GeoLink - Zonal Statistics Example

Luciano Perfetti Villa Nikos Tzavidis

```
## library(GeoLink)
devtools::load_all()
library(tidyverse)
library(sf)
library(emdi)

## Shapefile
data(shp_dt)
## Survey data
data(hhgeo_dt)
## Population data
data(popHDX_dt)
```

Introduction

The `GeoLink` package is a powerful tool designed to facilitate access to a wide range of geospatial data and enable advanced spatial analysis techniques. This package serves as a bridge between various geospatial databases and the R programming environment, allowing users to easily obtain and manipulate spatial information for diverse applications.

One particularly valuable application of the GeoLink package is in the field of poverty estimation using small-area estimation (SAE) techniques (see Molina and Rao 2010; Tzavidis et al. 2018). SAE is a statistical modeling approach that allows for the estimation of data for small domains where direct estimates are affected by small sizes of the sample, providing more granular insights than traditional survey methods. By combining the geospatial data capabilities of `GeoLink` with SAE methodologies, researchers and policymakers can generate more frequent and detailed poverty estimates when there is no access to the census or it is outdated.

To demonstrate the practical utility of the `GeoLink` package, we present an application focused on estimating poverty in Nigeria at the second administrative level. This case study showcases how `GeoLink` can be used to access relevant geospatial data, such as nighttime light intensity,

population density, and other metrics, which serve as covariates in the SAE model (see Masaki et al. 2022; Newhouse et al. 2025; Edochie et al. 2024). By integrating these geospatial indicators with available survey data, we can produce high-resolution poverty maps for Nigeria, offering valuable insights for targeted policy interventions and resource allocation.

This document will guide users through the process of utilizing GeoLink to obtain necessary geospatial data, implement SAE techniques, and visualize the resulting poverty estimates for Nigeria. The approach demonstrated here can be adapted to other countries or regions, making `GeoLink` a versatile tool for researchers and practitioners in the field of development economics and policy analysis.

Data

The data from Nigeria provides us with an interesting situation where we are able to obtain geospatial data at different levels, namely household-level and grid-generated or administrative zonal statistics. The household-level data can be extracted if we have access to the georeferenced location of the household, either by using a buffer zone around the location of the household or by creating a grid to obtain zonal statistics. On the other hand, if we have the location of the households in terms of administrative boundaries, we can obtain zonal statistics using a shapefile of the corresponding admin level.

In this vignette, we will explore the three scenarios. First, we will create a grid using Nigeria's original shapefile (`shp_dt`). The `gengrid2` function allows us to create a squared or polygonal grid with a defined cell size.

```
gengrid2(
  shp_dt = shp_dt,
  grid_size = 0.025,
  sqr = TRUE
) -> shp_dt_grid
```

The `hhgeo_dt` contains several geospatial variables, but we only want to keep the household identifier, administrative levels, location, sampling weights and total consumption for now. Therefore, each row contains a household with the location and the respective consumption.

```
hhgeo_dt %>%
  dplyr::select(
    hhid,                  ## Household identifier
    starts_with("ADM"),    ## Administrative level identifier
    geometry,               ## geometry with household location
    popw,                  ## population weights
    wt_wave4,               ## household weights
```

```
    totcons_adj_norm   ## total household consumption
    ) -> survey_dt
st_geometry(survey_dt) <- hhgeo_dt$geometry
```

Nighttime lights

We first obtain the annual average of the nighttime lights for 2018. However, there are other options in terms of frequency (i.e., monthly) or indicator (e.g., median, maximum, coefficient of variation, among others).

```
geolink_ntl(start_date = "2018-01-01",
            end_date = "2018-12-31",
            annual_version = "v21",
            indicator = "average_masked",
            shp_dt = shp_dt
            ) -> geolink_data_admin2

geolink_ntl(start_date = "2018-01-01",
            end_date = "2018-12-31",
            annual_version = "v21",
            indicator = "average_masked",
            shp_dt = shp_dt_grid) -> geolink_data_grid

geolink_ntl(start_date = "2018-01-01",
            end_date = "2018-12-31",
            annual_version = "v21",
            indicator = "average_masked",
            survey_dt = survey_dt,
            buffer_size = 600) -> geolink_data_survey_hh
```

CHIRPS

For the rainfall data, we will use a similar approach. We will obtain the annual average rainfall for 2018, but, as mentioned before, there are other frequencies and indicators to use.

```
geolink_chirps(start_date = "2018-01-01",
               end_date = "2018-12-31",
               time_unit = "annual",
               extract_fun = "mean",
               shp_dt = geolink_data_admin2
```

```
                ) -> geolink_data_admin2

geolink_chirps(start_date = "2018-01-01",
                end_date = "2018-12-31",
                time_unit = "annual",
                extract_fun = "mean",
                shp_dt = geolink_data_grid
                ) -> geolink_data_grid

geolink_chirps(start_date = "2018-01-01",
                end_date = "2018-12-31",
                time_unit = "annual",
                extract_fun = "mean",
                survey_dt = geolink_data_survey_hh,
                buffer_size = 500) -> geolink_data_survey_hh
```

Population

To estimate the SAE estimates at the desired level we need some information about the population in each level. Usually, under unit-level models we have the complete census (or administrative data) to predict the consumption of each household in the area of interest. When using geospatial data, we need a population estimate at the area of interest or aggregation level (i.e., zonal statistic). With this information, we can obtain the estimates acknowledging the number of household in each area and zonal statistic.

