Fundamentals in R: Extended Functionality

August 12, 2014

Module 5 - Extended Functionality

In this module we'll introduce how to extend the basic functionality of R by installing and loading additional packages. We'll introduce the most popular data visualization package in R.

The objectives are

- install and load R packages
- get a feel for ggplot2
- put your acquired skills to good use

Extending R

All of the functionality we've explored thus far ships standard with R. The beauty of R lies in it being *open source* and *extendable*. Additional functionality is provided by functions shipped in packages. Packages must be installed once (for every R version install) and loaded (every time you start R).

Identifying packages for download:

Search rseek.org Browse CRAN Task Views

Exercise

- Identify a business/analytics problem you would like to tackle with R (examples: importing data into R from Excel, running a logistic regression in R, R with Google Motion Charts)
- Search the community
  - Who else has been working on this? r-bloggers.com
  - What package(s) could prove useful? Search rseek.org
  - Can you leverage code that's already been written?
- Identify a package to download

Load Your Work

```
load("data/Module3_songData.RData")
load("data/Module4_songSubset3.RData")
```

Installing, Loading, Help

```r
install.packages("ggplot2")  # Install a package

library("ggplot2")  # Load a package

load("data/Module4_songSubset3.RData")

help(package = "ggplot2")  # Get help on a package

help("map_data")  # Get help for a specific function

# ... or ?map_data
```

ggplot2

ggplot2 is the most popular package for data visualization in R. It's syntax may be very different from anything you've worked with in the past but it is precisely what gives ggplot it flexibility and overall ease of use.

To see examples of how others have used this package/function, a recommended resource is www.rseek.org search. Here's an example.

ggplot2

```
mapWorld <- map_data(map = "world")

# What does map_data() function do?

# ?map_data
```

ggplot2

```r
# What type of object is mapWorld?

str(mapWorld)

## 'data.frame':    25553 obs. of  6 variables:
##  $ long     : num  -133 -132 -132 -132 -130 ...
##  $ lat      : num  58.4 57.2 57 56.7 56.1 ...
##  $ group    : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ order    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ region   : chr  "Canada" "Canada" "Canada" "Canada" ...
##  $ subregion: chr  NA NA NA NA ...

# What type of object is mapWorld?

head(mapWorld)

##      long   lat group order region subregion
## 1 -133.4 58.42     1     1 Canada      <NA>
```

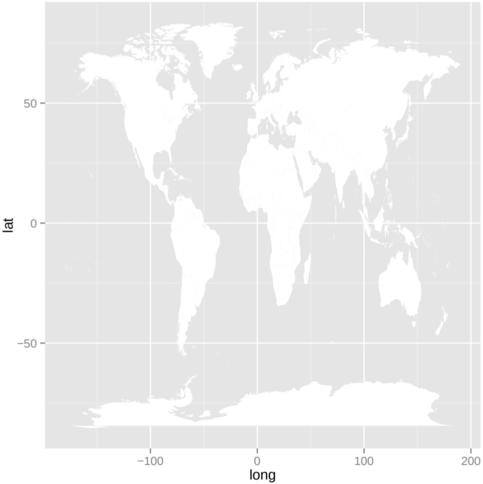
ggplot2

ggplot2 can take the data frame generated by map_data and plot it as polygons using latitude and longitude coordinates as well as group information.

```
songMap <- ggplot() + geom_polygon(data = mapWorld, aes(x = long, y = la
    fill = "white")

songMap
```

ggplot2

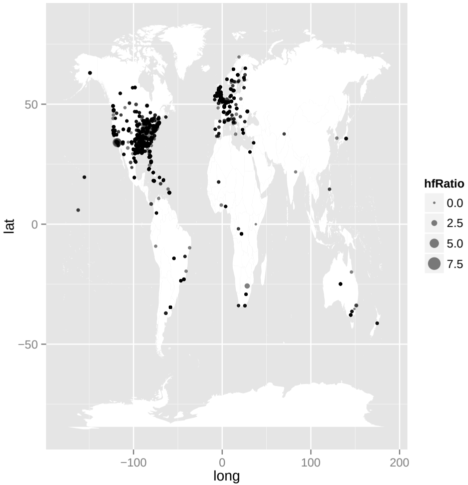


ggplot2

We can add data to our map. Each point will represent an observation in our song data frame (one song). The size of the point will be relative to the hfRatio.

```
songMap + geom_point(data = songSubset3, aes(x = artist.longitude, y = a
    size = hfRatio), alpha = I(0.5))
```

ggplot2



How can we improve this map?

