Code ▾

Population modelling for census support

Tutorial 3: How to model small-scale spatial variations?

Edith Darin and Douglas Leasure

Compiled on 21/08/2021

WorldPop

Code

Introduction

Tutorial 2 explored how to model large-scale spatial patterns, that is how population density differs per large grouping such as by region or settlement type. We integrated those large-scale variations in a Bayesian framework by using a hierarchical random intercept model.

Tutorial 3 aims at integrating small-scale variations of population density that are due to local context. High-resolution covariates are great to inform about the local context as shown in Figure 1.

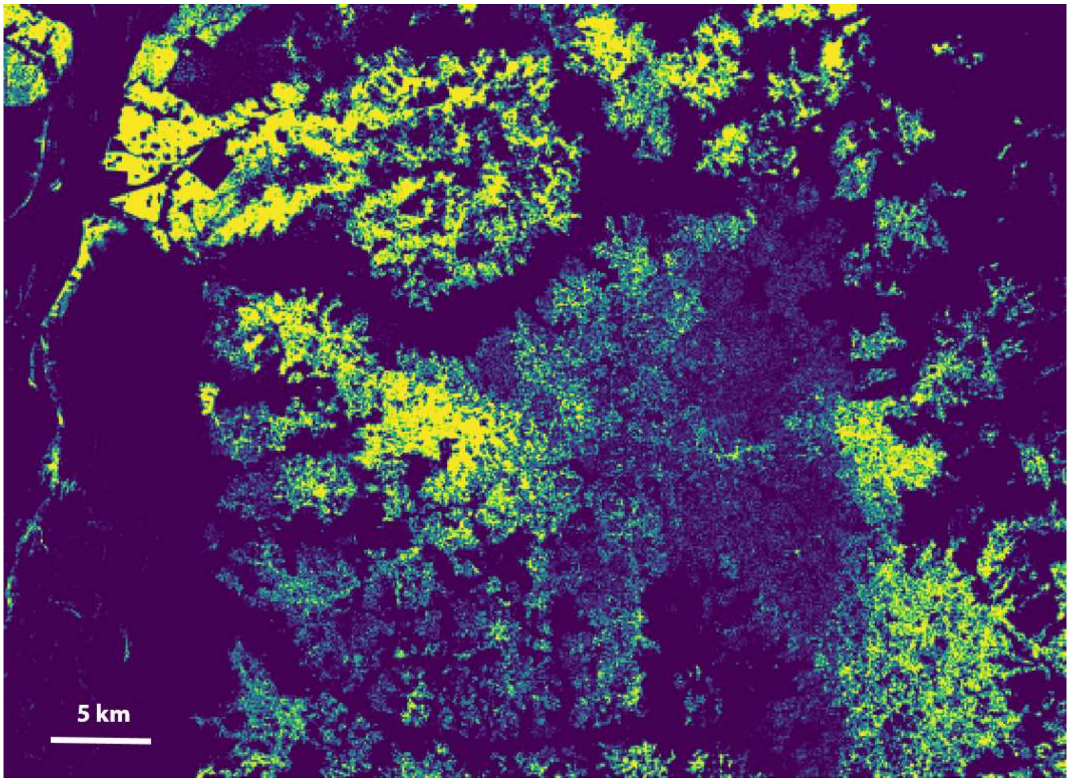


Figure 1: Example of a covariate (residential area) around Onitscha, Nigeria

Adding high-resolution covariates helps to improve the model fit as well as guides the population predictions in unsampled areas.

For data to be used as covariates in the model, they should be:

- correlated with differences in population density
- measured consistently and completely across the study space
- accurately mapped as high-resolution geospatial layers

High-resolution covariates that are suitable for the modelling are typically **spatial covariates with national coverage and consistent data collection**. While individual or household-level information, such as collected during surveys, is useful for understanding differences in demographic characteristics, that type of information is difficult to use in the bottom-up approaches because it only comes from a sample of households. The primary objective of the population model is to make a spatially-complete prediction.

Goals

1. Understand requirements for covariates to be included in the model
2. Get an overview of covariates used in WorldPop bottom-up population models
3. Familiarize with covariates processing for population model
4. Integrate a linear regression to refine the definition of the median of the Lognormal
5. Add a random slope model
6. Understand the difference between hierarchical modelling and random effect modelling

Supporting readings

To include a spatial covariate in the modelling and to use it as a support for prediction, we use gridded datasets, known as *rasters*. You would need thus to have basic GIS knowledge about raster and vector file management. It does not need to be in `R`, it can be in QGIS, ArcGIS, Python or any GIS software of your choice.

The purpose of this tutorial is **not** spatial data processing. We will just mention processing techniques that are required to prepare covariates data for population modelling.

However, here are some R resources on spatial manipulation:

- The bible (https://keen-swartz-3146c4.netlify.app/) by Pebesma and Bivand on the `sf` (vector data) and `stars` (raster data) R packages with excellent overview of the different spatial manipulations
- A hands-on introduction (https://geoscripting-wur.github.io/) to `sf` and `raster` package by the University of Wageningen
- A focus on raster manipulation (https://rspatial.org/) with `raster` and `terra` by Hijmans

Formal modelling

We will work from model 3 in tutorial 2 which is based on a Poisson distribution to model population count and on a Lognormal distribution with a hierarchical random intercept by settlement type and region to model population density .

Because the median of the population density is only defined with a random intercept, it results in 5 (number of settlement type) x 11 (number of region) options for the population density estimates. To add small-scale variations we **refine the median of the Lognormal with a regression model** that integrates the covariates.

