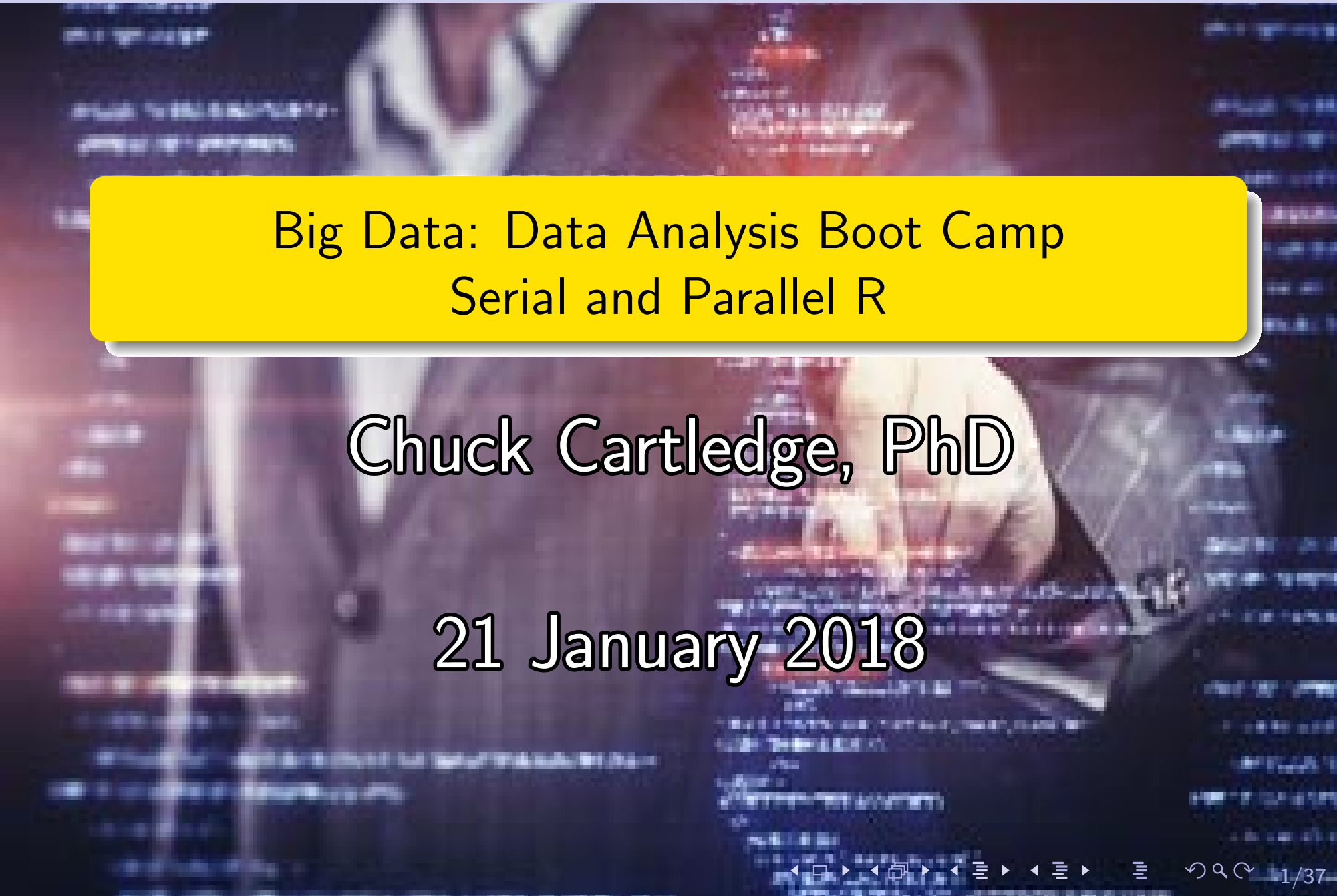


Big Data: Data Analysis Boot Camp
Serial and Parallel R

Chuck Cartledge, PhD

21 January 2018

Table of contents (1 of 1)

What are we going to cover?

1. Compare R to other languages
2. Identify R's memory limitations
3. Explore some of R's packages to overcome R's RAM limitations
4. Explore Medicare Part D data (a dataframe with over 25,000,000 rows)

R is relatively slow compared to other languages

- R is an interpreted language. Interpreted language execution is almost always slower than compiled language execution.
- R is not well defined. It is open source, and anyone can add extensions (known as libraries) without a formal review and development cycle. Collisions between library functions and inefficient pieces of software are common.
- R is single threaded. Regardless of how many cores, or threads the underlying CPU may have, only one will be used to run the R interpreter.

R is good for "proof of concept" and demonstration projects. It may not be fast enough for enterprise level tasks.

All data must fit into available RAM (1 of 2)

- **biglars** – least-angle regression, lasso and stepwise regression for numeric datasets in which the number of observations is greater than the number of predictors. The functions can be used with the *ff* library to accommodate datasets that are too large to be held in memory[6].
- **biglm** – regression for data too large to fit in memory[4].
- **ff** – package provides data structures that are stored on disk but behave (almost) as if they were in RAM by transparently mapping only a section (pagesize) into main memory - the effective virtual memory consumption per *ff* object[1].
- **ffbase** – extends the out of memory vectors of *ff* with statistical functions and other utilities to ease their usage[2].

All data must fit into available RAM (2 of 2)

- **stream** – a framework for data stream modeling and associated data mining tasks such as clustering and classification[3].

We'll be looking at **ff** and **ffbase**.

Modern OSs use virtual memory

"...Another important concept inherent in the hierarchy is virtual memory. The purpose of virtual memory is to use the hard disk as an extension of RAM, thus increasing the available address space a process can use."

