`{censobr}`: Easy Access to Brazilian Demographic Censuses

Rafael H. M. Pereira, IPEA   Rogério Jerônimo Barbosa, IESP/UERJ

2024-11-14

1. Introduction

Census data are indispensable resources for social research. In Brazil, the Demographic Censuses conducted by the Instituto Brasileiro de Geografia e Estatística (IBGE) serve as a foundational source for informing public policy and advancing academic research. However, the large scale of these datasets, often exceeding 20 gigabytes, presents substantial challenges for researchers, particularly those working with limited computing resources.

To address these issues, we developed `{censobr}`, an R package designed to facilitate the efficient access and manipulation of Brazilian census data, spanning from 1960 to 2022. Relying on the Apache Arrow platform, `{censobr}` enables users to manage larger-than-memory datasets using a columnar memory format that optimizes data access and processing. Arrow seamlessly integrates with `{dplyr}` syntax, allowing users to interact with in-disk datasets using familiar R commands, thus streamlining the analytical workflow without necessitating extensive new learning. `{censobr}` also integrates with `{geobr}`, an R package for geographic data in Brazil, using matching geographic identifiers that facilitate the merging of census data with spatial geometries for visualization and analysis. Our goal is to make census data more accessible while maintaining the flexibility and power of the R environment.

This paper introduces the `{censobr}` package, outlines its core functionalities, and provides illustrative examples that demonstrate its applications in social science research.

2. Brazilian Demographic Censuses: An Overview

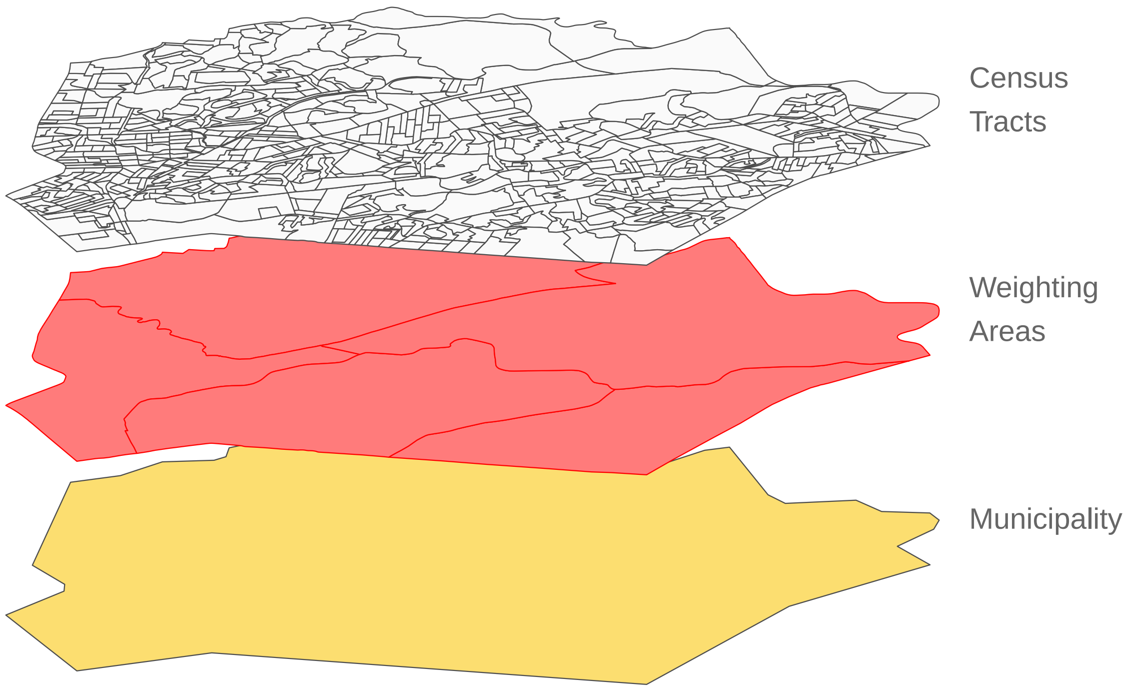
The Brazilian Demographic Censuses represent one of the most extensive population data collection initiatives worldwide. The first census was conducted in 1872, during Brazil's imperial period, while modern censuses began in 1940 under the coordination of the Instituto Brasileiro de Geografia e Estatística (IBGE).

Since 1960, each Brazilian census is divided into two principal components: the **universe survey** and the **sample survey** (ressembling the US Census). The universe survey, also called the basic questionnaire, is applied to the entire population and gathers essential demographic and housing information through a shorter form containing between 9 and 30 questions. In contrast, the sample survey targets a representative subset of the population and uses a longer questionnaire that includes the universe questions, along with more detailed questions on topics such as migration, religion, education, fertility, income, and employment. The sample fraction was 25% in the 1960, 1970, and 1980 censuses, but was reduced to 10% from 1991 onwards.

The **microdata** from the sample survey consist of a dataset in which each row represents an instance of data collection. In IBGE surveys and censuses, "persons" and "households" are tipically the units of analysis of microdata. "Persons" refer to individual members of the population, and their data typically includes personal characteristics such as age, gender, education, etc. "Households" represent residential units, which consist of one or more persons living together, sharing living expenses, and typically occupying a single housing unit. Household-level microdata include variables related to housing conditions, access to services such as water and sanitation, and household composition, such as the number of residents and their relationships.

On the other hand, **aggregated data** from the universe survey provide summary statistics for geographic units, such as census tracts, municipalities, and states. It is useful for spatial analysis and for examining regional disparities and trends at various levels of granularity. **Census tracts** are the smaller geographic units in the censuses. They are contiguous areas typically containing about 200 households, designed to facilitate efficient enumeration and data collection. Over time, the number of census tracts has expanded in response to population growth and changes in settlement patterns, with approximately 216,000 census tracts in 2000, 314,000 in 2010, and 452,000 in 2022. These tracts form the fundamental spatial units for census data collection, reflecting the evolution of Brazilian urban and rural landscapes.

