Ordinary Kriging with REML

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

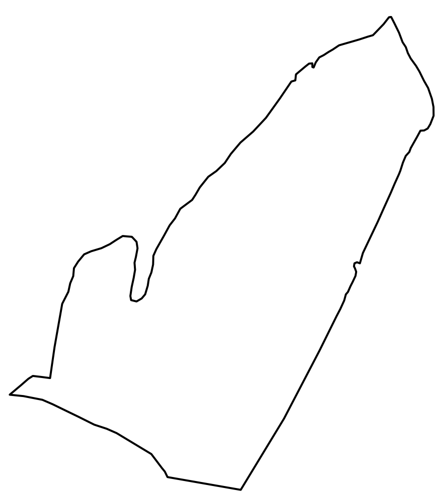
Loading the data

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)
```



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

Create a clean and empty grid to interpolate

```r
# 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)
```


```r
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.
```

```r
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

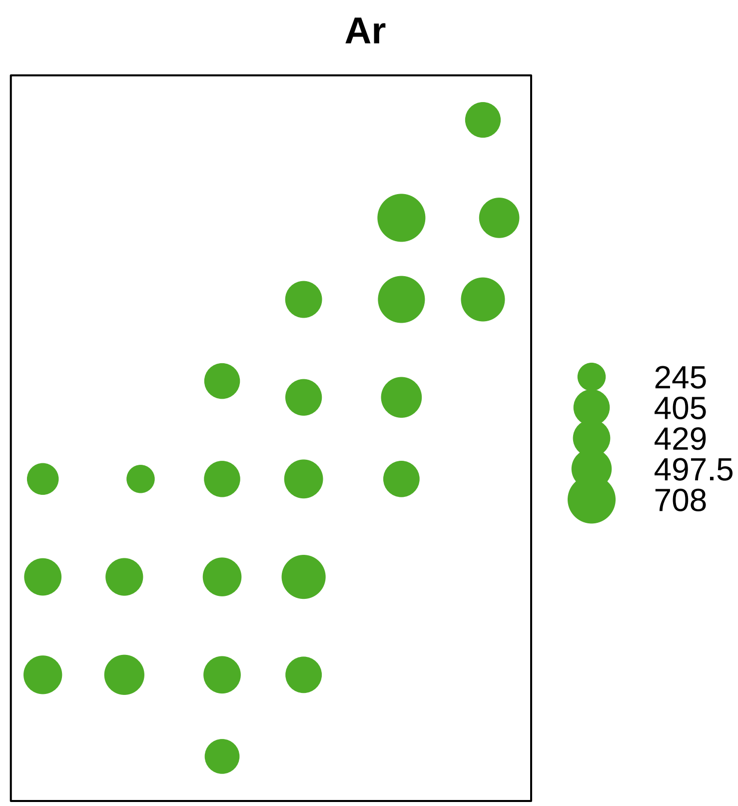
```r
#cargas los datos
original = data.frame(read.csv(file = "02_Cenarios_amostrais/04_CSVs/Paulinia_0.2am_cada1ha.csv",
                               header = TRUE, sep = ';'))
head(original)
```

```
##    pH  P   K CTC V. Argila        x       y
## 1 5.1 32 3.5  81 73    310 275859.8 7488081
## 2 5.8 39 3.7  96 81    429 275859.8 7487841
## 3 5.1 23 2.9  90 72    459 275859.8 7487601
## 4 6.0 42 3.7 100 84    435 276059.8 7487841
## 5 5.6 87 4.9 101 76    495 276059.8 7487601
## 6 5.9 60 4.2 102 85    245 276099.8 7488081
```

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

```
##          x       y Argila
## 1 275859.8 7488081    310
## 2 275859.8 7487841    429
## 3 275859.8 7487601    459
## 4 276059.8 7487841    435
## 5 276059.8 7487601    495
## 6 276099.8 7488081    245
```

```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 - first using MoM to get the initial values to use geoR in the REML model for 50 - 100 samples (Kerry and Oliver, 2007)

```
# Experimental variogram (MoM) and geodata prep
Variable<-"Ar"
data_geo <- as.geodata(dados, data.col = which(names(dados) == Variable))

# Build gstat object for the target variable
g = gstat(formula = Ar ~ 1, data=dados) #Change name of variable
# Useful distance helpers to design variogram cutoff/lag width
C<-print(max(dist(dados@coords))/2);C #Cutoff
```

