

# **SF Data Tips and Tricks**

OEWD Data and Performance Team

2024-04-03

# Table of contents

<b>Introduction</b>	<b>3</b>
<b>1 DataSF</b>	<b>4</b>
<b>2 Spatial Stuff</b>	<b>6</b>
2.1 ArcGIS REST API . . . . .	6
2.2 ArcGIS Pro . . . . .	10
2.3 Spatial Joins . . . . .	10
2.4 Removing Farallon Islands from Supervisor Districts . . . . .	13
2.5 Census Data . . . . .	15
<b>3 Geocoding</b>	<b>17</b>
<b>4 Sharepoint</b>	<b>20</b>
<b>5 Icons</b>	<b>22</b>
<b>References</b>	<b>23</b>

# Introduction

This Quarto ‘book’ is a collection of (mostly R) tips and tricks relevant to data professionals with the City and County of San Francisco.

To learn more about Quarto books visit <https://quarto.org/docs/books>.



# 1 DataSF

Getting data from DataSF is mostly a matter of copying the relevant URL. Many R ‘read’ functions can import data from a URL, e.g. `readr::read_csv`, `jsonlite::fromJSON`, `st::st_read`, etc.

```
library(readr)
library(sf)
```

Linking to GEOS 3.9.1, GDAL 3.4.3, PROJ 7.2.1; sf\_use\_s2() is TRUE

```
reg_businesses <- read_csv("https://data.sfgov.org/resource/g8m3-pdis.csv")
```

Rows: 1000 Columns: 37

-- Column specification -----

Delimiter: ","

chr (22): uniqueid, ttxid, certificate\_number, ownership\_name, dba\_name, fu...

dbl (7): business\_zip, supervisor\_district, :@computed\_region\_6qbp\_sg9q, :...

lgl (2): parking\_tax, transient\_occupancy\_tax

dtm (6): dba\_start\_date, dba\_end\_date, location\_start\_date, location\_end\_d...

i Use `spec()` to retrieve the full column specification for this data.

i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

## Warning

Behind the scenes there is a limit parameter that defaults to 1000, *even if the ‘All data’ radio button is selected*. To retrieve all the data, either read the same URL with the RSocrata package:

```
reg_businesses <- RSocrata::read.socrata("https://data.sfgov.org/resource/g8m3-pdis.csv")
```

Or append ?\$limit=9999999 to the end of the URL:

```
reg_businesses <- read_csv("https://data.sfgov.org/resource/g8m3-pdis.csv?$limit=9999999")
```

Read in a 'spatial' object with `st_read` and the URL with the geojson file extension:

```
sup_dists <- st_read("https://data.sfgov.org/api/geospatial/f2zs-jevy?accessType=DOWNLOAD&method=export&format=geojson")
```

Reading layer `OGRGeoJSON' from data source

```
`https://data.sfgov.org/api/geospatial/f2zs-jevy?accessType=DOWNLOAD&method=export&format=geojson'
using driver `GeoJSON'
```

Simple feature collection with 11 features and 7 fields

Geometry type: MULTIPOLYGON

Dimension: XY

Bounding box: xmin: -123.1738 ymin: 37.63983 xmax: -122.3279 ymax: 37.8632

Geodetic CRS: WGS 84

## 2 Spatial Stuff

```
library(sf)
```

Linking to GEOS 3.9.1, GDAL 3.4.3, PROJ 7.2.1; sf\_use\_s2() is TRUE

```
library(mapview)
library(arcgis)
```

Attaching core arcgis packages:

```
> arcgisutils v0.1.1.9000
> arcgislayers v0.1.0
```

```
library(tidyverse)
```

```
-- Attaching packages ----- tidyverse 1.3.2 --
v ggplot2 3.4.0      v purrr   0.3.4
v tibble  3.1.8      v dplyr   1.1.0
v tidyr   1.2.1      v stringr 1.5.0
v readr   2.1.3      v forcats 0.5.2
-- Conflicts ----- tidyverse_conflicts() --
x purrr::%||%()      masks arcgisutils::%||%()
x purrr::compact()  masks arcgisutils::compact()
x dplyr::filter()   masks stats::filter()
x dplyr::lag()       masks stats::lag()
```

