Webscraping in R - Final

a short primer

Mark Dayton

February 27, 2018

What is it? (webscraping using R)

Webscraping in R is basically....

- using various R package/functions to make retrieval and parsing of "web data" (fairly) easy

What is it? (webscraping using R)

Webscraping in R is basically....

- using various R package/functions to make retrieval and parsing of "web data" (fairly) easy
- caveats: whether the given web page is **"nicely formatted"** or not...
  as well as the underlying construction of the web page (site)

What is it? (webscraping using R)

Webscraping in R is basically....

- using various R package/functions to make retrieval and parsing of "web data" (fairly) easy
- caveats: whether the given web page is **"nicely formatted"** or not...
  as well as the underlying construction of the web page (site)
  often times *(but not always)* determines the level of effort required to do the job...

What is it? (webscraping using R)

Webscraping in R is basically....

- using various R package/functions to make retrieval and parsing of "web data" (fairly) easy

- caveats: whether the given web page is **"nicely formatted"** or not...
  as well as the underlying construction of the web page (site)
  often times *(but not always)* determines the level of effort required to do the job...

- HMTL/CSS complexity? HTML tables rendered via javascript? deciphering XML hierarchy? Password protected site?....
  - all can determine which packages/functions to utilize in your effort and the overall approach to take

Ok, so let's see it work...

by way of *illustration*, how about this....

- let's use parsed data from *one website* as a means to gather data from *another* website.....

Ok, so let's see it work...

by way of *illustration*, how about this....

- let's use parsed data from *one website* as a means to gather data from *another* website..... along the way we'll make use of the *rvest* package

Ok, so let's see it work...

by way of *illustration*, how about this....

- let's use parsed data from *one website* as a means to gather data from *another* website..... along the way we'll make use of the *rvest* package
  ...as well as selectorgadget

Ok, so let's see it work...

by way of *illustration*, how about this....

- let's use parsed data from *one website* as a means to gather data from *another* website..... along the way we'll make use of the *rvest* package
  ...as well as selectorgadget
  ...and the *right-click-inspect-element* technique of deciphering the correct css (or xpath)

Ok, so let's see it work...

by way of *illustration*, how about this....

- let's use parsed data from *one website* as a means to gather data from *another* website..... along the way we'll make use of the *rvest* package
  ...as well as selectorgadget
  ...and the *right-click-inspect-element* technique of deciphering the correct css (or xpath)
  - so we can *figure out* the 'correct' options to pass to rvest so it can *do its thing*

Ok, so let's see it work...

by way of *illustration*, how about this....

- let's use parsed data from *one website* as a means to gather data from *another* website..... along the way we'll make use of the *rvest* package
  ...as well as selectorgadget
  ...and the *right-click-inspect-element* technique of deciphering the correct css (or xpath)
  - so we can *figure out* the 'correct' options to pass to rvest so it can *do its thing*
  - after that, we'll go through dealing with XML-based webdata (using the *XML2R* package)

Ok, so let's see it work...

by way of *illustration*, how about this....

- let's use parsed data from *one website* as a means to gather data from *another* website..... along the way we'll make use of the *rvest* package
  ...as well as selectorgadget
  ...and the *right-click-inspect-element* technique of deciphering the correct css (or xpath)
  - so we can *figure out* the 'correct' options to pass to rvest so it can *do its thing*
  - after that, we'll go through dealing with XML-based webdata (using the *XML2R* package)
  - and finish up with using *rdom* or *RSelenium* to parse data from sites that use javascript to dynamically create website content (rvest can't help here...)

But first, some prerequisites for parsing HTML

things to study up on as you learn this topic

- Suggest to get comfortable with 1 of these 2 "utilities" (or rather *techniques*)....

But first, some prerequisites for parsing HTML

things to study up on as you learn this topic

- Suggest to get comfortable with 1 of these 2 **"utilities"** (or rather *techniques*)....
    - using selectorgadget

But first, some prerequisites for parsing HTML

things to study up on as you learn this topic

- Suggest to get comfortable with 1 of these 2 **"utilities"** (or rather *techniques*)....
  - using selectorgadget
  - using (my preference) **chrome developer tools** (i.e. *right-click-inspect-element*)

But first, some prerequisites for parsing HTML

things to study up on as you learn this topic

- Suggest to get comfortable with 1 of these 2 **"utilities"** (or rather *techniques*)....
  - using selectorgadget
  - using (my preference) **chrome developer tools** (i.e. *right-click-inspect-element*)

- Additionally, here's a link to a *css-for-dummies* interactive game for getting over the hump

But first, some prerequisites for parsing HTML

things to study up on as you learn this topic

- Suggest to get comfortable with 1 of these 2 **"utilities"** (or rather *techniques*)....
  - using selectorgadget
  - using (my preference) **chrome developer tools** (i.e. *right-click-inspect-element*)

- Additionally, here's a link to a *css-for-dummies* interactive game for getting over the hump
  - going though this will greatly ease anxiety when navigating webpages with complex HTML/CSS structures...

But first, some prerequisites for parsing HTML

things to study up on as you learn this topic

- Suggest to get comfortable with 1 of these 2 **"utilities"** (or rather *techniques*)....
  - using selectorgadget
  - using (my preference) **chrome developer tools** (i.e. *right-click-inspect-element*)

- Additionally, here's a link to a *css-for-dummies* interactive game for getting over the hump
  - going though this will greatly ease anxiety when navigating webpages with complex HTML/CSS structures...
  - And, a link to an interactive doc on the selectr::css_to_xpath function...(likely to come in handy at some point)

But first, some prerequisites for parsing HTML

things to study up on as you learn this topic

- Suggest to get comfortable with 1 of these 2 **"utilities"** (or rather *techniques*)....
  - using selectorgadget
  - using (my preference) **chrome developer tools** (i.e. *right-click-inspect-element*)

- Additionally, here's a link to a *css-for-dummies* interactive game for getting over the hump
  - going though this will greatly ease anxiety when navigating webpages with complex HTML/CSS structures...
  - And, a link to an interactive doc on the selectr::css_to_xpath function...(likely to come in handy at some point)

    Just a ***little*** bit of knowledge on deciphering CSS (and later, xpath hierarchies for XML-based data) will go a **looooong** ways!

ok - enough now - let's parse some (html) webdata

Example 1:

- from a webpage listing all R User Groups (*website #1*), grab the groups located in the U.S., and filter on the (fairly large) subset that use meetup.com (*website #2*) as their organizational hub.....so we can gather the following attributes:

ok - enough now - let's parse some (html) webdata

Example 1:

- from a webpage listing all R User Groups (*website #1*), grab the groups located in the U.S., and filter on the (fairly large) subset that use meetup.com (*website #2*) as their organizational hub.....so we can gather the following attributes:
    - location

ok - enough now - let's parse some (html) webdata

Example 1:

- from a webpage listing all R User Groups (*website #1*), grab the groups located in the U.S., and filter on the (fairly large) subset that use meetup.com (*website #2*) as their organizational hub.....so we can gather the following attributes:
    - location
    - inception date

ok - enough now - let's parse some (html) webdata

Example 1:

- from a webpage listing all R User Groups (*website #1*), grab the groups located in the U.S., and filter on the (fairly large) subset that use meetup.com (*website #2*) as their organizational hub.....so we can gather the following attributes:
  - location
  - inception date
  - nbr of members

ok - enough now - let's parse some (html) webdata

Example 1:

- from a webpage listing all R User Groups (*website #1*), grab the groups located in the U.S., and filter on the (fairly large) subset that use meetup.com (*website #2*) as their organizational hub.....so we can gather the following attributes:
  - location
  - inception date
  - nbr of members
  - nbr of meetings since inception

ok - enough now - let's parse some (html) webdata

Example 1:

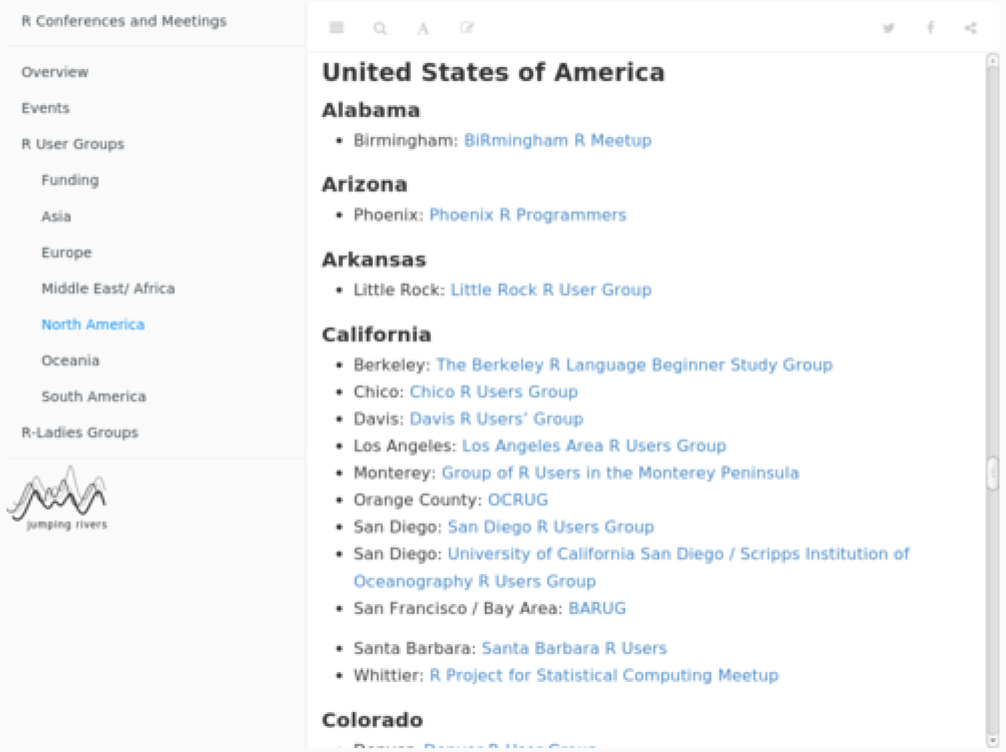
- from a webpage listing all R User Groups (*website #1*), grab the groups located in the U.S., and filter on the (fairly large) subset that use meetup.com (*website #2*) as their organizational hub.....so we can gather the following attributes:

    - location
    - inception date
    - nbr of members
    - nbr of meetings since inception

    NOTE: the 2nd website will be parsed as many times as there are user groups that use meetup.com...not just once!

Example 1 - Website 1: R User Groups (in U.S.) parsed one time

https://jumpingrivers.github.io/meetingsR/r-user-groups.html#united-states-of-america


R Conferences and Meetings

Overview

Events

R User Groups

Funding

Asia

Europe

Middle East/ Africa

North America

Oceania

South America

R-Ladies Groups

United States of America

Alabama

- Birmingham: BiRmingham R Meetup

Arizona

- Phoenix: Phoenix R Programmers

Arkansas

- Little Rock: Little Rock R User Group

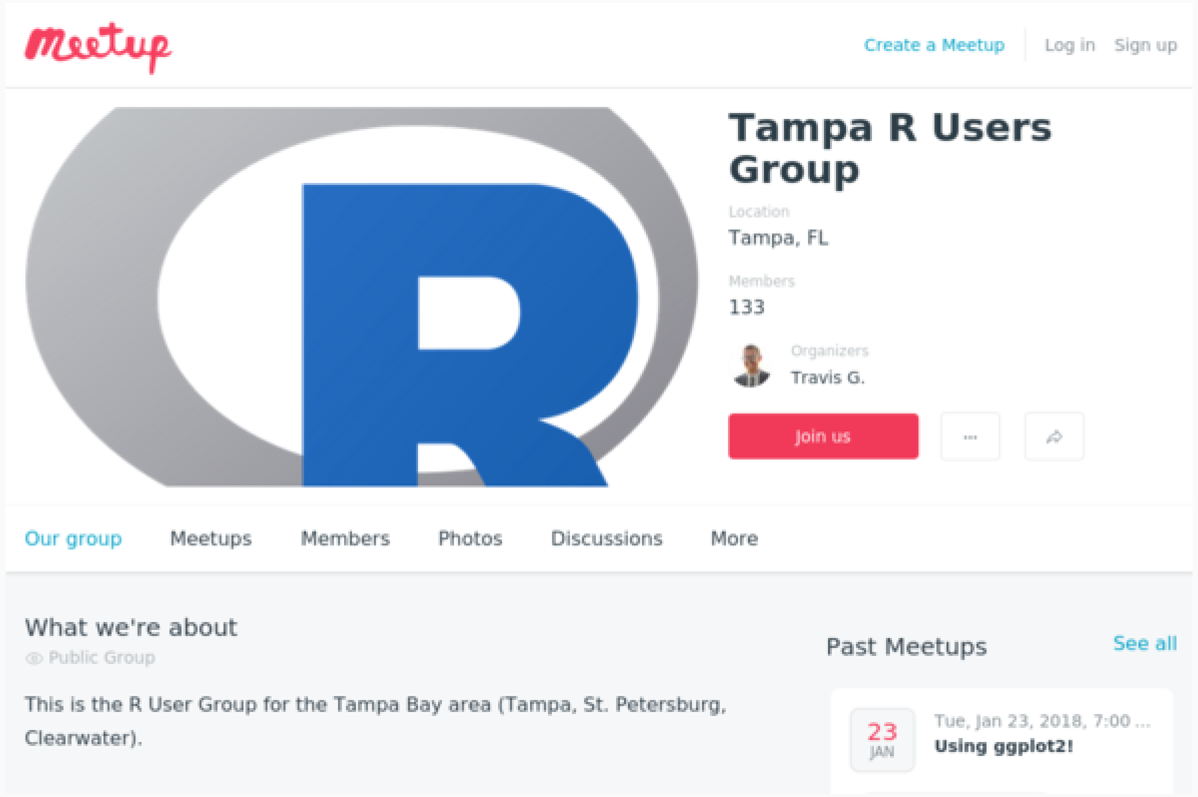
California

- Berkeley: The Berkeley R Language Beginner Study Group
- Chico: Chico R Users Group
- Davis: Davis R Users' Group
- Los Angeles: Los Angeles Area R Users Group
- Monterey: Group of R Users in the Monterey Peninsula
- Orange County: OCRUG
- San Diego: San Diego R Users Group
- San Diego: University of California San Diego / Scripps Institution of Oceanography R Users Group
- San Francisco / Bay Area: BARUG
- Santa Barbara: Santa Barbara R Users
- Whittier: R Project for Statistical Computing Meetup

Colorado

Example 1 - Website 2: meetup.com (parsed multiple times)

https://www.meetup.com/Tampa-R-Users-Group/



Tampa R Users Group

Location
Tampa, FL

Members
133

Organizers
Travis G.

What we're about

Public Group

This is the R User Group for the Tampa Bay area (Tampa, St. Petersburg, Clearwater).

