Area models for estimating poverty and labor market indicators

Andrés Gutiérrez[1], Stalyn Guerrero[2], Gabriel Nieto[3]

2023-12-14

[1]Experto Regional en Estadísticas Sociales - Comisión Económica para América Latina y el Caribe (CEPAL) - andres.gutierrez@cepal.org

[2]Consultor - Comisión Económica para América Latina y el Caribe (CEPAL) - guerrerostalyn@gmail.com

[3]Consultor - Comisión Económica para América Latina y el Caribe (CEPAL) - gabrieljose.nieto@gmail.com

Contents

List of Figures

List of Tables

Workshop material

Workshop material

In the following link you will find the R routines developed for the workshop. Download

Chapter 1

Session 1- Census and satellite information

1.1 Use of Satellite Imagery and SAE

One of the pioneering articles in small area estimation was the paper by Singh, R, et al. (2002), which addressed crop yield estimation for the tehsils (sub-administrative units) of Rohtak district in Haryana, India.

Raster images represent the world through a set of contiguous equally spaced cells known as pixels. These images contain information like a geographic information system and a coordinate reference system. Images store an identifier, a value in each pixel (or a vector with different values), and each cell is associated with a color scale.

Images can be obtained in raw and processed forms. The former contains only color layers, while the latter also contains values that have been processed in each cell (vegetation indices, light intensity, type of vegetation).

Raw information can be used to train desired features (roads, crop types, forest/non-forest). Fortunately, in Google Earth Engine, we find many processed indicators associated with a pixel. These indicators can be aggregated at a geographical area level.

1.2 Satellite Image Data Sources

Some of the main sources of satellite images include:

- USGS Earth Explorer
- Land Processes Distributed Active Archive Center (LP DAAC)
- NASA Earthdata Search

- Copernicus Open Access Hub
- AWS Public Dataset - Landsat

However, most of these sources are centralized within **Google Earth Engine**, which allows searching for satellite image data sources. GEE can be managed through APIs in different programming languages: JavaScript (by default), Python, and R (rgee package).

1.3 Google Earth Engine

Create an account at this link. Once logged in, you can search for datasets of interest:

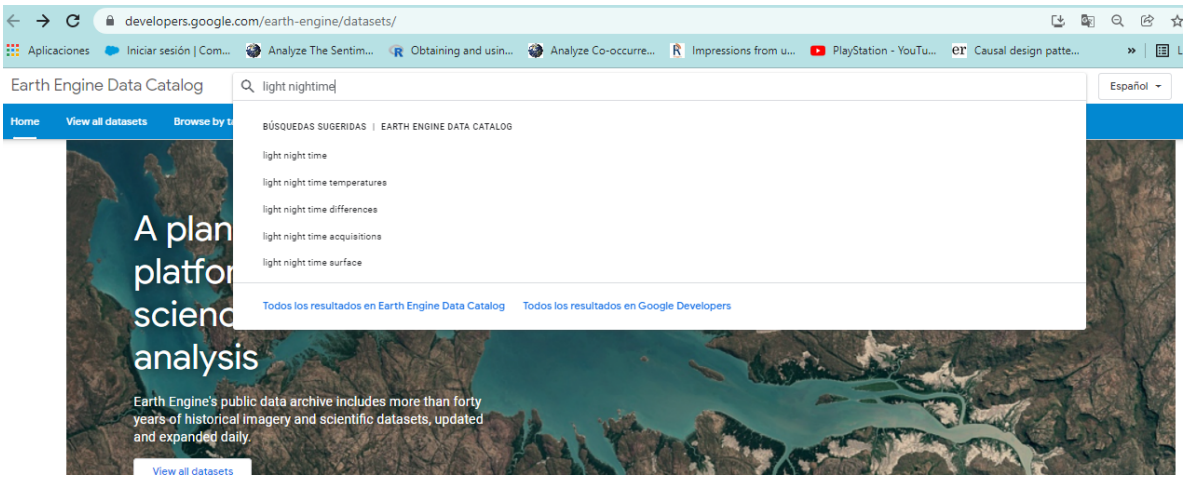


Figure 1.1: Night Lights Image

- Upon searching for the dataset, you can open a code editor provided by Google in JavaScript.
- Copy and paste the syntax provided by the dataset search to visualize the raster image and obtain statements allowing for the retrieval of the dataset of interest later in R.

1.4 Installing rgee

- Download and install Anaconda or Conda from here.
- Open **Anaconda Prompt** and set up a working environment (Python environment JAM2023) using the following commands:

```
conda env list
conda create -n JAM2023 python=3.9
```

New Script * Get Link Save

```
1  var dataset = ee.ImageCollection('NOAA/DMSP-OLS/NIGHTTIME_LIGHTS')
2                    .filter(ee.Filter.date('2010-01-01', '2010-12-31'));
3  var nighttimeLights = dataset.select('avg_vis');
4  var nighttimeLightsVis = {
5    min: 3.0,
6    max: 60.0,
7  };
8  Map.setCenter(7.82, 49.1, 4);
9  Map.addLayer(nighttimeLights, nighttimeLightsVis, 'Nighttime Lights');
10
```

Figure 1.2: Syntax in JavaScript

```
activate JAM2023
pip install google-api-python-client
pip install earthengine-api
pip install numpy
```

- List available Python environments in Anaconda Prompt:

```
conda env list
```

- Once you've identified the path of the JAM2023 environment, set it in R (**remember to change \ to /**).
- Install `reticulate` and `rgee`, load packages for spatial processing, and set up the working environment as follows:

```
library(reticulate) # Connection with Python
library(rgee) # Connection with Google Earth Engine
library(sf) # Package for handling geographic data
library(dplyr) # Package for data processing
library(magrittr)

rgee_environment_dir = "C:/Users/gnieto/Anaconda3/envs/JAM2023/python.exe"

# Set up Python (Sometimes not detected and R needs to be restarted)
reticulate::use_python(rgee_environment_dir, required=T)

rgee::ee_install_set_pyenv(py_path = rgee_environment_dir, py_env = "JAM2023")

Sys.setenv(RETICULATE_PYTHON = rgee_environment_dir)
Sys.setenv(EARTHENGINE_PYTHON = rgee_environment_dir)
```

- Once the environment is configured, you can initialize a Google Earth Engine session as follows:

```
rgee::ee_Initialize(drive = T)
```

```
── rgee 1.1.5 ──────────────────────────────────── earthengine-api 0.1.299 ──
 ✓ user: not_defined
 ✓ Google Drive credentials:  FOUND
 ✓ Initializing Google Earth Engine:*** Earth Engine *** FINAL DEADLINE: ee.Authenticate will fa
il after 2022-06-06. Please upgrade. https://developers.google.com/earth-engine/guides/python_ins
tall
 ✓ Initializing Google Earth Engine:  DONE!
 ✓ Earth Engine account: users/Stalyn
──────────────────────────────────────────────────────────────────────────────
```

Figure 1.3: Session started successfully

Notes:

- Each session must be initialized with the command `rgee::ee_Initialize(drive = T)`.
- JavaScript commands invoking methods with "." are replaced by the dollar sign ($), for example:

```
ee.ImageCollection().filterDate()  # JavaScript
ee$ImageCollection()$filterDate()  # R
```

1.4.1 Downloading Satellite Information

- **Step 1**: Have the shapefiles ready.

```
shape <- read_sf("Shapefile/JAM2_cons.shp")
plot(shape["geometry"])
```

Figure 1.4: Shapefile

- **Step 2**: Select the image file you want to process, for example, **night lights**.

```
lights <- ee$ImageCollection("NOAA/DMSP-OLS/NIGHTTIME_LIGHTS") %>%
  ee$ImageCollection$filterDate("2013-01-01", "2014-01-01") %>%
  ee$ImageCollection$map(function(x) x$select("stable_lights")) %>%
  ee$ImageCollection$toBands()
```

- **Step 3**: Download the information.

```
## Takes about 10 minutes
lights_shape <- map(unique(shape$dam2),
                    ~tryCatch(ee_extract(
                      x = lights,
                      y = shape["dam2"] %>% filter(dam2 == .x),
                      ee$Reducer$mean(),
                      sf = FALSE
                    ) %>% mutate(dam2 = .x),
                    error = function(e)data.frame(dam2 = .x)))

lights_shape %<>% bind_rows()

tba(lights_shape, cap = "Average of night lights")
```

Table 1.1: Average standardized night lights

dam2	stable_lights_mean
0101	0.9393
0102	1.6857
0103	1.6900
0201	-0.4380
0202	1.4627
0203	1.3519
0204	1.6333
0205	1.7522
0206	1.7522
0207	1.7444

Repeat the routine for:

- Soil type: **crops-coverfraction** (Percentage of crop cover) and **urban-coverfraction** (Percentage of urban cover) available at https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_Landcover_100m_Proba-V-C3_Global#description
- Travel time to the nearest hospital or clinic (**accessibility**) and travel time to the nearest hospital or clinic using non-motorized transport (**accessibility_walking_only**) information available at https://developers.google.com/earth-engine/datasets/catalog/Oxford_MAP_accessibility_to_healthcare_2019

- Human modification, considering human settlements, agriculture, transportation, mining, energy production, and electrical infrastructure. You can find satellite information at the following link: https://developers.google.com/earth-engine/datasets/catalog/CSP_HM_GlobalHumanModification#description

- **Paso 4**: Consolidate the information.

Table 1.2: Standardized satellite predictors

dam2	stable_lights_mean	crops.coverfraction_mean
0101	0.9393	-0.5459
0102	1.6857	-0.7090
0103	1.6900	-0.3571
0201	-0.4380	-0.0874
0202	1.4627	-0.6237
0203	1.3519	-0.6402
0204	1.6333	-0.5050
0205	1.7522	-0.6844
0206	1.7522	-0.4289
0207	1.7444	-0.4662

1.5 Population and Housing Censuses

It's necessary to define the variables for the country you want to work with. As a first step, access to the country's census data is required. You can access it from the following link: https://redatam.org/en/microdata, where you'll find a *.zip* file with the microdata for the country. To read this dataset, you'll need to use the *redatam.open* function from the `redatam` library. This function directly depends on the census dictionary from REDATAM software, which is a file with a .dicx extension and should be located in the same folder as the data being read. This is how an object is created within R that merges the dictionary with the microdata from the census database. After performing a process in R using REDATAM syntax, we have the following table:

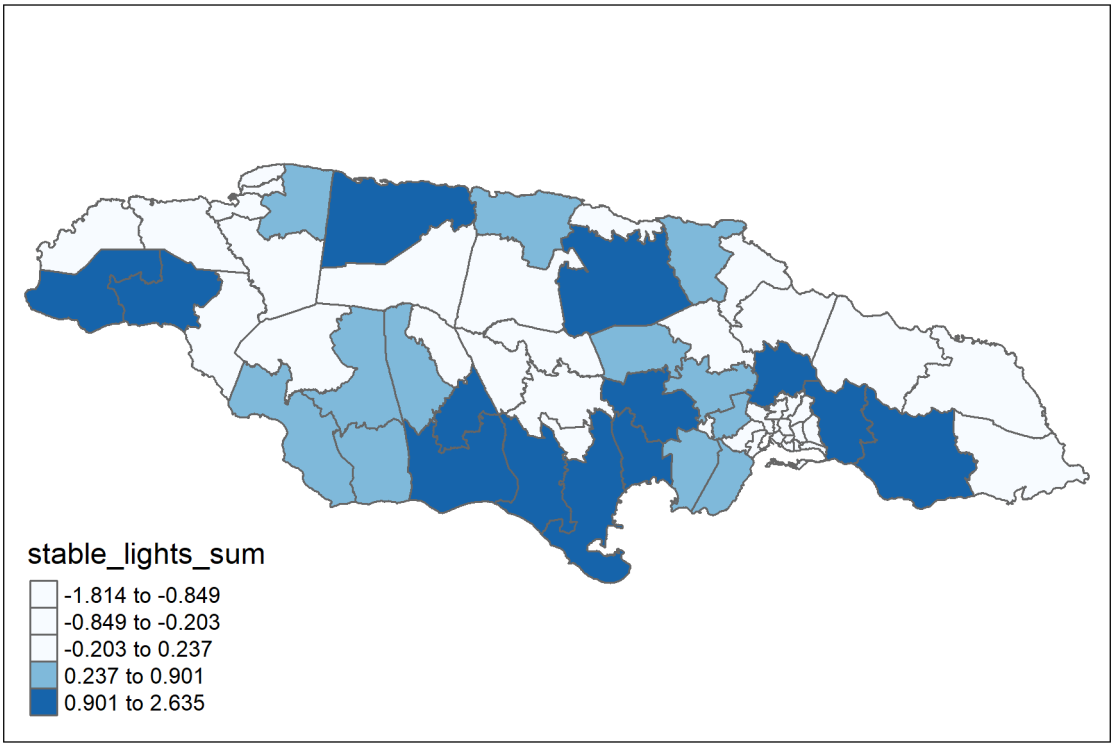


Figure 1.5: Night Lights Sum

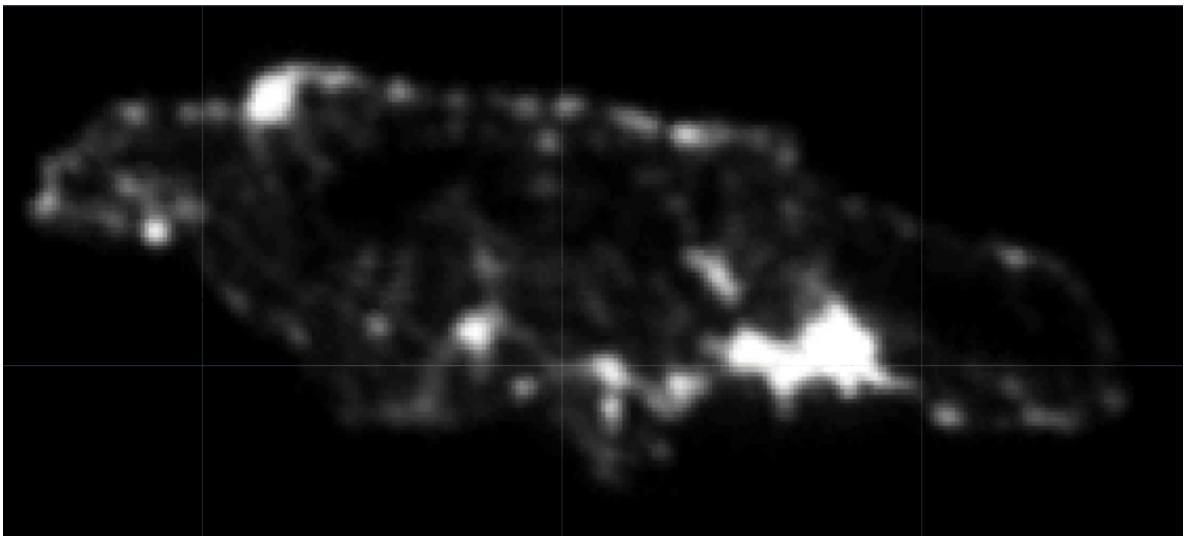


Figure 1.6: Night Lights Satellite

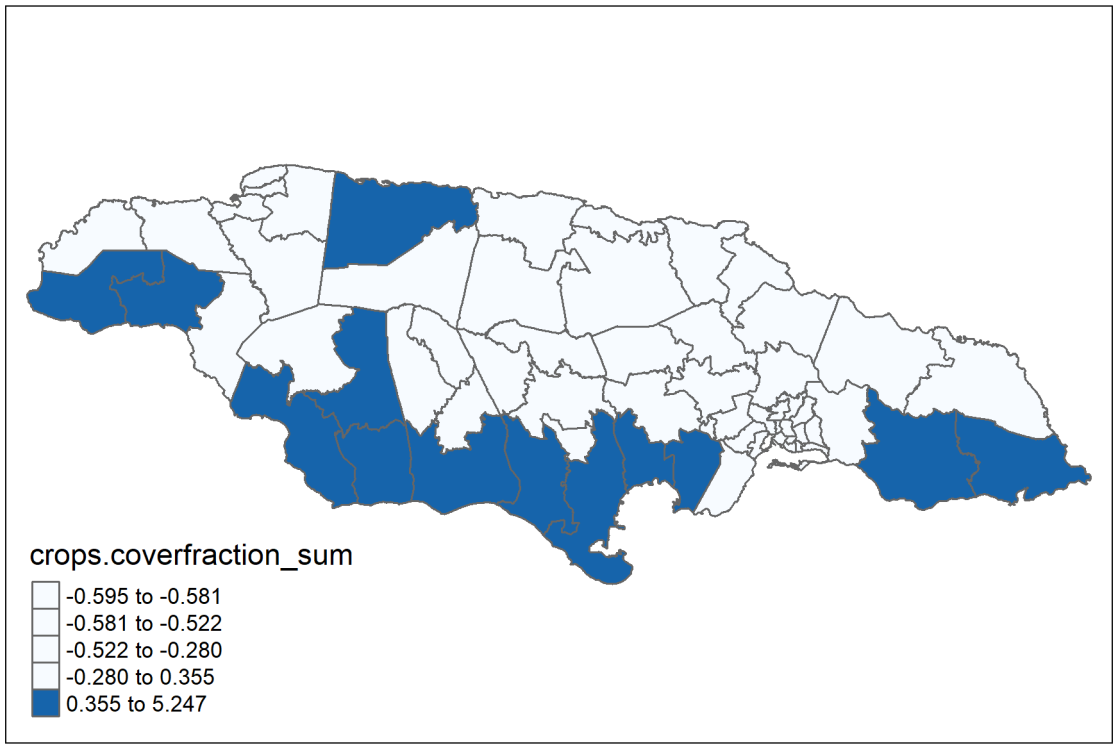


Figure 1.7: Crop Cover

Figure 1.8: Crop Cover Satellite

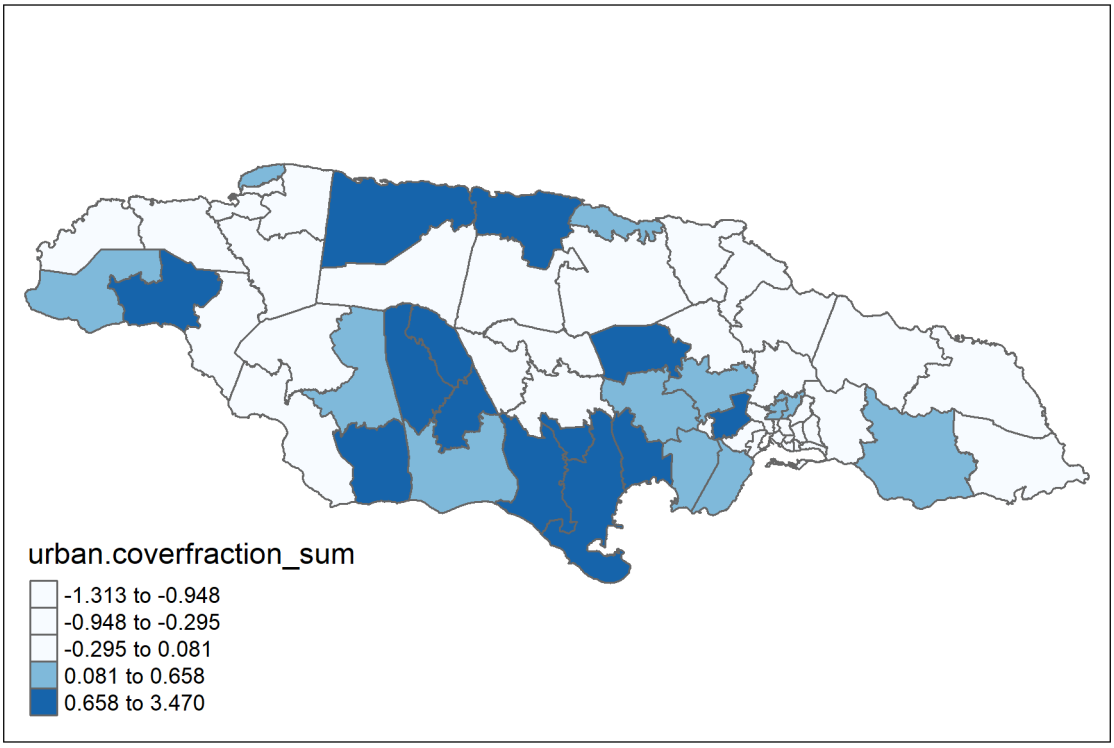


Figure 1.9: Urban Cover Sum

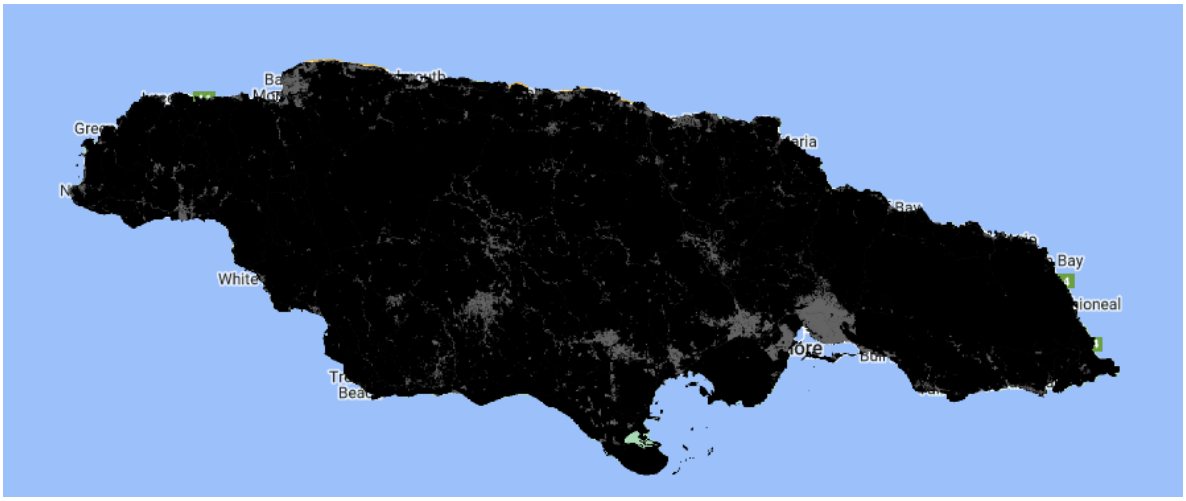


Figure 1.10: Urban Cover Satellite

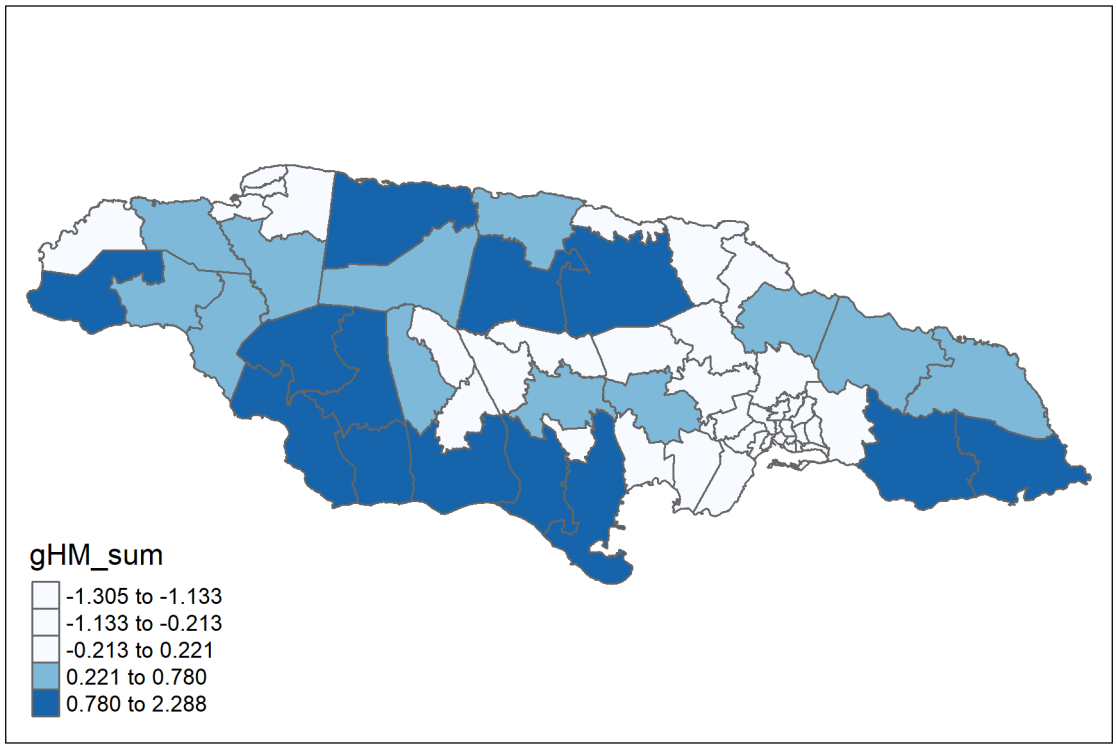


Figure 1.11: Human Modification Sum

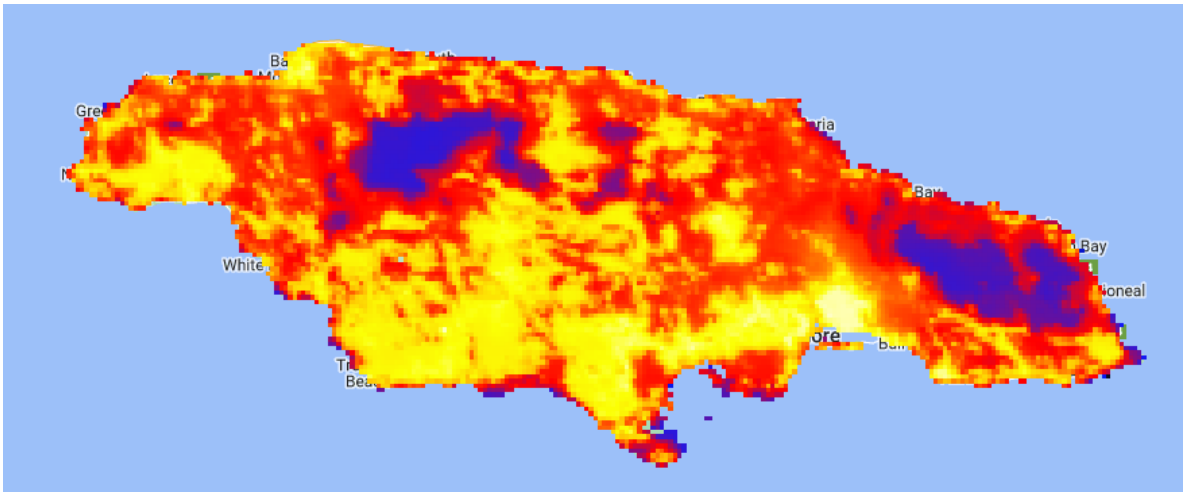


Figure 1.12: Human Modification Satellite

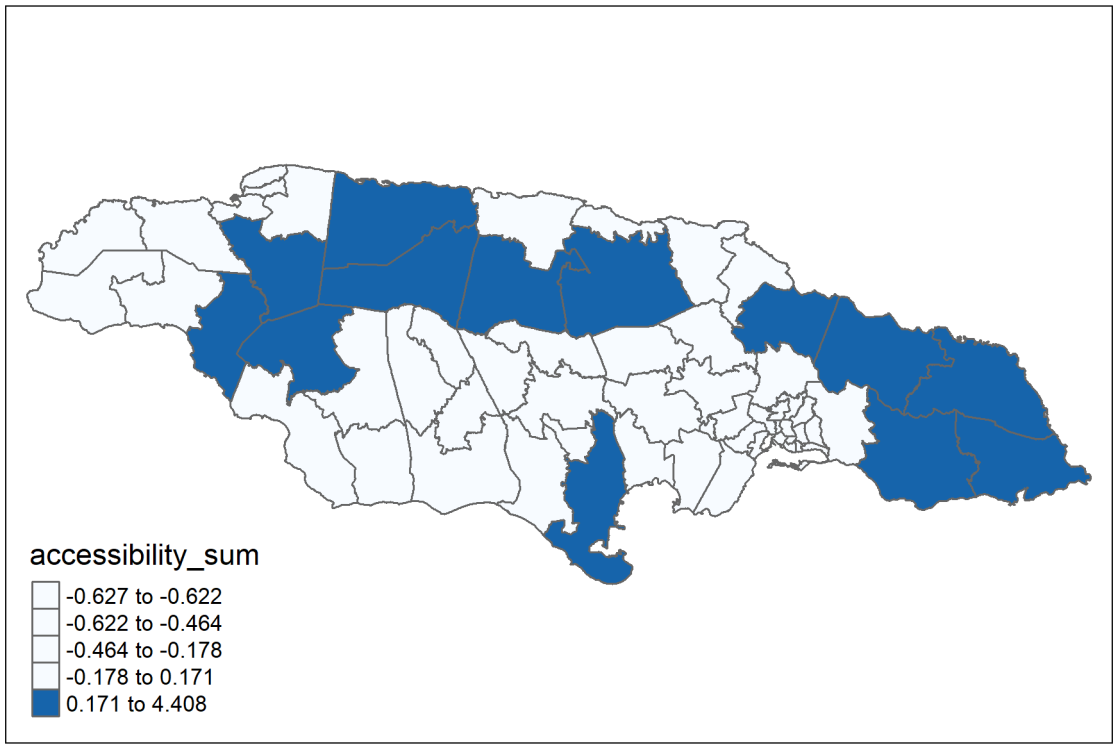


Figure 1.13: Average Travel Time to Hospital Sum

Figure 1.14: Average Travel Time to Hospital Satellite

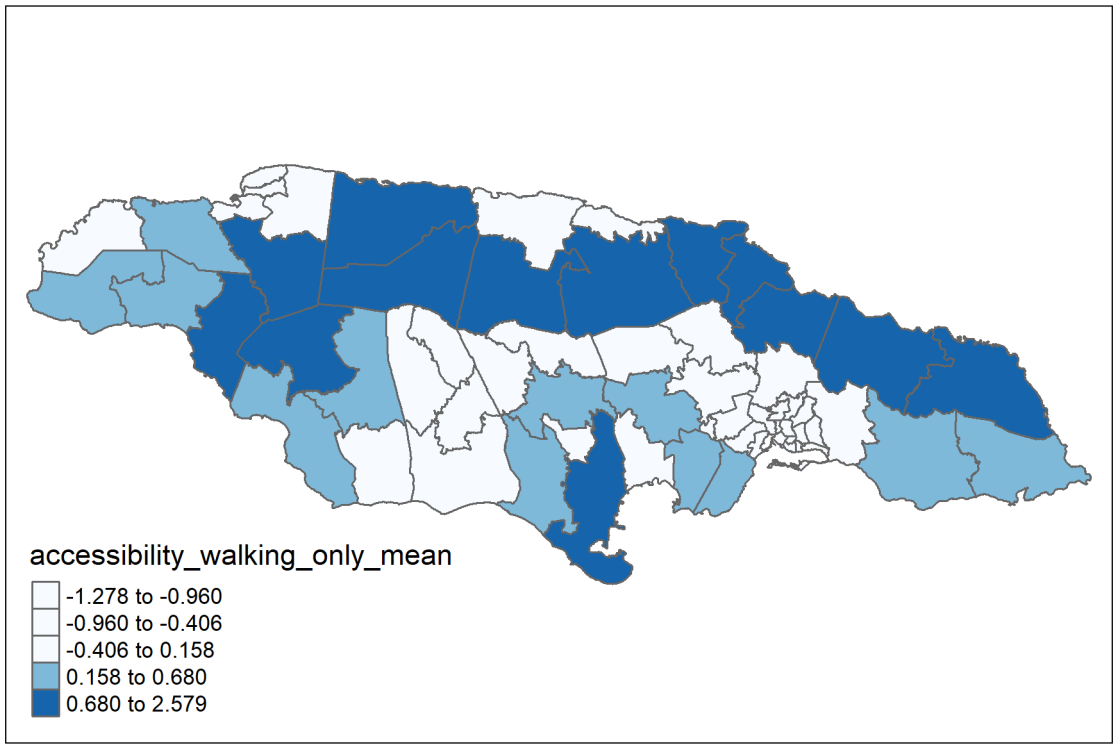
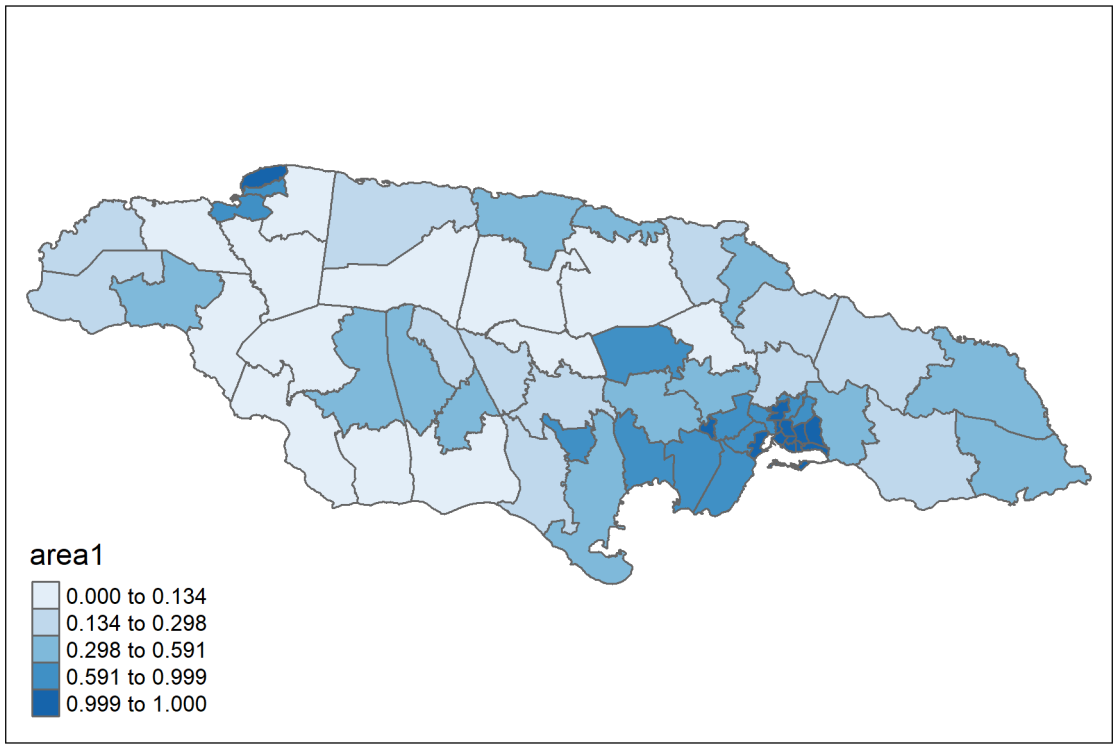


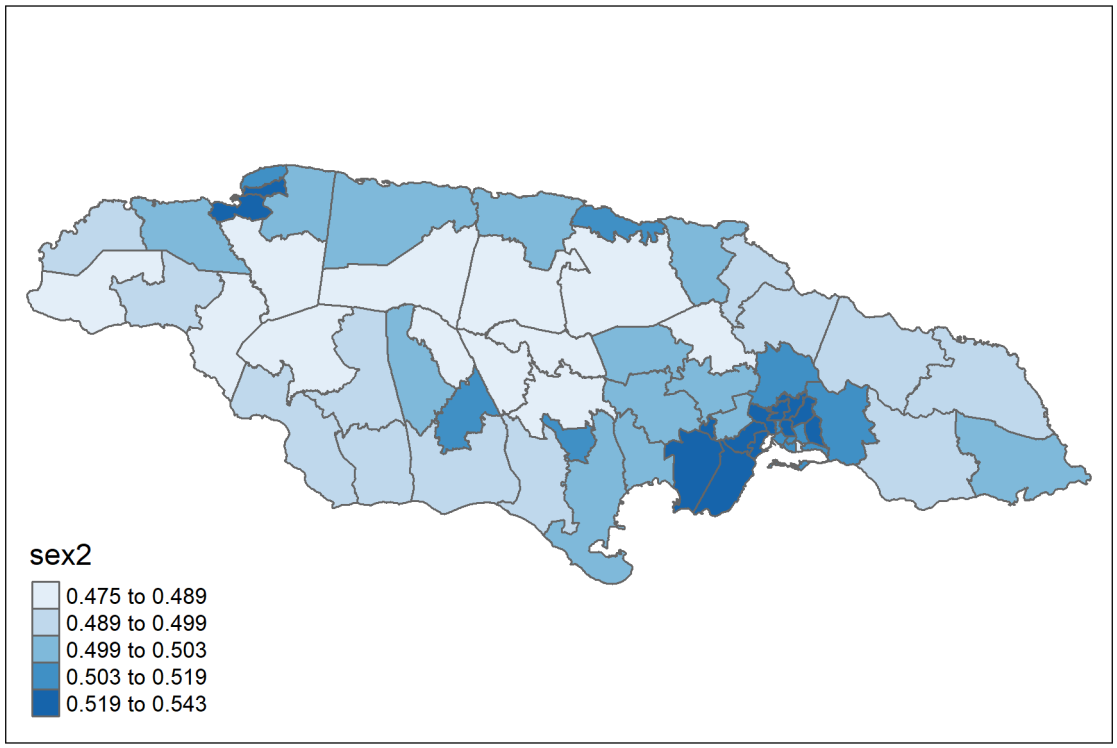
Figure 1.15: Average Travel Time to Hospital by Non-Motorized Vehicle Sum

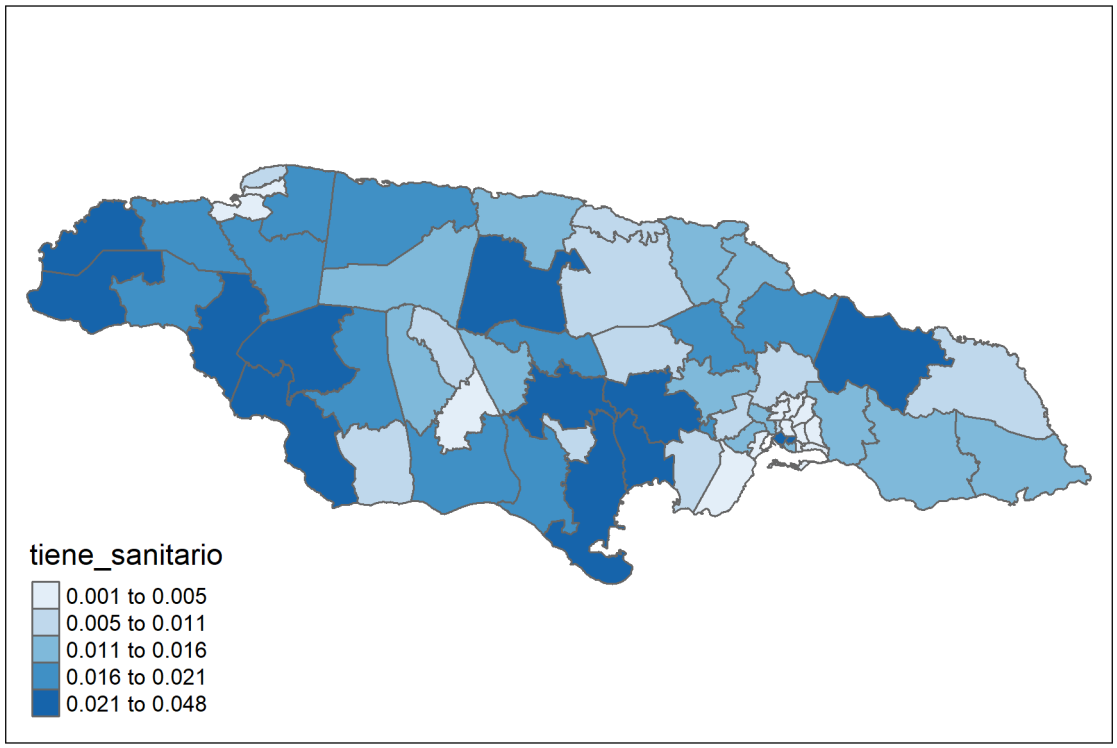
Figure 1.16: Average Travel Time to Hospital by Non-Motorized Vehicle Satellite

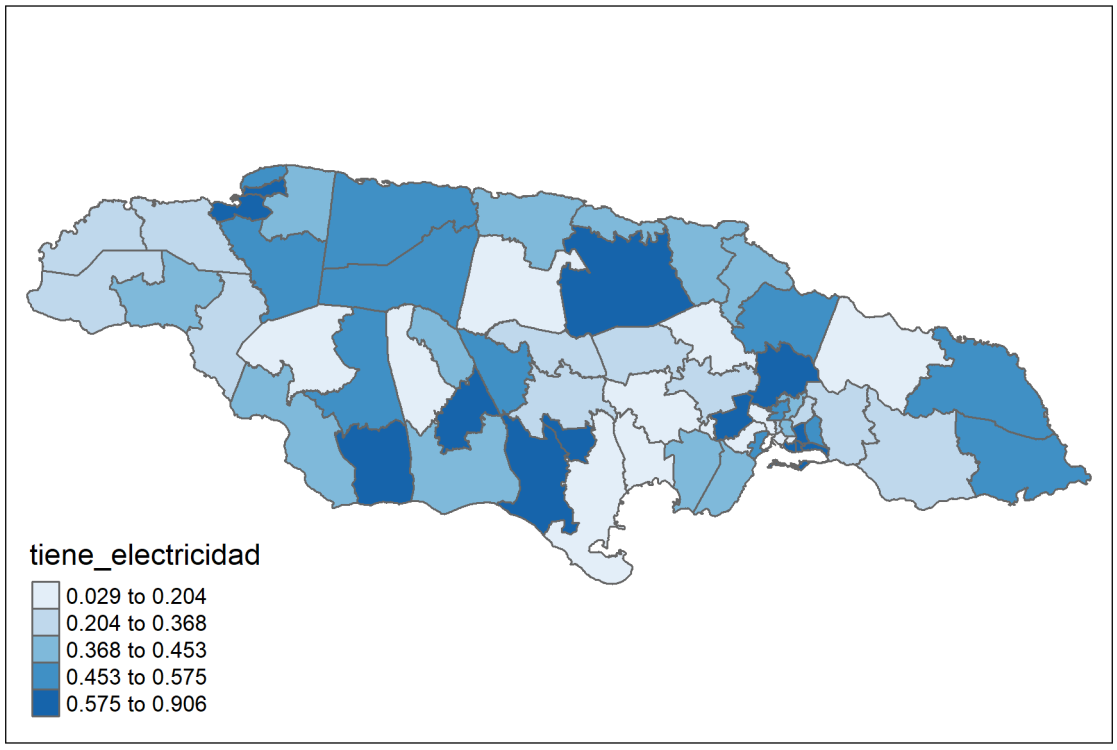
dam2	area1	sex2	age2	age3	age4
0101	1.0000	0.5087	0.2694	0.2297	0.1689
0102	1.0000	0.4754	0.2857	0.2261	0.1527
0103	1.0000	0.5037	0.3095	0.2015	0.1312
0201	0.5147	0.5060	0.2962	0.2090	0.1844
0202	0.9986	0.5376	0.2625	0.2226	0.2238
0203	0.9754	0.5432	0.2454	0.2254	0.2388
0204	1.0000	0.5300	0.3151	0.2022	0.2034
0205	1.0000	0.5182	0.3057	0.2286	0.1981
0206	1.0000	0.5157	0.3192	0.1959	0.1552
0207	1.0000	0.5097	0.3099	0.1966	0.1691

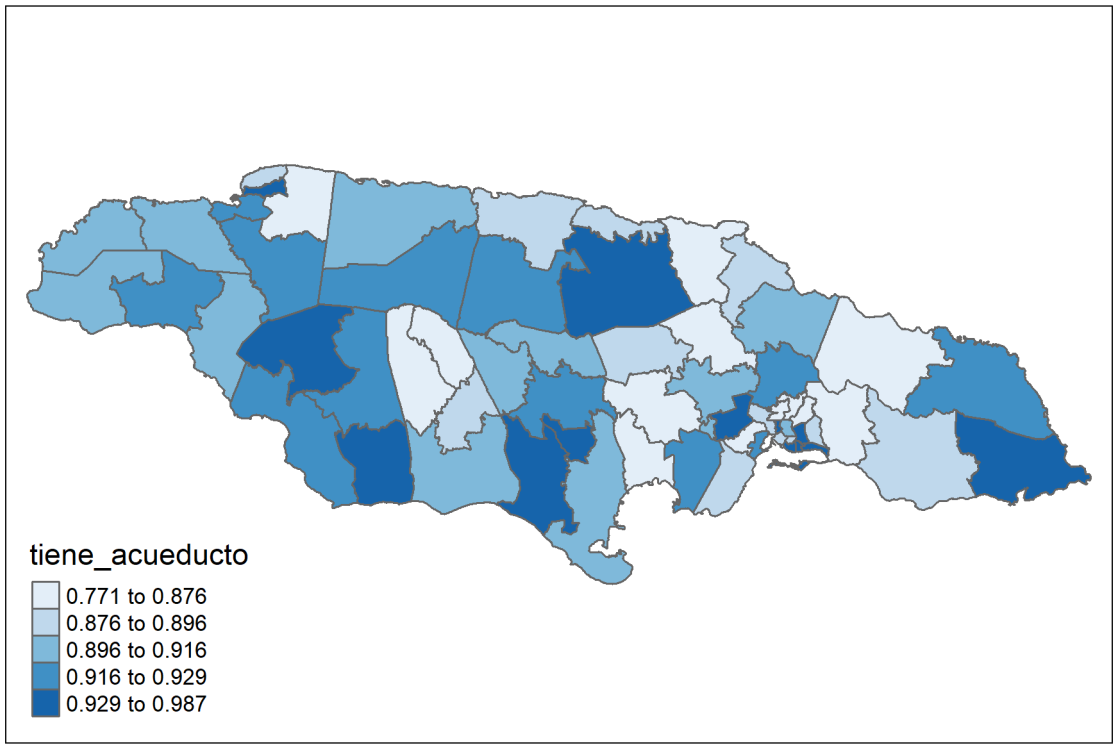
1.5.1 Mapas de las variables con información censal.

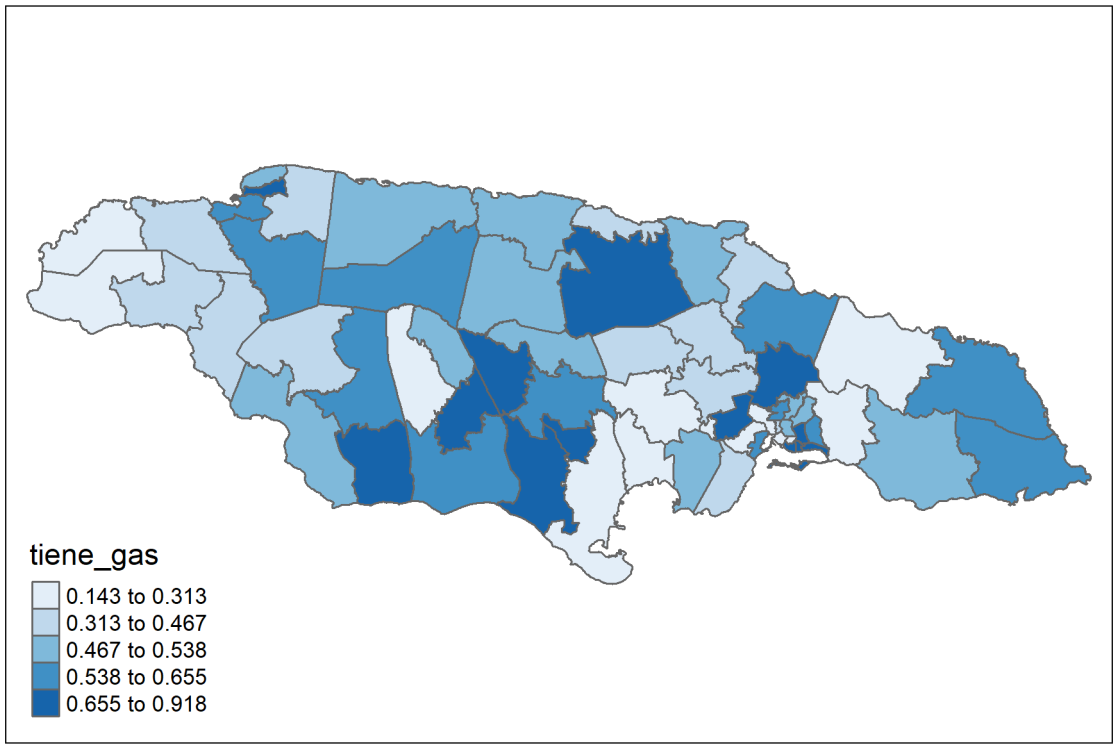


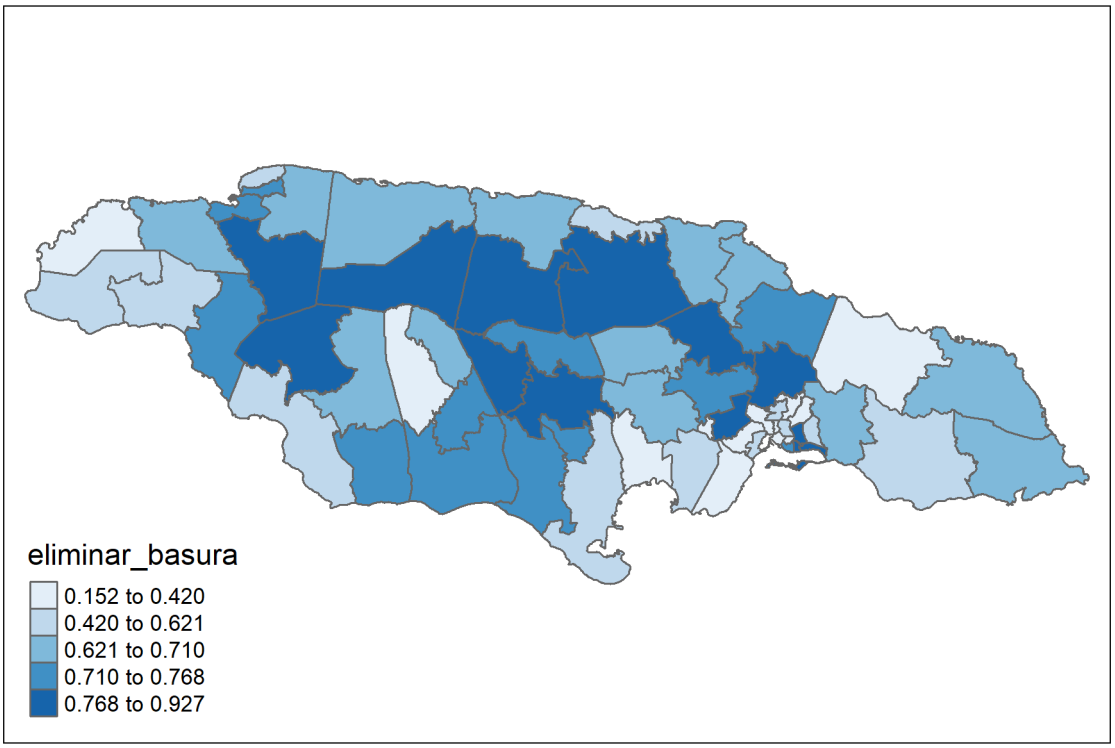


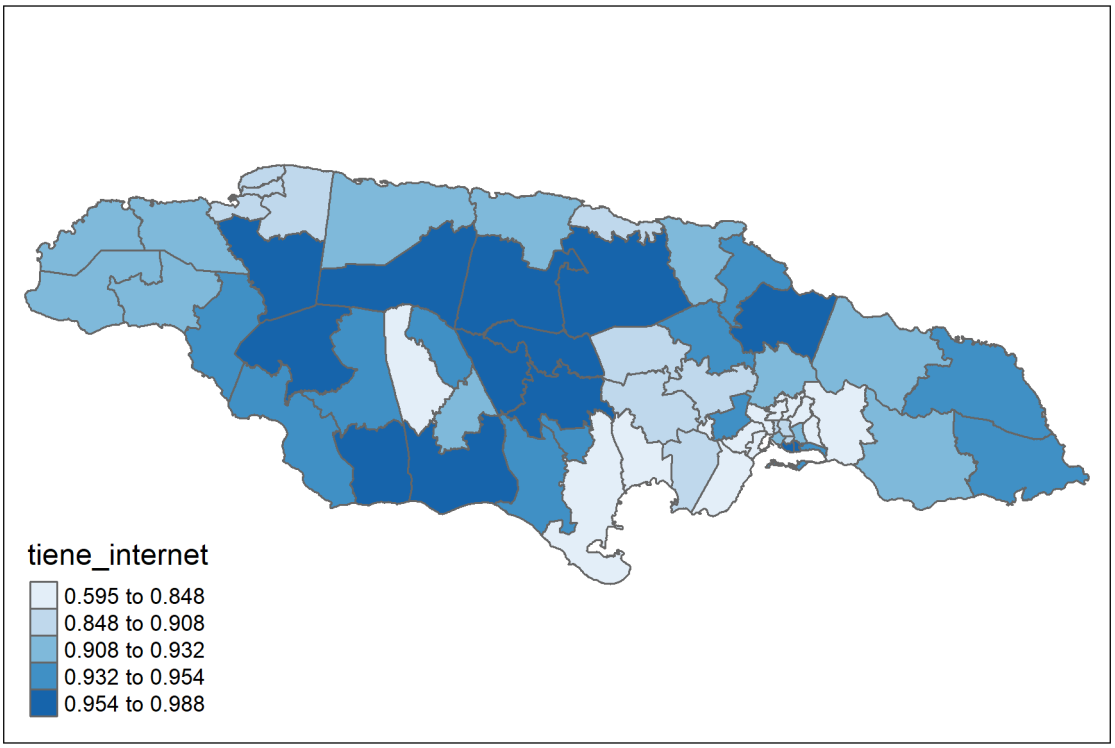

tiene_sanitario
0.001 to 0.005
0.005 to 0.011
0.011 to 0.016
0.016 to 0.021
0.021 to 0.048

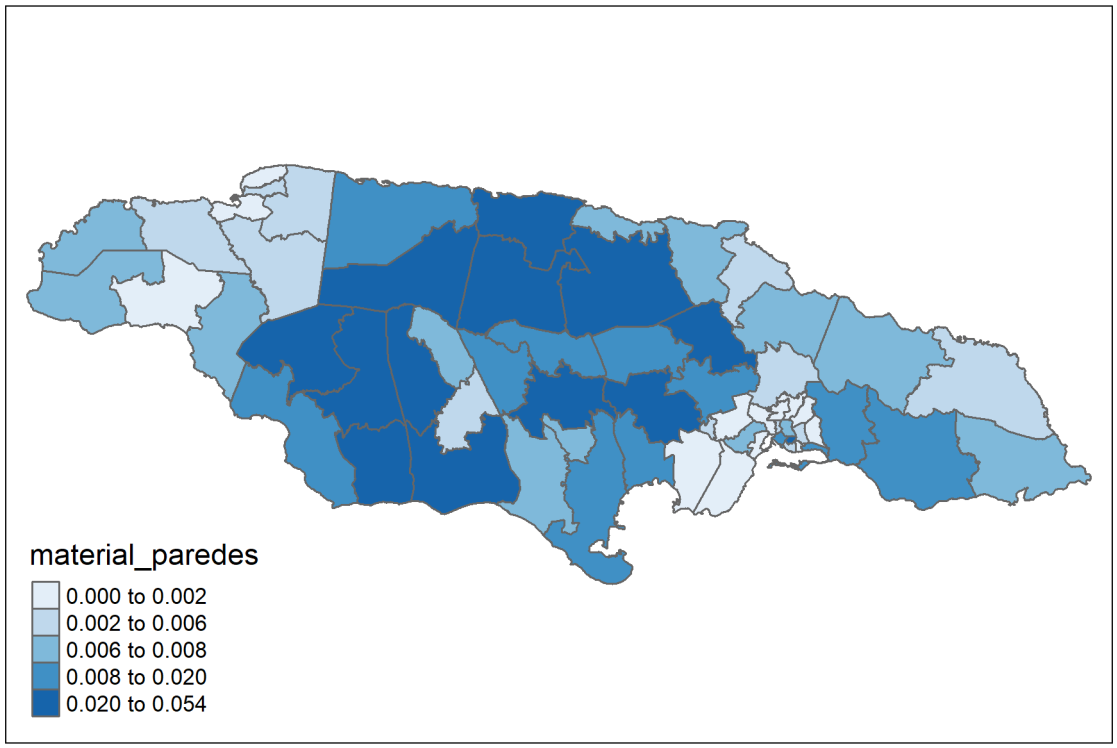

tiene_electricidad
0.029 to 0.204
0.204 to 0.368
0.368 to 0.453
0.453 to 0.575
0.575 to 0.906

