popRF: Random Forest-informed Disaggregative Population Modelling and Mapping

Jeremiah J. Nieves, Geographic Data Science Lab - University of Liverpool
Maksym Bondarenko, WorldPop - University of Southampton
Forrest R. Stevens, University of Louisville
Andrea E. Gaughan, University of Louisville
Warren C. Jochem, WorldPop - University of Southampton
David Kerr, WorldPop - University of Southampton
Andrew J. Tatem, WorldPop - University of Southampton
Alessandro Sorichetta, WorldPop- University of Southampton

August 2, 2021

Contents

1 Introduction

Methods of disaggregating census-based areal population counts to finer gridded population surfaces have become more prevalent since the 1990s with advances in computation and digitisation of data (Martin and Bracken 1991; Balk and Yetman 2004; Bhaduri, Bright, and Coleman 2007; CIESIN 2011; Doxsey-Whitfield et al. 2015; European Commission and Columbia University 2015; Friere et al. 2016; Esch et al. 2018; Palacios-Lopez et al. 2019; Kugler et al. 2019; Leyk et al. 2019). However, disaggregation, the process of distributing values in a “top-down” manner from a larger source area to a series of smaller target areas contained within the source area, has been utilised for population mapping since at least 1936 (Wright 1936). Mennis (2009) presents a short history of dasymetric disaggregation in its introduction. Within dasymetric disaggregations of populations, grids with uniform spatial resolution have become common target areas (Leyk et al. 2019), albeit these are still aggregate representation of population from, say, household data. These disaggregated gridded population data have the potential to better represent the true underlying spatial distribution of population densities (Eicher and Brewer 2001; J. Mennis 2003; J. Mennis and Hultgren 2006; Leyk et al. 2019) as compared to the areal census-based counts. While these census-based counts are often available at high resolution, i.e. small areas such as enumeration districts, within government, publicly released versions are often much coarser. These coarser areal, census-based counts, lacking any other information, can only imply a homogeneous population density within a given area. Further, gridded population data obtained from disaggregation allow for the ready incorporation of other gridded data, e.g. access to health facilities, for further analyses as well as the ability to aggregate up to non-census-based areas for further interpretation.

Given these characteristics, disaggregated gridded population data have been increasingly been used for a broad number of applications in sustainability (Dunnett et al. 2020), public health and surveys (James et al. 2018; Ruktanonchai et al. 2020; Thomson et al. 2020), and urban expansion projections (Nieves, Sorichetta, et al. 2020; Nieves, Bondarenko, et al. 2020), to name a few. While the specifics of the disaggregative method can vary, in general, dasymetric disaggregation has become the most prevalent means of top-down gridded population modelling (Balk and Yetman

2004; Bhaduri, Bright, and Coleman 2007; CIESIN 2011; Doxsey-Whitfield et al. 2015; European Commission and Columbia University 2015; Friere et al. 2016; Esch et al. 2018; Leyk et al. 2019; Palacios-Lopez et al. 2019). Dasymetric disaggregation being the redistribution of counts using the relationships between population counts and spatially-coincident ancillary geographic data (J. Mennis 2003; J. Mennis and Hultgren 2006; Jeremy Mennis 2009). In particular, the WorldPop dasymetrically mapped gridded population products (https://worldpop.org) have become widely used by researchers, government and non-governmental organisations, and international organisations, particularly within low- and middle-income contexts. Their dasymetric mapping procedure utilises a random forest model and environmental covariates to generate the weights used to disaggregate population counts to the final, gridded surface (Gaughan et al. 2013, 2014; Sorichetta et al. 2015; F. R. Stevens et al. 2015).

There have been some packages and code-based tools for the express purpose of disaggregative mapping, both past and current. The USGS has an ArcGIS-based tool (Sleeter 2008; Gould and Sleeter 2014). Within `Python`, there is a subpackage to **`pySAL`** (Rey and Anselin 2010; Rey 2019; Rey et al. 2021) called **`tobler`** which provides a library of tools for areal interpolation and dasymetric mapping (Knaap et al. 2021). Within `R`, there is the **`disaggregation`** package which takes an INLA (Nandi et al. 2020) Bayesian approach to disaggregative mapping. There is also the web-based, yet written with a `R` back-end, **`peanutbutter`** (Leasure et al. 2021) that disaggregates population counts into building footprints based upon manually defined average household densities and estimated proportion of buildings that are residential. A similar field to disaggregative count mapping, small area estimation, have packages in `R` as well, such as **`emdi`** (Kreutzmann et al. 2019). However, none of these disaggregative mapping methods utilise a random forest approach which provides great flexibility, automability, and parallelisation across different data landscapes (Gaughan et al. 2014; F. R. Stevens et al. 2015; Reed et al. 2018; Leyk et al. 2019). Further, with the exception of **`disaggregation`** (Nandi et al. 2020), these disaggregative mapping approaches are relatively naive and can rely on manually defined weights and or weighting parameters.

While the code for this random forest-informed disaggregative mapping procedure has been made openly available via GitHub since 2014 (Forrest R. Stevens 2014), its accessibility, utility and efficiency have noted limitations for practitioners and end users. Because it was a multi-script procedure written in a combination of `R` and `Python` (Forrest R. Stevens 2014), a high barrier of programming skill was required for its utilisation. The multi-script format made workflows and general folder structures largely static and difficult to customise. Further, the `Python` portions, which flexibly handled the geoprocessing of covariates, was dependent upon an ArcGIS license to utilise the **`arcpy`** library. This limited the running of the scripts to Windows-based environments and, more importantly, the license cost served as another access barrier. While capable of application across large spatial extents, given its time of development and reliance on **`arcpy`** for processing, the multi-script workflow was not optimised for efficiency or parallelisation outside of the RF portions. Lastly, because of the relatively high programming requirements for usage of this multi-script procedure, requests for disaggregative population data produced using end-user specified covariate sets was often done when the WorldPop group had spare time. Other times data security and license concerns precluded end-users from sharing data that would have led to improved disaggregative population data.

Here we introduce the **`popRF`** package in `R` that largely addresses these issues. This is done by functionalising the RF-informed dasymetric population modelling procedure (F. R. Stevens et al. 2015) in a single language that is completely free, open source, and environment agnostic. Further, the package has been parallelised where possible to achieve efficient prediction and geoprocessing over large extents, providing functions that have applied utility outside of simply performing disaggregative population modelling. This package was utilised already to predict population and inform the mapping of modelled human settlement (Nieves, Sorichetta, et al. 2020; Nieves, Bondarenko, et al. 2020; Nieves et al. 2021) at 100m resolution across 249 countries from 2000-2020, ingesting over 10TB of covariates (Lloyd et al. 2019) and producing another 70 TB of population and population related datasets.

1.1 Disaggregative population mapping

There are multiple means to disaggregate count type data, particularly population counts (Leyk et al. 2019; Palacios-Lopez et al. 2019; Martin and Bracken 1991; J. Mennis 2003; J. Mennis and Hultgren 2006; Jeremy Mennis 2009). In general however, the idea is to produce weights, whether by expert opinion, a function of the source and target area geometries, or statistical relationships between ancillary data and the counts or densities, that are used to inform the disaggregation of counts (J. Mennis 2003; J. Mennis and Hultgren 2006; Jeremy Mennis 2009). Areal reweighting is one of the simplest disaggregative processes which uses the relationship of the geometries of the source and target areas. While the `popRF` package does not perform simple areal reweighting, understanding areal reweighting in comparison to the, slightly, more involved dasymetric disaggregation is useful. We briefly cover area reweighting as a primer for dasymetric disaggregation.We then move to dasymetric disaggregation, specifically "intelligent" dasymetric disaggregation which uses a statistical or machine learning approach to generate the weights based upon spatially coincident ancillary data, such as land cover (J. Mennis and Hultgren 2006). We finally cover a specific case of dasymetric disaggregation of population counts using a random forest to generate the weights (F. R. Stevens et al. 2015), which is the focus of the `popRF` package.

1.1.1 Areal reweighting

Given some source area which is comprised of smaller target areas, areal reweighting is where the value of the source area is redistributed to the smaller constituent target areas in proportion to the targets' areas (Figure 1). That is a larger target area will receive a larger disaggregated value, compared to a smaller target area, because its area is proportionally larger relative to their common source area. The sum of all weights adds up to 1.

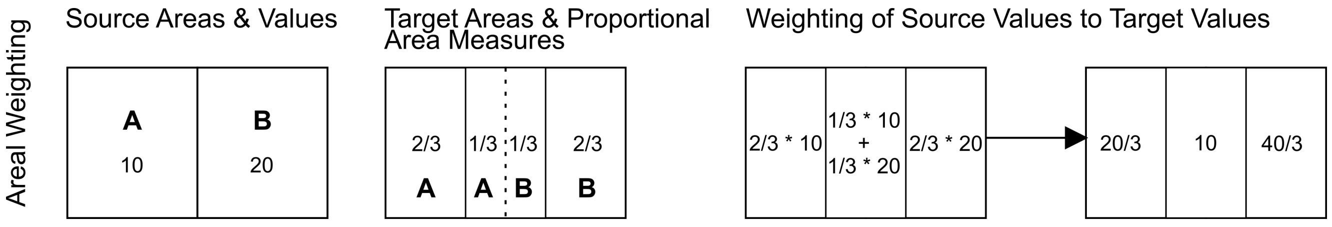


Figure 1: Basic diagram of areal reweighting with the original source area and its value; the target areas; the target area weights, as calculated by their proportional area of the source area; and the final disaggregated values of the target areas.

1.1.2 Dasymetric disaggregation

Dasymetric disaggregation is similar to areal reweighting in that the target areas have an assigned weight, which is relative to their larger common source area (Eicher and Brewer 2001). However, dasymetric weights are not determined by proportional area, but based upon ancillary or supporting data that is at the target area scale, such as classified landcover (J. Mennis 2003). These weights can be determined *a priori*, by expert opinion, or by statistical means, i.e. models (J. Mennis 2003; J. Mennis and Hultgren 2006). This last scenario has been referred to as "intelligent" dasymetric mapping (J. Mennis and Hultgren 2006). In this case, a modelling or algorithmic procedure determines a weighting value for each target area based upon the ancillary covariates provided (J. Mennis and Hultgren 2006). These weights are normalised within each source area, to ensure they add up to 1, before being used to disaggregate the value from the source area (J. Mennis and Hultgren 2006) (Figure 2).

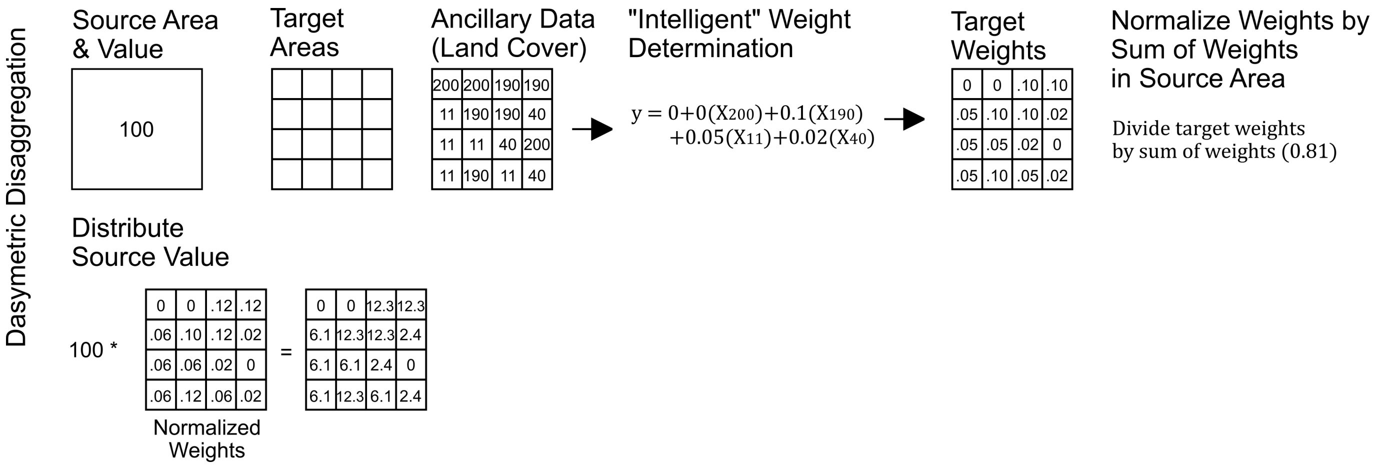


Figure 2: Basic diagram of dasymetric reweighting with the source area; the target areas; the ancillary information, in this case classified land cover; the source area normalised weights; and the final disaggregated values.

1.2 Random forest-informed population disaggregation

Building upon the dasymetric disaggregation concept, (F. R. Stevens et al. 2015) selected a random forest (RF) to be the algorithm to determine the target area weights. RFs are a non-parametric ensemble modelling method that are capable of handling continuous and categorical data and capture highly non-linear phenomena and complex interactions (Breiman 2001). A RF was selected due to these attributes and for: their robustness to noise and overfitting, their ability to handle small and large samples, their lack of user input, and their ability to be run in parallel (Breiman 2001; Liaw and Wiener 2002; F. R. Stevens et al. 2015).

RFs are an ensemble modelling method composed of numerous, typically hundreds, classification and regression trees (CARTs), hence the "forest" in RF (Breiman 2001). In the **`popRF`** package, we utilise the basic RF version given in the **`randomForest`** (Liaw and Wiener 2002) package as a regression model, although many alternative implementations exist. Given a set of data, each CART is independently constructed by first performing bagging, or sampling of the entire dataset with replacement, typically with each observation being included in 2/3 of all samples (Breiman 2001). The unsampled data is set aside as the out-of-bag (OOB) sample (Breiman 2001). The bagged sample is then used to construct the CART in an iterative fashion where, at each node (Figure 3), a random subsample of the input covariates are tested to determine a splitting value that maximises an information criterion and by which the data is split into two smaller subsets (Breiman 2001). This is repeated until some stopping criteria are met or there is a single observation in each "leaf" node (Breiman 2001). The OOB data is then run through the CART to determine its error and, when combined with the OOB error of other CARTS in the RF, serve as an internal cross-validation estimate of the RFs general error (Breiman 2001). After the total number of CARTs to be in the RF are constructed, predictions are made by giving the data to each CART and then their individual predictions are aggregated into a single prediction by taking the average (Breiman 2001).

