Package: AgePopDenom (via r-universe)

February 26, 2025

Type Package

Title Model Fine-Scale Age-Structured Population Data using Open-Source Data

Version 0.4.0

Description Automate the modelling of age-structured population data using survey data, grid population estimates and urban-rural extents.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

SystemRequirements C++17, TMB

Imports TMB, dplyr, curl, exactextractr, ggplot2, numDeriv, pdist, utils, sf, cli, tibble, terra, tidyr, httr

Suggests stats, stringr, openxlsx2, countrycode, crayon, scales, glue, haven, here, rstudioapi, geodata, pbmcapply, future, future.apply, purrr, rdhs, rlang, pak, sp, automap, testthat (>= 3.0.0), knitr, rmarkdown, matrixStats, mockery, gstat, RcppEigen

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

URL https://truenomad.github.io/AgePopDenom/

Depends R (>= 4.1.0)

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://truenomad.r-universe.dev

RemoteUrl https://github.com/truenomad/agepopdenom

RemoteRef HEAD

RemoteSha 73e4a016a812a7d9c9a7b691113dfaf3c28c11c0

Contents

`aggregate_and_extract_gamma` *Aggregate Individual Survey Data and Extract Gamma Parameters by Location*

Description

This script aggregates the individual Survey data to location level and extracts the gamma parameters for the locations.

Usage

```
aggregate_and_extract_gamma(
  data,
  lat_column = "lat",
  long_column = "long",
  age_column = "ageyrs",
  urban_column = "urban"
)
```

Arguments

data	Data frame containing individual-level age data with coordinates and urban/rural classification.
lat_column	Column name for latitude coordinates (default: "lat")
long_column	Column name for longitude coordinates (default: "long")
age_column	Column name for age values (default: "ageyrs")
urban_column	Column name for urban/rural classification (default: "urban")

Value

List containing:

- outlier_free_data: Original data with added spatial coordinates, outliers removed, and clusters with less than 10 samples removed
- age_param_data: Location-level gamma parameters with columns: lon, lat, web_x, web_y, log_scale, log_shape, urban, b1, c, b2, nsampled

---

compute_cov *Compute Covariance Matrix for Spatial Model*

Description

This function computes a block covariance matrix for a bivariate spatial model with age-structured parameters.

Usage

```
compute_cov(
  gamma,
  sigma2,
  phi,
  u_dist,
  n_x,
  tau2_1 = 1,
  tau2_2 = 1,
  age_param_data
)
```

Arguments

gamma	Correlation parameter between the two spatial processes
sigma2	Variance parameter for the spatial processes
phi	Range parameter for the spatial correlation
u_dist	Distance matrix between locations

n_x	Number of spatial locations
tau2_1	Variance parameter for first process (default = 1)
tau2_2	Variance parameter for second process (default = 1)
age_param_data	List containing age-structured parameters: • b1: Vector of age parameters for first process • b2: Vector of age parameters for second process • c: Vector of cross-process age parameters

Value

A sparse symmetric matrix of dimension 2n_x × 2n_x

---

create_prediction_data

Generate or Load Cached Predictors Data

Description

This function creates predictors data based on spatial inputs or loads cached predictors data if the file already exists. It saves the generated data to a specified directory for reuse and provides progress updates.

Usage

```
create_prediction_data(
  country_code,
  country_shape,
  pop_raster,
  ur_raster,
  adm2_shape,
  cell_size = 5000,
  ignore_cache = FALSE,
  output_dir = here::here("03_outputs", "3a_model_outputs")
)
```

Arguments

country_code	A string representing the country code (e.g., "KEN").
country_shape	An ‘sf‘ object representing the country’s administrative boundaries.
pop_raster	A ‘terra‘ raster object representing the population raster.
ur_raster	A ‘terra‘ raster object representing the urban extent raster.
adm2_shape	An ‘sf‘ object representing the administrative level 2 boundaries.
cell_size	An integer specifying the cell size for the prediction grid in meters (default is 5000).

ignore_cache	A boolean input which is set to determine whether to ignore the existing cache and write over it. Default is set to FALSE.
output_dir	A string specifying the directory where the predictors data file should be saved (default is "03_outputs/3a_model_outputs").

Value

A data object (‘predictor_data‘) containing the generated predictors.

Examples

```
tf <- file.path(tempdir(), "test_env")

# Initialize with normalized path
dir.create(tf, recursive = TRUE, showWarnings = FALSE)

init(
  r_script_name = "full_pipeline.R",
  cpp_script_name = "model.cpp",
  path = tf,
  open_r_script = FALSE
)

# Download shapefiles
download_shapefile(
  country_codes =  "COM",
  dest_file = file.path(
    tf, "01_data", "1c_shapefiles",
    "district_shape.gpkg"
  )
)

# Download population rasters from worldpop
download_pop_rasters(
  country_codes = "COM",
  dest_dir = file.path(tf, "01_data", "1b_rasters", "pop_raster")
)

# Extract urban extent raster
extract_afurextent(
  dest_dir = file.path(tf, "01_data", "1b_rasters", "urban_extent")
)


urban_raster <-  terra::rast(
  file.path(tf, "01_data", "1b_rasters",
            "urban_extent", "afurextent.asc"))

pop_raster <-  terra::rast(
  file.path(tf, "01_data", "1b_rasters", "pop_raster",
            "com_ppp_2020_constrained.tif")
```

```
)

adm2_sf <- sf::read_sf(
 file.path(tf, "01_data", "1c_shapefiles",
            "district_shape.gpkg"))

country_sf <- sf::st_union(adm2_sf)

predictors <- create_prediction_data(
  country_code =  "COM",
  country_shape = country_sf,
  pop_raster = pop_raster,
  ur_raster = urban_raster,
  adm2_shape = adm2_sf,
  cell_size = 5000,
  output_dir = file.path(
    tf, "03_outputs/3a_model_outputs"
  )
)
```

`create_project_structure` *Create a Standardized Project Folder Structure*

Description

This function creates a standardized folder structure for organizing data, scripts, and outputs within a project directory. It ensures consistency and reproducibility for data-related workflows.

Usage

```
create_project_structure(base_path = here::here())
```

Arguments

`base_path`	A character string specifying the root directory where the folder structure will be created. Defaults to ‘here::here()‘ to use the current project directory.

Details

The function generates the following folder structure:

```
# 01_data/
# +-- 1a_survey_data/
# |    +-- processed/
# |    \-- raw/
# +-- 1b_rasters/
```

```
# |    +-- urban_extent/
# |    \-- pop_raster/
# +-- 1c_shapefiles/
# 02_scripts/
# 03_outputs/
# +-- 3a_model_outputs/
# +-- 3b_visualizations/
# +-- 3c_table_outputs/
# \-- 3d_compiled_results/
```

Value

Creates directories under the specified ‘base_path‘. Returns invisible ‘NULL‘ and prints messages about folder creation status.

