

SIDS-Related Mortality in Cook County, IL

Daniel P. Hall Riggins

2022-02-23

Contents

1	SIDS-Related Mortality in Cook County, IL	5
1.1	Abstract	5
2	Import the Data	7
2.1	Dependencies	7
2.2	Initial import	7
2.3	Join census tract populations	7
2.4	Save for use in other chapters	9
I	Exploratory Analysis	11
3	Mapping SIDS Deaths	13
3.1	Code to produce the map	13
4	Describing SIDS Deaths	17

Chapter 1

SIDS-Related Mortality in Cook County, IL

1.1 Abstract

Although overall rates of Sudden Infant Death Syndrome are declining, dramatic racial disparities persist in many metropolitan regions, including Cook County, IL. In this analysis, we sought to identify specific micro-regions of the county where disparities have been most severe so that public health practitioners can target future interventions with high precision. We also sought to predictively model where SIDS incidence is expected to be highest based on census-level socio-ecologic factors. We hypothesized that the microregions with highest SIDS incidence would follow patterns of historic racial segregation. Similarly, we hypothesized that factors reflecting high levels of socioeconomic disadvantage would be predictive of high SIDS incidence. However, we were interested in identifying clusters in the city where our model would not perform well with the hopes that studying these areas of poor predictive performance might identify new factors associated with particularly harmful or protective effects.

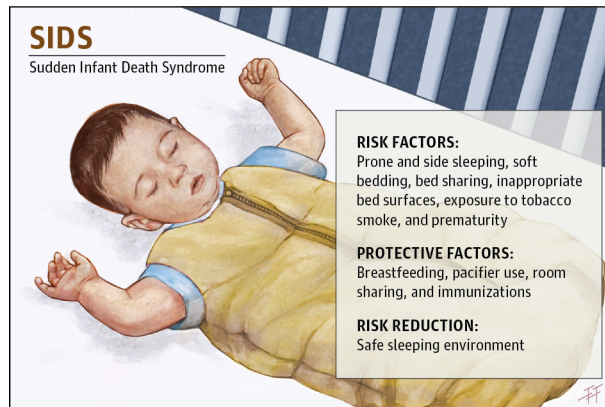


Image credit to JAMA Pediatrics

Chapter 2

Import the Data

2.1 Dependencies

```
# Load needed modules
box::use(
  dplyr[full_join, glimpse, select],
  janitor[clean_names],
  magrittr[`${}>%`],
  readxl[read_xlsx],
  sf[st_set_geometry],
  sids_data_wrangling = ./modules/sids_data_wrangling,
  tibble[as_tibble]
)
```

2.2 Initial import

First, read in the excel file that was originally shared for this project:

```
raw <-
  # Parse excel file
  read_xlsx("data/finaldataforanalysis3_220121.xlsx") %>%
  # Clean up variability in naming conventions
  clean_names()
```

