# Wallace Session overview:

The R code history from *Wallace* v2.0.5 session written as a R Markdown file in RStudio.

This R code was used in the manuscript Harman et al. 2023 Predicted range expansion of the larger grain borer, Prostephanus truncatus (Coleoptera: Bostrichidae), under projected climate change scenarios

Each code block is called a “chunk”, and you can run them either one-by-one or all at once by choosing an option in the “Run” menu at the top-right corner of the “Source” pane in RStudio.

For more detailed information see <http://rmarkdown.rstudio.com>).

### Package installation

Wallace uses the following R packages that must be installed and loaded before starting.

library(spocc)  
library(spThin)  
library(dismo)  
library(rgeos)  
library(ENMeval)  
library(wallace)

The *Wallace* session code .Rmd file is composed of a chain of module functions that are internal to *Wallace*. Each of these functions corresponds to a single module that the user ran during the session. To see the internal code for these module functions, click on the links in the .Rmd file. Users are encouraged to write custom code in the .Rmd directly to modify their analysis, and even modify the module function code to further customize. To see the source code for any module function, just type its name into the R console and press Return.

# example:  
# just type the function name and press Return to see its source code  
# paste this code into a new script to edit it  
occs\_queryDb

Analyses perfrmed in *Wallace* are below.

## Analysis for *P truncatus* (Pt)

User CSV path with occurrence data. If the CSV file is not in the current workspace, change to the correct file path (e.g. “/Users/darwin/Documents/occs/”).

# NOTE: provide the folder path of the .csv file  
occs\_path <- ""  
occs\_path <- file.path(occs\_path, "prostephanustruncatus\_WORLD southern pts Wallace.csv")  
# get a list of species occurrence data  
userOccs\_Pt <- occs\_userOccs(  
 txtPath = occs\_path,   
 txtName = "prostephanustruncatus\_WORLD southern pts Wallace.csv",   
 txtSep = ",",   
 txtDec = ".")  
occs\_Pt <- userOccs\_Pt$P\_truncatus$cleaned

### Obtain environmental data

Using WorldClim (<http://www.worldclim.org/>) bioclimatic dataset at resolution of 2.5 arcmin.

# Download environmental data   
envs\_Pt <- envs\_worldclim(  
 bcRes = 2.5,   
 bcSel = c('bio01', 'bio02', 'bio03', 'bio04', 'bio05', 'bio06', 'bio07', 'bio08', 'bio09', 'bio10', 'bio11', 'bio12', 'bio13', 'bio14', 'bio15', 'bio16', 'bio17', 'bio18', 'bio19'),   
 mapCntr = c(-37.039, 11.707), # Mandatory for 30 arcsec resolution   
 doBrick = FALSE)  
occs\_xy\_Pt <- occs\_Pt[c('longitude', 'latitude')]  
occs\_vals\_Pt <- as.data.frame(raster::extract(envs\_Pt, occs\_xy\_Pt, cellnumbers = TRUE))  
# Remove duplicated same cell values  
occs\_Pt <- occs\_Pt[!duplicated(occs\_vals\_Pt[, 1]), ]  
occs\_vals\_Pt <- occs\_vals\_Pt[!duplicated(occs\_vals\_Pt[, 1]), -1]  
# remove occurrence records with NA environmental values  
occs\_Pt <- occs\_Pt[!(rowSums(is.na(occs\_vals\_Pt)) >= 1), ]  
# also remove variable value rows with NA environmental values  
occs\_vals\_Pt <- na.omit(occs\_vals\_Pt)  
# add columns for env variable values for each occurrence record  
occs\_Pt <- cbind(occs\_Pt, occs\_vals\_Pt)

### Process Occurrence Data

Thinning the occurrences to 10 km

# Thin occurrences   
occs\_Pt <- poccs\_thinOccs(  
 occs = occs\_Pt,   
 thinDist = 10)

### Process environmental data

Sampling of 50000 background points and corresponding environmental data using a “point buffers” method with a 2 degree buffer.

# Generate background extent   
bgExt\_Pt <- penvs\_bgExtent(  
 occs = occs\_Pt,  
 bgSel = "point buffers",  
 bgBuf = 2)  
# Mask environmental data to provided extent  
bgMask\_Pt <- penvs\_bgMask(  
 occs = occs\_Pt,  
 envs = envs\_Pt,  
 bgExt = bgExt\_Pt)  
# Sample background points from the provided area  
bgSample\_Pt <- penvs\_bgSample(  
 occs = occs\_Pt,  
 bgMask = bgMask\_Pt,  
 bgPtsNum = 50000)  
# Extract values of environmental layers for each background point  
bgEnvsVals\_Pt <- as.data.frame(raster::extract(bgMask\_Pt, bgSample\_Pt))  
##Add extracted values to background points table  
bgEnvsVals\_Pt <- cbind(scientific\_name = paste0("bg\_", "P truncatus"), bgSample\_Pt,  
 occID = NA, year = NA, institution\_code = NA, country = NA,  
 state\_province = NA, locality = NA, elevation = NA,  
 record\_type = NA, bgEnvsVals\_Pt)

### Partition occurrence data

Partition occurrences and background points for model training and validation using “spatial block”, a spatial partition method with an aggregation factor of 2.