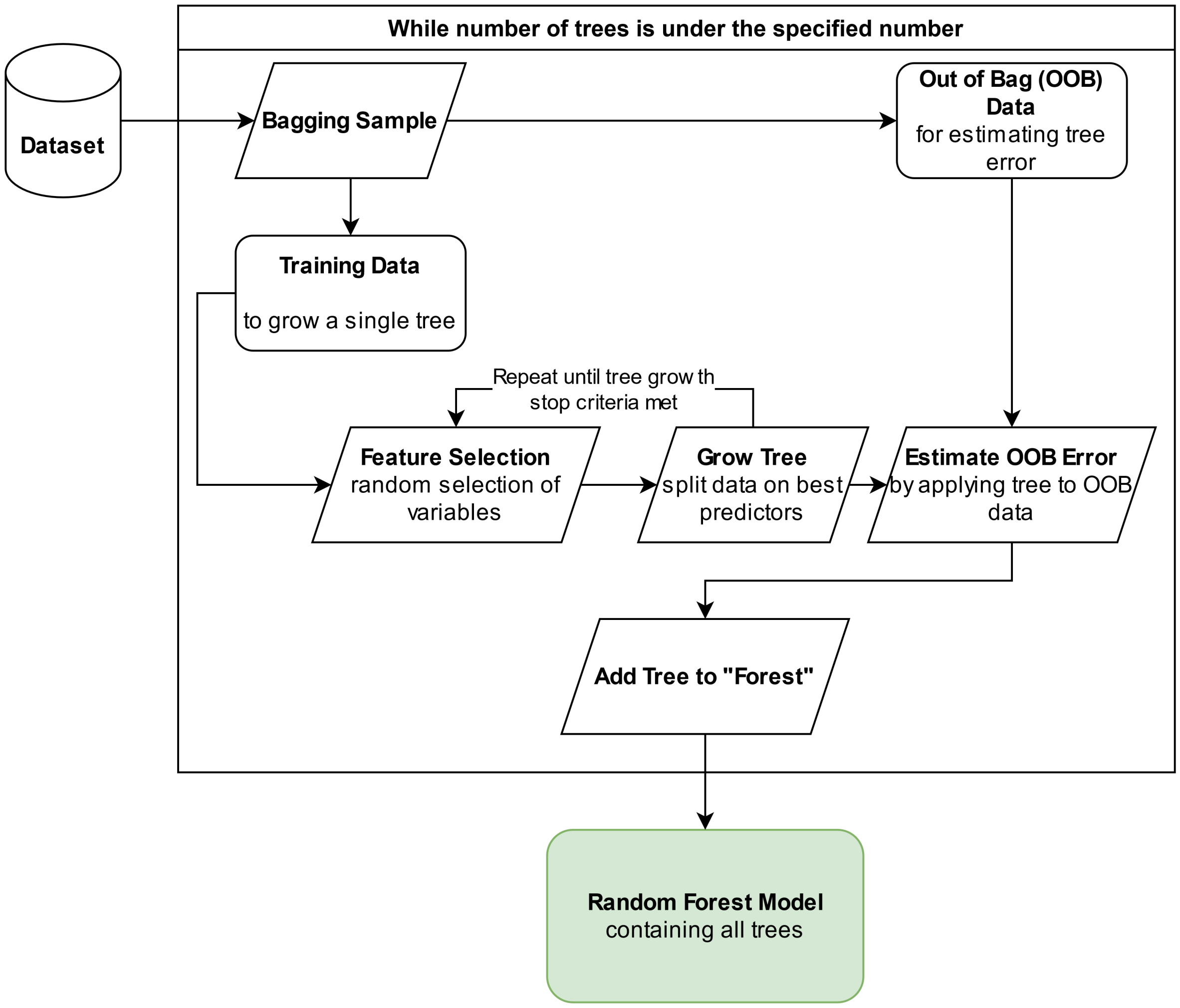

While number of trees is under the specified number
Dataset
Bagging Sample
Out of Bag (OOB) Data for estimating tree error
Training Data to grow a single tree
Repeat until tree growth stop criteria met
Feature Selection random selection of variables
Grow Tree split data on best predictors
Estimate OOB Error by applying tree to OOB data
Add Tree to "Forest"
Random Forest Model containing all trees

Figure 3: Generalised RF process diagram.

In this dasymetric mapping scenario, the source areas were subnational areal units with associated census-based population counts (Figure 4). The target areas were pixels representing approximately 100m at the equator (0.0008333°) (Figure 4). Population counts were first log transformed at the source area scale (F. R. Stevens et al. 2015). Ancillary data regarding land cover, topography, climate, and the built environment are aggregated to the source area scale and given to the RF as covariates to be trained against source area population density (Gaughan et al. 2014; F. R. Stevens et al. 2015) (Figure 4). Given the input covariates, a conservative covariate selection procedure is conducted. An initial RF is fit using all input covariates and each variables' importance recorded (F. R. Stevens et al. 2015). When using a RF as a regression, the variable importance is measured by the percent increase in the mean square error (Per.Inc.MSE) when a given covariate's values are randomly permutated, i.e. shuffled, effectively breaking its relationship with the outcome of interest (Breiman 2001). The larger the Per.Inc.MSE of the covariate, the more important the covariate within the RF (Breiman 2001). Covariates with Per.Inc.MSE values less than zero are removed, the RF refit on the reduced covariate set, and covariate importance rechecked (Figure 4) (F. R. Stevens et al. 2015). This is repeated until only non-zero, positive importance covariates remain. The final RF model is then fit and used to predict back-transformed population density at the pixel level for use as dasymetric weights (F. R. Stevens et al. 2015). These pixel level population density weights are then normalised by their sum within each source area, to have all normalised weights within a source area add up to one (Figure 4). The normalised weights are then used to disaggregate the source area population counts, with the disaggregated target counts always summing back up to the source area total (F. R. Stevens et al. 2015).

Because RFs are an ensemble modelling method, this means that multiple RFs can naturally be combined [Breiman (2001)}. This is particularly useful in scenarios where one country has poor quality areal population data, i.e. few and large areal units, or too few subnational units to train an RF model (Gaughan et al. 2014). In this case, the RF model trained on a similar country can be used in conjunction with poor quality data, or by itself to predict the density weights (Gaughan et al. 2014). This regional parameterisation scenario is detailed more in Gaughan et al. (2014) and Sinha et al. (2019). The **`popRF`** package follows the described RF-informed dasymetric disaggregation procedure and allows for the regional parameterisation scenarios.

2 Using popRF

The **`popRF`** package contains one primary function: `popRF()`. `popRF()` provides a single function call to produce RF-informed dasymetric disaggregations of population count data in a gridded format, following the process in Figure 4. The input data can either be from local paths, from the online geospatial library provided by WorldPop (Lloyd et al. 2019), other network/internet locations, or a mix of all of these. It is assumed that all of the input data is raster-based, unprojected, e.g. WGS84 coordinate system, and already spatially aligned. Models using the `popRF()` function can be run specifically for a single country, multiple countries, for a single country using the RF from a previously run country/countries, or for a single country combining the given country's RF with another previously run country/countries RF(s). In the remainder of this section, we'll cover the specifications of the input data, the types of model specifications that can be run using the `popRF()` function, as well as the how the internal helper functions carry out the modelling.

`popRF` utilises the following packages: **`Rcpp`** (Eddelbuettel et al. 2021; Eddelbuettel and François 2011; Eddelbuettel and Balamuta 2018), **`doParallel`** (Corporation et al. 2020), **`foreach`** (Revolution Analytics and Weston, n.d.), **`gdalUtils`** (Greenberg and Mattiuzzi 2020), **`plyr`** (Wickham 2011, 2020), **`quantregForest`** (Meinshausen 2017), **`randomForest`** (Liaw and Wiener 2002), **`raster`**(Hijmans 2021), **`rgdal`** (R. Bivand, Keitt, and Rowlingson 2021), and **`sp`** (Pebesma and Bivand 2021; R. S. Bivand, Pebesma, and Gomez-Rubio 2013). Additionally, throughout the rest of

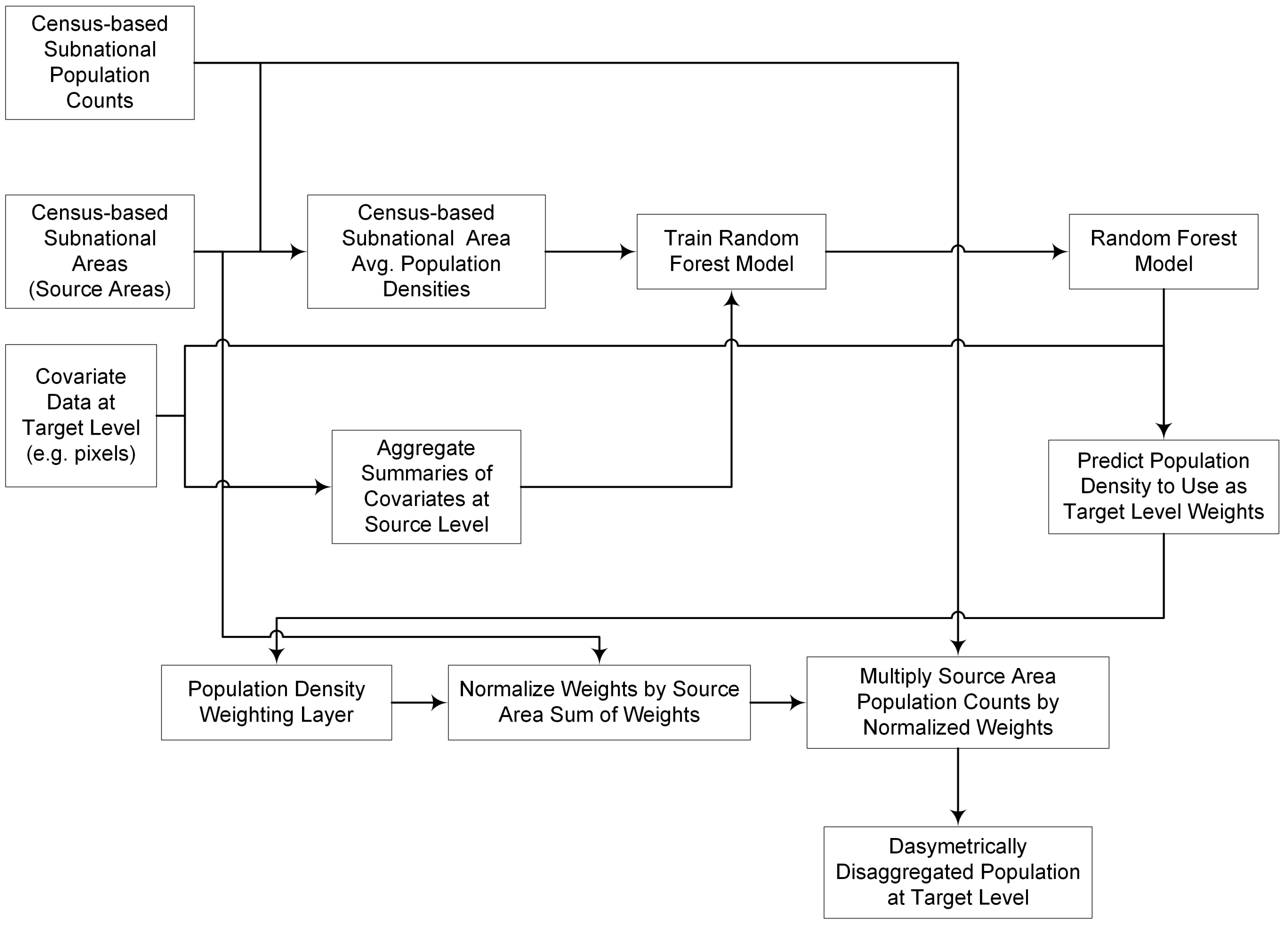

Census-based Subnational Population Counts
Census-based Subnational Areas (Source Areas)
Census-based Subnational Area Avg. Population Densities
Train Random Forest Model
Random Forest Model
Covariate Data at Target Level (e.g. pixels)
Aggregate Summaries of Covariates at Source Level
Predict Population Density to Use as Target Level Weights
Population Density Weighting Layer
Normalize Weights by Source Area Sum of Weights
Multiply Source Area Population Counts by Normalized Weights
Dasymetrically Disaggregated Population at Target Level

Figure 4: Generalised process diagram of a RF-informed dasymetric disaggregation of census-based population counts as described in Stevens et al. (2015) and implemented in the popRF package.

the article, we also utilise: **ggplot2** (Wickham 2016), **ggspatial** (Dunnington 2021), and **viridis** (Garnier et al. 2021). All operations were run using `R` 4.0.5 "Shake and Throw" (R Core Team 2021).

2.1 General data requirements

All input data should be of a raster format and is assumed to be unprojected but with a common, defined coordinate system, e.g. WGS84. This is why, as one of the parameters input to the `popRF()` function, we require a raster for "pixel area" whose values should contain the precalculated area of each output pixel. This unprojected approach to modelling was chosen to avoid the need for manual specification of projections, of which there are many for a given country, as well as facilitate the creation of models that incorporate multiple countries of data at once (Gaughan et al. 2014; Sinha et al. 2019). Further, all input raster data should be spatially aligned and harmonised, i.e. each raster layer should have the same extents, a common origin, and a common, uniform resolution. The `popRF()` function is focused on the modelling and disaggregation and as such, the geoprocessing that must occur prior to the use of the function is left to the user to facilitate.

2.2 Minimum data requirements

Data requirements for carrying out the `popRF()` function workflow will vary based upon the specific type of model being run, e.g. "single country general" versus "single country hybrid" (Table 1). Though, at a minimum, zonal data representing subnational areas, corresponding tabular population count data for those areas, a binary water mask indicating pixels of water (*1*) and no water (*0*), a raster containing the area of each pixel as its pixel values, and at least one ancillary covariate dataset must be provided. The areal zonal data must have a unique ID for each zone and these IDs must correspond to the tabular population count data. Common ways the `popRF()` function can be run and their data requirements are given below in Table 1.

```
#>
#> Attaching package: 'dplyr'
#> The following object is masked from 'package:randomForest':
#>
#>     combine
#> The following objects are masked from 'package:raster':
#>
#>     intersect, select, union
#> The following objects are masked from 'package:stats':
#>
#>     filter, lag
#> The following objects are masked from 'package:base':
#>
#>     intersect, setdiff, setequal, union
#>
#> Attaching package: 'kableExtra'
#> The following object is masked from 'package:dplyr':
#>
#>     group_rows
```

Table 1: Common popRF model run types and corresponding data requirements. For all model runs, zonal data, corresponding tabular population count data, and at least one ancillary dataset must be provided in addition to the below specified data. Each of these models can be run for either a single or multiple country scenario.

Model Type Designation	Input Data
General	Zonal Data (Raster)
	Population Counts (Tabular)
	Water Mask (Raster)
	Pixel Area (Raster)
	Ancillary Datasets (Rasters)
Hybrid-parameterised	Zonal Data (Raster)
	Population Counts (Tabular)
	Water Mask (Raster)
	Pixel Area (Raster)
	Ancillary Datasets (Rasters)
	RF Model Object from Other Country(ies) (.RData)
Fully-parameterised	Zonal Data (Raster)
	Population Counts (Tabular)
	Water Mask (Raster)
	Pixel Area (Raster)
	Ancillary Datasets (Rasters)
	RF Model Object from Other Country(ies) (.RData)

In the case of the hybrid- and fully-parameterised model types, the full path(s) of the directories containing one or more RF model objects from one or more countries must be provided to the parameters of the `popRF()` function by properly defining the relevant fixed set parameters. It is important that the names of the ancillary datasets match those of the previously run RF model otherwise an error will occur.

2.3 User settings and parameters

The `popRF()` function has five required parameters and 12 optional parameters related to the overall process (Table 2), as well as an additional five optional parameters that are passed to the internally called `randomForest()` function (Table 3). The required parameters involve the paths to the input files containing: the areal, population count data (*pop*); the "mastergrid" raster containing the areas with unique IDs corresponding to the population data (*mastergrid*); the covariates for the RF (*cov*); the raster indicating water extents to be used as a mask (*watermask*); and the raster containing the area of each unprojected pixel (*px_area*). All optional parameters involve: the location for writing outputs (*output_dir*), options for parallel processing (cores* and *minblocks*), running a quantile RF regression options (*quant*), model type options (*fset*, *fset_incl*, and *fset_cutoff*), covariate alignment options (*fix_cov*), output checking (*check*), and processing messaging options (*verbose* and *log*). We give a brief description of all parameters, but refer the reader to the **`popRF`** package documentation for additional detail not included here.

