Downloading files

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Get/set your working directory

- A basic component of working with data is knowing your working directory
- The two main commands are `getwd()` and `setwd()`.
- Be aware of relative versus absolute paths
  - **Relative** - `setwd("./data")`, `setwd("../")`
  - **Absolute** - `setwd("/Users/jtleek/data/")`
- Important difference in Windows `setwd("C:\\Users\\Andrew\\Downloads")`

Checking for and creating directories

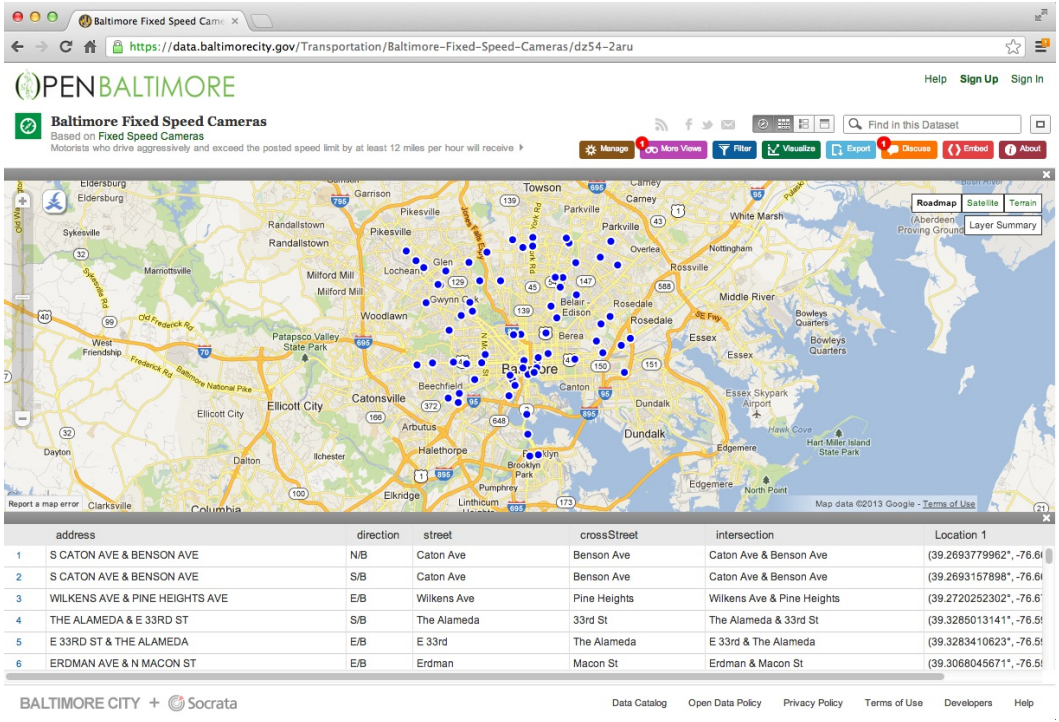
- `file.exists("directoryName")` will check to see if the directory exists
- `dir.create("directoryName")` will create a directory if it doesn't exist
- Here is an example checking for a "data" directory and creating it if it doesn't exist

```
if (!file.exists("data")) {
    dir.create("data")
}
```

Getting data from the internet - download.file()

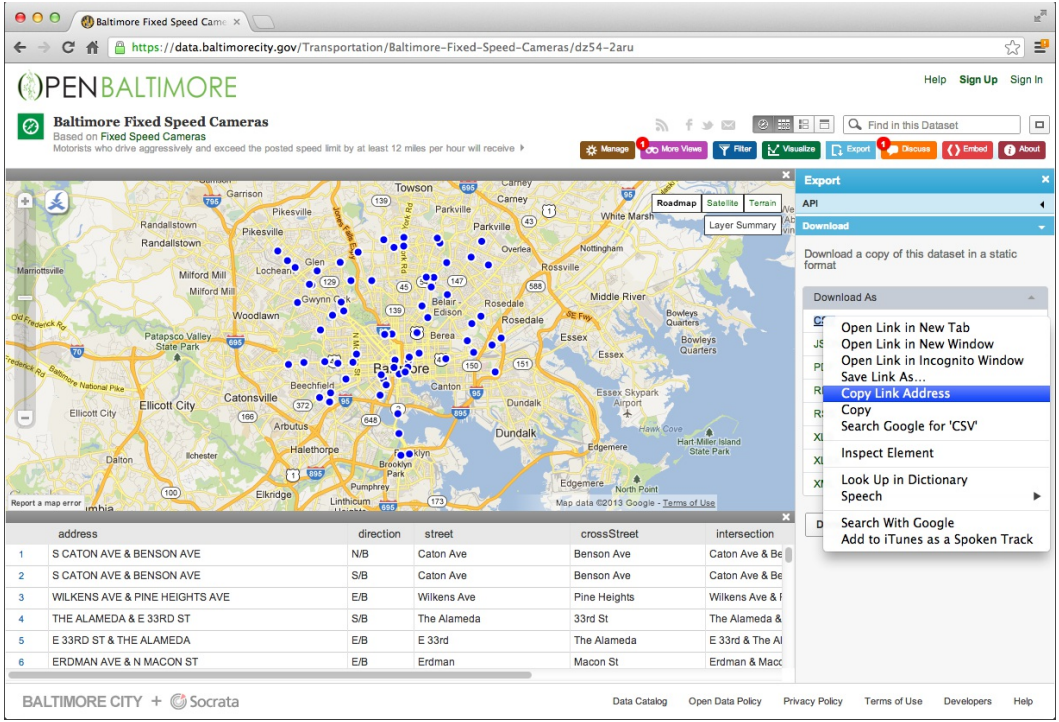
- Downloads a file from the internet
- Even if you could do this by hand, helps with reproducibility
- Important parameters are *url*, *destfile*, *method*
- Useful for downloading tab-delimited, csv, and other files

