Demo: modelling spatial correlation in damage

Michele Nguyen

03/02/2022

```
rm(list = ls())
library(fragilitycurves)
library(MASS)
library(gridExtra)
library(ggplot2)
library(sp)
library(lemon) # For grid_arrange_shared_legend.
library(raster)
library(TMB)
library(geoR) # For matern.
library(dplyr)
library(gstat) # For unconditional simulation of spatial fields.
library(rgdal)
library(rgeos)
```

Introduction

In this analysis, we aim to demonstrate the functions in `fragilitycurves` R package which fit the non-spatial and spatial damage models as well as to compute the difference in resulting annual loss estimates.

Sample data

We use simulated earthquake building damage data for the Haiti for illustration. To fit the spatial ordinal model (or the Damage-spatial model), we use data for two building categories and a raster surface of mean peak ground acceleration (PGA).

```
# Read in mean PGA surface:
data(mean_PGA)
res(mean_PGA) # About 1km pixels.
```

```
## [1] 1060 1110
```

```
# Read in sample datasets:
data(damage_simulation)
data.subset.1 <- damage_simulation[damage_simulation$building_cat == 1, ]
head(data.subset.1)
```

```
## # A tibble: 6 x 6
```

```
## # Groups:   CDF [1]
##   building_cat CDF   logPGA   PGA Easting Northing
##          <dbl> <ord>  <dbl> <dbl>   <dbl>    <dbl>
## 1            1 0       3.50 33.1  759701. 2039467.
## 2            1 0       3.65 38.6  751221. 2039467.
## 3            1 0       1.33  3.80 820121. 2138257.
## 4            1 0       3.52 33.9  738501. 2036137.
## 5            1 0       3.42 30.4  715181. 2033917.
## 6            1 0       3.74 42.2  737441. 2038357.
```

```
table(data.subset.1$CDF)
```

```
##
##   0 0.5   5  20  45  80 100
##  25  25  25  25  25  25  25
```

```
data.subset.2 <- damage_simulation[damage_simulation$building_cat == 2, ]
head(data.subset.2)
```

```
## # A tibble: 6 x 6
## # Groups:   CDF [1]
##   building_cat CDF   logPGA   PGA Easting Northing
##          <dbl> <ord>  <dbl> <dbl>   <dbl>    <dbl>
## 1            2 0       3.83 45.9  765001. 2042797.
## 2            2 0       3.40 30.1  709881. 2036137.
## 3            2 0       1.25  3.50 802101. 2153797.
## 4            2 0       3.97 52.8  726841. 2040577.
## 5            2 0       3.43 31.0  774541. 2052787.
## 6            2 0       3.41 30.3  724721. 2032807.
```

```
table(data.subset.2$CDF)
```

```
##
##   0 0.5   5  20  45  80 100
##  25  25  25  25  25  25  25
```

`data.subset.1` contains damage data for *Building Category 1 (Unreinforced block walls)* and `data.subset.2` for *Building Category 2 (Stone masonry)*. Both datasets contain information for 25 buildings per assessed damage grade including their Easting and Northing coordinates, the log peak ground acceleration (`log(PGA)`) experienced and the central damage factor recorded (`CDF`). Here, we have a 7-level damage scale. We make a note of the CDFs in our data:

```
CDF_breaks <- sort(unique(data.subset.1$CDF), decreasing = FALSE)
CDF_breaks
```

```
## [1] 0   0.5 5   20  45  80  100
## Levels: 0 < 0.5 < 5 < 20 < 45 < 80 < 100
```

Notice that the `CDF` column is ordered. We check the classes of the other columns:

```r
str(data.subset.1)
```

```
## grouped_df [175 x 6] (S3: grouped_df/tbl_df/tbl/data.frame)
##  $ building_cat: num [1:175] 1 1 1 1 1 1 1 1 1 1 ...
##  $ CDF         : Ord.factor w/ 7 levels "0"<"0.5"<"5"<..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ logPGA      : num [1:175] 3.5 3.65 1.33 3.52 3.42 ...
##  $ PGA         : num [1:175] 33.1 38.6 3.8 33.9 30.4 ...
##  $ Easting     : num [1:175] 759701 751221 820121 738501 715181 ...
##  $ Northing    : num [1:175] 2039467 2039467 2138257 2036137 2033917 ...
##  - attr(*, "groups")= tibble [7 x 2] (S3: tbl_df/tbl/data.frame)
##   ..$ CDF  : Ord.factor w/ 7 levels "0"<"0.5"<"5"<..: 1 2 3 4 5 6 7
##   ..$ .rows: list<int> [1:7]
##   .. ..$ : int [1:25] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..$ : int [1:25] 26 27 28 29 30 31 32 33 34 35 ...
##   .. ..$ : int [1:25] 51 52 53 54 55 56 57 58 59 60 ...
##   .. ..$ : int [1:25] 76 77 78 79 80 81 82 83 84 85 ...
##   .. ..$ : int [1:25] 101 102 103 104 105 106 107 108 109 110 ...
##   .. ..$ : int [1:25] 126 127 128 129 130 131 132 133 134 135 ...
##   .. ..$ : int [1:25] 151 152 153 154 155 156 157 158 159 160 ...
##   .. ..@ ptype: int(0)
##   ..- attr(*, ".drop")= logi TRUE
```

For most of the functions, the input datasets require at least the columns `CDF` (ordered factor), `PGA` (numeric), `logPGA` (numeric), `Easting` (numeric) and `Northing` (numeric). The functions also work if we do not a point-level dataset like that for Haiti, but instead have data aggregated into grids with the number of buildings of each type in each damage grade per grid square (this is the case for the Nepal 2015 earthquake damage data). To fit the non-spatial and spatial ordinal models, we can create an approximate point-level dataset by creating a dataframe with rows for each building and use the grid centroid coordinates, for example, to extract the PGA values.

Fitting a non-spatial ordinal model

We use the library `MASS` and its `polr` function to fit a non-spatial ordinal model to each of the data subsets.

```r
frag.model.1 <- polr(CDF ~ logPGA, data = data.subset.1, method = "probit", Hess = TRUE)
frag.model.1$coefficients
```

```
##    logPGA
## 0.4107456
```

```r
frag.model.1$zeta
```

```
##     0|0.5     0.5|5      5|20     20|45     45|80    80|100
## 0.1867683 0.7038374 1.1088331 1.4887481 1.8904535 2.4077337
```

```r
frag.model.2 <- polr(CDF ~ logPGA, data = data.subset.2, method = "probit", Hess = TRUE)
frag.model.2$coefficients
```

```
##    logPGA
## 0.2416098
```

```
frag.model.2$zeta
```

```
##       0|0.5      0.5|5       5|20      20|45      45|80     80|100
## -0.3349841  0.1804349  0.5724907  0.9380494  1.3298599  1.8324993
```

The first function in the `fragilitycurves` R package plots the fitted fragility curve against empirical proportions in the data subset.

```
?frag_curve
```

We illustrate this for Building Category 1 in Fig. 1.

```
ex.prob.1 <- frag_curve(frag.model.1, data = data.subset.1, plot = TRUE)
```

These exceedance probabilities were calculated by assuming that there is a latent variable with a normal distribution which has a mean of $\beta \log(PGA)$ and a standard deviation of 1. The estimated cut-off points $\{\xi_k\}$, which are represented by the bold vertical lines in Fig. 2, demarcate the damage states so that $P(DS \geq k) = P(Z \geq \xi_k)$.

With the estimated exceedance probabilities, we can compute the mean CDF given the PGA value. The second function in the R package is used for this but before that we define the unique upper limits of the damage bins as well as the bin lengths. For the Haiti 2010 earthquake damage data, these are described by the ATC-13 1985 damage scale.

```
upper.bin <- c(0, 1, 10, 30, 60, 100)
bin.length <- c(1, 1, 10, 20, 30, 40, 1)  # 0 and 100 are treated as point masses.
```

```
?mean_DF
```

```
mDF_vector <- mean_DF(frag.model.1, data.subset.1, upper.bin, bin.length,
                      ex.prob = ex.prob.1, plot = TRUE)
```

As shown in Figure 3(a), the non-spatial ordinal model produces probability density estimates that are somewhat consistent with the assumption of a Beta distribution with two point masses at damage factor 0 and 100, denoted by the red circles. Figure 3(b) illustrates how the estimated mean damage factor varies with PGA.