`GeoLink` allows us to extract the WorldPop gridded population estimates in a given year. The following code obtains the total population for each case, i.e. the shapefile (admin2), the generated grid, and the household location. Note that we can also use population as a predictor of the mean consumption; for instance, we could use it as a measure of population density.

```
geolink_population(start_year = 2018,
                   end_year = 2018,
                   iso_code = "NGA",
                   UN_adjst = "N",
                   constrained = "N",
                   shp_dt = geolink_data_admin2,
                   extract_fun = "sum"
                   ) -> geolink_data_admin2

geolink_population(start_year = 2018,
                     end_year = 2018,
```

```
                        iso_code = "NGA",
                        UN_adjst = "N",
                        constrained = "N",
                        shp_dt = geolink_data_grid,
                        extract_fun = "sum"
                        ) -> geolink_data_grid

geolink_population(start_year = 2018,
                           end_year = 2018,
                           iso_code = "NGA",
                           UN_adjst = "N",
                           constrained = "N",
                           survey_dt = geolink_data_survey_hh,
                           buffer_size = 500,
                           extract_fun = "sum"
                           ) -> geolink_data_survey_hh
```

Elevation

An additional layer to extract is the elevation of the terrain in the zonal statistic or the location of the households. We will continue to use the same buffer zone and the mean as the extraction function.

```
geolink_elevation(iso_code = "NGA",
                  shp_dt = geolink_data_admin2) -> geolink_data_admin2

geolink_elevation(iso_code = "NGA",
                  shp_dt = geolink_data_grid) -> geolink_data_grid

geolink_elevation(iso_code = "NGA",
                    survey_dt = geolink_data_survey_hh,
                    buffer_size = 500) -> geolink_data_survey_hh
```

Buildings (WorldPop)

WorldPop also produces a variable that estimates different measures of buildings in a given area (more information here). Such variables may help to differentiate highly populated areas where we could expect higher consumption due to urban economic development and less populated rural areas where the consumption might be lower. The survey was obtained in 2018, so we

will obtain the version 1.1 that is closest to that year. Also, due to the raster was initially created, we will need to expand the buffer zone in order to avoid missing values.

```
geolink_buildings(iso_code = "NGA",
                  version = "v1.1",
                  shp_dt = geolink_data_admin2) -> geolink_data_admin2

geolink_buildings(iso_code = "NGA",
                    version = "v1.1",
                    shp_dt = geolink_data_grid) -> geolink_data_grid

geolink_buildings(iso_code = "NGA",
                    version = "v1.1",
                    survey_dt = geolink_data_survey_hh,
                    buffer_size = 1000) -> geolink_data_survey_hh
```

DataFrame with Zonal Statistics

The data frame `geolink_data_admin2` contains all the layers of the geospatial data we obtained from the previous lines. Each row is a district, but we want to link each household in `survey_dt` with the value of the zonal statistic. Hence, we would ideally have the original `survey_dt` with consumption per household, and its geospatial predictors.

```
geolink_data_admin2 <- st_drop_geometry(geolink_data_admin2)
geolink_data_survey_hh <- st_drop_geometry(geolink_data_survey_hh)

geolink_data_admin2 %>%
  dplyr::select(ADM2_PCODE, ntl_annual1average_masked, rainfall_annual1,
         population_2018, NGA_elv_msk, count, cv_area, cv_length,
         density, mean_area, mean_length,
         total_area, total_length, urban) %>%
  right_join(survey_dt, by = c("ADM2_PCODE")) %>% #, multiple = "first", unmatched = "drop")
  distinct(hhid, ADM2_PCODE, .keep_all = TRUE) -> geolink_data_survey

geolink_data_grid %>%
  dplyr::select(poly_id, poly_area, ntl_annual1average_masked, rainfall_annual1,
         population_2018, NGA_elv_msk, count, cv_area, cv_length,
         density, mean_area, mean_length,
         total_area, total_length, urban) %>%
  st_join(x = survey_dt, y = ., join = st_within) -> geolink_data_survey_grid
```

Although there should be a thorough process to determine different possible transformations for each predictor, we will assume a log-shift transformation to improve the estimation of the model and reduce the impact of outliers in the covariates.

```
geolink_data_survey %>%
  mutate(across(c( ntl_annual1average_masked, rainfall_annual1,
         NGA_elv_msk, count, cv_area, cv_length,
         density, mean_area, mean_length,
         total_area, total_length), \(x) log(x + 0.001)),
         urban = if_else(urban < 0.5, 0, 1),
         ADM1_PCODE = factor(ADM1_PCODE),
         ADM2_PCODE = factor(ADM2_PCODE, levels = unique(geolink_data_admin2$ADM2_PCODE)),
         population_2018 = if_else(is.na(population_2018) | population_2018 == 0, 1, populati

geolink_data_survey_grid %>%
    mutate(across(c( ntl_annual1average_masked, rainfall_annual1,
             NGA_elv_msk, count, cv_area, cv_length,
             density, mean_area, mean_length,
             total_area, total_length), \(x) log(x + 0.001)),
             urban = if_else(urban < 0.5, 0, 1),
             ADM1_PCODE = factor(ADM1_PCODE),
             ADM2_PCODE = factor(ADM2_PCODE, levels = unique(geolink_data_grid$ADM2_PCODE)),
             population_2018 = if_else(is.na(population_2018) | population_2018 == 0, 1, popu

geolink_data_survey_hh %>%
  mutate(across(c( ntl_annual1average_masked, rainfall_annual1,
         NGA_elv_msk, count, cv_area, cv_length,
         density, mean_area, mean_length,
         total_area, total_length), \(x) log(x + 0.001)),
         urban = if_else(urban < 0.5, 0, 1),
         ADM1_PCODE = factor(ADM1_PCODE),
         ADM2_PCODE = factor(ADM2_PCODE, levels = unique(geolink_data_admin2$ADM2_PCODE)),
         population_2018 = if_else(is.na(population_2018) | population_2018 == 0, 1, populati
```

