IS5 in R: Stats Starts Here (Chapter 1)

Nicholas Horton (nhorton@amherst.edu)

2025-01-02

Table of contents

Introduction and background

This document is intended to help describe how to undertake analyses introduced as examples in the Fifth Edition of *Intro Stats* (2018) by De Veaux, Velleman, and Bock. This file as well as the associated Quarto reproducible analysis source file used to create it can be found at http://nhorton.people.amherst.edu/is5.

This work leverages initiatives undertaken by Project MOSAIC (http://www.mosaic-web.org), an NSF-funded effort to improve the teaching of statistics, calculus, science and computing in the undergraduate curriculum. In particular, we utilize the `mosaic` package, which was written to simplify the use of R for introductory statistics courses. A short summary of the R needed to teach introductory statistics can be found in the mosaic package vignettes (https://cran.r-project.org/web/packages/mosaic). A paper describing the mosaic approach was published in the *R Journal*: https://journal.r-project.org/archive/2017/RJ-2017-024.

We begin by loading packages that will be required for our analyses.

```
library(mosaic)
library(tidyverse)
```

Chapter 1: Stats Starts Here

Section 1.1: What is Statistics?

Section 1.2: Data

Section 1.3: Variables

See table on page 7.

```
library(mosaic)
options(digits = 3)
Tour <- read_csv(
  "http://nhorton.people.amherst.edu/is5/data/Tour_de_France_2016.csv"
) |>
  janitor::clean_names()
```

```
Rows: 103 Columns: 12
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (4): Winner, Country, Team, Total Time(h.min.sec)
dbl (8): Year, Age, Total Time(h), Average.Speed, Stages, Total Distance Rid...

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.
```

By default, `read_csv()` prints the variable names as it reads the online file. These messages can be suppressed using the `message: false` code chunk option to save space and improve readability. (I'll ask you to do this in future as you develop more familiarity with these systems.)

The variable names coming out of spreadsheet files can sometimes be quite wonky. The results from `read_csv()` are passed to the `clean_names()` function from the `janitor` package to make them more consistent.

```
names(Tour)
```

```
 [1] "year"                "winner"              "country"
 [4] "age"                 "team"                "total_time_h_min_sec"
 [7] "total_time_h"        "average_speed"       "stages"
[10] "total_distance_ridden" "starting_riders"     "finishing_riders"
```

```r
glimpse(Tour)
```

```
Rows: 103
Columns: 12
$ year                  <dbl> 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, ~
$ winner                <chr> "Maurice Garin", "Henri Cornet", "Louis Troussel~
$ country               <chr> "France", "France", "France", "France", "France"~
$ age                   <dbl> 32, 20, 24, 27, 24, 25, 22, 21, 27, 24, 23, 24, ~
$ team                  <chr> "La Fran\x8daise", "Cycles JC", "Peugeot", "Peug~
$ total_time_h_min_sec  <chr> "94.33.00", "96.05.56", "110.26.58", "189.34.00"~
$ total_time_h          <dbl> 94.5, 96.1, 110.4, 189.6, 158.8, 156.9, 157.0, 1~
$ average_speed         <dbl> 25.7, 25.3, 27.1, 24.5, 28.5, 28.7, 28.7, 29.1, ~
$ stages                <dbl> 6, 6, 11, 13, 14, 14, 14, 15, 15, 15, 15, 15, 15~
$ total_distance_ridden <dbl> 2428, 2428, 2994, 4637, 4488, 4488, 4497, 4734, ~
$ starting_riders       <dbl> 60, 88, 60, 82, 93, 112, 150, 110, 84, 131, 140,~
$ finishing_riders      <dbl> 21, 27, 24, 14, 33, 36, 55, 41, 28, 41, 25, 54, ~
```

```r
head(Tour, 3)
```

```
# A tibble: 3 x 12
   year winner            country   age team   total_time_h_min_sec total_time_h
  <dbl> <chr>             <chr>   <dbl> <chr>  <chr>                        <dbl>
1  1903 Maurice Garin     France     32 "La F~ 94.33.00                      94.6
2  1904 Henri Cornet      France     20 "Cycl~ 96.05.56                      96.1
3  1905 Louis Trousselier France     24 "Peug~ 110.26.58                     110.
# i 5 more variables: average_speed <dbl>, stages <dbl>,
#   total_distance_ridden <dbl>, starting_riders <dbl>, finishing_riders <dbl>
```

```r
tail(Tour, 8) |>
  select(winner, year, country)
```

```
# A tibble: 8 x 3
  winner              year country
  <chr>              <dbl> <chr>
1 Contador Alberto    2009 Spain
2 Andy Schleck        2010 Luxembourg
3 Cadel Evans         2011 Australia
4 Bradley Wiggins     2012 Great Britain
5 Christopher Froome  2013 Great Britain
6 Vincezo Nibali      2014 Italy
```

```
7 Cristopher Froome   2015 Great Britain
8 Cristopher Froome   2016 Great Britain
```

Piping (`|>`) takes the output of the line of code and passes it along to the next command. We will use this to "chain" together commands to do useful things. (`%>%` is an alternative pipe operator that you may sometimes see: it works in almost the same manner.)

Let's find who was the winner in 1998

We use the `filter()` command.

```
filter(Tour, year == 1998) |>
  select(winner, year, country)
```

```
# A tibble: 1 x 3
  winner         year country
  <chr>         <dbl> <chr>
1 Marco Pantani  1998 Italy
```

Several things are noteworthy here:

1. Two equal signs are used for "comparison" (we will see that one equal sign is used for options to commands).
2. Nothing is saved from this pipeline: the output is displayed. (Later we will "assign" the output to an object that can be reused.)

How many stages were there in the tour in the year that Alberto Contador won?

We can also use the `filter()` command.

```
filter(Tour, winner == "Contador Alberto") |>
  select(winner, year, stages)
```

```
# A tibble: 2 x 3
  winner            year stages
  <chr>            <dbl>  <dbl>
1 Contador Alberto  2007     21
2 Contador Alberto  2009     21
```

Note that the following command generates the same output.

```
Tour |>
  filter(winner == "Contador Alberto") |>
  select(winner, year, stages)
```

```
# A tibble: 2 x 3
  winner            year stages
  <chr>            <dbl>  <dbl>
1 Contador Alberto  2007     21
2 Contador Alberto  2009     21
```

As does:

```
select(filter(Tour, winner == "Contador Alberto"), winner, year, stages)
```

```
# A tibble: 2 x 3
  winner            year stages
  <chr>            <dbl>  <dbl>
1 Contador Alberto  2007     21
2 Contador Alberto  2009     21
```

The pipe operator (|>) can help improve the readability of code, since each step is clearly indicated.

What was the slowest average speed of any tour? Fastest?

Again, we use `filter()` but this time in conjunction with the `min()` function.

```
Tour |>
filter(average_speed == min(average_speed)) |>
  select(year, average_speed)
```

```
# A tibble: 1 x 2
   year average_speed
  <dbl>         <dbl>
1  1919          24.1
```

```
Tour |>
filter(average_speed == max(average_speed)) |>
  select(year, average_speed)
```

```
# A tibble: 1 x 2
   year average_speed
  <dbl>         <dbl>
1  2005          41.7
```

How can we summarize the distribution of Average Speeds?

```
df_stats(~ average_speed, data = Tour)
```

```
       response  min   Q1 median   Q3  max mean  sd   n missing
1 average_speed 24.1 29.5   35.4 38.7 41.7 34.1 5.2 103       0
```

Note that `~ x` denotes the simplest form of the general modelling language (used to indicate a single variable in using the `mosaic` package).