```
---
title: "R Scraping Demonstration"
author: "Trinh Nguyen"
institute: "IEP, CDFW"
format:
  revealjs:
    code-line-numbers: true
---

```{r}
library(kableExtra)
library(dplyr)
library(tabulizer)
library(tidyr)
library(stringr)
```

Purpose: There are several PDF scraping packages in R. I will focus on two:

1. `pdftools`
2. `tabulizer`

Pros and Cons

-   The main reason to consider this approach: cost. It is free.

::: incremental
-   The main reasons to **NOT** consider this approach: nearly everything else:
    1. very code heavy
    2. no support
    3. scrapes only text-based data
    4. `tabulizer` can be difficult to install if you run into Java restrictions
:::

Outline

1. Primer to 'pdftools'
2. Primer to 'tabulizer'
3. Case study: Bay Study datasheet
4. Case study: DGS Pay Period Tables
5. Conclusion

Primer to 'pdftools'

-   Primarily used to extract **ONLY** text-based data from a pdf
-   Will read back all text as a singular string (ignoring structure)

```{r}
#| echo: true
pdftools::pdf_text("https://water.ca.gov/-/media/DWR-Website/Web-Pages/Programs/State-Water-Project/Operations-And-Maintenance/Files/Operations-Control-Office/Delta-Status-And-
```
```

```
Operations/Delta-Operations-Daily-Summary.pdf")
```

Primer to 'tabulizer'

- Used to extract **ONLY** text-based data from a pdf
- Can extract formatted tabular data as tables
- However, tables must be formatted as tables

Case study: Bay Study

Bay Study Datasheet

Workflow: Bay Study

1. If the data is tabular, use `extract_table()` from `tabulizer`
2. Non-tabular data will require more creative approaches

Workflow: extracting tabular data

```{r}
#| echo: true
extract_tables(file.path("Bay Study field sheet front_fill2_data.pdf"))
```

Workflow: extracting tabular data

`extract_tables()` has two algorithms to try and scrape tables. Can try both

```{r}
#| eval: false
#| echo: true
extract_tables(file.path("pdfRead", "bayStudyExample.pdf"), method =
"lattice")
extract_tables(file.path("pdfRead", "bayStudyExample.pdf"), method =
"stream")
```


::: {.fragment}
Does not work still!
:::

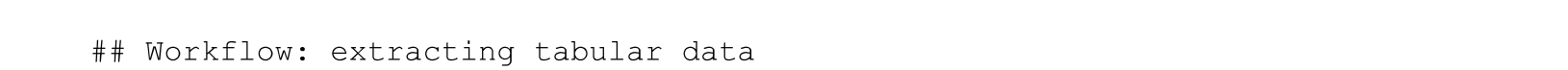
Workflow: extracting tabular data

One final way is to specify the specific location of the data on the pdf.

```{r}
#| eval: false
#| echo: true
```

```
locate_areas(file.path("Bay Study field sheet front_fill2_data.pdf"))
```

Workflow: extracting tabular data {.scrollable}



Workflow: extracting tabular data

Once selected:

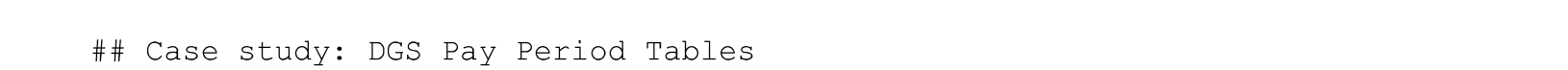
Workflow: extracting tabular data

```{r}
#| echo: true
table <- extract_tables(file.path("bayStudyExample.pdf"), pages = 1,
                        area = list(c(176.8358, 35.8913, 393.9738,
601.7659)),
                        guess = F, output = "data.frame")
table
```

Workflow: extracting tabular data
```{r}
#| echo: true
#| output-location: slide
table[[1]] %>%
  .[-1, ] %>%
  setNames(c("Species", "S", "Plus", "Specimen1", "Specimen2", "Specimen3",
             "Specimen4", "Specimen5")) %>%
  data.frame() %>%
  pivot_longer(c("Specimen1", "Specimen2", "Specimen3",
                 "Specimen4", "Specimen5"),
               # contains("Specimen"),
               names_to = "Specimen", values_to = "Length") %>%
  mutate(Sex = str_extract(Length, "[:upper:]"),
         Length = str_extract(Length, "\\d+"),
         across(c(S, Plus, Length),
                as.numeric)) %>%
  filter(!(is.na(S) & is.na(Plus) & is.na(Length) & is.na(Sex)))
```

Case study: DGS Pay Period Tables

`tabulizer` works well *IF* the datasheet is set up correctly.



Case study: DGS Pay Period Tables

```{r}
#|echo: true
```

```
#|output-location: slide
tables <- extract_tables(file.path("PayPeriodsDGS.pdf"), method = "stream")
tables
```

Case study: DGS Pay Period Tables

Easily workable, e.g., pay period information for the current month:

```{r}
masterTableLocation <- which(sapply(tables, function(x) x[1, 1] == "Year"))
# Cleaning up the tables
# Table 6 is the master table showing which schedule table each year will
follow:
masterTable <- data.frame(year = c(tables[[masterTableLocation]][1, 2:12],
                                   tables[[masterTableLocation]][4, 2:12],
                                   tables[[masterTableLocation]][7, 2:12]),
                          pattern = c(tables[[masterTableLocation]][2,
2:12],
                                      tables[[masterTableLocation]][5,
2:12],
                                      tables[[masterTableLocation]][8,
2:12])) %>%
  mutate(across(everything(), as.numeric)) %>%
  filter(!is.na(year))

priorityTable <- lapply(tables[-masterTableLocation], function(x) {

  if (grepl("PATTERN", x[1, 1])) {
    data.frame(x) %>%
      mutate(pattern = gsub("[^0-9]", "", X1[1])) %>%
      filter(X2 %in% c(month.name)) %>%
      separate(X3, into = c("startDate", "endDate"), sep = " to ") %>%
      transmute(pattern, month = X2, startDate, endDate, numberPaydays =
X4)
  } else {
    data.frame(x) %>%
      filter(X1 %in% c(month.name)) %>%
      separate(X2, into = c("startDate", "endDate"), sep = " to ") %>%
      transmute(month = X1, startDate, endDate, numberPaydays = X3)
  }
}) %>%
  bind_rows() %>%
  fill(pattern, .direction = "down")

priorityTable %>%
  filter(pattern == (masterTable %>%
                       filter(year == format(Sys.time(), "%Y")) %>%
                       pull(pattern)),
         month == month.name[as.numeric(format(Sys.time(), "%m"))])
```

Conclusion {.smaller}

```
- Possible
- Automate with `tabulizer` and `pdftools`

::: {.fragment}
However, there are major drawbacks:

::: {.incremental}
  1. requires a non-trivial amount of coding knowledge in R
  2. can only scrape text-based data
  3. sensitive to structure of the datasheet
  4. no established workflow and dedicated features
  5. no support
  6. requires Java

:::
:::

## Pros and Cons

```{r}
data.frame("Features" = c("Ease of installation", "Ease of use", "Ease of
application", "Extract tables", "Extract text", "Automation", "Code
Knowledge", "Recommend"),
           "tabulizer" = c("Hard", "Medium", "Hard", "Yes", "Yes", "Yes",
"Medium High", "No"),
           "pdftools" = c("Easy", "Easy", "Very Hard", "No", "Yes", "Yes",
"Very High", "Very No")) %>%
  setNames(c("Features", "'tabulizer'", "'pdftools'")) %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",
"responsive"))
```

Questions
```