Browser Automation for Web Experiments & Analytics
a.k.a Doing Your Job Without a Mouse

Amanda Ludden, Harvard Business Review
June 26, 2018
Case 1: Agile Role Playing at Scale

Experiments on HBR.org often involve showing different things to visitors based on attributes such as subscription status, article topic, device type, location, and page scroll depth.
Case 1: Agile Role Playing at Scale

These combinations can be tedious to QA manually:

```r
expand.grid("User State" = c("Guest", "Subscriber"),
    "Region" = c("US", "EU"),
    "Article Topic" = c("Technology", "Leadership"),
    "Browser" = c("Chrome", "Firefox", "Edge"),
    "Device" = c("Desktop", "Mobile")) %>
datatable(options=list(pageLength=8, searching=FALSE, lengthChange=FALSE))
```

<table>
<thead>
<tr>
<th>User State</th>
<th>Region</th>
<th>Article Topic</th>
<th>Browser</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guest</td>
<td>US</td>
<td>Technology</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
<tr>
<td>Subscriber</td>
<td>US</td>
<td>Technology</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
<tr>
<td>Guest</td>
<td>EU</td>
<td>Technology</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
<tr>
<td>Subscriber</td>
<td>EU</td>
<td>Technology</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
<tr>
<td>Guest</td>
<td>US</td>
<td>Leadership</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
<tr>
<td>Subscriber</td>
<td>US</td>
<td>Leadership</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
<tr>
<td>Guest</td>
<td>EU</td>
<td>Leadership</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
<tr>
<td>Subscriber</td>
<td>EU</td>
<td>Leadership</td>
<td>Chrome</td>
<td>Desktop</td>
</tr>
</tbody>
</table>

Showing 1 to 8 of 48 entries  Previous 1 2 3 4 5 6 Next
Case 1: Solution

Checklist A
- Open Chrome
- Go to article
- Log in
- Spoof Canadian IP
- Newsletter signup
- Hit quota
- Scroll
- See creative
- Click stuff
- Close creative
Case 1: Solution

Checklist A
- Open Chrome
- Go to article
- Log in
- Spoof Canadian IP
- Newsletter signup
- Hit quota
- Scroll
- See creative
- Click stuff
- Close creative
Crash Course: R (& Python, et al) can...

# Do math
2+2+mean(cumsum(1:100))

## [1] 1721

# Do not math (or: less obvious math)
data(state)
grep("a\">", state.name, value=TRUE) %>% paste(collapse="", ")

## [1] "Alabama, Alaska, Arizona, California, Florida, Georgia, Indiana, Iowa, Louisiana, ..."

# Visualize data
library(ggplot2); library(ggthemes); library(dplyr)
ggplot(starwars, aes(x=mass)) + geom_histogram(binwidth=5) + theme_fivethirtyeight() + labs(title="Histogram: Star Wars Character Weights (kg)") + scale_x_continuous(breaks=seq(6/18,...)"
DIY: Scripted Screen Capture

- In a few lines of code, we can remotely control a browser and record the experience
- Prerequisites: Install R, RSelenium, ffmpeg

```r
library(RSelenium)
myBrowser <- rsDriver(browser = c("chrome"))
browserController <- myBrowser[['client']]

browserController$navigate("https://hbr.org/2018/07/managers-dont-have-all-the-answers?hideIntromercial=true")
pageBody <- browserController$findElement("tag name", "body")
dir.create("DIY_screenshots", recursive=TRUE)
browserController$screenshot(file = paste0("DIY_screenshots/screenshot_001.png"))

for(i in 2:30) {
  pageBody$sendKeysToElement(list("dummy", key="page_down"))
  browserController$screenshot(file = paste0("DIY_screenshots/screenshot", sprintf("_%03d", i)))
}
myBrowser$server$stop()

shell(paste0('cd ', normalizePath("DIY_screenshots/"),
            
            '& ffmpeg -y -framerate 6 -i "screenshot_%03d.png" -vcodec libx264 -b:v 2000k demo.mp4'))
```

- Optional prerequisite: Install RStudio Desktop (interface)
- See RSelenium documentation for more options
DIY: Scripted Screen Capture (demo)
Case 2: Shaking Black Boxes

- HBR.org is evaluating two search engine providers, which are currently both implemented on the site as part of an A/B test.
- To contextualize any differences in engagement or revenue, we should understand basic differences in the search engines themselves: How different are the search results they return?
Case 2: Solution

1. Scrape results for 200 baseline queries on both search engines
2. Compare % of article vs. product results returned
3. Realize one engine surfaces far fewer products but only resulted in a small drop in revenue during test period
Case 2: Solution

% Search Results by Type*
Based on first 10 results for top 20 HBR.org search terms, i.e., same 200 queries run through each engine.

*Data fabricated for this presentation
Crash Course: cURL and HTML

```
C:\Users\amanda.ludden>curl -k -L hbr.org
<!DOCTYPE html>
<!-- [if IE 8]>
<html class="ie8 ">
<! [endif]-->
<!-[if IE 9]>
<html class="ie9 ">
<! [endif]-->
<!-[if IEM]
<html class="">
<!-[endif]-->
<!-[if IE]>
<! [endif]-->
<html class="">
<!-[endif]-->

To improve the practice of management and its impact in a changing world.
est. 1922

<head>
<meta charset="utf-8" />
```
Crash Course: cURL and HTML
DIY: Web Scraping

library(httr); library(xml2); library(dplyr)

search_terms <- c("hbr tools", "leadership", "culture", "blockchain", "strategy", "case studies")

getSearchResultCount <- Vectorize(function(searchTerm) {
  GET("https://hbr.org/search?search-all", query = list(term = searchTerm)) %>%
    content("text") %>% read_html %>%
    xml_find_all("//search-stream/div/div/div/div[2]/div/div[1]/h3") %>% xml_text
})

getSearchResultCount(search_terms[1])

##         hbr tools
## " 4,409  Results"

dir.create("DIY_search_results")

getSearchResultCount(search_terms) %>% data.frame %>% write.csv("DIY_search_results/search_term_results.csv")
library(RSelenium)
myBrowser <- rsDriver(browser = c("chrome"))  # Configure the kind of browser you want
browserController <- myBrowser[['client']]  

browserController$setWindowSize(width = 1200, height = 800)

for(i in search_terms) {
  browserController$navigate("https://hbr.org/search?search-all&term=" %>%
paste0(i, "&hideIntromercial=true&Ntt=1")
  )
  browserController$screenshot(file = paste0("DIY_search_results/", i, ".png"))
}

myBrowser$server$stop()
Summary

- Selenium can control browsers
- cURL gets HTML from websites
- ffmpeg is a tool for recording/making video
- R, Python, or a language of your choice can bring together Selenium, cURL, and ffmpeg (and more) in one place
- Benefits of automating: easier debugging, can pass on to coworkers, run remotely, free up time for other things
Tips

- Anyone can run code! There is plenty of pre-fab code that you can use as-is or easily adapt for your needs
- Don't store passwords in plain text!
- Be a good robot: observe rate limits and robots.txt
Thanks!

amanda.ludden@hbr.org