Sudden Unexpected Infant Death in Cook County, IL from 2015-2019

This manuscript (permalink) was automatically generated from andtheWings/cook_county_suid_manuscript@5fbd4e5 on July 22, 2022.

Authors

- **Daniel P. Riggins** ✉
  0000-0002-6240-6212 · andtheWings · DanRiggins
  Section of Preventive Medicine, Cook County Health; Program in Public Health, Feinberg School of Medicine, Northwestern University · Funded by none

- **Huiyuan Zhang**

  Center for Health Equity and Inclusion, Cook County Health

- **William E. Trick**
  0000-0002-3243-5098
  Center for Health Equity and Inclusion, Cook County Health

✉ — Correspondence possible via GitHub Issues or email to Daniel P. Riggins <daniel.riggins@protonmail.com>.

Abstract

BACKGROUND: Although overall rates of Sudden Infant Death Syndrome (SIDS) are declining, significant racial disparities persist in many metropolitan areas, including Cook County, IL. In some areas, public health campaigns have successfully reduced SIDS through promotion of safe sleep practices. To correct these disparities, public health practitioners might benefit from guidance to specific high-risk neighborhoods to precisely target interventions within Cook County's large geographic breadth.

METHODS: We used geocoded medical examiner data and manual review of death records to identify SIDS events and locations. We characterized the demographics of SIDS, sought to identify specific clusters of census tracts in the county where SIDS has been most prevalent, and modeled prevalence based on census-level ecologic factors obtained from the census bureau. We hypothesized that regions with greatest prevalence would follow patterns of historic racial segregation and that factors reflecting elevated levels of socioeconomic disadvantage would have high utility in the model.

RESULTS: Demographic analysis revealed that infant mortality due to SIDS had a much higher proportion of Black individuals (70%) than Cook County's children overall (25%). Mapping confirmed that SIDS mortality was most concentrated in historically disinvested portions of the West and South Sides of the county, specifically in Chicago neighborhoods like Englewood, Humboldt Park, and Pullman as well as suburbs like Chicago Heights, Harvey, and Matteson. Our model included the following variables: number of residents under age five, percent of residents on public insurance, number of opioid overdose deaths, and percent of residents self-identifying as Black. The model was able to accurately retrodict whether census tracts had or had not contained SIDS deaths, but it was not flexible enough to enumerate the number of deaths in a tract if that was any greater than a count of 2.

COUNCLUSIONS: Clinicians and public health practitioners should concentrate preventive efforts against SIDS in the specific geographic areas listed above. We shall explore why certain neighborhoods fared better or worse than expected by our model in order to identify additional harmful or protective factors. We intend to involve individuals and organizations from the Black community in the design of future interventions.

Results

Case Characteristics

Our automated method for identifying cases of SUID from medical examiner archives generated 333 prospective cases. After comparison with the gold standard list of cases identified via manual review by a panel of experts, we added XXX cases and removed XXX cases (Table XXX). There were XXX cases not originally identified by manual review that we still deemed to be valid. This process yielded XXX valid SUID cases in total.

We tabulated descriptive characteristics for valid SUID cases (Table XXX). We compared the distributions of select characteristics compared to those in the overall population of Cook County children under the age of 5 (derived from the census). SUID cases had a higher prevalence of Black infants and favored location in Chicago (vs. Cook County Suburbs) compared to the reference population.

Census Tract Aggregate Characteristics

We tabulated characteristics of census tracts that had at least one case of SUID versus those that did not (Table XXX). Census tracts with at least one case of SUID appeared to have higher proportions of black people, XXX and lower proportions of XXX.

Comparing Cook County, IL, Census Tracts by Presence of SUID from 2015-2019

	No SUID, n = 1,078[1]	**SUID Present**, n = 237[1]
Age		
Population under 5 years old (%)	5.80 (4.30, 7.70)	6.25 (4.43, 8.07)
Gender		
Male (%)	48.8 (46.5, 50.9)	47.5 (43.9, 50.4)
Race/Ethnicity		
Black (%)	5 (2, 30)	72 (11, 95)
White (%)	67 (42, 83)	15 (3, 52)

[1] Median (IQR)

Comparing Cook County, IL, Census Tracts by Presence of SUID from 2015-2019		
Hispanic (%)	12 (6, 33)	10 (2, 32)
Culture		
Foreign-born (%)	18 (8, 32)	7 (2, 22)
Spanish-speaking (%)	9 (4, 26)	7 (2, 27)
Education		
Less than high school (%)	10 (5, 19)	18 (10, 25)
High school diploma (%)	24 (13, 31)	30 (24, 36)
Some college (%)	18 (13, 22)	23 (17, 29)
College diploma (%)	43 (26, 64)	24 (16, 37)
Employment		
Employed (%)	62 (55, 68)	52 (43, 60)
Income		
Earning < national 10th percentile (%)	6 (3, 10)	11 (6, 18)
Earning < national 25th percentile (%)	12 (7, 18)	20 (14, 27)
Earning < national 50th percentile (%)	20 (14, 26)	25 (19, 30)
Earning < national 75th percentile (%)	15.9 (12.2, 19.6)	15.6 (12.3, 19.7)
Earning > national 75th percentile (%)	41 (27, 57)	21 (14, 33)
Marital Status		
Married males (%)	44 (33, 55)	31 (20, 42)
Married females (%)	41 (29, 50)	26 (17, 40)
Divorced, widowed males (%)	9.0 (6.4, 12.5)	11.5 (7.9, 15.0)
Divorced, widowed females (%)	18 (13, 22)	19 (16, 25)
Household Density		
Average number of people per household	2.59 (2.26, 2.92)	2.72 (2.39, 3.07)

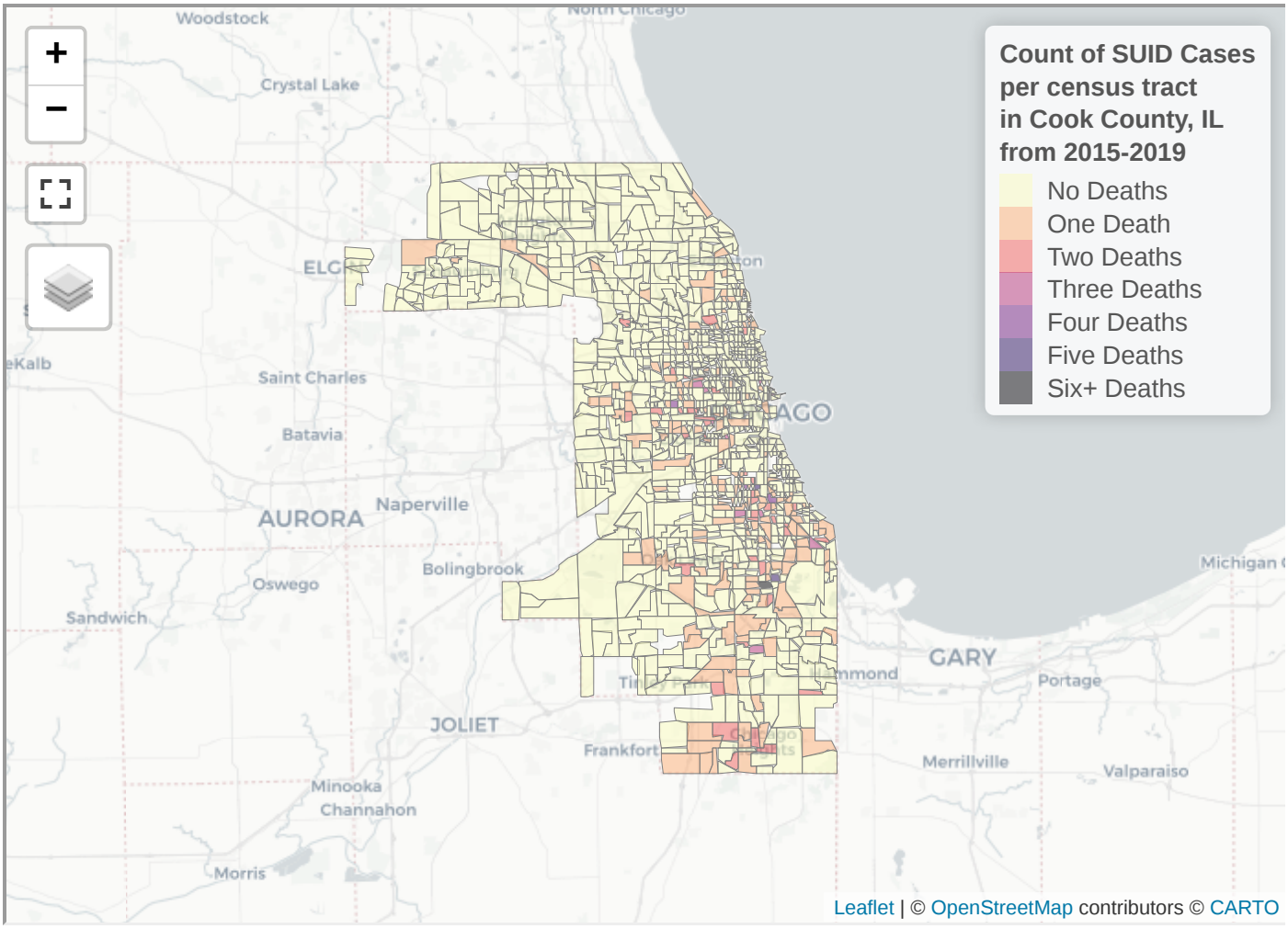
[1] Median (IQR)

