########## SPECIES DISTRIBUTION MODELING

### load packages

library(raster)

library(rgdal)

library(dismo)

library(ENMeval)

library(megaSDM)

library(SDMtune)

library(ecospat)

library(MASS)

library(dplyr)

library(ggplot2)

library(extrafont)

library(rasterVis)

########################################################################################################################################

#### import .grd layers

env <- raster::stack(list.files(path = 'layers/grd', pattern = '.grd$', full.names = T))

plot(env[[1]])

names(env)

#### load occs ::: full dataset

occs <- read.csv('occs/chosenicus\_occ\_full.csv') %>% select(1,2,3)

colnames(occs) = c('species', 'long', 'lat')

head(occs)

## thin occs data

occs <- SDMtune::thinData(coords = occs, env = env, x = 'long', y = 'lat')

write.csv(occs, 'occs/chosenicus\_occ\_full\_1km\_thinned.csv')

head(occs)

## generate calibration area ::: use megaSDM package

occlist <- list.files(path = 'buffers/buff\_occ', pattern = '.csv', full.names = T)

BackgroundBuffers(occlist = occlist, envdata = env, output = 'buffers',

buff\_distance = 0.45)

buff <- readOGR('buffers/chosenicus\_occ\_full\_1km\_thinned.shp')

## crop layers to calibration area

env\_calib <- raster::mask(env, buff)

plot(env\_calib[[1]])

#### target group background sampling

# create bias layer

# load thinned target group bg clipped to the calibration area

targ <- read.csv('targ\_bg2/targ/targ\_bg\_thinned\_calib.csv') %>% select(3,4)

head(targ)

plot(env[[1]], col = 'grey', legend = F) ## un-clipped raster for reference

plot(buff, border = 'blue', lwd = 3, add = T) ## plot buffer over the full raster

points(targ) ## plot target group points

targ.ras <- rasterize(targ, env\_calib, 1)

plot(targ.ras)

targ.pres <- which(values(targ.ras) == 1)

targ.pres.locs <- coordinates(targ.ras)[targ.pres, ]

targ.dens <- MASS::kde2d(targ.pres.locs[,1], targ.pres.locs[,2],

n = c(nrow(targ.ras), ncol(targ.ras)),

lims = c(extent(env\_calib)[1], extent(env\_calib)[2],

extent(env\_calib)[3], extent(env\_calib)[4]))

targ.dens.ras <- raster(targ.dens, env\_calib)

targ.dens.ras2 <- resample(targ.dens.ras, env\_calib)

bias.layer <- raster::mask(targ.dens.ras2, buff)

crs(bias.layer) = crs(env\_calib)

plot(bias.layer)

## export bias layer ::: grd & bil

writeRaster(bias.layer, 'targ\_bg2/targ/bias.file/bias\_layer.grd', overwrite = T) # grd

writeRaster(bias.layer, 'targ\_bg2/targ/bias.file/bias\_layer.bil', overwrite = T) # bil

writeLines(showWKT(crs(bias.layer, asText=T)), extension('targ\_bg2/targ/bias.file/bias\_layer.grd', 'prj'))

writeLines(showWKT(crs(bias.layer, asText=T)), extension('targ\_bg2/targ/bias.file/bias\_layer.bil', 'prj'))

## sample target group bg points

length(which(!is.na(values(subset(env\_calib, 1)))))

targ.bg <- xyFromCell(bias.layer,

sample(which(!is.na(values(subset(env\_calib, 1)))), 10000,

prob = values(bias.layer)[!is.na(values(subset(env\_calib, 1)))])) %>% as.data.frame()

colnames(targ.bg) = colnames(occs[, c(2,3)])

head(targ.bg)

write.csv(targ.bg, 'targ\_bg2/targ/bias.file/targ\_bg\_calib.csv')

plot(env[[1]], col = 'grey', legend = F) ## un-clipped raster for reference

plot(buff, border = 'blue', lwd = 3, add = T) ## plot buffer over the full raster

points(targ.bg) ## plot target group bg points

#########################################################################################################################################

