BioComp498 Introduction to the tidyverse

Shawn McCafferty

3/20/2022

This is an RMarkdown document. To step through it, text in the white background is simply that, text to read that explain what the code does. The R code is in the grey background area, called R *code chunks*. These can be executed by clicking on the green right handed triangle to the far right of the code chunk box. That will execute the code showing any critical messages in the Console and any output (e.g. graphs) will appear as a new box right below the code.

Stepping through the markdown document is one way to work through the document. More interactie than the alternative, which is you can "Knit" the entire document to make an HTML file of the entire document. This will automatically step through the entire document and execute all teh R code chunks, showing the results on the the HTML doc. This can take a while, so do not do this in class. We will be stepping through it together. FOr more information on RMarkdown, see https://rmarkdown.rstudio.com.

Preparing the RStudio environment

The very first step in a new analysis using R should always be some basic housekeeping. You really want to be sure that whatever you were doing with R in the past will not impact what you are trying to do. We start by basically clearing all objects and memory usage.

```
# preliminary housekeeping
rm(list = ls(all.names = TRUE)) #will clear all objects includes hidden objects.
gc() #free up memory and report the memory usage.
```

Loading the necessary libraries

Out next step is to load the libraries we need to make the code work. I prefer to load all the libraries needed right at the start, though this is simply a style thing. A lot of people will load them as they need them.

Before we load the libraries though, we need to make sure that the proper packages (that contain the libraries) have been installed in our instance of *R*. Usually if you get an error trying to load a library, it means you did not install the appropriate package correctly.

Important note You only need to install packages once. Do not re-install them every time you run these scripts. To install, un-comment (remove the #'s) from the install lines before executing the code.

```
# install some necessary libraries
# You only need to this once. Wait for it to do its thing, it may take a little bit.
# install.packages("tidyverse")
# install.packages("palmerpenguins")
```

Once you have successfully installed the packages, load the necessary libraries as follows:

```
library(tidyverse)
library(palmerpenguins)
```

Introduction to the tidyverse

What is the tidyverse?

The tidyverse has rapidly become the go-to approach to modern data analytics (but not without controversy). If you have not heard it yet, chances are you are already familiar with it if you have used ggplot2 for making graphs. ggplot2 is part of the tidyverse, and is entirely consistent with the "tidyverse" way of doing things. So what is the "tidyverse"? You can find that answer to this question easily enough by googling it, and undoubtedly you will find an avalanche hits describing it. Here is a simple explanation that I found at *random* from https://teachdatascience.com/tidyverse/.

> The tidyverse is a coherent system of R packages for data wrangling, exploration and visualization that share a common design philosophy. These packages are intended to make statisticians and data scientists more productive by guiding them through workflows that facilitate communication, and result in reproducible work products. Unpacking the tidyverse, all that it means and contains, could easily take a dedicated book (https://r4ds.had.co.nz) or blog in itself. Instead, what this post aims to do is give you an introduction and a clear, easy path forward to learning the tidyverse. A nice introduction and motivation can be found in this RStudio post (https://rviews.rstudio.com/2017/06/08/what-is-the-tidyverse/) and Hadley Wickham's keynote address at the 2017 RStudio Conference (no link).

How does it differ from core R?

Though the tidyverse works through (overlays?) core R and is fundamentally based on core R, the syntax and how it works can be very different. The best way to describe how the tidyverse works, how it is different from core R, is to ask "Why use the tidyverse?" Again, from the same blog post quoted above:

> One of the biggest reasons to learn the tidyverse is consistency. Throughout these packages, consistency comes in three primary forms:
>
> The first formal argument of tidyverse functions is always a data frame that provides the function's input. The idea of tidy data: a data frame where each row is an observation and each column contains the value of a single variable. Here is a decent explanation of tidy data (https://www.openscapes.org/blog/2020/10/12/tidy-data/) The pipe operator, %>%, guides the flow of operations on data. See (https://r4ds.had.co.nz/pipes.html)

The consistency of the tidyverse, together with the *goal of increasing* productivity, mean that the syntax of tidyverse functions is typically straightforward to learn. Function names like **mutate**, **select**, **filter**, **group_by** and so many others do exactly what they sound like and can be combined seamlessly with the pipe operator. As Hadley Wickham often quotes of Hal Abelson, "Programs must be written for people to read and only incidentally for machines to execute." The third point above, concerning the pipe operator, deserves an introduction of its own, but for now think of it as the way to chain your commands or functions together similar to piping in unix.

