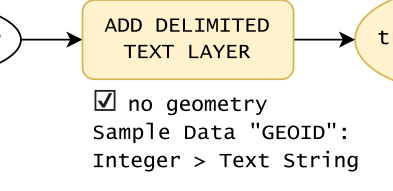
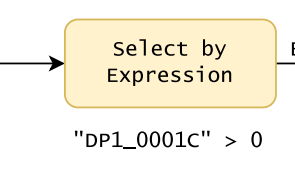
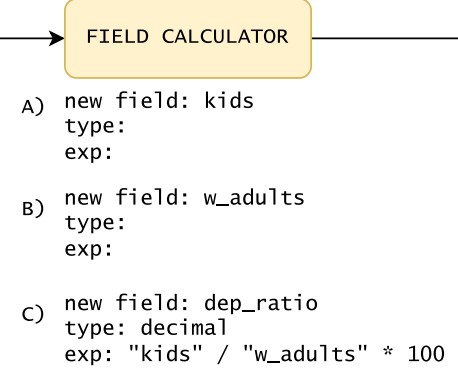
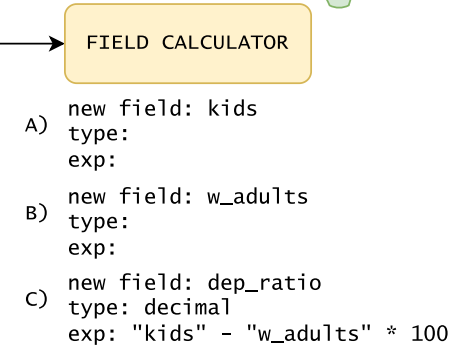
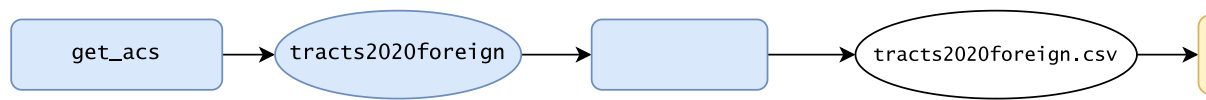
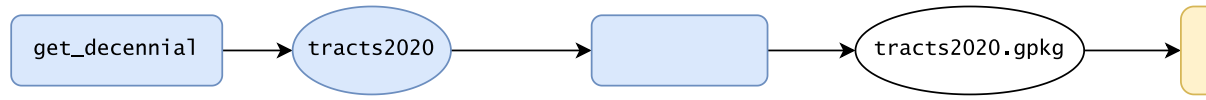
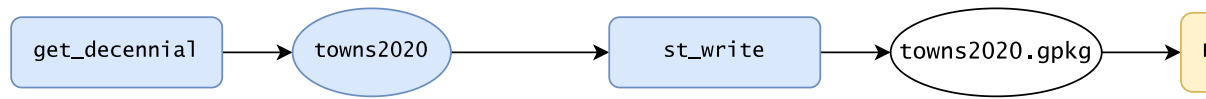
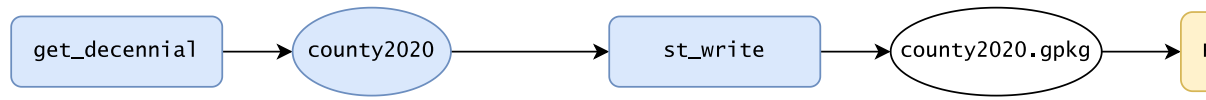
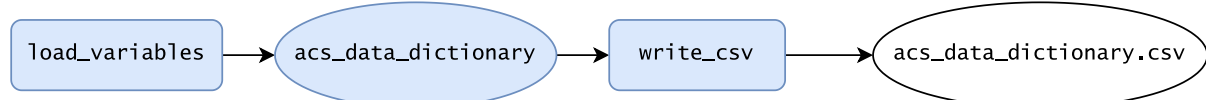
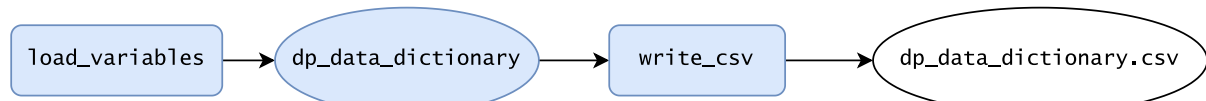
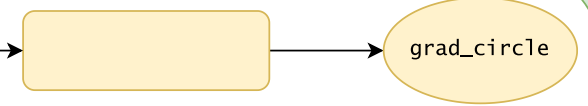


