Exploratory Data Analysis with RStudio

Noushin Nabavi

2018-10-23

Learning Goals:

- Describe the purpose of dplyr package in RStudio
- Learn dplyr commands and use them to manipulate tables
- Undertand when to use the following functions within dplyr
- Be able to use **select()** to subset columns; **filter()** to subset rows on conditions; **mutate()** to create new columns by using information from other columns; **group_by()** and **summarize()** to create summary statisitcs on grouped data; **arrange()** to sort results; **count()** to count discrete values.

A. Introduction

The focus of this lesson is to use an RStudio package to explore a data table. *dplyr* is used to wrangle, shape, or transform a dataset. By the end of this lesson, we will learn data cleaning and manipulation as well as exploratory data analysis. You'll learn how to manipulate data by filtering, sorting and summarizing and answering exploratory questions. This is a suitable introduction for people who have no previous experience in RStudio and are interested in learning to perform data analysis. We will be workign with an example data frame from Statistics Canada as part of the open Vital Statistics data throughout this lesson.

Load packages for this lesson:

```
library(dplyr)
library(ggplot2)
library(data.table)
library(Hmisc)
```

B. Data Input

Load and read the datatable into RStudio using data.table, an R package that provides an enhanced version of data.frames, which are the standard data structure for storing data in base R. In the load section below, we will create a data.table using fread(). We can also create one using the data.table() function.

Locate the working directory in RStudio:

```
setwd("~/Exploratory_R")
getwd()
```

Load the Canadian Life Expectancy datatable into RStudio:

```
data <- fread("Life_Expectancy.csv", sep = ",")
```

Explore the data structure:

```
class(data)
str(data)
nrow(data)
ncol(data)
describe(data)
print(data)
head(data)
dim(data)
names(data)
```

C. Data wrangling

Use the five *dplyr* verbs to transform or shape the data. Verbs are easy to understand by their name and include **select**, **filter**, **arrange**, **mutate**, and **summarize**. To use these verbs, we need to use the %>% [pipe] symbol which is unique to the *dplyr* package syntax.

Use the select verb to extract an observation in a dataset:

Select function returns the subset of the columns from a data frame.

```
data %>%
  select(YEAR)
```

Use the filter verb to extract an observation in a dataset:

filter() function allows to extract a subset of rows from a data frame based on logical condition.

```
data %>%
filter(YEAR == 2007, Element =="Life expectancy (in years) at age x (ex)", Sex == "Both")
```

Use the arrange verb to sort observations in ascending or descending order in a dataset:

arrange () function allow you to order the rows by one or more columns in ascending or descending order. This function can be combined with other verbs in *dplyr* package using %>%.

```
data %>%
filter(Element =="Life expectancy (in years) at age x (ex)", Sex == "Both") %>%
arrange(desc(YEAR))
```

Use the mutate verb to change a variable in the dataset:

mutate () function allow to adding new variable or transform variable in the data frame.

```
data %>%
mutate(minimum = min(AVG_VALUE))
```

Use the summarize verb to summarize a variable in the dataset:

summarise() function allow to generate or calculate summary statistics for a given columns in the data frame such as finding mean, median, minimum, maximum limits, etc.

```
data %>%
filter(Element =="Life expectancy (in years) at age x (ex)", Sex == "Both") %>%
summarise(mean = mean(AVG_VALUE))
```

A major strength of *dplyr* package is that it enables to group the data by a variable or variables and then operate on the data "by group".

```
data %>%
group_by (YEAR, Element) %>%
  summarise(mean = mean(AVG_VALUE))
```

D. Data visualization

Create a subset of data

```
data_visual <- data %>%
select(YEAR, GEO, AVG_VALUE, Element) %>%
  filter(Element == "Life expectancy (in years) at age x (ex)")
```

```
ggplot(data_visual, aes(x=GEO , y=AVG_VALUE, fill = YEAR)) + geom_line()
```

Create aesthetics in a data file

Other ggplot options: geom_line ; geom_col() ; geom_histogram() ; geom_boxplot() ; etc.

```
ggplot(data_visual, aes(x= , y=), color = continent, size = pop)
+ geom_point())
+ scale_x_log10()
+ facet_wrap(~ continents)
```

```
ggplot(data_visual, aes(x= , y=), color = continent, size = pop)
+ geom_point())
+ scale_x_log10()
+ facet_wrap(~ continents)
```