######## select environmental variables

## data-driven approach using SDMtune

cor.bg <- prepareSWD(species = 'bgs', a = targ.bg, env = env\_calib, categorical = NULL)

plotCor(cor.bg, method = 'spearman', cor\_th = 0.7)

corVar(cor.bg, method = 'spearman', cor\_th = 0.7)

## STEP 1 ::: remove variables with low importance

## first generate a default MaxEnt model

data\_def <- prepareSWD(species = 'pcho', p = occs[, c(2,3)],

a = targ.bg, env = env\_calib, categorical = NULL)

c(train, test) %<-% trainValTest(data\_def, test = 0.2)

(default\_model <- train(method = 'Maxent', data = train))

varImp(default\_model, permut = 10)

cat('Testing TSS before', tss(default\_model, test = test))

reduced\_var\_mod <- reduceVar(default\_model, th = 1, metric = 'tss',

test = test, permut = 10, use\_jk = T, use\_pc = T)

cat('Testing TSS after', tss(reduced\_var\_mod, test = test))

## STEP 2 ::: remove highly correlated variables

selected\_var\_mod <- varSel(reduced\_var\_mod, metric = 'tss', test = test,

bg4cor = cor.bg, method = 'spearman', cor\_th = 0.7,

permut = 10, use\_pc = T)

# env stack for downstream SDM operation ::: only containing selected env variables

names(env\_calib)

env\_select <- stack(subset(env\_calib, c(6,13,18,20,23,24,26,27)))

plot(env\_select[[1]])

nlayers(env\_select)

names(env\_select)

#########################################################################################################################################

#################### use dismo

#### first use ENMeval for parameter tuning

#### random k-fold partitioning

rand <- get.randomkfold(occs = occs[, c(2,3)], bg = targ.bg, k = 10)

evalplot.grps(pts = occs[, c(2,3)], pts.grp = rand$occs.grp, envs = env\_select[[1]])

#### run ENMevaluate

eval <- ENMevaluate(occs = occs[, c(2,3)],

envs = env\_select,

bg = targ.bg,

tune.args = list(fc = c('L', 'LQ', 'H', 'LQH', 'LQHP', 'LQHPT'), rm = 0.5:8),

partitions = 'randomkfold',

partition.settings = list(kfolds = 10),

algorithm = 'maxent.jar',

doClamp = T)

#### select optimal parameter

eval.res <- eval.results(eval)

## optimal AICc

(opt.aicc <- eval.res %>% filter(delta.AICc == 0)) ## == LQHP\_4.5

mod.aicc <- eval.predictions(eval)[[opt.aicc$tune.args]]

plot(mod.aicc)

## sequential selection

(opt.seq <- eval.res %>%

filter(auc.val.avg == max(auc.val.avg)) %>%

filter(or.10p.avg == min(or.10p.avg))) ## == H\_0.5

mod.seq <- eval.predictions(eval)[[opt.seq$tune.args]]

plot(mod.seq)

#### use sequentially selected model ::: H\_0.5

#### dismo for model run ::: MaxEnt model fitting

## CAUTION for using args :::

## fully spell out [ true ] instead of [ T ] or [ TRUE ]

## and [ no space ] between flags and values

## for example ::: [ pictures=true ] instead of [ pictures = true ]

## for crossvalidation ::: set randomtestpoints to 0

dismo.mod <- dismo::maxent(x = env\_select, p = occs[, c(2,3)], a = targ.bg,

path = 'SDM/dismo',

args = c('responsecurves=true',

'pictures=true',

'jackknife=true',

'outputformat=cloglog',

'outputfiletype=asc',

'randomseed=true',

'betamultiplier=0.5',

'replicates=10',

'randomtestpoints=0',

'replicatetype=crossvalidate',

'writebackgroundpredictions=true',

'writeplotdata=true',

'linear=false',

'quadratic=false',

'product=false',

'threshold=false',

'hinge=true',

'visible=true',

'autofeature=false',

'outputgrids=true',

'plots=true',

'maximumiterations=5000'))

######## dismo ::: make MaxEnt model predictions

#### predict to calibration area

dismo.pred <- dismo::predict(object = dismo.mod, x = env\_select)

