Getting Data

## Introductions

* Instructors: Nathan Byers and Kali Frost
* Students: Please introduce yourself by telling us your name, where you work, and your experience with R

## Class Design

* These tutorials are made available online so that you can go back and look through them when you return to work.
* The exercises are designed to help you repeat the commands that are in the tutorials with just a few small changes.
* Some tutorials will have material that is slightly more advanced and will be left to the student to read independently. Exercises will also have more difficult questions that will be clearly labeled as "Advanced".
* After each tutorial there will be an open lab during which students are encouraged to work on the exercises. There will also be time for discussion if there are questions about the material or the exercises.
* Please feel free to speak up and help your class-mates!

Below is a table outlining the activities for each subject we will cover.

|  |  |  |
| --- | --- | --- |
| Activity | Description | Minutes |
| Tutorial | Lecture | 45 |
| Lab | Exercises or review | 30 |
| Questions | Any topic | 15 |

## Expectations

* These tutorials and exercises are meant to introduce you to general topics and specific R functions.
* We do not expect you to master all of the material in two days--there's too much to learn!
* You should expect to understand the tasks that R can accomplish and use these tutorials as resources when you go back to work.

## Topics

In this session we will cover the following topics:

* [Getting AQS data using the raqdm package](#raqdm)
* [Reading in data from a website](#scraping)
* [Pulling a table from a local database](#database)

#### Advanced Topics

* [Downloading and extracting a zipped file](#zipped)
* [Querying a database using SQL in R](#query)

## Getting AQS data with raqdm

Eric Bailey has written an R package that makes it very easy to get AQS data into R. The package is located on GitHub. It utilizes U.S. EPA's AirData API. To use EPA's AirData you must register for an account. Once you register, you will use the username and password that the EPA provides to get data into R using raqdm.

**NOTE**: The code below will not work, because you need to provide your own username and password. Also, it wouldn't be a good idea for the entire class to download large data sets all at the same time. The code below is for illustration, but we encourage you to use raqdm in your work flow.

First, we need to install the raqdm package from GitHub, using devtools.

library(devtools)  
install\_github("FluentData/raqdm")  
library(raqdm)

Once the package is installed and we load it using the library() function, we use a function called setAQDMuser() which will set the username and password for us. If you set save = TRUE then your username and password will be saved for future R sessions.

setAQDMuser(user = "me@email.com", pw = "secret", save = TRUE)

Once your user information has been saved, you can use the getAQDMdata() function to make a request for a specific data set. Right now, synchronous pulls are not working in the AirData API, so you must set synchronous = FALSE.

benz\_req <- getAQDMdata(state = "18", county = "089", bdate = "20150101",  
 edate = "20151231", param = "45201", synchronous = FALSE)

The code above does not return a data set. A request has been sent to EPA to pull that data from AQS, and an email is sent to you when the data is ready. The request is for benzene values from Lake County, Indiana from January 1 to December 31, 2015.

Once that request has been processed and you receive an email notification, (usually no more than 1 or 2 minutes) you can run the getAQDMrequest() function to get the data in R.

benz <- getAQDMrequest(benz\_req)  
head(benz)

## Latitude Longitude Datum Horizontal.Accuracy State.Code County.Code  
## 1 41.60668 -87.30473 WGS84 30.36 18 89  
## 2 41.60668 -87.30473 WGS84 30.36 18 89  
## 3 41.60668 -87.30473 WGS84 30.36 18 89  
## 4 41.60668 -87.30473 WGS84 30.36 18 89  
## 5 41.60668 -87.30473 WGS84 30.36 18 89  
## 6 41.60668 -87.30473 WGS84 30.36 18 89

For more information about parameters that can be used in getAQDMdata() see the documentation on AQDM Variables:

?`AQDM Variables`

## Reading in data from a website

In a previous training we covered how to import a .csv file into R using the RStudio point-and-click option. We also covered how to use the read.csv() function to read in a .csv file. To use read.csv() you need to supply the file path as a quoted text in the first parameter. So if you had a file called "my\_data.csv" that was located on your "C:/" drive, you would use the function like this:

my\_data <- read.csv("C:/my\_data.csv", stringsAsFactors = FALSE)

We set stringsAsFactors = FALSE to make sure that our data that should be a character data type remains a character data type.

It is possible to read a website like a .csv file on your computer. Reading data from a website is sometimes called web-scraping. A good example of a url that can be easily scraped using R is https://aqs.epa.gov/aqsweb/codes/data/ParametersByDesc.csv. You can find the page by going to the EPA site for AQS codes (http://www.epa.gov/aqs/aqs-code-list) then clicking  
 Parameters and Download Delimited Version of the Code Table.