Examples

```
# Create temp directory with normalized path
tf <- file.path(tempdir(), "test_env")
dir.create(tf, recursive = TRUE, showWarnings = FALSE)

#  Initialize with normalized path
cpp_path <- file.path(tf, "02_scripts", "model")
dir.create(cpp_path, recursive = TRUE, showWarnings = FALSE)
cpp_path <- normalizePath(cpp_path, winslash = "/", mustWork = FALSE)

create_project_structure(base_path = tf)
```

---

download_dhs_datasets *Main Function to Download DHS Datasets*

Description

Downloads the latest PR (household) and GE (geographic) DHS datasets for specified countries.

Usage

```
download_dhs_datasets(
  country_codes,
  cache_path = here::here("01_data", "1a_survey_data", "raw"),
  output_dir_root = here::here("01_data", "1a_survey_data", "raw"),
  survey_id = NULL,
  email,
  project,
  verbose = TRUE,
  clear_cache = TRUE
)
```

Arguments

country_codes	A character vector of ISO3 country codes.
cache_path	A character string specifying the cache path for RDHS.
output_dir_root	A character string specifying the root directory for output.
survey_id	A character vector of survey IDs. If NULL, uses latest survey.
email	A character string. Email registered with DHS.
project	A character string. Project name as registered with DHS.
verbose	Logical for rdhs setup and messages to be printed.
clear_cache	Logical whether to clear cache before downloading.

Value

Invisibly returns a list of downloaded dataset filenames.

---

download_pop_rasters *Download population rasters for given country codes.*

Description

This function attempts to download population rasters from WorldPop for the specified country codes. If ‘dest_dir‘ is not provided, file paths will be generated based on the given country codes and saved into the current project directory (using ‘here::here()‘). It first checks if a local file already exists, and if so, it will skip downloading.

Usage

```
download_pop_rasters(
  country_codes,
  dest_dir = here::here("01_data", "1b_rasters", "pop_raster"),
  quiet = FALSE
)
```

Arguments

country_codes	A character vector of ISO3 country codes.
dest_dir	A character vector of file paths for saving rasters. If NULL, defaults to "<cc>_ppp_2020_constrained2.tif" in the project dir.
quiet	Logical; if TRUE, suppress status messages.

Details

The function tries a baseline URL (BSGM) for each country code. If the file is not found there, it then tries a secondary URL (maxar_v1). If neither location provides the file, it returns NA and prompts you to check the WorldPop website directly.

Value

Invisibly returns a vector of downloaded file paths (or NA if not found).

Examples

```
download_pop_rasters(country_codes = "COM", dest_dir = tempdir())
```

`download_shapefile` *Download WHO ADM2 Boundaries with Partial Update*

Description

This function ensures the specified shapefile (‘dest_file‘) contains boundaries for all requested ISO3 codes (‘country_codes‘). It dynamically Downloades only the missing codes and appends them to the file, ensuring no duplication.

Usage

```
download_shapefile(
  country_codes,
  dest_file = here::here("01_data", "1c_shapefiles", "district_shape.gpkg")
)
```

Arguments

`country_codes`	Character vector of ISO3 country codes (e.g. c("KEN","UGA")).
`dest_file`	File path where data is saved (default: "01_data/1c_shapefiles/district_shape.gpkg").

Value

An ‘sf‘ object of combined boundaries for all requested country codes.

Examples

```
tf <- file.path(tempdir(), "test_env")

# Download population rasters from worldpop
download_shapefile(
  country_codes = "COM",
  dest_file = here::here(tf, "district_shape.gpkg")
)
```

extract_afurextent *Extract Urban/Rural Extent Raster*

Description

Extracts the ‘afurextent.asc‘ raster file from the package’s ‘inst/extdata‘ directory to a specified destination.

Usage

```
extract_afurextent(
  dest_dir = here::here("01_data", "1b_rasters", "urban_extent"),
  overwrite = FALSE
)
```

Arguments

dest_dir	A character string specifying the directory to save the extracted raster file.
overwrite	Logical. Whether to overwrite an existing file in the destination directory. Default is FALSE.

Details

This function extracts the ‘afurextent.asc‘ file from the package’s ‘extdata‘ directory, where it is stored as a compressed ‘.zip‘ file. It requires the ‘raster‘ package to load the raster file.

Value

A character string representing the full path to the extracted raster file.

Examples

```
extract_afurextent(tempdir(), overwrite = TRUE)
```

extract_age_param *Extract Parameters and Optimization Details with Log-Likelihood*

Description

Reads files matching the pattern "age_param_spatial_urban" in a specified directory, extracts the country name, parameter values, and optimization details, combines the results into a data frame, and optionally saves the output to a file.

Usage

```
extract_age_param(
  dir_path = here::here("03_outputs", "3a_model_outputs"),
 output_file = here::here("03_outputs", "3d_compiled_results", "model_params.csv")
)
```

Arguments

dir_path	A character string specifying the directory containing the files.
output_file	A character string specifying the path to save the output data frame. If NULL, the output will not be saved.

Value

A data frame with the extracted parameters, log-likelihood, and optimization details.

Examples

```
# Create temporary directory for dummy parameter files
dummy_dir <- tempdir()

dummy_params <- list(
  par = c(0.5, 1.2, 0.8, log(2), log(3), log(4)),
  objective = -123.45,
  convergence = 0,
  iterations = 10,
  evaluations = c("function = 20, gradient = 15"),
  message = "Converged"
)

saveRDS(dummy_params, file = file.path(dummy_dir,
                                       "abc_age_param_spatial_urban.rds"))

params_df <- extract_age_param(dir_path = dummy_dir,
                               output_file = tempdir())
```

---

extract_betas *Extract Beta Parameters from Model Output*

Description

This function extracts beta coefficients from a model parameter object, separating them into beta1 and beta2 components.

Usage

```
extract_betas(
  params_result,
  params = c("gamma", "log_sigma2", "log_phi", "log_tau1")
)
```

Arguments

params_result	A model parameter object containing parameter estimates
params	A character vector specifying parameter names, defaults to c("gamma", "log_sigma2", "log_phi", "log_tau1")

Details

The function assumes the parameter vector contains beta coefficients followed by other model parameters. It splits the betas into two equal groups after removing the last 4 parameters.

