

Live Session 2: Getting Data

Zak Varty

Week in Review

Tabular data

- Reading with base R and {readr}
- Tibbles
- Tidy data, wide data and tall data

Web Scraping

- Intro to HTML and CSS
- {rvest} for scraping webpages and extracting content
- Amazon task (review to come)

APIs

- Ways of sharing and sourcing data
- HTTP requests and responses
- Use wrappers where you can

Discussion

Question 1: RDS files

-
1. Roger Peng states that files can be imported and exported using `readRDS()` and `saveRDS()` for fast and space efficient data storage. What is the downside to doing so?
-

2. What data types have you come across (that we have not discussed already) and in what context are they used?
-
-

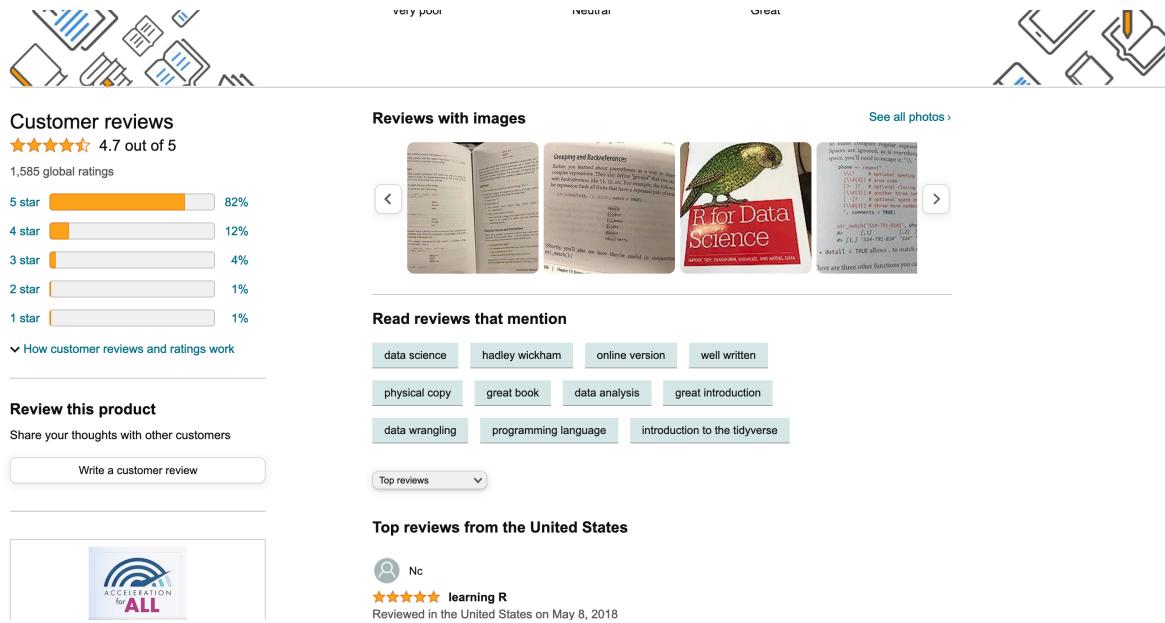
3. What do you have to give greater consideration to when scraping data than when using an API?
-

Scraping Book Reviews

Scrape R4DS Star Rating Percentages

```
library("rvest")
library("httr")
library("magrittr")
```

Visiting the R for Data Science webpage and scrolling down we find the review summaries giving the percentage of reviewers in each category.



The screenshot shows the product page for "R for Data Science". At the top, there's a navigation bar with icons for search, user profile, and cart. Below it, there are three tabs: "very poor", "neutral", and "great". On the left, there's a section titled "Customer reviews" with a 4.7 out of 5 star rating and 1,585 global ratings. It includes a horizontal bar chart showing the distribution of star ratings: 5 star (82%), 4 star (12%), 3 star (4%), 2 star (1%), and 1 star (1%). Below this is a link to "How customer reviews and ratings work". In the center, there's a "Reviews with images" section featuring a grid of images including a book cover, a parrot, and a person holding a book. To the right, there's a "See all photos" link. At the bottom, there's a "Read reviews that mention" section with tags like "data science", "hadley wickham", "online version", etc. A sidebar on the right contains R code for scraping reviews, including functions like `get_reviews`, `review_text`, and `reviewer`. At the very bottom, there's a "Top reviews from the United States" section with a review by "Nc" dated May 8, 2018.