```
## [1] 868.3342
```

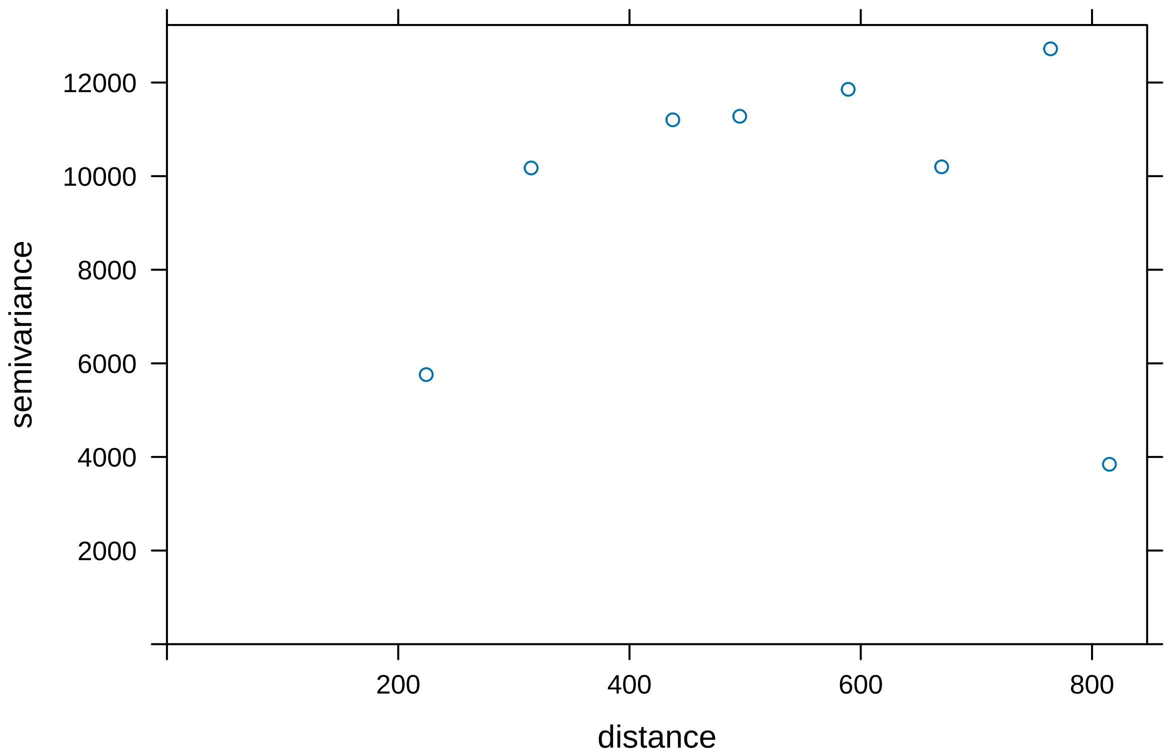
```
## [1] 868.3342
```