Value

A list with two components:

- beta1: First set of beta coefficients
- beta2: Second set of beta coefficients

---

fit_spatial_model *Fit a Spatial Model for Age Parameters using TMB*

Description

Fits a spatial model for age parameters using Template Model Builder (TMB) and C++. The model incorporates spatial correlation through distance matrices and handles both scale and shape parameters simultaneously. Progress is tracked with clear status updates. Can optionally load from cache.

Usage

```
fit_spatial_model(
  data,
  country_code = NULL,
  scale_outcome,
  shape_outcome,
  covariates,
  cpp_script_name,
  verbose = TRUE,
  control_params = list(trace = 2),
  manual_params = NULL,
  output_dir = NULL,
  ignore_cache = FALSE
)
```

Arguments

data	A data frame containing the response variables, covariates, and spatial coordinates (web_x, web_y)
country_code	Optional country code to save/load cached model. Default NULL runs model without caching.
scale_outcome	Character string specifying the column name for the scale parameter response variable
shape_outcome	Character string specifying the column name for the shape parameter response variable
covariates	Character vector of covariate names to include in both scale and shape models
cpp_script_name	Character string specifying the name of the C++ file (without extension) containing the TMB model definition
verbose	Logical indicating whether to show progress updates. Default TRUE
control_params	List of control parameters passed to nlminb optimizer. Default: list(trace = 2)
manual_params	Optional list of manual parameter values. If NULL (default), initial parameters are estimated from linear regression. The list should contain: • beta1: Vector of coefficients for scale model • beta2: Vector of coefficients for shape model • gamma: Scalar value (default 1.0) • log_sigma2: Log of sigma squared (default log(1.0)) • log_phi: Log of phi (estimated from variogram) • log_tau2_1: Log of tau squared (default log(1.0))
output_dir	Directory to save cached models. Only used if country_code is provided.
ignore_cache	Whether to ignore existing cache. Default FALSE.

Details

The function performs the following steps with progress tracking: 1. Fits initial linear models for scale and shape parameters 2. Calculates spatial distance matrix from web coordinates 3. Estimates optimal phi parameter using variogram: - Computes empirical variogram using automap - Automatically selects best theoretical variogram model - Range parameter is used to initialize spatial correlation - Default range of 100 used if estimation fails 4. Compiles and loads the TMB C++ template 5. Optimizes the joint likelihood using nlminb

The spatial correlation is modeled using an exponential variogram with parameters estimated from the data. The distance matrix is computed from the web coordinates (web_x, web_y) and used in the spatial covariance structure.

The C++ template should implement the joint spatial model for both parameters.

Value

An object of class 'nlminb' containing:

- par - Optimized parameter values

- objective - Final value of objective function
- convergence - Convergence code
- message - Convergence message
- iterations - Number of iterations
- evaluations - Number of function/gradient evaluations
- scale_formula - Formula used for scale model
- shape_formula - Formula used for shape model
- variogram - Fitted variogram model from automap containing:
  - range - Spatial correlation range parameter
  - psill - Partial sill (structured variance)
  - nugget - Nugget effect (unstructured variance)
  - kappa - Smoothness parameter for Matern models

Note

Requires TMB package and a working C++ compiler. The C++ template must be properly structured for TMB. The automap package is required for variogram fitting.

Examples

```
set.seed(123)
# Set parameters for simulation
total_population <- 266
urban_proportion <- 0.602
total_coords <- 266
lon_range <- c(-16.802, -13.849)
lat_range <- c(13.149, 13.801)
mean_web_x <- -1764351
mean_web_y <- 1510868

# Simulate processed survey dataset for Gambia
df_gambia <- NULL
df_gambia$age_param_data <- dplyr::tibble(
  country = "Gambia",
  country_code_iso3 = "GMB",
  country_code_dhs = "GM",
  year_of_survey = 2024,
  id_coords = rep(1:total_coords, length.out = total_population),
  lon = runif(total_population, lon_range[1], lon_range[2]),
  lat = runif(total_population, lat_range[1], lat_range[2]),
  web_x = rnorm(total_population, mean_web_x, 50000),
  web_y = rnorm(total_population, mean_web_y, 50000),
  log_scale = rnorm(total_population, 2.82, 0.2),
  log_shape = rnorm(total_population, 0.331, 0.1),
  urban = rep(c(1, 0), c(
    round(total_population * urban_proportion),
    total_population - round(total_population * urban_proportion)
```

```
 )),
 b1 = rnorm(total_population, 0.0142, 0.002),
 c = rnorm(total_population, -0.00997, 0.001),
  b2 = rnorm(total_population, 0.00997, 0.002),
  nsampled = sample(180:220, total_population, replace = TRUE)
)


tf <- file.path(tempdir(), "test_env")
dir.create(tf, recursive = TRUE, showWarnings = FALSE)

#initialise files and key scripts
init(
  r_script_name = "full_pipeline.R",
  cpp_script_name = "model.cpp",
  path = tf,
  open_r_script = FALSE
)

mod <- fit_spatial_model(
  df_gambia$age_param_data,
  scale_outcome = "log_scale",
  shape_outcome = "log_shape",
  covariates = "urban",
  cpp_script_name = file.path(tf, "02_scripts/model"),
  country_code = "GMB",
  output_dir = file.path(tf, "03_outputs/3a_model_outputs")
)
```

`generate_age_pop_raster` *Generate Age Population Raster*

Description

Creates age-stratified population raster layers from predictor data and gamma distribution parameters. Supports parallel processing and caching of results. The output is a multi-layer raster stack with each layer representing the population proportion for a specific age interval.

Usage

```
generate_age_pop_raster(
  predictor_data,
  scale_pred,
  shape_pred,
  age_range = c(0, 10),
  age_interval = 1,
```

```
  country_code,
  ignore_cache = FALSE,
  output_dir,
  n_cores = parallel::detectCores() - 2
)
```

Arguments

predictor_data	Data frame containing population and spatial data with columns: country, region, district, pop, web_x, web_y
scale_pred	Matrix of scale parameters for gamma distribution predictions
shape_pred	Matrix of shape parameters for gamma distribution predictions
age_range	Numeric vector of length 2 specifying min and max ages, default c(0,99)
age_interval	Numeric interval size between age groups in years, default 1
country_code	Character ISO3 country code
ignore_cache	Logical whether to ignore cached results, default FALSE
output_dir	Character path to output directory
n_cores	Integer number of cores for parallel processing, default detectCores()-2

Details

The function processes age intervals sequentially, computing population proportions using parallel processing. Results are cached as a GeoTIFF file for future use. The output raster maintains spatial properties of the input data and is suitable for GIS analysis and visualization.