More formally, let's define X a matrix of size *number of observations* x *number of covariates* that contains the covariates values for each study site and β a vector of size the number of covariates. Based on Equation 3 in tutorial 2 (and removing the prior distribution for $\alpha_{t,r}$ for sake of readibility), we define μ the median of the Lognormal distribution as follows:

$$
\begin{gathered}
population \sim Poisson(pop\_density * settled\_area) \\
pop\_density \sim Lognormal(\mu,\, \sigma) \\
\mu = \alpha_{t,r} + \beta X \\
\\
\beta \sim Normal(0, 10)
\end{gathered}
\tag{1}
$$

Note that the prior for β are identical normal distribution for each covariate with mean 0 and standard deviation 10 to avoid introducing any bias.

Figure 2 shows the updated dependent relationships of our model when integrating covariates.

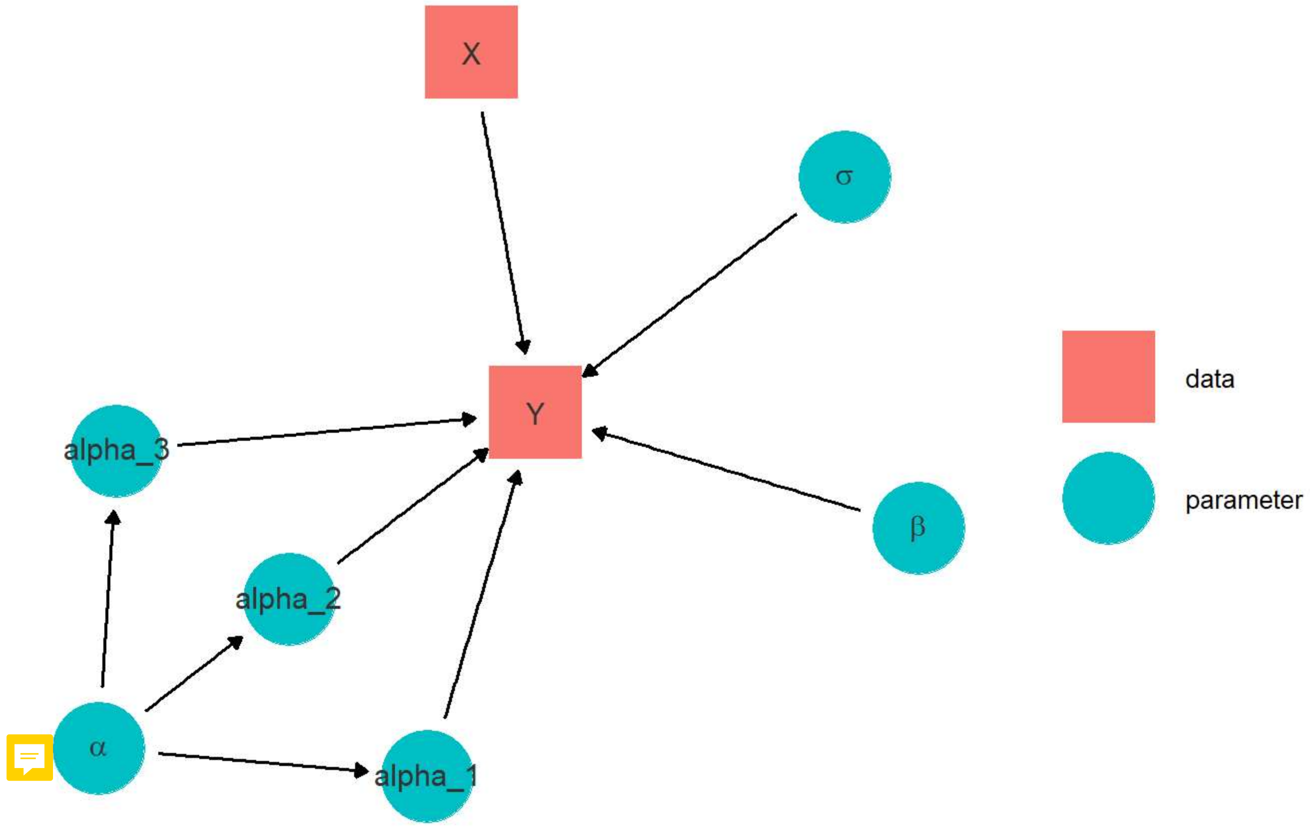


Figure 2: Dependancy graph for model 1 of tutorial 3

Review of the covariates used in WorldPop

To date, five bottom-up population models have been produced at WorldPop:

- WorldPop. 2019. Bottom-up gridded population estimates for **Nigeria, version 1.2**. WorldPop, University of Southampton. https://dx.doi.org/10.5258/SOTON/WP00655 (https://dx.doi.org/10.5258/SOTON/WP00655).
- WorldPop (School of Geography and Environmental Science, University of Southampton). 2020. Bottom-up gridded population estimates for **Zambia, version 1.0**. https://dx.doi.org/10.5258/SOTON/WP00662 (https://dx.doi.org/10.5258/SOTON/WP00662)
- WorldPop and Institut National de la Statistique et de la Démographie du Burkina Faso. 2021. Census-based gridded population estimates for **Burkina Faso (2019), version 1.0**. WorldPop, University of Southampton. https://dx.doi.org/10.5258/SOTON/WP00687 (https://dx.doi.org/10.5258/SOTON/WP00687).
- Boo G, Darin E, Leasure DR, Dooley CA, Chamberlain HR, Lazar AN, Tatem AJ. 2020. Modelled gridded population estimates for the Kinshasa, Kongo-Central, Kwango, Kwilu, and Mai-Ndombe provinces in the **Democratic Republic of the Congo, version 2.0**. WorldPop, University of Southampton. https://dx.doi.org/10.5258/SOTON/WP00669 (https://dx.doi.org/10.5258/SOTON/WP00669)

- WorldPop. 2020. Bottom-up gridded population estimates for the Kinshasa, Kongo-Central, Kwango, Kwilu, and Mai-Ndombe provinces in the **Democratic Republic of the Congo, version 1.0**. WorldPop, University of Southampton. https://dx.doi.org/10.5258/SOTON/WP00658 (https://dx.doi.org/10.5258/SOTON/WP00658)

In addition to that, two models are currently being updated: **Nigeria version 2.0** and **Democratic Republic of the Congo v3.0**.