Null and Lobur [5]

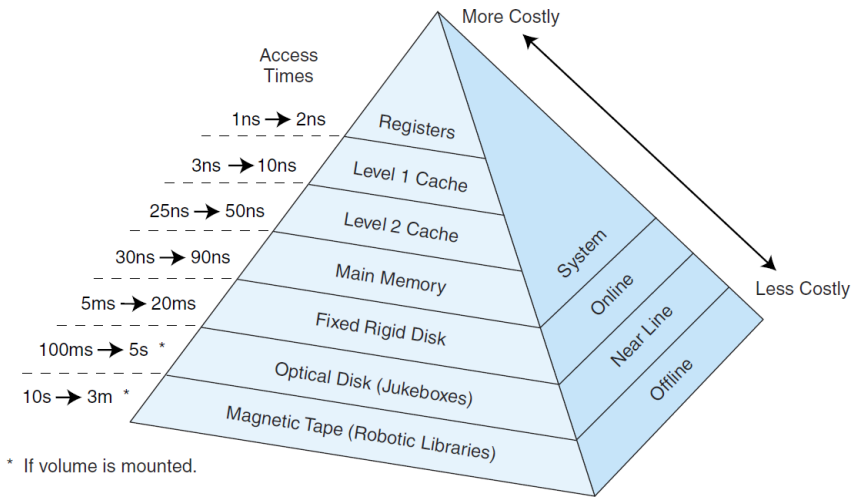


Image from [5].

Same image.

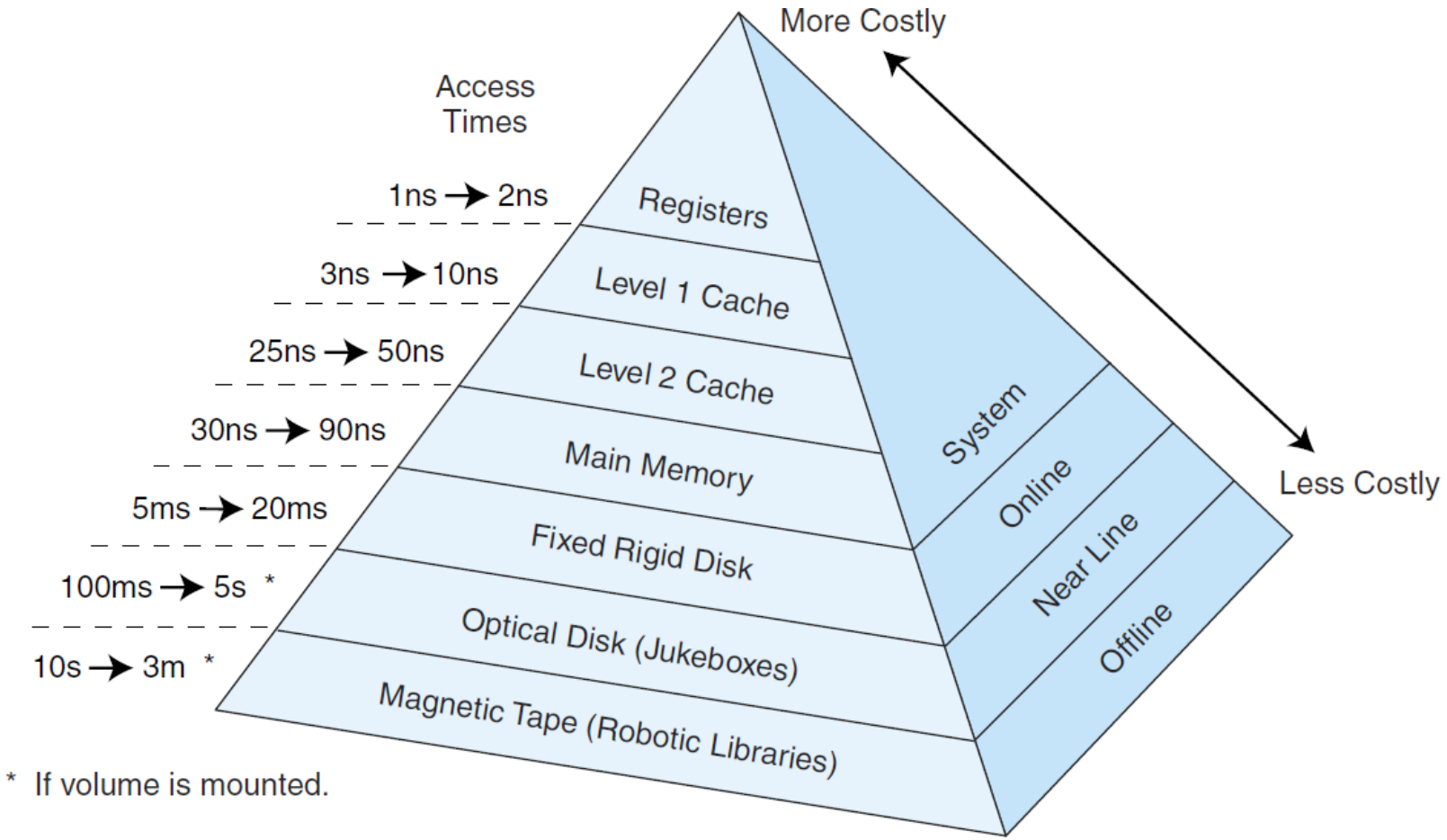


Image from [5].

R implements virtual memory via the *ff* package

How to read a 1TB CSV file:

```
library(ffbase)
options(fftempdir = saveDir)
filechunks.ff <-
read.csv.ffdf(file=‘‘big_data.csv’’,
header=T, sep=‘‘,’’, VERBOSE=T,
next.rows=500000, colClasses=NA)
```

Data is written to a series of files in `savedir`.

