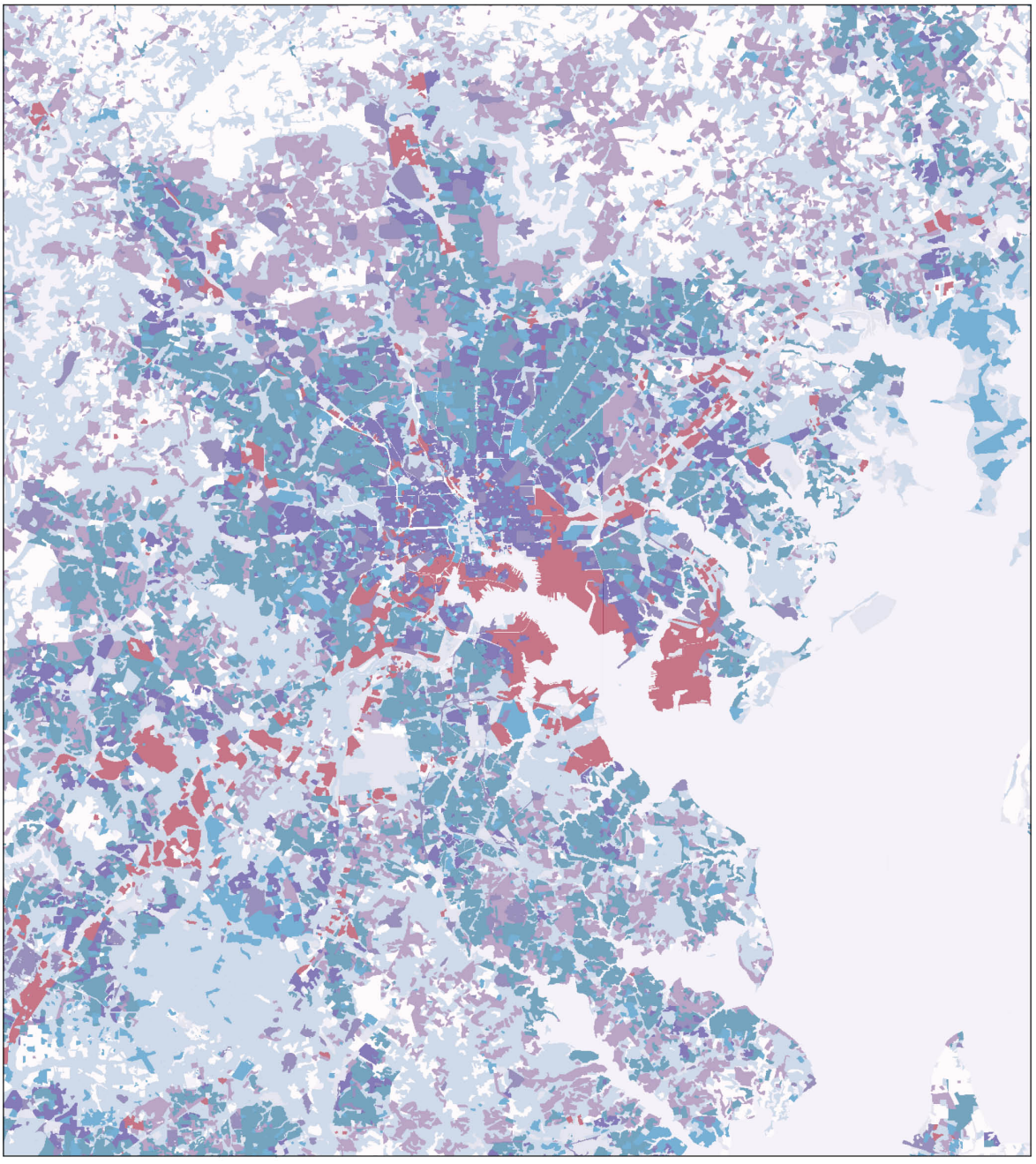
# Spatial Analysis and Mapping with R: A Short Tutorial

Wolf T. Pecher



*Wolf T. Pecher*

# *Spatial Analysis and Mapping with R: A Short Tutorial*



**Attributions**

"Spatial Analysis and Mapping with R: A Short Tutorial" by Wolf T. Pecher used under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License (online at https://ubalt.pressbooks.pub/spatialanalysisenvironmentalstatisticsr/#main).

**Cover**

Design: Nett Smith

Cover image: Land Use in Greater Baltimore (2002). Data source: Maryland Archived Land Use Land Cover 2002, Maryland Department of Planning, online at https://data.imap.maryland.gov/datasets/96116be90edb4e8d933048f345c3a487?geometry=-79.386%2C38.060%2C-75.096%2C39.558#www.mdp.state.md.us/OurWork/landuse.shtml (accessed on April 17, 2021).

[1]Published with bookdown (https://github.com/rstudio/bookdown). LaTeX class adapted from krantz.cls

# *Contents*

# *List of Figures*

# *Preface*

This tutorial introduces the reader to some of the amazing capabilities of **R** to work with and map geographic data. Geographic data are data that contain spatial attributes (or spatial data) that define a geographic space (location, area, elevation, etc.) and non spatial attributes (f.e., population density, pollutant concentrations, temperature).

This tutorial was developed for one the units of the course "ENVS 420: Research Seminar in Environmental Sciences" offered at the University of Baltimore. However, it is hoped that readers outside of ENVS 420 who are interested in geospatial analysis and with a basic familiarity of **R** find this tutorial useful.

The use of an integrated developer environment (IDE) or an IDE like configuration such as the IDE **RStudio** (https://rstudio.com/) or the **Nvim-R** plug-in for the integration of **vim/neovim** and **R** (https://github.com/jalvesaq/Nvim-R/tree/stable) is recommended but not necessary.

The tutorial was written with **RMarkdown** (v. 2.6) (Allaire *et al.*, 2020; Xie *et al.*, 2018, 2020) in **R** (v. 4.0.2) (R Core Team, 2020).

Required **R** packages:

- **dplyr** (Wickham *et al.*, 2020)
- **openxlsx** (Schauberger & Walker, 2020)
- **RColorBrewer** (Neuwirth, 2014)
- **sf** (Pebesma, 2018)
- **tmap** (Tennekes, 2018)
- **tidyr** (Wickham & Henry, 2020)

### Data

Datatasets used are archived in a zip compressed file (**SpatialAnalysisData.zip**) that can be downloaded at SpatialAnalysisData (**URL**: https://mega.nz/file/hYk02AyK#4knQ1zcaIxKTN_GX3JN8ZyOPD4lXjynIbA3PvpvSaG4). The link will connect you to a cloud storage service (MEGA, https://mega.nz) and ask you to download the file. By accessing the cloud storage service and downloading the file/data you agree the terms of service of MEGA and to the terms of use of the code and data.

# Chapter 1

## *Service Deserts*

In 2015 the United Nations General Assembly adopted 17 Sustainable Development Goals that are at the core of the 2030 Agenda for Sustainable Development. The goals address the three dimensions (pillows) of sustainable development, namely economic, social, and environmental sustainability. The goals range from eradicating poverty and reduction of inequality to improved education and health, improved resilience to climate change, and preservation of oceans and forests (United Nations General Assembly, 2015).

### 1.1 Definition of a Service Desert

This chapter is about mapping rural and urban areas that have limited access to basic services, and as such can be flagged as "Basic Service Deserts." Improving access to basic services is included in the 2030 Agenda for Sustainable Development. It is within the realms of Goal 3: "Ensure healthy lives and promote well-being for all at all ages," and Goal 11: "Make cities and human settlements inclusive, safe, resilient and sustainable" (United Nations General Assembly, 2015).

The definition of a "Service Desert" is based on the definition of a "Food Desert." The U.S. Departments of Agriculture (USDA), Treasury, and Health and Human Services (HHS) define census tracts as "Food Deserts" if the following criteria are met (Ver Ploeg *et al.*, 2011):

1. The census tract is considered a low-income area. Low-income areas are defined as areas with a poverty rate > 20% or with a median family income of less than 80% of the median family income of the state or the respective urban area.
2. The census tract is considered a low-access area. At least five hundred people or 33% of the population of the area live more than one mile away from a supermarket in urban centers, and more than 10 miles away in rural areas.

However, if it is assumed that access to services in urban centers should be available to residents without means of transportation arguably walking distance should be considered. Walking distance is a measure of how much distance can be covered by walking over time. What distance is deemed acceptable is somewhat arbitrary and is influenced by many factors. Not surprisingly, it is a heavily debated topic. Studies around transportation suggest that a walking distance of 800 m (0.5 mi) may be acceptable, which translates roughly to a 10 to 15-minute walk.

The open access datasets that are used do not allow to identify the precise location of each household/resident, making it impossible to calculate the number or proportion of residents that live outside a certain range. As a proxy, we use the area of a census tract.

Therefore, for the purpose of this chapter, we declare a census tract a "Service Desert" if the census tract is a low-income tract (as defined above) with *limited access* (not low access) to the service. A census tract is considered a limited-access tract if 33% or more of the area of the census tract is outside a 0.5 mi or 10 mi range of the service for urban centers and rural areas, respectively.

# Chapter 2

## *Coordinate Reference Systems*

When working with spatial data, coordinate reference systems (`CRS`) are a key component. `CRS` allow to determine positions on a three-dimensional surface (usually Earth) and project them onto a two-dimensional surface ("flattening"). Key components that define a `CRS` are:

- **Coordinate system**: A $x$ and $y$ and $z$ grid that defines where the data points are located in space.
- **Units**: Units of the distances on the $x$ and $y$ axis of the grid (horizontal unit) and the $z$ axis (vertical unit)
- **Datum**: Model of the shape of the object (in our case, Earth), that defines the origin of the coordinate system (or, the reference point). For a global system for example the reference point is the Prime Meridian and the Equator.
- **Projection**: The mathematical equation used to project positions on the 3D object onto a 2D flat surface.

### 2.1 Geographic and Projected `CRS`

**Geographic Coordinate Reference Systems** are used to map places on the surface of a globe (i.e., Earth) based on two values, longitude and latitude. Longitude is the location in East-West direction in angular distance (usually degrees) from the Prime Meridian. Latitude is the North/South location in angular distance from the Equator. Units of angular distances are not linear. Therefore, geographic `CRS` are not suitable to calculate and compare distances between locations.

Most geographic `CRS` model the Earth as an ellipsoid rather than a perfect sphere. What ellipsoid is used is defined by the **datum**.

There are two types of datum: a geocentric datum and a local datum. The geocentric (or geoid) datum (such as the `WGS84`) uses the Earth's center of gravity as its center. The ellipsoid is not optimized for local variations. A local datum, such as the North America Datum (`NAD`) optimize the ellipsoid to include local variations such as large mountain ranges.

**Projected Coordinate Reference Systems** are based on a Cartesian coordinate system on a flat surface. Map projections are used to convert the 3D surface of the Earth into $x$ and $y$ coordinates of the projected `CRS`. Projections are grouped into 3 main types, conic, cylindrical, and planar.

## 2.2 Formats of CRS Definitions

There are multiple formats that define **CRS**, including **WKT** (well-known text) strings, **proj4string**, and **EPSG**.

Spatial packages in **R** support two ways of describing **CRS**: **EPSG** code and **proj4string** definitions. **EPSG** code defines one specific **CRS** whereas **proj4string** definitions are more flexible and allow to define projection, datum etc.

## 2.3 Common CRS

The World Geodetic System 1984 (**WGS84**or **WGS 84**) and the North American Datum 1983 (**NAD83**) are two commonly used **CRS** in the U.S. Both systems are geocentric and use Greenwich as the Prime Meridian. Units of measurements are degrees, and the axes longitude ($x$) and latitude ($y$).

The **EPSG** code of the **WGS 84** is **4326**, and the **proj4string** is **+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs**. **WGS 84** is used by the global positioning system (GPS).

The **NAD83** is a local datum used in Canada and the U.S. The **EPSG** of the latest rendition of **NAD83** (2011) is **4269** and the **proj4string** is **+proj=longlat +ellps=GRS80 +datum=NAD83 +no_defs**. It uses a different ellipsoid model (**GRS80**). Furthermore, **NAD83** coordinates for points on North American Plate do not change over time. Thus, coordinates of locations on the North American Plate are not affected by plate tectonics. Having said that, position west of the San Andreas Fault and Hawaii are not on the North American Plate. In contrast, **WGS 84** position coordinates are defined based on the average of stations around the globe. Therefore coordinates of **WGS 84** defined positions deviate by 1 to 2 cm per year from coordinates established by **NAD83**. Today the deviations are large enough that they need to be taken into consideration.

If calculations of area or distance are required, and to avoid distortions, spatial data using a geocentric **CRS** need to be re-projected to a projected **CRS** with linear units (meter, feet, US survey feet). Which projected **CRS** to use depends on the region. For Maryland, **NAD83(2011)/Maryland** (**EPSG: 6487**, unit: meter; or **EPSG: 6488**, unit: US survey feet) are often used.

Note, there are (at least) two other **EPSG** codes that seem to be equivalent to **EPSG:6487** and **EPSG:6488**, namely **EPSG:26985** and **EPSG:2248**.

- **proj4string** of **EPSG:6487** and **26985**:
  - **EPSG:6487**: **+proj=lcc +lat_0=37.6666666666667 +lon_0=-77 +lat_1=39.45 +lat_2=38.3 +x_0=400000 +y_0=0 +ellps=GRS80 +units=m +no_defs.**
  - **EPSG:26985**: **+proj=lcc +lat_0=37.6666666666667 +lon_0=-77 +lat_1=39.45 +lat_2=38.3 +x_0=400000 +y_0=0 +datum=NAD83 +units=m +no_defs.**

- **proj4string** of **EPSG:6488** and **2248**:
  - **EPSG:6488**: **+proj=lcc +lat_0=37.6666666666667 +lon_0=-77 +lat_1=39.45 +lat_2=38.3 +x_0=399999.9998984 +y_0=0 +ellps=GRS80 +units=us-ft +no_defs.**
  - **EPSG:2248**: **+proj=lcc +lat_0=37.6666666666667 +lon_0=-77 +lat_1=39.45 +lat_2=38.3 +x_0=399999.9998984 +y_0=0 +datum=NAD83 +units=us-ft +no_defs.**

**EPSG:6487** and **6488** were released by the U.S. National Geodetic Survey (Revision date: 2013-10-09). The area covered is Maryland (IOGP Geomatics Committee, 2021). **EPGS:26985** and **2248** were released by the U.S. Defense Agency TR8350.2 (Revision date: 2014-11-19). The area covered appears to be much broader (**WGS 84** bounds: -172.54 23.81; -47.74 86.46) (MapTiler Team, 2019). These two **EPSG** are not included in the **EPSG** Geodeditc parameter dataset (v10.018).

For both, **EPSG:6487** and **EPSG:6488** the latitude/longitude at the (artificial) origin (**0**, **0**) is 37°34'38.14264"N and 81°31'45.07877"W. False Easting at the 77th meridian is 400,000 m (**EPSG:6487**) and 399,999.9998984 m (**EPSG:6488**) (Reger, 2013).

# Chapter 3

## *Data*

As an example we will map census tracts in Maryland and Baltimore City with limited access to COVID-19 vaccination sites using open access datasets.

### 3.1 Data

The following datasets were retrieved from different open access sources on April 4, 2021:

1. Food Access Research Atlas Data (Data Download 2015) (Excel Workbook format [**.xlsx**])
2. Census tract boundaries (ESRI shapefile collection [**.zip**])
3. List of COVID-19 vaccination sites in Maryland, with coordinates (comma separated values file, [**.csv**])
4. Maryland Physical Boundaries - County Boundaries (ESRI shapefile collection [**.zip**])

#### Access to Data

The datasets were archived in a zip compressed file (**SpatialAnalysisData.zip**) that can be downloaded at SpatialAnalysisData (**URL**: `https://mega.nz/file/hYk02AyK#4knQ1zcaIxKTN_GX3JN8ZyOPD4lXjynIbA3PvpvSaG4`). The link will connect you to a cloud storage service (MEGA, `https://mega.nz`) and ask you to download the file. By accessing the cloud storage service and downloading the file/data you agree to the terms of service of MEGA and to the terms of use of the code and data.

#### Data Retrieval

1. Create a directory called **data** in your R project folder.
2. Click on the SpatialAnalysisData link (**URL**: `https://mega.nz/file/hYk02AyK#4knQ1zcaIxKTN_GX3JN8ZyOPD4lXjynIbA3PvpvSaG4`).
3. Click on **Download** and save the file in the **data** directory. A zip compressed archive file (**SpatialAnalysisData.zip**; about 81.5 MB large) will be downloaded.
4. Unzip the archive file with the following **R** command:

```
destfile <- "data/SpatialAnalysisData.zip"

unzip(destfile, exdir = "data/")
```

If you get a warning message, such as

```
Warning:
In unzip(destfile, exdir = "data/") : error 1 in extracting from zip file"))
```

then the file may have not been downloaded into your directory.

Check:

- if you have a typo in the **destfile**,
- if you are in the correct working directory (with the command **getwd()**), or
- if the zip archive file is in the **data** directory of your working directory.

The latter can be checked with the **R** command **file.exists()**:

```
file.exists("data/SpatialAnalysisData.zip")
```

If the outcome is **[1] "FALSE"** then the file is missing. Find the file and move it into the **data** directory.

Once successful, you should have the following files in your **data** directory:

- **FoodAccess2015.xlsx**
- **gz_2010_24_140_00_500k.zip**
- **gz_2010_24_140_00_500k.dbf**
- **gz_2010_24_140_00_500k.prj**
- **gz_2010_24_140_00_500k.shp**
- **gz_2010_24_140_00_500k.shx**
- **gz_2010_24_140_00_500k.xml**
- **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).cpg**
- **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).csv**
- **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).dbf**
- **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).prj**
- **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).shp**
- **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).shx**
- **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).xml**
- **MD_counties_CT.dbf**

- `MD_counties_CT.prj`
- `MD_counties_CT.shp`
- `MD_counties_CT.shx`
- `MD_Covid19_VacSites_2021-04-04.csv`
- `SpatialAnalysisData.zip`

## 3.2 Data Sources and Description

### Food Access Research Atlas Data (2015 Dataset)

In 2008 the U.S. Congress directed the USDA to assess the extend of areas with limited access to fresh food (U.S. Department of Agriculture, 2009). Subsequently, the USDA Economic Research Service (USDA-ERS) developed a mapping tool called "the Food Desert Locator" (Ver Ploeg *et al.*, 2011). The Food Desert Locator allows to identify census tracts that qualify as food deserts [as defined by USDA, Treasury, and HHS (Ver Ploeg *et al.*, 2011)].

We are using this dataset as it lists (among more than 140 parameters) low-income and low-vehicle access census tracts.

The data is compiled into an Excel workbook (download size approx. 75 Mb). The workbook has three sheets, `Read me`, `Variable Lookup`, and `Food Access Research Atlas`. The latter contains the data aggregated based on data from the 2010 census.

Definitions and data sources can be accessed at https://www.ers.usda.gov/data-products/food-access-research-atlas/documentation/.

| | |
|---|---|
| Source: | USDA Economic Research Service (USDA-ERS), available at https://www.ers.usda.gov/data-products/food-access-research-atlas/download-the-data/ |
| Source URL | https://www.ers.usda.gov/webdocs/DataFiles/80591/DataDownload2015.xlsx?v=5276.5, retrieved on Apr 4, 2021 |

### Census Tracts

Food Atlas data is based on census tracts of the 2010 census. Census tracts are (relatively stable) statistical subdivisions of counties that ideally have 4,000 people (range: 1,200–8,000 people) (U.S. Census Bureau, 2019). To map, census tracts polygons are required.

| | |
|---|---|
| Source: | Census Tracts for Maryland, United States Census Bureau, Cartographic Boundary Files – Shapefile, available at https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.2010.html |

| | |
|---|---|
| URL (Maryland): | https://www2.census.gov/geo/tiger/GENZ2010/gz_2010_24_140_00_500k.zip, retrieved on Apr 4, 2021 |

Census tract polygons are stored in an ESRI shapefile collection as a **zip** archive file. The **zip** file contains 5 files, a shapefile (**.shp**), 3 necessary auxiliary files (**.dbf**, **.prj**, **.shx**), and an additional **.xml** file. The **.xml** file is not necessary for spatial analysis.

### List of COVID-19 Vaccination Sites

A list of COVID-19 vaccination sites in Maryland was retrieved from the Maryland Department of Health Coronavirus website (https://coronavirus.maryland.gov)

| | |
|---|---|
| Source: | All Maryland Vaccination Sites, https://coronavirus.maryland.gov/datasets/all-maryland-vaccination-sites-1 |
| URL (.csv) | https://opendata.arcgis.com/datasets/d677f143334648a1a40b84d94df8e134_4.csv, retrieved on April 4, 2021 |
| Credit: | Maryland Department of Health COVID-19 Testing Task Force |
| Metadata: | https://www.arcgis.com/sharing/rest/content/items/d677f143334648a1a40b84d94df8e134/info/metadata/metadata.xml?format=default&output=html |

The dataset was slightly modified for educational purposes. An additional variable (column) — **Location** — was added and variables rearranged. The **Location** variable contains GPS coordinates of the vaccination sites in a **(latitude, longitude)** format. This format is often encountered in datasets that provide GPS information in tabular form.