Fitting a spatial ordinal model for two building categories

Before we attempt to fit a spatial ordinal model which accounts for the spatial correlation in damage beyond that in ground motion intensity, we conduct some exploratory analysis on the data for the two selected building categories.

```
## Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj
## = prefer_proj): Discarded datum Unknown based on WGS84 ellipsoid in Proj4
## definition
```

Figure 4 shows the spatial distribution of the buildings in the simulated earthquake damage data which correspond to Building Category 1 (Unreinforced block walls) and Building Category 2 (Stone masonry). Figure 5 shows the modelled, mean PGA experienced during the event. We see that the PGA is highest

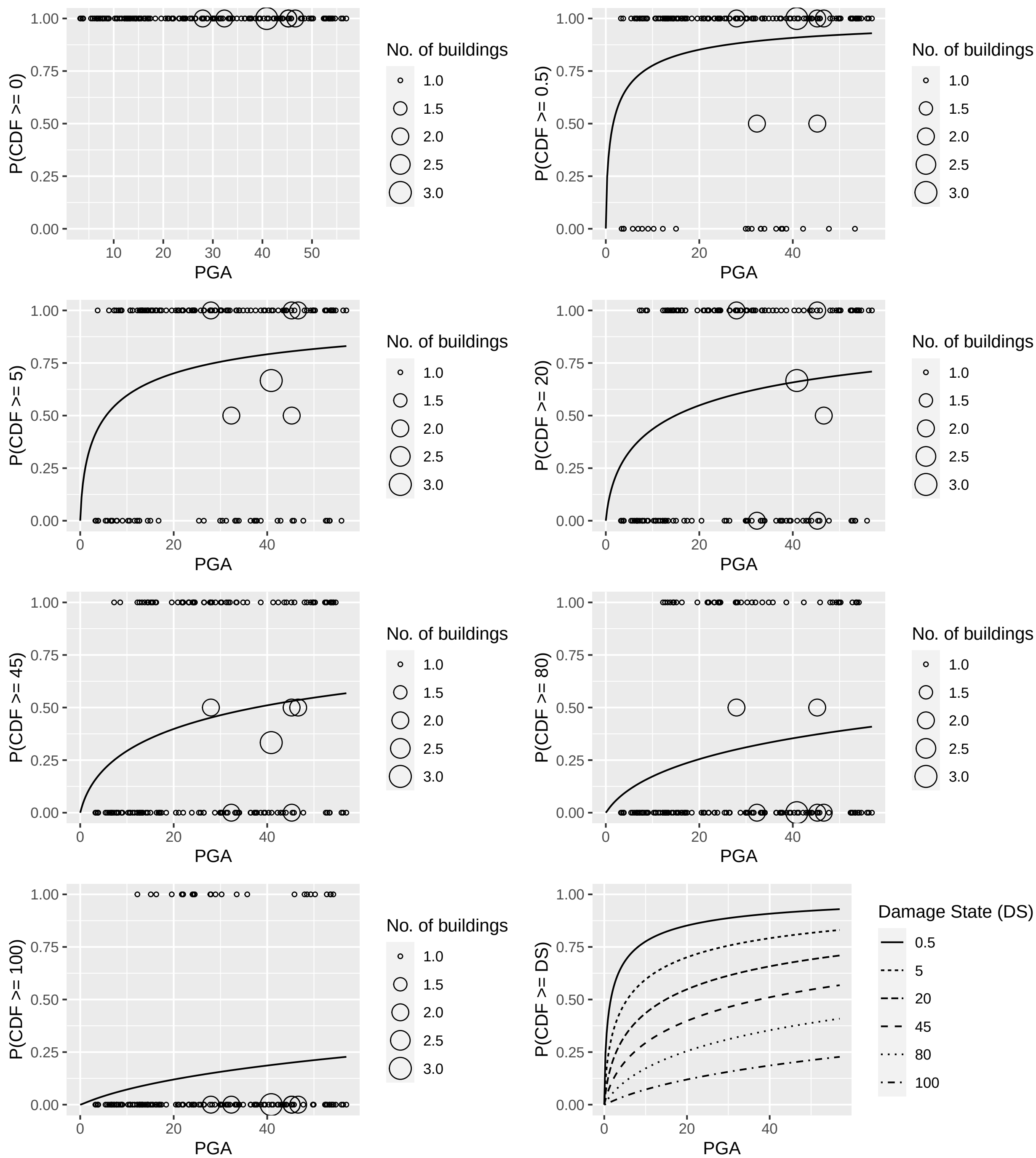

Figure 1: Plot of the fragility curves fitted using the non-spatial ordinal model and the observed empirical proportions of damage state exceedance. Here, CDF refers to the central damage factor of a damage state.

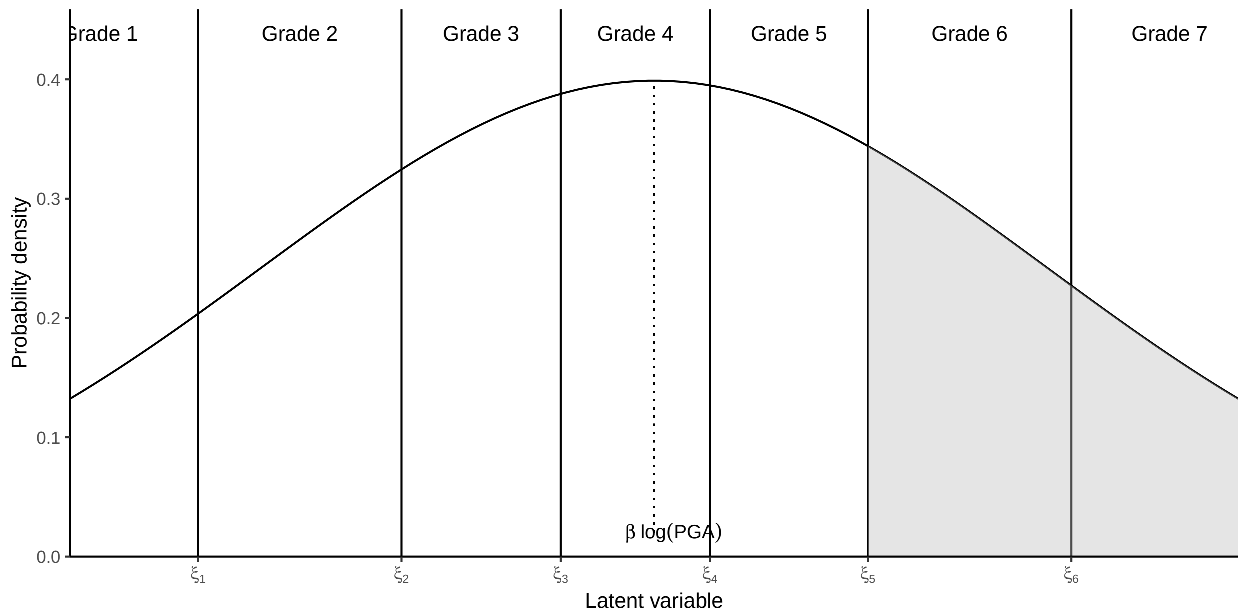

Figure 2: The distribution of the latent variable Z in the fitted non-spatial ordinal model for Building Category 1. The bold vertical lines denote the estimated cut-off points and the dotted vertical line denotes the mean which depends on the PGA value.

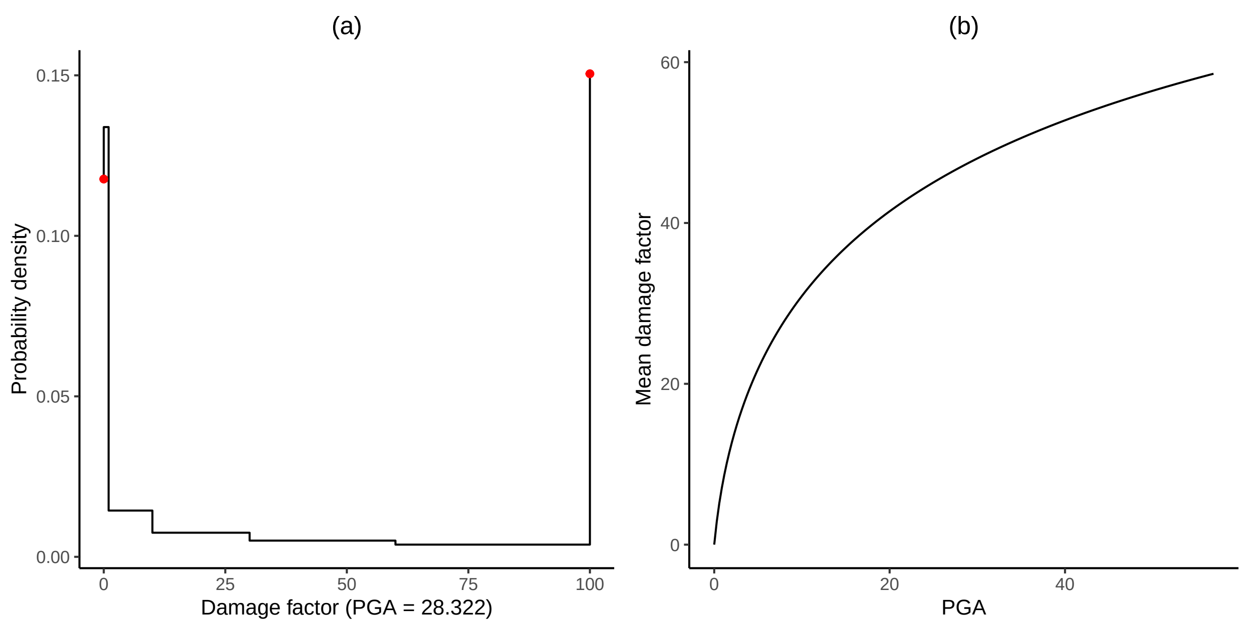

Figure 3: (a) Estimated probability density for the damage factor for a given value of the peak ground acceleration (PGA); (b) Plot of the mean damage factor against PGA as estimated using the non-spatial ordinal model for Building Category 1.