Use

```
ls(‘‘package:ffbase’’)
```

to see package offerings.

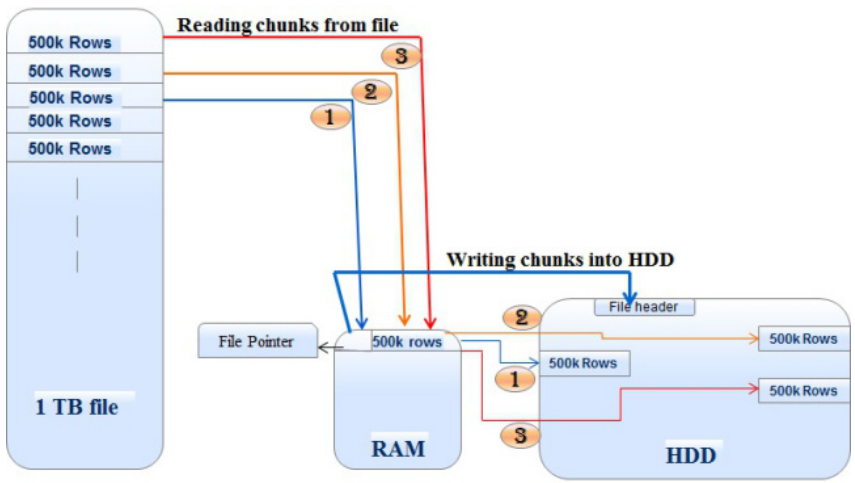


Image from [7].

Same image.

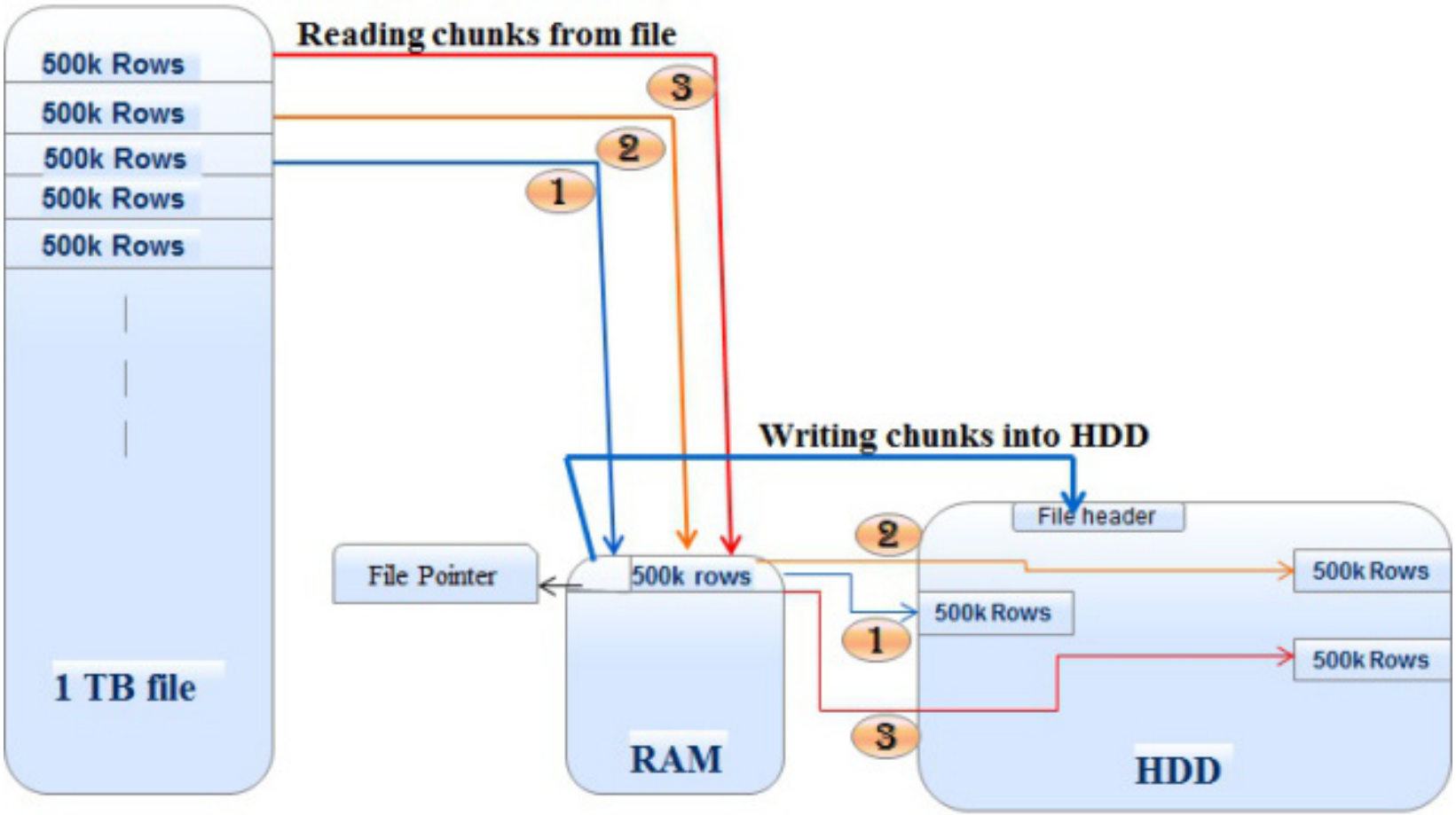


Image from [7].

