Tidyverse Essentials

One of the big impediments to learning R in the past was the vast ecosystem of packages.

- Many ways to do the same task via competing R packages
- Most packages written by different people
- Different syntax was used in different packages
- Required lots of reading of help pages to understand how to use each package/function

Along came the `tidyverse` collection of packages! While not the most efficient method for programming, the `tidyverse` provides a coherent ecosystem for almost all common data tasks! That is,

- (Almost) all packages have functions with the same syntax
- Functions are built to work together
- A plethora of help documentation and vignettes exists

`tidyverse` Syntax

As the `tidyverse` is mostly concerned with the analysis and manipulation of data, the main data object used is a special version of a data frame called a **tibble**.

```
iris_tbl <- dplyr::as_tibble(iris)
class(iris_tbl)
```

```
[1] "tbl_df"     "tbl"        "data.frame"
```

You can see the classes of a tibble include a data frame. When R functions do method dispatch, they look through the class list from first to last. If there is a method for `tbl_df` it uses that, if not, it looks for a method for a `tbl`. If that doesn't exist, it uses a method for `data.frame`s.

```
str(iris_tbl)
```

```
tibble [150 x 5] (S3: tbl_df/tbl/data.frame)
 $ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num [1:150] 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1
1 1 1 ...
```

We see the structure looks very similar to that of a `data.frame`.

```
iris_tbl
```

```
# A tibble: 150 x 5
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>  
 1          5.1         3.5          1.4         0.2 setosa 
 2          4.9         3            1.4         0.2 setosa 
 3          4.7         3.2          1.3         0.2 setosa 
 4          4.6         3.1          1.5         0.2 setosa 
 5          5           3.6          1.4         0.2 setosa 
 6          5.4         3.9          1.7         0.4 setosa 
 7          4.6         3.4          1.4         0.3 setosa 
 8          5           3.4          1.5         0.2 setosa 
 9          4.4         2.9          1.4         0.2 setosa 
10          4.9         3.1          1.5         0.1 setosa 
# i 140 more rows
```

However, we can see the default `print()` method for a tibble (which is used when you just type an R object into the console) is not the same. We get *fancy* printing that is more useful for us and doesn't clog up our output space. We get information on the number of observations, the columns, and see only the first few rows/columns.

Almost all of the `tidyverse` functions are built to work on a tibble. That is, they usually take in a tibble and output a tibble.

- (Almost) all functions have similar syntax!
  `function_name(tibble, other_arg, ...)`
- Makes them perfect for chaining!

```
tibble |>
    function(other_arg, ...) |>
    ...
```

Note: you'll often see the chain from the `magrittr` package used (`%>%`). Due to the popularity of this operator, R implemented its own pipe recently (`|>`). At this point, the functionality is almost the same so we'll use the `BaseR` pipe since it doesn't require a package load.

`tidyverse` Packages

The `tidyverse` consists of a large number of packages. However, `library(tidyverse)` loads only the eight core packages (which sometimes load other packages of course). Those are (from their website):

- `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
- `dplyr` - dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges
- `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

- `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
- `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
- `tibble` - tibble is a modern re-imagining 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
- `stringr` - stringr provides a cohesive set of functions designed to make working with strings as easy as possible. It is built on top of stringi, which uses the ICU C library to provide fast, correct implementations of common string manipulations
- `forcats` - forcats provides a suite of useful tools that solve common problems with factors. R uses factors to handle categorical variables, variables that have a fixed and known set of possible values

We'll spend a good bit of time on `ggplot2`, `dplry`, `tidyr`, and `readr`. `tibble` will get used implicitly along the way

Recap!

`tidyverse` provides a coherent ecosystem for almost all common data tasks!

- Works on `tibbles` (special data frames)
- (Almost) all packages have functions with the same syntax
- A plethora of help documentation and vignettes exist