Comparing Cook County, IL, Census Tracts by Presence of SUID from 2015-2019		
Health Insurance Status		
Private insurance (%)	68 (50, 83)	45 (33, 62)
Public insurance (%)	32 (23, 42)	50 (37, 60)
No insurance (%)	8 (4, 12)	10 (7, 15)
Social Vulnerability Index		
SVI socioeconomic status (national %ile)	53 (23, 80)	84 (65, 94)
SVI household composition & disability (national %ile)	38 (16, 65)	73 (47, 89)
SVI minority status & language (national %ile)	70 (50, 86)	75 (56, 88)
SVI housing type & transportation (national %ile)	54 (32, 76)	66 (40, 84)
SVI summary (national %ile)	57 (28, 79)	82 (67, 93)
Drug Use		
Opioid-related deaths (count)	2.0 (1.0, 5.0)	5.0 (2.0, 8.0)
approx_suid_incidence	0 (0, 0)	28 (16, 56)

[1] Median (IQR)

Mapping SUID Count per Census Tract

We created an interactive map showing the count of SUID cases color-coded over each census tract. We subjectively noted clusters of deaths on the Westside of Chicago in neighborhoods like Garfield Park, Humboldt Park, and North Lawndale; on the Southside in neighborhoods like Englewood, Pullman, and Woodlawn; and in Southern suburbs like Chicago Heights, Harvey, Hazel Crest, Olympia Fields, and Park Forest.



Modeling SUID Count per Census Tract

We fitted a negative-binomial model (estimated using ML) to predict suid_count with pop_under_five, public_insurance, count_opioid_death and white (formula: suid_count ~ pop_under_five + public_insurance + count_opioid_death + white + count_opioid_death:white). The model's explanatory power is substantial (Nagelkerke's R2 = 0.37). The model's intercept, corresponding to pop_under_five = 0, public_insurance = 0, count_opioid_death = 0 and white = 0, is at -2.44 (95% CI [-3.20, -1.70], p < .001). Within this model:

- The effect of pop under five is statistically significant and positive (beta = 1.01e-03, 95% CI [2.39e-04, 1.77e-03], p = 0.007; Std. beta = 0.15, 95% CI [0.04, 0.27])
- The effect of public insurance is statistically significant and positive (beta = 0.03, 95% CI [0.02, 0.04], p < .001; Std. beta = 0.52, 95% CI [0.33, 0.71])
- The effect of count opioid death is statistically non-significant and positive (beta = 2.27e-04, 95% CI [-0.02, 0.02], p = 0.984; Std. beta = 0.43, 95% CI [0.27, 0.59])
- The effect of white is statistically significant and negative (beta = -0.02, 95% CI [-0.03, -0.02], p < .001; Std. beta = -0.50, 95% CI [-0.70, -0.30])
- The interaction effect of white on count opioid death is statistically significant and positive (beta = 1.73e-03, 95% CI [9.32e-04, 2.52e-03], p < .001; Std. beta = 0.25, 95% CI [0.14, 0.37])

Standardized parameters were obtained by fitting the model on a standardized version of the dataset.

Supplement 1: Analytic Pipeline and Custom Functions

Pipeline Overview

We hosted both the code and manuscript on Github.

We orchestrated the data pipeline for the data analysis with the package {targets}.

```
library(targets)
```

If working with this repository and you want a visual overview of the pipeline, you can execute:

```
tar_visnetwork()
```

Here is information on the R session and its dependencies:

```
sessionInfo()

## R version 4.2.1 (2022-06-23)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Pop!_OS 22.04 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
## LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblasp-r0.3.20.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8
##  [2] LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8
##  [4] LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8
##  [6] LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8
##  [8] LC_NAME=C
##  [9] LC_ADDRESS=C
## [10] LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8
## [12] LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices
## [4] utils     datasets  methods
## [7] base
##
## other attached packages:
##  [1] magrittr_2.0.3
##  [2] gt_0.6.0
##  [3] lubridate_1.8.0
##  [4] tidycensus_1.2.2
##  [5] sf_1.0-8
##  [6] forcats_0.5.1
##  [7] stringr_1.4.0
##  [8] dplyr_1.0.9
##  [9] readr_2.1.2
## [10] tidyr_1.2.0
## [11] tibble_3.1.7
## [12] tidyverse_1.3.2
## [13] targets_0.12.1
## [14] purrr_0.3.4
## [15] ggplot2_3.3.6
## [16] performance_0.9.1
##
## loaded via a namespace (and not attached):
##   [1] googledrive_2.0.0
##   [2] colorspace_2.0-3
##   [3] ellipsis_0.3.2
```

```
##   [4] class_7.3-20
##   [5] leaflet_2.1.1
##   [6] rgdal_1.5-32
##   [7] estimability_1.4
##   [8] snakecase_0.11.0
##   [9] parameters_0.18.1
##  [10] fs_1.5.2
##  [11] rstudioapi_0.13
##  [12] proxy_0.4-27
##  [13] farver_2.1.1
##  [14] gtsummary_1.6.1
##  [15] fansi_1.0.3
##  [16] mvtnorm_1.1-3
##  [17] xml2_1.3.3
##  [18] splines_4.2.1
##  [19] codetools_0.2-18
##  [20] pscl_1.5.5
##  [21] knitr_1.39
##  [22] jsonlite_1.8.0
##  [23] broom_1.0.0
##  [24] dbplyr_2.2.1
##  [25] compiler_4.2.1
##  [26] httr_1.4.3
##  [27] emmeans_1.7.5
##  [28] backports_1.4.1
##  [29] Matrix_1.4-1
##  [30] assertthat_0.2.1
##  [31] fastmap_1.1.0
##  [32] gargle_1.2.0
##  [33] cli_3.3.0
##  [34] htmltools_0.5.3
##  [35] tools_4.2.1
##  [36] igraph_1.3.4
##  [37] gtable_0.3.0
##  [38] glue_1.6.2
##  [39] corrr_0.4.3
##  [40] rappdirs_0.3.3
##  [41] Rcpp_1.0.9
##  [42] cellranger_1.1.0
##  [43] vctrs_0.4.1
##  [44] nlme_3.1-157
##  [45] tigris_1.6.1
##  [46] broom.helpers_1.8.0
##  [47] crosstalk_1.2.0
##  [48] insight_0.18.0
##  [49] xfun_0.31
##  [50] networkD3_0.4
##  [51] ps_1.7.1
##  [52] rvest_1.0.2
##  [53] lifecycle_1.0.1
##  [54] googlesheets4_1.0.0
```

```
##  [55] MASS_7.3-58
##  [56] scales_1.2.0
##  [57] hms_1.1.1
##  [58] parallel_4.2.1
##  [59] yaml_2.3.5
##  [60] gridExtra_2.3
##  [61] see_0.7.1
##  [62] sass_0.4.2
##  [63] stringi_1.7.8
##  [64] highr_0.9
##  [65] bayestestR_0.12.1
##  [66] maptools_1.1-4
##  [67] e1071_1.7-11
##  [68] checkmate_2.1.0
##  [69] commonmark_1.8.0
##  [70] rlang_1.0.4
##  [71] pkgconfig_2.0.3
##  [72] evaluate_0.15
##  [73] lattice_0.20-45
##  [74] htmlwidgets_1.5.4
##  [75] labeling_0.4.2
##  [76] processx_3.7.0
##  [77] tidyselect_1.1.2
##  [78] DataExplorer_0.8.2
##  [79] R6_2.5.1
##  [80] generics_0.1.3
##  [81] base64url_1.4
##  [82] DBI_1.1.3
##  [83] mgcv_1.8-40
##  [84] pillar_1.8.0
##  [85] haven_2.5.0
##  [86] foreign_0.8-82
##  [87] withr_2.5.0
##  [88] units_0.8-0
##  [89] datawizard_0.4.1
##  [90] sp_1.5-0
##  [91] janitor_2.1.0
##  [92] modelr_0.1.8
##  [93] crayon_1.5.1
##  [94] uuid_1.1-0
##  [95] KernSmooth_2.23-20
##  [96] utf8_1.2.2
##  [97] correlation_0.8.1
##  [98] tzdb_0.3.0
##  [99] rmarkdown_2.14
## [100] grid_4.2.1
## [101] readxl_1.4.0
## [102] data.table_1.14.2
## [103] callr_3.7.1
## [104] reprex_2.0.1
## [105] digest_0.6.29
```

```
## [106] classInt_0.4-7
## [107] xtable_1.8-4
## [108] munsell_0.5.0
```

