Manipulating Spatial Data Using dplyr

Post 02 - Stat 133, Fall 2017

Will Freyman

November 27, 2017

Introduction

`dplyr` is a handy `R` package for manipulating data. Released by Hadley Wickham in 2016 (Wickham and Francois 2016), `dplyr` takes the concept behind the "grammar of graphics," in which a structured language is used to construct visual graphics, and applies it to create a structured language for data wrangling. Wickham has a lot of experience with this approach due to his fantastic `R` package `ggplot2` (Wickham and Chang 2017).

In this post I'll show how to use `dplyr` along with `ggplot2` to wrangle spatial data and map it. We'll use the `R` package `rgbif` to download raw spatial data from the Global Biodiversity Information Facility (GBIF), which is a huge database of spatial occurence data aggregated from museums all over the world (Chamberlain et al. 2016).

Whether you work in industry or in academia, every step of a large data analysis project should be computationally reproducible. In this post I walk through all the steps needed to acquire spatial datasets, wrangle the data, and generate the maps. All the code you need to reproduce the results is shown below.

Getting the necessary R packages

First, you'll need to install three packages for this tutorial: `rgbif`, `ggplo2`, and of course `dplyr`.

```
install.packages("rgbif")
install.packages("ggplot2")
install.packages("dplyr")
```

Now load the packages:

```
library(rgbif)
library(ggplot2)
library(dplyr)
```

I used `R` version 3.4.0, `rgbif` version 0.9.8.9112, `dplyr` version 0.7.0, and `ggplot2` version 2.2.1.

Getting raw spatial data

We'll download spatial data by querying GBIF. GBIF's data is notoriously dirty because it is aggregated from thousands of sources all over the world each with their own standards and quality controls. Using `dplyr` to clean up and manipulate the data can speed up your analysis. The actual data we download will consist of the spatial coordinates at which biological organisms were observed plus lots of details about the observation such as the date, time, the name of the observer, information about the habitat and whether or not the organisms were collected or only observed, etc.

Let's search GBIF for occurences of the grasshopper genus *Sphenarium*. Why *Sphenarium*? They are used for many people's favorite snack: fried grasshoppers! yum…

If you do not love fried grasshoppers you can search for whatever creature you like. See the wikipedia entry on Chapulines for background on this tasty food (Wikipedia, 2017). First we need to query GBIF's taxonomic database to get a taxon key:

```
key = name_suggest(q="Sphenarium", rank="genus")$key[1]
```

Now we use that key to download all the occurences:

```
occ = occ_search(taxonKey=key)
```