Past Meetups

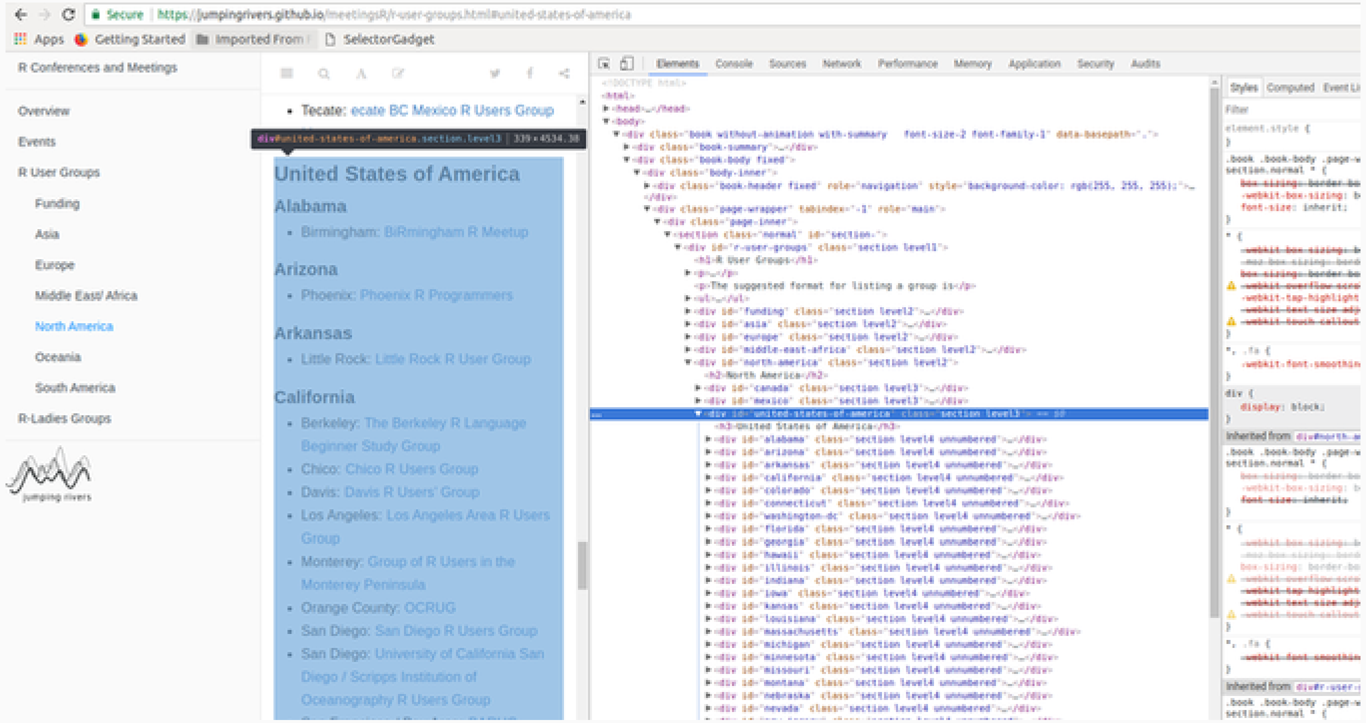
23 JAN Tue, Jan 23, 2018, 7:00 ...
Using ggplot2!

right-click-inspect-element (website #1)

In chrome, *right-click-inspect-element*, and mouse-over the list of R user groups highlighted (in blue!).

The CSS to pass to rvest::html_nodes() is in the **black retangular box**

div#united-states-of-america.section.level3



https://jumpingrivers.github.io/meetingsR/r-user-groups.html#united-states-of-america

R Conferences and Meetings

- Overview
- Events
- R User Groups
  - Funding
  - Asia
  - Europe
  - Middle East/ Africa
  - North America
  - Oceania
  - South America
- R-Ladies Groups

- Tecate: ecate BC Mexico R Users Group

div#united-states-of-america.section.level3 | 339×4534.38

United States of America

Alabama

- Birmingham: BiRmingham R Meetup

Arizona

- Phoenix: Phoenix R Programmers

Arkansas

- Little Rock: Little Rock R User Group

California

- Berkeley: The Berkeley R Language Beginner Study Group
- Chico: Chico R Users Group
- Davis: Davis R Users' Group
- Los Angeles: Los Angeles Area R Users Group
- Monterey: Group of R Users in the Monterey Peninsula
- Orange County: OCRUG
- San Diego: San Diego R Users Group
- San Diego: University of California San Diego / Scripps Institution of Oceanography R Users Group

```
<!DOCTYPE html>
<html>
▶<head>…</head>
▼<body>
  ▼<div class="book without-animation with-summary  font-size-2 font-family-1" data-basepath=".">
    ▶<div class="book-summary">…</div>
    ▼<div class="book-body fixed">
      ▼<div class="body-inner">
        ▶<div class="book-header fixed" role="navigation" style="background-color: rgb(255, 255, 255);">…
        </div>
        ▼<div class="page-wrapper" tabindex="-1" role="main">
          ▼<div class="page-inner">
            ▼<section class="normal" id="section-">
              ▼<div id="r-user-groups" class="section level1">
                <h1>R User Groups</h1>
                ▶<p>…</p>
                <p>The suggested format for listing a group is</p>
                ▶<ul>…</ul>
                ▶<div id="funding" class="section level2">…</div>
                ▶<div id="asia" class="section level2">…</div>
                ▶<div id="europe" class="section level2">…</div>
                ▶<div id="middle-east-africa" class="section level2">…</div>
                ▼<div id="north-america" class="section level2">
                  <h2>North America</h2>
                  ▶<div id="canada" class="section level3">…</div>
                  ▶<div id="mexico" class="section level3">…</div>
...               ▼<div id="united-states-of-america" class="section level3"> == $0
                    <h3>United States of America</h3>
                    ▶<div id="alabama" class="section level4 unnumbered">…</div>
                    ▶<div id="arizona" class="section level4 unnumbered">…</div>
                    ▶<div id="arkansas" class="section level4 unnumbered">…</div>
                    ▶<div id="california" class="section level4 unnumbered">…</div>
                    ▶<div id="colorado" class="section level4 unnumbered">…</div>
                    ▶<div id="connecticut" class="section level4 unnumbered">…</div>
                    ▶<div id="washington-dc" class="section level4 unnumbered">…</div>
                    ▶<div id="florida" class="section level4 unnumbered">…</div>
                    ▶<div id="georgia" class="section level4 unnumbered">…</div>
                    ▶<div id="hawaii" class="section level4 unnumbered">…</div>
                    ▶<div id="illinois" class="section level4 unnumbered">…</div>
                    ▶<div id="indiana" class="section level4 unnumbered">…</div>
                    ▶<div id="iowa" class="section level4 unnumbered">…</div>
                    ▶<div id="kansas" class="section level4 unnumbered">…</div>
                    ▶<div id="louisiana" class="section level4 unnumbered">…</div>
                    ▶<div id="massachusetts" class="section level4 unnumbered">…</div>
                    ▶<div id="michigan" class="section level4 unnumbered">…</div>
                    ▶<div id="minnesota" class="section level4 unnumbered">…</div>
                    ▶<div id="missouri" class="section level4 unnumbered">…</div>
                    ▶<div id="montana" class="section level4 unnumbered">…</div>
                    ▶<div id="nebraska" class="section level4 unnumbered">…</div>
                    ▶<div id="nevada" class="section level4 unnumbered">…</div>
```

scrape URLs via css (& xpath)

Now that we have the css, we can use it in the **rvest::html_nodes()** method... as well as pass it to the **selectr::css_to_xpath** method (as a way to show the results will be the same when using **css** or **xpath** parms in the call to **rvest::html_nodes()**

scrape URLs via css (& xpath)

Now that we have the css, we can use it in the **rvest::html_nodes()** method… as well as pass it to the **selectr::css_to_xpath** method (as a way to show the results will be the same when using **css** or **xpath** parms in the call to **rvest::html_nodes()**

Remember, for website #1, we will scrape just URLs within the U.S.

```r
library(rvest)
library(selectr)
url1 <- "https://jumpingrivers.github.io/meetingsR/r-user-groups.html#united-states-of-america"

us_rugs <- read_html(url1)                                # read url1 to create object for US R User Grps

css1a <- "div#united-states-of-america.section.level3"     # from "right-click-inspect"" method

xp1a  <- css_to_xpath(selector=css1a)                     # get xpath equivalent for css (to show all.equal==TRUE)

rugs_urls1a <- html_nodes(us_rugs,css=css1a) %>%
              html_nodes(.,"a") %>%
               html_attr(.,"href")

rugs_urlsxp1a <- html_nodes(us_rugs,xpath=xp1a) %>% # use xpath this time ... "a" --> denotes nodes w/a URL (with "href" attribute)
               html_nodes(.,"a") %>%
               html_attr(.,"href")

all.equal(rugs_urls1a,rugs_urlsxp1a)
```

```
## [1] TRUE
```

filter & clean URLs before passing to website #2

Not **all** of the 83 URLs use meetup.com (but most do), and we'll clean them up **before the next step**:

```
mu_urls ← rugs_urls1a[grep("meetup.com",rugs_urls1a)] %>%
          gsub("\\d+{4}|events/|members|files/","",.)
length(mu_urls)
```

```
## [1] 64
```

```
mu_urls[1:5]     # 64 out of 83 R User Groups in U.S. use meetup.com
```

```
## [1] "https://www.meetup.com/Bi-R-mingham-R-Meetup/"
## [2] "https://www.meetup.com/Central-Arkansas-R-User-Group/"
## [3] "https://www.meetup.com/r-enthusiasts/"
## [4] "https://www.meetup.com/Chico-R-Users-Group/"
## [5] "https://www.meetup.com/LAarea-R-usergroup/"
```

- OK...for the R-User Groups using meetup.com, *we've got their urls*

filter & clean URLs before passing to website #2

Not **all** of the 83 URLs use meetup.com (but most do), and we'll clean them up **before the next step**:

```
mu_urls <- rugs_urls1a[grep("meetup.com",rugs_urls1a)] %>%
          gsub("\\d+{4}|events/|members|files/","",.)
length(mu_urls)
```

```
## [1] 64
```

```
mu_urls[1:5]     # 64 out of 83 R User Groups in U.S. use meetup.com
```

```
## [1] "https://www.meetup.com/Bi-R-mingham-R-Meetup/"
## [2] "https://www.meetup.com/Central-Arkansas-R-User-Group/"
## [3] "https://www.meetup.com/r-enthusiasts/"
## [4] "https://www.meetup.com/Chico-R-Users-Group/"
## [5] "https://www.meetup.com/LAarea-R-usergroup/"
```

- OK...for the R-User Groups using meetup.com, *we've got their urls*
- and since **meetup.com** is a "well-formatted" website (i.e. consistent in HTML/CSS structure)....

filter & clean URLs before passing to website #2

Not **all** of the 83 URLs use meetup.com (but most do), and we'll clean them up **before the next step**:

```
mu_urls <- rugs_urls1a[grep("meetup.com",rugs_urls1a)] %>%
          gsub("\\d+{4}|events/|members|files/","",.)
length(mu_urls)
```

```
## [1] 64
```

```
mu_urls[1:5]     # 64 out of 83 R User Groups in U.S. use meetup.com
```

```
## [1] "https://www.meetup.com/Bi-R-mingham-R-Meetup/"
## [2] "https://www.meetup.com/Central-Arkansas-R-User-Group/"
## [3] "https://www.meetup.com/r-enthusiasts/"
## [4] "https://www.meetup.com/Chico-R-Users-Group/"
## [5] "https://www.meetup.com/LAarea-R-usergroup/"
```

- OK...for the R-User Groups using meetup.com, *we've got their urls*
- and since **meetup.com** is a "well-formatted" website (i.e. consistent in HTML/CSS structure)....
- it should be quick work to get the CSS (xpath) info we need to parse it

right-click-inspect-element (website #2)

- first, we're going to grab the info we need from the "members" page for each group:

right-click-inspect-element (website #2)

- first, we're going to grab the info we need from the **"members"** page for each group:

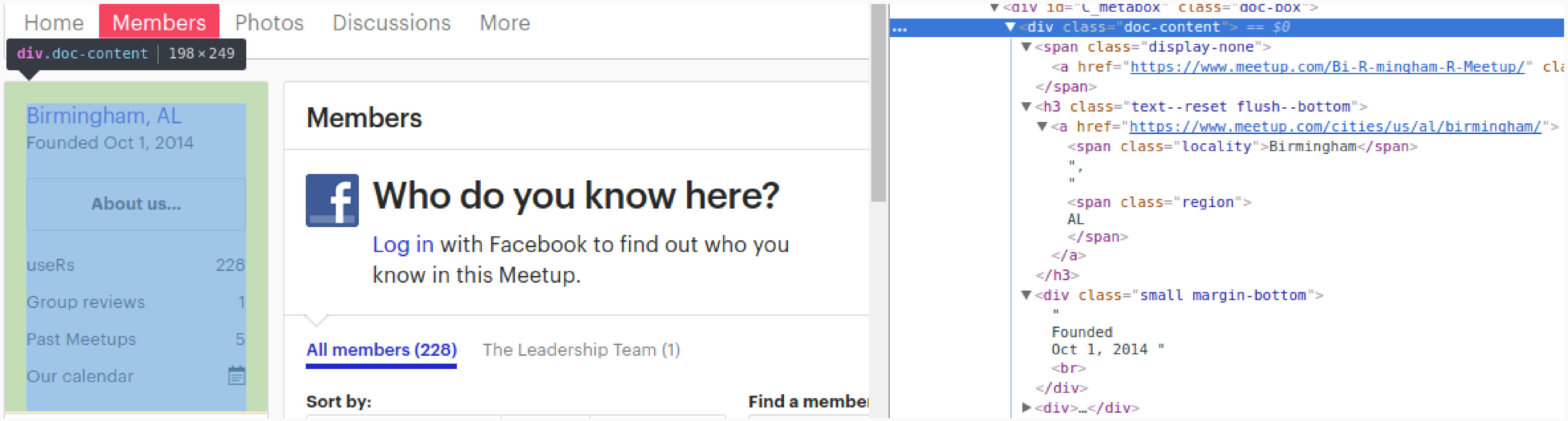
```r
# open up one of the R user group URLs from RStudio
browseURL(paste0(mu_urls[1],'/members/'))
```

right-click-inspect-element (website #2)

- first, we're going to grab the info we need from the "members" page for each group:

```r
# open up one of the R user group URLs from RStudio
browseURL(paste0(mu_urls[1],'/members/'))
```

  then, use selectorgadget and/or *right-click-inspect-method* to grab xpath to blue (or green) shaded box



Home Members Photos Discussions More

div.doc-content | 198 × 249

Birmingham, AL
Founded Oct 1, 2014

About us...

useRs 228
Group reviews 1
Past Meetups 5
Our calendar

Members

Who do you know here?

Log in with Facebook to find out who you know in this Meetup.

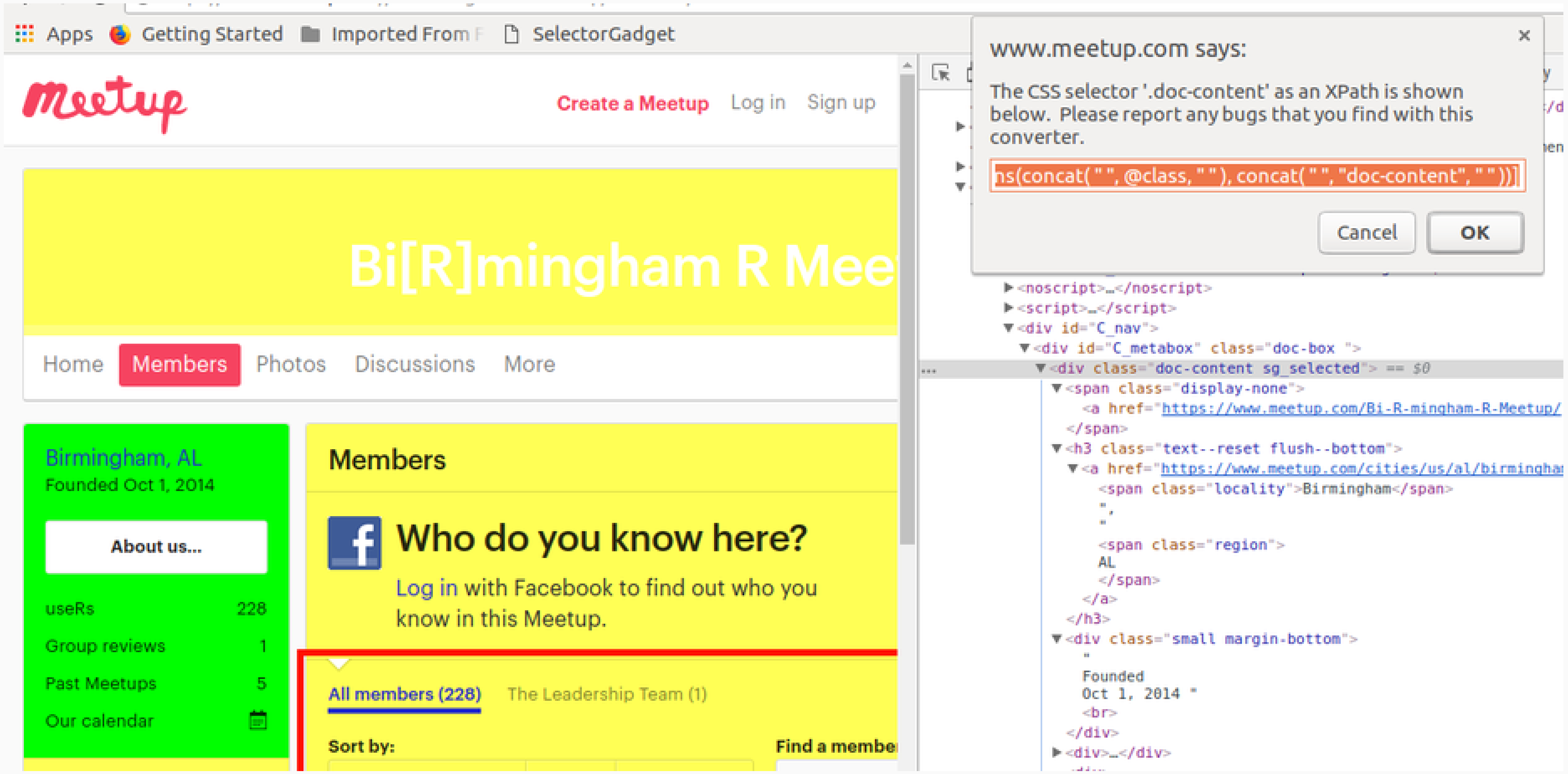
All members (228) The Leadership Team (1)