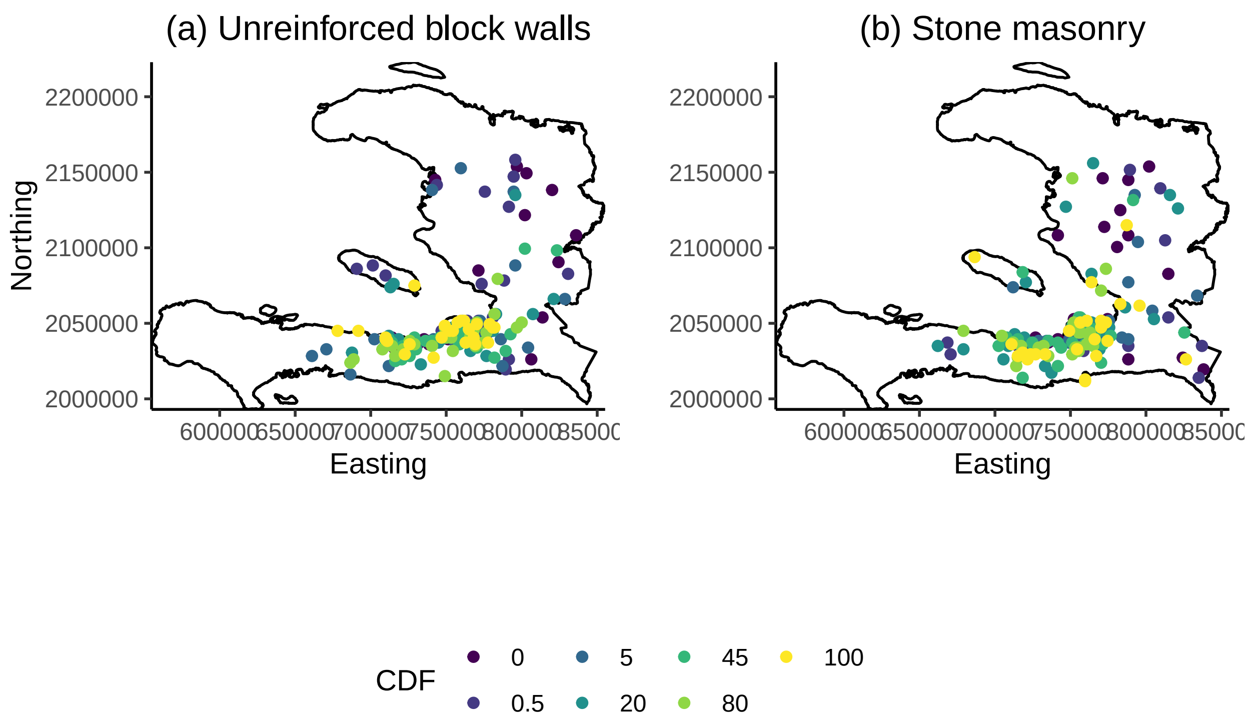

Figure 4: Spatial distribution of buildings from the two categories and their observed central damage factors (CDFs).

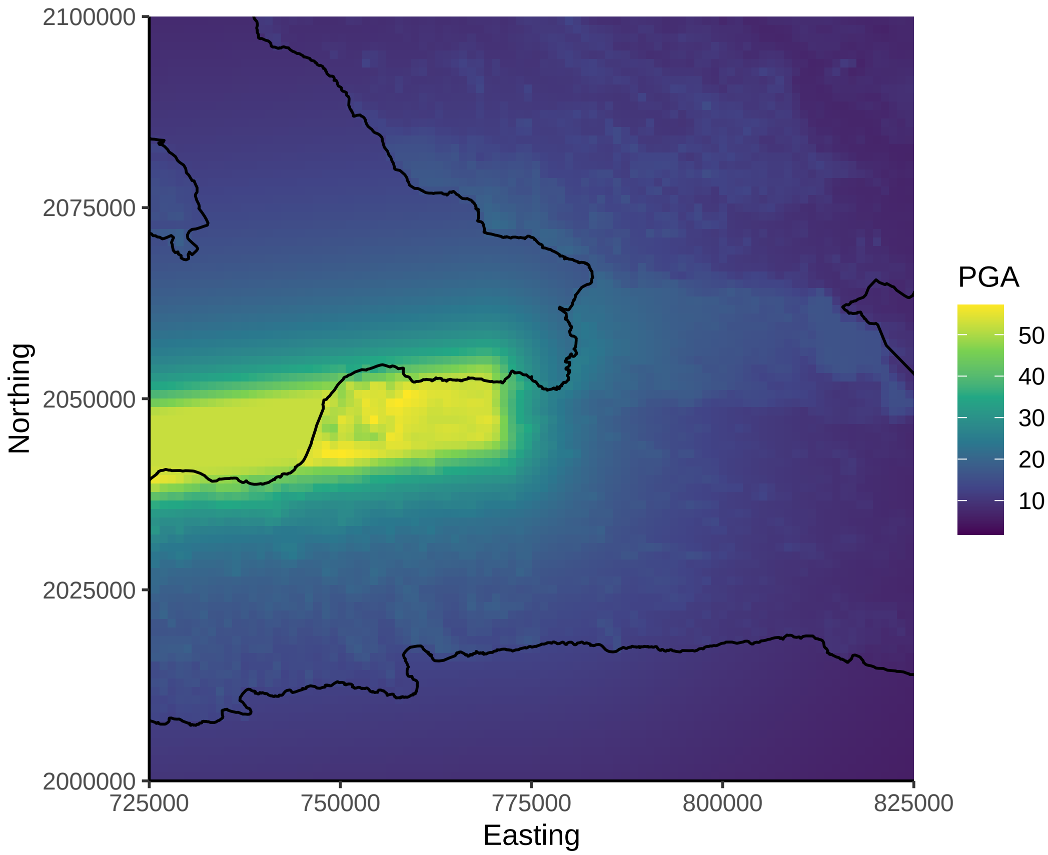

Figure 5: Map of the mean peak ground acceleration (PGA) modelled for the Haiti 2010 earthquake event.

nearer the fault and building damage for both building types seem to be greater towards this direction too. This ties in with the positive β estimates obtained via the non-spatial ordinal models in the previous section. Since there seems to be more yellow points in Figure 4(a) than Figure 4(b), it seems that Building Category 1 is more susceptible to damage than Building Category 2. This is somewhat consistent with the larger β estimate obtained for the former. We also notice that Building Category 2 (Stone masonry) has more damaged buildings in the lower part of the study region (near Northing 2025000) than Building Category 1 (Unreinforced block walls). This could allude to different amounts of random error or spatial pattern/correlation in the damage to the different building categories. By fitting a joint spatial model with common and building category specific spatial fields, we attempt to separate the spatial correlation in damage that is common to both categories through that in PGA and that which is unique to the categories.

The spatial correlation ranges which we can identify are dependent on the spatial resolution of our data. Previously we saw that the mean PGA raster had a resolution of about 1km by 1km. Next, we examine the distances between the buildings in our dataset:

```
data.subset <- rbind(data.subset.1, data.subset.2)
dist.mat <-as.matrix(stats::dist(data.subset[, c('Easting', 'Northing')],
                                 method = "euclidean", diag = TRUE))
dist.mat <- dist.mat/1000 # Work in km instead of m.
dist.mat.1 <-as.matrix(stats::dist(data.subset.1[, c('Easting', 'Northing')],
                                   method = "euclidean", diag = TRUE))
dist.mat.2 <-as.matrix(stats::dist(data.subset.2[, c('Easting', 'Northing')],
                                   method = "euclidean", diag = TRUE))
dist.mat.1 <- dist.mat.1/1000 # Work in km instead of m.
dist.mat.2 <- dist.mat.2/1000 # Work in km instead of m.
```

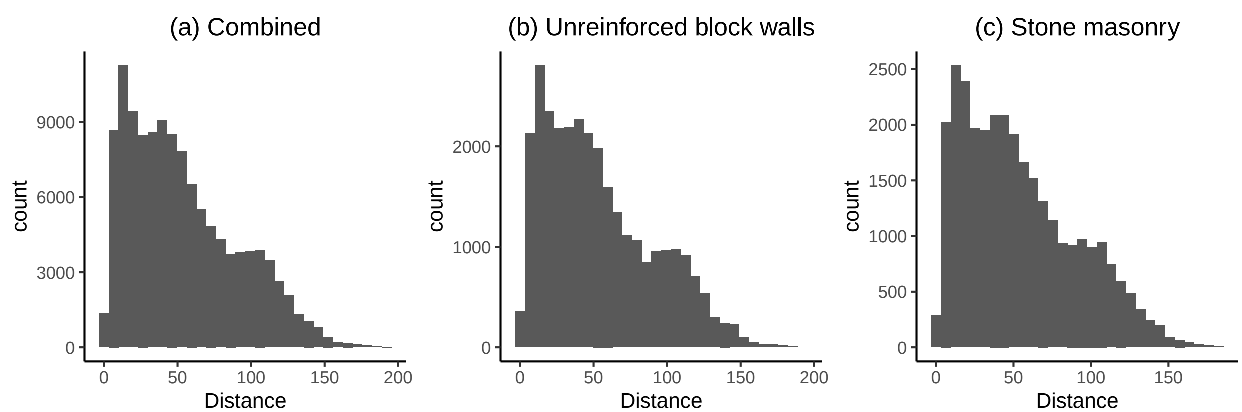

Figure 6: Histograms of pair-wise distances between observations: (a) from the combined dataset; (b) Building Category 1; and (c) Building Category 2.

```
## [1] 1.06
```

```
## [1] 1.06
```

```
## [1] 1.06
```

```
## [1] 193
```

Since the minimum inter-site distance in our dataset is 1.06km and the maximum inter-site distance is about 193km, we should not expect to estimate building category specific spatial correlation ranges of less than 1.06km and more than 193km. Due to the mean PGA raster resolution, the shared spatial field correlation range should also be more than 1km. In fact, literature suggests that this can range from 5km to 150km, depending on study site.