The computing workflow induced by the tidyverse necessarily means increased readability and reproducibility because functions are so organically named and every task can be so easily decomposed into single-function operations. As an example, rather than using the iris dataset Prof LeBlanc showed earlier, we are going to use a newer dataset based on penguins. You should have installed the package called "palmerpenguins" above and loaded the library. Our first step is going to initialize the data set using the following:

```
data(package = 'palmerpenguins')
```

To see details about the 'palmerpenguins' dataset, see (https://allisonhorst.github.io/palmerpenguins/)

Before doing all this, let's first look at the data using the tidyverse command *glimpse*.

```
glimpse(penguins)
```

```
## Rows: 344
## Columns: 8
## $ species           <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie…
## $ island            <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgersen…
## $ bill_length_mm    <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, …
## $ bill_depth_mm     <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, …
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186…
## $ body_mass_g       <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, …
## $ sex               <fct> male, female, female, NA, female, male, female, male…
## $ year              <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007…
```

This tells us the dimensions of the data set, and has eight columns which are composed of two factors (species and island), four variables measured as real numbers (bill_length_mm, bill_depth_mm, flipper_length_mm, body_mass_g), followed by sex and year sampled. The output also gives an example of what the data looks like. Glimpse is very similar to running str(). To see how the data actually looks, run:

```
head(penguins)
```

```
## # A tibble: 6 × 8
##   species island bill_length_mm bill_depth_mm flipper_length_… body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Adelie  Torge…           39.1          18.7              181        3750 male
## 2 Adelie  Torge…           39.5          17.4              186        3800 fema…
## 3 Adelie  Torge…           40.3          18                195        3250 fema…
## 4 Adelie  Torge…           NA            NA                NA          NA <NA>
## 5 Adelie  Torge…           36.7          19.3              193        3450 fema…
## 6 Adelie  Torge…           39.3          20.6              190        3650 male
## # … with 1 more variables: year <int>
```

This is core R, but notice it first tells us the data is a tibble (we will discuss tibbles), it formats it nicely (that's part of the tibble business), and gives us the dimensions. Also notice that we have some NA in the dataset. That is R speak for missing values. It is important to recognize them.

To be complete, use *str()* to see the structure of the tibble in standard R, and *view* to see a spreadsheet-like view in a tab above.

```
str(penguins)
```

```
## tibble [344 × 8] (S3: tbl_df/tbl/data.frame)
##  $ species          : Factor w/ 3 levels "Adelie","Chinstrap",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ island           : Factor w/ 3 levels "Biscoe","Dream",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ bill_length_mm   : num [1:344] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ...
##  $ bill_depth_mm    : num [1:344] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ...
##  $ flipper_length_mm: int [1:344] 181 186 195 NA 193 190 181 195 193 190 ...
##  $ body_mass_g      : int [1:344] 3750 3800 3250 NA 3450 3650 3625 4675 3475 4250 ...
##  $ sex              : Factor w/ 2 levels "female","male": 2 1 1 NA 1 2 1 2 NA NA ...
##  $ year             : int [1:344] 2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
```

```
view(penguins)
```

To see the column names (core R):

```
colnames(penguins)
```

```
## [1] "species"           "island"            "bill_length_mm"
## [4] "bill_depth_mm"     "flipper_length_mm" "body_mass_g"
## [7] "sex"               "year"
```

And the species names (core R):

```
levels(penguins$species)
```

```
## [1] "Adelie"    "Chinstrap" "Gentoo"
```

To demonstrate how the tidyverse works, for our data set we want to:

- Selects two variables of interest, then
- Filters down the data set to two of the three species, then
- Group the data by species, followed by
- Summarizes each variable by computing means

```
penguins %>%
  select(species, bill_length_mm, bill_depth_mm) %>%
  filter(species %in% c("Adelie", "Chinstrap")) %>%
  group_by(species) %>%
  summarize(
    Avg.bill.length = mean(bill_length_mm, na.rm=TRUE),
    Avg.bill.depth= mean(bill_depth_mm, na.rm=TRUE))
```

```
## # A tibble: 2 × 3
##   species   Avg.bill.length Avg.bill.depth
##   <fct>               <dbl>          <dbl>
## 1 Adelie               38.8           18.3
## 2 Chinstrap            48.8           18.4
```

The beauty of the code above is that it can actually be read exactly as the bullets above it are read, with the pipe operator being read as "then". Contrast this to the base R implementation,which might look something like the following:

```
peng_sub <- penguins[penguins$species %in% c("Adelie", "Chinstrap"),]
rbind(tapply(peng_sub$bill_length_mm, peng_sub$species, mean),
      tapply(peng_sub$bill_depth_mm, peng_sub$species, mean))
```

```
##      Adelie Chinstrap Gentoo
## [1,]     NA  48.83382     NA
## [2,]     NA  18.42059     NA
```

And of course this does not work because I am so used to doing things the tidyverse way, I messed the code up. But you get the drift. Core R certainly gets the job done efficiently but it is much harder to parse for students who are new to R.

Why would I want to learn it?

The tidyverse is really quite powerful, and though you use standard or core R to perform just about everything you can do with the tidyverse, it makes the steep learning curve of R a little less steep. The syntax just seems a little more natural, though it does take a little used to.