Let's look at the raw data we got back:

```
occ
```

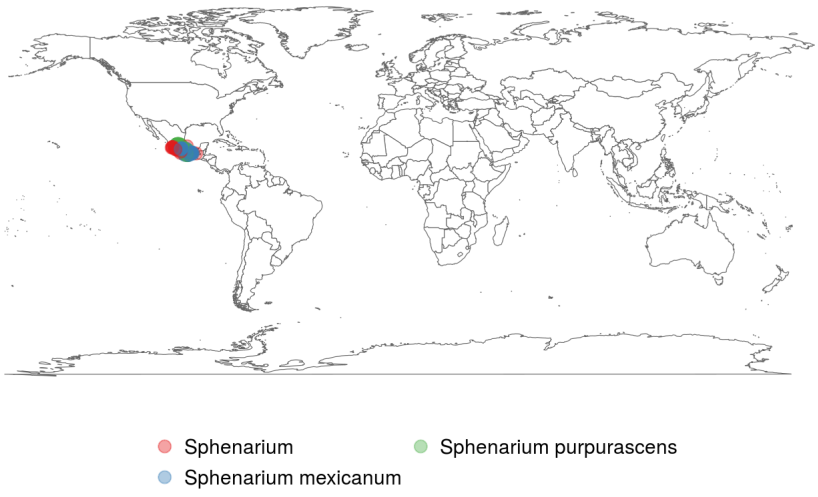
```
## Records found [356]
## Records returned [356]
## No. unique hierarchies [10]
## No. media records [69]
## No. facets [0]
## Args [limit=500, offset=0, taxonKey=1727717, fields=all]
## # A tibble: 356 x 120
##                       name        key decimalLatitude decimalLongitude
##                      <chr>      <int>           <dbl>            <dbl>
##  1 Sphenarium purpurascens 1453328623        18.93425        -99.15184
##  2 Sphenarium purpurascens 1455595024        20.68672        -99.80394
##  3 Sphenarium purpurascens 1453346573        19.30140        -99.18363
##  4 Sphenarium purpurascens 1453364158        19.29656        -99.09376
##  5 Sphenarium purpurascens 1668841895        19.27107        -98.97261
##  6 Sphenarium purpurascens 1668861509        18.99736        -98.19121
##  7 Sphenarium purpurascens 1668844826        20.90015       -100.77800
##  8 Sphenarium purpurascens 1668823231        19.06254        -98.34971
##  9 Sphenarium purpurascens 1668875777        19.27482        -98.97189
## 10 Sphenarium purpurascens 1668841958        19.27114        -98.97263
## # ... with 346 more rows, and 116 more variables: issues <chr>,
## #   datasetKey <chr>, publishingOrgKey <chr>, publishingCountry <chr>,
## #   protocol <chr>, lastCrawled <chr>, lastParsed <chr>, crawlId <int>,
## #   extensions <chr>, basisOfRecord <chr>, taxonKey <int>,
## #   kingdomKey <int>, phylumKey <int>, classKey <int>, orderKey <int>,
## #   familyKey <int>, genusKey <int>, speciesKey <int>,
## #   scientificName <chr>, kingdom <chr>, phylum <chr>, order <chr>,
## #   family <chr>, genus <chr>, species <chr>, genericName <chr>,
## #   specificEpithet <chr>, taxonRank <chr>, dateIdentified <chr>,
## #   coordinateUncertaintyInMeters <dbl>, year <int>, month <int>,
## #   day <int>, eventDate <chr>, modified <chr>, lastInterpreted <chr>,
## #   references <chr>, license <chr>, identifiers <chr>, facts <chr>,
## #   relations <chr>, geodeticDatum <chr>, class <chr>, countryCode <chr>,
## #   country <chr>, rightsHolder <chr>, identifier <chr>,
## #   verbatimEventDate <chr>, datasetName <chr>, verbatimLocality <chr>,
## #   collectionCode <chr>, gbifID <chr>, occurrenceID <chr>, taxonID <chr>,
## #   catalogNumber <chr>, recordedBy <chr>,
## #   http...unknown.org.occurrenceDetails <chr>, institutionCode <chr>,
## #   rights <chr>, eventTime <chr>, identificationID <chr>,
## #   informationWithheld <chr>, occurrenceRemarks <chr>,
## #   individualCount <int>, elevation <dbl>, stateProvince <chr>,
## #   typeStatus <chr>, locality <chr>, verbatimLabel <chr>,
## #   identificationRemarks <chr>, municipality <chr>,
## #   infraspecificEpithet <chr>, elevationAccuracy <dbl>, depth <dbl>,
## #   depthAccuracy <dbl>, scientificNameID <chr>, institutionID <chr>,
## #   dynamicProperties <chr>, county <chr>, fieldNumber <chr>,
## #   language <chr>, type <chr>, preparations <chr>, identifiedBy <chr>,
## #   acceptedNameUsage <chr>, verbatimCoordinateSystem <chr>,
## #   taxonomicStatus <chr>, locationID <chr>, ownerInstitutionCode <chr>,
## #   startDayOfYear <chr>, datasetID <chr>, bibliographicCitation <chr>,
## #   accessRights <chr>, higherClassification <chr>, collectionID <chr>,
## #   georeferenceSources <chr>, parentNameUsage <chr>,
## #   vernacularName <chr>, previousIdentifications <chr>,
## #   samplingProtocol <chr>, ...
```

The first thing you might notice is that 356 records were returned, and that most of the data is stored in a `tibble` with 120 columns. A `tibble` is an updated version of the `data.frame` data structure. They are particularly useful because they drop the nonsense of converting all character vectors to factors as `data.frames` do by default.

