The Bigmemory Suite of Packages

SCALABLE DATA PROCESSING IN R

Michael Kane
Assistant Professor, Yale University

So far ..

- Import
- Subset
- Assign values to `big.matrix` objects

Associated Packages

Tables and summaries

- `biganalytics`
- `bigtabulate`

Associated Packages

Linear algebra

- `bigalgebra`

Associated Packages

Fit Models

- `bigpca`
- `bigFastLM`
- `biglasso`
- `bigrf`

The FHFA's Mortgage Data Set

- Mortgages that were held or securitized by both Federal National Mortgage Association (Fannie Mae) and Federal Home Loan Mortgage Corporation (Freddie Mac) from 2009-2015
- FHFA Mortgage data is available online **here**
- We will focus on a random subset of 70000 loans

1st example: using bigtabulate with bigmemory

```
library(bigtabulate)
# How many samples do we have per year?
bigtable(mort, "year")
```

```
 2008  2009  2010  2011  2012  2013  2014  2015
 8468 11101  8836  7996 10935 10216  5714  6734
```

```
# Create nested tables
bigtable(mort, c("msa", "year"))
```

```
  2008 2009 2010 2011 2012 2013 2014 2015
0 1064 1343  998  851 1066 1005  504  564
1 7404 9758 7838 7145 9869 9211 5210 6170
```

Let's practice!

SCALABLE DATA PROCESSING IN R

Split-Apply-Combine

SCALABLE DATA PROCESSING IN R

Michael Kane
Assistant Professor, Yale University

Split-Apply-Combine

- Split: `split()`
- Apply: `Map()`
- Combine: `Reduce()`

Partition using split()

The `split()` function partitions data

- First argument is a vector or `data.frame` to split
- Second argument is a `factor` or `integer` whose values define the partitions

```
# Get the rows corresponding to each of the years in the mortgage data
 year_splits <- split(1:nrow(mort), mort["year"])
# year_splits is a list
class(year_splits)
```

```
"list"
```

```
# The years that we've split over
names(year_splits)
```

```
"2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015"
```

```
# The first few rows corresponding to the year 2010
year_splits[["2010"]][1:10]
```

```
1  6  7 10 21 23 24 27 29 38
```

Compute using Map()

The `Map()` function processes the partitions

- First argument is the function to apply to each parition
- Second argument is the partitions

Compute using Map()

```
col_missing_count <- function(mort) {
   apply(mort, 2, function(x) sum(x == 9))}
# For each of the years count the number of missing values for
# all columns
missing_by_year <- Map(
   function(x) col_missing_count(mort[x, ]),
   year_splits)

missing_by_year[["2008"]]
```

```
enterprise         record_number                   msa
         0                    12                     0
# ...
```

Combine using Reduce()

The `Reduce()` function combines the results for all partitions

- First argument is the function to combine with
- Second argument is the partitioned data

```
# Calculate the total missing values by column
Reduce(`+`, missing_by_year)
```

```
enterprise         record_number                   msa
         0                    64                     0
```

```
# ...
# Label the rownames with the year
mby <- Reduce(rbind, missing_by_year)
row.names(mby) <- names(year_splits)
mby[1:3, 1:3]
```

```
     enterprise record_number msa
2008          0            12   0
2009          0             8   0
2010          0            10   0
```

Let's practice!

SCALABLE DATA PROCESSING IN R

Visulize your results using Tidyverse

SCALABLE DATA PROCESSING IN R

Michael Kane
Assistant Professor, Yale University

datacamp

Missingness by Year

```
library(ggplot2)
library(tidyr)
library(dplyr)
```

```
mort %>%
    bigtable(c("borrower_gender", "year")) %>%
    as.data.frame()
```

Missingness by Year

```
library(ggplot2)
library(tidyr)
library(dplyr)

mort %>%
   bigtable(c("borrower_gender", "year")) %>%
   as.data.frame() %>%
   mutate(Category = c("Male", "Female", "Not Provided",
                       "Not Applicable", "Missing"))
```

