Web Scrapping Presentation_2

Cambridge

January 18, 2015

Web Scraping part 2: Diggint in

Rolf Fredheim and Aiora Zabala

University of Cambridge

25/02/2014

Variables

uni <- "The University of Cambridge" uni

Paying tax:

9400 tax free

(20000-9440)*20/100 #OR: wage <- 20000 taxFree <- 9400 rate <- 20 (wage-taxFree)*rate/100

Functions without variables

```
printName <- function(){ print ("My name is Rolf Fredheim") }
printName()
```

e.g. for simulations

```
sillySimulation <- function(){ x1 <- runif(500,80,100) x2 <-
runif(500,0,100) v1 <- c(x1,x2) x3 <- runif(1000,0,100) df <-
data.frame(v1,x3) require(ggplot2) print(ggplot(df,
aes(v1,x3))+geom_point()+ggtitle("simulation of some sort")) }
```

Inserting variables

desperateTimes <- function(){ print(paste0("Rolf is struggling to finish his PhD on time. Time remaining: 6 months")) }

Name

desperateTimes <- function(name){ print(paste0(name ," is struggling to finish his PhD on time. Time remaining: 6 months“)) } desperateTimes(name=”Tom“)

Gender==========

desperateTimes <- function(name,gender=“m”){
if(gender==“m”){ pronoun=“his” }else{ pronoun=“her” }

print(paste0(name ," is struggling to finish “,pronoun,” PhD on time. Time remaining: 6 months“)) }
desperateTimes(name=”Tanya“,gender=”f“)

degree==========

```
desperateTimes <- function(name,gender="m",degree){
if(gender=="m"){ pronoun="his" }else{ pronoun="her" }

print(paste0(name ," is struggling to finish ",pronoun," ",degree,"
on time. Time remaining: 6 months")) }
desperateTimes(name="Rolf",gender="m","Mphil")
```

Days til deadline

require(lubridate) require(ggplot2) deadline <- as.Date("2014-09-01") daysLeft <- deadline-Sys.Date() totDays <- deadline-as.Date("2011-10-01") print(daysLeft) print(paste0("Rolf is struggling to finish his PhD on time. Days remaining:", as.numeric(daysLeft)))

part2==========

print(paste0("Percentage to go:",round(as.numeric(daysLeft)/as.numeric(totDays)*100))) df <- data.frame(days=c(daysLeft,totDays-daysLeft),lab=c("to go","completed")) gg-plot(df,aes(1,days,fill=lab))+geom_bar(stat="identity",position="fill")

```
timeToWorry <- function(){ require(lubridate) deadline <-
as.Date("2014-09-01") daysLeft <- deadline-Sys.Date() totDays <-
deadline-as.Date("2011-10-01") print(daysLeft) print(paste0("Rolf is
struggling to finish his PhD on time. Days remaining:",
as.numeric(daysLeft))) print(paste0("Percentage to
go:",round(as.numeric(daysLeft)/as.numeric(totDays)*100))) df <-
data.frame(days=c(daysLeft,totDays-daysLeft),lab=c("to
go","completed")) gg-
plot(df,aes(1,days,fill=lab))+geom_bar(stat="identity",position="fill")
}
```

Last week's code==========

require(rjson) url <- “http://stats.grok.se/json/en/201201/web_scraping” raw.data <- readLines(url, warn=“F”) rd <- fromJSON(raw.data) rd.views <- rd$daily_v iewsrd.views <- -unlist(rd.views)rd <- -as.data.frame(rd.views)rddate <- rownames(rd) rownames(rd) <- NULL rd

Turn it into a function

getData <- function(url){ require(rjson) raw.data <- readLines(url, warn="F") rd <- fromJSON(raw.data) rd.views <- rd$daily_v iewsrd.views < -unlist(rd.views)rd < -as.data.frame(rd.views)rd$date <- rownames(rd) rownames(rd) <- NULL rd$date < -as.Date(rd$date) return(rd) }

getData("http://stats.grok.se/json/en/201201/web_scraping")

Creating the URLS

getUrls <- function(y1,y2,term){ root <- “http://stats.grok.se/json/en/” urls <- NULL for (year in y1:y2){ for (month in 1:9){ urls <- c(urls,(paste(root,year,0,month,“/”,term,sep=“”))) }

```
    for (month in 10:12){
    urls <- c(urls,(paste(root,year,month,"/",term,sep=""))
    }
}
return(urls)

}
```