Map the raw data

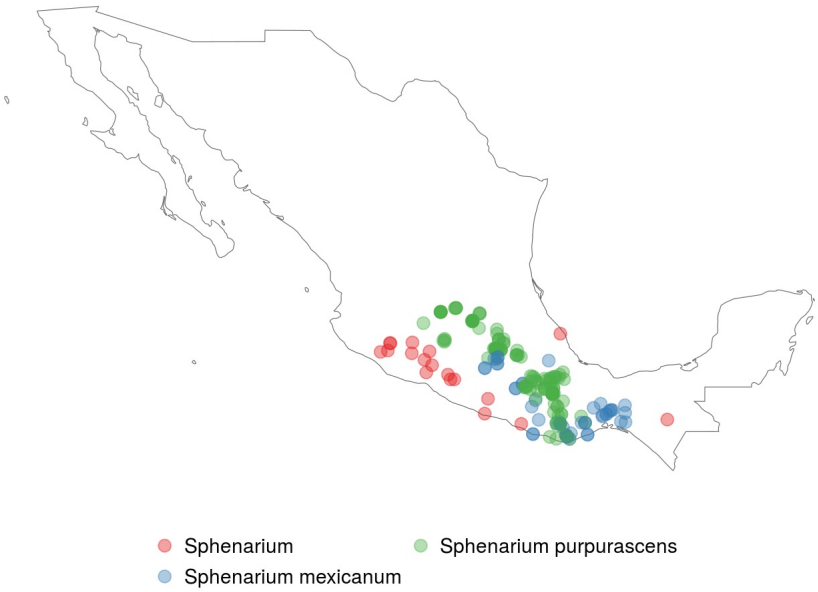
We can map the data very easily in `ggplot2` using the `gbifmap` function:

```
data = occ$data
gbifmap(data)
```



It looks like all the occurences are in Mexico, so let's zoom in there:

```
gbifmap(data, region="Mexico")
```



Nice! Now we know where to catch some delicious grasshoppers.

We see that there are two species of grasshoppers: *Sphenarium pupurascens* and *Sphenarium mexicanum*. Some of the occurences do not have species information and are only listed as *Sphenarium*.

Exploring the data with dplyr

Let's use `dplyr` to clean up the raw GBIF data and manipulate it in useful ways. First, we'll use the `select` function to get rid of all the columns of data we don't need.

```
data = select(data, name, decimalLatitude, decimalLongitude, species, year,
              month, occurrenceID)
head(data)
```

```
## # A tibble: 6 x 7
##                      name decimalLatitude decimalLongitude
##                     <chr>           <dbl>            <dbl>
## 1 Sphenarium purpurascens        18.93425        -99.15184
## 2 Sphenarium purpurascens        20.68672        -99.80394
## 3 Sphenarium purpurascens        19.30140        -99.18363
## 4 Sphenarium purpurascens        19.29656        -99.09376
## 5 Sphenarium purpurascens        19.27107        -98.97261
## 6 Sphenarium purpurascens        18.99736        -98.19121
## # ... with 4 more variables: species <chr>, year <int>, month <int>,
## #   occurrenceID <chr>
```

Let's use the 'filter' function to determine how many of the occurences are missing spatial data and so can't be mapped.

```
nrow(filter(data, is.na(decimalLatitude)))
```

```
## [1] 150
```

We see that 150 of the occurences are missing data.

Many people find the nested functions used in the line of code above to be difficult to read. We can use the pipe operator `%>%` to "pipe" the output from one function into another. The pipe operator makes for more readable code:

```
data %>% filter(is.na(decimalLatitude)) %>% nrow()
```

```
## [1] 150
```

OK, perhaps you think that fried *Sphenarium purpurascens* taste the best. We can use the `filter` function with the pipe operator to map the occurences for that species only:

```
data %>% filter("Sphenarium purpurascens" == species) %>% gbifmap(region="Mexico")
```

Let's use the `select` and `filter` functions chained together to get the range of dates for these occurences:

```
data %>% filter("Sphenarium purpurascens" == species) %>% select("year") %>%
  range(na.rm=TRUE)
```

```
## [1] 1910 2017
```

1910! Some of these observations are pretty old. You may want to know where there are currently populations of *Sphenarium purpurascens* so that you can catch fresh specimens.

Let's again use the `filter` function to map only the 2017 occurences:

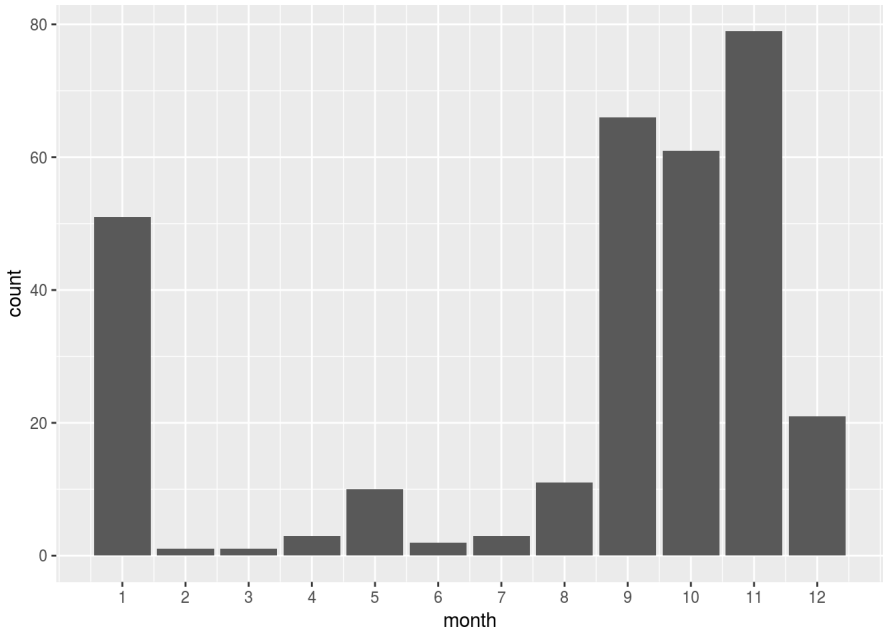
```
data %>% filter("Sphenarium purpurascens" == species & "2017" == year) %>%
  gbifmap(region="Mexico")
```

Awesome. Now we know where to go to catch fresh *Sphenarium purpurascens*.

But you may be asking what month should you go *Sphenarium* hunting? Are there particular months in the year that these grasshoppers are observed?

Again, we'll combine `dplyr` and `ggplot2` to visually identify the peak months to go *Sphenarium* hunting. We use `dplyr` functions `filter` and `select` and pipe the output directly into `ggplot`:

```
data %>% filter("Sphenarium purpurascens" == species && !is.na(month)) %>%
  select("month") %>% ggplot() + geom_bar(aes(x=month)) + scale_x_continuous(breaks=1:12)
```



Looks like the fall/winter months are the best time to go, probably the best is November.

We can use the `dplyr` function `mutate` to add a column with the name of the months.

```
months = c("January", "February", "March", "April", "May", "June", "July", "August",
           "September", "October", "November", "December")
data = data %>% mutate(named_month = months[month])
head(data$named_month)
```

```
## [1] "January"   "January"   "January"   "January"   "September" "September"
```

Finally, let's get even more specific information about populations where these grasshoppers were found. We'll randomly select the 37th occurence, and use `dplyr` to get some details:

```
links = data %>% filter("Sphenarium purpurascens" == species) %>% select("occurrenceID")
links[[37,1]]
```

```
## [1] "http://conabio.inaturalist.org/observations/4058435"
```

If you follow the link you'll see all sorts of detailed information, including the picture below of the actual grasshoppers that were observed at this location.

Those guys look pretty tasty, I think that would be a good spot to go next November!

Conclusion

We've seen examples of how to use `dyplr` to manipulate spatial data including examples of the pipe operator, the functions `select`, `filter`, and `mutate`. We've also used `ggplot2` to plot some of the manipulated data.

These examples are just the tip of the iceberg of what is possible using `dplyr` and `ggplot2`. Please see some of the tutorials in the references if you are interested in learning more, see these tutorials: dplyr tutorial 1 (BDS MIT 2017), dplyr tutorial 2 (CRAN 2017), and ggplot2 tutorial (IQSS Harvard 2017).

Take home message

This post demonstrated the steps necessary to download spatial datasets, wrangle data using `dplyr`, and generate maps from the data. These steps are useful for any data analysis project that uses spatial data and needs to be computationally reproducible.

References

Chamberlain, S., Ram, K., Barve, V., & Mcglinn, D. 2016. rgbif: Interface to the Global Biodiversity Information Facility. R package version 0.4. 0.

Comprehensive R Archive Network (CRAN). 2017. Introduction to dplyr. Retrieved from https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html

Institute For Quantitative Social Science (IQSS) at Harvard. 2017. Introduction to R graphics with ggplot2. Retrieved from http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

Biomedical Data Science (BDS) at MIT. 2017. dplyr tutorial. Retrieved from http://genomicsclass.github.io/book/pages/dplyr_tutorial.html

Wickham, H., & Francois, R. 2016. dplyr: A grammar of data manipulation [Software].

Wickham, H., Chang, W. 2017. ggplot2 official documentation. Retrieved from http://ggplot2.tidyverse.org/

Wikipedia, The Free Encyclopedia. 2017. Chapulines. Retrieved from https://en.wikipedia.org/wiki/Chapulines