ECO 274 LAB: Data Filtering, visualization, and exporting data

Learning Objectives

- Filtering data
- Data visualization/plotting using ggplo2 package
- Demonstrate how to subset data from data structures, exporting data to the directory.

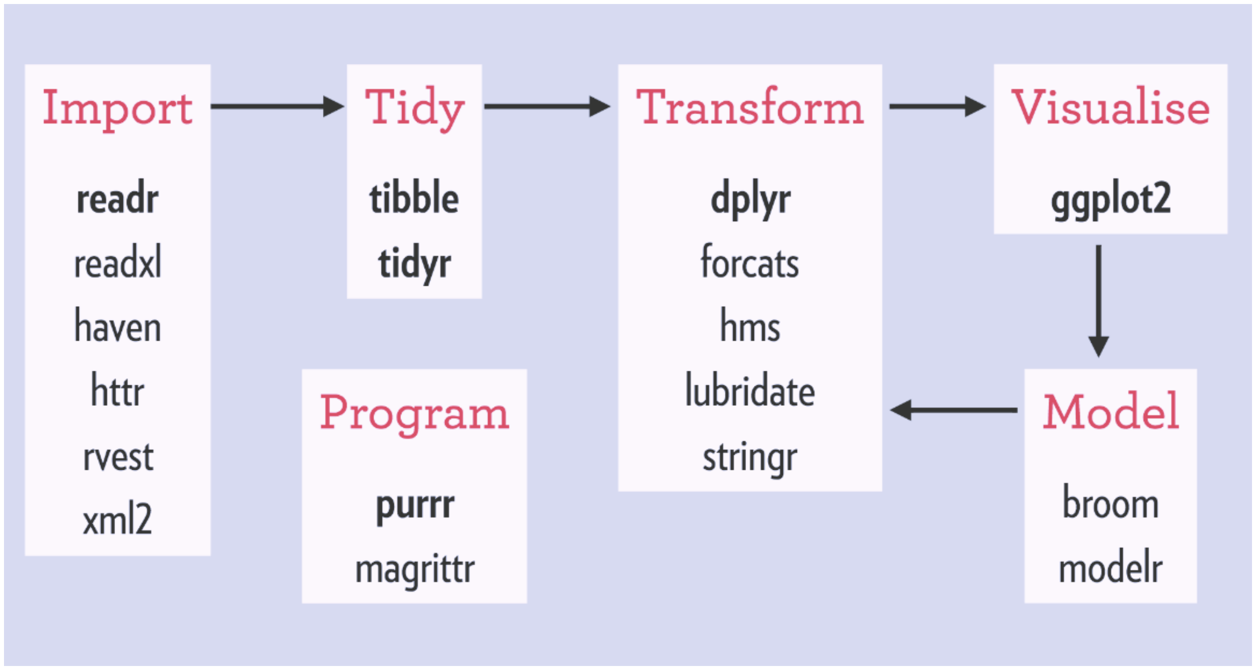
## Data filtering

This section is primarily based on tidyverse package. Data scientists spend close to 70% (if not more) of their time cleaning, massaging and preparing data. That's no secret – multiple surveys have confirmed that number. I can attest to it as well – it is simply the most time-taking aspect in a data science project.

Unfortunately, it is also among the least interesting things we do as data scientists. There is no getting around it, though. It is an inevitable part of our role. We simply cannot build powerful and accurate models without ensuring our data is well prepared.

So how can we make this phase of our job interesting?

Welcome to the wonderful world of Tidyverse! It is the most powerful collection of R packages for preparing, wrangling and visualizing data. Tidyverse has completely changed the way I work with messy data – it has actually made data cleaning and massaging fun!The tidyverse is a coherent system of packages for data manipulation, exploration and visualization that share a common design philosophy. These were mostly developed by Hadley Wickham himself, but they are now being expanded by several contributors. Tidyverse 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. Fundamentally, the tidyverse is about the connections between the tools that make the workflow possible.