Next, we set up the starting parameter values as well as parameter bounds for the optimisation. These were informed by both the non-spatial ordinal model fit and the literature. Readers are advised to refer to the spatial ordinal model formula in the main paper for more information on the model parameters.

```
# Set some reasonable upper and lower limits for parameters:
lower_lim <- rep(-Inf, 23); upper_lim <- rep(Inf, 23);
log_phi_max <- log(3); log_phi_min <- log(0.05);
log_slope1_max <- log(1.5*frag.model.1$coefficients);
log_slope1_min <- log(0.5*frag.model.2$coefficients)
log_slope2_max <- log(1.5*frag.model.1$coefficients);
log_slope2_min <- log(0.5*frag.model.2$coefficients)
cutoff.1.start <- frag.model.1$zeta
cutoff.2.start <- frag.model.2$zeta
# Reparameterising cut-offs to ensure increasing order in optimisation:
first_cutoff1 <-  cutoff.1.start[1]
first_cutoff2 <-  cutoff.2.start[1]
cutoff_factors <- function(cutoffs){
  temp <- rep(NA, length(cutoffs)-1)
  for (i in 2:length(cutoffs)){
    temp[i-1] <- cutoffs[i]-cutoffs[i-1]
  }
  return(temp)
}
cutoff_factors1 <- cutoff_factors(cutoff.1.start)
cutoff_factors2 <- cutoff_factors(cutoff.2.start)
cutoff11_max <- Inf; cutoff11_min <- -Inf
cutoff21_max <- Inf; cutoff21_min <- -Inf
factor_max <- 5*max(c(cutoff_factors1, cutoff_factors2));
factor_min <- 0.5*min(c(cutoff_factors1, cutoff_factors2))
lower_lim[1] <- log_phi_min;
lower_lim[10] <- log_slope1_min; lower_lim[11] <- log_slope2_min;
lower_lim[12:17] <- c(cutoff11_min, rep(factor_min, length(cutoff_factors1)));
upper_lim[c(1, 4, 7)] <- log_phi_max;
upper_lim[10] <- log_slope1_max;  upper_lim[11] <- log_slope2_max;
upper_lim[12:17] <- c(cutoff11_max, rep(factor_max, length(cutoff_factors1)));
upper_lim[18:23] <- c(cutoff21_max, rep(factor_max, length(cutoff_factors2)));
upper_lim[3] <- -2;
```

We will use the `spatial_ordinal` function to fit the spatial ordinal model.

```
?spatial_ordinal
```

The spatial ordinal model takes about 12 minutes in a PC with characteristics: Intel(R) Xeon (R) W-2112 CPU Processor @ 3.60GHz; 32GB of RAM; Windows 10 64-bit.

```
temp.time <- proc.time()[3]
spatial_fit <- spatial_ordinal(frag.model.1, frag.model.2,
                                data.subset.1, data.subset.2,
```

```
                                  lower.lim = lower_lim,
                                  upper.lim = upper_lim)
time.taken <- proc.time()[3] - temp.time
demo_spatial_fit <- spatial_fit
```

These are the parameter estimates.

```
demo_spatial_fit$par
```

```
##       log_phi  log_sigma_2    log_tau_2     log_phi1   log_tau1_2 log_sigma1_2
##     0.7030293    0.3613460  -15.0551272   -1.0316279   -0.3514987   -2.3727745
##      log_phi2   log_tau2_2 log_sigma2_2   log_slope1   log_slope2    c_factor1
##    -1.3988869   -0.3983044   -0.3446092   -0.6907916   -0.4903512    0.3140576
##     c_factor1    c_factor1    c_factor1    c_factor1    c_factor1    c_factor2
##     0.5456045    0.4257554    0.4022524    0.4240713    0.5549534    0.2720625
##     c_factor2    c_factor2    c_factor2    c_factor2    c_factor2
##     0.6945181    0.5445101    0.5182927    0.5598681    0.7135123
```

Next, we visualise the fitted variograms as well as estimated spatial fields in the capital of Haiti, Port-au-Prince, using the `vgm_plot`, `kriged_fields` and `latent_var` functions.

```
?vgm_plot
?kriged_fields
?latent_var
```

```
shared_range <- seq(0, 25, by = 0.2); cat_range <- seq(0, 10, by = 0.2)
vgm_plot(demo_spatial_fit$par, shared_range, cat_range)
```

From Figure 7, we estimate a spatial range for the shared spatial field of about 15 km, while those specific to Building category 1 and 2 are found to be about 2-2.5km. The effect of these are shown in the estimated spatial fields around the captial of Port-au-Prince in Figures 8(b)-(c) and 9(b)-(c).

```
# Convert the cut-off factors to cut-off values on the latent variable scale:
new_par <- convert_cutoffs(demo_spatial_fit$par)
# Obtain the shapefile for Port-au-Prince:
data(haiti_admin2)
pap_shp <- haiti_admin2[haiti_admin2$ADM2_EN == "Port-au-Prince", ]
pap_shp <- spTransform(pap_shp, CRS("+proj=utm +zone=18 ellps=WGS84"))
# Krige the field estimates from the spatial ordinal model fit:
kriged_rasters <- kriged_fields(demo_spatial_fit, data.subset.1, data.subset.2,
                                  pap_shp, mean_PGA)
```

```
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
```

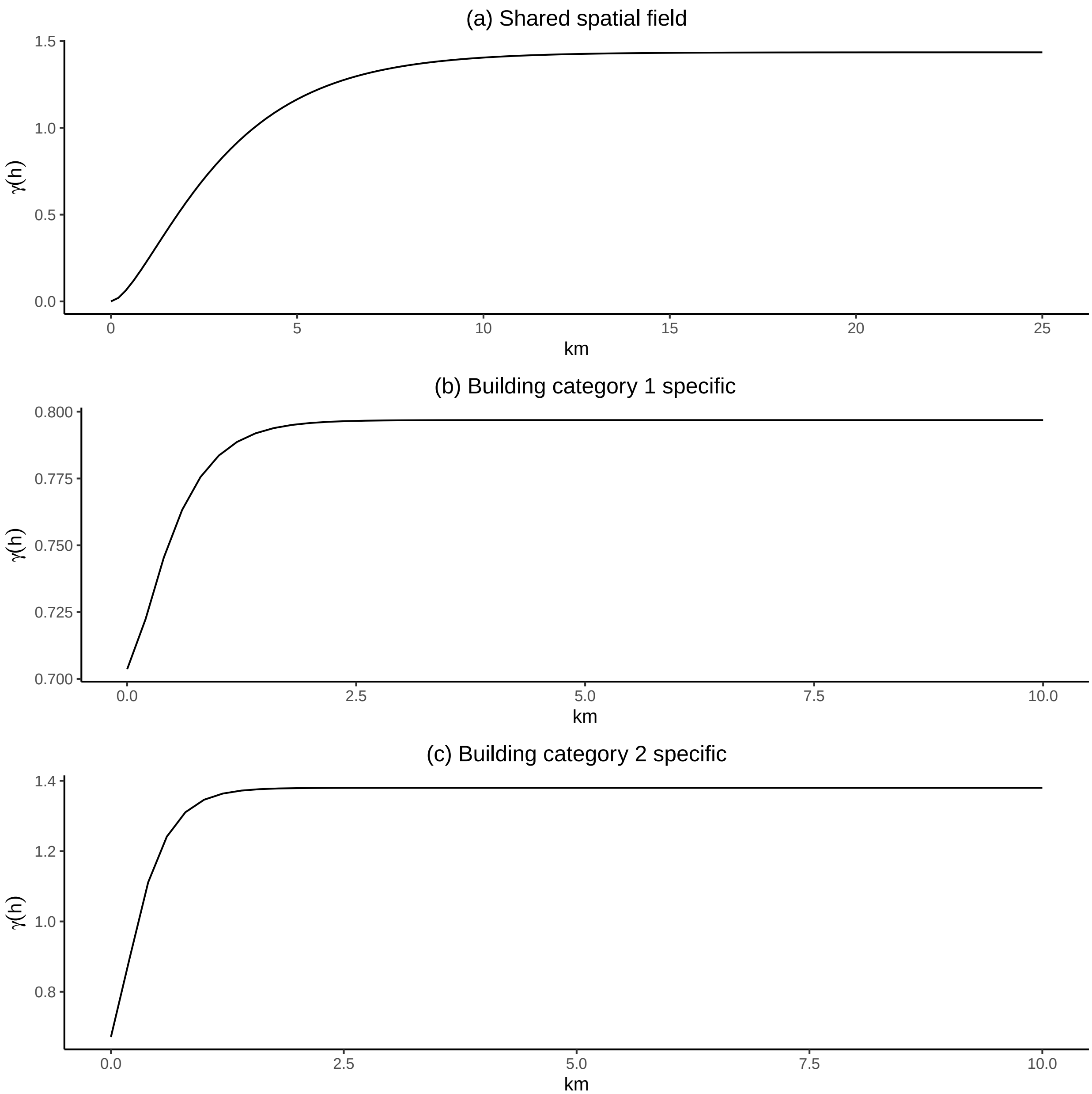

Figure 7: Fitted variograms of the spatial ordinal model for: (a) the shared spatial field; (b) Building Category 1 specific spatial field; and (c) Building Category 2 specific spatial field.

```
# Compute the latent variable mean raster for a building category and plot its contributing terms.
latent_var_1 <- latent_var(category = 1, new_par, kriged_rasters, mean_PGA, study_shp)
```