Example - Baltimore camera data



https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru

Example - Baltimore camera data



https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru

Download a file from the web

```
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl, destfile = "./data/cameras.csv", method = "curl")
list.files("./data")
```

```
## [1] "cameras.csv"
```

```
dateDownloaded <- date()
dateDownloaded
```

```
## [1] "Sun Jan 12 21:37:44 2014"
```

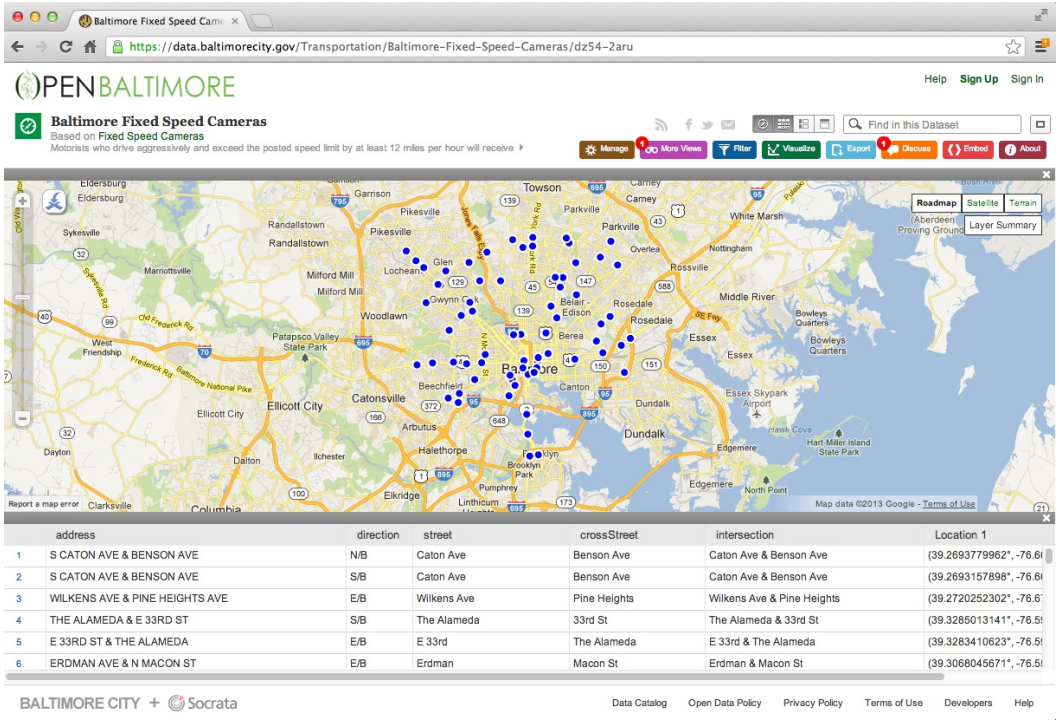
Some notes about download.file()

- If the url starts with *http* you can use download.file()
- If the url starts with *https* on Windows you may be ok
- If the url starts with *https* on Mac you may need to set *method="curl"*
- If the file is big, this might take a while
- Be sure to record when you downloaded.

Reading local flat files

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

Example - Baltimore camera data



https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru

Download the file to load

```
if (!file.exists("data")) {
    dir.create("data")
}
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.csv?accessType=DOWNLOAD"
download.file(fileUrl, destfile = "cameras.csv", method = "curl")
dateDownloaded <- date()
```

Loading flat files - read.table()

- This is the main function for reading data into R
- Flexible and robust but requires more parameters
- Reads the data into RAM - big data can cause problems
- Important parameters *file*, *header*, *sep*, *row.names*, *nrows*
- Related: *read.csv()*, *read.csv2()*

Baltimore example

```
cameraData <- read.table("./data/cameras.csv")
```

```
## Error: line 1 did not have 13 elements
```

```
head(cameraData)
```

```
## Error: object 'cameraData' not found
```

Example: Baltimore camera data

```
cameraData <- read.table("./data/cameras.csv", sep = ",", header = TRUE)
head(cameraData)
```

```
##                         address direction      street  crossStreet
## 1       S CATON AVE & BENSON AVE       N/B   Caton Ave   Benson Ave
## 2       S CATON AVE & BENSON AVE       S/B   Caton Ave   Benson Ave
## 3 WILKENS AVE & PINE HEIGHTS AVE       E/B Wilkens Ave Pine Heights
## 4        THE ALAMEDA & E 33RD ST       S/B The Alameda      33rd St
## 5        E 33RD ST & THE ALAMEDA       E/B      E 33rd  The Alameda
## 6        ERDMAN AVE & N MACON ST       E/B      Erdman     Macon St
##                intersection                      Location.1
## 1     Caton Ave & Benson Ave (39.2693779962, -76.6688185297)
## 2     Caton Ave & Benson Ave (39.2693157898, -76.6689698176)
## 3 Wilkens Ave & Pine Heights  (39.2720252302, -76.676960806)
## 4     The Alameda  & 33rd St (39.3285013141, -76.5953545714)
## 5      E 33rd  & The Alameda (39.3283410623, -76.5953594625)
## 6         Erdman  & Macon St (39.3068045671, -76.5593167803)
```

Example: Baltimore camera data

read.csv sets *sep=","* and *header=TRUE*

```
cameraData <- read.csv("./data/cameras.csv")
head(cameraData)
```

```
##                         address direction      street  crossStreet
## 1       S CATON AVE & BENSON AVE       N/B   Caton Ave   Benson Ave
## 2       S CATON AVE & BENSON AVE       S/B   Caton Ave   Benson Ave
## 3 WILKENS AVE & PINE HEIGHTS AVE       E/B Wilkens Ave Pine Heights
## 4        THE ALAMEDA & E 33RD ST       S/B The Alameda      33rd St
## 5        E 33RD ST & THE ALAMEDA       E/B      E 33rd  The Alameda
## 6        ERDMAN AVE & N MACON ST       E/B      Erdman     Macon St
##                intersection                       Location.1
## 1     Caton Ave & Benson Ave (39.2693779962, -76.6688185297)
## 2     Caton Ave & Benson Ave (39.2693157898, -76.6689698176)
## 3 Wilkens Ave & Pine Heights  (39.2720252302, -76.676960806)
## 4     The Alameda  & 33rd St (39.3285013141, -76.5953545714)
## 5      E 33rd  & The Alameda (39.3283410623, -76.5953594625)
## 6         Erdman  & Macon St (39.3068045671, -76.5593167803)
```

