Data Collection from Web Sources

Session 2 @ CBA Batch 12
April 2019

sudhir_voleti@isb.edu

1

Background to the Example: Evolv-Xerox

- When looking for workers to staff its call centers, Xerox Corp. used **traditional hiring methods** and valued past experience on the job.
- Then, a computer program told Xerox Inc. that **experience doesn't matter.**
- Evolv Inc., a San Francisco start-up designed this algorithm that crunches and matches *psychograhic* profiles with work outcomes.
- Claim was that conventional Hiring methods are remarkably short on rigor for this crucial business function
- Can a **statistical approach to hiring** improve results by reducing hiring managers' biases?

2

Background to the Example: Evolv-Xerox

- Evolv's Method: Job applicants take a 30-minute test that screens them for personality traits.
- Qs include Agree-Disagree on a 1-7 scale for "I ask more questions than most people do.", "People tend to trust what I say." Etc.
- The firm has a **database** of **current and past** worker psychographics and job performance.
- It *'statistically' matches* the psychographic patterns and looks for the best job performance prediction for each application.
- It then assigns a score - Green, Yellow or Red - to each applicant.

3

Evolv-Xerox Example: So what happened finally?

- Attrition dropped 20%. On-job performance improved similarly.
- The algo uncovers correlations the naked eye can’t see. Some fun facts the algo uncovered…
- “Who would you rather hire – A or B?”
- A) Candidate who used Internet Explorer or Safari to answer your questionnaire
- B) Candidate who used Firefox or Chrome?
- A) Candidate who has joined 1-2 social networks
- B) Candidate who has joined 3-4 social networks?

4

Evolv-Xerox: Inputs to the machine

- So, how did a set of humble Likerts provide inputs for this fancy analysis?
- Consider the data Evolv had: X
  - the individual side (demographics, psychographics, etc.)
  - the firm side (training costs, productivity & turnover rates etc.) y

f(.)

- Given this data, Evolv is somehow able to *connect* job outcomes to psychographic profiles… and thereafter *predict* outcomes for new psychographic profiles…
- So, what *principle* drives Evolv's approach?

$y = f(X) + e$

5

Motivating Example: Concluded

- There's a feeling that survey research is *so* 20th century.
- Pen-and-paper. Grunt work. Low-tech.
- That may be so.
- However, the primary DC it provides can yield insight-gold.
- Even humble Likert Qs on psyche-related phenomena can sometimes yield very powerful predictors (as latent feature combinations) …
- … that today's powerful machine learning algos can happily munch on.

6

Session Outline

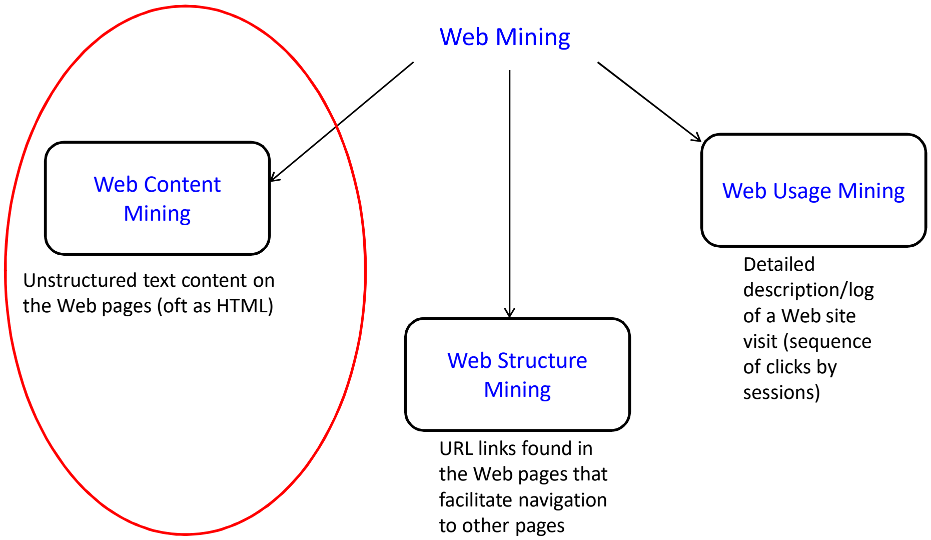
- A Motivating Example
  - Evolv-Xerox and Likert Scales
- Basic HTML primer
- DC from static web pages
  - Examples 1, 2, 3
- DC from dynamic webpages
  - Using webdrivers in py
- Session Wrap-up