Ingestion

At time of primary analysis, DPR combined a data file generated by collaborator HZ and additional data pulled via the R package {tidycensus}.

DPR imported HZ's excel file into the Targets pipeline using the function `readxl::read_xls()`. He pulled from the tidycensus API using his function `get_suid_from_tidycensus()`. This function asks for variables from the American Community Survey's 5-year estimate for 2015-2019. The variables describe count of total population, count of population under age 5 years, average number of people per household, and geospatial shape of census tracts in Cook County, IL. The function also does some reshaping of data as noted in the code comments below:

```
## function() {
##
##     df1 <-
##         # Call the census API
##         tidycensus::get_acs(
##             geography = "tract",
##             variables = c(
##                 "B01003_001", # TOTAL POPULATION
##                 "B06001_002",  # Total Under 5 years
##                 "B25010_001" # AVERAGE HOUSEHOLD SIZE OF OCCUPIED HOUSING
UNITS
##             ),
##             state = "IL", # Illinois
##             county = 031, # Cook County
##             geometry = TRUE # Import census tract polygons too
##         ) |>
##         st_transform(crs = 4326) |>
##         # Drop margin of estimate and name variables
##         select(-moe, -NAME) |>
##         # Convert GEOID to numeric type
##         mutate(
##             GEOID = as.numeric(GEOID)
##         ) |>
##         # Reshape variable estimates into separate columns
##         tidyr::pivot_wider(
##             names_from = variable,
##             values_from = estimate
##         ) |>
##         # Clean names for consistency
##         rename(
##             fips = GEOID,
##             pop_total = B01003_001,
##             pop_under_five = B06001_002,
##             avg_peop_per_household = B25010_001
##         ) |>
##         # Put geometry column at the end of the table
##         relocate(
##             geometry,
##             .after = avg_peop_per_household
##         ) |>
##         st_as_sf()
##
##     return(df1)
## }
```

DPR combined the two data sources with his function `assemble_suid()`, which performs a left join on FIPS ID for observations from HZ's excel data, then does some wrangling around naming conventions, adds a boolean variable for whether SUID is present in a given tract, and rounds estimates to the first decimal point for consistency:

```
## function(suid_from_internal_raw_df, suid_from_tidycensus_raw_sf) {
##
##     df1 <-
##         left_join(
##             suid_from_internal_raw_df,
##             suid_from_tidycensus_raw_sf,
##             by = c("FIPS" = "fips")
##         ) |>
##         clean_names() |>
##         # Rename variables for consistency
##         rename(
##             suid_count = count_asphyxia,
##             foreign_born = pe_foreignborn,
##             married_males = pe_marriedmales,
##             married_females = pe_marriedfemales,
##             divorced_widowed_males = pedivorcewidowedmale,
##             divorced_widowed_females = pedivorcewidowedfemale,
##             lt_high_school = pelessthanhighschool,
##             high_school_diploma = highschooldiploma,
##             some_college = somecollege,
##             college_diploma = collegediploma,
##             employed = percent_enployed,
##             income_lt_10 = incomelt10,
##             income_lt_25 = incomelt25,
##             income_lt_50 = incomelt50,
##             income_lt_75 = incomelt75,
##             income_gt_75 = incomegt75,
##             private_insurance = privateinsurance,
##             public_insurance = publicinsurance,
##             no_insurance = noinsurance
##         ) |>
##         # Add new variables
##         mutate(
##             # Binary variable on whether suid is present in a tract
##             suid_present = case_when(
##                 suid_count > 0 ~ TRUE,
##                 TRUE ~ FALSE
##             ),
##             suid_count_factor = factor(
##                 case_when(
##                     suid_count == 0 ~ "No Deaths",
##                     suid_count == 1 ~ "One Death",
##                     suid_count == 2 ~ "Two Deaths",
##                     suid_count == 3 ~ "Three Deaths",
##                     suid_count == 4 ~ "Four Deaths",
##                     suid_count == 5 ~ "Five Deaths",
##                     suid_count > 5 ~ "Six+ Deaths"
##                 ),
##                 ordered = TRUE,
##                 levels = c(
##                     "No Deaths",
```

```
##                     "One Death",
##                     "Two Deaths",
##                     "Three Deaths",
##                     "Four Deaths",
##                     "Five Deaths",
##                     "Six+ Deaths"
##                 )
##             ),
##             approx_suid_incidence =
##                 round(
##                     suid_count / (pop_under_five / 5) * 1000,
##                     2
##                 ),
##             across(
##                 .cols = starts_with("svi_"),
##                 .fns = ~ round((.x * 100), digits = 1)
##             )
##         ) |>
##         st_as_sf()
##
##     return(df1)
## }
```

Mapping

DPR generated maps using the {leaflet} package ecosystem in R. His function `make_suid_count_map()`, makes a chloropleth map that overlays onto CartoDB basemap tiles and colors census tracts with a {viridis}-derived palette.

```
## function(suid_sf) {
##
##     # Configure color palette
##     suid_palette <-
##         leaflet::colorFactor(
##             palette = "magma",
##             reverse = TRUE,
##             levels = c(
##                 "No Deaths",
##                 "One Death",
##                 "Two Deaths",
##                 "Three Deaths",
##                 "Four Deaths",
##                 "Five Deaths",
##                 "Six+ Deaths"
##             )
##         )
##
##     obj1 <-
##         # Assign map to a widget object
##         leaflet::leaflet(suid_sf) |>
##             # Use CartoDB's background tiles
##             leaflet::addProviderTiles("CartoDB.Positron") |>
##             # Center and zoom the map to Cook County
##             leaflet::setView(lat = 41.816544, lng = -87.749500, zoom = 9)
|>
##             # Add button to enable fullscreen map
##             leaflet.extras::addFullscreenControl() |>
##             # Add census tract polygons colored to reflect the number of
deaths
##             leaflet::addPolygons(
##                 # No borders to the polygons, just fill
##                 color = "gray",
##                 weight = 0.25,
##                 opacity = 1,
##                 # Color according to palette above
##                 fillColor = ~ suid_palette(suid_count_factor),
##                 # Group polygons by number of deaths for use in the layer
control
##                 group = ~ suid_count_factor,
##                 # Make slightly transparent
##                 fillOpacity = 0.5,
##                 label = "Click me for more details!",
##                 # Click on the polygon to get its ID
##                 popup =
##                     ~ paste0(
##                         "<b>FIPS ID</b>: ", as.character(fips), "</br>",
##                         "<b>SUID Count</b>: ", suid_count, " deaths</br>",
##                         "<b>Total Population</b>: ", pop_total, "
people</br>",
##                         "<b>Population Under 5 Years Old</b>: ",
```

```
pop_under_five, " children</br>",
##                          "<b>Rough Incidence</b>: ", approx_suid_incidence,
" deaths per 1,000 babies"
##                      )
##             ) |>
##             #Add legend
##             leaflet::addLegend(
##                 title = "Count of SUID Cases<br> per census tract <br> in
Cook County, IL <br> from 2015-2019",
##                 values = ~ suid_count_factor,
##                 pal = suid_palette,
##                 position = "topright"
##             ) |>
##             # Add ability to toggle each factor grouping on or off the map
##             leaflet::addLayersControl(
##                 overlayGroups = c(
##                     "No Deaths",
##                     "One Death",
##                     "Two Deaths",
##                     "Three Deaths",
##                     "Four Deaths",
##                     "Five Deaths",
##                     "Six+ Deaths"
##                 ),
##                 position = "topleft"
##             )
##
##     return(obj1)
##
## }
```

Supplement 2: Model Selection

```
targets::tar_load(suid)
library(dplyr)
library(magrittr)
```

This supplement lays out how we selected the final model for predicting the number of SUID cases per census tract (variable `suid_count`).