# RSTUDIO

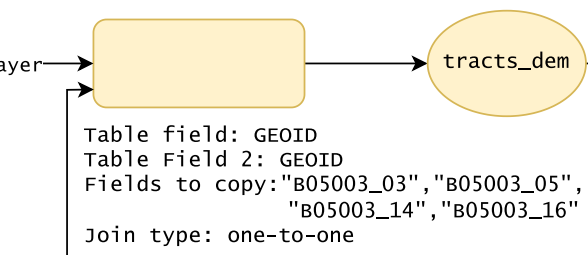
# QGIS



Map 1  
Counties Child Dependency



Map 2  
Towns Child Dependency

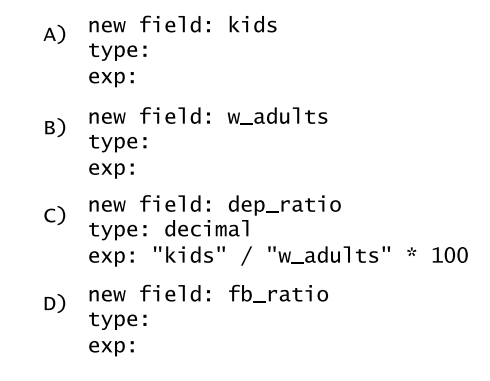


Map 3  
Towns Proportional Symbol

Map 4  
Towns Dot Density

Map 5  
Tracts Child Dependency

Map 6  
Tracts Foreign Born Kids



```

#=====  

# Setting Up Our Environment  

#=====  

# Install packages every time you use a new computer  

install.packages( c("sf", "tidycensus", "readr"))  

# Load packages every time you open RStudio  

library(sf)  

library(tidycensus)  

library(readr)  

#=====  

# Collecting Metadata  

#=====  

# To find the right variables to use in our analysis we want to use the  

# load_variables() to display all of the variables available from a data series  

# This table is data about our data (aka metadata)  

# Parameters:  

# year: the year for which you are requesting variables  

# dataset: the dataset name as used on the Census website  

dp_data_dictionary <- load_variables(year = 2020,  

                                   dataset = "dp")  

# We also want to find the right variables from a American Community Survey,  

# so we repeat the previous step with the other dataset  

acs_data_dictionary <- load_variables(year= 2020,  

                                   dataset= "acs5" )  

# We use write_csv() to translate that data into a CSV comma-delimited text file  

# for loading into other software like Excel or QGIS.  

# Parameters:  

# x: R data object you want to save  

# file: file path indicating where to save the obj as a CSV text file  

write_csv(x = dp_data_dictionary,  

         file = "raw/dp_data_dictionary.csv")  

# Repeat the the previous step with our acs_data_dictionary  

write_csv(x = ... ,  

         file = "...")  

#=====  

# Data Queries  

#=====  

# The get_decennial() function downloads data and feature geometry for the  

# decennial US Census

```

```

# Parameters:  

# geography: type of enumeration unit / level of census hierarchy, commonly:  

#   "country", "county subdivision", "tract", "block group", or "block"  

# variables or table:  

# variables: list of variables to download in format c("var1", ... "varN")  

# table: name of table for which to download all variables, typically  

#   the prefix of the variable name prior to the underscore "_"  

# year: the year for which you are requesting variables  

# sumfile: only needed for get_decennial(), designating which product from the  

#   decennial census to use, commonly "dp" for general demographic profile,  

#   "dhc" for more specific demographic and housing characteristics, and  

#   "pl" for public law redistricting data  

# state: specify the geographic extent of data to download using one or more  

#   state names or postal codes  

# output: specify "wide" so that results work in GIS with one row per feature  

# geometry: if TRUE, returns spatial geometry polygons for mapping but  

#   if FALSE, only returns a data table  

# keep_geo_vars: if TRUE, keep geographic attribute fields, e.g. name, area, etc.  

# Query demographic data for RI counties in 2020  

counties2020 <- get_decennial(geography = "county" ,  

                             table = "DP1",  

                             year = 2020,  

                             sumfile = "dp",  

                             state = "RI",  

                             output = "wide",  

                             geometry = TRUE,  

                             keep_geo_vars = TRUE )  

# To save counties as a geopackage, we use st_write()  

# Parameters:  

# obj: R data object you want to save  

# dsn: file path indicating where to save the obj as a geographic data file  

# layer: if saving to a geopackage, this is the name of the layer  

# append: add append = FALSE if you need to overwrite an existing data layer  

st_write(obj = counties,  

        dsn = "raw/counties2020.gpkg",  

        layer = "counties2020")

```

```

# Load the same census data for county subdivisions (towns)  

towns <- get_decennial(geography = "...",  

                      year = ... ,  

                      sumfile = "...",  

                      table = "...",  

                      state = "...",  

                      output = "...",  

                      geometry = ... ,  

                      keep_geo_vars = ... )  

# Save tracts2020 as a geopackage  

st_write(obj = ... ,  

        dsn = "raw/towns2020.gpkg",  

        layer = "towns2020")  

# Load the same census data for tracts  

... <- get_decennial(geography = "...",  

                    year = ... ,  

                    sumfile = "...",  

                    table = "...",  

                    state = "...",  

                    output = "...",  

                    geometry = ... ,  

                    keep_geo_vars = ... )  

# Save counties2020 as a geopackage  

st_write(obj = ...,  

        dsn = "...",  

        layer = "...")  

# To get foreign born and citizenship characteristics for every census tract  

# in Rhode Island we load ACS data using get_acs()  

# We are querying an attribute table with no geographic features, so we can omit  

# geometry and keep_geo_vars. Further, get_acs does not need the sumfile parameter  

tracts2020foreign <- get_acs(geography = "tract",  

                            year = 2020,  

                            survey = "acs5",  

                            table = "B05003",  

                            state = "RI",  

                            output = "wide")  

# Save the data table, adding the na parameter to specify what to write in place  

# of missing data, rather than writing the letters `na`.  

# Two double quotes "" results in a blank / empty table cell.  

write_csv(x = ... ,  

        file = "raw/tracts2020foreign.csv",  

        na = "")

```