Sort by: Find a membe

```
▼<div id="C_metabox" class="doc-box">
... ▼<div class="doc-content"> == $0
  ▼<span class="display-none">
     <a href="https://www.meetup.com/Bi-R-mingham-R-Meetup/" cla
   </span>
  ▼<h3 class="text--reset flush--bottom">
   ▼<a href="https://www.meetup.com/cities/us/al/birmingham/">
       <span class="locality">Birmingham</span>
       ",
       "
       <span class="region">
       AL
       </span>
     </a>
   </h3>
  ▼<div class="small margin-bottom">
     "
     Founded
     Oct 1, 2014 "
     <br>
   </div>
  ▶<div>…</div>
```

selectorgadget method (website #2)

The gadget info box has the xpath to use (click xpath when green area is correctly set)



www.meetup.com says:

The CSS selector '.doc-content' as an XPath is shown below. Please report any bugs that you find with this converter.

```
ns(concat(" ", @class, " "), concat(" ", "doc-content", " "))]
```

```
▶<noscript>…</noscript>
▶<script>…</script>
▼<div id="C_nav">
  ▼<div id="C_metabox" class="doc-box ">
... ▼<div class="doc-content sg_selected"> == $0
      ▼<span class="display-none">
         <a href="https://www.meetup.com/Bi-R-mingham-R-Meetup/
       </span>
      ▼<h3 class="text--reset flush--bottom">
        ▼<a href="https://www.meetup.com/cities/us/al/birmingha
           <span class="locality">Birmingham</span>
           ",
           "
           <span class="region">
           AL
           </span>
         </a>
       </h3>
      ▼<div class="small margin-bottom">
         "
         Founded
         Oct 1, 2014 "
         <br>
       </div>
      ▶<div>…</div>
```

scrape meetup.com/meetings for all R User Groups

```
#xpath for to obtain group's attributes
```

```
#from the meetup.com/members page
```

```
mu_htm ← read_html(paste0(mu_urls[1],'/members/'))
```

```
mu_infoxp ← '//*[@id="C_metabox"]/div[1]'
```

```
mu_info ← html_nodes(mu_htm,xpath=mu_infoxp)
```

- We've got the info - now just parse it for....

scrape meetup.com/meetings for all R User Groups

```
#xpath for to obtain group's attributes
```

```
#from the meetup.com/members page
```

```
mu_htm ← read_html(paste0(mu_urls[1],'/members/'))
```

```
mu_infoxp ← '//*[@id="C_metabox"]/div[1]'
```

```
mu_info ← html_nodes(mu_htm,xpath=mu_infoxp)
```

- We've got the info - now just parse it for....
  - location (city/state), founded-date, nbr-of-mems, nbr-of-past-mtgs

scrape meetup.com/meetings for all R User Groups

```
#xpath for to obtain group's attributes
```

```
#from the meetup.com/members page
```

```
mu_htm <- read_html(paste0(mu_urls[1],'/members/'))
```

```
mu_infoxp <- '//*[@id="C_metabox"]/div[1]'
```

```
mu_info <- html_nodes(mu_htm,xpath=mu_infoxp)
```

- We've got the info - now just parse it for....

  - location (city/state), founded-date, nbr-of-mems, nbr-of-past-mtgs

  ```
  ## founded on date has this CSS
  ```

  ```
  dob_css <- 'div.small.margin-bottom'
  ```

  ```
  mu_dob <- gsub("Founded\n","",html_text(html_nodes(mu_info,css=dob_css),trim=T))
  ```

scrape meetup.com/meetings for all R User Groups

- continuing the parsing setup....

scrape meetup.com/meetings for all R User Groups

- continuing the parsing setup….

```
# CSS for citi and state
```

```
mu_city ← html_text(html_nodes(mu_info,css=".locality"))
```

```
mu_st ← gsub("\\n","",html_text(html_nodes(mu_info,css=".region")))
```

```
# CSS for (text of) "data" elements
```

```
mu_datacss ← 'ul.paddedList.small.margin-bottom'
```

scrape meetup.com/meetings for all R User Groups

putting it together: run code for scraping & parsing the metrics....

```
all_grpinfo ← matrix(nrow=length(mu_urls), ncol=6)
colnames(all_grpinfo) ← c('mu_city', 'mu_st', 'mu_dob', 'mu_nbrmems', 'mu_pstmtgs', 'mu_urls')

mu_infoxp ← '//*[@id="C_metabox"]/div[1]'  # xpath for selecting group attributes from the meetup.com/members page for a given R user group URL
dob_css ← 'div.small.margin-bottom'        # CSS locator for founded date for group
mu_datacss ← 'ul.paddedList.small.margin-bottom' # CSS for (text of) data elements

n ← 5 # scraping 1st 5 urls - see later footnote on wrapping up scrape websites that may limit HTML requests
for (i in 1:n) {
mu_htm ← read_html(paste0(mu_urls[i],'/members/'))
mu_info ← html_nodes(mu_htm,xpath=mu_infoxp)
mu_city ← html_text(html_nodes(mu_info,css=".locality"))
mu_st ← html_text(html_nodes(mu_info,css=".region")) %>%
         gsub("\\n","",.)

mu_dob ← html_text(html_nodes(mu_info,css=dob_css),trim=T) %>%
          gsub("Founded\n","",.)

mu_data ← html_nodes(mu_info,css=mu_datacss) %>%
           html_nodes(.,css="a") %>%
           html_text(.)
mu_nbrmems ← strsplit(mu_data[1],"\n")[[1]][2]
mu_pstmtgs ← strsplit(mu_data[grep("Past Meet", mu_data)],"\n")[[1]][3]

all_grpinfo[i,] ← cbind(mu_city, mu_st, mu_dob, mu_nbrmems, mu_pstmtgs, mu_urls[i])
}
```

sample output

OK - what does it look like?.....

```
options(width = 150)
all_grpinfo[1:n,]
```

```
##      mu_city        mu_st mu_dob         mu_nbrmems mu_pstmtgs mu_urls
## [1,] "Birmingham"   "AL"  "Oct 1, 2014"  "228"      "5"        "https://www.meetup.com/Bi-R-mingham-R-Meetup/"
## [2,] "Little Rock"  "AR"  "Mar 4, 2017"  "79"       "3"        "https://www.meetup.com/Central-Arkansas-R-User-Group/"
## [3,] "Oakland"      "CA"  "Jan 27, 2012" "1,554"    "77"       "https://www.meetup.com/r-enthusiasts/"
## [4,] "Chico"        "CA"  "Dec 7, 2015"  "29"       "26"       "https://www.meetup.com/Chico-R-Users-Group/"
## [5,] "Santa Monica" "CA"  "Mar 21, 2009" "1,694"    "85"       "https://www.meetup.com/LAarea-R-usergroup/"
```

sample output

OK - what does it look like?.....

```
options(width = 150)
all_grpinfo[1:n,]
```

```
##      mu_city        mu_st mu_dob         mu_nbrmems mu_pstmtgs mu_urls
## [1,] "Birmingham"   "AL"  "Oct 1, 2014"  "228"      "5"        "https://www.meetup.com/Bi-R-mingham-R-Meetup/"
## [2,] "Little Rock"  "AR"  "Mar 4, 2017"  "79"       "3"        "https://www.meetup.com/Central-Arkansas-R-User-Group/"
## [3,] "Oakland"      "CA"  "Jan 27, 2012" "1,554"    "77"       "https://www.meetup.com/r-enthusiasts/"
## [4,] "Chico"        "CA"  "Dec 7, 2015"  "29"       "26"       "https://www.meetup.com/Chico-R-Users-Group/"
## [5,] "Santa Monica" "CA"  "Mar 21, 2009" "1,694"    "85"       "https://www.meetup.com/LAarea-R-usergroup/"
```

NOTE: [1]: we won't be scraping all websites here, but see footnote for example of throttling read_html() when websites may limit traffic

sample output

OK - what does it look like?.....

```
options(width = 150)
all_grpinfo[1:n,]
```

```
##      mu_city        mu_st mu_dob         mu_nbrmems mu_pstmtgs mu_urls
## [1,] "Birmingham"   "AL"  "Oct 1, 2014"  "228"      "5"        "https://www.meetup.com/Bi-R-mingham-R-Meetup/"
## [2,] "Little Rock"  "AR"  "Mar 4, 2017"  "79"       "3"        "https://www.meetup.com/Central-Arkansas-R-User-Group/"
## [3,] "Oakland"      "CA"  "Jan 27, 2012" "1,554"    "77"       "https://www.meetup.com/r-enthusiasts/"
## [4,] "Chico"        "CA"  "Dec 7, 2015"  "29"       "26"       "https://www.meetup.com/Chico-R-Users-Group/"
## [5,] "Santa Monica" "CA"  "Mar 21, 2009" "1,694"    "85"       "https://www.meetup.com/LAarea-R-usergroup/"
```

NOTE: [1]: we won't be scraping all websites here, but see footnote for example of throttling read_html() when websites may limit traffic

[1] https://stackoverflow.com/questions/39056103/iterating-rvest-scrape-function-gives-error-in-open-connectionx-rb-time/39057166/

some MOTO conclusions...

- playing the role of *master-of-the-obvious*, what can we conclude?
    - "if" one wanted to, one could create a nice dataset/database of R User Groups *just* by parsing data from meetup.com

some MOTO conclusions...

- playing the role of *master-of-the-obvious*, what can we conclude?
    - "if" one wanted to, one could create a nice dataset/database of R User Groups *just* by parsing data from meetup.com
    - with just a bit more effort, and one could get the 'joined date' for every member (and calculate a *bunch* of stuff)

some MOTO conclusions...

- playing the role of *master-of-the-obvious*, what can we conclude?
  - "if" one wanted to, one could create a nice dataset/database of R User Groups *just* by parsing data from meetup.com
  - with just a bit more effort, and one could get the 'joined date' for every member (and calculate a *bunch* of stuff)
  - one could parse the blurbs in the group organizers and analyze/segment based on textmining (e.g. *which industry is their focus?*)

some MOTO conclusions...

- playing the role of *master-of-the-obvious*, what can we conclude?
    - "if" one wanted to, one could create a nice dataset/database of R User Groups *just* by parsing data from meetup.com
    - with just a bit more effort, and one could get the 'joined date' for every member (and calculate a *bunch* of stuff)
    - one could parse the blurbs in the group organizers and analyze/segment based on textmining (e.g. *which industry is their focus?*)

some MOTO conclusions...

- or, one could parse meetings for the topic/subject matter, and *number of attendees*...etc etc etc

some MOTO conclusions...

- or, one could parse meetings for the **topic/subject** matter, and ***number of attendees***...etc etc etc
    - as a ***pedagogical device***, one could create an animated ***shiny app*** showing growth of R over time and geography (*West Coast to East Coast?*)

some MOTO conclusions...

- or, one could parse meetings for the topic/subject matter, and *number of attendees*...etc etc etc
  - as a *pedagogical device*, one could create an animated *shiny app* showing growth of R over time and geography (*West Coast to East Coast?*)
  - and on and on and on.....(what about expanding to non-U.S. groups?)

Summary take-away....

- learning to scrape data from the web allows one to *create* and *analyze* datasets that simply do not exist anywhere...

Summary take-away....

- learning to scrape data from the web allows one to *create* and *analyze* datasets that simply do not exist anywhere... ....**but in the rendering of webpages**

Summary take-away....

- learning to scrape data from the web allows one to *create* and *analyze* datasets that simply do not exist anywhere... ....**but in the rendering of webpages**

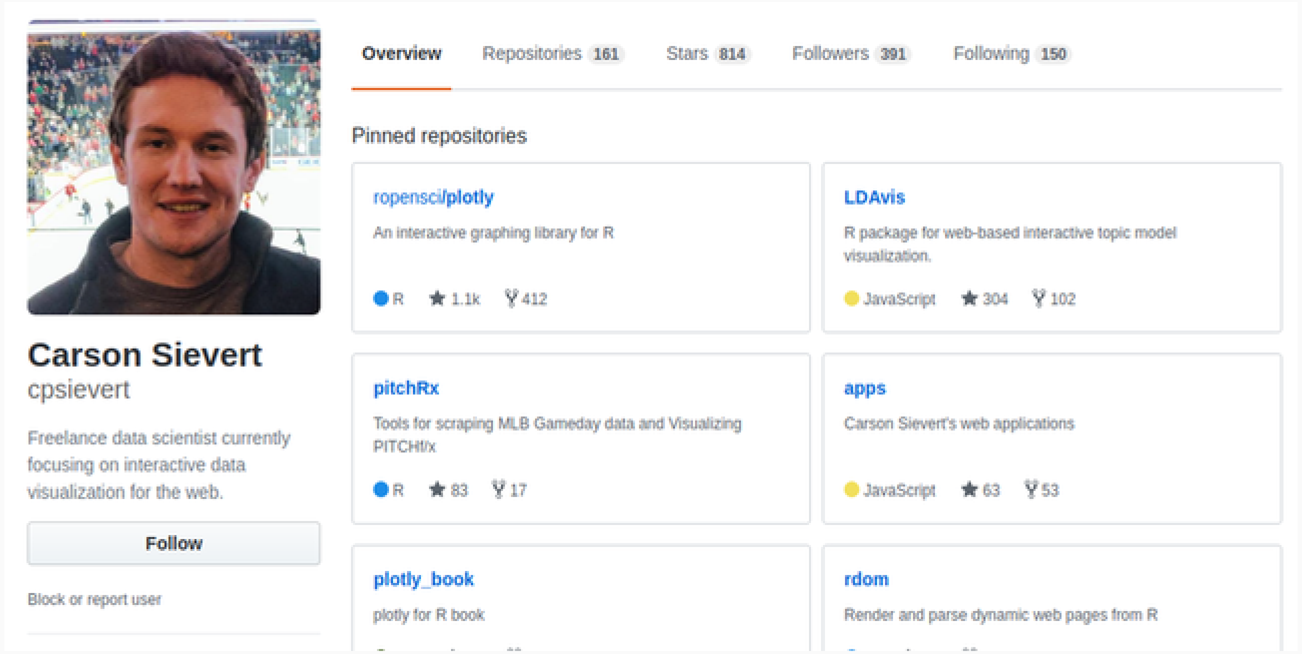
And now, let's move on to scraping some XML-formatted data

Scraping and Parsing XML

- *First*, let's mention one prolific contributor in the effort to **demystify** the scraping/manipulation of XML

Scraping and Parsing XML

- *First*, let's mention one prolific contributor in the effort to **demystify** the scraping/manipulation of XML
- Carson Seivert wrote the *pitchRx* package (for scraping MLB's "gameday" XML data)



Overview Repositories 161 Stars 814 Followers 391 Following 150

Carson Sievert
cpsievert

Freelance data scientist currently focusing on interactive data visualization for the web.