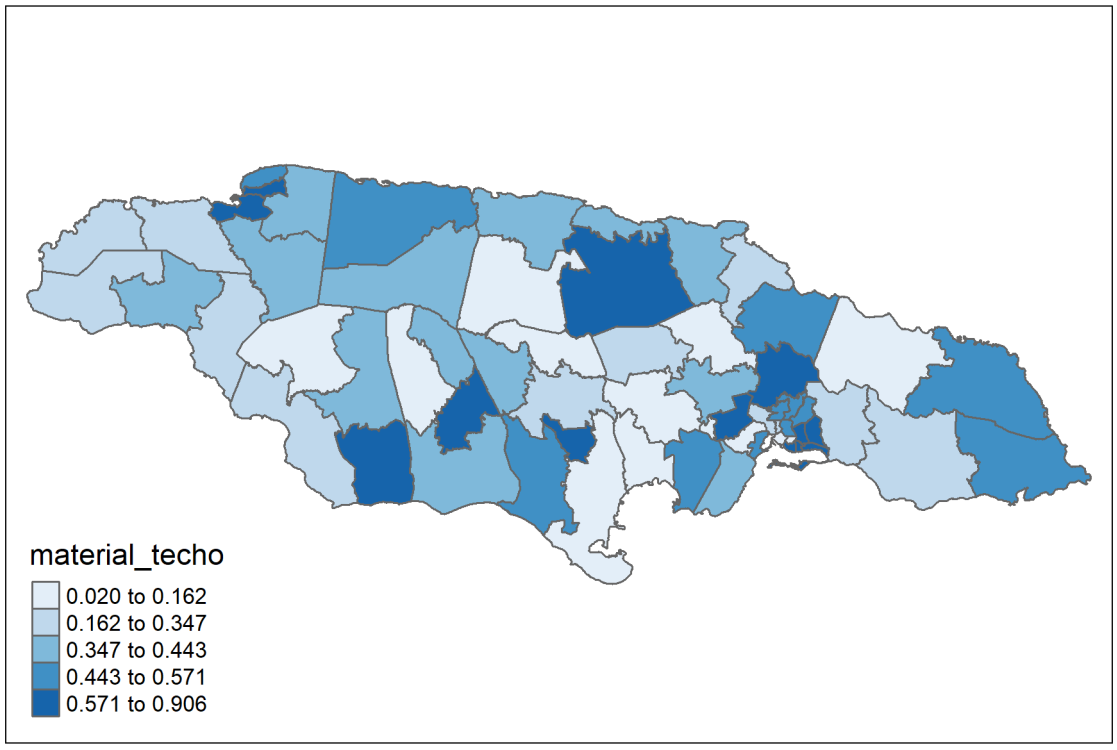

tiene_acueducto
0.771 to 0.876
0.876 to 0.896
0.896 to 0.916
0.916 to 0.929
0.929 to 0.987

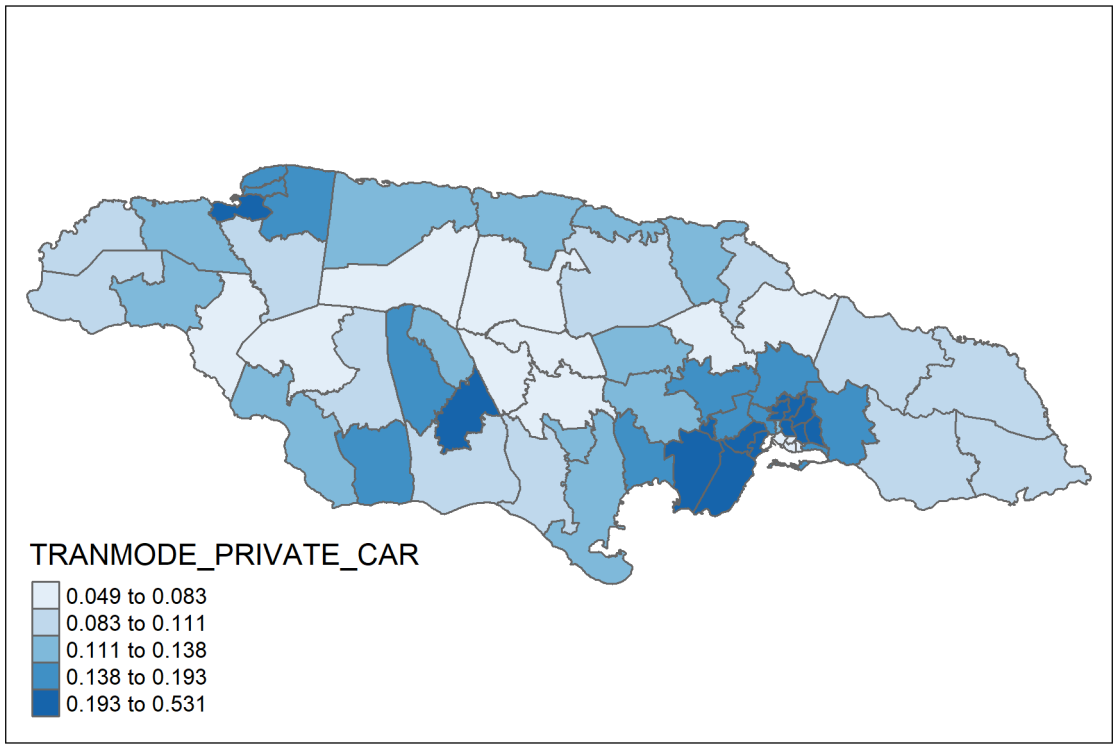

tiene_gas
0.143 to 0.313
0.313 to 0.467
0.467 to 0.538
0.538 to 0.655
0.655 to 0.918

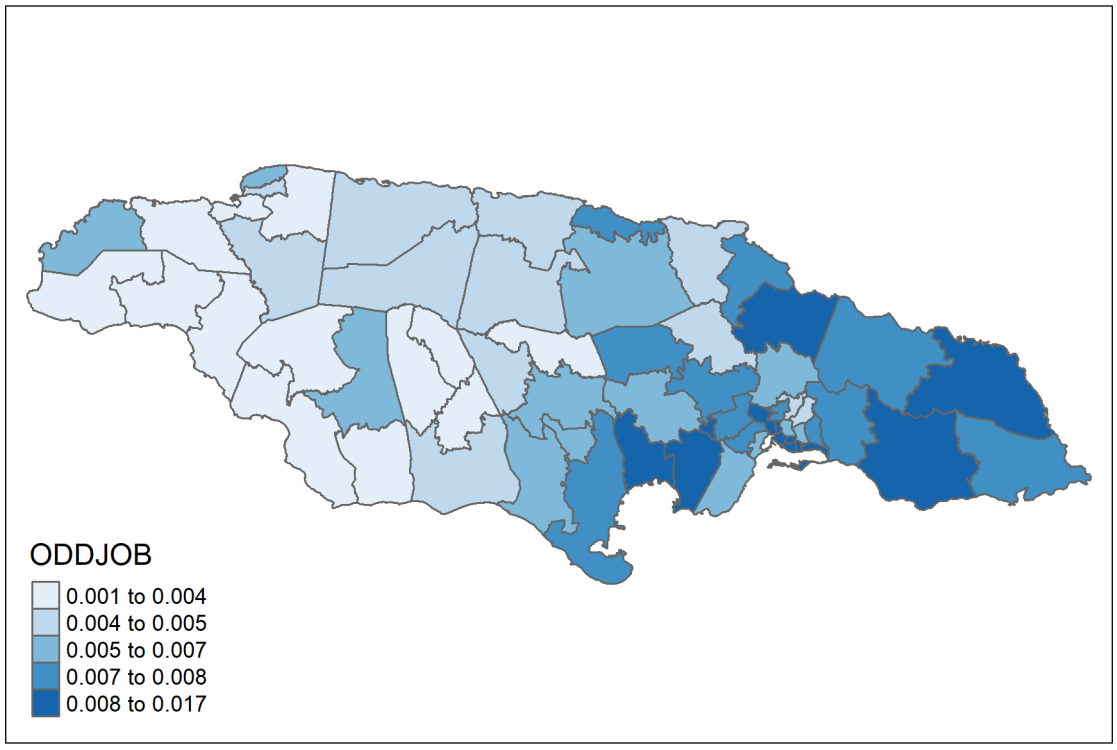

eliminar_basura
0.152 to 0.420
0.420 to 0.621
0.621 to 0.710
0.710 to 0.768
0.768 to 0.927

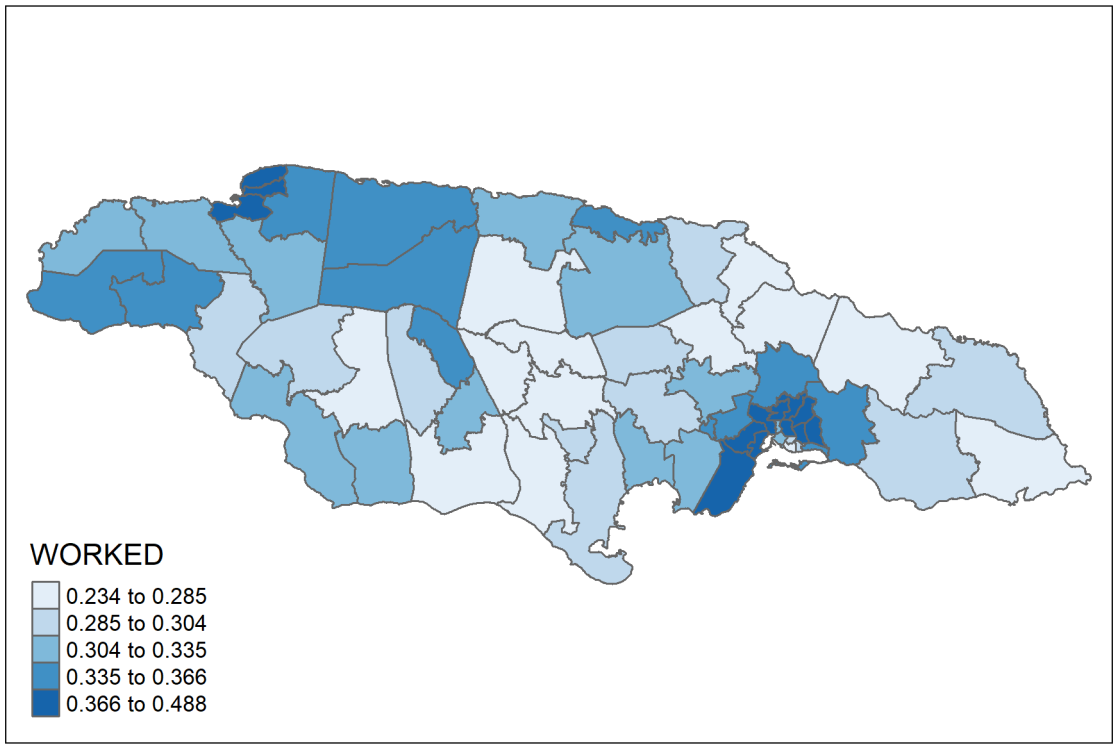

tiene_internet
0.595 to 0.848
0.848 to 0.908
0.908 to 0.932
0.932 to 0.954
0.954 to 0.988


material_paredes
0.000 to 0.002
0.002 to 0.006
0.006 to 0.008
0.008 to 0.020
0.020 to 0.054


material_techo
0.020 to 0.162
0.162 to 0.347
0.347 to 0.443
0.443 to 0.571
0.571 to 0.906


TRANMODE_PRIVATE_CAR
0.049 to 0.083
0.083 to 0.111
0.111 to 0.138
0.138 to 0.193
0.193 to 0.531


ODDJOB
0.001 to 0.004
0.004 to 0.005
0.005 to 0.007
0.007 to 0.008
0.008 to 0.017


WORKED
0.234 to 0.285
0.285 to 0.304
0.304 to 0.335
0.335 to 0.366
0.366 to 0.488

Chapter 2

Session 2- Generalized Variance Function

One of the most important inputs in the area model is the variance of the direct estimator at the domain level, which cannot be calculated directly. Accordingly, this value must be estimated from the data collected in each domain. However, in domains with very small sample sizes, these estimations will not perform well. Hence, it is very useful to use a **smoothing** model for variances to eliminate noise and volatility from these estimations and extract the true signal of the process.

Hidiroglou (2019) states that $E_{MP}\left(\hat{\theta}_d^{dir}\right) = \boldsymbol{x}_d^T\boldsymbol{\beta}$ and $V_{\mathscr{M}\mathscr{P}}\left(\hat{\theta}_d^{dir}\right) = \sigma_u^2 + \tilde{\sigma}_d^2$, where the subscript $\mathscr{M}\mathscr{P}$ refers to the double inference that must be taken into account in these adjustments and defines the joint probability measure between the model and the sampling design.

- $\mathscr{M}$ refers to the probability measure induced by modeling and the inclusion of auxiliary covariates ($\boldsymbol{x}_d$).
- $\mathscr{M}\mathscr{P}$ refers to the probability measure induced by the complex sampling design that yields direct estimations.

The solution proposed here is known as the Generalized Variance Function, which involves fitting a log-linear model to the estimated direct variance. Starting from the fact that an unbiased estimator of σ^2 denoted by $\hat{\sigma}^2$ is available, it follows that:

$$E_{\mathscr{M}\mathscr{P}}\left(\hat{\sigma}_d^2\right) = E_{\mathscr{M}}\left(E_{\mathscr{P}}\left(\hat{\sigma}_d^2\right)\right) = E_{\mathscr{M}}\left(\sigma_d^2\right) = \tilde{\sigma}_d^2$$

The above equality can be interpreted as an unbiased and simple estimator of $\tilde{\sigma}_d^2$, denoted as $\hat{\sigma}_d^2$. However, this sampling estimator is unstable when the sample size is small, which is precisely the dominant paradigm in small area estimation. Rivest and Belmonte (2000) consider smoothing models for the estimation of direct variances defined as follows:

$$\log\left(\hat{\sigma}_d^2\right) = \boldsymbol{z}_d^T\boldsymbol{\alpha} + \boldsymbol{\varepsilon}_d$$

Where $\boldsymbol{z}_d$ is a vector of explanatory covariates that are functions of $\boldsymbol{x}_d$, $\boldsymbol{\alpha}$ is a vector of parameters to be estimated, $\boldsymbol{\varepsilon}_d$ are random errors with zero mean and constant variance, assumed to be identically distributed conditionally on $\boldsymbol{z}_d$. From the above model, the smoothed estimation of the sampling variance is given by:

$$\tilde{\sigma}_d^2 = E_{\mathscr{MP}}\left(\sigma_d^2\right) = \exp\left(\boldsymbol{z}_d^T\boldsymbol{\alpha}\right) \times \Delta$$

Where $E_{\mathscr{MP}}\left(\varepsilon_d\right) = \Delta$. There's no need to specify a parametric distribution for the errors of this model. Using the method of moments, the following unbiased estimator for Δ is obtained:

$$\hat{\Delta} = \frac{\sum_{d=1}^{D}\hat{\sigma}_d^2}{\sum_{d=1}^{D}\exp\left(\boldsymbol{z}_d^T\boldsymbol{\alpha}\right)}$$

Similarly, using standard procedures in linear regression, the estimation of the regression parameter coefficients is given by the following expression:

$$\hat{\boldsymbol{\alpha}} = \left(\sum_{d=1}^{D}\boldsymbol{z}_d\boldsymbol{z}_d^T\right)^{-1}\sum_{d=1}^{D}\boldsymbol{z}_d\log\left(\hat{\sigma}_d^2\right)$$

Finally, the smoothed estimator of the sampling variance is defined as:

$$\hat{\tilde{\sigma}}_d^2 = \exp\left(\boldsymbol{z}_d^T\hat{\boldsymbol{\alpha}}\right)\hat{\Delta}$$

Survey Data:

The following code processes data using various R packages such as `survey`, `tidyverse`, `srvyr`, `TeachingSampling`, and `haven`.

1. **Library Loading:** The code loads the necessary libraries (`survey`, `tidyverse`, `srvyr`, `TeachingSampling`, `haven`) required for data manipulation and analysis.
2. **Survey Data Set Reading:** The code reads the dataset named 'data_JAM.rds' using the `readRDS` function.
3. **Data Manipulation:**
   - It creates new variables (`dam2`, `fep`, `upm`, `estrato`, `ingreso`, `pobreza`) using the `mutate` function from the `dplyr` package.
   - Filters the dataset based on a specific condition using the `filter` function.

```r
library(survey)
library(tidyverse)
library(srvyr)
library(TeachingSampling)
library(haven)

#read in data set

# Q518A: Gross average income From Employment

encuesta <- read_rds("Recursos/02_FGV/01_data_JAM.rds")

encuesta <-
  encuesta %>%
  mutate(
    dam2,
    fep = RFACT/4,
    upm = paste0(PAR_COD , CONST_NUMBER, ED_NUMBER),
    estrato = ifelse(is.na(STRATA) ,strata,STRATA),
    ingreso = case_when(!is.na(Q518A) & !is.na(Q518B) ~ Q518A + Q518B,
                        !is.na(Q518A) & !is.na(Q518B) ~ NA_real_,
                        !is.na(Q518A) & is.na(Q518B) ~ Q518A,
                        is.na(Q518A) & !is.na(Q518B) ~ Q518B),
    pobreza = ifelse(ingreso < 50,1,0)
  ) %>% filter(ingreso > 11)
```

dam2: Corresponds to the code assigned to the country's second administrative division.

The income definition is structured in this manner to illustrate the process utilized in the small area estimation methodology for poverty estimation.

ID	QUARTER	EMPSTATUS	STRATA	CLUSTER	CONST_NUMBER
02010105902170101	3	5	014	014011	01
02020207900130103	3	5	235	235472	02
02020404901820105	3	5	239	239488	04
04010400700370102	3	5	036	036073	04
04030406400160106	3	5	221	221453	04
05020506600550101	3	5	164	164352	05
06030106300920101	3	5	122	122227	01
06030200300140102	3	5	228	228464	02
06030204100960101	3	5	084	084180	02
06030208501550102	3	5	234	234474	02

In the following code block, the libraries `survey` and `srvyr` are used to create a sampling design from a survey database. The sampling design encompasses information about primary sampling units (PSUs), sampling weights (wkx), and strata (estrato) utilized in the sampling. Additionally, the "survey.lonely.psu" option is employed to adjust sample sizes within groups of primary sampling units that lack other primary sampling units within the same group.

```
library(survey)
library(srvyr)
options(survey.lonely.psu = "adjust")
id_dominio <- "dam2"

diseno <-
  as_survey_design(
    ids = upm,
    weights = fep,
    strata = estrato,
    nest = TRUE,
    .data = encuesta
  )

#summary(diseno)
```

´

1. **Indicator Calculation:**
    - Groups the sampling design by domain ID and calculates indicators related to poverty (`n_pobreza` and `pobreza`). `n_pobreza` counts the number of instances where `pobreza` equals 1 (indicating poverty), while `pobreza` computes the survey mean of the `pobreza` variable with specific variance estimations.

```
# Calculating indicators related to poverty and counts of UPMS per domain

# Indicator Calculation:
# Grouping the sampling design by domain ID and summarizing variables
# related to poverty.
indicador_dam <-
  diseno %>% group_by_at(id_dominio) %>%
  summarise(
    n_pobreza = unweighted(sum(pobreza == 1)),
    pobreza = survey_mean(pobreza,
                          vartype = c("se",  "var"),
                          deff = T
    )
```

```
  )
```

2. **Counts of UPMS per Domain:**
   - Extracts domain ID and UPMs from the survey dataset, obtaining unique UPMs per domain and counting them.
   - Joins the count of unique UPMs per domain with the previously calculated indicators related to poverty.

```
# Counts of UPMS per domain:
# Selecting domain ID and UPMs, obtaining unique UPMs per domain,
# and counting them.
# Joining the count of unique UPMs per domain with previously calculated
# indicators related to poverty.
indicador_dam <- encuesta %>% select(id_dominio, upm) %>%
  distinct() %>%
  group_by_at(id_dominio) %>%
  tally(name = "n_upm") %>%
  inner_join(indicador_dam, by = id_dominio)
# saveRDS(directodam2, "Recursos/02_FGV/indicador_dam.Rds")
```

dam2	n_upm	n_pobreza	pobreza	pobreza_se	pobreza_var	pobreza_deff
0101	17	237	0.9980	0.0021	0.0000	5.228000e-01
0102	12	197	0.9836	0.0089	0.0001	1.053700e+00
0103	9	65	1.0000	0.0000	0.0000	3.065785e+32
0201	11	244	1.0000	0.0000	0.0000	NaN
0202	11	141	0.9391	0.0292	0.0009	2.317700e+00
0203	12	101	0.8117	0.0775	0.0060	4.795200e+00
0204	11	224	1.0000	0.0000	0.0000	NaN
0205	9	85	0.9646	0.0331	0.0011	2.932600e+00
0206	8	102	1.0000	0.0000	0.0000	NaN
0207	11	149	1.0000	0.0000	0.0000	NaN

Domains with 5 or more UPMs and all those with a deff greater than 1 are now filtered.

```
base_sae <- indicador_dam %>%
  filter(n_upm >= 5,
         pobreza_deff > 1)
```

Next, the transformation $\log(\hat{\sigma}_d^2)$ is performed. Additionally, the selection of the municipality identifier columns (`dam2`), the direct estimation (`pobreza`), the number of people in the domain (`nd`), and the estimated variance for the direct estimation (`vardir`) is carried out, the latter being transformed using the `log()` function.

```
baseFGV <-  base_sae %>%
  select(dam2, pobreza, nd = n_pobreza, vardir = pobreza_var) %>%
  mutate(ln_sigma2 = log(vardir))
```

2.1 Graphical Analysis

The first graph, `p1`, displays a scatter plot of the variable `ln_sigma2` against the variable `pobreza`, with a smooth line representing a trend estimation. The x-axis is labeled as *pobreza*.

The second graph, `p2`, exhibits a scatter plot of the variable `ln_sigma2` against the variable `nd`, with a smooth line indicating a trend estimation. The x-axis is labeled as *Tamaño de muestra* (Sample Size).

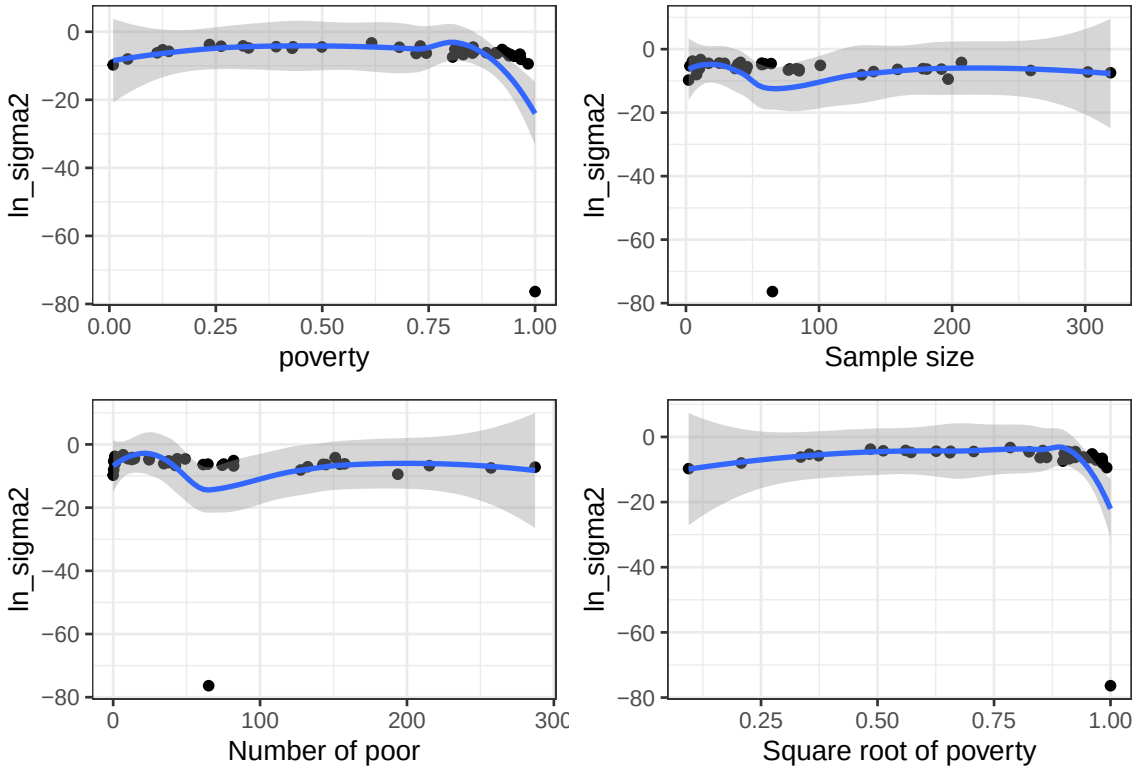
The third graph, `p3`, demonstrates a scatter plot of the variable `ln_sigma2` in relation to the product of `pobreza` and `nd`, with a smooth line representing a trend estimation. The x-axis is labeled as *Número de pobres* (Number of Poor).

The fourth graph, `p4`, shows a scatter plot of the variable `ln_sigma2` against the square root of the variable `pobreza`, with a smooth line representing a trend estimation. The x-axis is labeled as *Raiz cuadrada de pobreza* (Square Root of Poverty).

