Hierarchical Spatial Interpolation of Extremes

Prerequisite to MS Stat Thesis

Siegfred Roi L. Codia

Contents

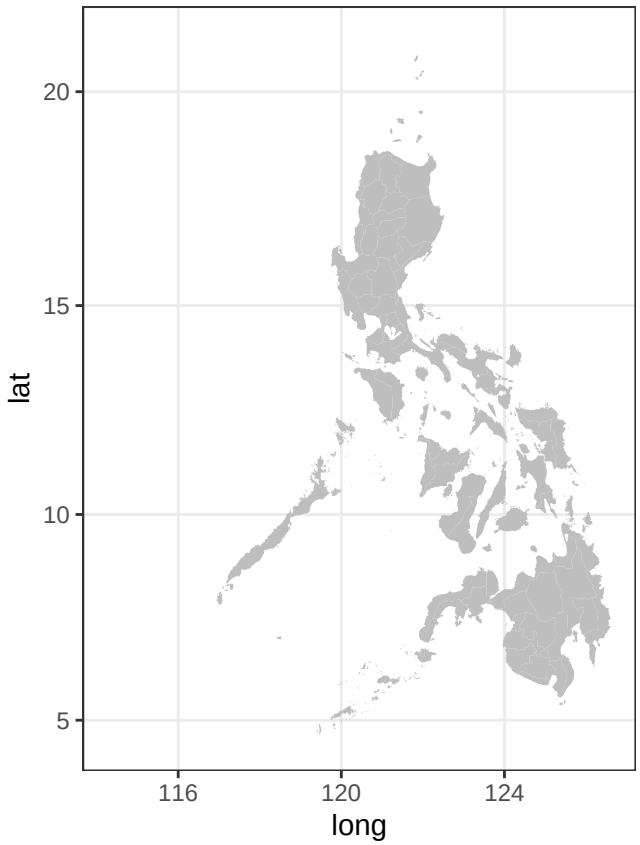
Importing Required Data

PH Provincial boundary data

Data obtained from NAMRIA

```
library(geojsonio)
library(broom)
provinces <- geojson_read("Data/Admin Boundaries/provinces.geojson",  what = "sp")
provinces <- tidy(provinces)
```

```
library(tidyverse)
ph_map <- ggplot() +
    geom_polygon(data = provinces,
                 aes( x = long, y = lat,
                      group = group), fill="gray") +
    theme_void() +
    coord_map()+
    theme_bw()
ph_map
```



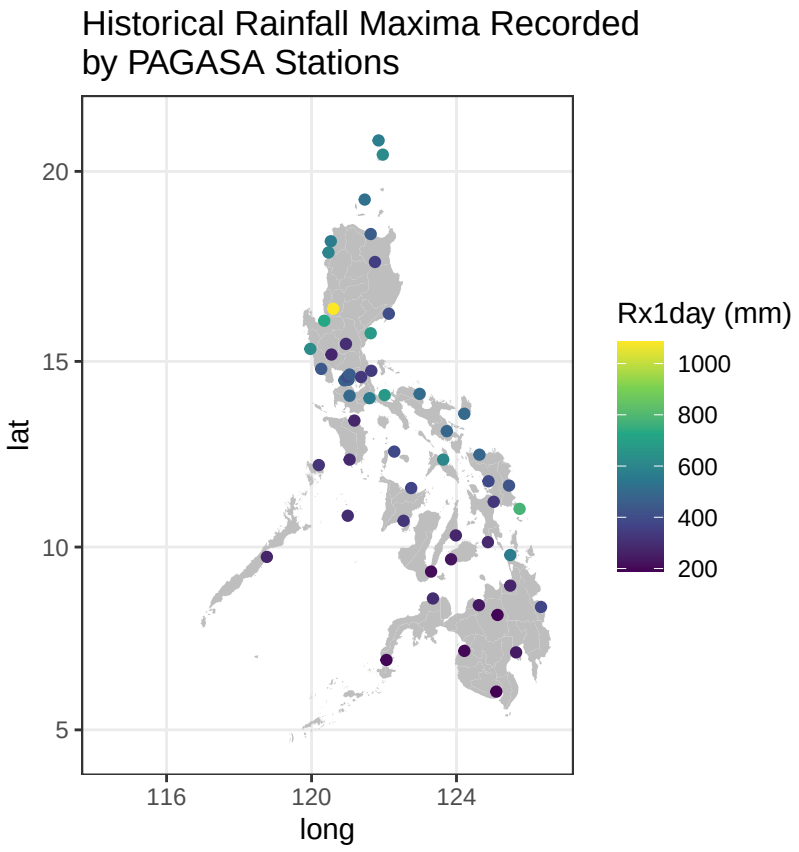
Rainfall data

```
stations <- read_csv("NOAA PH Weather Stations.csv")
max_data <- read_csv("max.spread.fixed.csv")
```

`max_data` contains annual rainfall maxima from each station obtained from NOAA GSOD server.