Preparations to process data

We'll be using the `chapter-03-unleashing-R.R` script.
Creating ff's working directory:

```
1 tempDir <- tempdir()
2 ffDir <- file.path( tempDir, "ffdf")
3 dumpObject(ffDir)
4 dir.create(ffDir, showWarnings = TRUE, recursive = TRUE
      , mode = "0777")
5 system(command)
6 options(fftempdir = ffDir)
```

Loading the data

Loading data the first time:

```
dumpObject(system.time(flights.ff <- loadFFdata(1,
    useSaved=TRUE, saveDir=ffDir, zipFile=zipFile,
    dataFile=``B05396_Ch03_Code/flights_sep_oct15.txt''
    )))
```

There are 951,111 records.
The initial load takes:

user	system	elapsed
30.224	0.476	30.776

Subsequent loads take:

user	system	elapsed
1.132	0.004	1.175

Exploring the data.

Column names:

```
[1] "   -- Dumping the object: names(flights.ff) (of type: character, class: character)"
 [1] "YEAR"               "MONTH"              "DAY_OF_MONTH"
 [4] "DAY_OF_WEEK"        "FL_DATE"            "UNIQUE_CARRIER"
 [7] "AIRLINE_ID"         "TAIL_NUM"           "FL_NUM"
[10] "ORIGIN_AIRPORT_ID" "ORIGIN"             "ORIGIN_CITY_NAME"
[13] "ORIGIN_STATE_NM"    "ORIGIN_WAC"         "DEST_AIRPORT_ID"
[16] "DEST"               "DEST_CITY_NAME"     "DEST_STATE_NM"
[19] "DEST_WAC"           "DEP_TIME"           "DEP_DELAY"
[22] "ARR_TIME"           "ARR_DELAY"          "CANCELLED"
[25] "CANCELLATION_CODE" "DIVERTED"           "AIR_TIME"
[28] "DISTANCE"
```

Bringing in airline data, and merging data.

```
dumpObject(system.time(airlines.ff <- loadFFdata(2,
    useSaved=TRUE, saveDir=ffDir, zipFile=zipFile,
    dataFile="B05396_Ch03_Code/airline_id.csv")))
names(airlines.ff) <- c("AIRLINE_ID", "AIRLINE_NM")
flights.data.ff <- merge.ffdf(flights.ff, airlines.ff,
    by="AIRLINE_ID")
dumpObject(names(flights.data.ff))
```

The variables `flights.ff` and `airlines.ff` are now gone.

How many flights per state?

```
    dumpObject(system.time(orig_state_tab <-
                table.ff(flights.data.ff$ORIGIN_STATE_NM,
                exclude = NA)))
    dumpObject(orig_state_tab)
...
   user  system elapsed
  1.380   0.004   1.392
[1] "   -- Dumping the object: orig_state_tab (of type:
           integer, class: table)"
                                        Alabama
                                           4744
                                         Alaska
                                           5697
                                        Arizona
                                          28060
(listing truncated)
```

Simple statistics to show what can be done.(1 of 2)

```
dumpObject(mean(flights.data.ff$DISTANCE))
dumpObject(quantile(flights.data.ff$DISTANCE))
dumpObject(range(flights.data.ff$DISTANCE))
dumpObject(Hmisc::describe(as.data.frame.ffdf(flights.
    data.ff$DISTANCE)))
dumpObject(summary(as.data.frame.ffdf(flights.data.ff$
    DISTANCE)))
```

Simple statistics to show what can be done.(2 of 2)

Resulting in:

```
[1] "   -- Dumping the object: mean(flights.data.ff$DISTANCE) (of type: double, class: numeric)"
[1] 816.1528
[1] "   -- Dumping the object: quantile(flights.data.ff$DISTANCE) (of type: integer, class: integer)"
  0%  25%  50%  75% 100%
  31  370  641 1050 4983
[1] "   -- Dumping the object: range(flights.data.ff$DISTANCE) (of type: integer, class: integer)"
[1]   31 4983
[1] "   -- Dumping the object: Hmisc::describe(as.data.frame.ffdf(flights.data.ff$DISTANCE)) (of type: lis
as.data.frame.ffdf(flights.data.ff$DISTANCE)
       n  missing distinct     Info     Mean      Gmd      .05      .10
  951111        0     1241        1    816.2    637.6      168      224
     .25      .50      .75      .90      .95
     370      641     1050     1721     2239
```

What is the average delay per departure city?(1 of 2)

```
1 DepDelayByOrigCity <- ffdfdply (flights.data.ff, split
    = flights.data.ff$ORIGIN_CITY_NAME, FUN=function(x)
    {summaryBy(DEP_DELAY~ORIGIN_CITY_NAME, data=x, FUN
    =mean, na.rm=TRUE)})
2 dumpObject(DepDelayByOrigCity)
3 plot1.df <- as.data.frame.ffdf(DepDelayByOrigCity)
4 dumpObject(str(plot1.df))
5 plot1.df <- orderBy(~-DEP_DELAY.mean, data=plot1.df)
6 dumpObject(plot1.df)
```

What is the average delay per departure city?(1 of 2)

Resulting in:

```
                     ORIGIN_CITY_NAME DEP_DELAY.mean
198                     Pago Pago, TT    49.11764706
286                   Adak Island, AK    21.23529412
289                 Christiansted, VI    20.46875000
268           North Bend/Coos Bay, OR    15.68055556
271                   Plattsburgh, NY    15.63333333
302                     Nantucket, MA    13.00952381
209          Scranton/Wilkes-Barre, PA    12.38565022
(listing truncated)
```

How many flights were canceled? (1 of 2)

```
1 subs1.ff <- subset.ffdf(flights.data.ff, CANCELLED ==
      1, select = c(FL_DATE, AIRLINE_ID, ORIGIN_CITY_NAME
      , ORIGIN_STATE_NM, DEST_CITY_NAME, DEST_STATE_NM,
      CANCELLATION_CODE))
2 dumpObject(subs1.ff)
```

How many flights were canceled? (2 of 2)

Resulting in:

	FL_DATE	AIRLINE_ID	ORIGIN_CITY_NAME
1	2015-09-29	19790	Melbourne, FL
2	2015-09-04	20366	Dallas/Fort Worth, TX
3	2015-09-04	20366	Lake Charles, LA
4	2015-09-03	20366	Dallas/Fort Worth, TX
5	2015-09-12	20366	Beaumont/Port Arthur, TX
6	2015-09-03	20366	Columbia, SC
7	2015-09-07	20366	Dallas/Fort Worth, TX
8	2015-09-09	20366	Columbia, SC
...			
4529	2015-10-19	20304	Houston, TX

What is Medicare Part D, and why could it be interesting?

