Ordinary Kriging

Laura Delgado Bejarano

2024-10-22

Cleaning the space

```
rm(list = ls())    # Clear all objects
graphics.off()     # Close graphics devices
cat("\014")        # Clear console
```

Libraries

Installing and loading the libraries that are going to be used

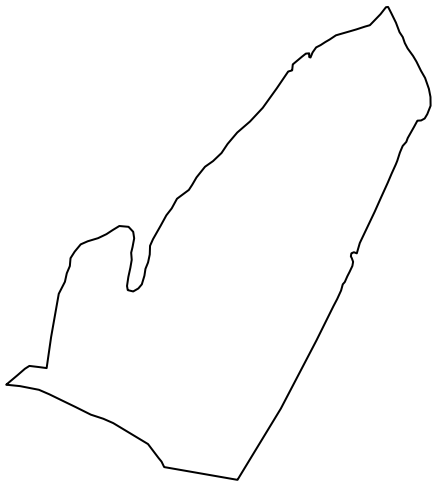
Loading the data

Load polygon of the area of study

```
poly <- st_read("00_Cartobase/01_Contornos/01_Paulinia/Contorno_Paulinia.shp")
```

```
## Reading layer 'Contorno_Paulinia' from data source
##   'G:\Mi unidad\02_Maestria\01_Projeito\00_Cartobase\01_Contornos\01_Paulinia\Contorno_Paulinia.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 1 feature and 0 fields
## Geometry type: POLYGON
## Dimension:     XYZ
## Bounding box:  xmin: 275598.9 ymin: 7487346 xmax: 277082.2 ymax: 7488999
## z_range:       zmin: 0 zmax: 0
## Projected CRS: WGS 84 / UTM zone 23S
```

```
plot(poly)
```



```
poly_sf <- st_zm(poly)
```

Create a clean and empty grid to interpolate

```
# Create base raster at 5 m resolution and convert to grid
r  <- raster(poly_sf, res = 5)          # base grid resolution (meters)
rp <- rasterize(poly_sf, r, 0)          # empty raster within polygon
grid <- as(rp, "SpatialPixelsDataFrame")
plot(grid)
```


```
proj4string(grid) <-CRS("+init=epsg:32723")
```

```
## Warning in CPL_crs_from_input(x): GDAL Message 1: +init=epsg:XXXX syntax is
## deprecated. It might return a CRS with a non-EPSG compliant axis order.
```

```
proj4string(grid)
```

```
## [1] "+proj=utm +zone=23 +south +datum=WGS84 +units=m +no_defs"
```

Load the csv with the data

Here is important to know if the csv are separed by , or ; and change it if necessary