`stations` data contains coordinates and historical maxima of each stations as posted by PAGASA

```
ph_map +
        geom_point(data = stations,
                   aes(x = LON, y = LAT, color = `Rx1day (mm)`))+
        scale_color_viridis_c()+
    ggtitle("Historical Rainfall Maxima Recorded \nby PAGASA Stations")
```



Digital Elevation Model data of the Philippines

Data is obtained from The Humanitarian Data Exchange

```
library(raster)
```

```
## Loading required package: sp
```

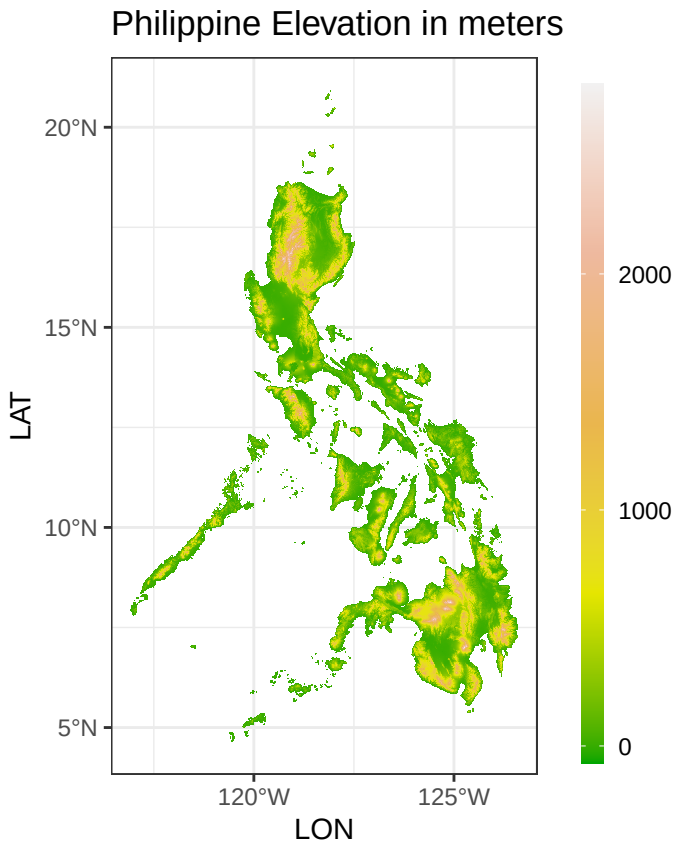
```
##
## Attaching package: 'raster'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
grd_file <- raster("Data/PHL_ALT/PHL_msk_alt.grd")
elev_df <- as.data.frame(rasterToPoints(grd_file))
colnames(elev_df) <- c("x", "y", "ALT")
```

```
library(ggplot2)
xlabs <- c(115, 120, 125, 130)
ylabs <- c(5,10,15,20)
ggplot()+
```

```
    geom_tile(data = elev_df, aes(x = x, y = y, fill = ALT))+
    scale_fill_gradientn(colors = terrain.colors(5))+
    coord_fixed()+
    theme_bw()+
    xlab("LON") + ylab("LAT")+
    scale_x_continuous(breaks = xlabs, labels = paste0(xlabs,'°W'))+
    scale_y_continuous(breaks = ylabs, labels = paste0(ylabs,'°N'))+
    theme(legend.title = element_blank(),
          legend.key.width = unit(0.3, "cm"),
          legend.key.height = unit(1.8, "cm"))+
    ggtitle("Philippine Elevation in meters")
```



Fitting pointwise GEVD distributions

We first load the EnvStats package and create custom function to extract parameters directly.

```
library(EnvStats)
gevd_pars <- function(x){egevd(x)$parameters}
```

Now performing the estimation per column in the data frame.

```
parameters <- max_data[, -1]%>%sapply(gevd_pars) %>% t() %>% data.frame()
```

Creating the data frame that contains the GEVD parameters per weather station...

```
parameters$"Station Name" <- row.names(parameters)
stations_pars <- full_join(stations,parameters, by = "Station Name")
stations_pars
```

```
## # A tibble: 55 x 11
##    Region `Administrative Prov~` `Station Name`   LAT   LON    ALT `Rx1day (mm)`
##    <chr>  <chr>                  <chr>          <dbl> <dbl>  <dbl>         <dbl>
##  1 I      Ilocos Norte           Laoag           18.2  121.    5.4          564.
##  2 I      Ilocos Sur             Sinait-Vigan    17.9  120.   58.1          594.
##  3 I      Pangasinan             Dagupan         16.1  120.    2            723.
##  4 CAR    Benguet                Baguio          16.4  121. 1510.         1086.
##  5 II     Batanes                Basco Radar     20.4  122.  167            616.
##  6 II     Batanes                Itbayat         20.8  122.  124            572.
##  7 II     Cagayan                Aparri          18.4  122.    3            453.
##  8 II     Cagayan                Calayan         19.3  121.   13            522.
##  9 II     Cagayan                Tuguegarao      17.6  122.   62            350.
## 10 III    Aurora                 Baler Radar     15.8  122.  173            676.
## # ... with 45 more rows, and 4 more variables: `Period Covered` <chr>,
## #   location <dbl>, scale <dbl>, shape <dbl>
```

