Reading Excel and Using Tidyverse

School of Information Studies
Syracuse University

Using readxl to Read an Excel File

```
#install.packages("readxl")
library("readxl")
library("tidyverse")

#Define the URL across multiple lines. (just to make it easier to cut and paste code)
part1 <-"http://www2.census.gov/programs-surveys/popest/tables/2010-2011/state/totals/"
part2 <- "nst-est2011-01.xls"
dataFile <- paste0(part1, part2)

#Download the file from the web, into tmpExcelFile
download.file(dataFile, "tmpExcelFile.xls")

#Now read the Excel file into R
testFrame <- read_excel("tmpExcelFile.xls")
```

Using Glimpse

glimpse(testFrame)

- Not that helpful (yet)

```
Rows: 66
Columns: 5
$ `table with row headers in column A and column headers in rows 3 through 4. (leading dots indicate sub-parts)` <chr> "Table 1. …
$ ...2                                                                                                           <chr> NA, "40269…
$ ...3                                                                                                           <chr> NA, NA, "E…
$ ...4                                                                                                           <chr> NA, "Popul…
$ ...5                                                                                                           <dbl> NA, NA, 20…
```

Take an Initial Look at the DataFrame

View(testFrame)

	table with row headers in column A and column headers in rows 3 through 4. (leading dots indicate sub-parts)	...2	...3	...4	...5
1	Table 1. Annual Estimates of the Population for the U...	*NA*	*NA*	*NA*	*NA*
2	Geographic Area	40269	*NA*	Population Estimates (as of July 1)	*NA*
3	*NA*	Census	Estimates Base	2010	2011
4	United States	308745538	308745538	309330219	311591917
5	Northeast	55317240	55317244	55366108	55521598
6	Midwest	66927001	66926987	66976458	67158835
7	South	114555744	114555757	114857529	116046736
8	West	71945553	71945550	72130124	72864748
9	.Alabama	4779736	4779735	4785401	4802740
10	.Alaska	710231	710231	714146	722718
11	.Arizona	6392017	6392013	6413158	6482505
12	.Arkansas	2915918	2915921	2921588	2937979
13	.California	37253956	37253956	37338198	37691912
14	.Colorado	5029196	5029196	5047692	5116796

Renaming the stateName Column

```
#Create a better column name
testFrame$stateName <- pull(testFrame, 1)
testFrame <- testFrame[,-1]

#Remove the '.'
testFrame$stateName <- str_replace(testFrame$stateName,"\\.","")
head(testFrame, 8)
```

```
# A tibble: 8 x 5
  ...2     ...3       ...4                    ...5 stateName
  <chr>    <chr>      <chr>                  <dbl> <chr>
1 NA       NA         NA                        NA Table 1 Annual Estimates
2 40269    NA         Population Estimates …    NA Geographic Area
3 Census   Estimates… 2010                    2.01e3 NA
4 3087455… 308745538  309330219              3.12e8 United States
5 55317240 55317244   55366108               5.55e7 Northeast
6 66927001 66926987   66976458               6.72e7 Midwest
7 1145557… 114555757  114857529              1.16e8 South
8 71945553 71945550   72130124               7.29e7 West
```

```
#Remove the first eight rows
testFrame < slice(testFrame, -1:-8)
```

Continue Cleaning the DataFrame

```
> glimpse(testFrame)
Rows: 58
Columns: 5
$ ...2      <chr> "4779736", "710231", "6392017", "29
$ ...3      <chr> "4779735", "710231", "6392013", "29
$ ...4      <chr> "4785401", "714146", "6413158", "29
$ ...5      <dbl> 4802740, 722718, 6482505, 2937979,
$ stateName <chr> "Alabama", "Alaska", "Arizona", "Ar
> tail(testFrame,8)
# A tibble: 8 x 5
  ...2    ...3    ...4      ...5 stateName
  <chr>   <chr>   <chr>    <dbl> <chr>
1 563626  563626  564554  568158 Wyoming
2 NA      NA      NA          NA NA
3 3725789 3725789 37219… 3706690 Puerto Rico
4 NA      NA      NA          NA Note: The April 1, 2
5 NA      NA      NA          NA Suggested Citation:
6 NA      NA      NA          NA Table 1 Annual Estim
7 NA      NA      NA          NA Source: US. Census E
8 NA      NA      NA          NA Release Date: Decemb
```