Value

SpatRaster object (terra package) containing multiple layers, where each layer represents the population proportion for an age interval. Layer names indicate the age range (e.g., "Age 0 to 1 years"). The raster uses EPSG:3857 projection with 5000m resolution.

Examples

```
predictor_data <- data.frame(
 country = rep("CountryX", 100),
 region = rep("RegionA", 100),
 district = rep("District1", 100),
 pop = sample(100:1000, 100, replace = TRUE),
 web_x = runif(100, -100, 100),
 web_y = runif(100, -50, 50)
)

scale_pred <- matrix(runif(100 * 10, 1, 5), nrow = 100, ncol = 10)
shape_pred <- matrix(runif(100 * 10, 1, 5), nrow = 100, ncol = 10)

res <- generate_age_pop_raster(predictor_data,
                       scale_pred,
                       shape_pred,
```

```
                        country_code = "COD",
                        output_dir = file.path(tempdir()),
                        n_cores = 1)
```

`generate_age_pop_table` *Generate Age Population Tables*

Description

Creates age-stratified population tables from predictor data and gamma distribution parameters. Supports parallel processing and caching of results.

Usage

```
generate_age_pop_table(
  predictor_data,
  scale_pred,
  shape_pred,
  age_range = c(0, 99),
  age_interval = 1,
  country_code,
  ignore_cache = FALSE,
  output_dir,
  n_cores = parallel::detectCores() - 2
)
```

Arguments

`predictor_data`	Data frame containing population data with columns: country, region, district, pop
`scale_pred`	Matrix of scale parameters for gamma distribution predictions
`shape_pred`	Matrix of shape parameters for gamma distribution predictions
`age_range`	Numeric vector of length 2 specifying min and max ages, default c(0,99)
`age_interval`	Numeric interval size between age groups in years, default 1
`country_code`	Character ISO3 country code
`ignore_cache`	Logical whether to ignore cached results, default FALSE
`output_dir`	Character path to output directory
`n_cores`	Integer number of cores for parallel processing, default detectCores()-2

Value

List containing two data frames: - prop_df: Age-stratified population proportions with uncertainty intervals - pop_df: Age-stratified population counts with uncertainty intervals

Examples

```
predictor_data <- data.frame(
 country = rep("ABC", 1100),
  region = rep("Region1", 1100),
  district = rep("District1", 1100),
  pop = rep(1000, 1100)
)
scale_pred <- matrix(rep(1:10, 1100), nrow = 1100, ncol = 10)
shape_pred <- matrix(rep(1:10, 1100), nrow = 1100, ncol = 10)
output <- generate_age_pop_table(
  predictor_data, scale_pred, shape_pred, age_range = c(0, 99),
  age_interval = 10, country_code = "ABC", ignore_cache = TRUE,
  output_dir = tempdir(), n_cores = 1
)
```

---

generate_age_pyramid_plot

Generate and Save Age Pyramid Plot

Description

This function processes an input dataset to compute age distribution, generates age pyramid plots by region showing both proportions and counts, and saves the plots to a specified directory.

Usage

```
generate_age_pyramid_plot(
  dataset,
  country_code,
  output_dir,
  line_color = "#67000d",
  fill_high = "#fee0d2",
  fill_low = "#a50f15",
  break_axis_by = 10,
  caption = paste0("Note: Total population includes ",
    "ages 99+, pyramid shows ages 0-99")
)
```

Arguments

dataset	A list containing two data frames: - prop_df: Population proportions data frame - pop_df: Population counts data frame Each with columns for ‘country‘, ‘region‘, ‘district‘, and columns ending with "mean"
country_code	A string representing the country code (e.g., "ken").

output_dir	A string specifying the directory where plots should be saved.
line_color	A string specifying the color of the plot's lines. Default is ‘"#67000d"‘.
fill_high	A string specifying the fill color for high values. Default is ‘"#fee0d2"‘.
fill_low	A string specifying the fill color for low values. Default is ‘"#a50f15"‘
break_axis_by	break axis to show less cluttered age groups. Default is 10
caption	A string specifying the caption text. Default is "Note: Total population includes ages 99+, pyramid shows ages 0-99"

Value

A list containing both proportion and count plots.

Examples

```
set.seed(123)
prop_df <- data.frame(
 country = rep("COD", 10),
 region = rep("RegionA", 10),
 district = paste("District", 1:10),
 popsize = runif(10, 2340, 28761),
 `0-4_mean` = runif(10, 0.1, 0.5),
 `5-9_mean` = runif(10, 0.05, 0.4),
 `10-14_mean` = runif(10, 0.03, 0.3)
)

pop_df <- data.frame(
 country = rep("COD", 10),
 region = rep("RegionA", 10),
 district = paste("District", 1:10),
 popsize = runif(10, 2340, 28761),
 `0-4_mean` = runif(10, 1000, 5000),
 `5-9_mean` = runif(10, 800, 4500),
 `10-14_mean` = runif(10, 700, 4000)
)

dataset <- list(prop_df = prop_df, pop_df = pop_df)

res <- generate_age_pyramid_plot(
         dataset = dataset,
         country_code = "COD",
         output_dir = file.path(tempdir()))
```

`generate_gamma_predictions` *Predict Gamma Distribution Parameters for Spatial Grid*

Description

This function predicts the scale and shape parameters of a Gamma distribution across a spatial grid using a bivariate spatial model. It can either generate new predictions or load cached results if available.

Usage

```
generate_gamma_predictions(
  country_code,
  age_param_data,
  model_params,
  predictor_data,
  shapefile,
  cell_size = 5000,
  n_sim = 5000,
  ignore_cache = FALSE,
  save_file = FALSE,
  output_dir = here::here("03_outputs", "3a_model_outputs")
)
```

Arguments

`country_code`	A string representing the country code (e.g., "KEN").
`age_param_data`	A data frame containing: • web_x, web_y: Spatial coordinates • urban: Urban/rural indicator • log_scale: Log of scale parameter at observed locations • log_shape: Log of shape parameter at observed locations
`model_params`	A list containing model parameters: • par: Named vector with gamma, log_sigma2, log_phi, log_tau1 • Additional parameters for extracting beta coefficients
`predictor_data`	A data object containing the predictors data.
`shapefile`	An sf object defining the boundary for predictions
`cell_size`	Numeric. Grid cell size in meters (default: 5000)
`n_sim`	Integer. Number of simulations for prediction (default: 5000)
`ignore_cache`	A boolean input which is set to determine whether to ignore the existing cache and write over it. Default is set to FALSE.
`save_file`	A boolean to determine whether to save prediction or not. Default is FALSE as this will require lots of space.

output_dir A string specifying the directory where the predictions file should be saved (default is "03_outputs/3a_model_outputs").