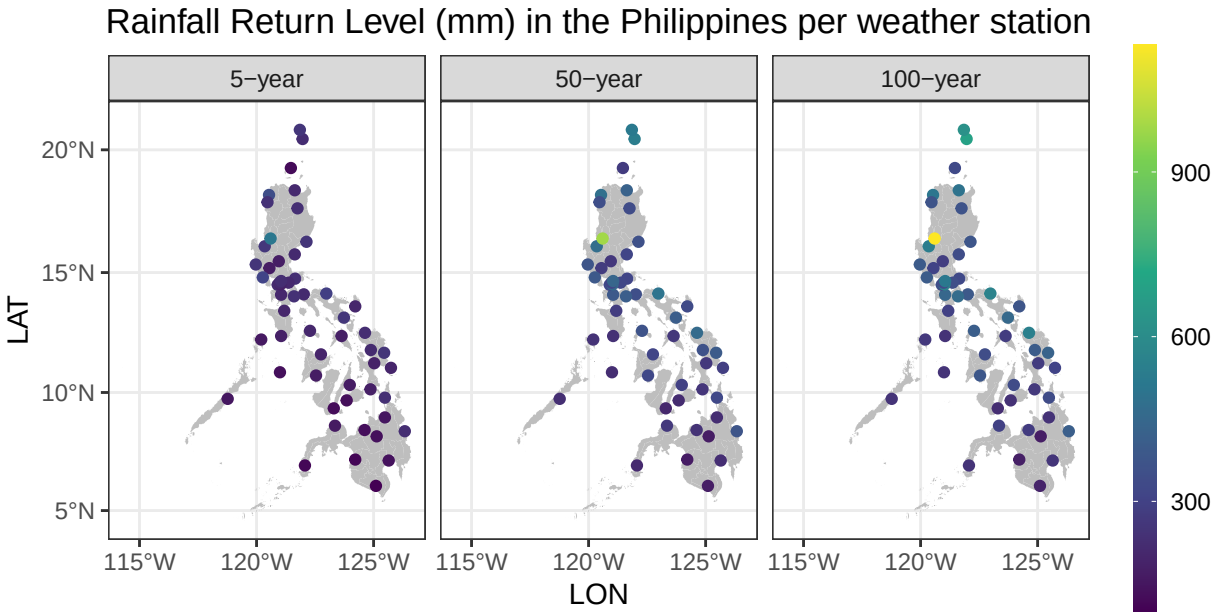
Visualization

```
library(tidyverse)
rp <- c("5-year", "50-year", "100-year")
rl_gathered <- stations_pars |>
    mutate("5-year" = qgevd(1-1/5, location = location,
                            scale = scale, shape = shape),
           "50-year" = qgevd(1-1/50, location = location,
                            scale = scale, shape = shape),
           "100-year" = qgevd(1-1/100, location = location,
                            scale = scale, shape = shape)) |>
    dplyr::select(`Station Name`, LAT, LON, all_of(rp)) |>
    pivot_longer(cols= rp,
                 names_to = "Return Period",
                 values_to = "Return Level") |>
    mutate("Return Period" = ordered(`Return Period`,
                                     levels = rp))
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(rp)` instead of `rp` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
ph_map +
    geom_point(data = rl_gathered, aes(x = LON, y = LAT, color = `Return Level`))+
    facet_wrap(.~`Return Period`)+
    scale_color_viridis_c()+
    theme_bw()+
    ggtitle("Rainfall Return Level (mm) in the Philippines per weather station")+
```

```
    xlab("LON") + ylab("LAT")+
    scale_x_continuous(breaks = xlabs, labels = paste0(xlabs,'°W'))+
    scale_y_continuous(breaks = ylabs, labels = paste0(ylabs,'°N'))+
    theme(plot.title=element_text(hjust=0.5),
          legend.title = element_blank(),
          legend.key.width = unit(0.3, "cm"),
          legend.key.height = unit(1.5, "cm"))
```