IBGE released the Census Data information in two main forms: the **microdata from the sample** and **aggregated data from the universe**. Microdata is a dataset in which the rows are the most detailed units to which information was collects – in this case,individual or household level. But once the sample is more sparse geographically, it does not allow for fine-grained spatial analyses. The smallest geographic units available in the microdata are **weighting areas** (Áreas de Ponderação), which are groups of contiguous census tracts ensuring statistical representativeness. For the 2010 Census, IBGE established that a weighting area should have at least 400 occupied households in the sample. In less densely populated regions, these areas cover a large territorial extension. So with the sample microdata, you gain the ability of doing individual- or household-level detailed analysis, but you loss some geographical granularity.



For privacy concerns, IBGE does not distribute microdata from the universe survey. Its information is released instead as data aggregated into census tracts. These are summary statistics (counts, proportions, and means) computed by IBGE from the short form questionnaire. The aggregated data from the universe allows then for analyses at very precise geographic levels, but prohibiting individual- and household-level analysis.

By using **sampling weights** included in the sample microdata, users can generate population estimates that accurately reflect the Brazilian entire population. As the sample was not produced using Simple Random Sampling (SRS), statistical inference requires accounting for the complex sampling design used in the census, which affects variance estimates and confidence interval calculations.

3. The 1960 Census

The microdata version available in `{censobr}` represents a combination of two distinct datasets drawn from the 1960 Demographic Census. The 1960 Census in Brazil marked a significant chapter in the nation's demographic data collection history, characterized by both methodological complexity and subsequent challenges in data processing. Initially, IBGE conducted a comprehensive 25% sample survey alongside the universe census, but technical issues delayed its processing, and several states' data remained incomplete and undigitized. The 25% sample currently available includes only 16 states of the Federation, excluding Maranhão, Piauí, Guanabara, Santa Catarina, Espírito Santo, and the Northern Region. It also contains data from a contested border region between Minas Gerais and Espírito Santo known as Serra dos Aimorés.

In the midst of the processing delay, in 1965, IBGE also created a probabilistic sub-sample representing approximately 1.27% of the population, covering all units of the Federation. This sub-sample was used to produce several official reports in the 1960s, and it remains an important source of data. Unlike the 25% sample, the 1.27% dataset is comprehensive in terms of geographic coverage, encompassing states not included in the 25% sample (which was never fully processed), albeit excluding rural areas of the state of Rondônia. We combined both the 25% and 1.27% samples to form a more complete dataset, which approximates the original design intended for the 1960 Census. The merging process involved substantial pre-processing to address data inconsistencies, especially since portions of the original 1.27% sample were corrupted, leading to missing or inaccurate information (more information here: https://github.com/antrologos/ConsistenciaCenso1960Br).

We development a crosswalk to align municipality codes from the 1960 Census with municipality names, using auxiliary documents from the IBGE's online library, with extensive manual digitization required due to the low quality of the scanned documents. Using the municipality names, we matched 1960 Census with the Brazilian Statistical Yearbook (Anuário Estatístico Brasileiro) and imported detailed information on the population totals of rural and urban areas for all municipalities and states. This allowed us to construct a sample weight variable, enabling proper population estimates and correcting for unbalanced data. For the 25% sample observations, the weights expand to the municipal level, while for the 1.27% sample observations, the expansion occurs at the state level. Additionally, to approximate the original complex sample design and allow for more accurate statistical analysis, variables representing stratification and clustering were incorporated. This feature enables the calculation of standard errors and confidence intervals that properly account for the combined sampling structure.

The availability of these combined datasets in `{censobr}` offers a unique opportunity for researchers interested in studying Brazil's demographic history during a period marked by rapid urbanization and socioeconomic transformation.

4. Installation and Basic Usage

`{censobr}` is available on CRAN, and the stable version can be installed like any other R package using the `install.packages()` function:

```r
install.packages("censobr")
```

Users interested in accessing the latest features (still in development!) can install the development version directly from GitHub using:

```r
# First, remove any existing version of censobr
utils::remove.packages("censobr")

# Install the development version from GitHub
remotes::install_github("ipeaGIT/censobr", ref = "dev")
```

After installing `{censobr}`, load it into your R session with:

```r
library(censobr)
```

The package includes several core functions to download and read different types of census data, as well as auxiliary functionalities:

Function	Description
`read_population()`	Download microdata of population records.
`read_households()`	Download microdata of household records.
`read_mortality()`	Download microdata of death records.
`read_emigration()`	Download microdata of emigration records.
`read_families()`	Download microdata of family records.
`read_tracts()`	Download census tract-level aggregated data .
`data_dictionary()`	Download and the data dictionary.
`questionnaire()`	Download the questionnaires used in the data collection.
`interview_manual()`	Download the interview manual used in the data collection.
`censobr_cache()`	Manage cached files from the censobr package.
`set_censobr_cache_dir()`	Set a custom cache directory for censobr files.

The `read_*` functions allow users to specify the year and type of data they want to access, and they return the data in a format compatible with Arrow (discussed in the next section).

To load household microdata from the 2010 census, use:

```r
households_2010 <- read_households(year = 2010)
```

To summarize the data and calculate the average household per capita income in 2010 Brazilian Reais (R$):

```r
library(dplyr)

avg_income_2010 <- households_2010 |>
        summarise(avg_income = mean(V6531, na.rm = TRUE)) |>
        collect()

print(avg_income_2010)
```

```
## # A tibble: 1 x 1
##   avg_income
##        <dbl>
## 1       781.
```

5 Census Documentation Available in {censobr}

In addition to functions for data reading, the `{censobr}` package also provides a set of functions for quick access to census documentation, including variable dictionaries, questionnaires, and interviewer manuals.