Value

A list containing:

- scale_pred: Matrix of simulated scale parameters
- shape_pred: Matrix of simulated shape parameters

`generate_gamma_raster_plot` *Generate and Save Raster Plot for Gamma Predictions*

Description

This function creates rasters from prediction data, combines them into a stack, and saves the faceted plot to a specified file.

Usage

```
generate_gamma_raster_plot(
  predictor_data,
  pred_list,
  country_code,
  output_dir,
  save_raster = TRUE,
  file_name_suffix = "gamma_prediction_rasters",
  width = 2500,
  height = 2000,
  png_resolution = 300
)
```

Arguments

predictor_data	A data frame containing ‘web_x‘ and ‘web_y‘ coordinates and associated prediction values.
pred_list	A list containing predictions (‘shape_hat‘, ‘scale_hat‘, ‘mean_age_pred‘) for creating rasters.
country_code	A string representing the lowercase country code, used for naming the output file.
output_dir	A string specifying the directory where the plot should be saved.
save_raster	A logical input specifying whether to save output or not. Default is TRUE.
file_name_suffix	A string specifying the suffix for the file name (default is "gamma_prediction_rasters").
width	Numeric. Width of output plot in pixels (default: 2500).
height	Numeric. Height of output plot in pixels (default: 2000).
png_resolution	An integer specifying the resolution of the plot in DPI (default: 300).

Value

The path to the saved raster plot.

Examples

```
predictor_data <- data.frame(
  web_x = runif(100, -100, 100),
  web_y = runif(100, -50, 50)
)

pred_list <- list(
  shape_hat = rnorm(100, mean = 2, sd = 0.5),
  scale_hat = rnorm(100, mean = 10, sd = 2),
  mean_age_pred = rnorm(100, mean = 30, sd = 5)
)

generate_gamma_raster_plot(predictor_data,
                           pred_list,
                           country_code = "COD",
                           output_dir = file.path(tempdir()))
```

---

generate_variogram_plot

Generate Variogram Plot

Description

Creates a variogram plot showing the spatial dependence structure of the data. The plot includes both the empirical variogram points and the fitted theoretical variogram line. The empirical variogram points show the actual semivariance values at different distances, while the red line shows the fitted exponential variogram model.

Usage

```
generate_variogram_plot(
  age_param_data,
  fit_vario,
  country_code,
  scale_outcome = "log_scale",
  output_dir,
  width = 12,
  height = 9,
  png_resolution = 300
)
```

Arguments

age_param_data	Data frame containing the age parameter data. Must include columns 'web_x' and 'web_y' for spatial coordinates and the response variable specified in scale_outcome.
fit_vario	Fitted variogram object from automap package. Should contain components $psill (partial sill) and $range (range parameter) for the exponential variogram model.
country_code	Character string of the country code (e.g. "TZA") used for plot title and output filename.
scale_outcome	Character string specifying the column name for the scale parameter response variable (default: "log_scale"). This is the variable for which the variogram is computed.
output_dir	Character string specifying the directory path where the plot will be saved as a PNG file.
width	Plot width in pixels (default: 2000). Controls the output image width.
height	Plot height in pixels (default: 1500). Controls the output image height.
png_resolution	PNG resolution in DPI (dots per inch, default: 300). Higher values create larger, higher quality images.

Details

The function creates a variogram plot with the following elements: - Points showing empirical semivariance values at different distances - A red line showing the fitted exponential variogram model - Clear axis labels and title - Comma-formatted distance values on x-axis - Clean theme with black and white style

The output filename is constructed as lowercase country code + "_variogram.png"

Value

Invisibly returns the ggplot object containing the variogram plot. The plot is also saved as a PNG file in the specified output directory.

Examples

```
set.seed(123)  # For reproducibility
age_param_data <- data.frame(
  country = rep("TZA", 100),
  web_x = runif(100, 0, 100),
  web_y = runif(100, 0, 100),
  log_scale = rnorm(100, mean = 5, sd = 2)
)

# Create a dummy fitted variogram object
fit_vario <- list(
  psill = c(0.1, 0.5),
  range = c(0, 50)
)
```

```
vario_plot <- generate_variogram_plot(
   age_param_data = age_param_data,
   fit_vario = fit_vario,
   country_code = "TZA",
   output_dir = file.path(tempdir()))
```

`init` *Initialize Full Pipeline Script and Model Script*

Description

Creates a full pipeline R script and a model C++ script, saving them to the appropriate folders within the project directory structure. The folder structure is created using 'AgePopDenom::create_project_structure()'. The scripts contain example code for downloading and processing DHS data, shapefiles, and running models.

Usage

```
init(
  r_script_name = "full_pipeline.R",
  cpp_script_name = "model.cpp",
  path = here::here(),
  open_r_script = TRUE,
  setup_rscript = TRUE
)
```

Arguments

`r_script_name`	Character. The name of the R script file to be created. Defaults to '"full_pipeline.R"'.
`cpp_script_name`	Character. The name of the C++ script file to be created. Defaults to '"model.cpp"'.
`path`	Character. The directory in which to create the scripts. Defaults to 'here::here()'.
`open_r_script`	Logical. Whether to open the R script automatically in RStudio (if available). Defaults to 'TRUE'.
`setup_rscript`	Logical. Whether to setup the R script with example code. Defaults to 'TRUE'.

Value

Invisibly returns a list containing:

- r_script_path: Path to the created R script
- cpp_script_path: Path to the created C++ script

The function also prints success messages upon script creation.

Examples

```
# Create temp directory with normalized path
tf <- file.path(tempdir(), "test_env")
dir.create(tf, recursive = TRUE, showWarnings = FALSE)

#  Initialize with normalized path
cpp_path <- file.path(tf, "02_scripts", "model")
dir.create(cpp_path, recursive = TRUE, showWarnings = FALSE)
cpp_path <- normalizePath(cpp_path, winslash = "/", mustWork = FALSE)

init(
r_script_name = "full_pipeline.R",
cpp_script_name = "model.cpp",
path = tf,
open_r_script = FALSE
)
```

---

log_lik *Log-Likelihood Function for Spatial Model*

Description

Computes the log-likelihood for a spatial statistical model with a covariance structure determined by parameters including spatial decay and variance.

Usage

```
log_lik(
  par,
  p1,
  p2,
  d1,
  d2,
  y,
  u_dist,
  n_x,
  tau2_1 = 1,
  tau2_2 = 1,
  age_param_data
)
```

Arguments

par	A numeric vector of parameters to estimate. The vector contains: • par[1:p1]: Coefficients for fixed effects in dataset 1 (β_1). • par[(p1 + 1):(p1 + p2)]: Coefficients for fixed effects in dataset 2 (β_2).