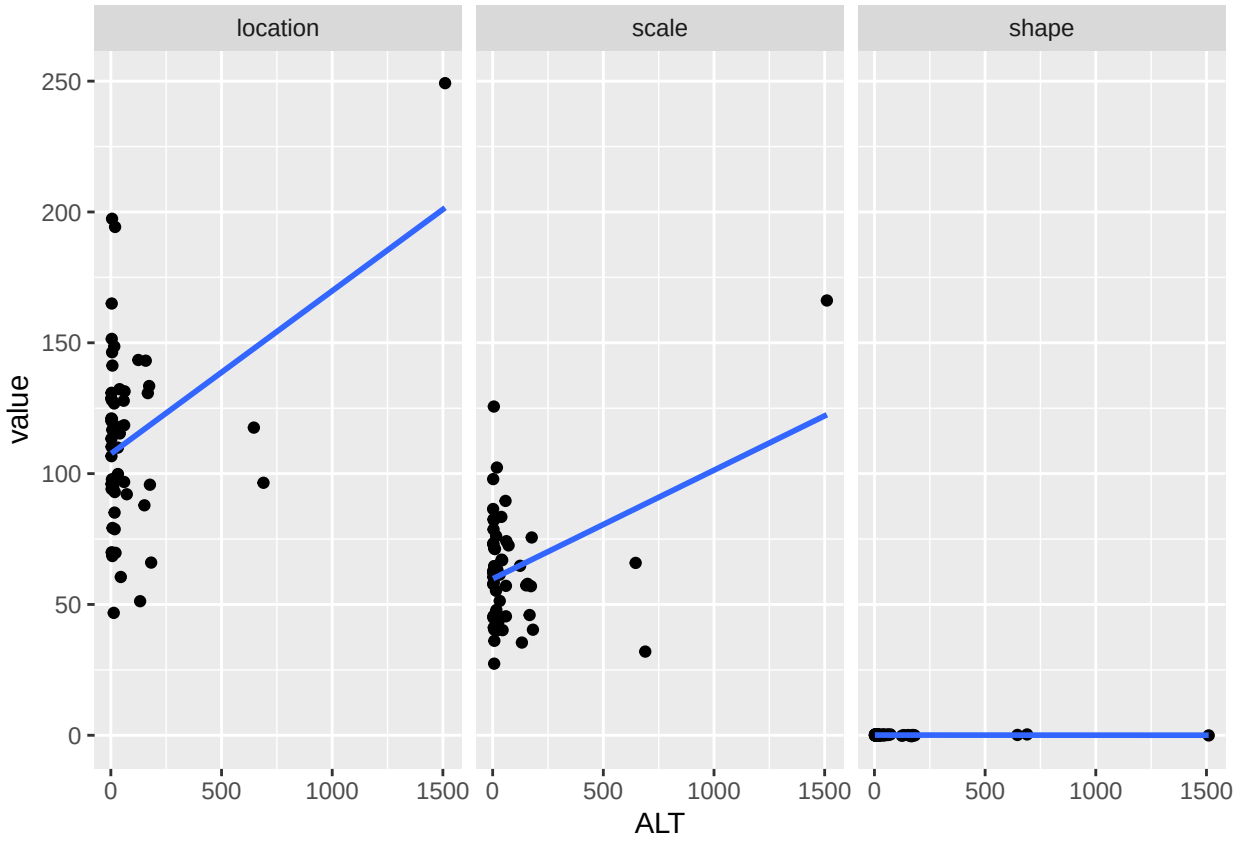


Relationships of Altitude and the GEV parameters

We also explore the relationship of Altitude and the estimated GEV parameters so we will have a basis if we will include them in the spatial interpolation models.

```
stations_pars |> pivot_longer(cols = c(location, scale, shape),
                              names_to = "parameter") |>
ggplot(aes(x = ALT, y = value))+
    geom_point()+
    geom_smooth(method = "lm", se = F)+
    facet_wrap(.~parameter)
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
cor(stations_pars[,c("ALT","location","scale","shape")])
```

```
##                  ALT  location     scale       shape
## ALT       1.00000000 0.3982270 0.4112922 -0.04962977
## location  0.39822700 1.0000000 0.8291777  0.18972890
## scale     0.41129219 0.8291777 1.0000000  0.24887884
## shape    -0.04962977 0.1897289 0.2488788  1.00000000
```

```
cor.test(stations_pars$ALT, stations_pars$location)
```

```
##
##  Pearson's product-moment correlation
##
## data:  stations_pars$ALT and stations_pars$location
## t = 3.1606, df = 53, p-value = 0.002603
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1486326 0.6001222
## sample estimates:
##      cor
## 0.398227
```

```
cor.test(stations_pars$ALT, stations_pars$scale)
```

```
##
##  Pearson's product-moment correlation
##
## data:  stations_pars$ALT and stations_pars$scale
## t = 3.285, df = 53, p-value = 0.001812
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1638763 0.6100265
## sample estimates:
##       cor
## 0.4112922
```

While the location and scale parameters have a significant positive correlation with the altitude, we still decide **not to include altitude in spatial interpolation** of the parameters since the correlation coefficient is not high enough (weak to moderate correlation only). To add, the following linear regression outputs have a very low R^2 values.

```
lm(location~ALT, data = stations_pars)%>%summary()
```

```
##
## Call:
## lm(formula = location ~ ALT, data = stations_pars)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -64.55 -23.15   2.49  20.07  89.50
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 107.56378    4.90493  21.930   <2e-16 ***
## ALT           0.06227    0.01970   3.161   0.0026 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.93 on 53 degrees of freedom
## Multiple R-squared:  0.1586, Adjusted R-squared:  0.1427
## F-statistic: 9.989 on 1 and 53 DF,  p-value: 0.002603
```

```
lm(scale~ALT, data = stations_pars)%>%summary()
```

```
##
## Call:
## lm(formula = scale ~ ALT, data = stations_pars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -56.438 -16.179   0.278  11.578  65.630
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 59.78341    3.14733  18.995  < 2e-16 ***
## ALT          0.04153    0.01264   3.285  0.00181 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.77 on 53 degrees of freedom
## Multiple R-squared:  0.1692, Adjusted R-squared:  0.1535
## F-statistic: 10.79 on 1 and 53 DF,  p-value: 0.001812
```

Spatial Interpolation

Recall that the `stations_pars` data frame contains longitude and latitude coordinates of each weather station, and the corresponding estimated GEVD parameters of the annual maxima. This will be the source data for interpolation.