7

Some Preliminaries

8

Preliminaries: Web Mining Types



Web Mining

Web Content Mining

Unstructured text content on the Web pages (oft as HTML)

Web Structure Mining

URL links found in the Web pages that facilitate navigation to other pages

Web Usage Mining

Detailed description/log of a Web site visit (sequence of clicks by sessions)

9

Preliminaries: HTML & the DOM

- What is *web-scraping*? What is its objective?
- What are webpages written in?
- What is HTML? What does it look like?
- What is a DOM? What does it look like?
- What converts HTML & CSS to human-readable form?
- How much of HTML code would typically be of interest in web-scraping?

10

Preliminaries: Static and Dynamic web pages

- Part of a web-page may have merely 'static' content in html.
- Part of the page may be dynamic - loading and updating constantly. (E.g., comments sections, twitter or FB timelines etc)
- Which of the 2 would be easier to scrape?
- For static pages, we'll use *rvest* in R and *beautiful soup* in Py.
- Sometimes we may need to evaluate conditions interactively and then only decide what to scrape and when. Enter *webdrivers*.
- Before we proceed, a basic primer on HTML …

11

A Basic HTML Primer

12

An *Elementary* HTML primer

- Aim of this primer is to demonstrate:

- Basic html structures
- Major html tag-types such as:
  - a. title tag
  - b. body tags
  - c. document structure tags
  - d. basic markup tags
  - e. links, image tags
  - f. table tags

- Idea is to enable easy scanning of the page-source of static webpages.

- Open file ‘html primer dc text-editor.txt' and *save-as* a '.html' file. Now open the .html file in browser.

13

Basic HTML Primer: Recap

- List some HTML tags we just saw.

- Consider the power and possibility to create variously designed and marked-up webpages simply remixing a handful of basic HTML tags.

- Can we now see page-sources on the web and estimate what the actual markups may look like?

- Ready to handle simple homework on building a static web-page?

- For scraping data off of web-pages, we'll need to identify particular patterns in html tags.

- Let's head there next, then.

14

Scraping Static Web-pages

rvest and beautifulSoup

15

A Roadmap for this Section

- Will need CSS selector gadget on chrome browser for this section.
- Plan is to cover 3 examples.
- 1. Scraping the IMDB - mini use-case as a walkthrough example.
- 2. Scraping a Telecom glossary @ www.atis.org/glossary
- 3. Py scraping news-stories @ news.ycombinator.com
- Ready? Open ‘rvest_example.html'

16

A Walk-through Example: IMDB

- Q: “What attributes characterize the *best* movies of all time?”

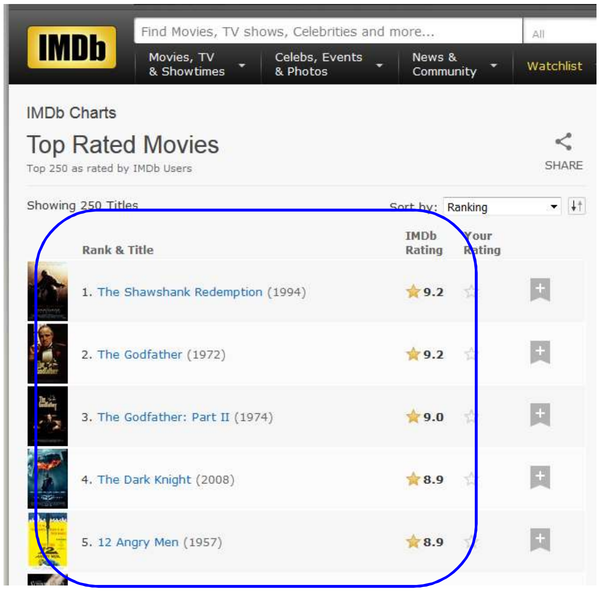
- How would you approach this Q? Where would you first look for data?

- Even before DC, what might “attributes” here mean? And what is ‘best’ anyway?

17

rvest for web scraping: Walkthrough Example

- Step 1: Go to the webpage, study it. ID features of interest such as ________.