- par[p1 + p2 + 1]: Spatial decay parameter (γ).
- par[p1 + p2 + 2]: Log of the variance parameter (σ^2).
- par[p1 + p2 + 3]: Log of the range parameter (ϕ).

p1	An integer. The number of fixed-effect parameters in dataset 1.
p2	An integer. The number of fixed-effect parameters in dataset 2.
d1	A numeric matrix. Design matrix for dataset 1 used to model the mean structure.
d2	A numeric matrix. Design matrix for dataset 2 used to model the mean structure.
y	A numeric vector. Observed response variable, including both datasets.
u_dist	A numeric matrix. Distance matrix for spatial locations.
n_x	An integer. The number of unique spatial locations.
tau2_1	Variance parameter for first process (default = 1)
tau2_2	Variance parameter for second process (default = 1)
age_param_data	A numeric matrix or vector. Additional parameters specific to age-based modeling.

Details

The log-likelihood is computed as:

$$-0.5 \left[\log(\det(M)) + (y - \mu)^T M^{-1} (y - \mu)\right]$$

where:

- M is the covariance matrix, computed using compute_cov.
- μ is the mean structure, determined by the design matrices d1, d2 and coefficients β_1, β_2.

The covariance matrix M is computed using spatial parameters (γ, σ^2, ϕ) and the distance matrix u_dist.

Value

A numeric scalar. The computed log-likelihood value.

Note

This function requires a helper function, compute_cov, to compute the covariance matrix based on spatial parameters.

process_dhs_data *Process DHS Data: Merge RDS Files with Shapefiles and Extract Gamma Parameters*

Description

This function processes DHS (Demographic and Health Survey) data by: 1. Reading RDS files and shapefiles for each country. 2. Merging demographic data with geographic information. 3. Cleaning and aggregating the data. 4. Extracting gamma parameters for age-related analysis.

Usage

```
process_dhs_data(
  rds_dir = here::here("01_data", "1a_survey_data", "raw", "pr_records"),
  shp_dir = here::here("01_data", "1a_survey_data", "raw", "shapefiles"),
  output_path = here::here("01_data", "1a_survey_data", "processed",
    "dhs_pr_records_combined.rds")
)
```

Arguments

rds_dir	Character. Path to the directory containing raw RDS files.
shp_dir	Character. Path to the directory containing shapefiles.
output_path	Character. Path to save the final processed dataset as an RDS file.

Details

The function loops through RDS files, processes each country's data by merging demographic information with shapefile data, and computes gamma parameters for age-related analysis. The progress is tracked and displayed for each country.

The function also filters out incomplete data (e.g., age values of '98') and handles labelled data using the 'haven::zap_labels' function.

The final output includes two datasets: 1. Outlier-free data. 2. Aggregated age parameter data.

Value

None. Saves the final combined dataset to the specified output path.

Examples

```
tf <- file.path(tempdir(), "test_env")
dir.create(tf, recursive = TRUE, showWarnings = FALSE)
tmp_rds_dir <- file.path(tf, "rds")
tmp_shp_dir <- file.path(tf, "shp")
tmp_output <- file.path(tf, "output.rds")

dir.create(tmp_rds_dir, recursive = TRUE, showWarnings = FALSE)
```

```
dir.create(tmp_shp_dir, recursive = TRUE, showWarnings = FALSE)

# Create fake DHS data
create_fake_dhs_data <- function(country_code) {
  set.seed(123) # For reproducibility
  n <- 100

  # Create labelled vectors
  hv007 <- haven::labelled(
    sample(c(2015, 2016), n, replace = TRUE),
    labels = c("2015" = 2015, "2016" = 2016)
  )

  hv001 <- haven::labelled(
    sample(1:20, n, replace = TRUE),
    labels = setNames(1:20, paste("Cluster", 1:20))
  )

  hv105 <- haven::labelled(
    sample(c(1:97, 98), n, replace = TRUE),
    labels = c(setNames(1:97, paste("Age", 1:97)), "Don't know" = 98)
  )

  # Combine into data frame
  data.frame(
    hv007 = hv007,
    hv001 = hv001,
    hv105 = hv105
  )
}

# Create fake shapefile data
# Create fake shapefile data with explicit CRS
create_fake_shapefile <- function(country_code) {
  set.seed(123)
  n_clusters <- 20

  # Create spatial data frame with explicit CRS
  sf_data <- sf::st_as_sf(
    dplyr::tibble(
      DHSCLUST = 1:n_clusters,
      URBAN_RURA = sample(c("R", "U"), n_clusters, replace = TRUE),
      LATNUM = runif(n_clusters, -10, 10),
    LONGNUM = runif(n_clusters, -10, 10)
  ),
    coords = c("LONGNUM", "LATNUM"),
    crs = 4326 # WGS84
  ) |>
    dplyr::mutate(
      LATNUM = runif(n_clusters, -10, 10),
      LONGNUM = runif(n_clusters, -10, 10)
    )
}
```

```
# Save test data for two countries
countries <- c("KE", "TZ")
for (country in countries) {
  saveRDS(
    create_fake_dhs_data(country),
    file = file.path(tmp_rds_dir, paste0(country, "HR71FL.rds"))
  )
  saveRDS(
    create_fake_shapefile(country),
    file = file.path(tmp_shp_dir, paste0(country, "HR7SHP.rds"))
  )
}

# Run the function
process_dhs_data(
  rds_dir = tmp_rds_dir,
  shp_dir = tmp_shp_dir,
  output_path = tmp_output
)
```

`process_final_population_data` *Process Final Population Data*

Description

Reads population and proportion data from RDS files, processes the data to generate age-group-based population summaries and proportions at different administrative levels (Country, Region, District), and writes the results to an Excel file with separate sheets for each level and metric.

Usage

```
process_final_population_data(
  input_dir = here::here("03_outputs", "3c_table_outputs"),
  excel_output_file = here::here("03_outputs", "3d_compiled_results",
    "age_pop_denom_compiled.xlsx")
)
```

Arguments

`input_dir`	A character string specifying the directory containing RDS files. Default is "03_outputs/3c_table_outputs" in the project directory.
`excel_output_file`	A character string specifying the output Excel file path. Default is "03_outputs/3d_compiled_results/age_pop_denom_compiled.xlsx" in the project directory.

Value

None. The function writes an Excel file to the specified output location with six sheets containing population counts and proportions at different administrative levels.

`process_gamma_predictions` *Process Gamma Prediction Results*

Description

This function processes gamma prediction results to calculate the mean age predictions, scale, and shape parameters efficiently.