### Maryland Physical Boundaries — County Boundaries (Detailed)

Shapefile collection containing boundaries of Maryland counties.

| | |
|---|---|
| Source: | Maryland Physical Boundaries - County Boundaries (Detailed) https://data.imap.maryland.gov/datasets/maryland-physical-boundaries-county-boundaries-detailed?geometry=-80.968%2C38.061%2C-73.569%2C39.559https://data.imap.maryland.gov/datasets/maryland-physical-boundaries-county- |
| URL (.zip) | https://opendata.arcgis.com/datasets/2315ef0b071a4ec59420e3d342dbcfe2_0.zip?outSR=%7B%22latestWkid%22%3A3857%2C%22wkid%22%3A102100%7D, retrieved on Apr 4, 2021 |
| Credit: | MD iMAP, SHA |

| | |
|---|---|
| Metadata: | https://www.arcgis.com/sharing/rest/content/items/2315ef0b071a4ec59420e3d342dbcfe2/info/metadata/metadata.xml?format=default&output=html |

### Maryland County Boundaries

Shapefile collection containing boundaries of Maryland counties. Based on unified census tract boundaries (source of the original data: Census Tracts).

# Chapter 4

## *Data Import and Wrangling*

This section shows how to import data, and perform some "data wrangling." Data wrangling refers to steps taken to make data more useful to downstream applications. Here we show approaches to clean up the raw data, i.e., remove missing values, filter out unnecessary information, and merge data sets. Data visualization and transformation of non-spatial data to spatial simple feature objects is introduced in the next section.

It is assumed that you already downloaded and unpacked the data into the **`data`** folder in your **`R project`** folder. If not, please do so before continuing (see Section 3.1).

### 4.1 Import Data into R

#### Food Access Research Atlas Data (2015 Data Set)

As mentioned in Section 3.2, the data is compiled into an Excel workbook. Excel workbooks (**`.xlsx`** format) can be imported directly into **`R`** by several packages, including **`xlsx`**, **`XLconnect`**, **`read_xlsx`**, and **`openxlsx`**. In this section we are using the package **`openxlsx`**. In contrast to the **`xlsx`** and **`XLConnect`** packages, **`openxlsx`** does not depend on **`Java`**. For detailed information on the **`openxlsx`** package please consult the documentation available:

- https://www.rdocumentation.org/packages/openxlsx/versions/4.2.3
- https://cran.r-project.org/web/packages/openxlsx/openxlsx.pdf

The **`openxlsx`** package provides functions to work with Excel worksheets, including functions to extract the of names of worksheets as well as to import selected worksheets.

##### Explore the structure of the workbook

The function **`openxlsx::getSheetNames()`** extracts the names of Excel worksheets.

```
openxlsx::getSheetNames("data/FoodAccess2015.xlsx")
```

```
## [1] "Read Me"                    "Variable Lookup"
## [3] "Food Access Research Atlas"
```

The output shows that the workbook has 3 worksheets:

1. Read Me
2. Variable Lookup
3. Food Access Research Atlas

#### Import relevant sheets

The worksheet **`Food Access Research Atlas`** contains the data we are interested in. The following code imports this worksheet as a data frame and assigns the data frame to the object **`FoodAccess2015`**. Note, this is a large file. Reading it into **`R`** will take a while. Be patient.

```
FoodAccess2015 <- openxlsx::read.xlsx("data/FoodAccess2015.xlsx",
                                      sheet = "Food Access Research Atlas")
```

Alternatively, the index number of the worksheet can be used:

```
FoodAccess2015 <- openxlsx::read.xlsx("data/FoodAccess2015.xlsx",
                                      sheet = 3)
```

Next, the structure of the data frame is explored with **`str()`**. The output of the following code lists the first 20 variables (set by the argument **`list.len = 20`**). If all variables should be displayed, the argument would be **`list.len = ncol()`**.

```
str(FoodAccess2015, list.len = 20, strict.width = "cut")
```

```
## 'data.frame':    72864 obs. of  147 variables:
##  $ CensusTract         : chr  "01001020100" "01001020200" "01001020300" "0100"..
##  $ State               : chr  "Alabama" "Alabama" "Alabama" "Alabama" ...
##  $ County              : chr  "Autauga" "Autauga" "Autauga" "Autauga" ...
##  $ Urban               : num  1 1 1 1 1 1 1 0 0 0 ...
##  $ POP2010             : num  1912 2170 3373 4386 10766 ...
##  $ OHU2010             : num  693 743 1256 1722 4082 ...
##  $ GroupQuartersFlag   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NUMGQTRS            : num  0 181 0 0 181 0 36 0 0 14 ...
##  $ PCTGQTRS            : num  0 0.0834 0 0 0.0168 ...
##  $ LILATracts_1And10   : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ LILATracts_halfAnd10: num  0 0 0 0 0 0 1 0 0 0 ...
##  $ LILATracts_1And20   : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ LILATracts_Vehicle  : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ HUNVFlag            : num  0 0 0 0 1 0 1 1 0 0 ...
##  $ LowIncomeTracts     : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ PovertyRate         : num  10 18.2 19.1 3.3 8.5 ...
```

```
##  $ MedianFamilyIncome  : num  74750 51875 52905 68079 77819 ...
##  $ LA1and10            : num  1 0 1 1 1 1 1 0 0 1 ...
##  $ LAhalfand10         : num  1 1 1 1 1 1 1 1 0 0 1 ...
##  $ LA1and20            : num  1 0 1 1 1 1 1 1 0 0 0 ...
##   [list output truncated]
```

The data frame has 72,864 observations (rows) and 147 variables (columns). Note that the variable **CensusTract** is a character string (**chr**). Furthermore, variables such as **Urban** or **POP2010** were imported as real numbers (double precision numbers, **num**).

### Census Tracts Polygons

The census tracts polygons are stored in a so called "shapefile collection" with the filename prefix **gz_2010_24_140_00_500k**. A shapefile collection consists of a number of different file types with the same filename prefix. These files contain geometric location and features as well attributes to these features. The files need to be stored in the same directory. This filing format was developed and is maintained by the Environmental Systems Research Institute (ESRI) (ESRI, 2020).

Four of these file types are required when performing spatial analysis:

- **.shp**, the shapefile itself, contains the features' geometry,
- **.shx**, contains the index of the feature geometry,
- **.dbf**, a table in **dBASE** that contains the attributes of the features, and
- **.prj**, a text file that contains the coordinate reference system information (**CRS**) of the features.

The function **st_read()** from the package **sf** imports shapefiles into **R**. While only the shapefile proper (**.shp**) is called, the function needs the other 3 files to properly import the geometric features and their attributes.

```
MD_CensusTracts_2010 <- sf::st_read("data/gz_2010_24_140_00_500k.shp")

MD_CensusTracts_2010
```

```
## Simple feature collection with 1403 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -79.48765 ymin: 37.91172 xmax: -75.04894 ymax: 39.72304
## Geodetic CRS:  NAD83
## First 10 features:
##                  GEO_ID STATE COUNTY  TRACT    NAME  LSAD CENSUSAREA
## 1  1400000US24015031400    24    015 031400     314 Tract     16.039
## 2  1400000US24017850101    24    017 850101 8501.01 Tract      9.718
## 3  1400000US24017850706    24    017 850706 8507.06 Tract      2.698
## 4  1400000US24017850710    24    017 850710 8507.10 Tract      1.401
## 5  1400000US24017850901    24    017 850901 8509.01 Tract      2.082
## 6  1400000US24017851500    24    017 851500    8515 Tract      5.010
```

```
## 7  1400000US24017990000    24    017 990000    9900 Tract      0.000
## 8  1400000US24019970702    24    019 970702 9707.02 Tract     43.137
## 9  1400000US24019970804    24    019 970804 9708.04 Tract     87.808
## 10 1400000US24019970900    24    019 970900    9709 Tract    211.650
##                          geometry
## 1  MULTIPOLYGON (((-76.15435 3...
## 2  MULTIPOLYGON (((-77.09875 3...
## 3  MULTIPOLYGON (((-76.94249 3...
## 4  MULTIPOLYGON (((-76.96267 3...
## 5  MULTIPOLYGON (((-76.90672 3...
## 6  MULTIPOLYGON (((-76.96267 3...
## 7  MULTIPOLYGON (((-77.08633 3...
## 8  MULTIPOLYGON (((-76.25027 3...
## 9  MULTIPOLYGON (((-76.16439 3...
## 10 MULTIPOLYGON (((-76.04837 3...
```

The first line of the code reads in the data. You may get a warning similar to the following:

```
## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

The warning can be ignored. It basically tells you that if the package **`dplyr`** would be loaded, "traditional" data frames may be replaced by **`tibble`** data frames. This will not affect our analysis.

The second line of code (**`MD_CensusTracts_2010`**) prints out the first 10 entries of the read in shapefile. It tells us that **`MD_CensusTracts_2010`** is a "simple feature" object (**`sf`**) with 1,403 and 7 fields (plus a geometry column). The geometric features are multi polygons. The coordinate reference system (**`CRS`**) is **`NAD83`**. **`NAD83`** stands for the North American Datum of 1983. The **`NAD83(2011)`** is the current geodetic system that is used for the continental U.S. (Section 2.3) (National Geodetic Survey, 2018).

### Maryland Counties (Physical) Boundaries

The **`Maryland_Physical_Boundaries_-County_Boundaries(Detailed)`** shapefile collection contains polygons of the Maryland counties (physical) boundaries that take into consideration waterways (such as the tributaries of Chesapeake Bay).

Import the file with **`st_read()`**.

```
MD_counties_map <-
sf::st_read("data/Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).shp")

MD_counties_map
```

```
## Simple feature collection with 24 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -8848486 ymin: 4564403 xmax: -8354439 ymax: 4825752
## Projected CRS: WGS 84 / Pseudo-Mercator
## First 10 features:
##    OBJECTID         COUNTY DISTRICT COUNTY_FIP COUNTYNUM CREATION_D LAST_UPDAT
## 1         1       Allegany        6          1         1 2010-01-28 2010-01-28
## 2         2   Anne Arundel        5          3         2 2006-04-18 2006-04-18
## 3         3      Baltimore        4          5         3 2006-10-09 2006-10-09
## 4         4 Baltimore City        0        510        24 2006-04-18 2009-11-16
## 5         5        Calvert        5          9         4 2010-01-28 2010-01-28
## 6         6       Caroline        2         11         5 2007-05-21 2008-07-30
## 7         7        Carroll        7         13         6 2008-06-16 2012-01-17
## 8         8          Cecil        2         15         7 2006-04-18 2008-08-20
## 9         9        Charles        5         17         8 2009-06-08 2009-06-08
## 10       10     Dorchester        1         19         9 2007-02-08 2007-02-22
##                          geometry
## 1  MULTIPOLYGON (((-8721085 48...
## 2  MULTIPOLYGON (((-8527741 47...
## 3  MULTIPOLYGON (((-8523507 48...
## 4  MULTIPOLYGON (((-8519244 47...
## 5  MULTIPOLYGON (((-8531762 46...
## 6  MULTIPOLYGON (((-8432189 47...
## 7  MULTIPOLYGON (((-8556981 47...
## 8  MULTIPOLYGON (((-8441053 48...
## 9  MULTIPOLYGON (((-8580309 46...
## 10 MULTIPOLYGON (((-8439760 46...
```

This dataset uses a projected **CRS** that is based on **WGS 84** using a pseudo-mercator projection. Pseudo-mercator projections are used by many web based mapping apps.

### List of Vaccination Sites

**MD_Covid19_VacSites_2021-04-04.csv** is a comma separated file that lists COVID-19 vaccination sites in Maryland. The table is read into **R** with **read.table()** and assigned to the object **VaccineSites**. Its structure is explored with **str()**.

```
VaccineSites <- read.table(file = "data/MD_Covid19_VacSites_2021-04-04.csv",
                           sep = ",",
                           na.strings = c("","" ", "NA"),
                           header = TRUE,
                           quote = "\"",
                           fill = TRUE)

str(VaccineSites, list.len = 20, strict.width = "cut")
```

```
## 'data.frame':    556 obs. of  48 variables:
##  $ OBJECTID               : int  84 87 92 93 95 96 98 99 100 102 ...
##  $ facilityid             : chr  "Anna_Lumi_83" "Lanh_Lumi_86" "Balt_Grac_91"..
##  $ name                   : chr  "Luminis Health Anne Arundel Medical Center"..
##  $ fulladdr               : chr  "2001 Medical Parkway, Annapolis, MD 21401""..
##  $ Location               : chr  "(38.990512076102, -76.5341664410021)" "(38"..
##  $ X                      : num  -76.5 -76.9 -76.6 -77 -76.9 ...
##  $ Y                      : num  39 39 39.3 39.6 39.2 ...
##  $ municipality           : chr  "Annapolis" "Lanham" "Baltimore" "Westminst"..
##  $ CreationDate           : chr  "1970/01/01 00:00:00+00" "1970/01/01 00:00:"..
##  $ Creator                : logi  NA NA NA NA NA NA ...
##  $ EditDate               : chr  "1970/01/01 00:00:00+00" "1970/01/01 00:00:"..
##  $ Editor                 : chr  NA NA NA NA ...
##  $ ActiveYesNo            : chr  "Yes" "Yes" "Yes" "Yes" ...
##  $ site_type              : chr  "Hospital" "Hospital" "Hospital" "Hospital" ..
##  $ appt_required          : logi  NA NA NA NA NA NA ...
##  $ operationalhours       : chr  "Mon - Fri 7 am - 7 pm" "Mon - Fri 7am - 7 "..
##  $ docorder_required      : logi  NA NA NA NA NA NA ...
##  $ costfree               : logi  NA NA NA NA NA NA ...
##  $ cost_outpocket         : logi  NA NA NA NA NA NA ...
##  $ drivethru              : logi  NA NA NA NA NA NA ...
##   [list output truncated]
```

This data frame contains 556 observations and 48 variables. The data frame as a variable called **`Location`** that has GPS coordinates presented as **`(longitude, latitude)`**. The data frame also has *xy* coordinates (variables **`X`** and **`Y`**). However for the purpose of practice, we will not use them.

## 4.2 Data Manipulation (Wrangling)

Once the data are imported, in most cases the data needs to be manipulated to continue with the analysis.

Our aim is to map "vaccination deserts" in Maryland and in Baltimore City. We have 3 datasets, namely 2 (non spatial) data frames, **`FoodAccess2015`** and **`VaccineSites`**, and a spatial object, **`MD_CensusTracts`**. The two data frames contain geographic and demographic information (loca-

tion of vaccination sites in Maryland, demographic profiles of the census tracts within the U.S.), and the spatial (simple feature) object the geographic boundaries of census tracts in Maryland.

We need to:

1. Extract the data for Maryland and Baltimore City, where possible,
2. Combine demographic profiles and geographic boundaries, and
3. Convert the data frame with location information on vaccination sites into a spatial (simple feature) object.

This section describes how to extract data from a data frame (spatial and non-spatial) as well as merging data frames. Conversion of data frames into spatial objects, manipulation of spatial objects, and mapping is discussed in the next section.

### Subsetting (Filtering) Data Frames

We are interested in data within Maryland and Baltimore City. To extract the data we use the function **`subset()`** (part of the core **`R`** installation).

#### Food Access Data

The data frame contains data for the entire US. List the first five variables with **`str()`**.

```
str(FoodAccess2015, list.len = 5, strict.width = "cut")
```

```
## 'data.frame':    72864 obs. of  147 variables:
##  $ CensusTract         : chr  "01001020100" "01001020200" "01001020300" "0100"..
##  $ State               : chr  "Alabama" "Alabama" "Alabama" "Alabama" ...
##  $ County              : chr  "Autauga" "Autauga" "Autauga" "Autauga" ...
##  $ Urban               : num  1 1 1 1 1 1 1 0 0 0 ...
##  $ POP2010             : num  1912 2170 3373 4386 10766 ...
##   [list output truncated]
```

The output shows that the data frame contains the variable **`State`** and **`County`**. The following code extracts only the rows for the state of Maryland, assigns the new data frame to the object **`FoodAccess2015_MD`**, and shows the structure of the first 20 variables of the new object.

```
FoodAccess2015_MD <- subset(FoodAccess2015, State == "Maryland")

str(FoodAccess2015_MD, list.len = 20, strict.width = "cut")
```

```
## 'data.frame':    1399 obs. of  147 variables:
##  $ CensusTract         : chr  "24001000100" "24001000200" "24001000300" "2400"..
##  $ State               : chr  "Maryland" "Maryland" "Maryland" "Maryland" ...
##  $ County              : chr  "Allegany" "Allegany" "Allegany" "Allegany" ...
##  $ Urban               : num  0 0 0 1 1 1 1 1 1 1 ...
##  $ POP2010             : num  3718 4564 2780 3022 2734 ...
##  $ OHU2010             : num  1523 1284 1133 1350 1044 ...
##  $ GroupQuartersFlag   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NUMGQTRS            : num  39 1517 155 14 345 ...
##  $ PCTGQTRS            : num  0.01049 0.33238 0.05576 0.00463 0.12619 ...
##  $ LILATracts_1And10   : num  1 0 0 1 1 0 0 0 0 0 ...
##  $ LILATracts_halfAnd10: num  1 0 0 1 1 1 1 1 0 0 ...
##  $ LILATracts_1And20   : num  0 0 0 1 1 0 0 0 0 0 ...
##  $ LILATracts_Vehicle  : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ HUNVFlag            : num  0 0 0 0 0 0 1 1 0 0 ...
##  $ LowIncomeTracts     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ PovertyRate         : num  6.6 13.7 20.2 12.8 56 ...
##  $ MedianFamilyIncome  : num  56875 60943 42727 48831 34519 ...
##  $ LA1and10            : num  1 0 0 1 1 0 0 0 0 0 ...
##  $ LAhalfand10         : num  1 0 0 1 1 1 1 1 0 0 ...
##  $ LA1and20            : num  0 0 0 1 1 0 0 0 0 0 ...
##   [list output truncated]
```

To verify that the new object only contains Maryland, call **`unique()`**:

```
unique(FoodAccess2015_MD$State)
```

```
## [1] "Maryland"
```

While it is possible to continue to work with the entire **`FoodAccess2015`** data frame, removing unwanted variables (columns) reduces the size of the data frame.

Based on the documentation https://www.ers.usda.gov/data-products/food-access-research-atlas/documentation/, we are interested in:

- Urban/Rural designation,
- County,
- Total population,
- Low Income, and
- Vehicles access.

To identify variable names in the **`FoodAccess2015`** data frame we import the **`Variable Lookup`** sheet from the **`FoodAccess2015.xlsx`** Excel workbook (sheet 2, see above), and show the first six entries (rows) with the **`head()`** function.

```
FoodAccessVar <- openxlsx::read.xlsx("data/FoodAccess2015.xlsx",
                                     sheet = "Variable Lookup")

head(FoodAccessVar, 6)
```

```
##         Field               LongName
## 1 CensusTract           Census tract
## 2       State                  State
## 3      County                 County
## 4       Urban            Urban tract
## 5     POP2010 Population, tract total
## 6     OHU2010    Housing units, total
##                                  Description
## 1                        Census tract number
## 2                                 State name
## 3                                 County name
## 4                        Flag for urban tract
## 5           Population count from 2010 census
## 6 Occupied housing unit count from 2010 census
```

To identify the variable name for low income, we can query the **FoodAccessVar** data frame using a regular expression (**regex**). Regular expressions are strings of text that help to find text pattern. The regular expression **"[Ll]ow[[:space:]][Ii]ncome"** for example will match the character strings **Low income**, **low income**, **Low Income**, and **low Income**.

The function **grepl()** is a function that will check if the content of an element of a vector (such the cell of a table) matches a regular expression. It will return a logical vector (**TRUE/FALSE**). Therefore we can use **grepl()** to extract rows from a data frame that contain the regular expression.