Some more important parameters

- *quote* - you can tell R whether there are any quoted values quote="" means no quotes.
- *na.strings* - set the character that represents a missing value.
- *nrows* - how many rows to read of the file (e.g. nrows=10 reads 10 lines).
- *skip* - number of lines to skip before starting to read

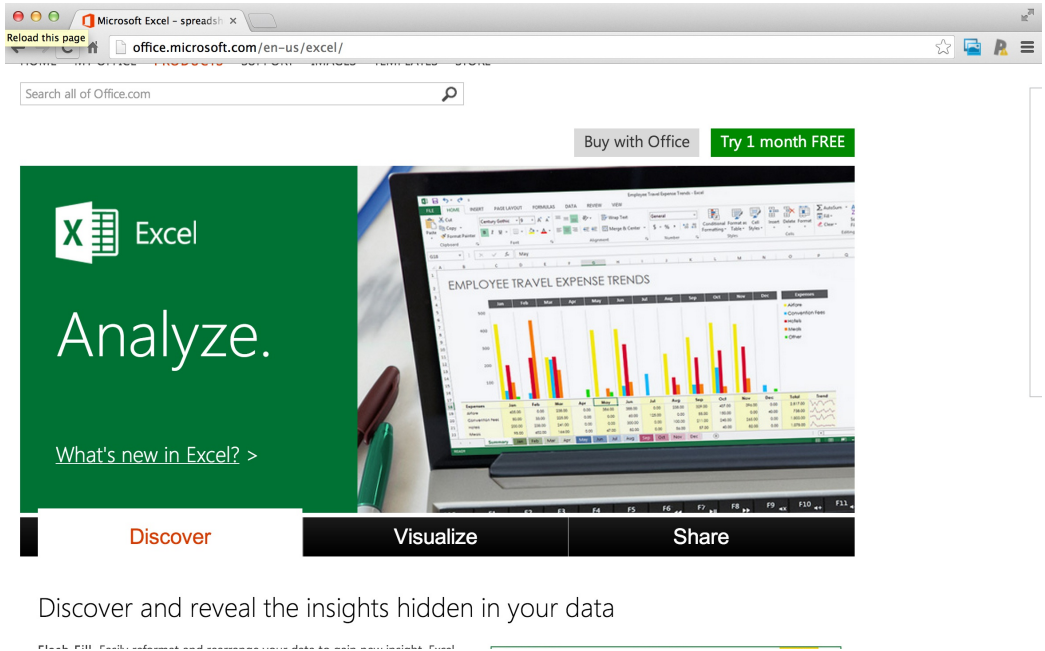
In my experience, the biggest trouble with reading flat files are quotation marks ` or " placed in data values, setting quote="" often resolves these.

Reading Excel files

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

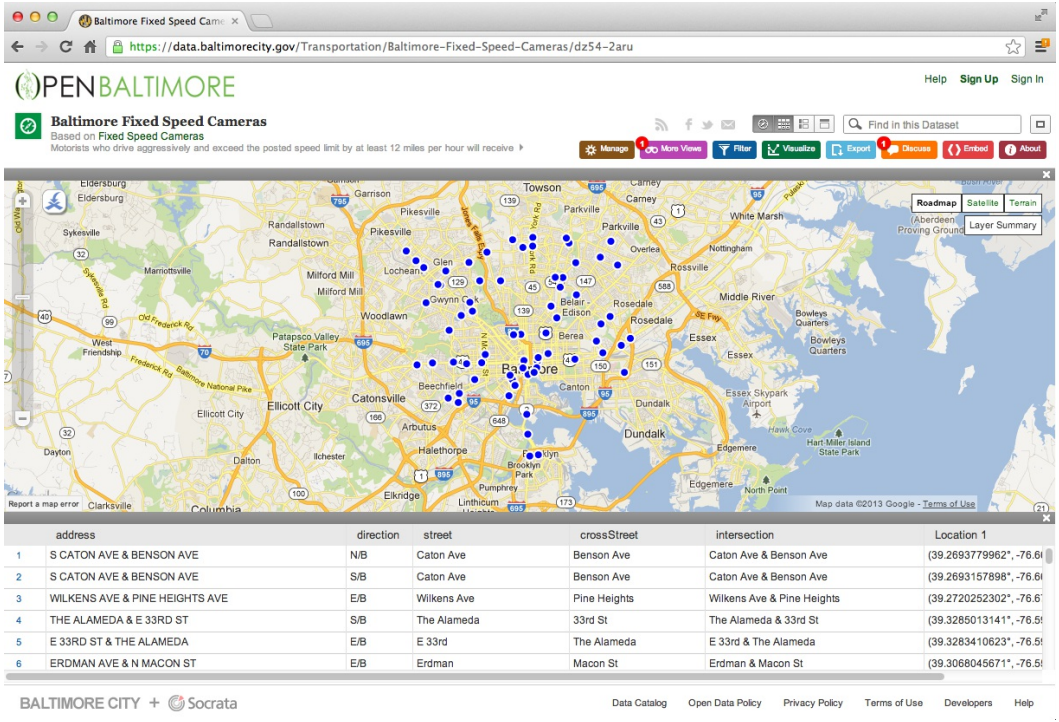
Excel files

Still probably the most widely used format for sharing data



http://office.microsoft.com/en-us/excel/

Example - Baltimore camera data



https://data.baltimorecity.gov/Transportation/Baltimore-Fixed-Speed-Cameras/dz54-2aru

