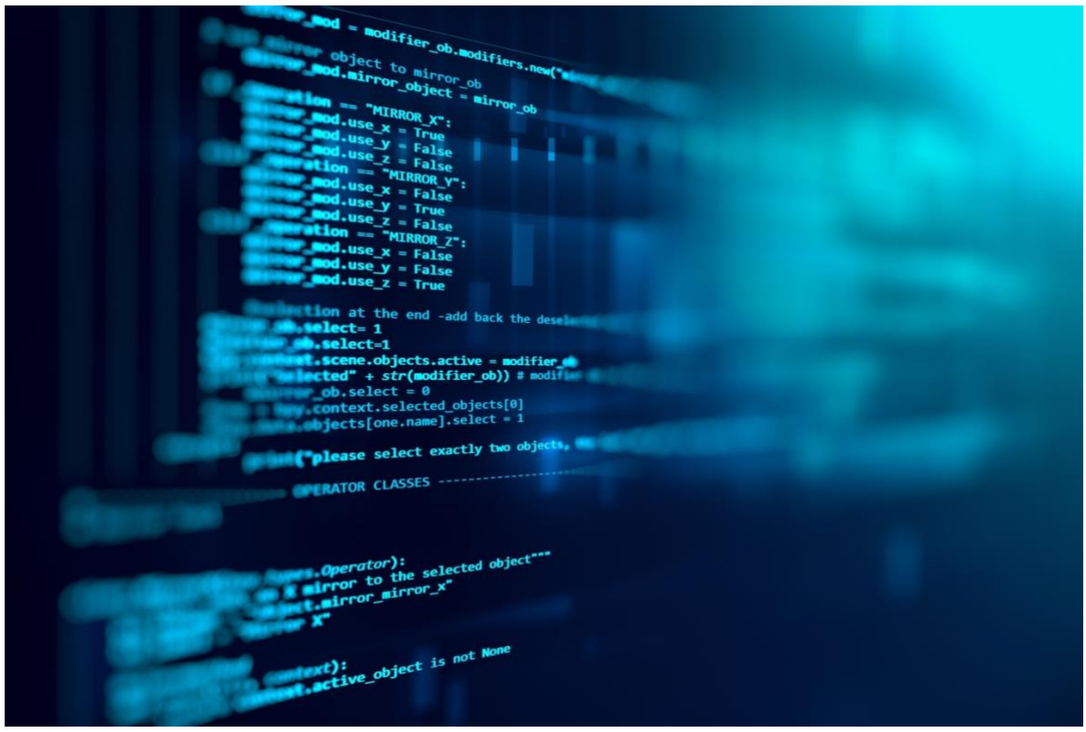
Learning to Code with R



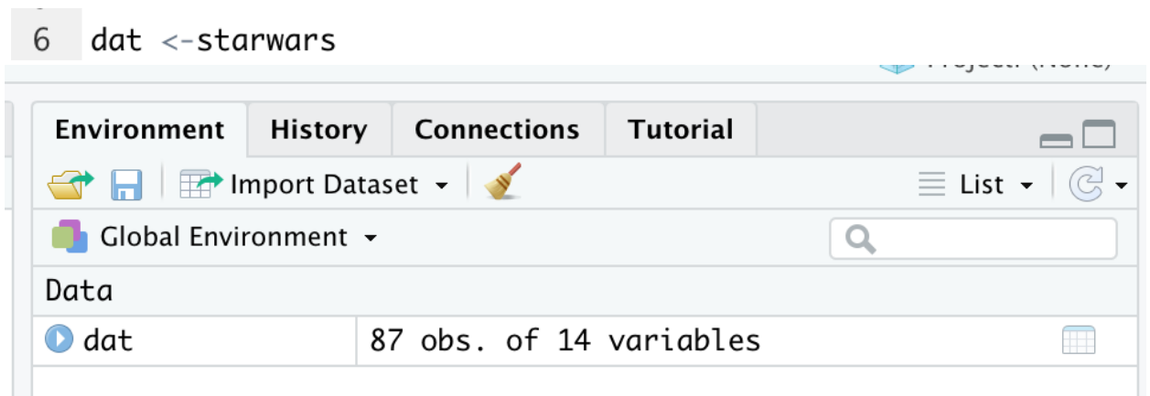
Learning how to code with a new software can be intimidating, especially for people with little to no experience in coding, like myself. I am here to tell you that R really isn't that bad, in fact it has made my life easier for managing my data! For this blog post, I'll be showing you some easy tips and tricks to simplify and easily visualize your data sets using some common functions in R! All you'll need to do is download R-studio and the R software then you're good to go.

