Computational Skills for Biostatistics I: Lecture 3

Amy Willis, Biostatistics, UW

October 12, 2017

This week

- Big challenge with Homework 2: working within requirements of each function
- Another challenging assignment, more great work
- Comments from homeworks go in README files
  - Each week, look in the repository for this

RStudio

- You should save all of your work as scripts (`.R` files)
- Laying out your workspace effectively
  - Rstudio -> Preferences -> Pane Layout
- Running code quickly
  - With the cursor on the line of script you want to run...
    - `cmd + return` (Mac)
    - `ctrl + enter` (Windows)
- Commenting: precede comments by a #

RStudio

R sessions are located somewhere on your computer

```
setwd("/Users/adwillis/Documents/")
getwd() # where am I?
```

```
## [1] "/Users/adwillis/Documents"
```

```
# change to:
setwd("/Users/adwillis/Documents/teaching/17-561/lecture3/")
```

R packages

- Most packages are distributed via CRAN, a global network for the distribution of R code
  - You may need to set your "mirror"
  - RStudio -> Preferences -> Packages
- Packages need to be installed, and then loaded

```
install.packages("tidyverse") # first download...
library(tidyverse) # ...then load
```

Writing beautiful code

Suppose you have a data frame of Star Wars characters `starwars` and you want to find the average weight of each species where we have data for more than 2 characters.

```
ns <- unclass(table(starwars$species))
means <- unclass(by(starwars, starwars$species,
                    function(x) mean(x$mass, na.rm = TRUE)))
means[ns > 2]
```

```
## starwars$species
##    Droid   Gungan    Human
## 69.75000 74.00000 82.78182
```

What is lousy about this code?

Writing beautiful code

Here is another way to write that code

```
starwars %>%
  group_by(species) %>%
  summarise(n = n(),
            mass = mean(mass, na.rm = TRUE)) %>%
  filter(n > 2)
```

```
## # A tibble: 4 x 3
##   species     n     mass
##     <chr> <int>    <dbl>
## 1   Droid     5 69.75000
## 2  Gungan     3 74.00000
## 3   Human    35 82.78182
## 4    <NA>     5 48.00000
```

Which is easier to read? Which is easier to debug?

tidyverse

The `tidyverse` is a collection of packages based on 4 principles for handling data:

1. Reuse existing data structures.
2. Compose simple functions with the pipe.
3. Embrace functional programming.
4. Design for humans.

The R project for Statistical Computing was built for a different age; the tidyverse is a collection of tools for *our* age

tidyverse

Core tidyverse

The core tidyverse includes the packages that you're likely to use in every day data analyses. As of tidyverse 1.1.0, the following packages are included in the core tidyverse:

ggplot2

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Learn more ...

dplyr

dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges. Learn more ...

tidyr

tidyr provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable. Learn more ...

readr

readr provides a fast and friendly way to read rectangular data (like csv, tsv, and fwf). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes. Learn more ...

purrr

purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors. Once you master the basic concepts, purrr allows you to replace many for loops with code that is easier to write and more expressive. Learn more ...

tibble

tibble is a modern re-imaginging of the data frame, keeping what time has proven to be effective, and throwing out what it has not. Tibbles are data.frames that are lazy and surly: they do less and complain more forcing you to confront problems earlier, typically leading to cleaner, more expressive code. Learn more ...

tibbles

Data frames are great! Except for

- printing them
- working with both characters and factors
- manipulating multiple columns

tibbles are the data frame alternative of the tidyverse

tibbles

```
starwars
```

```
## # A tibble: 87 x 13
##                    name height  mass    hair_color  skin_color eye_col
##                   <chr>  <int> <dbl>         <chr>       <chr>     <ch
##  1     Luke Skywalker    172    77         blond        fair      bl
##  2               C-3PO    167    75          <NA>        gold    yell
##  3               R2-D2     96    32          <NA> white, blue       r
##  4        Darth Vader    202   136          none       white    yell
##  5        Leia Organa    150    49         brown       light     bro
##  6          Owen Lars    178   120   brown, grey       light      bl
##  7 Beru Whitesun lars    165    75         brown       light      bl
##  8               R5-D4     97    32          <NA>  white, red       r
##  9  Biggs Darklighter    183    84         black       light     bro
## 10     Obi-Wan Kenobi    182    77 auburn, white        fair blue-gr
## # ... with 77 more rows, and 7 more variables: birth_year <dbl>,
## #   gender <chr>, homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

tibbles

A tibble, or tbl_df, is a modern reimagining of the data.frame, keeping what time has proven to be effective, and throwing out what is not. Tibbles are data.frames that are lazy and surly: they do less (i.e. they don't change variable names or types, and don't do partial matching) and complain more (e.g. when a variable does not exist). This forces you to confront problems earlier, typically leading to cleaner, more expressive code. Tibbles also have an enhanced print method() which makes them easier to use with large datasets containing complex objects.

- Hadley Wickham, Chief Scientist at RStudio

How do we read code?

Translate the following code into words:

```
length(unique(starwars$species))
```

```
## [1] 38
```

Intuitive coding

```
starwars$species %>%
  unique %>%
  length
```

```
## [1] 38
```

%>% is the "pipe operator"

- `f(x)` is the same as `x %>% f`
- "Take `x` and apply the function `f`"

Intuitive coding

Using native tidyverse functions `group_by` and `summarise`/`summarize`

```
starwars %>%
  group_by(species) %>%
  summarise(n()) %>%
  nrow
```

```
## [1] 38
```

More piping

Multiple summary statistics at once

```
starwars %>%
  group_by(species) %>%
  summarise(n = n(),
            mean.mass = mean(mass, na.rm = TRUE),
            sd.mass = sd(mass, na.rm = TRUE)) %>%
  filter(n > 2)
```

```
## # A tibble: 4 x 4
##   species     n mean.mass  sd.mass
##     <chr> <int>     <dbl>    <dbl>
## 1   Droid     5  69.75000 51.03185
## 2  Gungan     3  74.00000 11.31371
## 3   Human    35  82.78182 19.38334
## 4    <NA>     5  48.00000       NA
```

dplyr

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- `mutate()` adds new variables that are functions of existing variables
- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarise()` reduces multiple values down to a single summary.
- `arrange()` changes the ordering of the rows

dplyr: data manipulation

Show only certain variables

```
starwars %>%
  select(name, ends_with("color"))
```

```
## # A tibble: 87 x 4
##                    name    hair_color  skin_color eye_colo
##                   <chr>         <chr>       <chr>     <chr
##  1     Luke Skywalker          blond        fair      blu
##  2              C-3PO           <NA>        gold    yello
##  3              R2-D2           <NA> white, blue        re
##  4        Darth Vader           none       white    yello
##  5        Leia Organa          brown       light      brow
##  6          Owen Lars   brown, grey       light      blu
##  7 Beru Whitesun lars          brown       light      blu
##  8              R5-D4           <NA>  white, red        re
##  9  Biggs Darklighter          black       light      brow
## 10     Obi-Wan Kenobi auburn, white        fair blue-gra
```

dplyr: data manipulation

Show only certain variables

```
starwars %>%
  filter(hair_color == "black",
         skin_color %in% c("fair", "light"))
```

```
## # A tibble: 3 x 13
##                 name height  mass hair_color skin_color e
##                <chr>  <int> <dbl>      <chr>      <chr>
## 1 Biggs Darklighter    183  84.0      black      light
## 2         Boba Fett    183  78.2      black       fair
## 3    Shmi Skywalker    163    NA      black       fair
## # ... with 7 more variables: birth_year <dbl>, gender <c
## #   homeworld <chr>, species <chr>, films <list>, vehicl
## #   starships <list>
```

dplyr: data manipulation

Get summary statistics

```
starwars %>%
  filter(hair_color == "black",
         skin_color %in% c("fair", "light")) %>%
  summarise("mass" = mean(mass, na.rm = T))
```

```
## # A tibble: 1 x 1
##    mass
##   <dbl>
## 1  81.1
```

`starwars` is from the package `tibble`, %>% is from the package `magrittr`, `filter` is from the package `dplyr`... Hence `tidyverse`!

Advanced piping

- `x %>% f` is equivalent to `f(x)`
- `x %>% f(y)` is equivalent to `f(x, y)`
- `x %>% f(y, .)` is equivalent to `f(y, x)`

Watch out!

```
starwars
  %>% filter(hair_color == "black",
         skin_color %in% c("fair", "light"))
  %>% summarise("mass" = mean(mass, na.rm = T))
```

```
## Error: <text>:2:3: unexpected SPECIAL
## 1: starwars
## 2:   %>%
##      ^
```

tidyverse

- Programming in R using the tidyverse will require you to unlearn some bad habits, and may be more difficult for experienced R programmers
- Learning this style will make your code more readable, debugable, and efficient
- Graduate school is the time to learn!
- I will ask you to redo homework questions if you do not write them in the style; all of your code should be using this syntax starting now!

If wishes were horses. . .

. . . we would all be eating steak:

- "I wish that my output didn't spew numbers when I type `head(my_big_data_frame)`. . . "
- "I wish there was a function to tell me which elements in my vector that satisfy my condition. . . "
- "I wish there was a function to calculate minima pointwise. . . "

Similarly, "How do I turn a data frame into a tibble?"

Coming soon

- Homework 3 due next Thursday at 2 p.m.
  - All solutions need to be written in best coding practice (i.e., tidyverse)
  - Submission via github classroom
- Homework 2 feedback coming soon
- Next week: how to make awesome plots!