```
W<-print(min(dist(dados@coords)));W #Width
```

```
## [1] 199.992
```

```
## [1] 199.992
```

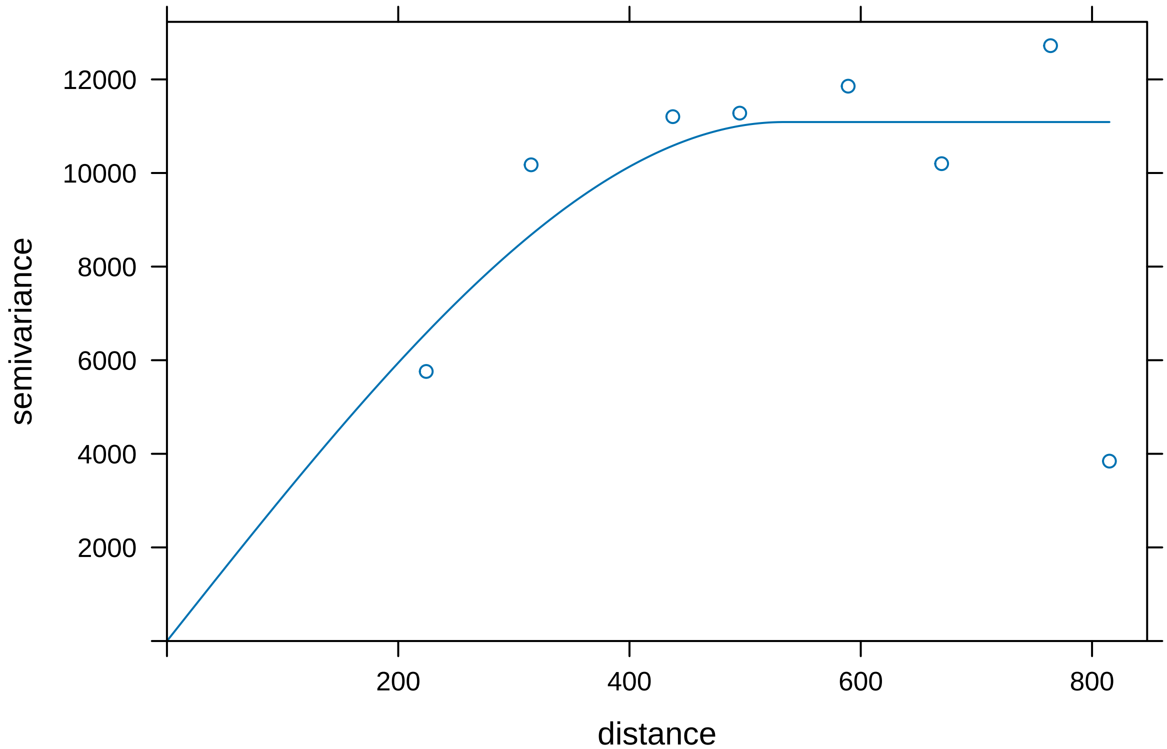
```r
var_exp = gstat::variogram(g, cutoff=C, width=89, cressie=F)
plot(var_exp)
```



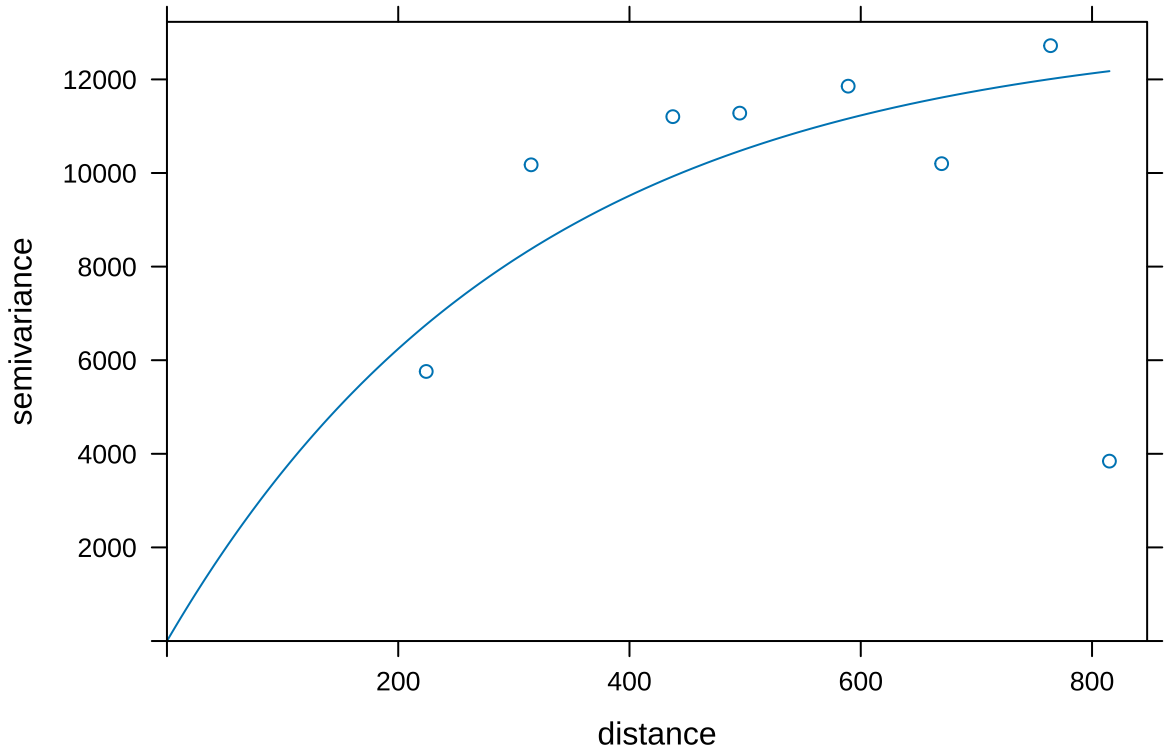

```r
# Initial MoM fits (keep if you want to compare visually)
fit.sph1   <- fit.variogram(var_exp, vgm(11800, "Sph", 300, 200))
fit.exp1   <- fit.variogram(var_exp, vgm(11800, "Exp", 300, 200))
fit.gauss1 <- fit.variogram(var_exp, vgm(11800, "Gau", 300, 200))
```

```
## Warning in fit.variogram(var_exp, vgm(11800, "Gau", 300, 200)): No convergence
## after 200 iterations: try different initial values?
```

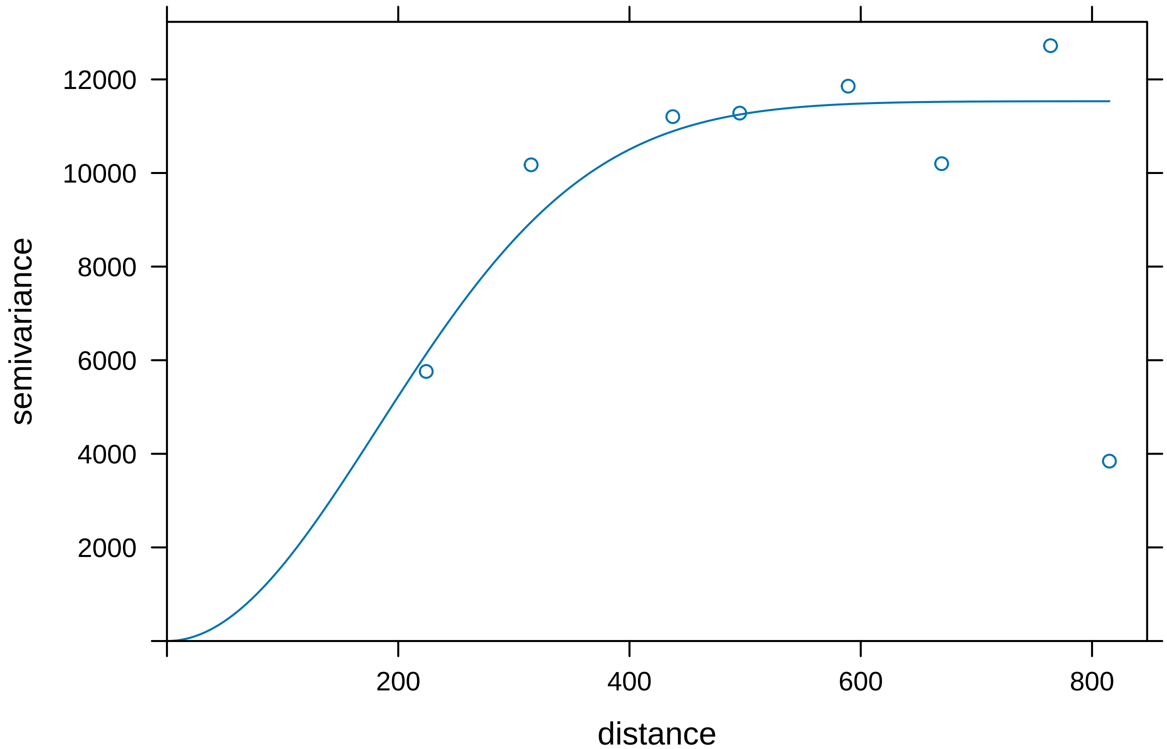
```r
plot(var_exp, fit.sph1)
```