You can actually see that the url has a .csv at the end of it. All we need to do to scrape that .csv file is to put the url as the file path in the read.csv() function. However, as you can see on the website, there is an initial line that isn't part of the table. We'll need to skip that line, so we use the skip parameter to tell the function that we want to skip the first line.

aqs\_params <- read.csv("https://aqs.epa.gov/aqsweb/codes/data/ParametersByDesc.csv",  
 stringsAsFactors = FALSE, skip = 1)  
head(aqs\_params)

## Parameter.Code Standard.Unit Parameter..Alt.Desc. CAS.Num  
## 1 43834 078 HFC-125 354-33-6  
## 2 43837 078 None 630-20-6  
## 3 43162 003 None 50-29-3  
## 4 43818 078 Ethane,1,1,2,2-tetrachloro- 79-34-5  
## 5 43821 078 None 76-13-1  
## 6 43820 078 Ethane,1,1,2-trichloro- 79-00-5  
## Parameter.Desc Conversion.Ind  
## 1 1,1,1,2,2-PENTAFLUOROETHANE R  
## 2 1,1,1,2-TETRACHLOROETHANE R  
## 3 1,1,1-TRICHLORO-2,2-BIS (P-CHLOROPHENYL) ETHANE R  
## 4 1,1,2,2-TETRACHLOROETHANE R  
## 5 1,1,2-TRICHLORO-1,2,2-TRIFLUOROETHANE R  
## 6 1,1,2-TRICHLOROETHANE R

This was a simple file to read from the web, but even data from more complicated websites can be scraped. Check out Hadley Wickham's rvest package if you would like to do more scraping: https://github.com/hadley/rvest. For advanced material on processing XML and JSON formats in web services, check out the book *XML and Web Technologies for Data Sciences with R*.

## Pulling a table from a local database

With R you can make connections to just about any kind of database. Below is a list of databases and the R packages that you can use to make connections and pull data.

|  |  |
| --- | --- |
| Database | R package |
| Microsoft Access | RODBC |
| Microsoft SQL Server | RSQLServer |
| Oracle | ROracle |
| MySQL and MariaDB | RMySQL |
| PostgreSQL | RPostgreSQL |
| SQLite | RSQLite |

We will illustrate how to pull a table out of a database by creating a simple SQLite database.

SQLite is an open source database that is very light and portable. Creating a database is very easy. We will use the Parameters table we downloaded from the AQS Codes site to make a database with just one table in it.

Here we load the RSQLite package and create the database in our working directory.

library(RSQLite)  
db <- dbConnect(drv = SQLite(), dbname = "Test.sqlite")

You make a connection to an SQLite database by providing the file path in the dbname parameter. If the database doesn't exist already then a new SQLite database will be created.

Now we will load the parameters table into the database.

dbWriteTable(conn = db, name = "Parameters", value = aqs\_params)

## [1] TRUE

Now we can check to see what tables are in the database.

dbListTables(db)

## [1] "Parameters"

And we can see what fields are in the table.

dbListFields(db, "Parameters")

## [1] "Parameter.Code" "Standard.Unit" "Parameter..Alt.Desc."  
## [4] "CAS.Num" "Parameter.Desc" "Conversion.Ind"

And finally, we can pull out the table and see that it's the same as the data frame that we put into it.

aqs\_params\_db <- dbReadTable(db, "Parameters")  
head(aqs\_params\_db)  
head(aqs\_params)

## Parameter.Code Standard.Unit Parameter..Alt.Desc. CAS.Num  
## 1 43834 078 HFC-125 354-33-6  
## 2 43837 078 None 630-20-6  
## 3 43162 003 None 50-29-3  
## 4 43818 078 Ethane,1,1,2,2-tetrachloro- 79-34-5  
## 5 43821 078 None 76-13-1  
## 6 43820 078 Ethane,1,1,2-trichloro- 79-00-5  
## Parameter.Desc Conversion.Ind  
## 1 1,1,1,2,2-PENTAFLUOROETHANE R  
## 2 1,1,1,2-TETRACHLOROETHANE R  
## 3 1,1,1-TRICHLORO-2,2-BIS (P-CHLOROPHENYL) ETHANE R  
## 4 1,1,2,2-TETRACHLOROETHANE R  
## 5 1,1,2-TRICHLORO-1,2,2-TRIFLUOROETHANE R  
## 6 1,1,2-TRICHLOROETHANE R

