Last week, [Axios published a very interesting piece reporting on  
Trump’s private schedule thanks to an insider’s  
leak](https://www.axios.com/donald-trump-private-schedules-leak-executive-time-34e67fbb-3af6-48df-aefb-52e02c334255.html).  
The headlines all were about Trump’s spending more than 60% of his time  
in “executive time” which admittedly was indeed the most important  
aspect of the story. I, however, also got curious about Axios’ work to  
go from the PDF schedules to the spreadsheet they made public. In this  
post, I’ll have a got at using rOpenSci’s Jeroen Ooms’ pdftools  
package and some data-wrangling stubborness of mine to try and rectangle  
Trump’s PDF schedules.

**Available data, available software**

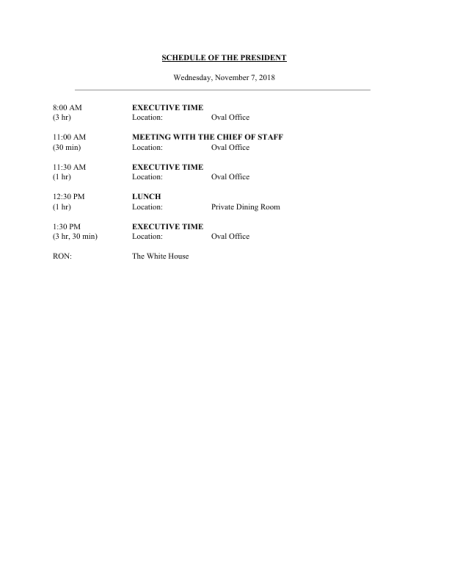
Axios published both [the spreadsheet they made from the data  
leaked](https://docs.google.com/spreadsheets/d/1oITCuVsYdhNXtY7GElLelsrbjRRIPJ1ce-_v-8J1X_A/edit#gid=0),  
and PDF corresponding to the schedules retyped in the format White House  
staff received them. I tried finding out how they went from the latter  
to the former, but this was apparently not reported. Quite a niche  
topic, I agree. Here’s the untidy PDF below.

pdf\_file <- "data/Axios-President-Donald-Trump-Private-Schedules.pdf"

library("magrittr")

pdftools::pdf\_render\_page(pdf\_file, page = 1) %>%

magick::image\_read()



Seeing the PDF format above reminded me of [the cool new pftools  
feature Jeroen Ooms announced back in  
December](https://ropensci.org/technotes/2018/12/14/pdftools-20/): low  
level text extraction.

# High level text extraction

pdftools::pdf\_text(pdf\_file)[1]

## [1] " SCHEDULE OF THE PRESIDENT\n Wednesday, November 7, 2018\n8:00 AM EXECUTIVE TIME\n(3 hr) Location: Oval Office\n11:00 AM MEETING WITH THE CHIEF OF STAFF\n(30 min) Location: Oval Office\n11:30 AM EXECUTIVE TIME\n(1 hr) Location: Oval Office\n12:30 PM LUNCH\n(1 hr) Location: Private Dining Room\n1:30 PM EXECUTIVE TIME\n(3 hr, 30 min) Location: Oval Office\nRON: The White House\n"

# Low level text extraction

head(pdftools::pdf\_data(pdf\_file)[[1]])

## width height x y space text

## 1 60 12 220 72 TRUE SCHEDULE

## 2 15 12 283 72 TRUE OF

## 3 23 12 301 72 TRUE THE

## 4 62 12 327 72 FALSE PRESIDENT

## 5 54 12 236 99 TRUE Wednesday,

## 6 46 12 293 99 TRUE November

While the pdftools::pdf\_text() function returns “a character vector of  
length equal to the number of pages in the file”, pdftools::pdf\_data  
returns a list of as many data.frame’s as there are pages, each one  
indicating the coordinates and width of each text piece. So powerful!  
Trump’s schedule appeared to be a perfect use case for this  
functionality!