From inception, we included in all models the variable `pop_under_five` (population of all children under age 5 years per census tract), which served as a quasi-exposure variable to account for variation in the number of people susceptible to an SUID incident.

Predictor Variable Candidates

To select predictor variable candidates, we sought to balance using variables that were highly correlated with `suid_count`, but not too highly correlated to each other to reduce multicollinearity.

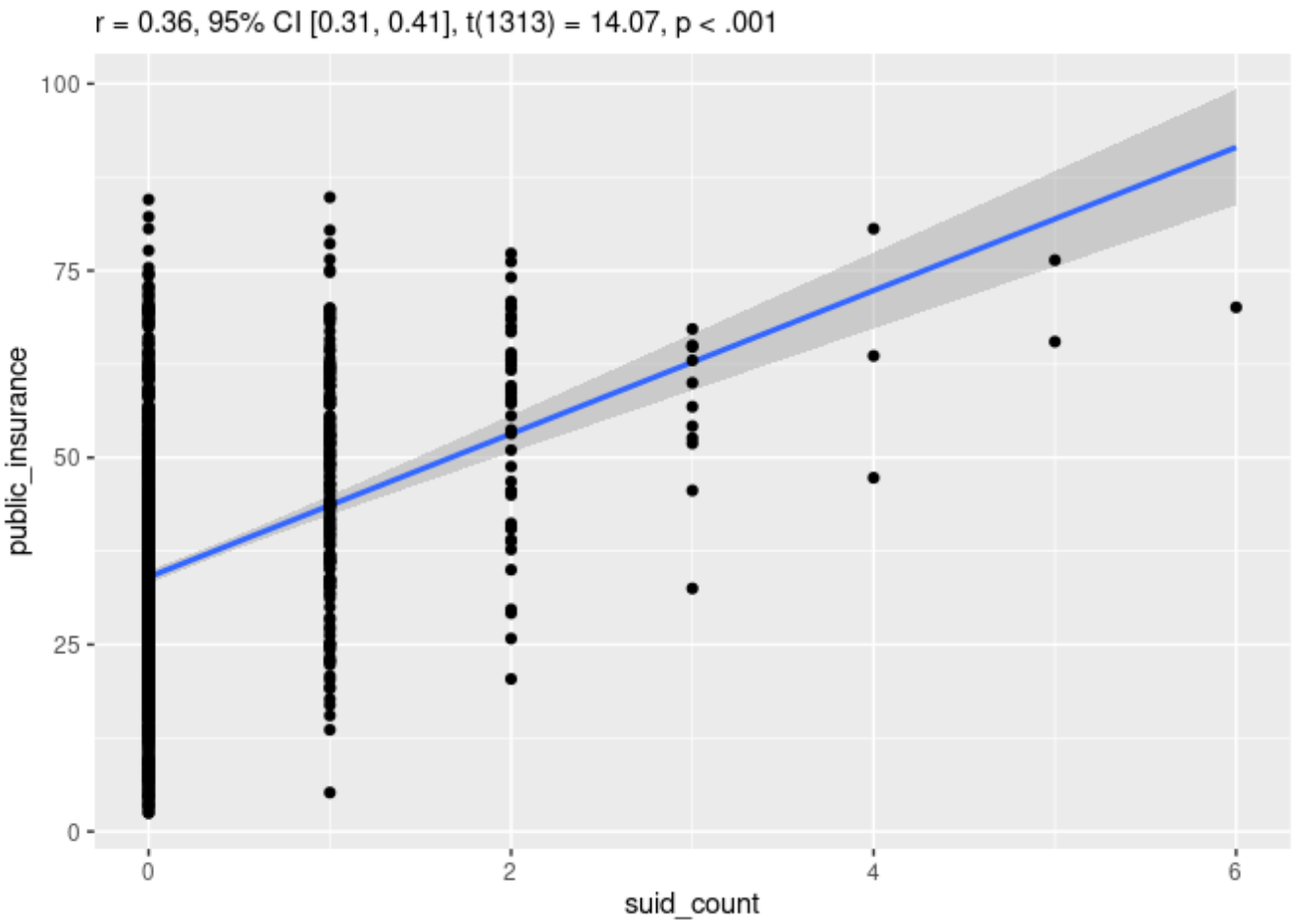
We started by generating a correlation dataframe and filtering for variables that had at least a weak association (> 0.20) with `suid_count`:

```
suid_correlations <-
    suid |>
    as_tibble() |>
    select(
        -fips,
        -geometry,
        -suid_present,
        -suid_count_factor
    ) |>
    relocate(suid_count) |>
    corrr::correlate() |>
    filter(abs(suid_count) > 0.20) |>
    arrange(desc(abs(suid_count)))

##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'

suid_correlations

## # A tibble: 17 × 35
##    term        suid_…¹ count…² svi_s…³
##    <chr>         <dbl>   <dbl>   <dbl>
##  1 public_ins…   0.362   0.354   0.819
##  2 white        -0.349  -0.332  -0.721
##  3 black         0.347   0.335   0.561
##  4 private_in…  -0.323  -0.356  -0.915
##  5 svi_househ…   0.311   0.319   0.672
##  6 income_gt_…  -0.310  -0.342  -0.898
##  7 married_fe…  -0.296  -0.355  -0.619
##  8 svi_socioe…   0.294   0.334  NA
##  9 income_lt_…   0.293   0.308   0.747
## 10 svi_summar…   0.291   0.349   0.941
## 11 employed     -0.291  -0.277  -0.625
## 12 income_lt_…   0.286   0.345   0.565
## 13 married_ma…  -0.286  -0.350  -0.621
## 14 college_di…  -0.272  -0.300  -0.856
## 15 count_opio…   0.248  NA       0.334
## 16 some_colle…   0.234   0.231   0.432
## 17 high_schoo…   0.217   0.243   0.714
## # … with 31 more variables:
## #   svi_household_composition_disability <dbl>,
## #   svi_minority_language <dbl>,
## #   svi_housing_transportation <dbl>,
## #   svi_summary_ranking <dbl>,
## #   foreign_born <dbl>,
## #   married_males <dbl>, …
## # ℹ Use `colnames()` to see all variable names
```

`publicinsurance`, the percentage of residents in each census tract on public insurance, had the strongest correlation with `suid_count`.