Table 2: Parameters of the popRF() function with brief descriptions.

Parameter Name	Type	Description
pop	Character	Contains the name of the file from which the unique area ID and corresponding population values are to be read from.
cov	List of named lists	The name of each named list corresponds to the 3-letter ISO code (ISO 3166-1 alpha-3) of a specified country. The elements within each named list define the specified input covariates to be used in the RF model, i.e. the name of the covariates and the corresponding, if applicable and local, path to them.
mastergrid	Named list	Each element of the list defines the path to the input mastergrid(s), i.e. the template gridded raster(s) that contains the unique area IDs as their value. The name(s) corresponds to the 3-letter ISO code(s) of a specified country(ies). Each corresponding element defines the path to the mastergrid(s).
watermask	Named list	Each element of the list defines the path to the input country-specific water mask. The name corresponds to the 3-letter ISO code of a specified country. Each corresponding element defines the path to the water mask, i.e. the binary raster that delineates the presence of water (1) and non-water (0), that is used to mask out areas from modelling.
px_area	Named list	Each element of the list defines the path to the input raster(s) containing the pixel area. The name corresponds to the 3-letter ISO code of a specified country. Each corresponding element defines the path to the raster whose values indicate the area of each unprojected (WGS84) pixel.
output_dir	Character	Optional. Contains the path to the directory for writing output files.
cores	Integer	Optional. Indicates the number of cores to use in parallel when executing the function.
quant	Logical	Optional. Indicates whether to produce the quantile regression forests (TRUE) to generate prediction intervals.
set_seed	Integer	Optional. Value to be used to set the random seed. Value is 2010 by default.
fset	Named list	Optional. Each element of the list defines the path to the directory(ies) containing the RF model objects (.RData) with which we are using as a 'fixed set' in this modeling, i.e. are we parameterising, in part or in full, this RF model run upon another country's(ies') RF model object. The name of each element corresponds to the 3-letter ISO code of a specified country.

Table 2: Parameters of the popRF() function with brief descriptions. *(continued)*

Parameter Name	Type	Description
fset_incl	Logical	Optional. Indicates whether the RF model object will or will not be combined with another RF model run upon another country's(ies') RF model object, i.e. "Hybrid Parameterised".
fset_cutoff	Integer	Optional. This parameter is only used if fset_incl is TRUE. If the country has less than fset_cutoff admin units, then RF popfit will not be combined with the RF model run upon another country's(ies') RF model object. This is to avoid trying to run a RF on countries with more covariates than admin units.
fix_cov	Logical	Optional. Indicates whether the raster extent of the covariates will be corrected if the extent does not match the mastergrid(s).
check_results	Logical	Optional. Indicates whether the results will be compared with input data, i.e. makes sure the sum of disaggregated pixel values equals the total of each source subnational unit the values were disaggregated from
verbose	Logical	Optional. Indicates whether to print intermediate output from the function to the console.
log	Logical	Optional. Indicates whether to print intermediate output from the function to the log.txt file.
...		Optional. Additional parameters passed directly to the 'randomForest()' function or influence the efficiency of parallel processing.

There are some additional optional parameters that are passed directly to the `randomForest()` function that is called internally which we detail in Table 3. Further details on the parameters of the `randomForest()` function are given in (Liaw and Wiener 2002).

2.3.1 Typical covariates and their sources

Covariates derived from remote sensing imagery, such as land cover, urban and settlement presence and density, nighttime lights, and vegetation, and other covariates derived from spatial environmental data, such as roads, health and education services, points of interest, water bodies, and protected land, are commonly used as "ancillary" data to produce the dasymetric weights used in disaggregating population (Martin and Bracken 1991; Balk and Yetman 2004; J. Mennis and Hultgren 2006; Bhaduri, Bright, and Coleman 2007; Gaughan et al. 2014; Sorichetta et al. 2015; F. R. Stevens et al. 2015; European Commission and Columbia University 2015; Friere et al. 2016; Reed et al. 2018; Palacios-Lopez et al. 2019; Nieves et al. 2021). Such covariate data can be locally developed, e.g. manual digitisation, or be derived from openly available sources as long as they are processed into raster format with a consistent resolution, are spatially aligned, and are unprojected in a common coordinate system (e.g. WGS84). The covariates can have any name, but will require consistency in the naming, between models, for model fitting and predictions to work when running wither a hybrid-parameterised or fully-parameterised model (Table 1).

2.3.2 Population and subnational boundary data

The population data that is input to the `popRF()` function is tabular in nature and should be in a comma separated value (CSV) file format. The data should consist of two columns: the first column containing rows of unique Geographic ID (GID), which correspond to the subnational areas defined by the raster provided to the *mastergrid* parameter, and the second column containing rows of values of population counts for each subnational area. The CSV file should not contain a header with column names.

2.4 Internal optimisation

One of the additional benefits of the `popRF()` function is that several portions of the process have been optimised for parallel processing, namely the RF predictions at the pixel/grid level and a zonal statistics function that calculates aggregate summaries of pixels within a given raster of zones. Both of these are carried out within internal functions that estimate the number of optimal number of blocks based upon the covariate and spatial contexts. These are written in `R` using the dependencies of the **`parallel`** (R Core Team 2021) and **`doParallel`** (Corporation et al. 2020) packages.

2.4.1 Random forest predictions

The RF prediction of the pixel level weights, used in the dasymetric redistribution of the population counts (Figure 2), has been parallelised since F. R. Stevens et al. (2015) and the corresponding code at (Forrest R. Stevens 2014). This portion of the process divides the study area covered by the covariates into subareas, referred to as "blocks," and then independently predicts for these areas using the same trained RF. The predicted values are then written out to a single raster, block by block. Much of the logic and code from F. R. Stevens et al. (2015) and Forrest R. Stevens (2014)

Table 3: Table of additional, optional parameters within the popfit() function that are passed internally to the randomForest() function.

Parameter Name	Type	Description
binc	Numeric	A constant value to increase the calculated number of blocks by. This can be used to set a larger 'safety' margin for the calculated allocation of computational resources.
boptimise	Logical	Indicates whether the calculation of blocksize should be 'optimised' to use 65 percent of available memory in the estimation.
bsoft	Logical	Indicates the 'softness' (or hardness) of the specification of cores to use in the modelling. If 'TRUE' and the task can be processed in a number of blocks less than the specified value of 'cores', then the smaller number of blocks will be used.
proximity	Logical	Indicates whether proximity measures among the rows should be computed.
nodesize	Integer	Declares a fixed value for the number of observations to be in each terminal node of the RF. Is passed to the *nodesize* parameter of the 'randomForest()' function. If 'NULL', defaults to 1/20th the number of observations.
maxnodes	Integer	Declares a fixed value for the maximum number of terminal nodes in a tree within the RF. Is passed to the *maxnodes* parameter of the 'randomForest()' function. Default, of 'NULL', is no limit.
ntree	Integer	Declares a fixed value for the number of trees to grow within the RF. Is passed to the *ntree* parameter of the 'randomForest()' function. Default, if 'NULL', is 500 trees.
mtry	Integer	Declares a fixed value for the number of covariates to try at each split within a tree. Is passed to the *mtry* parameter of the 'randomForest()' function. If 'NULL', the value fit from the 'tuneRF()' function is used instead.
const	Character	Contains the path to the binary raster file that will be used as a mask, where masked areas have value equal to zero. This mask is used to constrain the population modelling layers.

persists, however, we have provided an internal logic for determining the block size (see minblocks parameter) in relation to the overall study area size and the computational resources at hand. This is detailed in the internal `get_blocks_need()` function presented in Appendix A. This removes the need for hardcoding of the block size or number of blocks and allows for more standardised and efficient means, as compared to the previous trial and error, of subdividing the modelling task into subtasks for parallel processing.

2.4.2 Zonal statistics

The basic `zonal()` function in the **`raster`** package has long been known to be less than efficient when given large rasters and or many zonal features (Horning et al. 2013; Forrest R. Stevens 2014; Marrotte 2016; Lovelace, Nowosad, and Muenchow 2021). As a solution, we have created an internal function, `calculate_zs_parallel()`, that breaks the zonal statistic calculation task into many independent tasks that can be calculated in parallel. Again, using the concept of blocks, the function extracts the values from the value raster and the zonal raster into data.frames, carries out tabular summary calculations, and outputs the zonal statistics to a final `data.frame`. This all allows for more efficient calculation of zonal statistics. This function is called internally when summaries of the specified input covariates (see the *cov* parameter) do not already exist.

3 Exemplified test cases

For the remainder of this paper, and to focus more on the `popRF()` function modelling rather than geoprocessing, we will be using the geospatial library of data (Lloyd et al. 2019), created and hosted by the WorldPop research group (ftp://ftp.worldpop.org/GIS/). These data include spatially harmonised covariates pertaining to the environment as well as rasterised subnational areas that correspond to tabular, comma separated population data (.csv format). We give example `R` code in Appendix B on the programmatic downloading of these data, but they can be downloaded via a browser or FTP client. For the following examples we will use the countries of Guyana (ISO 3166-1 alpha-3: "GUY"), Suriname (ISO 3166-1 alpha-3: "SUR"), and French Guiana (ISO 3166-1 alpha-3: "GUF"). We download the data to the project root path of "`B:/Research/tmp_popRF/`," which we refer to simply as "`./`" hereafter for conciseness, but, of course, in replication the user may wish to change this directory location for their own file setup.

3.1 Single country, general run

For our "single country, general" model run, we will use Guyana (GUY). We have already downloaded all the data using the code in Appendix B and the "GUY" ISO tag. GUY consists of a number of subnational units ($n = 117$) and will serve as a reference model when running the later hybrid- and fully-parameterised model runs. Note, that for all of these countries, the number of subnational units that "exist" may be higher in the official public releases or in non-public datasets, but here, and for all other examples, we are using the boundary matched population data from Lloyd et al. Lloyd et al. (2019) based upon Doxsey-Whitfield et al.(Doxsey-Whitfield et al. 2015).

First, we need to make sure our folder structure is defined.

```r
country <- "GUY"
project_root_path <- "B:/Research/tmp_poprf/"
input_dir <- paste0(project_root_path, country, "/covariates/")
input_population <- paste0(input_dir,"guy_population.csv")
output_directory <- paste0(project_root_path,country,"/output")
if (!dir.exists(output_directory)) {
   dir.create(output_directory)
}
```

And we then define our input covariates in a nested list.

```r
input_covariates <- list(
      country = list(
         "dst011_2015" = file.path(input_dir,"esaccilc_dst011_100m_2015.tif"),
         "dst040_2015" = file.path(input_dir,"esaccilc_dst040_100m_2015.tif"),
         "dst130_2015" = file.path(input_dir,"esaccilc_dst130_100m_2015.tif"),
         "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
         "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
         "dst160_2015" = file.path(input_dir,"esaccilc_dst160_100m_2015.tif"),
         "dst190_2015" = file.path(input_dir,"esaccilc_dst190_100m_2015.tif"),
         "dst200_2015" = file.path(input_dir,"esaccilc_dst200_100m_2015.tif"),
         "dst_water" = file.path(input_dir,"esaccilc_dst_water_100m_2000_2012.tif"),
         "dst_bsgme_2020" = file.path(input_dir,"dst_bsgme_100m_2020.tif"),
         "dst_ghsl_2000" = file.path(input_dir,"dst_ghslesaccilc_100m_2000.tif"),
         "dst_intersec_2016" = file.path(input_dir,"osm_dst_roadintersec_100m_2016.tif"),
         "dst_waterway_2016" = file.path(input_dir,"osm_dst_waterway_100m_2016.tif"),
         "dst_road_2016" = file.path(input_dir,"osm_dst_road_100m_2016.tif"),
         "slope" = file.path(input_dir,"srtm_slope_100m.tif"),
         "topo" = file.path(input_dir,"srtm_topo_100m.tif"),
         "dst_coast" = file.path(input_dir,"dst_coastline_100m_2000_2020.tif"),
         "viirs_2016" = file.path(input_dir,"viirs_100m_2016.tif"),
         "wdpa_dst_2017" = file.path(input_dir,"wdpa_dst_cat1_100m_2017.tif")
      )
   )
names(input_covariates) <- c(country)
input_mastergrid <- list(
   country = file.path(input_dir,"subnational_admin_2000_2020.tif")
)
names(input_mastergrid) <- c(country)
input_watermask <- input_watermask <- list(
   country = file.path(input_dir,"esaccilc_water_100m_2000_2012.tif")
)
names(input_watermask) <- c(country)
input_px_area <- list(
   country = file.path(input_dir,"px_area_100m.tif")
)
names(input_px_area) <- c(country)
input_poptables <- list(
   country = input_population
)
names(input_poptables) <- c(country)
```

We can then call the `popRF()` function using these inputs. We will use the default blocks parameter of `NULL` to allow it to automatically determine the number of blocks to divide the modelling into, but this can be alternately specified by the user. We also set the log parameter to TRUE to write the intermediate messages to a text file and disable the calculation of proximity measures in the RF; the proximity measures are not of interest to us currently and this increases the computational load (particularly for large numbers of subnational units or large geographic extent).

```r
require(popRF)
guy <- popRF(pop = input_poptables,
             cov = input_covariates,
             mastergrid = input_mastergrid,
             watermask = input_watermask,
             px_area = input_px_area,
             output_dir = output_directory,
             cores = 8,
             quant = TRUE,
             proximity = FALSE,
             set_seed = 1964L,
             fix_cov = FALSE,
             check_result = TRUE,
             log = TRUE)
```

The `popRF()` function returns a named, two-item list with the first item, `pop`, being the raster containing the disaggregated and gridded population counts and the second item, `popfit`, which is the final RF model object. If the option `check_results = TRUE`, then a three-item list is returned with the addition of an item `error` which provides the average absolute difference in the modeled population versus the input subnational population counts. These output data/objects, and more, are written to the "`./*<ISO_CODE>*/output/`" folder. Of specific interest for the hybrid- and fully-parameterised model runs is that the final RF model object (filename pattern of "`popfit_final_*<ISO_CODE>*.RData`") and the quantile regression RF model object (filename pattern of "`popfit_quant_*<ISO_CODE>*.RData`") are written to the "`./*<ISO_CODE>*/output/tmp/`" folder. These model object files will need to be manually copied to the proper folders prior to any parameterised model run (Table 1).

