# SMT1

Script para el cálculo de cobertura arbórea

SMT1

Script to calculate tree cover

|  |
| --- |
| // Get the forest loss image.var gfc2014 = ee.Image('UMD/hansen/global\_forest\_change\_2021\_v1\_9');var lossImage = gfc2014.select(['loss']);var areaImage = lossImage.multiply(ee.Image.pixelArea());// Sum the values of forest loss pixels.var stats = areaImage.reduceRegion({ reducer: ee.Reducer.sum(), geometry: table, scale: 30, maxPixels: 1e9});print('pixels representing loss: ', stats.get('loss'), 'square meters');// Get the loss image.// This dataset is updated yearly, so we get the latest version.var gfc2021g = ee.Image('UMD/hansen/global\_forest\_change\_2021\_v1\_9');var lossImage = gfc2021g.select(['loss']);var lossAreaImage = lossImage.multiply(ee.Image.pixelArea());var lossYear = gfc2021g.select(['lossyear']);var lossByYear = lossAreaImage.addBands(lossYear).reduceRegion({ reducer: ee.Reducer.sum().group({groupField: 1 }), geometry: table, scale: 30,maxPixels: 1e9});print(lossByYear);var statsFormatted = ee.List(lossByYear.get('groups')).map(function(el) { var d = ee.Dictionary(el); return [ee.Number(d.get('group')).format("20 %02d"), d.get('sum')]; });var statsDictionary = ee.Dictionary(statsFormatted.flatten());print(statsDictionary);var chart = ui.Chart.array.values({ array: statsDictionary.values(), axis: 0,xLabels: statsDictionary.keys()}).setChartType('ColumnChart').setOptions({ title: 'Yearly Forest Loss',hAxis: {title: 'Year', format: '####'},vAxis: {title: 'Area (square meters)'}, legend: { position: "none" },lineWidth: 1,pointSize: 3});print(chart); |