Those six models encompass a large range of covariates, that are specific to each country. Table 1 offers an overview of the final covariates set selected for each model.

Code

Table 1: Review of covariates used in WorldPop bottom-up population models

Type	Covariate	Model	Source
Gridded population	UN-adjusted projected gridded estimates	Burkina Faso v1.0	WorldPop
Gridded population	Projected gridded estimates	Nigeria v1.2	WorldPop
Gridded population	Mean UN-adjusted projected gridded estimates within a 2km radius	Democratic Republic of Congo v1.0	WorldPop
Infrastucture	Fricition surface	Burkina Faso v1.0	Access to the Cities project
Infrastucture	Distance to secondary roads	Burkina Faso v1.0	National Geographical Office
Infrastucture	Household size	Nigeria v1.2, Nigeria v2.0	Demographic and Health Survey
Infrastucture	Residential roads density	Democratic Republic of Congo v1.0	OpenStreetMap
Infrastucture	Travel time to cities	Democratic Republic of Congo v1.0	Malaria Atlas Project
Infrastucture	Tertiary-sector activities density	Democratic Republic of Congo v1.0	OpenStreetMap
Natural feature	Distance to temporary rivers	Burkina Faso v1.0	National Geographical Office
Natural feature	Monthly variability of dry matter productivity	Democratic Republic of Congo v3.0	Copernicus
Natural feature	Monthly variability of surface air temperature	Democratic Republic of Congo v3.0	Copernicus
Natural feature	Land surface ‘roughness’ from Synthetic Aperture Radard VH	Nigeria v2.0	Sentinel-1
Natural feature	Land surface ‘roughness’ from Synthetic Aperture Radard VV	Nigeria v2.0	Sentinel-1
Settlement	Mean building count within a 5km radius	Burkina Faso v1.0	Ecopia & Maxar
Settlement	Mean building area within a 1km radius	Democratic Republic of Congo v2.0	Ecopia & Maxar
Settlement	Mean distance to nearest building within a 1km radius	Democratic Republic of Congo v2.0	Ecopia & Maxar
Settlement	Mean building count within a 1km radius	Democratic Republic of Congo v2.0	Ecopia & Maxar
Settlement	Mean building area	Zambia v1.0	Ecopia & Maxar
Settlement	Building density	Zambia v1.0	Ecopia & Maxar
Settlement	Coefficient of variation of building area	Zambia v1.0	Ecopia & Maxar
Settlement	Sum residential area within a 1km radius	Nigeria v1.2	Oak Ridge National Laboratory

Type	Covariate	Model	Source
Settlement	Sum nonresidential area within a 1km radius	Nigeria v1.2	Oak Ridge National Laboratory
Settlement	School density within a 1km radius	Nigeria v1.2	Oak Ridge National Laboratory
Settlement	Mean building perimeter	Democratic Republic of Congo v3.0	Ecopia & Maxar
Settlement	Compactness of building	Democratic Republic of Congo v3.0	Ecopia & Maxar

The covariates selected can be broadly classified as describing four main drivers of local population density variation:

1. *previous population spatial distribution* through WorldPop top-down population disaggregation (WorldPop Research Group et al. 2018)
2. *infrastructure* through OpenStreetMap (OpenStreetMap contributors 2018), country-specific resources (for example Institut Géographique du Burkina Faso (2015)]) or modelled resources (for example travel time to cities (Weiss et al. 2018))
3. *natural features* through remote sensing product such as radar data (Sentinel-1) or dry matter productivity (Copernicus)
4. *settlement shape* through the morphology of building fooptrints (Ecopia.AI and Maxar Technologies 2019) or settled area (Oak Ridge National Laboratory 2018)

Other covariates sources that were considered (but not selected in the final models) were:

- Conflict locations from the Armed Conflict Location and Event Data Project (https://acleddata.com/data-export-tool/)

- Climatic variables from the Climatic Research Unit at the university of Anglia (https://catalogue.ceda.ac.uk/uuid/10d3e3640f004c578403419aac167d82)

- Active mining concessions from the IPIS group (https://ipisresearch.be/publication/mining-concessions-dr-congo/)

- Land cover classification from the European Spatial Agency (https://climate.esa.int/en/esa-climate/esa-cci/)

- Global forest change from the University of Maryland (https://climate.esa.int/en/esa-climate/esa-cci/)

- Elevation and slope from WorldDEM (https://www.airbus.com/newsroom/press-releases/en/2018/10/WorldDEM-now-entirely-edited-and-available-via-streaming.html)

- Fossil fuel emissions from the Open-source Data Inventory for Anthropogenic CO2 Project (https://www.odiac.org/index.html)

To build a model, we first gather all covariates that can be related to our specific population data. It can reach up to 900+. We then use geospatial analysis techniques to obtain **gridded version of the covariates with identical spatial resolution, alignment and extent**. It involves resampling and clipping for covariates provided as raster files or for covariates provided as vector files computing count, density, distance to nearest features or even interpolation techniques.

Covariates engineering

Further covariates engineering steps can help extracting even more information from the gathered covariates.

1. Log-transformation

Considering the logarithm of the covariates can help handling extreme values.

2. Focal statistics

Focal statistics consist in summarising the covariate in a moving window around each grid cell. As seen in Table 1, we used different window sizes (1km, 2km or 5km) and summary statistics (mean or coefficient of variation). It provides contextual information around the grid cells.

3. Standardisation

Scaling the covariate (that is substrastring the mean and dividing by the standard deviation) can help enhancing meaningful local variations from the mean. The scaling can even be refined by computing the mean and the standard deviation by region, such that local variations are representative of the region.

A note on covariate selection

After engineering the gathered covariates, we might end up with 1000+ potential covariates.