Putting it all together

Putting it all together, let’s plot plot average hfRatio in the US by state – identifying states with up and coming artists.

Exercise

From songSubset3, remove any rows with NA's or NaN's for variables we'll be plotting. You will need to:

1. Identify which variables we need to screen for NA's and NaN's
2. Use !is.na() to identify observations in our data (songSubset3) where those variables *are not NA or NaN*
3. Use [] to extract a subset data frame (call it songState) based on the information ortained in 2.

Solution

```
songState <- songSubset3[!is.na(songSubset3$artist.hotttnesss) & !is.na
    !is.na(songSubset3$song.hotttnesss) & !is.na(songSubset3$artist.lat
    !is.na(songSubset3$artist.longitude), ]

dim(songSubset3)

## [1] 3518   14

dim(songState)

## [1] 2210   14
```

Solution

```
# Note, we can identify (remove) all rows where obervations of ANY of t
# variables are NA using complete.cases (na.omit).

dim(na.omit(songSubset3))

## [1] 1435   14

dim(songSubset3[complete.cases(songSubset3), ])

## [1] 1435   14
```

Putting it all together

We want to plot hotness and familiarity variables by state. Which variables can we use to do that?

```
View(songState)
```

Putting it all together

What are the next steps for plotting average hotness and familiarity variables by US state?

Putting it all together

1. Subset data to only include US
2. Aggregate hotness and familiarity variables by State

Exercise

1. Subset songState to only include rows where geoCountry is “United States”. Save this as an object called songStateSub. Hint: use [] or subset(). Under which state do most of our observations fall? How many observations?

Solution

```
songStateSub <- subset(songState, geoCountry == "United States")

summary(songStateSub)
```

```
## artist.latitude artist.longitude         geoCountry        geoState
## Min.   :19.6    Min.   :-155.4   United States:1448   California:263
## 1st Qu.:34.0    1st Qu.:-100.1   Afghanistan  :   0   New York  :217
## Median :37.3    Median : -87.7   Argentina    :   0   Texas     :120
## Mean   :37.1    Mean   : -92.0   Australia    :   0   Tennessee : 68
## 3rd Qu.:40.7    3rd Qu.: -77.3   Austria      :   0   Illinois  : 53
...
```

Exercise

2. Aggregate songState hfRatio variable by calculation the mean over the geoState variable. Call this data set songStateAgg.

Solution

```
songStateAgg <- aggregate(hfRatio ~ geoState, data = songStateSub, mean
```

Putting it all together

To generate the visualization of, say, average hfRatio by state, we will walk through a similar process as before.

As before, we use map_data to obtain a data frame which ggplot2 can use to plot. This time, we obtain the state map which corresponds to US states.

```r
usStateMap <- map_data("state")
head(usStateMap)
```

```
##      long   lat group order  region subregion
## 1 -87.46 30.39     1     1 alabama      <NA>
## 2 -87.48 30.37     1     2 alabama      <NA>
## 3 -87.53 30.37     1     3 alabama      <NA>
## 4 -87.53 30.33     1     4 alabama      <NA>
## 5 -87.57 30.33     1     5 alabama      <NA>
## 6 -87.59 30.33     1     6 alabama      <NA>
```

Exercise

This time, however, because we are plotting the songhottness variable by state and not simply by lat/long coordinates, we need to join our data with this map data frame on state name.

Join usStateMap and songStateAgg on the state names.

Solution

```
mapData <- merge(songStateAgg, usStateMap, by.x = "geoState", by.y = "re
```

This merge failed. Why? What do we need to ensure a successful merge?

```
head(mapData)
```

```
## [1] geoState  hfRatio   long      lat       group     order     subregion
## <0 rows> (or 0-length row.names)
```

Solution part 2

```
head(usStateMap, 2)
```

```
##     long   lat group order  region subregion
## 1 -87.46 30.39     1     1 alabama      <NA>
## 2 -87.48 30.37     1     2 alabama      <NA>
```

```
head(songStateAgg, 2)
```

```
##   geoState hfRatio
## 1  Alabama  0.6883
## 2   Alaska  0.8668
```

```
songStateAgg$geoState <- tolower(songStateAgg$geoState)
```

