784 Lab 2: Review of Tidyverse and ggplot

Eric Parajon

Tidyverse Review

1. Again, it is important to save the document you are using in a central folder. Then, set your working directory to that folder. It's also important to clear your environment before working in a new document.

```
getwd() #what is your current working directory?

setwd(____) #in the parentheses, set the working directory to the folder where
#you save this file. Realistically, this isn't super relevant now because
#you're not reading in external data, but it's a good habit to get into!

rm(list=ls()) #clear global environment


#The pacman package is required to load/install the additional packages. Uncomment the line below to in
#install.packages("pacman")

#Installing and loading packages for use
pacman::p_load(tidyverse,ggthemes)
```

2.The tidyverse allows us to use the piping operator `%>%`, which is a useful piece of code (base R can do this too with |>) .

It pipes an object (either by itself or created by another function) into the next function in the sequence, as the first argument. This makes the code much easier to read.

For example, we could calculate the standard deviation of the `mpg` variable in the `mtcars` dataset in the following ways:

```
library(tidyverse)
```

```
#normal
sd(mtcars$mpg)

#piping
mtcars$mpg %>%
  sd()

#Does it return the same value?
```

That is a very simple example, of course, and using a pipe in this case does not really gain you much. However, pipes and the myriad of tools in the `tidyverse` can make our work much, much easier. The `dplyr` package in the `tidyverse` allows us to manipulate data much more easily than in base R.

`dplyr` allows us to manipulate our data in a structured way. The key functions are `mutate()`, which creates new variables, `select()`, which allows us to pick variable by characteristics of their names, `filter()`, which allows us to choose rows according to criteria we define, `summarise()`, which allows us to "reduce multiple

values down to a single summary", and `arrange()`, which allows us to order the observations according values of 1 or more variables.

RStudio makes useful cheat sheets for their packages, including `dplyr`.

Star Wars example

Joins

Pivoting

Pivoting: Wide -> Long With pivot_longer(), you can take values of a variable that are stored as variable names pivot them onto the cells of a single (implicit) variable column

Pivoting: Long -> Wide

With pivot_wider(), we move in the opposite direction: your units are distributed over multiple rows, so pivot until you have a single case per row of your tibble

Part 2: ggplot

`ggplot` is very important, and requires some time and patience to get to know well. We won't have time to dive fully into it, but we will begin reviewing the following code so you become more familiar. R has its own built-in plot function, but ggplot is far more flexible and creates far more descriptive (and thus useful!) plots.

For a comprehensive 'cheat sheet', see: https://www.maths.usyd.edu.au/u/UG/SM/STAT3022/r/current/Misc/data-visualization-2.1.pdf

To begin, we can use the 'qplot' or 'ggplot' commands. I use ggplot. It's more customization.

What are some ways we can make this plot clearer?

You can also assign this plot to an object and save that object as a .pdf file.

Themes

As you've hopefully seen ggplot is highly customization. One way to standardize your graphs and avoid repeating your code to control ancillary parameters (e.g. color palettes, axes labels, etc.) is to use a ggplot() theme that includes bundles of code. This is what we were doing using theme_bw() above.

You can also create custom themes yourself!

As you can see there are tons of options that you can set yourself. Alternatively you can use existing gplot_themes. Here are some of the most popular

Facets

Using facets lets you create multi-panel plots based off a given variable. There are two different ggplot functions to use to facet.

What are other types of common geoms?

geom_histogram(): plots a histogram of single value; usually needs modification to number of bins.

geom_density(): density plot of single variable.

geom_bar(): vertical barplot by categorical variable

geom_point(): typical two-way scatterplot

geom_smooth(): simple line interpolator

Part 3: Practice

Tidyverse practice

Use the data below to first split the complete dataset into individual planes and then summarise each plane by counting the number of flights (count = n()) and computing the mean distance flown.

Next, determine what city is flown to the most often from NYC.

Finally, create a plot to create a barplot visualizing if flights are on time or delayed (defined as later than 5 minutes) by month. Additionally, ensure your plot has meaningful labels.