Follow

Block or report user

Pinned repositories

ropensci/plotly
An interactive graphing library for R
● R ★ 1.1k 412

LDAvis
R package for web-based interactive topic model visualization.
● JavaScript ★ 304 102

pitchRx
Tools for scraping MLB Gameday data and Visualizing PITCHf/x
● R ★ 83 17

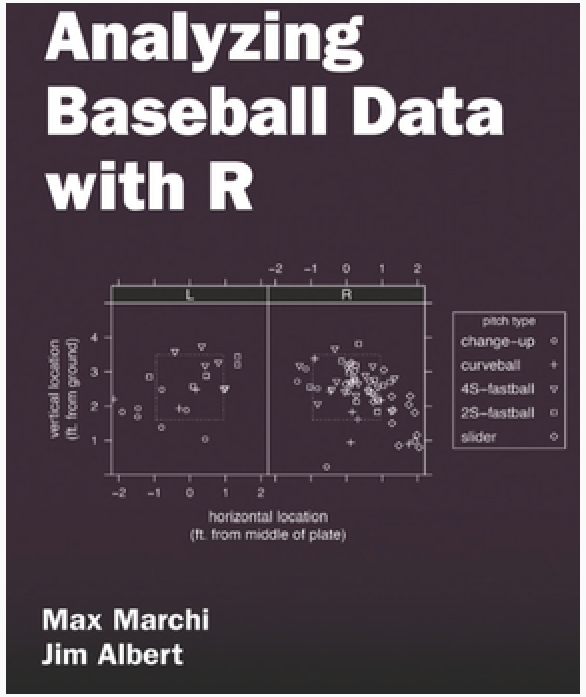
apps
Carson Sievert's web applications
● JavaScript ★ 63 53

plotly_book
plotly for R book

rdom
Render and parse dynamic web pages from R

Scraping and Parsing XML

- as referenced in the book *Analyzing Baseball Data in R*



Scraping and Parsing XML

- If there's one way to learn how to deal with XML, it's in putting together your own database of EVERY pitch thrown in EVERY MLB game over the past x years!

Scraping and Parsing XML

- If there's one way to learn how to deal with XML, it's in putting together your own database of EVERY pitch thrown in EVERY MLB game over the past x years!
- Since Carson has provided detailed examples of using XML2R[2] in the context of *baseball*,

Scraping and Parsing XML

- If there's one way to learn how to deal with XML, it's in putting together your own database of EVERY pitch thrown in EVERY MLB game over the past x years!
- Since Carson has provided detailed examples of using XML2R[2] in the context of *baseball*, let's do something related to a *different* sport....

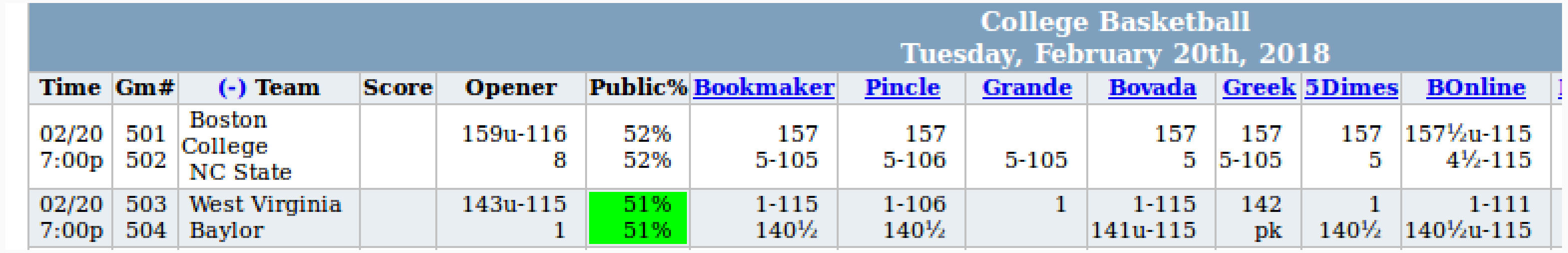
[2] https://xml2r.cpsievert.me/

Scraping and Parsing XML

In the spirit of ***March Madness***, let's grab some NCAA Basketball **"market"** data (i.e., Vegas Oddslines) from *wagertalk.com*

Scraping and Parsing XML

In the spirit of ***March Madness***, let's grab some NCAA Basketball **"market"** data (i.e., Vegas Oddslines) from *wagertalk.com*

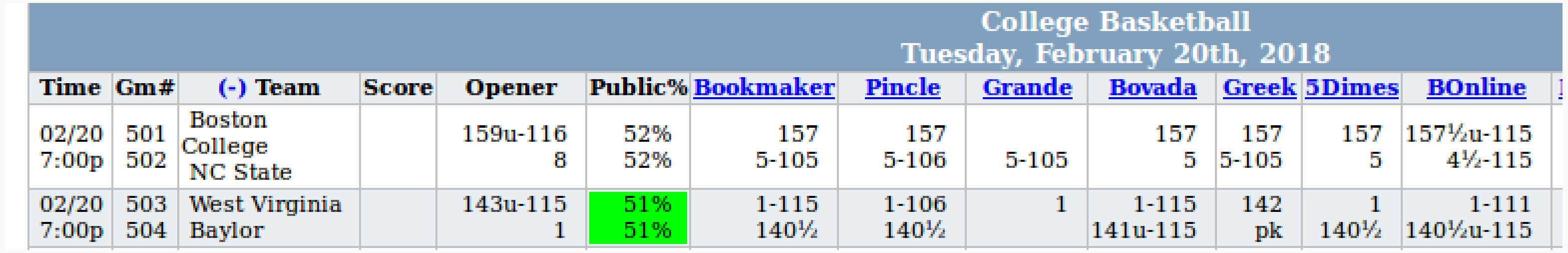


College Basketball
Tuesday, February 20th, 2018

Time	Gm#	(-) Team	Score	Opener	Public%	Bookmaker	Pincle	Grande	Bovada	Greek	5Dimes	BOnline
02/20 7:00p	501 502	Boston College NC State		159u-116 8	52% 52%	157 5-105	157 5-106	5-105	157 5	157 5-105	157 5	157½u-115 4½-115
02/20 7:00p	503 504	West Virginia Baylor		143u-115 1	51% 51%	1-115 140½	1-106 140½	1	1-115 141u-115	142 pk	1 140½	1-111 140½u-115

Scraping and Parsing XML

In the spirit of ***March Madness***, let's grab some NCAA Basketball **"market"** data (i.e., Vegas Oddslines) from *wagertalk.com*



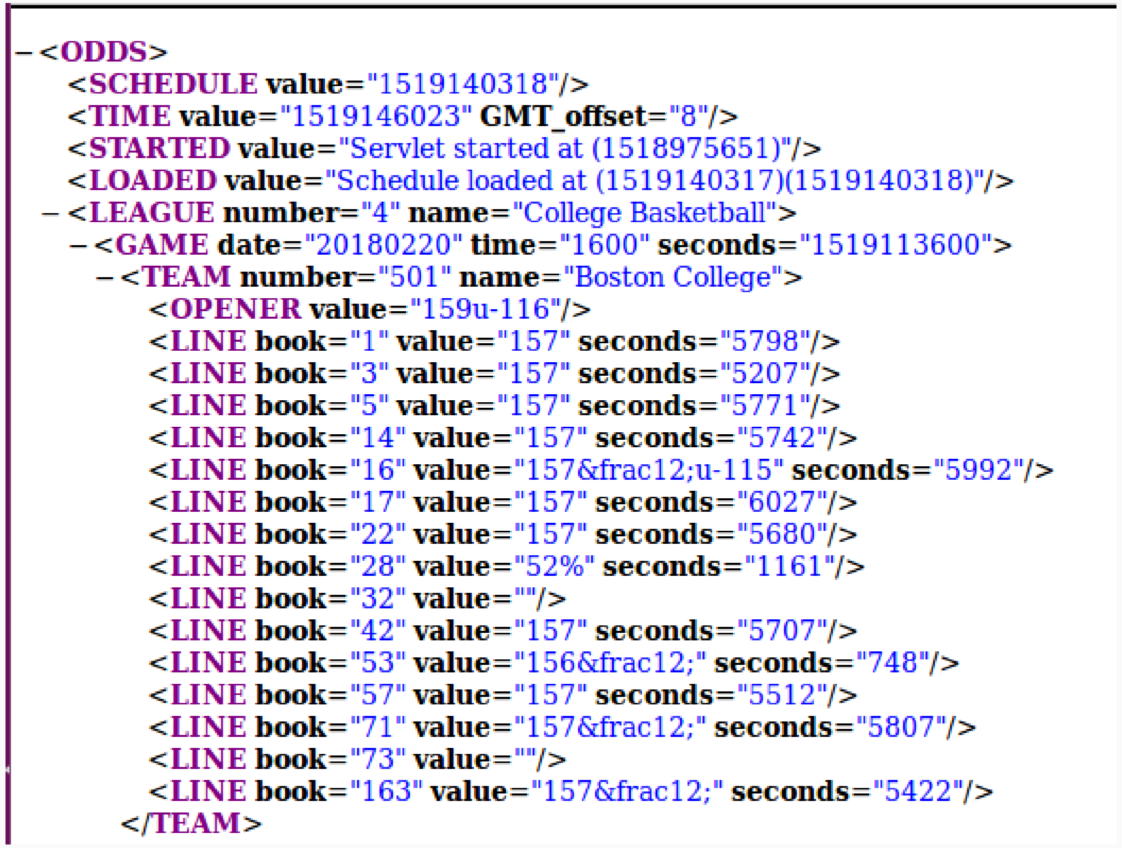
College Basketball Tuesday, February 20th, 2018												
Time	Gm#	(-) Team	Score	Opener	Public%	Bookmaker	Pincle	Grande	Bovada	Greek	5Dimes	BOnline
02/20 7:00p	501 502	Boston College NC State		159u-116 8	52% 52%	157 5-105	157 5-106	5-105	157 5	157 5-105	157 5	157½u-115 4½-115
02/20 7:00p	503 504	West Virginia Baylor		143u-115 1	51% 51%	1-115 140½	1-106 140½	1	1-115 141u-115	142 pk	1 140½	1-111 140½u-115

The above shows the odds/lines for NCAA BB as of around noon EST on Feb. 20th 2018....

Scraping and Parsing XML

Fortunately, that data is available in XML format here:

http://www.wagertalk.com/spt-opt/schedule.php?host=WAGERTALK&sport=ncaabb&period=0

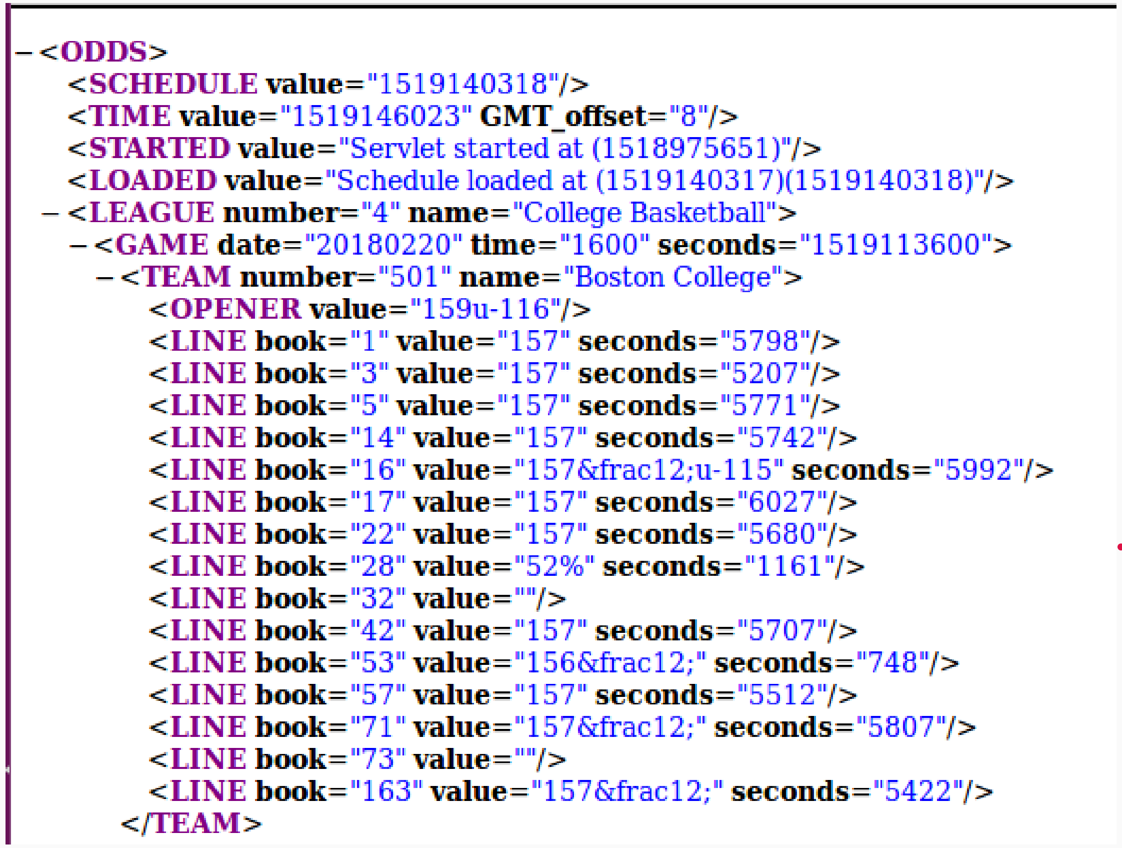


```
- <ODDS>
    <SCHEDULE value="1519140318"/>
    <TIME value="1519146023" GMT_offset="8"/>
    <STARTED value="Servlet started at (1518975651)"/>
    <LOADED value="Schedule loaded at (1519140317)(1519140318)"/>
  - <LEAGUE number="4" name="College Basketball">
    - <GAME date="20180220" time="1600" seconds="1519113600">
      - <TEAM number="501" name="Boston College">
          <OPENER value="159u-116"/>
          <LINE book="1" value="157" seconds="5798"/>
          <LINE book="3" value="157" seconds="5207"/>
          <LINE book="5" value="157" seconds="5771"/>
          <LINE book="14" value="157" seconds="5742"/>
          <LINE book="16" value="157&frac12;u-115" seconds="5992"/>
          <LINE book="17" value="157" seconds="6027"/>
          <LINE book="22" value="157" seconds="5680"/>
          <LINE book="28" value="52%" seconds="1161"/>
          <LINE book="32" value=""/>
          <LINE book="42" value="157" seconds="5707"/>
          <LINE book="53" value="156&frac12;" seconds="748"/>
          <LINE book="57" value="157" seconds="5512"/>
          <LINE book="71" value="157&frac12;" seconds="5807"/>
          <LINE book="73" value=""/>
          <LINE book="163" value="157&frac12;" seconds="5422"/>
        </TEAM>
```

Scraping and Parsing XML

Fortunately, that data is available in XML format here:

http://www.wagertalk.com/spt-opt/schedule.php?host=WAGERTALK&sport=ncaabb&period=0



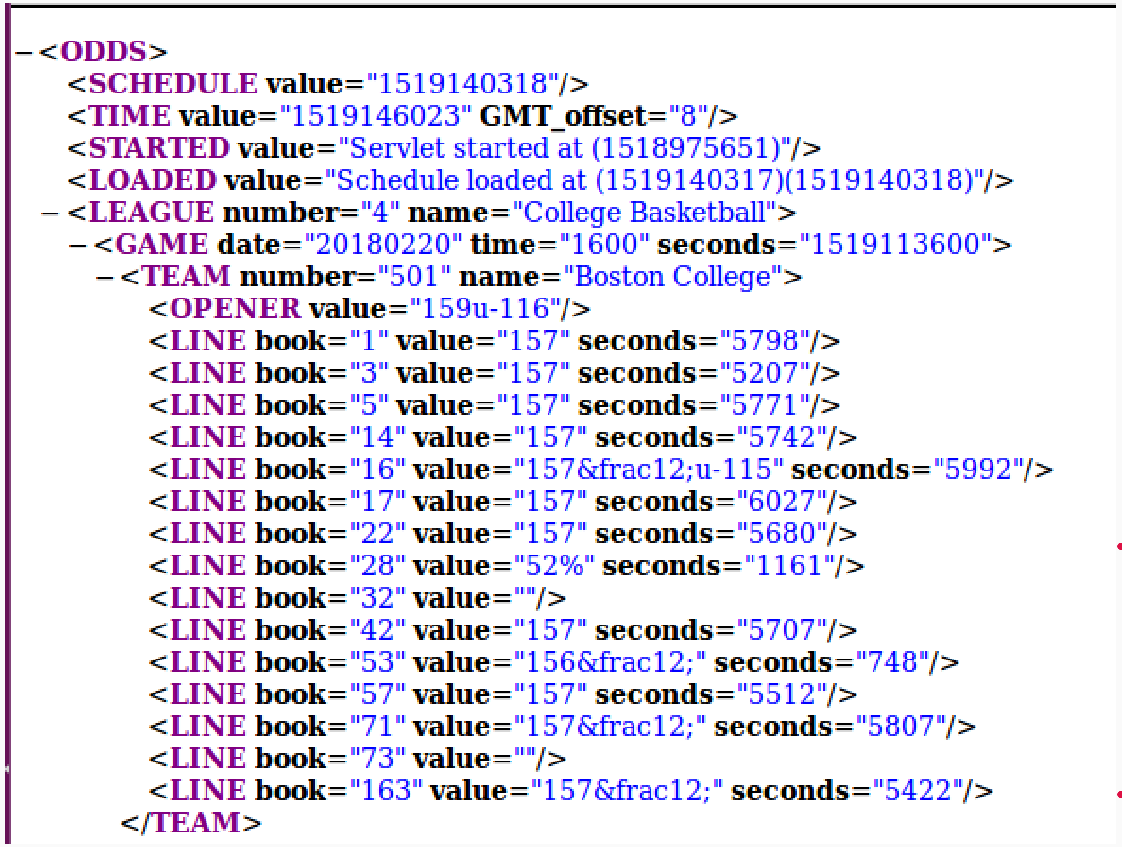
```
– <ODDS>
    <SCHEDULE value="1519140318"/>
    <TIME value="1519146023" GMT_offset="8"/>
    <STARTED value="Servlet started at (1518975651)"/>
    <LOADED value="Schedule loaded at (1519140317)(1519140318)"/>
  – <LEAGUE number="4" name="College Basketball">
    – <GAME date="20180220" time="1600" seconds="1519113600">
      – <TEAM number="501" name="Boston College">
          <OPENER value="159u-116"/>
          <LINE book="1" value="157" seconds="5798"/>
          <LINE book="3" value="157" seconds="5207"/>
          <LINE book="5" value="157" seconds="5771"/>
          <LINE book="14" value="157" seconds="5742"/>
          <LINE book="16" value="157&frac12;u-115" seconds="5992"/>
          <LINE book="17" value="157" seconds="6027"/>
          <LINE book="22" value="157" seconds="5680"/>
          <LINE book="28" value="52%" seconds="1161"/>
          <LINE book="32" value=""/>
          <LINE book="42" value="157" seconds="5707"/>
          <LINE book="53" value="156&frac12;" seconds="748"/>
          <LINE book="57" value="157" seconds="5512"/>
          <LINE book="71" value="157&frac12;" seconds="5807"/>
          <LINE book="73" value=""/>
          <LINE book="163" value="157&frac12;" seconds="5422"/>
        </TEAM>
```

.....now left's examine the XML hierarchy....