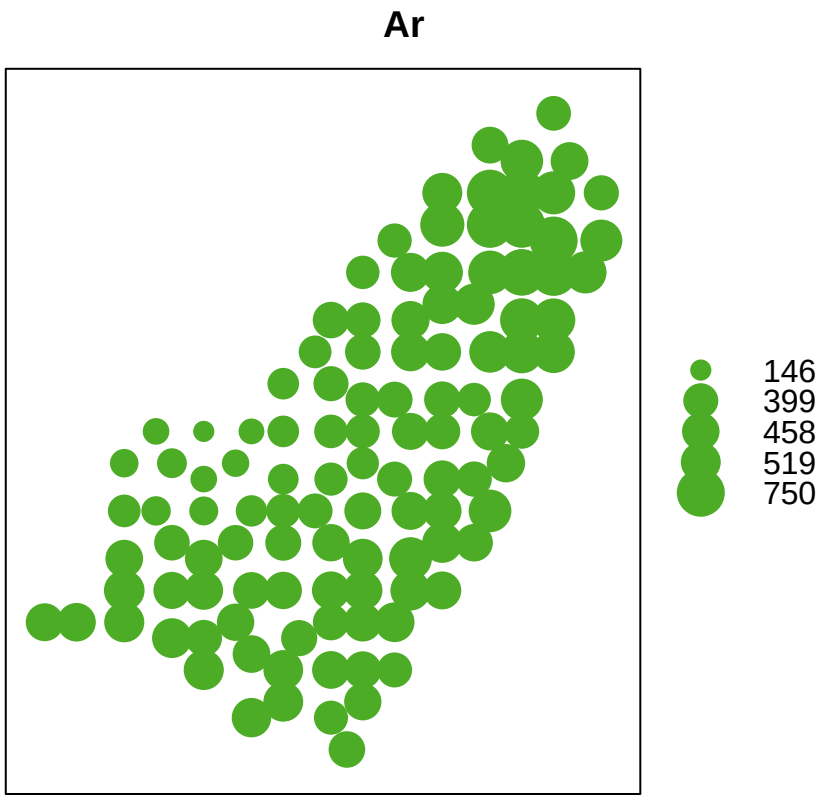
```r
#cargas los datos
original = data.frame(read.csv(file = "02_Cenarios_amostrais/04_CSVs/Paulinia_1am_cada1ha.csv",
                               header = TRUE, sep = ';'))
head(original)
```

```
##    pH  P   K CTC  V Argila        x       y
## 1 5.4 50 4.4  95 72    468 275619.8 7487681
## 2 5.8 75 4.6 102 78    484 275699.8 7487681
## 3 5.0 22 3.4  88 60    265 275819.8 7488081
## 4 4.6 18 2.8  70 53    348 275819.8 7487961
## 5 5.9 66 6.1 107 86    462 275819.8 7487841
## 6 5.5 51 5.0  94 78    537 275819.8 7487761
```

```r
dados = original[,c(7,8,6)] # select columns of interest (x,y,z)
head(dados)
```

```
##          x       y Argila
## 1 275619.8 7487681    468
## 2 275699.8 7487681    484
## 3 275819.8 7488081    265
## 4 275819.8 7487961    348
## 5 275819.8 7487841    462
## 6 275819.8 7487761    537
```

```r
dados = na.omit(dados)
names(dados) = c( "x", "y", "Ar") # ensure names x, y, variable
sp::coordinates(dados) = ~x+y
sp::bubble(dados, "Ar")
```



Geostatistics

Adjust the experimental semivariogram, here is necessary to define the initial values based on the experimental semivariogram: vgm(psill, model, range, nugget)

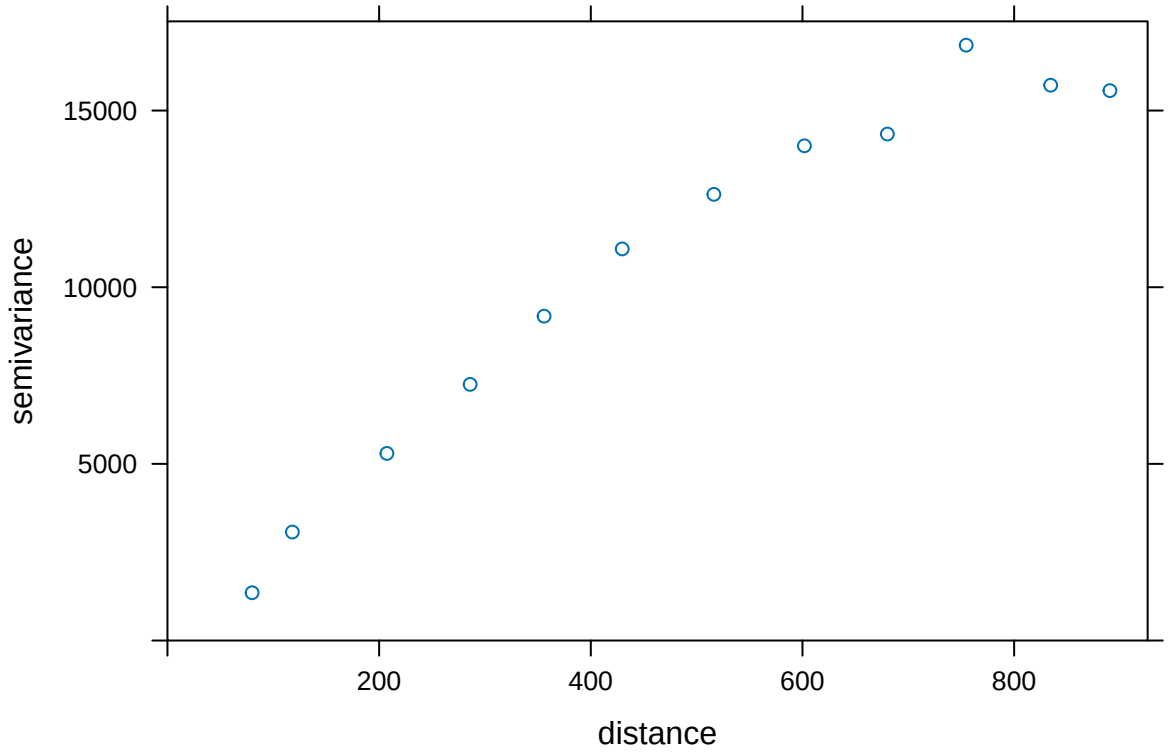
```
# Build gstat object
# Here is important to change to the variable that is going to be interpolated
g = gstat(formula = Ar ~ 1, data=dados)

# Distance helpers
print(max(dist(dados@coords))/2) #Cutoff
```

```
## [1] 905.0984
```

```
print(min(dist(dados@coords))) #Min distance to define the lags (width)
```

```
## [1] 79.991
```