The `geolink_data_admin2` and `geolink_data_survey_grid` data frames can be seen as the census, as they contain all the possible values of the covariates for each area or cell. To match the survey, we will transform the variables in these data frames. We will explain the additional adjustment to the population variable later.

```
geolink_data_admin2 %>%
  mutate(across(c( ntl_annual1average_masked, rainfall_annual1,
         NGA_elv_msk, count, cv_area, cv_length,
```

```
        density, imagery_year, mean_area, mean_length,
        total_area, total_length), \(x) log(x + 0.001)),
        urban = if_else(urban < 0.5, 0, 1),
        ADM1_PCODE = factor(ADM1_PCODE),
        ADM2_PCODE = factor(ADM2_PCODE),
        population_2018 = if_else(is.na(population_2018) | population_2018 <= 5, 1, populati
    ) -> geolink_data_admin2_log

geolink_data_grid %>%
  mutate(across(c( ntl_annual1average_masked, rainfall_annual1,
        NGA_elv_msk, count, cv_area, cv_length,
        density, imagery_year, mean_area, mean_length,
        total_area, total_length), \(x) log(x + 0.001)),
        urban = if_else(urban < 0.5, 0, 1),
        ADM1_PCODE = factor(ADM1_PCODE),
        ADM2_PCODE = factor(ADM2_PCODE),
        population_2018 = if_else(is.na(population_2018) | population_2018 <= 5, 1, populati
    ) -> geolink_data_grid_log
```

Population from external sources

We assume that the population of each admin2 area is obtained from a reliable external source (e.g., census, surveys, population projections, among others). In this case, we obtained it from the Humanitarian Data Exchange. Therefore, the results cannot be compared to official results but are good enough for the example. We merge the data frames so that we now have covariates and population weights for each admin2. It is worth mentioning that the population is in terms of people and not households, hence we are assuming an average household size of 5 members. In the `ebp` function of the `emdi` package, the `pop_weights` argument is used to provide a weighted indicator (e.g., mean or head count). Finally, we will estimate different poverty indicators at the district level.

```
geolink_data_admin2_log %>%
  left_join(popHDX_dt %>% dplyr::select(ADM2_PCODE, T_TL), by = c("ADM2_PCODE"), multiple = "
  mutate(pop_weights = if_else(is.na(T_TL), 1, T_TL/5),
        ADM2_PCODE = factor(ADM2_PCODE)) -> geolink_data_admin2_log
```

```
ebp_nga <- emdi::ebp(
                  fixed = totcons_adj_norm ~ ntl_annual1average_masked + rainfall_annual1 +
                    NGA_elv_msk + count + cv_area + cv_length +
                    density + mean_area + mean_length +
                    total_area + total_length + urban + ADM1_PCODE +
```

```
                  urban:( ntl_annual1average_masked + rainfall_annual1 +
                     NGA_elv_msk + count + cv_area + cv_length +
                     density + mean_area + mean_length +
                     total_area + total_length),
                  pop_data = geolink_data_admin2_log,
                  pop_domains = "ADM2_PCODE",
                  pop_weights = "pop_weights",
                  smp_data = geolink_data_survey_log,
                  smp_domains = "ADM2_PCODE",
                  transformation = "log",
                  na.rm = TRUE,
                  threshold = 137430        ## Taken from World Bank poverty assessment
                 )

ebp_nga_grid <- emdi::ebp(
                   fixed = totcons_adj_norm ~ ntl_annual1average_masked + rainfall_annual1 +
                     NGA_elv_msk + count + cv_area + cv_length +
                     density + mean_area + mean_length +
                     total_area + total_length + urban + ADM1_PCODE +
                   urban:( ntl_annual1average_masked + rainfall_annual1 +
                     NGA_elv_msk + count + cv_area + cv_length +
                     density + mean_area + mean_length +
                     total_area + total_length),
                   pop_data = st_drop_geometry(geolink_data_grid_log),
                   pop_domains = "ADM2_PCODE",
                   pop_weights = "population_2018",
                   smp_data = st_drop_geometry(geolink_data_survey_grid),
                   smp_domains = "ADM2_PCODE",
                   transformation = "log",
                   na.rm = TRUE,
                   threshold = 137430        ## Taken from World Bank poverty assessment
                  )

ebp_nga_hh <- emdi::ebp(
                    fixed = totcons_adj_norm ~ ntl_annual1average_masked + rainfall_annual1 +
                     NGA_elv_msk + count + cv_area + cv_length +
                     density + mean_area + mean_length +
                     total_area + total_length + urban + ADM1_PCODE +
                   urban:( ntl_annual1average_masked + rainfall_annual1 +
                     NGA_elv_msk + count + cv_area + cv_length +
                     density + mean_area + mean_length +
                     total_area + total_length),
```

```
                  pop_data = geolink_data_admin2_log,
                  pop_domains = "ADM2_PCODE",
                  pop_weights = "pop_weights",
                  smp_data = geolink_data_survey_hh_log,
                  smp_domains = "ADM2_PCODE",
                  transformation = "log",
                  na.rm = TRUE,
                  threshold = 137430        ## Taken from World Bank poverty assessment
                 )
```

Population from geospatial data

An alternative scenario might be when the researcher does not have access to population figures within each area. In such a case, alternative data sources like WorldPop may become an option. WorldPop produces gridded population figures, which allows us to obtain estimates of population in each area. However, since we are working with household level data, we need to transform the population into number of households. For this example we will assume a simple solution of 5 members per household, although there might be better ways to approach such transformation. As before, we use the `pop_weights` argument to include the estimated population into the `ebp` function.

```
ebp_nga_worldpop <- emdi::ebp(
                   fixed = totcons_adj_norm ~ ntl_annual1average_masked + rainfall_annual1 +
                     NGA_elv_msk + count + cv_area + cv_length +
                     density + mean_area + mean_length +
                     total_area + total_length + urban + ADM1_PCODE +
                   urban:( ntl_annual1average_masked + rainfall_annual1 +
                     NGA_elv_msk + count + cv_area + cv_length +
                     density + mean_area + mean_length +
                     total_area + total_length),
                   pop_data = geolink_data_admin2_log,
                   pop_domains = "ADM2_PCODE",
                   pop_weights = "population_2018",
                   smp_data = geolink_data_survey_log,
                   smp_domains = "ADM2_PCODE",
                   transformation = "log",
                   na.rm = TRUE,
                   threshold = 137430
                  )
```