Overall, these graphs are designed to explore the relationship between `ln_sigma2` and different independent variables such as `pobreza`, `nd`, and the square root of poverty. Choosing to use the "loess" function to smooth the lines instead of a straight line aids in visualizing general trends in the data more effectively.

```
theme_set(theme_bw())

# pobreza vs Ln_sigma2 #
p1 <- ggplot(baseFGV, aes(x = pobreza, y = ln_sigma2)) +
  geom_point() +
  geom_smooth(method = "loess") +
  xlab("poverty")

# Sample Size vs Ln_sigma2 #
p2 <- ggplot(baseFGV, aes(x = nd, y = ln_sigma2)) +
  geom_point() +
  geom_smooth(method = "loess") +
  xlab("Sample size")

# Number of Poor vs Ln_sigma2 #
p3 <- ggplot(baseFGV, aes(x = pobreza * nd, y = ln_sigma2)) +
  geom_point() +
```

```
  geom_smooth(method = "loess") +
  xlab("Number of poor")

# Square Root of Poverty vs Ln_sigma2 #
p4 <- ggplot(baseFGV, aes(x = sqrt(pobreza), y = ln_sigma2)) +
  geom_point() +
  geom_smooth(method = "loess") +
  xlab("Square root of poverty")

library(patchwork)
(p1 | p2) / (p3 | p4)
```



2.2 Variance Model

The code fits a multiple linear regression model (using the `lm()` function), where `ln_sigma2` is the response variable and the predictor variables include `pobreza`, `nd`, and various transformations of these variables. The goal of this model is to estimate the generalized variance function (FGV) for the observed domains.

```
library(gtsummary)
FGV1 <- lm(ln_sigma2 ~ -1 +  pobreza +
             I(pobreza*nd),
     data = baseFGV)

tbl_regression(FGV1) %>%
  add_glance_table(include = c(r.squared, adj.r.squared))
```

Characteristic	**Beta**	**95% CI**	**p-value**
pobreza	-12	-20, -4.5	0.003
I(pobreza * nd)	0.02	-0.04, 0.07	0.5
R^2	0.345		
Adjusted R^2	0.311		

After obtaining the model estimation, the value of the constant Δ must be obtained, for which the following code is used.

```
delta.hat = sum(baseFGV$vardir) /
  sum(exp(fitted.values(FGV1)))
```

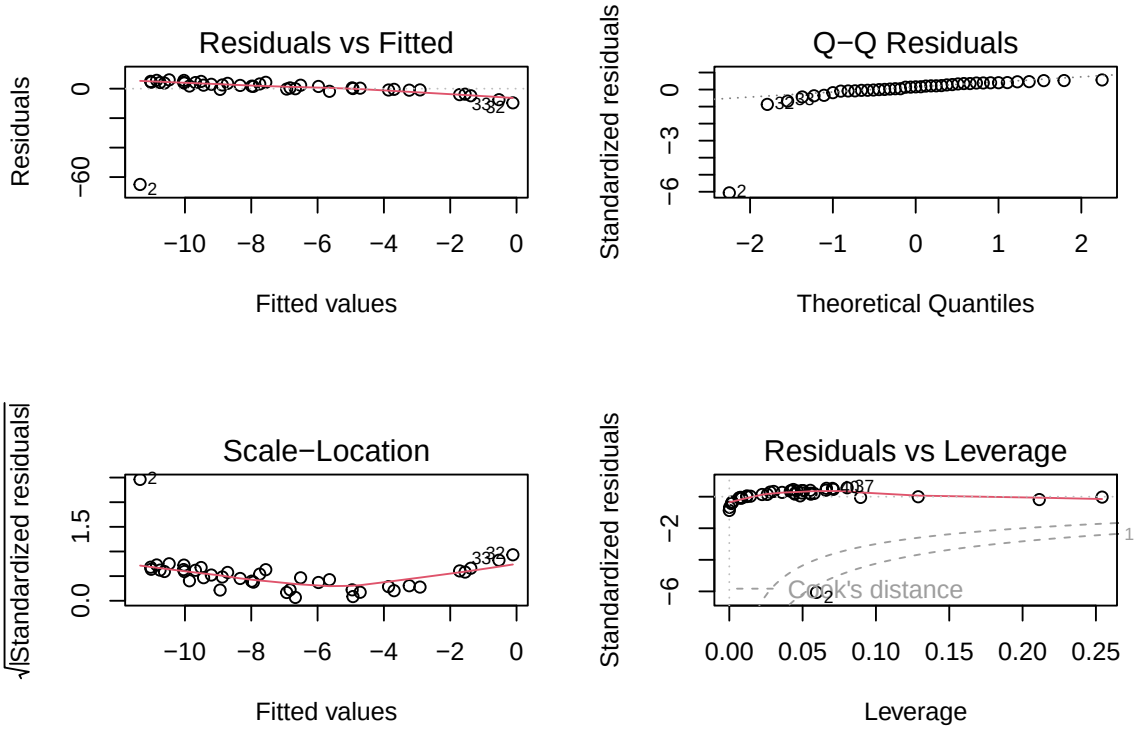
From which it is derived that $\Delta = 0.110434$. Finally, it is possible to obtain the smoothed variance by executing the following command.

```
hat.sigma <-
  data.frame(dam2 = baseFGV$dam2,
             hat_var = delta.hat * exp(fitted.values(FGV1)))

baseFGV <- left_join(baseFGV, hat.sigma)
tba(head(baseFGV, 10))
```

dam2	pobreza	nd	vardir	ln_sigma2	hat_var
0102	0.9836	197	0.0001	-9.4372	0e+00
0103	1.0000	65	0.0000	-76.3620	0e+00
0202	0.9391	141	0.0009	-7.0701	0e+00
0203	0.8117	101	0.0060	-5.1159	0e+00
0205	0.9646	85	0.0011	-6.8149	0e+00
0212	0.8304	59	0.0101	-4.5936	0e+00
0301	0.9419	45	0.0013	-6.6569	0e+00
0302	0.9109	159	0.0017	-6.3681	0e+00
0401	0.8063	319	0.0006	-7.4369	4e-04
0402	0.8311	259	0.0012	-6.7172	1e-04

Model validation for the FGV

```
par(mfrow = c(2, 2))
plot(FGV1)
```



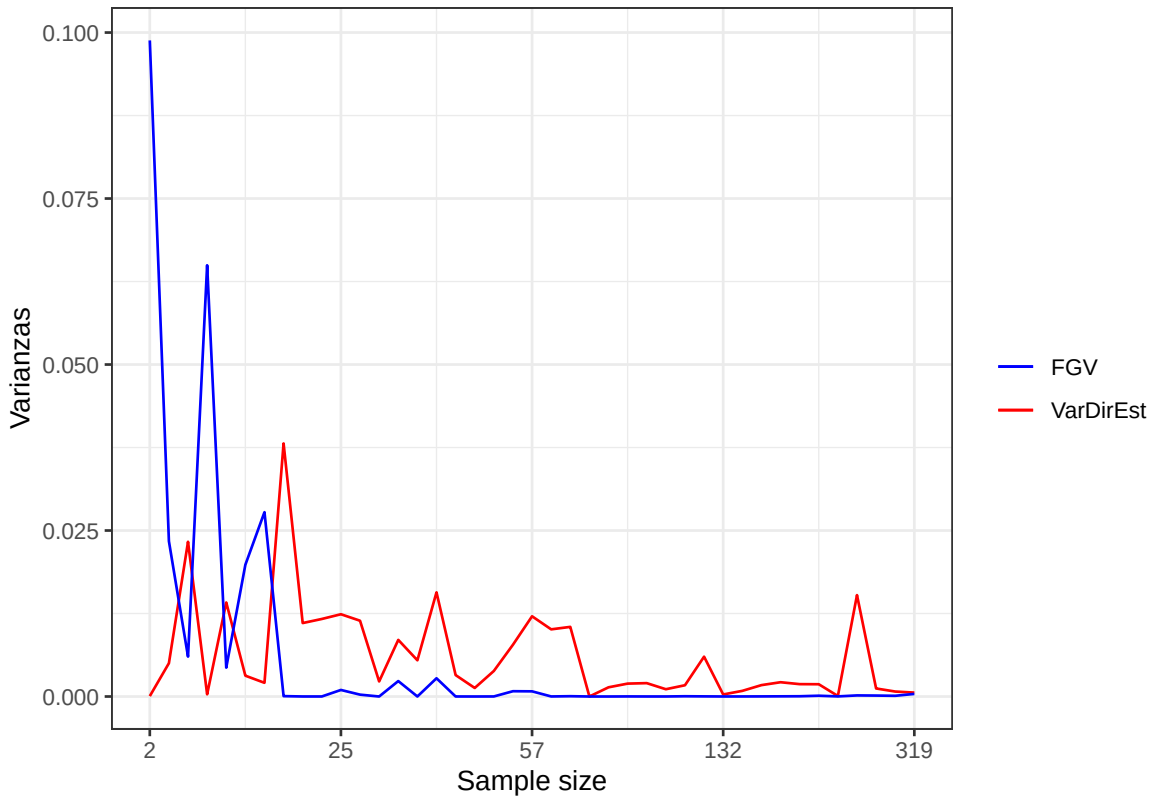
Smoothed variance prediction

```
base_sae <- left_join(indicador_dam,
                      baseFGV %>% select(id_dominio, hat_var),
                      by = id_dominio) %>%
  mutate(
    pobreza_var = ifelse(is.na(hat_var), NA_real_, pobreza_var),
    pobreza_deff = ifelse(is.na(hat_var), NA_real_, pobreza_deff)
  )
```

Now, we make a line graph to see the volatility and the estimates of the variances.

```
nDom <- sum(!is.na(base_sae$hat_var))
temp_FH <- base_sae %>% filter(!is.na(hat_var))
ggplot(temp_FH %>%
         arrange(n_pobreza), aes(x = 1:nDom)) +
  geom_line(aes(y = pobreza_var, color = "VarDirEst")) +
  geom_line(aes(y = hat_var, color = "FGV")) +
  labs(y = "Varianzas", x = "Sample size", color = " ") +
```

```
  scale_x_continuous(breaks = seq(1, nDom, by = 10),
labels = temp_FH$n_pobreza[order(temp_FH$n_pobreza)][seq(1, nDom, by = 10)]) +
  scale_color_manual(values = c("FGV" = "Blue", "VarDirEst" = "Red"))
```



This code performs several transformations on the dataset `base_sae`:

1. **Creation of new variables:**
   - `pobreza_deff`: Replaces NaN values with 1 if they exist; otherwise, it keeps the original value.
   - `deff_FGV`: Computes a new Design Effect (DEFF) using the formula `hat_var / (pobreza_var / pobreza_deff)` when `pobreza_var` is not equal to 0.
   - `n_eff_FGV`: Calculates the effective number of surveyed individuals as `n_pobreza / deff_FGV`.
2. **Modification of the variable `pobreza`:**
   - If `hat_var` is NA, it replaces `pobreza` values with NA; otherwise, it retains the original value.

```
base_FH <- base_sae %>%
  mutate(
    pobreza_deff = ifelse(is.nan(pobreza_deff), 1, pobreza_deff),
    deff_FGV = ifelse(pobreza_var == 0 ,
      1,
```

```
      hat_var / (pobreza_var / pobreza_deff) #Fórmula del nuevo DEFF
    ),
    # Criterio MDS para regularizar el DeffFGV
    n_eff_FGV = n_pobreza / deff_FGV, #Número efectivo de personas encuestadas

     pobreza = ifelse(is.na(hat_var), NA_real_, pobreza)
  )


#saveRDS(object = base_FH, "Recursos/02_FGV/base_FH_2020.rds")
```

Chapter 3

Session 3- Fay Herriot Model - Poverty Estimation

The Fay Herriot model, proposed by Fay and Herriot (1979), is a statistical area model and is the most commonly used. It's essential to note that within small area estimation methodology, area models are the most applied because having individual-level information is often not feasible. Instead, we typically have data at the area level along with associated auxiliary information. This mixed linear model was the first to include random effects at the area level, implying that most of the information fed into the model corresponds to usually aggregated units like departments, regions, provinces, municipalities, among others. The estimates obtained from the model are over these aggregations or subpopulations.

Now, the Fay Herriot model relates area indicators θ_d, where d ranges from 1 to D, assuming they vary with respect to a vector of p covariates $\boldsymbol{x}_d$. The model is defined by the equation $\theta_d = \boldsymbol{x}_d^T\boldsymbol{\beta} + u_d$, where u_d is the error term or random effect, distinct for each area and distributed as $u_d \overset{ind}{\sim} (0, \sigma_u^2)$.

However, the true values of the indicators θ_d are not observable. Hence, the direct estimator $\hat{\theta}_d^{DIR}$ is used to estimate them, leading to sampling error. This estimator is still considered unbiased under the sample design, i.e.,

$$\hat{\theta}_d^{DIR} = \theta_d + e_d$$

The model is then fitted using the sampling error term e_d, where $e_d \overset{ind}{\sim} \left(0, \sigma_{e_d}^2\right)$, and the variances $\sigma_{e_d}^2$ are estimated using survey microdata. The FH model is rewritten as

$$\hat{\theta}_d^{DIR} = \boldsymbol{x}_d^T\boldsymbol{\beta} + u_d + e_d$$

.

The best linear unbiased predictor (BLUP) under the FH model is given by

$$\tilde{\theta}_d^{FH} = \boldsymbol{x}_d^T \tilde{\boldsymbol{\beta}} + \tilde{u}_d$$

,

where $\tilde{u}_d = \gamma_d \left(\hat{\theta}_d^{DIR} - \boldsymbol{x}_d^T \tilde{\boldsymbol{\beta}} \right)$ and $\gamma_d = \frac{\sigma_u^2}{\sigma_u^2 + \sigma_{e_d}^2}$.

Area Model for Poverty Estimation

Let P_d be the probability of finding a person in a state of poverty in the d-th domain of the population. Then, the direct estimator of P_d can be written as:

$$\hat{P}_d^{DIR} = P_d + e_d$$

Now, P_d can be modeled as follows:

$$P_d = \boldsymbol{x}_d^T \boldsymbol{\beta} + u_d$$

Rewriting $\hat{P}_d^{DIR}$ in terms of the two previous equations, we have:

$$\hat{P}_d^{DIR} = \boldsymbol{x}_d^T \boldsymbol{\beta} + u_d + e_d$$

It is then possible to assume that $\hat{P}_d^{DIR} \sim N(\boldsymbol{x}_d^T \boldsymbol{\beta}, \sigma_u^2 + \sigma_{e_d}^2)$, $\hat{P}_d^{DIR} \mid u_d \sim N(\boldsymbol{x}_d^T \boldsymbol{\beta} + u_d, \sigma_{e_d}^2)$, and $u_d \sim N(0, \sigma_u^2)$.

Next, prior distributions are assumed for $\boldsymbol{\beta}$ and σ_u^2:

$$\begin{aligned} \beta_p &\sim N(0, 10000) \\ \sigma_u^2 &\sim IG(0.0001, 0.0001) \end{aligned}$$

Therefore, the Bayesian estimator for P_d is given as $\tilde{P}_d = E\left(P_d \mid \hat{P}_d^{DIR} \right)$.

Optimal Predictor of P_d

The optimal predictor of P_d is given by:

$$E(P_d | \hat{P}_d^{DIR}) = \gamma_d \hat{P}_d^{DIR} + (1 - \gamma_d) \boldsymbol{x}_d^T \boldsymbol{\beta}$$

where $\gamma_d = \frac{\sigma_u^2}{\sigma_u^2 + \sigma_{e_d}^2}$.

We know that $\hat{P}_d^{DIR} \sim N(\boldsymbol{x}_d^T\boldsymbol{\beta}, \sigma_u^2 + \sigma_{e_d}^2)$, $\hat{P}_d^{DIR} \mid u_d \sim N(\boldsymbol{x}_d^T\boldsymbol{\beta} + u_d, \sigma_{e_d}^2)$, and $u_d \sim N(0, \sigma_u^2)$.

Therefore, the optimal predictor is determined by a weighted combination of the direct estimator $\hat{P}_d^{DIR}$ and the linear predictor $\boldsymbol{x}_d^T\boldsymbol{\beta}$ where the weights are determined by the ratio of the variance components, providing an optimal balance between the direct estimate and the model-based prediction.

$$
\begin{aligned}
f(u_d|\hat{P}_d^{DIR}) \propto f(\hat{P}_d^{DIR}|u_d)f(u_d) &= \frac{1}{\sigma_{e_d}^2\sqrt{2\pi}}\exp\left\{-\frac{1}{2\sigma_{e_d}^2(\hat{P}_d^{DIR} - \boldsymbol{x}_d^T\boldsymbol{\beta} - u_d)^2}\right\}\frac{1}{\sigma_u^2\sqrt{2\pi}}\exp\left\{-\frac{1}{2\sigma_u^2}u_d^2\right\} \\
&\propto \exp\left\{-\frac{u_d^2 - 2u_d(\hat{P}_d^{DIR} - \boldsymbol{x}_d^T\boldsymbol{\beta})}{2\sigma_{e_d}^2} - \frac{u_d^2}{2\sigma_u^2}\right\} \\
&= \exp\left\{-\frac{1}{2}\left[(\frac{1}{\sigma_{e_d}^2} + \frac{1}{\sigma_u^2})u_d^2 - 2\frac{\hat{P}_d^{DIR} - \boldsymbol{x}_d^T\boldsymbol{\beta}}{\sigma_{e_d}^2}u_d\right]\right\} \\
&= \exp\left\{-\frac{1}{2\frac{\sigma_u^2\sigma_{e_d}^2}{\sigma_u^2+\sigma_{e_d}^2}}\left[u_d^2 - 2\frac{\sigma_u^2}{\sigma_u^2 + \sigma_{e_d}^2}(\hat{P}_d^{DIR} - \boldsymbol{x}_d^T\boldsymbol{\beta})u_d\right]\right\} \\
&\propto \exp\left\{-\frac{1}{2\frac{\sigma_u^2\sigma_{e_d}^2}{\sigma_u^2+\sigma_{e_d}^2}}\left[u_d - \frac{\sigma_u^2}{\sigma_u^2 + \sigma_{e_d}^2}(\hat{P}_d^{DIR} - \boldsymbol{x}_d^T\boldsymbol{\beta})\right]^2\right\} \\
&\propto N(E(u_d|\hat{P}_d^{DIR}), \mathrm{Var}(u_d|P^{DIR}))
\end{aligned}
$$

with $E(u_d|\hat{P}_d^{DIR}) = \frac{\sigma_u^2}{\sigma_u^2+\sigma_{e_d}^2}(\hat{P}_d^{DIR} - \boldsymbol{x}_d^T\boldsymbol{\beta})$ y $\mathrm{Var}(u_d|P^{DIR}) = \frac{\sigma_u^2\sigma_{e_d}^2}{\sigma_u^2+\sigma_{e_d}^2}$. Therefore you have,

$$
\begin{aligned}
E(P_d|\hat{P}_d^{DIR}) = \boldsymbol{x}_d^T\boldsymbol{\beta} + E(u_d|\hat{P}_d^{DIR}) &= \boldsymbol{x}_d^T\boldsymbol{\beta} + \frac{\sigma_u^2}{\sigma_u^2 + \sigma_{e_d}^2}(\hat{P}_d^{DIR} - \boldsymbol{x}_d^T\boldsymbol{\beta}) \\
&= \frac{\sigma_{e_d}^2}{\sigma_u^2 + \sigma_{e_d}^2}\hat{P}_d^{DIR} + \frac{\sigma_u^2}{\sigma_u^2 + \sigma_{e_d}^2}\boldsymbol{x}_d^T\boldsymbol{\beta} \\
&= \gamma_d\hat{P}_d^{DIR} + (1 - \gamma_d)\boldsymbol{x}_d^T\boldsymbol{\beta}
\end{aligned}
$$

3.1 Estimation Procedure

This code utilizes the libraries `tidyverse` and `magrittr` for data processing and analysis.

The function `readRDS()` is used to load a data file in RDS format, containing direct estimates and smoothed variance for the proportion of individuals in poverty for the year 2018. Subsequently, the `%>%` operator from the `magrittr` library is employed to chain the selection of specific columns, namely `dam2`, `nd`, `pobreza`, `vardir`, and `hat_var`.

```
library(tidyverse)
library(magrittr)
base_FH <- readRDS("Recursos/03_FH_normal/01_base_FH.Rds") %>%
  select(dam2,pobreza,hat_var)
```

Reading the covariates, which have been previously obtained. Due to the difference in scales between variables, an adjustment is necessary.

```
statelevel_predictors_df <-
  readRDS('Recursos/03_FH_normal/02_statelevel_predictors_dam.rds') %>%
  mutate(id_order = 1:n())
```

Next, a full join (`full_join`) is performed between the dataset `base_FH` and the predictors `statelevel_predictors_df` using the variable `dam2` as the joining key.

The function `tba()` is used to display the first 10 rows and 8 columns of the resulting dataset from the previous join.

A full join (`full_join`) combines data from both datasets, preserving all rows from both and filling in missing values (NA) if matches aren't found based on the joining variable (dam2 in this case).

The `tba()` function displays an HTML-formatted table in the R console showing the first 10 rows and 8 columns of the resulting dataset from the join.

```
base_FH <- full_join(base_FH, statelevel_predictors_df, by = "dam2" )
tba(base_FH[1:10,1:8])
```

dam2	pobreza	hat_var	area1	sex2	age2	age3	age4
0101	NA	NA	1.0000	0.5087	0.2694	0.2297	0.1689
0102	0.9836	0	1.0000	0.4754	0.2857	0.2261	0.1527
0103	1.0000	0	1.0000	0.5037	0.3095	0.2015	0.1312
0201	NA	NA	0.5147	0.5060	0.2962	0.2090	0.1844
0202	0.9391	0	0.9986	0.5376	0.2625	0.2226	0.2238
0203	0.8117	0	0.9754	0.5432	0.2454	0.2254	0.2388
0204	NA	NA	1.0000	0.5300	0.3151	0.2022	0.2034
0205	0.9646	0	1.0000	0.5182	0.3057	0.2286	0.1981
0206	NA	NA	1.0000	0.5157	0.3192	0.1959	0.1552
0207	NA	NA	1.0000	0.5097	0.3099	0.1966	0.1691

```
# View(base_FH)
```

3.2 Preparing the supplies for STAN

1. Splitting the database into observed and unobserved domains.

   Observed domains.

```
data_dir <- base_FH %>% filter(!is.na(pobreza))
```

Unobserved domains.

```
data_syn <-
  base_FH %>% anti_join(data_dir %>% select(dam2))
tba(data_syn[1:10, 1:8])
```

dam2	pobreza	hat_var	area1	sex2	age2	age3	age4
0101	NA	NA	1.0000	0.5087	0.2694	0.2297	0.1689
0201	NA	NA	0.5147	0.5060	0.2962	0.2090	0.1844
0204	NA	NA	1.0000	0.5300	0.3151	0.2022	0.2034
0206	NA	NA	1.0000	0.5157	0.3192	0.1959	0.1552
0207	NA	NA	1.0000	0.5097	0.3099	0.1966	0.1691
0208	NA	NA	1.0000	0.5256	0.2880	0.2218	0.1974
0209	NA	NA	1.0000	0.5149	0.3018	0.2100	0.1759
0210	NA	NA	0.9779	0.5194	0.3000	0.2111	0.1828
0211	NA	NA	1.0000	0.5427	0.2645	0.2192	0.2267
0502	NA	NA	0.3699	0.4974	0.2727	0.1824	0.1795

2. Defining the fixed-effects matrix.

   Defines a linear model using the `formula()` function, incorporating various predictor variables such as age, ethnicity, unemployment rate, among others.

   Utilizes the `model.matrix()` function to generate design matrices (`Xdat` and `Xs`) from the observed (`data_observed`) and unobserved (`data_unobserved`) data to use in building regression models. The `model.matrix()` function transforms categorical variables into binary (dummy) variables, allowing them to be used in modeling.

```
formula_mod  <- formula(~  ODDJOB + WORKED +
                           stable_lights_mean +
                           accessibility_mean +
                           urban.coverfraction_sum)
## Dominios observados
Xdat <- model.matrix(formula_mod, data = data_dir)

## Dominios no observados
```

```r
Xs <- model.matrix(formula_mod, data = data_syn)
dim(Xs)
```

```
## [1] 22  6
```

```r
dim(data_syn)
```

```
## [1] 22 33
```

3. Creando lista de parámetros para STAN

```r
sample_data <- list(
  N1 = nrow(Xdat),   # Observed.
  N2 = nrow(Xs),   # Not observed.
  p  = ncol(Xdat),       # Number of predictors.
  X  = as.matrix(Xdat),  # Observed covariates.
  Xs = as.matrix(Xs),    # Not observed covariates.
  y  = as.numeric(data_dir$pobreza), # Direct estimation
  sigma_e = sqrt(data_dir$hat_var)   # Estimation error
)
```

Rutina implementada en STAN

```
data {
  int<lower=0> N1;   // number of data items
  int<lower=0> N2;   // number of data items for prediction
  int<lower=0> p;   // number of predictors
  matrix[N1, p] X;   // predictor matrix
  matrix[N2, p] Xs;   // predictor matrix
  vector[N1] y;      // predictor matrix
  vector[N1] sigma_e; // known variances
}

parameters {
  vector[p] beta;       // coefficients for predictors
  real<lower=0> sigma2_u;
  vector[N1] u;
}

transformed parameters{
  vector[N1] theta;
  vector[N1] thetaSyn;
  vector[N1] thetaFH;
  vector[N1] gammaj;
  real<lower=0> sigma_u;
```

```
  thetaSyn = X * beta;
  theta = thetaSyn + u;
  sigma_u = sqrt(sigma2_u);
  gammaj =  to_vector(sigma_u ./ (sigma_u + sigma_e));
  thetaFH = (gammaj) .* y + (1-gammaj).*thetaSyn;
}

model {
  // likelihood
  y ~ normal(theta, sigma_e);
  // priors
  beta ~ normal(0, 100);
  u ~ normal(0, sigma_u);
  sigma2_u ~ inv_gamma(0.0001, 0.0001);
}

generated quantities{
  vector[N2] y_pred;
  for(j in 1:N2) {
    y_pred[j] = normal_rng(Xs[j] * beta, sigma_u);
  }
}
```

4. Compiling the model in `STAN`. Here's the process to compile the `STAN` code from R:

This code utilizes the `rstan` library to fit a Bayesian model using the file `17FH_normal.stan`, which contains the model written in the Stan probabilistic modeling language.

Initially, the `stan()` function is employed to fit the model to the `sample_data`. The arguments passed to `stan()` include the file containing the model (`fit_FH_normal`), the data (`sample_data`), and control arguments for managing the model fitting process, such as the number of iterations for the warmup period (`warmup`), the sampling period (`iter`), and the number of CPU cores to use for the fitting process (`cores`).

Additionally, the `parallel::detectCores()` function is used to automatically detect the number of available CPU cores. The `mc.cores` option is then set to utilize the maximum number of available cores for the model fitting.

The outcome of the model fitting is stored in `model_FH_normal`, which contains a sample from the posterior distribution of the model. This sample can be employed for inferences about the model parameters and predictions. Overall, this code is useful for fitting Bayesian models using Stan and conducting subsequent inferences.

```
library(rstan)
fit_FH_normal <- "Recursos/03_FH_normal/modelosStan/17FH_normal.stan"
options(mc.cores = parallel::detectCores())
rstan::rstan_options(auto_write = TRUE) # speed up running time
model_FH_normal <- stan(
  file = fit_FH_normal,
  data = sample_data,
  verbose = FALSE,
  warmup = 2500,
  iter = 3000,
  cores = 4
)
saveRDS(object = model_FH_normal,
        file = "Recursos/03_FH_normal/03_model_FH_normal.rds")
```

Leer el modelo

```
model_FH_normal<- readRDS("Recursos/03_FH_normal/03_model_FH_normal.rds")
```

3.2.1 Results of the model for observed domains.

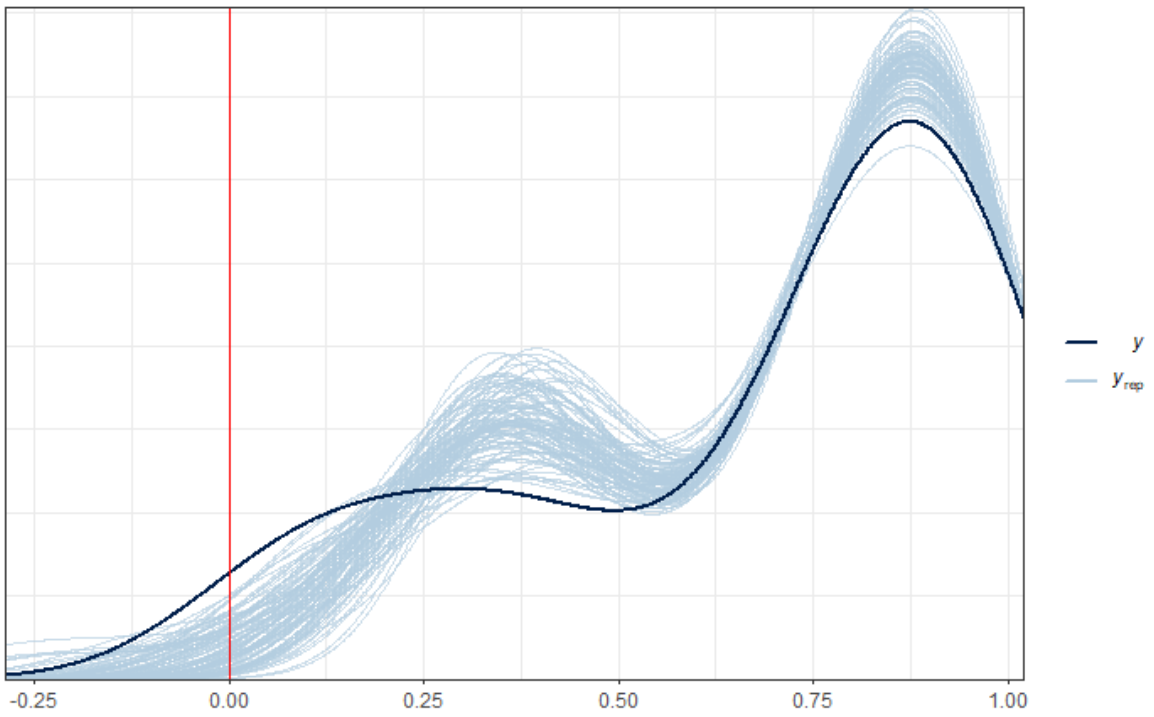
In this code, the `bayesplot`, `posterior`, and `patchwork` libraries are loaded to create graphics and visualizations of the model results.