Predictions will be made on the data frame `elev_df` which contains coordinates of Philippine land masses with cell size $0.0083°W \times 0.0083°N$.

We will convert the two data frames to SpatialDataFrame for better data manipulation using functions for spatial data analysis.

```
library(sp)
# converting the dataframes to spatial data frames
stations_pars_spdf <- SpatialPointsDataFrame(stations_pars,
                                             coords = stations[,c("LON", "LAT")])
elev_spdf <- SpatialPointsDataFrame(elev_df,
                                             coords = elev_df[,c("x", "y")])
```

For a short demonstration of hierarchical spatial interpolation of extremes, we will perform three spatial interpolation procedures of the GEV parameters μ, σ, ξ .

Inverse Distance Weighting

```
idw_loc   <- gstat::idw(location ~ 1, stations_pars_spdf, elev_spdf, idp = 2.0)
```

```
## [inverse distance weighted interpolation]
```

```
idw_scale <- gstat::idw(scale ~ 1, stations_pars_spdf, elev_spdf, idp = 2.0)
```

```
## [inverse distance weighted interpolation]
```

```
idw_shape <- gstat::idw(shape ~ 1, stations_pars_spdf, elev_spdf, idp = 2.0)
```

```
## [inverse distance weighted interpolation]
```

```
df_idw <- data.frame(x = elev_df$x, y = elev_df$y,
                     location = idw_loc$var1.pred,
                     scale = idw_scale$var1.pred,
                     shape = idw_shape$var1.pred)
```

Computation of return level values

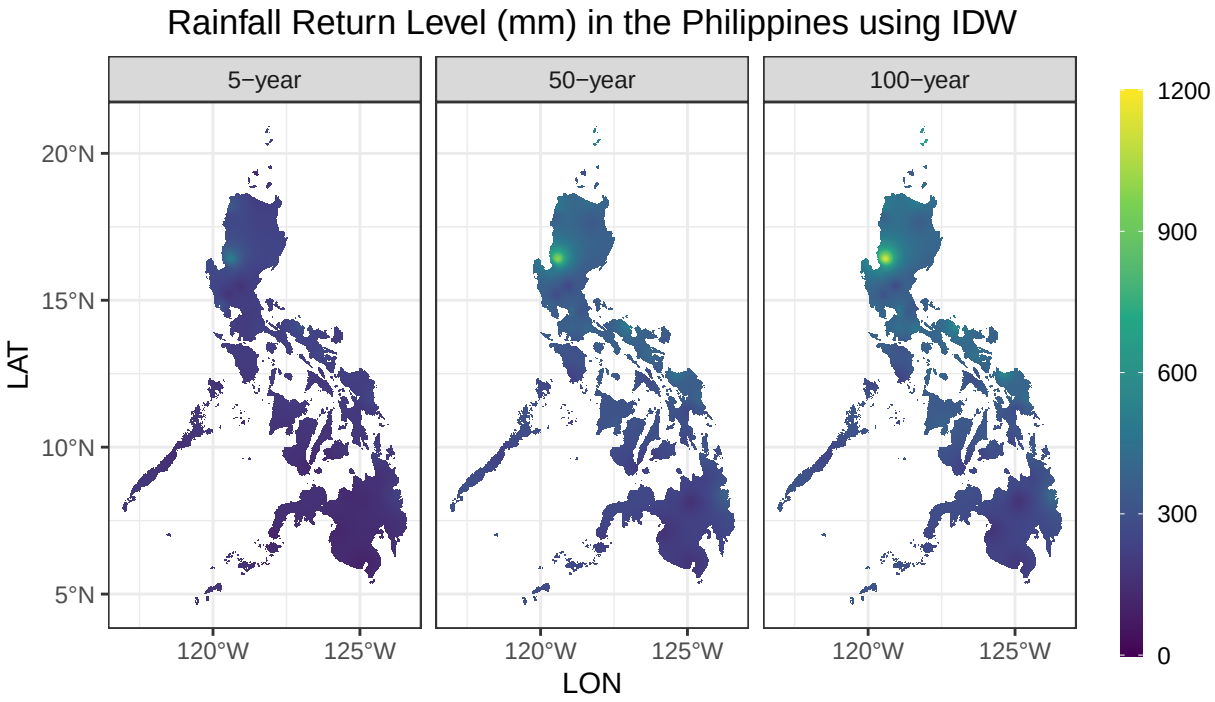
```
library(EnvStats)

rl_idw <- df_idw |>
    mutate("5-year" = qgevd(1-1/5, location = location,
                            scale = scale, shape = shape),
           "50-year" = qgevd(1-1/50, location = location,
                            scale = scale, shape = shape),
           "100-year" = qgevd(1-1/100, location = location,
                            scale = scale, shape = shape)) |>
    dplyr::select(x, y, all_of(rp)) |>
    pivot_longer(cols= rp,
                 names_to = "Return Period",
                 values_to = "Return Level") |>
    mutate("Return Period" = ordered(`Return Period`, levels = rp))
rl_idw
```

```
## # A tibble: 1,077,081 x 4
##        x     y `Return Period` `Return Level`
##    <dbl> <dbl> <ord>                     <dbl>
##  1  122.  20.9 5-year                     252.
##  2  122.  20.9 50-year                    516.
##  3  122.  20.9 100-year                   619.
##  4  122.  20.9 5-year                     253.
##  5  122.  20.9 50-year                    517.
##  6  122.  20.9 100-year                   620.
##  7  122.  20.8 5-year                     256.
##  8  122.  20.8 50-year                    520.
##  9  122.  20.8 100-year                   622.
## 10  122.  20.8 5-year                     256.
## # ... with 1,077,071 more rows
```

```
ggplot()+
    geom_tile(data = rl_idw, aes(x = x, y = y, fill = `Return Level`))+
    scale_fill_viridis_c(limits = c(0, 1200), oob = scales::squish)+
    coord_fixed()+
    theme_bw()+
    facet_wrap(.~`Return Period`)+
    ggtitle("Rainfall Return Level (mm) in the Philippines using IDW")+

    xlab("LON") + ylab("LAT")+
    scale_x_continuous(breaks = xlabs, labels = paste0(xlabs,'°W'))+
    scale_y_continuous(breaks = ylabs, labels = paste0(ylabs,'°N'))+
    theme(plot.title=element_text(hjust=0.5),
          legend.title = element_blank(),
          legend.key.width = unit(0.3, "cm"),
          legend.key.height = unit(1.5, "cm"))
```