```
plot(var_exp, fit.exp1)
```




```
plot(var_exp, fit.gauss1)
```




```r
# Cross-validation per model

# Spherical model CV
xvalid.sph <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.sph1)
xvalid_df_sp <- as.data.frame(xvalid.sph)
lm_sph <- lm(xvalid.sph$var1.pred ~ xvalid.sph$observed)

# Metrics
r2_sph   <- summary(lm_sph)$r.squared          # R²
rmse_sph <- hydroGOF::rmse(xvalid.sph$var1.pred, xvalid.sph$observed)  # RMSE
slope_sph <- lm_sph$coefficients[2]            # regression slope

# Exponential model CV
xvalid.exp <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.exp1)
xvalid_df_ex <- as.data.frame(xvalid.exp)
lm_exp <- lm(xvalid.exp$var1.pred ~ xvalid.exp$observed)

# Metrics
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 model CV
xvalid.gau <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.gauss1)
xvalid_df_ga <- as.data.frame(xvalid.gau)
lm_gau <- lm(xvalid.gau$var1.pred ~ xvalid.gau$observed)
```

```r
# Metrics
r2_gau   <- summary(lm_gau)$r.squared
rmse_gau <- hydroGOF::rmse(xvalid.gau$var1.pred, xvalid.gau$observed)
slope_gau <- lm_gau$coefficients[2]


# Build summary table for the CV performance

df.r2    <- data.frame(r2_sph, r2_exp, r2_gau)          # R² per model
df.rmse  <- data.frame(rmse_sph, rmse_exp, rmse_gau)    # RMSE per model
df.slope <- data.frame(slope_sph, slope_exp, slope_gau) # slope per model

# Combine into a single table
temp <- data.frame(cbind(t(df.r2), t(df.rmse), t(df.slope)))
colnames(temp) <- c("R2", "RMSE", "slope")

# Clean row names (remove the 'r2_' prefix)
rnames <- gsub("r2_", "", rownames(temp))
rownames(temp) <- rnames

print(temp)
```

```
##            R2     RMSE     slope
## sph 0.5703577 70.46190 0.4580396
## exp 0.4391114 79.34631 0.3667608
## gau 0.6198053 65.23336 0.5595322
```

REML semivariogram

Based on the best model in MoM we use the initial parameters to adjust the REML kriging - When this adjustment is done the lags are not fitted to the model It is just a visual guide but it isnt equal to the MoM adjustment