Put it together

create some URLS

urls <- getUrls(y1=2013,y2=2014,"Euromaidan")

get data for each of them and store that data

```
results=NULL for (url in urls){ results <-
rbind(results,getData(url)) } head(results)

ggplot(tail(results,100),aes(date,rd.views))+geom_line()
```

Download the web page==========

require(RCurl) require(XML)

url <- "http://en.wikipedia.org/wiki/Euromaidan" SOURCE <- getURL(url,encoding="UTF-8") #Download the page #this is a very very long line. Let's not print it. Instead: substring (SOURCE,1,200) PARSED <- htmlParse(SOURCE) #Format the html code d

Accessing HTML elements in R

xpathSApply(PARSED, "//h1")

xpathSApply(PARSED, "//h1",xmlValue)

Digging deeper==========

xpathSApply(PARSED, “//h3”,xmlValue)

========== length(xpathSApply(PARSED, "//a/@href"))

Get references

head(xpathSApply(PARSED, “//span[@class=’citation news’]”,xmlValue)) head(xpathSApply(PARSED, “//span[@class=’citation news’]/a/@href"))

links <- (xpathSApply(PARSED, “//span[@class=’citation news’]/a/@href")) browseURL(links[1])

XPath2

```
(xpathSApply(PARSED, "//*[@class='citation news'][17]/a/@*"))
(xpathSApply(PARSED, "//span[@class='citation
news'][17]/a/@*"))
```

XPath3

```
head(xpathSApply(PARSED,
“//span[starts-with(@class,’citation’)][17]/a/@href"))
head(xpathSApply(PARSED,”//span[contains(@class,’citation’)][17]/a/@hr
```

Example - BBC article==========

url <- “http://www.bbc.co.uk/news/world-europe-26333587” SOURCE <- getURL(url,encoding=“UTF-8”) # Specify encoding when dealing with non-latin characters PARSED <- htmlParse(SOURCE) (xpathSApply(PARSED, “//h1[@class=’story-header’]”,xmlValue)) (xpathSApply(PARSED, “//span[@class=’date’]”,xmlValue)) #Meta field for better formatting (xpathSApply(PARSED, “//meta[@name=’OriginalPublicationDate’]/@content"))

Make a scraper

```
bbcScraper <- function(url){ SOURCE <-
getURL(url,encoding="UTF-8") PARSED <- htmlParse(SOURCE)
title <- (xpathSApply(PARSED,
"//h1[@class='story-header']",xmlValue)) date <-
as.character(xpathSApply(PARSED,
"//meta[@name='OriginalPublicationDate']/@content"))
return(c(title,date)) }
```

test

lt==========

bbcScraper("http:
//www.bbc.co.uk/news/world-middle-east-26333533")
bbcScraper("http:
//www.bbc.co.uk/sport/0/football/26332893")

Adding exceptions

```
bbcScraper2 <- function(url){ title <- date=NA #Return empty values in case field not found SOURCE <- getURL(url,encoding="UTF-8") PARSED <- htmlParse(SOURCE) title=(xpathSApply(PARSED, "//h1[@class='story-header']",xmlValue)) date <- (xpathSApply(PARSED, "//meta[@name='OriginalPublicationDate']/@content")) if (is.null(date)){ date=(xpathSApply(PARSED,"//span[@class='date']",xmlValue)) } return(c(title,as.character(date))) }

bbcScraper2("http://www.bbc.co.uk/news/world-middle-east-26333533")
bbcScraper2("http://www.bbc.co.uk/sport/0/football/26332893")
```

Guardian

```
url <- "http://www.theguardian.com/commentisfree/2014/feb/25/how-much-cost-growers-bananas-68p-per-kilo"
SOURCE <- getURL(url,encoding="UTF-8") PARSED <- htmlParse(SOURCE) xpathSApply(PARSED, "//h1[contains(@itemprop,'headline')]",xmlValue)
xpathSApply(PARSED, "//a[@class='contributor']",xmlValue)
xpathSApply(PARSED, "//time[@itemprop='datePublished']",xmlValue)
xpathSApply(PARSED, "//time[@itemprop='datePublished']/@datetime")
xpathSApply(PARSED,"//a[@rel='tag']",xmlValue)
unique(xpathSApply(PARSED,"//a[@rel='tag']",xmlValue))
xpathSApply(PARSED,"//div[@id='article-body-blocks']",xmlValue)
xpathSApply(PARSED,"//div[@id='article-body-blocks']/p",xmlValue)
```