Comparison with direct estimates

The `emdi` package also estimates the direct estimates at the target areas. Note, however, that in SAE, we would expect these estimates' Mean Square Error (MSE) to be higher due to the small sample in the domains. The direct estimates are unbiased, on average, so we could use them as a benchmark to identify any potential bias from the estimates obtained from the geospatial data models. Nonetheless, it is important to mention that in this case some of the direct estimates, at admin2, have samples of only 1 household. Therefore, we cannot expect to have unbiased direct estimates in these cases. For now, we will use the direct estimates as comparison but it is not recommended when we have such small samples.

```
direct_nga <- emdi::direct(
                      y = "totcons_adj_norm",
                      smp_data = geolink_data_survey_log %>%
                        filter(!is.na(totcons_adj_norm)),
                      smp_domains = "ADM2_PCODE",
                      weights = "wt_wave4",
                      threshold = 137430
                    )
```

We can obtain some comparison plots using the `compare_plot` of the `emdi` package. The first plot compares the direct estimates with the EBP model by illustrating the correlation between both estimates. The second plot shows the levels of each estimate arranged by area.

```
emdi::compare_plot(model = ebp_nga, direct = direct_nga, indicator = "Mean")
```

```
## Not all domains contained in the model estimation have been found in the direct estimatio
```

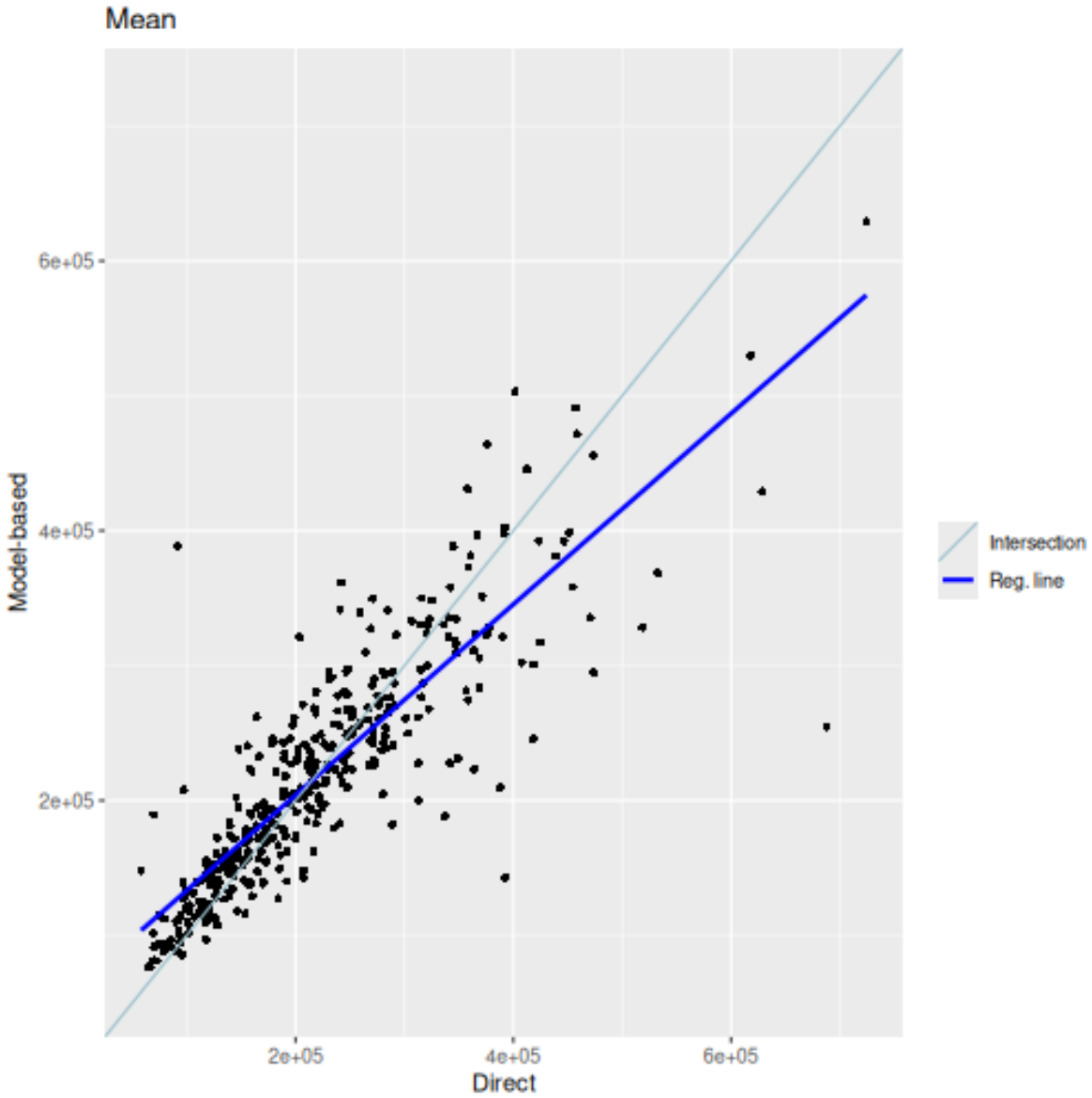

Figure 1: Comparison of Head Count Ratio with direct estimates using census population

```
## Press [enter] to continue
```