Function	Documentation Type	Availability
`data_dictionary()`	Variable dictionary	Microdata: 1960, 1970, 1980, 1991, 2000, 2010 (2022: coming soon) Aggregates: 2000, 2010 (coming soon for 2022)
`questionnaire()`	Questionnaires	1960, 1970, 1980, 1991, 2000, 2010, 2022
`interview_manual()`	Interviewer manual	1970, 1980, 1991, 2000, 2010, 2022

All documentation functions download the files in `.html` or `.pdf` format and open the document in the browser. Similar to the data reading functions of `{censobr}`, these documentation functions also save the files in a local cache the first time the function is run. Thus, when the user runs the function again, the package simply loads the local file almost instantly.

5.1. Data Dictionary

The `data_dictionary()` function loads the variable dictionary, pointing to the definition of each variable and the meaning of its categories for categorical variables. Currently, the function covers the sample microdata dictionaries for all Brazilian censuses since 1960: `c(1960, 1970, 1980, 1991, 2000, and 2010)`. Additionally, the function also includes the dictionaries for census tract-level aggregate data for the years 2000 and 2010.

```r
# dictionary of individual variables (sample microdata)
data_dictionary(year = 2010,
                dataset = 'population')

# dictionary of household variables (sample microdata)
data_dictionary(year = 2010,
                dataset = 'households')

# dictionary of census tract variables (aggregate data)
data_dictionary(year = 2010,
                dataset = 'tracts')
```

5.2. Questionnaires

Often, it is important to understand the structure and flow of the questionnaire used in data collection surveys. The `questionnaire()` function includes the questionnaires used in data collection for all Brazilian censuses since 1960.

In addition to passing the `year` parameter, the user needs to indicate the type of questionnaire of interest, whether it is the short form for the universe survey (`type = 'short'`) or the long form used in the sample survey (`type = 'long'`).

```r
# short form for the universe survey (short questionnaire)
questionnaire(year = 2022,
              type = 'short')

# long form for the sample survey (long questionnaire)
questionnaire(year = 2022,
              type = 'long')
```

5.3. Interviewer Manual

Finally, the `interview_manual()` function downloads and opens in the browser the "Manual do Recenseador," i.e., the instruction manual for IBGE enumerators on how to collect census data. Manuals for all censuses since 1960 are available.

```r
# 2010 Census
interview_manual(year = 2022)

# 1970 Census
interview_manual(year = 1960)
```

6. Handling with Larger-Than-Memory Data

6.1. In-disk Data wrangling with `{arrow}` and working with databases with `{duckdb}` and `{duckplyr}`

One of the most essential features of `{censobr}` is its capability to handle larger-than-memory datasets. The Brazilian census datasets are often too large to load directly into a user's RAM. To address this, `{censobr}` uses files saved in `.parquet` format, and by default, returns an "Arrow table" rather than a conventional `data.frame`. An Arrow table is an object that points to the dataset stored on disk, allowing for basic data manipulation without loading it into memory entirely.

Once the desired data are accessed through any `read_*` function, users can employ `{dplyr}` functions to select columns, filter cases, recode variables, or aggregate observations into larger units. Operations on Arrow tables are executed lazily; that is, they are only evaluated when explicitly requested, allowing researchers to delay heavy computations until they are necessary. After processing, smaller, more manageable datasets can be collected for advanced analysis.

To retrieve the results, users have two options:

- `collect()`: Converts the results into a regular `data.frame` in memory.
- `compute()`: Materializes the results as a new Arrow table, keeping it in Arrow format.

```r
# Read 2010 mortality data
mortality_2010 <- censobr::read_mortality(year = 2010, add_labels = 'pt', showProgress = FALSE)

# Filter deaths of men in the state of Rio de Janeiro
rio <- mortality_2010 |>
  filter(V0704 == 'Masculino' & abbrev_state == 'RJ')

# Collect the data, loading it into the memory as a data.frame
rio_df <- rio |> collect()
```

Another approach allowed by `{censobr}` is using `{duckdb}`, a library that enables Arrow tables to be queried as if they were part of a database, allowing researchers to use SQL-like syntax for data operations.

Users can register the Arrow table with `{duckdb}` and `{DBI}` to query the data:

```r
library(duckdb)
library(DBI)

# Read 2010 mortality data
mortality_2010 <- censobr::read_mortality(year = 2010, add_labels = 'pt', showProgress = FALSE)

# Create a database connection
con <- duckdb::dbConnect(duckdb::duckdb())

# Register the Arrow table in the database
duckdb::duckdb_register_arrow(con, 'mortality_2010_tbl', mortality_2010)

# Execute an SQL query to filter data
rio2 <- DBI::dbGetQuery(con, "SELECT * FROM 'mortality_2010_tbl' WHERE V0704 LIKE '%Masculino%' AND abb
```

The `{duckplyr}` package is another promising tool that extends `{duckdb}`'s capabilities, allowing `{dplyr}`-like operations directly on larger-than-memory datasets:

```r
library(duckplyr)

# Filter data using duckplyr
rio3 <- mortality_2010 |>
```

```
  duckplyr::filter(V0704 == 'Masculino' & abbrev_state == 'RJ')

# Collect the data as data.frame
rio3_df <- rio3 |> collect()
```

6.2. Data Cache

The first time a user runs a function, `{censobr}` will download the file and store it locally. This way, the data only needs to be downloaded once. When the cache parameter is set to `TRUE` (the default behavior of the package), the function will read the data already stored in the cache, which is practically instantaneous.

Users can manage cached datasets using the `censobr_cache()` function. For example, users can:

- List cached files:

```
censobr_cache(list_files = TRUE)
```

- Delete a specific file:

```
censobr_cache(delete_file = "2010_emigration")
```

- Delete all files from the cache:

```
censobr_cache(delete_file = "all")
```

By default, `{censobr}` files are saved in the 'User' directory. However, users can run the `set_censobr_cache_dir()` function to set a custom cache directory. Note that this custom setting needs to be specified at the beginning of each new R session.

```
tempf <- tempdir()

set_censobr_cache_dir(path = tempf)
```

7. Practical Applications

Here we present several use cases that illustrate the versatility of the package, supplemented by empirical examples and R code to demonstrate practical applications.

