

Comparative Examples of Geocoding and Mapping Techniques in SAS® and R Using Centers for Disease Control Survey Data

Joshua J. Cook, M.S. DS, M.S. CRM, ACRP-PM, CCRC, University of South Carolina (USC)

Louise S. Hadden, B.A., Independent Consultant

Swann Arp Adams, Ph.D, M.S., FACE, University of South Carolina (USC)

ABSTRACT

The Centers for Disease Control (CDC) maintains the Behavioral Risk Factor Surveillance System (BRFSS), which is a rich source of survey data for public health and epidemiological research. BRFSS collects data in all 50 states as well as the District of Columbia and three U.S. territories, and includes data on health-related risk behaviors, chronic health conditions, and use of preventive services. Geospatial analysis is a crucial tool in public health and epidemiology, offering insights into geographic patterns and trends. This presentation aims to provide a comprehensive comparison of geocoding and mapping techniques using SAS® and R, utilizing data from the CDC BRFSS. By demonstrating how to geocode and visually map public health data, we seek to empower researchers and practitioners with the skills necessary to conduct basic spatial analyses. In this session, we will illustrate the process of geocoding and mapping BRFSS data using SAS® PROC GEOCODE and PROC SGMAP procedures, alongside equivalent methods in R, using packages such as tidygeocoder, ggplot2, and plotly. Attendees will gain insights into the strengths and limitations of each approach, understanding how to choose the appropriate tools for their specific needs. Key topics will include an overview of the CDC BRFSS dataset and its relevance to public health, step-by-step guidance on geocoding and generating spatial coordinates, techniques for creating informative and aesthetically pleasing maps, with direct comparisons between SAS® and R in terms of functionality, ease of use, and output quality. By the end of this lecture, participants will be equipped with practical knowledge to apply geocoding and mapping techniques in their work, enhancing their ability to analyze spatial data and make data-driven decisions in public health.

INTRODUCTION

THE BEHAVIORAL RISK FACTOR SURVEILLANCE SYSTEM (BRFSS)

The Behavioral Risk Factor Surveillance System (BRFSS), maintained by the Centers for Disease Control and Prevention (CDC), is one of the largest ongoing public health surveillance systems in the world. Since its inception in 1984, the BRFSS has collected state-level data on a wide array of health-related risk behaviors, chronic health conditions, and use of preventive services among U.S. residents. This extensive dataset has become a vital resource for public health and epidemiological research, providing insights that drive policy decisions, healthcare interventions, and public health strategies across the nation.

The design of the CDC BRFSS survey is complex, utilizing a multistage probability sampling method that ensures the data collected is representative of the population. Key components of the survey design include the primary sampling unit (PSU), the stratum weight (STSTR), and the design weight used in raking (WT2RAKE), which are crucial for producing valid statistical estimates. The survey collects data across all 50 states, the District of Columbia, and three U.S. territories, covering a diverse set of variables that address various aspects of public health.

Among the vast array of variables collected by the BRFSS, specific groups of variables are of particular interest to us, especially those related to cancer and its determinants. For this analysis, we focus on variables such as STATE, which identifies the respondent's state of residence by the Federal Information Processing Standard (FIPS) state code, and CNCRDIF, which signifies the number of types of cancer the individual has had. Understanding the geographic distribution of cancer prevalence is critical for designing effective public health interventions.

One of the significant challenges in analyzing BRFSS data is accounting for the survey's complex design, particularly the weighting and stratification. The BRFSS data must be weighted to correct for the survey's unequal probabilities of selection and to make the data representative of the broader population. Stratification further complicates the analysis, as it requires careful consideration of how the data is segmented during the survey process.

When it comes to mapping and visualizing BRFSS data, another challenge arises—the need to appropriately “bin” the data. Binning involves categorizing continuous data into discrete intervals, which is essential for creating clear and interpretable maps. Effective binning can reveal patterns in the data that might otherwise be obscured, enabling researchers and policymakers to identify areas of concern and target interventions more precisely. Additionally, variation in survey questions if being analyzed over time may need to be harmonized to a single variable to ensure appropriate analysis.

This paper aims to explore these challenges and provide a comparative analysis of geocoding and mapping techniques in SAS® and R, specifically focusing on the CDC BRFSS dataset from 2023. By demonstrating how to geocode and visually map public health data, we hope to empower researchers and practitioners with the skills necessary to conduct spatial analyses and make data-driven decisions in public health.

GEOCODING & MAPPING

Geocoding

Geocoding is the process of converting addresses or other geographic identifiers into spatial coordinates, typically latitude and longitude. This transformation allows health data tied to specific locations—like residences, businesses, pharmacies, health providers, or community centers—to be placed on a map. The level of granularity of geocoding required for a specific research question can vary greatly, from latitude and longitude points to larger area estimates such as census block, zip code tabular area, county, state, or even country, and is dependent upon the level of specificity needed to answer the research question. The accuracy of geocoding can significantly impact the quality of subsequent spatial analyses, as it determines how precisely these locations are represented geographically. In general, the greater the granularity in geocoding estimates is the greater the probability for error in these estimates. This tool can illuminate such socioeconomic challenges such as food and/or pharmacy deserts.

Importantly, geocoding into latitude and longitude coordinates is not always required prior to mapping. Some packages, such as ggplot2 or plotly, can use native country and state names, abbreviations, or codes so long as they are formatted correctly in the dataframe. This paper provides examples of both cases.

Mapping

Mapping involves the visualization of geocoded data, often integrating multiple data layers to produce thematic maps. These thematic maps can display variations in health-related factors, such as disease incidence, access to healthcare facilities, environmental exposures, or social determinants of health across different geographic areas. Overall cancer prevalence is used as an example throughout this paper. By enabling visual assessment of these factors, mapping helps to identify spatial trends, clusters, or anomalies.

METHODS USING SAS®

SAS® PROCEDURES AND OTHER OPTIONS

SAS® has an amazing arsenal of tools to use and display geographic information that is relatively unknown and underutilized. We intend to highlight both new and existing capacities for creating stunning, informative maps as well as using geographic data in other ways. The two primary sources of mapping capacity in SAS are SG procedures (SGMAP and SGPLOT, as well as ATTRMAPs and annotate tools; and SAS/GRAPH, which evolved from SAS GIS many years ago.) Almost all of the tools described below

were originally provided to users through SAS/GRAPH; however, since SAS/GRAPH requires a separate license, many procedures and techniques were moved to BASE SAS so all users can access the geographic tools. As noted above, we will discuss custom mapping of geographic areas with base SAS techniques, as well as R, and compare and contrast.

There is an underutilized resource on the SAS website to help users make choices of how to use SAS for custom mapping and geocoding, SAS Maps Online. We strongly urge interested users to check this resource out (Figure 1), if nothing else to obtain geocode data and free code samples.

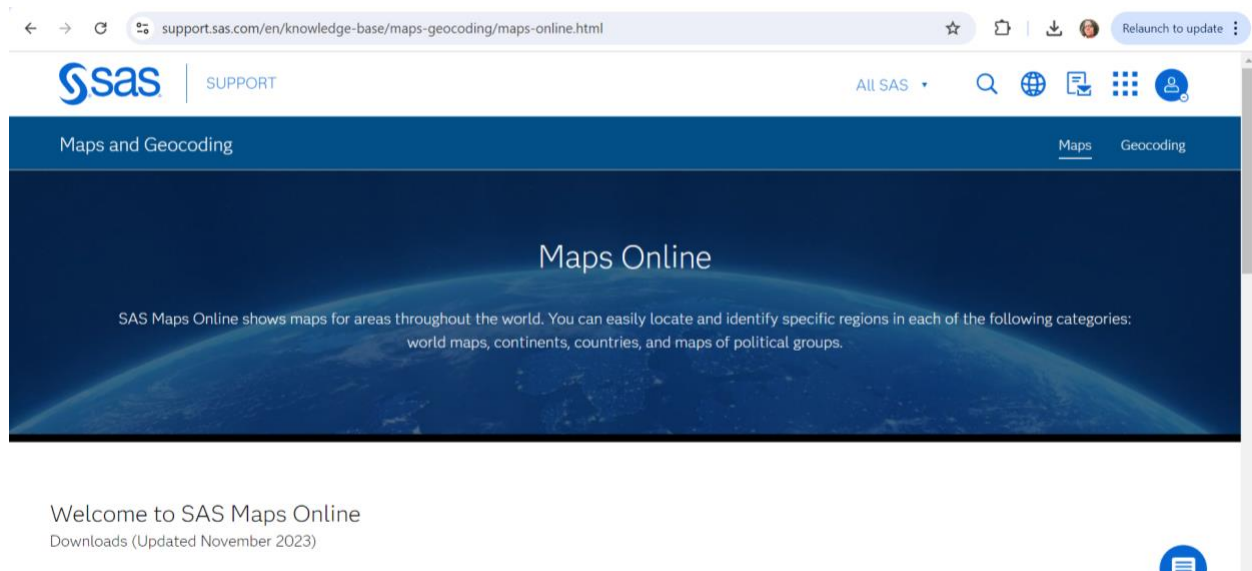


Figure 1. SAS Mapping and Geocoding Resources

The splash page of Maps Online is the Maps page, and this allows you to view and download (with an active SAS license) various map files and data sets which assist in visualizations.

Below, (Figure 2) we see the Geocoding page, which allows you access to geocode data compatible with SAS procedures such as street address geocoding files that do not automatically download with SAS installations.