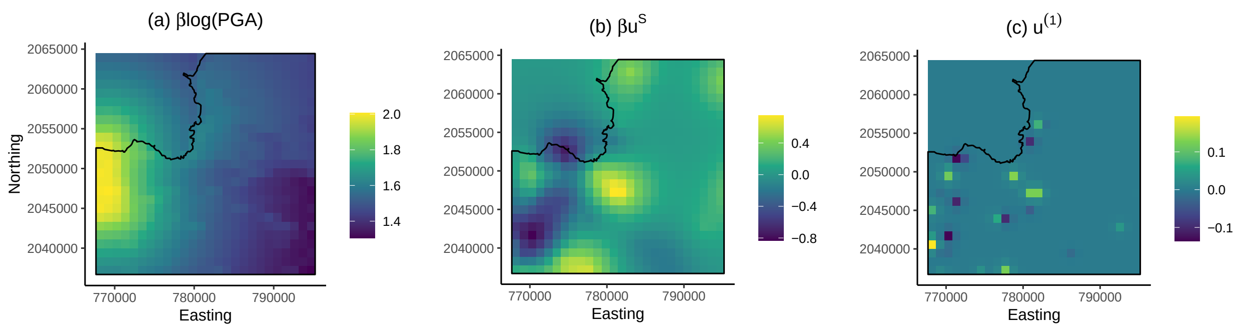

Figure 8: Contributing terms of Building category 1's latent variable mean surface.

```
# Compute the latent variable mean raster for a building category and plot its contributing terms.
latent_var_2 <- latent_var(category = 2, new_par, kriged_rasters, mean_PGA, study_shp)
```

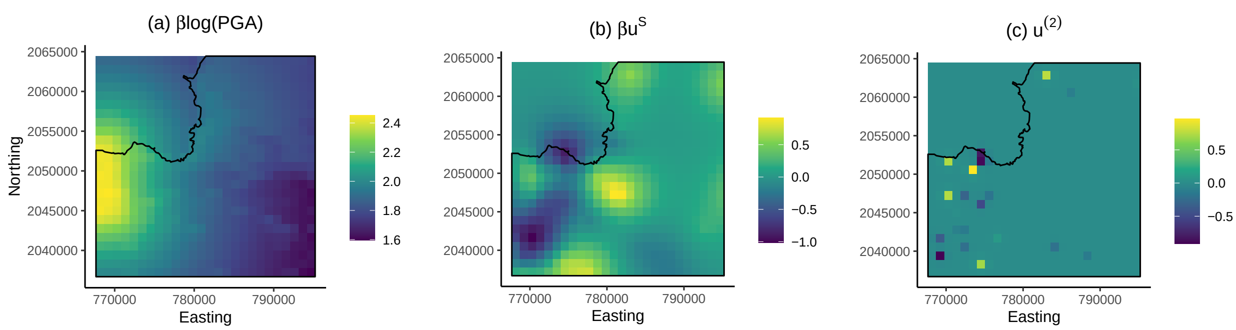

Figure 9: Contributing terms of Building category 2's latent variable mean surface.

From Figures 8 and 9, we also see that the $\beta log(PGA)$ component is the larger of the three components that are added up to form the latent variable means for the two building categories. But as we will show later, the spatial intricacies that are modelled via spatial fields can have noticeable effects on the estimated damage probabilities and loss estimation.

In Figure 10(a), we chose two locations (Site 1 and 2) to illustrate the effect of the Building category 1 specific spatial field on its ordinal probability distributions. Nearer Site 2, higher latent variable mean values lead to a shift in the probability density to the right. This in turn leads to higher exceedance probabilities for higher damage states. The converse holds for locations near Site 1. This can also be seen in the maps of exceedance probabilities in Figure 11.

```
exceed_prob_1 <- prob_exceed(1, new_par, CDF_breaks, latent_var_1, study_shp)
```

For completeness, we also show the maps of exceedance probabilities for Building category 2 in Figure 12. These are computed during the `prob_exceed` function in the R package.

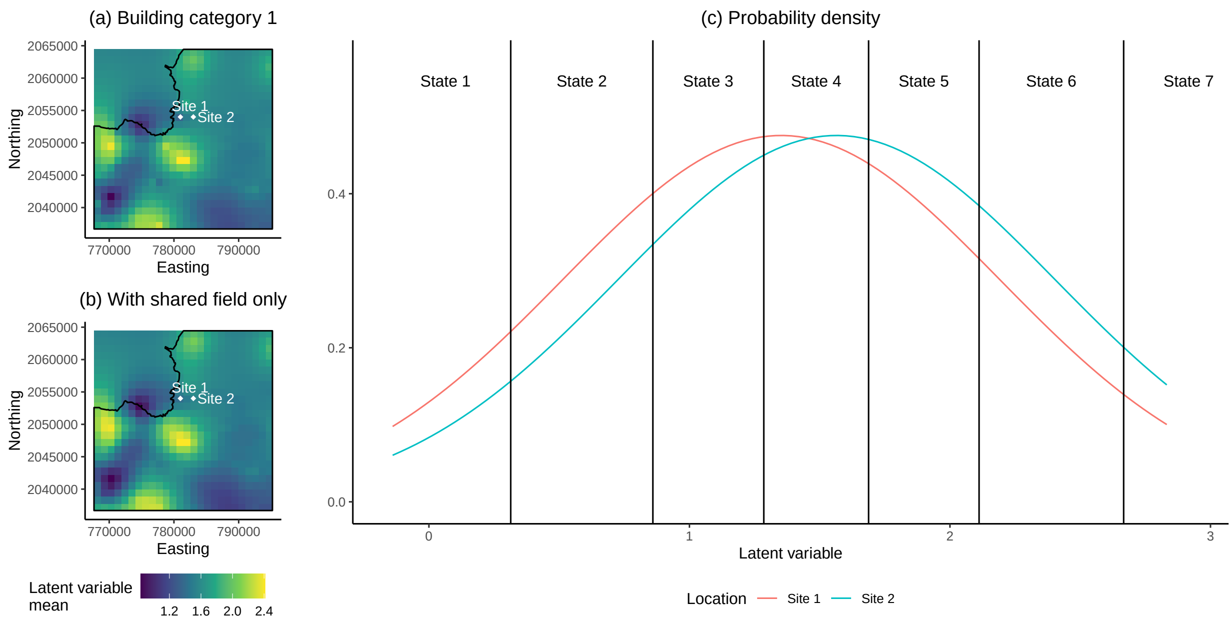

Figure 10: Illustrative plots of how the spatial fields shift the ordinal distributions for Building category 1.

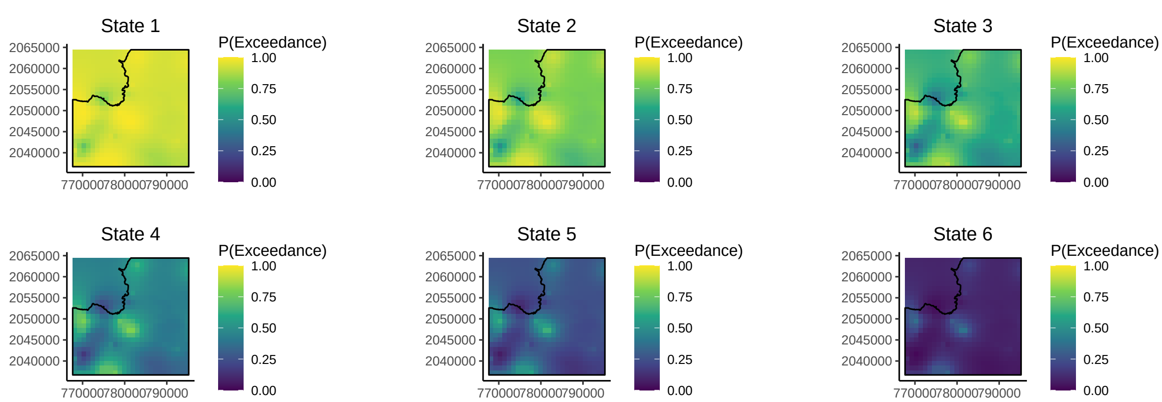

Figure 11: Maps of the exceedance probabilities for different damage states (Building category 1).

```r
exceed_prob_2 <- prob_exceed(2, new_par, CDF_breaks, latent_var_2, study_shp)
```