Using the httr selector gadget we can identify that the elements we want to scrape are given by

Customer reviews

★★★★★ 4.7 out of 5

1,585 global ratings

Rating	Percentage
5 star	82%
4 star	12%
3 star	4%
2 star	1%
1 star	1%

How customer reviews and ratings work

Review this product

Share your thoughts with other customers

Write a customer review

Reviews with images

See all photos >

Read reviews that mention

- data science
- hadley wickham
- online version
- well written
- physical copy
- great book
- data analysis
- great introduction
- data wrangling
- programming language
- introduction to the tidyverse

Top reviews from the United States

Nc
★★★★★ learnin... .a-text-right .a-link-normal

Reviewed in the United States on May 8, 2010

Clear (5) Toggle Position XPath Help X

We first scrape the entire page.

```
r4ds_url <- "https://www.amazon.com/dp/1491910399/"
r4ds_html <- rvest::read_html(r4ds_url)
```

Then we can use Rvest functions to extract the elements that we care about and convert these to strings.

```
data_strings <- r4ds_html %>%
  #rvest::html_elements(".a-nnowrap .a-link-normal") %>%
  rvest::html_elements(".a-text-right .a-link-normal") %>%
  rvest::html_text2()

data_strings
```

```
[1] "82%" "12%" "4%" "1%" "1%"
```

Finally, we want to drop the percentage sign from each element of the vector and convert this to a vector of integers, rather than strings.

```

data_values_as_character <- stringr::str_sub(data_strings, start = 1, end = -2)
data_values <- as.integer(data_values_as_character)
data_values

```

```
[1] 82 12 4 1 1
```

Scrape R4DS Number of Ratings

Similarly, we can scrape the number of reviews using the selectors

```
.averageStarRatingNumerical .a-color-secondary
```

Customer reviews

★★★★★ 4.7 out of 5

1,586 global ratings

Star Rating	Percentage
5 star	82%
4 star	12%
3 star	4%
2 star	1%
1 star	1%

How customer reviews and ratings work

Review this product

Share your thoughts with other customers

Write a customer review

Reviews with images

See all photos >

Read reviews that mention

data science hadley wickham online version well written
physical copy great book data analysis great introduction
data wrangling programming language introduction to the tidyverse

Top reviews

Top reviews from the United States

Nc
★★★★★ learning R
Reviewed in the United States on April 1, 2015
Verified Purchase

Clear (1) Toggle Position XPath Help X

https://www.amazon.com/product-reviews/1491910399/ref=acr_dp_hist_4?ie=UTF8&filterByStar=four_star&reviewerType=all_reviews#reviews-filter-bar

We extract the text element in the same way as before.

```

r4ds_review_count <- r4ds_html %>%
  rvest::html_elements(".averageStarRatingNumerical") %>%
  rvest::html_text2()

r4ds_review_count

```

```
[1] "1,586 global ratings"
```

To convert this to an integer we can work with, we first drop the 15 characters " global ratings" from the end.

```
r4ds_review_count <- r4ds_html %>%
  rvest::html_elements(".averageStarRatingNumerical .a-color-secondary") %>%
  rvest::html_text2() %>%
  stringr::str_sub(start = 1, end = -16)

r4ds_review_count

[1] "1,586"
```

The last things we need to do is get rid of the comma and convert this to an integer.

```
r4ds_review_count <- r4ds_html %>%
  rvest::html_elements(".averageStarRatingNumerical .a-color-secondary") %>%
  rvest::html_text2() %>%
  stringr::str_sub(start = 1, end = -16) %>%
  stringr::str_split_1(",") %>%
  stringr::str_flatten() %>%
  as.integer()

r4ds_review_count

[1] 1586
```

Summary table R4DS reviews

```
r4ds_data <- tibble::tibble(
  product = "R4DS",
  n_reviews = r4ds_review_count,
  percent_5_star = data_values[1],
  percent_4_star = data_values[2],
  percent_3_star = data_values[3],
  percent_2_star = data_values[4],
  percent_1_star = data_values[5],
  url = r4ds_url)

r4ds_data
```

```

# A tibble: 1 x 8
  product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
  <chr>      <int>          <int>          <int>          <int>          <int>
1 R4DS        1586            82             12              4              1
# i 2 more variables: percent_1_star <int>, url <chr>