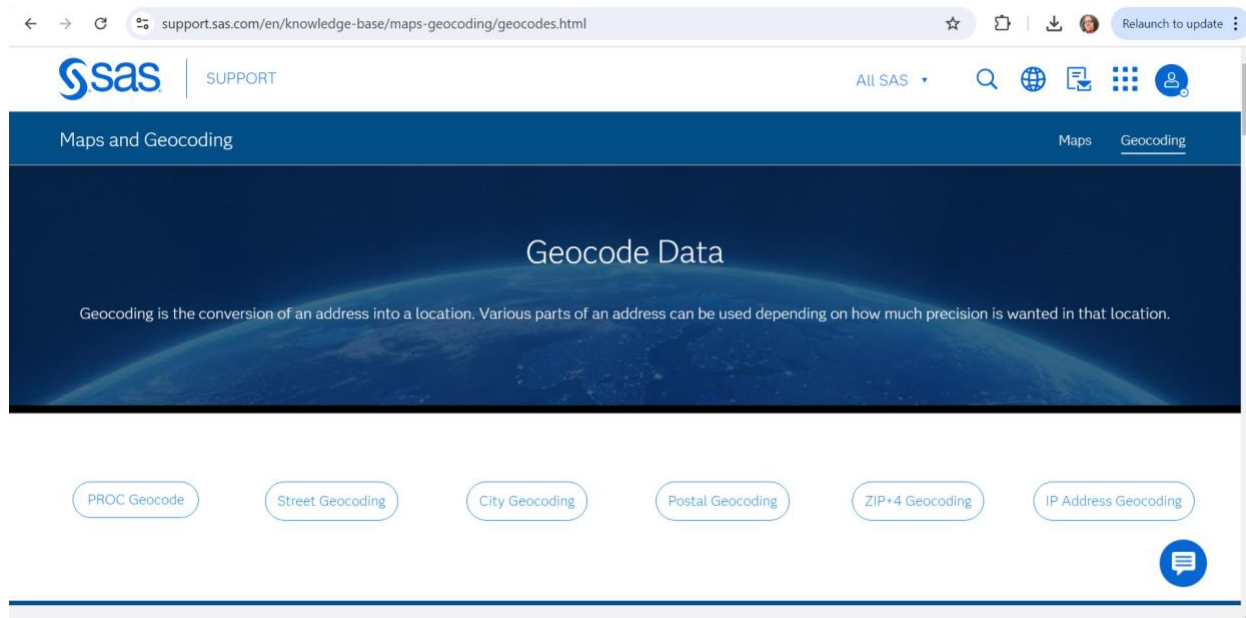


Figure 2. SAS Mapping and Geocoding Resources – Geocoding Resources

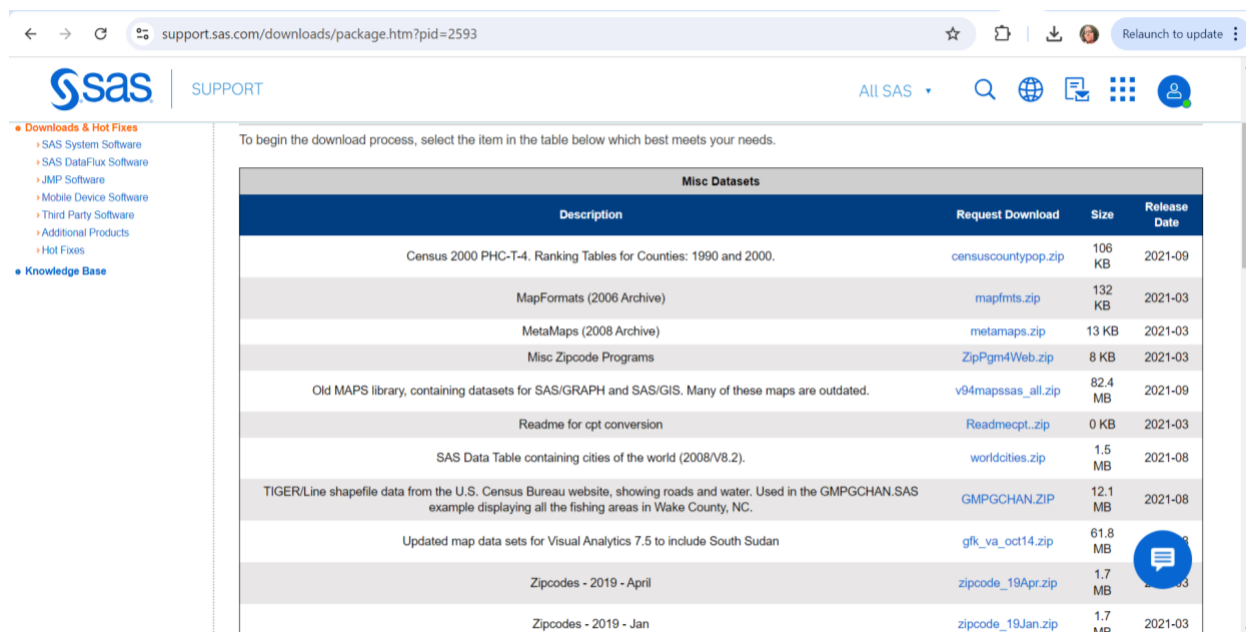


Figure 3. SAS Mapping and Geocoding Resources – Miscellaneous Data Sets

The above screenshot (Figure 3) shows some miscellaneous data sets, the most important of which are zipcode centroid files for every quarter since 2019. You must sign in with your SAS account credentials to access these files, including the geocoding data files.

A history of SAS Graphics

SAS introduced a version of ODS GRAPHICS, a graphics system which was template based, following soon after the Output Delivery System was released. In ODS GRAPHICS, many statistical procedures could produce one or more graphics, depending on options within the procedure, etc., simply by turning on the ODS GRAPHICS option. SAS 9.2 introduced the SG family of procedures. These procedures

assist users in the creation of effective graphics displaying information from statistical procedures, by using statements which invoke statistical procedures. As with ODS, SAS developed templates for each procedure and option which work seamlessly with the ODS style templates and destinations, and other SAS standards, creating attractive and meaningful graphics quickly and efficiently “out of the box”, with little effort by the programmer.

The initial SG procedures included PROC SGPLOT (single-celled graph, with many options), PROC SGSCATTER (paneled graphs with scatter plots), and PROC SGPANEL (classification panels). All these procedures support bygroup processing and paneling when applicable. The first MAPPING use of SG procedures was PROC SGPLOT with a Polygon statement (analogous to GGPLOT2), and then PROC SGMAP came into the picture.

As you might guess, there are both similarities and differences between the two systems. SG procedures are completely layout based, while SAS/GRAPH is device driven, which makes sense, since graphics were literally sent to devices. The second author's very first SAS paper created SAS/GRAPH maps that were sent to a very expensive color printer. Because the SG procedures are available to all users, SAS examples in the R & SAS comparison will be conducted with the SG procedures.

No setup is required to use the SURVEY, BASE SAS mapping and SG procedures in SAS. However, as with R, it is necessary to set up formats, color schemas, etc. in order to produce attractive maps. While R uses geocoding to establish a link between geographic areas on a map and in the data, SAS provides some special tools for state and county level mapping in the US. SAS provides geocoding capacity for finer areas down to congressional districts and street address coding, as well as other geographic entities.

A brief discussion of SAS procedures relating to the geographic presentations of data in this paper / presentation follows below.

PROC GEOCODE:

This procedure is used for **geocoding**—translating addresses into geographical coordinates (latitude and longitude). It supports various methods of geocoding, including street-level, ZIP code, city, and area-level geocoding. PROC GEOCODE is available with Base SAS:

```
PROC GEOCODE /* Invoke geocoding procedure */
  method=STREET /* Specify geocoding method */
  data=input /* Input data set of addresses */
  out=dd.GEOCODED /* Output data set with X/Y values */
  lookupstreet=street.usm /* Primary street lookup data set */
  type=SASHELP.GCTYPE; /* Lookup data set-added street type */
RUN;
```

PROC SGMAP:

Introduced in SAS 9.4M3, this procedure is part of the ODS Graphics procedures and is used for creating **modern, high-quality maps**. It supports various layers including scatter and series plots on maps, and it can integrate with PROC GEOCODE output for mapping geocoded data. There are many optional statements available for PROC SGMAP. There is a noticeable reduction in the lines of code necessary to generate a map. Like PROC GEOCODE and all SG procedures, PROC SGMAP is available with Base SAS:

```
ods graphics / border outputfmt=png;

data states;
  set mapssas.states;
  if state ^in(2,15,72);
  x = -x * 45/atan(1);
  y = y * 45/atan(1);
run;
```

```

data plot_data;
  set mapssas.uscenter;
  if state ^in(2,15,72) and ocean^='Y';
  long = -long;
  statename = fipstate(state);
run;

title 'Population from 2020 US Census';
proc sgmap mapdata=states
  maprespdata=sashelp.us_data
  plotdata=plot_data;
  esrimap

url='https://services.arcgis.com/arcgis/rest/services/Canvas/
World_Light_Gray_Base/MapServer';
choromap population_2020 / mapid=state density=2
name='choro';
text x=long y=lat text=statename /
textattrs=(size=6pt);
gradlegend 'choro' / title='2020 Population'
extractscale;
run;

ods graphics / reset;
title;

```

ASSOCIATING FIPS STATE CODES WITH GEOGRAPHIC DATA USING BRFS

By default, SAS uses a numeric state code and/or character state abbreviation (2 columns) that is consistent with FIPS state codes. Additionally, SAS provides built in functions or subroutines which align different state codes, character or numeric, with FIPS state codes, state abbreviations, and state names. SAS geographic functions related to states include: FIPNAME, FIPNAMEL, FIPSTATE, STNAME, STNAMEL. Thus, BRFS geographic data can easily be combined with SAS geographic files.