Continue Cleaning the DataFrame (cont.)

```
#Remove the bottom lines
testFrame <- slice(testFrame, -52:-58)

#Rename the columns
testFrame <- testFrame %>% rename(april10census = '...2')
testFrame <- testFrame %>% rename(april10base = '...3')
testFrame <- testFrame %>% rename(july10pop = '...4')
testFrame <- testFrame %>% rename(july11pop = '...5')

#Make the populations numeric
testFrame <- testFrame %>% mutate_at(vars(april10census,april10base,july10pop,july11pop),
as.numeric)
```

The Cleaned DataFrame

#Look at the first few rows

head(testFrame)

```
# A tibble: 6 x 5
  april10census april10base july10pop july11pop stateName
          <dbl>       <dbl>     <dbl>     <dbl> <chr>
1       4779736     4779735   4785401   4802740 Alabama
2        710231      710231    714146    722718 Alaska
3       6392017     6392013   6413158   6482505 Arizona
4       2915918     2915921   2921588   2937979 Arkansas
5      37253956    37253956  37338198  37691912 California
6       5029196     5029196   5047692   5116796 Colorado
```

Using group_by

```
#Remove the District of Columbia column
testFrame <- testFrame %>%  filter(stateName != "District of Columbia")

#Add the region for each state
testFrame <- testFrame %>% add_column(region=state.region)

#Calculate the region mean
testFrame %>%. group_by(region) %>%  summarise( regionMean = mean(july11pop),  .groups="drop")
```

```
# A tibble: 4 x 2
  region        regionMean
  <fct>              <dbl>
1 Northeast       6169066.
2 South           7214296.
3 North Central   5596570.
4 West            5604981.
```

Using group_by (cont.)

```
#Store the region mean as a new column
testFrame <-testFrame %>%  group_by(region) %>%
            mutate(regionMean = mean(july11pop))

#Look at the updated dataframe
head(testFrame, 2)
```

```
# A tibble: 2 x 7
# Groups:   region [2]
  april10census april10base july10pop july11pop stateName region regionMean
          <dbl>       <dbl>     <dbl>     <dbl> <chr>     <fct>        <dbl>
1       4779736     4779735   4785401   4802740 Alabama   South     7214296.
2        710231      710231    714146    722718 Alaska    West      5604981.
```

Using JSON in R

School of Information Studies
Syracuse University

Common Vocabulary

JSON—JavaScript Object Notation: a human and machine readable data interchange format; not XML

XML—Extensible markup language: a human and machine readable data interchange format; an international standard

SOAP—Simple object access protocol: a standardized method, using XML, for exchanging structured information among clients and servers

REST—Representational state transfer: a software design style or strategy for organizing interactions between clients and servers; not a standard