```
FoodAccessVar_low_income <- subset(FoodAccessVar,
                                   grepl("[Ll]ow[[:space:]][Ii]ncome",
                                   FoodAccessVar$LongName) == TRUE)

FoodAccessVar_low_income[,1:2]
```

```
##                   Field
## 10    LILATracts_1And10
## 11 LILATracts_halfAnd10
## 12    LILATracts_1And20
## 13   LILATracts_Vehicle
## 15      LowIncomeTracts
##                                                 LongName
## 10       Low income and low access tract measured at 1 mile for urban areas and 10 miles
            for rural areas
```

```
## 11      Low income and low access tract measured at 1/2 mile for urban areas and 10 miles
          for rural areas
## 12        Low income and low access tract measured at 1 mile for urban areas and 20 miles
            for rural areas
## 13      Low income and low access tract using vehicle access or low income and low access
          tract measured at 20 miles
## 15                                            Low income tract
```

The first line of the code extracts the rows of the **FoodAccessVar** data frame that have regular expression in the variable **LongName**. The filtered data frame is assigned to the object **FoodAccessVar_low_income**.

The second line displays the first two columns of the new data frame/object.

We repeat the code above for vehicle access, and extract rows that contain the expression **[Vv]ehicle [Aa]ccess]**. The filtered data frame is assigned to the object **FoodAccessVar_vehicle_access**.

```
FoodAccessVar_vehicle_access <- subset(FoodAccessVar,
                                       grepl("[Vv]ehicle[[:space:]][Aa]ccess",
                                       FoodAccessVar$LongName) == TRUE)

FoodAccessVar_vehicle_access[,1:2]
```

```
##                  Field
## 13  LILATracts_Vehicle
## 14            HUNVFlag
## 25  LATractsVehicle_20
## 54          lahunvhalf
## 55     lahunvhalfshare
## 80             lahunv1
## 81        lahunv1share
## 106           lahunv10
## 107      lahunv10share
## 132           lahunv20
## 133      lahunv20share
##                              LongName
## 13      Low income and low access tract using vehicle access or low income and low access
          tract measured at 20 miles
## 14                                        Vehicle access, tract with low vehicle access
## 25                  Low access tract using vehicle access and at 20 miles  in rural areas
## 54               Vehicle access, housing units without and low access at 1/2 mile, number
## 55                Vehicle access, housing units without and low access at 1/2 mile, share
## 80                 Vehicle access, housing units without and low access at 1 mile, number
## 81                  Vehicle access, housing units without and low access at 1 mile, share
## 106              Vehicle access, housing units without and low access at 10 miles, number
## 107               Vehicle access, housing units without and low access at 10 miles, share
## 132              Vehicle access, housing units without and low access at 20 miles, number
## 133               Vehicle access, housing units without and low access at 20 miles, share
```

Based on the above outputs, it looks like that the variables **HUNVFlag** and **LowIncomeTracts** are of interest. The first identifies tracts with low vehicle access, the second low-income tracts.

We therefore subset the **FoodAccess2015_MD** data frame to only keep the **CensusTract**, **County**, **Urban**, **POP2010**, **LowIncomeTracts**, and **HUNVFlag** variables (columns).

```
FoodAccess2015_MD <- subset(FoodAccess2015_MD, select = c(CensusTract, County, Urban,
                                                          POP2010, LowIncomeTracts,
                                                          HUNVFlag))

str(FoodAccess2015_MD, strict.width = "cut")
```

```
## 'data.frame':    1399 obs. of  6 variables:
##  $ CensusTract    : chr  "24001000100" "24001000200" "24001000300" "240010004"..
##  $ County         : chr  "Allegany" "Allegany" "Allegany" "Allegany" ...
##  $ Urban          : num  0 0 0 1 1 1 1 1 1 1 ...
##  $ POP2010        : num  3718 4564 2780 3022 2734 ...
##  $ LowIncomeTracts: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ HUNVFlag       : num  0 0 0 0 0 0 1 1 0 0 ...
```

Let's see if the data set contains suspicious data. In particular, we should check whether the data for the census tracts are complete and make sense.

First, let's verify that there are no missing values with the **colSums(is.na())** nested function.

```
colSums(is.na(FoodAccess2015_MD))
```

```
##     CensusTract          County           Urban         POP2010 LowIncomeTracts
##               0               0               0               0               0
##        HUNVFlag
##               0
```

The output shows that there are no missing values.

The variable **POP2010** reports the number of residents in a census tract. If reported correctly, all entries should be greater than 0. A census tract with no residents would just not make any sense. The population should range between 1,800 and 8,000 (U.S. Census Bureau, 2019). To check whether there are entries without residents, we first search for entries without residents and then count them with **nrow()**. **nrow()** is a function that shows you how many rows are in a data frame.

```
bad_census_tracts <- subset(FoodAccess2015_MD, POP2010 == 0)

nrow(bad_census_tracts)
```

```
## [1] 9
```

It looks like there are nine entries without residents. Let's see who they are by calling **`bad_census_tracts`**. To shorten the output, we only display the census tract, the county, (columns/-variables 1 and 2) and **`POP2010`** (variables 4).

```
bad_census_tracts[,c(1:2, 4)]
```

```
##       CensusTract       County POP2010
## 30065 24005980000    Baltimore       0
## 30066 24005980100    Baltimore       0
## 30067 24005980200    Baltimore       0
## 30191 24019990000   Dorchester       0
## 30821 24035990100 Queen Anne's       0
## 30839 24037990000   St. Mary's       0
## 30847 24039990100     Somerset       0
## 30924 24047980000    Worcester       0
## 30925 24047990000    Worcester       0
```

We remove these tracts by keeping all the entries that have a **`POP2010`** value that is not 0 (hence the **`!=`** operator).

```
FoodAccess2015_MD_good <- subset(FoodAccess2015_MD, POP2010 != 0)
```

Calling **`nrow(subset(FoodAccess2015_MD_good, POP2020 == 0))`** should return **`[1] 0`**

```
nrow(subset(FoodAccess2015_MD_good, POP2010 == 0))
```

```
## [1] 0
```

And it does.

We are also interested in identifying possible vaccination deserts in Baltimore City, and could subset the Maryland Food Access data set for Baltimore City. However, at this point the Maryland data set

does not contain any geographic information. The subsection "Merging Data Frames" shows how to add geographic information to the data set. Once that is done we will create a subset for Baltimore City.

### Census Tracts

The census tract data is for the entire state of Maryland and contains the census tract boundaries (as polygons). For our purposes, we will combine this data set with the Food Access data set.

Let's check if all entries do have geographic information (i.e., a geometry entry).

```
colSums(is.na(MD_CensusTracts_2010))
```

```
##     GEO_ID      STATE     COUNTY      TRACT       NAME       LSAD CENSUSAREA
##          0          0          0          0          0          0          0
##   geometry
##          0
```

There are no missing values.

### Vaccination Sites

The **`VaccineSites`** data frame lists COVID-19 vaccination sites in Maryland. To list all the variables (columns) we can use the function **`colnames()`**.

```
colnames(VaccineSites)
```

```
##  [1] "OBJECTID"                "facilityid"
##  [3] "name"                    "fulladdr"
##  [5] "Location"                "X"
##  [7] "Y"                       "municipality"
##  [9] "CreationDate"            "Creator"
## [11] "EditDate"                "Editor"
## [13] "ActiveYesNo"             "site_type"
## [15] "appt_required"           "operationalhours"
## [17] "docorder_required"       "costfree"
## [19] "cost_outpocket"          "drivethru"
## [21] "pedaccess"               "transitAccess"
## [23] "test_antigen"            "test_antibody"
## [25] "other_notes"             "insurance_accepted"
## [27] "medicaid"                "cost_other"
## [29] "cost_notes"              "test_rapid"
## [31] "schedule_url"            "online_scheduling"
## [33] "scheduling_contact"      "scheduling_contact_phone"
## [35] "scheduling_contact_email" "test_pcr"
```

```
## [37] "website_url"              "X_coord"
## [39] "Y_coord"                  "test_pediatric"
## [41] "multi_language"           "test_pediatric_notes"
## [43] "created_user"             "created_date"
## [45] "last_edited_user"         "last_edited_date"
## [47] "County"                   "PreRegistrationURL"
```

The data frame has 48 variables including a variable with GPS information (**Location**). Let's check if all entries have GPS information.

```
sum(is.na(VaccineSites$Location))
```

```
## [1] 0
```

All entries have GPS coordinates.

***Remove unwanted variables***

The **VaccineSites** data frame contains information that is not relevant for our analysis. We are interested in the GPS coordinates (**Location**), and the name of the facility (**name**). Furthermore, information on whether a physician referral is required (**docorder_required**), whether it is cost-free (**costfree**), and whether the facility is accessible by foot (**pedaccess**) would be interesting in the context of affordable service.

The following code subsets the **VaccineSites** data frame for the above variables, and checks whether data for all variables is available.

```
VaccineSites_mod <- subset(VaccineSites, select = c("name", "Location", "pedaccess",
                                                    "docorder_required", "costfree"))

str(VaccineSites_mod, strict.width = "cut")
```

```
## 'data.frame':    556 obs. of  5 variables:
##  $ name             : chr  "Luminis Health Anne Arundel Medical Center" "Lumi"..
##  $ Location         : chr  "(38.990512076102, -76.5341664410021)" "(38.982597"..
##  $ pedaccess        : logi  NA NA NA NA NA NA ...
##  $ docorder_required: logi  NA NA NA NA NA NA ...
##  $ costfree         : logi  NA NA NA NA NA NA ...
```

```
colSums(is.na(VaccineSites_mod))
```

```
##             name         Location         pedaccess docorder_required
##                0                0              556              556
##         costfree
##              556
```

It turns out that we only have information on the name and coordinates for all facilities. Therefore, we drop the other variables.

```
VaccineSites_mod <- subset(VaccineSites, select = c("name", "Location"))
```

```
str(VaccineSites_mod, strict.width = "cut")
```

```
## 'data.frame':    556 obs. of  2 variables:
##  $ name    : chr  "Luminis Health Anne Arundel Medical Center" "Luminis Healt"..
##  $ Location: chr  "(38.990512076102, -76.5341664410021)" "(38.9825972406456, "..
```

```
colSums(is.na(VaccineSites_mod))
```

```
##     name Location
##        0        0
```

As so often with **R**, there are alternative ways to accomplish the same task. **R** has functions that remove entries if they have missing values. For example the function **`na.omit()`** removes all entries that have missing values. Another function, **`complete.cases()`** removes either all variables or all rows that have some missing values. We only want to drop variables that only have missing values. We can do so with the following code. It will remove all variables that only contain missing values from the subsetted data frame **`VaccineSites_mod2`**.

```
VaccineSites_mod2 <- subset(VaccineSites, select = c("name",
                                                     "Location",
                                                     "pedaccess",
                                                     "docorder_required",
                                                     "costfree"))

VaccineSites_mod2 <- VaccineSites_mod2[, colSums(is.na(VaccineSites_mod2))
                                       != nrow(VaccineSites_mod2)]

colSums(is.na(VaccineSites_mod2))
```

```
##     name Location
##        0        0
```

```
str(VaccineSites_mod2, strict.width = "cut")
```

```
## 'data.frame':    556 obs. of  2 variables:
##  $ name    : chr  "Luminis Health Anne Arundel Medical Center" "Luminis Healt"..
##  $ Location: chr  "(38.990512076102, -76.5341664410021)" "(38.9825972406456, "..
```

### Merging Data Frames

**MD_CensusTracts_2010** has the geographic boundaries of the census tracts in Maryland. To add the attributes/properties to the census tracts that will allow us to map census tracts that have limited access to vaccination sites, and/or are defined as low income, and/or have limited access to a vehicle, etc. the **FoodAccess2015_MD** data frame is merged with **MD_CensusTracts_2010**.

The function **merge()** allows to combine data frames based on a common variable. Here the common variable is the census tract. Naturally, the requirement for this function is that both data frames have one variable in common.

#### Prepare data frames for merging

Both data frames/objects (**FoodAccess2015_MD_good**, **MD_CensusTracts_2010**) have census tract information. However, the data frames encode the census tracts differently, and have different variable names. Furthermore, **FoodAccess2015_MB_good** has 13 fewer observations (in part because we removed a few observations).

```
# Structure of Food Access
str(FoodAccess2015_MD_good, list.len = 20, strict.width = "cut")
```

```
## 'data.frame':    1390 obs. of  6 variables:
##  $ CensusTract    : chr  "24001000100" "24001000200" "24001000300" "240010004"..
##  $ County         : chr  "Allegany" "Allegany" "Allegany" "Allegany" ...
##  $ Urban          : num  0 0 0 1 1 1 1 1 1 1 ...
##  $ POP2010        : num  3718 4564 2780 3022 2734 ...
##  $ LowIncomeTracts: num  1 1 1 1 1 1 1 1 1 1 ...
##  $ HUNVFlag       : num  0 0 0 0 0 0 1 1 0 0 ...
```

```
# Struture of MD_CensusTracts
str(MD_CensusTracts_2010, list.len = 5, strict.width = "cut")
```

```
## Classes 'sf' and 'data.frame':   1403 obs. of  8 variables:
##  $ GEO_ID    : chr  "1400000US24015031400" "1400000US24017850101" "1400000US2"..
##  $ STATE     : chr  "24" "24" "24" "24" ...
##  $ COUNTY    : chr  "015" "017" "017" "017" ...
##  $ TRACT     : chr  "031400" "850101" "850706" "850710" ...
##  $ NAME      : chr  "314" "8501.01" "8507.06" "8507.10" ...
##   [list output truncated]
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA N..
##   ..- attr(*, "names")= chr [1:7] "GEO_ID" "STATE" "COUNTY" "TRACT" ...
```

The tract ID of the **FoodAccess2015_MD** data frame contains the state ID (**24**), a three-digit county ID (f.e. **001** for Allegany county), and a six-digit identifier of the census tract which unique within each county. In contrast, **MD_CensusTracts_2010** shows only the six-digit identifier of the census tract in each county. The two-digit state ID and the three-digit county ID are stored in a separate variable.

| | |
|---|---|
| FoodAccess2015_MD | 24001000100 |
| | 24001000200 |
| | ... |
| MD_CensusTracts_2010 | 000100 |
| | 000200 |
| | ... |

Furthermore, the variable name is different (**CensusTract** vs. **TRACT**).

The function **paste()** allows to combine variables (columns). The following code merges the three variables (columns) **STATE**, **COUNTY** and **TRACT** of the **MD_CensusTracts_2010** data frame and puts the new ID into a new variable **(CensusTract**)

```
# merge columns
MD_CensusTracts_2010$CensusTract <- paste(MD_CensusTracts_2010$STATE,
                                          MD_CensusTracts_2010$COUNTY,
                                          MD_CensusTracts_2010$TRACT,
                                          sep = "")
```

The code below checks the class of the modified data frame, and confirms that the **MD_CensusTracts_2010** is still a (spatial) simple feature (**sf**) object.

```
class(MD_CensusTracts_2010)
```

```
## [1] "sf"         "data.frame"
```

### Merge data frames

**MD_CensusTracts_2010** is a **sf** object and contains spatial data. When merging with non spatial data frames, the **sf** object that contains spatial data has to come first.

```
MD_CensusTracts_Map <- merge(MD_CensusTracts_2010, FoodAccess2015_MD_good,
                             by = "CensusTract", all.x = TRUE)

# dimension
dim(MD_CensusTracts_Map)
```

```
## [1] 1403   14
```

```
class(MD_CensusTracts_Map)
```

```
## [1] "sf"         "data.frame"
```

```
# structure (columns 1:13), 14 = geometry
str(MD_CensusTracts_Map[,c(1:13)], list.len = 13, strict.width = "cut")
```

```
## Classes 'sf' and 'data.frame':   1403 obs. of  14 variables:
##  $ CensusTract    : chr  "24001000100" "24001000200" "24001000300" "240010004"..
##  $ GEO_ID         : chr  "1400000US24001000100" "1400000US24001000200" "14000"..
##  $ STATE          : chr  "24" "24" "24" "24" ...
##  $ COUNTY         : chr  "001" "001" "001" "001" ...
##  $ TRACT          : chr  "000100" "000200" "000300" "000400" ...
##  $ NAME           : chr  "1" "2" "3" "4" ...
##  $ LSAD           : chr  "Tract" "Tract" "Tract" "Tract" ...
##  $ CENSUSAREA     : num  187.94 48.07 8.66 3.72 4.42 ...
##  $ County         : chr  "Allegany" "Allegany" "Allegany" "Allegany" ...
##  $ Urban          : num  0 0 0 1 1 1 1 1 1 1 ...
##  $ POP2010        : num  3718 4564 2780 3022 2734 ...
##  $ LowIncomeTracts: num  1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ HUNVFlag       : num  0 0 0 0 0 0 1 1 0 0 ...
##   [list output truncated]
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA N..
##   ..- attr(*, "names")= chr [1:13] "CensusTract" "GEO_ID" "STATE" "COUNTY" ...
```

The **by = "CensusTract"** argument of the **merge()** function merges the data frames based on the variable **CensusTract**. The argument **all.x = TRUE** will keep all entries of the first data frame and will add missing values if there is no match of the common variable (**CensusTract**) in the 2nd data frame. Entries that do have a matching **CensusTract** in the first data frame are excluded.

We should have at least 13 entries with missing data.

```
colSums(is.na(MD_CensusTracts_Map))
```

```
##     CensusTract          GEO_ID           STATE          COUNTY           TRACT
##               0               0               0               0               0
##            NAME            LSAD      CENSUSAREA          County           Urban
##               0               0               0              13              13
##         POP2010 LowIncomeTracts        HUNVFlag        geometry
##              13              13              13               0
```

Indeed, we have 13 entries with missing values. These census tracts will be mapped as having "no data."

***Subset for Baltimore City***

Last, we extract the data for Baltimore City from the **MD_CensusTracts_Map** data set. We can subset using the last three digits of the FIPS county code (stored in the variable **COUNTY**), which for Baltimore City is **510**.

```
BC_CensusTracts_Map <- subset(MD_CensusTracts_Map, COUNTY == "510")

str(BC_CensusTracts_Map, list.len = 13, strict.width = "cut")
```

```
## Classes 'sf' and 'data.frame':   200 obs. of  14 variables:
##  $ CensusTract    : chr  "24510010100" "24510010200" "24510010300" "245100104"..
##  $ GEO_ID         : chr  "1400000US24510010100" "1400000US24510010200" "14000"..
##  $ STATE          : chr  "24" "24" "24" "24" ...
##  $ COUNTY         : chr  "510" "510" "510" "510" ...
##  $ TRACT          : chr  "010100" "010200" "010300" "010400" ...
##  $ NAME           : chr  "101" "102" "103" "104" ...
##  $ LSAD           : chr  "Tract" "Tract" "Tract" "Tract" ...
##  $ CENSUSAREA     : num  0.152 0.137 0.26 0.144 0.06 0.067 0.076 0.256 0.165 0..
```

```
##  $ County         : chr  "Baltimore City" "Baltimore City" "Baltimore City" ""..
##  $ Urban          : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ POP2010        : num  3022 3009 2208 2870 1724 ...
##  $ LowIncomeTracts: num  0 0 0 0 0 0 0 0 1 1 ...
##  $ HUNVFlag       : num  0 0 0 0 0 0 0 0 1 0 ...
##   [list output truncated]
##  - attr(*, "sf_column")= chr "geometry"
##  - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA N..
##   ..- attr(*, "names")= chr [1:13] "CensusTract" "GEO_ID" "STATE" "COUNTY" ...
```

```
colSums(is.na(BC_CensusTracts_Map))
```

```
##     CensusTract          GEO_ID           STATE          COUNTY           TRACT
##               0               0               0               0               0
##            NAME            LSAD      CENSUSAREA          County           Urban
##               0               0               0               0               0
##         POP2010 LowIncomeTracts        HUNVFlag        geometry
##               0               0               0               0
```

Baltimore City has 200 entries, and there are no missing values.

# Chapter 5

## *Spatial Analysis — Part 1*

This section introduces spatial analysis and mapping with **R**. It covers creation of spatial data, manipulation of spatial data, and mapping.

### Creation and Manipulation of Spatial Data

Most manipulations are performed with functions from the **sf** package which is attached with the call **library(sf)**.

```
library(sf)
```

#### Conversion of non-spatial data into a (spatial) simple feature (sf) object

In Section 4.2 we created a smaller data frame of vaccination sites in Maryland (**VaccineSites_mod**). It is a non-spatial data frame that contains GPS coordinates in the variable **Location**. The format is **(latitude, longitude)**.

```
str(VaccineSites_mod, strict.width = "cut")
```

```
## 'data.frame':    556 obs. of  2 variables:
##  $ name    : chr  "Luminis Health Anne Arundel Medical Center" "Luminis Healt"..
##  $ Location: chr  "(38.990512076102, -76.5341664410021)" "(38.9825972406456, "..
```

```
## 'data.frame':    556 obs. of  2 variables:
##  $ name    : chr  "Luminis Health Anne Arundel Medical Center" "Luminis Health"..
##  $ Location: chr  "(38.990512076102, -76.5341664410021)" "(38.9825972406456, -"..")
```

Since the data frame has geometric information, it can be converted into a **sf** (simple feature) object with the function **st_as_sf()** from the **sf** package. **sf** objects contain non-spatial attributes

(variables/columns) and spatial features (geometries) associated with the attributes (Pebesma & Bivand, 2021, Chapter 7). However, first we have to convert the geometric information into a format that **sf** can read.

**st_as_sf()** takes coordinates (in separate variables) and converts them into a single geometry (list) column. We therefore modify the **Location** variable, and split the variable into a **lon** and a **lat** variable that contain the longitude and latitude coordinates (as numbers), respectively.