Guardian scraper

guardianScraper <- function(url){ SOURCE <- getURL(url,encoding="UTF-8") # Specify encoding when dealing with non-latin characters PARSED <- htmlParse(SOURCE) title <- xpathSApply(PARSED, "//h1[contains(@itemprop,'headline')]",xmlValue) author <- xpathSApply(PARSED, "//a[@class='contributor']",xmlValue) time <- xpathSApply(PARSED, "//time[@itemprop='datePublished']/@datetime") tags <- unique(xpathSApply(PARSED,"//a[@rel='tag']",xmlValue)) text <- xpathSApply(PARSED,"//div[@id='article-body-blocks']/p",xmlValue) return(list(title=title, author=author, time=time, tags=paste(tags,collapse="|") ,text=paste(text,collapse="|"))) }

Using the scraper

a <- guardianScraper(url) a[“title”] a[[“title”]] a[[“tags”]]

Example with dataframe.

url <- “http://www.theguardian.com/uk” SOURCE <- getURL(url,encoding=“UTF-8”) PARSED <- htmlParse(SOURCE) urls <- xpathSApply(PARSED, “//div[@class=’tabs-container’]//*/@href")

This is a bit tricky, apologies. There may be a better way

d <- lapply(urls,guardianScraper) data <- data.frame(matrix(unlist(d),ncol=5,byrow=T)) colnames(data) <- c("title","author","time","tags","text") as.character(data$tags)

Accessing this data later on==========

require(stringr) #return title of texts mentioning Chelsea
data[grep("Chelsea",data$tags),"title"] #return tags of texts mentioning Chelsea
unlist(str_split(data[grep("Chelsea",data$tags),"tags"],"\|"))

Advanced stuff: comments==========

url <- “http://discussion.theguardian.com/discussion/p/3n34d” SOURCE <- getURL(url,encoding=“UTF-8”) # Specify encoding when dealing with non-latin characters PARSED <- htmlParse(SOURCE) xpathSApply(PARSED, “//div[@class=’d2-body’][1]”,xmlValue) xpathSApply(PARSED, “//a[@class=’d2-username’]”,xmlValue)

Comments2

url <- “http://www.theguardian.com/commentisfree/2014/feb/25/how-much-cost-growers-bananas-68p-per-kilo” SOURCE <- getURL(url,encoding=“UTF-8”) PARSED <- htmlParse(SOURCE) links <- xpathSApply(PARSED, "//@href") shortUrl <- links[grep(“//gu\.”,links)][1] require(stringr) temp <- unlist(str_split(shortUrl,“/”)) target <- temp[length(temp)]

discussionURL <- paste0(“http://discussion.theguardian.com/discussion/p/”,target) SOURCE <- getURL(discussionURL,encoding=“UTF-8”) PARSED <- htmlParse(SOURCE) xpathSApply(PARSED, “//a[@class='d2-username']”,xmlValue)

unique(c(“r”,“fes”,“r”)) duplicated

author=“By Rolf” author <- gsub(“By”,“”,author) author

Solutions (1: Mirror)

MIRROR

```
url <- "http://www.mirror.co.uk/news/world-news/
oscar-pistorius-trial-murder-reeva-3181393" SOURCE
<- getURL(url,encoding="UTF-8") PARSED <-
htmlParse(SOURCE) title <- xpathSApply(PARSED,
"//h1",xmlValue) author <- xpathSApply(PARSED,
"//li[@class='author']",xmlValue) time <- xpathSApply(PARSED,
"//time[@itemprop='datePublished']/@datetime")
```

Telegraph

```
url <- "http://www.telegraph.co.uk/news/uknews/
terrorism-in-the-uk/10659904/
Former-Guantanamo-detainee-Moazzam-Begg-one-of-four-arrest
html" SOURCE <- getURL(url,encoding="UTF-8") PARSED <-
htmlParse(SOURCE) title <- xpathSApply(PARSED,
"//h1[@itemprop='headline name']",xmlValue) author <-
xpathSApply(PARSED, "//p[@class='bylineBody']",xmlValue) time
<- xpathSApply(PARSED, "//p[@class='publishedDate']",xmlValue)
```

Independent

```
url <- "http://www.independent.co.uk/news/world/asia/
leopard-on-the-loose-puts-indian-city-of-meerut-on-lockdow
html" SOURCE <- getURL(url,encoding="UTF-8") PARSED <-
htmlParse(SOURCE) title <- xpathSApply(PARSED,
"//h1",xmlValue) author <- xpathSApply(PARSED,
"//span[@class='authorName']",xmlValue) time <-
xpathSApply(PARSED, "//p[@class='dateline']",xmlValue)
```