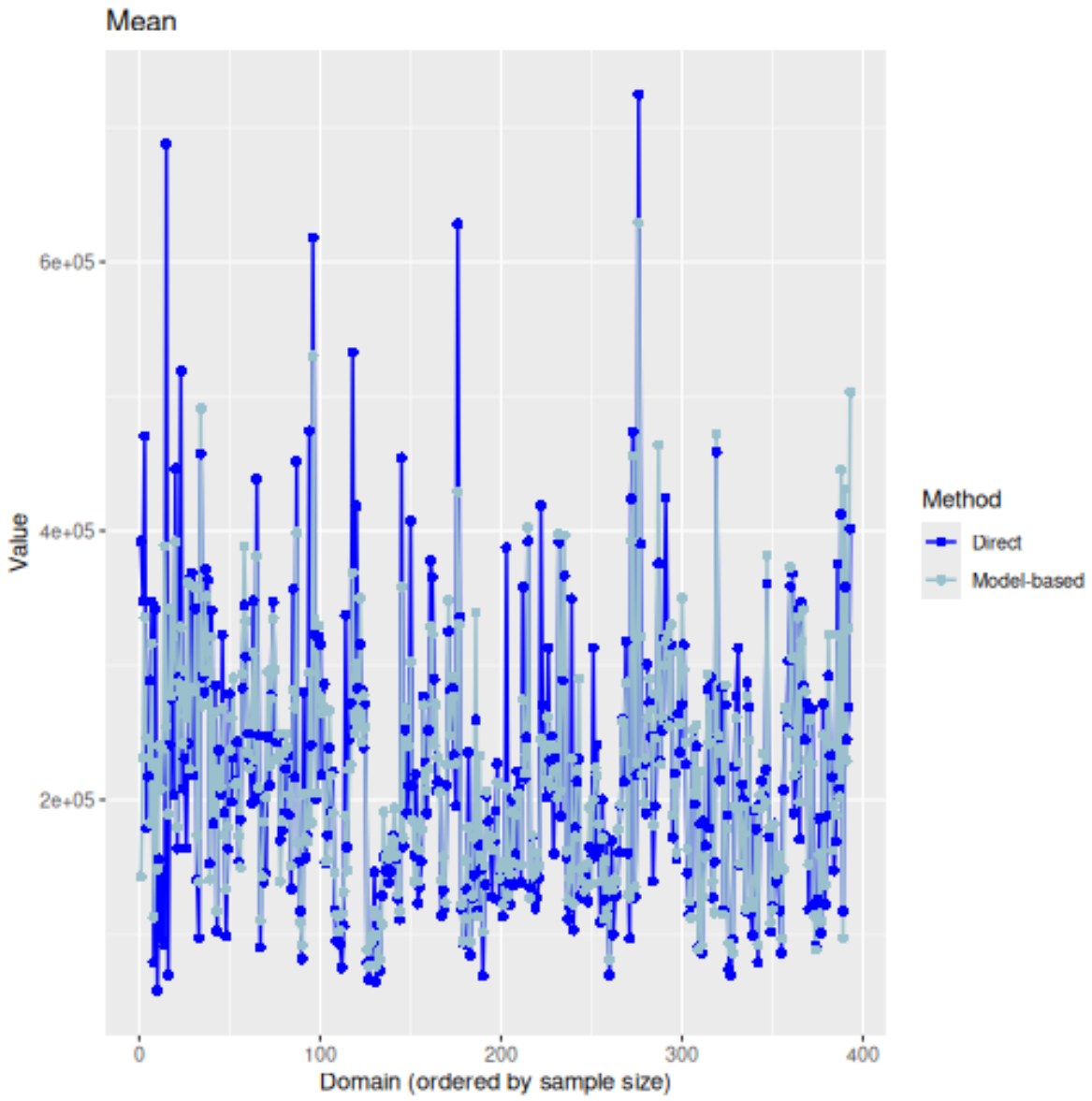

Figure 2: Arranged comparison of Head Count Ratio with direct estimates using census population

To compare all models, we can define a simple function that extracts the results from each `ebp` and `direct` result. The following code extracts the `Mean` and `Head_Count` ratio from the objects and then joins these in a `tibble` with a variable holding the model's name. We could replicate the same to extract the MSE by changing `ind` for `MSE`.

```
extract_ebp <- function(ebps, indicators = c("Mean", "Head_Count")){
  map(ebps,
      \(x) {
        x$ind[, c("Domain", indicators)]
      }) %>%
    bind_rows(.id = "Model")
}

results <- extract_ebp(list(
  "Direct" = direct_nga,
  "EBP_Zonal" = ebp_nga,
  "EBP_Grid" = ebp_nga_grid,
  "EBP_HH" = ebp_nga_hh,
  "EBP_WorldPop" = ebp_nga_worldpop))
```

?@fig-comparison-mean-ebps and **?@fig-comparison-hcr-ebps** show the different models compared to the direct estimates. The estimates lie close to the diagonal, suggesting that the models track the direct estimates slightly well. Note that the models do not reasonably estimate the highest values.

```
results %>%
  dplyr::select(-Head_Count) %>%
  pivot_wider(id_cols = c(Domain), names_from = Model, values_from = Mean) %>%
  pivot_longer(starts_with("EBP"), names_to = "Model", values_to = "Mean") %>%
  ggplot(aes(x = Direct, y = Mean, color = Model)) +
  geom_point() +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  facet_wrap(~Model, scales = "free") +
  labs(
    title = "Comparison of mean consumption estimates from EBP models with direct estimates"
    x = "Direct",
    y = "Model"
  )
```

```
## Warning: Removed 1520 rows containing missing values or values outside the scale range (`g
```