Download the file to load

```
if(!file.exists("data")){dir.create("data")}
fileUrl <- "https://data.baltimorecity.gov/api/views/dz54-2aru/rows.xlsx?accessType=DOWNLOAD"
download.file(fileUrl,destfile="./data/cameras.xlsx",method="curl")
dateDownloaded <- date()
```

read.xlsx(), read.xlsx2() {xlsx package}

```
library(xlsx)
cameraData <- read.xlsx("./data/cameras.xlsx",sheetIndex=1,header=TRUE)
head(cameraData)
```

```
                        address direction      street  crossStreet                intersection
1       S CATON AVE & BENSON AVE       N/B   Caton Ave   Benson Ave     Caton Ave & Benson Ave
2       S CATON AVE & BENSON AVE       S/B   Caton Ave   Benson Ave     Caton Ave & Benson Ave
3 WILKENS AVE & PINE HEIGHTS AVE       E/B Wilkens Ave Pine Heights Wilkens Ave & Pine Heights
4        THE ALAMEDA & E 33RD ST       S/B The Alameda       33rd St     The Alameda  & 33rd St
5        E 33RD ST & THE ALAMEDA       E/B      E 33rd  The Alameda      E 33rd  & The Alameda
6
1 (39.2693779962, -76.6688185297)
2 (39.2693157898, -76.6689698176)
3  (39.2720252302, -76.676960806)
4 (39.3285013141, -76.5953545714)
5 (39.3283410623, -76.5953594625)
6 (39.3068045671, -76.5593167803)
```

Reading specific rows and columns

```
colIndex <- 2:3
rowIndex <- 1:4
cameraDataSubset <- read.xlsx("./data/cameras.xlsx",sheetIndex=1,
                              colIndex=colIndex,rowIndex=rowIndex)
cameraDataSubset
```

```
  direction      street
1       N/B   Caton Ave
2       S/B   Caton Ave
3       E/B Wilkens Ave
```

Further notes

- The *write.xlsx* function will write out an Excel file with similar arguments.
- *read.xlsx2* is much faster than *read.xlsx* but for reading subsets of rows may be slightly unstable.
- The XLConnect package has more options for writing and manipulating Excel files
- The XLConnect vignette is a good place to start for that package
- In general it is advised to store your data in either a database or in comma separated files (.csv) or tab separated files (.tab/.txt) as they are easier to distribute.

Reading JSON

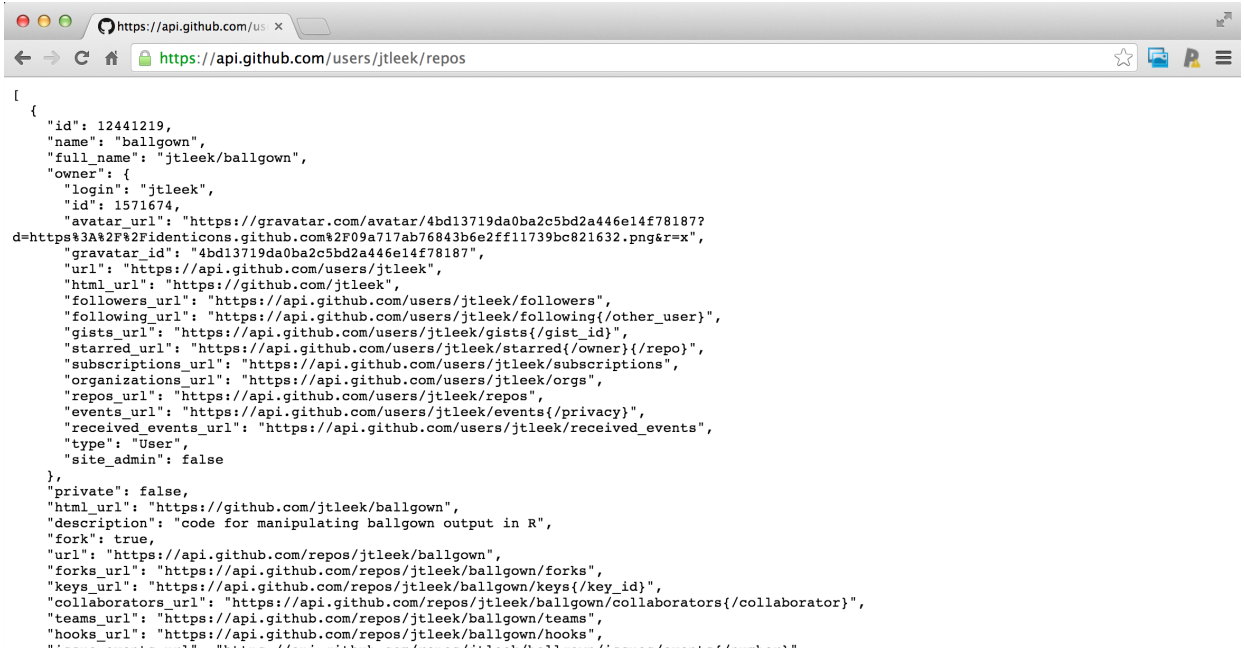
Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