### 2.1 ArcGIS REST API

The `{arcgislayers}` package allows users to read/write data to/from the ArcGIS REST API.

```
sf_libraries_url <- "https://services.arcgis.com/Zs2aNLfN00jrS4gG/arcgis/rest/services/SF_
# arc_open can read a FeatureServer or a FeatureLayer directly
(sf_libraries_fs <- arc_open(sf_libraries_url))
```

```
<FeatureServer <2 layers, 0 tables>>
CRS: 3857
Capabilities: Query
  0: Libraries (esriGeometryPoint)
  1: Libraries with Air Conditioning (esriGeometryPoint)
```

```
(sf_libraries_lyr <- get_layer(sf_libraries_fs, name = "Libraries"))
```

```
<FeatureLayer>
Name: Libraries
Geometry Type: esriGeometryPoint
CRS: 3857
Capabilities: Query
```

```
sf_libraries <- arc_select(sf_libraries_lyr)
```

```
Registered S3 method overwritten by 'jsonify':
  method      from
  print.json  jsonlite
```

```
glimpse(sf_libraries)
```

```
Rows: 33
Columns: 23
$ objectid      <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
$ gross_sq_ft   <chr> "8500", "6465", "6096", "8536", "7633", "6100", "8000~
$ block_lot     <chr> "8708110", "3564095", "6539034", "2919031", "0469001"~
$ zip_code      <chr> "94158", "94114", "94114", "94127", "94123", "94112",~
$ facility_id   <chr> "1113", "648", "1184", "1853", "1053", "896", "1858",~
$ city          <chr> "San Francisco", "San Francisco", "San Francisco", "S~
$ latitude      <chr> "37.775369728", "37.76406037", "37.750228042", "37.74~
$ department    <chr> "Public Library", "Public Library", "Public Library",~
$ longitude     <chr> "-122.393097384", "-122.431881717", "-122.435090242",~
```

```

$ dept_id      <chr> "48", "48", "48", "48", "48", "48", "48", "48", "48", ~
$ common_nam   <chr> "Mission Bay Library", "Eureka Valley Branch Library/~
$ address      <chr> "960 04th St", "1 Jose Sarria Ct", "451 Jersey St", "~
$ supervisor   <chr> "6", "8", "8", "7", "2", "7", "5", "6", "10", "8", "9~
$ city_tenan   <chr> " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " ", " "
$ owned_leas   <chr> "Own", "Own", "Own", "Own", "Own", "Own", "Own", "Own~
$ globalid     <chr> "755632fc-18d9-4c0e-b65d-ec53d18a132b", "195a2a2b-302~
$ created_user  <chr> "nancy.milholland_sfdem", "nancy.milholland_sfdem", "~
$ created_date  <dtm> 2019-06-07 21:44:34, 2019-06-07 21:44:34, 2019-06-07~
$ last_edited_user <chr> "nancy.milholland_sfdem", "nancy.milholland_sfdem", "~
$ last_edited_date <dtm> 2019-06-07 21:44:34, 2019-06-07 21:44:34, 2019-06-07~
$ eas_id       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N~
$ air_conditioning <chr> "Yes", "No", "No", "No", "No", "No", "No", "No", "Sec~
$ geometry     <POINT [m]> POINT (-13624737 4547742), POINT (-13629055 454~

```

```

# You can also specify which columns to select, e.g.:

```

```

# arc_select(
#   sf_libraries_lyr,
#   fields = c("common_nam", "gross_sq_f", "address"),
#   where = "gross_sq_f < 8000"
# )

```

```

# With pipes:

```

```

# sf_libraries_url %>%
#   arc_open() %>%
#   get_layer(name = "Libraries") %>%
#   arc_select()

```

```

# With pipes and tidyverse:

```

```

# (if the url points to a FeatureLayer instead of a FeatureServer)
# sf_libraries_url %>%
#   arc_open() %>%
#   select(common_nam, gross_sq_f, address) %>%
#   filter(gross_sq_f < 8000) %>%
#   collect()

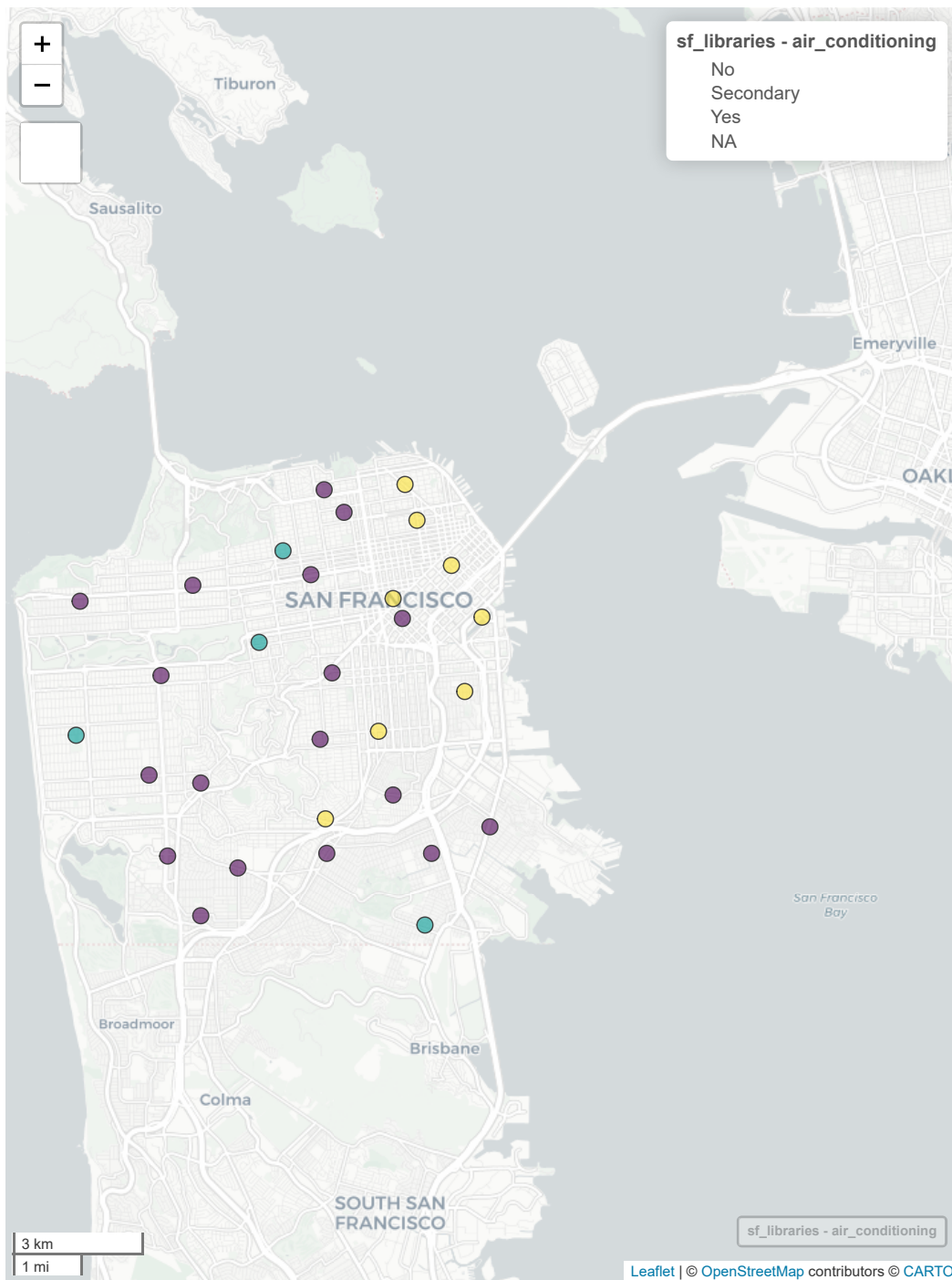
```

```

mapview(sf_libraries, zcol = "air_conditioning")

```





It is also convenient to wrap this up into the body of a single function:

```
get_arcgis_layer <- function(lyr_name) {  
  url <- glue::glue("https://services.arcgis.com/Zs2aNLFN00jrS4gG/arcgis/rest/services/{lyr_name}/FeatureServer/0")  
  out <- arcgislayers::arc_select(arcgislayers::arc_open(url))  
  return(out)  
}  
  
libraries <- get_arcgis_layer("SF_Libraries")
```

## 2.2 ArcGIS Pro

You can write to geodatabases within ArcGIS Pro projects with the {arcgisbinding} package.