Scraping and Parsing XML

Fortunately, that data is available in XML format here:

http://www.wagertalk.com/spt-opt/schedule.php?host=WAGERTALK&sport=ncaabb&period=0



```
– <ODDS>
    <SCHEDULE value="1519140318"/>
    <TIME value="1519146023" GMT_offset="8"/>
    <STARTED value="Servlet started at (1518975651)"/>
    <LOADED value="Schedule loaded at (1519140317)(1519140318)"/>
  – <LEAGUE number="4" name="College Basketball">
    – <GAME date="20180220" time="1600" seconds="1519113600">
      – <TEAM number="501" name="Boston College">
          <OPENER value="159u-116"/>
          <LINE book="1" value="157" seconds="5798"/>
          <LINE book="3" value="157" seconds="5207"/>
          <LINE book="5" value="157" seconds="5771"/>
          <LINE book="14" value="157" seconds="5742"/>
          <LINE book="16" value="157&frac12;u-115" seconds="5992"/>
          <LINE book="17" value="157" seconds="6027"/>
          <LINE book="22" value="157" seconds="5680"/>
          <LINE book="28" value="52%" seconds="1161"/>
          <LINE book="32" value=""/>
          <LINE book="42" value="157" seconds="5707"/>
          <LINE book="53" value="156&frac12;" seconds="748"/>
          <LINE book="57" value="157" seconds="5512"/>
          <LINE book="71" value="157&frac12;" seconds="5807"/>
          <LINE book="73" value=""/>
          <LINE book="163" value="157&frac12;" seconds="5422"/>
        </TEAM>
```

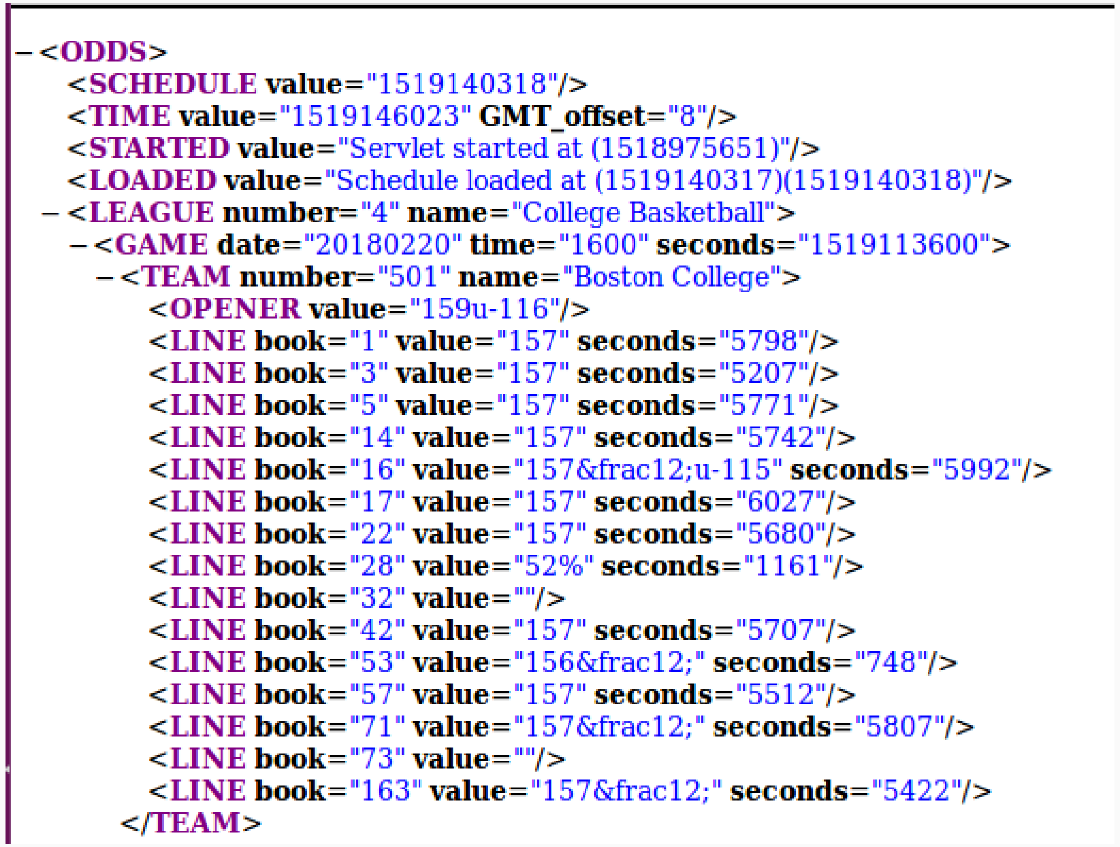
.....now left's examine the XML hierarchy....

.....before running some simple code....

Examining the XML structure

XML Hierarchy

1. open the file and example the structure
   - expand and collapse the XML hierarchy to geta feel for how the parent/child relationships are formatted
2. think of the XML in terms of *facts* and *dimensions*[3], and in terms of 'global' information.
   - in this case, the (my phrase) 'global' tags are SCHEDULE, TIME, STARTED, LOADED, LEAGUE
3. each GAME node has associated date/datetime elements, but also has 2 TEAM child nodes....
4. finally, TEAM node has set of LINE tags
   - LINE tags relate to the line's "provider" (bookmaker)
   - one set of LINEs relate to "point spread", and the other relates to game totals (OVER/UNDER)



```
- <ODDS>
    <SCHEDULE value="1519140318"/>
    <TIME value="1519146023" GMT_offset="8"/>
    <STARTED value="Servlet started at (1518975651)"/>
    <LOADED value="Schedule loaded at (1519140317)(1519140318)"/>
  - <LEAGUE number="4" name="College Basketball">
    - <GAME date="20180220" time="1600" seconds="1519113600">
      - <TEAM number="501" name="Boston College">
          <OPENER value="159u-116"/>
          <LINE book="1" value="157" seconds="5798"/>
          <LINE book="3" value="157" seconds="5207"/>
          <LINE book="5" value="157" seconds="5771"/>
          <LINE book="14" value="157" seconds="5742"/>
          <LINE book="16" value="157&frac12;u-115" seconds="5992"/>
          <LINE book="17" value="157" seconds="6027"/>
          <LINE book="22" value="157" seconds="5680"/>
          <LINE book="28" value="52%" seconds="1161"/>
          <LINE book="32" value=""/>
          <LINE book="42" value="157" seconds="5707"/>
          <LINE book="53" value="156&frac12;" seconds="748"/>
          <LINE book="57" value="157" seconds="5512"/>
          <LINE book="71" value="157&frac12;" seconds="5807"/>
          <LINE book="73" value=""/>
          <LINE book="163" value="157&frac12;" seconds="5422"/>
        </TEAM>
```

[3] https://en.wikipedia.org/wiki/Dimensional_modeling

Why XML2R?

Why XML2R?

XML2R makes scraping/parsing XML 'easy' because it....

Why XML2R?

XML2R makes scraping/parsing XML 'easy' because it....

- reduces the effort required to transform XML into tables

Why XML2R?

XML2R makes scraping/parsing XML 'easy' because it....

- reduces the effort required to transform XML into tables
- and does so while preserving parent to child relationships
    - (which is ***why*** we examined the XML structure as a **first step**)

Why XML2R?

XML2R makes scraping/parsing XML 'easy' because it....

- reduces the effort required to transform XML into tables
- and does so while preserving parent to child relationships
    - (which is ***why*** we examined the XML structure as a **first step**)
- for this exercise, we'll use just 3 functions:

Why XML2R?

XML2R makes scraping/parsing XML 'easy' because it....

- reduces the effort required to transform XML into tables
- and does so while preserving parent to child relationships
    - (which is *why* we examined the XML structure as a first step)
- for this exercise, we'll use just 3 functions:
    - XML2R::XML2Obs
    - XML2R::add_key
    - XML2R::collapse_obs

Parsing the XML via XML2R functions

let's use XML2R::XML2Obs against the oddsline URL and look at what it returns

Parsing the XML via XML2R functions

let's use XML2R::XML2Obs against the oddsline URL and look at what it returns

```r
library(XML2R)
xml1 <- "http://www.wagertalk.com/spt-opt/schedule.php?host=WAGERTALK&sport=ncaabb&period=0"
xmlobs1 <- XML2Obs(xml1, quiet=TRUE)
table(names(xmlobs1))
```

```
##
##                        ODDS//LEAGUE                   ODDS//LEAGUE//GAME             ODDS//LEAGUE//GAME//TEAM   ODDS//LEAGUE//GAME//TEAM//LINE
##                                   1                                   88                                  176                              656
## ODDS//LEAGUE//GAME//TEAM//OPENER                         ODDS//LOADED                       ODDS//SCHEDULE                    ODDS//STARTED
##                                 176                                    1                                    1                                1
##                          ODDS//TIME
##                                   1
```

Parsing the XML via XML2R functions

let's use XML2R::XML2Obs against the oddsline URL and look at what it returns

```
library(XML2R)
xml1 <- "http://www.wagertalk.com/spt-opt/schedule.php?host=WAGERTALK&sport=ncaabb&period=0"
xmlobs1 <- XML2Obs(xml1, quiet=TRUE)
table(names(xmlobs1))
```

```
##
##                      ODDS//LEAGUE                 ODDS//LEAGUE//GAME           ODDS//LEAGUE//GAME//TEAM   ODDS//LEAGUE//GAME//TEAM//LINE
##                                 1                                 88                                176                              656
## ODDS//LEAGUE//GAME//TEAM//OPENER                       ODDS//LOADED                     ODDS//SCHEDULE                    ODDS//STARTED
##                               176                                  1                                  1                                1
##                        ODDS//TIME
##                                 1
```

After having perused the XML file by collapsing/expanding nodes, the output *makes sense*....

- each GAME has **2** TEAMS

Parsing the XML via XML2R functions

let's use XML2R::XML2Obs against the oddsline URL and look at what it returns

```r
library(XML2R)
xml1 <- "http://www.wagertalk.com/spt-opt/schedule.php?host=WAGERTALK&sport=ncaabb&period=0"
xmlobs1 <- XML2Obs(xml1, quiet=TRUE)
table(names(xmlobs1))
```

```
##
##                        ODDS//LEAGUE                  ODDS//LEAGUE//GAME            ODDS//LEAGUE//GAME//TEAM   ODDS//LEAGUE//GAME//TEAM//LINE
##                                   1                                  88                                 176                              656
## ODDS//LEAGUE//GAME//TEAM//OPENER                        ODDS//LOADED                      ODDS//SCHEDULE                    ODDS//STARTED
##                                 176                                   1                                   1                                1
##                          ODDS//TIME
##                                   1
```

After having perused the XML file by collapsing/expanding nodes, the output *makes sense*....

- each GAME has **2** TEAMS
- the 'global' information is associated with a **single** count of a node ('time', 'loaded', etc)

Parsing the XML via XML2R functions

let's use XML2R::XML2Obs against the oddsline URL and look at what it returns

```
library(XML2R)
xml1 <- "http://www.wagertalk.com/spt-opt/schedule.php?host=WAGERTALK&sport=ncaabb&period=0"
xmlobs1 <- XML2Obs(xml1, quiet=TRUE)
table(names(xmlobs1))
```

```
##
##                    ODDS//LEAGUE                 ODDS//LEAGUE//GAME         ODDS//LEAGUE//GAME//TEAM   ODDS//LEAGUE//GAME//TEAM//LINE
##                               1                                 88                              176                              656
## ODDS//LEAGUE//GAME//TEAM//OPENER                       ODDS//LOADED                   ODDS//SCHEDULE                    ODDS//STARTED
##                             176                                  1                                1                                1
##                      ODDS//TIME
##                               1
```

After having perused the XML file by collapsing/expanding nodes, the output *makes sense*....

- each GAME has **2** TEAMS
- the 'global' information is associated with a **single** count of a node ('time', 'loaded', etc)
- now, we'll make sure the PARENT node data is passed to the CHILD nodes (before creating data.frame type output)

mapping parent-to-child (via XML2R::add_key)

Remember - the LINE nodes are the "facts", and we need to be able to map those facts to their parent attributes...

mapping parent-to-child (via XML2R::add_key)

Remember - the LINE nodes are the "facts", and we need to be able to map those facts to their parent attributes...

- under GAME, we want the "date" and the "time" values passed down

mapping parent-to-child (via XML2R::add_key)

Remember - the LINE nodes are the "facts", and we need to be able to map those facts to their parent attributes...

- under GAME, we want the "date" and the "time" values passed down

```
# adding key - new col of "gamedate"
```

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="date", key.name="gamedate")
```

```
# adding key - new col of "gametime"
```

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="time", key.name="gametime")
```

mapping parent-to-child (via XML2R::add_key)

Remember - the LINE nodes are the "facts", and we need to be able to map those facts to their parent attributes...

- under GAME, we want the "date" and the "time" values passed down

```
# adding key - new col of "gamedate"
```

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="date", key.name="gamedate")
```

```
# adding key - new col of "gametime"
```

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="time", key.name="gametime")
```

- under GAME//TEAM, we want the (team) "name" and the (game-wager) "number" passed down

mapping parent-to-child (via XML2R::add_key)

Remember - the LINE nodes are the "facts", and we need to be able to map those facts to their parent attributes...

- under GAME, we want the "date" and the "time" values passed down

```
# adding key - new col of "gamedate"
```

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="date", key.name="gamedate")
```

```
# adding key - new col of "gametime"
```

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="time", key.name="gametime")
```

- under GAME//TEAM, we want the (team) "name" and the (game-wager) "number" passed down

```
# new col of "gamewgrnbr"
```

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME//TEAM", recycle="number", key.name="gamewgrnbr")
```

```
# new col of "teamname"
```

```
xmlobs1 <- add_key(xmlobs1 , parent="ODDS//LEAGUE//GAME//TEAM", recycle="name", key.name="teamname")
```

collapsing observations (via XML2R::collapse_obs)

Now, we can simply use XML2R::collapse_obs to get a matrix

collapsing observations (via XML2R::collapse_obs)

Now, we can simply use XML2R::collapse_obs to get a matrix

```
#collapse observations from the ....//LINE nodes
```

```
#and remove the "url" colname (don't need it)
```

```
oddsline ← collapse_obs(xmlobs1[grep("^ODDS//LEAGUE//GAME//TEAM//LINE$",names(xmlobs1))])
```

```
oddsline ← oddsline[,-grep("url",colnames(oddsline))]
```

run code and show output

We'll just run the previously shown code and show some output to verify all 3 XML2R functions we've used did their job...

run code and show output

We'll just run the previously shown code and show some output to verify all 3 XML2R functions we've used did their job...

