Volpe R Course: Session 5, Data Manipulation

Instructors: Don Fisher, Dan Flynn, Jessie Yang
Course webpage: `http://bit.ly/volpeR`

5/11/2017

Overview

Last session: Communicating results

- Basic graphics
- `ggplot2` package
- `Rmarkdown` introduction

Today: Data manipulation

- Merging data frames
- Basic manipulations
- Plotting spatial data
- Explaining your reports!

Merging data frames

Merging data between different sources requires having a common column name or row name. The two functions most often used are `merge` and `match`. Type the following:

```
example(merge)
authors
books

merge(authors, books, by.x = "surname", by.y = "name")
# and do it again again, adding `all.y = TRUE`
```

We can use match to carry out a similar operation. Match is more flexible than merge, and therefore more difficult to use!

```
m1 <- match(authors$surname, books$name) # returns index for the second element

books[m1,]

newdata <- data.frame(authors, books[m1,])
```

Merging data frames

`match` can also be used in many other contexts, besides merging data frames. Additional functions for matching to know are `which`, `grep`, and `intersect`.

```
which(books$name == "Ripley")
```

```
## [1] 4 5
```

```
grep("Rip", books$name) # partial string matching
```

```
## [1] 4 5
```

```
intersect(books$name, authors$surname)
```

```
## [1] "Tukey"    "Venables" "Tierney"  "Ripley"   "McNeil"
```

```
# unique shared elements, order unimportant.
```

Merging data frames

We will work with two data files, on the course Google Drive, `http://bit.ly/volpeR`:
- `US_State_Avg_Ann_Temp.csv` -
`Net_generation_for_all_fuels_(utility-scale)_monthly.csv`

Download them and read them in. For the Fuel generation file, downloaded from the Energy Information Administration, we need to skip several lines at the top:

```
avg.temp <- read.csv("data/US_State_Avg_Ann_Temp.csv")
net.fuel <-
  read.csv("data/Net_generation_for_all_fuels_(utility-scale)_monthly.csv",
                       skip = 4)
```

The fuel data also needs some cleaning up with the file names, which we can do using `sub`:

```
names(net.fuel)
names(net.fuel) <- sub("...all.sectors.thousand.megawatthours", # Pattern
                       "",                                      # Replacement
                       names(net.fuel))                         # Vector
names(net.fuel)
```

Regular expressions

The use of `sub` above is an example of *regular expression*. These are very flexible and form the basis for all search engines. In R, there are number of ways to use regular expressions, with `grep` and `sub` being two of the most common ones. `regexpr` also provides additional options for those familiar with Perl or Python.

```
text1 <- c("Testing", "matching ", " and subustitutions")
sub(" +$", "", text1) # find and replace trailing whitespace
```

```
## [1] "Testing"              "matching"             " and subustitutions"
```

```
grep("^ ", text1) # find elements which start with a whitespace
```

```
## [1] 3
```

```
grep("^[A-Z]", text1) # find elements which start with a capital letter
```

```
## [1] 1
```

Exercise: regular expressions

Use the `authors` data frame and come up with a two ways to select the rows containing "Australia" in the `nationality` column.

Merging data frames

Using the example data, let's reshape and merge the data together.

```
library(reshape)
net.melt <- melt(net.fuel)


## Using Month as id variables
```

```
names(net.melt) = c("Month", "Region", "NetFuel")

avg.temp$Region <- gsub(" ", ".", avg.temp$Region)
# gsub matches multiple instances

fuel.temp <- merge(avg.temp, net.melt, all.y = T)
```

Aggregating data

Three functions are most useful for aggregating data: `aggregate`, `tapply`, and `apply`. Using the merged file, create mean net fuel consumption tables:

```
# Two index vectors, using aggregate
aggregate(fuel.temp$NetFuel,
          by = list(fuel.temp$Region, fuel.temp$State), FUN = mean)

# Or with only one index vector, using tapply
tapply(fuel.temp$NetFuel, fuel.temp$Region, mean)
```

Use apply to look across multiple columns or rows

```
apply(fuel.temp[3:4],      # data frame to look at (should be two-dimensional)
      2,                   # 1 is for rows, 2 is for columns
      mean)                # Function
```

Exercise:

Create a table of *annual* net fuel consumption for each region.

Merging data for spatial analysis

One application of the merge data is to plot it spatially.

```
library(maps)
map('state')
# Plot avg temp
statenames  <- map('state', plot = FALSE)$names
statematches <- match(statenames, tolower(avg.temp$State))
tempmatches <- avg.temp$Avg.degF[statematches]

colorpal <- heat.colors(100)

temp.colors <- colorpal[round(tempmatches)]

map('state',
    col = temp.colors,
    fill = T)
```

Exercise

Look at the states where the colors were not matched. Why did the names not match? Compare `statenames` and `avg.temp$State`.

Saving your work

After doing all this work, you will want to save the results. You can save R objects in your current workspace to file type called `.RData`. Then you can load these R objects back in at a later session, without having to carry out this preparation work.

For example, to save all objects in the current workspace:

```
save(list = ls(), file = "My_Prepped_Data.RData")
```

and load it back in using

```
load("My_Prepped_Data.RData")
```

To save specific R objects, you will name them in the list as strings, such as

```
save(list = c('fuel.temp', 'avg.temp'), file = "My_Prepped_Data.RData")
```

You can also consider making a script that does all this preparation work, and save it as a file called something like `Analysis Prep.R`. Then, in your data analysis script or `.Rmd` file, you can automatically run that script by inserting the command: `source('Analysis Prep.R')`.

Explaining your reports!

Given remaining time, take a few minutes to work with the person next to you on your Markdown report.

Show them the report you generated for this session (using `.Rmd`) and explain the story it tells. Show them how you used Markdown in your case, and get their feedback on elements to edit or improve. Leave enough time for both people to present.

Homework

Now that you have mastered all these elements of R, finalize your Markdown report to present a compelling story. A few of you will be asked to present these analyses in the final session.

The last session will also briefly cover two statistical topics which seem to be of interest for several of you:

- hierarchical modeling
- Time series and repeated measures modeling

Homework review: Loops

Loop to generate random subsets of your data:

```
# Get 10 samples of data from each species:
sampx <- vector()
for(i in levels(iris$Species)){
  iris.sp.i <- iris[iris$Species == i,]
  sampx <- rbind(sampx, iris.sp.i[sample(rownames(iris.sp.i), 10),])
  }
```

Loop over rows in a data frame and perform some operation:

```
# Subtract mean from each
means <- apply(iris[,1:4], 2, mean)
newdat <- vector()
for(i in 1:nrow(iris)){
  newdat.x <- iris[i,1:4] - means
  newdat <- rbind(newdat, data.frame(newdat.x, iris[i,5]))
  }
# Examine it: hist(newdat[,2]); hist(iris[,2])
```