```
# Copy VaccineSites_mod into new object
VaccineSites_mod2 <- VaccineSites_mod

# Remove round brackets
VaccineSites_mod2$Location <- gsub("[(]|[)]", "", VaccineSites_mod2$Location)

# Split columns
VaccineSites_mod2 <- tidyr::separate(VaccineSites_mod2, Location, c("lat", "lon"),
                                     sep = ", ", remove = TRUE, convert = TRUE)

str(VaccineSites_mod2, strict.width = "cut")
```

```
## 'data.frame':    556 obs. of  3 variables:
##  $ name: chr  "Luminis Health Anne Arundel Medical Center" "Luminis Health Do"..
##  $ lat : num  39 39 39.3 39.6 39.2 ...
##  $ lon : num  -76.5 -76.9 -76.6 -77 -76.9 ...
```

```
## 'data.frame':    556 obs. of  3 variables:
##  $ name: chr  "Luminis Health Anne Arundel Medical Center" "Luminis Health Doctors ...
##  $ lon : num  -76.5 -76.9 -76.6 -77 -76.9 ...
```

The first line of the code copies the original data frame into a new object (**VaccineSites_mod2**). The next line removes the **()** from the entries in **Location** using a **regex** expression and the function **gsub()**. The **regex** identifies round brackets. **gsub()** substitutes the round brackets with no space (**""**). The 3rd line splits the variable into two variables (**lat** for latitude, and **lon** for longitude) with the function **separate()** from the package **tidyr**. The call **tidyr::separate()** allows to call the function **separate()** without attaching the package (**tidyr**). The original variable **Location** is removed (**remove = TRUE**), and the GPS coordinates are converted into real numbers (**convert = TRUE**).

Now we can use **st_as_sf()** to convert the coordinate variables into a single geometry column. GPS coordinates are based on the **WGS 84 CRS** (**EPSG:4326**), which is assigned to the geometries with the argument **crs = 4326**. Note, that coordinates in a **sf** geometry follow a **lon, lat** format.

```
VaccineSites_sf <- st_as_sf(VaccineSites_mod2,
                            coords = c("lon","lat"),
                            crs = 4326)

VaccineSites_sf
```

```
## Simple feature collection with 556 features and 1 field
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -79.40422 ymin: 37.99461 xmax: -75.11762 ymax: 39.7205
## Geodetic CRS:  WGS 84
## First 10 features:
##                                                  name
## 1            Luminis Health Anne Arundel Medical Center
## 2       Luminis Health Doctors Community Medical Center
## 3  Grace Medical Center (formerly Bon Secours Hospital)
## 4                              Carroll Hospital Center
## 5                        Howard County General Hospital
## 6                                 Holy Cross Hospital
## 7                       MedStar Good Samaritan Hospital
## 8                              MedStar Harbor Hospital
## 9                     Greater Baltimore Medical Center
## 10            MedStar Southern Maryland Hospital Center
##                       geometry
## 1  POINT (-76.53417 38.99051)
## 2   POINT (-76.86539 38.9826)
## 3  POINT (-76.64888 39.28823)
## 4  POINT (-76.99063 39.55787)
## 5  POINT (-76.88587 39.21429)
## 6  POINT (-77.03653 39.01438)
## 7  POINT (-76.58784 39.35864)
## 8   POINT (-76.6152 39.25186)
## 9  POINT (-76.62761 39.39109)
## 10 POINT (-76.87562 38.74815)
```

### Changing Coordinate References Systems (Re-projection)

As mentioned in Section 2.3, spatial analyses involving distances require a projected coordinate reference system (**CRS**) with linear distance units (meter, US survey feet, etc.). For Maryland we use the **NAD83(2011) Maryland CRS** (**EPSG:6487**). It uses a Lambert Conformal Conic projection (**lcc**) and meters.

The function **st_transform()** from the **sf** package re-projects coordinates of simple feature objects.

```
VaccineSites_6487 <- st_transform(VaccineSites_sf, crs = 6487)

VaccineSites_6487
```

```
## Simple feature collection with 556 features and 1 field
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 192974.5 ymin: 37051.97 xmax: 564558.6 ymax: 228199.9
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##                                                    name
## 1            Luminis Health Anne Arundel Medical Center
## 2       Luminis Health Doctors Community Medical Center
## 3  Grace Medical Center (formerly Bon Secours Hospital)
## 4                               Carroll Hospital Center
## 5                        Howard County General Hospital
## 6                                   Holy Cross Hospital
## 7                       MedStar Good Samaritan Hospital
## 8                               MedStar Harbor Hospital
## 9                     Greater Baltimore Medical Center
## 10           MedStar Southern Maryland Hospital Center
##                      geometry
## 1  POINT (440356.8 147055.9)
## 2    POINT (411663 146082.9)
## 3  POINT (430291.3 180062.2)
## 4      POINT (400805 209940)
## 5  POINT (409856.7 171801.7)
## 6  POINT (396836.2 149602.8)
## 7  POINT (435522.7 187900.9)
## 8  POINT (433214.1 176036.3)
## 9  POINT (432080.3 191488.2)
## 10 POINT (410811.8 120056.4)
```

In Section 4.1 — Maryland Counties (Physical) Boundaries we already read in a map showing the boundaries of Maryland's counties (**MD_counties_map**).

```
MD_counties_map
```

```
## Simple feature collection with 24 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -8848486 ymin: 4564403 xmax: -8354439 ymax: 4825752
## Projected CRS: WGS 84 / Pseudo-Mercator
```

```
## First 10 features:
##    OBJECTID         COUNTY DISTRICT COUNTY_FIP COUNTYNUM CREATION_D LAST_UPDAT
## 1         1       Allegany        6          1         1 2010-01-28 2010-01-28
## 2         2   Anne Arundel        5          3         2 2006-04-18 2006-04-18
## 3         3      Baltimore        4          5         3 2006-10-09 2006-10-09
## 4         4 Baltimore City        0        510        24 2006-04-18 2009-11-16
## 5         5        Calvert        5          9         4 2010-01-28 2010-01-28
## 6         6       Caroline        2         11         5 2007-05-21 2008-07-30
## 7         7        Carroll        7         13         6 2008-06-16 2012-01-17
## 8         8          Cecil        2         15         7 2006-04-18 2008-08-20
## 9         9        Charles        5         17         8 2009-06-08 2009-06-08
## 10       10     Dorchester        1         19         9 2007-02-08 2007-02-22
##                           geometry
## 1  MULTIPOLYGON (((-8721085 48...
## 2  MULTIPOLYGON (((-8527741 47...
## 3  MULTIPOLYGON (((-8523507 48...
## 4  MULTIPOLYGON (((-8519244 47...
## 5  MULTIPOLYGON (((-8531762 46...
## 6  MULTIPOLYGON (((-8432189 47...
## 7  MULTIPOLYGON (((-8556981 47...
## 8  MULTIPOLYGON (((-8441053 48...
## 9  MULTIPOLYGON (((-8580309 46...
## 10 MULTIPOLYGON (((-8439760 46...
```

The output tells us that the data set is using a projected **CRS** based on the **WGS 84 CRS** with a pseudo-mercator projection.

To map, all data objects need to have the same **CRS**. We therefore re-project the **MD_counties_map** to the **NAD83(2011) Maryland CRS** (**EPSG:6487**).

```
MD_counties_6487 <- st_transform(MD_counties_map, crs = 6487)

MD_counties_6487
```

```
## Simple feature collection with 24 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 185218.2 ymin: 25771.53 xmax: 570274.7 ymax: 230947
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##    OBJECTID         COUNTY DISTRICT COUNTY_FIP COUNTYNUM CREATION_D LAST_UPDAT
## 1         1       Allegany        6          1         1 2010-01-28 2010-01-28
## 2         2   Anne Arundel        5          3         2 2006-04-18 2006-04-18
## 3         3      Baltimore        4          5         3 2006-10-09 2006-10-09
## 4         4 Baltimore City        0        510        24 2006-04-18 2009-11-16
## 5         5        Calvert        5          9         4 2010-01-28 2010-01-28
## 6         6       Caroline        2         11         5 2007-05-21 2008-07-30
## 7         7        Carroll        7         13         6 2008-06-16 2012-01-17
## 8         8          Cecil        2         15         7 2006-04-18 2008-08-20
```

```
## 9         9        Charles        5         17         8 2009-06-08 2009-06-08
## 10       10     Dorchester        1         19         9 2007-02-08 2007-02-22
##                           geometry
## 1  MULTIPOLYGON (((284864.2 22...
## 2  MULTIPOLYGON (((434019 1735...
## 3  MULTIPOLYGON (((437063.4 22...
## 4  MULTIPOLYGON (((440528 1894...
## 5  MULTIPOLYGON (((431100.8 12...
## 6  MULTIPOLYGON (((508262.9 16...
## 7  MULTIPOLYGON (((411297.8 20...
## 8  MULTIPOLYGON (((500588 2260...
## 9  MULTIPOLYGON (((393194.3 11...
## 10 MULTIPOLYGON (((503022.8 11...
```

## 5.1 First Maps

Lets plot the vaccination sites onto a map of Maryland counties using **tmap**. **tmap** is a **R** package designed to create thematic maps to visualize spatial distributions. It uses a layer based approach. Every map starts with **tm_shape()**. **tm_shape()** defines the extension of the map and which method can be used to draw the features. It is followed by other layer elements, such as **tm_fill()** (which colors polygons), **tm_borders()** (which outlines polygons), or **tm_symbols** (which plots markers and symbols), to name a few (Lovelace *et al.*, 2021, Chapter 8). Each layer can be stored in or assigned to an object that can be called separately.

### Mapping Vaccination Sites

To start, we create a "base map" of Maryland's counties that is stored in the object **map_MD_counties**. **tm_shape()** is called to create a layer based on the shape object **MD_counties_6487**. **tm_polygon()** plots the polygons defined in **MD_counties_6487**. We could use the argument **"MAP_COLORS"** to color the counties as it attempts to color the polygons in such a way that neighboring polygons have different colors. However, it does not work too well for our data set. Furthermore, its color attribution is not consistent. Therefore, we will define our own color scheme that is based on the color palette **GnBu** from the **RColorBrewer** package.

```
library(tmap)
library(RColorBrewer)

#Define colors
(cols <- brewer.pal(brewer.pal.info["GnBu", "maxcolors"], "GnBu"))

mypal <- c(cols[1], cols[4], cols[3], cols[5], cols[3],
           cols[1], cols[4], cols[4], cols[5], cols[4],
           cols[2], cols[2], cols[1], cols[5], cols[5],
           cols[3], cols[1], cols[3], cols[1], cols[4],
```

```
            cols[5], cols[4], cols[3], cols[4])

# map counties
map_MD_counties <- tm_shape(MD_counties_6487) +
          tm_polygons(col = "COUNTY",
          palette = mypal,
          legend.show = FALSE)

map_MD_counties
```

**map_MD_counties** plots the map and you should see a plot similar to Figure 5.1 A.

Create a directory **figures** in your project directory. Save the map of the county with **tmap_save()** as a **.png** file (**map_MD_counties.png**) in the directory **figures** (within the current working directory):

```
# save to png
tmap_save(map_MD_counties, filename = "figures/map_MD_counties.png")
```

Then we create a layer that has the labels of the counties. The layer is assigned to the object **MD_counties_label**. Again, **tm_shape(MD_counties_6487)** is called. **tm_text** plots the name of the counties (listed in the variable **COUNTY** of the **MD_counties_6487** data set). We set the text color to **brown** (argument **col**), and the size to **0.75**. We plot the labels by calling the object **MD_counties_label**. A plot similar to Figure 5.1 B should be produced.

```
#label
MD_counties_label <- tm_shape(MD_counties_6487) +
                     tm_text("COUNTY",
                             col = "brown",
                             size = 0.75)

MD_counties_label
```

Last, we create a layer with the vaccination sites and store the layer in the object **map_VaccineSites**. **tm_shape()** is called again to create the layer based on the **VaccineSites_6487** object. The geometric features of this layer are points. The points are plotted with the function **tm_symbols()**. The **shape** argument plots symbols available to **R** (**shape 0** to **25**). The fill color, outline (border) color, and line width of the outline of shapes **21** to **25** can be changed with the arguments **col**, **border.col**, and **border.lwd**. respectively (Figure 5.1 D). For the vaccine sites (Figures 5.1 C, 5.2, and 5.3) the fill color is set to **green**, the outline color to **red**, and the line width to **1**. Calling **map_VaccineSites** should plot a plot similar to Figure 5.1 C.

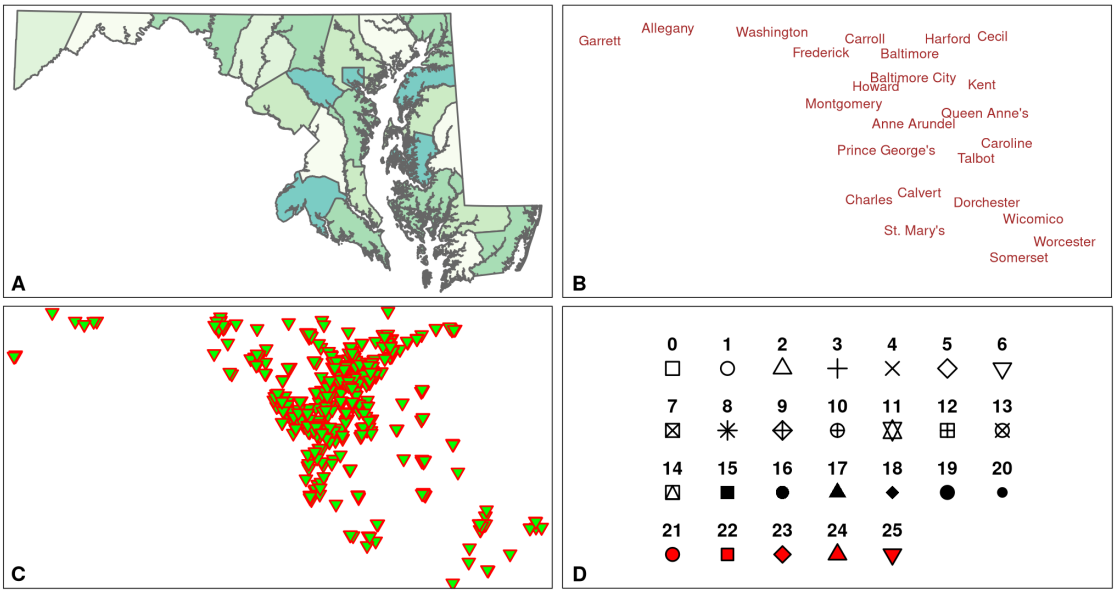



***FIGURE 5.1**. **A.** Map of Maryland counties. **B.** Plot showing the names of the counties. **C.** Plot of the vaccination sites in Maryland (on April 4, 2021). **D.** Plotting symbols in* **R**.

We can put together the final map by calling the layers, and add a legend with **`tmap_add_label()`**.

```
MD_vac1 <- map_MD_counties +
           MD_counties_label +
           map_VaccineSites +
           tm_add_legend(type = "symbol",
                         shape = 25,
                         size = 0.75,
                         col = "green",
                         border.col = "red",
                         label = "COVID-19 Vaccination Site") +
            tm_layout(legend.text.size = 1)

MD_vac1
```

**`tm_add_legends()`** adds (manually) a legend to the plot. We just want to indicate what the plotted symbols stand for (i.e, COVID-19 Vaccination Sites). Therefore, the **`type`** is set to **`"symbol."`** We define the shape, fill and outline color (copy the respective parameters (arguments) from **`tm_symbols`**), and the size of the symbol for the legend. **`label`** adds the legend text (**`"COVID-19 Vaccination Site"`**). The last function, **`tm_layout()`** allows to fine tune the plot. Here we only specify the size of the legend label. The last line (**`MD_vac1`**) plots the map. The plot should look similar to the plot in Figure 5.2.

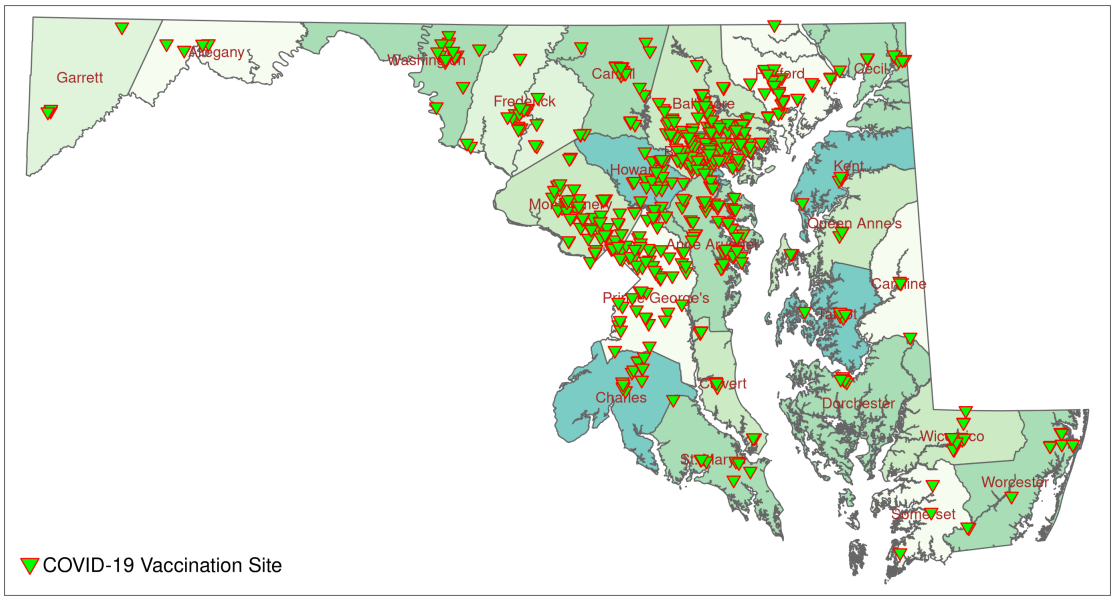



***FIGURE 5.2**. Location of COVID-19 vaccination sites in Maryland (last updated: April 4, 2021)*

Save the plot as a **`.png`** file (**`MD_vac_sites.png`**) in the directory **`figures`** (within the current working directory).

```
# save to png
tmap_save(MD_vac1, filename = "figures/MD_vac_sites.png")
```

The map shows that there are clusters of vaccination sites in the I-95 and I-270 corridors. Rural areas seem to be less covered. To check we can add urban census tracts to the plot. The data frame **`MD_CensusTracts_Map`** has all the information we need. However, its **`CRS`** is **`NAD83`**.

```
MD_CensusTracts_Map
```

```
## Simple feature collection with 1403 features and 13 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -79.48765 ymin: 37.91172 xmax: -75.04894 ymax: 39.72304
## Geodetic CRS:  NAD83
## First 10 features:
##    CensusTract               GEO_ID STATE COUNTY  TRACT NAME  LSAD CENSUSAREA
## 1  24001000100 1400000US24001000100    24    001 000100    1 Tract    187.937
## 2  24001000200 1400000US24001000200    24    001 000200    2 Tract     48.067
## 3  24001000300 1400000US24001000300    24    001 000300    3 Tract      8.656
## 4  24001000400 1400000US24001000400    24    001 000400    4 Tract      3.723
```

```
## 5  24001000500 1400000US24001000500    24    001 000500    5 Tract      4.422
## 6  24001000600 1400000US24001000600    24    001 000600    6 Tract      1.577
## 7  24001000700 1400000US24001000700    24    001 000700    7 Tract      0.712
## 8  24001000800 1400000US24001000800    24    001 000800    8 Tract      1.255
## 9  24001001000 1400000US24001001000    24    001 001000   10 Tract      0.448
## 10 24001001100 1400000US24001001100    24    001 001100   11 Tract      0.301
##      County Urban POP2010 LowIncomeTracts HUNVFlag
## 1  Allegany     0    3718               1        0
## 2  Allegany     0    4564               1        0
## 3  Allegany     0    2780               1        0
## 4  Allegany     1    3022               1        0
## 5  Allegany     1    2734               1        0
## 6  Allegany     1    2965               1        0
## 7  Allegany     1    3387               1        1
## 8  Allegany     1    2213               1        1
## 9  Allegany     1    2547               1        0
## 10 Allegany     1    1493               1        0
##                          geometry
## 1  MULTIPOLYGON (((-78.42058 3...
## 2  MULTIPOLYGON (((-78.71775 3...
## 3  MULTIPOLYGON (((-78.72077 3...
## 4  MULTIPOLYGON (((-78.70206 3...
## 5  MULTIPOLYGON (((-78.75435 3...
## 6  MULTIPOLYGON (((-78.74107 3...
## 7  MULTIPOLYGON (((-78.75033 3...
## 8  MULTIPOLYGON (((-78.76588 3...
## 9  MULTIPOLYGON (((-78.77711 3...
## 10 MULTIPOLYGON (((-78.76443 3...
```

We re-project **MD_CensusTracts_Map** to **NAD83(2011) Maryland** (**EPSG:6487**).