IMDb Charts

Top Rated Movies

Top 250 as rated by IMDb Users

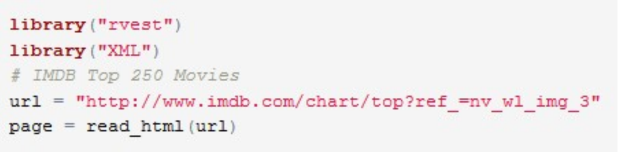
Showing 250 Titles

Rank & Title	IMDb Rating	Your Rating
1. The Shawshank Redemption (1994)	9.2	
2. The Godfather (1972)	9.2	
3. The Godfather: Part II (1974)	9.0	
4. The Dark Knight (2008)	8.9	
5. 12 Angry Men (1957)	8.9	

18

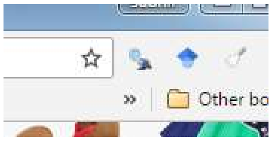
rvest for web scraping: Walkthrough Example

- Step 2: Read it into R using _________ function.



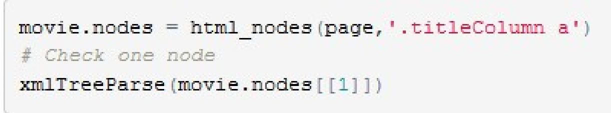
```
library("rvest")
library("XML")
# IMDB Top 250 Movies
url = "http://www.imdb.com/chart/top?ref_=nv_wl_img_3"
page = read_html(url)
```

- Step 3: Activate the SelectorGadget tool on Chrome browser.

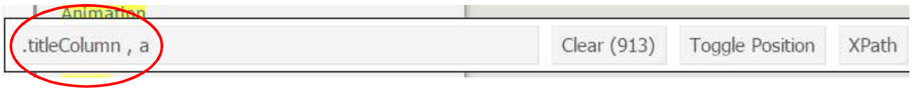


rvest for web scraping: Walkthrough Example

- Step 4: ID the CSS tags needed and read them into _______ function.



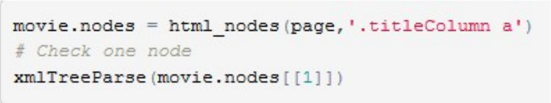
```
movie.nodes = html_nodes(page,'.titleColumn a')
# Check one node
xmlTreeParse(movie.nodes[[1]])
```



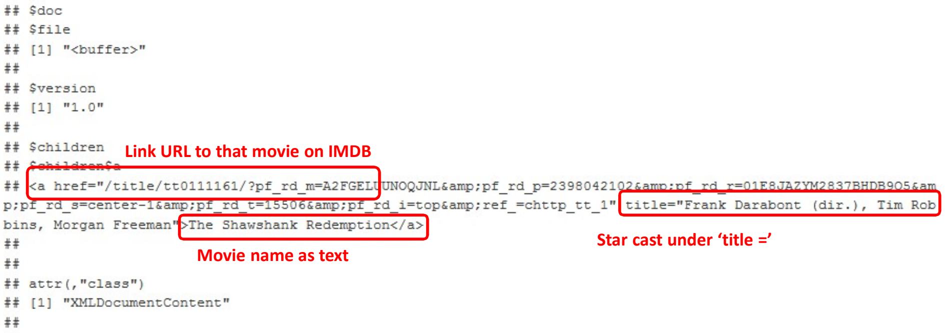
.titleColumn , a

rvest for web scraping: Walkthrough Example

- Step 5: Scan the structure of extracted nodes. ID that tiny % that is useful using the _____ function.



```
movie.nodes = html_nodes(page,'.titleColumn a')
# Check one node
xmlTreeParse(movie.nodes[[1]])
```



```
## $doc
## $file
## [1] "<buffer>"
##
## $version
## [1] "1.0"
##
## $children
## ~~$children$a~~
## <a href="/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&amp;pf_rd_p=2398042102&amp;pf_rd_r=01E8JAZYM2837BHDB9O5&am
p;pf_rd_s=center-1&amp;pf_rd_t=15506&amp;pf_rd_i=top&amp;ref_=chttp_tt_1" title="Frank Darabont (dir.), Tim Rob
bins, Morgan Freeman">The Shawshank Redemption</a>
##
##
## attr(,"class")
## [1] "XMLDocumentContent"
##
```