To select the best one for prediction purposes, we generally used two methods:

- pairwise correlation and scatter plot with the population density at study site level
- univariate model, testing each covariate successively

Covariates in Nigeria, v1.2

We focused in the remaining parts of the tutorial on the data we downloaded from Leasure et al. (2020) which corresponds to the Nigeria model v1.2.

Overview of the covariates

Six covariates were used in Nigeria v1.2 model:

- `x_1` : gridded population estimates from WorldPop Global
- `x_2` : school densities within a 1km radius
- `x_3` : household sizes by interpolating Demographic Health Survey results from 2013
- `x_4` : settled area within a 1km radius
- `x_5` : residential area in a 1km radius
- `x_6` : nonresidential settled area within a 1km radius

Covariate `x_4` was scaled based on its mean and standard deviation nationally, whereas covariates `x_5` and `x_6` were scaled based on their mean and standard deviation within a 50-km radius. Leasure et at scaled `x_5` and `x_6` in this way because wethey suspected that neighborhood types may not be directly comparable across regions (especially northern versus southern Nigeria). This scaling also reduced correlation with `x_4` .

They scaled the WorldPop Global estimates (`x_1`) based on their mean and standard deviation nationally. They averaged these values among pixels within each microcensus study site. They treated this covariate as an indicator of relative population densities based on geospatial covariates that were in the random forest model.

Covariate `x_2` was scaled using its mean and standard deviation within a 50km radius. They scaled this covariate within a 50km moving window because what constitutes a “high density” of schools varies by region and this distinction was lost when the covariate was scaled nationally. This also helped to control for possible differences in school mapping effort in different regions.

They scaled `x_3` based on its mean and standard deviation nationally. One key reason for including this covariate was to account for a strong north–south gradient in household sizes, with significantly more people per household in northern Nigeria than in southern Nigeria.

Data preparation

To integrate the covariates in the model, we build first a dataset with the average of the covariate values for each study site using zonal statistics.

Note that this constitutes a change in support: we might want to check if the range of covariates values at study site level is representative of the covariate values at grid cell level.

Figure 3 shows the relation between the covariates and the population density at study site level. We see that household size (`x_3`) is strongly positively correlated with population density. The negative value are due to the scaling method adopted. On the opposite, nonresidential settled area (`x_6`) is negatively correlated which is expected: the more the surroundings are nonresidential the lower the population density.

Code

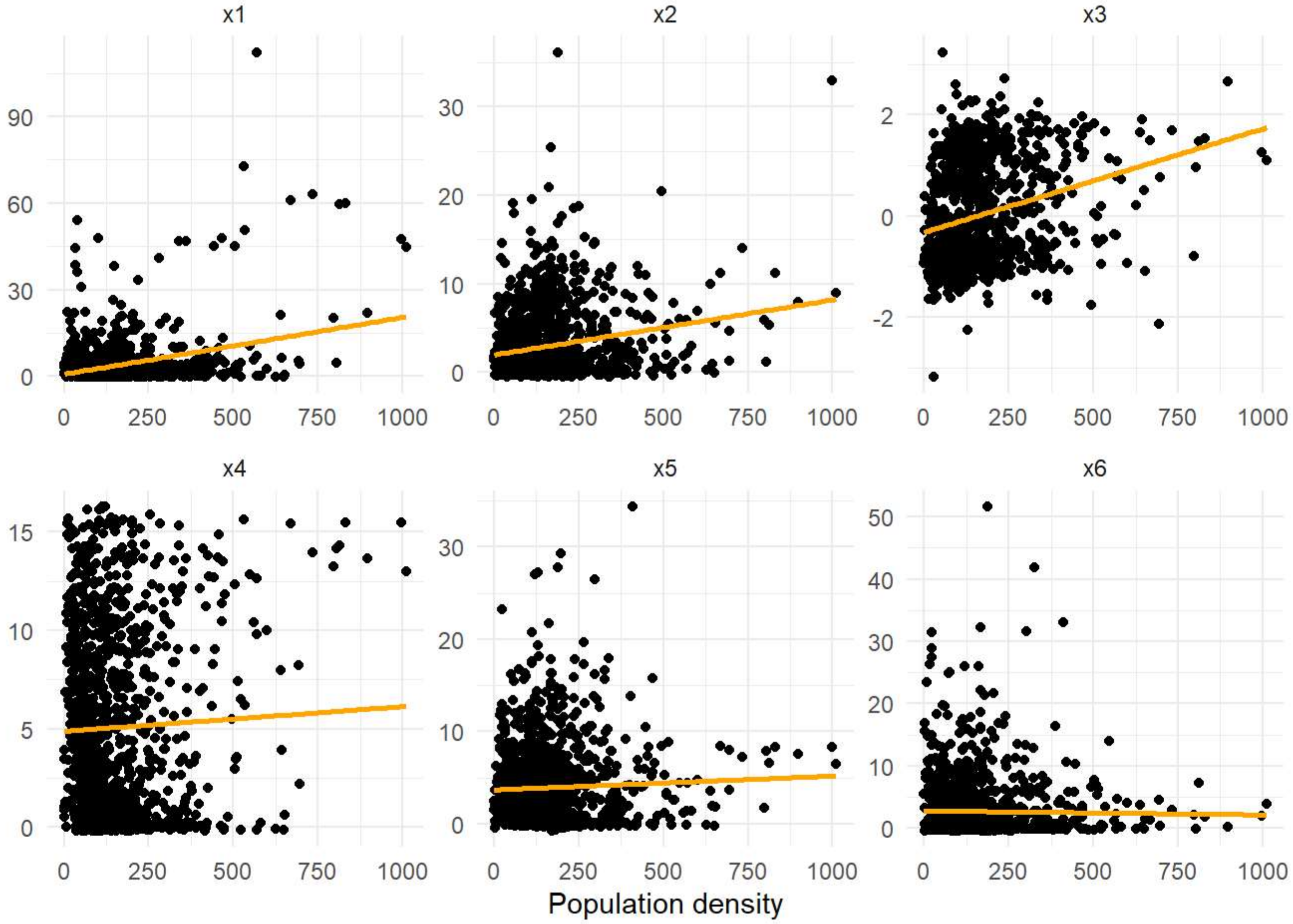


Figure 3: Scatterplot of covariates values vs population density for each study site