```
MD_CensusTracts_6487 <- st_transform(MD_CensusTracts_Map, crs = 6487)

MD_CensusTracts_6487
```

```
## Simple feature collection with 1403 features and 13 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 185230.9 ymin: 27801.06 xmax: 570294.2 ymax: 230941.9
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##    CensusTract               GEO_ID STATE COUNTY  TRACT NAME  LSAD CENSUSAREA
## 1  24001000100 1400000US24001000100    24    001 000100    1 Tract    187.937
## 2  24001000200 1400000US24001000200    24    001 000200    2 Tract     48.067
## 3  24001000300 1400000US24001000300    24    001 000300    3 Tract      8.656
## 4  24001000400 1400000US24001000400    24    001 000400    4 Tract      3.723
## 5  24001000500 1400000US24001000500    24    001 000500    5 Tract      4.422
## 6  24001000600 1400000US24001000600    24    001 000600    6 Tract      1.577
## 7  24001000700 1400000US24001000700    24    001 000700    7 Tract      0.712
```

```
## 8  24001000800 1400000US24001000800    24    001 000800    8 Tract      1.255
## 9  24001001000 1400000US24001001000    24    001 001000   10 Tract      0.448
## 10 24001001100 1400000US24001001100    24    001 001100   11 Tract      0.301
##      County Urban POP2010 LowIncomeTracts HUNVFlag
## 1  Allegany     0    3718               1        0
## 2  Allegany     0    4564               1        0
## 3  Allegany     0    2780               1        0
## 4  Allegany     1    3022               1        0
## 5  Allegany     1    2734               1        0
## 6  Allegany     1    2965               1        0
## 7  Allegany     1    3387               1        1
## 8  Allegany     1    2213               1        1
## 9  Allegany     1    2547               1        0
## 10 Allegany     1    1493               1        0
##                          geometry
## 1  MULTIPOLYGON (((277992.7 21...
## 2  MULTIPOLYGON (((252677.1 22...
## 3  MULTIPOLYGON (((252464.7 22...
## 4  MULTIPOLYGON (((253978.3 22...
## 5  MULTIPOLYGON (((249423 2212...
## 6  MULTIPOLYGON (((250523.8 21...
## 7  MULTIPOLYGON (((249749.1 22...
## 8  MULTIPOLYGON (((248425.6 22...
## 9  MULTIPOLYGON (((247506.5 22...
## 10 MULTIPOLYGON (((248561.4 22...
```

For the map we are only interested in urban census tracts. We extract the urban tracts from **`MD_CensusTracts_6487`** (**`Urban == 1`**), and re-draw the map. Urban tracts are highlighted in orange.

```
# Filter out urban centers
urban_tracts <- subset(MD_CensusTracts_6487, Urban == 1)

urban_tracts

# Map urban tracts
# Layer for urban tracts
map_urban_tracts <- tm_shape(urban_tracts) +
                    tm_fill(col = "orange")

# save as png
tmap_save(map_urban_tracts, filename = "figures/map_urban_tracts.png")
```

Compile the map.

```
MD_vac2 <- map_MD_counties +
           map_urban_tracts +
           MD_counties_label +
           map_VaccineSites +
```

```
            tm_add_legend(type = "symbol",
                          shape = c(22, 25),
                          size = c(1, 0.75),
                          col = c("orange", "green"),
                          border.col = c("black", "red"),
                          label = c("Urban Census Tract",
                                    "COVID-19 Vaccination Site")) +
            tm_layout(legend.text.size = 1)

MD_vac2

# save to png
tmap_save(MD_vac2, filename = "figures/MD_vac2_sites_census_tract.png")
```

We added an orange square (symbol **22**) in **`tm_add_legend()`**

```
tm_add_legend(type = "symbol",
                      shape = c(22, 25),
                      size = c(1, 0.75),
                      col = c("orange", "green"),
                      border.col = c("black", "red"),
                      label = c("Urban Census Tract",
                                "COVID-19 Vaccination Site"))
```

Calling **`Map_vac2`** should produce a plot similar to Figure 5.3.

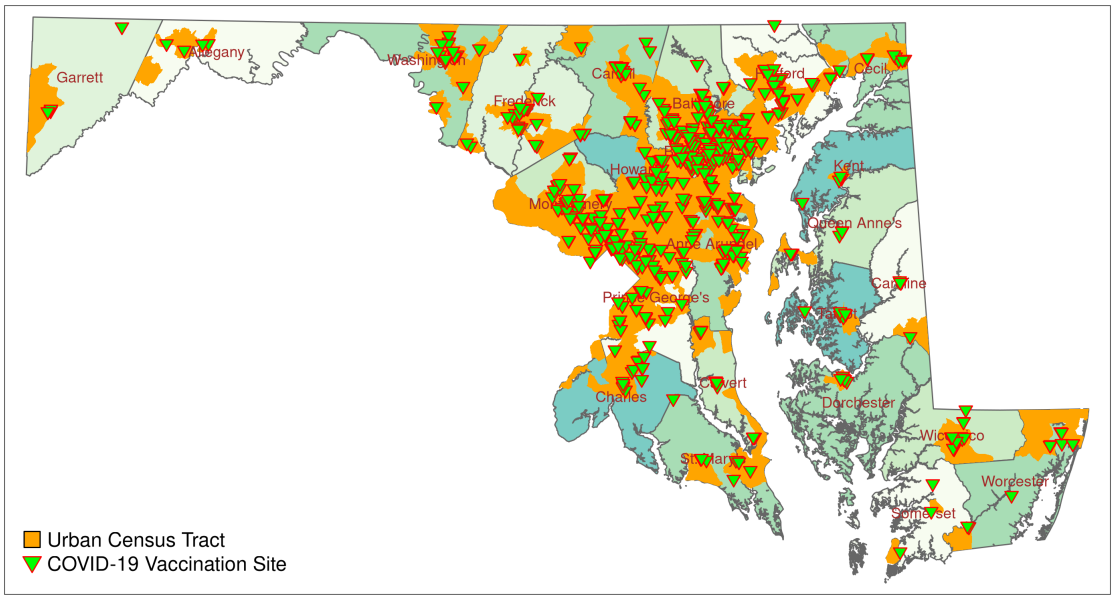



***FIGURE 5.3**. Location of COVID-19 vaccination sites in Maryland (last updated: April 4, 2021). Urban census tracts are highlighted in orange.*

## 5.2 Manipulating Geometries

The **sf** package has functions that allow us to manipulate the spatial features of a **sf** object, including

- identify features based on their spatial relation to other features,
- clip geometries, and
- merge geometries.

### Identify spatial features based on spatial relations to other features

To identify or visualize vaccination sites in rural and urban centers, we plotted urban tracts onto the vaccination site map. Alternatively, we can extract vaccination sites that are located in rural tracts and urban centers with the function **st_intersection()**.

We already created a **sf** object that features urban census tracts (**urban_tracts**). Lets also create a **sf** object for rural census tracts.

```
rural_tracts <- subset(MD_CensusTracts_6487, Urban == 0)
```

The following lines of code identify the vaccination sites that are located within or on the border of urban and rural census tracts, respectively. You will likely see warnings. They can be ignored.

```
vac_urban <- st_intersection(urban_tracts, VaccineSites_6487)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
vac_urban
```

```
## Simple feature collection with 494 features and 14 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 192974.5 ymin: 46073.83 xmax: 560461.9 ymax: 224285.4
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##      CensusTract              GEO_ID STATE COUNTY  TRACT    NAME  LSAD
## 40   24003702701 1400000US24003702701    24    003 702701 7027.01 Tract
## 1070 24033806711 1400000US24033806711    24    033 806711 8067.11 Tract
## 1301 24510200100 1400000US24510200100    24    510 200100    2001 Tract
```

```
## 394  24013507801 1400000US24013507801    24    013 507801 5078.01 Tract
## 629  24027605602 1400000US24027605602    24    027 605602 6056.02 Tract
## 829  24031703901 1400000US24031703901    24    031 703901 7039.01 Tract
## 1369 24510270803 1400000US24510270803    24    510 270803 2708.03 Tract
## 1323 24510250203 1400000US24510250203    24    510 250203 2502.03 Tract
## 314  24005490605 1400000US24005490605    24    005 490605 4906.05 Tract
## 926  24033801207 1400000US24033801207    24    033 801207 8012.07 Tract
##      CENSUSAREA          County Urban POP2010 LowIncomeTracts HUNVFlag
## 40        1.812    Anne Arundel     1    4815               0        0
## 1070      0.548 Prince George's     1    5146               1        0
## 1301      0.102  Baltimore City     1    1846               1        1
## 394       2.423         Carroll     1    5652               0        1
## 629       2.809          Howard     1    7610               0        0
## 829       0.554      Montgomery     1    2957               0        0
## 1369      0.845  Baltimore City     1    6268               1        0
## 1323      0.357  Baltimore City     1    2018               1        1
## 314       0.844       Baltimore     1    5182               1        0
## 926       3.141 Prince George's     1    4432               0        0
##                                                    name
## 40             Luminis Health Anne Arundel Medical Center
## 1070      Luminis Health Doctors Community Medical Center
## 1301 Grace Medical Center (formerly Bon Secours Hospital)
## 394                              Carroll Hospital Center
## 629                        Howard County General Hospital
## 829                                Holy Cross Hospital
## 1369                      MedStar Good Samaritan Hospital
## 1323                             MedStar Harbor Hospital
## 314                     Greater Baltimore Medical Center
## 926              MedStar Southern Maryland Hospital Center
##                       geometry
## 40   POINT (440356.8 147055.9)
## 1070   POINT (411663 146082.9)
## 1301 POINT (430291.3 180062.2)
## 394      POINT (400805 209940)
## 629  POINT (409856.7 171801.7)
## 829  POINT (396836.2 149602.8)
## 1369 POINT (435522.7 187900.9)
## 1323 POINT (433214.1 176036.3)
## 314  POINT (432080.3 191488.2)
## 926  POINT (410811.8 120056.4)
```

```
vac_rural <- st_intersection(rural_tracts, VaccineSites_6487)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
vac_rural
```

```
## Simple feature collection with 62 features and 14 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 219869 ymin: 37051.97 xmax: 564558.6 ymax: 228199.9
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##      CensusTract               GEO_ID STATE COUNTY  TRACT    NAME  LSAD
## 353  24009860702 1400000US24009860702    24    009 860702 8607.02 Tract
## 1091 24035810400 1400000US24035810400    24    035 810400    8104 Tract
## 1106 24037875500 1400000US24037875500    24    037 875500    8755 Tract
## 1120 24039930300 1400000US24039930300    24    039 930300    9303 Tract
## 1198 24047951300 1400000US24047951300    24    047 951300    9513 Tract
## 366  24011955301 1400000US24011955301    24    011 955301 9553.01 Tract
## 465  24019970702 1400000US24019970702    24    019 970702 9707.02 Tract
## 1129 24041960501 1400000US24041960501    24    041 960501 9605.01 Tract
## 1185 24045010702 1400000US24045010702    24    045 010702  107.02 Tract
## 1194 24047950900 1400000US24047950900    24    047 950900    9509 Tract
##      CENSUSAREA       County Urban POP2010 LowIncomeTracts HUNVFlag
## 353       9.430      Calvert     0    2974               1        0
## 1091     47.327 Queen Anne's     0    6053               0        0
## 1106     34.068   St. Mary's     0    8631               0        1
## 1120     88.028     Somerset     0    2539               1        0
## 1198      5.936    Worcester     0    2816               1        0
## 366      42.331     Caroline     0    4111               0        0
## 465      43.137   Dorchester     0    3987               0        0
## 1129     22.790       Talbot     0    4752               0        0
## 1185     27.469     Wicomico     0    8671               1        1
## 1194     26.689    Worcester     0    2017               0        0
##                                      name                   geometry
## 353       Calvert County Health Department    POINT (435102 99044.7)
## 1091 Queen Anne's County Health Department POINT (481028.7 153466.8)
## 1106 Saint Mary's County Health Department POINT (431747.9 70311.09)
## 1120     Somerset County Health Department POINT (513236.4 51719.08)
## 1198    Worcester County Health Department POINT (542330.9 57518.24)
## 366                        Walmart Denton   POINT (502252.9 134723)
## 465                     Walmart Cambridge POINT (482594.3 98828.14)
## 1129                       Walmart Easton POINT (482184.2 123210.8)
## 1185                    Walmart Salisbury POINT (524947.9 84315.16)
## 1194                       Walmart Berlin POINT (560573.2 76351.68)
```

Sixty-two vaccination sites are in rural tracts, and 494 in urban tracts. Let's see how many vaccination sites per 100,000 (or in scientific notation **`1e05`**) residents are in rural and urban tracts (based on population data from the 2010 census). The variable **`POP2010`** lists the total number of residents in each tract. The number of vaccination sites for each subset equals the number of entries and can be displayed with **`nrow()`**. **`nrow()`** lists the number of rows of a data frame.

```
rural_pop_100k <- nrow(vac_rural)/sum(vac_rural$POP2010)*1e05

rural_pop_100k
```

```
## [1] 21.07353
```

```
urban_pop_100k <- nrow(vac_urban)/sum(vac_urban$POP2010)*1e05

urban_pop_100k
```

```
## [1] 21.07604
```

The result suggests that based on the plain number of vaccination sites, rural and urban tracts are overall equally covered. Rural tracts have 21.07 vaccination sites per 100,000 residents, and urban tracts 21.08.

To visualize, we map the urban and rural vaccination sites (green inverted triangles (symbol 25): urban, blue diamonds (symbol 23): rural sites; for symbols, see Figure 5.1 D). We compile the map as we have done before. First, we create map layers for the urban and rural vaccination sites. Then we add these two layers to the county map featuring urban tracts.

```
map_vac_urban <- tm_shape(vac_urban) +
                 tm_symbols(shape = 25,
                            size =0.3,
                            col = "green",
                            border.col = "red",
                            border.lwd = 1)

map_vac_rural <- tm_shape(vac_rural) +
                 tm_symbols(shape = 23,
                            size =0.3,
                            col = "blue",
                            border.col = "red",
                            border.lwd = 1)

MD_vac3 <- map_MD_counties +
           map_urban_tracts +
           MD_counties_label +
           map_vac_urban +
           map_vac_rural +
           tm_add_legend(type = "symbol",
                         shape = c(22, 25, 23),
```

```
                          size = c(1, 0.75, 0.75),
                          col = c("orange", "green", "blue"),
                          border.col = c("black", "red", "red"),
                          label = c("Urban Census Tract",
                                    "Urban COVID-19 Vaccination Site",
                                    "Rural COVID-19 Vaccination Site")) +
            tm_layout(legend.text.size = 1)

#call map
MD_vac3

# save to png
tmap_save(MD_vac3, filename = "figures/MD_vac3_urban_rural.png")
```

The code should print a map similar to Figure 5.4.

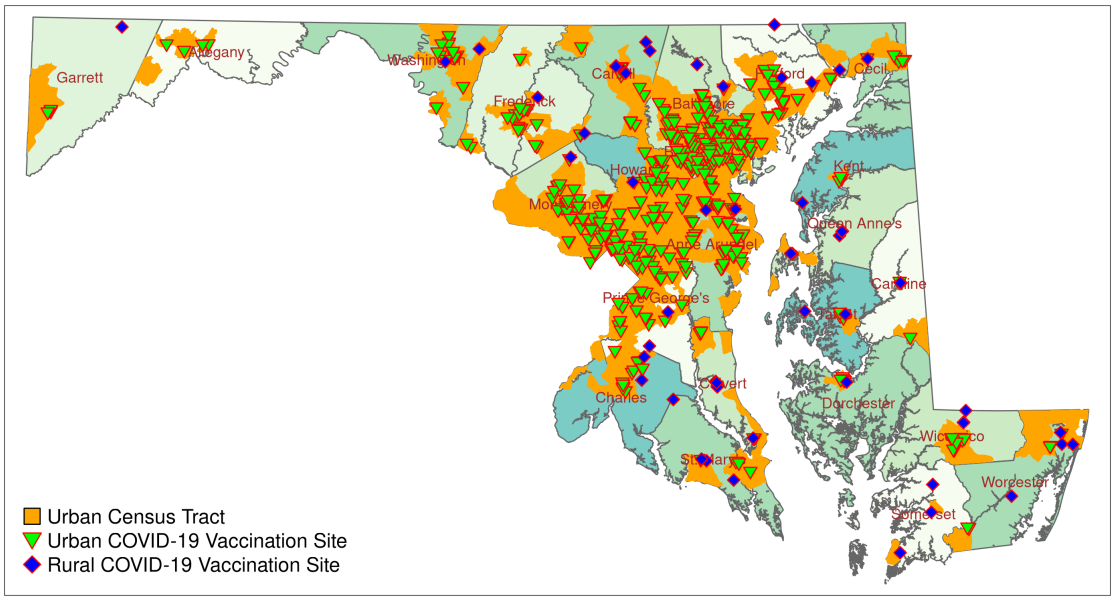



***FIGURE 5.4**. Location of COVID-19 vaccination sites in urban and rural census tracts in Maryland (last updated: April 4, 2021). Urban tracts are highlighted in orange.*

# Chapter 6

## *Spatial Analysis — Part 2*

In the previous section we have created maps that show the locations of vaccination sites in rural and urban census tracts.

That required us to:

- filter spatial objects,
- identify and extract spatial feature based on their spatial relation to other spatial features, and
- map (visualize) these features.

In this section we expand our analysis and assess rural communities in a selected county in Maryland and Baltimore City for the potential presence of vaccination deserts. Per our definition, a low-income census tract qualifies as a vaccination desert if 33% of its area is outside a 0.5 mile (urban) or 10 mile (rural) range of the vaccination site (Section 1.1).

Conceptually we need to identify:

1. low-income census tracts that are outside of a certain range of a vaccination site, and
2. low-income tracts that have less than 33% of their area within the range of a vaccination site.

To identify these tracts, we need to further manipulate spatial objects, including:

- merge spatial features,
- buffer,
- clip, and
- perform simple mathematical operations on spatial features.

Before we continue, let us create a Maryland state boundary map and a map containing county boundaries. For the latter, read in the shapefile **`MD_counties_CT.shp`**. This map contains boundaries of the counties of Maryland. I created the file by extracting all counties from the census tract map (**`MD_CensusTracts_6487`**), and unifying the census tracts of each county. The Maryland state boundary map is created by unifying the census tracts of the entire state. The following script produces these maps that should look similar to the maps in Figure 6.1.

```
MD_counties <- st_read("data/MD_counties_CT.shp")

# draw Map (Fig. 6.1 A)
map_MD_counties <- tm_shape(MD_counties) +
                   tm_borders()

map_MD_counties

# create MD_state (Fig. 6.1 B)
MD_state <- st_union(MD_counties)

# draw Map
map_MD_state <- tm_shape(MD_state) +
                   tm_fill(col = "antiquewhite") +
                   tm_borders()

map_MD_state
```

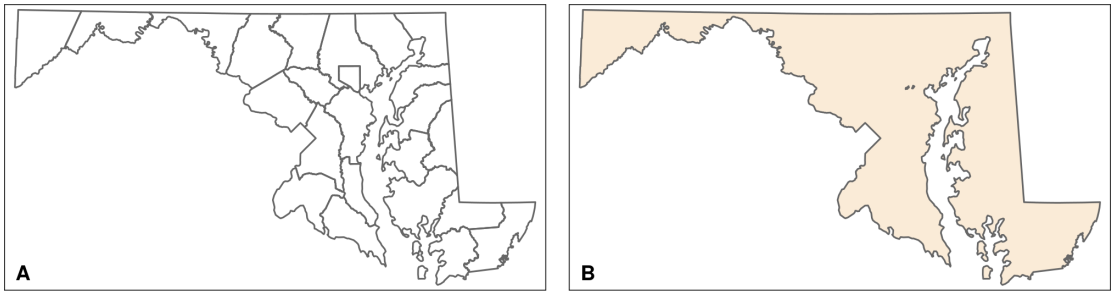



***FIGURE 6.1**. Map of political boundaries of Maryland, based on census tract boundaries. **A.** County boundaries. **B.** State boundaries.*

## 6.1 Identification of Possible Rural Vaccination Deserts

To identify areas with limited access to vaccination sites, we first create a 10-mile buffer around all Maryland vaccination sites (listed in **`VaccineSites_6487`**) with the function **`st_buffer()`**. The **`CRS`** of our maps uses meter. Thus, we need to convert miles to meter. One mile is approximately 1.690344 meters. Ten miles are therefore 16,093 meters, which is provided to the **`dist`** argument of **`st_buffer()`**. We join (unify) overlapping buffers with **`st_union()`**, and crop (clip) the ten mile ranges with **`st_intersection()`** to the state boundaries of Maryland.