Usage

```
process_gamma_predictions(gamma_prediction)
```

Arguments

`gamma_prediction`
: A list containing ‘scale_pred‘ and ‘shape_pred‘ matrices from the gamma prediction model.

Value

A list containing the following elements: - ‘mean_age_pred‘: A vector of mean age predictions. - ‘scale_hat‘: A vector of mean scale parameters. - ‘shape_hat‘: A vector of mean shape parameters.

`rasterize_data` *Rasterize Spatial Data*

Description

This function converts spatial data with x, y coordinates and a value field into a raster using a specified resolution and CRS.

Usage

```
rasterize_data(x_coords, y_coords, values, cell_size = 5000, crs, fun = mean)
```

Arguments

x_coords	Numeric vector of x-coordinates (e.g., longitude).
y_coords	Numeric vector of y-coordinates (e.g., latitude).
values	Numeric vector of values associated with each point.
cell_size	Numeric. Grid cell size in meters (default: 5000).
crs	Character, the coordinate reference system in EPSG format (e.g., "EPSG:3857").
fun	Function to aggregate values in cells (default is 'mean').

Value

A 'terra::SpatRaster' object.

Examples

```
x_coords <- runif(100, -100, 100)
y_coords <- runif(100, -50, 50)
values <- rnorm(100, mean = 10, sd = 5)

rasterize_data(x_coords, y_coords, values,
               cell_size = 5000, crs = "EPSG:3857", fun = mean)
```

---

run_full_workflow *Run Country-Specific Spatial Modeling Workflow with Logging*

Description

This function runs the entire spatial modeling workflow for a given country code and logs the results. It processes Survey data, fits a spatial model, generates predictions, creates population tables, and produces raster outputs. The function is modular and can be reused for different countries with minimal adjustments.

Usage

```
run_full_workflow(
  country_code,
  survey_data_path = here::here("01_data", "1a_survey_data", "processed"),
  survey_data_suffix = "dhs_pr_records_combined.rds",
  shape_path = here::here("01_data", "1c_shapefiles"),
  shape_suffix = "district_shape.gpkg",
  pop_raster_path = here::here("01_data", "1b_rasters", "pop_raster"),
  pop_raster_suffix = "_ppp_2020_constrained.tif",
  ur_raster_path = here::here("01_data", "1b_rasters", "urban_extent"),
  ur_raster_suffix = "afurextent.asc",
```

```
  pred_save_file = FALSE,
  raster_width = 2500,
  raster_height = 2000,
  raster_resolution = 300,
  save_raster = TRUE,
  generate_pop_raster = FALSE,
  pyramid_line_color = "#67000d",
  pyramid_fill_high = "#fee0d2",
  pyramid_fill_low = "#a50f15",
  pyramid_caption = paste0("Note: Total population includes ",
    "ages 99+, pyramid shows ages 0-99"),
  output_paths = list(),
  model_params = list(),
  return_results = FALSE,
  n_cores = parallel::detectCores() - 2,
  ...
)
```

Arguments

country_code	Character. The ISO3 country code (e.g., "TZA").
survey_data_path	Character. Path to Survey data. Default: "01_data/1a_survey_data/processed".
survey_data_suffix	Character. Suffix for Survey data files. Default: "dhs_pr_records_combined.rds".
shape_path	Character. Path to shapefile data. Default: "01_data/1c_shapefiles".
shape_suffix	Character. Suffix for shapefile data. Default: "district_shape.gpkg".
pop_raster_path	Character. Path to population raster data. Default: "01_data/1b_rasters/pop_raster".
pop_raster_suffix	Character. Suffix for population raster files. Default: "_ppp_2020_constrained.tif".
ur_raster_path	Character. Path to urban-rural extent data. Default: "01_data/1b_rasters/urban_extent".
ur_raster_suffix	Character. Suffix for urban-rural raster. Default: "afurextent.asc".
pred_save_file	Logical. Whether to save prediction files. Default: FALSE
raster_width	Integer. Width of raster plots in pixels. Default: 2500
raster_height	Integer. Height of raster plots in pixels. Default: 2000
raster_resolution	Integer. Resolution of PNG outputs. Default: 300
save_raster	Logical. Whether to save raster outputs to disk. Default: TRUE
generate_pop_raster	Logical. Whether to generate population raster. Default: FALSE
pyramid_line_color	Character. Hex color code for the age pyramid's outline. Default: "#67000d"

`pyramid_fill_high`
: Character. Hex color code for the age pyramid's higher values fill. Default: "#fee0d2"

`pyramid_fill_low`
: Character. Hex color code for the age pyramid's lower values fill. Default: "#a50f15"

`pyramid_caption`
: Character. Caption text for the age pyramid plot. Default: "Note: Total population includes ages 99+, pyramid shows ages 0-99"

`output_paths`
: List of output paths:
  - model: Path for model outputs. Default: "03_outputs/3a_model_outputs"
  - plot: Path for plots. Default: "03_outputs/3b_visualizations"
  - raster: Path for rasters. Default: "03_outputs/3c_raster_outputs"
  - table: Path for tables. Default: "03_outputs/3c_table_outputs"
  - compiled: Path for compiled results. Default: "03_outputs/3d_compiled_results"
  - excel: Path for Excel outputs. Default: "03_outputs/3d_compiled_results/age_pop_denom_compiled
  - log: Path for logs. Default: "03_outputs/3a_model_outputs/modelling_log.rds"

`model_params`
: List of model parameters:
  - cell_size: Cell size in meters. Default: 5000
  - n_sim: Number of simulations. Default: 5000
  - ignore_cache: Whether to ignore cache. Default: FALSE
  - age_range: Age range vector. Default: c(0, 99)
  - age_interval: Age interval. Default: 1
  - return_prop: Return proportions. Default: TRUE
  - scale_outcome: Scale outcome variable. Default: "log_scale"
  - shape_outcome: Shape outcome variable. Default: "log_shape"
  - covariates: Model covariates. Default: "urban"
  - cpp_script: C++ script path. Default: "02_scripts/model"
  - control_params: Control parameters. Default: list(trace = 2)
  - manual_params: Manual parameters. Default: NULL
  - verbose: Verbose output. Default: TRUE
  - age_range_raster: Age range for raster output. Default: c(0, 10)
  - age_interval_raster: Age interval for raster output. Default: 1

`return_results`
: Logical. Whether to return results. Default: FALSE.

`n_cores`
: Integer number of cores for parallel processing for age population table, default detectCores()-2

`...`
: Additional arguments passed to subfunctions.