Example Architecture

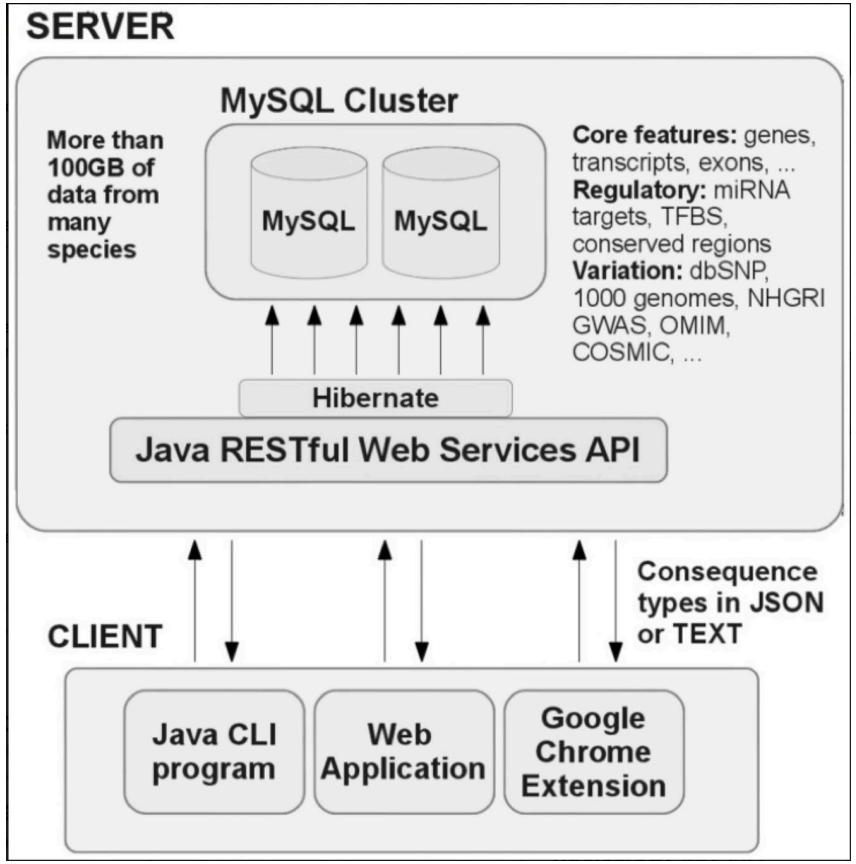


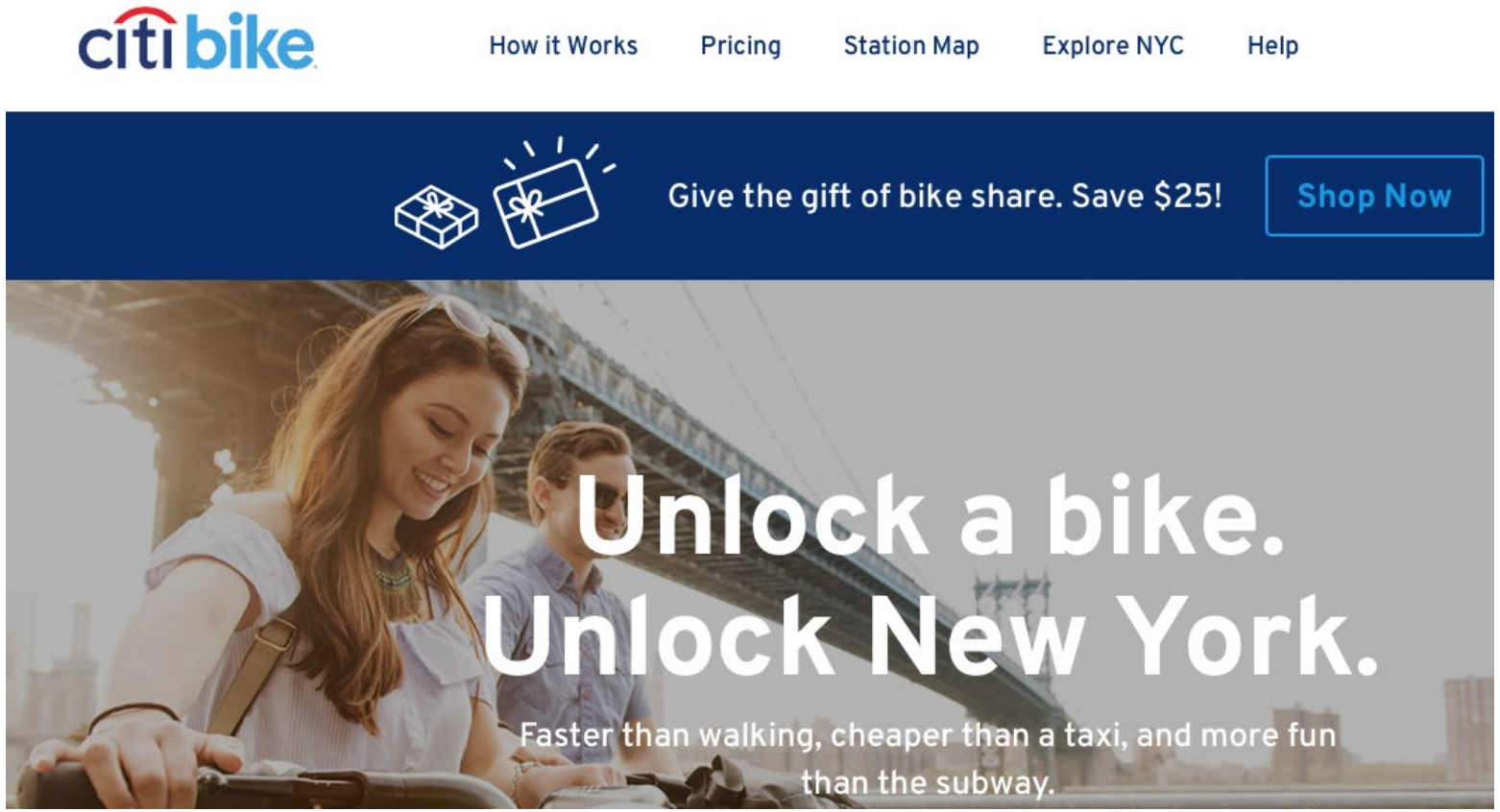
Image credit: Medina et al., 2012

Data and applications pertinent to research on genetic data

Java package for mapping an object-oriented data model to standard SQL

JSON is used to transmit data objects to and from several possible clients

JSON Data Access Example, Part I



JSON Data Access Example, Part II

```
#For access to Internet data
library(RCurl)

#For decoding JSON
library(jsonlite)

#URL of JSON data
bikeURL <- 'https://gbfs.citibikenyc.com/gbfs/en/station_status.json’
```

JSON Data Access Example, Part III

```
#Grab the JSON data
apiResult <- getURL(bikeURL)

#Parse the data
results <- fromJSON(apiResult)

#Look at our data
str(results)

List of 3
 $ data       :List of 1
  …$ stations:'data.frame':     1365 obs. of  13 variables:
```

Parsing JSON Data

#See the data generated

stations <- results$data$stations

glimpse(stations)

```
Rows: 1,367
Columns: 15
$ last_reported                <int> 1614275588, 1614275427, 1614276152, 1614274376, 1614274653, 1614276021, 1614275964, 161427…
$ station_id                   <chr> "72", "79", "82", "83", "116", "119", "120", "127", "128", "143", "144", "146", "150", "15…
$ num_bikes_disabled           <int> 3, 0, 2, 0, 3, 1, 1, 0, 3, 3, 1, 0, 0, 4, 1, 2, 1, 1, 3, 2, 3, 1, 1, 0, 1, 1, 0, 4, 2, 1, …
$ num_bikes_available          <int> 21, 14, 21, 34, 5, 17, 4, 4, 15, 8, 47, 19, 11, 25, 15, 39, 12, 20, 18, 45, 25, 5, 29, 3, …
$ station_status               <chr> "active", "active", "active", "active", "active", "active", "active", "active", "active", …
$ is_renting                   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ num_docks_available          <int> 31, 19, 4, 28, 42, 35, 14, 27, 38, 13, 10, 20, 45, 4, 33, 22, 9, 14, 26, 0, 42, 29, 30, 34…
$ num_ebikes_available         <int> 0, 2, 0, 3, 0, 4, 2, 0, 7, 1, 2, 4, 1, 6, 1, 7, 1, 0, 2, 2, 1, 0, 0, 1, 4, 3, 2, 5, 1, 0, …
