Extracting data from PDF files using the `pdftools` package

LJ Valencia*

Introduction

This paper shows how to extract data from PDF files using the `pdftools` package, clean the extracted information, convert it into a dataframe, and then export this dataframe as an excel file. There are alternative packages that could be used to scrape PDF files such as the `tabulizer` package. However, `tabulizer` requires Java while `pdftools` does not. I use the `pdftools` package to avoid debugging and the hassle of additional software installations.[1] A few R packages are used in this demonstration: `tidyverse`, `pdftools`, `readr`, `dplyr`, and `writexl`.

Data Files

The data file used for this demonstration is a time-series, historical data of annual mineral production in Canada from Natural Resources Canada (NRCAN).[2]

Importing Packages

The relevant packages are loaded using the `library()` function.

```
# import packages
library(tidyverse)
library(writexl)
library(pdftools)
```

Scraping Process

The pdf_text() function extracts text from the PDF file. The read_lines() function from the `readr` package is used to read the lines extracted using the pdf_text() function. To see what the pdf file looks like, I used the print(PDF) function to do a preliminary assessment of the raw data.

```
file <- "mineral-prod-canada-2018.pdf" #filename
PDF <- pdf_text(file) %>% readr::read_lines() # call read_lines() function to read lines
print(PDF) # print PDF to show lines
```

*Bachelor of Arts (Honors) in Economics, University of Alberta

[1] This article demonstrates the somewhat tedious installation process for Java when using `tabulizer`: https://blog.az.sg/posts/reading-pdfs-in-r/

[2] Mineral production in Canada, 2018, link: https://mmsd.nrcan-rncan.gc.ca/PDF/MIS2018TableG01c-en.pdf

At first glance, there are some details that are needed to be removed to create a clean dataframe, this is done in the next line of code below. I used a for-loop to ensure row length consistency in the raw data. This allows for easy conversion to a dataframe. The need to address lenght consistency arose from missing/suppressed observations in the original table which was skipped during data extraction via the PDF function.

```
PDF <- PDF[-c(1:5,37:46)] # remove the lines
all_stat_lines <- PDF[1:30] %>%
  str_squish() %>% # str_squish() function removes the whitespace in the strings
  strsplit(split = " ") # remove empty spaces
# Use a for-loop to enable length consistency when converting to a dataframe
for (i in 1:length(all_stat_lines)){
  if (length(all_stat_lines[[i]]) < 18){
    all_stat_lines[[i]][18] <- all_stat_lines[[i]][17] # add a new point;
    all_stat_lines[[i]][17] <- "NA" # fill as NA
  }
  else{
    print("Consistent character length.") # else print "Consistent character length."
  }
}
```

The dataframe conversion is done using the plyr function. Then I used the unite() to merge some of the columns and make them appear like the original pdf file. Doing this is a matter of personal preference and depends on a user's desired specifications with the data. I created a list of variable names to define the column names of the dataframe.

```
df <- plyr::ldply(all_stat_lines) # create a data frame using the ldply() function
head(df) # show first few observations
df <- df %>% unite(V1.2, V1, V2, V3, sep = " ") # unite rows
# variable names
var_lines <- c("Year",
               "Units",
               "NL",
               "PEI",
               "NS",
               "NB",
               "QC",
               "ON",
               "MB",
               "SK",
               "AB",
               "BC",
               "YT",
               "NWT",
               "NU",
               "Canada") # create your variable names
# change column names
colnames(df) <- var_lines
# export dataframe as excel file
write_xlsx(df, path="mineral-production-2018.xlsx") # export data
```

Full Implementation

The code below shows the full implementation.

```
# import packages
library(tidyverse)
library(writexl)
library(pdftools)

file <- "mineral-prod-canada-2018.pdf" #filename
PDF <- pdf_text(file) %>% readr::read_lines() # call read_lines() function to read lines
print(PDF) # print PDF to show lines
PDF <- PDF[-c(1:5,37:46)] # remove the lines

all_stat_lines <- PDF[1:30] %>%
  str_squish() %>% # str_squish() function removes the whitespace in the strings
  strsplit(split = " ") # remove empty spaces
# Use a for-loop to enable length consistency when converting to a dataframe
for (i in 1:length(all_stat_lines)){
  if (length(all_stat_lines[[i]]) < 18){
    all_stat_lines[[i]][18] <- all_stat_lines[[i]][17] # add a new point;
    all_stat_lines[[i]][17] <- "NA" # fill as NA
  }
  else{
    print("Consistent character length.") # else print "Consistent character length."
  }
}

df <- plyr::ldply(all_stat_lines) # create a data frame using the ldply() function
head(df) # show first few observations
df <- df %>% unite(V1.2, V1, V2, V3, sep = " ") # unite rows
# variable names
var_lines <- c("Year",
               "Units",
               "NL",
               "PEI",
               "NS",
               "NB",
               "QC",
               "ON",
               "MB",
               "SK",
               "AB",
               "BC",
               "YT",
               "NWT",
               "NU",
               "Canada") # create your variable names
# change column names
colnames(df) <- var_lines
# export dataframe as excel file
write_xlsx(df, path="mineral-production-2018.xlsx") # export data
```

Conclusion

I have demonstrated how to extract data from a PDF File using a variety of functions. This is only a demonstration and can provide inspiration in crafting unique solutions relevant to data extraction in PDF files.

References

"Mineral Production in Canada, 2018." 2020. *Natural Resources Canada*. Natural Resources Canada. https://mmsd.nrcan-rncan.gc.ca/MIS/MISTable.aspx?FileT=G01&Year=2018.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.