Link URL to that movie on IMDB

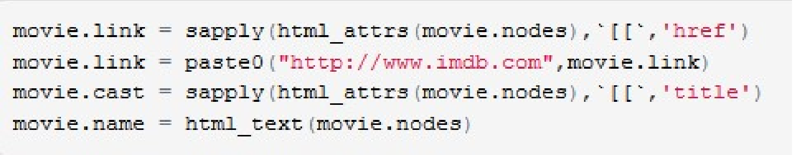
Star cast under 'title ='

Movie name as text

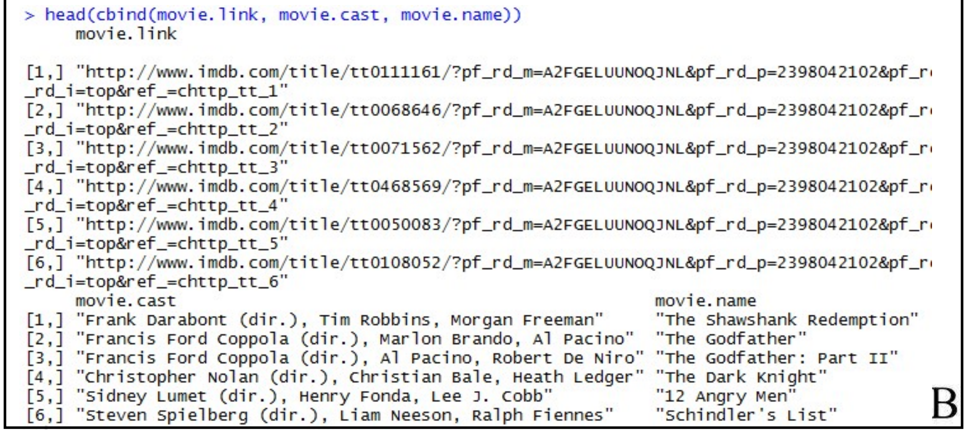
21

rvest for web scraping: Walkthrough Example

- Step 6: Store fields of interest as separate vectors because _____.



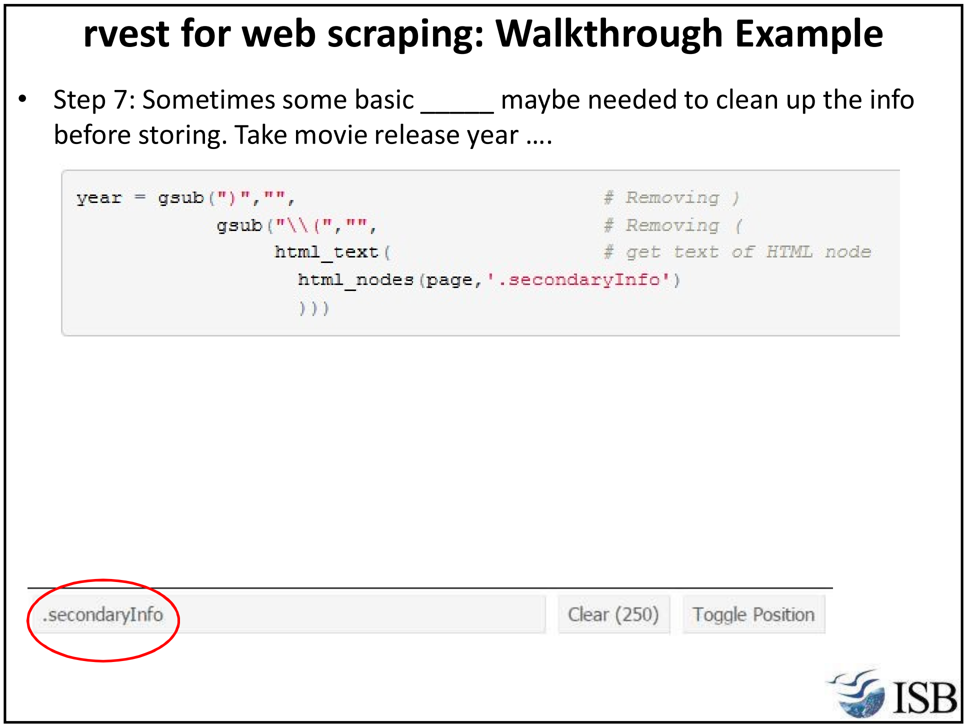
```
movie.link = sapply(html_attrs(movie.nodes),`[[`,'href')
movie.link = paste0("http://www.imdb.com",movie.link)
movie.cast = sapply(html_attrs(movie.nodes),`[[`,'title')
movie.name = html_text(movie.nodes)
```



```
> head(cbind(movie.link, movie.cast, movie.name))
     movie.link

[1,] "http://www.imdb.com/title/tt0111161/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_1"
[2,] "http://www.imdb.com/title/tt0068646/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_2"
[3,] "http://www.imdb.com/title/tt0071562/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_3"
[4,] "http://www.imdb.com/title/tt0468569/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_4"
[5,] "http://www.imdb.com/title/tt0050083/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_5"
[6,] "http://www.imdb.com/title/tt0108052/?pf_rd_m=A2FGELUUNOQJNL&pf_rd_p=2398042102&pf_r
_rd_i=top&ref_=chttp_tt_6"
     movie.cast                                              movie.name
[1,] "Frank Darabont (dir.), Tim Robbins, Morgan Freeman"      "The Shawshank Redemption"
[2,] "Francis Ford Coppola (dir.), Marlon Brando, Al Pacino"   "The Godfather"
[3,] "Francis Ford Coppola (dir.), Al Pacino, Robert De Niro"  "The Godfather: Part II"
[4,] "Christopher Nolan (dir.), Christian Bale, Heath Ledger"  "The Dark Knight"
[5,] "Sidney Lumet (dir.), Henry Fonda, Lee J. Cobb"           "12 Angry Men"
[6,] "Steven Spielberg (dir.), Liam Neeson, Ralph Fiennes"     "Schindler's List"
```