Kriging

We will perform an Ordinary Kriging. To determine which type of variogram will be used, the following checks are performed:

```
library(gstat)
v_loc <- variogram(location ~ 1, stations_pars_spdf)

fit_loc_sph <- fit.variogram(v_loc, vgm(model = "Sph"))
```

```
## Warning in fit.variogram(v_loc, vgm(model = "Sph")): singular model in variogram
## fit
```

```
fit_loc_lin <- fit.variogram(v_loc, vgm(model = "Lin"))
```

```
## Warning in fit.variogram(v_loc, vgm(model = "Lin")): singular model in variogram
## fit
```

```
fit_loc_pow <- fit.variogram(v_loc, vgm(model = "Pow"))

pred_sph <- gstat::krige(location ~ 1,
                         stations_pars_spdf,
                         stations_pars_spdf,
                         model = fit_loc_sph)
```

```
## [using ordinary kriging]
```

```
pred_lin <- gstat::krige(location ~ 1,
                         stations_pars_spdf,
                         stations_pars_spdf,
                         model = fit_loc_lin)
```

```
## [using ordinary kriging]
```

```
pred_pow <- gstat::krige(location ~ 1,
                         stations_pars_spdf,
                         stations_pars_spdf,
                         model = fit_loc_pow)
```

```
## [using ordinary kriging]
```

```
Metrics::rmse(stations_pars_spdf$location, pred_sph$var1.pred)
```

```
## [1] 5.58661e-15
```

```
Metrics::rmse(stations_pars_spdf$location, pred_lin$var1.pred)
```

```
## [1] 7.664765e-15
```

```
Metrics::rmse(stations_pars_spdf$location, pred_pow$var1.pred)
```

```
## [1] 1.145716e-14
```

```
Metrics::mae(stations_pars_spdf$location, pred_sph$var1.pred)
```

```
## [1] 1.808654e-15
```

```
Metrics::mae(stations_pars_spdf$location, pred_lin$var1.pred)
```

```
## [1] 3.10055e-15
```

```
Metrics::mae(stations_pars_spdf$location, pred_pow$var1.pred)
```

```
## [1] 5.813531e-15
```

Spherical Model has the lowest RMSE and MAE. Hence, we will use Spherical Variogram Model Type in our Ordinary Kriging Spatial Interpolation of the parameters.

```
v_loc <- variogram(location ~ 1, stations_pars_spdf)
fit_loc <- fit.variogram(v_loc, vgm(model = "Sph"))
```

```
## Warning in fit.variogram(v_loc, vgm(model = "Sph")): singular model in variogram
## fit
```

```
krig_loc   <- gstat::krige(location ~ 1,
                           stations_pars_spdf,
                           elev_spdf,
                           model = fit_loc)
```

```
## [using ordinary kriging]
```

```
v_scale <- variogram(scale ~ 1, stations_pars_spdf)
fit_scale <- fit.variogram(v_scale, vgm(model = "Sph"))
krig_scale <- gstat::krige(scale ~ 1,
                           stations_pars_spdf,
                           elev_spdf,
                           model = fit_scale)
```

```
## [using ordinary kriging]
```

```
v_shape <- variogram(shape ~ 1, stations_pars_spdf)
fit_shape <- fit.variogram(v_shape, vgm(model = "Sph"))
krig_shape <- gstat::krige(shape ~ 1,
                           stations_pars_spdf,
                           elev_spdf,
                           model = fit_shape)
```

```
## [using ordinary kriging]
```

Creating data frame for interpolated values of the parameters.

```
df_krig <- data.frame(x = elev_df$x, y = elev_df$y,
                      location = krig_loc$var1.pred,
                      scale = krig_scale$var1.pred,
                      shape = krig_shape$var1.pred)
```

