Random Forest Interpolation (RFSI)

Laura Delgado Bejarano
Agricultural Engineer, Universidad Nacional de Colombia
Master’s Candidate, University of Campinas (UNICAMP)

2024-10-23

Random forest Spatial interpolation based in Sekulic et al 2020 http://dx.doi.org/10.3390/rs12101687

Cleaning the space

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

```
knitr::opts_chunk$set(echo = TRUE)
# --- Directory settings (edit these paths as needed) ---
WORK_DIR <- "C:/Users/Gitap/Desktop/LauraBejarano/" # <- edit if needed
WORK_DIR <- normalizePath(WORK_DIR, winslash = "/", mustWork = TRUE)
knitr::opts_knit$set(root.dir = WORK_DIR)  # make this the working dir for all chunks
```

Libraries

Installing and loading the packages that are going to be used

```
# cargar/instalar librerías
if (!require("pacman")) install.packages("pacman")
```

```
## Carregando pacotes exigidos: pacman
```

```
pacman::p_load(
  meteo,        # RF spatial Sekulic
  sp, gstat,ggspatial,raster, ggplot2,gridExtra,ggstar,
  mapview,terra,sf, akima,automap,paletteer,
  viridis,dplyr,mlr,parallelMap,parallel,ggthemes
)
```

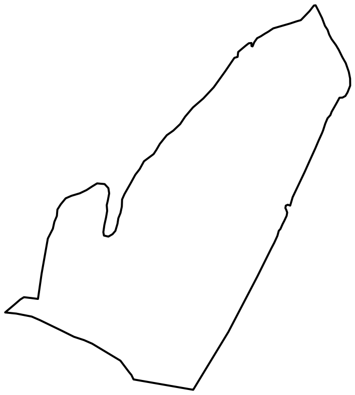
Loading the data

Load polygon of the area of study

```
poly <- st_read("03_Test/Contorno_Paulinia/Contorno_Paulinia.shp")
```

```
## Reading layer ‘Contorno_Paulinia’ from data source
##   ‘C:\Users\Gitap\Desktop\LauraBejarano\03_Test\Contorno_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
```