7.1. Population data: Making age pyramids

One of the key applications of census data is to analyze demographic trends over time. In this example, we use the `read_population()` function for downloading data from 1970 and 2010:

```
# Load population data for 1970 and 2010
pop_1970 <- read_population(year = 1970)
pop_2010 <- read_population(year = 2010)
```

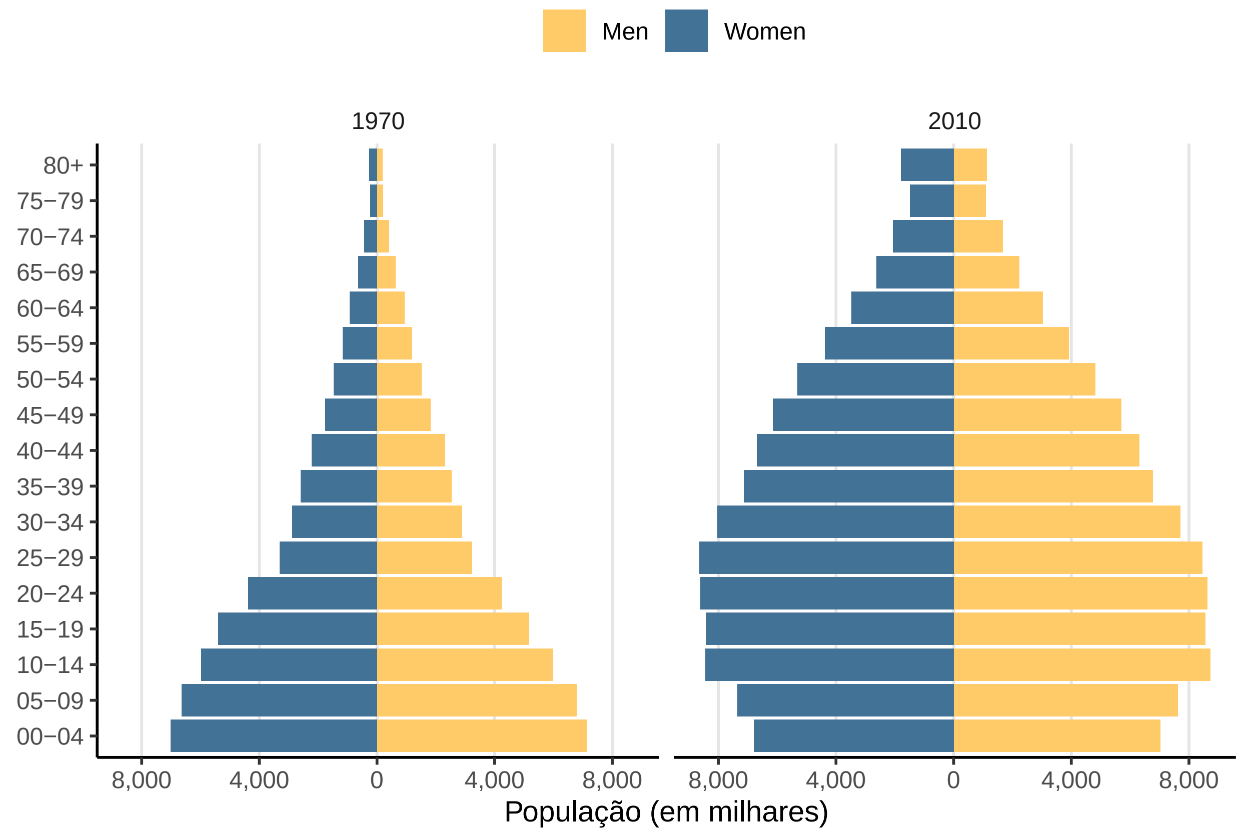
Then we recode them (still as Arrow tables), collect, and gather into a age pyramid dataset, which is ready to be plotted.

```
# Summarize age distribution
age_dist_1970 <- pop_1970 |>
        mutate(age    = ifelse(V026 %in% c(3, 4),
                               as.numeric(V027), 0),
               gender = ifelse(V023 == 0, "Men", "Women"),
               year   = 1970) |>
        group_by(age, gender, year) |>
        summarise(count = sum(V054)) |>
        collect()
```

```r
age_dist_2010 <- pop_2010 |>
        mutate(age    = as.numeric(V6036),
               gender = ifelse(V0601  == 1, "Men", "Women"),
               year   = 2010) |>
        group_by(age, gender, year) |>
        summarise(count = sum(V0010)) |>
        collect()

# Gathering, recoding, and aggregating by age groups
pyramid_df = bind_rows(age_dist_1970,
                       age_dist_2010) |>
        filter(!is.na(age)) |>
        mutate(count = ifelse(gender == "Men", count, -count),
               age_group = dplyr::case_when(
                       age <= 04            ~ "00-04",
                       age >= 05 & age <= 09 ~ "05-09",
                       age >= 10 & age <= 14 ~ "10-14",
                       age >= 15 & age <= 19 ~ "15-19",
                       age >= 20 & age <= 24 ~ "20-24",
                       age >= 25 & age <= 29 ~ "25-29",
                       age >= 30 & age <= 34 ~ "30-34",
                       age >= 35 & age <= 39 ~ "35-39",
                       age >= 40 & age <= 44 ~ "40-44",
                       age >= 45 & age <= 49 ~ "45-49",
                       age >= 50 & age <= 54 ~ "50-54",
                       age >= 55 & age <= 59 ~ "55-59",
                       age >= 60 & age <= 64 ~ "60-64",
                       age >= 65 & age <= 69 ~ "65-69",
                       age >= 70 & age <= 74 ~ "70-74",
                       age >= 75 & age <= 79 ~ "75-79",
                       age >= 80             ~ "80+")) |>
        count(year, gender, age_group, wt = count)
```

And using {ggplot2} we can plot the age pyramid for the two census years:

```r
library(ggplot2)

# Plotting the figure
pyramid_df |>
        ggplot(aes(x = n / 1000,
                   y = age_group,
                   fill = gender)) +
        geom_col() +
        scale_fill_discrete(name="", type=c("#ffcb69","#437297")) +
        scale_x_continuous(labels = function(x){scales::comma(abs(x))},
                           breaks = c(-8000, -4000,0,4000, 8000),
                           name = "População (em milhares)") +
        facet_wrap(~year) +
        theme_classic() +
        theme(legend.position = "top",
              axis.title.y = element_blank(),
              panel.grid.major.x = element_line(color = "grey90"),
              strip.background.x = element_blank())
```



7.2. Household Data: Sanitation Conditions

In this example, we will use the `read_households()` function to obtain data from the 2010 Census – and then analyze sanitation conditions in Brazil.