**Special tale for Ubuntu users**

*Edit:* [*Benjamin Louis*](https://www.benjaminlouis-stat.fr/) *posted a link to* [*a helpful thread about installing Poppler’s latest version and pdftools on Ubuntu 18.04*](https://askubuntu.com/questions/1112856/how-to-install-poppler-0-73-on-ubuntu-18-04/1112947#1112947) *in the comments. Thank you!*

Sadly, as noted in Jeroen’s tech note, the new Poppler version that is  
necessary for pdftools to support low level text extraction… is not  
available on Linux yet, and I’ve just switched to Ubuntu from Windows  
(an accidental New Year’s resolution after Windows disappeared from my  
laptop). Jeroen wrote “Linux users probably have to wait for the latest  
version of poppler to become available in their system package manager  
(or compile from source).”. I thought, OK, let me *just* compile Poppler  
from source then. I’m not sure I’d recommend trying this at home so take  
my experience report with a bit of caution. I downloaded [Poppler’s  
latest version](https://poppler.freedesktop.org/#Download), unzipped it,  
created a build folder inside the poppler folder and ran something  
like this (inspired by [this  
gist](https://gist.github.com/Dayjo/618794d4ff37bb82ddfb02c63b450a81)):

cmake ..

make

sudo make install

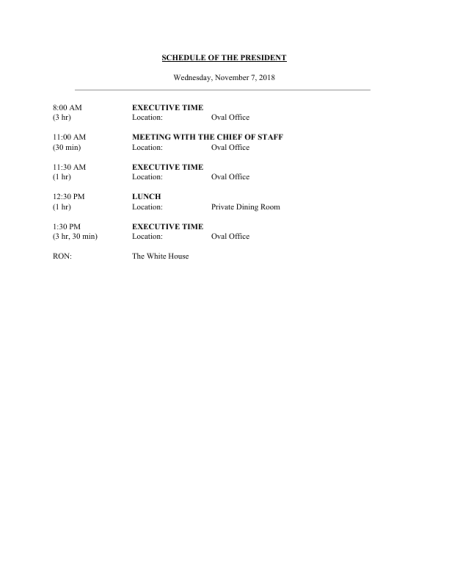
It seemed to work but then re-installing pdftools failed. Jeroen  
thankfully advised me to remove my old Poppler installation with a  
drastic command that affected all softwares depending on Poppler (that’s  
why you might not want to reproduce this at home) like GIMP or Libre  
Office Writer: sudo apt-get remove libpoppler\*. But, after I did that,  
I was able to install pdftools using the newest Poppler version.

**General strategy**

I started work by carefully looking at the PDF. There were one-page  
daily schedules like on page 1,

pdftools::pdf\_render\_page(pdf\_file, page = 1) %>%

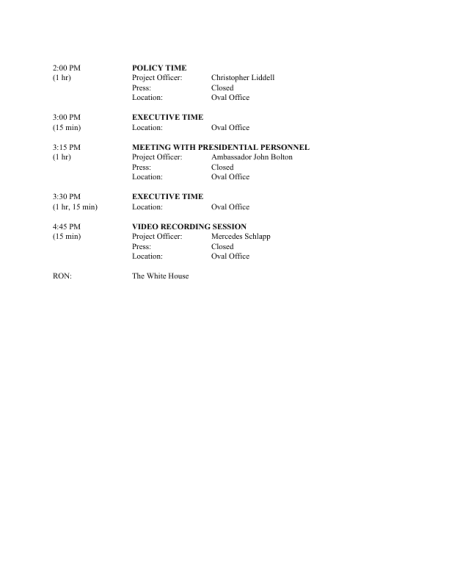
magick::image\_read()



but also schedules continued on several pages,

pdftools::pdf\_render\_page(pdf\_file, page = 3) %>%

magick::image\_read()



Because of that, I decided to extract the dates of pages that had one,  
then assign to each undated page the latest available date, before  
splitting the PDF low level text data into dates and working on a  
pipeline by date.