JSON

- Javascript Object Notation
- Lightweight data storage
- Common format for data from application programming interfaces (APIs)
- Similar structure to XML but different syntax/format
- Data stored as
  - Numbers (double)
  - Strings (double quoted)
  - Boolean (*true* or *false*)
  - Array (ordered, comma separated enclosed in square brackets *[]*)
  - Object (unorderd, comma separated collection of key:value pairs in curley brackets *{}*)

http://en.wikipedia.org/wiki/JSON

Example JSON file




```
[
  {
    "id": 12441219,
    "name": "ballgown",
    "full_name": "jtleek/ballgown",
    "owner": {
      "login": "jtleek",
      "id": 1571674,
      "avatar_url": "https://gravatar.com/avatar/4bd13719da0ba2c5bd2a446e14f781877
d=https%3A%2F%2Fidenticons.github.com%2F09a717ab76843b6e2ff11739bc821632.png&r=x",
      "gravatar_id": "4bd13719da0ba2c5bd2a446e14f78187",
      "url": "https://api.github.com/users/jtleek",
      "html_url": "https://github.com/jtleek",
      "followers_url": "https://api.github.com/users/jtleek/followers",
      "following_url": "https://api.github.com/users/jtleek/following{/other_user}",
      "gists_url": "https://api.github.com/users/jtleek/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/jtleek/starred{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/jtleek/subscriptions",
      "organizations_url": "https://api.github.com/users/jtleek/orgs",
      "repos_url": "https://api.github.com/users/jtleek/repos",
      "events_url": "https://api.github.com/users/jtleek/events{/privacy}",
      "received_events_url": "https://api.github.com/users/jtleek/received_events",
      "type": "User",
      "site_admin": false
    },
    "private": false,
    "html_url": "https://github.com/jtleek/ballgown",
    "description": "code for manipulating ballgown output in R",
    "fork": true,
    "url": "https://api.github.com/repos/jtleek/ballgown",
    "forks_url": "https://api.github.com/repos/jtleek/ballgown/forks",
    "keys_url": "https://api.github.com/repos/jtleek/ballgown/keys{/key_id}",
    "collaborators_url": "https://api.github.com/repos/jtleek/ballgown/collaborators{/collaborator}",
    "teams_url": "https://api.github.com/repos/jtleek/ballgown/teams",
    "hooks_url": "https://api.github.com/repos/jtleek/ballgown/hooks",
```

Reading data from JSON {jsonlite package}

```
library(jsonlite)
jsonData <- fromJSON("https://api.github.com/users/jtleek/repos")
names(jsonData)
```

```
 [1] "id"               "name"             "full_name"        "owner"
 [5] "private"          "html_url"         "description"      "fork"
 [9] "url"              "forks_url"        "keys_url"         "collaborators_url"
[13] "teams_url"        "hooks_url"        "issue_events_url" "events_url"
[17] "assignees_url"    "branches_url"     "tags_url"         "blobs_url"
[21] "git_tags_url"     "git_refs_url"     "trees_url"        "statuses_url"
[25] "languages_url"    "stargazers_url"   "contributors_url" "subscribers_url"
[29] "subscription_url" "commits_url"      "git_commits_url"  "comments_url"
[33] "issue_comment_url" "contents_url"    "compare_url"      "merges_url"
[37] "archive_url"      "downloads_url"    "issues_url"       "pulls_url"
[41] "milestones_url"   "notifications_url" "labels_url"     "releases_url"
[45] "created_at"       "updated_at"       "pushed_at"        "git_url"
[49] "ssh_url"          "clone_url"        "svn_url"          "homepage"
[53] "size"             "stargazers_count" "watchers_count"   "language"
```

Nested objects in JSON

```
names(jsonData$owner)
```

```
 [1] "login"             "id"                "avatar_url"          "gravatar_id"
 [5] "url"               "html_url"          "followers_url"       "following_url"
 [9] "gists_url"         "starred_url"       "subscriptions_url"   "organizations_url"
[13] "repos_url"         "events_url"        "received_events_url" "type"
[17] "site_admin"
```

```
jsonData$owner$login
```

```
 [1] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
[11] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
[21] "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek" "jtleek"
```

Writing data frames to JSON

```
myjson <- toJSON(iris, pretty=TRUE)
cat(myjson)
```

```
[
    {
        "Sepal.Length" : 5.1,
        "Sepal.Width" : 3.5,
        "Petal.Length" : 1.4,
        "Petal.Width" : 0.2,
        "Species" : "setosa"
    },
    {
        "Sepal.Length" : 4.9,
        "Sepal.Width" : 3,
        "Petal.Length" : 1.4,
        "Petal.Width" : 0.2,
        "Species" : "setosa"
    },
    {
        "Sepal.Length" : 4.7,
```

Convert back to JSON

```
iris2 <- fromJSON(myjson)
head(iris2)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

http://www.r-bloggers.com/new-package-jsonlite-a-smarter-json-encoderdecoder/

Further resources

- http://www.json.org/
- A good tutorial on jsonlite - http://www.r-bloggers.com/new-package-jsonlite-a-smarter-json-encoderdecoder/
- jsonlite vignette

Reading XML

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

XML

- Extensible markup language
- Frequently used to store structured data
- Particularly widely used in internet applications
- Extracting XML is the basis for most web scraping
- Components
    - Markup - labels that give the text structure
    - Content - the actual text of the document

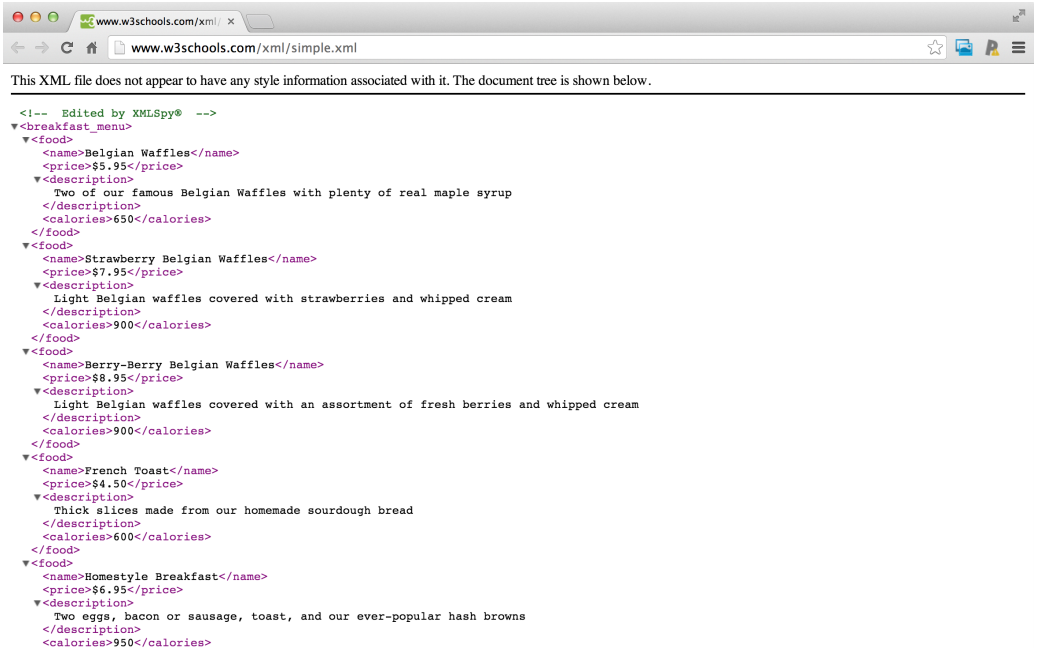
http://en.wikipedia.org/wiki/XML

Tags, elements and attributes

- Tags correspond to general labels
  - Start tags `<section>`
  - End tags `</section>`
  - Empty tags `<line-break />`
- Elements are specific examples of tags
  - `<Greeting> Hello, world </Greeting>`
- Attributes are components of the label
  - `<img src="jeff.jpg" alt="instructor"/>`
  - `<step number="3"> Connect A to B. </step>`

http://en.wikipedia.org/wiki/XML

Example XML file



http://www.w3schools.com/xml/simple.xml

Read the file into R