```
# create a 10 mi buffer
vac_10mi <- st_buffer(VaccineSites_6487, dist = 1.609344*1e04)

# join/unify overlapping buffers
vac_10mi_union <- st_union(vac_10mi)

# clip/crop
vac_10mi_state <- st_intersection(MD_state, vac_10mi_union)
```

Next we plot Maryland's vaccination sites with the ten-mile buffer onto the counties "base map" (**map_MD_counties**). The **alpha** argument of the **tm_fill()** function sets a transparency level. One would mean no transparency (100% opacity), and zero 100% transparency (0% opacity). We set it to 65% (**alpha = 0.35**). The code produces a map similar to Figure 6.2.

```
# draw map
map_vac_10mi_md <- map_MD_counties +
                   tm_shape(vac_10mi_state) +
                   tm_fill(col = "orange",
                               alpha = 0.35) +
                   tm_borders(col = "red") +
                   map_VaccineSites

map_vac_10mi_md
```

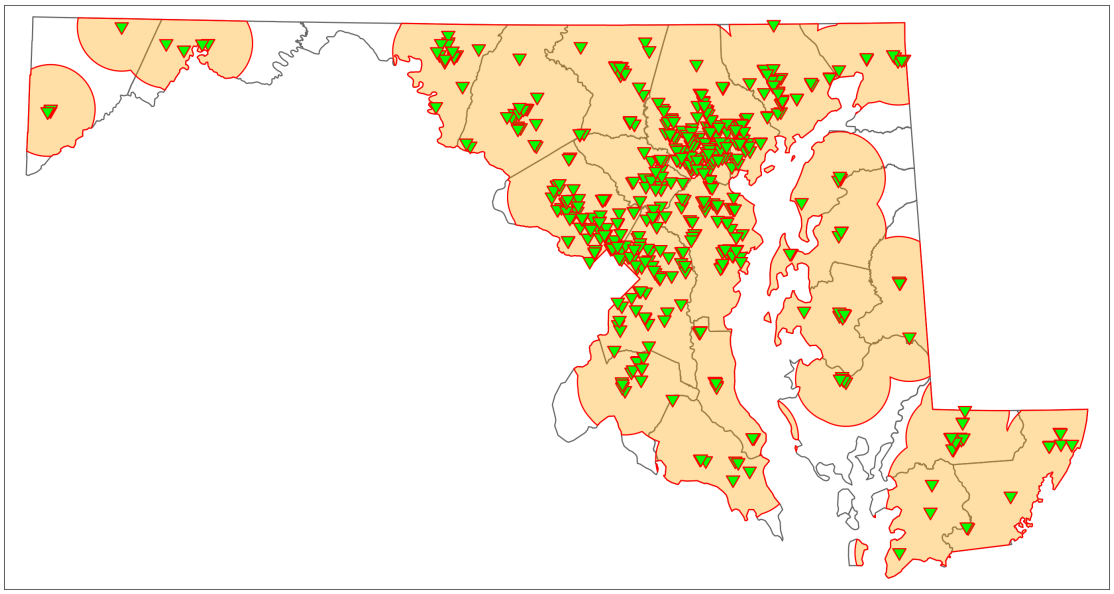


***FIGURE 6.2**. Vaccination sites with a 10 mi buffer, clipped to the Maryland state border.*

Now we identify low-income rural census tracts. We subset **MD_CensusTracts_6487** for **Urban == 0** and **LowIncomeTracts == 1**.

```
rural_LowIncome <- subset(MD_CensusTracts_6487, Urban == 0 & LowIncomeTracts == 1)

rural_LowIncome
```

```
## Simple feature collection with 55 features and 13 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 185230.9 ymin: 27801.06 xmax: 547910.5 ymax: 230941.9
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##     CensusTract               GEO_ID STATE COUNTY  TRACT    NAME  LSAD
## 1   24001000100 1400000US24001000100    24    001 000100       1 Tract
## 2   24001000200 1400000US24001000200    24    001 000200       2 Tract
## 3   24001000300 1400000US24001000300    24    001 000300       3 Tract
## 15  24001001502 1400000US24001001502    24    001 001502   15.02 Tract
## 16  24001001503 1400000US24001001503    24    001 001503   15.03 Tract
## 20  24001001900 1400000US24001001900    24    001 001900      19 Tract
## 21  24001002000 1400000US24001002000    24    001 002000      20 Tract
## 23  24001002200 1400000US24001002200    24    001 002200      22 Tract
## 299 24005451900 1400000US24005451900    24    005 451900    4519 Tract
## 353 24009860702 1400000US24009860702    24    009 860702 8607.02 Tract
##     CENSUSAREA    County Urban POP2010 LowIncomeTracts HUNVFlag
## 1      187.937  Allegany     0    3718               1        0
## 2       48.067  Allegany     0    4564               1        0
## 3        8.656  Allegany     0    2780               1        0
## 15       9.148  Allegany     0    2055               1        0
## 16      11.539  Allegany     0    1968               1        0
## 20      24.855  Allegany     0    2623               1        0
## 21      26.800  Allegany     0    5552               1        1
## 23      23.497  Allegany     0    3874               1        0
## 299      5.187 Baltimore     0    2445               1        0
## 353      9.430   Calvert     0    2974               1        0
##                           geometry
## 1   MULTIPOLYGON (((277992.7 21...
## 2   MULTIPOLYGON (((252677.1 22...
## 3   MULTIPOLYGON (((252464.7 22...
## 15  MULTIPOLYGON (((243626.5 22...
## 16  MULTIPOLYGON (((241785.2 22...
## 20  MULTIPOLYGON (((234463.9 22...
## 21  MULTIPOLYGON (((241888.4 21...
## 23  MULTIPOLYGON (((231767.6 19...
## 299 MULTIPOLYGON (((456497.8 17...
## 353 MULTIPOLYGON (((434677 1017...
```

There are 55 census tracts that qualify. The code below plots these tracts onto the map (Figure 6.3).

```
map_Rural_LowIncome_md <- map_MD_counties +
                          tm_shape(rural_LowIncome) +
                          tm_fill(col = "red",
                                  alpha = 0.75) +
                          tm_shape(vac_10mi_state) +
                          tm_fill(col = "orange",
                                  alpha = 0.5) +
                          tm_borders(col = "blue") +
                          map_VaccineSites

map_Rural_LowIncome_md
```

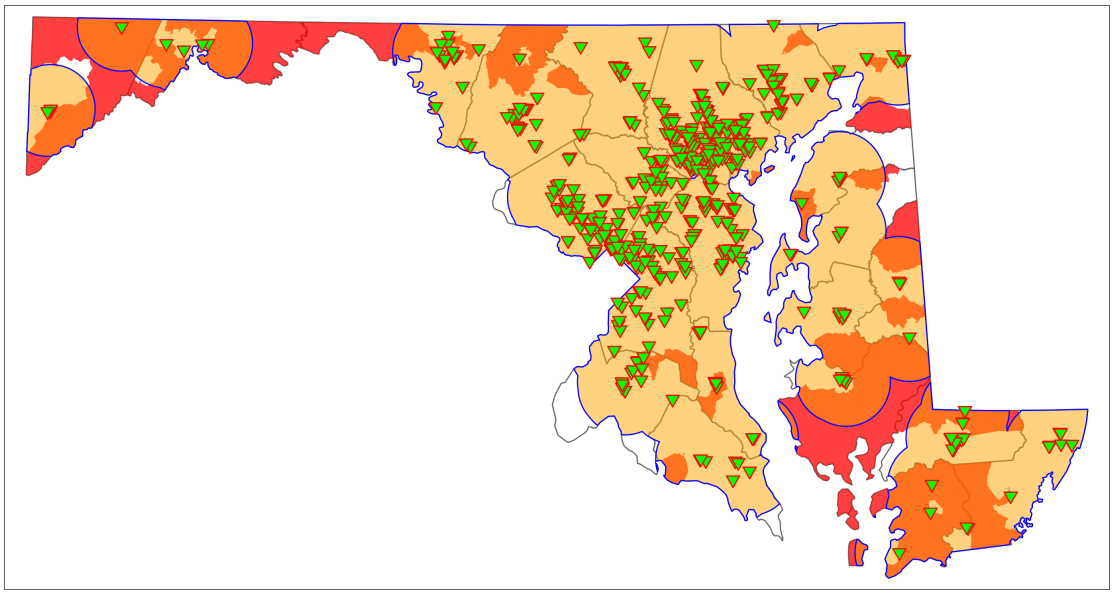


***FIGURE 6.3**. Rural low-income tracts and vaccination sites with a 10 mi buffer, clipped to the Maryland state border.*

We can narrow down the rural regions that may contain vaccination deserts by identifying rural low-income areas that are outside the 10-mile range of a vaccination site. The function **`st_difference()`** clips features that do not intersect with other features or are within other features. (Note that the code may produce a warning that can be ignored). The resulting plot should be similar to Figure 6.4. It shows rural areas (not census tracts) that would have limited access to vaccination sites in "hot pink." They are mainly located on the Eastern Shore (parts of Dorchester, Queen Anne's, and Kent County) and in Western Maryland (parts of Washington, Allegany, and Garrett County).

```
rural_vac_desert <- st_difference(rural_LowIncome, vac_10mi_state)

map_Rural_vac_desert <- map_MD_counties +
                        tm_shape(rural_vac_desert) +
                        tm_fill(col = "hotpink",
                                alpha = 0.75) +
                        tm_shape(vac_10mi_state) +
                        tm_fill(col = "orange",
                                alpha = 0.5) +
                        tm_borders(col = "blue") +
                        map_VaccineSites

map_Rural_vac_desert
```

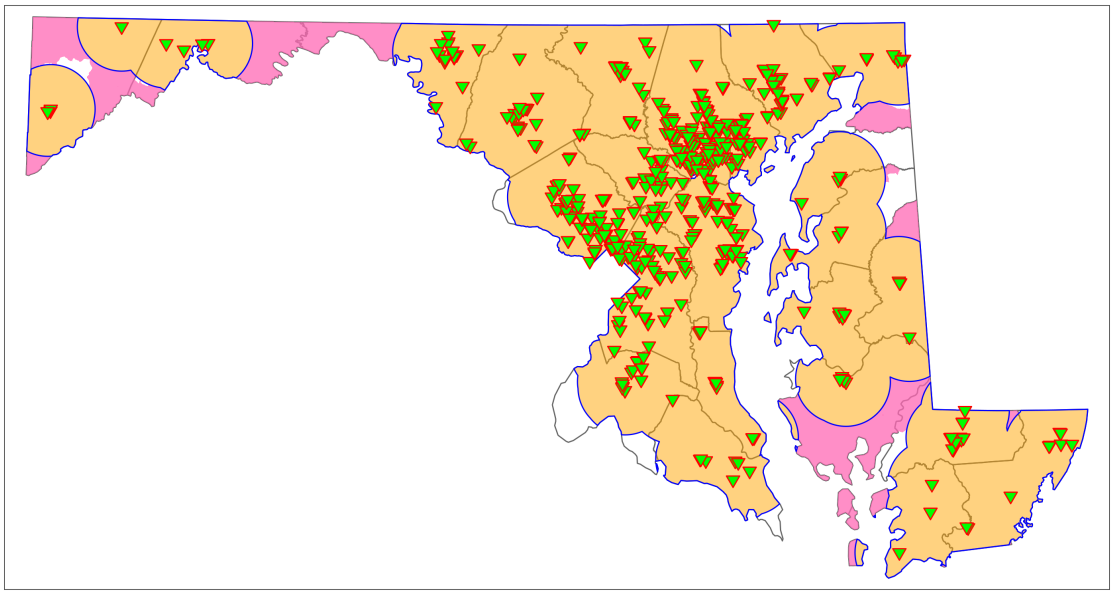


***FIGURE 6.4**. The map shows rural areas with potential vaccination deserts (highlighted in "hot pink").*

## 6.2 Possible Vaccination Deserts in Garrett County

The analysis above suggests that Garrett County in Western Maryland may have rural vaccination deserts, which warrants further analysis. Before we continue, we create a basemap for Garrett County that allows us to map possible vaccination deserts in Garrett County. The code below extracts Garrett County from **`MD_CensusTracts_6487`** using the **`County`** variable. It unifies the census tracts and plots a map. The plot should look similar to Figure 6.5 A.

```
Garrett_CensusTracts <- subset(MD_CensusTracts_6487, County == "Garrett")

Garrett_CensusTracts

Garrett_County <- st_union(Garrett_CensusTracts)

map_Garrett <- tm_shape(Garrett_County) +
               tm_fill(col = "antiquewhite") +
               tm_borders(col = "black")

map_Garrett
```

Next, we extract and map rural low-income tracts, that are highlighted in red. For reference, all census tracts are outlined (Figure 6.5 B).

```
Garrett_RuralLowIncome <- subset(Garrett_CensusTracts,
                                 Urban == 0 & LowIncomeTracts == 1)
Garrett_RuralLowIncome

map_Garrett_RuralLowIncome <- map_Garrett +
                              tm_shape(Garrett_CensusTracts) +
                              tm_borders(col = "black") +
                              tm_shape(Garrett_RuralLowIncome) +
                              tm_fill(col = "orange") +
                              tm_borders(col = "black")

map_Garrett_RuralLowIncome
```

Five of the seven census tracts in Garrett County qualify as rural low-income tracts. Next, we identify vaccination sites located in Garrett County. To identify areas with limited access to these sites we create again a 10-mile buffer around the Garrett County vaccination sites, join overlapping buffers, and clip the 10-mile buffers to the boundaries of Garrett County. For reference we label the tracts with their names (listed in the variable **NAME**) (Figure 6.5 C).

```
# Identify vaccination sites
vac_Garrett <- st_intersection(VaccineSites_6487, Garrett_County)

# Create a 10 mile buffer around vaccination sites
vac_Garrett_10mi <- st_buffer(vac_Garrett, dist = 1.690344*1e04)

# unify
vac_Garrett_10mi <- st_union(vac_Garrett_10mi)

# Clip to Garrett County Boundaries
vac_Garrett_10mi_clipped <- st_intersection(Garrett_County, vac_Garrett_10mi)
```

```
# map
# Garrett County Vaccine Sites
map_vac_Garrett <- tm_shape(vac_Garrett) +
                   tm_symbols(shape = 25,
                              size = 0.75,
                              col = "green",
                              border.col = "red")

map_Garrett_RuralLowIncome_10mi <- map_Garrett_RuralLowIncome +
                                   tm_shape(vac_Garrett_10mi_clipped) +
                                   tm_fill(col = "purple",
                                           alpha = 0.5) +
                                   map_vac_Garrett
map_Garrett_RuralLowIncome_10mi
```

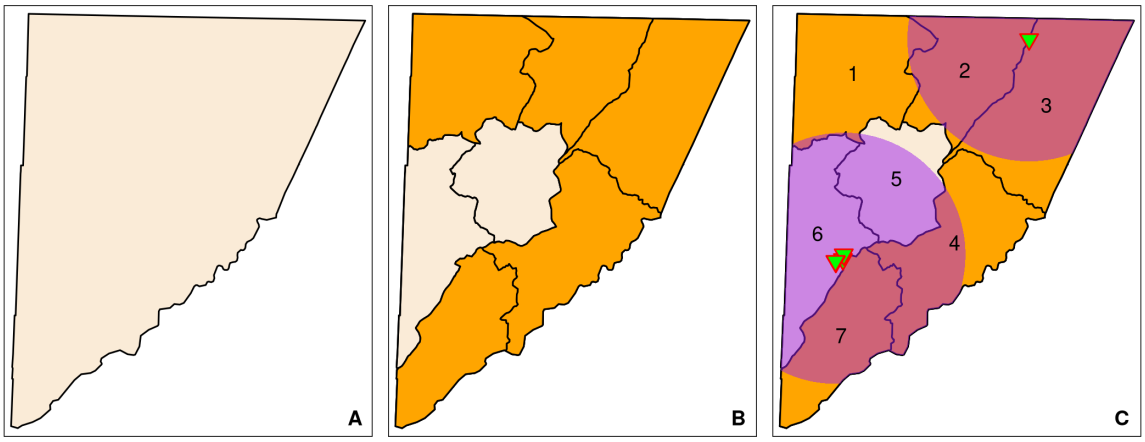



***FIGURE 6.5**. Maps of Garrett County. **A.** County boundaries. **B.** Rural low-income census tracts (highlighted in orange). **C.** Rural low-income census tracts (highlighted in orange) and ranges of vaccination sites (highlighted in purple). Location of vaccination sites are shown as green inverted triangles.*

Garrett County has five rural low-income tracts: tracts 1–4, and tract 7. Tracts 2–4 and 7 have large areas that are within the range of a vaccination site. Tract 1 has only two small sections that are within the range (Figure 6.5 C).

As discussed in Section 1.1, a low-income census tract should have at least 33% of its area outside of the range of a vaccination site to be flagged as a possible vaccination desert. Remember however that this definition has its limitations. It suggests that residential housing is evenly distributed throughout the census tract. This is likely not true for many rural areas.

Regardless, let's determine the relative sizes of the low-income census tracts sections that are outside of the range of a vaccination site. To do so, we first identify the sections that are within the range of a vaccination site, and determine their sizes. We use **`st_intersection()`** to extract the portions of the census tracts that is within the range of a vaccination site. Note that the low-income census tracts are listed first, followed by the 10-mile vaccination site buffer. Also, **`st_intersection()`** will

issue a warning that can be ignored. The resulting spatial object is assigned to **vac_assess**. Next, we calculate the area of the spatial features with **st_area()** (and assign the outcome to **vac_access_area**). Then we calculate the total area of the low-income census tracts (**st_area(Garrett_RuralIncome)**) and assign the output to the object **Garrett_RuralLowIncome_area**.

Since all five low-income tracts of Garrett County overlap with the 10-mile range of a vaccination site, we can calculate the relative area of a low-income tract that is outside the reach of a vaccination site by dividing **vac_access_area** by **Garrett_RuralLowIncome_area**. The quotient (result of the division) is converted into a vector (function **as.vector()**) and subtracted from one. The final ratio is assigned to a new variable (**outside_range_ratio**) to the **Garrett_RuralLowIncome_vac_area** created beforehand.

```
# determine region within range
vac_access <- st_intersection(Garrett_RuralLowIncome, vac_Garrett_10mi)
```

```
## Warning: attribute variables are assumed to be spatially constant throughout all
## geometries
```

```
vac_access_area <- st_area(vac_access)

vac_access_area
```

```
## Units: [m^2]
## [1]  21235139 190265302 267057684 137166602 177668272
```

```
# calcualting total area of each census tract (low income)
Garrett_RuralLowIncome_area <- st_area(Garrett_RuralLowIncome)

Garrett_RuralLowIncome_area
```

```
## Units: [m^2]
## [1] 275216882 208291591 323333623 268461042 214505992
```

```
# copy Garrett_RuralLowIncome
Garrett_RuralLowIncome_vac_area <- Garrett_RuralLowIncome

# Calculate area outside of the range
Garrett_RuralLowIncome_vac_area$outside_range_ratio <-
1 - as.vector(vac_access_area/Garrett_RuralLowIncome_area)

Garrett_RuralLowIncome_vac_area
```

```
## Simple feature collection with 5 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 185230.9 ymin: 173522.8 xmax: 234427.1 ymax: 230941.9
## Projected CRS: NAD83(2011) / Maryland
##     CensusTract               GEO_ID STATE COUNTY  TRACT NAME  LSAD CENSUSAREA
## 530 24023000100 1400000US24023000100    24    023 000100    1 Tract    105.372
## 531 24023000200 1400000US24023000200    24    023 000200    2 Tract     80.293
## 532 24023000300 1400000US24023000300    24    023 000300    3 Tract    124.156
## 533 24023000400 1400000US24023000400    24    023 000400    4 Tract    102.178
## 536 24023000700 1400000US24023000700    24    023 000700    7 Tract     82.476
##      County Urban POP2010 LowIncomeTracts HUNVFlag
## 530 Garrett     0    4003               1        1
## 531 Garrett     0    3937               1        1
## 532 Garrett     0    2857               1        0
## 533 Garrett     0    3337               1        0
## 536 Garrett     0    5726               1        1
##                           geometry outside_range_ratio
## 530 MULTIPOLYGON (((187332 2221...          0.92284216
## 531 MULTIPOLYGON (((202819.1 23...          0.08654353
## 532 MULTIPOLYGON (((222190.9 20...          0.17404914
## 533 MULTIPOLYGON (((205719.8 18...          0.48906329
## 536 MULTIPOLYGON (((197757.6 18...          0.17173283
```

We then extract low-income tracts with an **`outside_range_ratio`** larger than 33% (0.33), which represent possible vaccination deserts.