22

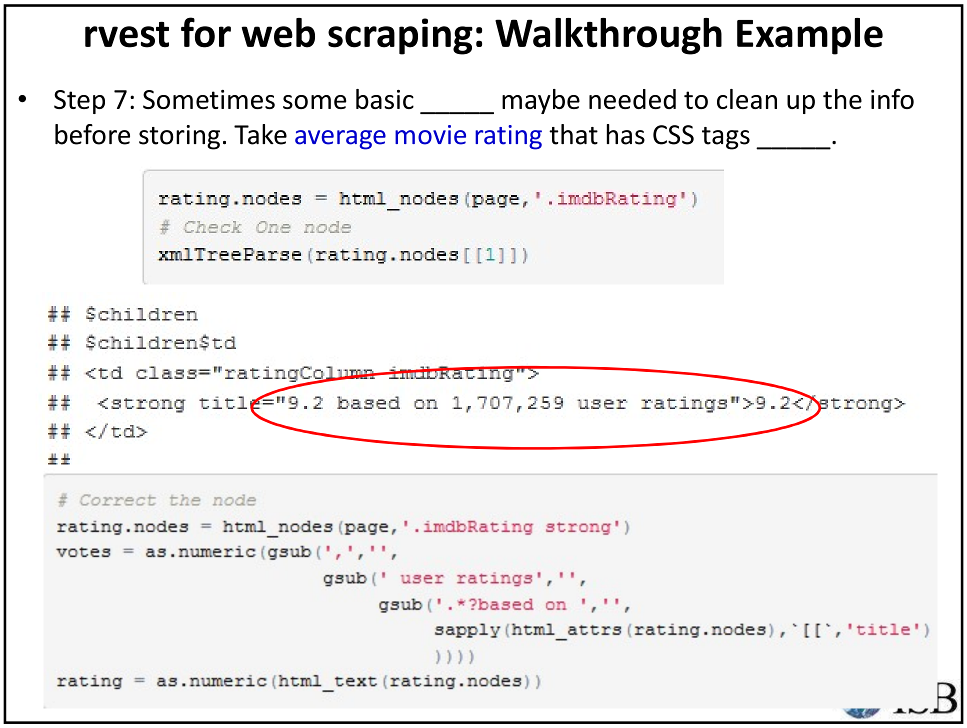


rvest for web scraping: Walkthrough Example

- Step 7: Sometimes some basic ______ maybe needed to clean up the info before storing. Take movie release year ….

```
year = gsub(")","",                         # Removing )
            gsub("\\(","",                  # Removing (
                 html_text(                 # get text of HTML node
                   html_nodes(page,'.secondaryInfo')
                   )))
```