Subsequently, the `as.array()` and `as_draws_matrix()` functions are used to extract samples from the posterior distribution of the parameter `theta` from the model. Then, 100 rows of these samples are randomly selected using the `sample()` function, resulting in the `y_pred2` matrix.

Finally, the `ppc_dens_overlay()` function from `bayesplot` is utilized to plot a comparison between the empirical distribution of the observed variable `pobreza` in the data (`data_dir$pobreza`) and the simulated posterior predictive distributions for the same variable (`y_pred2`). The `ppc_dens_overlay()` function generates a density plot for both distributions, facilitating the visualization of their comparison.

```
library(bayesplot)
library(posterior)
library(patchwork)
y_pred_B <- as.array(model_FH_normal, pars = "theta") %>%
  as_draws_matrix()
rowsrandom <- sample(nrow(y_pred_B), 100)
y_pred2 <- y_pred_B[rowsrandom, ]
p1 <-  ppc_dens_overlay(y = as.numeric(data_dir$pobreza), y_pred2)
```

```
# ggsave(plot = p1,
#        filename = "Recursos/Día2/Sesion1/0Recursos/FH1.png",
#        scale = 2)
p1 + geom_vline(xintercept = 0, color = "red")
```



The results indicate that using the Fay-Herriot normal method is not possible, as we are obtaining outcomes outside the boundaries.

3.3 Area models - ArcSin transformation.

In its most basic conception, the **Fay-Herriot** model is a linear combination of covariates. However, the result of this combination can take values that fall outside the acceptable range for a proportion; that is, generally, the Fay-Herriot estimator $\theta \in \mathbb{R}$, whereas the direct estimator $\theta \in (0, 1)$. The arcsine transformation is given by:

$$\hat{z}_d = \arcsin\left(\sqrt{\hat{\theta}_d}\right)$$

where

$$Var\left(\hat{z}_d\right) = \frac{\widehat{DEFF}_d}{4 \times n_d} = \frac{1}{4 \times n_{d,\text{effective}}}$$

The Fay-Herriot model is defined as follows:

$$
\begin{aligned}
Z_d \mid \mu_d, \sigma_d^2 &\sim N(\mu_d, \sigma_d^2) \\
\mu_d &= \boldsymbol{x}_d^T \boldsymbol{\beta} + u_d \\
\theta_d &= (\sin(\mu_d))^2
\end{aligned}
$$

where $u_d \sim N(0, \sigma^2)$.

Let the prior distributions for $\boldsymbol{\beta}$ and σ_u^2 be given by:

$$
\begin{aligned}
\boldsymbol{\beta} &\sim N\left(0, 1000\right) \\
\sigma_u^2 &\sim \text{IG}\left(0.0001, 0.0001\right)
\end{aligned}
$$

3.3.1 Estimation procedure

Reading the database that resulted in the previous step and selecting the columns of interest

```
library(tidyverse)
library(magrittr)

base_FH <- readRDS('Recursos/04_FH_Arcosin/01_base_FH.Rds') %>%
transmute(dam2,                                ## id dominios
          pobreza,
          T_pobreza = asin(sqrt(pobreza)),     ## creando zd
          n_effec = n_eff_FGV,                 ## n efectivo
          varhat = 1/(4*n_effec)               ## varianza para zd
)
```

Joining the two databases.

```
statelevel_predictors_df <-
  readRDS('Recursos/03_FH_normal/02_statelevel_predictors_dam.rds') %>%
  mutate(id_order = 1:n())
base_FH <-
  full_join(base_FH, statelevel_predictors_df, by = "dam2")
tba(base_FH[, 1:8] %>% head(10))
```

dam2	pobreza	T_pobreza	n_effec	varhat	area1	sex2	age2
0101	NA	NA	NA	NA	1.0000	0.5087	0.2694
0102	0.9836	1.4422	1029.570	2.000000e-04	1.0000	0.4754	0.2857
0103	1.0000	1.5708	0.000	2.226882e+57	1.0000	0.5037	0.3095
0201	NA	NA	NA	NA	0.5147	0.5060	0.2962
0202	0.9391	1.3215	5881.354	0.000000e+00	0.9986	0.5376	0.2625
0203	0.8117	1.1219	6965.029	0.000000e+00	0.9754	0.5432	0.2454
0204	NA	NA	NA	NA	1.0000	0.5300	0.3151
0205	0.9646	1.3814	11790.358	0.000000e+00	1.0000	0.5182	0.3057
0206	NA	NA	NA	NA	1.0000	0.5157	0.3192
0207	NA	NA	NA	NA	1.0000	0.5097	0.3099

Selecting the covariates for the model.

```
names_cov <-
    c(
       "ODDJOB","WORKED",
      "stable_lights_mean",
      "accessibility_mean",
      "urban.coverfraction_sum"
    )
```

3.3.2 Preparing Inputs for STAN

1. Splitting the database into observed and unobserved domains

Observed domains.

```
data_dir <- base_FH %>% filter(!is.na(T_pobreza))
```

Unobserved domains.

```
data_syn <-
  base_FH %>% anti_join(data_dir %>% select(dam2))
```

2. Defining the fixed-effects matrix.

```
## Observed domains
Xdat <- cbind(inter = 1,data_dir[,names_cov])

## Unobserved domains
Xs <-  cbind(inter = 1,data_syn[,names_cov])
```

3. Creating a parameter list for STAN.

```
sample_data <- list(
  N1 = nrow(Xdat),        # Observed.
  N2 = nrow(Xs),          # Unobserved.
  p  = ncol(Xdat),        # Number of regressors.
  X  = as.matrix(Xdat),   # Observed Covariates.
  Xs = as.matrix(Xs),     # Unobserved Covariates
  y  = as.numeric(data_dir$T_pobreza),
  sigma_e = sqrt(data_dir$varhat)
)
```

4. Compiling the model in STAN.

```
library(rstan)
fit_FH_arcoseno <- "Recursos/04_FH_Arcosin/modelosStan/15FH_arcsin_normal.stan"
options(mc.cores = parallel::detectCores())
rstan::rstan_options(auto_write = TRUE) # speed up running time
model_FH_arcoseno <- stan(
  file = fit_FH_arcoseno,
  data = sample_data,
  verbose = FALSE,
  warmup = 2500,
  iter = 3000,
  cores = 4
)
saveRDS(model_FH_arcoseno,
        "Recursos/04_FH_Arcosin/02_model_FH_arcoseno.rds")
```

```
model_FH_arcoseno <- readRDS("Recursos/04_FH_Arcosin/02_model_FH_arcoseno.rds")
```

3.3.2.1 Model results for the observed domains.

```
library(bayesplot)
library(patchwork)
library(posterior)

y_pred_B <- as.array(model_FH_arcoseno, pars = "theta") %>%
  as_draws_matrix()
rowsrandom <- sample(nrow(y_pred_B), 100)

y_pred2 <- y_pred_B[rowsrandom, ]
ppc_dens_overlay(y = as.numeric(data_dir$pobreza), y_pred2)
```

Graphical analysis of the convergence of σ_u^2 chains.

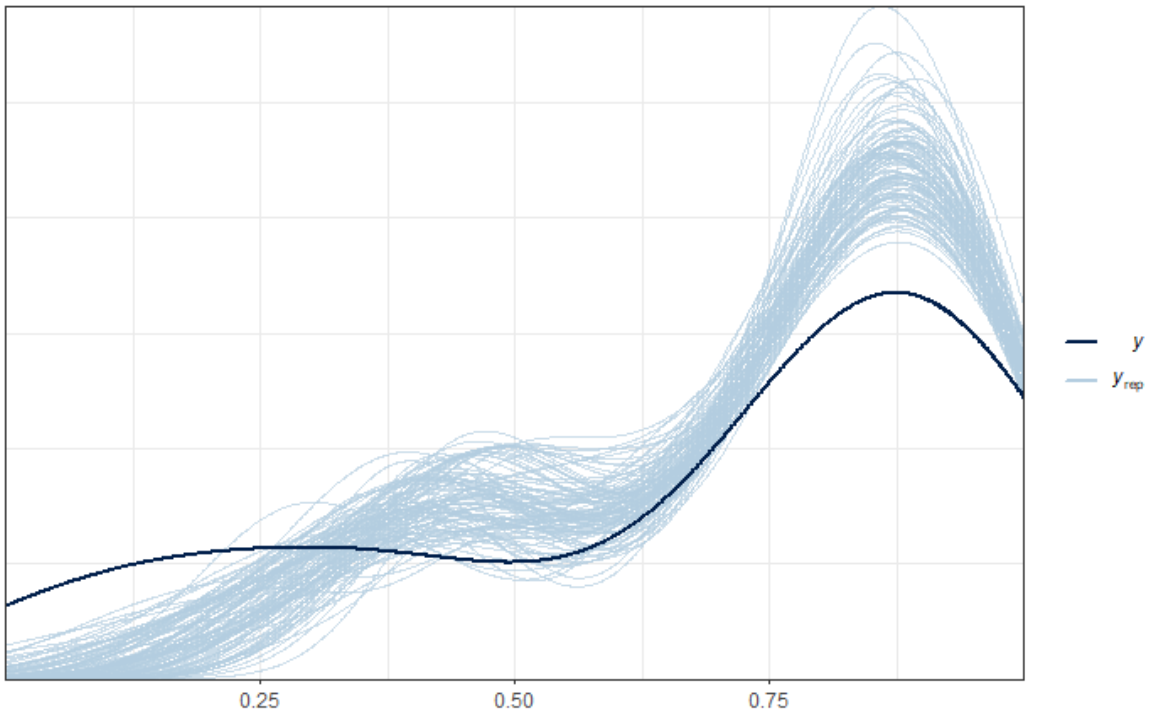


Figure 3.1: ppc arcosin

```
posterior_sigma2_u <- as.array(model_FH_arcoseno, pars = "sigma2_u")
(mcmc_dens_chains(posterior_sigma2_u) +
    mcmc_areas(posterior_sigma2_u) ) /
  mcmc_trace(posterior_sigma2_u)
```

To validate the convergence of all chains, the *R-hat* is used.

```
parametros <- summary(model_FH_arcoseno,
                      pars =  c("theta", "theta_pred")
                      )$summary %>%   data.frame()
p1 <- mcmc_rhat(parametros$Rhat)
p1
```

Estimation of the FH of poverty in the observed domains.

```
theta_FH <-   summary(model_FH_arcoseno,pars =  "theta")$summary %>%
  data.frame()
data_dir %<>% mutate(pred_arcoseno = theta_FH$mean,
                     pred_arcoseno_EE = theta_FH$sd,
                     Cv_pred = pred_arcoseno_EE/pred_arcoseno)
```

Estimation of the FH of poverty in the NOT observed domains.

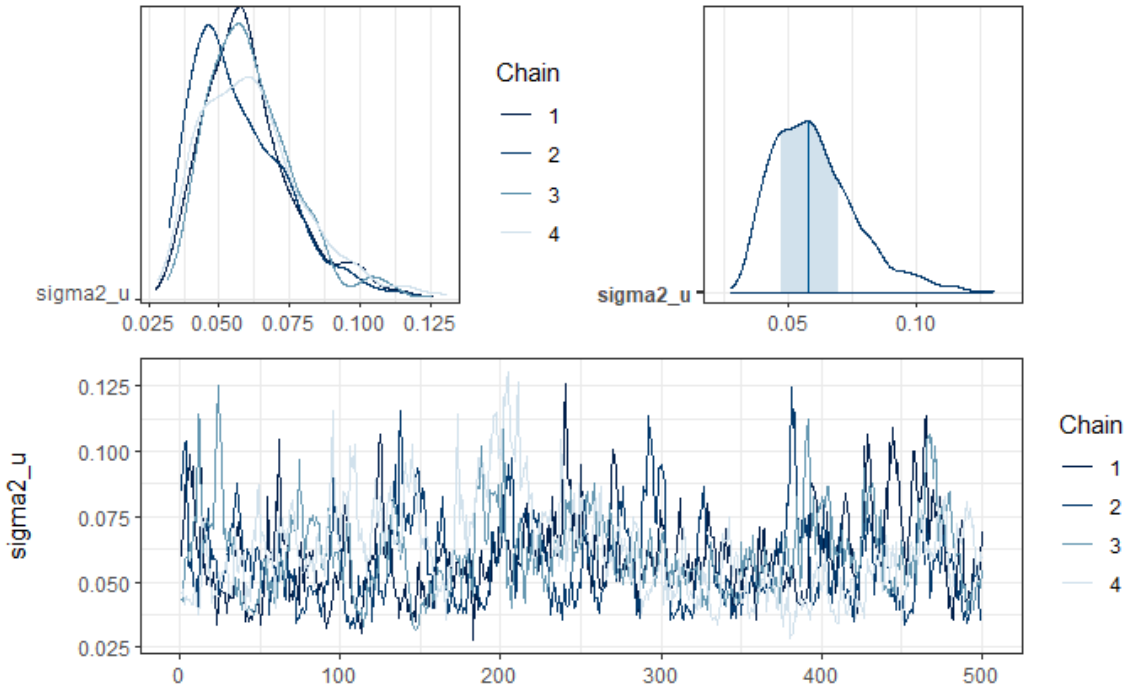


Figure 3.2: Posterior sigma2

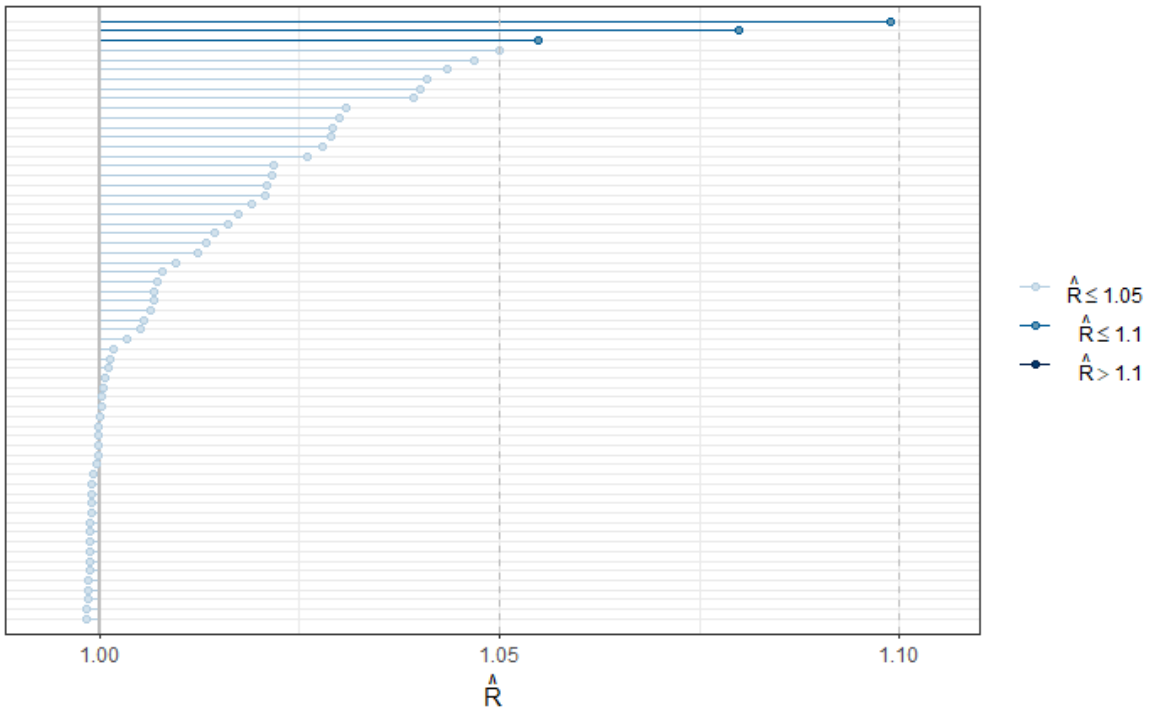


Figure 3.3: Rhat

```
theta_FH_pred <- summary(model_FH_arcoseno,pars =  "theta_pred")$summary %>%
  data.frame()
data_syn <- data_syn %>%
  mutate(pred_arcoseno = theta_FH_pred$mean,
         pred_arcoseno_EE = theta_FH_pred$sd,
         Cv_pred = pred_arcoseno_EE/pred_arcoseno)
```

Table 3.1: Estimation

dam2	pobreza	pred_arcoseno	pred_arcoseno_EE	Cv_pred
0101	NA	0.8630	0.1463	0.1696
0201	NA	0.8797	0.1426	0.1621
0204	NA	0.8412	0.1592	0.1893
0206	NA	0.8980	0.1296	0.1443
0207	NA	0.8958	0.1321	0.1475
0208	NA	0.7818	0.1893	0.2421
0209	NA	0.8910	0.1313	0.1474
0210	NA	0.8990	0.1290	0.1435
0211	NA	0.8872	0.1389	0.1566
0502	NA	0.7649	0.1889	0.2470

consolidating the bases of estimates for observed and UNobserved domains.

```
estimacionesPre <- bind_rows(data_dir, data_syn) %>%
  select(dam2, theta_pred = pred_arcoseno) %>%
  mutate(dam = substr(dam2,1,2))
```

3.4 Benchmark Process

1. From the census extract the total number of people by DAM2

```
total_pp <- readRDS(file = "Recursos/04_FH_Arcosin/06_censo_mrp.rds") %>%
   mutate(dam = substr(dam2,1,2))


N_dam_pp <- total_pp %>%   group_by(dam,dam2) %>%
            summarise(total_pp = sum(n) ) %>%
  group_by(dam) %>% mutate(dam_pp = sum(total_pp))

tba(N_dam_pp %>% data.frame() %>% slice(1:10),
    cap = "Number of people by DAM2")
```

Table 3.2: Number of people by DAM2

dam	dam2	total_pp	dam_pp
01	0101	35750	88799
01	0102	26458	88799
01	0103	26591	88799
02	0201	58937	573065
02	0202	43700	573065
02	0203	35309	573065
02	0204	47230	573065
02	0205	35694	573065
02	0206	38832	573065
02	0207	42103	573065

2. Obtaining direct estimates by DAM or the level of aggregation at which the survey is representative.

In this code, an RDS file of a survey (`07_data_JAM.rds`) is read, and the `transmute()` function is used to select and transform the variables of interest.

```
encuesta <- readRDS("Recursos/04_FH_Arcosin/07_encuesta.rds") %>%
   mutate(dam = substr(dam2,1,2))
```

The code is conducting survey data analysis using the `survey` package in R. Initially, an object `design` is created as a survey design using the `as_survey_design()` function from the `srvyr` package. This design includes primary sampling unit identifiers (`upm`), weights (`fep`), strata (`estrato`), and survey data (`encuesta`). Subsequently, the `design` object is grouped by the variable "Aggregate," and the mean of the variable "pobreza" with a confidence interval for the entire population is calculated using the `survey_mean()` function. The result is stored in the `directoDam` object and displayed in a table.

```
library(survey)
library(srvyr)
options(survey.lonely.psu = "adjust")

diseno <-
  as_survey_design(
    ids = upm,
    weights = fep,
    strata = estrato,
    nest = TRUE,
```

```
    .data = encuesta
  )
directoDam <- diseno %>%
    group_by(dam) %>%
  summarise(
    theta_dir = survey_mean(pobreza, vartype = c("ci"))
    )
tba(directoDam %>% slice(1:10), cap = "Direct estimation")
```

Table 3.3: Direct estimation

dam	theta_dir	theta_dir_low	theta_dir_upp
01	0.9926	0.9848	1.0004
02	0.9760	0.9638	0.9881
03	0.9191	0.8566	0.9816
04	0.8160	0.7742	0.8578
05	0.7486	0.6943	0.8029
06	0.9419	0.8993	0.9844
07	0.8019	0.7076	0.8962
08	0.5933	0.4904	0.6961
09	0.8628	0.8167	0.9090
10	0.7379	0.6348	0.8410

3. Carry out the consolidation of information obtained in *1* and *2*.

```
temp <- estimacionesPre %>%
  inner_join(N_dam_pp ) %>%
  inner_join(directoDam )

tba(temp %>% slice(1:10), cap = "Join datas")
```

Table 3.4: Join datas

dam2	theta_pred	dam	total_pp	dam_pp	theta_dir	theta_dir_low	theta_dir_upp
0102	0.9832	01	26458	88799	0.9926	0.9848	1.0004
0103	0.8759	01	26591	88799	0.9926	0.9848	1.0004
0202	0.9391	02	43700	573065	0.9760	0.9638	0.9881
0203	0.8117	02	35309	573065	0.9760	0.9638	0.9881
0205	0.9645	02	35694	573065	0.9760	0.9638	0.9881
0212	0.8303	02	58509	573065	0.9760	0.9638	0.9881
0301	0.9419	03	41762	93896	0.9191	0.8566	0.9816
0302	0.9109	03	52134	93896	0.9191	0.8566	0.9816
0401	0.8079	04	49909	81732	0.8160	0.7742	0.8578
0402	0.8307	04	31823	81732	0.8160	0.7742	0.8578

4. With the organized information, calculate the weights for the Benchmark

```
R_dam2 <- temp %>% group_by(dam) %>%
  summarise(
  R_dam_RB = unique(theta_dir) / sum((total_pp  / dam_pp) * theta_pred)
) %>%
  left_join(directoDam, by = "dam")

tba(R_dam2 %>% arrange(desc(R_dam_RB)), cap = "Weights for the Benchmark")
```

Table 3.5: Weights for the Benchmark

dam	R_dam_RB	theta_dir	theta_dir_low	theta_dir_upp
13	1.1793	0.6461	0.5226	0.7696
02	1.1158	0.9760	0.9638	0.9881
01	1.0996	0.9926	0.9848	1.0004
09	1.0669	0.8628	0.8167	0.9090
14	1.0550	0.8792	0.8376	0.9207
10	1.0465	0.7379	0.6348	0.8410
05	1.0098	0.7486	0.6943	0.8029
06	1.0031	0.9419	0.8993	0.9844
04	0.9991	0.8160	0.7742	0.8578
08	0.9973	0.5933	0.4904	0.6961
03	0.9940	0.9191	0.8566	0.9816
07	0.9474	0.8019	0.7076	0.8962
11	0.8631	0.4277	0.3056	0.5498
12	0.0480	0.0292	0.0120	0.0463

calculating the weights for each domain.

```
pesos <- temp %>%
  mutate(W_i = total_pp / dam_pp) %>%
  select(dam2, W_i)
tba(pesos %>% slice(1:10), cap = "Weights")
```

Table 3.6: Weights

dam2	W_i
0102	0.2980
0103	0.2995
0202	0.0763
0203	0.0616
0205	0.0623
0212	0.1021
0301	0.4448
0302	0.5552
0401	0.6106
0402	0.3894

5. Perform FH Benchmark Estimation

```
estimacionesBench <- estimacionesPre %>%
  left_join(R_dam2, by = c("dam")) %>%
  mutate(theta_pred_RBench = R_dam_RB * theta_pred) %>%
  left_join(pesos) %>%
  select(dam, dam2, W_i, theta_pred, theta_pred_RBench)

  tba(estimacionesBench %>% slice(1:10), cap = "Estimation Benchmark")
```

Table 3.7: Estimation Benchmark

dam	dam2	W_i	theta_pred	theta_pred_RBench
01	0102	0.2980	0.9832	1.0811
01	0103	0.2995	0.8759	0.9632
02	0202	0.0763	0.9391	1.0478
02	0203	0.0616	0.8117	0.9057
02	0205	0.0623	0.9645	1.0762
02	0212	0.1021	0.8303	0.9264
03	0301	0.4448	0.9419	0.9363
03	0302	0.5552	0.9109	0.9054
04	0401	0.6106	0.8079	0.8072
04	0402	0.3894	0.8307	0.8299

6. Validation: FH Estimation with Benchmark

```
estimacionesBench %>% group_by(dam) %>%
  summarise(theta_reg_RB = sum(W_i * theta_pred_RBench)) %>%
  left_join(directoDam, by = "dam") %>%
  tba(cap = "FH Estimation with Benchmark")
```

Table 3.8: FH Estimation with Benchmark

dam	theta_reg_RB	theta_dir	theta_dir_low	theta_dir_upp
01	0.9926	0.9926	0.9848	1.0004
02	0.9760	0.9760	0.9638	0.9881
03	0.9191	0.9191	0.8566	0.9816
04	0.8160	0.8160	0.7742	0.8578
05	0.7486	0.7486	0.6943	0.8029
06	0.9419	0.9419	0.8993	0.9844
07	0.8019	0.8019	0.7076	0.8962
08	0.5933	0.5933	0.4904	0.6961
09	0.8628	0.8628	0.8167	0.9090
10	0.7379	0.7379	0.6348	0.8410
11	0.4277	0.4277	0.3056	0.5498
12	0.0292	0.0292	0.0120	0.0463
13	0.6461	0.6461	0.5226	0.7696
14	0.8792	0.8792	0.8376	0.9207

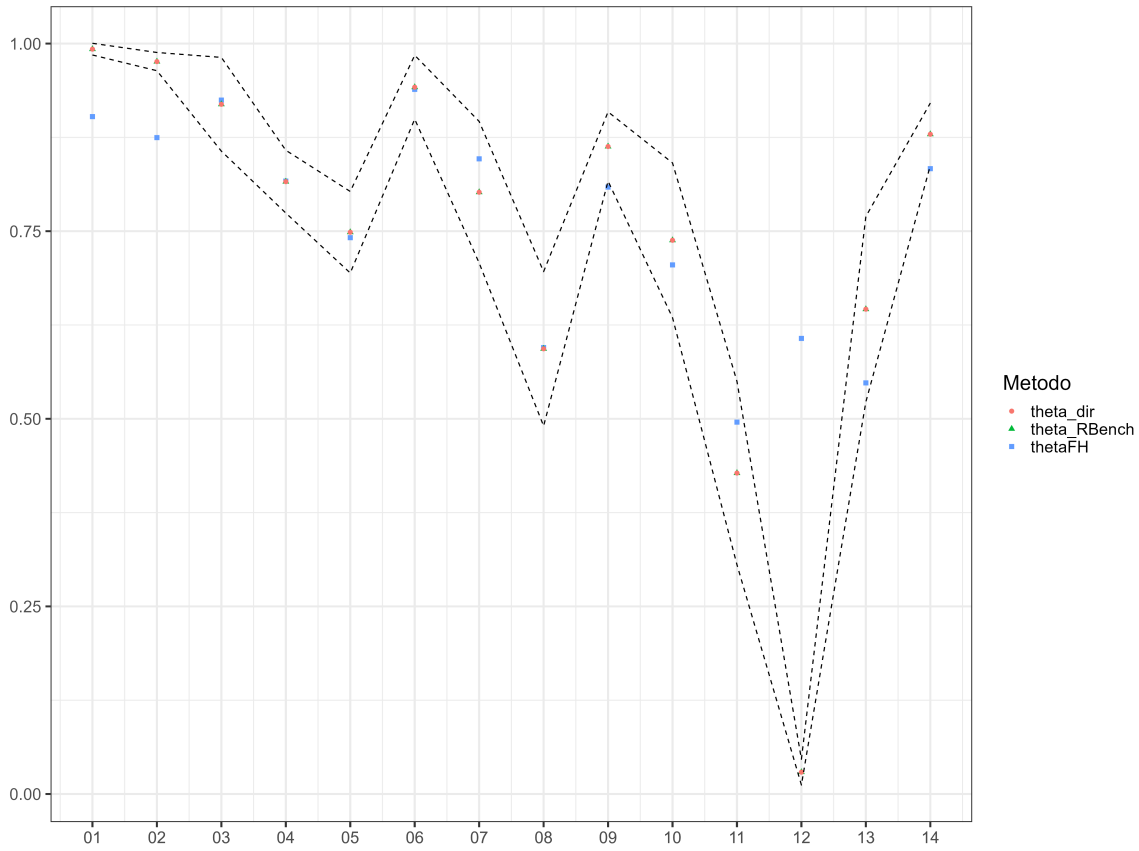
3.4.1 Results Validation

This code conducts data analysis and visualization using the `ggplot2` library. Specifically, it merges two `data frames` using the `left_join()` function, groups the data by the `dam` variable, and performs some operations to transform the `thetaFH` and `theta_pred_RBench` variables. Afterwards, it utilizes the `gather()` function to organize the data in a long format and visualizes it with `ggplot()`.