3.1.1 Investigating the model and outputs

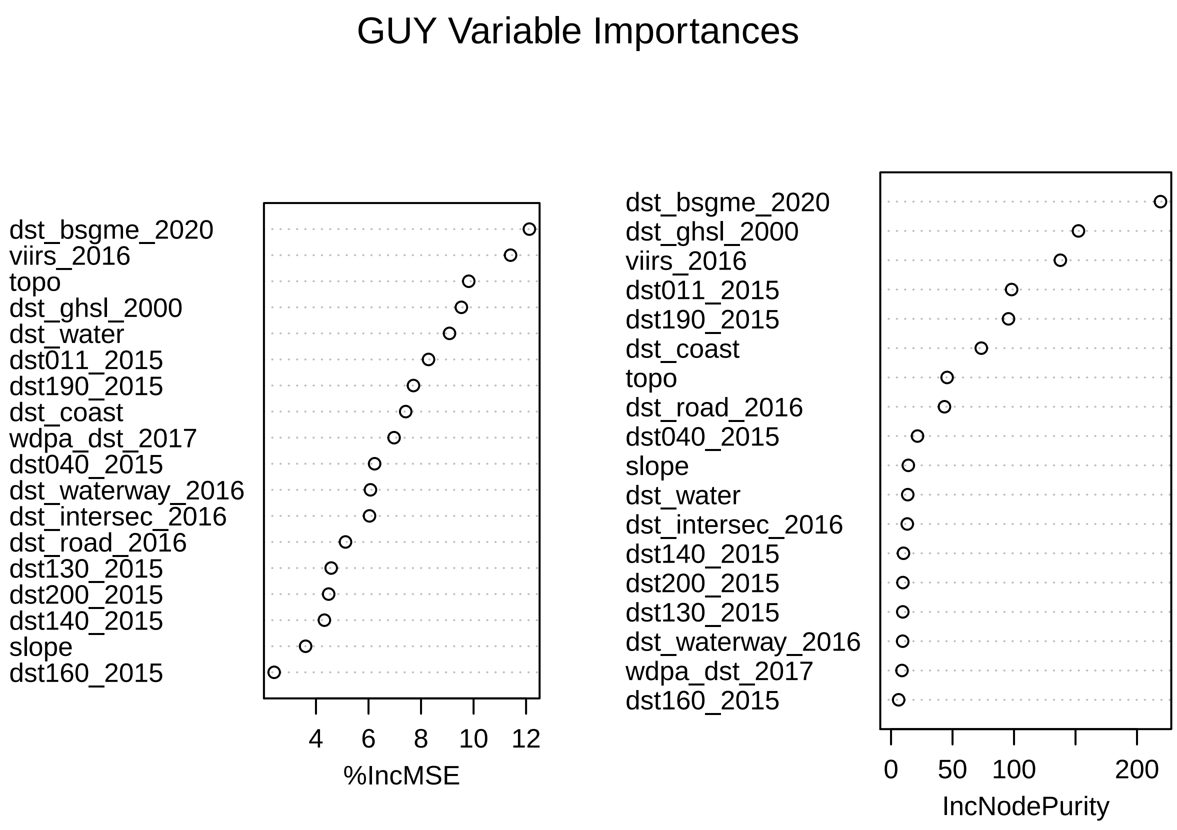
We can examine some of the outputs from the model run now. First, we'll look at the overall model fit of the RF.

```
require(randomForest)
guy$popfit
```

```
#> Call:
#>   randomForest(x = x_data y = y_data ntree = rf_ntree mtry = rf_mtry
#>        nodesize = rf_nodesize maxnodes = rf_maxnodes importance = TRUE
#>        proximity = proximity do.trace = F)
#>                 Type of random forest: regression
#>                       Number of trees: 500
#> No. of variables tried at each split: 6
#>
#>           Mean of squared residuals: 1.25317
#>                     % Var explained: 85.51
```

This gives us the standard summary output for RF model objects from the **`randomForest`** package (Liaw and Wiener 2002). We can see the original, internal `randomForest()` function call, that we grew 500 trees overall, and the auto-adjusted parameters that resulted in six variables tried at each split. Additionally, we can see that 85 percent of the data variance are explained in this model, as estimated from the out-of-bag sample validations. We can also look at the variable importances within the RF model.

```
require(randomForest)
randomForest::varImpPlot(guy$popfit,
                        main = "GUY Variable Importances",
                        cex = 0.8)
```



Further explorations of the RF model object are possible as with any standard RF model object from the **`randomForest`** package, such as partial dependency plots (Liaw and Wiener 2002), Accumulated Local Effect plots (Apley and Zhu 2016), and others.

We can also plot the final population raster to get an idea of the overall output modelled population surface.

```
require(raster)
require(ggplot2)
require(ggspatial)
require(viridis)
ggplot() +
  layer_spatial(guy$pop) +
  scale_fill_viridis(limits = c(0,5), na.value = NA) +
  ggtitle("Guyana, Modelled Population, 2020") +
  labs(fill = "Population\nPer Pixel") +
  theme(panel.background = element_rect(fill = "gray20"),
        panel.grid = element_line(color = "gray60"))
```

There are other output files, written to the "`./*<ISO_CODE>*/output/`" folder, that may be of interest that are not returned by the `popRF()` function. The complete list of output files, their format, and their description are given in Table 3.

\begin{table}

\caption{Files written to the '/output/' folder by the popRF() function, their format, and a description of what they include. This includes all files produced when popRF() function defaults are used. The names associated with the 'ZONAL' files (last row) will vary based upon user input

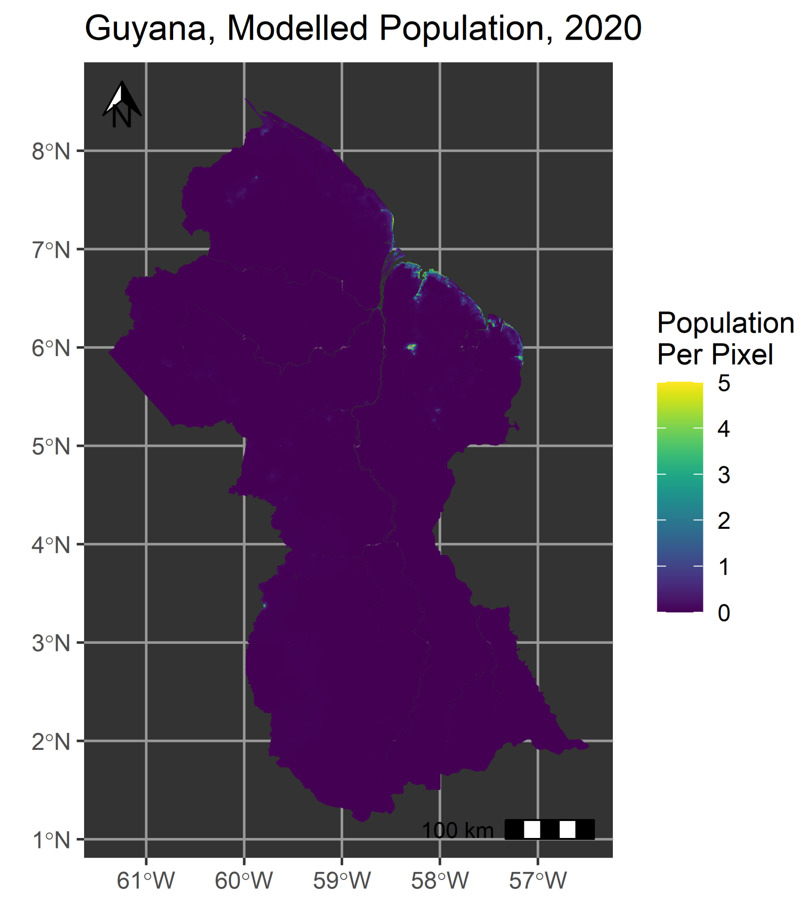


Figure 5: Population surface for Guyana using estimated 2020 population subnational counts and modelled using a RF-informed dasymetric disaggregation. Note, the population legend uses a maximum of 5 people per pixel for visualisation purposes; the people per pixel in the modelled data has a maximum of 91.

and naming conventions, but in general will follow the pattern of '__*ZS*.csv'. Optional files, such as those produced when log is TRUE, are not included here.}

File	Format	Folder	Description
*\<ISO\>*__census__data	.CSV	'../<ISO>/output/zonal__stats/'	Subnational population count data with columns representing the zonal statistics summaries of all input covariates.
check__result__prj__*\<ISO\>*	.CSV	'../<ISO>/output/tmp/'	CSV file containing the difference between the input subnational population counts and the sum of the modelled population counts per pixel in each subnational unit.
init__popfit__*\<ISO\>*	.RData	'../<ISO>/output/tmp/'	Initial RF model that tunes in the parameters for number of covariates to use at each split.
pop__census__mask__*\<ISO\>	.TIF	'../<ISO>/output/tmp/'	Binary mask of where predictions should or should not be made based upon where the input mastergrid has coverage.
popfit__*\<ISO\>*	.RData	'../<ISO>/output/tmp/'	Random forest model that has the final set of covariates that resulted from the selection procedure.
popfit__final__*\<ISO\>*	.RData	'../<ISO>/output/tmp/'	Final RF model that is used to produce the dasymetric weights.
popfit__quant__*\<ISO\>*	.RData	'../<ISO>/output/tmp/'	Quantile regression forest model object.
ppp__*\<ISO\>*	.TIF	'../<ISO>/output/tmp/'	Population Per Pixel (PPP) where pixel values are counts of people in the pixel.
predict__density__rf__pred__*\<ISO\>*	.TIF	'../<ISO>/output/tmp/'	Values to be used as dasymetric weights where pixel value is predicted population from the popfit__final model.
predict__density__rf__pred__*\<ISO\>*__ZS__sum	.TIF	'../<ISO>/output/tmp/'	Sum of values from predict__density__rf__pred__\<ISO\> by subnational unit; used to produce the dasymetric weights.
predict__density__rf__pred__05__*\<ISO\>*	.TIF	'../<ISO>/output/tmp/'	5th percentile of RF predictions.
predict__density__rf__pred__50__*\<ISO\>*	.TIF	'../<ISO>/output/tmp/'	50th percentile of RF predictions.
predict__density__rf__pred__90__*\<ISO\>*	.TIF	'../<ISO>/output/tmp/'	90th percentile of RF predictions.
predict__density__rf__sd__*\<ISO\>*	.TIF	'../<ISO>/output/tmp/'	Standard deviation of RF predictions.
ZONAL	.CSV	'../<ISO>/output/zonal__stats/'	Zonal statistic summaries by subnational unit.

\end{table}

3.2 Single country, hybrid-parameterised model run

Now we'll look at the example of Suriname (SUR), which has a relatively small number of subnational units ($n = 80$). Let's say we assume that while we can create a model using only SUR, that we believe it might be beneficial to include information from the GUY model we previously ran (see Section Single Country). In this case, we want to combine the RF model object trained on the GUY data with the RF we train on only the SUR data. We refer to this as a "hybrid" parametrised model since we are combining two independent random forests from two different study areas, e.g. countries. This method builds from F. R. Stevens et al. (2015) and was put forth in Gaughan et al. (2014).

To facilitate the hybrid model that will incorporate information from the GUY model in conjunction with the SUR trained model, we created a folder "`/final/`" and a folder "`/quant/`"} within country specific folder within the project root path, in this case "`B:/Research/tmp_poprf/SUR/`"}. From the "`./output/`" folder, where our GUY model and intermediaries were output, we have copied the "final" RF model for GUY into the "`/final/`"} folder as well as the quantile regression model object for GUY into the "`/quant/`"} folder. Both of these model objects are saved in `.RData` format. Note, if the quantile regression was not run for GUY and it is desired for SUR, then the GUY model would need to be rerun to include a quantile regression. Alternatively, if the quantile regression was run for GUY but was not desired for SUR, then there is no need to copy over the quantile regression model object (`.RData`) to the "`/quant/`" folder.

After copying the necessary model objects to the correct folders, we first download the data for SUR using the code in Appendix B and then define our parameters that we will pass to the `popRF()` function.

```r
country <- "SUR"
project_root_path <- "B:/Research/tmp_poprf/"
input_dir <- paste0(project_root_path, country, "/covariates/")
input_population <- paste0(input_dir, tolower(country), "_population.csv")

output_directory <- paste0(project_root_path,country,"/output/")
##  If the directory does not exist, create it:
if (!dir.exists(output_directory)) {
   dir.create(output_directory)
}
```

Take note that since we are parameterising the model on a previously run RF, we will need to make sure we have the same covariates and covariates names, otherwise conflicts will occur when trying to provide the data to the RF.

```r
input_covariates <- list(
      country = list(
         "dst011_2015" = file.path(input_dir,"esaccilc_dst011_100m_2015.tif"),
         "dst040_2015" = file.path(input_dir,"esaccilc_dst040_100m_2015.tif"),
         "dst130_2015" = file.path(input_dir,"esaccilc_dst130_100m_2015.tif"),
         "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
         "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
         "dst160_2015" = file.path(input_dir,"esaccilc_dst160_100m_2015.tif"),
         "dst190_2015" = file.path(input_dir,"esaccilc_dst190_100m_2015.tif"),
         "dst200_2015" = file.path(input_dir,"esaccilc_dst200_100m_2015.tif"),
         "dst_water" = file.path(input_dir,"esaccilc_dst_water_100m_2000_2012.tif"),
         "dst_bsgme_2020" = file.path(input_dir,"dst_bsgme_100m_2020.tif"),
         "dst_ghsl_2000" = file.path(input_dir,"dst_ghslesaccilc_100m_2000.tif"),
         "dst_intersec_2016" = file.path(input_dir,"osm_dst_roadintersec_100m_2016.tif"),
         "dst_waterway_2016" = file.path(input_dir,"osm_dst_waterway_100m_2016.tif"),
         "dst_road_2016" = file.path(input_dir,"osm_dst_road_100m_2016.tif"),
         "slope" = file.path(input_dir,"srtm_slope_100m.tif"),
         "topo" = file.path(input_dir,"srtm_topo_100m.tif"),
         "dst_coast" = file.path(input_dir,"dst_coastline_100m_2000_2020.tif"),
         "viirs_2016" = file.path(input_dir,"viirs_100m_2016.tif"),
         "wdpa_dst_2017" = file.path(input_dir,"wdpa_dst_cat1_100m_2017.tif")
      )
   )
names(input_covariates) <- c(country)
input_mastergrid <- list(
   country = file.path(input_dir,"subnational_admin_2000_2020.tif")
)
names(input_mastergrid) <- c(country)
input_watermask <- input_watermask <- list(
   country = file.path(input_dir,"esaccilc_water_100m_2000_2012.tif")
)
names(input_watermask) <- c(country)
input_px_area <- list(
   country = file.path(input_dir,"px_area_100m.tif")
)
names(input_px_area) <- c(country)
input_poptables <- list(
   country = input_population
)
names(input_poptables) <- c(country)
```

We can now pass these arguments to the `popRF()` function, taking care to set our "fixed set" parameters (e.g. `fset`, `fset_incl`) for a hybrid model run such that a RF is created based solely on the SUR data and then merged with the GUY RF, i.e. `fset_incl = TRUE`.

```
require(popRF)
sur <- popRF(pop = input_poptables,
             cov = input_covariates,
             mastergrid = input_mastergrid,
             watermask = input_watermask,
             px_area = input_px_area,
             output_dir = output_directory,
             cores = cores,
             fset = list("final " = paste0(project_root_path, country,
                                          "/final/"),
                         "quant" = paste0(project_root_path, country,
                                         "/quant/")),
             fset_incl = TRUE,
             quant = FALSE,
             set_seed = 1964L,
             proximity = FALSE,
             fix_cov = FALSE,
             check_result = TRUE,
             log = TRUE)
```