> *"Medicare Part D, also called the Medicare prescription drug benefit, is a United States federal-government program to subsidize the costs of prescription drugs and prescription drug insurance premiums for Medicare beneficiaries."*
>
> *W. Staff [8]*

It is interesting because of the size of the program and the number of people affected by it.

Where can we get data about the program?

Information is available the Centers for Medicare and Medicaid Services (www.cms.gov)
Data specific to the Part D program is at:

`https://www.cms.gov/Research-Statistics-Data-and-Systems/Statistics-Trends-and-Reports/Medicare-Provider-Charge-Data/Part-D-Prescriber.html`

We will be looking at CY 2015 data (about 551 MB compressed).

Now into the analysis:

We'll be using the `chapter-03-unleashing-R-medicare.R` script.
First to download the data:

```
downloadURL <- "http: ... long URL ..."
zipFile <- "PartD_Prescriber_PUF_NPI_DRUG_15.zip"
medicare.ff <- loadFFdata(3, useSaved=TRUE, saveDir=
    ffDir, zipFile=zipFile, dataFile="PartD_Prescriber_
    PUF_NPI_Drug_15.txt", downloadURL=downloadURL)
```

The first time the program is run, the download takes a few minutes. It is much, much faster after that.
The program downloads 578,938,303 bytes of compressed data.

We'll take a peek at the data:

```
1 dumpObject(names(medicare.ff))
```

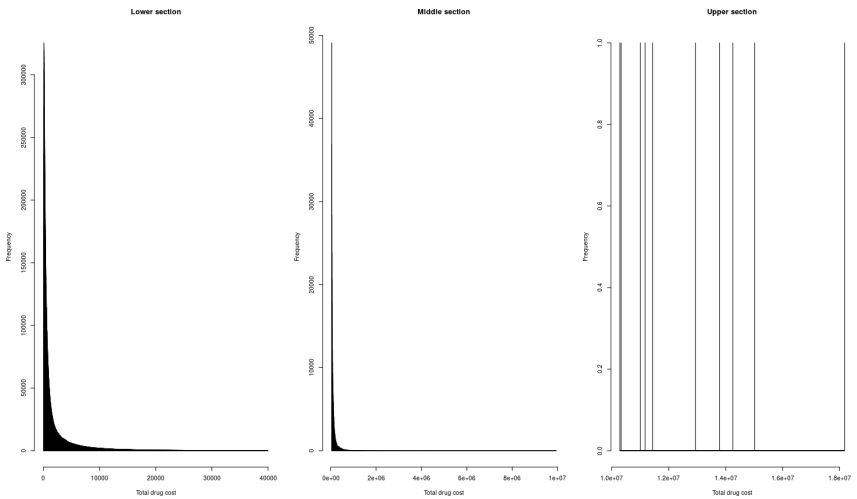
Resulting in:

```
[1] "   -- Dumping the object: names(medicare.ff) (of type: character, class: character)"
 [1] "npi"                          "nppes_provider_last_org_name"
 [3] "nppes_provider_first_name"    "nppes_provider_city"
 [5] "nppes_provider_state"         "specialty_description"
 [7] "description_flag"             "drug_name"
 [9] "generic_name"                 "bene_count"
[11] "total_claim_count"            "total_30_day_fill_count"
[13] "total_day_supply"             "total_drug_cost"
[15] "bene_count_ge65"              "bene_count_ge65_suppress_flag"
[17] "total_claim_count_ge65"       "ge65_suppress_flag"
[19] "total_30_day_fill_count_ge65" "total_day_supply_ge65"
[21] "total_drug_cost_ge65"
```

There are 24,525,869 records.

