Text analysis workshop: Getting data from PDFs

Casey O'Hara

Overview

How often have you run across a published paper with awesome-looking data, but it is only available in PDF format? ARGHHH! But using the `pdftools` package and some `stringr` functions with regex patterns, we can get that data out and into a usable format.

```
library(tidyverse)
library(stringr)

library(pdftools)

# ?pdftools
```

The `pdftools` package

The pdftools package basically has five functions:

- `pdf_info(pdf, opw = "", upw = "")` to get metadata about the pdf itself
- `pdf_text(pdf, opw = "", upw = "")` to get the text out of the pdf.
- `pdf_fonts(pdf, opw = "", upw = "")` to find out what fonts are used (including embedded fonts)
- `pdf_attachments(pdf, opw = "", upw = "")` umm attachments I guess?
- `pdf_toc(pdf, opw = "", upw = "")` and a table of contents.

Really we'll just focus on `pdf_text()`.

```
pdf_smith <- file.path('pdfs/smith_wilen_2003.pdf')

smith_text <- pdf_text(pdf_smith)
```

`pdf_text()` returns a vector of strings, one for each page of the pdf. So we can mess with it in tidyverse style, let's turn it into a dataframe, and keep track of the pages.

Then we can use `stringr::str_split()` to break the pages up into individual lines. Each line of the pdf is concluded with a backslash-n, so split on this. We will also add a line number in addition to the page number.

```
smith_df <- data.frame(text = smith_text)

smith_df <- data.frame(text = smith_text) %>%
  mutate(page = 1:n()) %>%
  mutate(text_sep = str_split(text, '\\n')) %>%
  unnest(text_sep)

smith_df <- data.frame(text = smith_text) %>%
  mutate(page = 1:n()) %>%
  mutate(text_sep = str_split(text, '\\n')) %>%
  unnest(text_sep) %>%
  group_by(page) %>%
  mutate(line = 1:n()) %>%
  ungroup()
```

Getting the table out of the PDF

Let's look at the PDF, specifically we want to get the data in the table on page 8 of the document. More specifically, the table data is in lines 8 to 18 on page 8. This is a table comparing the number of active urchin divers to the number of patches in which they dove for urchins, from 1988 to 1999.

- The column headings are annoyingly just years, and R doesn't like numbers as column names, so we'll rename them as 'y####' for year.
- Break up the columns separated by (probably tabs but we can just use) spaces. We'll use the `tidyr::separate` function to separate into columns by spaces. Note, one space and multiple spaces should count the same way - how to do that in regex?

```
### We want to extract data from the table on page 8
page8_df <- smith_df %>%
  filter(page == 8)

### Let's just brute force cut out the table
col_lbls <- c('n_patches', paste0('y', 1988:1999))

table1_df <- page8_df %>%
  filter(line %in% 8:18) %>%
  separate(text_sep, col_lbls, ' +')
```

Now we can ditch the `text`, `page`, and `line` columns, and pull the result into a tidy format (long format rather than wide) for easier operations.

- When we pull the 'y####' columns into a year column, let's turn those into integers instead of a character
- Same goes for the number of patches and number of divers - they're all character instead of integer (because it started out as text).

```
table1_tidy_df <- table1_df %>%
  select(-text, -line, -page) %>%
  gather(year, n_divers, starts_with('y')) %>%
  mutate(year = str_replace(year, 'y', ''), ### or str_extract(year, '[0-9]{4}')
         year = as.integer(year),
         n_patches = as.integer(n_patches),
         n_divers = as.integer(n_divers))
```

With pdftools, we extracted a data table, but we could also just extract the text itself if that's what we really wanted. We'll do this in the next exercise, but first:

`tabulizer` package!

Another package for extracting table data from pdfs is the `tabulizer` package from ROpenSci. This takes advantage of Java functionality, so you will need Java installed on your computer for it to work. It is pretty finicky, and the results seem so-so in terms of formatting, requiring a lot of manual cleanup. The one advantage is that it automatically pulls ALL tables from a paper without having to know page/line numbers, etc - so it might be great if you need to automate table extraction from dozens of papers.

Installing Java

I downloaded/installed the JDK (includes JRE): http://www.oracle.com/technetwork/java/javase/downloads/index.html

It's finicky so if it doesn't work properly right away, here are a couple of notes I found through the Googles that helped:

- in R: `dyn.load(paste0(system2('/usr/libexec/java_home', stdout = TRUE), '/jre/lib/server/libjvm.dyli`
- in terminal: `R CMD javareconf`
- Once it's working, the `rJava` should load:
  - `# install.packages('rJava')`
  - `library(rJava)` should not return any errors.

Note that the `tabulizer::extract_tables()` function returned some warnings, but still returned the tables as well. Compare table 1 (first item in the list) to our results using `pdftools`...

```
# devtools::install_github(c("ropensci/tabulizerjars"))
# devtools::install_github(c("ropensci/tabulizer"))
library(tabulizer)

pdf_smith <- file.path('pdfs/smith_wilen_2003.pdf')

tables_list <- tabulizer::extract_tables(pdf_smith)

tab1_df <- tables_list[[1]] %>%
  as.data.frame(stringsAsFactors = FALSE)
```