```r
# ---------------------------
# Selected model (from MoM)
# ---------------------------
co    <- fit.gauss1$psill[1]      # nugget from MoM spherical fit
psill <- fit.gauss1$psill[2]      # partial sill (MoM)
sill  <- fit.gauss1 + co             # total sill
```

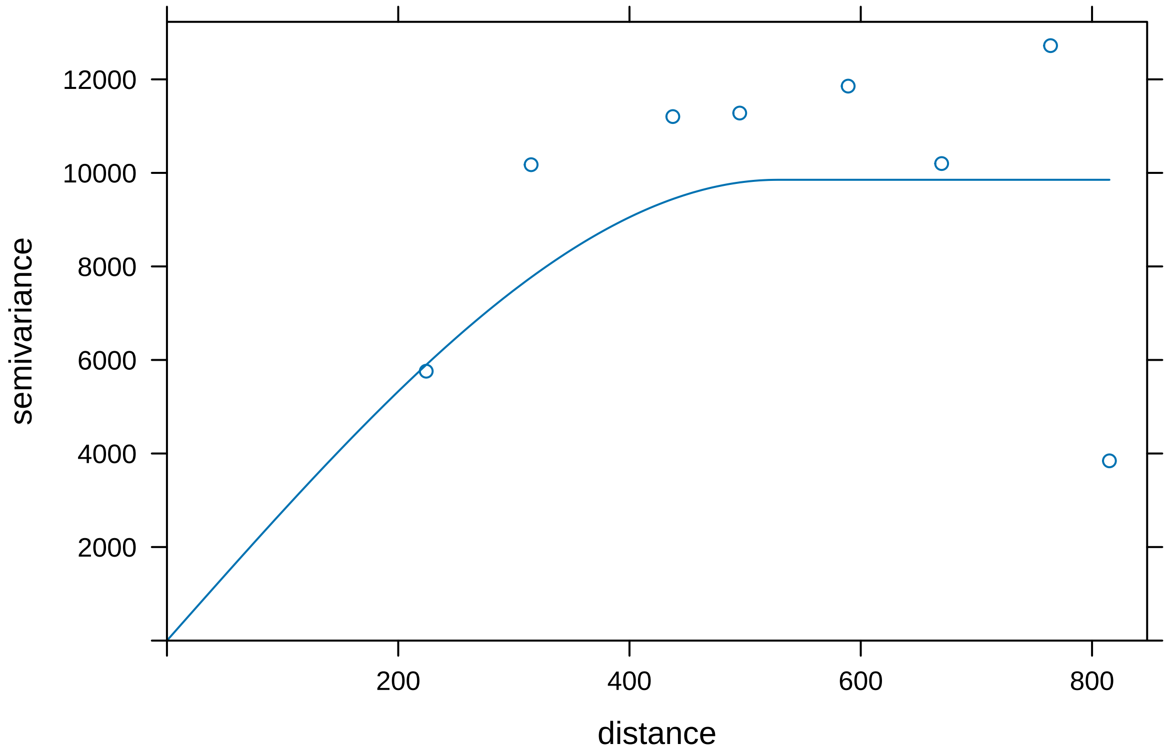
```
## Warning in Ops.factor(left, right): '+' not meaningful for factors
```

```r
range <- fit.gauss1$range[2]      # range (MoM)

# ------------------------------------------
# REML fit (geoR) -> convert to gstat::vgm
# ------------------------------------------

# Spherical model (REML)
fit_sph <- geoR::likfit(
  data_geo,
```

```r
  ini.cov.pars = c(psill, range),  # c(sigmasq, phi)
  nugget       = co,
  fix.nugget   = FALSE,
  lik.method   = "REML",
  cov.model    = "spherical"
)
```

```
## kappa not used for the spherical correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
fit.sph <- vgm(
  psill  = fit_sph$sigmasq,   # partial sill (sigmasq)
  model  = "Sph",             # Model: Spherical
  range  = fit_sph$phi,       # range (phi)
  nugget = fit_sph$tausq      # nugget (tausq)
)