## generate averaged model prediction

dismo.pred.avg <- mean(dismo.pred[[1]], dismo.pred[[2]], dismo.pred[[3]],

dismo.pred[[4]], dismo.pred[[5]], dismo.pred[[6]],

dismo.pred[[7]], dismo.pred[[8]], dismo.pred[[9]],

dismo.pred[[10]])

plot(dismo.pred.avg)

writeRaster(dismo.pred.avg, 'SDM/output\_model\_grids/calibration\_area\_avg\_dismo.tif')

## plot ggplot style ::: continuous ::: calibration area

gplot(dismo.pred.avg) +

geom\_tile(aes(fill = value)) +

coord\_equal() +

scale\_fill\_gradientn(colours = c("#2c7bb6", "#abd9e9", "#ffffbf", "#fdae61", "#d7191c"),

na.value = "transparent",

name = 'Suitability') +

geom\_polygon(data = mask, aes(x = long, y = lat, group = group),

color = 'black', fill = 'transparent', size = 1.0) +

xlab('Longitude (°)') + ylab('Latitude (°)') +

theme\_minimal() +

theme(axis.line = element\_line(size = 1.0, colour = 'black'),

axis.title = element\_text(size = 14, face = 'bold'),

axis.title.x = element\_text(margin = margin(t = 20)),

axis.title.y = element\_text(margin = margin(r = 20)),

axis.text = element\_text(size = 12),

legend.title = element\_text(size = 14, face = 'bold', margin = margin(b = 10)),

legend.text = element\_text(size = 12))

######## predict to whole study extent

env\_select2 <- raster::stack(subset(env, names(env\_select)))

plot(env\_select2[[1]])

dismo.pred2 <- dismo::predict(object = dismo.mod, x = env\_select2)

## generate averaged model prediction

dismo.pred2.avg <- mean(dismo.pred2[[1]], dismo.pred2[[2]], dismo.pred2[[3]],

dismo.pred2[[4]], dismo.pred2[[5]], dismo.pred2[[6]],

dismo.pred2[[7]], dismo.pred2[[8]], dismo.pred2[[9]],

dismo.pred2[[10]])

plot(dismo.pred2.avg)

writeRaster(dismo.pred2.avg, 'SDM/output\_model\_grids/full\_area\_avg\_dismo.tif')

## plot ggplot style ::: continuous ::: whole area

## set font

windowsFonts(a = windowsFont('Times New Roman'))

gplot(dismo.pred2.avg) +

geom\_tile(aes(fill = value)) +

coord\_equal() +

scale\_fill\_gradientn(colours = c('#2c7bb6', '#abd9e9', '#ffffbf', '#fdae61', '#d7191c'),

na.value = 'transparent',

name = 'Suitability') +

geom\_polygon(data = mask, aes(x = long, y = lat, group = group),

color = 'black', fill = 'transparent', size = 1.0) +

geom\_polygon(data = buff, aes(x = long, y = lat, group = group),

color = '#FF5722', fill = 'transparent', size = 1.7,

linetype = 1) +

xlab('Longitude (°)') + ylab('Latitude (°)') +

theme\_minimal() +

theme(text = element\_text(family = 'a'),

axis.title = element\_text(size = 14, face = 'bold'),

axis.title.x = element\_text(margin = margin(t = 20)),

axis.title.y = element\_text(margin = margin(r = 20)),

axis.text = element\_text(size = 12),

legend.title = element\_text(size = 14, face = 'bold', margin = margin(b = 10)),

legend.text = element\_text(size = 12),

panel.border = element\_rect(fill = 'transparent',

color = 'black', size = 1.0))

###### generate binary grids ::: get value from MaxEnt result .csv file

## 10p threshold = 0.2578

## MTSS (test) threshold = 0.4154

bin.full.mod.10p <- ecospat::ecospat.binary.model(Pred = dismo.pred2.avg, Threshold = 0.2578)

bin.full.mod.mtss <- ecospat::ecospat.binary.model(Pred = dismo.pred2.avg, Threshold = 0.4154)

plot(bin.full.mod.10p)

plot(bin.full.mod.mtss)

writeRaster(bin.full.mod.10p, 'SDM/output\_model\_grids/bin.full.mod.10p.tif')

writeRaster(bin.full.mod.mtss, 'SDM/output\_model\_grids/bin.full.mod.mtss.tif')