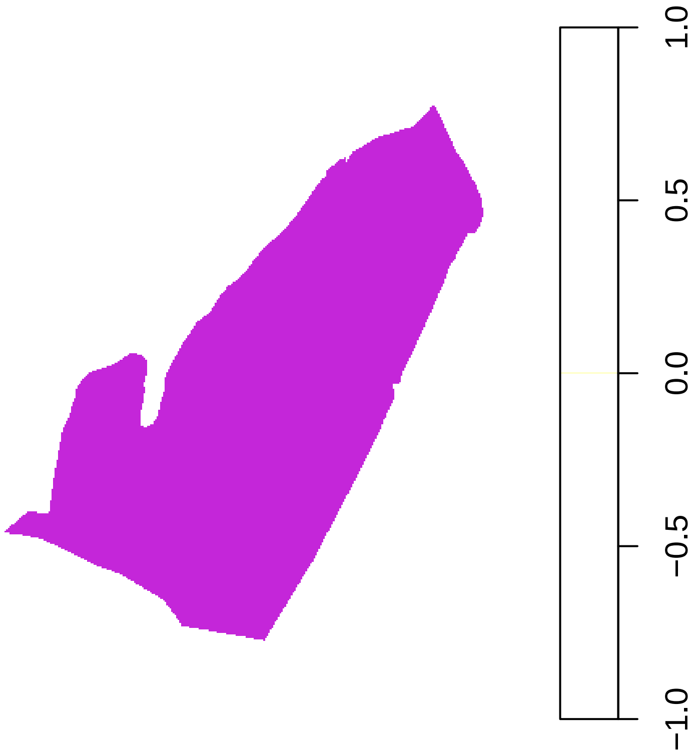
```
plot(poly)
```



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

Create a clean and empty grid to interpolate

```
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") #CRS area
```

```
## 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. Further
## messages of this type will be suppressed.
```

```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

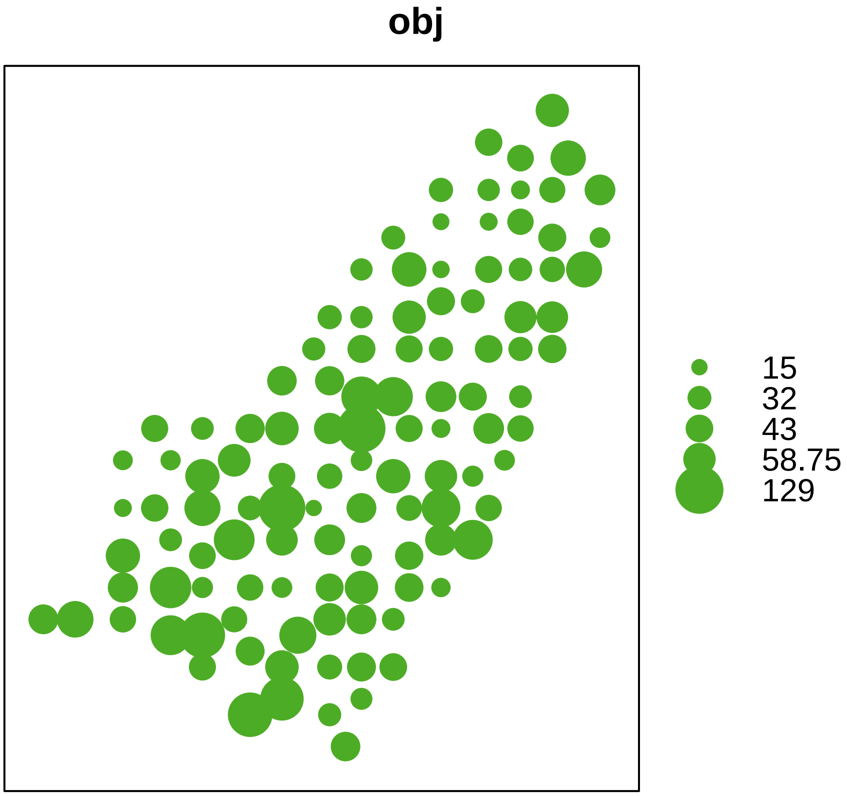
```r
original = data.frame(read.csv(file = "03_Test/Paulinia_1am_cada1ha.csv", header = TRUE, sep = ';'))
head(original)
```

```
##    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
```

```
dados = original[,c(7,8,2)] #select the columns x,y and obj for interpolation
head(dados)
```