```
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="date", key.name="gamedate") # adding key - new col of "gamedate"
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME", recycle="time", key.name="gametime") # adding key - new col of "gametime"
xmlobs1 <- add_key(xmlobs1, parent="ODDS//LEAGUE//GAME//TEAM", recycle="number", key.name="gamewgrnbr") # new col of "gamewgrnbr"
xmlobs1 <- add_key(xmlobs1 , parent="ODDS//LEAGUE//GAME//TEAM", recycle="name", key.name="teamname")
oddsline <- collapse_obs(xmlobs1[grep("^ODDS//LEAGUE//GAME//TEAM//LINE$",names(xmlobs1))])  # this returns a matrix
oddsline <- oddsline[,-grep("url",colnames(oddsline))]            # get rid of the long "url" colname
head(oddsline)
```

```
##      book value           seconds gamedate   gametime gamewgrnbr teamname
## [1,] "1"  "1&frac12;"     "41302" "20180226" "1600"   "721"      "Marquette"
## [2,] "3"  "1&frac12;"     "36722" "20180226" "1600"   "721"      "Marquette"
## [3,] "5"  "1&frac12;"     "35957" "20180226" "1600"   "721"      "Marquette"
## [4,] "14" "1&frac12;"     "41362" "20180226" "1600"   "721"      "Marquette"
## [5,] "16" "1&frac12;-106" "6009"  "20180226" "1600"   "721"      "Marquette"
## [6,] "17" "1&frac12;-108" "2671"  "20180226" "1600"   "721"      "Marquette"
```

Post-processing: task list

Here's a list of things to do to get the data into a "usable" data frame....

Post-processing: task list

Here's a list of things to do to get the data into a "usable" data frame....

We won't go through them with code, but a screenshot of the *final persistance* to a mysql table will be shown

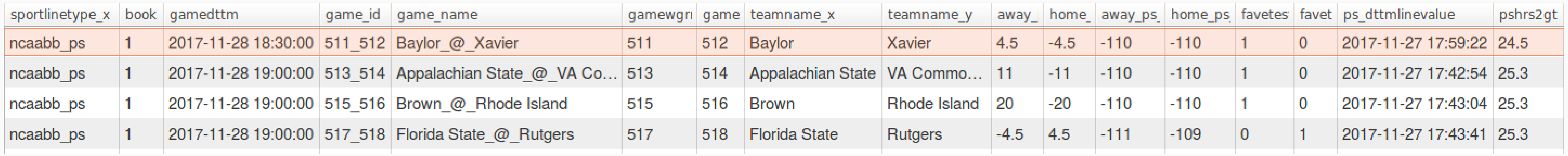
Post-processing: task list

Here's a list of things to do to get the data into a "usable" data frame....

We won't go through them with code, but a screenshot of the *final persistance* to a mysql table will be shown

- parse the lines for half-points (gsub the "ampersand frac12")
- perform necessary character-to-numeric conversions
- ...and then for dates/datetimes, convert to appropriate POSIX.. classes
- get the "datetime-line-refresh" from the global //TIME attribute and...
- use that with the "seconds" value to calculate the datetime for a given "LINE"
- calculate an "hours-to-gametime" column (optional - but easier to digest!)
- load to a database table (mysql is my preference) for tracking line movements over time

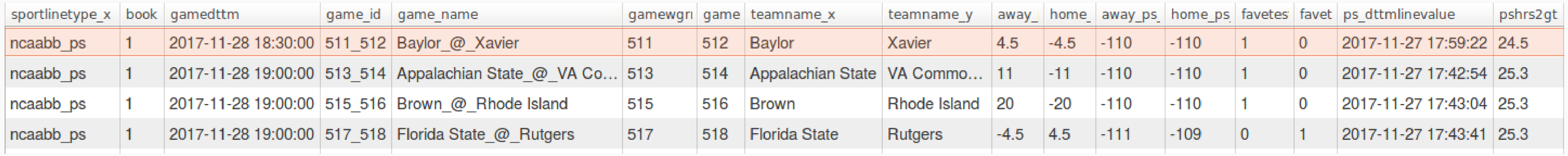
Final resting place (for the parsed & processed XML)



sportlinetype_x	book	gamedttm	game_id	game_name	gamewgr	game	teamname_x	teamname_y	away_	home_	away_ps_	home_ps	favetes	favet	ps_dttmlinevalue	pshrs2gt
ncaabb_ps	1	2017-11-28 18:30:00	511_512	Baylor_@_Xavier	511	512	Baylor	Xavier	4.5	-4.5	-110	-110	1	0	2017-11-27 17:59:22	24.5
ncaabb_ps	1	2017-11-28 19:00:00	513_514	Appalachian State_@_VA Co…	513	514	Appalachian State	VA Commo…	11	-11	-110	-110	1	0	2017-11-27 17:42:54	25.3
ncaabb_ps	1	2017-11-28 19:00:00	515_516	Brown_@_Rhode Island	515	516	Brown	Rhode Island	20	-20	-110	-110	1	0	2017-11-27 17:43:04	25.3
ncaabb_ps	1	2017-11-28 19:00:00	517_518	Florida State_@_Rutgers	517	518	Florida State	Rutgers	-4.5	4.5	-111	-109	0	1	2017-11-27 17:43:41	25.3

Notice the "pshr2gt" column...(point-spread-hours-to-gametime)

Final resting place (for the parsed & processed XML)

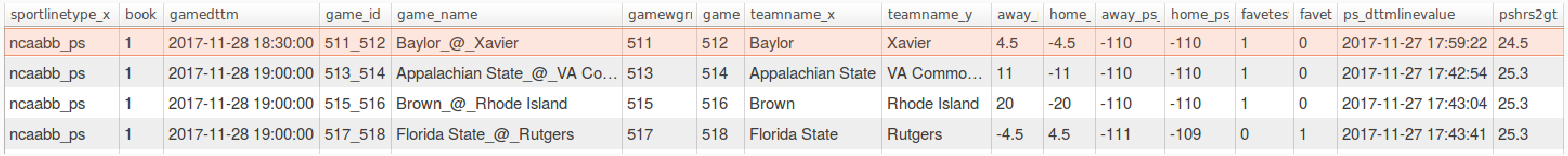


sportlinetype_x	book	gamedttm	game_id	game_name	gamewgri	game	teamname_x	teamname_y	away_	home_	away_ps_	home_ps	favetes	favet	ps_dttmlinevalue	pshrs2gt
ncaabb_ps	1	2017-11-28 18:30:00	511_512	Baylor_@_Xavier	511	512	Baylor	Xavier	4.5	-4.5	-110	-110	1	0	2017-11-27 17:59:22	24.5
ncaabb_ps	1	2017-11-28 19:00:00	513_514	Appalachian State_@_VA Co…	513	514	Appalachian State	VA Commo…	11	-11	-110	-110	1	0	2017-11-27 17:42:54	25.3
ncaabb_ps	1	2017-11-28 19:00:00	515_516	Brown_@_Rhode Island	515	516	Brown	Rhode Island	20	-20	-110	-110	1	0	2017-11-27 17:43:04	25.3
ncaabb_ps	1	2017-11-28 19:00:00	517_518	Florida State_@_Rutgers	517	518	Florida State	Rutgers	-4.5	4.5	-111	-109	0	1	2017-11-27 17:43:41	25.3

Notice the "pshr2gt" column...(point-spread-hours-to-gametime)

- that's from adding "seconds" to the global "TIME" attribute and subtracting game-datetime

Final resting place (for the parsed & processed XML)



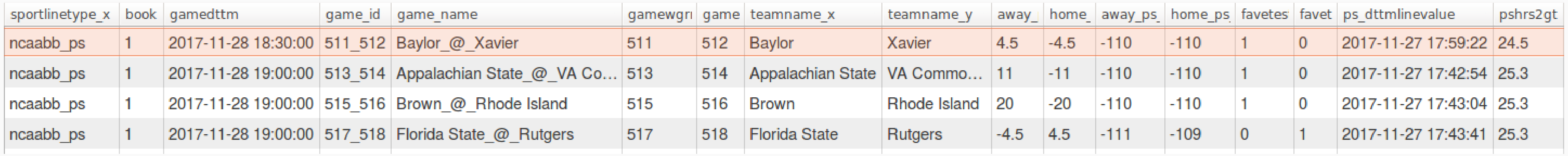
sportlinetype_x	book	gamedttm	game_id	game_name	gamewgr	game	teamname_x	teamname_y	away_	home_	away_ps_	home_ps	favetes	favet	ps_dttmlinevalue	pshrs2gt
ncaabb_ps	1	2017-11-28 18:30:00	511_512	Baylor_@_Xavier	511	512	Baylor	Xavier	4.5	-4.5	-110	-110	1	0	2017-11-27 17:59:22	24.5
ncaabb_ps	1	2017-11-28 19:00:00	513_514	Appalachian State_@_VA Co...	513	514	Appalachian State	VA Commo...	11	-11	-110	-110	1	0	2017-11-27 17:42:54	25.3
ncaabb_ps	1	2017-11-28 19:00:00	515_516	Brown_@_Rhode Island	515	516	Brown	Rhode Island	20	-20	-110	-110	1	0	2017-11-27 17:43:04	25.3
ncaabb_ps	1	2017-11-28 19:00:00	517_518	Florida State_@_Rutgers	517	518	Florida State	Rutgers	-4.5	4.5	-111	-109	0	1	2017-11-27 17:43:41	25.3

Notice the "pshr2gt" column...(point-spread-hours-to-gametime)

- that's from adding "seconds" to the global "TIME" attribute and subtracting game-datetime

  Moving on - knowing when it's time to use RSelenium

Final resting place (for the parsed & processed XML)



sportlinetype_x	book	gamedttm	game_id	game_name	gamewgri	game	teamname_x	teamname_y	away_	home_	away_ps_	home_ps	favetes	favet	ps_dttmlinevalue	pshrs2gt
ncaabb_ps	1	2017-11-28 18:30:00	511_512	Baylor_@_Xavier	511	512	Baylor	Xavier	4.5	-4.5	-110	-110	1	0	2017-11-27 17:59:22	24.5
ncaabb_ps	1	2017-11-28 19:00:00	513_514	Appalachian State_@_VA Co…	513	514	Appalachian State	VA Commo…	11	-11	-110	-110	1	0	2017-11-27 17:42:54	25.3
ncaabb_ps	1	2017-11-28 19:00:00	515_516	Brown_@_Rhode Island	515	516	Brown	Rhode Island	20	-20	-110	-110	1	0	2017-11-27 17:43:04	25.3
ncaabb_ps	1	2017-11-28 19:00:00	517_518	Florida State_@_Rutgers	517	518	Florida State	Rutgers	-4.5	4.5	-111	-109	0	1	2017-11-27 17:43:41	25.3

Notice the "pshr2gt" column...(point-spread-hours-to-gametime)

- that's from adding "seconds" to the global "TIME" attribute and subtracting game-datetime

  Moving on - knowing when it's time to use RSelenium

- (or its easy-to-use wrapper form via the rdom package)

Intro to RSelenium

Intro to RSelenium - What is it?

- serves as a powerful tool for screen scraping due to:

Intro to RSelenium - What is it?

- serves as a powerful tool for screen scraping due to:
    - ability to send "events" (submit forms, clicks, etc)

Intro to RSelenium - What is it?

- serves as a powerful tool for screen scraping due to:
    - ability to send "events" (submit forms, clicks, etc)
    - ability to execute javascript

Intro to RSelenium - What is it?

- serves as a powerful tool for screen scraping due to:
    - ability to send "events" (submit forms, clicks, etc)
    - ability to execute javascript
    - provides shortcuts for scraping via the built-in functions/methods

Intro to RSelenium - What is it?

- having DOM access means we can navigate the DOM using $methods to do anything a human user would do..

Intro to RSelenium - What is it?

- having DOM access means we can navigate the DOM using $methods to do anything a human user would do..
- send mouse events, clicks, submit forms, take screenshots

Intro to RSelenium - What is it?

- having DOM access means we can navigate the DOM using $methods to do anything a human user would do..
- send mouse events, clicks, submit forms, take screenshots
- and makes it easier to extract text and attributes within a webpage much more easily than previously shown

Getting started with RSelenium

- **Suggest** to watch these videos hosted by the RSelenium package author (John Harrison)

Getting started with RSelenium

- **Suggest** to watch these videos hosted by the RSelenium package author (John Harrison) John Harrison video 1 - soup2nuts
  John Harrison video 2 - javascript example

- print out the RSelenium doc and follow along some examples.....

  https://cran.r-project.org/web/packages/RSelenium/RSelenium.pdf

Using RSelenium: Ex. #1

- Now we'll go to a website that requires logging in
- and that has some javascript that we can execute

Using RSelenium: Ex. #1

- Now we'll go to a website that requires logging in
- and that has some javascript that we can execute
- Returning to NCAA Hoops, we'll scrape a day of past predictions:

Using RSelenium: Ex. #1

- Now we'll go to a website that requires logging in
- and that has some javascript that we can execute
- Returning to NCAA Hoops, we'll scrape a day of past predictions:
    - predicted score and margin of victory

Using RSelenium: Ex. #1

- Now we'll go to a website that requires logging in
- and that has some javascript that we can execute
- Returning to NCAA Hoops, we'll scrape a day of past predictions:
    - predicted score and margin of victory
- AND, we'll get some data that is related to "within-game win probabilities"

Using RSelenium: Ex. #1

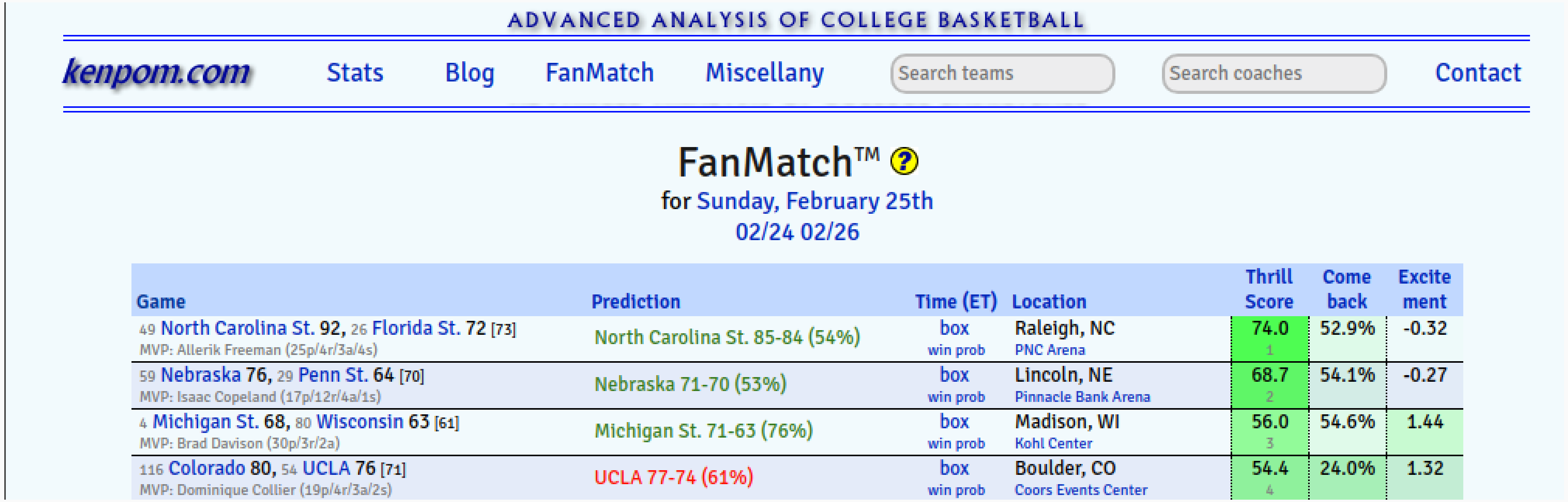
- Now we'll go to a website that requires logging in
- and that has some javascript that we can execute
- Returning to NCAA Hoops, we'll scrape a day of past predictions:
    - predicted score and margin of victory
- AND, we'll get some data that is related to "within-game win probabilities"
    - the data is NOT in a table....

Using RSelenium: Ex. #1

- Now we'll go to a website that requires logging in
- and that has some javascript that we can execute
- Returning to NCAA Hoops, we'll scrape a day of past predictions:
  - predicted score and margin of victory
- AND, we'll get some data that is related to "within-game win probabilities"
  - the data is NOT in a table....
  - it's accessable via executing javascript (another thing that makes RSelenium so powerful)

Using RSelenium: Ex. #1

Data to get from kenpom.com….



ADVANCED ANALYSIS OF COLLEGE BASKETBALL

kenpom.com Stats Blog FanMatch Miscellany Search teams Search coaches Contact

FanMatch™

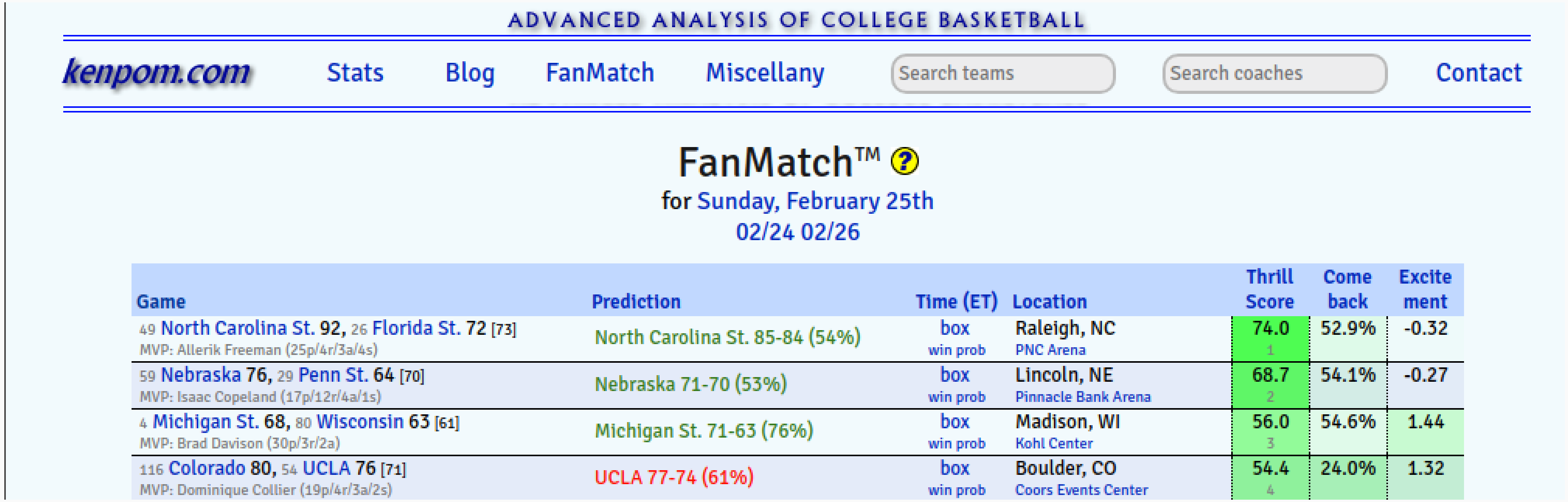
for Sunday, February 25th

02/24 02/26

Game	Prediction	Time (ET)	Location	Thrill Score	Come back	Excite ment
49 North Carolina St. 92, 26 Florida St. 72 [73] MVP: Allerik Freeman (25p/4r/3a/4s)	North Carolina St. 85-84 (54%)	box win prob	Raleigh, NC PNC Arena	74.0 1	52.9%	-0.32
59 Nebraska 76, 29 Penn St. 64 [70] MVP: Isaac Copeland (17p/12r/4a/1s)	Nebraska 71-70 (53%)	box win prob	Lincoln, NE Pinnacle Bank Arena	68.7 2	54.1%	-0.27
4 Michigan St. 68, 80 Wisconsin 63 [61] MVP: Brad Davison (30p/3r/2a)	Michigan St. 71-63 (76%)	box win prob	Madison, WI Kohl Center	56.0 3	54.6%	1.44
116 Colorado 80, 54 UCLA 76 [71] MVP: Dominique Collier (19p/4r/3a/2s)	UCLA 77-74 (61%)	box win prob	Boulder, CO Coors Events Center	54.4 4	24.0%	1.32

Using RSelenium: Ex. #1

Data to get from kenpom.com....



ADVANCED ANALYSIS OF COLLEGE BASKETBALL

kenpom.com | Stats | Blog | FanMatch | Miscellany | Search teams | Search coaches | Contact

FanMatch™

for Sunday, February 25th

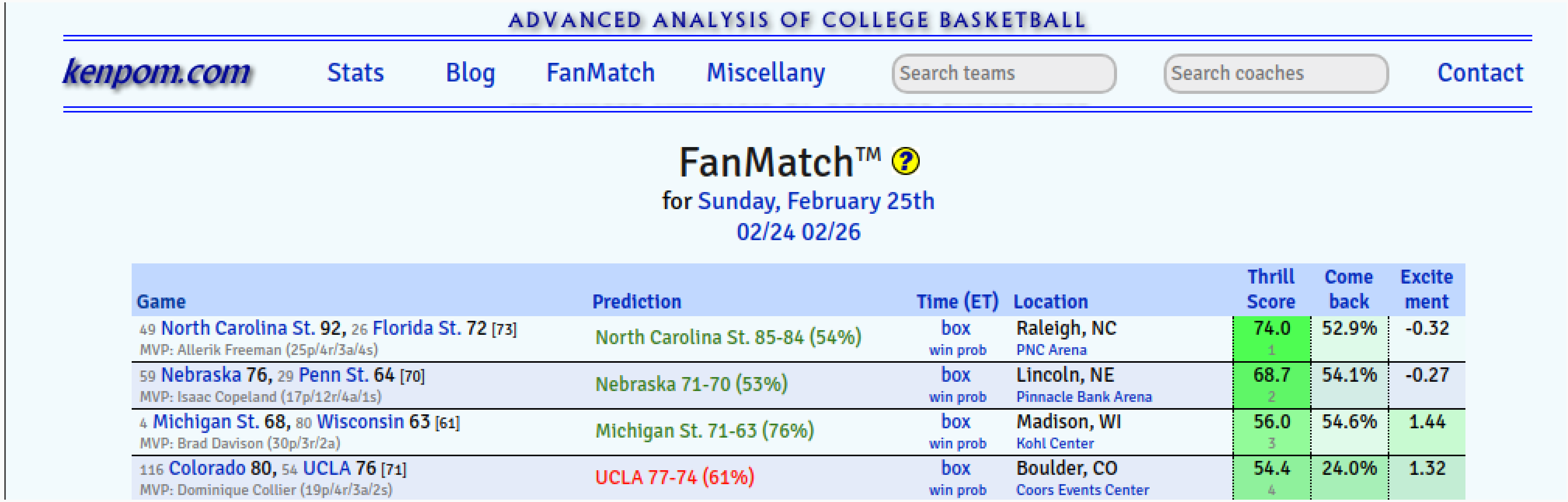
02/24 02/26

Game	Prediction	Time (ET)	Location	Thrill Score	Come back	Excite ment
49 North Carolina St. 92, 26 Florida St. 72 [73] MVP: Allerik Freeman (25p/4r/3a/4s)	North Carolina St. 85-84 (54%)	box win prob	Raleigh, NC PNC Arena	74.0 1	52.9%	-0.32
59 Nebraska 76, 29 Penn St. 64 [70] MVP: Isaac Copeland (17p/12r/4a/1s)	Nebraska 71-70 (53%)	box win prob	Lincoln, NE Pinnacle Bank Arena	68.7 2	54.1%	-0.27
4 Michigan St. 68, 80 Wisconsin 63 [61] MVP: Brad Davison (30p/3r/2a)	Michigan St. 71-63 (76%)	box win prob	Madison, WI Kohl Center	56.0 3	54.6%	1.44
116 Colorado 80, 54 UCLA 76 [71] MVP: Dominique Collier (19p/4r/3a/2s)	UCLA 77-74 (61%)	box win prob	Boulder, CO Coors Events Center	54.4 4	24.0%	1.32

- We will get values from the Prediction column

Using RSelenium: Ex. #1

Data to get from kenpom.com....



ADVANCED ANALYSIS OF COLLEGE BASKETBALL

kenpom.com Stats Blog FanMatch Miscellany Search teams Search coaches Contact

FanMatch™

for Sunday, February 25th

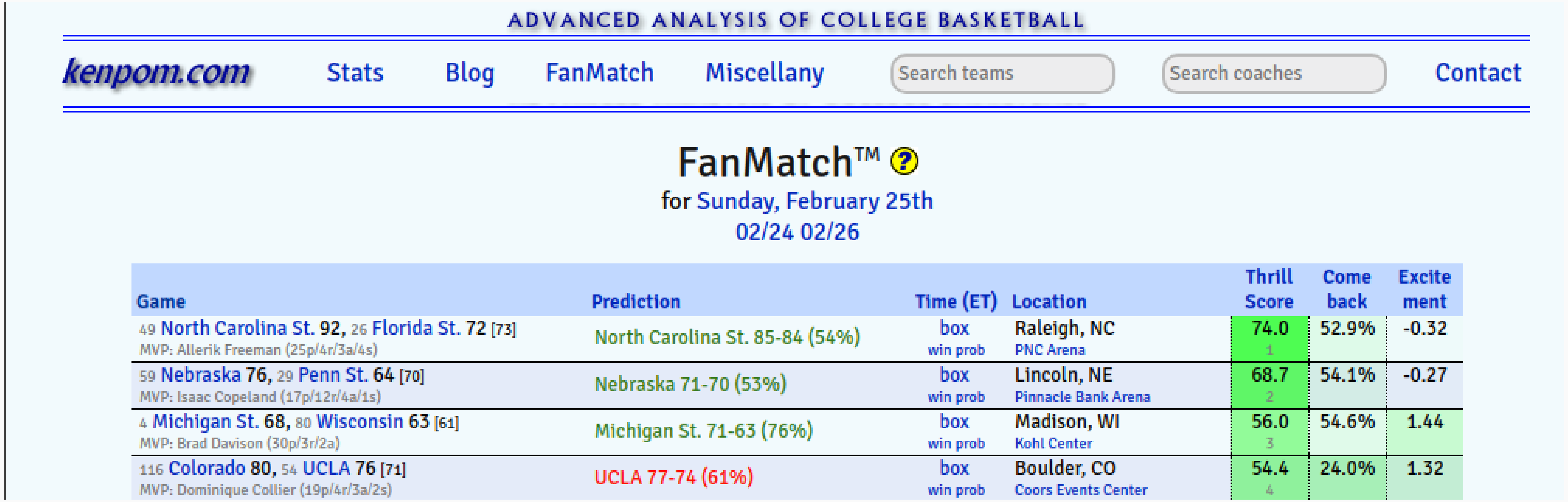
02/24 02/26

Game	Prediction	Time (ET)	Location	Thrill Score	Come back	Excite ment
49 North Carolina St. 92, 26 Florida St. 72 [73] MVP: Allerik Freeman (25p/4r/3a/4s)	North Carolina St. 85-84 (54%)	box win prob	Raleigh, NC PNC Arena	74.0 1	52.9%	-0.32
59 Nebraska 76, 29 Penn St. 64 [70] MVP: Isaac Copeland (17p/12r/4a/1s)	Nebraska 71-70 (53%)	box win prob	Lincoln, NE Pinnacle Bank Arena	68.7 2	54.1%	-0.27
4 Michigan St. 68, 80 Wisconsin 63 [61] MVP: Brad Davison (30p/3r/2a)	Michigan St. 71-63 (76%)	box win prob	Madison, WI Kohl Center	56.0 3	54.6%	1.44
116 Colorado 80, 54 UCLA 76 [71] MVP: Dominique Collier (19p/4r/3a/2s)	UCLA 77-74 (61%)	box win prob	Boulder, CO Coors Events Center	54.4 4	24.0%	1.32

- We will get values from the Prediction column
- as well as the href's from the win prob link (under the "Time" column)

Using RSelenium: Ex. #1

Data to get from kenpom.com....



ADVANCED ANALYSIS OF COLLEGE BASKETBALL

kenpom.com Stats Blog FanMatch Miscellany Search teams Search coaches Contact

FanMatch™

for Sunday, February 25th

02/24 02/26

Game	Prediction	Time (ET)	Location	Thrill Score	Come back	Excite ment
49 North Carolina St. 92, 26 Florida St. 72 [73] MVP: Allerik Freeman (25p/4r/3a/4s)	North Carolina St. 85-84 (54%)	box win prob	Raleigh, NC PNC Arena	74.0 1	52.9%	-0.32
59 Nebraska 76, 29 Penn St. 64 [70] MVP: Isaac Copeland (17p/12r/4a/1s)	Nebraska 71-70 (53%)	box win prob	Lincoln, NE Pinnacle Bank Arena	68.7 2	54.1%	-0.27
4 Michigan St. 68, 80 Wisconsin 63 [61] MVP: Brad Davison (30p/3r/2a)	Michigan St. 71-63 (76%)	box win prob	Madison, WI Kohl Center	56.0 3	54.6%	1.44
116 Colorado 80, 54 UCLA 76 [71] MVP: Dominique Collier (19p/4r/3a/2s)	UCLA 77-74 (61%)	box win prob	Boulder, CO Coors Events Center	54.4 4	24.0%	1.32

- We will get values from the Prediction column
- as well as the href's from the win prob link (under the "Time" column)
- and parse the Prediction column for Winner, score, prob-win, etc.

Using RSelenium: Ex. #1

The win prob graph and javascript-generated data looks like this:

Using RSelenium: Ex. #1

- After installing RSelenium (see videos!, etc), this is how one starts it up

Using RSelenium: Ex. #1

- After installing RSelenium (see videos!, etc), this is how one starts it up

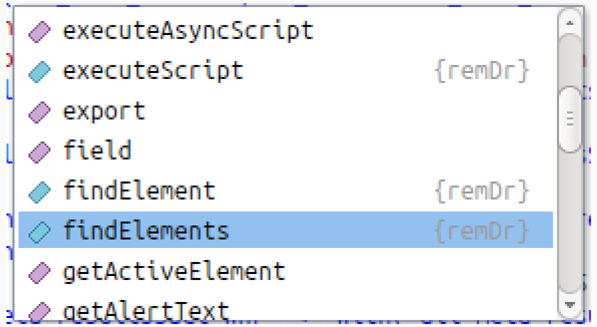
```
library(RSelenium)
rD ← rsDriver(port=4445L, browser="chrome")
remDr ← rD[["client"]]
```

Using RSelenium: Ex. #1

- After installing RSelenium (see videos!, etc), this is how one starts it up