BASE MAPPING MATERIALS IN SAS

SAS also provides projected SAS state maps and state centroids for labeling. Generally, when creating SAS overlap maps with either PROC GMAP or PROC SGMAP, different “layers” are reprojected together using PROC GPROJECT to make sure all the coordinates mesh correctly. With PROC SGMAP, it may be possible to accomplish overlays without reprojecting.

SAS has the ability to create choropleth (heat) maps as well as importing ESRI background maps and OpenStreetMaps to use as backgrounds to map plots. Using PROC SGMAP, users can place data on maps with SCATTER, BUBBLE, TEXT and SERIES statements.

SAS map data sets can be downloaded from SAS MAPS ONLINE with a Base SAS license, or can be imported into SAS using PROC MAPIMPORT, which converts ESRI shape files into maps data sets with accompanying metadata.

Aside from PROC GMAP, PROC SGMAP, PROC GPROJECT, and PROC MAPIMPORT, other mapping related SAS procedures include: the GEOCODE procedure; the GINSIDE procedure; the GREduce procedure, and the GREMOVE procedure. Additionally, SAS provides the %CENTROID macro which calculated centroids for polygons. The %CENTROID macro is part of two suites of SAS geographic macros called %ANNOMAC for the “G” procedures, and %SGANNO for the “SG” procedures.

GEOCODING IN SAS

PROC GEOCODE has been available in SAS/GRAPH since Version 8.2, and recently became available in BASE SAS with a number of other tools in Version 9.4 Maintenance Release M5. SAS geocoding is

available at the city, zip, zip+4, street and custom levels. It produces x and y coordinates for matches between your data and the lookup table. It is important to note that SAS already provides geocoding and links for states, counties, zip codes, etc. so that it is not necessary to employ geocoding at the state level.

PREPARATION OF BRFSS DATA IN SAS

To create a binary for any cancer, a data step is used to transform the CNCRDIF variable. This variable has the values 1, 2, 3, 7, and 9, which correspond to 1 type of cancer, 2 types of cancer, 3+ types of cancer, don't know/not sure, and refused. Choices should be made regarding the 7 and 9 categories. Additionally, the question is asked with a skip pattern, so choices should be made about incorporating 0s from the originating variables (CHCSCNC1 and CHCOCNC1) as 0s in the CNCRDIF variable or binary:

```
data want;
  retain binary_for_cancer;
  set have;
  if CHCSCNC1 in(2) or CHCOCNC1 in(2) then binary_for_cancer=0;
  else if CNCRDIF in(1,2,3) then binary_for_cancer=1;
  /* any coding required for 7,9, and blank in the three input
variables */
run;
```

As noted above, BRFSS is a survey with a complex weighting scheme. In order to properly represent statistics on a map, procedures which take complex samples into account must be used. Therefore, BRFSS data must be pre-process to weight statistics. To summarize and categorize the data, SAS "Survey" procedures such as PROC SURVEYMEANS, PROC SURVEYFREQ, PROC SURVEYLOGISTIC, and PROC SURVEYREG must be used to produce weighted tabulations, means, and other statistics. Once weighted statistics are produced they can be mapped, or further pre-processed to bin, etc.:

```
proc surveymeans data=brfss_data mean stderr clm;
  strata _STSTR;      /* Specifies the strata */
  cluster _PSU;       /* Specifies the primary sampling units (PSUs) */
  weight _LLCPWT;     /* Specifies the survey weight variable */
  var binary_for_cancer; /* Specify the variable to transform */
run;
```

Once the binary is created, a map can be generated by "binning" the chosen statistic. One method we use frequently in preparing data for mapping is PROC RANK. PROC RANK will rank records in ascending order into a specified number of groups. For mapping, our research indicated that viewers can't really distinguish more than 5 different colors in a heat map, so we chose 5. There are some cons to PROC RANK – there may be issues with "tie-breaking" and your groups may not be equal in size. The blog below presents a number of different options for grouping observations in SAS, and offers some good ideas for visualizing "binned" data:

```
PROC RANK DATA = have
  GROUPS = 5
  OUT = ranks
  (KEEP=state statistic: );
VAR binary_for_cancer_stat;
RANKS statistic_5 ;
RUN;
```

The "binned" and properly weighted statistic we've created can be used to generate a choropleth map similar to what can be created in R (see examples Figure 4 and Figure 5 in the sections below):

```
ods graphics / border outputfmt=png;
title 'Population from 2010 US Census';
proc sgmap mapdata=states maprespdata=sashelp.us_data plotdata=plot_data;
  esrimap
  url='https://services.arcgis.com/arcgis/rest/services/Canvas/
```

```
World_Light_Gray_Base/MapServer';
choromap population_2010 / mapid=state density=2 name='choro';
text x=long y=lat text=statename / textattrs=(size=6pt);
gradlegend 'choro' / title='2010 Population' extractscale;
run;

ods graphics / reset;
title;
```

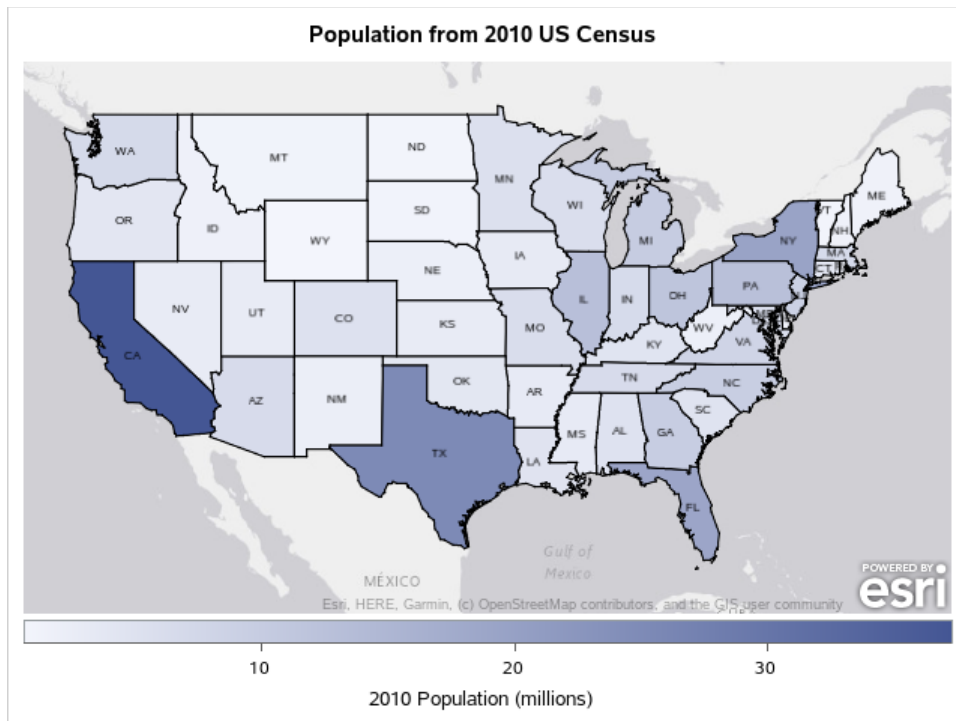


Figure 4. A number of different options for grouping observations in SAS, and offers some good ideas for visualizing “binned” data.

We can add Hawaii and Alaska, and insert “hot links” as well:

```
ods graphics / reset;
title;

ods graphics /
  imagemap
  width=500px height=375px
  drilltarget="_blank";

title1 h=18pt "Click a State to Find Its Home Page";

data my_data;
set sashelp.us_data;
length my_link $100;
my_link='https://www.google.com/search?q=state+page+for+
'||trim(left(statename));
run;

proc sgmap maprespdata=my_data mapdata=mapsgfk.us noautolegend;
```

```

choromap population_2010 /
    mapid=statecode
    id=statecode
    url=my_link
    tip=(statename population_2010)
    tiplabel=("State" "Population");

run;

ods graphics / reset;
title;

```

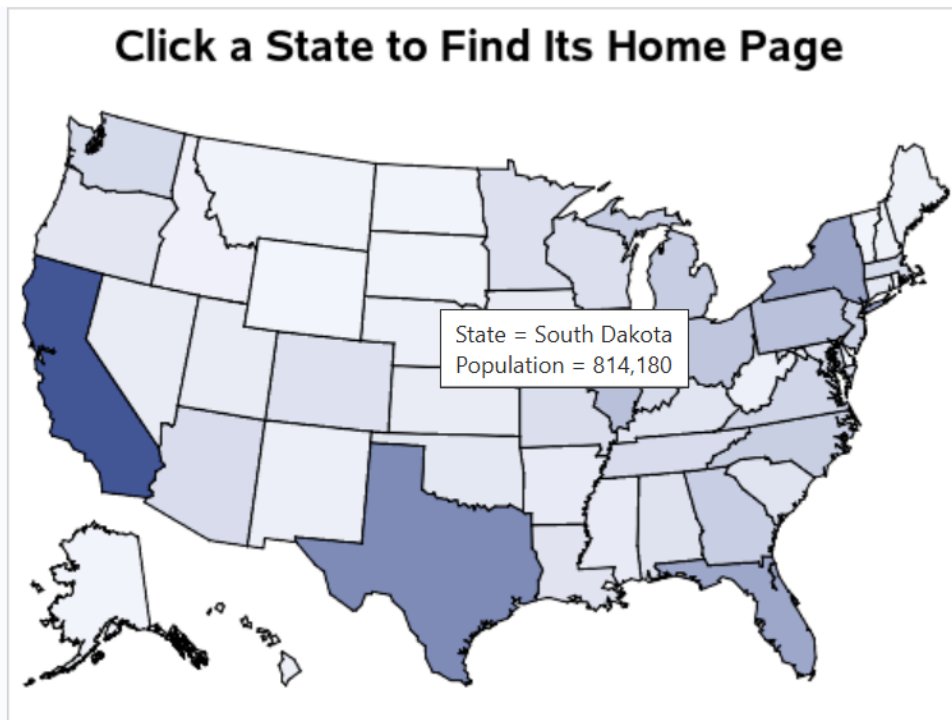


Figure 5. A number of different options for grouping observations in SAS, and offers some good ideas for visualizing “binned” data.