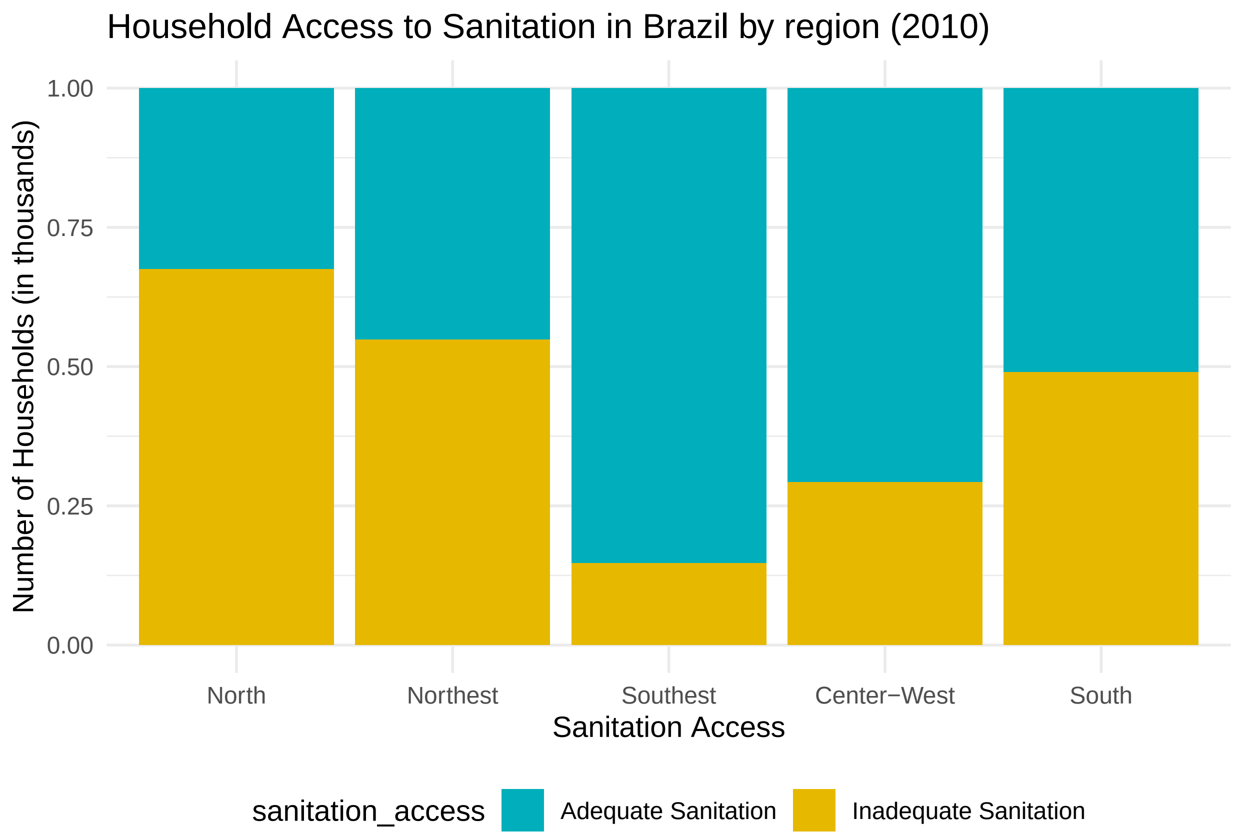
The variable `V0207` lists several types of sanitation, such as connected sewage systems or septic tanks. We recode this variable to differentiate between "Adequate Sanitation" (including connection to public sewage systems or septic tanks) and "Inadequate Sanitation" (e.g., rudimentary or no sewage treatment).

First, let's download the data and summarise it:

```r
# Load household data for 2010
households_2010 <- read_households(year = 2010)

# Summarize the number of households with and without adequate sanitation facilities
sanitation_summary <- households_2010 |>
        mutate(region = trunc(V0001/10),
               sanitation_access = case_when(V0207 %in% c(1, 2) ~ "Adequate Sanitation",
                                             TRUE ~ "Inadequate Sanitation")) |>
        group_by(region, sanitation_access) |>
        summarise(count = sum(V0010)) |> # weighted count
        group_by(region) |>
        mutate(Percentage = count/sum(count)) |>
        collect() |>
        mutate(region = factor(region,
                               levels = 1:5,
                               labels = c("North", "Northest", "Southest", "Center-West", "South"),
                                   ordered = T))
```

Using this summary table, we plot and can visualize disparities in sanitation access by region:

```
# Plotting sanitation access by region
sanitation_summary |>
  ggplot(aes(x = region, y = Percentage, fill = sanitation_access)) +
  geom_col() +
  labs(
    title = "Household Access to Sanitation in Brazil by region (2010)",
    x = "Sanitation Access",
    y = "Number of Households (in thousands)"
  ) +
  scale_fill_manual(values = c("#00AFBB", "#E7B800")) +
  theme_minimal() +
        theme(legend.position = "bottom")
```



7.3. Using Complex Sample Design in the 1960 Census

As we explained above, the 1960 Census microdata provided by `{censobr}` is an integration of two distinct datasets: a comprehensive 25% sample survey and a 1.27% probabilistic sub-sample. The dataset provide the variables for incorporating the Complex Sample Design with stratification, clustering, and weighting. Stratification ensures that different population groups are represented with certainty, while clustering addresses the correlation among observations within the same sampling units. Sampling weights, by their turn, corrects for disproportions and incorporates an expansion factor, making the counts sum up to the population totals.