**Splitting the low level text data by date**

In a first step, I wrote a pipeline adding the page number to each  
data.frame from pdftools::pdf\_data().

pdf\_data <- pdftools::pdf\_data(pdf\_file)

read\_page <- function(page, pdf\_data){

df <- pdf\_data[[page]]

if(nrow(df) > 0) {

df$page <- page

df

} else {

NULL

}

}

pdf\_text <- purrr::map(1:pdftools::pdf\_info(pdf\_file)$pages,

read\_page,

pdf\_data = pdf\_data)

# remove empty pages

pdf\_text <- purrr::discard(pdf\_text, is.null)

head(pdf\_text[[1]])

## width height x y space text page

## 1 60 12 220 72 TRUE SCHEDULE 1

## 2 15 12 283 72 TRUE OF 1

## 3 23 12 301 72 TRUE THE 1

## 4 62 12 327 72 FALSE PRESIDENT 1

## 5 54 12 236 99 TRUE Wednesday, 1

## 6 46 12 293 99 TRUE November 1

Then for each of them I tried extracting the date.

get\_date <- function(page\_data){

if(page\_data$text[[1]] == "SCHEDULE"){

date <- as.character(

glue::glue\_collapse(page\_data$text[6:8],

sep = " "))

date <- lubridate::parse\_date\_time(date,

orders = "BdY",

locale = "en\_US.utf8", tz = "EST")

date <- as.Date(date)

}else{

date <- NA

}

tibble::tibble(date = date,

page = page\_data$page[1])

}

dates <- purrr::map\_df(pdf\_text, get\_date)

Finally, I created a gigantic data.frame of all the data.frame’s  
from pdftools::pdf\_text(), sorted it by page just to be sure it was  
sorted by page, assigned to each row the latest non missing page, and  
then split the data by date which was a better unit to work from. In  
order to not mess up coordinates, within each date I added 700, 1400,  
etc. to y-coordinates from the second, third, etc. page.

split\_days <- function(pdf\_text, dates){

pdf\_text %>%

dplyr::bind\_rows() %>%

dplyr::left\_join(dates, by = "page") %>%

dplyr::arrange(page) %>%

dplyr::mutate(date = zoo::na.locf(date)) %>%

dplyr::group\_by(date) %>%

dplyr::mutate(y = y + (page - min(page)) \* 700) %>%

dplyr::ungroup() %>%

split(.$date)

}

pdf\_text <- split\_days(pdf\_text, dates)

length(pdf\_text)

## [1] 50

pdf\_text[[2]]

## # A tibble: 243 x 8

## width height x y space text page date

##

## 1 60 12 220 72 TRUE SCHEDULE 2 2018-11-08

## 2 15 12 283 72 TRUE OF 2 2018-11-08

## 3 23 12 301 72 TRUE THE 2 2018-11-08

## 4 62 12 327 72 FALSE PRESIDENT 2 2018-11-08

## 5 44 12 240 99 TRUE Thursday, 2 2018-11-08

## 6 46 12 287 99 TRUE November 2 2018-11-08

## 7 8 12 336 99 TRUE 8, 2 2018-11-08

## 8 21 12 347 99 FALSE 2018 2 2018-11-08

## 9 19 12 72 140 TRUE 8:00 2 2018-11-08

## 10 17 12 94 140 FALSE AM 2 2018-11-08

## # … with 233 more rows

tail(pdf\_text[[2]])

## # A tibble: 6 x 8

## width height x y space text page date

##

## 1 28 12 229 1069 FALSE House 3 2018-11-08

## 2 51 12 288 799 TRUE Christopher 3 2018-11-08

## 3 31 12 342 799 FALSE Liddell 3 2018-11-08

## 4 30 12 288 813 FALSE Closed 3 2018-11-08

## 5 21 12 288 826 TRUE Oval 3 2018-11-08

## 6 28 12 312 826 FALSE Office 3 2018-11-08

At that point I took a minute to compare the dates I had with the ones  
in Axios’ clean data.

clean\_data <- readr::read\_csv("data/axios\_data.csv")

unique(clean\_data$date[!clean\_data$date %in% dates$date])

## [1] "2018-11-12" "2018-11-16" "2018-11-20" "2018-11-21" "2018-11-22"

## [6] "2018-12-06" "2018-12-24" "2018-12-25" "2018-12-26" "2018-12-27"

## [11] "2018-12-28" "2018-12-31" "2019-01-01"

Wat?! I had a look at the PDF, and actually, these dates are not even in  
the PDF I’m fighting, so no need for me to try and revise my R script.