There are also many ways to enhance and annotate SAS maps, including bubble plots, etc. Below (Figure 6) follows code to create a Massachusetts county map with plots of binned address points. Although this map does not have shading to indicate binned statistics, it could, and layer the address points on as well:

```

ods graphics / noscale imagefmt=png imagename="&name"
width=1200px height=900px noborder;

proc sgmap mapdata=my_map (drop = lat long) maprespdata=my_counties
    plotdata=my_data noautolegend;
    choromap counter / mapid=id lineattrs=(color=black);
    bubble x=x1 y=y1 size=or1 / fillattrs=(color=red) group=or1 transparency=.8
    bradiusmin=3px bradiusmax=11px ;
    bubble x=x2 y=y2 size=or2 / fillattrs=(color=lightorange) group=or2
    transparency=.8 bradiusmin=4px bradiusmax=12px ;
    bubble x=x3 y=y3 size=or3 / fillattrs=(color=lightyellow) group=or3
    transparency=.8 bradiusmin=5px bradiusmax=13px ;

```

```

bubble x=x4 y=y4 size=or4 / fillattrs=(color=lightgreen) group=or4
transparency=.8 bradiusmin=6px bradiusmax=14px ;
bubble x=x5 y=y5 size=or5 / fillattrs=(color=lightblue) group=or5
transparency=.8 bradiusmin=7px bradiusmax=15px ;
run;

ods graphics / reset;
title;

```

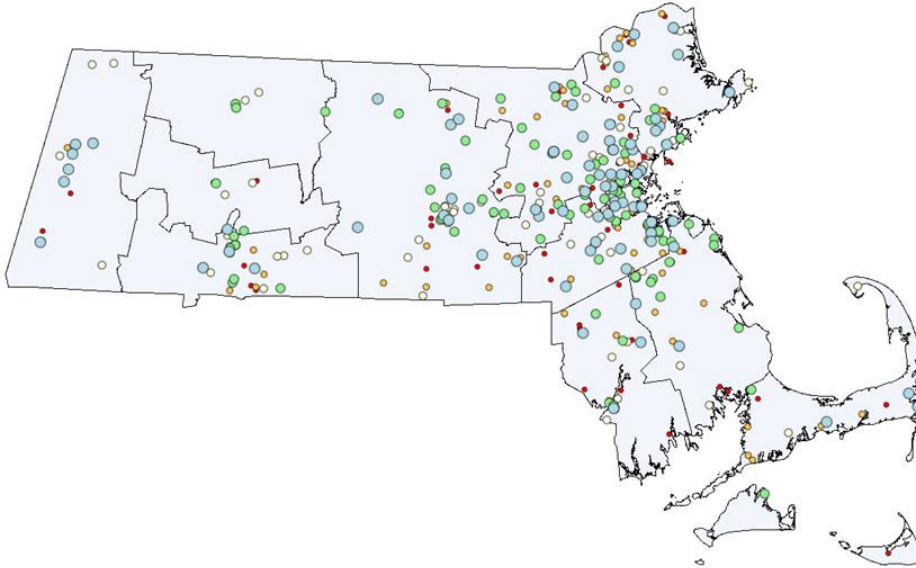


Figure 6. Massachusetts county map with plots of binned address points.

OPEN-SOURCE METHODS USING R

R offers a robust, flexible, and analogous set of tools for geocoding and mapping, which are increasingly popular in the fields of data science and public health. This section highlights both established and cutting-edge packages in R that empower users to create detailed, informative maps and perform complex geographic analyses. The versatility of R's geospatial capabilities makes it a valuable resource for researchers and practitioners alike, providing a wide range of options for visualizing and analyzing spatial data.

The primary package for geocoding in R that we'll be focusing on is `tidygeocoder`. This package allows users to easily convert addresses into geographic coordinates and vice versa, integrating seamlessly with the tidyverse suite of packages. With `tidygeocoder`, users can access multiple geocoding services, ensuring flexibility and reliability in obtaining accurate geographic data. This package exemplifies R's strength in handling spatial data efficiently and with minimal setup.

For mapping, `ggplot2` serves as the standard in R, renowned for its elegance and ease of use in creating visually appealing plots. `ggplot2` offers a solid foundation for building static maps, providing extensive customization options through its layering system. This package is particularly powerful for producing publication-quality maps that clearly convey complex data. Building on `ggplot2`, the `ggmap` package extends its functionality by integrating with Google Maps and other map services, enabling users to overlay data on various types of basemaps. `ggmap` enhances the visual appeal and context of maps, allowing for more detailed and informative geographic visualizations. `ggmap` is deserving of its own paper and thus will not be covered here.

In addition to these tools, the `mapsf` package introduces a straightforward approach to creating static maps, particularly useful for displaying administrative boundaries and thematic mapping. `mapsf` is designed to be user-friendly, offering a simple yet effective way to produce maps that communicate geographic information clearly.

For those interested in creating interactive maps, `plotly` offers a powerful solution. `plotly` allows users to build interactive maps that can be explored and manipulated by the viewer, providing an engaging way to present spatial data. This capability is particularly useful for web-based applications and dashboards, where user interaction with the map adds significant value.

As with SAS, R's geocoding and mapping capabilities require some setup, such as establishing appropriate formats and color schemes to produce aesthetically pleasing maps. However, the flexibility and range of tools available in R provide users with the ability to tailor their maps to specific needs, whether they are producing static visualizations for reports or interactive maps for web applications.

In this section, we will explore how to implement these geocoding and mapping techniques in R, providing a direct comparison with the tools available in SAS. Through this comparison, we aim to equip users with the knowledge to choose the most appropriate tools for their geographic analyses, whether they are working in SAS or R.

HANDLING BRFSS SURVEY DATA

Before we get started, it is important to properly format the BRFSS data as a survey object within the R environment, with the aforementioned survey design using the primary sampling unit, stratum weight, and design weight used for raking. This is accomplished using the tidyverse group of packages for data wrangling, as well as the `survey` package for analyzing complex survey data. The `here` package is utilized to ease the navigation of subdirectories, by using the top-level directory of a project to easily build paths to files:

```
# Load necessary libraries
library(here)
library(tidyverse)
library(survey)

# Reading in the 2022 BRFSS data
data_2022 <- readRDS(here("temp_data", "2022_data.rds"))
str(data_2022)
colnames(data_2022)

# Remove underscores from all variable names as they may cause issues in
subsequent processing
data_2022 <- data_2022 %>%
  rename_with(~str_remove(., "_"), starts_with("_"))

data_2022 <- data_2022 %>%
  select(STATE, STSTR, PSU, WT2RAKE, CNCRDIF)

print (data_2022)
```

```
# A tibble: 445,132 × 5
  STATE STSTR      PSU WT2RAKE CNCRDIF
<dbl> <dbl>    <dbl>   <dbl>   <dbl>
1     1 11011 2022000001    74.8     NA
2     1 11011 2022000002    37.4     NA
3     1 11011 2022000003    37.4     NA
4     1 11011 2022000004   112.     NA
5     1 11011 2022000005    74.8     NA
6     1 11011 2022000006    37.4     NA
```