First, let’s talk about PIPE operator. Simplify Your Code with %>% (ctrl+shift+M)

In R, the pipe operator is, as you have already seen, %>% . It takes the output of one function and passes it into another function as an argument. This allows us to link a sequence of analysis steps.

```
### Using pipe operator

library(gapminder)
library(tidyverse)
gapminder %>% filter(country=="Oman") %>% head(10)
gapminder %>% filter(country=="Oman" & year>1980 & year<=2000
                     ) %>% head()


data("starwars")
View(starwars)


starwars %>% filter(height>150 & mass<200) %>%
  mutate(height_in_meters=height/100) %>%
  select(height_in_meters,mass) %>%
  arrange(mass) %>%
  View()

plot()

## Exploring Data structure and variable
data("msleep")
View(msleep)
glimpse(msleep)
head(msleep)
class(msleep)
class(msleep$name)
length(msleep)
length(msleep$name)
names(msleep)
unique(msleep$vore)
missing <- !complete.cases(msleep)
msleep[missing,]


## Select variables

starwars %>%
  select(name,height,mass)

starwars %>% select(1:3)

## Changing variable name
starwars %>%
  rename("characters"="name") %>%
  head()

## filter rows
starwars %>%
  select(mass,sex) %>%
  filter(mass<55 & sex=="male")


##Recode data

starwars %>%
  select(sex) %>%
  mutate(sex=recode(sex,"male"="man", "female"="women"))


## Dealing with missing data

mean(starwars$height, na.rm = T)


## create or change a new variable
starwars %>%
  mutate(height_m=height/100) %>%
  select(name,height,height_m)


## conditional statement

starwars %>%
  mutate(height_m=height/100) %>%
  select(name,height,height_m) %>%
  mutate(tallness=if_else(height_m<1,"short","tall"))



## Describing data, use msleep data set

min(msleep$awake)
```

```
max(msleep$awake)
range(msleep$awake)
IQR(msleep$awake)
mean((msleep$awake))
median((msleep$awake))
var((msleep$awake))
summary(msleep$awake)
msleep %>% select(awake,sleep_total) %>%
  summary()
```

### Visualization

MUST READ: https://ggplot2-book.org/getting-started.html

http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html

https://www.cedricscherer.com/2019/08/05/a-ggplot2-tutorial-for-beautiful-plotting-in-r/

```
## Visualization

## The Grammar of plot or graphics: ggplot2
## data, mapping, geometry
## data must in data frame format
## Bar plots
require(ggplot2)
ggplot(data = starwars, mapping = aes(x=gender))+
  geom_bar()

## Histogram

starwars %>%
  drop_na(height) %>%
  ggplot(mapping = aes(x=height))+
  geom_histogram()


## Box plots

starwars %>%
  drop_na(height) %>%
  ggplot(mapping = aes(x=height))+
  geom_boxplot(fill="steelblue")+
  theme_bw()+
  labs(title = "Boxplot of Height",x="Height of Characters")


# density plot


starwars %>%
  drop_na(height) %>%
  filter(sex %in% c("male","female")) %>%
  ggplot(mapping = aes(x=height, color=sex, fill=sex))+
  geom_density(alpha=0.2)+
  theme_bw()+
  labs(title = "Boxplot of Height",x="Height of Characters")

# Scatter plot


starwars %>%
  filter(mass<200) %>%
  ggplot(mapping = aes(height, mass,color=sex))+
  geom_point(size=5,alpha=0.5)+
  theme_bw()+
  labs(title = "Height and mass by sex")
```

###Writing to file/save/export data file

Everything we have done so far has only modified the data in R; the files have remained unchanged. Whenever we want to save our datasets to file, we need to use a `write` function in R.

To write our matrix to file in comma separated format (.csv), we can use the `write.csv` function. There are two required arguments: the variable name of the data structure you are exporting, and the path and filename that you are exporting to. By default the delimiter is set, and columns will be separated by a comma:

```
write.csv(starwars, file="starwars2.csv")
```

Similar to reading in data, there are a wide variety of functions available allowing you to export data in specific formats. Another commonly used function is `write.table`, which allows you to specify the delimiter you wish to use. This function is commonly used to create tab-delimited files.

NOTE: Sometimes when writing a dataframe with row names to file, the column names will align starting with the row names column. To avoid this, you can include the argument `col.names = NA` when writing to file to ensure all of the column names line up with the correct column values.

Reference and acknowledgement:

1. The materials used in this lesson are adapted from work that is Copyright © Data Carpentry (http://datacarpentry.org/), data camp, data quest, Kaggle, and Harvard Chan Bioinformatics Core (HBC) under the open access terms of the Creative Commons Attribution license (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

2. The Book of R: A First Course in Programming and Statistics by Tilman M. Davies