Value

If return_results is TRUE, a list containing:

- spat_model_param: Fitted spatial model parameters
- predictor_data: Predictor dataset

- gamma_prediction: Generated gamma predictions
- pred_list: Processed gamma prediction results
- final_age_pop_table: Age-population table data
- final_pop: Compiled population data
- all_mod_params: Compiled model parameters

If return_results is FALSE, the function saves all outputs to disk and returns NULL invisibly.

See Also

- fit_spatial_model: Fits the spatial model for age parameters
- create_prediction_data: Creates predictor dataset from spatial inputs
- generate_gamma_predictions: Generates predictions using fitted model
- process_gamma_predictions: Processes gamma prediction results
- generate_gamma_raster_plot: Creates prediction raster plots
- generate_age_pop_table: Generates age-population tables
- generate_age_pyramid_plot: Creates age pyramid plots
- extract_age_param: Extracts and compiles model parameters
- process_final_population_data: Processes final population data
- generate_variogram_plot: Creates variogram plots showing spatial dependence structure

Examples

```
# set country code
country_codeiso <- "GMB"

set.seed(123)
# Set parameters for simulation
total_population <- 266
urban_proportion <- 0.602
total_coords <- 266
lon_range <- c(-16.802, -13.849)
lat_range <- c(13.149, 13.801)
mean_web_x <- -1764351
mean_web_y <- 1510868

# Simulate processed survey dataset for Gambia
df_gambia <- NULL
df_gambia$age_param_data <- dplyr::tibble(
  country = "Gambia",
  country_code_iso3 = "GMB",
  country_code_dhs = "GM",
  year_of_survey = 2024,
  id_coords = rep(1:total_coords, length.out = total_population),
  lon = runif(total_population, lon_range[1], lon_range[2]),
  lat = runif(total_population, lat_range[1], lat_range[2]),
```

```
  web_x = rnorm(total_population, mean_web_x, 50000),
  web_y = rnorm(total_population, mean_web_y, 50000),
  log_scale = rnorm(total_population, 2.82, 0.2),
  log_shape = rnorm(total_population, 0.331, 0.1),
  urban = rep(c(1, 0), c(
    round(total_population * urban_proportion),
    total_population - round(total_population * urban_proportion)
  )),
  b1 = rnorm(total_population, 0.0142, 0.002),
  c = rnorm(total_population, -0.00997, 0.001),
  b2 = rnorm(total_population, 0.00997, 0.002),
 nsampled = sample(180:220, total_population, replace = TRUE)
)


# Create temp directory with normalized path
tf <- file.path(tempdir(), "test_env")
dir.create(tf, recursive = TRUE, showWarnings = FALSE)
tf <- normalizePath(tf, winslash = "/", mustWork = FALSE)

AgePopDenom::init(
  r_script_name = "full_pipeline.R",
  cpp_script_name = "model.cpp",
  path = tf,
  open_r_script = FALSE
)

# save as processed dhs data
saveRDS(
  df_gambia,
  file = file.path(
    tf, "01_data", "1a_survey_data", "processed",
    "dhs_pr_records_combined.rds"
  ) |>
    normalizePath(winslash = "/", mustWork = FALSE)
)

# Download shapefiles
download_shapefile(
  country_codes = country_codeiso,
  dest_file = file.path(
    tf, "01_data", "1c_shapefiles",
    "district_shape.gpkg"
  ) |>
    normalizePath(winslash = "/", mustWork = FALSE)
)

# Download population rasters from worldpop
download_pop_rasters(
 country_codes = country_codeiso,
  dest_dir = file.path(tf, "01_data", "1b_rasters", "pop_raster") |>
    normalizePath(winslash = "/", mustWork = FALSE)
)
```

```
# Extract urban extent raster
extract_afurextent(
  dest_dir = file.path(tf, "01_data", "1b_rasters", "urban_extent") |>
    normalizePath(winslash = "/", mustWork = FALSE)
)

# Modelling ------------------------------------------------------------

run_full_workflow(
  country_code = country_codeiso,
  survey_data_path = file.path(
    tf, "01_data", "1a_survey_data", "processed"
  ) |>
    normalizePath(winslash = "/", mustWork = FALSE),
  survey_data_suffix = "dhs_pr_records_combined.rds",
  shape_path = file.path(
    tf, "01_data", "1c_shapefiles"
  ) |>
    normalizePath(winslash = "/", mustWork = FALSE),
  shape_suffix = "district_shape.gpkg",
  pop_raster_path = file.path(
    tf, "01_data", "1b_rasters", "pop_raster"
  ) |>
    normalizePath(winslash = "/", mustWork = FALSE),
  pop_raster_suffix = "_ppp_2020_constrained.tif",
  ur_raster_path = file.path(
    tf, "01_data", "1b_rasters", "urban_extent"
  ) |>
    normalizePath(winslash = "/", mustWork = FALSE),
  ur_raster_suffix = "afurextent.asc",
  pred_save_file = FALSE,
  raster_width = 2500,
  raster_height = 2000,
  raster_resolution = 300,
  save_raster = TRUE,
  pyramid_line_color = "#67000d",
  pyramid_fill_high = "#fee0d2",
  pyramid_fill_low = "#a50f15",
  pyramid_caption = paste0(
    "Note: Total population includes ",
    "ages 99+, pyramid shows ages 0-99"
  ),
  generate_pop_raster = TRUE,
  output_paths = list(
    model = file.path(tf, "03_outputs", "3a_model_outputs"),
    plot = file.path(tf, "03_outputs", "3b_visualizations"),
    raster = file.path(tf, "03_outputs", "3c_raster_outputs"),
    table = file.path(tf, "03_outputs", "3c_table_outputs"),
    compiled = file.path(tf, "03_outputs", "3d_compiled_results"),
    excel = file.path(
      tf, "03_outputs", "3d_compiled_results",
      "age_pop_denom_compiled.xlsx"
    ),
```

```
    log = file.path(
      tf, "03_outputs", "3a_model_outputs", "modelling_log.rds"
    )
  ) |> lapply(\(x) normalizePath(x, winslash = "/", mustWork = FALSE)),
  model_params = list(
    cell_size = 5000,
    n_sim = 10,
    ignore_cache = FALSE,
    age_range = c(0, 1),
    age_interval = 1,
    return_prop = TRUE,
    scale_outcome = "log_scale",
    shape_outcome = "log_shape",
    covariates = "urban",
    cpp_script = file.path(tf, "02_scripts", "model") |>
      normalizePath(winslash = "/", mustWork = FALSE),
    control_params = list(trace = 2),
    manual_params = NULL,
    verbose = TRUE
  ),
  return_results = FALSE,
  n_cores = 1
)
```

Index