In this section, we illustrate how to use data from the states of Guanabara (nowadays, Rio de Janeiro city) and Rio de Janeiro to construct a survey design object. We then estimate the distribution of the population across urban, suburban, and rural areas and calculate the confidence intervals. We start by downloading and recoding the data:

```r
# Load household data for 1960
census_1960 = read_population(year = 1960)

# Filtering cases, selecting and recoding varibles
census_rj_gb = census_1960 |>
        select(uf, Urban = V118,
               censobr_upa, censobr_usa, censobr_estrato, censobr_weight,
               censobr_source) |>
        filter(uf %in% c(52, 54)) |>
        collect() |>
        mutate(uf    = ifelse(uf == 52, "Rio de Janeiro", "Guanabara"),
               Urban = factor(Urban,
                              levels = c(1,3, 5),
                              labels = c("Urban", "Suburban", "Rural"),
                              ordered = T))

# Weighted (N) and unweighted cases (n)
census_rj_gb |>
        group_by(uf) |>
        summarise(N = sum(censobr_weight),
                  n = n())
```

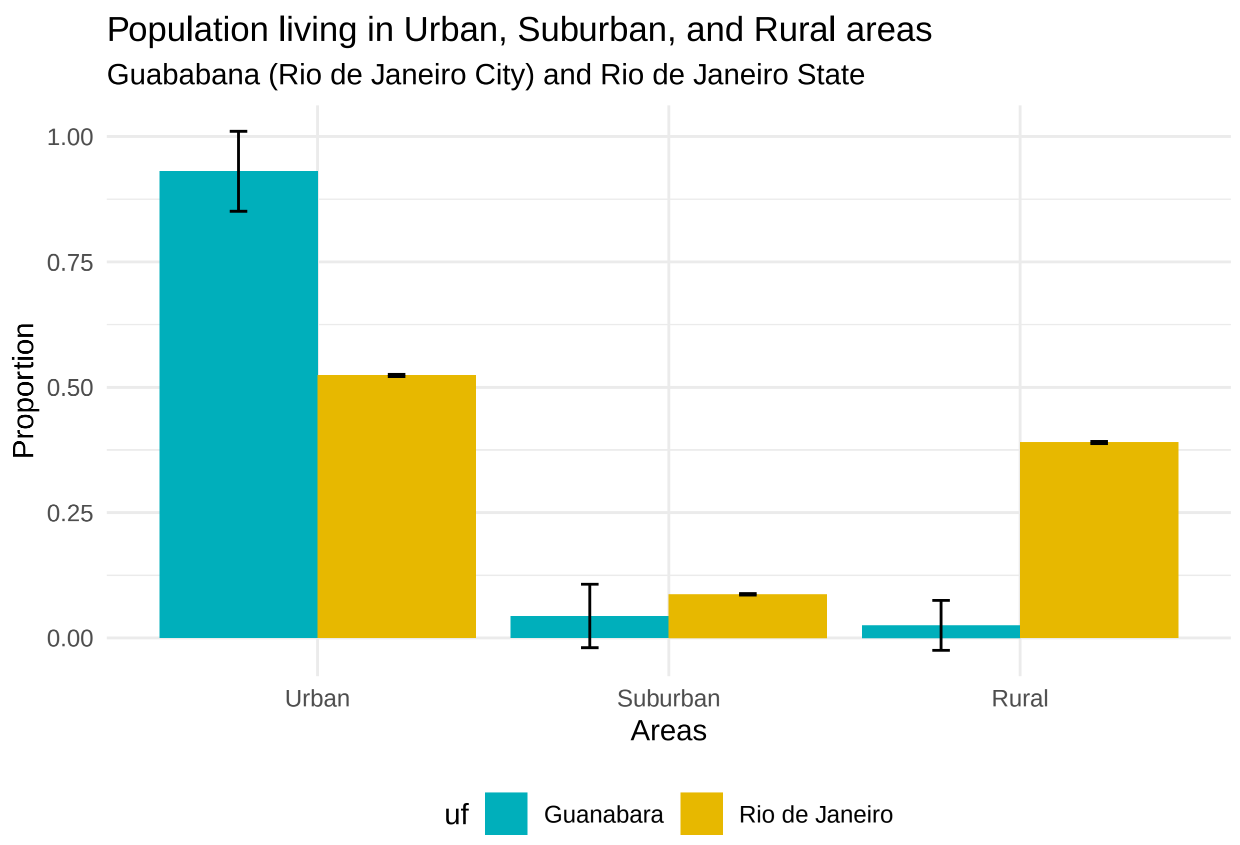
```
## # A tibble: 2 x 3
##   uf                    N      n
##   <chr>             <dbl>  <int>
## 1 Guanabara      3307163   37134
## 2 Rio de Janeiro 3402728. 878115
```

Notice that despite having very similar populations, Guanabara and Rio de Janeiro have very different sample sizes in the {censobr}'s version of the 1960 Census. Guanabara is not present in the 25% sample – only in the 1.27%. So we expect it to have larger error margins.

Now let's transform the `census_rj_gb` data.frame into a survey object usign the function `as_survey_design`, from `{srvyr}`. This package is built on top of the `{survey}`, allowing for a more accessible pipe-friendly syntax that integrated with `{dplyr}`. We inform the primary sampling units (PSUs) and secondary sampling units (SSU), strata, and weights.