```

Making this a function

Let's abstract out the URL and product name to turn this into a function.

```

get_amazon_reviews <- function(product_name, url){

  # Scrape Amazon page of product
  product_html <- rvest::read_html(url)

  # Extract percentage receiving each number of stars
  review_percentages <- product_html %>%
    rvest::html_elements(".a-text-right .a-link-normal") %>% # extract information
    rvest::html_text2() %>% # convert to text
    stringr::str_sub(start = 1, end = -2) %>% # remove "%" from string
    as.integer() # convert to integer

  # Extract total number of reviews
  review_count <- product_html %>%
    rvest::html_elements(".averageStarRatingNumerical .a-color-secondary") %>%
    rvest::html_text2() %>%
    stringr::str_sub(start = 1, end = -16) %>%
    stringr::str_split_1(",") %>%
    stringr::str_flatten() %>%
    as.integer()

  # Construct Tibble
  product_data <- tibble::tibble(
    product = product_name,
    n_reviews = review_count,
    percent_5_star = review_percentages[1],
    percent_4_star = review_percentages[2],
    percent_3_star = review_percentages[3],
    percent_2_star = review_percentages[4],
    percent_1_star = review_percentages[5],
    url = url)
}

```

```
product_data  
}
```

We can test that this works for R4DS.

```
get_amazon_reviews("R4DS", url = r4ds_url)  
  
# A tibble: 1 x 8  
#> product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star  
#> <chr>      <int>          <int>          <int>          <int>          <int>  
#> 1 R4DS       1586           82            12             4              1  
#> i 2 more variables: percent_1_star <int>, url <chr>
```

This function is doing a lot, let's move some of the stages out to helper functions. This will make life easier for us if the structure of the webpages change over time and also if we need to debug the function.

We will have one function to extract the review percentages from the scraped html.

```
extract_review_percentages <- function(scraped_html, css_selector = ".a-text-right .a-link  
scraped_html %>%  
rvest::html_elements(css_selector) %>% # extract information  
rvest::html_text2() %>% # convert to text  
stringr::str_sub(start = 1, end = -2) %>% # remove "%" from string  
as.integer()  
}
```

A second function to extract the review count from the scraped html.

```
extract_review_count <- function(scraped_html, css_selector = ".averageStarRatingNumerical  
scraped_html %>%  
rvest::html_elements(".averageStarRatingNumerical .a-color-secondary") %>%  
rvest::html_text2() %>%  
stringr::str_sub(start = 1, end = -16) %>%  
stringr::str_split_1(",") %>%  
stringr::str_flatten() %>%  
as.integer()  
}
```

And a third function to assemble this information into a tibble.

```

construct_product_review_tibble <- function(product_name, url, review_count, review_percentages) {
  tibble::tibble(
    product = product_name,
    n_reviews = review_count,
    percent_5_star = review_percentages[1],
    percent_4_star = review_percentages[2],
    percent_3_star = review_percentages[3],
    percent_2_star = review_percentages[4],
    percent_1_star = review_percentages[5],
    url = url)
}

```

Each of these can then be called from within an updated version of `get_amazon_reviews()`.

```

get_amazon_reviews <- function(product_name, url){

  # Scrape Amazon page of product
  product_html <- rvest::read_html(url)

  # Extract percentage receiving each number of stars
  review_percentages <- extract_review_percentages(product_html)

  # Extract total number of reviews
  review_count <- extract_review_count(product_html)

  # Construct Tibble
  construct_product_review_tibble(product_name, url, review_count, review_percentages)
}

```

Again, we should test that this still works.

```

get_amazon_reviews("R4DS", url = r4ds_url)

# A tibble: 1 x 8
#>   product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
#>   <chr>      <int>          <int>          <int>          <int>          <int>
#> 1 R4DS       1586            82             12              4              1
#> # i 2 more variables: percent_1_star <int>, url <chr>

```

We can also try it with the ggplot2 book

```
ggplot2_url <- "https://www.amazon.com/dp/331924275X"
get_amazon_reviews("ggplot2", url = ggplot2_url)
```

Warning in scraped_html %>% rvest::html_elements(css_selector) %>%
rvest::html_text2() %>% : NAs introduced by coercion

```
# A tibble: 1 x 8
  product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
  <chr>      <int>           <int>           <int>           <int>           <int>
1 ggplot2       160             NA              71              12              10
# i 2 more variables: percent_1_star <int>, url <chr>
```