Select and Filter

To start out we will use some data simplifying functions. Select and filter are from the dplyr package, so to load these functions, install dplyr, then call them in your library. Uncomment by removing the "#" in the image below if you have not downloaded this package.

```
1
2  ####Step 1####
3  #install.packages("dplyr")
4  library (dplyr)
5
6
```

Let's use a data set that is built into dpylr, called starwars. We will save this in our working directory as dat to make it easier to work with. "dat" should now show up in your global environment on R-studio.



To see what variables the function has, call the function “names( )” in the terminal, and enter dat within the brackets.

```
> names(dat)
 [1] "name"       "height"     "mass"       "hair_color" "skin_color" "eye_color"  "birth_year"
 [8] "sex"        "gender"     "homeworld"  "species"    "films"      "vehicles"   "starships"
```

Since this data set is pretty big, I only want to focus on the variables name, mass, homeworld, sex, and species. To do this, I will use the select function! Our new data set will be saved as “smaller.dat” by an arrow pointing towards the new name beside the function.

```
11  smaller.dat<- select(dat, name, mass, homeworld, species, sex)
```

Now let’s say we want to look at specific aspects within our data, for this example, we want to narrow down our data to only include information about humans. For this, we would use the filter function. We need to focus on the species column, and tell R to only include humans in our data set. After typing the filter function, we will tell R what data set we are working with, then type in the species variable and use “== “Human”, which will let R understand we only want to look at humans in this data frame. We will also save this as a new dataframe by naming it “filter.dat”.

```
filter.dat <- filter(smaller.dat, species == "Human")
```

Now that we know were working with one species, we can get rid of the species variable all together by using select again.

```
16  filter.dat <- select(filter.dat, name, mass, homeworld, sex) #removing the species column
```

Omitting data

You may have noticed that there are a lot of “NA” values in the starwars data set, and even our filtered data set. Sometimes, we can’t have values for every variable for every participant, and sometimes we don’t want to include this in our statistical evaluations. R has a great function for this within the tidyr package that automatically gets rid of all NA values in your data set, and it’s called “na.omit”. Let’s do an example below with the same data we’ve been working with. Don’t forget to install the new package and call it using library.

```
####Step 2####
#install.packages("tidyr")
library(tidyr)

is.na(filter.dat) #looks at missingness
omit.dat<- na.omit(filter.dat) #deletes all missing data
```

Now that we have a tidy data frame, let’s use the ggplot package to make some simple graphs to visualize our data. Install the package and call ggplot2 in the library. To make a simple bar graph we just need to know our x and y variables, and to pick a colour for the bar graph fill. The function below is the skeleton we will be using.

```
ggplot(data= df, aes(x= variable1, y= variable2)) +
  geom_bar(stat="identity", fill="colour")+
  theme_minimal()
```