```r
library(RSelenium)
rD <- rsDriver(port=4445L, browser="chrome")
remDr <- rD[["client"]]
```

- we instantiated the "client" as **remDr**...
- remDr$ - see doc for the class methods that make using RSelenium really SMOOTH!



Using RSelenium: Ex. #1

```r
remDr$navigate("https://kenpom.com")
```

```r
#login required - see companion script
```

Using RSelenium: Ex. #1

```r
remDr$navigate("https://kenpom.com")
```

```r
#login required - see companion script
```

```r
#after login, navigate to fanmatch-table
```

```r
remDr$navigate("https://kenpom.com/fanmatch.php?d=2018-02-25")
```

Using RSelenium: Ex. #1

```
remDr$navigate("https://kenpom.com")
```

```
#login required - see companion script
```

```
#after login, navigate to fanmatch-table
```

```
remDr$navigate("https://kenpom.com/fanmatch.php?d=2018-02-25")
```

```
# concact xpaths to search for wnrs and losers WITHIN the table
```

```
game_preds ← remDr$findElements(value = "//td[@class = 'wrong'] | //td[@class = 'correct']")
```

Using RSelenium: Ex. #1

```
remDr$navigate("https://kenpom.com")
```

```
#login required - see companion script
```

```
#after login, navigate to fanmatch-table
```

```
remDr$navigate("https://kenpom.com/fanmatch.php?d=2018-02-25")
```

```
# concact xpaths to search for wnrs and losers WITHIN the table
```

```
game_preds ← remDr$findElements(value = "//td[@class = 'wrong'] | //td[@class = 'correct']")
```

```
# Now, we can use RSelenium's class ('$') methods technique to get the text which we will (later) parse
```

```
game_predsTxt ← sapply(game_preds,function(x) x$getElementText())
```

Using RSelenium: Ex. #1

- to get the within-game probabilities, we need to get all hrefs that point to the graphs
- …we will (later) iterate over those links