Missingness by Year

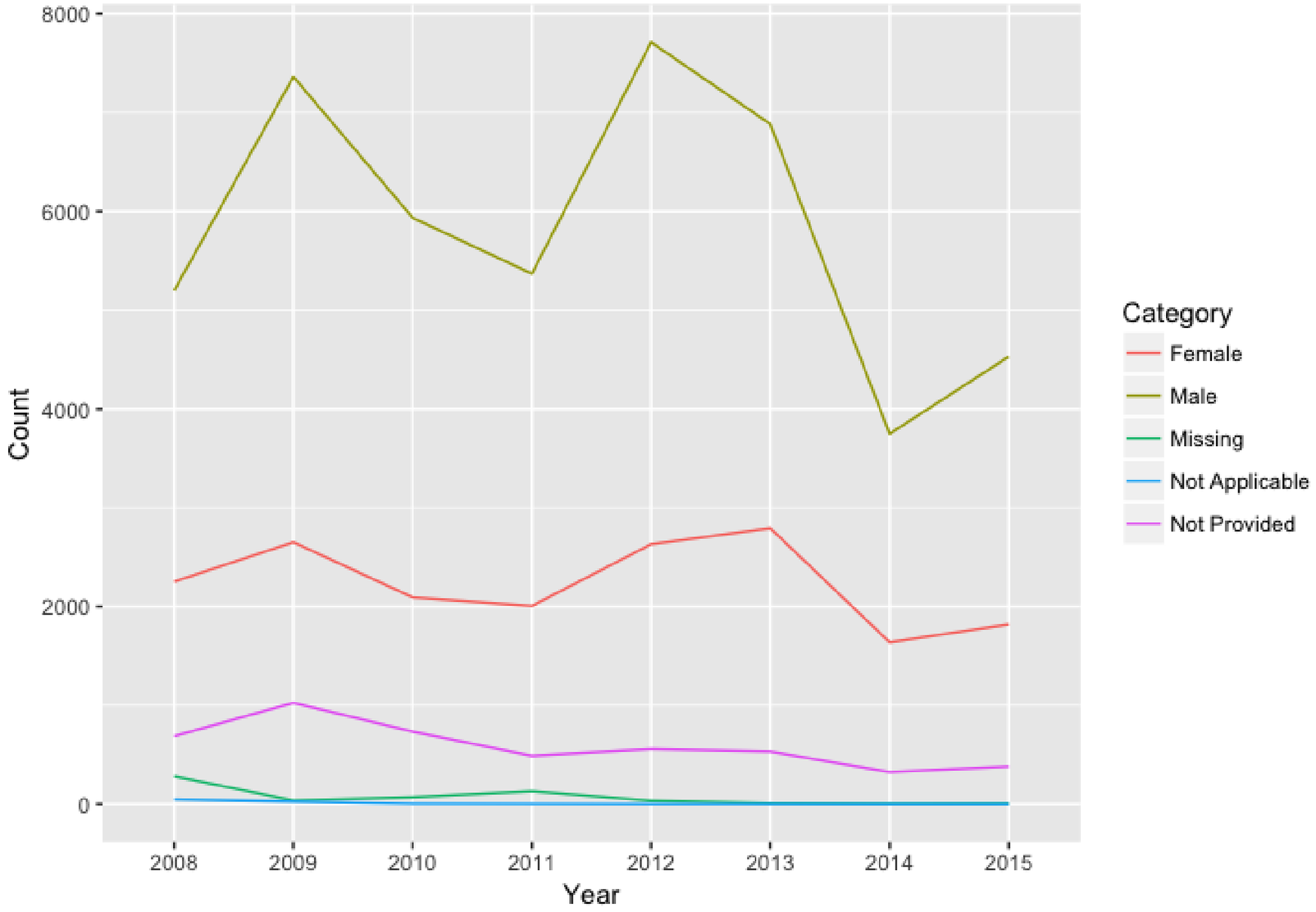
```
library(ggplot2)
library(tidyr)
library(dplyr)

mort %>%
   bigtable(c("borrower_gender", "year")) %>%
   as.data.frame %>%
   mutate(Category = c("Male", "Female", "Not Provided",
                       "Not Applicable", "Missing")) %>%
    gather(Year, Count, -Category)
```

Missingness by Year

```
library(ggplot2)
library(tidyr)
library(dplyr)

mort %>%
    bigtable(c("borrower_gender", "year")) %>%
    as.data.frame %>%
    mutate(Category = c("Male", "Female", "Not Provided",
                        "Not Applicable", "Missing")) %>%
    gather(Year, Count, -Category) %>%
    ggplot(aes(x = Year, y = Count, group = Category,
               color = Category))                          +
    geom_line()
```


8000
6000
4000
2000
0
Count
2008
2009
2010
2011
2012
2013
2014
2015
Year
Category
Female
Male
Missing
Not Applicable
Not Provided

Let's practice!

SCALABLE DATA PROCESSING IN R

Limitations of bigmemory

SCALABLE DATA PROCESSING IN R

Michael Kane
Assistant Professor, Yale University

datacamp

Where can you use bigmemory?

- You can use bigmemory when your data are
  - matrices
  - dense
  - numeric
- Underlying data structures are compatible with low-level linear algebra libraries for fast model fitting
- If you have different column types, you could try the `ff` package

Understanding disk access

> A big.matrix is a data structure designed for random access

Disadvantages of random access

- Can't add rows or columns to an existing `big.matrix` object
- You need to have enough disk space to hold the entire matrix in one big block

Let's practice!

SCALABLE DATA PROCESSING IN R