23



rvest for web scraping: Walkthrough Example

- Step 7: Sometimes some basic ______ maybe needed to clean up the info before storing. Take average movie rating that has CSS tags ______.

```
rating.nodes = html_nodes(page,'.imdbRating')
# Check One node
xmlTreeParse(rating.nodes[[1]])
```

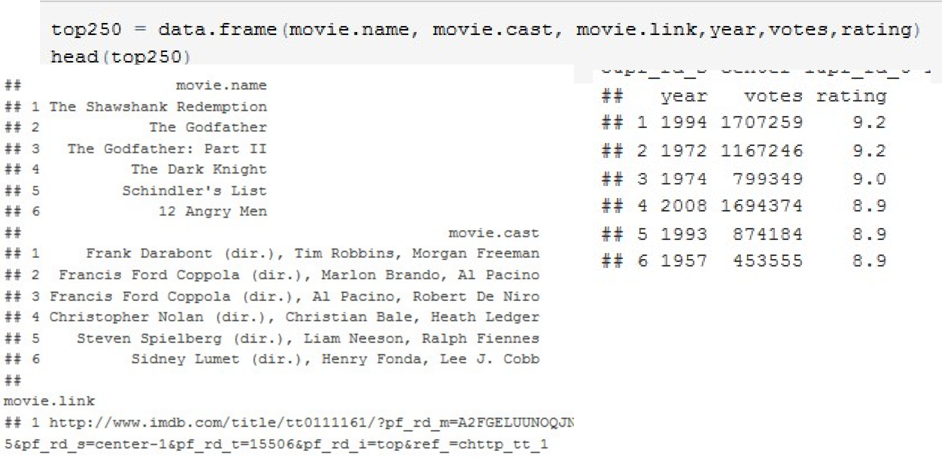
```
## $children
## $children$td
## <td class="ratingColumn imdbRating">
##  <strong title="9.2 based on 1,707,259 user ratings">9.2</strong>
## </td>
##
```

```
# Correct the node
rating.nodes = html_nodes(page,'.imdbRating strong')
votes = as.numeric(gsub(',','',
                        gsub(' user ratings','',
                             gsub('.*?based on ','',
                                  sapply(html_attrs(rating.nodes),`[[`,'title')
                                  ))))
rating = as.numeric(html_text(rating.nodes))
```

24

rvest for web scraping: Walkthrough Example

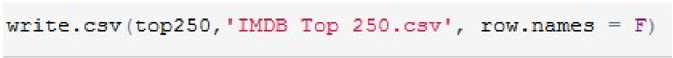
- Step 8: Build a dataframe with all extracted info and write to disk.



```
top250 = data.frame(movie.name, movie.cast, movie.link,year,votes,rating)
head(top250)
```

```
##                  movie.name
## 1 The Shawshank Redemption
## 2            The Godfather
## 3   The Godfather: Part II
## 4          The Dark Knight
## 5         Schindler's List
## 6             12 Angry Men
##                                               movie.cast
## 1     Frank Darabont (dir.), Tim Robbins, Morgan Freeman
## 2  Francis Ford Coppola (dir.), Marlon Brando, Al Pacino
## 3 Francis Ford Coppola (dir.), Al Pacino, Robert De Niro
## 4 Christopher Nolan (dir.), Christian Bale, Heath Ledger
## 5    Steven Spielberg (dir.), Liam Neeson, Ralph Fiennes
## 6          Sidney Lumet (dir.), Henry Fonda, Lee J. Cobb
##
movie.link
## 1 http://www.imdb.com/title/tt0111161/?pf_rd_m=A2FGELUUNOQJN
5&pf_rd_s=center-1&pf_rd_t=15506&pf_rd_i=top&ref_=chttp_tt_1
```

```
##    year   votes rating
## 1 1994 1707259    9.2
## 2 1972 1167246    9.2
## 3 1974  799349    9.0
## 4 2008 1694374    8.9
## 5 1993  874184    8.9
## 6 1957  453555    8.9
```



```
write.csv(top250,'IMDB Top 250.csv', row.names = F)
```

25

rvest for web-scraping: Recap

- We saw just how easy it can be to scrape data off (reasonably structured) web-pages.

- But hey, suppose you really needed to collect movie genre, then what?

- Turns out 'movie.link' links to the movie's IMDB page where such info is available.