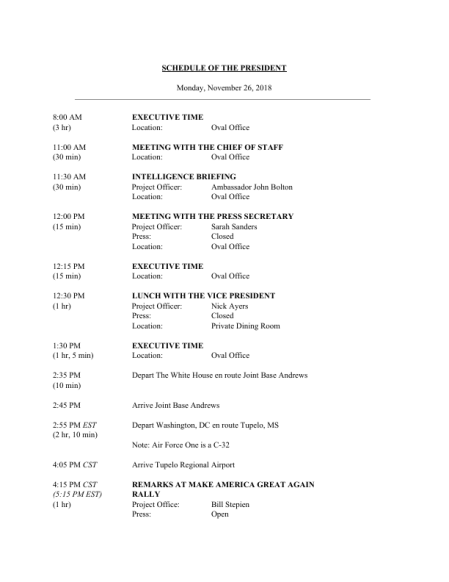
**Digesting each date’s data**

This part was the most crucial, and difficult one. I wanted to obtain a  
table with one line by event. My strategy was to first extract times by  
filtering text data whose x-coordinate was in an interval I had decided  
on by looking at some pages, clean these times to obtain time, duration  
and y-coordinate of the top of the time info, and then assign words from  
the right of the page to each event thanks to fuzzy joins: words  
correspond to an event if their y-coordinate is between the y-coordinate  
of the time of that event and the y-coordinate of next event.

On top of that I had to decide what to do with pages with times in  
several timezones, like below

pdftools::pdf\_render\_page(pdf\_file, page = 13) %>%

magick::image\_read()



I simply removed all times that where in parentheses, assumed a time  
with no timezone information had the same timezone as the last event  
with a timezone, and that the first event of the day, if lacking a  
timezone, was in EST. In real life if I were really working on such data  
I’d probably still make such a general assumption but then review the  
result page by page, preferably with several pairs of eyes.

get\_times <- function(page\_data){

times <- dplyr::filter(page\_data,

dplyr::between(x, 72, 117))

rest <- dplyr::filter(page\_data,

!dplyr::between(x, 72, 117))

times <- times %>%

dplyr::group\_by(y) %>%

dplyr::summarize(text = glue::glue\_collapse(text, sep = " "))

times <- dplyr::filter(times,

!(grepl("\\(", text)&grepl("M", text)))

clean\_times <- dplyr::mutate(times,

time = dplyr::if\_else(

grepl("M", text)|grepl("RON", text),

text,

""),

duration = dplyr::if\_else(

grepl("\\(", dplyr::lead(text)),

dplyr::lead(text),

"")

)

clean\_times <- dplyr::filter(clean\_times, time != "")

clean\_times <- dplyr::select(clean\_times, y, time, duration)

clean\_times <- dplyr::mutate(clean\_times,

timezone = gsub(".\*\\:.\*M ", "", time),

timezone = ifelse(timezone == time,

NA, timezone))

if(is.na(clean\_times$timezone[1])){

clean\_times$timezone[1] <- "EST"

}

clean\_times <- dplyr::mutate(clean\_times,

timezone = zoo::na.locf(timezone))

clean\_times <- dplyr::mutate(clean\_times,

timezone = trimws(timezone),

timezone = gsub(" ", "",timezone),

time = gsub("M.\*", "M", time))

clean\_times <- dplyr::mutate(clean\_times,

start = y,

end = ifelse(dplyr::lead(y) != start,

dplyr::lead(y) - 1, start),

end = ifelse(is.na(end),

99999, end))

if(nrow(clean\_times) == 0){

print(page\_data$date[1])

return(NULL)

}

rest <- dplyr::mutate(rest,

start = y, end = y)

df <- fuzzyjoin::interval\_left\_join(clean\_times,

rest,

by = c("start", "end"))

df <- dplyr::select(df, time, duration, timezone, x, y.y, text)

df <- dplyr::rename(df, y = y.y)

df <- df %>%

dplyr::group\_by(time, duration, timezone) %>%

dplyr::mutate(event = as.character(

glue::glue\_collapse(text[y == min(y)], sep = " "))) %>%

dplyr::group\_by(time, duration, timezone, y, event) %>%

dplyr::summarize(text = as.character(

glue::glue\_collapse(text, sep = " ")

)) %>%

dplyr::group\_by(time, duration, timezone, event) %>%

dplyr::mutate(text = ifelse(toupper(event) != event,

as.character(

glue::glue\_collapse(text, sep = " ")

), text)) %>%

dplyr::ungroup()

if(nrow(df) == 0){

df <- tibble::tibble(event = as.character(

glue::glue\_collapse(page\_data$text, sep = " ")