This gives us the following model summary and population map. Note that the summary does not have an estimate of the variance explained since we merged two independent models together.

```
require(raster)
require(randomForest)
sur$popfit
```

```
#> Call:
#>   randomForest(x = x_data y = y_data ntree = rf_ntree mtry = rf_mtry
#>        nodesize = rf_nodesize maxnodes = rf_maxnodes importance = TRUE
#>        proximity = proximity do.trace = F)
#>                  Type of random forest: regression
#>                        Number of trees: 500
#>   No. of variables tried at each split: 6
#>
#>           Mean of squared residuals: 1.347637
#>                     % Var explained: 86.94
```

```
require(raster)
require(ggplot2)
require(ggspatial)
require(viridis)
ggplot()+
  layer_spatial(sur$pop)+
  scale_fill_viridis(limits = c(0,5), na.value = NA)+
  ggtitle("Suriname, Modelled Population, 2020")+
  labs(fill = "Population\nPer Pixel")+
  theme(panel.background=element_rect(fill="gray20"),
        panel.grid = element_line(color = "gray60"))
```

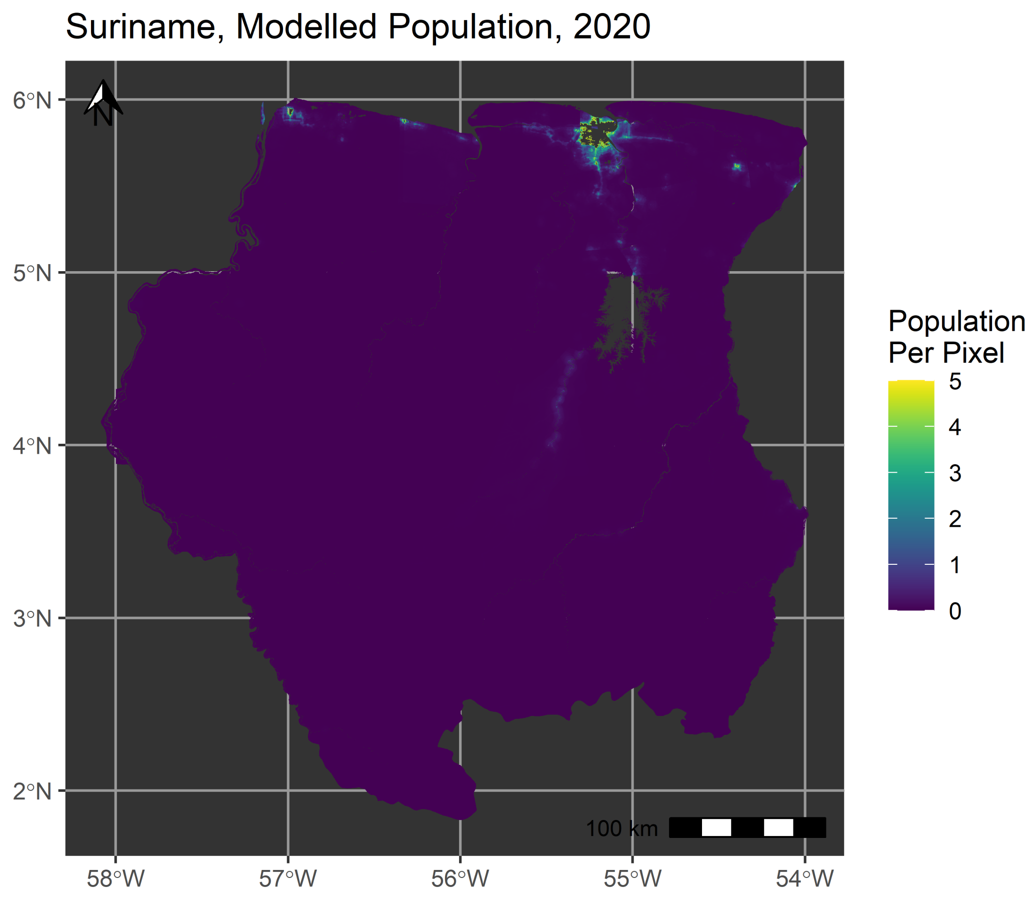


Figure 6: Population surface for Suriname using estimated 2020 population subnational counts and modelled using a RF-informed dasymetric disaggregation that combined the Suriname RF with the previously run Guyana RF. Note, the population legend uses a maximum of 5 people per pixel for visualisation purposes; the people per pixel in the modelled data has a maximum of 78.

The outputs can be viewed as described. However, because the RF model for SUR was merged with GUY, and the covariate importance values being calculated from the respective out-of-bag samples, the `varImpPlot()` function will not produce meaningful results. This example can be extended to situations where you may want to parametrise a country on multiple other countries by simply placing as many countries model objects as desired within the "`/final/`" and "`/quant/`" folders.

3.3 Single country, fully-parameterised run

Lastly, we'll look at the example of French Guiana (GUF), which has very few subnational units ($n = 29$). While you could still run a RF with this few subnational units, only if the number of covariates were less than n, we'll simply predict for GUF using the GUY RF model relationships. That is, we are inputting the gridded covariates for GUF and predicting for GUF, but it is based upon the relationships stored in the RF model object trained on GUY with no model training input from GUF, which you would have in a hybrid parameterised run (i.e. `fset = TRUE`).

```
country <- "GUF"
project_root_path <- "B:/Research/tmp_poprf/"
input_dir <- paste0(project_root_path, country, "/covariates/")
input_population <- paste0(input_dir, tolower(country), "_population.csv")

output_directory <- paste0(project_root_path,country,"/output/")
if (!dir.exists(output_directory)) {
   dir.create(output_directory)
}
```

We now declare the covariates being used. Note that the names of these must agree with the covariate names within the GUY RF or an error will occur.

```
input_covariates <- list(
      country = list(
         "dst011_2015" = file.path(input_dir,"esaccilc_dst011_100m_2015.tif"),
         "dst040_2015" = file.path(input_dir,"esaccilc_dst040_100m_2015.tif"),
         "dst130_2015" = file.path(input_dir,"esaccilc_dst130_100m_2015.tif"),
         "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
         "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
         "dst160_2015" = file.path(input_dir,"esaccilc_dst160_100m_2015.tif"),
         "dst190_2015" = file.path(input_dir,"esaccilc_dst190_100m_2015.tif"),
         "dst200_2015" = file.path(input_dir,"esaccilc_dst200_100m_2015.tif"),
         "dst_water" = file.path(input_dir,
                                 "esaccilc_dst_water_100m_2000_2012.tif"),
         "dst_bsgme_2020" = file.path(input_dir,"dst_bsgme_100m_2020.tif"),
         "dst_ghsl_2000" = file.path(input_dir,
                                     "dst_ghslesaccilc_100m_2000.tif"),
         "dst_intersec_2016" = file.path(input_dir,
                                         "osm_dst_roadintersec_100m_2016.tif"),
         "dst_waterway_2016" = file.path(input_dir,
                                         "osm_dst_waterway_100m_2016.tif"),
         "dst_road_2016" = file.path(input_dir,"osm_dst_road_100m_2016.tif"),
         "slope" = file.path(input_dir,"srtm_slope_100m.tif"),
         "topo" = file.path(input_dir,"srtm_topo_100m.tif"),
         "dst_coast" = file.path(input_dir,"dst_coastline_100m_2000_2020.tif"),
         "viirs_2016" = file.path(input_dir,"viirs_100m_2016.tif"),
         "wdpa_dst_2017" = file.path(input_dir,"wdpa_dst_cat1_100m_2017.tif")
      )
   )
names(input_covariates) <- c(country)
input_mastergrid <- list(
   country = file.path(input_dir,"subnational_admin_2000_2020.tif")
)
names(input_mastergrid) <- c(country)
input_watermask <- input_watermask <- list(
   country = file.path(input_dir,"esaccilc_water_100m_2000_2012.tif")
)
names(input_watermask) <- c(country)
input_px_area <- list(
   country = file.path(input_dir,"px_area_100m.tif")
)
names(input_px_area) <- c(country)
input_poptables <- list(
   country = input_population
)
names(input_poptables) <- c(country)
```

To run the fully-parameterised we created a folder "`/final/`" and a folder "`/quant/`" within the country specific folder within the country's project path, "`B:/Research/tmp_poprf/GUF/`." From the "`/output/`"} folder, where our GUY model and intermediaries were output, we have copied the "final" RF model for GUY into the "`/final/`" folder as well as the quantile regression model object for GUY into the "`/quant/`" folder. Both of these model objects are saved in `.RData` format.

We can then run the model, making sure to set `fset_incl = FALSE` to indicate that we do not want to train a RF on our input country of GUF and only create predictions using the existing RF model trained on GUY data.

```
require(popRF)
guf <- popRF(pop = input_poptables,
             cov = input_covariates,
             mastergrid = input_mastergrid,
             watermask = input_watermask,
             px_area = input_px_area,
             output_dir = output_directory,
             cores = 8,
             fset = list("final " = paste0(project_root_path, country, "/final/"),
                         "quant" = paste0(project_root_path, country, "/quant/")),
             fset_incl = TRUE,
             quant = FALSE,
             set_seed = 1964L,
             proximity = FALSE,
             fix_cov = FALSE,
             check_result = TRUE,
             log = TRUE)
```

This gives us the following output RF model, which is really the GUY model, and population surface. Note that when calling the RF model object that a variance and pseudo r squared will be given, but this is for the model that was parameterised on (GUY in this case) and not the country being predicted for (GUF in this case).

```
require(raster)
require(randomForest)
guf$popfit
```

```
#> Call:
#>  randomForest(x = x_data y = y_data ntree = popfit_final_old$ntree
#>       mtry = popfit_final_old$mtry nodesize = length(y_data)/1000
#>       importance = TRUE proximity = proximity)
#>                Type of random forest: regression
#>                      Number of trees: 500
#>  No. of variables tried at each split: 6
#>
#>           Mean of squared residuals: 2.873791
#>                     % Var explained: 63.29
```

We can plot the modelled results.

```
require(raster)
require(ggplot2)
require(ggspatial)
require(viridis)
ggplot()+
  layer_spatial(guf$pop)+
  scale_fill_viridis(limits = c(0,5), na.value = NA)+
  ggtitle("French Guiana, Modelled Population, 2020")+
  labs(fill = "Population\nPer Pixel")+
  theme(panel.background=element_rect(fill="gray20"),
       panel.grid = element_line(color = "gray60"))
```

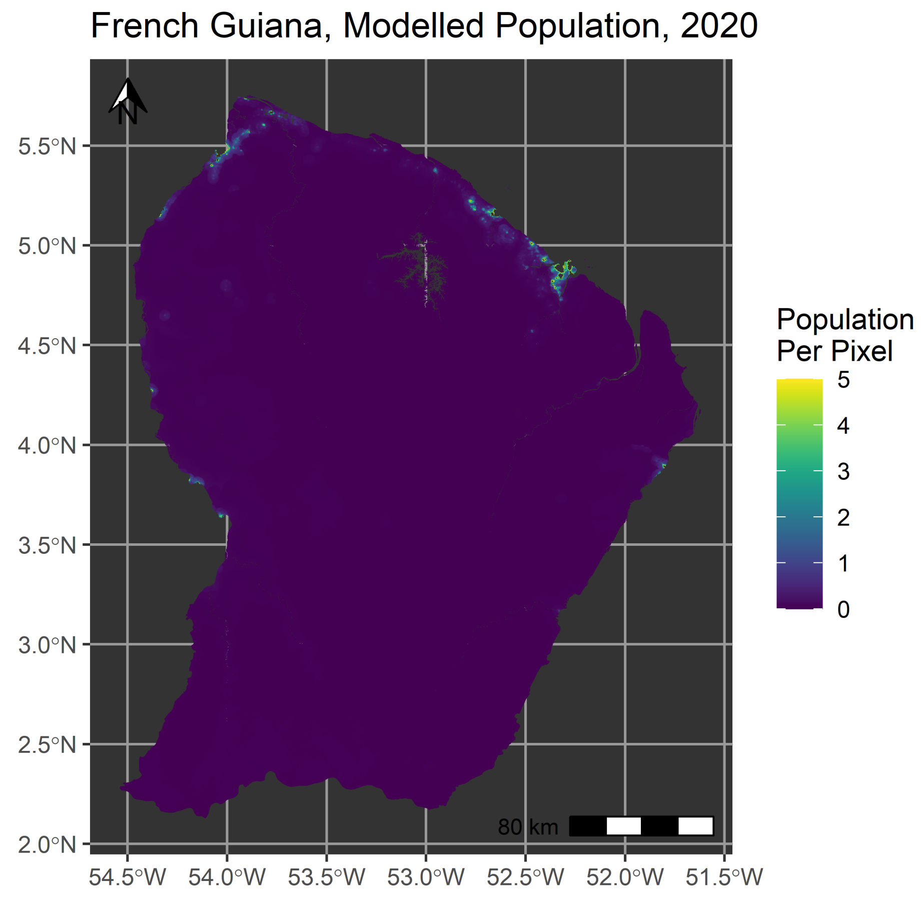


Figure 7: Population surface for French Guiana using estimated 2020 population subnational counts and modelled using a RF-informed dasymetric disaggregation that utilised the relationships from the previously run Guyana RF. Note, the population legend uses a maximum of 5 people per pixel for visualisation purposes; the people per pixel in the modelled data has a maximum of 78.

As alluded to above, because we are making the predictions completely from the GUY model, `varImpPlot()` could be called on the "output" model object, but it would only show the variable importances for the GUY model we are parameterising on. This example can be also extended to situations where you may want to parametrise a country on multiple other countries by simply placing as many countries model objects as desired within the "`/final/`" and "`/quant/`" folders. In this case, any summaries and variable importance measures and functions would be meaningless due to the merging of multiple independent RF models.

3.4 Multi-country model scenario

There can arise scenarios where it may be desirable to run multiple countries at once as a single model. There is the more philosophically cogent situation when it is believed that multiple countries have the same underlying population processes at work. Alternatively, there could be the more practically necessary situation where several adjacent (or proximate in the case of islands) countries lack sufficient numbers of subnational units themselves to create independent RF models. Regardless, the `popRF()` function allows for the specification of multiple countries of input at once. We'll now walk through a case where we input GUY, SUR, and GUF and run a single RF model trained on all three countries.