Before implementing the model in `stan`, we **uniformly scale the covariates at study site level**, such that the β_k have the same scale. We first compute the scaling coefficients (mean and standard deviation) for each covariate:

Hide

```
covariatesScaling <- function(var){
  mean_var <- mean(var)
  sd_var <- sd(var)
  return(
    data.frame(
      'cov_mean'= mean_var,
      'cov_sd' = sd_var
    )
  )
}

covs <- data %>%
  select(starts_with('x'))

scale_factor <- bind_rows(apply(covs, 2, covariatesScaling))
scale_factor$cov <- colnames(covs)

scale_factor %>% select(cov, cov_mean, cov_sd) %>% kbl %>%  kable_minimal()
```

cov	cov_mean	cov_sd
x1	3.913963	8.071879
x2	2.950790	3.970139
x3	0.000000	1.000000
x4	5.081461	4.748466
x5	3.941279	4.377713
x6	2.729769	5.028868

We then apply the scaling coefficient to the covariates:

Hide

```
covs_scaled <-  covs %>%
  mutate(cluster_id = 1:nrow(covs)) %>%
  pivot_longer(-cluster_id,names_to = 'cov') %>%
  left_join(scale_factor, by="cov") %>%
  mutate(value= (value-cov_mean)/cov_sd ) %>%
  select(-cov_mean, -cov_sd) %>%
  pivot_wider(names_from = cov, values_from = value, id_cols = cluster_id) %>%
  select(-cluster_id)
```

We store the scaled covariates and the scaling coefficients for the prediction stage (in Tutorial 4).

Hide

```
write_csv(covs_scaled, 'tutorials/data/covs_scaled.csv')
write_csv(scale_factor, 'tutorials/data/scale_factor.csv')
```

Model implementation

Equation (1) is implemented in `stan` as follows:

Hide

```
// Model 1: Independent alpha by settlement type

data{
  ...
  // slope
  int<lower=0> ncov; // number of covariates
  matrix[n, ncov] cov; // covariates
}
parameters{
  ...
  // slope
  row_vector[ncov] beta;
}
transformed parameters{
  ...
  for(idx in 1:n){
    pop_density_mean[idx] = alpha_t_r[type[idx], region[idx]] + sum( cov[idx,] .* be
ta );
  }
}
model{
  ...
  //slope
  beta ~ normal(0,10);
}
generated quantities{
  ...
   for(idx in 1:n){
    density_hat[idx] = lognormal_rng( alpha_t_r[type[idx], region[idx]] + sum(cov[id
x,] .* beta), sigma );
   }
}
```

Note the two new data type, `matrix` for the covariates values and `row_vector` for β. This is due to the way we have coded the covariates: in a for loop that runs through each study site observations, such that `cov[idx,]` is a (column)-vector and requires a `row_vector` as multiplier.

We keep the same set up for the `stan` Markov chains:

Hide

```
# mcmc settings
chains <- 4
warmup <- 500
iter <- 500
seed <- 1789
```

And we add the covariates to `stan` input data:

Hide

```
# prepare data for stan
stan_data_model1 <- list(
  population = data$N,
  n = nrow(data),
  area = data$A,
  type = data$type,
  ntype= n_distinct(data$type),
  region = data$region,
  nregion = n_distinct(data$region),
  seed=seed,
  cov = covs_scaled,
  ncov = ncol(covs_scaled)
  )
```

We add `beta` as parameter to monitor and run the model.

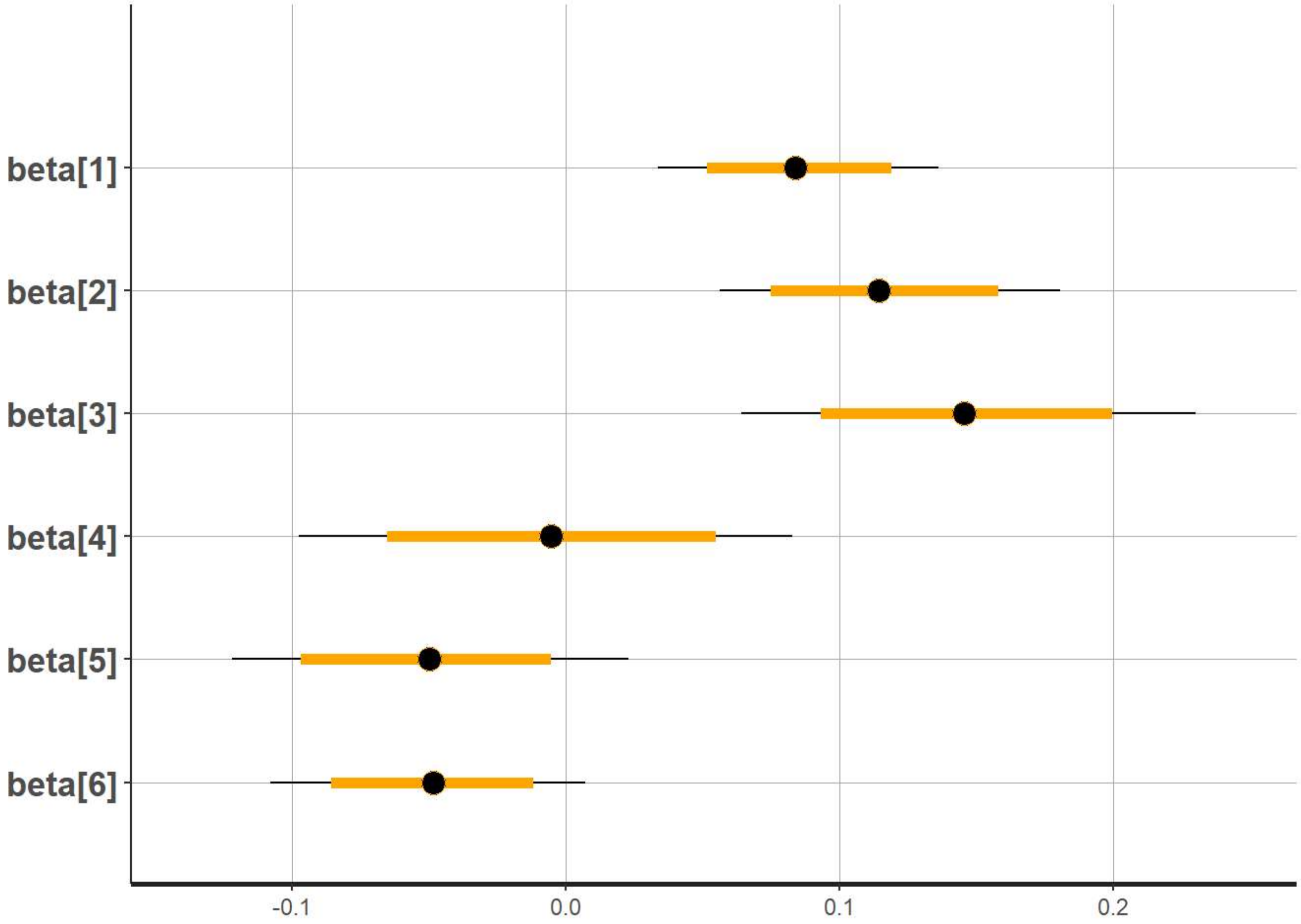
Hide

```
pars <- c('alpha','sigma','beta','alpha_t', 'population_hat',  'density_hat')

# mcmc
fit1 <- rstan::stan(file = file.path('tutorials/tutorial3/tutorial3_model1.stan'),
                   data = stan_data_model1,
                   iter = warmup + iter,
                   chains = chains,
                   warmup = warmup,
                   pars = pars,
                   seed = seed)
```