plot(var_exp, fit.sph)
```

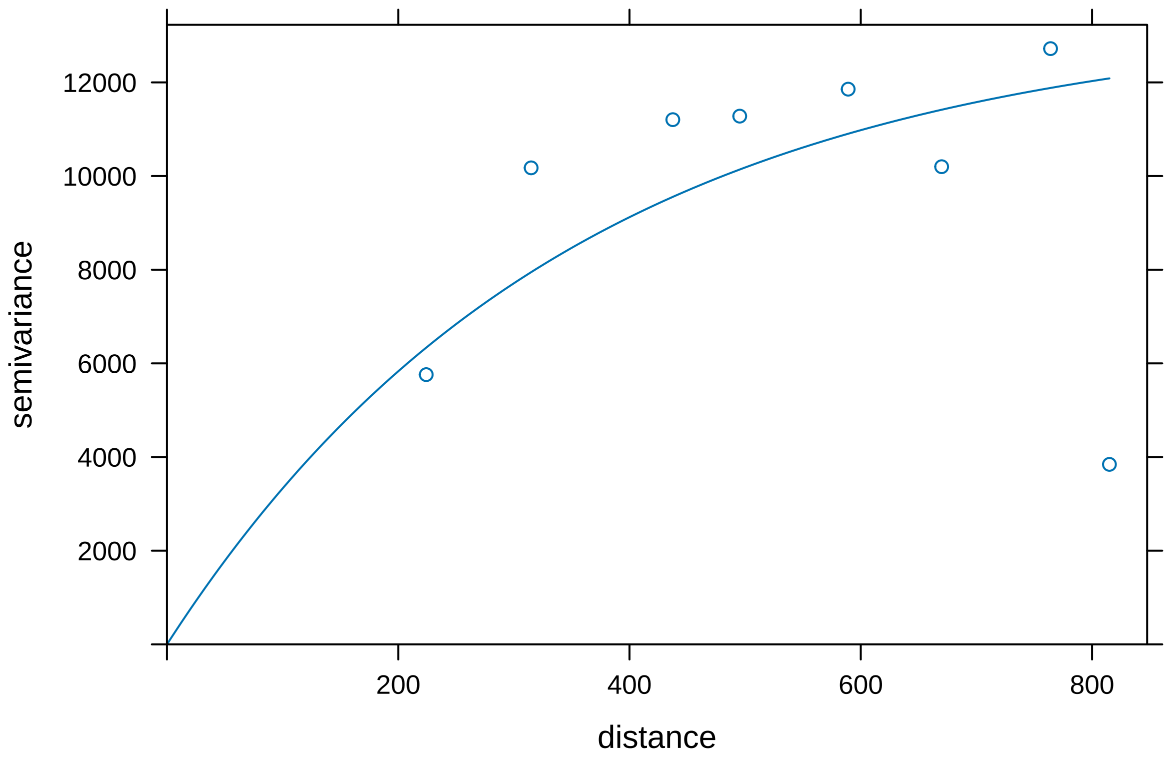



```r
# Exponential model (REML)
fit_exp <- geoR::likfit(
  data_geo,
  ini.cov.pars = c(psill, range),  # c(sigmasq, phi)
  nugget       = co,
  fix.nugget   = FALSE,
  lik.method   = "REML",
  cov.model    = "exponential"
)
```

```
## kappa not used for the exponential correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
fit.exp <- vgm(
  psill  = fit_exp$sigmasq,    # partial sill
  model  = "Exp",              # Model: Exponential
```

```r
  range  = fit_exp$phi,        # range
  nugget = fit_exp$tausq       # nugget
)

plot(var_exp, fit.exp)
```

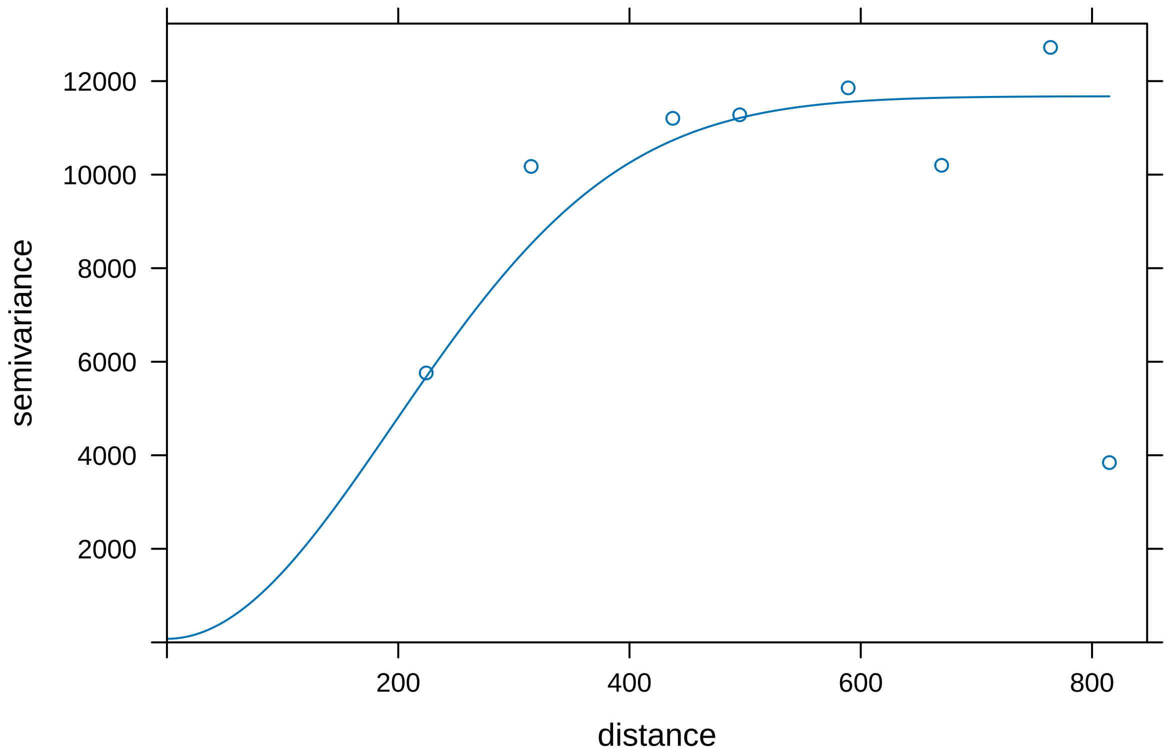



```r
# Gaussian model (REML)
fit_gau <- geoR::likfit(
  data_geo,
  ini.cov.pars = c(psill, range),  # c(sigmasq, phi)
  nugget       = co,
  fix.nugget   = FALSE,
  lik.method   = "REML",
  cov.model    = "gaussian"
)
```

```
## kappa not used for the gaussian correlation function
## ---------------------------------------------------------------
## likfit: likelihood maximisation using the function optim.
## likfit: Use control() to pass additional
##          arguments for the maximisation function.
##         For further details see documentation for optim.
## likfit: It is highly advisable to run this function several
##         times with different initial values for the parameters.
## likfit: WARNING: This step can be time demanding!
```

```
## ---------------------------------------------------------------
## likfit: end of numerical maximisation.
```