The screenshot shows an Amazon product page for the book 'ggplot2'. At the top, there's a navigation bar with categories like 'plots', 'useful', 'advanced', 'graphs', 'introduction', 'reference', 'hadley', 'level', 'package', 'syntax', 'basics', 'coding', and 'date'. Below the navigation bar, there's a section titled 'Customer reviews' with a star rating of 4.4 out of 5 based on 160 global ratings. It includes a horizontal bar chart showing the percentage of reviews for each star rating: 5 star (71%), 4 star (12%), 3 star (10%), 2 star (4%), and 1 star (4%). There's also a link to 'How customer reviews and ratings work'. To the right, there's a section titled 'Read reviews that mention' with tags for 'ggplot2 is an excellent', 'learn ggplot', 'data', 'examples', 'learning', 'plots', 'useful', 'advanced', 'graphs', 'introduction', 'reference', 'hadley', 'level', 'package', 'syntax', 'basics', 'coding', and 'date'. Below that is a 'Top reviews' dropdown menu. Further down, there's a section titled 'Top reviews from the United States' featuring a review by Brandon Hurr. The review is for the 2nd edition and is highly rated. It mentions that the book has changed significantly and is no longer useful, as Hadley has rewritten it. The review is a 'Verified Purchase'.

Hooray! It works! How about the R packages?

```
r_packages_url <- "https://www.amazon.com/dp/1491910593/"
get_amazon_reviews("R packages", url = r_packages_url)
```

```
# A tibble: 1 x 8
  product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
  <chr>      <int>           <int>           <int>           <int>           <int>
1 R packa~       107            81              15               4              NA
# i 2 more variables: percent_1_star <int>, url <chr>
```

Once again, this has worked out.



But those NA values worry me. Let's take a look at where they are coming from.

```
r_packages_html <- rvest::read_html(r_packages_url)
extract_review_percentages(r_packages_html)
```

```
[1] 81 15 4
```

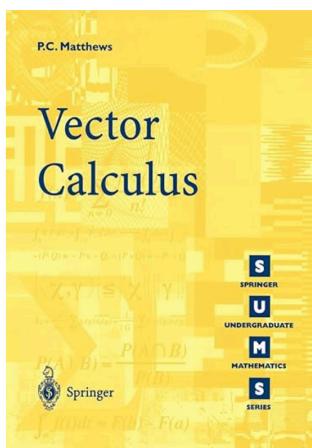
We only have three values being extracted. This is likely because only the non-zero values were clickable on the webpage. It seems we got lucky and those happened to be the first three, but what would have happened if that were not the case?

To find out, we need to identify a product which satisfies:

- (at least) one star category $x \in \{2, 3, 4, 5\}$ that has zero percent
- a second star category $y \in \{1, 2, 3, 4\}$ such that $y < x$ and y has non-zero percentage of reviews.

To get an empty star category, we can maximise our chances by looking at products with a low total number of reviews. Staying on topic, I decided to look at mathematics textbooks.

It took a bit of digging (lots of books received only 5-star and 4-star reviews) to find [Vector Calculus](#) which, *at the time of writing* has no 2-star reviews.



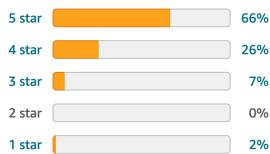
Roll over image to zoom in

[Read sample](#)

Customer reviews

★★★★★ 4.5 out of 5

55 global ratings



[▼ How customer reviews and ratings work](#)

Review this product

Share your thoughts with other customers

[Write a customer review](#)

Vector Calculus (Springer Undergraduate Mathematics Series) Paperback – Illustrated, 14 Jan. 1998

by Paul C. Matthews (Author)

4.5 ★★★★★ 55 ratings

[Part of: Springer Undergraduate Mathematics \(82 books\)](#)

[See all formats and editions](#)

Vector calculus is the foundation stone on which a vast amount of applied mathematics is based. Topics such as fluid dynamics, solid mechanics and electromagnetism depend heavily on the calculus of vector quantities in three dimensions. This book covers the material in a comprehensive but concise manner, combining mathematical rigour with physical insight. There are many diagrams to illustrate the physical meaning of the mathematical concepts, which is essential for a full understanding of the subject. Each chapter concludes with a summary of the most important points, and there are worked examples that cover all of the material. The final chapter introduces some of the most important applications of vector calculus, including mechanics and electromagnetism.