))

}

df$date <- page\_data$date[1]

df

}

all\_events <- purrr::map\_df(pdf\_text,

get\_times)

head(all\_events)

## # A tibble: 6 x 7

## time duration timezone y event text date

##

## 1 1:30 PM (3 hr, 30 … EST 302 EXECUTIVE TI… EXECUTIVE TI… 2018-11-07

## 2 1:30 PM (3 hr, 30 … EST 315 EXECUTIVE TI… Location: Ov… 2018-11-07

## 3 11:00 … (30 min) EST 180 MEETING WITH… MEETING WITH… 2018-11-07

## 4 11:00 … (30 min) EST 194 MEETING WITH… Location: Ov… 2018-11-07

## 5 11:30 … (1 hr) EST 221 EXECUTIVE TI… EXECUTIVE TI… 2018-11-07

## 6 11:30 … (1 hr) EST 234 EXECUTIVE TI… Location: Ov… 2018-11-07

**Cleaning each event**

At that point I was fairly happy but I knew that a code review would  
have been great in real life, and that in any case I needed more work. I  
wanted to have a duration in minutes for each event as well as a start  
and end as times, and I wanted each event’s info to be in a single row.

I first made each event a row.

gather\_events <- function(all\_events){

all\_events %>%

dplyr::group\_by(event, text) %>%

dplyr::mutate(clean\_text = trimws(gsub(event[1], "", text))) %>%

dplyr::ungroup() %>%

dplyr::select(- text) %>%

dplyr::rename(text = clean\_text)

# events with no text

no\_text\_events <- all\_events %>%

dplyr::group\_by(time, duration,

timezone, date) %>%

dplyr::filter(all(!nzchar(text))) %>%

dplyr::summarise(event = unique(event),

text = NA,

y = min(y))

# events with text

text\_events <- all\_events %>%

dplyr::group\_by(time, duration,

timezone, date) %>%

dplyr::filter(any(nzchar(text))) %>%

dplyr::filter(nzchar(text)) %>%

dplyr::summarize(y = min(y),#y gives us the \*certain\* order of events in a day

event = ifelse(length(text[!grepl("\\:", text)]) > 0,

as.character(

glue::glue\_collapse(paste(unique(event, text[!grepl("\\:", text)&text!=event]))), sep = " "),

event),

text = ifelse(length(text[grepl("\\:", text)]) > 0,

toString(text[grepl("\\:", text)]),

""))

all\_events <- dplyr::bind\_rows(no\_text\_events, text\_events)

all\_events <- dplyr::ungroup(all\_events)

all\_events <- dplyr::arrange(all\_events, date, y)

all\_events

}

all\_events <- gather\_events(all\_events)

Before binding the events with text and no text I had checked by had  
that no events got lost (using dplyr::n\_groups()).

head(all\_events)

## # A tibble: 6 x 7

## time duration timezone date event text y

##

## 1 8:00 AM (3 hr) EST 2018-11-07 EXECUTIVE TIME Location: Ov… 140

## 2 11:00 … (30 min) EST 2018-11-07 MEETING WITH … Location: Ov… 180

## 3 11:30 … (1 hr) EST 2018-11-07 EXECUTIVE TIME Location: Ov… 221

## 4 12:30 … (1 hr) EST 2018-11-07 LUNCH Location: Pr… 261

## 5 1:30 PM (3 hr, 30… EST 2018-11-07 EXECUTIVE TIME Location: Ov… 302

## 6 RON: "" EST 2018-11-07 The White Hou… "" 342

head(all\_events$text)

## [1] "Location: Oval Office" "Location: Oval Office"

## [3] "Location: Oval Office" "Location: Private Dining Room"

## [5] "Location: Oval Office" ""

Further work aimed at extracting “Location”, “Note”, etc. into specific  
columns. This code is especially ugly.

