FEMA Fact Sheet

OpenFEMA Claims and Policy Data Interaction Guide July 2019

Introduction

The [FIMA/NFIP] claims and policy files available on OpenFEMA are too large for most spreadsheet applications like Microsoft Excel. This guide is a technical document which will provide an overview for compiling the data and extracting subsets for further analysis. Users do not have to be well-versed in the data or computer science to use this guide, but a basic familiarity with data analytics tools will help. For this guide we will be using R and Apache Spark, two free opensource tools, however there are many other excellent options for conducting this type of analysis. Our use of these tools in this guide does not constitute endorsement of the products or an assurance of their efficacy or security.

Tools

R is statistical programming language which natively includes a suite of statistical and modeling functions as well as flexible data structures. R is often compared to SAS, SPSS and Stata in terms of functionality. Apache Spark is an opensource tool for distributed processing across large datasets. In this guide we will initiate and interface with Spark via the R library SparkR. This guide will focus on installing and using these tools on a windows desktop, however, the process should be similar in other environments.

Installing the Tools

Step 1: Download the OpenFEMA datasets

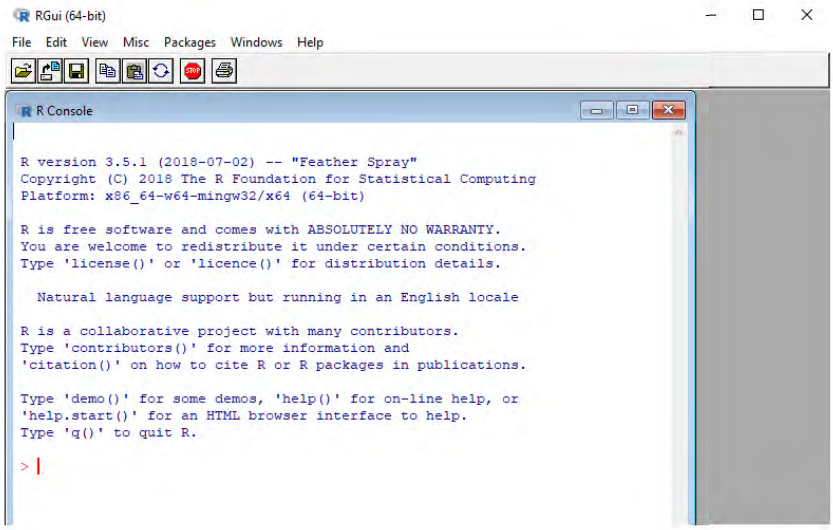
The claim and policy datasets posted to OpenFEMA are compressed into multiple zip files. Download all files and decompress to a convenient directory.

Step 2: Download and install R

The R project is located here: https://www.r-project.org/. The website includes links and directions for downloads. Identify the mirror and installation method you would like to use. Follow directions provided by the r-project to install R on your computer.

Step 3: Open R

Once R is installed open the RGui application. The interface should appear similar to the one below if you are working in a windows environment



Step 4: Install the SparkR package

Confirm you are connected to the internet. In the R terminal you've opened (the prompt after '>') issue the following command to install the SparkR package.

```
install.packages ("SparkR")
```

Similar to the installation of R you may be prompted to select the mirror you would like to download from; choose whichever is appropriate for you. You can find additional information about the installation function and how to install from local files here however for this guide we recommend accepting all defaults.

Step 5: Initiate an Apache Spark session via R.

Load the spark library and Initiate a new spark session in the R terminal by entering the following commands.

```
library(SparkR)
sparkR.session()
```

There are additional arguments you can provide however in this guide we will assume you don't have concurrent spark sessions running. When you first initiate a session it may be necessary to download and install additional files.