The resulting visualization displays points in different shapes and colors, representing various estimation methods. Additionally, it includes two dashed lines that depict the upper and lower confidence intervals for the observed values in the `theta_dir` variable.

```
temp <- estimacionesBench %>% left_join(
bind_rows(
data_dir %>% select(dam2, thetaFH = pred_arcoseno),
data_syn %>% select(dam2, thetaFH = pred_arcoseno))) %>%
group_by(dam) %>%
summarise(thetaFH = sum(W_i * theta_pred),
          theta_RBench = sum(W_i * theta_pred_RBench)
          ) %>%
left_join(directoDam, by = "dam")  %>%
mutate(id = 1:n())
```

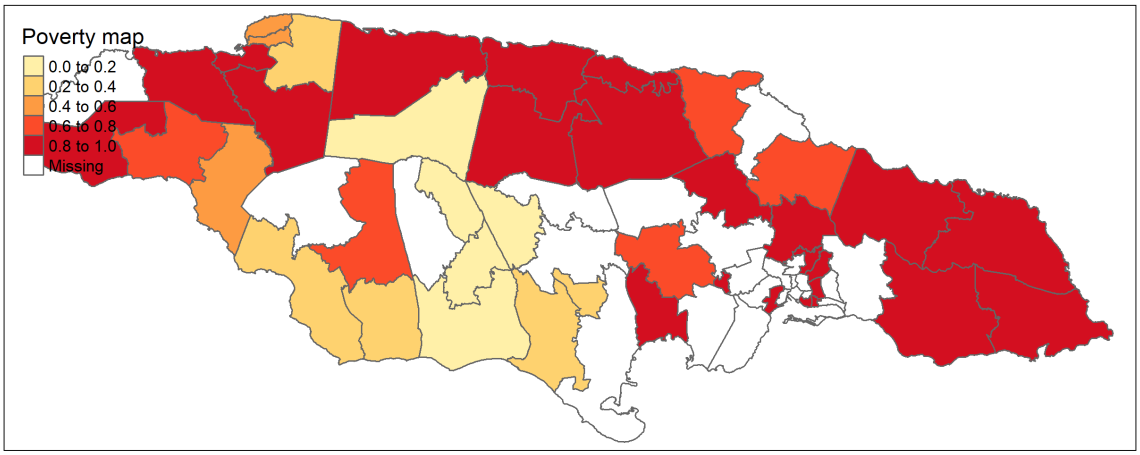
```
temp %<>% gather(key = "Metodo",value = "Estimacion",
                 -id, -dam, -theta_dir_upp, -theta_dir_low)

p1 <- ggplot(data = temp, aes(x = id, y = Estimacion, shape = Metodo)) +
  geom_point(aes(color = Metodo), size = 2) +
  geom_line(aes(y = theta_dir_low), linetype  = 2) +
  geom_line(aes(y = theta_dir_upp),  linetype  = 2) +
  theme_bw(20) +
  scale_x_continuous(breaks = temp$id,
    labels =  temp$dam) +
  labs(y = "", x = "")

# ggsave(plot = p1,
#        filename = "Recursos/04_FH_Arcosin/08_validar_bench.png",width = 16,height
p1
```

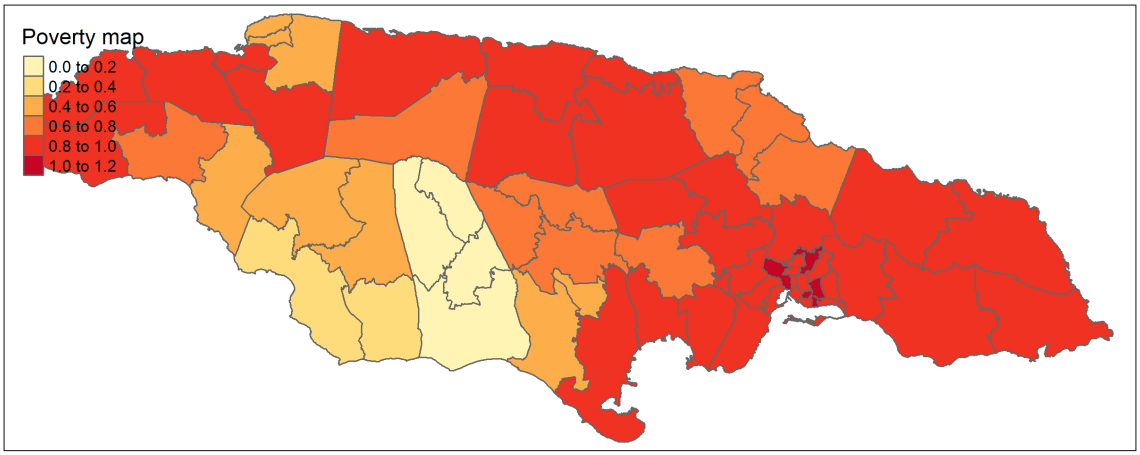


3.5 Poverty Map

This code block loads various packages (`sf`, `tmap`) and performs several operations. Initially, it conducts a `left_join` between the benchmark-adjusted estimates (`estimacionesBench`) and the model estimates (`data_dir`, `data_syn`), utilizing the `dam2` variable as the key for the join. Subsequently, it reads a `Shapefile` containing geospatial information for the country. Then, it creates a thematic map (`tmap`) using the `tm_shape()` function and adds layers using the `tm_polygons()` function. The map represents a variable `theta_pred_RBench` utilizing a color palette named "YlOrRd" and sets the intervals' breaks for the variable with the variable `brks_lp`. Finally, the `tm_layout()` function sets some design parameters for the map, such as the aspect ratio (asp).

```
library(sf)
library(tmap)

estimacionesBench %<>% left_join(
bind_rows(
data_dir %>% select(dam2, pobreza, pred_arcoseno_EE , Cv_pred),
data_syn %>% select(dam2,pobreza, pred_arcoseno_EE , Cv_pred)))

## Leer Shapefile del país
ShapeSAE <- read_sf("Shapefile/JAM2_cons.shp")


mapa <- tm_shape(ShapeSAE %>%
                   left_join(estimacionesBench,  by = "dam2"))

tmap_options(check.and.fix = TRUE)
Mapa_lp <-
  mapa + tm_polygons(
    c("pobreza","theta_pred_RBench"),
    title = "Poverty map",
    palette = "YlOrRd",
    colorNA = "white"
  ) + tm_layout(asp = 1.5)

tmap_save(Mapa_lp,
          filename = "Recursos/04_FH_Arcosin/09_map.png",
           width = 2500,
  height = 2000,
  asp = 0)
Mapa_lp
```


Poverty map
0.0 to 0.2
0.2 to 0.4
0.4 to 0.6
0.6 to 0.8
0.8 to 1.0
Missing


Poverty map
0.0 to 0.2
0.2 to 0.4
0.4 to 0.6
0.6 to 0.8
0.8 to 1.0
1.0 to 1.2

Chapter 4

Session 4 - Area model for labor market statistics

The National Labour Force Survey (NLFS) is a key survey in Jamaica conducted by the Statistical Institute of Jamaica (STATIN). This survey provides detailed and up-to-date information on the dynamics of the country's labor market. Some highlights of the NLFS include:

1. **Employment and Unemployment Measurement:** The survey gathers comprehensive data on the employment status of working-age individuals, identifying the employed, unemployed populations, and the unemployment rate across different demographic segments and regions of the country.
2. **Underemployment and Labor Conditions:** In addition to measuring unemployment, the survey assesses employment conditions, including underemployment, involuntary part-time work, and other forms of inadequate employment.
3. **Demographic Variables:** Important demographic data such as age, gender, education, and geographic location are collected, enabling the identification of specific labor patterns within different population groups.
4. **Frequency and Scope:** The survey is conducted periodically to capture changes in the labor market over time and covers a broad, representative sample of the population to ensure precise and reliable results.

4.1 Definition of the Multinomial Model

- Let K be the number of categories of the variable of interest $Y \sim multinomial\,(\boldsymbol{\theta})$, with $\boldsymbol{\theta} = (p_1, p_2, \ldots, p_k)$ and $\sum_{k=1}^{K} p_k = 1$.
- Let N_i be the number of elements in the i-th domain and N_{ik} be the number of

elements in the k-th category. Note that $\sum_{k=1}^{K} N_{ik} = N_i$ and $p_{ik} = \frac{N_{ik}}{N_i}$.

- Let $\hat{p}_{ik}$ be the direct estimation of p_{ik} and $v_{ik} = Var\left(\hat{p}_{ik}\right)$, and denote the estimator of the variance by $\hat{v}_{ik} = \widehat{Var}\left(\hat{p}_{ik}\right)$

Note that the design effect changes between categories; therefore, the first step is to define the effective sample size per category. This is:

The estimation of $\tilde{n}$ is given by $\tilde{n}_{ik} = \frac{(\tilde{p}_{ik} \times (1 - \tilde{p}_{ik}))}{\hat{v}_{ik}}$,

$\tilde{y}_{ik} = \tilde{n}_{ik} \times \hat{p}_{ik}$

Then, $\hat{n}_i = \sum_{k=1}^{K} \tilde{y}_{ik}$

From where it follows that $\hat{y}_{ik} = \hat{n}_i \times \hat{p}_{ik}$

Let $\boldsymbol{\theta} = (p_1, p_2, p_3)^T = \left(\frac{N_{i1}}{N_i}, \frac{N_{i2}}{N_i} \frac{N_{i3}}{N_i}\right)^T$, then the multinomial model for the i-th domain would be:

$$(\tilde{y}_{i1}, \tilde{y}_{i2}, \tilde{y}_{i3}) \mid \hat{n}_i, \boldsymbol{\theta}_i \sim multinomial\left(\hat{n}_i, \boldsymbol{\theta}_i\right)$$

Now, you can write p_{ik} as follows:

$\ln\left(\frac{p_{i2}}{p_{i1}}\right) = \boldsymbol{X}_i^T \beta_2 + u_{i2}$ and $\ln\left(\frac{p_{i3}}{p_{i1}}\right) = \boldsymbol{X}_i^T \beta_3 + u_{i3}$

Given the restriction $1 = p_{i1} + p_{i2} + p_{i3}$ then

$$p_{i1} + p_{i1}(e^{\boldsymbol{X}_i^T \boldsymbol{\beta_2}} + u_{i2}) + p_{i1}(e^{\boldsymbol{X}_i^T \beta_3} + u_{i3})$$

from where it follows that

$$p_{i1} = \frac{1}{1 + e^{\boldsymbol{X}_i^T \boldsymbol{\beta_2}} + u_{i2} + e^{\boldsymbol{X_i}^T \boldsymbol{\beta_3}} + u_{i3}}$$

The expressions for p_{i2} and p_{i3} would be:

$$p_{i2} = \frac{e^{\boldsymbol{X}_i^T \beta_2} + u_{i2}}{1 + e^{\boldsymbol{X}_i^T \boldsymbol{\beta_2}} + u_{i2} + e^{\boldsymbol{X_i}^T \boldsymbol{\beta_3}} + u_{i3}}$$

$$p_{i3} = \frac{e^{\boldsymbol{X}_i^T \beta_3} + u_{i3}}{1 + e^{\boldsymbol{X}_i^T \boldsymbol{\beta_2}} + u_{i2} + e^{\boldsymbol{X_i}^T \boldsymbol{\beta_3}} + u_{i3}}$$

4.2 Loading Libraries

- The `survey` library is a statistical analysis tool in R that allows working with complex survey data, such as stratified, multistage, or weighted surveys. It provides functions for parameter estimation, sample design, analysis of variance and regression, and calculation of standard errors.

- The `tidyverse` library is a collection of R packages used for data manipulation and visualization. It includes `dplyr`, `ggplot2`, `tidyr`, and others, characterized by its focus on 'tidy' or organized programming, making data exploration and analysis easier.
- The `srvyr` library is an extension of the `survey` library that integrates `survey` functions with `dplyr` syntax, facilitating the manipulation of complex survey data. It includes functions for grouping, filtering, and summarizing survey data using 'tidy' syntax.
- The `TeachingSampling` library is an R tool used for teaching statistical sampling methods. It includes functions for simulating different types of samples, estimating parameters, calculating standard errors, and constructing confidence intervals, among others.
- The `haven` library is an R tool that allows importing and exporting data in different formats, including SPSS, Stata, and SAS. It works with survey data files, providing functions for labeling variables, coding missing data, and converting data across formats.
- The `bayesplot` library is an R tool used for visualization and diagnostics of Bayesian models. It includes functions for plotting posterior distributions, convergence diagnostics, residual diagnostic plots, and other graphics related to Bayesian analysis.
- The `patchwork` library is an R tool that allows simple and flexible combination of plots. This library makes creating complex plots easier by allowing the combination of multiple plots into a single visualization, especially useful in data analysis and modeling.
- The `stringr` library is an R tool used for string manipulation. It includes functions for extracting, manipulating, and modifying text strings, particularly useful in data cleaning and preparation before analysis.
- The `rstan` library is an R tool used for Bayesian model estimation using the Markov Chain Monte Carlo (MCMC) method. This library allows specifying and estimating complex models using a simple and flexible language, offering various tools for diagnosis and visualization of results.

```r
library(survey)
library(tidyverse)
library(srvyr)
library(TeachingSampling)
library(haven)
library(bayesplot)
library(patchwork)
library(stringr)
```

```
library(rstan)
```

4.3 Reading the survey and direct estimates

This code performs several operations on a labor survey in Jamaica, represented by the `encuesta` object, which is read from a file in RDS format. Here's the breakdown:

1. **Data Reading:** The code reads data from the Jamaican labor survey from an RDS file located at 'Resources/05_Employment/01_data_JAM.rds'. The data is stored in the `encuesta` object.
2. **Data Transformation:** Through a sequence of operations using the `%>%` pipe and the `transmute()` function from the `dplyr` package, the following transformations are performed on the survey:
    - Specific columns `dam2`, `RFACT`, `PAR_COD`, `CONST_NUMBER`, `ED_NUMBER`, `STRATA`, and `EMPSTATUS` are selected from the survey.
    - More descriptive names are assigned to some columns such as `fep` for `RFACT`, `upm` by combining `PAR_COD`, `CONST_NUMBER`, and `ED_NUMBER`, `estrato` using conditions to define the value based on `STRATA`, and `empleo_label` and `empleo` representing specific categories derived from `EMPSTATUS` with labeled levels and categorical values.

In summary, the code reads a labor survey in Jamaica and performs a series of transformations on selected columns, renaming and reorganizing them for future analyses or processing.

```
encuesta <- readRDS('Recursos/05_Empleo/01_data_JAM.rds')
##
id_dominio <- "dam2"

encuesta <-
  encuesta %>%
  transmute(
    dam2,
    fep = RFACT,
    upm = paste0(PAR_COD , CONST_NUMBER, ED_NUMBER),
    estrato = ifelse(is.na(STRATA) ,strata,STRATA),
    empleo_label = as_factor(EMPSTATUS ,levels  = "labels"),
    empleo = as_factor(EMPSTATUS ,levels  = "values")
  )
```

The presented code defines the sampling design for the analysis of the "survey" in R. The first line sets an option for handling singleton PSU (primary sampling units), indicating

that adjustments need to be applied in standard error calculations. The second line uses the "as_survey_design" function from the "survey" library to define the sampling design. The function takes "encuesta" as an argument and the following parameters:

- `strata`: The variable defining the strata in the survey, in this case, the "estrato" variable.
- `ids`: The variable identifying the PSUs in the survey, here, the "upm" variable.
- `weights`: The variable indicating the survey weights of each observation, in this case, the "fep" variable.
- `nest`: A logical parameter indicating whether the survey data is nested or not. In this case, it's set to "TRUE" because the data is nested by domain.

Together, these steps allow defining a sampling design that takes into account the sampling characteristics and the weights assigned to each observation in the survey. This is necessary to obtain precise and representative estimations of the parameters of interest.

```
options(survey.lonely.psu= 'adjust' )
diseno <- encuesta %>%
  as_survey_design(
    strata = estrato,
    ids = upm,
    weights = fep,
    nest=T
  )
```

The following code conducts a descriptive analysis based on a survey design represented by the object `diseno`.

1. **Grouping and Filtering:** It uses the `%>%` function to chain operations. Initially, it groups the data by the domain identifier (`id_dominio`) using `group_by_at()` and subsequently filters observations where the variable `empleo` falls within the range of 3 to 5.
2. **Variable Summary:** With the `summarise()` function, it computes various summaries for different categories of the variable `empleo`. These summaries include the weighted count for employed, unemployed, and inactive individuals (`n_ocupado`, `n_desocupado`, `n_inactivo`). Furthermore, it utilizes the `survey_mean()` function to obtain weighted mean estimates for each category of `empleo`, considering the variable type (`vartype`) and design effect (`deff`).

```
indicador_dam <-
  diseno %>% group_by_at(id_dominio) %>%
  filter(empleo %in% c(3:5)) %>%
  summarise(
```

```
    n_ocupado = unweighted(sum(empleo == 3)),
    n_desocupado = unweighted(sum(empleo == 4)),
    n_inactivo = unweighted(sum(empleo == 5)),

    Ocupado = survey_mean(empleo == 3,
      vartype = c("se",  "var"),
      deff = T
    ),
    Desocupado = survey_mean(empleo == 4,
                             vartype = c("se",  "var"),
                             deff = T
    ),
    Inactivo = survey_mean(empleo == 5,
                             vartype = c("se",  "var"),
                             deff = T
    )
  )
```

3. **Upms counts by domains:** This code performs operations on the survey data. First, it selects the columns id_dominio and upm, removes duplicate rows, and then counts the number of unique upm values for each id_dominio. Subsequently, it performs an inner join of these results with an existing object indicador_dam based on the id_dominio column, thus consolidating information about the quantity of unique upm values per identified domain in the survey.

```
indicador_dam <- encuesta %>% select(id_dominio, upm) %>%
  distinct() %>%
  group_by_at(id_dominio) %>%
  tally(name = "n_upm") %>%
  inner_join(indicador_dam, by = id_dominio)
#Save data----------------------------
saveRDS(indicador_dam,'Recursos/05_Empleo/indicador_dam.Rds' )
```

4.4 Domain Selection

After conducting the necessary validations, the rule is set to include in the study the domains that have:

- Two or more PSUs per domain.
- An estimated design effect greater than 1 in all categories.

```
indicador_dam1 <- indicador_dam %>%
  filter(n_upm >= 2,
```

```
         Desocupado_deff > 1,
         Ocupado_deff > 1,
         Inactivo_deff > 1)  %>%
  mutate(id_orden = 1:n())

saveRDS(object = indicador_dam1, "Recursos/05_Empleo/02_base_modelo.Rds")
```

dam2	n_upm	n_ocupado	n_desocupado	n_inactivo	Ocupado	Ocupado_se
0101	17	784	40	487	0.6000	0.0188
0201	11	733	40	420	0.6033	0.0213
0202	11	617	25	332	0.6448	0.0470
0203	12	645	17	363	0.6409	0.0184
0204	11	638	69	402	0.5574	0.0258
0207	11	473	22	374	0.5350	0.0193
0209	8	434	4	307	0.5666	0.0405
0211	13	713	11	396	0.6232	0.0314
0212	9	639	49	301	0.6179	0.0274
0302	10	676	91	327	0.5907	0.0285

4.5 Modeling in STAN

The code presents the implementation of a multinomial logistic response area model using the programming language STAN. In this model, it is assumed that the response variable in each domain follows a multinomial distribution. The parameters governing the relationship between the predictor variables and the response variable are assumed to be different in each domain and are modeled as random effects.

The *functions* section defines an auxiliary function called `pred_theta()`, used to predict the values of the response variable in the unobserved domains. The `data` section contains the model's input variables, including the number of domains, the number of response variable categories, direct estimates of the response variable in each domain, observed covariates in each domain, and covariates corresponding to the unobserved domains.

The *parameters* section defines the model's unknown parameters, including the *beta* parameter matrix, containing coefficients relating covariates to the response variable in each category. Standard deviations of the random effects are also included.

The *transformed parameters* section defines the `theta` parameter vector, containing the probabilities of belonging to each category of the response variable in each domain. Random effects are used to adjust the values of `theta` in each domain.

The *model* section defines the model structure and includes prior distributions for the

unknown parameters. Particularly, a normal distribution is used for the coefficients of the beta matrix. Finally, it calculates the likelihood function of the multinomial distribution for the direct estimates of the response variable in each domain.

The *generated quantities* section is used to compute predictions of the response variable in the unobserved domains using the previously defined auxiliary function.

```
functions {
  matrix pred_theta(matrix Xp, int p, matrix beta){
  int D1 = rows(Xp);
  real num1[D1, p];
  real den1[D1];
  matrix[D1,p] theta_p;
  matrix[D1,p] tasa_pred;

  for(d in 1:D1){
    num1[d, 1] = 1;
    num1[d, 2] = exp(Xp[d, ] * beta[1, ]' ) ;
    num1[d, 3] = exp(Xp[d, ] * beta[2, ]' ) ;

    den1[d] = sum(num1[d, ]);
  }

  for(d in 1:D1){
    for(i in 2:p){
    theta_p[d, i] = num1[d, i]/den1[d];
    }
    theta_p[d, 1] = 1/den1[d];
   }

for(d in 1:D1){
    tasa_pred[d, 1] = theta_p[d,2]/(theta_p[d,1] + theta_p[d,2]);// TD
    tasa_pred[d, 2] = theta_p[d,1];                              // TO
    tasa_pred[d, 3] = theta_p[d,1] + theta_p[d,2];               // TP
    }

  return tasa_pred  ;
  }

}

data {
  int<lower=1> D; // número de dominios
  int<lower=1> P; // categorías
```

```
  int<lower=1> K; // cantidad de regresores
  int hat_y[D, P]; // matriz de datos
  matrix[D, K] X_obs; // matriz de covariables
  int<lower=1> D1; // número de dominios
  matrix[D1, K] X_pred; // matriz de covariables
}


parameters {
  matrix[P-1, K] beta;// matriz de parámetros
  vector<lower=0>[P-1] sigma_u;       // random effects standard deviations
  // declare L_u to be the Choleski factor of a 2x2 correlation matrix
  cholesky_factor_corr[P-1] L_u;
  matrix[P-1, D] z_u;
}

transformed parameters {
  simplex[P] theta[D];// vector de parámetros;
  real num[D, P];
  real den[D];
  matrix[D,P] tasa_obs;
  // this transform random effects so that they have the correlation
  // matrix specified by the correlation matrix above
  matrix[P-1, D] u; // random effect matrix
  u = diag_pre_multiply(sigma_u, L_u) * z_u;

  for(d in 1:D){
    num[d, 1] = 1;
    num[d, 2] = exp(X_obs[d, ] * beta[1, ]' + u[1, d]) ;
    num[d, 3] = exp(X_obs[d, ] * beta[2, ]' + u[2, d]) ;

    den[d] = sum(num[d, ]);
  }

  for(d in 1:D){
    for(p in 2:P){
    theta[d, p] = num[d, p]/den[d];
    }
    theta[d, 1] = 1/den[d];
  }

  for(d in 1:D){
    tasa_obs[d, 1] = theta[d,2]/(theta[d,1] + theta[d,2]);// TD
```

```
    tasa_obs[d, 2] = theta[d,1];                                    // TO
    tasa_obs[d, 3] = theta[d,1] + theta[d,2];                  // TP
    }

}

model {
  L_u ~ lkj_corr_cholesky(1); // LKJ prior for the correlation matrix
  to_vector(z_u) ~ normal(0, 10000);
  // sigma_u ~ cauchy(0, 50);
  sigma_u ~ inv_gamma(0.0001, 0.0001);

  for(p in 2:P){
    for(k in 1:K){
      beta[p-1, k] ~ normal(0, 10000);
    }
    }

  for(d in 1:D){
    target += multinomial_lpmf(hat_y[d, ] | theta[d, ]);
  }
}


generated quantities {
  matrix[D1,P] tasa_pred;
  matrix[2, 2] Omega;
  Omega = L_u * L_u'; // so that it return the correlation matrix

 tasa_pred = pred_theta(X_pred, P, beta);
}
```

4.6 Preparing supplies for STAN

1. Reading and adaptation of covariates

```
statelevel_predictors_df <-
  readRDS('Recursos/05_Empleo/03_statelevel_predictors_dam.rds') %>%
  mutate(id_orden =1:n())

head(statelevel_predictors_df[,1:9],10) %>% tba()
```

dam2	area1	sex2	age2	age3	age4	age5	tiene_sanitario	tiene_electricidad
0101	1.0000	0.5087	0.2694	0.2297	0.1689	0.0672	0.0019	0.7596
0102	1.0000	0.4754	0.2857	0.2261	0.1527	0.0683	0.0011	0.9064
0103	1.0000	0.5037	0.3095	0.2015	0.1312	0.0449	0.0152	0.6930
0201	0.5147	0.5060	0.2962	0.2090	0.1844	0.0711	0.0138	0.2342
0202	0.9986	0.5376	0.2625	0.2226	0.2238	0.0961	0.0028	0.3852
0203	0.9754	0.5432	0.2454	0.2254	0.2388	0.1160	0.0015	0.3326
0204	1.0000	0.5300	0.3151	0.2022	0.2034	0.0776	0.0042	0.5720
0205	1.0000	0.5182	0.3057	0.2286	0.1981	0.0768	0.0013	0.8060
0206	1.0000	0.5157	0.3192	0.1959	0.1552	0.0496	0.0290	0.0285
0207	1.0000	0.5097	0.3099	0.1966	0.1691	0.0538	0.0465	0.1581

2. Select the model variables and create a covariate matrix.

```
names_cov <-
  c(
   "dam2",
    "ODDJOB","WORKED",
    "stable_lights_mean",
    "accessibility_mean",
    "urban.coverfraction_sum",
    "id_orden"
  )
X_pred <-
  anti_join(statelevel_predictors_df %>% select(all_of(names_cov)),
            indicador_dam1 %>% select(dam2))
```

The code block identifies which domains will be the predicted ones.

```
X_pred %>% select(dam2, id_orden) %>%
  saveRDS(file = "Recursos/05_Empleo/dam_pred.rds")
```

Creating the covariate matrix for the unobserved (`X_pred`) and observed (`X_obs`) domains

```
## Obteniendo la matrix
X_pred %<>%
  data.frame() %>%
  select(-dam2,-id_orden)  %>%  as.matrix()

## Identificando los dominios para realizar estimación del modelo

X_obs <- inner_join(indicador_dam1 %>% select(dam2),
                    statelevel_predictors_df[,names_cov]) %>%
  arrange(id_orden) %>%
```