- One can replicate the previous analysis this time on the movie's IMDB page --> ID & extract more features --> loop over the 250 movies.

- Heck, if there are more links of interest, then one can "chain" together web scraping ops on successive pages.

26

rvest : Recap and Ponder

- A good time to take a step back and review learnings from rvest
- What libraries did we call?
- What main inbuilt functions did we use?
- What user-defined functions did we use?
- What inputs and outputs to the web-scraping did we see?
- Any other comment on learnings? Applications? Assignments?

27

Another rvest Exercise

- So far we built vectors of features of interest (e.g., movie name, cast etc) by *applying* funcs over the entire webpage, then *column-binding* into DFs.
- Alternately, we could've taken each unit (e.g., movie) as an R object, extracted features of interest and *row-bound* the result into a DF.
- Open the HTML file 'Webscraping Basics with R and Py'.
- The file contains 2 examples - one for scraping a telecom industry glossary page-by-page, and the other for scraping the ycombinator main news page.

28

Scraping a Glossary: Example

- Why scrape atis.org?
  - Well, this exercise is illustrative. For the tools & skills that're more generally applicable.
- Note change in URLs as we click on each link.
- In structured web-pages, there is a **pattern** to that URL change that we can leverage to scrape the site easily.
- Each glossary term in turn is a link, with an id that leads to a new page (check its URL) with the text definition of that term.
- Our aim scrape the glossary's terms alone.
  - But we could, if need be, scrape the text definition as well.

29

Scraping a Glossary: Recap

- What did we just do?
- What libraries did we call?
- What functions did we use?
- What kind of structure did we rely on?
- P.S. What if we also needed to capture each glossary term's text definition?

30

Scraping news.ycombinator: Another Exercise

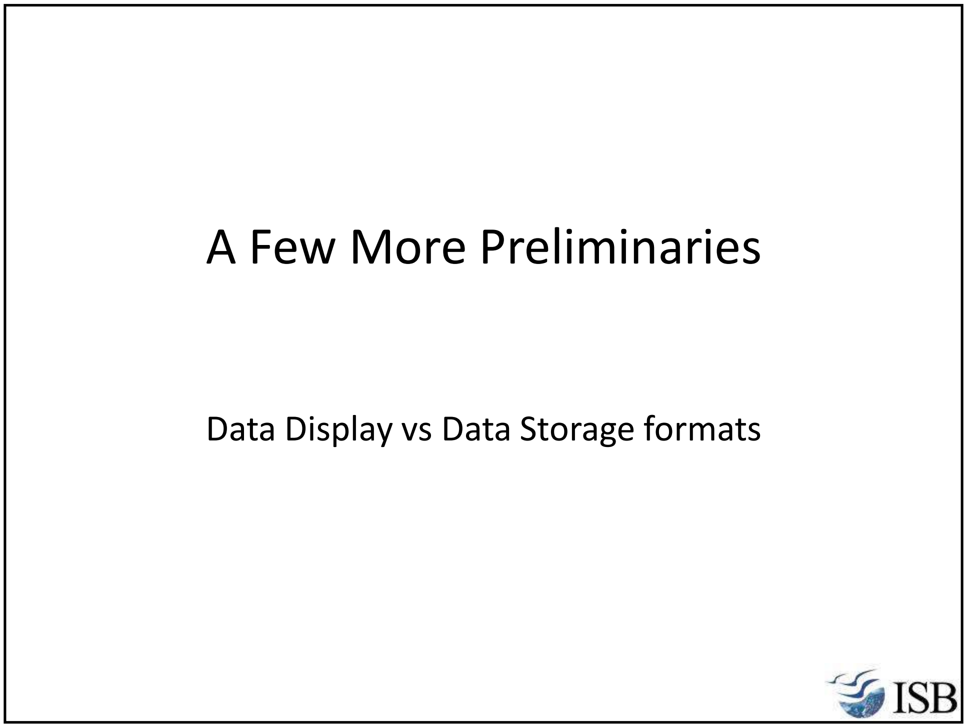
- Along the same lines, however there's a twist here.
- This one requires we study the view-source more deeply.
- From page-source, we can id the nodes of interest (e.g., 'a') and the attributes in that node (e.g., 'storylink', 'sitestr') that contain quantities of interest like text, links etc.
- Let's do this one in Python's Beautiful Soup.
- Open spyder and let's begin.