7	1	11011	2022000007	37.4	NA
8	1	11011	2022000008	37.4	NA
9	1	11011	2022000009	74.8	NA
10	1	11011	2022000010	74.8	NA

i 445,122 more rows
i Use `print(n = ...)` to see more rows

```
# Create a lookup table for state FIPS codes based on the 2022 codebook
state_fips <- tibble(
  "STATE" = c(1, 2, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39,
40, 41, 42, 44, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56, 66, 72, 78),
  state_name = as.factor(c("Alabama", "Alaska", "Arizona", "Arkansas",
"California", "Colorado", "Connecticut", "Delaware", "District of
Columbia", "Florida", "Georgia", "Hawaii", "Idaho", "Illinois", "Indiana",
"Iowa", "Kansas", "Kentucky", "Louisiana", "Maine", "Maryland",
"Massachusetts", "Michigan", "Minnesota", "Mississippi", "Missouri",
"Montana", "Nebraska", "Nevada", "New Hampshire", "New Jersey", "New
Mexico", "New York", "North Carolina", "North Dakota", "Ohio", "Oklahoma",
"Oregon", "Pennsylvania", "Rhode Island", "South Carolina", "South Dakota",
"Tennessee", "Texas", "Utah", "Vermont", "Virginia", "Washington", "West
Virginia", "Wisconsin", "Wyoming", "Guam", "Puerto Rico", "Virgin
Islands"))
)

# Join BRFSS data with the state FIPS lookup table to add state names
data_2022 <- data_2022 %>%
  left_join(state_fips, by = "STATE")

# Verify the levels of the state factor to ensure correct matching
levels(data_2022$state_name)

# Create a survey design object accounting for the complex survey design
data_2022_survey <- svydesign(
  id = ~PSU,                # Primary Sampling Units (PSUs)
  strata = ~STSTR,          # Stratification variable
  weights = ~WT2RAKE,       # Weights variable
  data = data_2022,         # Data set
  nest = TRUE               # Indicates nesting of the design
)

# Recode the CNCRDIF variable as a factor with descriptive labels for each
level
data_2022$CNCRDIF <- factor(data_2022$CNCRDIF, levels = c(1, 2, 3, 7, 9),
  labels = c("One", "Two", "Three or more",
"Don't know/Not Sure", "Refused"))

# Set survey options to adjust for lonely PSUs, which may have a single
observation
options(survey.lonely.psu = "adjust")

# Recreate the survey design object with adjusted survey options
data_2022_survey_adjusted <- svydesign(
  id = ~PSU,
```

```

    strata = ~STSTR,
    weights = ~WT2RAKE,
    data = data_2022,
    nest = TRUE
  )

# Calculate weighted COUNTS for the cancer type variable (CNCRDIF) by
state
cancer_counts_by_state <- svyby(
  ~CNCRDIF,                # Outcome variable
  ~STATE,                  # Grouping variable (State)
  data_2022_survey_adjusted, # Survey design object
  svytotal,                # Function to compute weighted totals
  na.rm = TRUE             # Remove missing values
)

# Reshape data for easier merging
cancer_counts_long <- cancer_counts_by_state %>%
  pivot_longer(cols = starts_with("CNCRDIF"), names_to = "Cancer_Type",
values_to = "Count") %>%
  mutate(Type = case_when(
    str_detect(Cancer_Type, "One") ~ "One",
    str_detect(Cancer_Type, "Two") ~ "Two",
    str_detect(Cancer_Type, "Three.or.more") ~ "Three or more",
    TRUE ~ NA_character_
  )) %>%
  filter(!is.na(Type)) %>%
  select(-Cancer_Type)

# Join BRFSS data with the state FIPS lookup table to add state names
cancer_counts_long <- cancer_counts_long %>%
  left_join(state_fips, by = "STATE")

# Calculate the "At least one" count per state by binning counts for "One,"
"Two," and "Three or more"
at_least_one_counts <- cancer_counts_long %>%
  group_by(STATE) %>%
  summarise(AtLeastOneCount = sum(Count, na.rm = TRUE))

# Calculate the total number of "At least one" cases in the U.S.
total_us_cases <- sum(at_least_one_counts$AtLeastOneCount, na.rm = TRUE)

# Calculate the proportion of "At least one" cases for each state relative
to the U.S. total
at_least_one_counts <- at_least_one_counts %>%
  mutate(USPercent = (AtLeastOneCount / total_us_cases) * 100)

sum(at_least_one_counts$USPercent) # Should be 100

# Merge the "At least one" data with the state FIPS lookup for full state
names
at_least_one_counts <- at_least_one_counts %>%
  left_join(state_fips, by = c("STATE" = "STATE")) %>%
  select(1, 4, 2, 3)

print(at_least_one_counts)

```

```
# A tibble: 54 × 4
  STATE state_name      AtLeastOneCount USPercent
  <dbl> <fct>          <dbl>      <dbl>
1     1 Alabama             0          0
2     2 Alaska              0          0
3     4 Arizona             0          0
4     5 Arkansas          21449.        2.62
5     6 California          0          0
6     8 Colorado          61051.        7.47
7     9 Connecticut       23741.        2.90
8    10 Delaware           2926.        0.358
9    11 District of ...      0          0
10   12 Florida             0          0
# i 44 more rows
# i Use `print(n = ...)` to see more rows
```

Before mapping, it is important to note that although 53 states and U.S. territories were surveyed, some states do not have any survey data with cancer cases. In other words, a 0% cancer rate does not necessarily mean that there are no cases of cancer within that state, that we are visualizing a shortcoming of the survey implementation.

MAPPING WITH GGLOT2

With `ggplot2`, state and country names can be used natively and thus geocoding is not required. However, if you need to geocode, `tidygeocoder` is the go-to package in R. The `tidygeocoder` package is a powerful tool for converting addresses or place names into geographic coordinates (latitude and longitude). It supports multiple geocoding services, including OpenStreetMap (OSM), which is used in the example code below. This package allows for both forward geocoding (converting place names to coordinates) and reverse geocoding (converting coordinates to place names):

```
# Load necessary libraries
library(here)
library(tidygeocoder)
library(tidyverse)

# Forward geocode the state names using OpenStreetMap (OSM)
geocoded_states <- state_fips %>%
  geocode(
    address = state_name,      # Column containing the state names
    method = 'osm',           # Geocoding service ('osm' for OpenStreetMap)
    full_results = FALSE      # Only return basic results (latitude and
                                longitude)
  )

# Print the geocoded results
print(geocoded_states)
```

```
# A tibble: 54 × 4
  STATE state_name      lat   long
  <dbl> <fct>          <dbl> <dbl>
1     1 Alabama       33.3  -86.8
2     2 Alaska        64.4 -150.
3     4 Arizona       34.4 -112.
```

```

4      5 Arkansas      35.2 -92.4
5      6 California    36.7 -119.
6      8 Colorado      38.7 -106.
7      9 Connecticut   41.7 -72.7
8     10 Delaware      38.7 -75.4
9     11 District of Columbia 38.9 -77.0
10    12 Florida       27.8 -81.5
# i 44 more rows
# i Use `print(n = ...)` to see more rows

```

```

# Example data frame with lat/long coordinates
coords <- data.frame(
  lat = c(34.0522, 40.7128, 37.7749),
  long = c(-118.2437, -74.0060, -122.4194)
)

# Reverse geocode the coordinates to obtain place names
reverse_geocoded <- coords %>%
  reverse_geocode(
    lat = lat,      # Latitude column
    long = long,    # Longitude column
    method = 'osm'  # Reverse geocoding service (OSM)
  )

# Print the reverse geocoded results
print(reverse_geocoded)

```

```

# A tibble: 3 × 3
  lat   long address
<dbl> <dbl> <chr>
1  34.1 -118. Los Angeles Police Department Headquarters, South M...
2  40.7  -74.0 New York City Hall, 260, Broadway, Lower Manhattan,...
3  37.8 -122. South Van Ness Avenue, Hayes Valley, San Francisco,...

```

This mapping method will be convenient for those who are familiar with the `ggplot2` package syntax even if this is your first time dealing with geographical or spatial data. In this example, we utilized the state names (`state_name`) for the purpose of mapping. We start by visualizing the data using a basic bar plot to show how hard the data is to interpret. Then, we build a basic map using `ggplot2`, slowly building up to a publication-worthy map with a custom palette, state abbreviations, and enhanced labeling for the axes. The `maps` package is used to obtain polygon data for the continental United States, while the `usmap` package is used to obtain polygon data for the entire United States (including Hawaii and Alaska). `PNWColors` is used to build a custom Pacific Northwest color palette for the purposes of this paper. The `sf` package is used to create and map geospatial vectors. Without getting into mapping, `ggplot2` is frequently used for other visualizations such as bar plots (Figure 7). Instead, geospatial data may be easier to read using a map (Figure 8). Customizations can be made to make visualizations easier to read and more interpretable (Figure 9). A final map that fits all our needs can be found in Figure 10.

```

# Load necessary libraries
library(tidyverse)
library(maps)
library(usmap)
library(PNWColors)
library(sf)

# Basic bar plot

```

```
ggplot(at_least_one_counts, aes(x = state_name, y = USPercent)) +
  geom_bar(stat = "identity", fill = "#9DBF9E") +
  labs(title = "US Percent of At Least One Count by State",
       x = "State",
       y = "US Percent") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

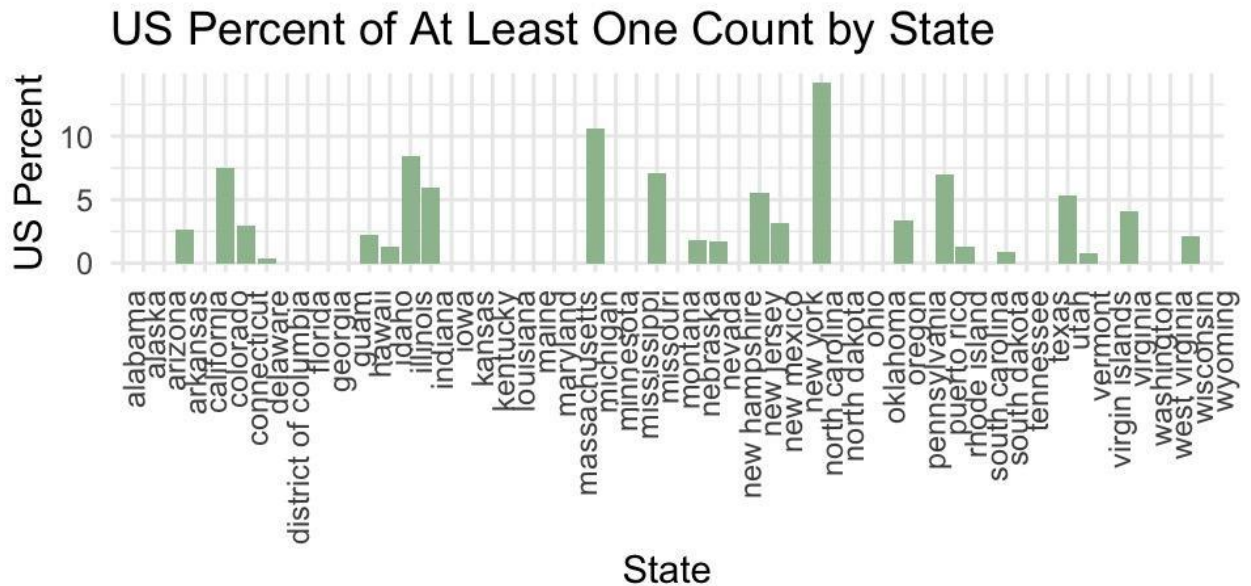


Figure 7. Basic bar plot using ggplot2.