$ is_returning                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ num_docks_disabled           <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, …
$ legacy_id                    <chr> "72", "79", "82", "83", "116", "119", "120", "127", "128", "143", "144", "146", "150", "15…
$ eightd_has_available_keys    <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,…
$ is_installed                 <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, …
$ eightd_active_station_services <list> [NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL…
$ valet                        <df[,7]> <data.frame[44 x 7]>
```

Munging the Data

```
#Keep a subset of the columns
colsToKeep <- c('num_bikes_disabled','num_docks_disabled', 'station_id',
                'num_ebikes_available', 'num_bikes_available', 'num_docks_available')

#Both of these commands do the same thing
stations1 <- stations[,colsToKeep]
stations2 <- stations %>% select(colsToKeep)

str(stations2)
```

```
'data.frame':   1367 obs. of  6 variables:
 $ num_bikes_disabled  : int  3 0 2 0 3 1 1 0 3 3 ...
 $ num_docks_disabled  : int  0 0 0 0 0 0 0 0 0 0 ...
 $ station_id          : chr  "72" "79" "82" "83" ...
 $ num_ebikes_available: int  0 2 0 3 0 4 2 0 7 1 ...
 $ num_bikes_available : int  21 14 21 34 5 17 4 4 15 8 ...
 $ num_docks_available : int  31 19 4 28 42 35 14 27 38 13 ...
```

Exploring the Parsed Data

```
#Keep a subset of the columns
mean(stations2$num_docks_available)
[1] 18.17557

mean(stations2$num_bikes_available)
[1] 10.71836

#How many stations have a bike available – could have used ‘filter’
bikesAvailDF <- stations2[stations2$num_bikes_available>0,]
nrow(bikesAvailDF)
[1] 1255
```

Questions

Why are these data available via JSON?

What would be some other good (and bad) alternatives for publishing these data?

Why did they make citibike data available at all?