```
# subset for vaccination deserts
Garrett_RuralVacDeserts <- subset(Garrett_RuralLowIncome_vac_area,
                                  outside_range_ratio > 0.33)

Garrett_RuralVacDeserts
```

```
## Simple feature collection with 2 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
```

```
## Bounding box:  xmin: 186998.7 ymin: 183652.7 xmax: 222098.7 ymax: 230941.9
## Projected CRS: NAD83(2011) / Maryland
##     CensusTract               GEO_ID STATE COUNTY  TRACT NAME  LSAD CENSUSAREA
## 530 24023000100 1400000US24023000100    24    023 000100    1 Tract    105.372
## 533 24023000400 1400000US24023000400    24    023 000400    4 Tract    102.178
##      County Urban POP2010 LowIncomeTracts HUNVFlag
## 530 Garrett     0    4003               1        1
## 533 Garrett     0    3337               1        0
##                           geometry outside_range_ratio
## 530 MULTIPOLYGON (((187332 2221...           0.9228422
## 533 MULTIPOLYGON (((205719.8 18...           0.4890633
```

```
# map
map_Garrett_RuralVacDeserts <- map_Garrett +
                               tm_shape(Garrett_CensusTracts) +
                               tm_borders(col = "black") +
                               tm_shape(Garrett_RuralLowIncome) +
                               tm_polygons(col = "orange") +
                               tm_shape(Garrett_RuralVacDeserts) +
                               tm_polygons(col = "red") +
                               map_vac_Garrett +
                               tm_add_legend(type = "symbol",
                                         shape = c(22, 22, 25),
                                         size = c(0.9, 0.9, 0.65),
                                         col = c("orange", "red", "green"),
                                         border.col = c("black", "black", "red"),
                                         label = c("Rural Low-income Tract",
                                                   "Potential Vaccination Desert",
                                                   "COVID-19 Vaccination Site")) +
                              tm_legend(position = c(0.65, 0.02),
                                        width = 0.6) +
                              tm_layout(legend.text.size = 0.85)

map_Garrett_RuralVacDeserts

#save map
tmap_save(map_Garrett_RuralVacDeserts,
          filename = "figures/map_Garrett_RuralVacDeserts.png")
```

Note that you may need to adjust the position of the legend manually for the saved file. I adjusted the legend position as follows to save the map to my liking (just replace the lines containing the **`tm_legend(position ...)`**).

```
tm_legend(position = c(0.65, 0.02),
                      width = 0.6) +
```

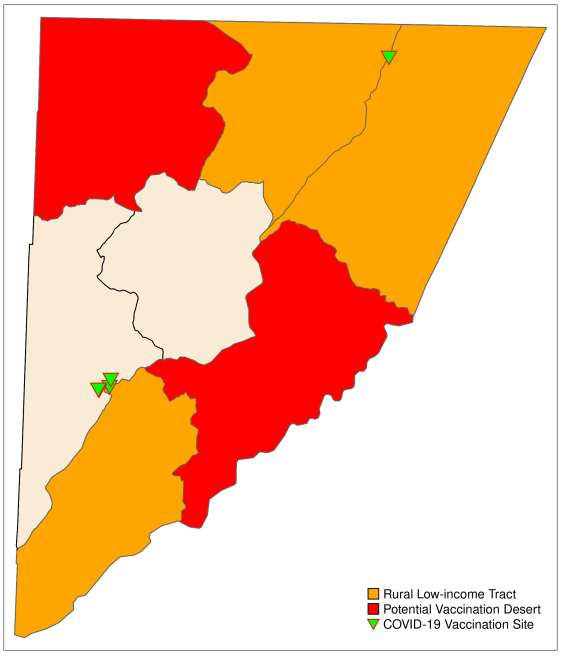



***FIGURE 6.6**. Map of Garrett County showing rural low-income census tracts (highlighted in red) that are potential vaccination deserts.*

## 6.3 Possible Vaccination Deserts in Baltimore City

We can use a similar approach to assess whether Baltimore City has "COVID-19 Vaccination Deserts." Of course, there are some modifications necessary.

- Remember, in an urban setting, we defined a limited-access census tract as a census tract where more than 33% of the area is outside of a 0.5 mile range to a vaccination site (0.5 mile is roughly 800 m). Thus, instead of creating a 10-mile buffer around vaccination sites, we create a 0.5 mile (800 m) buffer.
- Baltimore City is surrounded by other (Maryland) counties, in which Baltimore City residents can get vaccinated. Therefore, vaccination sites will be clipped to Baltimore City limits that are "expanded" by 800 m.
- As you will see, in contrast to Garrett County, Baltimore City has low-income tracts that are outside of the 0.5 miles range of vaccination sites (Figure 6.7 C), which we have to account for.

The first steps (up to mapping low-income tracts and buffered vaccination sites) in general follow the example we used for Garrett County:

1. We create a base map for Baltimore City and a map showing census tracts.
2. We extract low-income tracts and double check that we only have urban tracts.
3. We create a 800 m buffer around Baltimore City limits and identify vaccination sites that are within the extended Baltimore City limits.
4. We create a 800 m buffer around the vaccination sites and clip the buffered sites to the city limits (for aesthetics).
5. We plot Baltimore City's census tracts, low-income census tracts, and buffered vaccination sites.

```
# Create Baltimore City map
# From Census Tract Map
BC_CensusTracts <- subset(MD_CensusTracts_6487, County == "Baltimore City")

# Baltimore City outline
BC <- st_union(BC_CensusTracts)

# Map BC (Figure 6.7 A)

map_BC <- tm_shape(BC) +
          tm_fill(col = "antiquewhite") +
          tm_borders(col = "black")

map_BC
```

```
# Extract low income census tracts.
# check if non urban tracts are present.
subset(BC_CensusTracts, Urban == 0)  # just to be sure
```

```
## Simple feature collection with 0 features and 13 fields
## Bounding box:  xmin: NA ymin: NA xmax: NA ymax: NA
## Projected CRS: NAD83(2011) / Maryland
##  [1] CensusTract     GEO_ID          STATE           COUNTY
##  [5] TRACT           NAME            LSAD            CENSUSAREA
##  [9] County          Urban           POP2010         LowIncomeTracts
## [13] HUNVFlag        geometry
## <0 rows> (or 0-length row.names)
```

```
# <0 rows>, we are good to go
# Extract LowIncomeTracts
BC_LowIncome <- subset(BC_CensusTracts, LowIncomeTracts == "1")

# Map census tracts, highlight low-income census tracts orange (Fig. 6.7 B)
map_BC_LowIncome <- map_BC +
                    tm_shape(BC_CensusTracts) +
                    tm_borders(col = "black") +
                    tm_shape(BC_LowIncome) +
                    tm_fill(col = "orange") +
                    tm_borders(col = "black")

map_BC_LowIncome
```

```
# Create 800 m buffer for Baltimore City
BC_800m <- st_buffer(BC, dist = 800)

# Identify Vaccination Sites in BC (for mapping sites); ignore warning
vac_BC <- st_intersection(VaccineSites_6487, BC)

# Identify Vaccination Sites within "expanded" City; ignore warning
vac_BC_ext <- st_intersection(VaccineSites_6487, BC_800m)

# Create 0.5 mi (800m) buffer
vac_BC_800m <- st_buffer(vac_BC_ext, dist = 800)

# Unify
vac_BC_800m <- st_union(vac_BC_800m)

# Clip to BC boundaries (NOT BC_800m)
vac_BC_800m_clipped <- st_intersection(BC, vac_BC_800m)

# Map Vaccination Sites and buffer, only map sites in the city (Figure 6.7 C)
map_vac_BC <- tm_shape(vac_BC) +
              tm_symbols(shape = 25,
                         size = 0.5,
                         col = "green",
                         border.col = "black")

# map (figure 6.7 C)
map_BC_LowIncome_800m <- map_BC_LowIncome +
                         tm_shape(vac_BC_800m_clipped) +
                         tm_fill(col = "purple",
                                 alpha = 0.5) +
                         map_vac_BC

map_BC_LowIncome_800m
```

The **R** script produces maps similar to the plots in Figure 6.7 A–C. Indeed, Baltimore City has quite a few low-income tracts that are outside the 0.5 mile range of vaccination sites. These by definition are possible vaccination deserts. We need to separate them from the low-income tracts that are within the range of vaccination sites. Otherwise, we cannot calculate the area that overlap. The package **dplyr** has the function **anti_join()** that removes rows of a data frame if the rows are present in another data frame. However, the function does not work with 2 **sf** objects. The 2nd object needs to be a non spatial data frame.

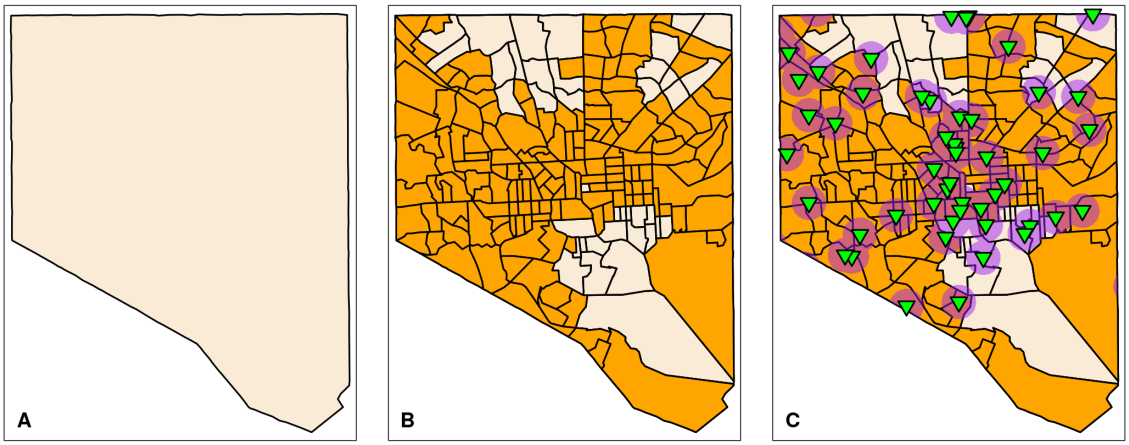



***FIGURE 6.7****. Maps of Baltimore City.* ***A.*** *Outline of Baltimore City.* ***B.*** *Low-income census tracts (highlighted in orange).* ***C.*** *Low-income census tracts (highlighted in orange) and a range of vaccination sites (highlighted in purple). Location of vaccination sites are shown as green inverted triangles.*

To separate the low-income tracts that share some area with the buffered vaccination sites, we use the following approach.

1. We identify the area of the census tracts that overlap with the buffered vaccination site and name these **BC_vac_access**. We are using the function **st_intersection()**. As the function name implies, a spatial data frame is returned that only contains the intersection of both the census tracts and the buffered vaccination sites (Figure 6.8 A).

2. We obtain the tracts that do not overlap (the "no-access" tracts) by removing the tracts with access from the overall low-income tracts with the function **dplyr::anti_join()**. This possible because in **sf** objects the function **st_intersection()** does not alter non spatial information.

3. We obtain the entire census tracts that overlap with the buffered vaccination sites. We assign these tracts to the object **BC_withvac_access**.

The following **R** scripts produces a few plots that visualize the above process. The plots are shown in Figure 6.8 A–C. **st_intersection()** will again give warnings that can be ignored.

```
#=====================================================
# Step 1: identify the area that overlaps
#=====================================================

BC_vac_access <- st_intersection(BC_LowIncome, vac_BC_800m_clipped)

# Map BC_vac_access (Fig. 6.8 A)
# create an outline of Low Access census tracts
BC_LowIncome_outline <- st_union(BC_LowIncome)

# base map
BC_LI <- tm_shape(BC_LowIncome_outline) +
      tm_borders(col = "darkorange3",
                 lwd = 1.5) +
      tm_shape(BC_LowIncome_outline) +
      tm_fill(col = "antiquewhite") +
      tm_borders(col = "darkorange3",
                 lwd = 1.5)

map_BC_vac_access <- BC_LI +
      tm_shape(BC_vac_access) +  # area of overlap
      tm_fill(col = "purple",
              alpha = 0.25) +
      tm_borders(col = "darkblue")

map_BC_vac_access

#=====================================================
# Step 2: identify census tracts that do not overlap
#=====================================================

# Remove geometries from BC_vac_access
# drop spatial information from overlap (BC_vac_access
BC_vac_access_nsp <- st_drop_geometry(BC_vac_access)

BC_novac_access <- dplyr::anti_join(BC_LowIncome, BC_vac_access_nsp, by = "NAME")

# Map BC_novac_access (Fig. 6.8 B)
map_BC_novac_access <- BC_LI +
    tm_shape(BC_novac_access) + # tracts w/ no access
    tm_polygons(col = "red")

map_BC_novac_access

#=============================================
# Step 3 get census tracts with access (overlap)
#=============================================

# Remove geometries from BC_novac_access
BC_novac_access_nsp <- st_drop_geometry(BC_novac_access)

# Remove no-access from low-income tract
```

```
BC_withvac_access <- dplyr::anti_join(BC_LowIncome, BC_novac_access_nsp, by = "NAME")

# Map BC_withvac_access (Fig. 6.8 C)
map_BC_withvac_access <- BC_LI +
            tm_shape(BC_withvac_access) +
            tm_fill(col = "orange",
                    alpha = 0.5) +
            tm_borders(col = "darkorange3")

map_BC_withvac_access
```

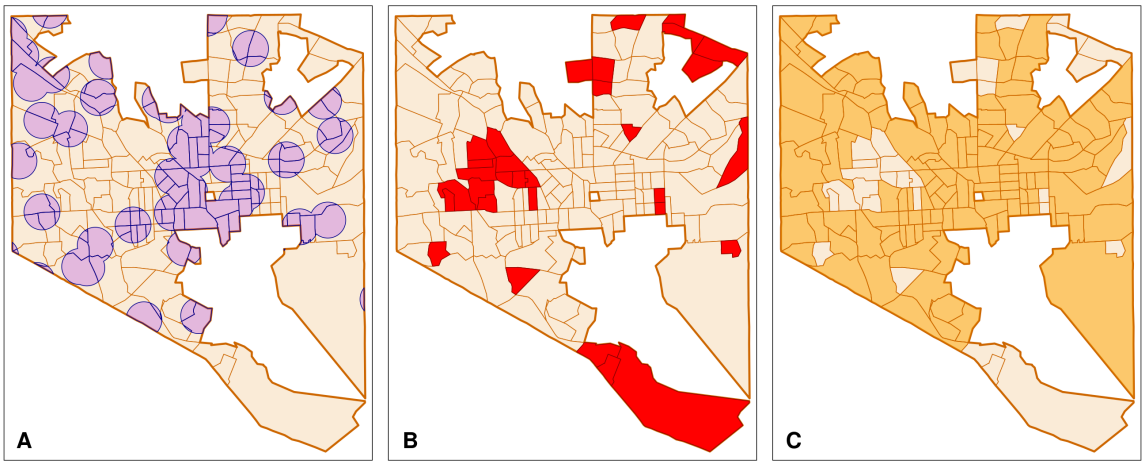



***FIGURE 6.8**. Outcome of spatial manipulations on low-income tracts of Baltimore City. **A.** Intersection of low-income tracts and buffered vaccination sites (highlighted in purple) generated by* **`st_intersection()`**. ***B.** Low-income tracts outside of the 0.5 mi buffer around vaccination sites ("no-access" low-income tracts; highlighted in red). **C.** Low-income census tracts that intersect with the vaccination site buffer ("with-access" low-income tracts; highlighted in orange).*

We have two sets of complete low-income census tracts: (1) census tracts that do not overlap ("no-access" low-income tracts; Figure 6.8 B), and (2) those that have areas within the 0.5 mile range to vaccination sites ("with-access" low-income tracts; Figure 6.8 C). These data sets allow us to calculate the portion of the area of a low-income tract that overlaps with the 0.5 mile buffer around vaccination sites. For these calculations it is important that the census tracts in both data frames (**`BC_vac_access`** and **`BC_withvac_access`**) are in the same order. To verify, we use the function **`identical()`**. It tests if two objects are exactly the same. We cannot compare the entire data frames since the geometry columns of the data frames differ. Furthermore, **`anti_join()`** from the **`dplyr`** package unfortunately does not preserve the row numbers. Instead we compare the entries in the variable **`GEO_ID`** as it is unique to each tract.

```
# check if NAME viable is the same in both data frames
identical(BC_vac_access$GEO_ID, BC_withvac_access$GEO_ID)
```

```
## [1] TRUE
```

**TRUE** is returned, confirming that in both data sets the order of the census tracts is identical. We now identify possible low-income tracts with limited-access to vaccination sites following the example of Garrett County. We first divide the area of the intersection (**BC_vac_access**) by the total area of the tract (**BC_withvac_access**), then subtract the quotient from 1, and lastly extract tracts that share no more than 33% with the buffer around vaccination sites.

```
# calculate area of the portions that are within the 0.5 mile range
BC_vac_access_area <- st_area(BC_vac_access)

BC_vac_access_area
```

```
## Units: [m^2]
##   [1]  426051.567  595722.353 1204742.968  437131.380    1595.308   68083.216
##   [7]  420835.141  133338.221  225786.014  326691.483   15006.228  281497.037
##  [13]  247188.754   74697.281   38425.821  202105.805  291927.929  251092.218
##  [19]  282982.776  327353.482   53235.925  128529.689  534636.724  291398.885
##  [25]  354049.787   87394.618  787116.713  544379.226   91860.517  204809.073
##  [31]  308259.900  706061.179  356875.579  934568.962  408658.063  270852.756
##  [37]  382688.840  503295.282 1020097.808   24903.375  318708.773   26947.737
##  [43]    8289.565  507238.123  293322.753  314815.420  593204.994  316978.938
##  [49]  259617.680  438479.266  298329.823  783814.209  681257.558   97632.114
##  [55]  875160.584  143299.770   12652.807   70685.473   46630.607  276051.933
##  [61]  484918.575  283264.985  340923.971  177094.232  160975.836  248396.176
##  [67]   92184.117  271324.877  265150.352  386301.387  207259.904  240586.509
##  [73]  180069.772 1248149.304   92392.360  130851.261  481958.333  989663.497
##  [79]    3416.019  292827.299   74827.221 1723568.441 1051350.778   29350.392
##  [85] 1186969.701  541803.150  407432.500   90406.402   44231.254  662949.100
##  [91] 1248459.714  173237.048  671154.321  130106.429  606352.391  536479.279
##  [97]   10152.565  423422.520  648875.862 1576870.203  171886.689  730677.745
## [103]  311145.862   84521.796  481436.695  801923.164  783199.034  575371.412
## [109]   62320.814   94295.486  305834.031   42781.522  753480.037  940759.662
## [115]  401882.661   46069.995  418841.769  294534.122  426904.915  758439.728
## [121] 1306801.799 2602953.864 1175111.918 1363236.743  113529.070  854045.245
## [127]  228777.089  408589.497  480149.958  924871.056
```

```
# Calculate total area of each low-income census tracts that intersect with a buffer
BC_withvac_access_area <- st_area(BC_withvac_access)

BC_withvac_access_area
```

```
## Units: [m^2]
##   [1]   427345.4   595722.4  1204743.0   437131.4   257719.1   186938.5
##   [7]   429559.1   314502.5   225786.0   326691.5  2299202.9   364736.1
##  [13]   863578.2   268338.1   440547.7   251437.0   350511.1   336465.9
##  [19]   298794.8   327485.2   659655.6  2031460.0   795632.8   350196.9
##  [25]   744957.2   446013.3   844748.8   685535.9   453673.4   304861.7
##  [31]   308259.9   706061.2   356875.6   991486.1   408658.1   270852.8
##  [37]   532064.0   503295.3  1040072.6   298538.9   362711.1   357715.6
##  [43]  2825307.7   904919.4   679703.3   718015.6   593205.0   395240.6
##  [49]   391087.0  1090207.3  1455134.1  1489384.5  1691295.7   775813.7
##  [55]   938545.1   382271.0   255419.0   398742.1  1050387.4   276051.9
##  [61]   487427.0   310599.0   340924.0   234698.9   290125.6   382635.7
##  [67]   277363.1   380917.3   265150.4   931568.8   238474.1   398644.6
##  [73]   683109.9  1809786.4  1199178.7   288614.6  1082656.5  1477763.3
##  [79]  2358753.8   330709.1   975141.6  2585246.1  1283257.1   816551.0
##  [85]  2325837.4  2578126.4  1306904.1  2240766.9  1637342.2  1457068.4
##  [91]  1632642.9  1025175.6   885456.9   612403.1   885014.5  1236183.8
##  [97]  1334571.4  1365204.3  4464233.6  3218863.2 17165266.5   762390.6
## [103]   311481.4   310251.5   972807.3  1097472.2  2177944.3   696604.8
## [109]  1269181.3   681341.4  1132474.6  1085186.7   845572.7  1735994.4
## [115]   566402.8   516018.6   729234.7   924512.3   739565.7   804749.8
## [121]  1418617.5  3493791.3  2414014.1  3224540.7   979278.6  1645592.4
## [127]   553237.0  1819413.2   870269.1   924956.5
```