Reading layer 'OGRGeoJSON' from data source

```
`https://data.sfgov.org/api/geospatial/f2zs-jevy?accessType=DOWNLOAD&method=export&format=GeoJSON`  
using driver 'GeoJSON'
```

Simple feature collection with 11 features and 7 fields

Geometry type: MULTIPOLYGON

Dimension: XY

Bounding box: xmin: -123.1738 ymin: 37.63983 xmax: -122.3279 ymax: 37.8632

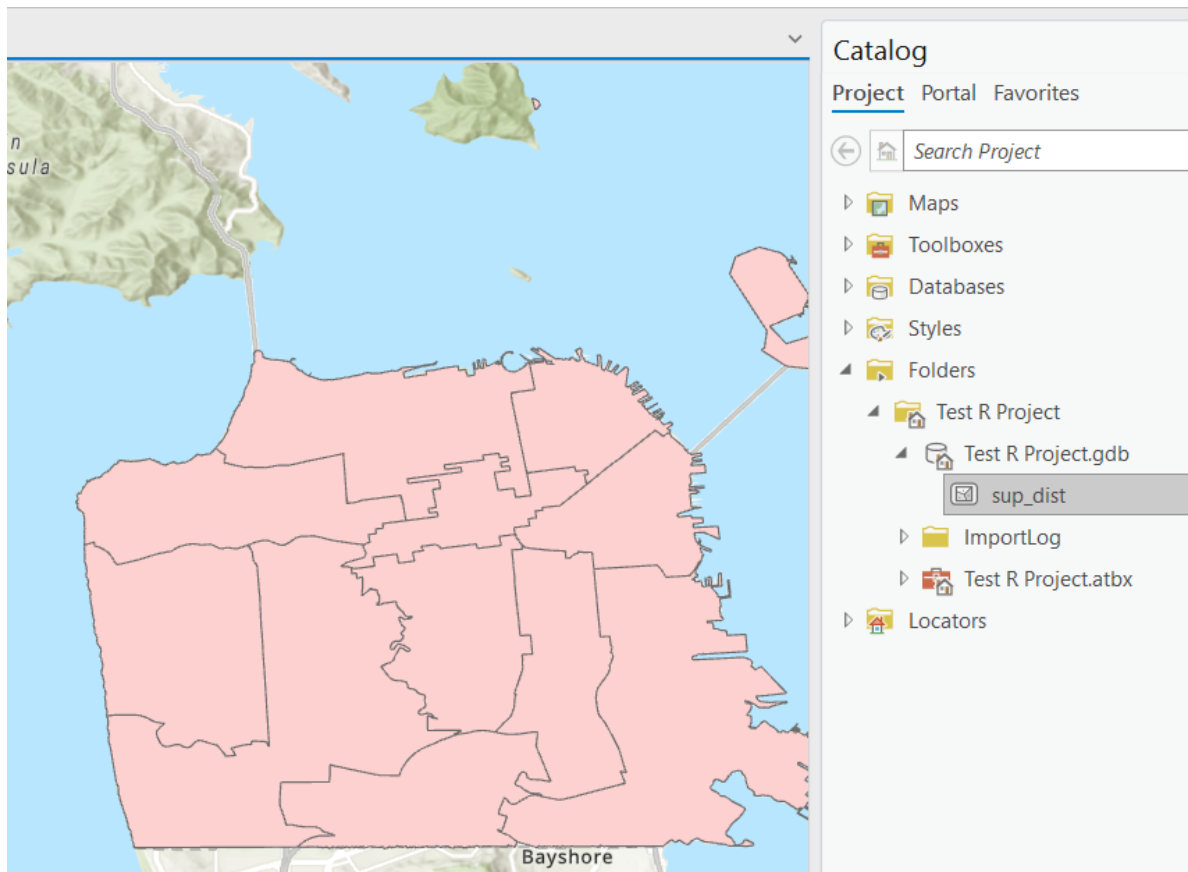
Geodetic CRS: WGS 84

```
library(arcgisbinding)  
# arc.check_product()  
  
# Get Supervisor Districts from DataSF:  
sup_dists <- st_read("https://data.sfgov.org/api/geospatial/f2zs-jevy?accessType=DOWNLOAD&method=export&format=GeoJSON")  
  
# Write to ArcGIS Pro project geodatabase  
proj_path <- "...<full_path>.../ArcGIS/Projects/Test R Project/Test R Project.gdb/sup_dists.gdb" # path to project geodatabase  
arc.write(path = proj_path, data = sup_dists)
```