Getting to know ggplot

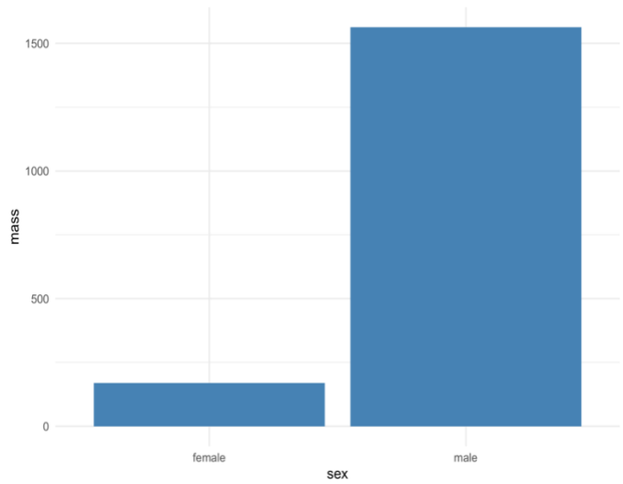
Let's make a graph to see how mass varies by sex with our new tidy data frame.

```
33  ggplot(data=omit.dat, aes(x=sex, y=mass)) +    #how mass varies by sex
34    geom_bar(stat="identity", fill="steelblue")+
35    theme_minimal()
```

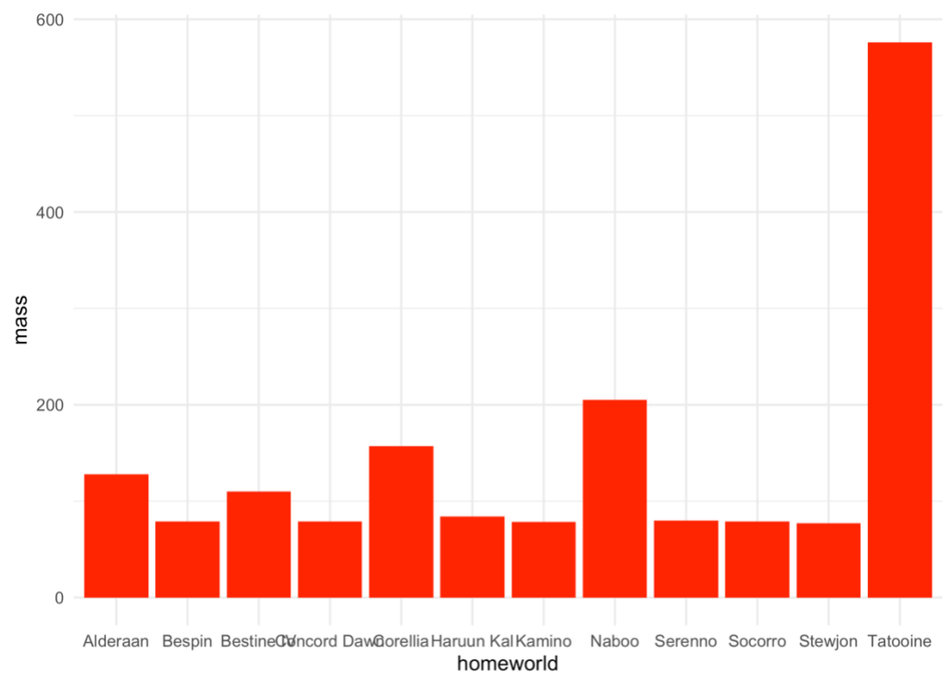
Your bar graph should show up on R-studio in the plots section. To save this as a shareable image, we use the ggsave function. First, let's save the ggplot function we previously performed, as mass.sex. The ggsave function is able to save your plot as a high quality image on your computer by setting a dpi of 300 or more, with options to edit the height and width of the photo.

```
31  mass.sex<- ggplot(data=omit.dat, aes(x=sex, y=mass)) +    #how mass varies by sex
32    geom_bar(stat="identity", fill="steelblue")+
33    theme_minimal()
34
35  ggsave(mass.sex, filename = "./mass.sex.png", width = 7, height = 5, units = "in",
36         dpi = 300) #save the file as a good quality png
```

Your image should look like mine:



Now it's your turn! Try to create a ggplot with homeworld and mass as your x and y variables, and choose red as your fill coulour. Save your function as mass.homeworld. Does your function look like mine?



Hopefully this tutorial helped ease your anxiety about learning to code. It takes practice, but once you get your head around it, it can be easy. There are limitless things you can do with R, and this is only the start!

Cashmeira Tyson