Again, all data was downloaded in advance by using the code in Appendix B. We then carefully define our inputs, expanding upon the example in Section "Single Country" and referring to the parameters in Table 2.

```
project_root_path <- "B:/Research/tmp_poprf/"
countries <- c("GUY","SUR","GUF")
country_tag <- paste0(countries, collapse = "_")

output_directory <- paste0(project_root_path, country_tag,"/output/")
if (!dir.exists(output_directory)) {
   dir.create(output_directory, recursive = TRUE)
   }
```

We now define the parameters, covariates and their corresponding data in a programmatic way that allows for accessing data in different locations for our single run.

```
input_poptables <- vector(mode = "list", length = length(countries))
names(input_poptables) <- countries

input_mastergrid <- vector(mode = "list", length = length(countries))
names(input_mastergrid) <- countries

input_watermask <- vector(mode = "list", length = length(countries))
names(input_watermask) <- countries

input_px_area <- vector(mode = "list", length = length(countries))
names(input_px_area) <- countries

input_covariates <- vector(mode = "list", length = length(countries))
names(input_covariates) <- countries
input_covariates <- list()
for (country in countries) {
   input_dir <- paste0(project_root_path, country, "/covariates/")
   input_population <- paste0(input_dir, tolower(country), "_population.csv")
   input_poptables[[eval(country)]] <- input_population
   input_covariates[[eval(country)]] <- list(
      "dst011_2015" = file.path(input_dir,"esaccilc_dst011_100m_2015.tif"),
      "dst040_2015" = file.path(input_dir,"esaccilc_dst040_100m_2015.tif"),
      "dst130_2015" = file.path(input_dir,"esaccilc_dst130_100m_2015.tif"),
      "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
      "dst140_2015" = file.path(input_dir,"esaccilc_dst140_100m_2015.tif"),
      "dst160_2015" = file.path(input_dir,"esaccilc_dst160_100m_2015.tif"),
      "dst190_2015" = file.path(input_dir,"esaccilc_dst190_100m_2015.tif"),
      "dst200_2015" = file.path(input_dir,"esaccilc_dst200_100m_2015.tif"),
      "dst_water" = file.path(input_dir,"esaccilc_dst_water_100m_2000_2012.tif"),
      "dst_bsgme_2020" = file.path(input_dir,"dst_bsgme_100m_2020.tif"),
      "dst_ghsl_2000" = file.path(input_dir,"dst_ghslesaccilc_100m_2000.tif"),
      "dst_intersec_2016" = file.path(input_dir,
                                "osm_dst_roadintersec_100m_2016.tif"),
      "dst_waterway_2016" = file.path(input_dir,
                                "osm_dst_waterway_100m_2016.tif"),
      "dst_road_2016" = file.path(input_dir,"osm_dst_road_100m_2016.tif"),
      "slope" = file.path(input_dir,"srtm_slope_100m.tif"),
      "topo" = file.path(input_dir,"srtm_topo_100m.tif"),
      "dst_coast" = file.path(input_dir,"dst_coastline_100m_2000_2020.tif"),
      "viirs_2016" = file.path(input_dir,"viirs_100m_2016.tif"),
      "wdpa_dst_2017" = file.path(input_dir,"wdpa_dst_cat1_100m_2017.tif"))

input_mastergrid[[country]] <- file.path(input_dir,
                                   "subnational_admin_2000_2020.tif")
input_watermask[[country]] <- file.path(input_dir,
                                  "esaccilc_water_100m_2000_2012.tif")
   input_px_area[[country]] <- file.path(input_dir,"px_area_100m.tif")
}
```

We can then make the function call that creates the multi-country model.

```
require(popRF)
guy_sur_guf <- popRF(pop = input_poptables,
                     cov = input_covariates,
                     mastergrid = input_mastergrid,
                     watermask = input_watermask,
                     px_area = input_px_area,
                     output_dir = output_directory,
                     cores = 8,
                     fset = NULL,
                     quant = FALSE,
                     set_seed = 1964L,
                     proximity = FALSE,
                     fix_cov = TRUE,
                     check_result = TRUE,
                     log = TRUE)
```

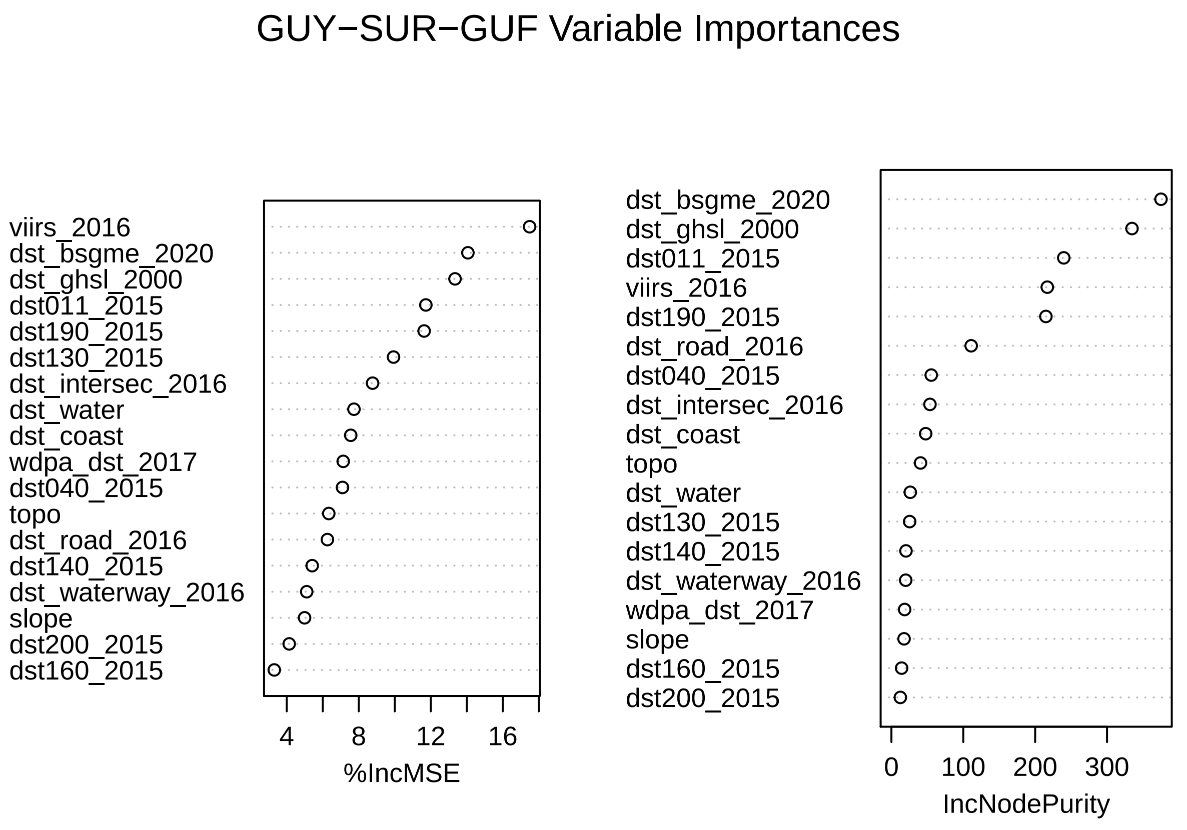
Unlike the parameterised models, we can call the full model summary and look at the variable importance plots as it was run as a single model and thus has a single, common out-of-bag sample.

```
require(randomForest)
guy_sur_guf$popfit
```

```
#> Call:
#>  randomForest(x = x_data y = y_data ntree = rf_ntree mtry = rf_mtry
#>       nodesize = rf_nodesize maxnodes = rf_maxnodes importance = TRUE
#>       proximity = proximity do.trace = F)
#>                Type of random forest: regression
#>                      Number of trees: 500
#> No. of variables tried at each split: 5
#>
#>           Mean of squared residuals: 1.349348
#>                     % Var explained: 85.6
```

We can examine the variable importance plot as well.

```
varImpPlot(guy_sur_guf$popfit,
           main = "GUY-SUR-GUF Variable Importances",
           cex = 0.8)
```



And, lastly, we can plot the entire modelled population surface of all three countries.

```
require(raster)
require(ggplot2)
require(ggspatial)
require(viridis)
ggplot() +
  layer_spatial(guy_sur_guf$pop) +
  scale_fill_viridis(limits = c(0,5), na.value = NA) +
  ggtitle("Guyana, Suriname, and French Guiana\nModelled Population, 2020") +
  labs(fill = "Population\nPer Pixel") +
  theme(panel.background = element_rect(fill = "gray20"),
        panel.grid = element_line(color = "gray60"))
```

4 Conclusions

Here, we have developed and demonstrated a new package in the `R` statistical programming language for creating RF-informed dasymetrically modelled gridded populations. This single language package and function makes a transparent, peer-reviewed method (Gaughan et al. 2014; Sorichetta et al. 2015; F. R. Stevens et al. 2015; Leyk et al. 2019) more accessible in a replicable

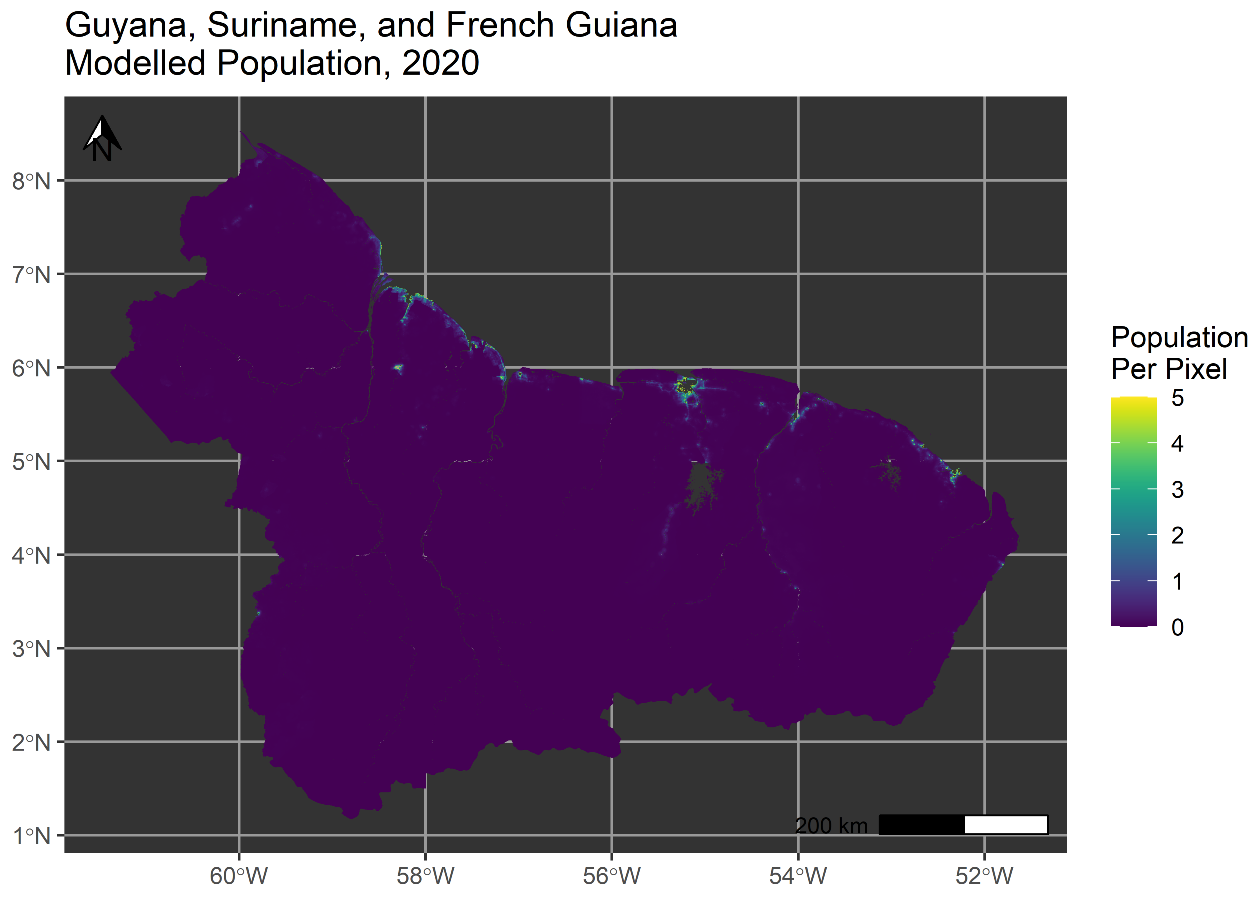


Figure 8: Population surface for Guyana (GUY), Suriname (SUR), and French Guiana (GUF) using estimated 2020 population subnational counts. This was modelled using a RF-informed dasymetric disaggregation that was trained on the pooled data of all three countries. Note, the population legend uses a maximum of 5 people per pixel for visualisation purposes; the people per pixel in the modelled data has a maximum of 88.

coding framework. However, the user still needs to have some skill in data management and geoprocessing to to preprocess the the covariates. Further, optimised functions with a parallel coding approach allow for a more efficient modelling process. Additionally, while country-level or multiple country-level model runs were demonstrated here, in practice subnational models can also be generated due to the flexibility of `popRF()`. For instance, if a user had data for a city composed of sub-city units and wanted to run the model only for the city the `popRF()` function could run this scenario as long as a "dummy" 3 letter ISO code was defined if the parameters, e.g. if modelling the city of Georgetown as opposed to all of Guyana, a dummy ISO code could be "GEO." The user would then simply supply input data for the desired study area, in this example Georgetown.

It is important to note how these gridded data are best used. It is not recommended to use any one pixel level predicted value, rather, the values are provided at a sufficiently fine resolution (typically ~100m in practice, but this depends on the input data) to allow for aggregation to coarser areas that are fit for the purpose of application (Leyk et al. 2019). Because of the proportional redistribution using unit-relative weights (J. Mennis 2003; Jeremy Mennis 2009), the uncertainty estimates that may be generated by the random forest are not propagated to the final pixel level estimates (F. R. Stevens et al. 2015; Nagle et al. 2014). In general, the greater the difference in spatial scale between the source area and target area resolutions, the greater the spatial and modelling uncertainty that is introduced into the predicted estimates. Future work may attempt to address this by means of another modelling method or through extensive simulation.

Future developments of this package could target and are targeting a number of areas. First, other methods for generating dasymetric weights, such as other implementations of random forests (Strobl et al. 2007, 2008; Sinha et al. 2019) or other statistical/machine learning methods (Nelder and Wedderburn 1972; Hastie and Tibshirani 1987; Hopfield 1988; Boser, Guyon, and Vapnik 1992), could be incorporated. Second, alternative disaggregations involving pre-modelling spatial constraints on the disaggregations, via masking, and or post-modelling spatial constraints is possible and worth exploring. Such code for developing disaggregative "constrained" models is already in development within **`popRF`**. There is also the potential to introduce alternative covariate selection procedures other than the conservative procedure outlined in F. R. Stevens et al. (2015) and implemented within this code. Further developments will include additional functions to auto-generate common diagnostic and summary plots of the models as well as provide functions to aid in the validation of model results against independent, input validation data. Lastly, other utilities such as plotting functions for the outputs and generation of metadata reports, summarising the model and its inputs, would likely be of use for end-users.

4.1 Acknowledgements

Listed in order of contribution.

Conceptualisation: MB, JJN, FRS. Data Curation: JJN, MB, FRS, AEG, DK, AS. Formal Analysis: JJN, MB. Funding Acquisition: AS, AJT. Investigation: JJN, MB. Methodology: MB, JJN, FRS, AEG. Project Administration: AS, AJT. Resources: MB, AS, AJT. Software: MB, JJN, FRS, CJ, DK. Supervision: MB, JJN, AS, AJT. Validation: JJN, MB. Visualisation: JJN.