## 2.3 Spatial Joins

Use spatial joins to determine which points are 'within' which polygon:

```
nhoods <- st_read("https://data.sfgov.org/resource/j2bu-swwd.geojson")
```



```

Reading layer `j2bu-swwd' from data source
  `https://data.sfgov.org/resource/j2bu-swwd.geojson' using driver `GeoJSON'
Simple feature collection with 41 features and 1 field
Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:   xmin: -122.5149 ymin: 37.70813 xmax: -122.357 ymax: 37.8333
Geodetic CRS:   WGS 84

```

```

# The coordinate reference systems must match
st_crs(sf_libraries) == st_crs(sup_dists)

```

```
[1] FALSE
```

```

sf_libraries %>%
  select(common_nam) %>%
  st_transform(st_crs(sup_dists)) %>%
  st_join(sup_dists %>% select(sup_dist), join = st_within) %>%
  st_join(nhoods, join = st_within)

```

```

Simple feature collection with 33 features and 3 fields
Geometry type: POINT
Dimension:      XY
Bounding box:   xmin: -122.4981 ymin: 37.71245 xmax: -121.9681 ymax: 37.80253
Geodetic CRS:   WGS 84
First 10 features:

```

	common_nam	sup_dist
1	Mission Bay Library	6
2	Eureka Valley Branch Library/ Harvey Milk Memorial Branch Library	8
3	Noe Valley Branch Library	8
4	West Portal Branch Library	7
5	Marina Branch Library	2
6	Ingleside Branch	7
7	Western Addition Branch Library	5
8	Library Support Services	6
9	Visitation Valley Branch Library	10
10	Glen Park Branch Library	8

	nhood	geometry
1	Mission Bay	POINT (-122.3931 37.77537)
2	Castro/Upper Market	POINT (-122.4319 37.76406)
3	Noe Valley	POINT (-122.4351 37.75023)

```

4 West of Twin Peaks POINT (-122.4661 37.74137)
5 Marina POINT (-122.4341 37.80137)
6 West of Twin Peaks POINT (-122.4563 37.72406)
7 Japantown POINT (-122.4375 37.78412)
8 South of Market POINT (-122.4137 37.77501)
9 Visitacion Valley POINT (-122.4079 37.71245)
10 Glen Park POINT (-122.4338 37.73398)

```

## 2.4 Removing Farallon Islands from Supervisor Districts

```

d4 <- sup_dists %>%
  filter(sup_dist == 4) %>%
  st_cast("POLYGON") %>%
  slice(1) %>%
  st_cast("MULTIPOLYGON")