```r
fit.gauss <- vgm(
  psill  = fit_gau$sigmasq,    # partial sill
  model  = "Gau",              # Model: Gaussian
  range  = fit_gau$phi,        # range
  nugget = fit_gau$tausq       # nugget
)

plot(var_exp, fit.gauss)
```



LOOCV Cross validation to select the model to kriging

```r
# -------------------------------
# Cross-validation to select model
# -------------------------------

# Spherical model CV
xvalid.sph <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.sph)
xvalid_df_sp <- as.data.frame(xvalid.sph)
lm_sph <- lm(xvalid.sph$var1.pred ~ xvalid.sph$observed)

# Metrics
r2_sph    <- summary(lm_sph)$r.squared      # R²
rmse_sph  <- hydroGOF::rmse(xvalid.sph$var1.pred, xvalid.sph$observed)  # RMSE
```

```r
slope_sph <- coef(lm_sph)[2]                # regression slope

# Exponential model CV
xvalid.exp <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.exp)
xvalid_df_ex <- as.data.frame(xvalid.exp)
lm_exp <- lm(xvalid.exp$var1.pred ~ xvalid.exp$observed)

# Metrics
r2_exp    <- summary(lm_exp)$r.squared
rmse_exp  <- hydroGOF::rmse(xvalid.exp$var1.pred, xvalid.exp$observed)
slope_exp <- coef(lm_exp)[2]

# Gaussian model CV
xvalid.gau <- gstat::krige.cv(Ar ~ 1, locations = dados, model = fit.gauss)
xvalid_df_ga <- as.data.frame(xvalid.gau)
lm_gau <- lm(xvalid.gau$var1.pred ~ xvalid.gau$observed)

# Metrics
r2_gau    <- summary(lm_gau)$r.squared
rmse_gau  <- hydroGOF::rmse(xvalid.gau$var1.pred, xvalid.gau$observed)
slope_gau <- coef(lm_gau)[2]

# --------------------------------------------
# Build CV summary table and pick the best fit
# --------------------------------------------

cv_tbl <- data.frame(
  Model = c("Sph", "Exp", "Gau"),
  R2    = c(r2_sph, r2_exp, r2_gau),
  RMSE  = c(rmse_sph, rmse_exp, rmse_gau),
  Slope = c(slope_sph, slope_exp, slope_gau),
  stringsAsFactors = FALSE
)

# Print table sorted by RMSE (ascending = better)
cv_tbl_sorted <- cv_tbl[order(cv_tbl$RMSE), ]
print(cv_tbl_sorted, row.names = FALSE)
```

```
##  Model        R2     RMSE     Slope
##    Gau 0.6175515 65.00674 0.5969078
##    Sph 0.5752196 70.28557 0.4559113
##    Exp 0.4356654 79.29646 0.3784514
```

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

```r
# 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
```

```r
# 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 75.8431564912699
## 2      Sill 11676.2329312138
## 3     Range 276.120401103335
## 4      SDI% 99.3504484114177
## 5     Class        Very high
```

Kriging

Doing the kriging with the selected model

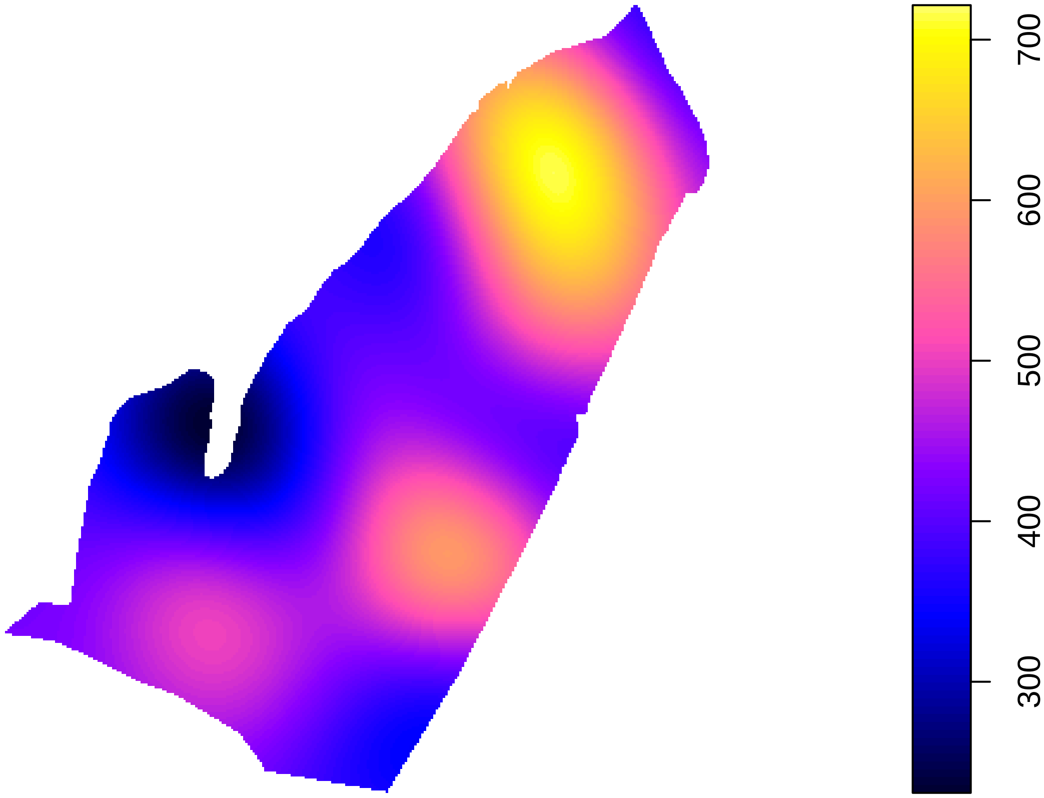
```r
### Interpolacao por krigagem
proj4string(dados) <- CRS("+init=epsg:32723")
proj4string(dados)
```

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

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

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