No convergence issue. We can plot the $\hat{\beta}_k$. The sign and magnitude of the covariate effects are inline with the correlation shown in Figure 3.

Hide

```
stan_plot(fit1, pars='beta', fill_color='orange')
```



The next question is: How does integrating covariates improve the model?

We run Tutorial 2 model 3 to compare:

Hide

```
pars <- pars[!grepl('beta', pars)]
fit0 <- rstan::stan(file = file.path('tutorials/tutorial2/tutorial2_model3.stan'),
                   data = stan_data_model1,
                   iter = warmup + iter,
                   chains = chains,
                   warmup = warmup,
                   pars = pars,
                   seed = seed)
```

We can then compute the predictions for every study site and compare the goodness-of-fit when adding the covariates.

Code

Table 2: Goodness-of-metrics comparison with and without covariates

Model	Bias	Inaccuracy	Imprecision	Mean Confidence Interval Size
With covariates	33.56579	193.6548	275.4327	1251.533
Without covariates	35.43037	200.0768	284.2697	1272.677

We see an improvement on every goodness-of-fit metrics.

A random slope

In tutorial 2 we have seen that a model fitting a unique α for all the observations could be improved by splitting the observations into groupings that would share a similar pattern of population density.

The idea is similar with β: **some covariates effects might vary by settlement type**. Example: `x_4` , the sum of settled area within a 1km radius might, might have a greater predictive power in rural areas than in urban areas. The difference in correlation between covariate and population density by settlement is highlighted in Figure 4.