By using a simple dplyr-like syntax, we can produce a contingency table with the correct confidence intervals.

```r
# Using dplyr-like syntax to manipulate the survey object
table = census_rj_gb_svy |>
        mutate(one = 1) |>
        group_by(uf, Urban) |>
        summarize(pct = survey_mean(vartype = "ci"),
                  n = unweighted(n()))

# Plotting the results
table |>
        ggplot(aes(x = Urban,
                   y = pct,
                   fill = uf,
                   ymax = pct_upp,
                   ymin = pct_low)) +
        geom_bar(stat = "identity", position = "dodge") +
        geom_errorbar(position = position_dodge(width = 0.9), width = 0.1) +
        scale_fill_manual(values = c("#00AFBB", "#E7B800")) +
        labs(
                title = "Population living in Urban, Suburban, and Rural areas",
                subtitle = "Guabanana (Rio de Janeiro City) and Rio de Janeiro State",
                x = "Areas",
                y = "Proportion") +
        theme_minimal() +
        theme(legend.position = "bottom")
```



7.4. Working with Census Tracts

As previously mentioned, IBGE does not distribute microdata from the universe survey. And as most users are interested in individual- or household-level analysis, the sample microdata often becomes the focus of attention. However, the aggregated data at the census tract level provides very rich data on population and environmental characteristics on a very detailed geographical resolution.

In its original format, these aggregated data are divided into different and separate datasets, organized by themes and types of variables (e.g., variables related to individuals, households, etc.). In many cases, the same theme is spread across multiple files (sometimes with hundreds of variables). To simplify understanding of these data, `{censobr}` consolidates all files/variables into eight tables:

- “Basico” (Basic Variables)
- “Entorno” (Household surroundings/neighborhood)
- “Domicilio” (Aggregated household information)
- “Pessoa” (Aggregated persons information)

- “Responsavel” (Aggregated information on the household heads)
- “PessoaRenda” (Aggregated information on persons’ income)
- “DomicilioRenda” (Aggregated information on households’ income)
- “ResponsavelRenda” (Aggregated information on the household heads’ income)

When variables in a table originate from different files, we added a prefix to the variable names, indicating its original IBGE source table. For instance, let us look at the “Domicilio” table. This `{censobr}` table actually comes from two separate original files: Domicilio01 and Domicilio02. Thus, the column names in this table are organized as follows:

```r
# download aggregated data for census tracts: household variables
dom <- read_tracts(year = 2010,
                   dataset = 'Domicilio')

names(dom)[c(1:12,301:306)]
```

```
##  [1] "code_tract"        "code_weighting"    "code_muni"
##  [4] "code_state"        "abbrev_state"      "name_state"
##  [7] "code_region"       "name_region"       "domicilio01_V1005"
## [10] "domicilio01_V001"  "domicilio01_V002"  "domicilio01_V003"
## [13] "domicilio02_V050"  "domicilio02_V051"  "domicilio02_V052"
## [16] "domicilio02_V053"  "domicilio02_V054"  "domicilio02_V055"
```

This organization of data aggregated by census tracts may seem confusing at first glance—and it is. However, it becomes clearer with some practical examples.

7.4.1. Spatial Distribution of Income in 2010 In this example, we will create a map of the spatial distribution of average per capita income. Information on the total number of residents in each census tract is available in the “Basico” block variable set, in the variable “V002”. The information on total income for the census tract can be found in the “DomicilioRenda” block, in the variable “V003”.

Using the code below, we can download the data and calculate the per capita income of all census tracts in Brazil. We will later filter these results to include only the tracts in Belo Horizonte.

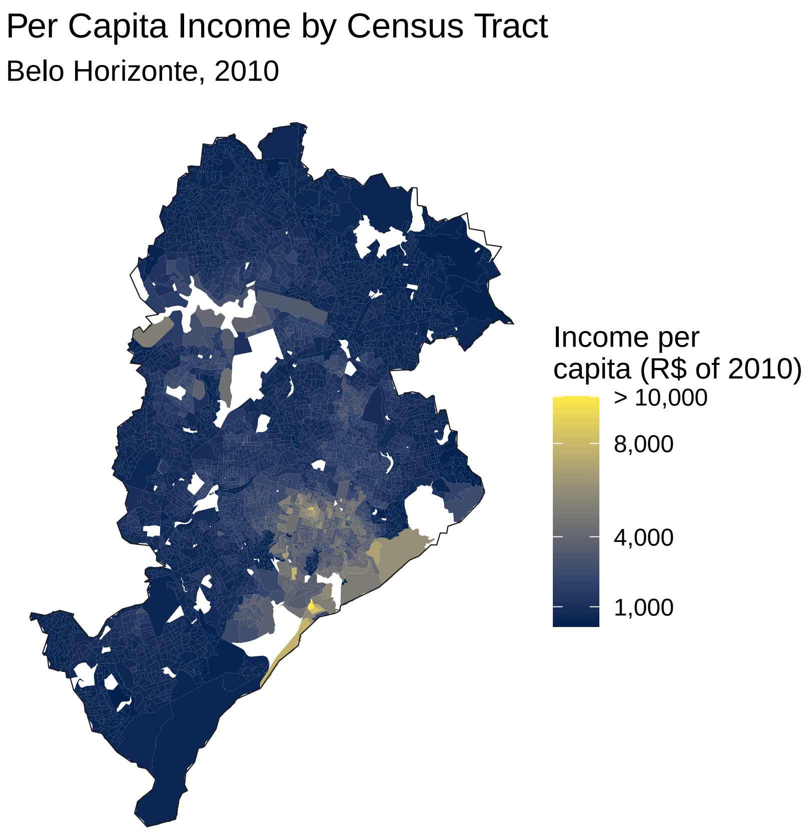
```r
# download the data
tract_basico <- read_tracts(year = 2010,
                            dataset = "Basico")

tract_income <- read_tracts(year = 2010,
                            dataset = "DomicilioRenda")

# select columns
tract_basico <- tract_basico |> select('code_tract','V002')
tract_income <- tract_income |> select('code_tract','V003')

# merge tables
tracts_df10 <- left_join(tract_basico, tract_income)

# calculate per capita income
tracts_df10 <- tracts_df10 |>
               mutate(income_pc = V003 / V002) |>
               collect()
```