```r
plot(mapa)
```



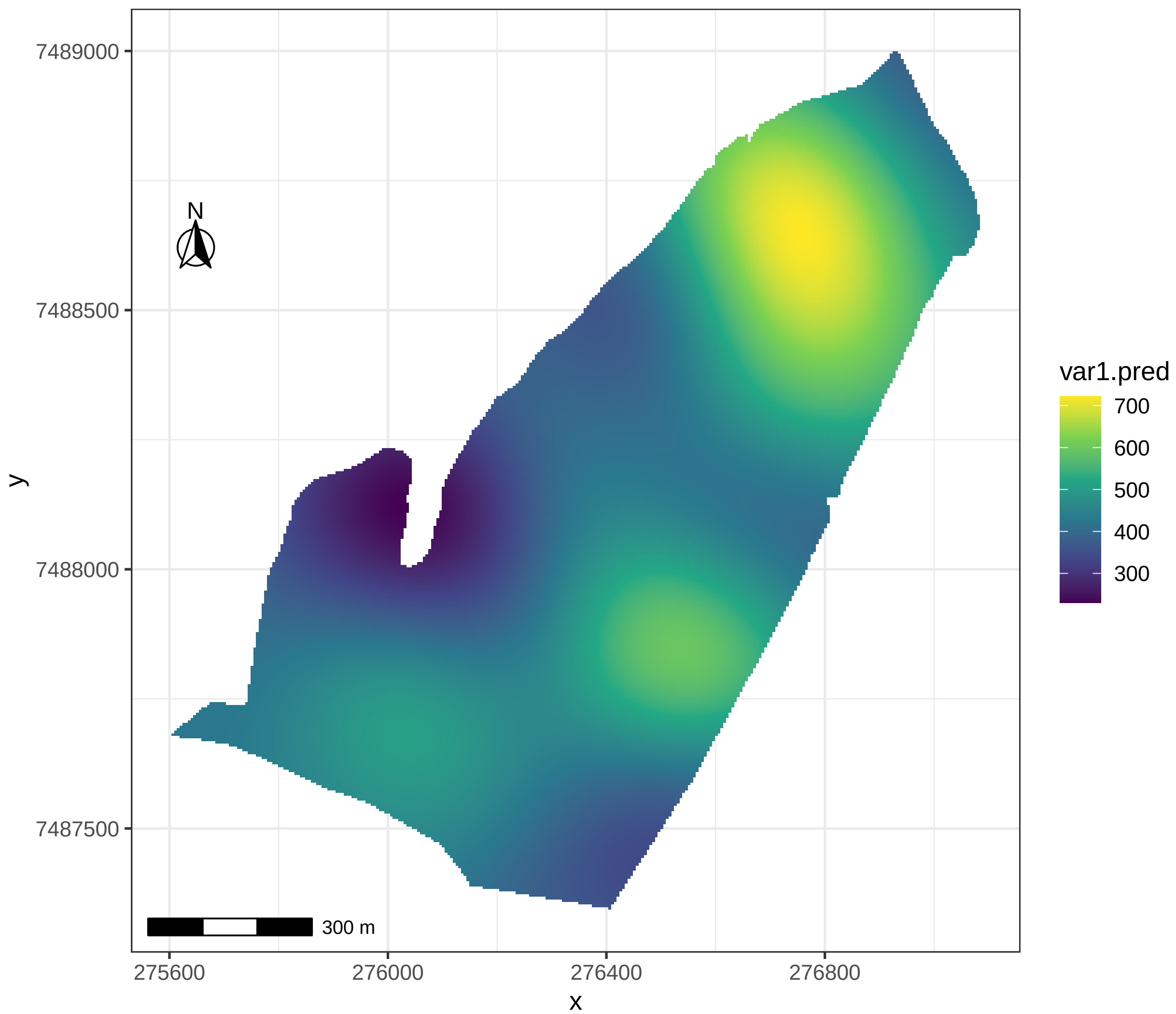
Exporting the raster

```r
mapaRaster <- raster(mapa)

filename<-'13_GitHub/Ar_OK_0.2cada1_Paulinia_REML.tiff'
writeRaster(mapaRaster, filename , format = 'GTiff', overwrite = T)
```

Graphic with ggplot

```r
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'
```


N
300 m
7489000
7488500
7488000
7487500
y
275600
276000
276400
276800
x
var1.pred
700
600
500
400
300