31

Beautiful soup-ing the webpage in py

- Py up and running for everyone? Open spyder (or Jupyter).
- Copy paste the code, chunk by chunk.
- Examine the variables get created in the variable explorer pane.
- Recall that the hard work of navigating the page-source was done already.

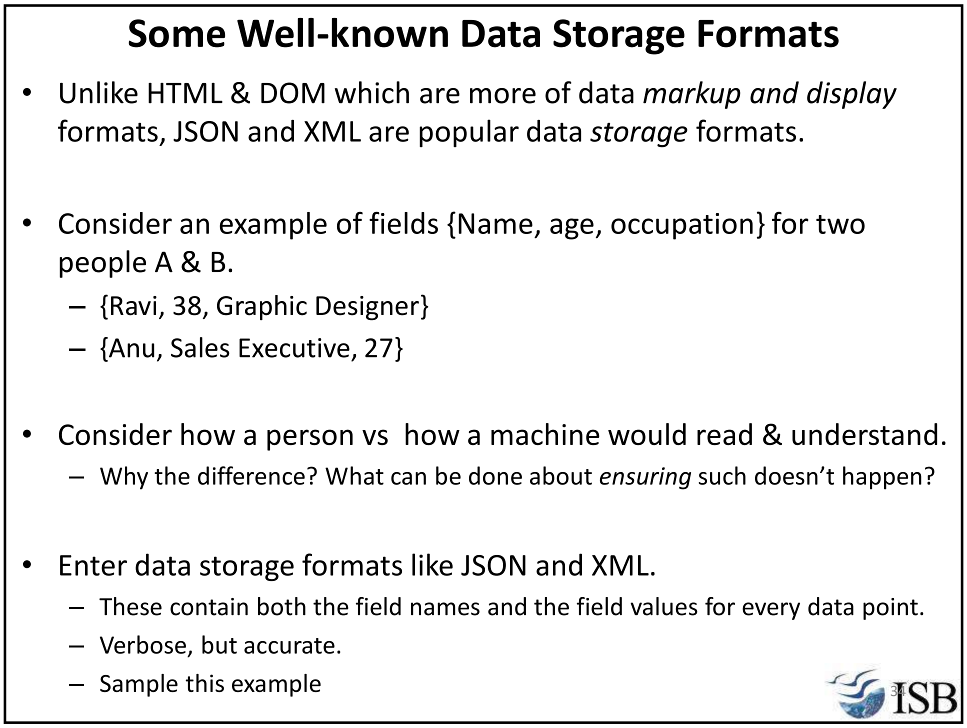
32



A Few More Preliminaries

Data Display vs Data Storage formats

33



Some Well-known Data Storage Formats

- Unlike HTML & DOM which are more of data *markup and display* formats, JSON and XML are popular data *storage* formats.
- Consider an example of fields {Name, age, occupation} for two people A & B.
  - {Ravi, 38, Graphic Designer}
  - {Anu, Sales Executive, 27}
- Consider how a person vs how a machine would read & understand.
  - Why the difference? What can be done about *ensuring* such doesn’t happen?
- Enter data storage formats like JSON and XML.
  - These contain both the field names and the field values for every data point.
  - Verbose, but accurate.
  - Sample this example

34

JSON & XML Data Storage Formats

- Here’s a quick view of what JSON output looks like …
- Can you ID the field names (or ‘keys’) and values?
- And now a quick view of what XML output looks like …
- Can you ID the field names (or ‘keys’) and values?
- Note the ability to nest and build hierarchical data storage structures

Webdrivers for Web-scraping

Using Selenium in Py

- Open the HTML file 'Intro to webdrivers - Selenium in Py’
- Let's walk though it step by step.
- What I'll show next is fairly basic. However, ...
- If you're aware of alternatives, better ways to do the same thing etc., pls speak up and share with the class.

37

Using Webdrivers: Recap

- What are webdrivers and where are they most used?
- What modules did we invoke for using webdrivers?
- What main functions were called? What did they do?
- What further possibilities come to mind with webdriver use?
- Ready for some basic homework involving py and selenium?

38

Session Wrap-up

39

Session Wrap-up

- This was more of a ‘workshop’ session than the 2 preceding it.
- Expect the trend to continue here onwards for the rest of DC and for Text-analytics.
- Group formation issues remain.
- Any Qs or comments etc?

40