```
library(XML)
fileUrl <- "http://www.w3schools.com/xml/simple.xml"
doc <- xmlTreeParse(fileUrl,useInternal=TRUE)
rootNode <- xmlRoot(doc)
xmlName(rootNode)
```

```
[1] "breakfast_menu"
```

```
names(rootNode)
```

```
  food   food   food   food   food
"food" "food" "food" "food" "food"
```

Directly access parts of the XML document

```
rootNode[[1]]
```

```
<food>
  <name>Belgian Waffles</name>
  <price>$5.95</price>
  <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
  <calories>650</calories>
</food>
```

```
rootNode[[1]][[1]]
```

```
<name>Belgian Waffles</name>
```

Programatically extract parts of the file

```
xmlSApply(rootNode,xmlValue)
```

```
                                "Belgian Waffles$5.95Two of our famous Belgian Waffles with plenty of rea

                          "Strawberry Belgian Waffles$7.95Light Belgian waffles covered with strawberries and

"Berry-Berry Belgian Waffles$8.95Light Belgian waffles covered with an assortment of fresh berries and

                                                    "French Toast$4.50Thick slices made from our homemade so

                          "Homestyle Breakfast$6.95Two eggs, bacon or sausage, toast, and our ever-popula
```

Programatically extract parts of the file

```
xmlSApply(rootNode,xmlValue)
```

```
                                "Belgian Waffles$5.95Two of our famous Belgian Waffles with plenty of rea

                          "Strawberry Belgian Waffles$7.95Light Belgian waffles covered with strawberries and

"Berry-Berry Belgian Waffles$8.95Light Belgian waffles covered with an assortment of fresh berries and

                                                        "French Toast$4.50Thick slices made from our homemade so

                           "Homestyle Breakfast$6.95Two eggs, bacon or sausage, toast, and our ever-popula
```

XPath

- */node* Top level node
- *//node* Node at any level
- *node[@attr-name]* Node with an attribute name
- *node[@attr-name='bob']* Node with attribute name attr-name='bob'

Information from: http://www.stat.berkeley.edu/~statcur/Workshop2/Presentations/XML.pdf

Get the items on the menu and prices

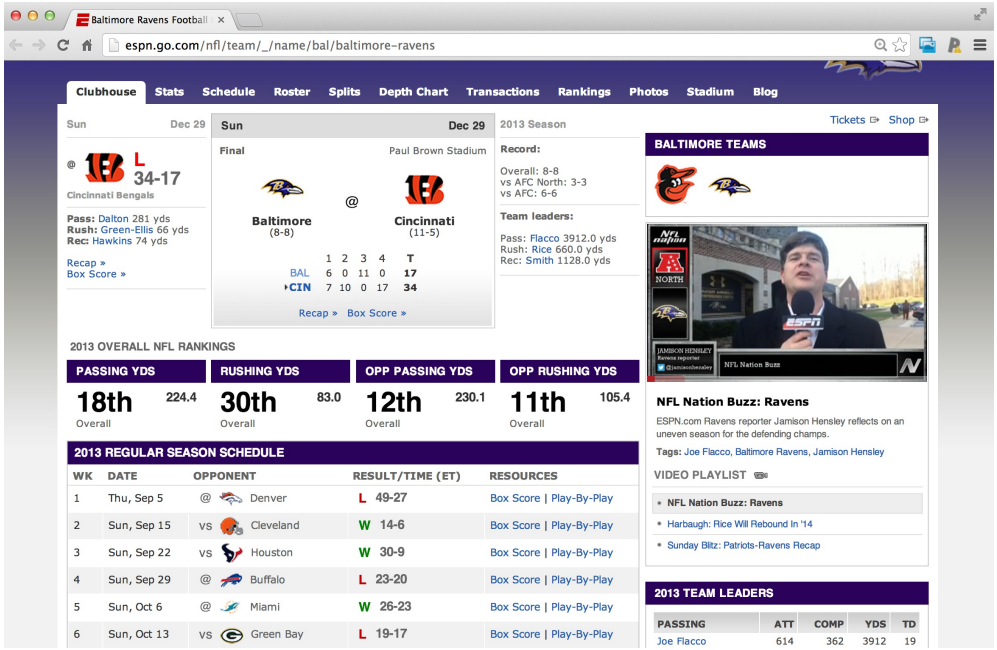
```
xpathSApply(rootNode,"//name",xmlValue)
```

```
[1] "Belgian Waffles"             "Strawberry Belgian Waffles"  "Berry-Berry Belgian Waffles"
[4] "French Toast"                "Homestyle Breakfast"
```