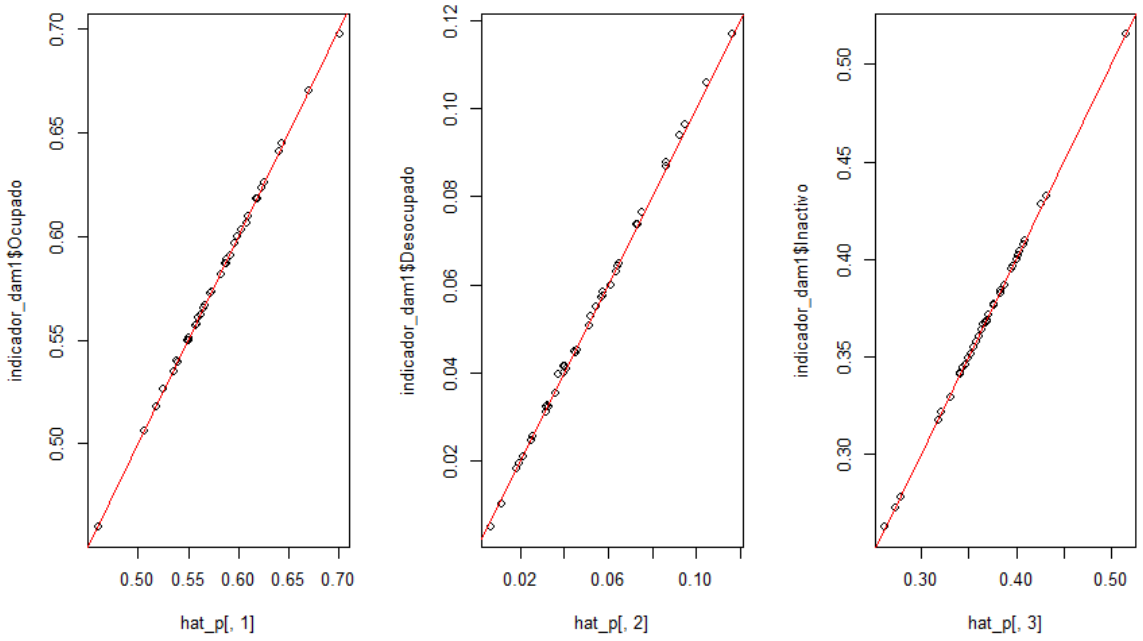
```
  data.frame() %>%
  select(-dam2, -id_orden)  %>%  as.matrix()
```

3. Calculating the n_cash and the $\tilde{y}$

```
D <- nrow(indicador_dam1)
P <- 3 # Ocupado, desocupado, inactivo.
Y_tilde <- matrix(NA, D, P)
n_tilde <- matrix(NA, D, P)
Y_hat <- matrix(NA, D, P)

# n efectivos ocupado
n_tilde[,1] <- (indicador_dam1$Ocupado*(1 - indicador_dam1$Ocupado))/
  indicador_dam1$Ocupado_var
Y_tilde[,1] <- n_tilde[,1]* indicador_dam1$Ocupado


# n efectivos desocupado
n_tilde[,2] <- (indicador_dam1$Desocupado*(1 - indicador_dam1$Desocupado))/
  indicador_dam1$Desocupado_var
Y_tilde[,2] <- n_tilde[,2]* indicador_dam1$Desocupado

# n efectivos Inactivo
n_tilde[,3] <- (indicador_dam1$Inactivo*(1 - indicador_dam1$Inactivo))/
  indicador_dam1$Inactivo_var
Y_tilde[,3] <- n_tilde[,3]* indicador_dam1$Inactivo
```

Now, we validate the consistency of the calculations carried out

```
ni_hat = rowSums(Y_tilde)
Y_hat[,1] <- ni_hat* indicador_dam1$Ocupado
Y_hat[,2] <- ni_hat* indicador_dam1$Desocupado
Y_hat[,3] <- ni_hat* indicador_dam1$Inactivo
Y_hat <- round(Y_hat)

hat_p <- Y_hat/rowSums(Y_hat)
par(mfrow = c(1,3))
plot(hat_p[,1],indicador_dam1$Ocupado)
abline(a = 0,b=1,col = "red")
plot(hat_p[,2],indicador_dam1$Desocupado)
abline(a = 0,b=1,col = "red")
plot(hat_p[,3],indicador_dam1$Inactivo)
abline(a = 0,b=1,col = "red")
```



4. Compiling the model

```
X1_obs <- cbind(matrix(1,nrow = D,ncol = 1),X_obs)
K = ncol(X1_obs)
D1 <- nrow(X_pred)
X1_pred <- cbind(matrix(1,nrow = D1,ncol = 1),X_pred)

sample_data <- list(D = D,
                    P = P,
                    K = K,
                    y_tilde = Y_hat,
                    X_obs = X1_obs,
                    X_pred = X1_pred,
                    D1 = D1)


library(rstan)
model_bayes_mcmc2 <- stan(
   # Stan program
  file = "Recursos/05_Empleo/modelosStan/00 Multinomial_simple_no_cor.stan",
  data = sample_data,    # named list of data
  verbose = TRUE,
  warmup = 1000,          # number of warmup iterations per chain
  iter = 2000,            # total number of iterations per chain
```

```
  cores = 4,              # number of cores (could use one per chain)
)

saveRDS(model_bayes_mcmc2,
        "Recursos/05_Empleo/05_model_bayes_multinomial_cor.Rds")
```

4.7 Model validation

Model validation is essential to assess a model's ability to accurately and reliably predict future outcomes. In the case of a multinomial response area model, validation focuses on measuring the model's accuracy in predicting different response categories. The main objective of validation is to determine if the model can generalize well to unseen data and provide accurate predictions. This involves comparing the model's predictions to observed data and using evaluation metrics to measure model performance. Model validation is crucial to ensure prediction quality and the model's reliability for use in future applications.

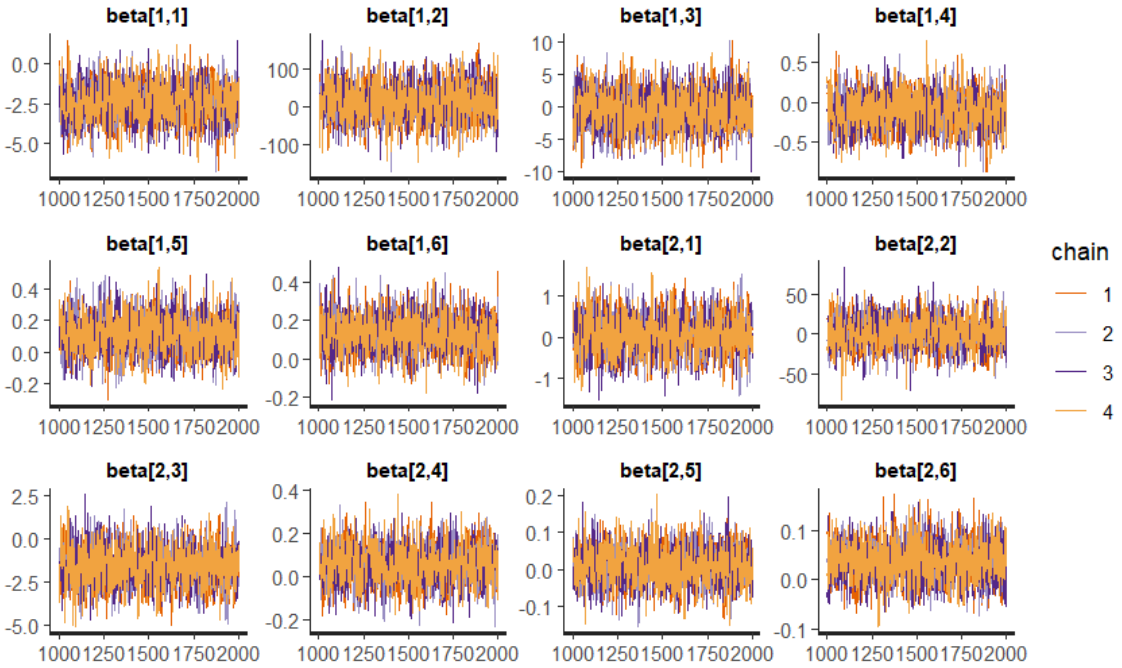
```
infile <- paste0("Recursos/05_Empleo/05_model_bayes_multinomial_cor.Rds")
model_bayes <- readRDS(infile)

#--- Exporting Bayesian Multilevel Model Results ---#

paramtros <- summary(model_bayes)$summary %>% data.frame()

tbla_rhat <- mcmc_rhat_data(paramtros$Rhat) %>%
  group_by(description) %>%
  tally() %>% mutate(Porcen = n/sum(n)*100)

tbla_rhat %>% tba()
```

description	n	Porcen
hat(R) <= 1.05	599	100

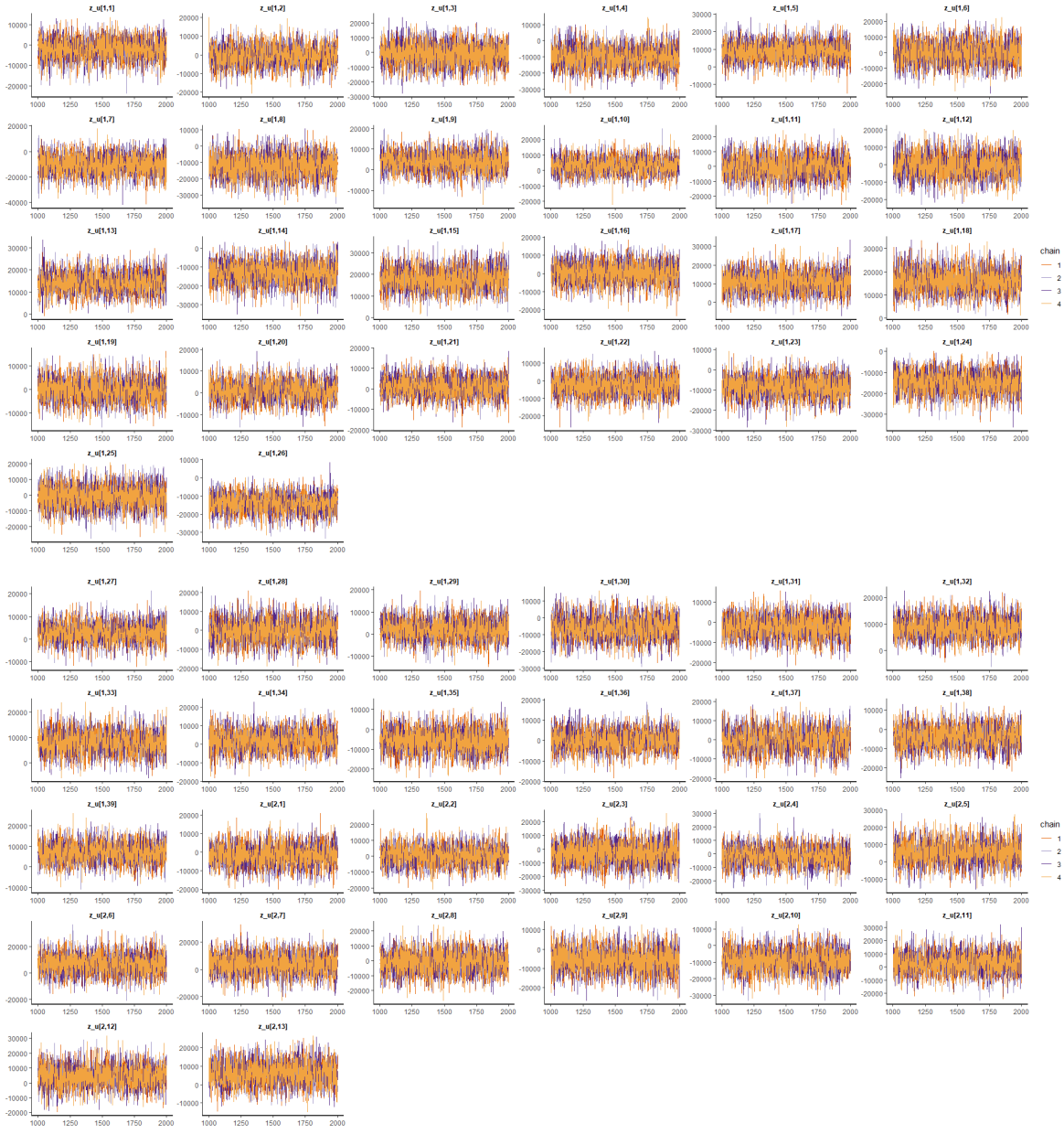
Fixed effects

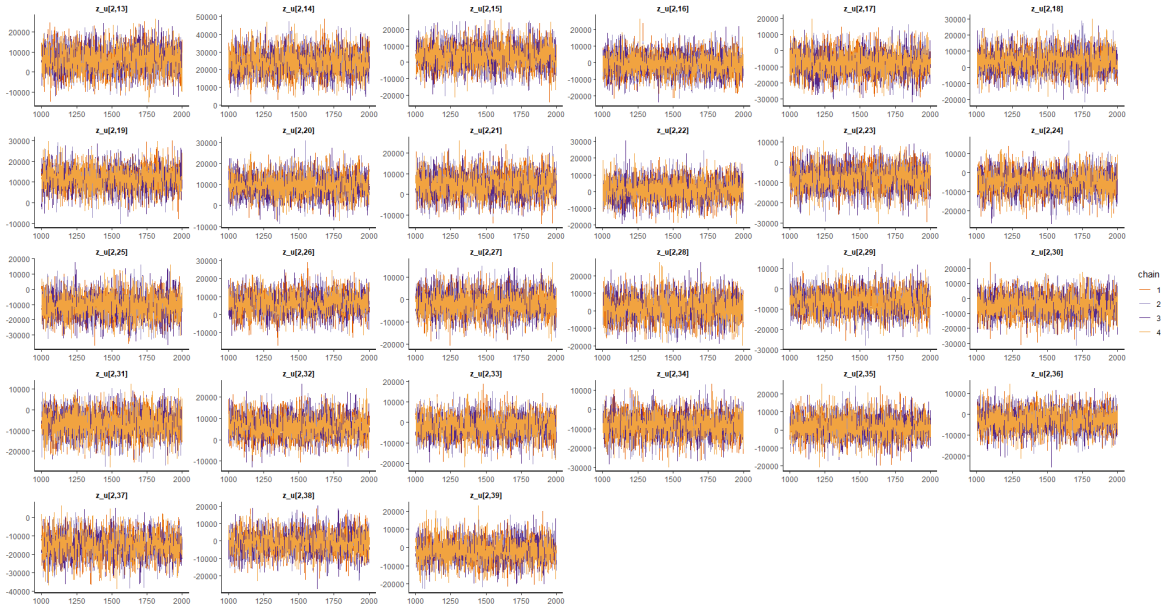
```
efecto_fijo <-  grep(pattern = "beta",
                     x = rownames(paramtros),
                     value = TRUE)

p_fijo <- traceplot(model_bayes, pars = efecto_fijo)
```

```
p_fijo
```



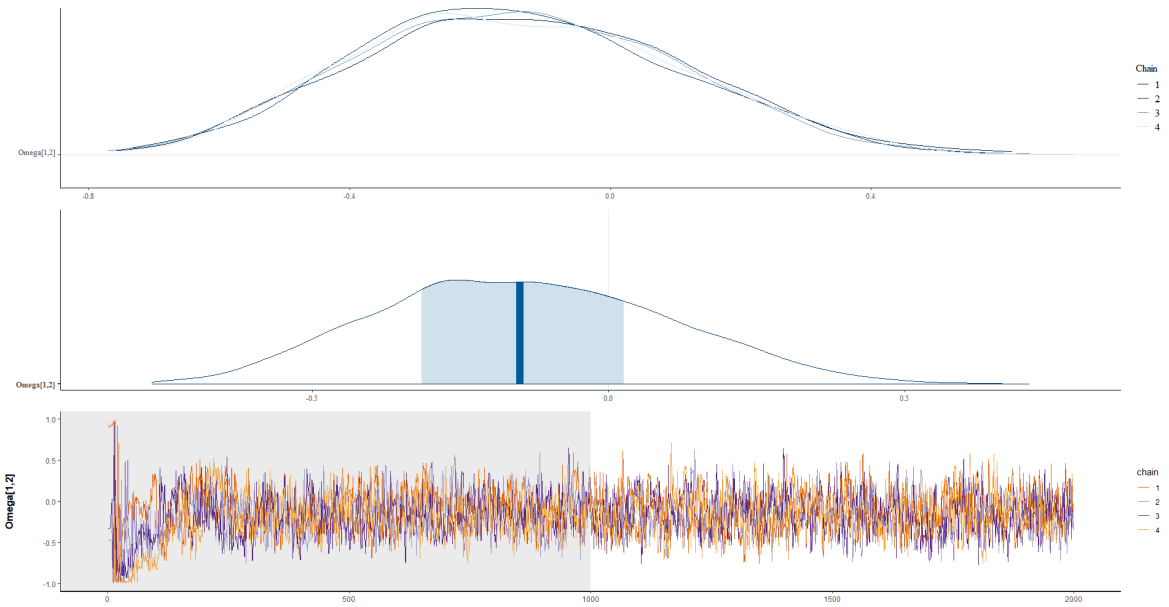

```
efecto_aleatorio <-  grep(pattern = "z_u",
                          x = rownames(paramtros),
                          value = TRUE)

p_alea1 <- traceplot(model_bayes, pars = efecto_aleatorio[1:26])
p_alea2 <- traceplot(model_bayes, pars = efecto_aleatorio[27:(26*2)])
p_alea3 <- traceplot(model_bayes, pars = efecto_aleatorio[(26*2):78])
```


z_u[1,1]
z_u[1,2]
z_u[1,3]
z_u[1,4]
z_u[1,5]
z_u[1,6]
z_u[1,7]
z_u[1,8]
z_u[1,9]
z_u[1,10]
z_u[1,11]
z_u[1,12]
z_u[1,13]
z_u[1,14]
z_u[1,15]
z_u[1,16]
z_u[1,17]
z_u[1,18]
z_u[1,19]
z_u[1,20]
z_u[1,21]
z_u[1,22]
z_u[1,23]
z_u[1,24]
z_u[1,25]
z_u[1,26]
z_u[1,27]
z_u[1,28]
z_u[1,29]
z_u[1,30]
z_u[1,31]
z_u[1,32]
z_u[1,33]
z_u[1,34]
z_u[1,35]
z_u[1,36]
z_u[1,37]
z_u[1,38]
z_u[1,39]
z_u[2,1]
z_u[2,2]
z_u[2,3]
z_u[2,4]
z_u[2,5]
z_u[2,6]
z_u[2,7]
z_u[2,8]
z_u[2,9]
z_u[2,10]
z_u[2,11]
z_u[2,12]
z_u[2,13]
chain
1
2
3
4



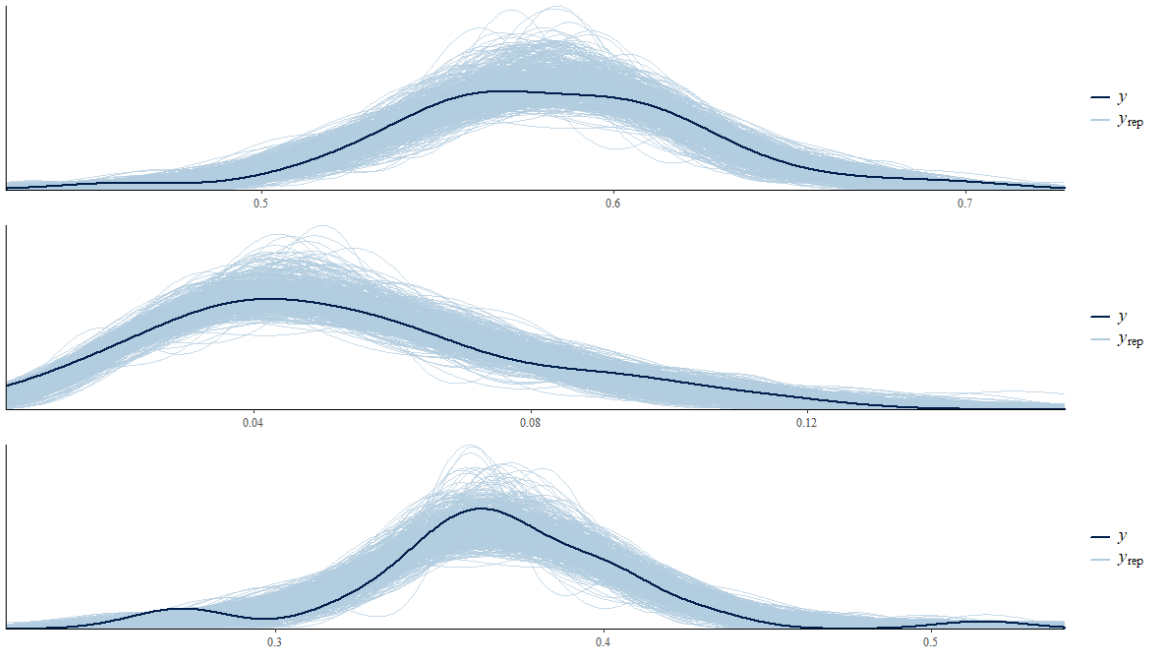
Estimated values for the correlation matrix {-}

```
## Valores estimados para la matriz de correlación
omega12 <- summary(model_bayes, pars = "Omega[1,2]")$summary

plot_omega <- Plot_dens_draws(model_bayes, pars = "Omega[1,2]")
```



4.7.1 Posterior predictive distribution

```
theta_dir <- indicador_dam1 %>%
  transmute(dam2,
    n = n_desocupado + n_ocupado + n_inactivo,
        Ocupado, Desocupado, Inactivo)

color_scheme_set("brightblue")
theme_set(theme_bw(base_size = 15))
y_pred_B <- as.array(model_bayes, pars = "theta") %>%
  as_draws_matrix()

rowsrandom <- sample(nrow(y_pred_B), 100)

theta_1<-  grep(pattern = "1]",x = colnames(y_pred_B),value = TRUE)
theta_2<-  grep(pattern = "2]",x = colnames(y_pred_B),value = TRUE)
theta_3<-  grep(pattern = "3]",x = colnames(y_pred_B),value = TRUE)
y_pred1 <- y_pred_B[rowsrandom,theta_1 ]
y_pred2 <- y_pred_B[rowsrandom,theta_2 ]
y_pred3 <- y_pred_B[rowsrandom,theta_3 ]

p1 <- ppc_dens_overlay(y = as.numeric(theta_dir$Ocupado), y_pred1)/
  ppc_dens_overlay(y = as.numeric(theta_dir$Desocupado), y_pred2)/
  ppc_dens_overlay(y = as.numeric(theta_dir$Inactivo), y_pred3)
```



4.8 Parameter estimation.

This code block starts by importing an `dam_pred.rds` file and sets some variables for the number of parameters and the number of domains. Then, it extracts summaries from a model named `model_bayes` for observed and predicted rates. Next, it organizes these rates into ordered matrices, assigning appropriate column names and converting them into data frames. These data frames are concatenated with the `dam2` column from the original `indicador_dam1` and `dam_pred` data, respectively, creating two data frames (`tasa_obs_ordered` and `tasa_pred_ordered`) containing organized parameter estimates ready for further analysis.

```
dam_pred <- readRDS("Recursos/05_Empleo/dam_pred.rds")
P <- 3
D <- nrow(indicador_dam1)
D1 <- nrow(dam_pred)
## Estimación del modelo.
theta_dir <- indicador_dam1
tasa_obs <- summary(model_bayes,pars = "tasa_obs")$summary
tasa_pred <-  summary(model_bayes,pars = "tasa_pred")$summary

## Ordenando la matrix de theta
tasa_obs_ordenado <- matrix(tasa_obs[,"mean"],
                            nrow = D,
                            ncol = P,byrow = TRUE)

colnames(tasa_obs_ordenado) <- c("TD_mod", "TO_mod", "TP_mod")
tasa_obs_ordenado%<>% as.data.frame()
tasa_obs_ordenado <- cbind(dam2 = indicador_dam1$dam2,
                           tasa_obs_ordenado)



tasa_pred_ordenado <- matrix(tasa_pred[,"mean"],
                             nrow = D1,
                             ncol = P,byrow = TRUE)

colnames(tasa_pred_ordenado) <- c("TD_mod", "TO_mod", "TP_mod")
tasa_pred_ordenado%<>% as.data.frame()
tasa_pred_ordenado <- cbind(dam2 = dam_pred$dam2, tasa_pred_ordenado)

estimaciones_obs <- full_join(theta_dir,
                              bind_rows(tasa_obs_ordenado, tasa_pred_ordenado))
```

4.9 Estimation of Standard Deviation and Coefficient of Variation

This code block computes the standard deviations (sd) and coefficients of variation (cv) for the `theta` parameters, both observed and predicted. Initially, the `summary()` function from the `rstan` package is used to extract the `sd` values for observed and predicted `theta` parameters from the Bayesian estimation model (`model_bayes`). Subsequently, the `sd` values are organized into matrices ordered by `dam2` and given corresponding names. Using these matrices, another matrix is generated containing the coefficients of variation for the observed `theta` parameters (`theta_obs_ordenado_cv`). Similarly, ordered matrices are constructed by `dam2` for both the `sd` and `cv` values of the predicted `theta` parameters (`theta_pred_ordenado_sd` and `theta_pred_ordenado_cv`, respectively).

```
tasa_obs_ordenado_sd <- matrix(tasa_obs[,"sd"],
                               nrow = D,
                               ncol = P,byrow = TRUE)

colnames(tasa_obs_ordenado_sd) <- c("TD_mod_sd", "TO_mod_sd", "TP_mod_sd")
tasa_obs_ordenado_sd%<>% as.data.frame()
tasa_obs_ordenado_sd <- cbind(dam2 = indicador_dam1$dam2,
                              tasa_obs_ordenado_sd)
tasa_obs_ordenado_cv <- tasa_obs_ordenado_sd[,-1]/tasa_obs_ordenado[,-1]

colnames(tasa_obs_ordenado_cv) <- c("TD_mod_cv", "TO_mod_cv", "TP_mod_cv")

tasa_obs_ordenado_cv <- cbind(dam2 = indicador_dam1$dam2,
                              tasa_obs_ordenado_cv)

tasa_pred_ordenado_sd <- matrix(tasa_pred[,"sd"],
                                nrow = D1,
                                ncol = P,byrow = TRUE)

colnames(tasa_pred_ordenado_sd) <- c("TD_mod_sd", "TO_mod_sd", "TP_mod_sd")
tasa_pred_ordenado_sd%<>% as.data.frame()
tasa_pred_ordenado_sd <- cbind(dam2 = dam_pred$dam2, tasa_pred_ordenado_sd)

tasa_pred_ordenado_cv <- tasa_pred_ordenado_sd[,-1]/tasa_pred_ordenado[,-1]

colnames(tasa_pred_ordenado_cv) <- c("TD_mod_cv", "TO_mod_cv", "TP_mod_cv")

tasa_pred_ordenado_cv <- cbind(dam2 = dam_pred$dam2, tasa_pred_ordenado_cv)
```

The last step is to consolidate the bases obtained for the point estimate, standard

deviation and coefficient of variation.

```
tasa_obs_ordenado <-
  full_join(tasa_obs_ordenado, tasa_obs_ordenado_sd) %>%
  full_join(tasa_obs_ordenado_cv)

tasa_pred_ordenado <-
  full_join(tasa_pred_ordenado, tasa_pred_ordenado_sd) %>%
  full_join(tasa_pred_ordenado_cv)


estimaciones_obs <- full_join(indicador_dam1,
                              bind_rows(tasa_obs_ordenado, tasa_pred_ordenado))



saveRDS(object = estimaciones_obs, file = "Recursos/05_Empleo/11_estimaciones.rds")
tba(head(estimaciones_obs[,1:7],10))
```

dam2	n_upm	n_ocupado	n_desocupado	n_inactivo	Ocupado	Ocupado_se
0101	17	784	40	487	0.6000	0.0188
0201	11	733	40	420	0.6033	0.0213
0202	11	617	25	332	0.6448	0.0470
0203	12	645	17	363	0.6409	0.0184
0204	11	638	69	402	0.5574	0.0258
0207	11	473	22	374	0.5350	0.0193
0209	8	434	4	307	0.5666	0.0405
0211	13	713	11	396	0.6232	0.0314
0212	9	639	49	301	0.6179	0.0274
0302	10	676	91	327	0.5907	0.0285