```

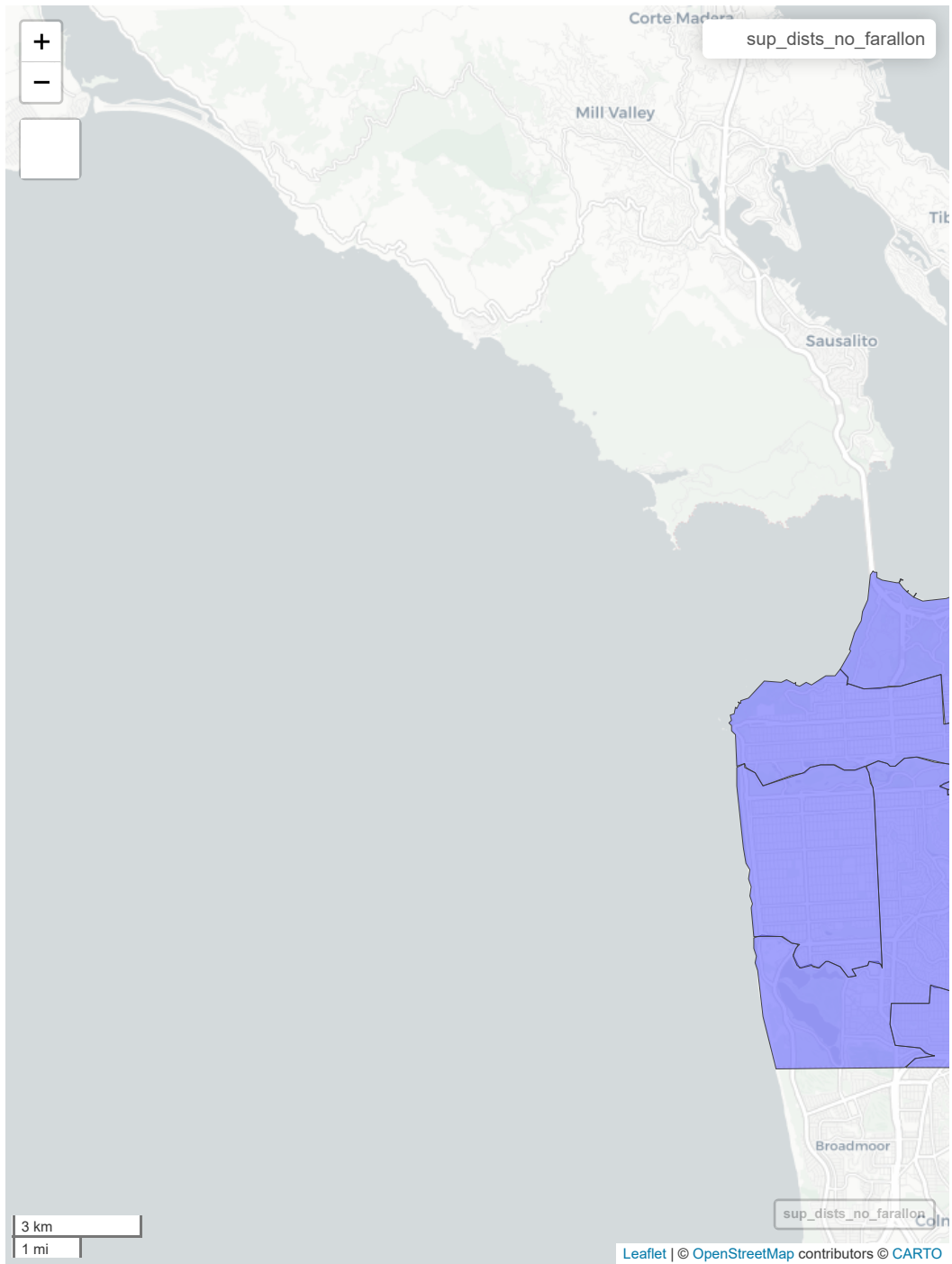
Warning in `st_cast.sf(., "POLYGON")`: repeating attributes for all sub-geometries for which they may not be constant

```

sup_dists_no_farallon <- sup_dists %>%
  filter(sup_dist != 4) %>%
  bind_rows(d4)

mapview(sup_dists_no_farallon)

```



## 2.5 Census Data

The `{tidycensus package}` is fantastic, and the documentation is full of helpful examples.

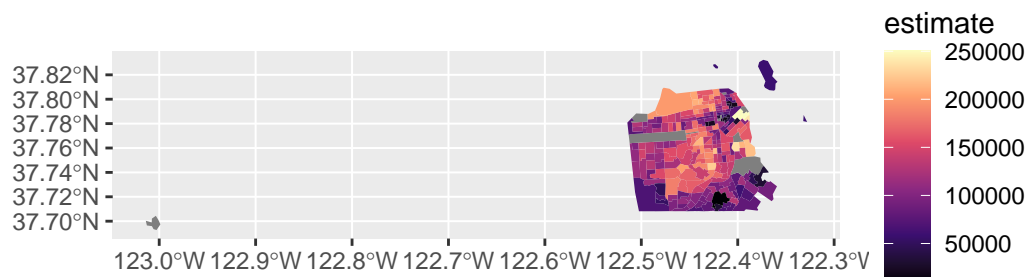
```
library(tidycensus)

sf <- get_acs(
  state = "CA",
  county = "San Francisco",
  geography = "tract",
  variables = "B19013_001",
  geometry = TRUE,
  year = 2020
)
```

Getting data from the 2016-2020 5-year ACS

Downloading feature geometry from the Census website. To cache shapefiles for use in future

```
sf %>%
  ggplot(aes(fill = estimate)) +
  geom_sf(color = NA) +
  scale_fill_viridis_c(option = "magma")
```





## 3 Geocoding

There are several ways to geocode addresses from R, but the easiest (and cheapest) way is with the `{tidygeocoder}` package and one of the city's [internal locators](#).