## plot ggplot style ::: binary ::: 10p

gplot(bin.full.mod.10p) +

geom\_tile(aes(fill = value)) +

coord\_equal() +

scale\_fill\_gradientn(colors = c('lightgrey', '#1E88E5'),

na.value = 'transparent') +

geom\_polygon(data = mask, aes(x = long, y = lat, group = group),

fill = 'transparent', color = 'black', size = 1.0) +

geom\_polygon(data = buff, aes(x = long, y = lat, group = group),

fill = 'transparent', color = '#FF5722', size = 1.7,

linetype = 1) +

xlab('Longitude (°)') + ylab('Latitude (°)') +

theme\_minimal() +

theme(text = element\_text(family = 'a'),

axis.title = element\_text(size = 14, face = 'bold'),

axis.title.x = element\_text(margin = margin(t = 20)),

axis.title.y = element\_text(margin = margin(r = 20)),

axis.text = element\_text(size = 12),

legend.position = 'none',

panel.border = element\_rect(fill = 'transparent',

color = 'black', size = 1.0))

## plot ggplot style ::: binary ::: mtss (test)

gplot(bin.full.mod.mtss) +

geom\_tile(aes(fill = value)) +

coord\_equal() +

scale\_fill\_gradientn(colors = c('lightgrey', '#1E88E5'),

na.value = 'transparent') +

geom\_polygon(data = mask, aes(x = long, y = lat, group = group),

fill = 'transparent', color = 'black', size = 1.0) +

geom\_polygon(data = buff, aes(x = long, y = lat, group = group),

fill = 'transparent', color = '#FF5722', size = 1.7,

linetype = 1) +

xlab('Longitude (°)') + ylab('Latitude (°)') +

theme\_minimal() +

theme(text = element\_text(family = 'a'),

axis.title = element\_text(size = 14, face = 'bold'),

axis.title.x = element\_text(margin = margin(t = 20)),

axis.title.y = element\_text(margin = margin(r = 20)),

axis.text = element\_text(size = 12),

legend.position = 'none',

panel.border = element\_rect(fill = 'transparent',

color = 'black', size = 1.0))

## plot ggplot style ::: in rasterStack ::: side-by-side mapping

bin.stack <- raster::stack(bin.full.mod.10p, bin.full.mod.mtss)

names(bin.stack) <- c('P10', 'MTSS')

gplot(bin.stack) +

geom\_tile(aes(fill = value)) +

facet\_wrap(~ variable) +

coord\_equal() +

scale\_fill\_gradientn(colors = c('lightgrey', '#1E88E5'),

na.value = 'transparent') +

geom\_polygon(data = mask, aes(x = long, y = lat, group = group),

fill = 'transparent', color = 'black', size = 1.0) +

geom\_polygon(data = buff, aes(x = long, y = lat, group = group),

fill = 'transparent', color = '#FF5722', size = 1.7,

linetype = 1) +

xlab('Longitude (°)') + ylab('Latitude (°)') +

theme\_bw() +

theme(text = element\_text(family = 'a'),

axis.title = element\_text(size = 14, face = 'bold'),

axis.title.x = element\_text(margin = margin(t = 20)),

axis.title.y = element\_text(margin = margin(r = 20)),

axis.text = element\_text(size = 12),

strip.text = element\_text(size = 14),

legend.position = 'none')

######## evaluate models

## TSS

## null models ::: 100 iterations

mod.null <- ENMnulls(eval, mod.settings = list(fc = 'H', rm = 0.5), no.iter = 100)

null.res <- null.results(mod.null)

null.algorithm(mod.null)

head(null.res$auc.val.avg)

mean(null.res$auc.val.avg)

evalplot.nulls(mod.null, stats = c('auc.val', 'or.10p'), plot.type = 'histogram')