[Report an issue with this product](#)

ISBN-10	ISBN-13	Edition	Publisher	Publication date
3540761802	978-3540761808	# 1st ed. 1998. Corr. 3rd printing 2000	Springer	14 Jan. 1998

Read reviews that mention

vector calculus good book basic chapter course advanced

applications topics understanding easier exercises tensors

[Top reviews](#) ▾

Top reviews from United Kingdom

ab.c VINE VOICE

★★★★★ Satisfying exploration of essential part mathematical physics

Reviewed in the United Kingdom on 8 November 2011

Verified Purchase

Physical

The book is slim, (182 pages) and printed upon quality paper, but not the glossy kind. The font size is just the right size, so those requiring reading glasses will not struggle.

```
vector_calc_url <- "https://www.amazon.co.uk/dp/3540761802"
get_amazon_reviews(product_name = "vector calculus", url = vector_calc_url)
```

```
# A tibble: 1 x 8
  product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
  <chr>      <int>           <int>           <int>           <int>           <int>
1 vector ~       55            66            26             7            2
# i 2 more variables: percent_1_star <int>, url <chr>
```

As we suspected - the one star reviews are misplaced.

I spent a long time trying to get workarounds, but missing values are tricky to deal with. I got some code working, but it was very clunky and involved using `try()` within a `for` loop.

A much simpler solution is to return to the Selector gadget and update our CSS selectors within the extraction function.

The screenshot shows the Selector gadget interface. At the top, there are icons for various selection types like 'Customer reviews' (star rating), 'Read reviews that mention' (tags like 'vector calculus', 'good book', etc.), and 'Top reviews' dropdown. Below this, the 'Top reviews from United Kingdom' section displays a review by 'ab..c VINE VOICE' with a 5-star rating and the title 'Satisfying exploration of essential part mathematical physics'. The review text discusses the book's physicality and readability. On the left, there's a 'Customer reviews' summary with a 4.5 out of 5 rating and a star distribution chart. A 'Review this product' section allows users to share their thoughts. At the bottom, there's a product image for 'GrinGrab SuperGel' gloves with a 'Shop now >' button. The CSS selector used is `#histogramTable .a-text-right .a-size-base`.

This more careful selection gives the following CSS selector:

```
#histogramTable .a-text-right .a-size-base
```

We can use this to update the default value in `extract_review_percentages()`

```
extract_review_percentages <- function(scraped_html, css_selector = "#histogramTable .a-text-right .a-size-base") {
  scraped_html %>%
    rvest::html_elements(css_selector) %>%
    rvest::html_text2() %>%
    stringr::str_sub(start = 1, end = -2) %>%
    as.integer()
}
```

This works for our vector calculus example.

```
get_amazon_reviews(product_name = "vector calculus", url = vector_calc_url)

# A tibble: 1 x 8
product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
```

```
<chr>      <int>      <int>      <int>      <int>      <int>
1 vector ~      55       66       26        7        0
# i 2 more variables: percent_1_star <int>, url <chr>
```

It corrects also corrects our output for the R packages example to be 0 rather than NA,

```
get_amazon_reviews(product_name = "R packages", url = r_packages_url)

# A tibble: 1 x 8
product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
<chr>      <int>      <int>      <int>      <int>      <int>
1 R packa~     107       81       15        4        0
# i 2 more variables: percent_1_star <int>, url <chr>
```

and it has not broken any of our complete examples

```
get_amazon_reviews(product_name = "R4DS", url = r4ds_url)

# A tibble: 1 x 8
product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
<chr>      <int>      <int>      <int>      <int>      <int>
1 R4DS       1586       82       12        4        1
# i 2 more variables: percent_1_star <int>, url <chr>
```

```
get_amazon_reviews(product_name = "ggplot2", url = ggplot2_url)

# A tibble: 1 x 8
product n_reviews percent_5_star percent_4_star percent_3_star percent_2_star
<chr>      <int>      <int>      <int>      <int>      <int>
1 ggplot2     160        71        12        10        4
# i 2 more variables: percent_1_star <int>, url <chr>
```

Discussion

- What did you do differently to me?
- What was easy, what was difficult?
- How could we formalise and automate this testing workflow? What might be make this difficult?