To be clear, the tidyverse is really nothing more than a useful tool to get a job done. Some jobs are more efficient using the tidyverse tool chest (e.g. data wrangling and graphing), while others are best approached using core R (this seems particularly true for many higher level statistical tools). However, the two approaches (tidyverse and core R) work pretty seamlessly with each other for the most part. Though there can be some problems jumping between the two, particularly when using very field specific high-end packages that were written prior to the tidyverse, usually there is no problem working with either in the same script.

An another way to ask the question "why would I want to learn it?" is "Why Teach The Tidyverse?" And once again, from the above blog site:

> David Robinson has been a major proponent of not only teaching the tidyverse, but teaching it first. Whereas the authors of this blog and many more out there probably grew up learning base R first, the above small irs example is likely sufficient to convince a great many why the tidyverse should be taught before base R. The tidyverse is actually EASIER for students to learn than base R because the code flows naturally from the goals of the data anlaysis: subset, group, aggregate. In Robinson's words (http://varianceexplained.org/r/teach-tidyverse/), "get students doing powerful things quickly." Low barriers to entry of data analysis will resonate quite strongly with those teaching or learning statistical computing and data science. The vast majority of exploratory data analysis is accessible via the tidyverse packages.

We have *already* seen how descriptive statistics are easily calculated. We will see even more examples of this over the next two weeks.

Where the tidyverse really excels is in data wrangling. Some of the most commonly encountered data wrangling challenges involve creating new variables and sorting. Below, two new variables are created then the dataset is sorted (arranged) by them.

```
penguins %>%
  mutate(BillRatio = bill_length_mm/bill_depth_mm,
         FlipperSizeCor = flipper_length_mm/body_mass_g) %>%
  arrange(desc(BillRatio), FlipperSizeCor) %>%
  slice(1:5)
```

```
## # A tibble: 5 × 10
##   species island bill_length_mm bill_depth_mm flipper_length_… body_mass_g sex
##   <fct>   <fct>           <dbl>         <dbl>            <int>       <int> <fct>
## 1 Gentoo  Biscoe            51.3          14.2              218        5700 male
## 2 Gentoo  Biscoe            50.2          14.3              218        5700 male
## 3 Gentoo  Biscoe            59.6          17                230        6050 male
## 4 Gentoo  Biscoe            46.1          13.2              211        4500 fema…
## 5 Gentoo  Biscoe            54.3          15.7              231        5650 male
## # … with 3 more variables: year <int>, BillRatio <dbl>, FlipperSizeCor <dbl>
```

Of course there are limitations of the tidyverse, and nobody is suggesting that the tidyverse be the only set of tools taught. The philosophy and consistency of the tidyverse are still fairly young and do not sync nicely with all R packages. Some uses of R for, say, certain types of software development or advanced statistical modeling are not covered by the current collection of tidyverse packages. And so broader knowledge of base R and core computer science techniques are still needed at certain levels. However, the tidyverse is constantly evolving with more and more high level tools available. We will discuss some of these tools in next two weeks.

Some essential components of the tidyverse for data wrangling and plotting data.

The tidyverse is actually composed of a series of packages that perform specific functions all following the philosophy and format of the tidy-universe. These include the routine packages: readr, tibble, tidyr, dplyr, purrr, and ggplot2. Together these form the basis of the standard data science workflow. We have already encountered a number of commands above that are part of these tools: glimpse(), select(), filter(), group_by(), summarize(), and mutate(). Other important tools include pivot_longer and pivot_wider, which allow for the easy transition for tidy data in long format to non-tidy wide format used for many non-tidyverse packages and analyses.

There are other tidyverse packages like *stringr* and lubridate which must be loaded explicitly. stringr is incredibly useful for working with string data, and even allows the use of regex. lubridate is an essential package for working with time/date data. If you have ever struggled with dates using EXCEL or doing time series analysis, you will love lubridate.

You can find all the information you could possibly want for the tidyverse at the website https://www.tidyverse.org. We do not have the time to go over all these packages and what they can do, but it certainly is worth checking out. In addition, the tidyverse ecosystem is constantly changing, with new packages being developed for specific jobs, older packages tweaked and improved, and the entire universe evolving as it becomes more and more popular.

The purpose of this exercise is to introduce you to the basics of the tidyverse in terms of data wrangling and plotting genomics data. Our exercise will by necessity be relative simple, yet hopefully it will demonstrate the power of the tidyverse approach in bioinformatics as well as data analytics and statistics.

Some examples of using R and the tidyverse with penguins

There are a lot out there. Here are 3 to get you started.

Penguins Dataset Overview – iris alternative in R https://www.r-bloggers.com/2020/06/penguins-dataset-overview-iris-alternative-in-r/

Get started with tidymodels and #TidyTuesday Palmer penguins (machine learning) https://juliasilge.com/blog/palmer-penguins/

PCA with penguins and recipes https://allisonhorst.github.io/palmerpenguins/articles/articles/pca.html