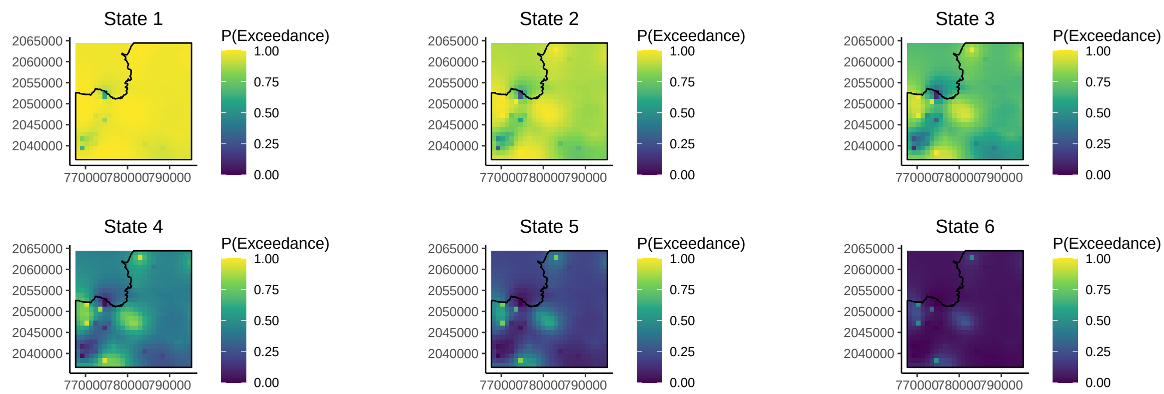

Figure 12: Maps of the exceedance probabilities for different damage states (Building category 2).

The spatial patterns in the latent variable mean surface translate to similar patterns in the exceedance probability maps.

Next, we will use one million one-year stochastic event set (SES) simulations from the OpenQuake engine as well as a portfolio of 150 buildings of each building type within a 2km x 2km grid to illustrate the effect of modelling spatial correlation in damage.

Computing annual loss curves using OpenQuake and spatial field simulations

OpenQuake enables us to obtain rasters of simulated PGA over our study region. We can extract the simulated PGA for our buildings in Port-au-Prince per event.

```r
data(oq_sim)
oq_sim[1:5, 1:5]
```

```
## # A tibble: 5 x 5
## # Groups:   building_cat [1]
##     lon   lat building_cat   `10`   `40`
##   <dbl> <dbl>        <dbl>  <dbl>  <dbl>
## 1 -72.3  18.6            1 -0.227   0.887
## 2 -72.3  18.5            1 -0.0266  0.308
## 3 -72.3  18.5            1  0.772  -0.149
## 4 -72.3  18.6            1 -0.227   0.887
## 5 -72.3  18.5            1  0.772  -0.149
```

```r
events_affected <- ncol(oq_sim) - 3
```

Each row denotes a building with its corresponding longitude and latitude coordinates as well as category (`building_cat = 1` or `2`). The numbered columns correspond to the $\log(PGA)$ values for the simulated events which are identified by their `event_id`. Note that we only record events which affect at least one building of our concern. Hence, the `event_id`s are not consecutive. The `NA` values which occur as a result of the location being outside the range of the calculation from the earthquake source (set to 200km in our case) are replaced by `-Inf`.

In addition to the PGA per building per event, we need to associate events with the year of occurrence or SES (there can be multiple events in one year). The dataframe `event_ses_list` contains this information:

```
data("event_ses_list")
head(event_ses_list)
```

```
##   event_id rlz_id
## 1        0    472
## 2        1  49429
## 3        2  26865
## 4        3  11310
## 5        4  45005
## 6        5  47568
```

```
ses_list <- unique(event_ses_list$rlz_id)
```

Note it is likely that the `ses_id` will be available in the `events` file given by an updated version of OpenQuake. However, for now, we use `rlz_id` in place of this because it is equivalent when we set `ses_per_logic_tree_path = 1` for multiple logic tree samples in the OpenQuake `job.ini` file.

```
# Change coordinates to easting and northing:
obs.pts <- SpatialPoints(oq_sim[, c("lon", "lat")],
                         proj4string = CRS("+proj=longlat"))
UTM.pts <- spTransform(obs.pts, CRS("+proj=utm +zone=18 ellps=WGS84"))
colnames(UTM.pts@coords) <- c("Easting", "Northing")
UTM.pts@coords <- UTM.pts@coords/1000 # Parameters relate to km for distance not m.
```

We convert the coordinates to Easting and Northing to simulate the spatial fields required to compute the latent variable means. For Building category $i = 1, 2$, the latent variable values for the three models are defined as follows:

$$\mathbf{Z}^{(i)} = \begin{cases} [\beta_i(\log(IM) + \mathbf{u}^S + \mathbf{e}^S) + \mathbf{u}^i + \mathbf{e}^{D,i}] + \mathbf{e}^{(i)} & \text{[Damage-spatial]} \\ [\beta_i(\log(IM) + \mathbf{u}^S + \mathbf{e}^S)] + \tilde{\mathbf{e}}^{(i)} & \text{[IM-spatial]} \\ [\beta_i(\log(IM) + \tilde{\mathbf{e}}^S)] + \tilde{\mathbf{e}}^{(i)} & \text{[Non-spatial]} \end{cases}$$

where β_i is the associated slope coefficient, $\log(IM)$ is the log-transformed PGA value, $\mathbf{u}^S$ is the shared spatial field with a Matérn covariance and $\mathbf{e}^S$ is its dummy nugget component. Here, $\mathbf{u}^i$ is the building category specific field with a different Matérn covariance and $\mathbf{e}^{(i)}$ is its nugget or random error component while $\mathbf{e}^{D,i}$ is its dummy nugget component for kriging. Comparing the Damage-spatial case to the Non-spatial case, we see that the former decomposes the random error terms $\tilde{\mathbf{e}}^S \sim N(0, (\tau^2+\sigma^2)I$ and $\tilde{\mathbf{e}}^{(i)} \sim N(0, (\tau^2+\tau_i^2+\sigma_i^2)I)$ into $\mathbf{u}^S + \mathbf{e}^S$ and $\mathbf{u}^{(i)} + \mathbf{e}^{D,i} + \mathbf{e}^{(i)}$ respectively. The parameters τ^2 and τ_i^2 refer to the estimated nugget variances from the spatial ordinal model while σ^2 and σ_i^2 denote the partial sills. We have used I to represent the identity matrix.

To compute the exceedance probabilities and mean replacement cost (the product of the replacement cost and mean central damage factor), we simulate the latent variable means, μ_{LV}, denoted in the square brackets and compute the exceedance probability of damage state k by $1 - \Phi\left(\frac{\xi_k - \mu_{LV}}{\tau_i}\right)$ for the Damage-spatial case and $1 - \Phi\left(\frac{\xi_k - \mu_{LV}}{\sqrt{\tau_i^2 + \sigma_i^2}}\right)$ for the IM-spatial and Non-spatial cases.

The shared and building category specific fields are simulated once for each of the 156491 events which affect Port-au-Prince:

```r
# Spatial model parameters:
field.phi <- exp(new_par["log_phi"]); field.sigma2 <- exp(new_par["log_sigma_2"]);
field.tau2 <- exp(new_par["log_tau_2"]);
field1.phi <- exp(new_par["log_phi1"]); field1.sigma2 <- exp(new_par["log_sigma1_2"]);
field1.tau2 <- exp(new_par["log_tau1_2"]);
field2.phi <- exp(new_par["log_phi2"]); field2.sigma2 <- exp(new_par["log_sigma2_2"]);
field2.tau2 <- exp(new_par["log_tau2_2"]);
beta1 <- exp(new_par["log_slope1"]); beta2 <- exp(new_par["log_slope2"]);
```

```r
# a. Shared field:
# Define the gstat object (spatial model)
g.dummy <- gstat::gstat(formula=z~1, locations=~Easting+Northing, dummy=T, beta=0,
                        model=vgm(psill=field.sigma2,range=field.phi,nugget=field.tau2,
                                  kappa=1,model="Mat"))
# Make simulations based on the stat object
set.seed(2)
temp.time <- proc.time()[3]
field.sim <- predict(g.dummy, newdata=UTM.pts, nsim=events_affected)
time.taken.2 <- proc.time()[3] - temp.time # 1h 2 min.
# b. Field for Building Cat 1:
# Define the gstat object (spatial model)
g.dummy1 <- gstat(formula=z~1, locations=~Easting+Northing, dummy=T, beta=0,
                  model=vgm(psill=field1.sigma2,range=field1.phi,nugget=field.tau2,
                            kappa=1,model="Mat"))
set.seed(3)
temp.time <- proc.time()[3]
field1.sim <- predict(g.dummy1, newdata=UTM.pts[oq_sim$building_cat == 1, ],
                      nsim=events_affected)
time.taken.3 <- proc.time()[3] - temp.time # 17 min.
# c. Field for Building Cat 2:
# Define the gstat object (spatial model)
g.dummy2 <- gstat(formula=z~1, locations=~Easting+Northing, dummy=T, beta=0,
                  model=vgm(psill=field2.sigma2,range=field2.phi,nugget=field.tau2,
                            kappa=1,model="Mat"))
set.seed(4)
temp.time <- proc.time()[3]
field2.sim <- predict(g.dummy2, newdata=UTM.pts[oq_sim$building_cat == 2, ],
                      nsim=events_affected)
time.taken.4 <- proc.time()[3] - temp.time # 17 min.
```

```r
head(field.sim[, 1:10])
```

```
##         sim1        sim2       sim3       sim4       sim5        sim6
## 1 -0.7574305  0.71290189 -0.9826702 1.04457211  0.5270404  0.18533704
## 2 -0.3719270 -0.03601102 -0.2330767 1.16007423  0.7996938 -0.38303107
## 3 -1.0583454  0.41599184 -0.9255580 0.09179646 -0.2591991 -0.01000454
## 4 -0.5956984  0.71563202 -0.7804698 1.29579914  0.6774712  0.25006470
## 5 -0.6491634  0.56003535 -0.6502938 1.71608746  0.9437344 -0.24261074
## 6 -0.4263102  0.33664823 -0.4941177 1.05840814  0.8453147 -0.58190131
##         sim7       sim8       sim9     sim10
## 1  0.3951398 -1.4778619 -0.2885464 0.2858995
## 2  0.6611152 -1.4150561  1.0754936 1.4583837
## 3  0.9764307 -1.1234163  0.5424443 0.6082234
```

```
## 4  0.2158263 -1.5532991 -0.3076010 0.3251065
## 5  0.3036644 -1.7506132  0.6103598 0.8830219
## 6 -0.2289381 -0.7938935  1.4718192 1.6525806
```