```
# Issue - hard to read, not very informative
```

```
# Enter: Mapping
# Get the map data for the U.S.
states_map <- map_data("state")

# Ensure your state names are lowercase to match with the map data
at_least_one_counts <- at_least_one_counts %>%
  mutate(state_name = tolower(as.character(state_name)))

# Join the map data with your data
map_data <- states_map %>%
  left_join(at_least_one_counts, by = c("region" = "state_name"))

# Custom color palette for the Pacific Northwest
palette <- pnw_palette(name="Shuksan2",n=4,type="continuous")

ggplot(map_data, aes(x = long, y = lat, group = group, fill = USPercent)) +
  geom_polygon(color = "white") +
  scale_fill_gradientn(colors = palette, na.value = "grey50") +
  labs(title = "US Percent of At Least One Count by State",
       fill = "US Percent") +
  theme_minimal() +
  theme(axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
```

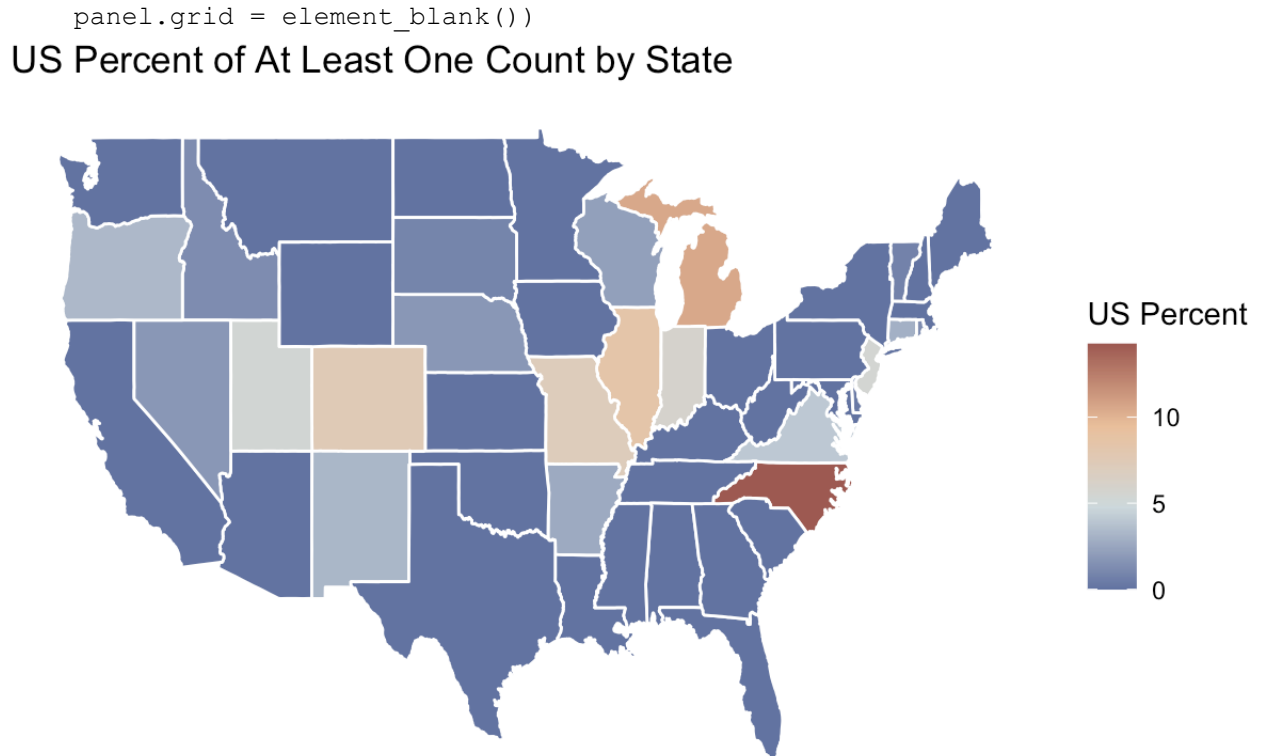


Figure 8. Basic map using ggplot2.

```
# Issue - the "maps" package does not include non-continental U.S. (Hawaii,
# Issue - hard to read 0 as blue
# Use `usmap`'s state data to create a map of the U.S. with Hawaii and
# Alaska
us_data <- us_map()

# Ensure your state names are lowercase to match with the map data
us_data <- us_data %>%
  mutate(full = tolower(as.character(full)))

# Join with your data
us_data <- us_data %>%
  left_join(at_least_one_counts, by = c("full" = "state_name"))

# Define the custom palette with grey for zero
palette_grey <- c("grey50", pnw_palette(name = "Shuksan2", n = 4, type =
"continuous"))

# Plot the basic map using ggplot2 and usmap with the custom palette
us_plot <- plot_usmap(data = us_data, values = "USPercent", regions =
"states") +
  scale_fill_gradientn(
    colors = custom_palette,
    values = scales::rescale(c(0, 0.25, 0.5, 0.75, 1)), # Adjust positions
of the colors
    na.value = "grey50",
    name = "US Percent"
```

```

) +
theme_void() +
labs(title = "US Percent of At Least One Count by State") +
theme(legend.position = "right")

# Display the final map
print(us_plot)

```

US Percent of At Least One Count by State

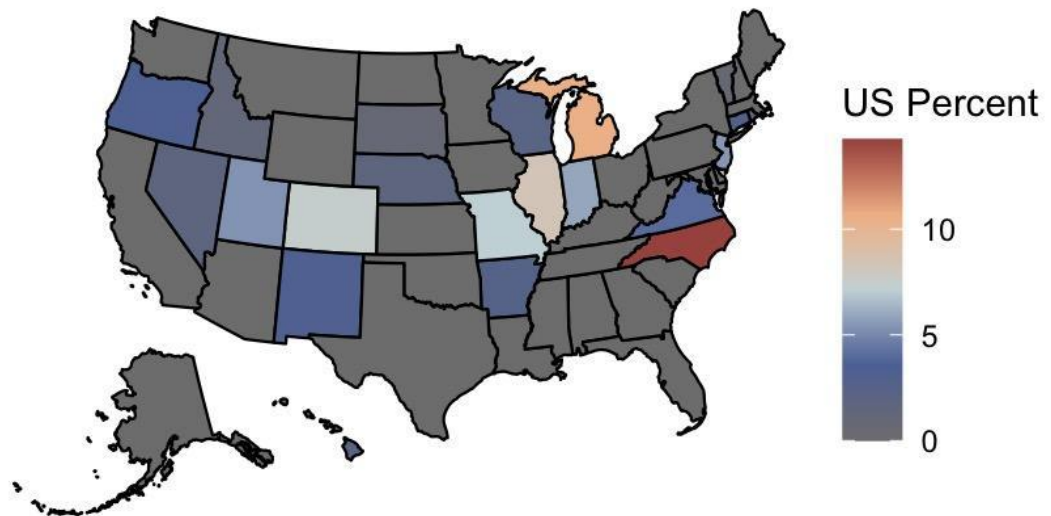


Figure 9. Improved map using ggplot2.

```

# Issue - which states are which?
# Issue - axes, data source, customization

# Calculate the centroids of the states
us_data_centroids <- st_centroid(us_data)

# Extract the coordinates from the centroids
us_data_centroids <- cbind(us_data_centroids,
st_coordinates(us_data_centroids))

# Plot the map with state abbreviations and a footnote
us_plot <- plot_usmap(data = us_data, values = "USPercent", regions =
"states") +
  scale_fill_gradientn(
    colors = custom_palette,
    values = scales::rescale(c(0, 0.25, 0.5, 0.75, 1)), # Adjust positions
of the colors
    na.value = "grey50",
    name = "Percent of U.S. Cases (%)"
  ) +
  geom_text(
    data = us_data_centroids,
    aes(x = X, y = Y, label = abbr),
    color = "black",
    size = 3,

```

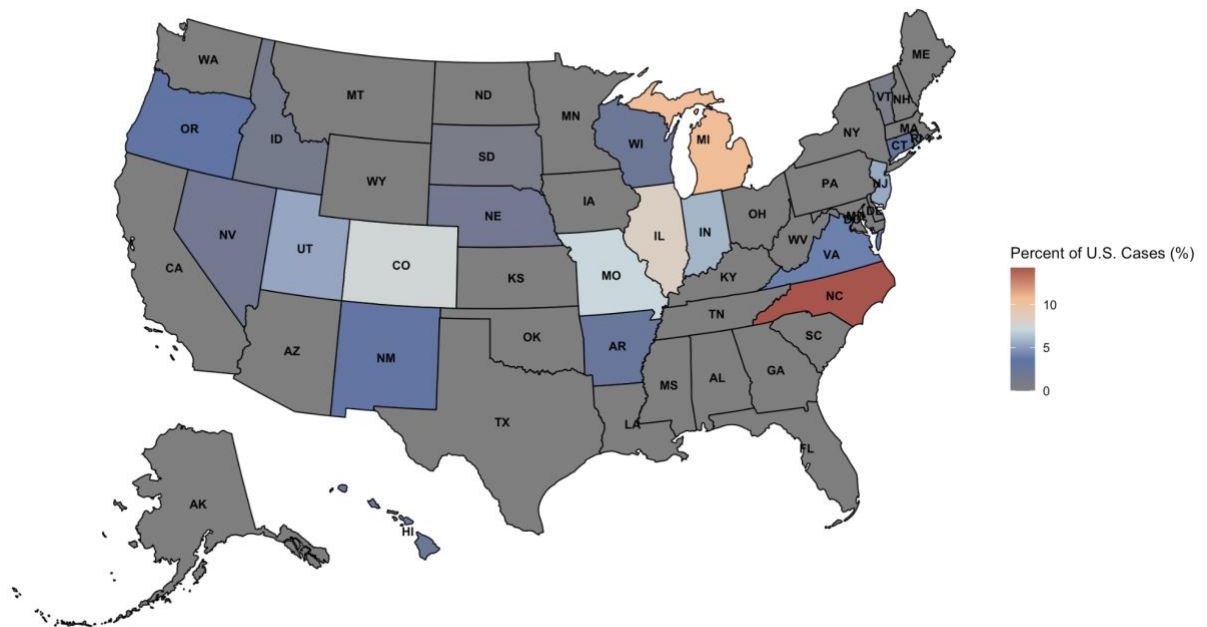
```