# R code to get partitioned data  
groups\_Pt <- part\_partitionOccs(  
 occs = occs\_Pt ,  
 bg = bgSample\_Pt,   
 method = "block",  
 bgMask = bgMask\_Pt,  
 aggFact = 2)

### Build and Evaluate Niche Model

Generating a species distribution model using the maxnet algorithm as implemented in ENMeval V2.0 (with clamping = TRUE). For tuning using L, LQ, H, LQH feature classes and regularization multipliers in the 1, 4 range increasing by 1. Not using any categorical predictor variables.

# Run maxent model for the selected species  
model\_Pt <- model\_maxent(  
 occs = occs\_Pt,  
 bg = bgEnvsVals\_Pt,  
 user.grp = groups\_Pt,   
 bgMsk = bgMask\_Pt,  
 rms = c(1, 4),   
 rmsStep = 1,  
 fcs = c('L', 'LQ', 'H', 'LQH'),  
 clampSel = TRUE,  
 algMaxent = "maxnet",  
 parallel = FALSE,  
 numCores = 11)

### Visualize

Generate a map of the maxnet generated model with no threshold

# Select current model and obtain raster prediction  
m\_Pt <- model\_Pt@models[["fc.LQ\_rm.1"]]  
predSel\_Pt <- predictMaxnet(m\_Pt, bgMask\_Pt,  
 type = "cloglog",   
 clamp = TRUE)  
#Get values of prediction  
mapPredVals\_Pt <- getRasterVals(predSel\_Pt, "cloglog")  
#Define colors and legend   
rasCols <- c("#2c7bb6", "#abd9e9", "#ffffbf", "#fdae61", "#d7191c")  
legendPal <- colorNumeric(rev(rasCols), mapPredVals\_Pt, na.color = 'transparent')  
rasPal <- colorNumeric(rasCols, mapPredVals\_Pt, na.color = 'transparent')  
#Generate map  
m <- leaflet() %>% addProviderTiles(providers$Esri.WorldTopoMap)   
m %>%  
 leaflet::addLegend("bottomright", pal = legendPal,  
 title = "Predicted Suitability<br>(Training)",  
 values = mapPredVals\_Pt, layerId = "train",  
 labFormat = reverseLabel(2, reverse\_order = TRUE)) %>%   
 #add occurrence data  
 addCircleMarkers(data = occs\_Pt, lat = ~latitude, lng = ~longitude,  
 radius = 5, color = 'red', fill = TRUE, fillColor = "red",  
 fillOpacity = 0.2, weight = 2, popup = ~pop) %>%   
 ##Add model prediction  
 addRasterImage(predSel\_Pt, colors = rasPal, opacity = 0.7,  
 group = 'vis', layerId = 'mapPred', method = "ngb") %>%  
 ##add background polygons  
 addPolygons(data = bgExt\_Pt,fill = FALSE,  
 weight = 4, color = "blue", group = 'proj')

### Visualize

Generate a Maxent evaluation plot using “auc.val” as evaluation statistic.

# Generate an evaluation plot  
maxentEvalPlot\_Pt <- ENMeval::evalplot.stats(  
 model\_Pt,  
 "auc.val",  
 "rm",  
 "fc")  
#plot  
maxentEvalPlot\_Pt

## Transfer model

Transfering the model to a new user provided area

# First must generate the transfer area based on user provided files  
 ##User must input the path to shapefile or csv file and the file name   
xfer\_userExt\_Pt <- xfer\_userExtent(  
 bgShp\_path = "Input path here",  
 bgShp\_name = "Input file name here",  
 userBgBuf = 0.5)  
  
# Create object of transfer variables  
xferAreaEnvs\_Pt <- envs\_Pt  
# Generate a transfer of the model to the desired area  
xfer\_area\_Pt <- xfer\_area(  
 evalOut = model\_Pt,  
 curModel = "fc.LQ\_rm.1",  
 envs = xferAreaEnvs\_Pt ,   
 outputType = "cloglog",  
 alg = "maxnet",  
 clamp = TRUE,  
 xfExt = xfer\_userExt\_Pt)   
#store the cropped transfer variables  
xferExt\_Pt <- xfer\_area\_Pt$xferExt  
   