The next step is to download the geometries of Belo Horizonte census tracts for 2010 using the `read_census_tract` function from the `{geobr}` package. Here, we pass the parameter `code_tract = "MG"` to download all tracts in the state of Minas Gerais and then filter only for the municipality of Belo Horizonte.

```r
library(geobr)

# download Belo Horizonte municipality
muni_bh <- read_municipality(code_muni = 3106200,
                             year = 2010)

# download all tracts in Minas Gerais
tracts_2010 <- geobr::read_census_tract(code_tract = "MG",
                                        year = 2010,
                                        simplified = FALSE,
                                        showProgress = FALSE)

# filter tracts in Belo Horizonte
tracts_2010 <- filter(tracts_2010, name_muni == 'Belo Horizonte')
```

Finally, we can merge the spatial data with the income data of the census tracts using the key variable `code_tract` and create a map of the spatial distribution of per capita income in the municipality.

```r
bh_tracts <- left_join(tracts_2010, tracts_df10, by = 'code_tract')

ggplot() +
  geom_sf(data = bh_tracts, aes(fill = ifelse(income_pc<10000,income_pc,10000)),
          color=NA) +
  geom_sf(data = muni_bh, color='gray10', fill=NA) +
  labs(title = 'Per Capita Income by Census Tract',
       subtitle= 'Belo Horizonte, 2010') +
  scale_fill_viridis_c(name = "Income per
capita (R$ of 2010)",
                       na.value="white",
                       option = 'cividis',
                       breaks = c(0,  1e3, 4e3, 8e3, 1e4),
                       labels  = c('0',  '1,000', '4,000', '8,000', '> 10,000')
                       ) +
  theme_void()
```


Per Capita Income by Census Tract
Belo Horizonte, 2010
Income per capita (R$ of 2010)
> 10,000
8,000
4,000
1,000

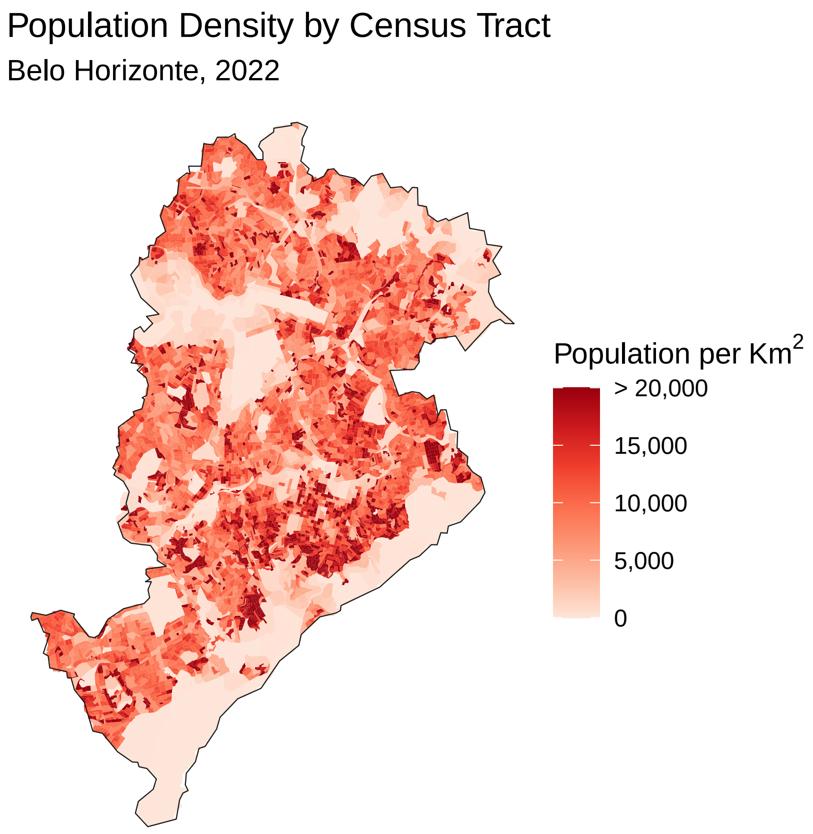
7.4.2. Spatial Distribution of Population in 2022 In this second example using the aggregated data, we will use the "Preliminares" results from the 2022 census, which were released by IBGE in March 2024. Specifically, we will use variable "V0001", which provides the total population of the tracts. Additionally, since there was a change in the spatial grid of census tracts between 2010 and 2022, we need to download the grid for the corresponding year.

```
# download preliminary data from the 2022 tracts
tracts_df22 <- read_tracts(year = 2022,
                           dataset = "Preliminares") |>
               filter(name_muni == 'Belo Horizonte') |>
               collect()

# download all tracts in Minas Gerais
tracts_2022 <- geobr::read_census_tract(code_tract = "MG",
                                        year = 2022,
                                        simplified = FALSE)

# filter tracts in Belo Horizonte
tracts_2022 <- filter(tracts_2022, name_muni == 'Belo Horizonte')
```

Now we can merge the population table with the spatial data, calculate the area of the tracts in square kilometers, and compute the population density of each tract to create the map.

```
# merge tables
tracts_df22$code_tract <- as.numeric(tracts_df22$code_tract)
bh_tracts22 <- left_join(tracts_2022, tracts_df22, by = 'code_tract')
```

```
# calculate the area of the tracts
bh_tracts22 <- bh_tracts22 |>
              mutate(tract_aream2 = sf::st_area(tracts_2022),
                     tract_areakm2 = units::set_units(tract_aream2, km2))

# calculate population density
bh_tracts22 <- bh_tracts22 |>
               mutate(pop_km2 = as.numeric(V0001/ tract_areakm2))

# map
ggplot() +
  geom_sf(data = bh_tracts22, color=NA,
          aes(fill = ifelse(pop_km2<20000,pop_km2,20000))) +
  geom_sf(data = muni_bh, color='gray10', fill=NA) +
  labs(title = 'Population Density by Census Tract',
       subtitle= 'Belo Horizonte, 2022') +
  scale_fill_distiller(palette = "Reds", direction = 1,
                       name='Population per'~Km^2,
                       breaks = c(0,  5e3, 10e3, 15e3, 2e4),
                       labels  = c('0',  '5,000', '10,000', '15,000', '> 20,000')) +
  theme_void()
```



8.