Code

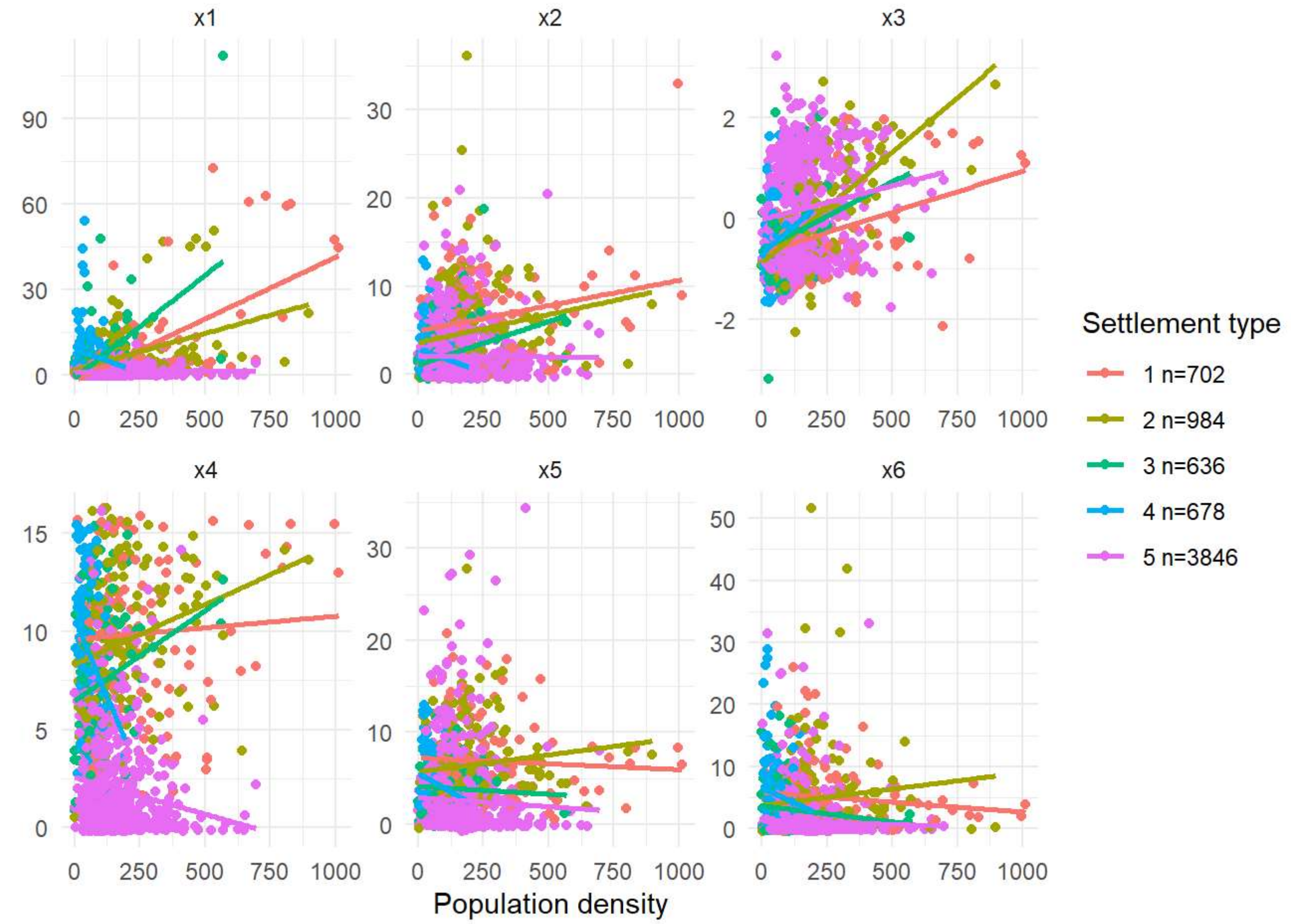


Figure 4: Scatterplot of covariates vs population density by settlement type

Modelling β_k by settlement type is called a **random slope model**.

> **Question**: Do we want to model the β_k hierarchically?

▶ Click for the solution

Formally, a random slope model is written as follows:

$$population \backsim Poisson(pop\_density * settled\_area)$$

$$pop\_density \backsim Lognormal(\mu,\ \sigma)$$

$$\mu = \alpha_{t,r} + \beta_t^{random} X^{random} + \beta^{fixed} X^{fixed}$$

$$\beta_t^{random} \backsim Normal(0, 10)$$

$$\beta^{fixed} \backsim Normal(0, 10)$$

The difference can be seen in the indexing: β^{random} is indexed by t. Similarly as in the no-pooling framework, we set the priors to be independent Normal(0,10).

The `stan` implementation is as follows:

Hide

```
// Model 1: Independent alpha by settlement type

data{
  ...
    // fixed slope
  int<lower=0> ncov_fixed; // number of covariates -1
  matrix[n, ncov_fixed] cov_fixed; // covariates
  // random slope
  vector[n] cov_random;
}
parameters{
  ...
  // slope
  row_vector[ncov_fixed] beta_fixed;
  vector[ntype] beta_random;
}
transformed parameters{
  ...
  vector[n] beta;

  for(idx in 1:n){
    beta[idx] = sum( cov_fixed[idx,] .* beta_fixed) + cov_random[idx] * beta_random[type[idx]];
    pop_density_mean[idx] = alpha_t_r[type[idx], region[idx]] + beta[idx];
  }
}
model{
  ...
  //slope
  beta_fixed ~ normal(0,10);
  beta_random ~ normal(0,10);
}
generated quantities{
  ...
 vector[n] beta_hat;

  for(idx in 1:n){
    beta_hat[idx] = sum( cov_fixed[idx,] .* beta_fixed) + cov_random[idx] * beta_random[type[idx]];
    density_hat[idx] = lognormal_rng( alpha_t_r[type[idx], region[idx]] + beta_hat[idx], sigma );
  ...
}
```

Note that we wrote the code to model only **one** random covariate.

To run the model, we distinguish in the input data between the covariates that are fixed and the one that is random. We choose `x_4` , the sum of settled area within a 1km radius, to be modelled with a random effect.

Note that this setting allows to test the model with different covariate candidates for the random effect.