Load NFIP data into spark Data frames

With the R terminal confirm the spark library is loaded and initiate or connect to an existing spark session

```
library(SparkR)
sparkR.session()
```

Identify where you have decompressed the NFIP files you would like to load (it is easiest if you have all the data extracted to the same directory). Run the following command to establish the path to the directory where the data is stored.

```
nfipdir <- choose.dir()
```

Once you have identified the directory, load each data file into a sparkdata frame (note: the names of the files may have changed since they were originally posted). The claims file and the first segment of the policy file include headers and should be loaded into two tables titled clm and pol1 respectively:

```
clm <- read.df(paste(nfipdir, "openfema_claims20190331.csv", sep ="\\"), "csv", header = 'true')
pol1 <- read.df(paste(nfipdir, "openfema_policies20190331_00.csv", sep ="\\"), "csv", header = 'true')
```

The remaining policy files should be loaded without headers.

```
pol2 <- read.df(paste(nfipdir, "openfema_policies20190331_01.csv", sep ="\\"), "csv", header = 'false')
pol3 <- read.df(paste(nfipdir, "openfema_policies20190331_02.csv", sep ="\\"), "csv", header = 'false')
pol4 <- read.df(paste(nfipdir, "openfema_policies20190331_03.csv", sep ="\\"), "csv", header = 'false')
pol5 <- read.df(paste(nfipdir, "openfema_policies20190331_04.csv", sep ="\\"), "csv", header = 'false')
pol6 <- read.df(paste(nfipdir, "openfema_policies20190331_05.csv", sep ="\\"), "csv", header = 'false')
pol7 <- read.df(paste(nfipdir, "openfema_policies20190331_06.csv", sep ="\\"), "csv", header = 'false')
pol8 <- read.df(paste(nfipdir, "openfema_ policies20190331_07.csv", sep ="\\"), "csv", header = 'false')
pol9 <- read.df(paste(nfipdir, "openfema_policies20190331_08.csv", sep ="\\"), "csv", header = 'false')
pol10 <- read.df(paste(nfipdir, "openfema_policies20190331_09.csv", sep ="\\"), "csv", header = 'false')
```

Step 6.: Build a unified policy table

First load the data without headers into a single table which we'll call policies

```
policies <- rbind(pol2, pol3, pol4, pol5, pol6, pol7, pol8, pol09, pol10)
```

Assign policies the headers for the first policy file

```
names(policies) <- names(pol1)
```

Add the first policy file to the policy table

```
policies <- rbind(pol1, policies)
```

Subsetting the data

Now that the data is loaded we'll focus on extracting smaller subsets. We will be using the table names established in the guide and we will describe writing the results to a csv file. The assumption is that users will open the subsets as spreadsheets we therefore recommend that you limit the number of records to fewer than 1,000,000 in any subset.

Extracting claims for given state

Establish the state id, the directory where you would like to export the result as well as the resulting file name. Your file name should end in '.csv'.

```
lkupstate <- readline("Enter a two character state abbreviation and press ENTER:  ")
filestate <- readline("Enter a file name and press ENTER:  ")
savedir <- choose.dir()
filestate <- paste(savedir,filestate, sep= "\\")
```

Filter the data by state

```
clmstate <- filter(clm, clm$state == lkupstate)
```

Return the results as an R data frame (you may be limited by the amount of memory on your computer)

```
rclmstate <- collect(clmstate)
```

Write the results to your computer

```
write.csv(rclmstate, filestate, na = "", row.names = FALSE)
```

Extracting policies contracts in force for a given state at a given moment in time.

Establish the state id and date as well as the directory where you would like to export the result and the resulting file name.

```
lkupstate <- readline("Enter a two character state abbreviation and press ENTER:  ")
lkupdate <- readline("Enter the date (in the following format: YYYY-MM-DD) at which
you would like to view policy contracts in force and press ENTER:  ")
filestate <- readline("Enter a file name and press ENTER:  ")
savedir <- choose.dir()
filestate <- paste(savedir,filestate, sep= "\\")
```

Set date fields to date format and filter the data by state and date in force.

```
to_date(policies$policyeffectivedate)
to_date(policies$policyterminationdate)
polstate <- filter(policies, policies$propertystate == lkupstate &
policies$policyeffectivedate <= lkupdate & policies$policyterminationdate >
lkupdate)
```

Return the results as an R data frame (you may be limited by the amount of memory on your computer)

```
rpolstate <- collect(polstate)
```

Write the results to your computer

```
write.csv(rpolstate, filestate, na = "", row.names = FALSE,)
```