```
# find ALL elements, then sapply that result to get the element attribute="href" (a link!)
fmUrls  ← remDr$findElements(value = "//span/a")
fmHrefs ← sapply(fmUrls,function(x) x$getElementAttribute('href'))

#each game also has a link to the "MVP" for the game, so filter those out
# keeping only "winprob" links
fmWinProbs ← fmHrefs[grep("winprob",fmHrefs)]
```

Using RSelenium: Ex. #1

- iterate over the URLs that have the win-prob graphs...
- and extract the data via remDr$executeScript()

```r
for (i in 1:length(fmWinProbs)){
    remDr$navigate(fmWinProbs[[i]])
    winprob <- remDr$executeScript("return dataset;")
winprobDf <- do.call(rbind.data.frame,winprob)
    gamedata <- remDr$executeScript("return data;")
gamedataDf <- do.call(cbind.data.frame,gamedata[!names(gamedata) %in% 'input'])
    wpdf <- merge(winprobDf, gamedataDf[names(gamedataDf) %in% c('gid','ymd','gameTime','team1','team2')])
names(wpdf) <- gsub("WP","visitorWP",names(wpdf))
names(wpdf) <- gsub("team1","team_away",names(wpdf))
names(wpdf) <- gsub("team2","team_home",names(wpdf))
wpdf$homeWP <- with(wpdf,1-visitorWP)
if (i==1) {all_games <- wpdf
           all_meta <- gamedataDf }
    else {all_games <- rbind(all_games,wpdf)
          all_meta <- rbind(all_meta,gamedataDf)}
}
results <- cbind.data.frame(all_meta, pred_wnr, pred_pwin, pred_wnrscr, pred_loserscr,stringsAsFactors=FALSE)
results$pred_ttl <- with(results, pred_wnrscr+pred_loserscr)
results$act_ttl <- with(results, score1+score2)
results$pred_diff <- pred_wnrscr - pred_loserscr
results$act_diff  <- ifelse(results$pred_wnr==results$team2,results$score2-results$score1,results$score1-results$score2)
```

Using RSelenium: Ex. #1

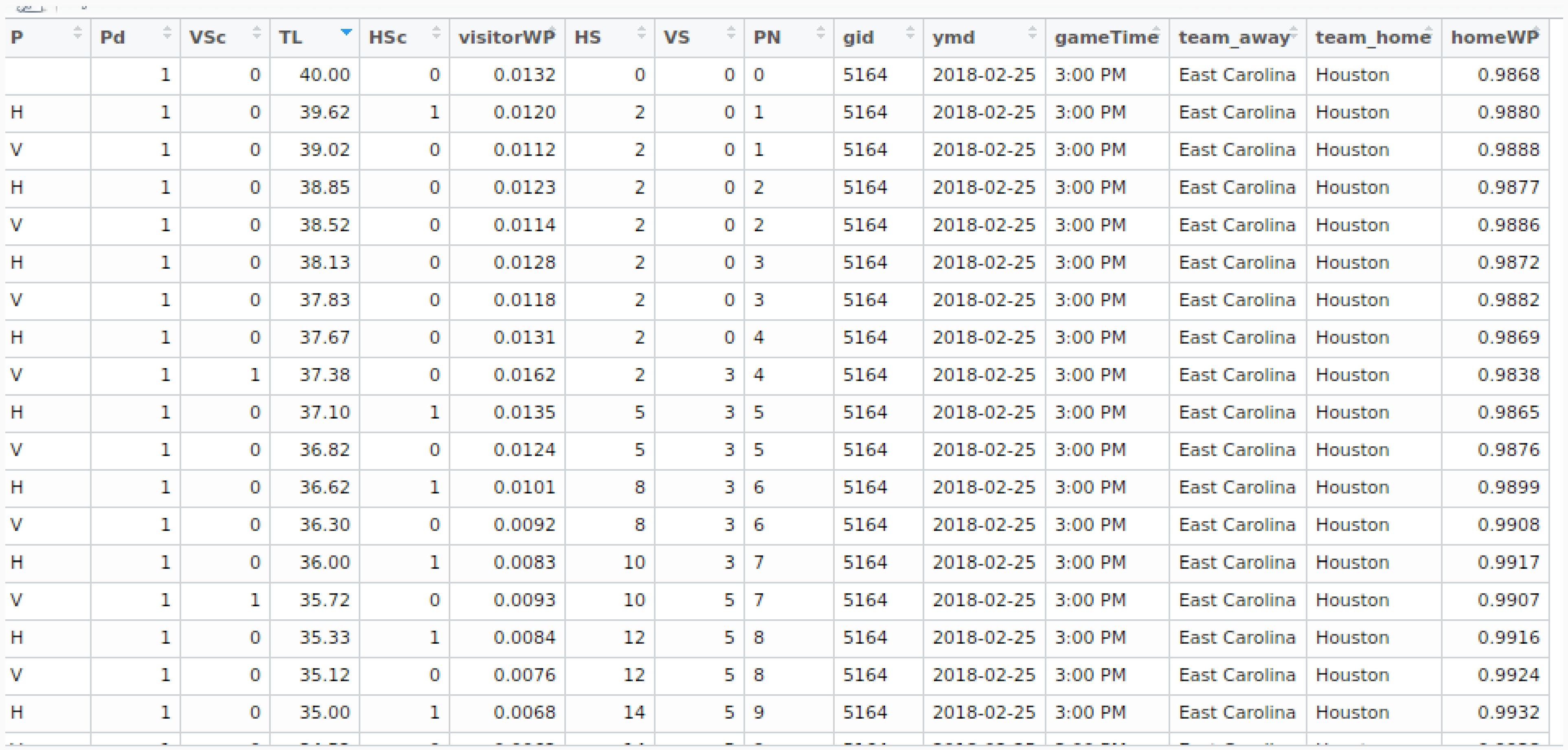
- There are TWO data frames obtained from the javascript calls

Using RSelenium: Ex. #1

- There are TWO data frames obtained from the javascript calls
- the detailed one with the within-game probabilities (all_games data frame)

Using RSelenium: Ex. #1

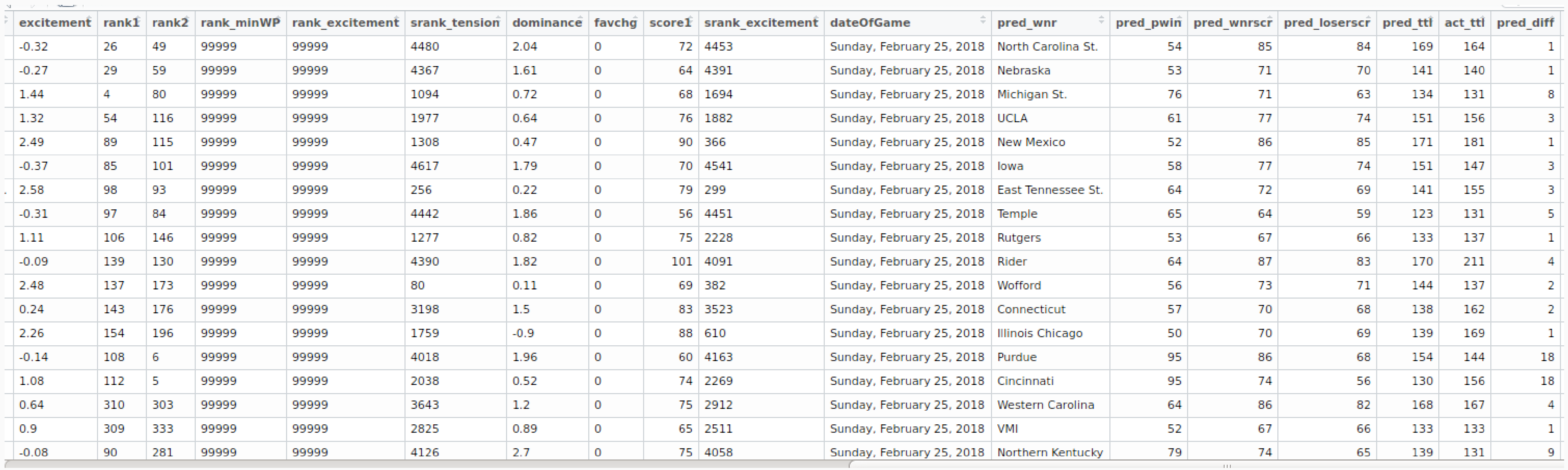
- There are TWO data frames obtained from the javascript calls
- the detailed one with the within-game probabilities (all_games data frame)



P	Pd	VSc	TL	HSc	visitorWP	HS	VS	PN	gid	ymd	gameTime	team_away	team_home	homeWP
	1	0	40.00	0	0.0132	0	0	0	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9868
H	1	0	39.62	1	0.0120	2	0	1	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9880
V	1	0	39.02	0	0.0112	2	0	1	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9888
H	1	0	38.85	0	0.0123	2	0	2	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9877
V	1	0	38.52	0	0.0114	2	0	2	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9886
H	1	0	38.13	0	0.0128	2	0	3	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9872
V	1	0	37.83	0	0.0118	2	0	3	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9882
H	1	0	37.67	0	0.0131	2	0	4	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9869
V	1	1	37.38	0	0.0162	2	3	4	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9838
H	1	0	37.10	1	0.0135	5	3	5	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9865
V	1	0	36.82	0	0.0124	5	3	5	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9876
H	1	0	36.62	1	0.0101	8	3	6	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9899
V	1	0	36.30	0	0.0092	8	3	6	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9908
H	1	0	36.00	1	0.0083	10	3	7	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9917
V	1	1	35.72	0	0.0093	10	5	7	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9907
H	1	0	35.33	1	0.0084	12	5	8	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9916
V	1	0	35.12	0	0.0076	12	5	8	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9924
H	1	0	35.00	1	0.0068	14	5	9	5164	2018-02-25	3:00 PM	East Carolina	Houston	0.9932

Using RSelenium: Ex. #1

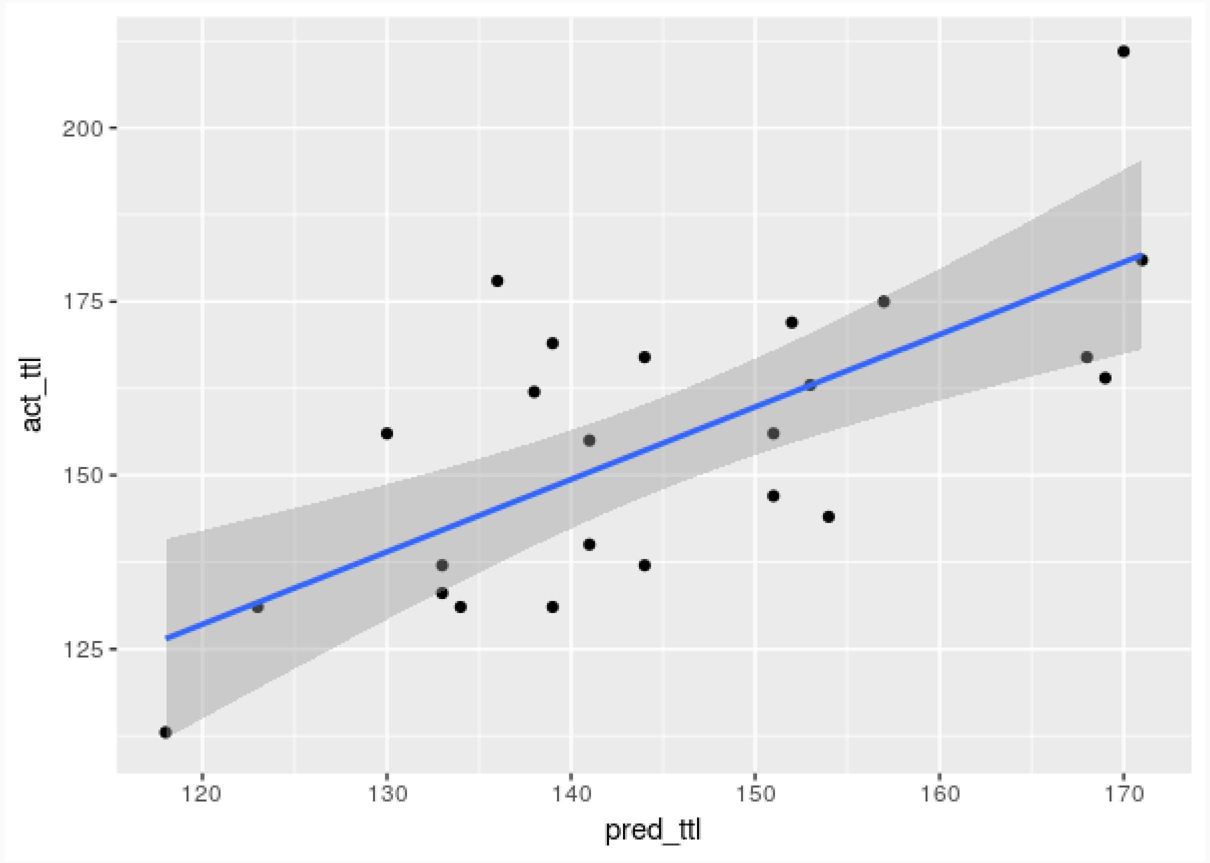
- and the "meta" data one with the predictions & outcomes (results data frame)



excitement	rank1	rank2	rank_minWP	rank_excitement	srank_tension	dominance	favchg	score1	srank_excitement	dateOfGame	pred_wnr	pred_pwin	pred_wnrscr	pred_loserscr	pred_ttl	act_ttl	pred_diff
-0.32	26	49	99999	99999	4480	2.04	0	72	4453	Sunday, February 25, 2018	North Carolina St.	54	85	84	169	164	1
-0.27	29	59	99999	99999	4367	1.61	0	64	4391	Sunday, February 25, 2018	Nebraska	53	71	70	141	140	1
1.44	4	80	99999	99999	1094	0.72	0	68	1694	Sunday, February 25, 2018	Michigan St.	76	71	63	134	131	8
1.32	54	116	99999	99999	1977	0.64	0	76	1882	Sunday, February 25, 2018	UCLA	61	77	74	151	156	3
2.49	89	115	99999	99999	1308	0.47	0	90	366	Sunday, February 25, 2018	New Mexico	52	86	85	171	181	1
-0.37	85	101	99999	99999	4617	1.79	0	70	4541	Sunday, February 25, 2018	Iowa	58	77	74	151	147	3
2.58	98	93	99999	99999	256	0.22	0	79	299	Sunday, February 25, 2018	East Tennessee St.	64	72	69	141	155	3
-0.31	97	84	99999	99999	4442	1.86	0	56	4451	Sunday, February 25, 2018	Temple	65	64	59	123	131	5
1.11	106	146	99999	99999	1277	0.82	0	75	2228	Sunday, February 25, 2018	Rutgers	53	67	66	133	137	1
-0.09	139	130	99999	99999	4390	1.82	0	101	4091	Sunday, February 25, 2018	Rider	64	87	83	170	211	4
2.48	137	173	99999	99999	80	0.11	0	69	382	Sunday, February 25, 2018	Wofford	56	73	71	144	137	2
0.24	143	176	99999	99999	3198	1.5	0	83	3523	Sunday, February 25, 2018	Connecticut	57	70	68	138	162	2
2.26	154	196	99999	99999	1759	-0.9	0	88	610	Sunday, February 25, 2018	Illinois Chicago	50	70	69	139	169	1
-0.14	108	6	99999	99999	4018	1.96	0	60	4163	Sunday, February 25, 2018	Purdue	95	86	68	154	144	18
1.08	112	5	99999	99999	2038	0.52	0	74	2269	Sunday, February 25, 2018	Cincinnati	95	74	56	130	156	18
0.64	310	303	99999	99999	3643	1.2	0	75	2912	Sunday, February 25, 2018	Western Carolina	64	86	82	168	167	4
0.9	309	333	99999	99999	2825	0.89	0	65	2511	Sunday, February 25, 2018	VMI	52	67	66	133	133	1
-0.08	90	281	99999	99999	4126	2.7	0	75	4058	Sunday, February 25, 2018	Northern Kentucky	79	74	65	139	131	9

Using RSelenium: Ex. #1

- Now that there's some data, let's at least show it!



Using RSelenium: Conclusions

- RSelenium makes scraping much less cumbersome than via other methods....
- it's worth the associated learning curve

Don't need the "full" power of RSelenium?

- try using rdom
- it allows access to the DOM (for parsing dynamic content)

  https://github.com/cpsievert/rdom