```
xpathSApply(rootNode,"//price",xmlValue)
```

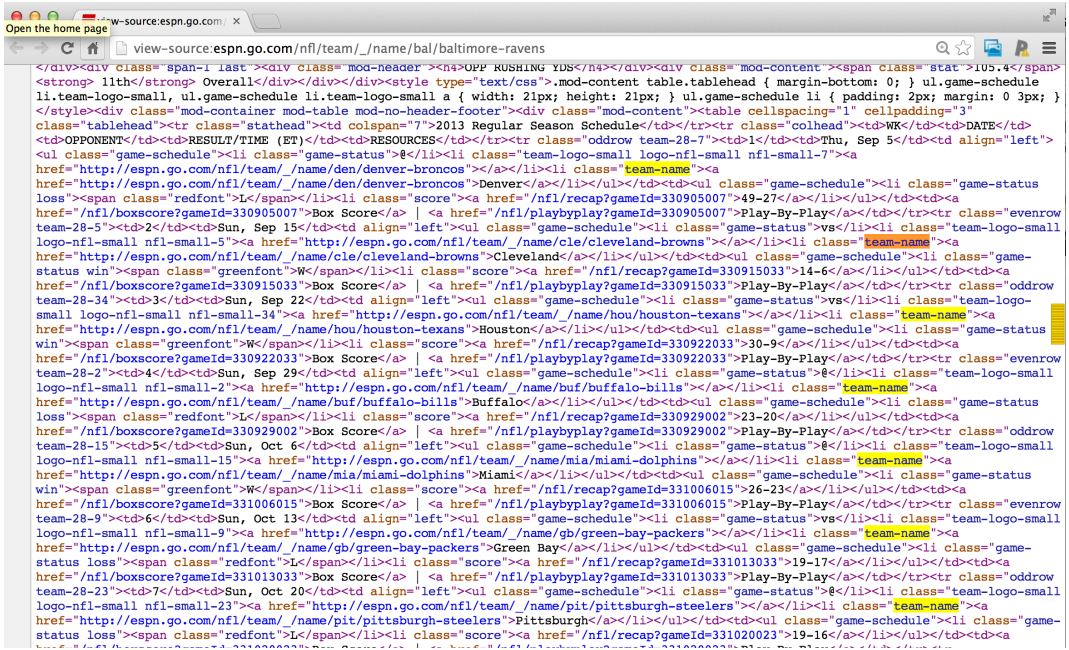
```
[1] "$5.95" "$7.95" "$8.95" "$4.50" "$6.95"
```

Another example



http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens

Viewing the source



http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens

Extract content by attributes

```
fileUrl <- "http://espn.go.com/nfl/team/_/name/bal/baltimore-ravens"
doc <- htmlTreeParse(fileUrl,useInternal=TRUE)
scores <- xpathSApply(doc,"//li[@class='score']",xmlValue)
teams <- xpathSApply(doc,"//li[@class='team-name']",xmlValue)
scores
```

```
 [1] "49-27"    "14-6"     "30-9"     "23-20"    "26-23"    "19-17"    "19-16"    "24-18"
 [9] "20-17 OT" "23-20 OT" "19-3"     "22-20"    "29-26"    "18-16"    "41-7"     "34-17"
```

```
teams
```

```
 [1] "Denver"       "Cleveland"    "Houston"      "Buffalo"      "Miami"        "Green Bay"
 [7] "Pittsburgh"   "Cleveland"    "Cincinnati"   "Chicago"      "New York"     "Pittsburgh"
[13] "Minnesota"    "Detroit"      "New England"  "Cincinnati"
```

Notes and further resources

- Official XML tutorials short, long
- An outstanding guide to the XML package

Using data.table

Jeffrey Leek
Johns Hopkins Bloomberg School of Public Health

data.table

- Inherets from data.frame
    - All functions that accept data.frame work on data.table
- Written in C so it is much faster
- Much, much faster at subsetting, group, and updating

Create data tables just like data frames

```
library(data.table)
DF = data.frame(x=rnorm(9),y=rep(c("a","b","c"),each=3),z=rnorm(9))
head(DF,3)
```

```
       x y        z
1 0.4159 a -0.05855
2 0.8433 a  0.13732
3 1.0585 a  2.16448
```

```
DT = data.table(x=rnorm(9),y=rep(c("a","b","c"),each=3),z=rnorm(9))
head(DT,3)
```

```
          x y      z
1: -0.27721 a 0.2530
2:  1.00158 a 1.5093
3: -0.03382 a 0.4844
```

See all the data tables in memory

```
tables()
```

```
     NAME NROW MB COLS  KEY
[1,] DT      9 1  x,y,z
Total: 1MB
```

Subsetting rows

```
DT[2,]
```

```
      x y     z
1: 1.002 a 1.509
```

```
DT[DT$y=="a",]
```

```
         x y      z
1: -0.27721 a 0.2530
2:  1.00158 a 1.5093
3: -0.03382 a 0.4844
```

Subsetting rows

```
DT[c(2,3)]
```

```
         x y      z
1:  1.00158 a 1.5093
2: -0.03382 a 0.4844
```

Subsetting columns!?