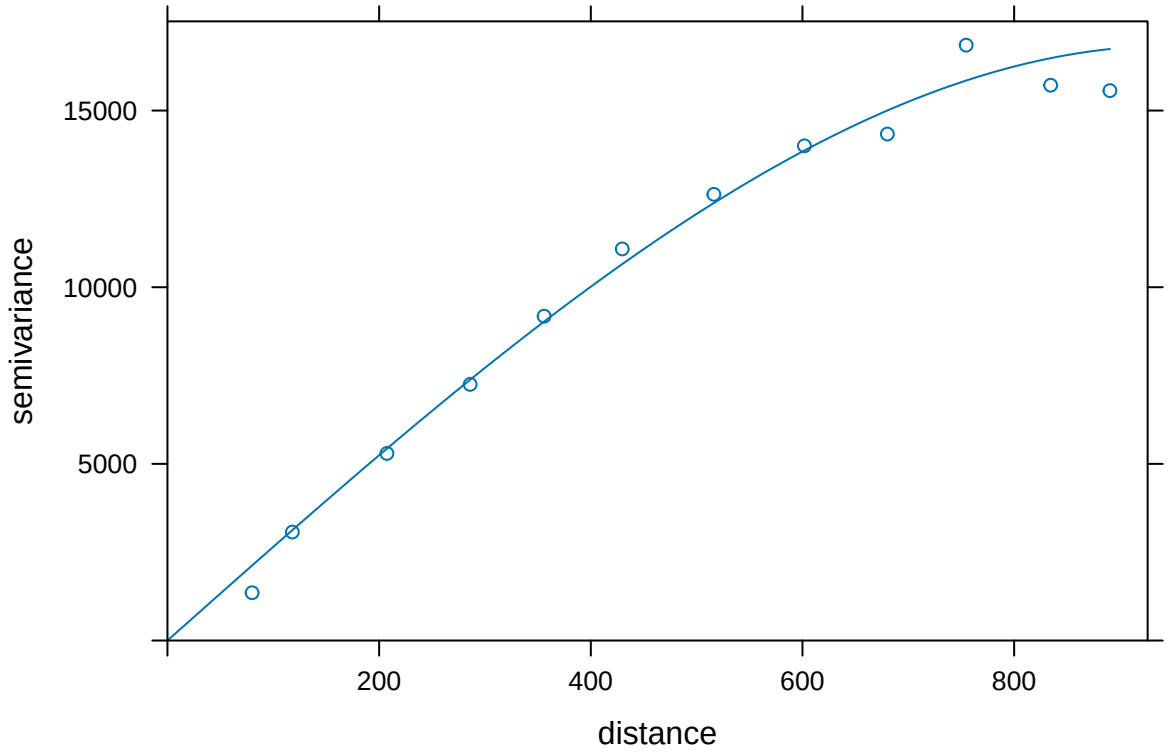
```
# Experimental variogram
var_exp = gstat::variogram(g, cutoff=905, width=80, cressie=F)
plot(var_exp)
```