2.3 Join census tract populations

Then, import census population data:

```
## This is a custom function I wrote that
## pulls data from the TidyCensus API about
## the population count of people under five
## years old and about spatial features
## for each census tract. I have commented it
## out and saved the result in an RDS file
## so as to not make a new call to the API
## every time this script is run. You can
## inspect the function definition in the
## modules folder of the source code.

# coords_and_pop_est <-
#   sids_data_wrangling$get_coords_and_pop_est(raw)
#
# saveRDS(coords_and_pop_est, "data/coords_and_pop_est.RDS")

coords_and_pop_est <- readRDS("data/coords_and_pop_est.RDS")

# Join the population counts to the imported dataframe
df <-
  coords_and_pop_est %>%
  # Drop geospatial features
  st_set_geometry(NULL) %>%
  # Convert to tibble format
  as_tibble() %>%
  # And join to raw
  full_join(raw)
#> Joining, by = "fips"

# Preview the data
glimpse(df)
#> Rows: 1,315
#> Columns: 32
#> $ fips                                <dbl> 17031807500, ~
#> $ pop_under_five                      <dbl> 151, 192, 21, ~
#> $ count_asphyxia                      <dbl> 0, 0, 1, 0, 0~
#> $ count_opioid_death                  <dbl> 1, 7, 2, 2, 6~
#> $ svi_socioeconomic                   <dbl> 0.1269, 0.593~
#> $ svi_household_composition_disability <dbl> 0.1728, 0.803~
#> $ svi_minority_language                <dbl> 0.7024, 0.677~
#> $ svi_housing_transportation           <dbl> 0.3690, 0.528~
#> $ svi_summary_ranking                  <dbl> 0.2470, 0.679~
#> $ pe_foreignborn                      <dbl> 31.6, 2.0, 1.~
#> $ pe_marriedmales                     <dbl> 62.5, 23.0, 3~
#> $ pe_marriedfemales                   <dbl> 56.6, 23.0, 2~
```



```

#> $ pedivorcewidowedmale <dbl> 6.4, 16.9, 7.~
#> $ pedivorcewidowedfemale <dbl> 16.8, 34.7, 3~
#> $ pelessthanhighschool <dbl> 7.1, 9.2, 8.0~
#> $ high schooldiploma <dbl> 14.6, 28.4, 2~
#> $ somecollege <dbl> 12.8, 26.4, 3~
#> $ collegediploma <dbl> 65.5, 36.0, 3~
#> $ black <dbl> 2.5, 97.4, 96~
#> $ white <dbl> 58.3, 0.7, 1.~
#> $ hispanic <dbl> 5.6, 0.0, 2.2~
#> $ male <dbl> 48.8, 50.8, 3~
#> $ percent_employed <dbl> 61.6, 49.0, 4~
#> $ incomelt10 <dbl> 0.0, 15.7, 10~
#> $ incomelt25 <dbl> 3.6, 15.6, 22~
#> $ incomelt50 <dbl> 10.9, 15.9, 2~
#> $ incomelt75 <dbl> 15.7, 27.6, 1~
#> $ incomegt75 <dbl> 69.8, 25.3, 2~
#> $ privateinsurance <dbl> 78.9, 55.5, 5~
#> $ publicinsurance <dbl> 26.4, 43.5, 5~
#> $ noinsurance <dbl> 2.8, 12.2, 13~
#> $ spanish_language <dbl> 6.0, 2.1, 0.7~

```

2.4 Save for use in other chapters

```
saveRDS(df, file = "data/df.RDS")
```


Part I

Exploratory Analysis

Chapter 3

Mapping SIDS Deaths

#> PhantomJS not found. You can install it with `webshot::install_phantomjs()`. If it is installed, Visit http://danielriggins.com/widgets/cook_county_sids_deaths.html for a full-screen view.

3.1 Code to produce the map

3.1.1 Load Dependencies

```
box::use(
  dplyr[
    case_when,
    full_join,
    mutate,
    select
  ],
  leaflet[
    addLayersControl,
    addLegend,
    addPolygons,
    addProviderTiles,
    leaflet,
    setMaxBounds,
    setView
  ],
  leaflet.extras[addFullscreenControl],
  magrittr[`%>%`],
  sf[...],
  tibble[view]
```

```
)
```

3.1.2 Reshape data for use in the map

```
# Load SIDS death data
df <-
  # Load cached geospatial features
  readRDS("data/coords_and_pop_est.RDS") %>%
  # Join to cached dataframe
  full_join(readRDS("data/df.RDS")) %>%
  # Select ID and outcome variables
  select(fips, count_asphyxia) %>%
  # Turn outcome into an ordinal factor
  mutate(
    death_count = factor(
      case_when(
        count_asphyxia == 0 ~ "No Deaths",
        count_asphyxia == 1 ~ "One Death",
        count_asphyxia == 2 ~ "Two Deaths",
        count_asphyxia == 3 ~ "Three Deaths",
        count_asphyxia == 4 ~ "Four Deaths",
        count_asphyxia == 5 ~ "Five Deaths",
        count_asphyxia == 6 ~ "Six Deaths"
      ),
      ordered = TRUE,
      levels = c(
        "No Deaths",
        "One Death",
        "Two Deaths",
        "Three Deaths",
        "Four Deaths",
        "Five Deaths",
        "Six Deaths"
      )
    )
  )

# Configure color palette
sids_palette <-
  leaflet::colorFactor(
    palette = "magma",
    reverse = TRUE,
    levels = c(
      "No Deaths",
      "One Death",
```

```

        "Two Deaths",
        "Three Deaths",
        "Four Deaths",
        "Five Deaths",
        "Six Deaths"
      )
    )
  )
)