Solution part 2

```r
mapData <- merge(songStateAgg, usStateMap, by.x = "geoState", by.y = "re
    all.y = TRUE)
head(mapData)
```

```
##   geoState hfRatio   long   lat group order subregion
## 1  alabama  0.6883 -87.46 30.39     1     1      <NA>
## 2  alabama  0.6883 -87.48 30.37     1     2      <NA>
## 3  alabama  0.6883 -85.10 32.64     1   119      <NA>
## 4  alabama  0.6883 -85.07 32.61     1   120      <NA>
## 5  alabama  0.6883 -85.05 32.56     1   121      <NA>
## 6  alabama  0.6883 -87.53 30.37     1     3      <NA>
```

Putting it all together

Notice that in merging, our data frame got a little bit jumbled up. We'll re-order the observations based on the order variable.

```
head(usStateMap, 3)
```

```
##      long   lat group order  region subregion
## 1 -87.46 30.39     1     1 alabama      <NA>
## 2 -87.48 30.37     1     2 alabama      <NA>
## 3 -87.53 30.37     1     3 alabama      <NA>
```

```
head(mapData, 3)
```

```
##   geoState hfRatio   long   lat group order subregion
## 1  alabama  0.6883 -87.46 30.39     1     1      <NA>
## 2  alabama  0.6883 -87.48 30.37     1     2      <NA>
## 3  alabama  0.6883 -85.10 32.64     1   119      <NA>
```

```
mapData <- mapData[order(mapData$order), ]
```

Putting it all together

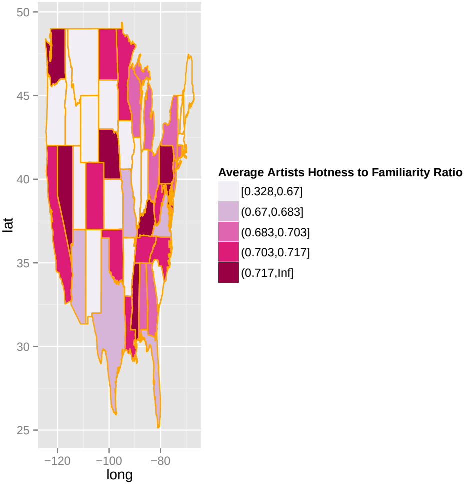
Our ggplot code will be very much the same as it was before. Instead of geom_point, we will use geom_polygon as we are plotting states, not coordinates.

```
(p1 <- ggplot(data = mapData, aes(x = long, y = lat, group = group)) + g
    5))))

(p1 <- p1 + geom_path(colour = "orange", linestyle = 2))

(p1 <- p1 + scale_fill_brewer("Average Artists Hotness to Familiarity Ra
    palette = "PuRd"))
```

Putting it all together



Putting it all together

We add state names (abbrev.) using a built in dataset containing lat/long state centers. We clean up our map, removing axes and background. We save the image to a jpeg file using ggsave().

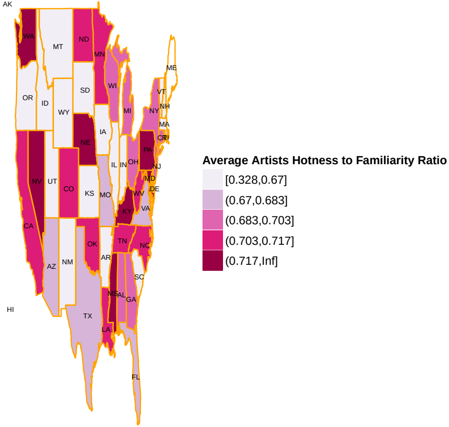
```
states <- data.frame(state.center, state.abb)

(p1 <- p1 + geom_text(data = states, aes(x = x, y = y, label = state.ab
    size = 2))

(p1 <- p1 + theme(line = element_blank(), axis.text = element_blank(),
    panel.background = element_blank()))

ggsave(filename = "p1.jpeg", width = 11, height = 8.5)
```

Putting it all together



Where to next?

Now that we know which states, on average, are associated with the most up & coming artists by average rating, what else might we want to investigate?

Where to next?

How does number of observations vary across states? How can we incorporate this information into our plot?

How do the hotttness and familiarity variables vary by release year? How does this affect the ratio we calculated?

...

Thank You!

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com, 1.855.GET.REVO, Twitter: @RevolutionR