```
##          x       y  P
## 1 275619.8 7487681 50
## 2 275699.8 7487681 75
## 3 275819.8 7488081 22
## 4 275819.8 7487961 18
## 5 275819.8 7487841 66
## 6 275819.8 7487761 51
```

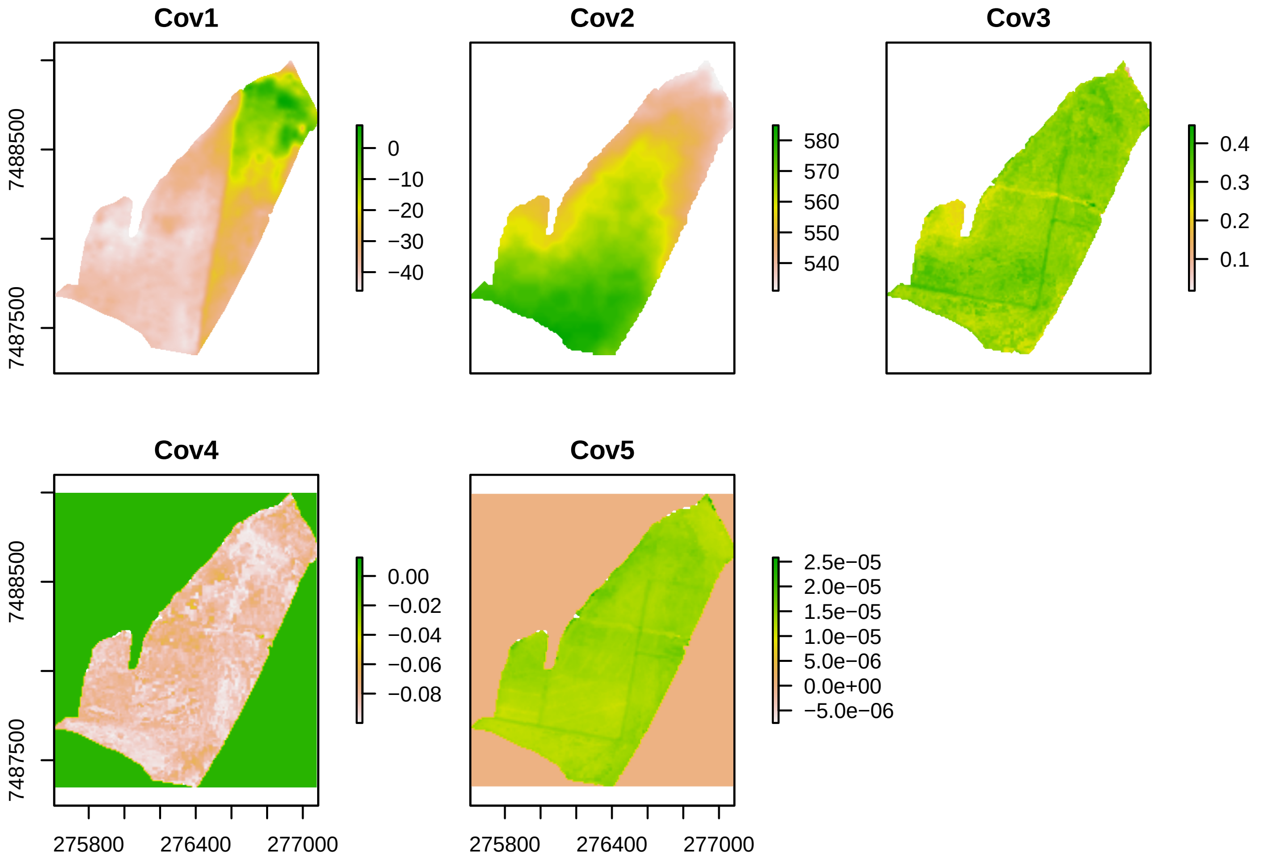
```
dados = na.omit(dados)
names(dados) = c( "x", "y", "obj") #change names
sp::coordinates(dados) = ~x+y
sp::bubble(dados, "obj")
```



#Covariates

Load covariates and resample with bilinear method in the spatial resolution wanted

```
# Covariates
# Load covariates and resample with bilinear method at desired spatial resolution

# List auxiliary rasters (.tif/.tiff)
covariates <- list.files(
  path = "03_Test/04_Stack",
  pattern = "\\.tif(f)?$",
  full.names = TRUE)

# Use elevation raster as template if available, otherwise the first raster
elev_idx <- grep("Elevacao_", basename(covariates), ignore.case = TRUE)
template_path <- if (length(elev_idx) > 0) covariates[elev_idx[1]] else covariates[1]
reference_raster <- raster::raster(template_path)
raster::res(reference_raster) <- 5 #Pixel size

# Read and resample rasters
all_rasters <- lapply(covariates, raster::raster)
resampled   <- raster::stack(lapply(all_rasters, function(r)
  raster::resample(r, reference_raster, "bilinear")
))

# Rename layers
names(resampled) <- paste0("Cov", seq_len(raster::nlayers(resampled)))

# Quick plot
plot(resampled)
```



##Extraction

Extraction of auxiliary information co-located with soil sampling points for data

```
Values<-raster::extract(resampled,dados)
head(Values)
```

```
##           Cov1     Cov2      Cov3        Cov4         Cov5
## [1,] -40.94244 576.8875 0.3357456 -0.08360102 6.577062e-06
## [2,] -38.96170 577.6394 0.2892047 -0.09212512 1.229361e-05
## [3,] -43.20070 558.0656 0.2549428 -0.03880936 5.203175e-06
## [4,] -40.64766 561.0250 0.2619579 -0.08977283 1.383433e-05
## [5,] -39.37175 570.1350 0.3265654 -0.08547828 1.227005e-05
## [6,] -38.68437 573.0000 0.3322796 -0.08169111 1.136339e-05
```