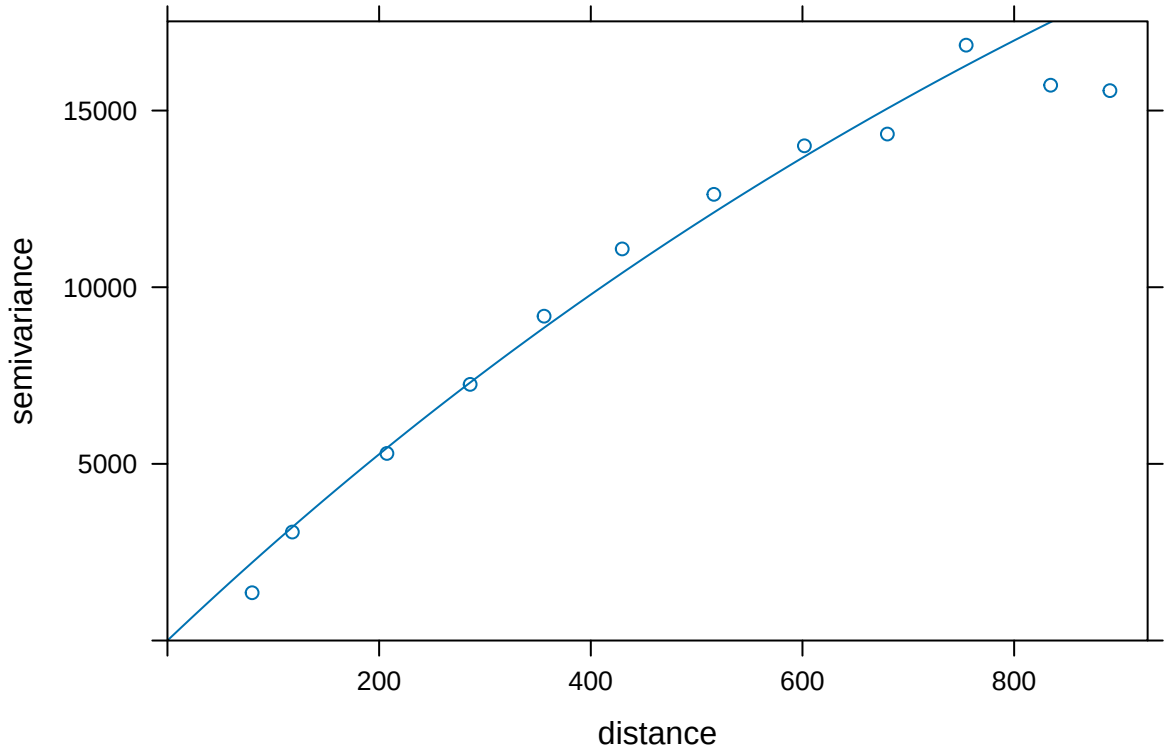

```r
var(dados$Ar) # Data variance (should be close to variogram sill)
```

```
## [1] 12874.14
```

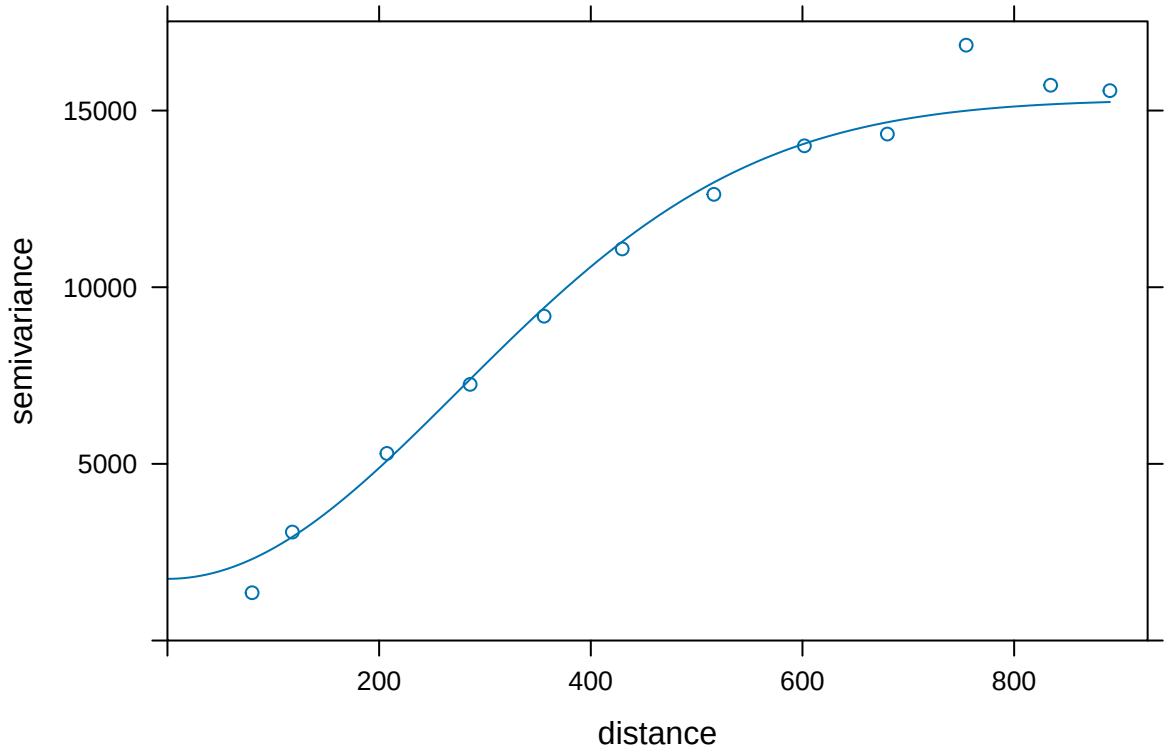
```r
# Spherical model
fit.sph = fit.variogram(var_exp, vgm(16000, "Sph", 700, 0))
plot(var_exp, fit.sph)
```




```
# Exponential model
fit.exp = fit.variogram(var_exp, vgm(16000, "Exp", 700, 0))
plot(var_exp, fit.exp)
```




```
# Gaussian model
fit.gauss = fit.variogram(var_exp, vgm(16000, "Gau", 700, 0))
plot(var_exp, fit.gauss)
```