## plot auc.val of empirical & null models

# make data

head(eval.res$auc.val.avg)

head(null.res$auc.val.avg)

emp.auc <- as.data.frame(eval.res$auc.val.avg)

emp.auc$type <- 'Empirical'

colnames(emp.auc) = c('AUC', 'type')

head(emp.auc)

null.auc <- as.data.frame(null.res$auc.val.avg)

null.auc$type <- 'Null'

colnames(null.auc) = colnames(emp.auc)

head(null.auc)

auc <- rbind(emp.auc, null.auc)

# plot

auc %>%

ggplot(aes(x = type, y = AUC, fill = type, group = type)) +

geom\_boxplot(size = 1.2, width = 0.5) +

xlab('Model') +

ylab('Test AUC') +

theme(text = element\_text(family = 'a'),

panel.border = element\_rect(size = 1.0, fill = 'transparent'),

legend.position = 'none',

axis.title = element\_text(size = 14, face = 'bold'),

axis.title.x = element\_text(margin = margin(t = 20)),

axis.title.y = element\_text(margin = margin(r = 20)),

axis.text = element\_text(size = 12),

axis.text.x = element\_text(margin = margin(t = 8)),

axis.text.y = element\_text(margin = margin(r = 8)))

# statistical comparisons

head(emp.auc)

histogram(emp.auc[[1]])

shapiro.test(emp.auc[[1]])

head(null.auc)

histogram(null.auc[[1]])

shapiro.test(null.auc[[1]])

wilcox.test(x = emp.auc[[1]], y = null.auc[[1]])

######## plot response curves

## read plot data

bio3 <- read.csv('SDM/resp\_csv/species\_wc2.1\_30s\_bio\_3.csv')

bio8 <- read.csv('SDM/resp\_csv/species\_wc2.1\_30s\_bio\_8.csv')

bio14 <- read.csv('SDM/resp\_csv/species\_wc2.1\_30s\_bio\_14.csv')

cultivated <- read.csv('SDM/resp\_csv/species\_cultivated.csv')

herb <- read.csv('SDM/resp\_csv/species\_herb.csv')

water <- read.csv('SDM/resp\_csv/species\_open\_water.csv')

slope <- read.csv('SDM/resp\_csv/species\_slope\_1km.csv')

urban <- read.csv('SDM/resp\_csv/species\_urban.csv')

head(bio3)

head(bio8)

head(bio14)

head(cultivated)

head(herb)

head(water)

head(slope)

head(urban)

## combine & recode

resp <- rbind(bio3, bio8, bio14, cultivated, herb, water, slope, urban)

resp$variable <- recode\_factor(resp$variable,

'wc2.1\_30s\_bio\_3' = 'Bio 3',

'wc2.1\_30s\_bio\_8' = 'Bio 8',

'wc2.1\_30s\_bio\_14' = 'Bio 14',

'cultivated' = 'Cultivated',

'herb' = 'Herbaceous',

'open\_water' = 'Open water',

'slope\_1km' = 'Slope',

'urban' = 'Urban area')

head(resp)

tail(resp)

## plot

min <- resp$y - sd(resp$y, na.rm = T) # sd

max <- resp$y + sd(resp$y, na.rm = T) # sd

resp$variable = factor(resp$variable,

levels = c('Bio 3', 'Bio 8', 'Bio 14', 'Slope', 'Cultivated',

'Herbaceous', 'Open water', 'Urban area'))

resp %>%

ggplot(aes(x = x, y = y)) +

geom\_ribbon(aes(ymin = min, ymax = max), fill = 'lightgrey') +

geom\_line(size = 1.2, color = '#1976D2') +

facet\_wrap(~ variable, nrow = 2, ncol = 4, scales = 'free\_x') +

xlab('Value') + ylab('Suitability') +

theme\_bw() +

theme(text = element\_text(family = 'a'),

axis.title = element\_text(size = 16, face = 'bold'),

axis.title.x = element\_text(margin = margin(t = 20)),

axis.title.y = element\_text(margin = margin(r = 20)),

axis.text = element\_text(size = 12),

strip.text = element\_text(size = 14))