extract\_info <- function(all\_events){

all\_events %>%

dplyr::group\_by(time, duration, timezone, date, event, y) %>%

dplyr::mutate(location = ifelse(any(grepl("Location",

unlist(strsplit(text, ",")))),

gsub("Location\\: ", "",

unlist(strsplit(text, ","))[grepl("Location",

unlist(strsplit(text, ",")))]),

NA),

project\_officer = ifelse(any(grepl("Project Officer",

unlist(strsplit(text, ",")))),

gsub("Project Officer\\: ", "",

unlist(strsplit(text, ","))[grepl("Project Officer",

unlist(strsplit(text, ",")))]),

NA),

press = ifelse(any(grepl("Press",

unlist(strsplit(text, ",")))),

gsub("Press\\: ", "",

unlist(strsplit(text, ","))[grepl("Press",

unlist(strsplit(text, ",")))]),

NA)) %>%

dplyr::ungroup() %>%

dplyr::mutate(location = trimws(location),

project\_officer = trimws(project\_officer),

press = trimws(press))

}

all\_events <- extract\_info(all\_events)

At this point, again, in a real work environment I’d spend time checking  
data carefuly, but instead, I’ll just have a rough look at the clean  
data vs my own cleaned data.

dplyr::count(clean\_data, listed\_project\_officer, sort = TRUE)

## # A tibble: 39 x 2

## listed\_project\_officer n

##

## 1 369

## 2 Ambassador John Bolton 33

## 3 Sarah Sanders 25

## 4 Lindsay Reynolds 18

## 5 Shahira Knight 18

## 6 William McGinley 18

## 7 Christopher Liddell 13

## 8 John DeStefano 10

## 9 Mercedes Schlapp 9

## 10 Tim Pataki 7

## # … with 29 more rows

dplyr::count(all\_events, trimws(project\_officer), sort = TRUE)

## # A tibble: 32 x 2

## `trimws(project\_officer)` n

##

## 1 467

## 2 Ambassador John Bolton 32

## 3 Sarah Sanders 23

## 4 Shahira Knight 18

## 5 William McGinley 16

## 6 Lindsay Reynolds 15

## 7 Christopher Liddell 13

## 8 John DeStefano 10

## 9 Mercedes Schlapp 9

## 10 Tim Pataki 7

## # … with 22 more rows

dplyr::count(clean\_data, listed\_location, sort = TRUE)

## # A tibble: 36 x 2

## listed\_location n

##

## 1 Oval office 310

## 2 75

## 3 Oval Office 48

## 4 Private dining room 34

## 5 Buenos Aires, Argentina 21

## 6 oval office 9

## 7 Grand foyer 7

## 8 Roosevelt room 6

## 9 Cabinet Room 5

## 10 East Room 5

## # … with 26 more rows

dplyr::count(all\_events, trimws(location), sort = TRUE)

## # A tibble: 34 x 2

## `trimws(location)` n

##

## 1 Oval Office 377

## 2 151

## 3 Private Dining Room 35

## 4 Buenos Aires 18

## 5 Grand Foyer 8

## 6 Cabinet Room 7

## 7 Roosevelt Room 7

## 8 Biloxi 4

## 9 Diplomatic Reception Room 4

## 10 McAllen 4

## # … with 24 more rows

I surprisingly have *more* events in my data than in Axios’ clean data,  
which would require further exploration. Part of that might be due to my  
data listing “RON” as events, where RON means “Remainder of the Night”  
which is not an event, but a way to indicate where the President spent  
the rest of his time on that day. On one day, the whole schedule is  
actually “RON”.

dplyr::filter(all\_events, date == lubridate::ymd("2018-11-23"))

## # A tibble: 1 x 10

## time duration timezone date event text y location

##

## 1 RON: "" EST 2018-11-23 Mar-… "" 180

## # … with 2 more variables: project\_officer , press

For some reasons, in its clean data Axios classified that day as  
executive time, although the PDF does not indicate anything about it.

dplyr::filter(clean\_data, date == lubridate::ymd("2018-11-23"))

## # A tibble: 1 x 11

## week date time\_start time\_end duration listed\_title top\_category

##

## 1 3 2018-11-23 08:00 17:00 9 Executive t… executive\_t…

## # … with 4 more variables: listed\_location ,

## # listed\_project\_officer , detail\_category , notes

I then had a go at cleaning times and durations.