LOOCV

Cross validation to select the best fit model to krige, if is going to be a high density kriging (high number of samples as ECa or productivity) is necessary to add nfold=10 and nmax=200

```
# Spherical
xvalid.sph <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.sph)
lm_sph     <- lm(xvalid.sph$var1.pred ~ xvalid.sph$observed)
r2_sph     <- summary(lm_sph)$r.squared
rmse_sph   <- hydroGOF::rmse(xvalid.sph$var1.pred, xvalid.sph$observed)
slope_sph  <- lm_sph$coefficients[2]


# Exponential
xvalid.exp <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.exp)
lm_exp     <- lm(xvalid.exp$var1.pred ~ xvalid.exp$observed)
r2_exp     <- summary(lm_exp)$r.squared
rmse_exp   <- hydroGOF::rmse(xvalid.exp$var1.pred, xvalid.exp$observed)
slope_exp  <- lm_exp$coefficients[2]


# Gaussian
xvalid.gau <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.gauss)
lm_gau     <- lm(xvalid.gau$var1.pred ~ xvalid.gau$observed)
r2_gau     <- summary(lm_gau)$r.squared
```

```
rmse_gau   <- hydroGOF::rmse(xvalid.gau$var1.pred, xvalid.gau$observed)
slope_gau  <- lm_gau$coefficients[2]
```

Creating the metrics table

```
df.r2    <- data.frame(r2_sph, r2_exp, r2_gau)
df.rmse  <- data.frame(rmse_sph, rmse_exp, rmse_gau)
df.slope <- data.frame(slope_sph, slope_exp, slope_gau)

temp <- data.frame(cbind(t(df.r2), t(df.rmse), t(df.slope)))
colnames(temp) <- c("R2", "RMSE", "slope")

rnames <- gsub("r2_", "", rownames(temp))
rownames(temp) <- rnames

print(temp)
```

```
##            R2     RMSE     slope
## sph 0.8253346 47.26145 0.8397348
## exp 0.8281230 46.86623 0.8375304
## gau 0.8383710 45.42464 0.8355115
```

Taking theorical parameters - it has to change for the best model

```
# Extract nugget, partial sill, total sill, and range
nugget        <- fit.gauss$psill[1]
partial_sill  <- fit.gauss$psill[2]
sill          <- partial_sill + nugget
range         <- fit.gauss$range[2]

# Spatial Dependence Index (SDI) as percentage of partial sill over total sill
SDI <- (partial_sill / sill) * 100

# Classify SDI into categories
Class <- ifelse(SDI < 20, "Very low",
         ifelse(SDI < 40, "Low",
         ifelse(SDI < 60, "Medium",
         ifelse(SDI < 80, "High", "Very high"))))

# Build summary table (note: mixing numeric and character will coerce to character)
table_params <- data.frame(
  Parameter = c("Nugget", "Sill", "Range", "SDI%", "Class"),
  Value     = c(nugget, sill, range, SDI, Class),
  stringsAsFactors = FALSE
)

table_params
```

```
##   Parameter            Value
## 1    Nugget 1745.38362991187
## 2      Sill 15315.2789519115
## 3     Range 389.854507343428
```

```
## 4      SDI% 88.6036445343751
## 5     Class         Very high
```