Computation of return level values

```
rl_krig <- df_krig |>
     mutate("5-year" = qgevd(1-1/5, location = location,
                             scale = scale, shape = shape),
            "50-year" = qgevd(1-1/50, location = location,
                             scale = scale, shape = shape),
            "100-year" = qgevd(1-1/100, location = location,
                             scale = scale, shape = shape)) |>
    dplyr::select(x, y, all_of(rp)) |>
    pivot_longer(cols= rp,
                 names_to = "Return Period",
                 values_to = "Return Level") |>
    mutate("Return Period" = ordered(`Return Period`, levels = rp))
rl_krig
```

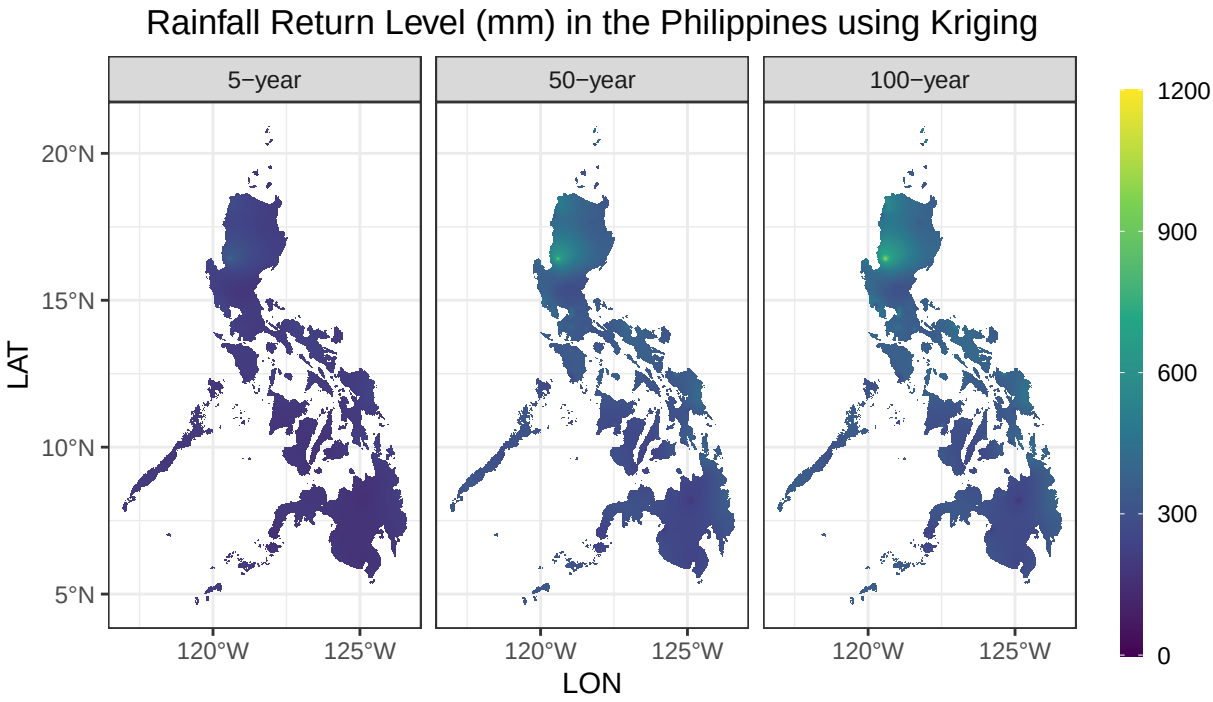
```
## # A tibble: 1,077,081 x 4
##        x     y ‘Return Period‘ ‘Return Level‘
```

```
##    <dbl> <dbl> <ord>                    <dbl>
##  1  122.  20.9 5-year                    207.
##  2  122.  20.9 50-year                   345.
##  3  122.  20.9 100-year                  382.
##  4  122.  20.9 5-year                    209.
##  5  122.  20.9 50-year                   361.
##  6  122.  20.9 100-year                  404.
##  7  122.  20.8 5-year                    222.
##  8  122.  20.8 50-year                   439.
##  9  122.  20.8 100-year                  514.
## 10  122.  20.8 5-year                    227.
## # ... with 1,077,071 more rows
```

Plotting return level values

```
ggplot()+
    geom_tile(data = rl_krig, aes(x = x, y = y, fill = `Return Level`))+
    scale_fill_viridis_c(limits = c(0, 1200), oob = scales::squish)+
    coord_fixed()+
    theme_bw()+
    facet_wrap(.~`Return Period`)+
    ggtitle("Rainfall Return Level (mm) in the Philippines using Kriging")+

    xlab("LON") + ylab("LAT")+
    scale_x_continuous(breaks = xlabs, labels = paste0(xlabs,'°W'))+
    scale_y_continuous(breaks = ylabs, labels = paste0(ylabs,'°N'))+
    theme(plot.title=element_text(hjust=0.5),
          legend.title = element_blank(),
          legend.key.width = unit(0.3, "cm"),
          legend.key.height = unit(1.5, "cm"))
```