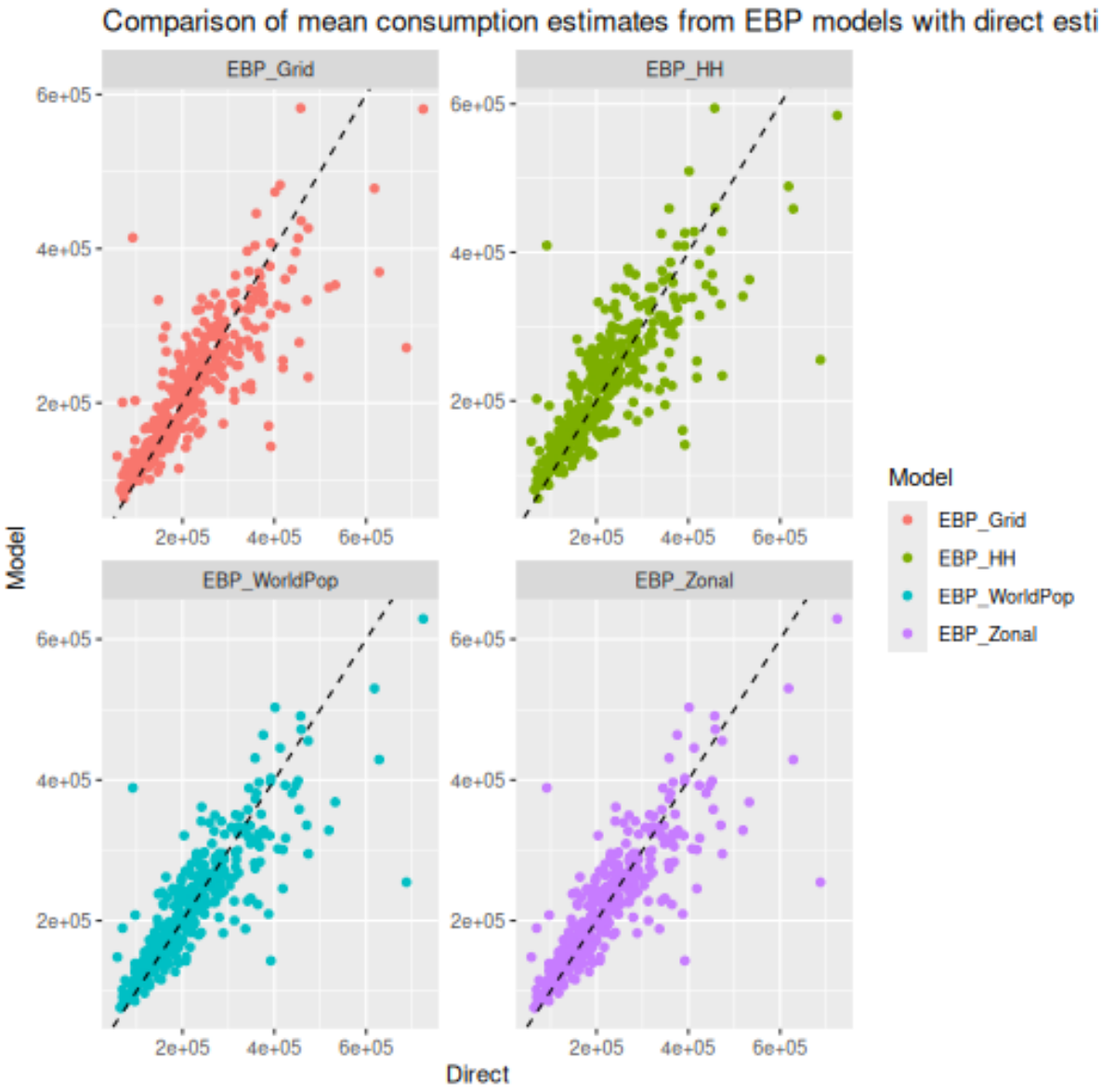

Figure 3: Comparison of mean consumption estimates with direct estimates between the different models

```
results %>%
  dplyr::select(-Mean) %>%
  pivot_wider(id_cols = c(Domain), names_from = Model, values_from = Head_Count) %>%
  pivot_longer(starts_with("EBP"), names_to = "Model", values_to = "Head_Count") %>%
  ggplot(aes(x = Direct, y = Head_Count, color = Model)) +
```

```
  geom_point() +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  facet_wrap(~Model, scales = "free") +
  labs(
    title = "Comparison of Head Count Ratio estimates from EBP models with direct estimates"
    x = "Direct",
    y = "Model"
  )
```

```
## Warning: Removed 1520 rows containing missing values or values outside the scale range (`g
```