The simulation of the shared field (`field.sim`) takes about 1 hour while the simulation of the building category specific fields (`field1.sim` and `field2.sim`) take about 17 min each. Note that we do not include the nugget for the latter two because from the equation because they do not contribute to the latent variable means. As illustrated for `field.sim` above, the columns of the storage dataframes corresponding to the simulation numbers while the rows correspond to the individual buildings which are ordered according to their order in the `oq_sim` dataframe.

With the estimated fields, we can compute the latent variable means corresponding to the spatial ordinal model for damage correlation as well as its submodels: the non-spatial model and the spatial model considering only the shared field, i.e. only spatial correlation due to $\log(PGA)$. To differentiate between these models, we will refer to them as the "Non-spatial", the "IM-spatial" and the "Damage-spatial" models. The `lv_sim` function compute the latent variable means for the different models:

```
temp.time <- proc.time()[3]
nonspat_lv <- lv_sim(model = "Non-spatial", data = oq_sim, fieldsim = NULL,
                     field1sim = NULL, field2sim = NULL, slope1 = beta1, slope2 = beta2,
                     shared_sill = field.tau2 + field.sigma2)
time.taken.5 <- proc.time()[3] - temp.time # 2 min.
temp.time <- proc.time()[3]
pgaspat_lv <- lv_sim(model = "IM-spatial", data = oq_sim, fieldsim = field.sim,
                     field1sim = NULL, field2sim = NULL, slope1 = beta1, slope2 = beta2,
                     shared_sill = field.tau2 + field.sigma2)
time.taken.6 <- proc.time()[3] - temp.time # 3 min.
temp.time <- proc.time()[3]
damagespat_lv <- lv_sim(model = "Damage-spatial", data = oq_sim, fieldsim = field.sim,
                        field1sim = field1.sim, field2sim = field2.sim, slope1 = beta1,
                        slope2 = beta2, shared_sill = field.tau2 + field.sigma2)
time.taken.7 <- proc.time()[3] - temp.time # 3 min.
```

Based on the simulated latent variable means, we compute the annual losses by defining a replacement cost per building (here we use 1 unit cost) and computing the mean damage factor for each building as the weighted mean of the central damage factors where the weights are the estimated probabilities of being in each damage state. Then, by multiplying the replacement cost with the mean damage factor and summing this up over all the buildings affected and event in the year, we obtain the estimated annual loss. First, we compute the mean replacement cost of our building portfolios per event for the two building types separately:

```
# Cutoff values:
cutoffs1 <- new_par[names(new_par) == "cutoffs1"];
cutoffs2 <- new_par[names(new_par) == "cutoffs2"];
temp.time <- proc.time()[3]
nonspat.rc1 <- portfolio_rc(cutoffs1, CDF_breaks, nonspat_lv$lv1,
                            sqrt(field.tau2 + field1.tau2 + field1.sigma2),
                            replacement.cost = 1)
time.taken.8 <- proc.time()[3] - temp.time # 4 min.
temp.time <- proc.time()[3]
pgaspat.rc1 <- portfolio_rc(cutoffs1, CDF_breaks, pgaspat_lv$lv1,
                            sqrt(field.tau2 + field1.tau2 + field1.sigma2),
                            replacement.cost = 1)
time.taken.9 <- proc.time()[3] - temp.time # 4 min.
temp.time <- proc.time()[3]
```

```
damagespat.rc1 <- portfolio_rc(cutoffs1, CDF_breaks, damagespat_lv$lv1,
                               sqrt(field1.tau2), replacement.cost = 1)
time.taken.10 <- proc.time()[3] - temp.time # 4 min.
temp.time <- proc.time()[3]
nonspat.rc2 <- portfolio_rc(cutoffs2, CDF_breaks, nonspat_lv$lv2,
                            sqrt(field.tau2 + field2.tau2 + field2.sigma2),
                            replacement.cost = 1)
time.taken.11 <- proc.time()[3] - temp.time # 4 min.
temp.time <- proc.time()[3]
pgaspat.rc2 <- portfolio_rc(cutoffs2, CDF_breaks, pgaspat_lv$lv2,
                            sqrt(field.tau2 + field2.tau2 + field2.sigma2),
                            replacement.cost = 1)
time.taken.12 <- proc.time()[3] - temp.time # 4 min.
temp.time <- proc.time()[3]
damagespat.rc2 <- portfolio_rc(cutoffs2, CDF_breaks, damagespat_lv$lv2,
                               sqrt(field2.tau2), replacement.cost = 1)
time.taken.13 <- proc.time()[3] - temp.time # 4 min.
```

If we have exposure data aggregated up into grids, we can calculate the latent variable means per grid cell and use the `no.building` argument in `portfolio_rc` function to indicate the number of buildings per grid location so that we can multiply this with the estimated replacement costs per grid cell. Summing this up over all the grid cells gives us the estimated portfolio loss. Next, we match the loss per event to the SES to calculate annual losses:

```
# Compute annual losses and exceedance rates:
event_list <- colnames(oq_sim)[!(colnames(oq_sim) %in% c("loc_id", "lon", "lat",
                                                          "building_cat"))]
```

```
nonspat.al1 <- nonspat.al2 <- pgaspat.al1 <- pgaspat.al2 <-
damagespat.al1 <- damagespat.al2 <- rep(0, length(ses_list))
temp.time <- proc.time()[3]
for (i in 1:length(event_list)){
  ses_event <- event_ses_list$rlz_id[event_ses_list$event_id == as.numeric(event_list[i])]
  nonspat.al1[ses_list == ses_event] <-  nonspat.al1[ses_list == ses_event] +
                                         nonspat.rc1[i]
  nonspat.al2[ses_list == ses_event] <-  nonspat.al2[ses_list == ses_event] +
                                         nonspat.rc2[i]
  pgaspat.al1[ses_list == ses_event] <-  pgaspat.al1[ses_list == ses_event] +
                                         pgaspat.rc1[i]
  pgaspat.al2[ses_list == ses_event] <-  pgaspat.al2[ses_list == ses_event] +
                                         pgaspat.rc2[i]
  damagespat.al1[ses_list == ses_event] <-  damagespat.al1[ses_list == ses_event] +
                                            damagespat.rc1[i]
  damagespat.al2[ses_list == ses_event] <-  damagespat.al2[ses_list == ses_event] +
                                            damagespat.rc2[i]
  if ((i %% 1000)==0) { # reduce logging
    print(paste(i, "/", length(event_list), " done.", sep = ""))
  }
}
time.taken.14 <- proc.time()[3] - temp.time # 14 min.
```

This operation takes about 14 minutes. As a check, we compute the average annual loss and associated standard deviation for each model. The results for building category 1 and 2 are given in Table 1 and 2 respectively.

Table 1: Building category 1: Average annual losses (AAL) and associated standard deviations (sd) for the spatial ordinal model (Damage-spatial) and its submodels (Non-spatial, IM-spatial).

Model	AAL	sd
Non-spatial	16.60	20.70
IM-spatial	16.68	24.59
Damage-spatial	16.66	24.98

Table 2: Building category 2: Average annual losses (AAL) and associated standard deviations (sd) for the spatial ordinal model (Damage-spatial) and its submodels (Non-spatial, IM-spatial).

Model	AAL	sd
Non-spatial	18.78	22.51
IM-spatial	18.85	26.17
Damage-spatial	18.89	27.62

The models should produce similar average annual losses so that adding the spatial fields do not introduce bias. Similar to what was noted on p.20 of Silva (2019), "Uncertainty and Correlation in Seismic Vulnerability Functions of Building Classes", we also observe larger standard deviations associated with higher spatial correlation.