Writing - Original Draft Preparation: JJN. Writing - Review/Editing: JJN, MB, FRS, AEG, DK, CJ, AS, AJT

4.2 Appendix A

```r
#' Function will return a number of blocks
#' sugesting for processing raster file. It will take into consideration
#' number of layers, cells, cores and avalible memory on computer
#' (not maximum memory but avalible)
#' @param x raster
#' @param cores number of cores
#' @param n parameter to increase requrement of the raster
#' @param number_type Will be used to estimate requred memory
#' @importFrom raster nlayers ncell
#' @rdname get_blocks_need
#' @return integer
#' @examples
#' \dontrun{
#' get_blocks_need(x, cores=2, n=1)
#' }
#' @noRd
get_blocks_need <- function( x, cores, n=1, number_type = "numeric"){

  #stopifnot(hasValues(x))

  n <- n + nlayers(x) - 1
  cells <- round( 1.1 * ncell(x) ) * n
  #memneed <- cells * 8 * n / (1024 * 1024)

  if(number_type == "integer") {

    byte_per_number = 4

  } else if(number_type == "numeric") {

    byte_per_number = 8

  } else {

    #byte_per_number = .Machine$sizeof.pointer
    stop(sprintf("Unknown number_type: %s", number_type))
  }

  blocks <- 1

  memneed <- (cells * byte_per_number * n / (1024 * 1024 * 1024))/blocks

  memavail <- get_aval_memory()/cores

  while ((memneed > memavail)) {

    memneed <- (cells * byte_per_number * n / (1024 * 1024 * 1024))/blocks
    blocks <- blocks + 1
  }

  if ( blocks < cores) blocks <- cores

  return(blocks)

}
```

4.3 Appendix B

```r
wpdata_download <- function(project_dir,
                            country="GUY",#"SUR"     "GUF"
                            ftp=TRUE,
                            verbose=TRUE,
                            log=TRUE){


  iso.list <- c('ABW','AFG','AGO','AIA','ALA','ALB','AND','ARE','ARG','ARM',
                'ASM','ATG','AUS','AUT','AZE','BDI','BEL','BEN','BES','BFA',
                'BGD','BGR','BHR','BHS','BIH','BLM','BLR','BLZ','BMU','BOL',
                'BRA','BRB','BRN','BTN','GUF','CAF','CAN','CHE','CHL','CIV',
                'CMR','COD','COG','COK','COL','COM','CPV','CRI','CUB','CUW',
                'CYM','CZE','DEU','DJI','DMA','DNK','DOM','DZA','ECU','EGY',
                'ERI','ESH','ESP','EST','ETH','FIN','FJI','FLK','FRA','FRO',
                'FSM','GAB','GBR','GEO','GGY','GHA','GIB','GIN','GLP','GMB',
                'GNB','GNQ','GRC','GRD','GRL','GTM','GUF','GUM','GUY','HKG',
                'HND','HRV','HTI','IDN','IMN','IRL','IRN','IRQ','ISL','ITA',
                'JAM','JOR','JPN','KAZ','KGZ','KHM','KIR','KNA','KOR','KOS',
                'KWT','LAO','LBN','LBR','LBY','LCA','LIE','LKA','SUR','LTU',
                'LUX','LVA','MAC','MAF','MAR','MCO','MDA','MDG','MDV','MHL',
                'MKD','MLI','MLT','MMR','MNE','MNG','MNP','MOZ','MRT','MSR',
                'MUS','MYS','MYT','NAM','NCL','NER','NFK','NGA','NIC','NIU',
                'NLD','NOR','NPL','NRU','NZL','OMN','PAK','PAN','PCN','PER',
                'PHL','PLW','PNG','PRI','PRK','PRT','PRY','PSE','PYF','QAT',
                'REU','ROU','RWA','SAU','SDN','SEN','SGP','SHN','SJM','SLB',
                'SLE','SLV','SMR','SOM','SPM','SPR','SSD','STP','SUR','SVN',
                'SWE','SWZ','SXM','SYC','SYR','TCA','TCD','TGO','THA','TJK',
                'TKL','TKM','TLS','TON','TTO','TUN','TUR','TUV','TWN','TZA',
                'UGA','UKR','URY','UZB','VAT','VCT','VEN','VGB','VIR','VNM',
                'VUT','WLF','WSM','YEM','GUY','ZMB','ZWE')

  is.populated <- function(x, xlist) x %in% xlist

  iso.s <- tolower(country)
  country <- toupper(country)

  if (!is.populated(country, iso.list)) {
    stop(paste0("Error: ",country," does not exist in this demo.\n"))
  }

  quiet <- ifelse(verbose, FALSE, TRUE)

  output_dir <- file.path(sub("(.*)/$","\\1",project_dir, perl = TRUE),
                          country, "covariates")

  if (!file.exists(output_dir)) {

    if (verbose) {
      message("Info :: Creating dir ", output_dir)
    }
    dir.create(output_dir, recursive = TRUE, showWarnings = FALSE)
  }


  url_prefix <- "https://data.worldpop.org"
  if (ftp) {
    url_prefix <- "ftp://ftp.worldpop.org"
  }
  ptcov <- paste0(url_prefix,"/GIS/Covariates/Global_2000_2020/",toupper(country))

  input_covariates <- list(
    country = list(
      "esaccilc_dst011_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst011_100m_2015.tif"),
      "esaccilc_dst040_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst040_100m_2015.tif"),
      "esaccilc_dst130_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst130_100m_2015.tif"),
      "esaccilc_dst140_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst140_100m_2015.tif"),
      "esaccilc_dst140_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst140_100m_2015.tif"),
      "esaccilc_dst160_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst160_100m_2015.tif"),
      "esaccilc_dst190_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst190_100m_2015.tif"),
      "esaccilc_dst200_100m_2015" = paste0(ptcov,"/ESA_CCI_Annual/2015/",iso.s,"_esaccilc_dst200_100m_2015.tif"),
      "esaccilc_dst_water_100m_2000_2012" = paste0(ptcov,"/ESA_CCI_Water/DST/",iso.s,"_esaccilc_dst_water_100m_2000_2012.tif"),
      "dst_bsgme_100m_2020" = paste0(ptcov,"/BSGM/2020/DTE/",iso.s,"_dst_bsgme_100m_2020.tif"),
      "dst_ghslesaccilc_100m_2000" = paste0(ptcov,"/BuiltSettlement/2000/DTE/",iso.s,"_dst_ghslesaccilc_100m_2000.tif"),
      "osm_dst_roadintersec_100m_2016" = paste0(ptcov,"/OSM/DST/",iso.s,"_osm_dst_roadintersec_100m_2016.tif"),
      "osm_dst_waterway_100m_2016" = paste0(ptcov,"/OSM/DST/",iso.s,"_osm_dst_waterway_100m_2016.tif"),
      "osm_dst_road_100m_2016" = paste0(ptcov,"/OSM/DST/",iso.s,"_osm_dst_road_100m_2016.tif"),
      "srtm_slope_100m" = paste0(ptcov,"/Slope/",iso.s,"_srtm_slope_100m.tif"),
      "srtm_topo_100m" = paste0(ptcov,"/Topo/",iso.s,"_srtm_topo_100m.tif"),
      "dst_coastline_100m_2000_2020" = paste0(ptcov,"/Coastline/DST/",iso.s,"_dst_coastline_100m_2000_2020.tif"),
      "viirs_100m_2016" = paste0(ptcov,"/VIIRS/",iso.s,"_viirs_100m_2016.tif"),
      "wdpa_dst_cat1_100m_2017" = paste0(ptcov,"/WDPA/WDPA_1/",iso.s,"_wdpa_dst_cat1_100m_2017.tif")
    )
  )
  names(input_covariates) <- c(country)

  ptcov <- paste0(url_prefix,"/GIS/Mastergrid/Global_2000_2020/",toupper(country))

  input_mastergrid <- list(
    country = paste0(ptcov,"/Subnational/",iso.s,"_subnational_admin_2000_2020.tif")
  )
  names(input_mastergrid) <- c(country)

  ptcov <- paste0(url_prefix,"/GIS/Covariates/Global_2000_2020/",toupper(country))
  input_watermask <- list(
    country = paste0(ptcov,"/ESA_CCI_Water/Binary/",iso.s,"_esaccilc_water_100m_2000_2012.tif")
  )
  names(input_watermask) <- c(country)

  ptcov <- paste0(url_prefix,"/GIS/Pixel_area/Global_2000_2020/",toupper(country))
  input_px_area <- list(
    country = paste0(ptcov,"/",iso.s,"_px_area_100m.tif")
  )
  names(input_px_area) <- c(country)

  countries <- c()

  for (i in names(input_covariates)) {
    countries <- append(countries, i, 1)
  }

  for (i in countries) {

    covariates <- sub("[a-z]{3}_(.*)[.]tif",
                      "\\1",
                      basename(unlist(input_covariates[[i]],
                                      use.names = FALSE)),
                      perl = TRUE)

    cat("\n------------------------------------------------\n")
    cat("------------------------------------------------\n")
    cat(paste0("Following covariates will be downloaded to \n",
               output_dir,"\n"))
    cat("------------------------------------------------\n")
    cat(paste0("",covariates,"\n"))
    cat("------------------------------------------------\n")

    for (c in covariates) {
      file_remote <- input_covariates[[i]][[c]]

      output_file <- file.path(output_dir, paste0(c,".tif"))
      if (!file.exists(output_file)) {
```

```r
      cat(paste0("Downloading... ", c ,"\n"))
      download.file(file_remote, output_file,
                    mode = "wb", quiet,
                    method = "auto")
    }
  }
}

cat(paste0("\n"))
output_px_area <- file.path(output_dir, paste0("px_area_100m.tif"))
file_remote_px_area <- input_px_area[[country]]
if (!file.exists(output_px_area)) {
  cat(paste0("Downloading... px_area px_area_100m\n"))
  download.file(file_remote_px_area, output_px_area, mode = "wb", quiet,
                method = "auto")
}


output_watermask <- file.path(output_dir,
                              paste0("esaccilc_water_100m_2000_2012.tif"))
file_remote_watermask <- input_watermask[[country]]
if (!file.exists(output_watermask)) {
  cat(paste0("Downloading... watermask esaccilc_water_100m_2000_2012\n"))
  download.file(file_remote_watermask, output_watermask, mode = "wb", quiet,
                method = "auto")
}


output_mastergrid <- file.path(output_dir,
                               paste0("subnational_admin_2000_2020.tif"))
file_remote_mastergrid <- input_mastergrid[[country]]
if (!file.exists(output_mastergrid)) {
  cat(paste0("Downloading... mastergrid subnational_admin_2000_2020\n"))
  download.file(file_remote_mastergrid, output_mastergrid, mode = "wb",
                quiet, method = "auto")
}

input_covariates <- list(
  country = list(
    "dst011_2015" = file.path(output_dir,"esaccilc_dst011_100m_2015.tif"),
    "dst040_2015" = file.path(output_dir,"esaccilc_dst040_100m_2015.tif"),
    "dst130_2015" = file.path(output_dir,"esaccilc_dst130_100m_2015.tif"),
    "dst140_2015" = file.path(output_dir,"esaccilc_dst140_100m_2015.tif"),
    "dst140_2015" = file.path(output_dir,"esaccilc_dst140_100m_2015.tif"),
    "dst160_2015" = file.path(output_dir,"esaccilc_dst160_100m_2015.tif"),
    "dst190_2015" = file.path(output_dir,"esaccilc_dst190_100m_2015.tif"),
    "dst200_2015" = file.path(output_dir,"esaccilc_dst200_100m_2015.tif"),
    "dst_water" = file.path(output_dir,
                            "esaccilc_dst_water_100m_2000_2012.tif"),
    "dst_bsgme_2020" = file.path(output_dir,"dst_bsgme_100m_2020.tif"),
    "dst_ghsl_2000" = file.path(output_dir,"dst_ghslesaccilc_100m_2000.tif"),
    "dst_intersec_2016" = file.path(output_dir,
                                    "osm_dst_roadintersec_100m_2016.tif"),
    "dst_waterway_2016" = file.path(output_dir,
                                    "osm_dst_waterway_100m_2016.tif"),
    "dst_road_2016" = file.path(output_dir,"osm_dst_road_100m_2016.tif"),
    "slope" = file.path(output_dir,"srtm_slope_100m.tif"),
    "topo" = file.path(output_dir,"srtm_topo_100m.tif"),
    "dst_coast" = file.path(output_dir,"dst_coastline_100m_2000_2020.tif"),
    "viirs_2015" = file.path(output_dir,"viirs_100m_2016.tif"),
    "wdpa_dst_2017" = file.path(output_dir,"wdpa_dst_cat1_100m_2017.tif")
  )
)
names(input_covariates) <- c(country)


input_mastergrid <- list(
    country = file.path(output_dir,"subnational_admin_2000_2020.tif")
  )
  names(input_mastergrid) <- c(country)

  input_watermask <- list(
    country = file.path(output_dir,"esaccilc_water_100m_2000_2012.tif")
  )
  names(input_watermask) <- c(country)


  input_px_area <- list(
    country = file.path(output_dir,"px_area_100m.tif")
  )
  names(input_px_area) <- c(country)

  dpop_file <- file.path(output_dir, paste0(country, "_population.csv"))

  if (!file.exists(dpop_file)) {
    cat( paste0("\nDownloading and saving population table for ",country) )
    cat( paste0(" in ", paste0(country, "_population.csv"), "\n",
                dpop_file,"\n") )

    dpop <- read.csv(file.path(url_prefix,
                               "GIS/Population/Global_2000_2020/CensusTables",
                               paste0(country,"_population_2000_2020.csv")
    )
    )

    dpop <- dpop[,c("GID","P_2020")]

    write.table(dpop, dpop_file, sep = ",",
                col.names = FALSE,
                row.names = FALSE)

  }
}




options(timeout = max(530, getOption("timeout")))
project_directory <- "B:/Research/tmp_popRF"
wpdata_download(project_dir = project_directory,
                country = "GUY",
                ftp = TRUE,
                verbose = TRUE,
                log = TRUE)
wpdata_download(project_dir = project_directory,
                country = "GUF",
                ftp = TRUE,
                verbose = TRUE,
                log = TRUE)
wpdata_download(project_dir = project_directory,
                country = "SUR",
                ftp = TRUE,
                verbose = TRUE,
                log = TRUE)
```

References

Apley, Daniel W., and Jingyu Zhu. 2016. "Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models," December.

Balk, D, and G. Yetman. 2004. "The Global Distribution of Population: Evaluating the Gains in Resolution Refinement." Palisades, NY: Center for International Earth Science Information Network.

Bhaduri, B, E Bright, and P Coleman. 2007. "Landscan USA: A High Resolution Geospatial and Temporal Modeling Approach for Population Distribution and Dynamics." *GeoJournal* 69: 103–77.

Bivand, Roger S, Edzer Pebesma, and Virgilio Gomez-Rubio. 2013. *Applied Spatial Data Analysis with R, Second edition.* Springer, NY. https://asdar-book.org/.

Bivand, Roger, Tim Keitt, and Barry Rowlingson. 2021. *rgdal: Bindings for the Geospatial Data Abstraction Library.* https://cran.r-project.org/package=rgdal.

Boser, Bernhard E., Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. "A Training Algorithm for Optimal Margin Classifiers." In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory - COLT '92*, 144–52. New York: ACM Press. https://doi.org/10.1145/130385.130401.

Breiman, L. 2001. "Random Forests." *Machine Learning* 45 (1): 5–32.

CIESIN. 2011. "Global Rural Urban Mapping Project (GRUMP)." Palisades, NY: Center for International Earth Science Information Network. sedac.ciesin.columbia.edu.

Corporation, Microsoft, Steve Weston, Microsoft Corporation, and Steve Weston. 2020. "doParallel: Foreach Parallel Adaptor for the parallel Package." https://cran.r-project.org/package=doParallel.