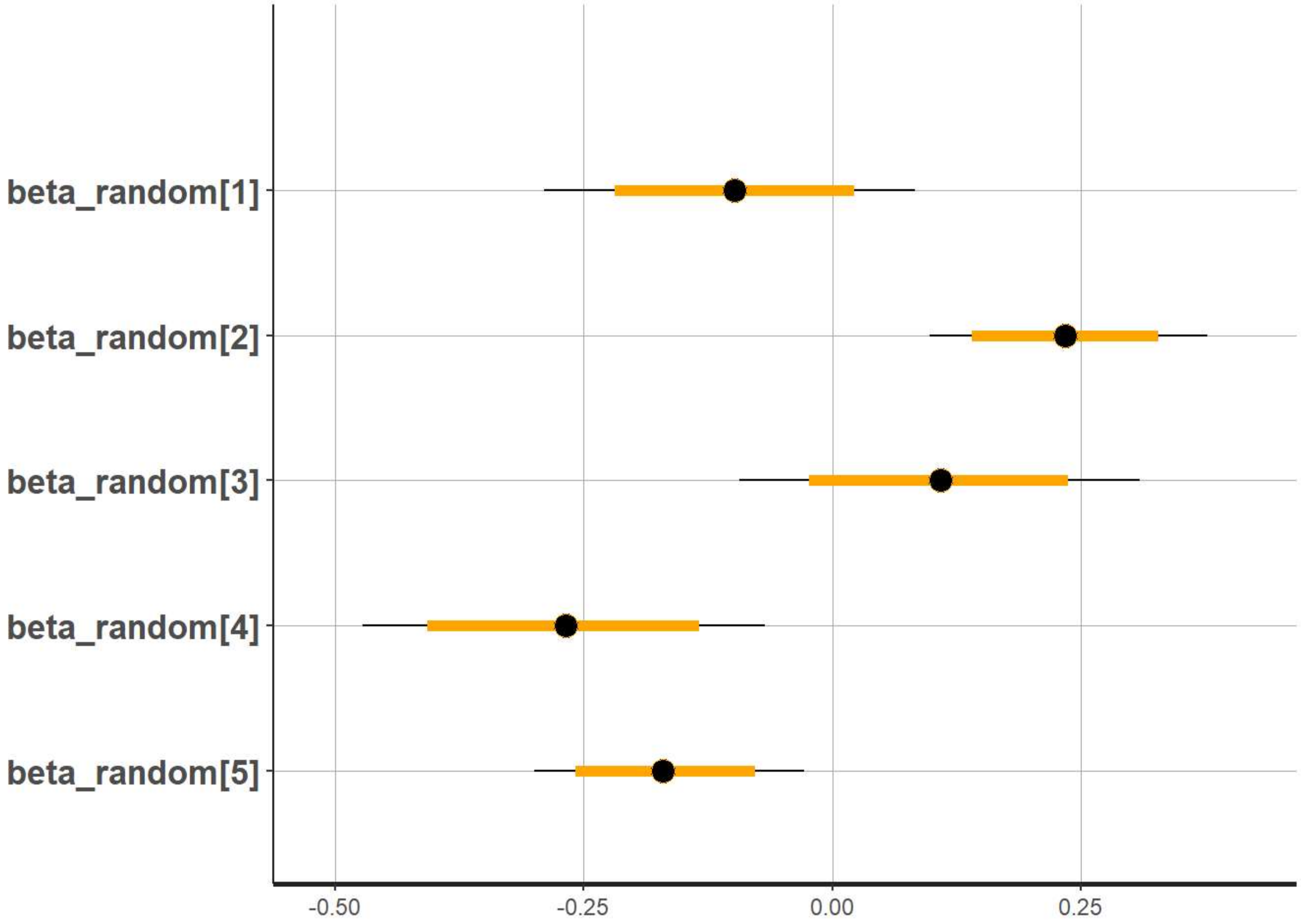
Hide

```
stan_data_model2 <- list(
  population = data$N,
  n = nrow(data),
  area = data$A,
  type = data$type,
  ntype= n_distinct(data$type),
  region = data$region,
  nregion = n_distinct(data$region),
  seed=seed,
  cov_fixed = covs_scaled %>% select(-x4),
  ncov_fixed = ncol(covs_scaled) -1,
  cov_random = covs_scaled$x4
  )

pars <- c('alpha','sigma','beta_fixed','beta_random','alpha_t','alpha_t_r', 'populat
ion_hat',  'density_hat')

# mcmc
fit2 <- rstan::stan(file = file.path('tutorials/tutorial3/tutorial3_model2.stan'),
                   data = stan_data_model2,
                   iter = warmup + iter,
                   chains = chains,
                   warmup = warmup,
                   pars = pars,
                   seed = seed)
```

No convergence issue. We plot `beta_random` that is a vector with a $\hat{\beta}_t$ for each settlement type.

Hide

```
stan_plot(fit2, pars='beta_random', fill_color='orange')
```



We see that modelling β^{x4} by settlement type unravels different patterns: for settlement 2 we observe a positive effect for settlement 2, a weak effect for settlement 1, and 3 and a negative effect for settlement type 4 and 5. TBC with settlement type definition ask Chris

We now want to evaluate the effect on the predicted population count for each study site.

Hide

```
comparison_df <- rbind(
 getPopPredictions(fit1) %>%
   mutate(model='Fixed effect'),
 getPopPredictions(fit2) %>%
   mutate(model='Random effect in x4'))
# compute goodness-of-fit metrics
comparison_df %>% group_by(model) %>%
  summarise( `Bias`= mean(residual),
    `Inaccuracy` = mean(abs(residual)),
        `Imprecision` = sd(residual),
    `Mean Confidence Interval Size` = mean(ci_size)
) %>%  kbl(caption = 'Goodness-of-metrics comparison with and without random effect
 in x4 ') %>% kable_minimal()
```

Table 3: Goodness-of-metrics comparison with and without random effect in x4

model	Bias	Inaccuracy	Imprecision	Mean Confidence Interval Size
Fixed effect	33.56579	193.6548	275.4327	1251.533

model	Bias	Inaccuracy	Imprecision	Mean Confidence Interval Size
Random effect in x4	33.26182	192.1447	274.5590	1240.142

We see a slight decrease of bias and an increase of the precision of the estimates with shorter credible intervals.

We will save the results of this final model as a RDS file to explore it in Tutorial 4.

Hide

```
saveRDS(fit2, 'tutorials/tutorial3/tutorial3_model2_fit.rds')
```

References

Ecopia.AI, and Maxar Technologies. 2019. *Digitize Africa Data*.

Institut Géographique du Burkina Faso. 2015. “Base Nationale de Données Topographiques.”

Leasure, Douglas R, Warren C Jochem, Eric M Weber, Vincent Seaman, and Andrew J Tatem. 2020. “National Population Mapping from Sparse Survey Data: A Hierarchical Bayesian Modeling Framework to Account for Uncertainty.” *Proceedings of the National Academy of Sciences*.

Oak Ridge National Laboratory. 2018. “LandScan HD: Nigeria.”

OpenStreetMap contributors. 2018. *Planet Dump Retrieved from Https://Planet.osm.org*.

Weiss, D J, A Nelson, HS Gibson, W Temperley, S Peedell, A Lieber, M Hancher, et al. 2018. “A Global Map of Travel Time to Cities to Assess Inequalities in Accessibility in 2015.” *Nature* 553 (7688): 333336.

WorldPop Research Group, Department of Geography and Geosciences, University of Louisville, Departement de Geographie, Universite de Namur, and Center for International Earth Science Information Network (CIESIN), Columbia University. 2018. “Global High Resolution Population Denominators Project - Funded by the Bill and Melinda Gates Foundation (Opp1134076).” https://doi.org/10.5258/SOTON/WP00645 (https://doi.org/10.5258/SOTON/WP00645).