```
dados@data<-cbind(dados@data,Values)
head(dados)
```

```
##   obj      Cov1     Cov2      Cov3        Cov4         Cov5
## 1  50 -40.94244 576.8875 0.3357456 -0.08360102 6.577062e-06
## 2  75 -38.96170 577.6394 0.2892047 -0.09212512 1.229361e-05
## 3  22 -43.20070 558.0656 0.2549428 -0.03880936 5.203175e-06
## 4  18 -40.64766 561.0250 0.2619579 -0.08977283 1.383433e-05
## 5  66 -39.37175 570.1350 0.3265654 -0.08547828 1.227005e-05
## 6  51 -38.68437 573.0000 0.3322796 -0.08169111 1.136339e-05
```

```
#Combine with interpolation grid
grid_cov <- raster::extract(resampled, grid)
grid@data  <- cbind(grid@data, grid_cov)
```

Random Forest

##Hiperparameters

```
dados_df <- as.data.frame(dados)
regr.task <- makeRegrTask(data = dados_df, target = "obj")
rf_learner <- makeLearner(
  "regr.randomForest",
  predict.type = "response",
  par.vals = list(importance = TRUE)
)

n_feats   <- mlr::getTaskNFeats(regr.task)
mtry_max  <- max(1, n_feats)

rf_param <- makeParamSet(
  makeIntegerParam("ntree",    lower = 200,  upper = 5000),
  makeIntegerParam("mtry",     lower = 1,    upper = mtry_max),
  makeIntegerParam("nodesize", lower = 1,    upper = 50)
)

ctrl <- makeTuneControlRandom(maxit = 200)
cv   <- makeResampleDesc("CV", iters = 5)
measure_list <- list(mlr::rmse)
```

```
n_cores<-parallel::detectCores()-1
set.seed(10)
t0 <- Sys.time()

rf_tune <- tuneParams(
  learner      = rf_learner,
  task         = regr.task,
  resampling   = cv,
  par.set      = rf_param,
  control      = ctrl,
  measures     = measure_list,
  show.info    = TRUE
)

tuning_time <- Sys.time() - t0
print(rf_tune)
print(tuning_time)

parallelStop()
```

##Parameters

```
print(tuning_time)
```

```
## Time difference of 2.763389 mins
```

```
numtrees_best   <- rf_tune$x$ntree;numtrees_best
```

```
## [1] 3761
```

```
mtry_best       <- rf_tune$x$mtry;mtry_best
```

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

```
nodesize_best   <- rf_tune$x$nodesize;nodesize_best
```

```
## [1] 47
```

Training

Training the RFSI

```
dados <- st_as_sf(dados)
fm.RFSI <- as.formula("obj~Cov1+Cov2+Cov3+Cov4+Cov5") # Reeplace with covariates


rfsi_model <- rfsi(formula = fm.RFSI,
                   data = dados,
                   zero.tol = 0,
                   n.obs = 8, # number of nearest observations
                   # s.crs = st_crs(data), # nedded only if the coordinates are lon/lat (WGS84)
                   # p.crs = st_crs(data), # nedded only if the coordinates are lon/lat (WGS84)
                   cpus = detectCores()-1,
                   progress = TRUE,
                   # ranger parameters
                   importance = "impurity",
                   seed = 315,
                   num.trees = numtrees_best, #Values in the last chunk
                   mtry = mtry_best,#Values in the last chunk
                   splitrule = "variance",
                   min.node.size = nodesize_best, #nodesize #Values in the last chunk
                   sample.fraction = 0.95,
                   quantreg = FALSE) # quantile regression model
```

```
## Preparing data ...
```

```
## Warning in rfsi(formula = fm.RFSI, data = dados, zero.tol = 0, n.obs = 8, :
## Source CRS is NULL! Using given coordinates for Euclidean distances
## calculation.
```

```
## Spatial process ...
```

```
## Calculating distances to the nearest observations ...
```

```
## Fitting RFSI model ...
```

```
## Warning in ranger(formula, data = data.df, ...): Unused arguments: zero.tol
```

```
## Done!
```

```
rfsi_model
```

```
## Ranger result
##
## Call:
##  ranger(formula, data = data.df, ...)
##
## Type:                             Regression
## Number of trees:                  3761
## Sample size:                      114
## Number of independent variables:  21
## Mtry:                             1
## Target node size:                 47
## Variable importance mode:         impurity
## Splitrule:                        variance
## OOB prediction error (MSE):       536.5389
## R squared (OOB):                  0.03124533
```

Predicting