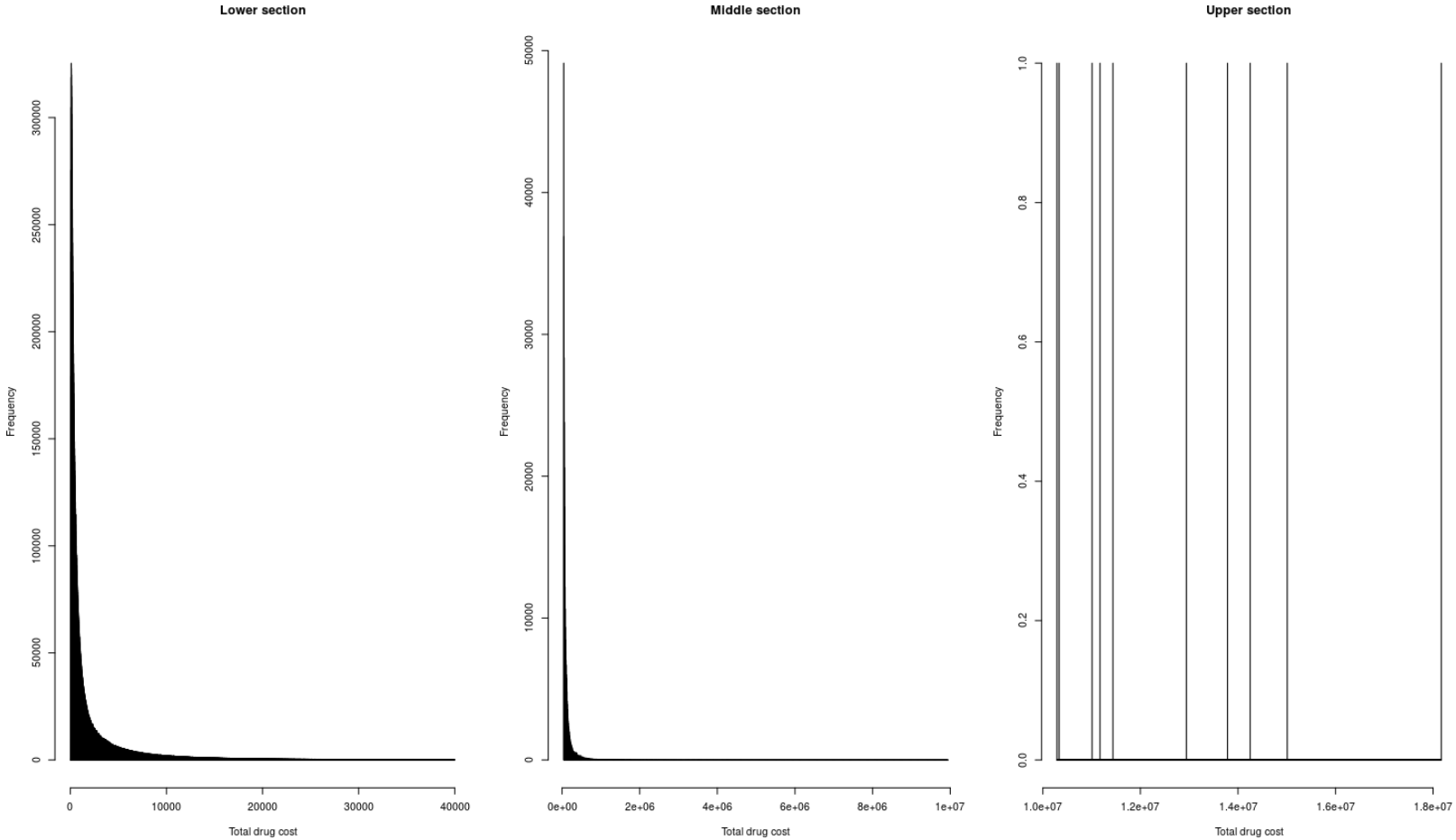
We're interested in the field `total_drug_cost`

The range of values is too large to be plotted in a meaningful way on a single histogram, so three histograms were created. The range of values are from 0 to <40,000, from 40,000 to <10,000,000, and from 10,000,000 to the maximum.



Attached file.

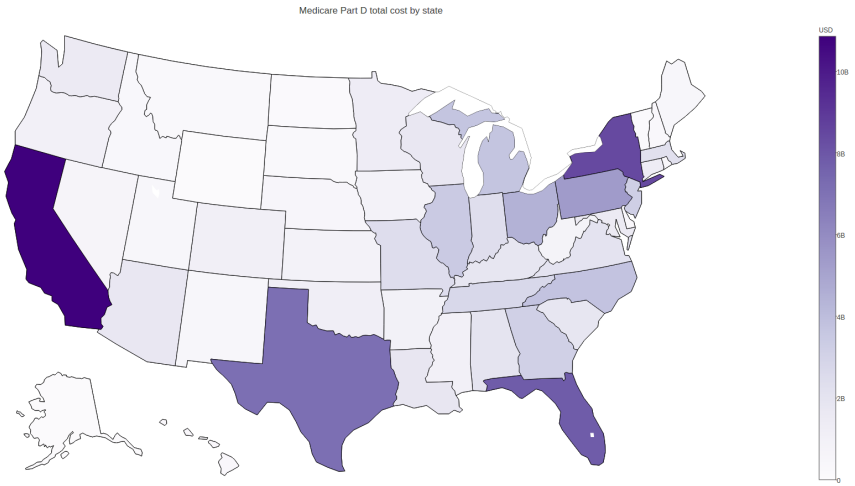
Same image.



Attached file.