```
DT[,c(2,3)]
```

```
[1] 2 3
```

Column subsetting in data.table

- The subsetting function is modified for data.table
- The argument you pass after the comma is called an "expression"
- In R an expression is a collection of statements enclosed in curley brackets

```
{
  x = 1
  y = 2
}
k = {print(10); 5}
```

```
[1] 10
```

```
print(k)
```

```
[1] 5
```

Calculating values for variables with expressions

```
DT[,list(mean(x),sum(z))]
```

```
        V1     V2
1: 0.05637 0.5815
```

```
DT[,table(y)]
```

```
y
a b c
3 3 3
```

Adding new columns

```
DT[,w:=z^2]
```

```
          x y        z        w
1: -0.27721 a  0.25300 0.064009
2:  1.00158 a  1.50933 2.278091
3: -0.03382 a  0.48437 0.234619
4: -0.70493 b -1.22755 1.506885
5: -1.36402 b -0.64624 0.417631
6: -0.26224 b -0.51427 0.264475
7: -0.10929 c  1.21445 1.474901
8:  1.40234 c  0.07493 0.005614
9:  0.85494 c -0.56652 0.320948
```

Adding new columns

```
DT2 <- DT
DT[, y:= 2]
```

```
          x y        z        w
1: -0.27721 2  0.25300 0.064009
2:  1.00158 2  1.50933 2.278091
3: -0.03382 2  0.48437 0.234619
4: -0.70493 2 -1.22755 1.506885
5: -1.36402 2 -0.64624 0.417631
6: -0.26224 2 -0.51427 0.264475
7: -0.10929 2  1.21445 1.474901
8:  1.40234 2  0.07493 0.005614
9:  0.85494 2 -0.56652 0.320948
```

Careful

```
head(DT,n=3)
```

```
          x y      z       w
1: -0.27721 2 0.2530 0.06401
2:  1.00158 2 1.5093 2.27809
3: -0.03382 2 0.4844 0.23462
```

```
head(DT2,n=3)
```

```
          x y      z       w
1: -0.27721 2 0.2530 0.06401
2:  1.00158 2 1.5093 2.27809
3: -0.03382 2 0.4844 0.23462
```

Multiple operations

```
DT[,m:= {tmp <- (x+z); log2(tmp+5)}]
```

```
          x y        z        w     m
1: -0.27721 2  0.25300 0.064009 2.315
2:  1.00158 2  1.50933 2.278091 2.909
3: -0.03382 2  0.48437 0.234619 2.446
4: -0.70493 2 -1.22755 1.506885 1.617
5: -1.36402 2 -0.64624 0.417631 1.580
6: -0.26224 2 -0.51427 0.264475 2.078
7: -0.10929 2  1.21445 1.474901 2.610
8:  1.40234 2  0.07493 0.005614 2.695
9:  0.85494 2 -0.56652 0.320948 2.403
```

plyr like operations

```
DT[,a:=x>0]
```

```
          x y        z        w     m     a
1: -0.27721 2  0.25300 0.064009 2.315 FALSE
2:  1.00158 2  1.50933 2.278091 2.909  TRUE
3: -0.03382 2  0.48437 0.234619 2.446 FALSE
4: -0.70493 2 -1.22755 1.506885 1.617 FALSE
5: -1.36402 2 -0.64624 0.417631 1.580 FALSE
6: -0.26224 2 -0.51427 0.264475 2.078 FALSE
7: -0.10929 2  1.21445 1.474901 2.610 FALSE
8:  1.40234 2  0.07493 0.005614 2.695  TRUE
9:  0.85494 2 -0.56652 0.320948 2.403  TRUE
```

plyr like operations

```
DT[,b:= mean(x+w),by=a]
```

```
          x y        z        w     m     a      b
1: -0.27721 2  0.25300 0.064009 2.315 FALSE 0.2018
2:  1.00158 2  1.50933 2.278091 2.909  TRUE 1.9545
3: -0.03382 2  0.48437 0.234619 2.446 FALSE 0.2018
4: -0.70493 2 -1.22755 1.506885 1.617 FALSE 0.2018
5: -1.36402 2 -0.64624 0.417631 1.580 FALSE 0.2018
6: -0.26224 2 -0.51427 0.264475 2.078 FALSE 0.2018
7: -0.10929 2  1.21445 1.474901 2.610 FALSE 0.2018
8:  1.40234 2  0.07493 0.005614 2.695  TRUE 1.9545
9:  0.85494 2 -0.56652 0.320948 2.403  TRUE 1.9545
```

Special variables

`.N` An integer, length 1, containing the numbe r

```
set.seed(123);
DT <- data.table(x=sample(letters[1:3], 1E5, TRUE))
DT[, .N, by=x]
```

```
   x     N
1: a 33387
2: c 33201
3: b 33412
```

Keys

```
DT <- data.table(x=rep(c("a","b","c"),each=100), y=rnorm(300))
setkey(DT, x)
DT['a']
```

```
    x        y
  1: a  0.25959
  2: a  0.91751
  3: a -0.72232
  4: a -0.80828
  5: a -0.14135
  6: a  2.25701
  7: a -2.37955
  8: a -0.45425
  9: a -0.06007
 10: a  0.86090
 11: a -1.78466
 12: a -0.13074
 13: a -0.36984
 14: a -0.18066
 15: a -1.04973
```

Joins

```
DT1 <- data.table(x=c('a', 'a', 'b', 'dt1'), y=1:4)
DT2 <- data.table(x=c('a', 'b', 'dt2'), z=5:7)
setkey(DT1, x); setkey(DT2, x)
merge(DT1, DT2)
```

```
   x y z
1: a 1 5
2: a 2 5
3: b 3 6
```

Fast reading

```
big_df <- data.frame(x=rnorm(1E6), y=rnorm(1E6))
file <- tempfile()
write.table(big_df, file=file, row.names=FALSE, col.names=TRUE, sep="\t", quote=FALSE)
system.time(fread(file))
```

```
   user  system elapsed
  0.312   0.015   0.326
```

```
system.time(read.table(file, header=TRUE, sep="\t"))
```

```
   user  system elapsed
  5.702   0.048   5.755
```

Summary and further reading

- The latest development version contains new functions like `melt` and `dcast` for data.tables
  - https://r-forge.r-project.org/scm/viewvc.php/pkg/NEWS?view=markup&root=datatable
- Here is a list of differences between data.table and data.frame
  - http://stackoverflow.com/questions/13618488/what-you-can-do-with-data-frame-that-you-cant-in-data-table
- Notes based on Raphael Gottardo's notes https://github.com/raphg/Biostat-578/blob/master/Advanced_data_manipulation.Rpres, who got them from Kevin Ushey.