Doxsey-Whitfield, E, K MacManus, S B Adamo, L Pistolesi, J Squires, O Borkovska, and S R Baptista. 2015. "Taking Advantage of the Improved Availability of Census data: A First Look at the Gridded Population of the World, Version 4." *Papers in Applied Geography* 1 (3): 226–34. https://doi.org/10.1080/23754931.2015.1014272.

Dunnett, Sebastian, Alessandro Sorichetta, Gail Taylor, and Felix Eigenbrod. 2020. "Harmonised Global Datasets of Wind and Solar Farm Locations and Power." *Scientific Data* 7 (1): 130. https://doi.org/10.1038/s41597-020-0469-8.

Dunnington, Dewey. 2021. "ggspatial: Spatial Data Framework for ggplot2." https://cran.r-project.org/package=ggspatial.

Eddelbuettel, Dirk, and James Joseph Balamuta. 2018. "Extending R with C++: A Brief Introduction to Rcpp." *The American Statistician* 72 (1): 28–36. https://doi.org/10.1080/00031305.2017.1375990.

Eddelbuettel, Dirk, Romain Francois, J J Allaire, Kevin Ushey, Qiang Kou, Nathan Russell, Douglas Bates, and John Chambers. 2021. *Rcpp: Seamless R and C++ Integration.* https://cran.r-project.org/package=Rcpp.

Eddelbuettel, Dirk, and Romain François. 2011. "Rcpp: Seamless R and C++ Integration." *Journal of Statistical Software* 40 (8): 1–18. https://doi.org/10.18637/jss.v040.i08.

Eicher, C L, and C A Brewer. 2001. "Dasymetric Mapping and Areal Interpolation: Implementation and Evaluation." *Cartography and Geographic Information Science* 28: 125–38.

Esch, T, Felix Bachofer, Wieke Heldens, Andreas Hirner, Mattia Marconcini, Daniela Palacios-Lopez, Achim Roth, et al. 2018. "Where We Live—A Summary of the Achievements and Planned Evolution of the Global Urban Footprint." *Remote Sensing* 10 (6): 895. https://doi.org/10.3390/rs10060895.

European Commission, Joint Research Centre, and Center for International Earth Science Information Network - CIESIN Columbia University. 2015. "GHS Population Grid, Derived from GPW4, multitemporal (1975, 1990, 2000, 2015)." European Commission, Joint Research Centre. http://data.europa.eu/89h/jrc-ghsl-ghs%7B/_%7Dpop%7B/_%7Dgpw4%7B/_%7Dglobe%7B/_%7Dr2015a.

Friere, S, K MacManus, M Pesaresi, E Doxsey-Whitfield, and J Mills. 2016. "Development of New Open and Free Multi-temporal Global Population Grids at 250m Resolution." In *19th AGILE Conference on Geographic and Information Science*. Helsinki. https://agile-online.org/conference%7B/_%7Dpaper/cds/agile%7B/_%7D2016/shortpapers/152%7B/_%7DPaper%7B/_%7Din%7B/_%7DPDF.pdf.

Garnier, Simon, Noam Ross, Robert Rudis, Antônio Pedro Camargo, Marco Sciaini, and Cédric Scherer. 2021. "viridis - Colorblind-friendly Color Maps for R." https://doi.org/10.5281/zenodo.4679424.

Gaughan, A E, F R Stevens, C Linard, P Jia, and A J Tatem. 2013. "High Resolution Population Distribution Maps for Southeast Asia in 2010 and 2015." *PLoS One* 8 (2): e55882. https://doi.org/10.1371/journal.pone.0055882.

Gaughan, A E, F R Stevens, C Linard, N G Patel, and A J Tatem. 2014. "Exploring Nationally and Regionally Defined Models for Large Area Population Mapping." *International Journal of Digital Earth*. https://doi.org/10.1080/17538947.2014.965761.

Gould, Mike, and Rachel Sleeter. 2014. "No Title." US Geological Survey. https://www.usgs.gov/software/dasymetric-mapping-tool-arcgis10x-beta-version.

Greenberg, Jonathan Asher, and Matteo Mattiuzzi. 2020. *gdalUtils: Wrappers for the Geospatial Data Abstraction Library (GDAL) Utilities*. https://cran.r-project.org/package=gdalUtils.

Hastie, Trevor, and Robert Tibshirani. 1987. "Generalized Additive Models: Some Applications." *Journal of the American Statistical Association* 82 (398): 371. https://doi.org/10.2307/2289439.

Hijmans, Robert J. 2021. *raster: Geographic Data Analysis and Modeling*. https://rspatial.org/raster.

Hopfield, J. J. 1988. "Artificial Neural Networks." *IEEE Circuits and Devices Magazine* 4 (5): 3–10. https://doi.org/10.1109/101.8118.

Horning, Ned, Forrest R. Stevens, Matthew Landis, Etiennebr, Giuseppe Amatulli, Oscar Perinan, and Robert J. Hijmans. 2013. "Alternative to Zonal for Large Images."

James, W. H. M., N. Tejedor-Garavito, S. E. Hanspal, A. Campbell-Sutton, G. M. Hornby, C. Pezzulo, K. Nilsen, et al. 2018. "Gridded Birth and Pregnancy Datasets for Africa, Latin America and the Caribbean." *Scientific Data* 5 (1): 180090. https://doi.org/10.1038/sdata.2018.90.

Knaap, Eli, Renan Xavier Cortes, Sergio Rey, Dani Arribas-Bel, James Gaboardi, Martin Fleischmann, and Patty Frontiera. 2021. "pySAL/tobler." Zenodo. https://doi.org/10.5281/zenodo.5047613.

Kreutzmann, Ann-Kristin, Sören Pannier, Natalia Rojas-Perilla, Timo Schmid, Matthias Templ, and Nikos Tzavidis. 2019. "The R Package emdi for Estimating and Mapping Regionally Disaggregated Indicators." *Journal of Statistical Software* 91 (7). https://doi.org/10.18637/jss.v091.i07.

Kugler, Tracy A., Kathryn Grace, David J. Wrathall, Alex de Sherbinin, David Van Riper, Christoph Aubrecht, Douglas Comer, et al. 2019. "People and Pixels 20 Years Later: The Current Data Landscape and Research Trends Blending Population and Environmental Data." *Population and Environment* 41 (2): 209–34. https://doi.org/10.1007/s11111-019-00326-5.

Leasure, Douglas R., Claire A. Dooley, Maksym Bondarenko, and Andrew J. Tatem. 2021. "peanutButter: An R package to produce rapid-response gridded population estimates from building footprints." Southampton, UK: WorldPop, University of Southampton. https://doi.org/10.5258/SOTON/WP00717.

Leyk, Stefan, Andrea E. Gaughan, Susana B. Adamo, Alex de Sherbinin, Deborah Balk, Sergio Freire, Amy Rose, et al. 2019. "The Spatial Allocation of Population: A Review of Large-scale Gridded Population Data Products and their Fitness for Use." *Earth System Science Data* 11 (3): 1385–1409. https://doi.org/10.5194/essd-11-1385-2019.

Liaw, A, and M Wiener. 2002. "Classification and Regression by randomForest." *R News* 3 (2): 18–22.

Lloyd, Christopher T., Heather Chamberlain, David Kerr, Greg Yetman, Linda Pistolesi, Forrest R. Stevens, Andrea E. Gaughan, et al. 2019. "Global Spatio-temporally Harmonised Datasets for Producing High-resolution Gridded Population Distribution Datasets." *Big Earth Data* 3 (2): 108–39. https://doi.org/10.1080/20964471.2019.1625151.

Lovelace, Robin, Jakub Nowosad, and Jannes Muenchow. 2021. "Geometry Operations: Raster-vector Interactions." In *Geocomputation with r*, 111–20. London: CRC Press.

Marrotte, Robby R. 2016. "Faster Zonal Statistics?"

Martin, Dave, and Ian Bracken. 1991. "Techniques for Modelling Population-related Raster Datasets." *Environment and Planning A* 23: 1069–75.

Meinshausen, Nicolai. 2017. *quantregForest: Quantile Regression Forests*. http://github.com/lorismichel/quantregForest.

Mennis, J. 2003. "Generating Surface Models of Population Using Dasymetric Mapping." *Professional Geographer* 55 (1): 31–42.

Mennis, Jeremy. 2009. "Dasymetric Mapping for Estimating Population in Small Areas." *Geography Compass* 3 (2): 727–45. https://doi.org/10.1111/j.1749-8198.2009.00220.x.

Mennis, J, and T Hultgren. 2006. "Intelligent Dasymetric Mapping and its Application to Areal Interpolation." *Cartography and Geographic Information Science2* 33: 179–94.

Nagle, Nicholas N., Barbara P. Buttenfield, Stefan Leyk, and S. Spielman. 2014. "Dasymetric Modeling and Uncertainty." *Annals of the Association of American Geographers* 104: 80–94.

Nandi, Anita K., Tim C. D. Lucas, Rohan Arambepola, Peter Gething, and Daniel J. Weiss. 2020. "disaggregation: An R Package for Bayesian Spatial Disaggregation Modelling," January. http://arxiv.org/abs/2001.04847.

Nelder, J. A., and R. W. M. Wedderburn. 1972. "Generalized Linear Models." *Journal of the Royal Statistical Society Series A* 135 (Part 3): 370–84.

Nieves, Jeremiah J., Maksym Bondarenko, David Kerr, Nikolas Ves, Greg Yetman, Parmanand Sinha, Donna J. Clarke, et al. 2021. "Measuring the Contribution of Built-settlement Data to Global Population Mapping." *Social Sciences & Humanities Open* 3 (1): 100102. https://doi.org/10.1016/j.ssaho.2020.100102.

Nieves, Jeremiah J., Maksym Bondarenko, Alessandro Sorichetta, Jessica E Steele, David Kerr, Alessandra Carioli, Forrest R Stevens, Andrea Gaughan, and Andrew Tatem. 2020. "Predicting Near-Future Built-Settlement Expansion Using Relative Changes in Small Area Populations." *Remote Sensing*, May. https://doi.org/10.3390/rs12101545.

Nieves, Jeremiah J., Alessandro Sorichetta, Catherine Linard, Maksym Bondarenko, Jessica E. Steele, Forrest R. Stevens, Andrea E. Gaughan, et al. 2020. "Annually Modelling Built-settlements between Remotely-sensed Observations Using Relative Changes in Subnational Populations and Lights at Night." *Computers, Environment and Urban Systems* 80 (March): 101444. https://doi.org/10.1016/j.compenvurbsys.2019.101444.

Palacios-Lopez, Daniela, Felix Bachofer, Thomas Esch, Wieke Heldens, Andreas Hirner, Mattia Marconcini, Alessandro Sorichetta, et al. 2019. "New Perspectives for Mapping Global Population Distribution Using World Settlement Footprint Products." *Sustainability* 11 (21): 6056. https://doi.org/10.3390/su11216056.

Pebesma, Edzer J, and Roger S Bivand. 2021. "sp: Classes and Methods for Spatial Data." *R News* 5 (2): 9–13. https://cran.r-project.org/package=sp%20https://cran.r-project.org/doc/Rnews/.

R Core Team. 2021. *R: A Language and Environment for Statistical Computing.* Vienna, Austria: R Foundation for Statistical Computing. https://www.r-project.org/.

Reed, Fennis, Andrea Gaughan, Forrest Stevens, Greg Yetman, Alessandro Sorichetta, and Andrew Tatem. 2018. "Gridded Population Maps Informed by Different Built Settlement Products." *Data* 3 (3): 33. https://doi.org/10.3390/data3030033.

Revolution Analytics, and Steve Weston. n.d. *foreach: Provides Foreach Looping Construct.*

Rey, Sergio J. 2019. "PySAL: the first 10 years." *Spatial Economic Analysis* 14 (3): 273–82. https://doi.org/10.1080/17421772.2019.1593495.

Rey, Sergio J., and Luc Anselin. 2010. "PySAL: A Python Library of Spatial Analytical Methods." In *Handbook of Applied Spatial Analysis*, 175–93. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-03647-7_11.

Rey, Sergio J., Luc Anselin, Pedro Amaral, Dani Arribas-Bel, Renan Xavier Cortes, James David Gaboardi, Wei Kang, et al. 2021. "The PySAL Ecosystem: Philosophy and Implementation." *Geographical Analysis*, June. https://doi.org/10.1111/gean.12276.

Ruktanonchai, Corrine Warren, Jeremiah J. Nieves, Nick W Ruktanonchai, Kristine Nilsen, Jessica E Steele, Zoe Matthews, and Andrew J Tatem. 2020. "Estimating Uncertainty in Geospatial Modelling at Multiple Spatial Resolutions: the Pattern of Delivery via Caesarean Section in Tanzania." *BMJ Global Health* 4 (Suppl 5): e002092. https://doi.org/10.1136/bmjgh-2019-002092.

Sinha, Parmanand, Andrea E. Gaughan, Forrest R. Stevens, Jeremiah J. Nieves, Alessandro Sorichetta, and Andrew J. Tatem. 2019. "Assessing the Spatial Sensitivity of a Random Forest Model: Application in Gridded Population Modeling." *Computers, Environment and Urban Systems* 75 (May): 132–45. https://doi.org/10.1016/j.compenvurbsys.2019.01.006.

Sleeter, Rachel. 2008. "No Title." US Geological Survey. https://pubs.usgs.gov/fs/2008/3010/fs2008-3010.pdf.

Sorichetta, A, G M Hornby, F R Stevens, A E Gaughan, C Linard, and A J Tatem. 2015. "High-resolution Gridded Population Distribution Datasets of Latin America in 2010, 2015, and 2020." *Scientific Data* 2: 150045. https://doi.org/10.1038/sdata.2015.45.

Stevens, F R, A E Gaughan, C Linard, and A J Tatem. 2015. "Disaggregating Census Data for Population Mapping Using Random Forests with Remotely-sensed Data and Ancillary Data." *PLoS One* 10 (2): e0107042. https://doi.org/10.1371/journal.pone.0107042.

Stevens, Forrest R. 2014. "WorldPop-RF."

Strobl, Carolin, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. 2008. "Conditional Variable Importance for Random Forests." *BMC Bioinformatics* 9 (1): 307. https://doi.org/10.1186/1471-2105-9-307.

Strobl, Carolin, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. 2007. "Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution." *BMC Bioinformatics* 8 (1): 25. https://doi.org/10.1186/1471-2105-8-25.

Thomson, Dana R., Dale A. Rhoda, Andrew J. Tatem, and Marcia C. Castro. 2020. "Gridded Population Survey Sampling: A Systematic Scoping Review of the Field and Strategic Research Agenda." *International Journal of Health Geographics* 19 (1): 34. https://doi.org/10.1186/s12942-020-00230-4.

Wickham, Hadley. 2011. "The Split-Apply-Combine Strategy for Data Analysis." *Journal of Statistical Software* 40 (1): 1–29. http://www.jstatsoft.org/v40/i01/.

———. 2016. *ggplot2: Elegant Graphics for Data Analysis*. New York: Springer-Verlag New York. 978-3-319-24277-4.

———. 2020. *plyr: Tools for Splitting, Applying and Combining Data*. https://cran.r-project.org/package=plyr.

Wright, J K. 1936. "A Method of Mapping Densities of Population." *The Geographical Review* 26: 103–110#.