```r
newdata <- resampled
names(newdata)<-c("Cov1","Cov2","Cov3","Cov4","Cov5")
newdata<-rast(newdata)

rfsi_prediction <- pred.rfsi(model = rfsi_model,
                             data = dados,
                             obs.col = "obj",
                             newdata = newdata,
                             output.format = "SpatRaster", # "sf", # "SpatVector",
                             zero.tol = 0,
                             cpus = detectCores()-1,
                             progress = TRUE,
                             soil3d=FALSE
)
```

```
## Preparing data ...
```

```
## Spatial process ...
## Spatial process ...
```

```
## Warning in pred.rfsi(model = rfsi_model, data = dados, obs.col = "obj", : Data
## source CRS is NA! Using given coordinates for Euclidean distances calculation.
```

```
## Warning in pred.rfsi(model = rfsi_model, data = dados, obs.col = "obj", :
## Newdata projection CRS is NA. Using source CRS for Euclidean distances
## calculation:
```

```
## PROJCRS["unknown",
##     BASEGEOGCRS["unknown",
##         DATUM["World Geodetic System 1984",
##             ELLIPSOID["WGS 84",6378137,298.257223563,
##                 LENGTHUNIT["metre",1]],
##             ID["EPSG",6326]],
##         PRIMEM["Greenwich",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8901]]],
##     CONVERSION["UTM zone 23S",
##         METHOD["Transverse Mercator",
##             ID["EPSG",9807]],
##         PARAMETER["Latitude of natural origin",0,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8801]],
##         PARAMETER["Longitude of natural origin",-45,
##             ANGLEUNIT["degree",0.0174532925199433],
##             ID["EPSG",8802]],
##         PARAMETER["Scale factor at natural origin",0.9996,
##             SCALEUNIT["unity",1],
##             ID["EPSG",8805]],
##         PARAMETER["False easting",500000,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8806]],
##         PARAMETER["False northing",10000000,
##             LENGTHUNIT["metre",1],
##             ID["EPSG",8807]],
##         ID["EPSG",16123]],
##     CS[Cartesian,2],
##         AXIS["(E)",east,
##             ORDER[1],
##             LENGTHUNIT["metre",1,
##                 ID["EPSG",9001]]],
##         AXIS["(N)",north,
##             ORDER[2],
##             LENGTHUNIT["metre",1,
##                 ID["EPSG",9001]]]]
```

```
## Calculating distances to the nearest observations ...
```

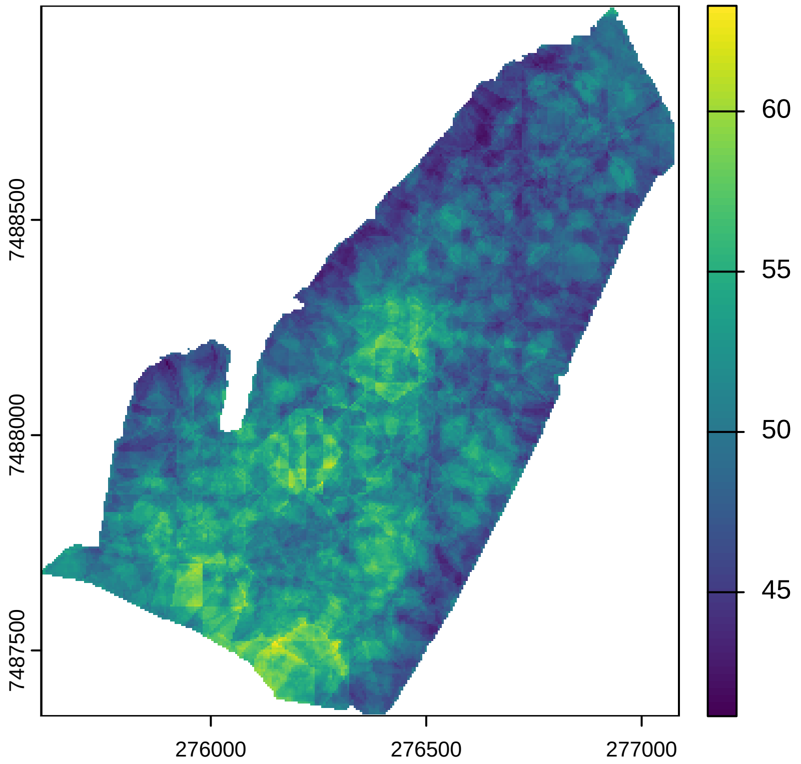
```
## Doing RFSI predictions ...
```

