Intro to piping and Data Manipulation

JEPA

02/06/2017

Contents

Libraries and Data

```
#install.packages('dplyr')
library(dplyr)

#install.packages('tidyr')
library(tidyr)

#install.packages('ggplot2')
library(ggplot2)
```

```
Alaska <- read.csv("./Data/Alaska.csv") #Sea around Us data for Alaska
USA <- read.csv("./Data/USAP.csv") #Sea around Us data for USA
```

Dplyr and Tidyr

Despite beign separate, these two packadges work together as one. Theyr main function is to manipulate data frames and keep things "tydi". In some cases you can also make basic data creation. Both packadges follow the same syntax and can use the pipe opperator, I normally don't even know which function is from what packadge so I oftenly just call both.

Plus: Most functions are self explanatory like `select` or `filter`!

Dplyr

Arrange

The `arrange`function allows you to, literaly, arrenge your data by any value of a column

Basic structure:

New_Table <- arrange(Data, column_to_arange_by)

Note: If you eant to do from Top <- Bottom you can use `desc()` within the function

Note: when doing multiple variables the order is important since it will start with the first one

```
#You can arrange by characters (A -> Z)
Arrange_Example <- arrange(Alaska,common_name)

head(Arrange_Example[5:7], 3)
```

```
##   year scientific_name common_name
## 1 1964        Haliotis    Abalones
## 2 1964        Haliotis    Abalones
## 3 1966        Haliotis    Abalones
```

```
#You can arrange by characters (A <- Z) using desc()
Arrange_Example2 <- arrange(Alaska,desc(common_name))


head(Arrange_Example2[5:7], 3)
```

```
##   year   scientific_name         common_name
## 1 1984 Sebastes flavidus Yellowtail rockfish
## 2 1985 Sebastes flavidus Yellowtail rockfish
## 3 1987 Sebastes flavidus Yellowtail rockfish
```

```
# you can do multiple characters:
Arrange_Example3 <- arrange(Alaska,common_name,functional_group, desc(commercial_group))

head(Arrange_Example3[7:9],3)
```

```
##   common_name             functional_group commercial_group
## 1    Abalones Other demersal invertebrates         Molluscs
```

```
## 2    Abalones Other demersal invertebrates         Molluscs
## 3    Abalones Other demersal invertebrates         Molluscs
```

```
# And naturally, you can also arrange by numeric factors

Arrange_Example4 <- arrange(Alaska, uncertainty_score, desc(tonnes))

head(Arrange_Example4[4:6],3)
```

```
##   uncertainty_score year          scientific_name
## 1                 1 2010 Oncorhynchus tshawytscha
## 2                 1 1989 Oncorhynchus tshawytscha
## 3                 1 1988 Oncorhynchus tshawytscha
```

Filter

The `filter`function allows you to, literaly, filter your data by any category or numer.

Basic structure:

New_Table <- filter(Data, column_to_filter_by == "category")

```
#You can filter by character
Filter_Example <- filter(Alaska,common_name =="Clams")

head(Filter_Example[1:5], 5)
```

```
##                 area_name area_type                  data_layer
## 1 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 2 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 3 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 4 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 5 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
##   uncertainty_score year
## 1                 1 1950
## 2                 1 1951
## 3                 1 1952
## 4                 1 1953
## 5                 1 1954
```

```
#You can filter by numeric input too
Filter_Example2 <- filter(Alaska,
                          year == 2009)
head(Filter_Example2[1:5], 5)
```

```
##                 area_name area_type                  data_layer
## 1 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 2 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 3 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 4 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 5 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
##   uncertainty_score year
## 1                 4 2009
## 2                 4 2009
## 3                 2 2009
## 4                 4 2009
```

```
## 5                 4 2009
```

```
# Note: you can do =>, <= or !=

# you can do multiple characters:

Selection <- c("Clams","Octopuses")

Filter_Example3 <- filter(Alaska,common_name %in% Selection)

head(Filter_Example3[4:8], 5)
```

```
##   uncertainty_score year scientific_name common_name
## 1                 1 1950        Bivalvia       Clams
## 2                 1 1950     Octopodidae   Octopuses
## 3                 1 1951        Bivalvia       Clams
## 4                 1 1951     Octopodidae   Octopuses
## 5                 1 1952        Bivalvia       Clams
##               functional_group
## 1 Other demersal invertebrates
## 2                  Cephalopods
## 3 Other demersal invertebrates
## 4                  Cephalopods
## 5 Other demersal invertebrates
```

```
# NOTE: remember that in R there are multiple ways to get to the same result!

#Wait! What if I want to filter by multiple columns!?

Filter_Example4 <- filter(Alaska,common_name == "Clams" &
                            reporting_status =="Unreported")

#You can also filter by NA

Filter_NA_Example1 <- filter(Alaska,is.na(uncertainty_score)) #Extract only NA's
head(Filter_NA_Example1[1:4],3)
```

```
##                 area_name area_type             data_layer
## 1 USA (Alaska, Subarctic)       eez Inferred foreign catch
## 2 USA (Alaska, Subarctic)       eez Inferred foreign catch
## 3 USA (Alaska, Subarctic)       eez Inferred foreign catch
##   uncertainty_score
## 1                NA
## 2                NA
## 3                NA
```

```
Filter_NA_Example2 <- filter(Alaska,!is.na(uncertainty_score)) #Clear NA's
```

Group_by* (plus summarise)

The `group_by`function allows you to group your data by common variables for future (inmidiate) calculations. This function needs the "pipe opperator"

Basic structure:

New_Table <- Data %>% group_by(column_1,column_2...) %>% second_function()

```
#Simple group_by
Group_by_Example <- Alaska %>%
  group_by(common_name) %>%
  summarise(n()) #tells you how many rows of each common_name you have

head(Group_by_Example, 3)
```

```
## # A tibble: 3 x 2
##      common_name   n()
##           <fctr> <int>
## 1       Abalones    52
## 2  Alaska plaice     9
## 3 Alaska pollock   290
```

```
#Multiple
Group_by_Example2 <- Alaska %>%
  group_by(common_name,uncertainty_score) %>%
  summarise(n()) %>% #tells you how many rows of each common_name you have
  arrange(uncertainty_score)

head(Group_by_Example, 3)
```

```
## # A tibble: 3 x 2
##      common_name   n()
##           <fctr> <int>
## 1       Abalones    52
## 2  Alaska plaice     9
## 3 Alaska pollock   290
```

Mutate

The `mutate`function allows you to create a new column in the dataset. The new columb can have characters or numbers.

Basic structure:

New_Table <- mutate(Data, Name_New_Column = action)

```
#Functions
Mutate_Example1 <- mutate(Alaska, Log = log(tonnes))

head(Mutate_Example1[13:16], 3)
```

```
##   reporting_status    tonnes landed_value      Log
## 1       Unreported   13.8030      20235.2 2.624886
## 2         Reported 1483.9740    2175505.9 7.302479
## 3       Unreported  389.9891     571724.0 5.966119
```

```
#In data calculations (per row)
Mutate_Example2 <- mutate(Alaska, Price_plus_Ton = (landed_value+tonnes))

head(Mutate_Example2[13:16], 3)
```

```
##   reporting_status    tonnes landed_value Price_plus_Ton
```

```
## 1       Unreported   13.8030      20235.2          20249
## 2         Reported 1483.9740    2175505.9        2176990
## 3       Unreported  389.9891     571724.0         572114
```

```
#Or characters...
Mutate_Example3 <- mutate(Alaska, Country = "USA")

head(Mutate_Example3[13:16], 3)
```

```
##   reporting_status    tonnes landed_value Country
## 1       Unreported   13.8030      20235.2     USA
## 2         Reported 1483.9740    2175505.9     USA
## 3       Unreported  389.9891     571724.0     USA
```

```
Mutate_Example4 <- mutate(Mutate_Example3, Country = paste("In",year,Country,"harvested",round(tonnes,2)

paste(Mutate_Example4[1,16])
```

```
## [1] "In 1950 USA harvested 13.8 tonnes of Marine fishes nei"
```

```
paste(Mutate_Example4[5387,16])
```

```
## [1] "In 1979 USA harvested 18.7 tonnes of Squids"
```

select

The `select`function is one of those "of-course it does that" function cus it allows you to, wait for it... SELECT any column you want.