Here is the relationship visualized:

```
plot(correlation::cor_test(as_tibble(suid), "suid_count",
"public_insurance"))
```

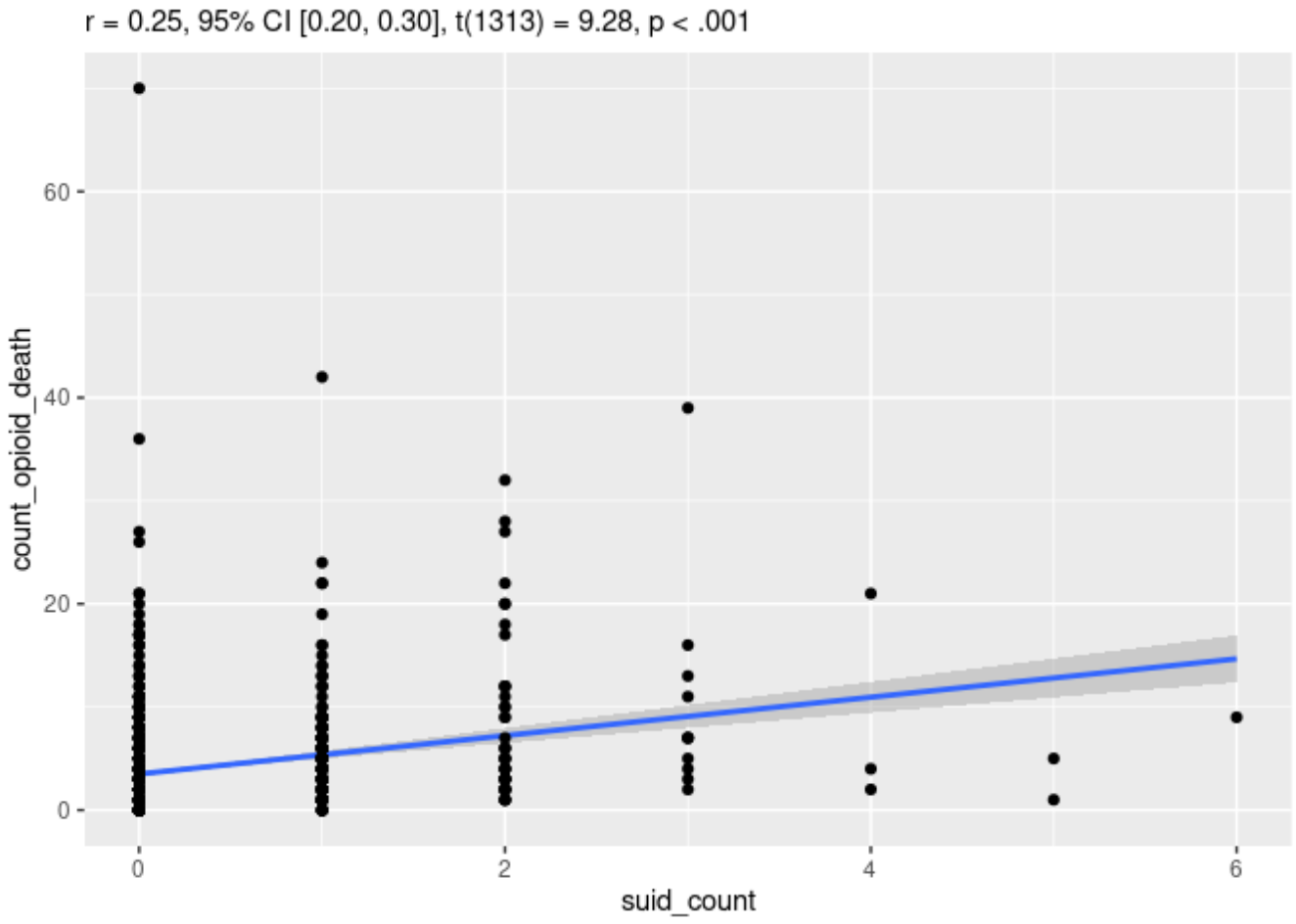


In order to minimize multicollinearity with additional predictor variables, we next selected for variables that had some degree of correlation with `suid_count`, but had no more than weak correlation (< 0.5) with `publicinsurance`:

```
suid_correlations |>
    filter(abs(suid_count) > 0.20) |>
    filter(abs(public_insurance) < 0.5)

## # A tibble: 1 × 35
##   term         suid_…¹ count…² svi_s…³
##   <chr>          <dbl>   <dbl>   <dbl>
## 1 count_opioi…   0.248      NA   0.334
## # … with 31 more variables:
## #   svi_household_composition_disability <dbl>,
## #   svi_minority_language <dbl>,
## #   svi_housing_transportation <dbl>,
## #   svi_summary_ranking <dbl>,
## #   foreign_born <dbl>,
## #   married_males <dbl>, …
## # ℹ Use `colnames()` to see all variable names
```

This just left `count_opioid_death`, the count of opioid-related deaths in each census tract.

```
plot(correlation::cor_test(as_tibble(suid), "suid_count",
"count_opioid_death"))
```



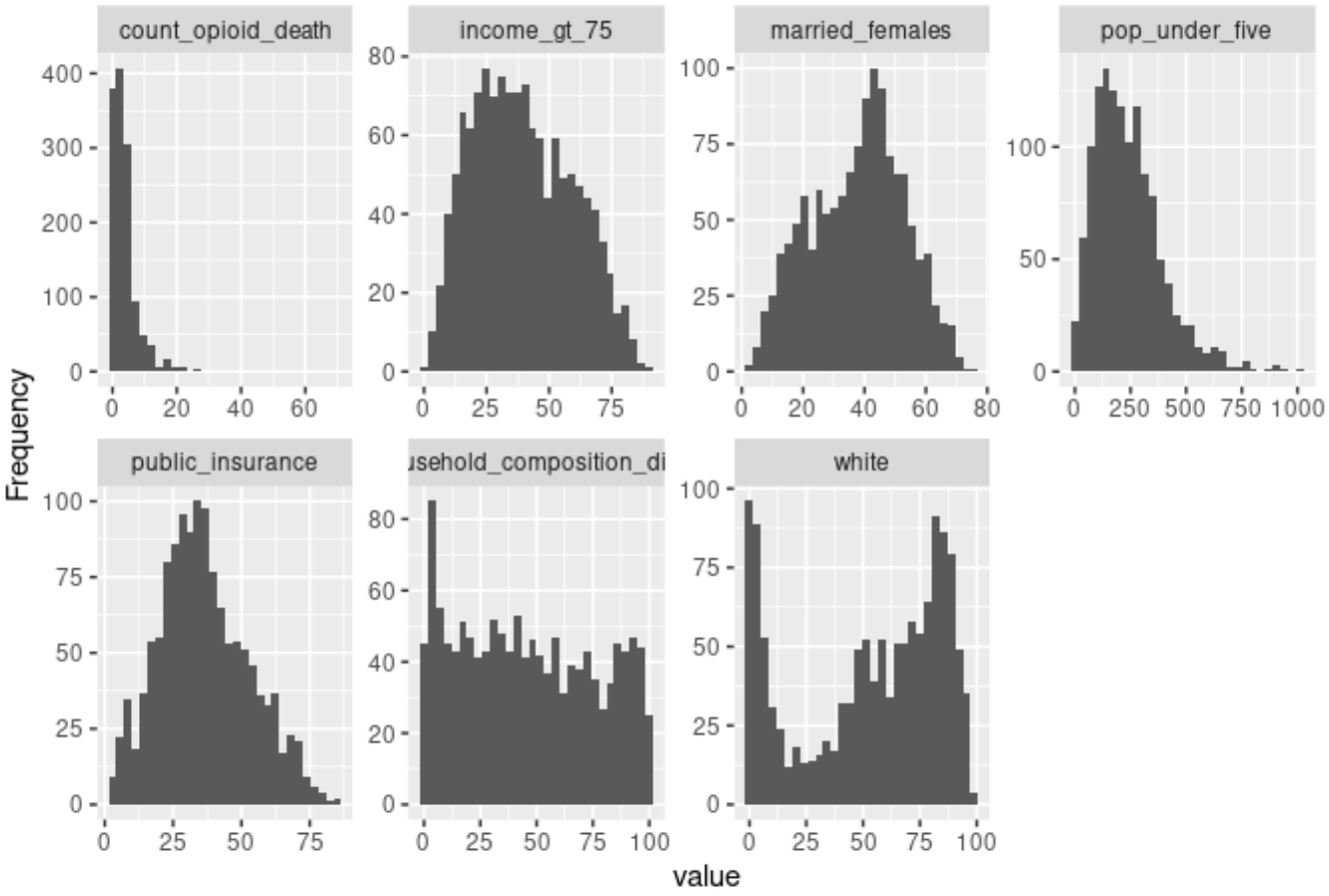
There was a large outlier of many opioid deaths in a tract with no SUID deaths, but otherwise, there seemed to be a strong positive trend.