### **i** Note

The locator will only geocode San Francisco addresses.

```
library(tidygeocoder)
library(sf)
```

Linking to GEOS 3.9.1, GDAL 3.4.3, PROJ 7.2.1; `sf_use_s2()` is TRUE

```
library(mapview)
library(tidyverse)
```

```
-- Attaching packages ----- tidyverse 1.3.2
--
```

```
v ggplot2 3.4.0      v purrr   0.3.4
v tibble  3.1.8      v dplyr   1.1.0
v tidyr   1.2.1      v stringr 1.5.0
v readr   2.1.3      v forcats 0.5.2
```

```
-- Conflicts ----- tidyverse_conflicts() --
```

```
x dplyr::filter() masks stats::filter()
x dplyr::lag()     masks stats::lag()
```

```
df <- tibble(address = c("1 South Van Ness", "1 Dr Carlton B Goodlett Pl"))
locator <- "https://gis.sf.gov/svc/rest/services/loc/c83_eas_str_ctrl_composite/GeocodeSer

coords <- df %>%
  geocode(
```

```

    api_url = locator,
    address = address,
    custom_query = list(outSR = "4326"), # outSR (Spatial Reference) is a required parameter
    method = "arcgis"
  )

```

Passing 2 addresses to the ArcGIS single address geocoder  
 Query completed in: 0.3 seconds

```

coords

```

```

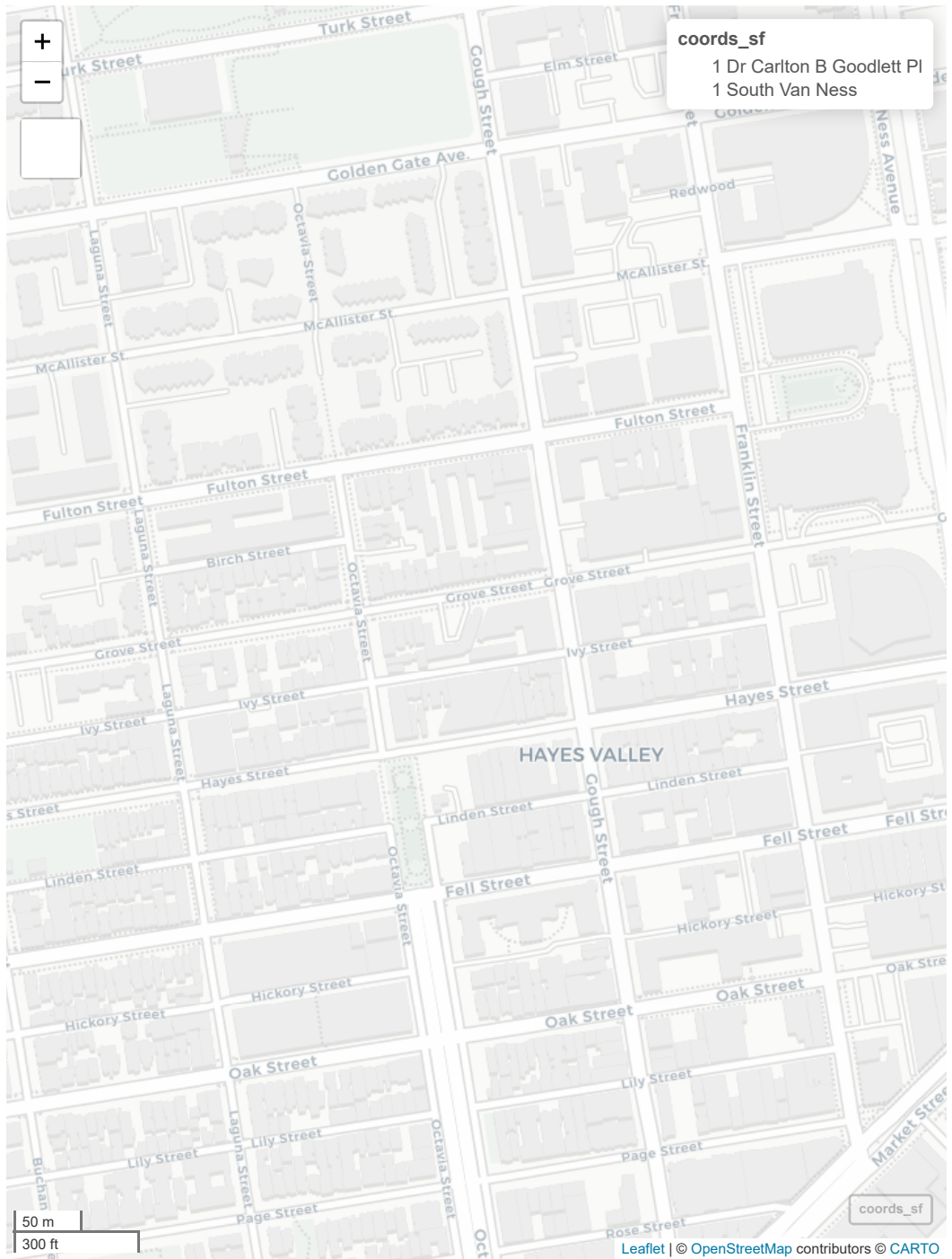
# A tibble: 2 x 3
  address          lat long
  <chr>          <dbl> <dbl>
1 1 South Van Ness      37.8 -122.
2 1 Dr Carlton B Goodlett Pl 37.8 -122.