Spline

```
library(fields)

spline_loc    <- Tps(coordinates(stations_pars_spdf),
                     stations_pars_spdf$location, lambda = 0)
spline_scale   <- Tps(coordinates(stations_pars_spdf),
                     stations_pars_spdf$scale, lambda = 0)
spline_shape   <- Tps(coordinates(stations_pars_spdf),
                     stations_pars_spdf$shape, lambda = 0)
```

```
df_spline <- elev_df |>
    mutate(location = predict(spline_loc, coordinates(elev_spdf)),
           scale  = predict(spline_scale, coordinates(elev_spdf)),
           shape = predict(spline_shape, coordinates(elev_spdf)))
```

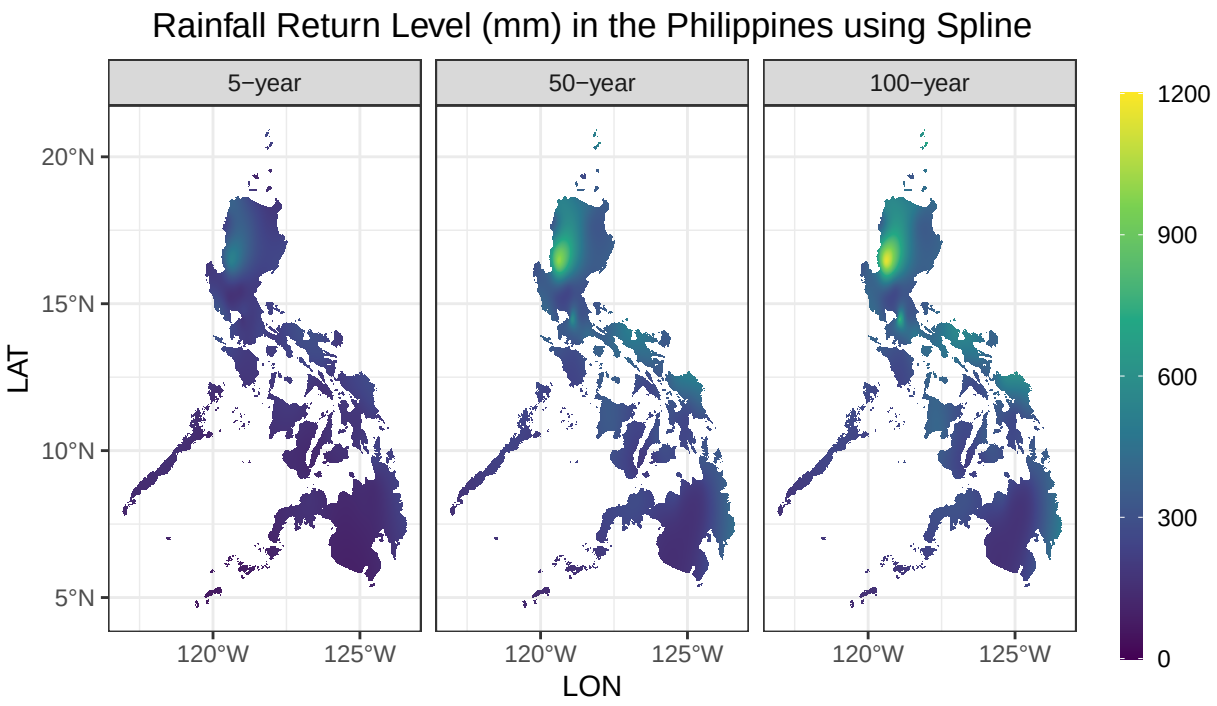
Computation of return level values

```
rl_spline <- df_spline |>
    mutate("5-year" = qgevd(1-1/5, location = location,
                            scale = scale, shape = shape),
           "50-year" = qgevd(1-1/50, location = location,
                            scale = scale, shape = shape),
```

```
          "100-year" = qgevd(1-1/100, location = location,
                             scale = scale, shape = shape)) |>
    dplyr::select(x, y, all_of(rp)) |>
    pivot_longer(cols= rp,
                 names_to = "Return Period",
                 values_to = "Return Level") |>
    mutate("Return Period" = ordered(`Return Period`, levels = rp))
```

```
ggplot()+
    geom_tile(data = rl_spline, aes(x = x, y = y, fill = `Return Level`))+
    scale_fill_viridis_c(limits = c(0, 1200), oob = scales::squish)+
    coord_fixed()+
    theme_bw()+
    facet_wrap(.~`Return Period`)+
    ggtitle("Rainfall Return Level (mm) in the Philippines using Spline")+

    xlab("LON") + ylab("LAT")+
    scale_x_continuous(breaks = xlabs, labels = paste0(xlabs,'°W'))+
    scale_y_continuous(breaks = ylabs, labels = paste0(ylabs,'°N'))+
    theme(plot.title=element_text(hjust=0.5),
          legend.title = element_blank(),
          legend.key.width = unit(0.3, "cm"),
          legend.key.height = unit(1.5, "cm"))
```



In the spline interpolation, from the graphs, there is a tendency that this model will over estimate when it comes to extrapolation of values outside the study region