Basic structure:

New_Table <- select(Data,number or name of colum)

```
#Select by column number
Select_Example1 <- select(Alaska, 6)

head(Select_Example1,3)
```

```
##                scientific_name
## 1 Marine fishes not identified
## 2 Marine fishes not identified
## 3 Marine fishes not identified
```

```
#Select by multiple column numbers
Select_Example2 <- select(Alaska, 4,5,6,7)

head(Select_Example2, 3)
```

```
##   uncertainty_score year              scientific_name       common_name
## 1                 1 1950 Marine fishes not identified Marine fishes nei
## 2                 3 1950 Marine fishes not identified Marine fishes nei
## 3                 3 1950 Marine fishes not identified Marine fishes nei
```

```
# You can also do (4:7) and even (4:6,15)

#Select by name
Select_Example3 <- select(Alaska, area_name,year,scientific_name,tonnes)
```

```r
head(Select_Example3, 3)
```

```
##                 area_name year              scientific_name    tonnes
## 1 USA (Alaska, Subarctic) 1950 Marine fishes not identified   13.8030
## 2 USA (Alaska, Subarctic) 1950 Marine fishes not identified 1483.9740
## 3 USA (Alaska, Subarctic) 1950 Marine fishes not identified  389.9891
```

```r
# You can substract columns from a dataframe

Select_Example4 <- select(Select_Example3, -area_name,year)

head(Select_Example4, 3)
```

```
##   year              scientific_name    tonnes
## 1 1950 Marine fishes not identified   13.8030
## 2 1950 Marine fishes not identified 1483.9740
## 3 1950 Marine fishes not identified  389.9891
```

```r
#Note, you can also substract using -1

#And you can also re-order your columns!

Select_Example5 <- select(Select_Example3, scientific_name,year,tonnes,area_name)

head(Select_Example5, 3)
```

```
##                scientific_name year    tonnes               area_name
## 1 Marine fishes not identified 1950   13.8030 USA (Alaska, Subarctic)
## 2 Marine fishes not identified 1950 1483.9740 USA (Alaska, Subarctic)
## 3 Marine fishes not identified 1950  389.9891 USA (Alaska, Subarctic)
```

slice

The `slice`function works like the `select`function but for rows. So, if you want to extract an specific row, a set of rows, or a range between values, use slice!

Basic Structure

New_Data <- slice(Old_Data, number)

```r
#Select by row number
Slice_Example1 <- slice(Alaska, 3948)

Slice_Example1
```

```
##                 area_name area_type                   data_layer
## 1 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
##   uncertainty_score year          scientific_name     common_name
## 1                 3 1973 Clupea pallasii pallasii Pacific herring
##              functional_group commercial_group fishing_entity
## 1 Medium pelagics (30 - 89 cm)    Herring-likes            USA
##   fishing_sector catch_type reporting_status  tonnes landed_value
## 1     Industrial   Landings         Reported 15792.9     23152391
```

```r
#Select by multiple rows
Slice_Example2 <- slice(Alaska, 1000:3948)

head(Slice_Example2, 3)
```

```
##                 area_name area_type                   data_layer
## 1 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 2 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 3 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
##   uncertainty_score year          scientific_name     common_name
## 1                 3 1957  Hippoglossus stenolepis Pacific halibut
## 2                 1 1957  Hippoglossus stenolepis Pacific halibut
## 3                 3 1957 Clupea pallasii pallasii Pacific herring
##               functional_group commercial_group fishing_entity
## 1   Large flatfishes (>=90 cm)       Flatfishes            USA
## 2   Large flatfishes (>=90 cm)       Flatfishes            USA
## 3 Medium pelagics (30 - 89 cm)    Herring-likes            USA
##   fishing_sector catch_type reporting_status      tonnes landed_value
## 1      Artisanal   Landings         Reported 12564.60000  18419703.60
## 2   Recreational   Landings       Unreported    11.14694     16341.42
## 3     Industrial   Landings         Reported 53656.10001  78659842.61
```