```
# copy BC_withvac_access
BC_withvac_access_vac_area <- BC_withvac_access

# calcuate area outside of the range
BC_withvac_access_vac_area$outside_range_ratio <-
    1 - as.vector(BC_vac_access_area/BC_withvac_access_area)

BC_withvac_access_vac_area
```

```
## Simple feature collection with 130 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 424860.5 ymin: 172357 xmax: 440631.8 ymax: 189404.6
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##    CensusTract               GEO_ID STATE COUNTY  TRACT NAME  LSAD CENSUSAREA
## 1  24510030100 1400000US24510030100    24    510 030100  301 Tract      0.165
```

```
## 2  24510030200 1400000US24510030200    24    510 030200  302 Tract      0.195
## 3  24510040100 1400000US24510040100    24    510 040100  401 Tract      0.463
## 4  24510040200 1400000US24510040200    24    510 040200  402 Tract      0.169
## 5  24510060200 1400000US24510060200    24    510 060200  602 Tract      0.100
## 6  24510060300 1400000US24510060300    24    510 060300  603 Tract      0.072
## 7  24510060400 1400000US24510060400    24    510 060400  604 Tract      0.166
## 8  24510070200 1400000US24510070200    24    510 070200  702 Tract      0.120
## 9  24510070300 1400000US24510070300    24    510 070300  703 Tract      0.088
## 10 24510070400 1400000US24510070400    24    510 070400  704 Tract      0.126
##            County Urban POP2010 LowIncomeTracts HUNVFlag
## 1  Baltimore City     1    3065               1        1
## 2  Baltimore City     1    2342               1        0
## 3  Baltimore City     1    4006               1        0
## 4  Baltimore City     1     838               1        0
## 5  Baltimore City     1    3265               1        0
## 6  Baltimore City     1    1800               1        0
## 7  Baltimore City     1    1183               1        1
## 8  Baltimore City     1    3782               1        0
## 9  Baltimore City     1    1042               1        0
## 10 Baltimore City     1    1241               1        0
##                          geometry outside_range_ratio
## 1  MULTIPOLYGON (((434910.7 17...         0.003027589
## 2  MULTIPOLYGON (((433985.8 18...         0.000000000
## 3  MULTIPOLYGON (((433926.2 18...         0.000000000
## 4  MULTIPOLYGON (((432439.3 18...         0.000000000
## 5  MULTIPOLYGON (((436345.8 18...         0.993809897
## 6  MULTIPOLYGON (((435592.6 18...         0.635798801
## 7  MULTIPOLYGON (((435161.1 18...         0.020309062
## 8  MULTIPOLYGON (((436406 1810...         0.576034505
## 9  MULTIPOLYGON (((435524.7 18...         0.000000000
## 10 MULTIPOLYGON (((435290.8 18...         0.000000000
```

```
# subset for limited access
BC_limvac_access <- subset(BC_withvac_access_vac_area, outside_range_ratio > 0.33)

BC_limvac_access
```

```
## Simple feature collection with 75 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 424876.1 ymin: 172357 xmax: 440631.8 ymax: 189404.6
## Projected CRS: NAD83(2011) / Maryland
## First 10 features:
##    CensusTract               GEO_ID STATE COUNTY  TRACT   NAME  LSAD CENSUSAREA
## 5  24510060200 1400000US24510060200    24    510 060200    602 Tract      0.100
## 6  24510060300 1400000US24510060300    24    510 060300    603 Tract      0.072
## 8  24510070200 1400000US24510070200    24    510 070200    702 Tract      0.120
## 11 24510080101 1400000US24510080101    24    510 080101 801.01 Tract      0.894
```

```
## 13 24510080200 1400000US24510080200    24    510 080200    802 Tract      0.332
## 14 24510080301 1400000US24510080301    24    510 080301 803.01 Tract      0.104
## 15 24510080302 1400000US24510080302    24    510 080302 803.02 Tract      0.170
## 21 24510090100 1400000US24510090100    24    510 090100    901 Tract      0.254
## 22 24510090200 1400000US24510090200    24    510 090200    902 Tract      0.671
## 25 24510090500 1400000US24510090500    24    510 090500    905 Tract      0.286
##            County Urban POP2010 LowIncomeTracts HUNVFlag
## 5  Baltimore City     1    3265               1        0
## 6  Baltimore City     1    1800               1        0
## 8  Baltimore City     1    3782               1        0
## 11 Baltimore City     1    3881               1        1
## 13 Baltimore City     1    1585               1        0
## 14 Baltimore City     1    2084               1        0
## 15 Baltimore City     1    2937               1        1
## 21 Baltimore City     1    4251               1        1
## 22 Baltimore City     1    3243               1        0
## 25 Baltimore City     1    1964               1        1
##                          geometry outside_range_ratio
## 5  MULTIPOLYGON (((436345.8 18...           0.9938099
## 6  MULTIPOLYGON (((435592.6 18...           0.6357988
## 8  MULTIPOLYGON (((436406 1810...           0.5760345
## 11 MULTIPOLYGON (((435823.6 18...           0.9934733
## 13 MULTIPOLYGON (((435460.7 18...           0.7137622
## 14 MULTIPOLYGON (((436130 1816...           0.7216300
## 15 MULTIPOLYGON (((436252.3 18...           0.9127772
## 21 MULTIPOLYGON (((433659.8 18...           0.9192974
## 22 MULTIPOLYGON (((435030.3 18...           0.9367304
## 25 MULTIPOLYGON (((434050.3 18...           0.5247381
```

**BC_limvac_access** contains all the low-income census tracts with limited access to a vaccination site which as such qualify as vaccination deserts (based on our definition in Section 1.1). To obtain a list or data frame with all possible vaccination deserts we have to add the "no-access" low-income tracts. We can do so with the function **rbind()**, but we need to add an **outside_range_ratio** variable to the **BC_novav_access** data frame. Since the all census tracts in this data frame are outside the range, all entries are set to one.

```
# combine no access with limited acces
BC_novac_access$outside_range_ratio <- 1

BC_VacDeserts <- rbind(BC_novac_access, BC_limvac_access)

BC_VacDeserts
```

```
## Simple feature collection with 106 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: 424876.1 ymin: 169994.5 xmax: 440631.8 ymax: 189417.3
## Projected CRS: NAD83(2011) / Maryland
```

```
## First 10 features:
##    CensusTract               GEO_ID STATE COUNTY  TRACT    NAME  LSAD
## 1  24510060100 1400000US24510060100    24    510 060100     601 Tract
## 2  24510070100 1400000US24510070100    24    510 070100     701 Tract
## 3  24510090600 1400000US24510090600    24    510 090600     906 Tract
## 4  24510150100 1400000US24510150100    24    510 150100    1501 Tract
## 5  24510150200 1400000US24510150200    24    510 150200    1502 Tract
## 6  24510150300 1400000US24510150300    24    510 150300    1503 Tract
## 7  24510150400 1400000US24510150400    24    510 150400    1504 Tract
## 8  24510150500 1400000US24510150500    24    510 150500    1505 Tract
## 9  24510150600 1400000US24510150600    24    510 150600    1506 Tract
## 10 24510150701 1400000US24510150701    24    510 150701 1507.01 Tract
##    CENSUSAREA         County Urban POP2010 LowIncomeTracts HUNVFlag
## 1       0.092 Baltimore City     1    3222               1        0
## 2       0.112 Baltimore City     1    2957               1        0
## 3       0.154 Baltimore City     1    3402               1        1
## 4       0.143 Baltimore City     1    3211               1        0
## 5       0.162 Baltimore City     1    2699               1        0
## 6       0.154 Baltimore City     1    2478               1        1
## 7       0.315 Baltimore City     1    3724               1        0
## 8       0.367 Baltimore City     1    1543               1        0
## 9       0.370 Baltimore City     1    3412               1        1
## 10      0.331 Baltimore City     1    1696               1        1
##    outside_range_ratio                       geometry
## 1                    1 MULTIPOLYGON (((436868.1 18...
## 2                    1 MULTIPOLYGON (((436868.1 18...
## 3                    1 MULTIPOLYGON (((435043.8 18...
## 4                    1 MULTIPOLYGON (((431016 1817...
## 5                    1 MULTIPOLYGON (((430202.4 18...
## 6                    1 MULTIPOLYGON (((429669 1824...
## 7                    1 MULTIPOLYGON (((429829.8 18...
## 8                    1 MULTIPOLYGON (((428945.2 18...
## 9                    1 MULTIPOLYGON (((428244.4 18...
## 10                   1 MULTIPOLYGON (((428077.1 18...
```

Last, we map the potential vaccination deserts of Baltimore City. To make the map more appealing, we will clip the census tract map to the Baltimore City physical boundaries map.

```
BC_county_boundary <- subset(MD_counties_6487, COUNTY == "Baltimore City")

BC_county_boundary <- st_union(st_buffer(BC_county_boundary, dist = 20))

BC_CensusTracts_phys <- st_intersection(BC_CensusTracts, BC_county_boundary)

BC_LowIncome_phys <- st_intersection(BC_LowIncome, BC_county_boundary)

BC_VacDeserts_phys <- st_intersection(BC_VacDeserts, BC_county_boundary)

map_BC_phys <- tm_shape(BC) +
               tm_fill(col = "lightcyan3") +
```

```
                tm_borders() +
                tm_shape(BC_CensusTracts_phys) +
                tm_fill(col = "antiquewhite") +
                tm_shape(BC_LowIncome_phys) +
                tm_fill(col = "orange") +
                tm_shape(BC_VacDeserts_phys) +
                tm_fill(col = "red") +
                tm_shape(BC_CensusTracts_phys) +
                tm_borders() +
                tm_shape(BC) +
                tm_borders(col = "black",
                           lwd = 1.5) +
                map_vac_BC +
#Legend
                tm_add_legend(type = "symbol",
                    labels =c('Vaccination Site',
                              'Low-income Tract',
                              'Low-income Tract flagged',
                              'as Vaccination Desert'),
                    size = c(0.5, 0.65, 0.65, 0),
                    shape = c(25, 22, 22, 20),
                    col = c('green', 'orange', 'red', 'white'),
                    title = 'Legend') +
                tm_layout(fontfamily = 'Times',
                    legend.position = c(0.025, 0.195),
                    legend.text.size = 0.75,
                    legend.width = 1) +
                tm_scale_bar(position = c("0.015", "0.0015"),
                             text.size = 0.5) + #add (default) scale
                tm_compass(type = "8star", size = 2.5,
                           position = c("0.1125", 0.07)) #add compass

map_BC_phys
```

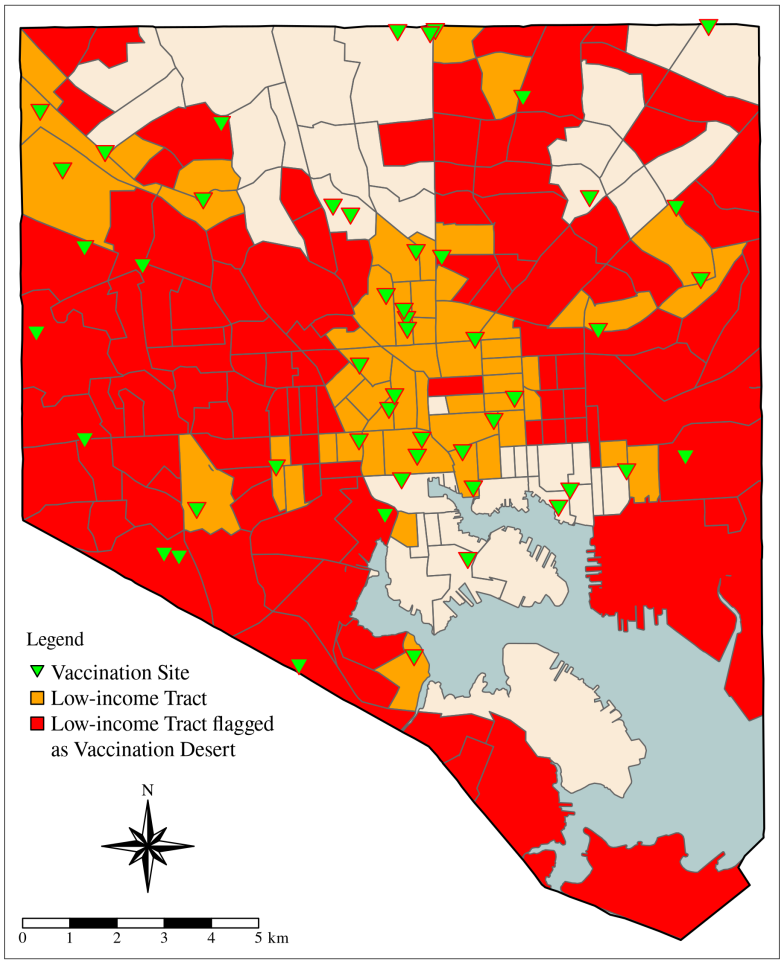



***FIGURE 6.9**. Map of Baltimore City showing low-income census tracts (highlighted in orange) and possible vaccination deserts (highlighted in red).*

# *Bibliography*


Allaire, J., *et al.* (2020) *rmarkdown: Dynamic Documents for R*. R package version 2.6 https://github.com/rstudio/rmarkdown

IOGP Geomatics Committee (2021) *EPSG Geodetic Parameter Dataset*. https://epsg.org/home.html (accessed on Apr 5, 2021)

Lovelace, R., Nowosad, J., and Jannes, M. (2021) *Geocomputation with R*. https://geocompr.robinlovelace.net/index.html (version: 2021-04-06)

MapTiler Team (2019) *epsg.io, Coordinate Systems Worldwide*. https://epsg.io (accessed on Apr 5, 2021)

National Geodetic Survey (2018) *North American Datum of 1983*. https://www.ngs.noaa.gov/datums/horizontal/north-american-datum-1983.shtml (accessed on Mar 9, 2021)

Neuwirth, E. (2014) *RColorBrewer: ColorBrewer Palettes*. R package version 1.1-2 https://CRAN.R-project.org/package=RColorBrewer

Pebesma, E. (2018) *Simple Features for R: Standardized Support for Spatial Vector Data*. The R Journal **10**:439–446. doi:10.32614/RJ-2018-009

Pebesma, E. and Bivand, R. (2021) *Spatial data science with applications in R*. https://keen-swartz-3146c4.netlify.app/index.html (version: 2021-04-04)

R Core Team (2020) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing Vienna, Austria. https://www.R-project.org/

Reger, J.P. (2013) *Maryland Geological Survey: a user's guide to Maryland coordinate system*. http://www.mgs.md.gov/geology/maryland_coordinate_system.html (accessed on Apr 6, 2021)

Schauberger, P. and Walker, A. (2020) *openxlsx: Read, Write and Edit xlsx Files*. R package version 4.2.3 https://CRAN.R-project.org/package=openxlsx

Tennekes, M. (2018) *tmap: Thematic maps in R*. Journal of Statistical Software **84**:1–39. doi:10.18637/jss.v084.i06

United Nations General Assembly (2015) *Resolution 70/1. Transforming our world: the 2030 Agenda for Sustainable Development*. Tech. rep. https://www.un.org/en/development/desa/population/migration/generalassembly/docs/globalcompact/A_RES_70_1_E.pdf

U.S. Census Bureau (2019) *Glossary*. https://www.census.gov/programs-surveys/geography/about/glossary.html (accessed on Apr 5, 2021)

U.S. Department of Agriculture (2009) *Access to affordable and nutritious food: measuring and understanding food deserts and their consequences*. Tech. rep. https://www.ers.usda.gov/webdocs/publications/42711/12716_ap036_1_.pdf?v=8903.9

Ver Ploeg, M., Nulph, D., and Williams, R. (2011) *Mapping food deserts in the United States*. https://www.ers.usda.gov/amber-waves/2011/december/data-feature-mapping-food-deserts-in-the-us/ (accessed on Apr 5, 2021)

Wickham, H., François, R., Henry, L., and Müller, K. (2020) *dplyr: A Grammar of Data Manipulation*. R package version 1.0.0 https://CRAN.R-project.org/package=dplyr

Wickham, H. and Henry, L. (2020) *tidyr: Tidy Messy Data*. R package version 1.1.0 https://CRAN.R-project.org/package=tidyr

Xie, Y., Allaire, J., and Grolemund, G. (2018) *R Markdown: The Definitive Guide*. Chapman and Hall/CRC, Boca Raton, Florida. https://bookdown.org/yihui/rmarkdown iSBN 9781138359338

Xie, Y., Dervieux, C., and Riederer, E. (2020) *R Markdown Cookbook*. Chapman and Hall/CRC, Boca Raton, Florida. https://bookdown.org/yihui/rmarkdown-cookbook iSBN 9780367563837

# *Appendices*

## Appendix A: Terms of Use

By downloading the data from the cloud storage service you agree to and accept the terms of service of MEGA LIMITED (https://mega.nz/terms).

**Use constraints**: Code and Spatial Data, and the information therein, (collectively the "Data") is provided "as is" without warranty of any kind, either expressed, implied, or statutory. You (the user) assume the entire risk as to quality and performance of the Data. No guarantee of accuracy is granted, nor is any responsibility for reliance thereon assumed. In no event shall the author be liable for direct, indirect, incidental, consequential or special damages of any kind. The author does not accept liability for any damages or misrepresentation caused by inaccuracies in the Data or as a result to changes to the Data. No guarantee of accuracy is granted, nor is any responsibility for reliance thereon assumed. In no event shall the author be liable for direct, indirect, incidental, consequential or special damages of any kind, nor is there responsibility assumed to maintain the Data in any manner or form. Data can be freely distributed as long as the conditions and user constraints of the original data sources are observed. It is the responsibility of you (the user) to ensure compliance. Data sources and links to data sources are listed in Appendix B: Data Sources.

## Appendix B: Data Sources

### Food Access Research Atlas Data (2015 Dataset)

| | |
|---|---|
| File: | FoodAccess2015.xlsx |
| Source: | https://www.ers.usda.gov/data-products/food-access-research-atlas/download-the-data/ |

### Maryland Census Tracts (2010)

| | |
|---|---|
| Files: | • **gz_2010_24_140_00_500k.dbf**<br>• **gz_2010_24_140_00_500k.prj**<br>• **gz_2010_24_140_00_500k.shp**<br>• **gz_2010_24_140_00_500k.shx**<br>• **gz_2010_24_140_00_500k.xml** |
| Source: | https://www.census.gov/geographies/mapping-files/time-series/geo/carto-boundary-file.2010.html |
| **URL** (Block Groups: Maryland MD): | https://www2.census.gov/geo/tiger/GENZ2010/gz_2010_24_140_00_500k.zip |

### List of COVID-19 Vaccination Sites

| | |
|---|---|
| File: | **MD_Covid19_VacSites_2021-04-04.csv** |

The dataset was slightly modified for educational purposes. An additional variable (column) — **Location** — was added and variables rearranged.

#### Original dataset

| | |
|---|---|
| Source: | https://coronavirus.maryland.gov/datasets/all-maryland-vaccination-sites-1 |
| Credit: | Maryland Department of Health COVID-19 Testing Task Force |
| Metadata: | https://www.arcgis.com/sharing/rest/content/items/d677f143334648a1a40b84d94df8e134/info/metadata/metadata.xml?format=default&output=html |

### Maryland Physical Boundaries — County Boundaries (Detailed)

| | |
|---|---|
| Files: | • **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).cpg**<br>• **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).dbf**<br>• **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).prj**<br>• **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).shp**<br>• **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).shx**<br>• **Maryland_Physical_Boundaries_-_County_Boundaries_(Detailed).xml** |
| Source: | https://data.imap.maryland.gov/datasets/maryland-physical-boundaries-county-boundaries-detailed?geometry=-80.968%2C38.061%2C-73.569%2C39.559 |
| Credit: | MD iMAP, SHA |
| Metadata: | https://www.arcgis.com/sharing/rest/content/items/2315ef0b071a4ec59420e3d342dbcfe2/info/metadata/metadata.xml?format=default&output=html |

### Maryland County Boundaries

| | |
|---|---|
| Files: | • **r(fsize(ftype('MD_counties_CT.dbf'))**<br>• **r(fsize(ftype('MD_counties_CT.prj'))**<br>• **r(fsize(ftype('MD_counties_CT.shp'))**<br>• **r(fsize(ftype('MD_counties_CT.shx'))** |
| Source: | Based on Census Tracts (2010) polygons (see above). The file was created by unifying the census tracts of each county. |