To fill out our candidates, we selected the next four variables that correlated most with `suid_count`, without being fully redundant (e.g. not selecting `black` when `white` was already in the list):

- `white` = the percentage of residents in each census tract identifying their race as White
- `svi_household_composition_disability` = percentile ranking for each census tract on the Social Vulnerability Index for Household Composition & Disability, which is a mash-up of information about households that include people older than 65, people younger than 17, people with disabilities, and/or single-parents
- `income_gt_75` = the percentage of residents in each census tract whose income were greater than the national 75th percentile
- `married_females` = the percentage of female residents who were married in each census tract

Here are the summary statistics and visualized distributions of our candidate predictors:

```
as_tibble(suid) |>
    select(
        pop_under_five,
        public_insurance,
        count_opioid_death,
        white,
        svi_household_composition_disability,
        income_gt_75,
        married_females
    ) %T>%
    DataExplorer::plot_histogram() |>
    summary()
```


count_opioid_death
income_gt_75
married_females
pop_under_five
public_insurance
ısehold_composition_di
white
Frequency
value

```
##  pop_under_five   public_insurance
##  Min.   :   0.0   Min.   : 2.50
##  1st Qu.: 127.0   1st Qu.:24.80
##  Median : 213.5   Median :34.70
##  Mean   : 237.7   Mean   :36.40
##  3rd Qu.: 318.0   3rd Qu.:47.65
##  Max.   :1011.0   Max.   :84.80
##  NA's   :31
##  count_opioid_death     white
##  Min.   : 0.000     Min.   : 0.00
##  1st Qu.: 1.000     1st Qu.:24.00
##  Median : 3.000     Median :59.50
##  Mean   : 3.957     Mean   :52.84
##  3rd Qu.: 5.000     3rd Qu.:81.00
##  Max.   :70.000     Max.   :99.00
##
##  svi_household_composition_disability
##  Min.   :  0.10
##  1st Qu.: 19.30
##  Median : 43.60
##  Mean   : 46.02
##  3rd Qu.: 71.75
##  Max.   :100.00
##
##   income_gt_75   married_females
##  Min.   : 0.70   Min.   : 1.90
##  1st Qu.:23.50   1st Qu.:26.10
##  Median :37.30   Median :39.60
##  Mean   :39.33   Mean   :37.77
##  3rd Qu.:54.50   3rd Qu.:48.90
##  Max.   :90.60   Max.   :75.20
##
```

Model Type

We explored the general family of models that expect an outcome distribution to be a count variable. The distribution types included Poisson, Negative Binomial, and their zero-inflated variants.

First, we compared each model type on its predictive performance using just the exposure variable and an intercept:

```
list(
    poisson =
        glm(
            suid_count ~ pop_under_five,
            family = poisson(),
            data = suid
        ),
    zero_infl_poisson =
        pscl::zeroinfl(
            suid_count ~ pop_under_five,
            data = suid
        ),
    neg_bin =
        MASS::glm.nb(
            suid_count ~ pop_under_five,
            data = suid
        ),
    zero_infl_neg_bin =
        pscl::zeroinfl(
            suid_count ~ pop_under_five,
            dist = "negbin",
            data = suid
        )
) |>
compare_performance() |>
print_html()
```

Comparison of Model Performance Indices

Name	Model	AIC	AIC weights	BIC	BIC weights	RMSE	Sigma	Score_log	Score_spherical	R2	R2 (adj.)	Nagelkerke's R2
poisson	glm	1674.76	< 0.001	1685.07	< 0.001	0.62	0.95	-0.65	0.03			6.66e-04
zero_infl_poisson	zeroinfl	1595.73	0.003	1616.37	< 0.001	0.62	0.62	-0.62	0.03	1.01e-04	-1.46e-03	
neg_bin	negbin	1584.44	0.829	1599.92	0.999	0.62	0.74	-0.63	0.03			6.21e-04
zero_infl_neg_bin	zeroinfl	1587.63	0.168	1613.42	0.001	0.62	0.62	-0.62	0.03	1.22e-04	-1.44e-03	

Although the models seemed generally comparable, the negative binomial model type had both the best Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) scores. Minimizing these two scores theoretically optimizes balance between over- and under-fitting to the observed data.

Next, we compared performance of a nested sequence of predictor candidates:

```
list(
    base_formula = suid_count ~ pop_under_five,
    base_formula_plus_one = suid_count ~ pop_under_five + public_insurance,
    base_formula_plus_two = suid_count ~ pop_under_five + public_insurance +
count_opioid_death,
    base_formula_plus_three = suid_count ~ pop_under_five + public_insurance
+ count_opioid_death + white,
    base_formula_plus_four = suid_count ~ pop_under_five + public_insurance +
count_opioid_death + white + svi_household_composition_disability,
    base_formula_plus_five = suid_count ~ pop_under_five + public_insurance +
count_opioid_death + white + svi_household_composition_disability +
income_gt_75,
    base_formula_plus_six = suid_count ~ pop_under_five + public_insurance +
count_opioid_death + white + svi_household_composition_disability +
income_gt_75 + married_females
) |>
    purrr::map(~MASS::glm.nb(.x, data = suid)) |>
    compare_performance(rank = TRUE) |>
    print_html()
```

Comparison of Model Performance Indices

Name	Model	Nagelkerke's R2	RMSE	Sigma	Score_log	Score_spherical	AIC weights	BIC weights	Performance-Score
base_formula_plus_three	neg bin	0.34	0.59	0.74	-0.53	0.03	0.387	0.963	82.02%
base_formula_plus_five	neg bin	0.35	0.58	0.74	-0.53	0.03	0.325	0.005	67.65%
base_formula_plus_six	neg bin	0.35	0.58	0.74	-0.53	0.03	0.122	< 0.001	59.69%
base_formula_plus_four	neg bin	0.34	0.59	0.74	-0.53	0.03	0.165	0.031	59.69%
base_formula_plus_two	neg bin	0.32	0.62	0.74	-0.54	0.03	< 0.001	< 0.001	39.77%

NA

Comparison of Model Performance Indices									
base_formula_plus_one	neg bin	0.30	0.58	0.75	-0.54	0.03	< 0.001	< 0.001	39.47%
base_formula	neg bin	6.21e-04	0.62	0.74	-0.63	0.03	< 0.001	< 0.001	20.55%
NA									

Since all models were of the same type, in this comparison, we used the `compare_performance()` function's ranking algorithm, which chose the model with the exposure variable plus 3 covariates to perform the best.

Then we used the `select_parameters()` function's heuristic algorithm to check if any interaction terms were worth including in the model.

```
MASS::glm.nb(
    suid_count ~ (pop_under_five + public_insurance + count_opioid_death +
white)^2,
    data = suid
) |>
parameters::select_parameters() |>
parameters::parameters() |>
print_html()
```

Model Summary					
Parameter	Coefficient	SE	95% CI	z	p
(Intercept)	-2.10	0.44	(-2.98, -1.24)	-4.75	< .001
pop under five	9.07e-04	3.81e-04	(1.23e-04, 1.68e-03)	2.38	0.017
public insurance	0.03	7.38e-03	(0.01, 0.04)	3.45	< .001
count opioid death	2.89e-03	0.01	(-0.02, 0.03)	0.26	0.798
white	-0.03	7.83e-03	(-0.05, -0.02)	-4.12	< .001
public insurance * white	2.44e-04	1.73e-04	(-8.97e-05, 5.88e-04)	1.41	0.158
count opioid death * white	1.67e-03	3.86e-04	(8.59e-04, 2.47e-03)	4.32	< .001

The interaction between `ccount_opioid_death` and `white` was proposed by the algorithm and statistically significant, so we included it as the only interaction term.