Joining Data with dplyr

The "bind" family

bind_cols

```r
#Lets just asume that we have two different data sets
Data1 <- select(Alaska, 1)
Data2 <- select(Alaska, 2)

#Now we bind the columns together
Bind_Cols_1 <- bind_cols(Data1,Data2)

head(Bind_Cols_1, 3)
```

```
##                 area_name area_type
## 1 USA (Alaska, Subarctic)       eez
## 2 USA (Alaska, Subarctic)       eez
## 3 USA (Alaska, Subarctic)       eez
```

bind_rows

```r
#Lets just asume that we have two different data sets
Data1 <- slice(Alaska, 1:3)
Data2 <- slice(Alaska, 10800:10802)

#Now we bind the columns together
Bind_Row_1 <- bind_cols(Data1,Data2)

head(Bind_Row_1, 6)
```

```
##                 area_name area_type                   data_layer
## 1 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 2 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 3 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
##   uncertainty_score year              scientific_name       common_name
## 1                 1 1950 Marine fishes not identified Marine fishes nei
## 2                 3 1950 Marine fishes not identified Marine fishes nei
## 3                 3 1950 Marine fishes not identified Marine fishes nei
##                functional_group       commercial_group fishing_entity
## 1 Medium demersals (30 - 89 cm) Other fishes & inverts            USA
## 2 Medium demersals (30 - 89 cm) Other fishes & inverts            USA
## 3 Medium demersals (30 - 89 cm) Other fishes & inverts            USA
##   fishing_sector catch_type reporting_status    tonnes landed_value
## 1    Subsistence   Landings       Unreported   13.8030      20235.2
## 2      Artisanal   Landings         Reported 1483.9740    2175505.9
## 3      Artisanal   Landings       Unreported  389.9891     571724.0
##                 area_name area_type                   data_layer
## 1 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 2 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
## 3 USA (Alaska, Subarctic)       eez Reconstructed domestic catch
##   uncertainty_score year    scientific_name common_name
## 1                 4 2009 Anoplopoma fimbria   Sablefish
## 2                 2 2009 Anoplopoma fimbria   Sablefish
## 3                 4 2009 Anoplopoma fimbria   Sablefish
##                functional_group commercial_group fishing_entity
## 1 Large bathydemersals (>=90 cm)   Scorpionfishes            USA
## 2 Large bathydemersals (>=90 cm)   Scorpionfishes            USA
## 3 Large bathydemersals (>=90 cm)   Scorpionfishes            USA
##   fishing_sector catch_type reporting_status       tonnes landed_value
## 1     Industrial   Landings         Reported  1074.516856  1575241.711
## 2    Subsistence   Landings       Unreported     5.002588     7333.794
## 3      Artisanal   Landings         Reported 11175.083144 16382671.889
```

The "join" family

anti_join

```
#Lets asume we want to know how many species are fished in Alaska and not in the continental US
Similar_Species <- anti_join(Alaska, USA, by="scientific_name")

#You can also do it by more than one variable
Similar_Species2 <- anti_join(Alaska, USA, by=c("scientific_name","reporting_status"))
```

semi_join

```
#Now we want to know how many species are fished in BOTH Alaska and the continental US
Diff_Species <- semi_join(Alaska, USA, by="scientific_name")

#Not just like anti_join, you can do it for more than one variable
```

inner_join

left_join

right_join

intersect

union

setdiff

The Piping opperator %>%

Many R packadges like `dplyr`, `tidyr` and `leaflet`, allows you to use the pipe (%>%) operator to chain functions together. Chaining code allows you to streamline your workflow and make it easier to read.

When using the %>% operator, first specify the data frame that all following functions will use. For the rest of the chain the data frame argument can be omitted from the remaining functions.

NOTE: for Mac users the pipe simbol "%>%" shortcut is command + shit + m