Based on the annual losses per SES year, we plot the rates of exceedance as shown in Figure 13 for the building categories separately and Figure 14 for the combined portfolio of building category 1 and 2.

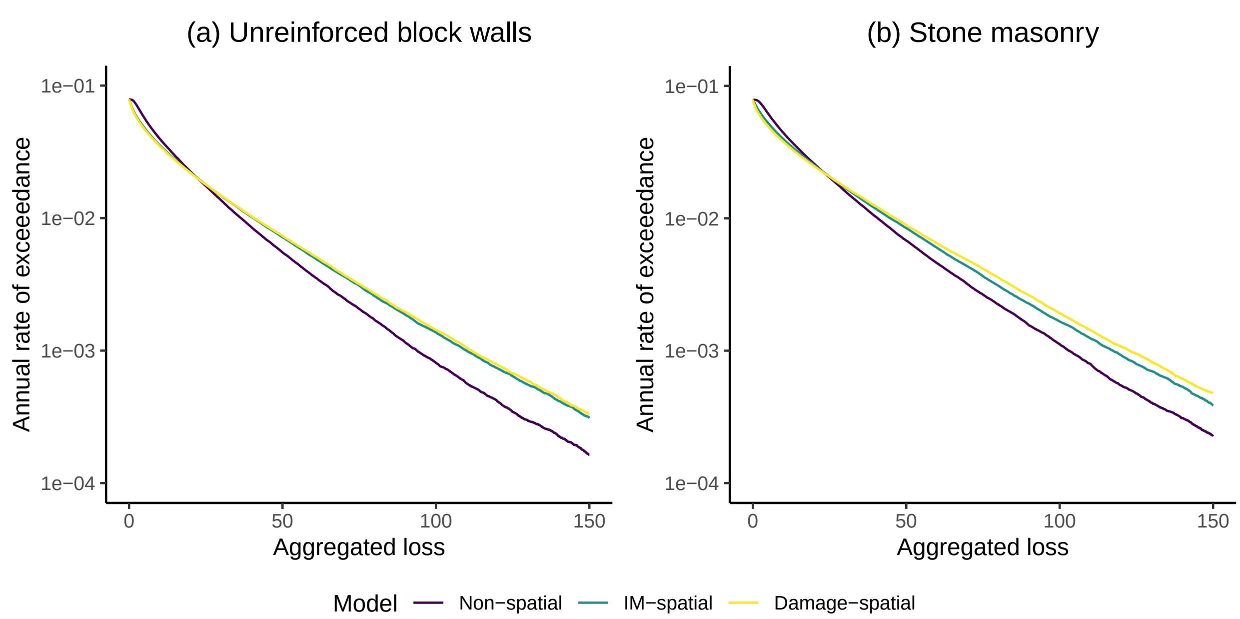

Figure 13: Loss exceedance curves for the portfolios of (a) 150 Category 1 (unreinforced block walls) and (b) 150 Category 2 (stone masonry) buildings within a 2km by 2km region in Port-au-Prince.

Next, we compute the Akaike Information Criterion (AIC) values for the three damage models.

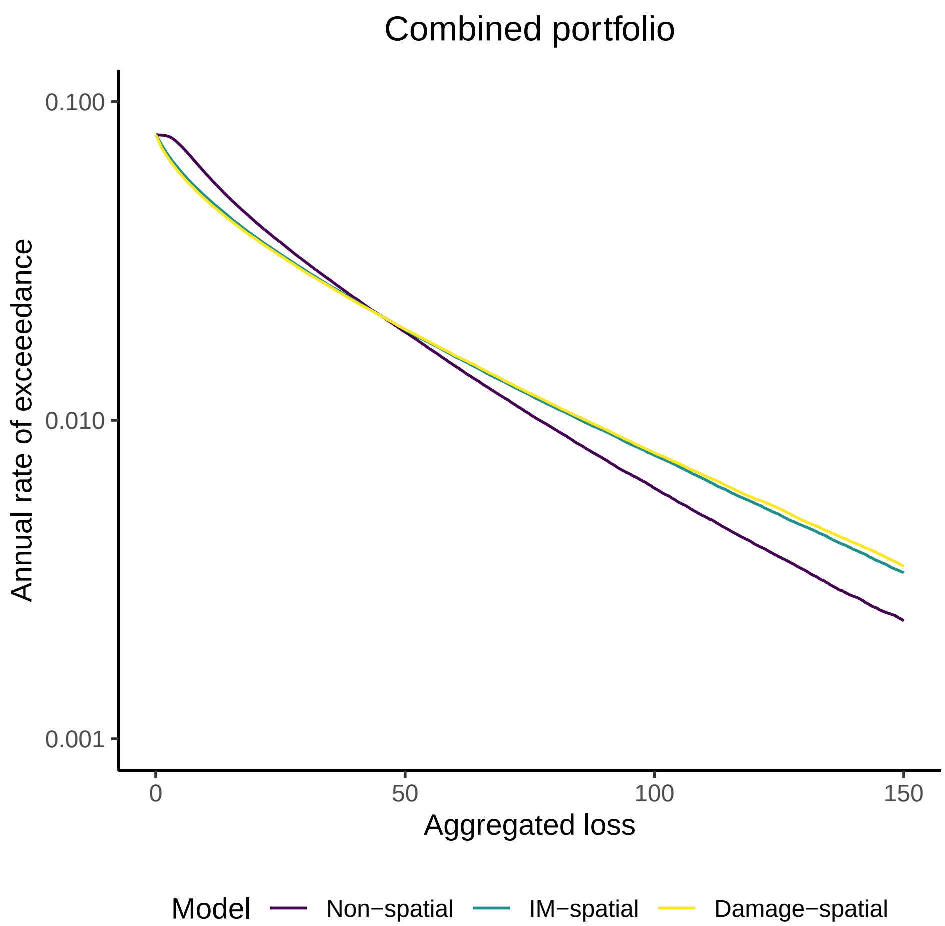

Figure 14: Loss exceedance curves for the portfolio containing 150 Category 1 (unreinforced block walls) and 150 Category 2 (stone masonry) buildings within a 2km by 2km region in Port-au-Prince.

```r
?submodel_aic
```

```r
temp.time <- proc.time()[3]
aic_val <- submodel_aic(data.1 = data.subset.1,
                        data.2 = data.subset.2, model.fit = demo_spatial_fit)
```

```
## Order of parameters:
##  [1] "c_factor1"   "c_factor2"   "log_phi"     "log_sigma_2" "log_tau_2"
##  [6] "log_tau1_2"  "log_tau2_2"  "field"       "log_slope1"  "log_slope2"
## Not matching template order:
##  [1] "log_phi"     "log_sigma_2" "log_tau_2"   "log_tau1_2"  "log_tau2_2"
##  [6] "log_slope1"  "log_slope2"  "c_factor1"   "c_factor2"   "field"
## Your parameter list has been re-ordered.
## (Disable this warning with checkParameterOrder=FALSE)
## Constructing atomic bessel_k_10
## Constructing atomic D_lgamma
## Constructing atomic invpd
## Constructing atomic pnorm1
## Constructing atomic bessel_k_10
## Constructing atomic D_lgamma
## Constructing atomic invpd
## Constructing atomic pnorm1
## Constructing atomic matmul
## Constructing atomic bessel_k_10
## Constructing atomic D_lgamma
## Constructing atomic invpd
## Constructing atomic pnorm1
## Constructing atomic matmul
## Optimizing tape... Done
## iter: 1  value: 750.5877 mgc: 0.5523639 ustep: 1
## iter: 2  value: 750.5877 mgc: 0.0003374215 ustep: 1
## iter: 3  mgc: 1.868453e-09
## Order of parameters:
## [1] "c_factor1"  "c_factor2"  "log_tau_2"  "log_tau1_2" "log_tau2_2"
## [6] "field"      "log_slope1" "log_slope2"
## Not matching template order:
## [1] "log_tau_2"  "log_tau1_2" "log_tau2_2" "log_slope1" "log_slope2"
## [6] "c_factor1"  "c_factor2"  "field"
## Your parameter list has been re-ordered.
## (Disable this warning with checkParameterOrder=FALSE)
## Constructing atomic invpd
## Constructing atomic pnorm1
## Constructing atomic invpd
## Constructing atomic pnorm1
## Constructing atomic matmul
## Constructing atomic invpd
## Constructing atomic pnorm1
## Constructing atomic matmul
## Optimizing tape... Done
## iter: 1  value: -1628.589 mgc: 6205591 ustep: 1
## iter: 2  value: -1628.589 mgc: 0.03504226 ustep: 1
## iter: 3  mgc: 1.44329e-15
```

```r
time.taken.15 <- proc.time()[3] - temp.time
aic.df <- data.frame("Model" = c("Non-spatial", "IM-spatial", "Damage-spatial"),
                     "AIC" = round(aic_val, 2))
```

Table 3: Akaike Information Criterion (AIC) values for the non-spatial, IM-spatial and Damage-spatial models.

	Model	AIC
Non-spatial	Non-spatial	1402.86
IM-spatial	IM-spatial	1369.72
Damage-spatial	Damage-spatial	1376.05

From Table 3, we see that for this simulated damage dataset, the Damage-spatial model has a lower Akaike Information Criterion (AIC) value than the Non-spatial submodel but not the IM-spatial model.