    fontface = "bold"
  ) +
  theme_void() +
  labs(
    title = "Which States Have the Highest Percentage of Cancer Cases?",
    caption = "Source: CDC BRFSS Data, 2022. Data represents the percentage
of at least one cancer case by state compared to the entire U.S."
  ) +
  theme(legend.position = "right")

# Display the final map
print(us_plot)

```

Which States Have the Highest Percentage of Cancer Cases?



Source: CDC BRFSS Data, 2022. Data represents the percentage of at least one cancer case by state compared to the entire U.S.

Figure 10. An advanced map with ggplot2.

MAPPING WITH MAPSF

The `mapsf` package offers a powerful alternative to the `ggplot2` approach for creating thematic maps, with a focus on providing extensive customization options specifically tailored for geographic visualizations. Unlike `ggplot2`, which is a general-purpose plotting tool, `mapsf` is entirely dedicated to thematic mapping, allowing users to create detailed and visually compelling maps with minimal effort. The package includes built-in functions for creating various types of thematic maps, such as choropleth, proportional symbols, and typology maps, while also offering fine-grained control over map elements like titles, legends, and labels. Additionally, `mapsf` supports advanced cartographic features, such as easy-to-use functions for adding credits, north arrows, and scale bars, making it an excellent choice for users looking to produce professional-quality maps that go beyond the basic capabilities of `ggplot2`. With `mapsf`, users can focus on the cartographic aspects of their visualizations, leveraging a suite of tools designed to enhance the aesthetic and functional quality of thematic maps. Importantly, the syntax is not a huge learning curve as with `ggplot2` (Figure 11):

```

# Load necessary libraries
library(sf)

```

```

library(mapsf)
library(PNWColors)

# set a theme
mf_theme("agolalight")

# Start the mapsf plot with base map
mf_map(
  x = us_data,
  var = "USPercent",
  type = "choro", # Choropleth map
  pal = custom_palette, # Custom palette
  border = "white", # State borders
  lwd = 0.5, # Line width for borders
  leg_horiz = TRUE, # Horizontal legend
  leg_title = "Percent of U.S. Cases (%)", # Legend title
  leg_val_rnd = 2, # Rounding for legend values
  add = FALSE
)

# Add state abbreviations
mf_label(
  x = us_data_centroids,
  var = paste(c("abbr", n/US_Percent))
  col = "black",
  cex = 0.5,
  font = 2
)

# Add a title with flair
# Add a title without color adjustment
mf_title(
  txt = "Which States Have the Highest Percentage of Cancer Cases?",
  pos = "left",
  cex = 1.3,
  font = 3,
)

# Add a footnote/caption
mf_credits(
  txt = "Source: CDC BRFSS Data, 2022. Data represents the percentage of at
least one cancer case by state compared to the entire U.S.",
  pos = "bottomleft",
  cex = 0.6,
  col = "black"
)

```

Which States Have the Highest Percentage of Cancer Cases?

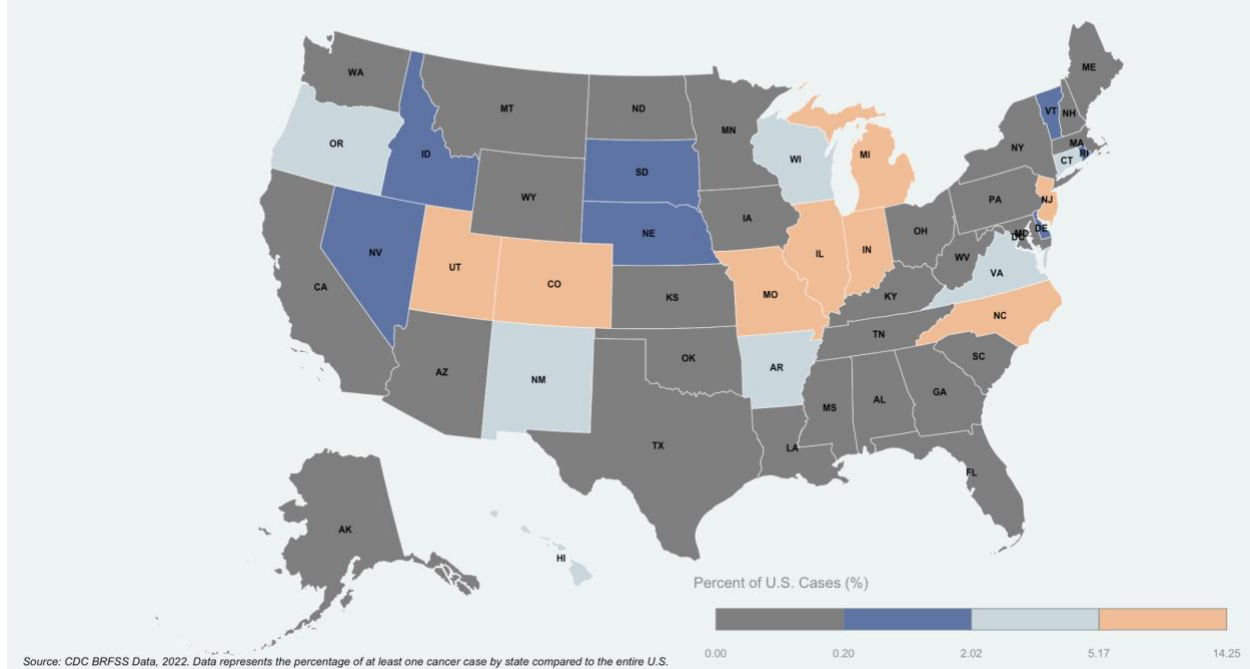


Figure 11. An introduction to mapping with mapsf.

INTERACTIVE MAPPING WITH PLOTLY

The `plotly` package offers a range of interactive features that enhance the user experience compared to static visualizations created with `ggplot2` or `maps`. While `ggplot2` and `maps` are powerful tools for creating detailed and customizable static maps, `plotly` transforms these maps into interactive experiences. It is worth mentioning that addon packages, such as `ggiraph` also extend `ggplot2` with interactive features. With `plotly`, users can hover over different map regions to view detailed tooltips, zoom in and out for closer examination, and pan across the map to explore different areas. Additionally, `plotly` supports dynamic legends, clickable elements, and responsive design, allowing users to engage with the data in real-time. This interactivity makes `plotly` particularly useful for presentations, dashboards, and web applications where users need to explore data more deeply and intuitively than static maps typically allow (Figure 12):

```
library(plotly)
library(sf)
library(tidyverse)

# Convert the geometry to a format compatible with plotly
us_data <- us_data %>%
  mutate(geometry = st_as_text(st_geometry(.))) %>%
  st_drop_geometry()

us_data <- us_data %>%
  mutate(hover = paste0(
    abbr, "<br>",
    "Percent of U.S. Cases: ", round(USPercent, 2), "%"
  ))

graph_properties <- list(
  scope = 'usa',
  projection = list(type = 'albers usa'),
```

```

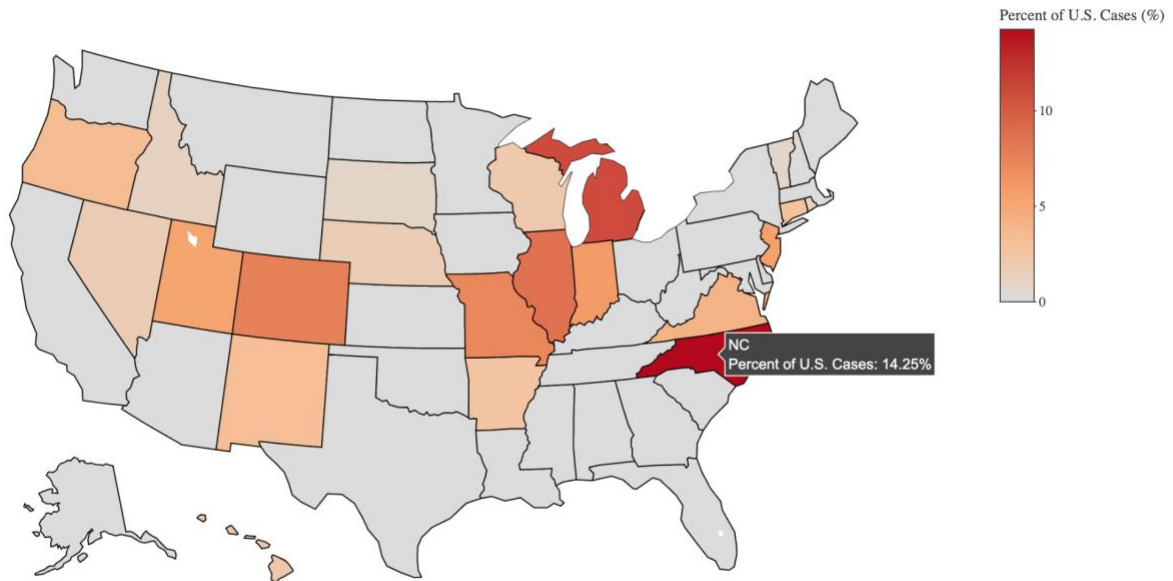
    showlakes = TRUE,
    lakecolor = toRGB('white')
  )