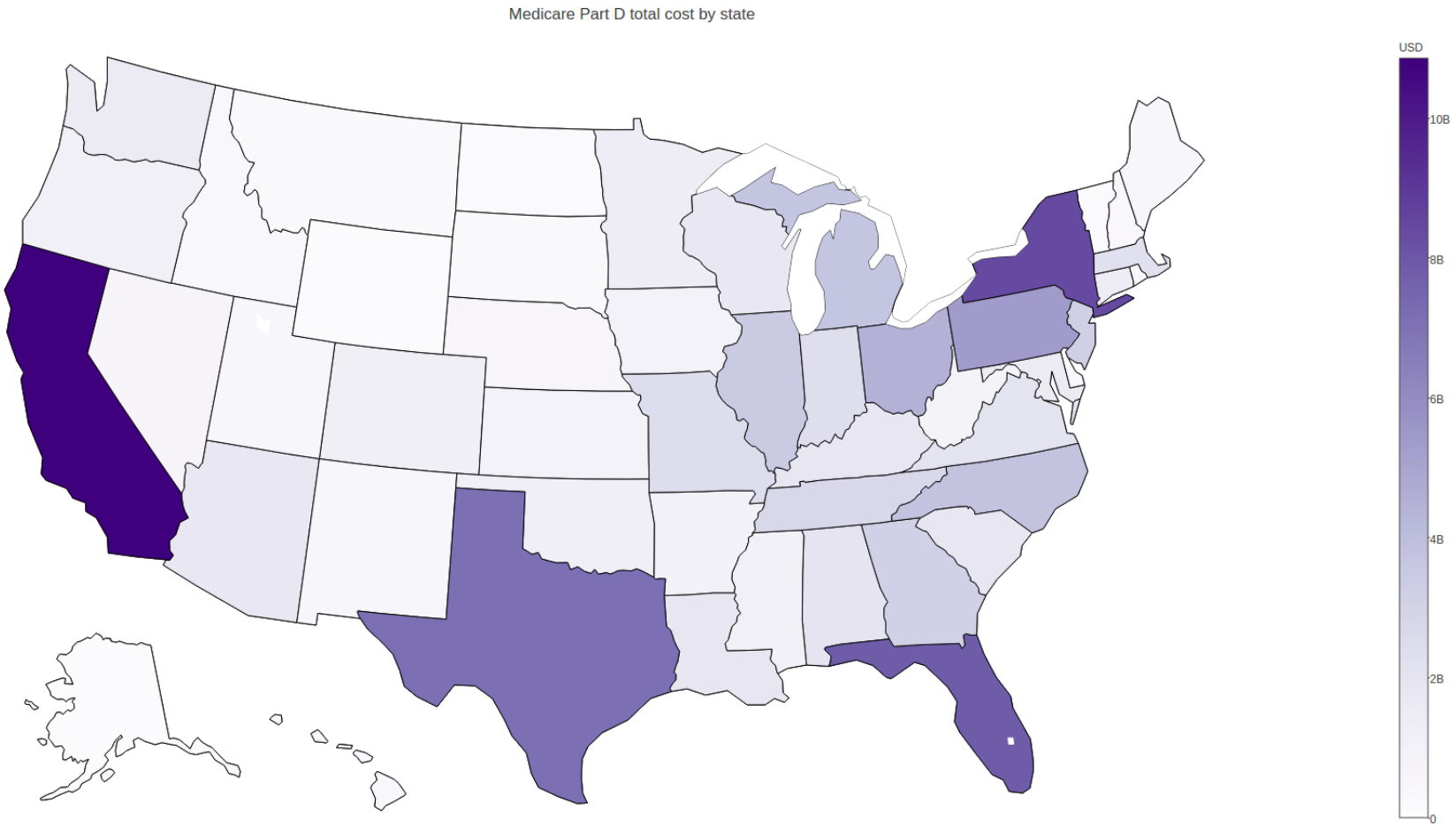
Total drug cost by state.

We are interested in the total payments to each state, and to present that in a graphical format.
The image is presented in a web page with "hover" explanations.



Attached file.

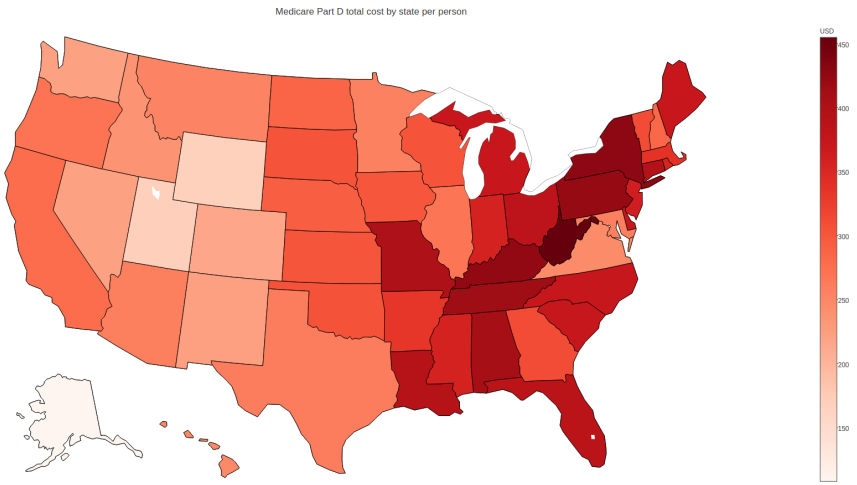
Same image.



Attached file.