## Parameter.Code Standard.Unit Parameter..Alt.Desc. CAS.Num  
## 1 43834 078 HFC-125 354-33-6  
## 2 43837 078 None 630-20-6  
## 3 43162 003 None 50-29-3  
## 4 43818 078 Ethane,1,1,2,2-tetrachloro- 79-34-5  
## 5 43821 078 None 76-13-1  
## 6 43820 078 Ethane,1,1,2-trichloro- 79-00-5  
## Parameter.Desc Conversion.Ind  
## 1 1,1,1,2,2-PENTAFLUOROETHANE R  
## 2 1,1,1,2-TETRACHLOROETHANE R  
## 3 1,1,1-TRICHLORO-2,2-BIS (P-CHLOROPHENYL) ETHANE R  
## 4 1,1,2,2-TETRACHLOROETHANE R  
## 5 1,1,2-TRICHLORO-1,2,2-TRIFLUOROETHANE R  
## 6 1,1,2-TRICHLOROETHANE R

Now we'll disconnect from the database and delete it from the working directory.

dbDisconnect(db)

## [1] TRUE

unlink("Test.sqlite")

## Exercises

Exercises for this tutorial can be found here: http://rpubs.com/NateByers/GettingDataExercises.

## Advanced Topics

### Downloading and extracting a zipped file

Sometimes you may want to automate the download of a large file that's zipped and read it into R. If we look at the National Emissions Inventory page for 2011 we can see that there are ZIP CSV files under the heading "SCC Data Files". Below is code that will download the Onroad zipped file to a temporary file, read the region 5 .csv file into R, and delete the temporary file on your machine.

# create a temporary file  
temp <- tempfile()  
  
# download the .zip file to the temporary file  
download.file("ftp://ftp.epa.gov/EmisInventory/2011/2011neiv2\_onroad\_byregions.zip", temp)  
  
# read the "onroad\_5.csv" file into R--this will take a while  
data <- read.csv(unz(temp, "onroad\_5.csv"))  
  
# delete the temporary file  
unlink(temp)  
  
head(data)

## state\_and\_county\_fips\_code  
## 1 88180  
## 2 88180  
## 3 88180  
## 4 88180  
## 5 88181  
## 6 88181  
## tribal\_name st\_usps\_cd  
## 1 Shoshone-Bannock Tribes of the Fort Hall Reservation of Idaho   
## 2 Shoshone-Bannock Tribes of the Fort Hall Reservation of Idaho   
## 3 Shoshone-Bannock Tribes of the Fort Hall Reservation of Idaho   
## 4 Shoshone-Bannock Tribes of the Fort Hall Reservation of Idaho   
## 5 Coeur d'Alene Tribe of the Coeur d'Alene Reservation, Idaho   
## 6 Coeur d'Alene Tribe of the Coeur d'Alene Reservation, Idaho   
## county\_name data\_category\_cd scc  
## 1 NP 2103008000  
## 2 NP 2302002100  
## 3 NP 2102008000  
## 4 NP 2302002200  
## 5 NP 2102008000  
## 6 NP 2302002200

### Querying a database using SQL in R

Not only can you make a connection to a database with R, but you can pass SQL statements to the database and retrieve the result inside of R. Here's an example of querying an SQLite database in R.

library(RSQLite)  
  
# Create an SQLite database and save connection in db\_cities object  
db\_cities <- dbConnect(drv = SQLite(), dbname = "QueryTest.sqlite")  
  
# read in the AQS codes table for city names  
aqs\_cities <- read.csv("https://aqs.epa.gov/aqsweb/codes/data/CityNamesByState.csv",  
 stringsAsFactors = FALSE, skip = 1)  
  
# replace . with \_ in table names--otherwise you will not be able to use field names  
names(aqs\_cities) <- sub("\\.", "\_", names(aqs\_cities))  
  
# make a Cities table in the database  
dbWriteTable(db\_cities, "Cities", aqs\_cities)  
  
# send a query to the database for cities in Illinois  
query <- dbSendQuery(db\_cities, "SELECT \* FROM Cities WHERE State\_Name = 'Illinois'")  
  
# get a data frame of cities in Illinois  
illinois\_cities <- dbFetch(query)  
  
head(illinois\_cities)

## State\_Name State\_Code City\_Code City\_Name County\_Code County\_Name  
## 1 Illinois 17 113 Abingdon 95 Knox  
## 2 Illinois 17 230 Addieville 189 Washington  
## 3 Illinois 17 243 Addison 43 DuPage  
## 4 Illinois 17 295 Adeline 141 Ogle  
## 5 Illinois 17 516 Albany 195 Whiteside  
## 6 Illinois 17 555 Albers 27 Clinton

It's always good practice to clear the query result and disconnect from the database.

dbClearResult(query)  
dbDisconnect(db\_cities)

Then we delete the database.

unlink("QueryTest.sqlite")