double()  
## .. )  
## - attr(\*, "problems")=<externalptr>

lm(formula = weight ~ year, data = surveys)

##   
## Call:  
## lm(formula = weight ~ year, data = surveys)  
##   
## Coefficients:  
## (Intercept) year   
## 2752.137 -1.361

Sometimes, us coders are lazy. We don’t want to put in every variable detail if we can avoid it. So we use the pipeline.

surveys %>%   
 lm(formula = weight ~ year, data = \_)  
surveys %>%   
 lm(formula = weight ~ year, data = .)  
surveys |>   
 lm(formula = weight ~ year, data = \_)

## Piping Placeholders

filter(surveys, species\_id == "DS", !is.na(weight)) %>%   
 lm(formula = weight ~ year, data = .) %>%   
 summary()

##   
## Call:  
## lm(formula = weight ~ year, data = .)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -109.787 -12.440 3.723 14.886 69.886   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -709.1968 263.2510 -2.694 0.00711 \*\*  
## year 0.4184 0.1328 3.150 0.00165 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 22.86 on 2342 degrees of freedom  
## Multiple R-squared: 0.00422, Adjusted R-squared: 0.003795   
## F-statistic: 9.925 on 1 and 2342 DF, p-value: 0.001651

## Data Grouping / Data Aggregation

The function group\_by() combines rows based on *matching columns*. group\_by([data], [column])

group\_by(surveys,year)

## # A tibble: 35,549 × 9  
## # Groups: year [26]  
## record\_id month day year plot\_id species\_id sex hindfoot\_length weight  
## <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl>  
## 1 1 7 16 1977 2 NL M 32 NA  
## 2 2 7 16 1977 3 NL M 33 NA  
## 3 3 7 16 1977 2 DM F 37 NA  
## 4 4 7 16 1977 7 DM M 36 NA  
## 5 5 7 16 1977 3 DM M 35 NA  
## 6 6 7 16 1977 1 PF M 14 NA  
## 7 7 7 16 1977 2 PE F NA NA  
## 8 8 7 16 1977 1 DM M 37 NA  
## 9 9 7 16 1977 1 DM F 34 NA  
## 10 10 7 16 1977 6 PF F 20 NA  
## # … with 35,539 more rows

surveys %>%   
group\_by(year)

## # A tibble: 35,549 × 9  
## # Groups: year [26]  
## record\_id month day year plot\_id species\_id sex hindfoot\_length weight  
## <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <chr> <dbl> <dbl>  
## 1 1 7 16 1977 2 NL M 32 NA  
## 2 2 7 16 1977 3 NL M 33 NA  
## 3 3 7 16 1977 2 DM F 37 NA  
## 4 4 7 16 1977 7 DM M 36 NA  
## 5 5 7 16 1977 3 DM M 35 NA  
## 6 6 7 16 1977 1 PF M 14 NA  
## 7 7 7 16 1977 2 PE F NA NA  
## 8 8 7 16 1977 1 DM M 37 NA  
## 9 9 7 16 1977 1 DM F 34 NA  
## 10 10 7 16 1977 6 PF F 20 NA  
## # … with 35,539 more rows

surveys %>%   
group\_by(sex, year) %>%   
 summarize()

## `summarise()` has grouped output by 'sex'. You can override using the `.groups`  
## argument.

## # A tibble: 78 × 2  
## # Groups: sex [3]  
## sex year  
## <chr> <dbl>  
## 1 F 1977  
## 2 F 1978  
## 3 F 1979  
## 4 F 1980  
## 5 F 1981  
## 6 F 1982  
## 7 F 1983  
## 8 F 1984  
## 9 F 1985  
## 10 F 1986  
## # … with 68 more rows

Okay, this is an alright tool, but it’s better when we know how to use it.

group\_by(surveys, sex, year) %>%   
 summarize(count = n())

## `summarise()` has grouped output by 'sex'. You can override using the `.groups`  
## argument.

## # A tibble: 78 × 3  
## # Groups: sex [3]  
## sex year count  
## <chr> <dbl> <int>  
## 1 F 1977 204  
## 2 F 1978 503  
## 3 F 1979 327  
## 4 F 1980 605  
## 5 F 1981 631  
## 6 F 1982 823  
## 7 F 1983 771  
## 8 F 1984 445  
## 9 F 1985 636  
## 10 F 1986 414  
## # … with 68 more rows

group\_by(surveys, sex, year) %>%   
 summarize(mean = mean(weight, na.rm = TRUE))

## `summarise()` has grouped output by 'sex'. You can override using the `.groups`  
## argument.

## # A tibble: 78 × 3  
## # Groups: sex [3]  
## sex year mean  
## <chr> <dbl> <dbl>  
## 1 F 1977 47.6  
## 2 F 1978 70.0  
## 3 F 1979 65.6  
## 4 F 1980 57.4  
## 5 F 1981 63.4  
## 6 F 1982 55.4  
## 7 F 1983 55.9  
## 8 F 1984 49.0  
## 9 F 1985 47.1  
## 10 F 1986 54.7  
## # … with 68 more rows

surveys %>%   
 group\_by(species\_id) %>%   
 summarize(count = n())

## # A tibble: 49 × 2  
## species\_id count  
## <chr> <int>  
## 1 AB 303  
## 2 AH 437  
## 3 AS 2  
## 4 BA 46  
## 5 CB 50  
## 6 CM 13  
## 7 CQ 16  
## 8 CS 1  
## 9 CT 1  
## 10 CU 1  
## # … with 39 more rows

surveys %>%   
 group\_by(species\_id, year) %>%   
 summarize(count = n())

## `summarise()` has grouped output by 'species\_id'. You can override using the  
## `.groups` argument.

## # A tibble: 535 × 3  
## # Groups: species\_id [49]  
## species\_id year count  
## <chr> <dbl> <int>  
## 1 AB 1980 5  
## 2 AB 1981 7  
## 3 AB 1982 34  
## 4 AB 1983 41  
## 5 AB 1984 12  
## 6 AB 1985 14  
## 7 AB 1986 5  
## 8 AB 1987 35  
## 9 AB 1988 39  
## 10 AB 1989 31  
## # … with 525 more rows

surveys %>%   
 filter(species\_id == "DO") %>%   
 group\_by(year) %>%   
 summarize(mean = mean(weight, na.rm = TRUE))

## # A tibble: 26 × 2  
## year mean  
## <dbl> <dbl>  
## 1 1977 42.7  
## 2 1978 45   
## 3 1979 45.9  
## 4 1980 48.1  
## 5 1981 49.1  
## 6 1982 47.9  
## 7 1983 47.2  
## 8 1984 48.4  
## 9 1985 48.0  
## 10 1986 49.4  
## # … with 16 more rows