# Create the interactive map
interactive_map <- plot_geo(us_data, locationmode = "USA-states") %>%
  add_trace(
    locations = ~abbr,          # Use the abbr column for state
    codes
    z = ~USPercent,
    zmin = 0,
    zmax = max(us_data$USPercent, na.rm = TRUE),
    color = ~USPercent,
    colorscale = palette,      # Use custom color scale
    text = ~hover,
    hoverinfo = 'text'
  ) %>%
  layout(
    geo = graph_properties,
    title = list(
      text = "Which States Have the Highest Percentage of Cancer Cases?",
      font = list(family = "DM Sans", size = 24) # Increase title font
    size
  ),
  font = list(family = "DM Sans"),
  annotations = list(
    list(
      x = 0.5,
      y = -0.15, # Adjusted value to bring annotation closer to the plot
      text = "Source: CDC BRFSS Data, 2022. Data represents the
percentage of at least one cancer case by state compared to the entire
U.S.",
      showarrow = FALSE,
      xref = "paper",
      yref = "paper",
      xanchor = "center",
      yanchor = "top",
      font = list(size = 12)
    )
  ),
  margin = list(t = 100, b = 100) # Increase bottom margin
) %>%
config(displayModeBar = FALSE) %>%
colorbar(
  title = "Percent of U.S. Cases (%)",          # Rename legend title
  tickprefix = ''
)

# Display the interactive map
interactive_map

```

Which States Have the Highest Percentage of Cancer Cases?



Source: CDC BRFSS Data, 2022. Data represents the percentage of at least one cancer case by state compared to the entire U.S.

Figure 12. An introduction to interactive maps with plotly.

RESOURCES AVAILABLE TO BOTH SAS AND R

colorbrewer2/RColorBrewer for color selection (Figure 13):

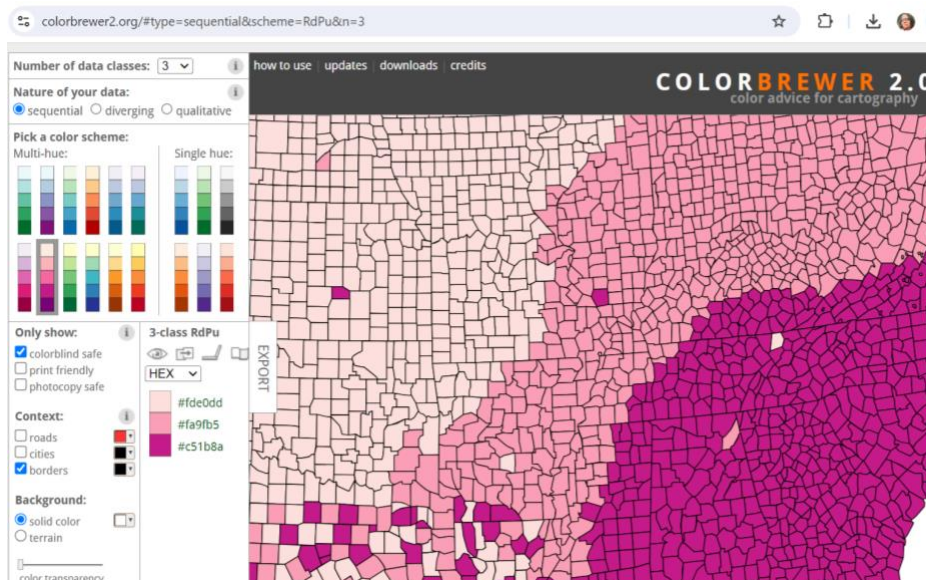


Figure 13. The colorbrewer2 interface.

CONCLUSION

This paper has highlighted the importance of geospatial analysis in public health and epidemiology, particularly through the use of the CDC's Behavioral Risk Factor Surveillance System (BRFSS) data. We have provided a detailed comparison of geocoding and mapping techniques using both SAS® and R, demonstrating how these tools can be leveraged to generate valuable geographic insights from public health data. It is important to note that both SAS® and R are constantly evolving and improving visualization capability, so it is important to keep up with developments in whichever package is being used. In SAS®, the modern geocoding and mapping procedures to keep up with include PROC GEOCODE and PROC SGMAP. In R, the packages to watch are ggplot2, mapsf, and plotly. Other packages that may be the focus of a future paper include ggmap, ggiraph, and tmap.

REFERENCES

- Centers for Disease Control and Prevention (CDC). 2022. *Behavioral Risk Factor Surveillance System Survey Data*. U.S. Department of Health and Human Services, Centers for Disease Control and Prevention. Available at: <https://www.cdc.gov/brfss/index.html>
- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Müller, K. (2020). *here: A Simpler Way to Find Your Files*. R package version 1.0.1. URL <https://CRAN.R-project.org/package=here>.
- Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., & Yutani, H. (2019). "Welcome to the tidyverse." *Journal of Open Source Software*, 4(43), 1686. doi:10.21105/joss.01686.
- Lumley, T. (2024). *survey: analysis of complex survey samples*. R package version 4.4.
- Cambon, J., Hernangómez, D., Belanger, C., & Possenriede, D. (2021). "tidygeocoder: An R package for geocoding." *Journal of Open Source Software*, 6(65), 3544. doi:10.21105/joss.03544.
- Becker, R. A., Wilks, A. R., Brownrigg, R., Minka, T. P., & Deckmyn, A. (2023). *maps: Draw Geographical Maps*. R package version 3.4.2. URL <https://CRAN.R-project.org/package=maps>.
- Di Lorenzo, P. (2024). *usmap: US Maps Including Alaska and Hawaii*. R package version 0.7.1. URL <https://CRAN.R-project.org/package=usmap>.
- Lawlor, J. (2020). *PNWColors: Color Palettes Inspired by Nature in the US Pacific Northwest*. R package version 0.1.0. URL <https://CRAN.R-project.org/package=PNWColors>.
- Pebesma, E., & Bivand, R. (2023). *Spatial Data Science: With Applications in R*. Chapman and Hall/CRC. doi:10.1201/9780429459016.
- Giraud, T. (2024). *mapsf: Thematic Cartography*. R package version 0.10.1. URL <https://CRAN.R-project.org/package=mapsf>.
- Sievert, C. (2020). *Interactive Web-Based Data Visualization with R, plotly, and shiny*. Chapman and Hall/CRC, Florida. URL <https://plotly-r.com>.
- https://documentation.sas.com/doc/en/pgmsascdc/9.4_3.5/grmapref/titlepage.htm

RECOMMENDED READING

- [CDC - BRFSS](#)
- [CDC - 2022 BRFSS Survey Data and Documentation](#)

- [*Geocoding Made Easy • tidygeocoder \(jessecambon.github.io\)*](#)
- [*Making beautiful inset maps in R using sf, ggplot2 and cowplot | UPGo-McGill*](#)
- [*usmap*](#)
- [*dkahle/ggmap: A package for plotting maps in R with ggplot2 \(github.com\)*](#)
- [*mapsf: thematic cartography • mapsf \(riatelab.github.io\)*](#)
- [*Make ggplot2 Graphics Interactive • ggiraph package \(davidgohe.github.io\)*](#)
- [*Maps in R \(plotly.com\)*](#)
- [*Choropleth maps in R \(plotly.com\)*](#)
- [*The essential guide to binning in SAS - The DO Loop*](#)
- [*SAS® PROC GEOCODE by Example: A Case Study - SAS Support Communities*](#)
- [*How the RANK procedure calculates ranks with groups and ties - SAS Users*](#)

ACKNOWLEDGMENTS

Joshua J. Cook thanks his mentors, Louise Hadden, Dr. Achraf Cohen, Dr. Swann Adams, and Kirk Paul Lafler, as well as Richann Watson, Troy Martin Hughes, and Dr. Lisa Mendez for their general guidance and mentorship.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Joshua J. Cook, M.S. DS, M.S. CRM, ACRP-PM, CCRC
 Adjunct Professor / Clinical Researcher / Data Scientist
 Website: <https://joshua-j-cook-portfolios.netlify.app/>
 Email: jcook0312@outlook.com

—

Louise S. Hadden, B.A.
 Statistical Programmer
 Email: louisesquibbhadden@gmail.com

—

Swann Arp Adams, Ph.D, M.S., FACE
 Professor
 Department of Biobehavioral Health and Nursing Science
 College of Nursing and the Department of Epidemiology & Biostatistics
 Arnold School of Public Health
 Email: swann.adams@sc.edu

Any brand and product names are trademarks of their respective companies.