clean\_time\_info <- function(all\_events){

all\_events %>%

dplyr::mutate(start = lubridate::parse\_date\_time(paste(date, time),

orders = "Ymd HM p",

locale = "en\_US.utf8", tz = "EST")) %>%

dplyr::mutate(duration = gsub("\\(", "", duration),

duration = gsub("\\)", "", duration)) %>%

dplyr::mutate(hours = ifelse(grepl("hr", duration),

as.numeric(trimws(gsub("hr.\*$", "", duration))),

0),

minutes = ifelse(grepl("min", duration),

as.numeric(trimws(stringr::str\_remove(

gsub("^.\*\\,", "", duration),

"min"))),

0)) %>%

dplyr::mutate(duration = 60\*hours + minutes) %>%

dplyr::select(- hours, - minutes) %>%

dplyr::group\_by(date) %>%

dplyr::mutate(end = ifelse(is.na(duration),

as.POSIXct(dplyr::lead(start),

origin = "1970-01-01"),

as.POSIXct(start + lubridate::minutes(duration),

origin = "1970-01-01")))

}

all\_events <- clean\_time\_info(all\_events)

head(all\_events)

## # A tibble: 6 x 12

## # Groups: date [1]

## time duration timezone date event text y location

##

## 1 8:00… 180 EST 2018-11-07 EXEC… Loca… 140 Oval Of…

## 2 11:0… 30 EST 2018-11-07 MEET… Loca… 180 Oval Of…

## 3 11:3… 60 EST 2018-11-07 EXEC… Loca… 221 Oval Of…

## 4 12:3… 60 EST 2018-11-07 LUNCH Loca… 261 Private…

## 5 1:30… 210 EST 2018-11-07 EXEC… Loca… 302 Oval Of…

## 6 RON: 0 EST 2018-11-07 The … "" 342

## # … with 4 more variables: project\_officer , press ,

## # start , end

I stopped there, not willing to further work on the data. More work on  
timezones, and on filling missing durations based on context, would be  
useful. I’d be delighted to see someone cleaning the dataset more  
thoroughly than I did here!

**Conclusion**

In this post I took advantage of the nifty pdftools::pdf\_data()  
function to extract text from Trump’s PDF schedule along with  
coordinates, in the hope to be able to rectangle the data. Part of that  
went well in particular thanks to fuzzyjoin and Tidyverse tooling,  
part of that went less well, in particular due to the inexact data  
formatting and less than optimal indication of timezones… and my not  
desiring to spend my week cleaning the data. I have no idea how Axios  
prepared the dataset, but if I were to clean such data for actual work,  
after scripting as much as possible I’d request at least one colleague  
to review the code, and this person and I would also sample a few if not  
all events and compare their information in the PDF and the resulting  
data.

The advantage of scripting at least part of the cleaning is  
reproducibility, for being able to explain what happened to the data,  
but also to apply the code to new data. This morning I read that [more  
schedules had been  
leaked](https://www.axios.com/trump-schedule-leaks-4840f751-e663-49c0-b288-2dd39bde9c79.html)  
so I tried my code on it.

pdf\_file2 <- "data/newschedules.pdf"

pdf\_data2 <- pdftools::pdf\_data(pdf\_file2)

pdf\_text2 <- purrr::map(1:pdftools::pdf\_info(pdf\_file2)$pages,

read\_page,

pdf\_data = pdf\_data2) %>%

purrr::discard(is.null)

dates2 <- purrr::map\_df(pdf\_text2, get\_date)

all\_events2 <- pdf\_text2 %>%

split\_days(dates2) %>%

purrr::map\_df(get\_times) %>%

gather\_events() %>%

extract\_info() %>%

clean\_time\_info()

head(all\_events2)

## # A tibble: 6 x 12

## # Groups: date [1]

## time duration timezone date event text y location

##

## 1 8:00… 180 EST 2019-02-04 EXEC… Loca… 141 Oval Of…

## 2 11:0… 30 EST 2019-02-04 MEET… Loca… 180 Oval Of…

## 3 11:3… 15 EST 2019-02-04 EXEC… Loca… 219 Oval Of…

## 4 11:4… 30 EST 2019-02-04 INTE… Proj… 258 Oval Of…

## 5 12:1… 30 EST 2019-02-04 EXEC… Loca… 310 Oval Of…

## 6 12:4… 60 EST 2019-02-04 LUNC… Proj… 349 Private…

## # … with 4 more variables: project\_officer , press ,

## # start , end