```

3.1.3 Create the Map

```

# Assign map to a widget object
m <- leaflet(df) %>%
  # Use CartoDB's background tiles
  addProviderTiles("CartoDB.Positron") %>%
  # Center and zoom the map to Cook County
  setView(lat = 41.816544, lng = -87.749500, zoom = 9) %>%
  # Add button to enable fullscreen map
  addFullscreenControl() %>%
  # Add census tract polygons colored to reflect the number of deaths
  addPolygons(
    # No borders to the polygons, just fill
    stroke = FALSE,
    # Color according to palette above
    color = ~ sids_palette(death_count),
    # Group polygons by number of deaths for use in the layer control
    group = ~ death_count,
    # Make slightly transparent
    fillOpacity = 0.7,
    # Click on the polygon to get its ID
    popup = ~ paste0("<b>FIPS ID:</b> ", as.character(fips))
  ) %>%
  #Add legend
  addLegend(
    title = "Count of SIDS deaths <br> per census tract <br> from 2015-2019",
    values = ~ death_count,
    pal = sids_palette,
    position = "topright"
  ) %>%
  # Add ability to toggle each factor grouping on or off the map
  addLayersControl(overlayGroups = c(
    "No Deaths",
    "One Death",
    "Two Deaths",
    "Three Deaths",
    "Four Deaths",

```

```
        "Five Deaths",  
        "Six Deaths"  
    ),  
    position = "topleft"  
)
```


Chapter 4

Describing SIDS Deaths

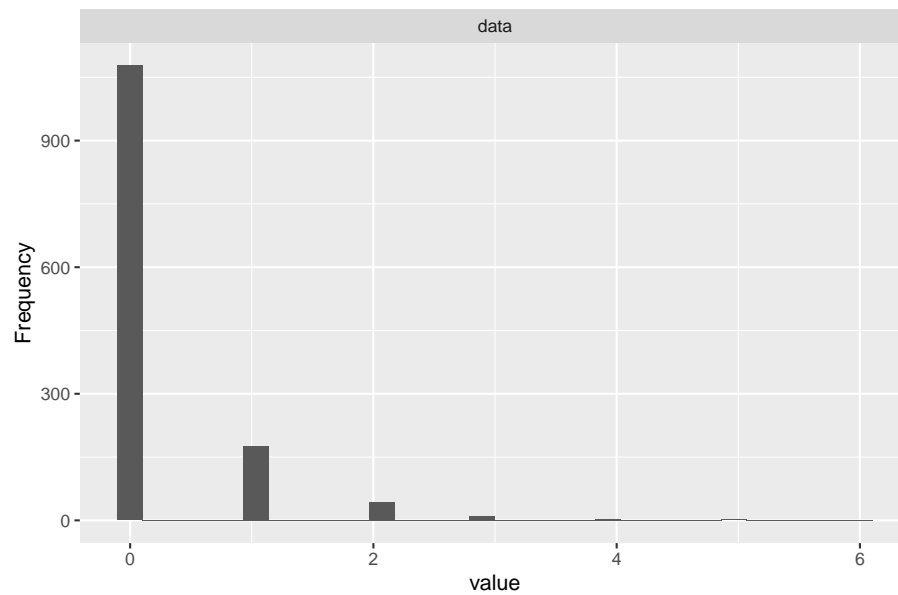
Our primary outcome in the analysis is:

`count_asphyxia` = the count of SIDS-related deaths in each census tract

This is a count-type outcome, which is typically modeled using the Poisson or Negative Binomial distributions. I will be using the Negative Binomial, since the Poisson is limited by having its variance needing to be equal to its mean.

Here we visualize the `count_asphyxia` distribution of count frequencies.

```
DataExplorer::plot_histogram(df$count_asphyxia)
```



As expected, the majority of census tracts do not have any SIDS-related deaths.

It's a little hard to see the tail of the distribution so let's tabulate:

```
janitor::tabyl(df$count_asphyxia)[,1:2]
#> df$count_asphyxia    n
#>      0 1078
#>      1  176
#>      2   44
#>      3   11
#>      4    3
#>      5    2
#>      6    1
```