```
## Warning in pred.rfsi(model = rfsi_model, data = dados, obs.col = "obj", :
## Newdata is in SpatRaster format. output.format ignored, returning SpatRaster!
```

```r
summary(rfsi_prediction)
```

```
##       pred
##  Min.   :41.13
##  1st Qu.:47.52
##  Median :49.88
##  Mean   :50.22
##  3rd Qu.:52.68
##  Max.   :63.29
##  NA's   :54755
```

```r
plot(rfsi_prediction)
```



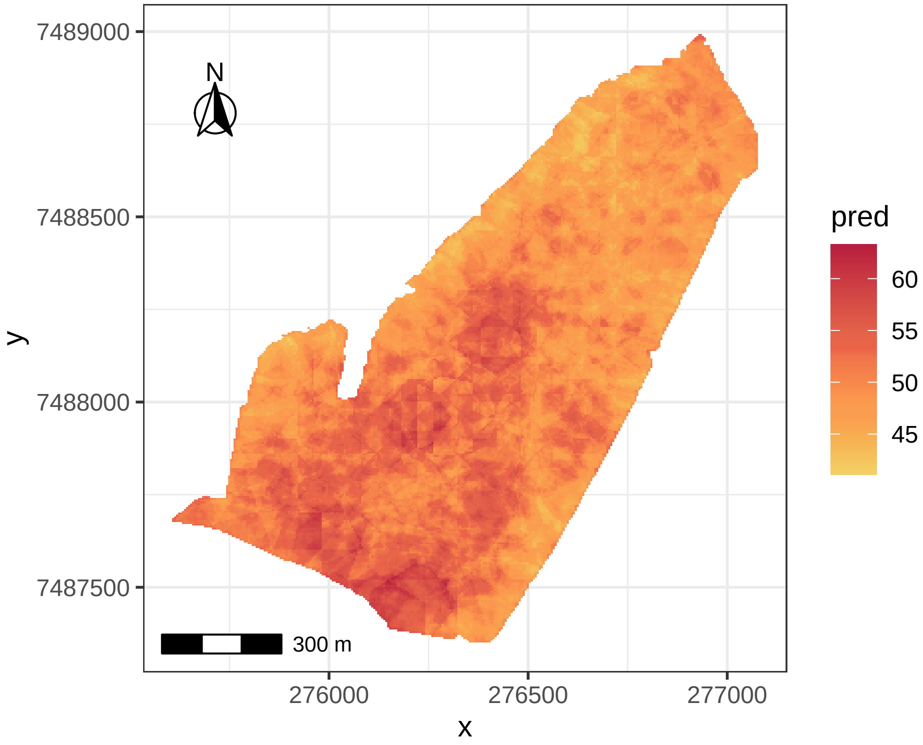
## Exporting the raster

```
mapaRaster <- raster(rfsi_prediction)

filename<-'C:/Users/Gitap/Desktop/LauraBejarano/03_Test/Result_rf.tiff'
writeRaster(mapaRaster, filename , format = 'GTiff', overwrite = T)
```

##Graphic with ggplot

```
rfsi_df <- as.data.frame(rfsi_prediction, xy = TRUE)
ggplot() +
  geom_raster(data = as.data.frame(rfsi_df), aes(fill = pred, x = x, y = y)) +
  scale_fill_paletteer_c("ggthemes::Red-Gold") +
  # labs(fill = expression('Argila (g*kg'^-1*')'))+
  annotation_north_arrow(which_north = "grid",height = unit(1, "cm"),
                         width = unit(0.9, "cm"),
                         pad_x = unit(0.5, "cm"),
                         pad_y = unit(7, "cm"),
                         style=north_arrow_fancy_orienteering())+
  annotation_scale(location = "bl", width_hint = 0.2)+
  theme_bw()
```

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


7489000
7488500
7488000
7487500
y
276000
276500
277000
x
N
300 m
pred
60
55
50
45