SMT2

Script del modelo empleado en Wallace 2.0

SMT2

Script of the model used in Wallace 2.0

|  |
| --- |
| library(spocc)library(spThin)library(dismo)library(rgeos)library(ENMeval)library(wallace)# just type the function name and press Return to see its source code# paste this code into a new script to edit itoccs\_queryDbYouranalyses are below.# NOTE: provide the folder path of the .csv fileoccs\_path<- ""occs\_path<- file.path(occs\_path, "Presas\_japu\_processed\_occs.csv")# get a list of species occurrence datauserOccs\_Pj<- occs\_userOccs(txtPath = occs\_path, txtName = "Presas\_japu\_processed\_occs.csv", txtSep = ",", txtDec = ".")occs\_Pj<- userOccs\_Pj$Presas\_japu$cleaned# Download environmental data envs\_Pj<- envs\_worldclim(bcRes = 2.5, bcSel = c('bio01', 'bio02', 'bio03', 'bio04', 'bio05', 'bio07', 'bio12', 'bio13', 'bio14', 'bio15'), mapCntr = c(-79.74, -1.047), # Mandatory for 30 arcsec resolution doBrick = TRUE)occs\_xy\_Pj<- occs\_Pj[c('longitude', 'latitude')]occs\_vals\_Pj<- as.data.frame(raster::extract(envs\_Pj, occs\_xy\_Pj, cellnumbers = TRUE))# Remove duplicated same cell valuesoccs\_Pj<- occs\_Pj[!duplicated(occs\_vals\_Pj[, 1]), ]occs\_vals\_Pj<- occs\_vals\_Pj[!duplicated(occs\_vals\_Pj[, 1]), -1]# remove occurrence records with NA environmental valuesoccs\_Pj<- occs\_Pj[!(rowSums(is.na(occs\_vals\_Pj)) >= 1), ]# also remove variable value rows with NA environmental valuesoccs\_vals\_Pj<- na.omit(occs\_vals\_Pj)# add columns for env variable values for each occurrence recordoccs\_Pj<- cbind(occs\_Pj, occs\_vals\_Pj)# Load the user provided shapefile or csv file with the desired extent. ##User must input the path to shapefile or csv file and the file name # Define pathbgPath\_Pj<- ""bgExt\_Pj<- penvs\_userBgExtent(bgShp\_path = paste0(bgPath\_Pj, "Occidente", ".shp"),bgShp\_name = paste0("Occidente", c(".shp", ".shx", ".dbf")),userBgBuf = 0,occs = occs\_Pj)# Mask environmental data to provided extentbgMask\_Pj<- penvs\_bgMask(occs = occs\_Pj,envs = envs\_Pj,bgExt = bgExt\_Pj)# Sample background points from the provided areabgSample\_Pj<- penvs\_bgSample(occs = occs\_Pj,bgMask = bgMask\_Pj,bgPtsNum = 1000)# Extract values of environmental layers for each background pointbgEnvsVals\_Pj<- as.data.frame(raster::extract(bgMask\_Pj, bgSample\_Pj))##Add extracted values to background points tablebgEnvsVals\_Pj<- cbind(scientific\_name = paste0("bg\_", "Presas japu"), bgSample\_Pj,occID = NA, year = NA, institution\_code = NA, country = NA,state\_province = NA, locality = NA, elevation = NA,record\_type = NA, bgEnvsVals\_Pj)# R code to get partitioned datagroups\_Pj<- part\_partitionOccs(occs = occs\_Pj ,bg = bgSample\_Pj,  method = "block",bgMask = bgMask\_Pj,aggFact = 2) # Run maxent model for the selected speciesmodel\_Pj<- model\_maxent(occs = occs\_Pj,bg = bgEnvsVals\_Pj,user.grp = groups\_Pj, bgMsk = bgMask\_Pj, rms = c(0.5, 5), rmsStep = 0.5, fcs = c('L', 'LQ', 'H', 'LQH'),clampSel = TRUE,algMaxent = "maxnet", parallel = FALSE,numCores = 7)# Select current model and obtain raster predictionm\_Pj<- model\_Pj@models[["fc.H\_rm.0.5"]]predSel\_Pj<- predictMaxnet(m\_Pj, bgMask\_Pj, type = "cloglog",  clamp = TRUE)#Get values of predictionmapPredVals\_Pj<- getRasterVals(predSel\_Pj, "cloglog")#Define colors and legend rasCols<- c("#2c7bb6", "#abd9e9", "#ffffbf", "#fdae61", "#d7191c")legendPal<- colorNumeric(rev(rasCols), mapPredVals\_Pj, na.color = 'transparent')rasPal<- colorNumeric(rasCols, mapPredVals\_Pj, na.color = 'transparent')#Generate mapm <- leaflet() %> %addProviderTiles(providers$Esri.WorldTopoMap) m %> % leaflet::addLegend("bottomright", pal = legendPal, title = "Predicted Suitability<br>(Training)", values = mapPredVals\_Pj, layerId = "train",labFormat = reverseLabel(2, reverse\_order = TRUE)) %> % #add occurrence dataaddCircleMarkers(data = occs\_Pj, lat = ~latitude, lng = ~longitude, radius = 5, color = 'red', fill = TRUE, fillColor = "red",fillOpacity = 0.2, weight = 2, popup = ~pop) %> % ##Add model predictionaddRasterImage(predSel\_Pj, colors = rasPal, opacity = 0.7, group = 'vis', layerId = 'mapPred', method = "ngb") %> % ##add background polygonsaddPolygons(data = bgExt\_Pj,fill = FALSE, weight = 4, color = "blue", group = 'proj')#Download variables for transferring xferTimeEnvs\_Pj<- raster::getData( 'CMIP5', var = "bio", res = round((raster::res(bgMask\_Pj) \* 60)[1],1),rcp = 85, model = "CC", year = 50)names(xferTimeEnvs\_Pj) <- paste0('bio', c(paste0('0',1:9), 10:19))# Select variables for transferring to match variables used for modelling xferTimeEnvs\_Pj<- xferTimeEnvs\_Pj[[names(bgMask\_Pj)]]# Generate a transfer of the model to the desired area and timexfer\_time\_Pj<-xfer\_time(evalOut = model\_Pj,curModel = "fc.H\_rm.0.5",envs = xferTimeEnvs\_Pj,xfExt = bgExt\_Pj,alg = "maxnet",outputType = "cloglog", clamp = TRUE ) # store the cropped variables of transferxferExt\_Pj<- xfer\_time\_Pj$xferExt###Make map of transferbb\_Pj<- bgExt\_Pj@bboxbbZoom<- polyZoom(bb\_Pj[1, 1], bb\_Pj[2, 1], bb\_Pj[1, 2], bb\_Pj[2, 2], fraction = 0.05)mapXferVals\_Pj<- getRasterVals(xfer\_time\_Pj$xferTime,"cloglog")rasCols\_Pj<- c("#2c7bb6", "#abd9e9", "#ffffbf", "#fdae61", "#d7191c")# if no threshold specifiedlegendPal<- colorNumeric(rev(rasCols\_Pj), mapXferVals\_Pj, na.color = 'transparent')rasPal\_Pj<- colorNumeric(rasCols\_Pj, mapXferVals\_Pj, na.color = 'transparent')m <- leaflet() %> %addProviderTiles(providers$Esri.WorldTopoMap) m %> %fitBounds(bbZoom[1], bbZoom[2], bbZoom[3], bbZoom[4]) %> % leaflet::addLegend("bottomright", pal = legendPal, title = "Predicted Suitability<br>(Transferred)", values = mapXferVals\_Pj, layerId = 'xfer',labFormat = reverseLabel(2, reverse\_order = TRUE)) %> %# map model prediction raster and polygon of transferclearMarkers() %> %clearShapes() %> %removeImage('xferRas') %> %addRasterImage(xfer\_time\_Pj$xferTime, colors = rasPal\_Pj, opacity = 0.7,layerId = 'xferRas', group = 'xfer', method = "ngb") %> % ##add polygon of transfer (same modeling area)addPolygons(data = bgExt\_Pj, fill = FALSE, weight = 4, color = "blue", group = 'xfer')# R code to generate MESS rasterxferMess\_Pj<- xfer\_mess(occs = occs\_Pj,bg = bgEnvsVals\_Pj ,bgMsk = bgMask\_Pj,xferExtRas = xferExt\_Pj)# Generate MESS maprasVals\_Pj<- getRasterVals(xferMess\_Pj) # define colorRamp for mess if (max(rasVals\_Pj) > 0 & min(rasVals\_Pj) < 0) { rc1 <- colorRampPalette(colors = rev(RColorBrewer::brewer.pal(n = 3, name = 'Reds')), space = "Lab")(abs(min(rasVals\_Pj))) rc2 <- colorRampPalette(colors = RColorBrewer::brewer.pal(n = 3, name = 'Blues'), space = "Lab")(max(rasVals\_Pj))rasCols\_Pj<- c(rc1, rc2) } else if (max(rasVals\_Pj) < 0 & min(rasVals\_Pj) < 0) {rasCols\_Pj<- colorRampPalette(colors = rev(RColorBrewer::brewer.pal(n = 3, name = 'Reds')), space = "Lab")(abs(min(rasVals))) } else if (max(rasVals\_Pj) > 0 & min(rasVals\_Pj) > 0) {rasColsPj<- colorRampPalette(colors = RColorBrewer::brewer.pal(n = 3, name = 'Blues'), space = "Lab")(max(rasVals\_Pj)) }legendPal\_Pj<- colorNumeric(rev(rasCols\_Pj), rasVals\_Pj, na.color='transparent')rasPal\_Pj<- colorNumeric(rasCols\_Pj, rasVals\_Pj, na.color='transparent') #Create map m <- leaflet() %> %addProviderTiles(providers$Esri.WorldTopoMap) m %> % leaflet::addLegend("bottomright", pal = legendPal\_Pj, title = "MESS Values", values = rasVals\_Pj, layerId = 'xfer',labFormat = reverseLabel(2, reverse\_order=TRUE)) %> % # map model prediction raster and transferring polygonclearMarkers() %> %clearShapes() %> %removeImage('xferRas') %> %addRasterImage(xferMess\_Pj, colors = rasPal\_Pj , opacity = 0.9,layerId = 'xferRas', group = 'xfer', method = "ngb") %> % ##add transferring polygon: this we need to fix for now please replace bgExt\_Pj for the name of your transferring polygon.addPolygons(data = bgExt\_Pj, fill = FALSE, weight = 4, color = "blue", group = 'xfer') |