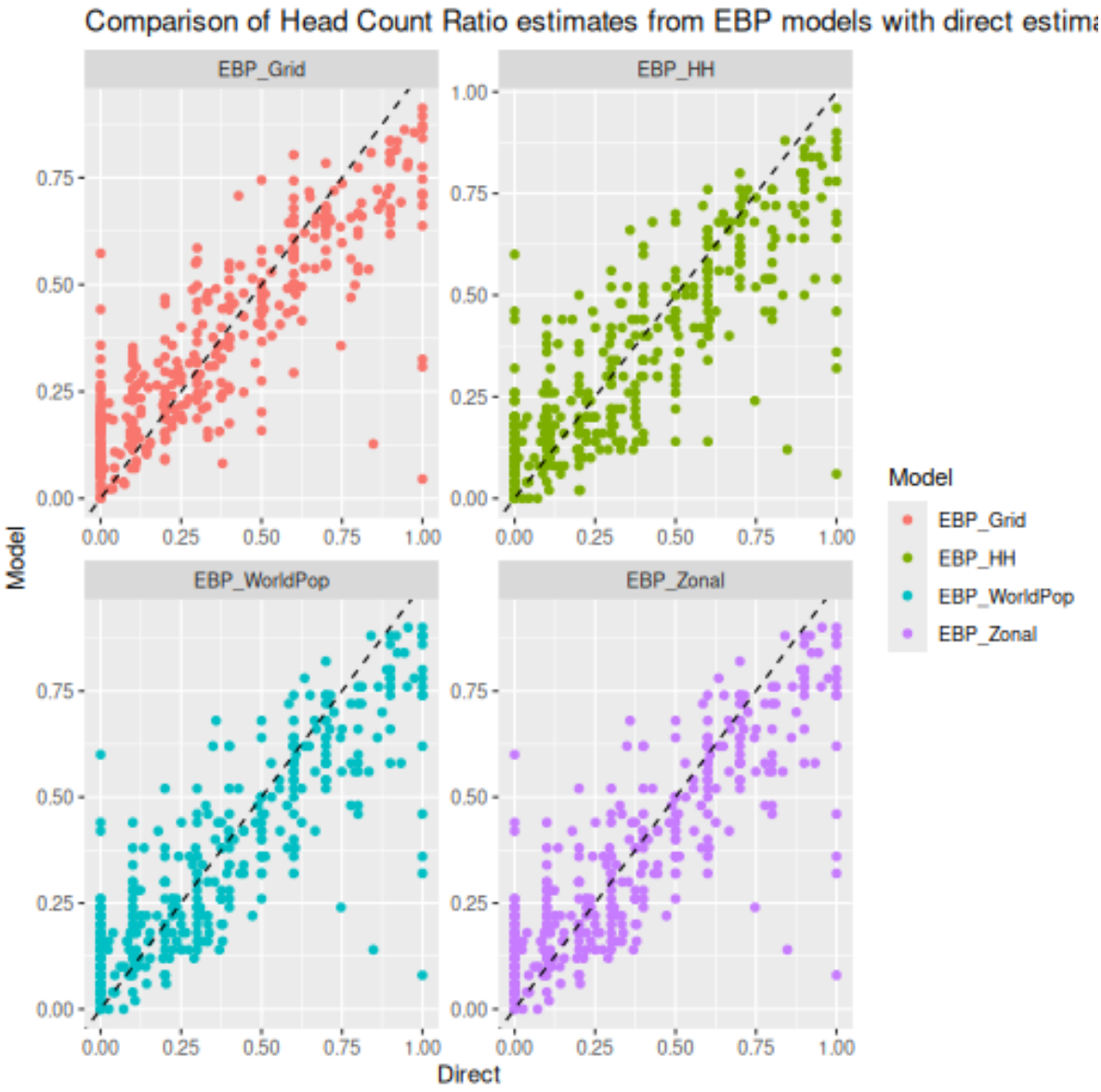

Figure 4: Comparison of Head Count Ratio estimates with direct estimates between the different models

We can also check the spatial distribution of the estimates and compare the results. Note that the direct estimate have out-of-sample areas that we cannot estimate since there are no samples in these areas. On the other hand, the EBP allows us to produce out-of-sample estimates with the caveat of having larger uncertainty around these estimates. In both **?@fig-comparison-mean-maps** and **?@fig-comparison-hcr-maps**, we can observe similar patterns with some

slight differences in some regions.

```
shp_dt %>%
  dplyr::select(ADM2_PCODE, geometry) %>%
  right_join(results, by = c("ADM2_PCODE" = "Domain")) -> results_sf

ggplot() +
  geom_sf(data = results_sf, aes(fill = Mean)) +
  facet_wrap(~Model) +
  scale_fill_viridis_c(option = "plasma") +
  labs(fill = "Mean consumption")
```

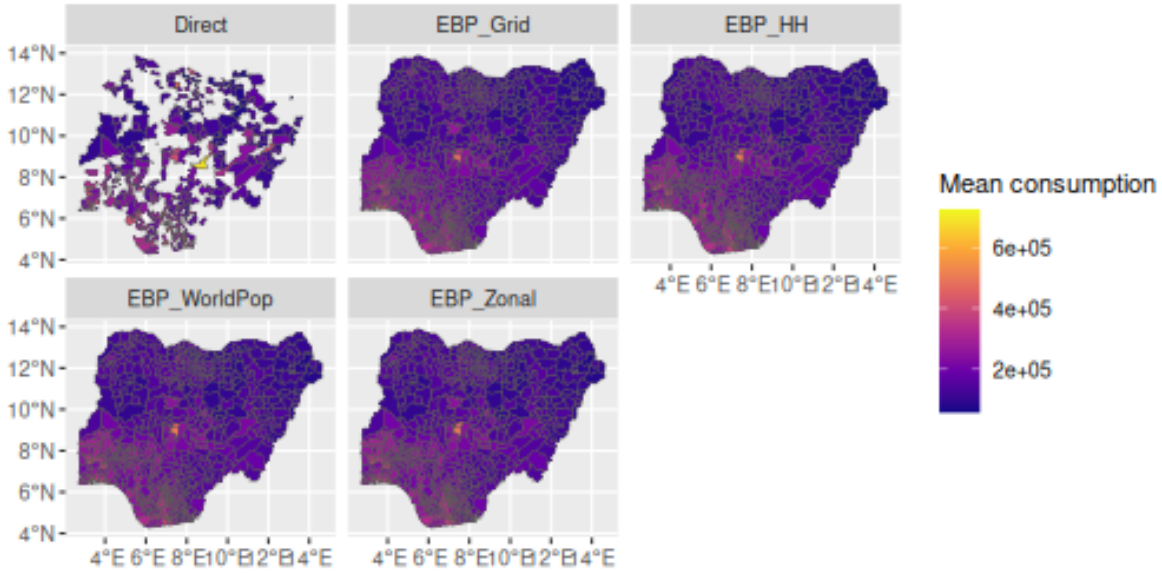

Figure 5: Comparison of mean consumption estimates using maps

```
ggplot() +
  geom_sf(data = results_sf, aes(fill = Head_Count)) +
  facet_wrap(~Model) +
  scale_fill_viridis_c(option = "plasma") +
  labs(fill = "Head Count Ratio")
```