#Kriging

Doing the kriging with the selected model

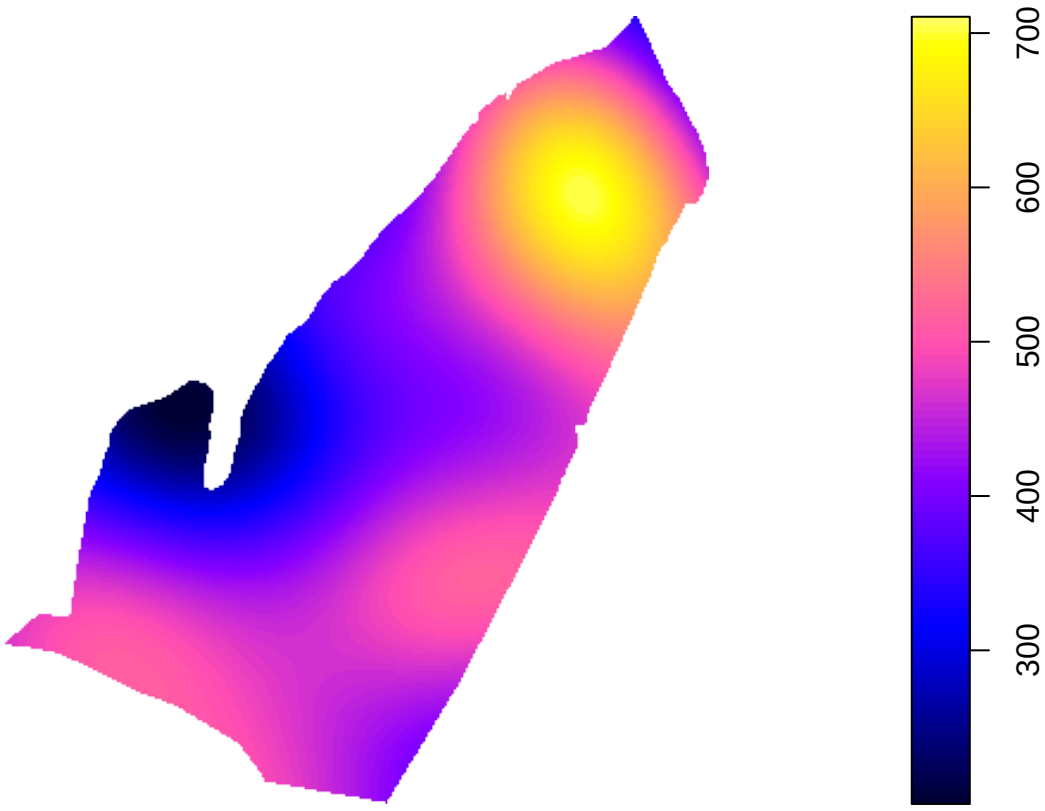
```
# Kriging with the selected model
proj4string(dados) <- CRS("+init=epsg:32723")
proj4string(dados)
```

```
## [1] "+proj=utm +zone=23 +south +datum=WGS84 +units=m +no_defs"
```

```
mapa = krige(Ar ~ 1, dados, grid, model = fit.gauss)
```

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

```
plot(mapa)
```



Exporting the raster

```
mapaRaster <- raster(mapa)

filename<-'13_GitHub/Ar_OK_1cada1_Paulinia.tiff'
writeRaster(mapaRaster, filename , format = 'GTiff', overwrite = T)
```

Graphic with ggplot

```
ggplot() +
  geom_raster(data = as.data.frame(mapa), aes(fill = var1.pred, x = x, y = y)) +
  scale_fill_viridis_c() +
    annotation_north_arrow(which_north = "grid",height = unit(1, "cm"),
                           width = unit(0.9, "cm"),
                           pad_x = unit(0.5, "cm"),
                           pad_y = unit(10, "cm"),
                           style=north_arrow_fancy_orienteering())+
  annotation_scale(location = "bl", width_hint = 0.2)+
  theme_bw()
```

```
## Using plotunit = 'm'
```