4.10 Metodología de Benchmarking

1. People counts aggregated by dam2, people over 15 years of age.

```
conteo_pp_dam <- readRDS("Recursos/05_Empleo/12_censo_mrp.rds") %>%
   filter(age > 1)  %>%
   mutate(dam = str_sub(dam2,1,2)) %>%
  group_by(dam, dam2) %>%
  summarise(pp_dam2 = sum(n),.groups = "drop") %>%
  group_by(dam) %>%
```

```
  mutate(pp_dam = sum(pp_dam2))
head(conteo_pp_dam) %>% tba()
```

dam	dam2	pp_dam2	pp_dam
01	0101	26282	64195
01	0102	19385	64195
01	0103	18272	64195
01	0198	256	64195
02	0201	44833	443954
02	0202	35177	443954

2. Estimation of the `theta` parameter at the level that the survey is representative.

```
indicador_agregado <-
  diseno %>%
   mutate(dam = str_sub(dam2,1,2)) %>%
  group_by(dam) %>%
  filter(empleo %in% c(3:5)) %>%
  summarise(
    TO = survey_mean(empleo == 3,
                     vartype = c("se", "cv")),
    TD = survey_ratio(empleo == 4,
                      empleo != 5,
                      vartype = c("se",  "cv")),
    TP = survey_mean(empleo != 5,
                     vartype = c("se",  "cv"))
  ) %>% select(dam,TO,TD, TP)


tba(indicador_agregado)
```

dam	TO	TD	TP
01	0.5990	0.0609	0.6378
02	0.6065	0.0462	0.6359
03	0.5722	0.1573	0.6790
04	0.5459	0.1137	0.6160
05	0.5132	0.1029	0.5720
06	0.5328	0.1499	0.6267
07	0.5003	0.0683	0.5369
08	0.5741	0.1177	0.6506
09	0.5506	0.0726	0.5938
10	0.5815	0.0666	0.6230
11	0.6059	0.0400	0.6311
12	0.5746	0.0909	0.6320
13	0.5753	0.0951	0.6358
14	0.5946	0.0852	0.6500

Organizing the output as a vector.

```
temp <-
  gather(indicador_agregado, key = "agregado",
         value = "estimacion", -dam) %>%
  mutate(nombre = paste0("dam_", dam,"_", agregado))

Razon_empleo <- setNames(temp$estimacion, temp$nombre)
```

3. Define the weights by domains.

```
names_cov <-  "dam"
estimaciones_mod <- estimaciones %>%
  transmute(
    dam = str_sub(dam2,1,2),
    dam2,
    TO_mod,TD_mod,TP_mod) %>%
  inner_join(conteo_pp_dam ) %>%
  mutate(wi = pp_dam2/pp_dam)
```

4. Create dummy variables

```
estimaciones_mod %<>%
  fastDummies::dummy_cols(select_columns = names_cov,
                          remove_selected_columns = FALSE)

Xdummy <- estimaciones_mod %>% select(matches("dam_")) %>%
```

```
  mutate_at(vars(matches("_\\d")) ,
            list(TO = function(x) x*estimaciones_mod$TO_mod,
                 TD = function(x) x*estimaciones_mod$TD_mod,
                 TP = function(x) x*estimaciones_mod$TP_mod)) %>%
  select((matches("TO|TD|TP")))

# head(Xdummy) %>% tba()
```

Some validations carried out

```
colnames(Xdummy) == names(Razon_empleo)
data.frame(Modelo = colSums(Xdummy*estimaciones_mod$wi),
Estimacion_encuesta = Razon_empleo)
```

5. Calculate the weight for each level of the variable.

Occupancy Rate

```
library(sampling)
names_ocupado <- grep(pattern = "TO", x = colnames(Xdummy),value = TRUE)

gk_TO <- calib(Xs = Xdummy[,names_ocupado],
            d =  estimaciones_mod$wi,
            total = Razon_empleo[names_ocupado],
            method="logit",max_iter = 5000,)

checkcalibration(Xs = Xdummy[,names_ocupado],
                 d =estimaciones_mod$wi,
                 total = Razon_empleo[names_ocupado],
                 g = gk_TO)
```

Unemployment Rate

```
names_descupados <- grep(pattern = "TD", x = colnames(Xdummy),value = TRUE)

gk_TD <- calib(Xs = Xdummy[,names_descupados],
                   d =  estimaciones_mod$wi,
                   total = Razon_empleo[names_descupados],
                   method="logit",max_iter = 5000,)

checkcalibration(Xs = Xdummy[,names_descupados],
                 d =estimaciones_mod$wi,
```

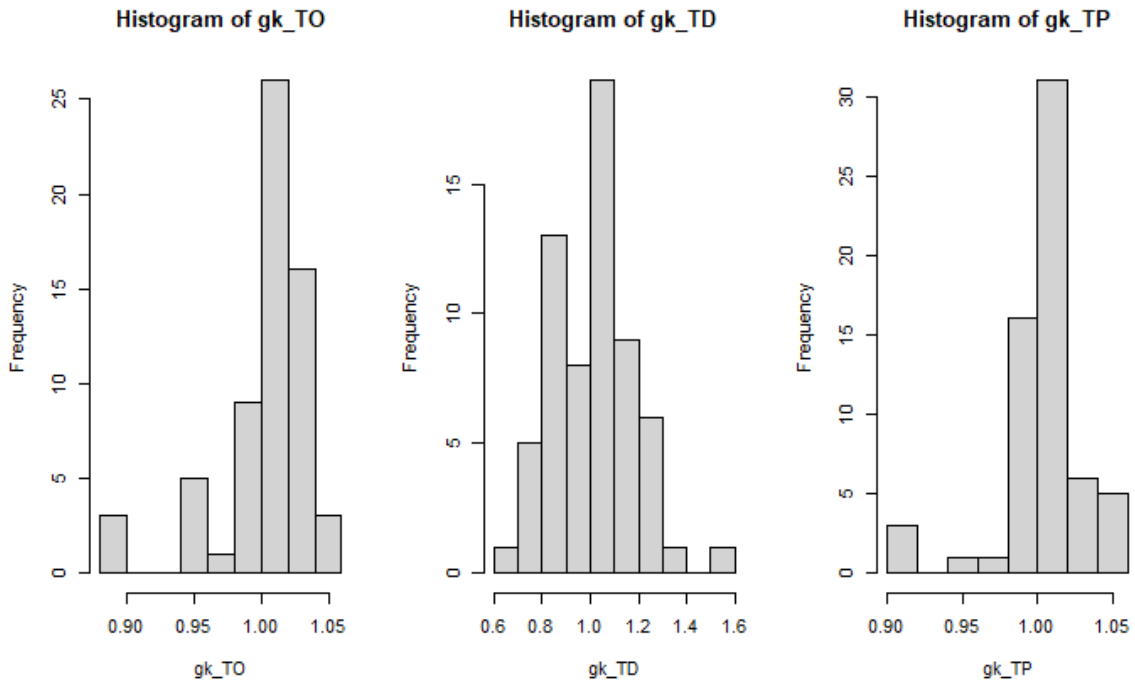
```
                  total = Razon_empleo[names_descupados],
                  g = gk_TD)
```

Participation Rate

```
names_inactivo <- grep(pattern = "TP", x = colnames(Xdummy),value = TRUE)

gk_TP <- calib(Xs = Xdummy[,names_inactivo],
                    d =  estimaciones_mod$wi,
                    total = Razon_empleo[names_inactivo],
                    method="logit",max_iter = 5000,)

checkcalibration(Xs = Xdummy[,names_inactivo],
                  d =estimaciones_mod$wi,
                  total = Razon_empleo[names_inactivo],
                  g = gk_TP)
```

6. Validate the results obtained.

```
par(mfrow = c(1,3))
hist(gk_TO)
hist(gk_TD)
hist(gk_TP)
```



7. Estimates adjusted by the weighter

```
estimacionesBench <- estimaciones_mod %>%
  mutate(gk_TO, gk_TD, gk_TP) %>%
  transmute(
    dam,dam2,
    wi,gk_TO, gk_TD, gk_TP,
    TO_Bench = TO_mod*gk_TO,
    TD_Bench = TD_mod*gk_TD,
    TP_Bench = TP_mod*gk_TP
  )
```

8. Validation of results.

```
tabla_validar <- estimacionesBench %>%
  group_by(dam) %>%
  summarise(TO_Bench = sum(wi*TO_Bench),
            TD_Bench = sum(wi*TD_Bench),
            TP_Bench = sum(wi*TP_Bench)) %>%
  inner_join(indicador_agregado)
tabla_validar %>% tba()
```

dam	TO_Bench	TD_Bench	TP_Bench	TO	TD	TP
01	0.5990	0.0609	0.6378	0.5990	0.0609	0.6378
02	0.6065	0.0462	0.6359	0.6065	0.0462	0.6359
03	0.5722	0.1573	0.6790	0.5722	0.1573	0.6790
04	0.5459	0.1137	0.6160	0.5459	0.1137	0.6160
05	0.5132	0.1029	0.5720	0.5132	0.1029	0.5720
06	0.5328	0.1499	0.6267	0.5328	0.1499	0.6267
07	0.5003	0.0683	0.5369	0.5003	0.0683	0.5369
08	0.5741	0.1177	0.6506	0.5741	0.1177	0.6506
09	0.5506	0.0726	0.5938	0.5506	0.0726	0.5938
10	0.5815	0.0666	0.6230	0.5815	0.0666	0.6230
11	0.6059	0.0400	0.6311	0.6059	0.0400	0.6311
12	0.5746	0.0909	0.6320	0.5746	0.0909	0.6320
13	0.5753	0.0951	0.6358	0.5753	0.0951	0.6358
14	0.5946	0.0852	0.6500	0.5946	0.0852	0.6500

9. Save results

```
estimaciones <- inner_join(estimaciones,estimacionesBench)
saveRDS(object = estimaciones, file = "Recursos/05_Empleo/15_estimaciones_Bench.rds")
```

4.10.1 Grafico de validación del Benchmarking

1. Perform model estimates before and after Benchmarking

```
estimaciones_agregada <- estimaciones %>%
  group_by(dam) %>%
  summarise(
    TO_mod = sum(wi * TO_mod),
    TD_mod = sum(wi * TD_mod),
    TP_mod = sum(wi * TP_mod),
    TO_bench = sum(wi * TO_Bench),
    TD_bench = sum(wi * TD_Bench),
    TP_bench = sum(wi * TP_Bench))
```

2. Obtain the confidence intervals for the direct estimates

```
indicador_agregado <-
  diseno %>%
   mutate(dam = str_sub(dam2,1,2)) %>%
  group_by(dam) %>%
  filter(empleo %in% c(3:5)) %>%
  summarise(
     nd = unweighted(n()),
    TO = survey_mean(empleo == 3,
                     vartype = c("ci")),
    TD = survey_ratio(empleo == 4,
                      empleo != 5,
                      vartype = c("ci")),
    TP = survey_mean(empleo != 5,
                     vartype = c("ci"))
  )

data_plot <- left_join(estimaciones_agregada, indicador_agregado)
```

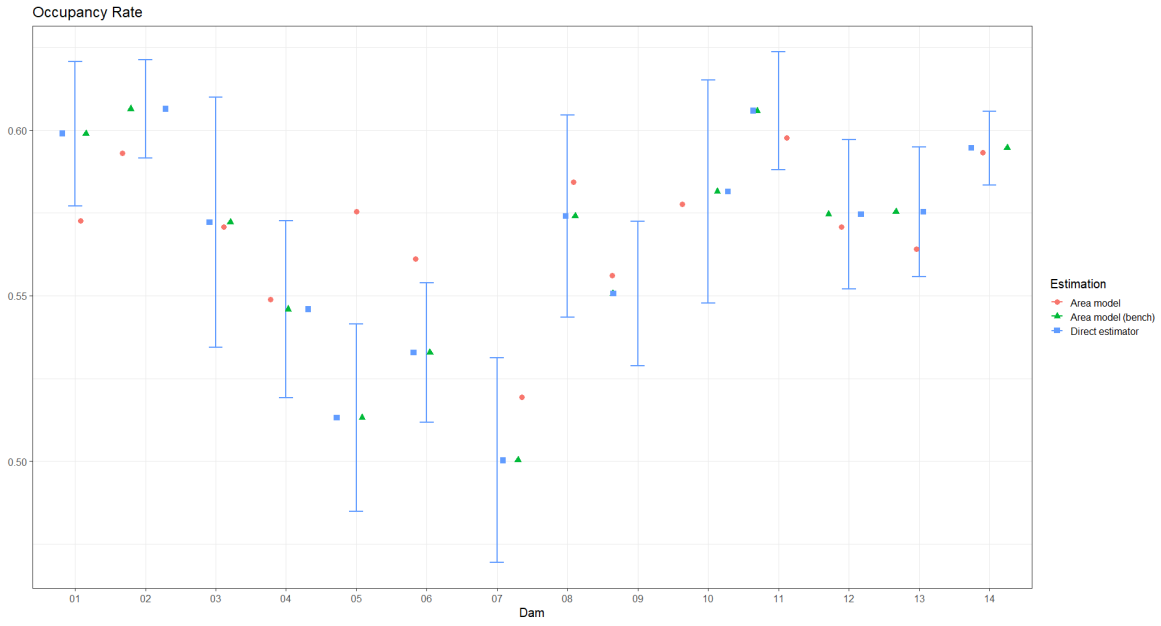
3. Select the results for an indicator (Occupancy rate)

```
temp_TO <- data_plot %>% select(dam,nd, starts_with("TO"))


temp_TO_1 <- temp_TO %>% select(-TO_low, -TO_upp) %>%
  gather(key = "Estimacion",value = "value", -nd,-dam) %>%
  mutate(Estimacion = case_when(Estimacion == "TO_mod" ~ "Area model",
                                Estimacion == "TO_bench" ~ "Area model (bench)",
                                Estimacion == "TO"~ "Direct estimator"))
```

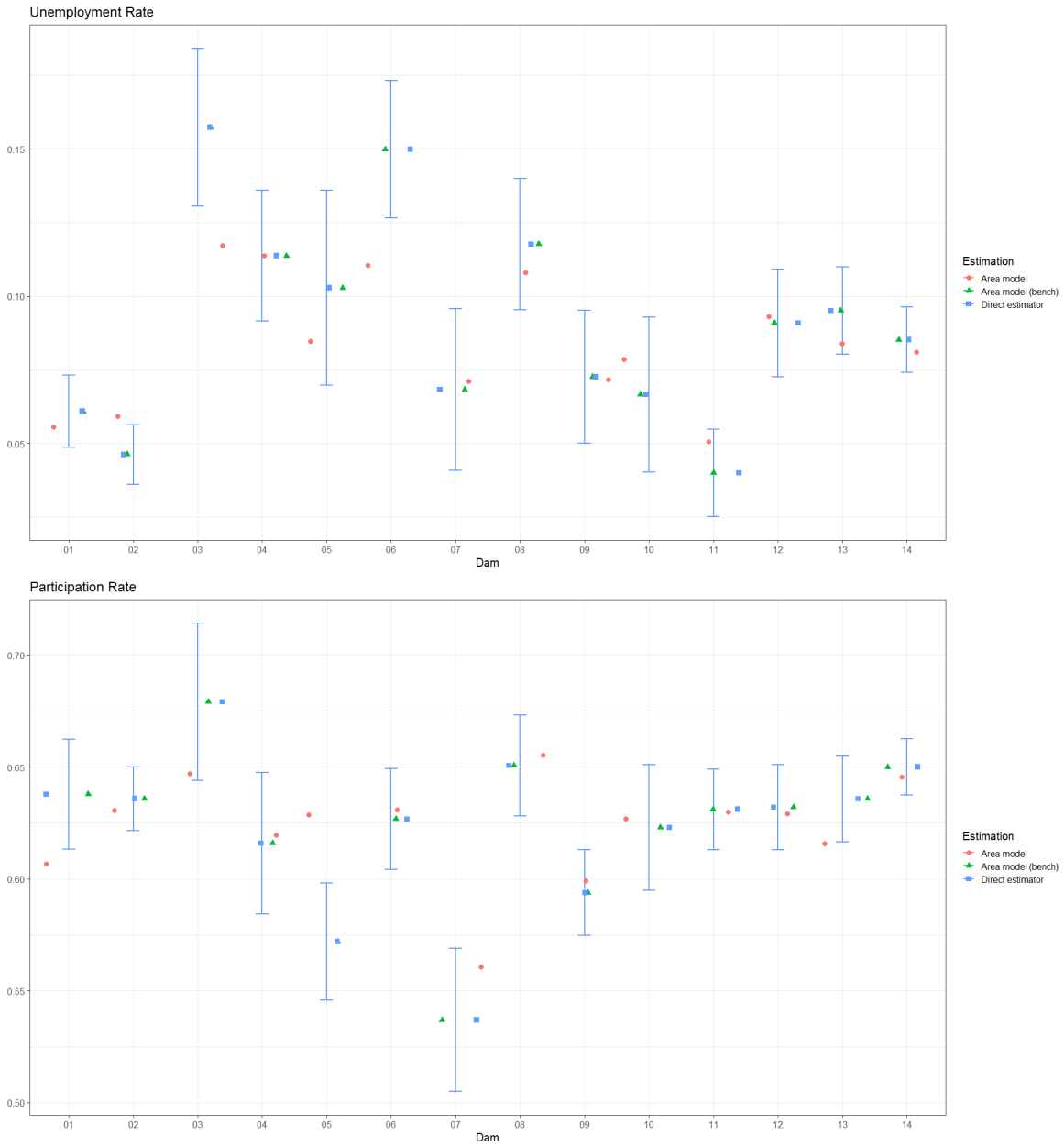
```
lims_IC_ocupado <-  temp_TO %>%
  select(dam,nd,value = TO,TO_low, TO_upp) %>%
  mutate(Estimacion = "Direct estimator")
```

4. Make the graph

```
p_TO <- ggplot(temp_TO_1,
                    aes(
                      x = fct_reorder2(dam, dam, nd),
                      y = value,
                      shape = Estimacion,
                      color = Estimacion
                    )) +
  geom_errorbar(
    data = lims_IC_ocupado,
    aes(ymin = TO_low ,
        ymax = TO_upp, x = dam),
    width = 0.2,
    linewidth = 1
  )  +
  geom_jitter(size = 3) +
  labs(x = "Dam", title = "Occupancy Rate", y = "",
       color= "Estimation", shape = "Estimation")
```



5. Repeat the process with the other indicators.



4.11 Labor market maps.

The code loads the libraries `sf` and `tmap`. Subsequently, it reads a shapefile containing geographic information and uses the 'inner_join' function to merge it with previously calculated survey estimates.

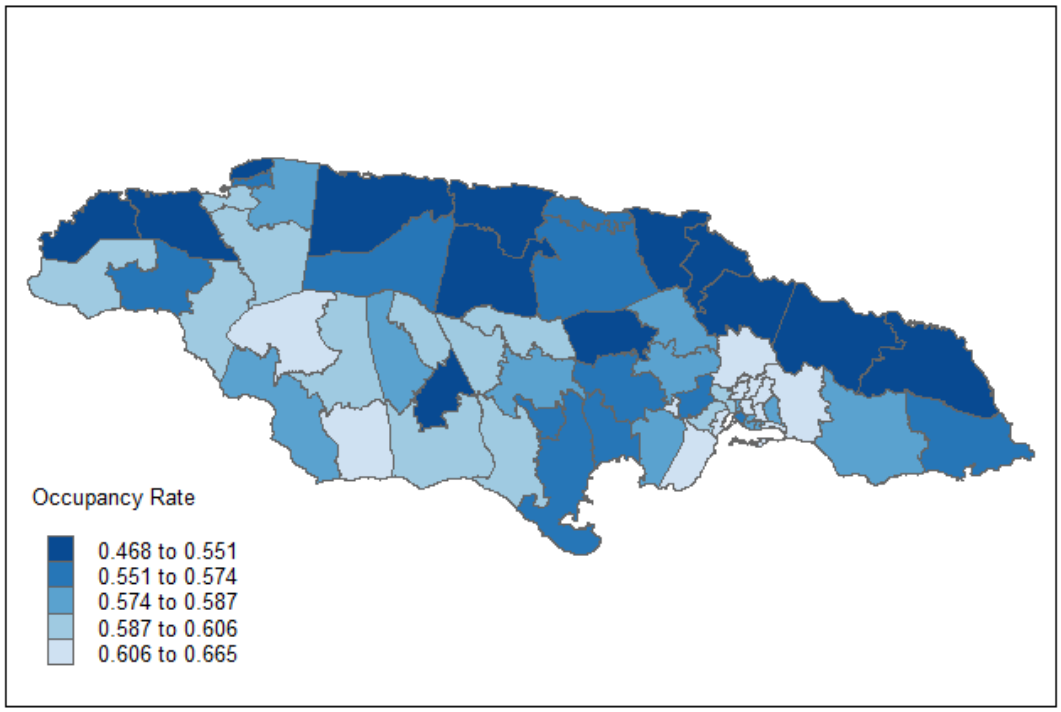
```
library(sf)
library(tmap)
ShapeSAE <- read_sf("Shapefile/JAM2_cons.shp")
```

```
P1_empleo <- tm_shape(ShapeSAE %>%
                            inner_join(estimaciones))
```

Occupancy Rate

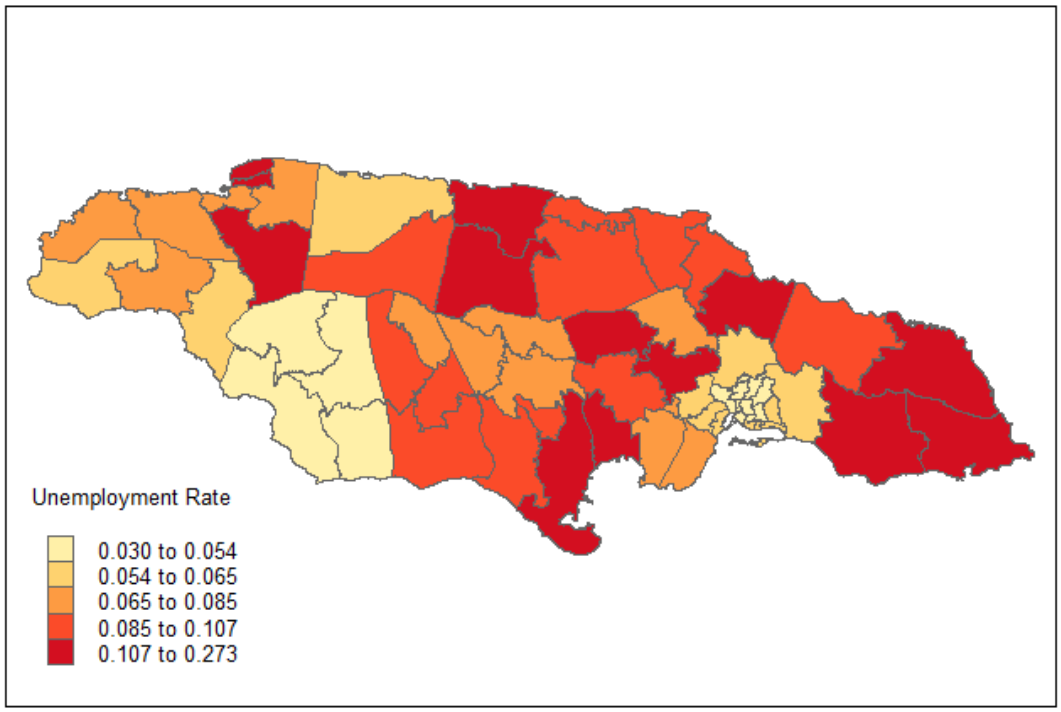
The following code creates the map of the occupancy rate using the tmap library. It begins by setting some global mapping options with tmap_options. Then, it uses the P1_employment variable as the base and adds polygon layers using tm_polygons. In this layer, the variable "TO_Bench" is defined as representing the occupancy rate, a title is assigned to the map, a color palette ("Blues") is chosen, and the data classification style ("quantile") is set. Subsequently, with tm_layout, various aspects of the map's presentation are defined, such as the position and size of the legend, the aspect ratio, and the text size of the legend. Finally, the resulting map is displayed.

```
tmap_options(check.and.fix = TRUE)
Mapa_TO <-
  P1_empleo +
  tm_polygons("TO_Bench",
          title = "Occupancy Rate",
          palette = "-Blues",
          colorNA = "white",
          style= "quantile") +
  tm_layout(
    legend.only = FALSE,
    legend.height = -0.3,
    legend.width = -0.5,
    asp = 1.5,
    legend.text.size = 3,
    legend.title.size = 3)

Mapa_TO
```

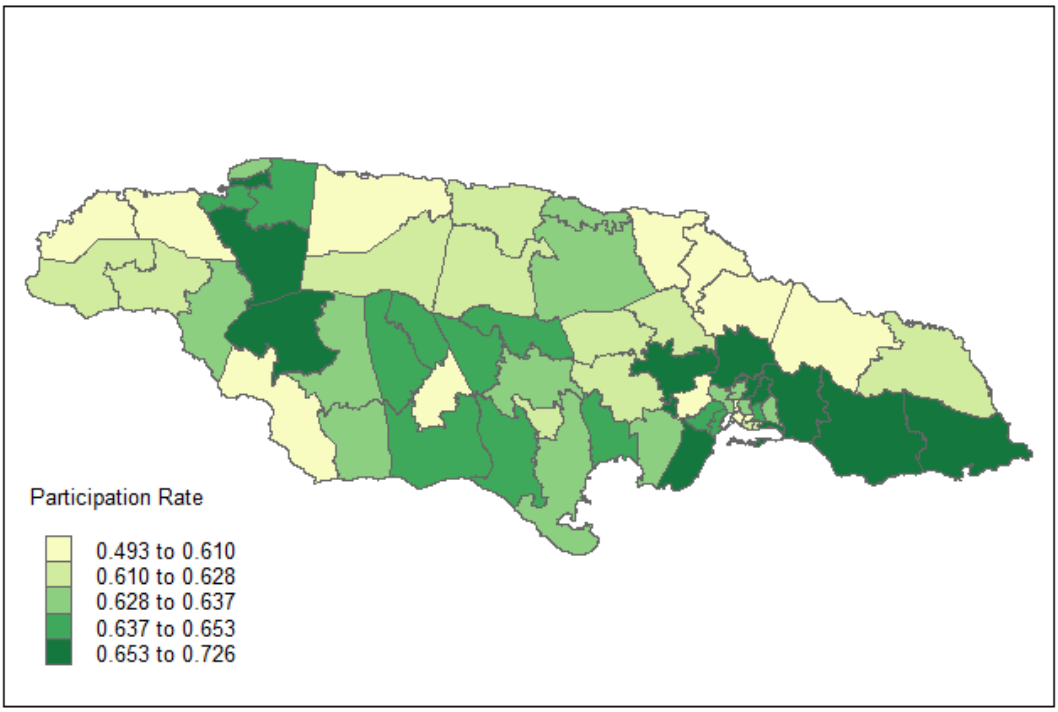


Unemployment Rate

```
Mapa_TD <-
  P1_empleo + tm_polygons(
    "TD_Bench",
    title =  "Unemployment Rate",
    palette = "YlOrRd",
    colorNA = "white",
    style= "quantile"
  ) + tm_layout(
    legend.only = FALSE,
    legend.height = -0.3,
    legend.width = -0.5,
    asp = 1.5,
    legend.text.size = 3,
    legend.title.size = 3)


Mapa_TD
```




```
Mapa_TP <-
  P1_empleo + tm_polygons(
    "TP_Bench",
    title =  "Participation Rate",
    colorNA = "white",
    palette = "YlGn",
    style= "quantile"
  ) + tm_layout(
    legend.only = FALSE,
    legend.height = -0.3,
    legend.width = -0.5,
    asp = 1.5,
    legend.text.size = 3,
    legend.title.size = 3)

Mapa_TP
```


Participation Rate
0.493 to 0.610
0.610 to 0.628
0.628 to 0.637
0.637 to 0.653
0.653 to 0.726