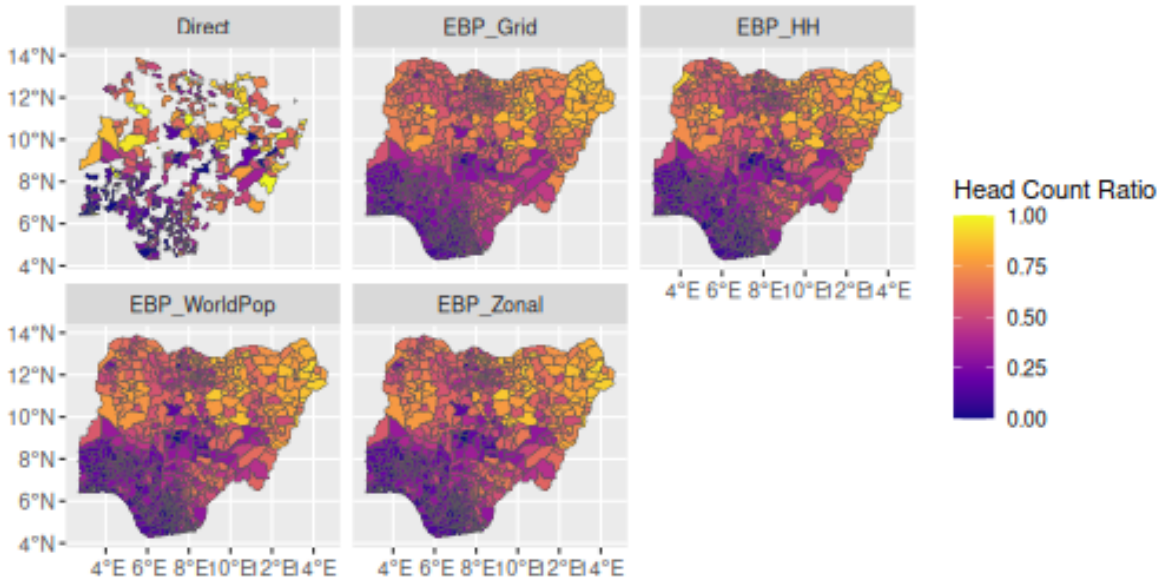

Figure 6: Comparison of Head Count Ratio estimates using maps

References

Edochie, Ifeanyi, David Newhouse, Nikos Tzavidis, Timo Schmid, Elizabeth Foster, Angela Luna Hernandez, Aissatou Ouedraogo, Aly Sanoh, and Aboudrahyme Savadogo. 2024. "Small Area Estimation of Poverty in Four West African Countries by Integrating Survey and Geospatial Data." *Policy Research Working Paper*, no. 10892

(September). https://documents1.worldbank.org/curated/en/099158209042442519/pdf/IDU170f4516f1b8a7146a619630191a7420b4631.pdf.

Masaki, Takaaki, David Newhouse, Ani Rudra Silwal, Adane Bedada, and Ryan Engstrom. 2022. "Small Area Estimation of Non-Monetary Poverty with Geospatial Data." *Statistical Journal of the IAOS* 38 (3): 1035–51. https://doi.org/10.3233/SJI-210902.

Molina, Isabel, and J. N. K. Rao. 2010. "Small Area Estimation of Poverty Indicators." *Canadian Journal of Statistics* 38 (3): 369–85. https://doi.org/10.1002/cjs.10051.

Newhouse, David, Joshua D Merfeld, Anusha Pudugramam Ramakrishnan, Tom Swartz, and Partha Lahiri. 2025. "Small Area Estimation of Monetary Poverty in Mexico Using Satellite Imagery and Machine Learning." *Oxford Bulletin of Economics and Statistics*.

Tzavidis, Nikos, Li-Chun Zhang, Angela Luna, Timo Schmid, and Natalia Rojas-Perilla. 2018. "From Start to Finish: A Framework for the Production of Small Area Official Statistics." *Journal of the Royal Statistical Society Series A: Statistics in Society* 181 (4): 927–79. https://doi.org/10.1111/rssa.12364.

Session

```
sessionInfo()
```

```
## R version 4.4.3 (2025-02-28)
## Platform: x86_64-redhat-linux-gnu
## Running under: Fedora Linux 41 (Forty One)
##
## Matrix products: default
## BLAS/LAPACK: FlexiBLAS OPENBLAS-OPENMP;  LAPACK version 3.12.0
##
## locale:
##  [1] LC_CTYPE=en_GB.UTF-8       LC_NUMERIC=C               LC_TIME=en_GB.UTF-8        LC_C
##  [6] LC_MESSAGES=en_GB.UTF-8    LC_PAPER=en_GB.UTF-8       LC_NAME=C                  LC_A
## [11] LC_MEASUREMENT=en_GB.UTF-8 LC_IDENTIFICATION=C
##
## time zone: Europe/London
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] GeoLink_0.1.0    emdi_2.2.2       sf_1.0-19        lubridate_1.9.4  forcats_1.0.0
##  [9] readr_2.1.5      tidyr_1.3.1      tibble_3.2.1     ggplot2_3.5.1    tidyverse_2.0.0
```

```
##
## loaded via a namespace (and not attached):
##   [1] wk_0.9.4             rstudioapi_0.16.0    jsonlite_1.9.1       magrittr_2.0.3
##   [7] fs_1.6.4             vctrs_0.6.5          spdep_1.3-5          memoise_2.0.1
##  [13] janitor_2.2.0        htmltools_0.5.8.1    usethis_3.1.0        progress_1.2.3
##  [19] s2_1.1.7             raster_3.6-31        spData_2.3.1         KernSmooth_2.23-26
##  [25] plyr_1.8.9           cachem_1.1.0         mime_0.12            lifecycle_1.0.4
##  [31] Matrix_1.7-2         R6_2.6.1             fastmap_1.2.0        shiny_1.9.1
##  [37] digest_0.6.35        colorspace_2.1-0     mapview_2.11.2       diagonals_6.4.0
##  [43] leafem_0.2.3         pkgload_1.3.4        crosstalk_1.2.1      labeling_0.4.3
##  [49] mgcv_1.9-1           httr_1.4.7           compiler_4.4.3       proxy_0.4-27
##  [55] withr_3.0.2          doParallel_1.0.17    backports_1.5.0      DBI_1.2.3
##  [61] R.utils_2.13.0       MASS_7.3-64          rappdirs_0.3.3       sessioninfo_1.2.2
##  [67] tools_4.4.3          units_0.8-7          zip_2.3.1            parallelMap_1.5.1
##  [73] R.oo_1.27.0          glue_1.8.0           satellite_1.0.5      dbscan_1.2-0
##  [79] grid_4.4.3           checkmate_2.3.2      reshape2_1.4.4       generics_0.1.3
##  [85] gtable_0.3.5         tzdb_0.5.0           formula.tools_1.7.1  R.methodsS3_1.8.2
##  [91] hms_1.1.3            sp_2.2-0             xml2_1.3.8           foreach_1.5.2
##  [97] splines_4.4.3        moments_0.14.1       BBmisc_1.13          lattice_0.22-6
## [103] miniUI_0.1.1.1       rstac_1.0.1          knitr_1.47           gridExtra_2.3
## [109] devtools_2.4.5       brio_1.1.5           stringi_1.8.4        boot_1.3-31
## [115] codetools_0.2-20     cli_3.6.4            xtable_1.8-4         reticulate_1.41.0.1
## [121] png_0.1-8            parallel_4.4.3       ellipsis_0.3.2       prettyunits_1.2.0
## [127] profvis_0.4.0        urlchecker_1.0.1     viridisLite_0.4.2    scales_1.3.0
## [133] ncdf4_1.23           crayon_1.5.3         rlang_1.1.5          rvest_1.0.4
```