```

```

coords_sf <- coords %>% st_as_sf(coords = c("long", "lat"), crs = 4326)
mapview(coords_sf)

```



## 4 Sharepoint

Sharepoint is part of many data pipelines, and the easiest way to interact with Sharepoint from R is through the [{Microsoft365R}](#) package. This package has been vetted by the Department of Technology (DT) and authentication should be straight-forward after calling one of the initial functions.

```
library(Microsoft365R)

wp <- get_sharepoint_site("ECN-Workforce Programs")
wp_docs <- wp$get_drive("Documents")

wp_docs$list_files()

wp_docs$download_file("path_to_file.xlsx")
```

It is helpful to wrap common read/write operations into more usable functions:

```
connect_to_sharepoint_site_docs <- function(sp_site) {
  sp_site <- Microsoft365R::get_sharepoint_site(sp_site)
  sp_site_doc <- sp_site$get_drive("Documents")
  return(sp_site_doc)
}

connect_to_wp_docs <- function() connect_to_sharepoint_site_docs("ECN-Workforce Programs")

download_from_wp <- function(sp_file, destination) {
  docs <- connect_to_wp_docs()
  docs$download_file(sp_file, dest = destination)
  cli::cli_alert_success(glue::glue("{sp_file} downloaded."))
}

upload_to_wp <- function(file, sp_destination) {
  docs <- connect_to_wp_docs()
  docs$upload_file(
    file,
```

```

    sp_location
  )
  cli::cli_alert_success("File uploaded.")
}

read_wp <- function(path) {
  tmp <- tempfile(fileext = "xlsx")
  download_from_wp(
    sp_file = glue("{path}.xlsx"),
    destination = tmp
  )
  out <- readxl::read_excel(tmp, .name_repair = janitor::make_clean_names)
  return(out)
}

```

## 5 Icons

The Digital Services team has provided [a nifty set of icons on the San Francisco Design System website](#). You can use these icons in Quarto (HTML) documents by installing [the sficons extension from GitHub here](#).

```
quarto install extension SFOEWD/sficons
```

To embed an icon, use the `sficon` shortcode. Some examples:

```
{{< sficon wip >}}  
{{< sficon alert >}}
```

```
{{< sficon arrow-right color=firebrick >}}
```

```
{{< sficon globe color=green size=5em >}}
```

```
{{< sficon pencil color=gold size=10em >}}
```

Control the color and size of the icons:

## References