Next, we compared our final set of predictors in the panel of model types again:

```
final_models <-
    list(
        poisson =
            glm(
                suid_count ~ pop_under_five + public_insurance +
count_opioid_death + white + count_opioid_death:white,
                family = poisson(),
                data = suid
            ),
        zero_infl_poisson =
            pscl::zeroinfl(
                suid_count ~ pop_under_five + public_insurance +
count_opioid_death + white + count_opioid_death:white,
                data = suid
            ),
        neb_bin =
            MASS::glm.nb(
                suid_count ~ pop_under_five + public_insurance +
count_opioid_death + white + count_opioid_death:white,
                data = suid
            ),
        zero_infl_neg_bin =
            pscl::zeroinfl(
                suid_count ~ pop_under_five + public_insurance +
count_opioid_death + white + count_opioid_death:white,
                dist = "negbin",
                data = suid
            )
    )

final_models |>
    compare_performance() |>
    print_html()
```

Comparison of Model Performance Indices

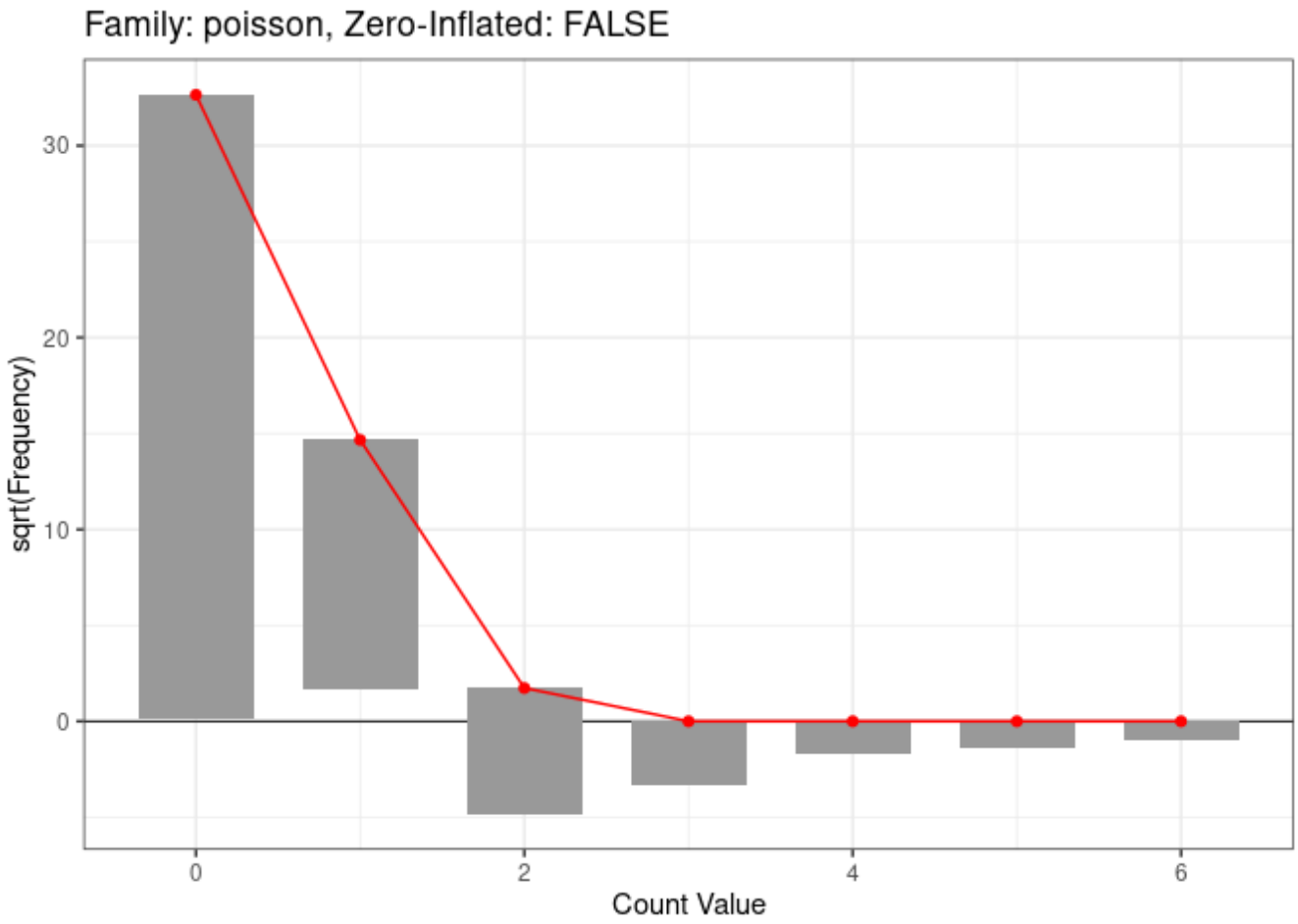
Name	Model	AIC	AIC weights	BIC	BIC weights	RMSE	Sigma	Score_log	Score_spherical	R2	R2 (adj.)	Nagelkerke's R2
poisson	glm	13647.76	< 0.001	13955.71	0.009	0.56	0.81	-0.53	0.03			0.37
zero_infl_poisson	zeroinfl	13499.84	0.310	14111.73	< 0.001	0.56	0.56	-0.52	0.03	0.07	0.07	

Comparison of Model Performance Indices												
neb_bin	neg bin	135 0.11	0.270	138 6.21	0.991	0.5 6	0.7 4	-0.52	0.03			0.37
zero_infl_ neg_bin	zer oin fl	134 9.23	0.420	141 6.28	< 0.001	0.5 6	0.5 6	-0.52	0.03	0. 0 7	0.07	

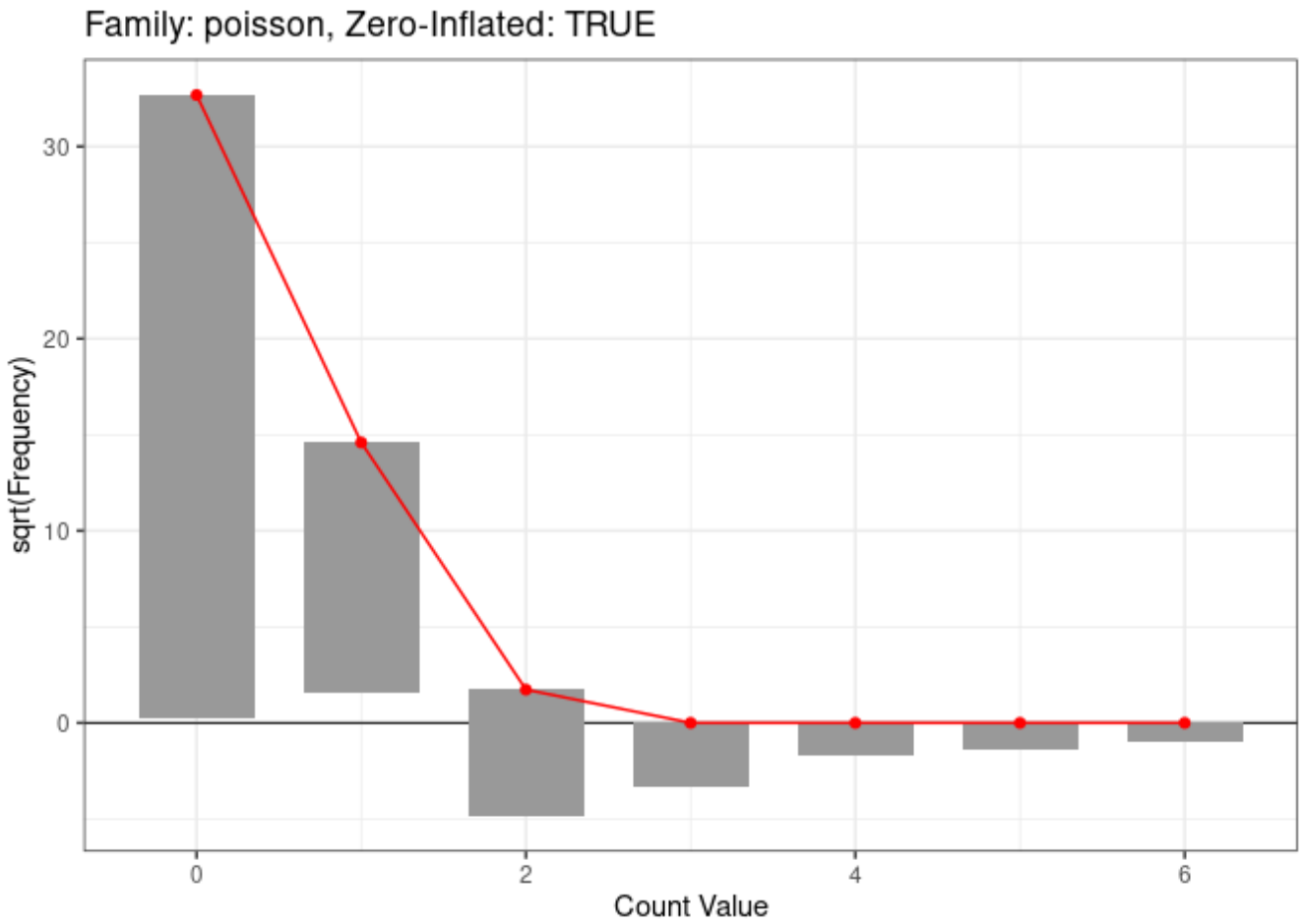
Precision-related scores were fairly similar and the R^2 terms were not directly comparable, so we again paid most attention to AIC and BIC. These two scores disagreed on which model type had the best fit, but BIC more emphatically chose the negative binomial model, so we stuck with this type.