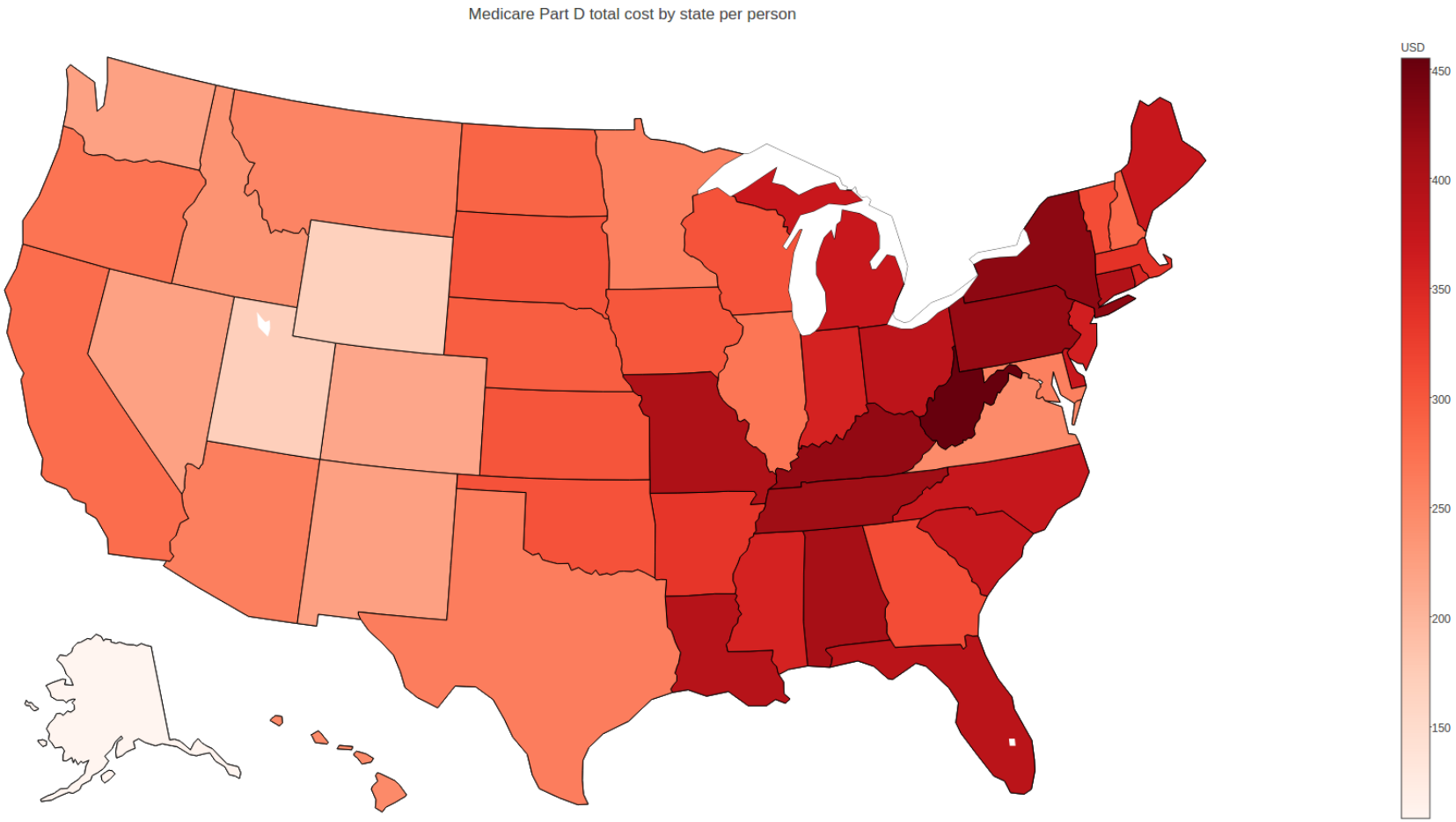
Total drug cost by state by person.

The data by state, tends to show that the areas with the greatest population gets the most money. But how do the states vary as a function of their population?



Attached file.

Same image.



Attached file.

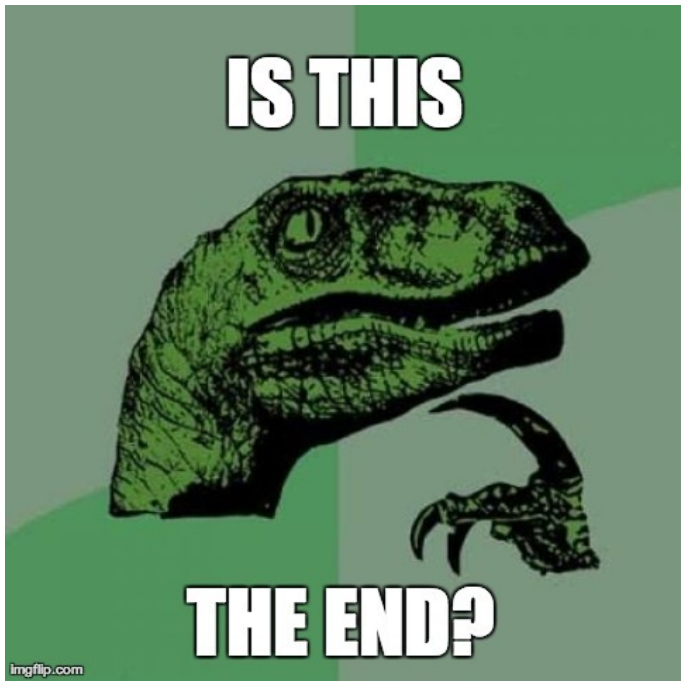
What other questions can be asked and answered with the data?

Q & A time.

Q: Why is it that the more accuracy you demand from an interpolation function, the more expensive it becomes to compute?
A: That's the Law of Spline Demand.

What have we covered?

- We've looked at how the `ff` package can extend R's memory.
- We've processed dataframes with more that 25,000,000 records.
- We've seen different ways to present voluminous data.



Next: BDAR Chapter 4, Hadoop and R

References (1 of 2)

[1] D Adler, C Gläser, O Nenadic, J Oehlschlägel, and W Zucchini, ff: Memory-efficient Storage of Large Data on Disk and Fast Access F R package version (2014), 2–2.

[2] Edwin De Jonge, Jan Wijffels, and Jan van der Laan, ffbase: Basic Statistical Functions for Package ff, 2014.

[3] Michael Hahsler, Matthew Bolanos, and John Forrest, Introduction to stream: An Extensible Framework for Data Stream Cl Journal of Statistical Software **76** (2017), no. 14, 1–50.

[4] Thomas Lumley, biglm: Bounded Memory Linear and Generalized Linear Models, R package version 0.8 (2011).

References (2 of 2)

[5] Linda Null and Julia Lobur,
The Essentials of Computer Organization and Architecture,
Jones & Bartlett Publishers, 2010.

[6] M Seligman, C Fraley, and T Hesterberg,
biglars: Scalable Least-Angle Regression and Lasso, R package version **1** (2010), no. 1.

[7] RHandbook Staff, Big Data Analysis using ff and ffbase,
`https://rhandbook.wordpress.com/2013/11/12/ff-ffbase/`,
2013.

[8] Wikipedia Staff, Medicare Part D,
`https://en.wikipedia.org/wiki/Medicare_Part_D`, 2017.

Files of interest

1. BDAR chapter 3 code
2. Looking at Medicare Part D data
3. R library script file
4. BDAR source scripts and zip file
5. Excel census population spreadsheet