Another way to compare the goodness of fit was with a visual check of rootograms:

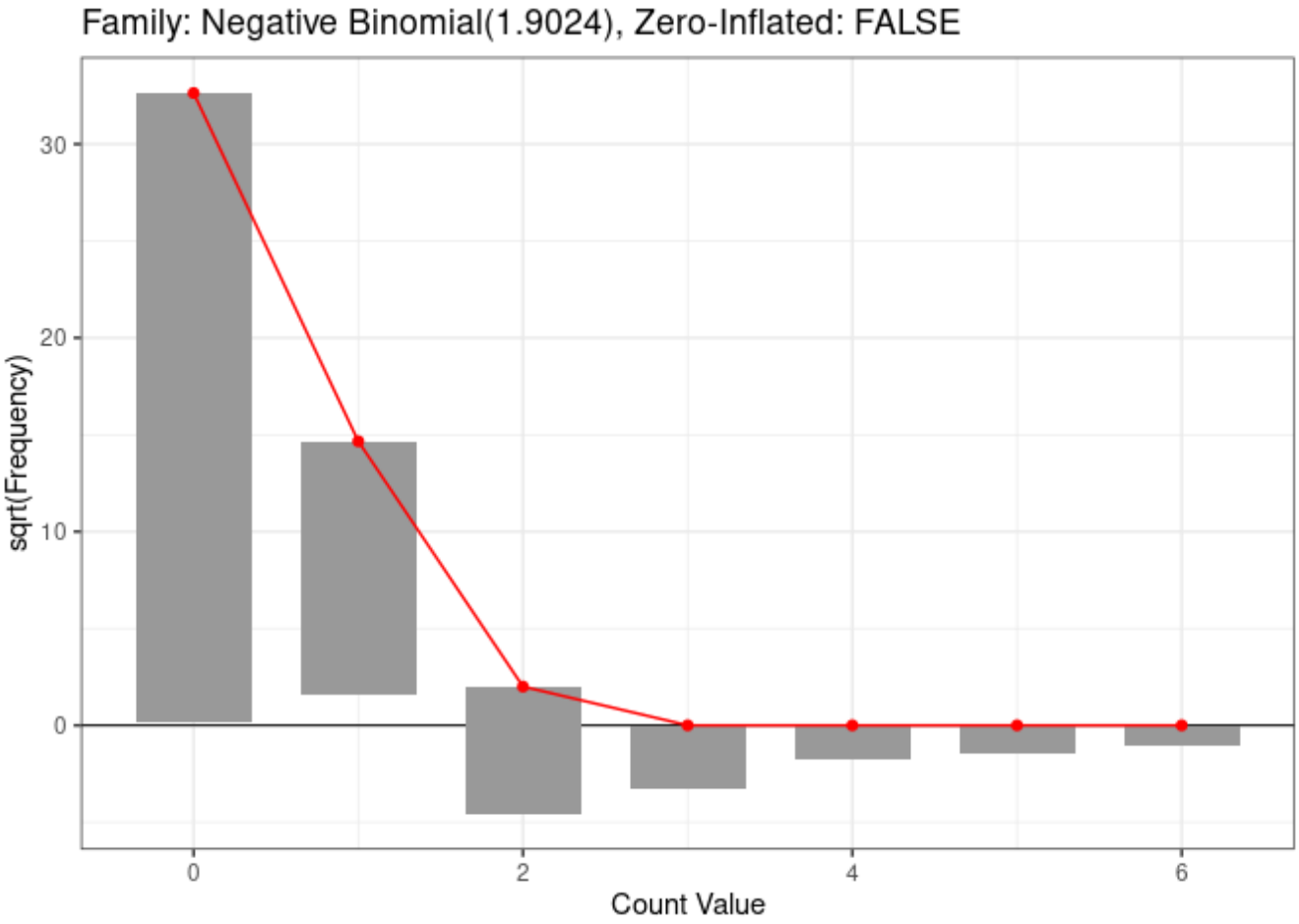
```
source("R/plot_rootogram.R")

final_models |>
    purrr::map(plot_rootogram)

## $poisson
```



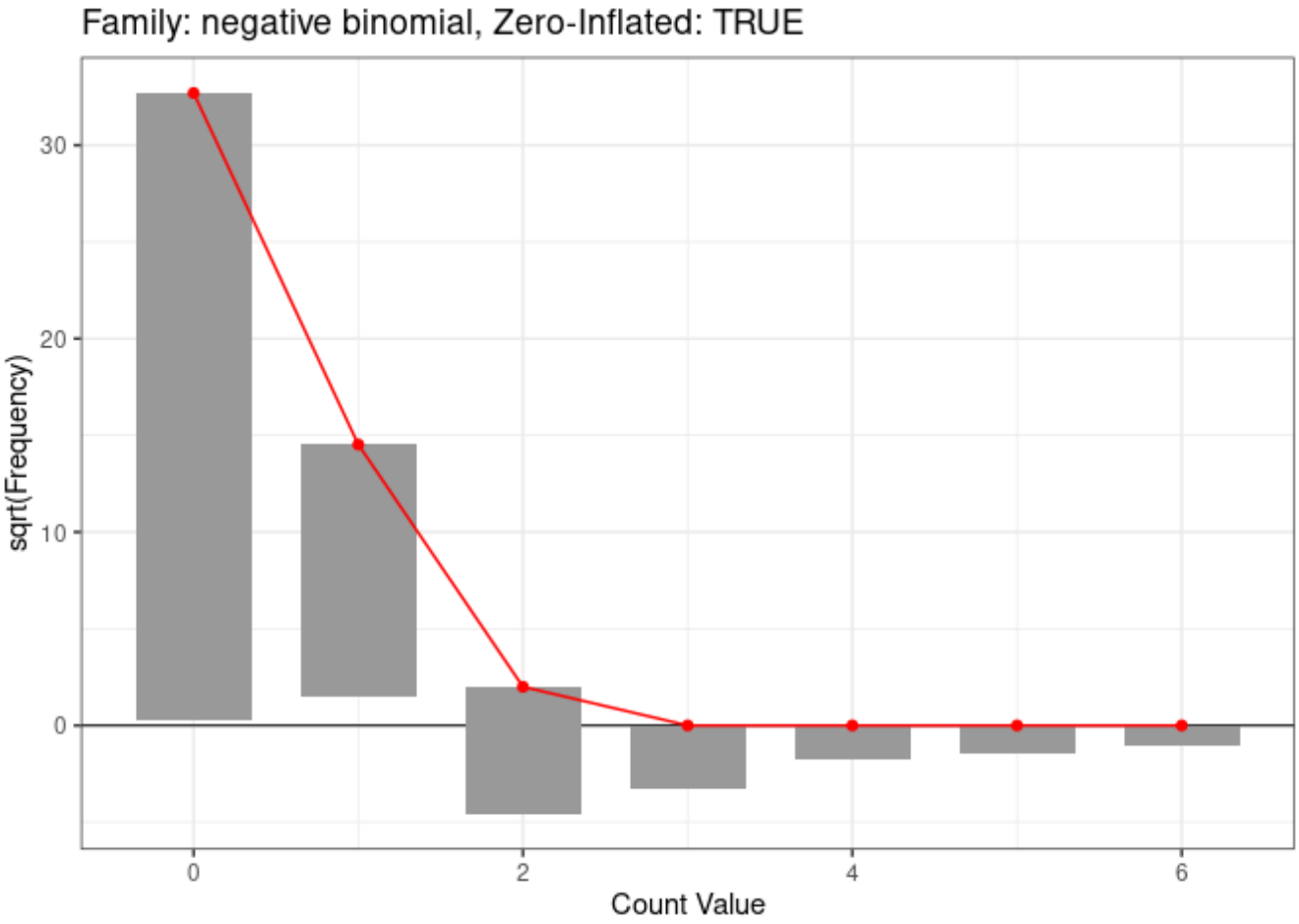
```
##
## $zero_infl_poisson
```




```
##
## $neb_bin
```




```
##
## $zero_infl_neg_bin
```



Visually speaking, the negative binomial model type and its zero-inflated variant looked almost identical, so we felt further reassured in selecting the plain, non-zero-inflated type.

References