 ###Make map of transfer  
bb\_Pt <- bgExt\_Pt@bbox  
bbZoom <- polyZoom(bb\_Pt[1, 1], bb\_Pt[2, 1], bb\_Pt[1, 2],   
 bb\_Pt[2, 2], fraction = 0.05)  
mapXferVals\_Pt <- getRasterVals(xfer\_area\_Pt$xferArea,"cloglog")  
rasCols\_Pt <- c("#2c7bb6", "#abd9e9", "#ffffbf", "#fdae61", "#d7191c")  
# if no threshold specified  
legendPal <- colorNumeric(rev(rasCols\_Pt), mapXferVals\_Pt, na.color = 'transparent')  
rasPal\_Pt <- colorNumeric(rasCols\_Pt, mapXferVals\_Pt, na.color = 'transparent')  
m <- leaflet() %>% addProviderTiles(providers$Esri.WorldTopoMap)   
m %>%  
 fitBounds(bbZoom[1], bbZoom[2], bbZoom[3], bbZoom[4]) %>%  
 leaflet::addLegend("bottomright", pal = legendPal,  
 title = "Predicted Suitability<br>(Transferred)",  
 values = mapXferVals\_Pt, layerId = 'xfer',  
 labFormat = reverseLabel(2, reverse\_order = TRUE)) %>%  
# map model prediction raster and transfer polygon  
 clearMarkers() %>% clearShapes() %>% removeImage('xferRas') %>%  
 addRasterImage(xfer\_area\_Pt$xferArea, colors = rasPal\_Pt, opacity = 0.7,  
 layerId = 'xferRas', group = 'xfer', method = "ngb") %>%  
 ##add transfer polygon (user provided area)  
 addPolygons(data = xfer\_userExt\_Pt, fill = FALSE,  
 weight = 4, color = "blue", group = 'xfer')

### Transfer model

Transferring the model to a user provided area area with no threshold. New time based on “WorldClim 1.4” variables for 2050 using a “CC” GCM and an RCP of *2.6*.

# Download variables for transferring   
xferTimeEnvs\_Pt <- raster::getData(  
 'CMIP5',  
 var = "bio",  
 res = round((raster::res(bgMask\_Pt) \* 60)[1],1),  
 rcp = 26,  
 model = "CC",  
 year = 50)  
  
names(xferTimeEnvs\_Pt) <- paste0('bio', c(paste0('0',1:9), 10:19))  
  
# Select variables for transferring to match variables used for modelling   
xferTimeEnvs\_Pt <- xferTimeEnvs\_Pt[[names(bgMask\_Pt)]]  
  
# Generate the area of transfer based on user provided files  
 ##User must input the path to shapefile or csv file and the file name   
xfer\_userExt\_Pt <- xfer\_userExtent(  
 bgShp\_path = "Input path here",  
 bgShp\_name = "Input file name here",  
 userBgBuf = 0.5)  
  
# Generate a transfer of the model to the desired area and time  
xfer\_time\_Pt <-xfer\_time(  
 evalOut = model\_Pt,  
 curModel = "fc.LQ\_rm.1",  
 envs = xferTimeEnvs\_Pt,  
 xfExt = xfer\_userExt\_Pt,  
 alg = "maxnet",  
 outputType = "cloglog",  
 clamp = TRUE  
 )   
# store the cropped variables of transfer  
xferExt\_Pt <- xfer\_time\_Pt$xferExt  
  
###Make map of transfer  
bb\_Pt <- bgExt\_Pt@bbox  
bbZoom <- polyZoom(bb\_Pt[1, 1], bb\_Pt[2, 1], bb\_Pt[1, 2],   
 bb\_Pt[2, 2], fraction = 0.05)  
mapXferVals\_Pt <- getRasterVals(xfer\_time\_Pt$xferTime,"cloglog")  
rasCols\_Pt <- c("#2c7bb6", "#abd9e9", "#ffffbf", "#fdae61", "#d7191c")  
# if no threshold specified  
legendPal <- colorNumeric(rev(rasCols\_Pt), mapXferVals\_Pt,   
 na.color = 'transparent')  
rasPal\_Pt <- colorNumeric(rasCols\_Pt, mapXferVals\_Pt,   
 na.color = 'transparent')  
m <- leaflet() %>% addProviderTiles(providers$Esri.WorldTopoMap)   
m %>%  
 fitBounds(bbZoom[1], bbZoom[2], bbZoom[3], bbZoom[4]) %>%  
 leaflet::addLegend("bottomright", pal = legendPal,  
 title = "Predicted Suitability<br>(Transferred)",  
 values = mapXferVals\_Pt, layerId = 'xfer',  
 labFormat = reverseLabel(2, reverse\_order = TRUE)) %>%  
# map model prediction raster and polygon of transfer  
 clearMarkers() %>% clearShapes() %>% removeImage('xferRas') %>%  
 addRasterImage(xfer\_time\_Pt$xferTime, colors = rasPal\_Pt, opacity = 0.7,  
 layerId = 'xferRas', group = 'xfer', method = "ngb") %>%  
 ##add polygon of transfer (user provided area)  
 addPolygons(data = xfer\_userExt\_Pt, fill = FALSE,  
 weight = 4, color = "blue", group = 'xfer')

Generate a MESS map for the transferring variables given the variables used for modelling

# R code to generate MESS raster  
xferMess\_Pt <- xfer\_mess(  
 occs = occs\_Pt,  
 bg = bgEnvsVals\_Pt ,  
 bgMsk = bgMask\_Pt,  
 xferExtRas = xferExt\_Pt)  
  
# Generate MESS map  
rasVals\_Pt <- getRasterVals(xferMess\_Pt)  
  
 # define colorRamp for mess  
 if (max(rasVals\_Pt) > 0 & min(rasVals\_Pt) < 0) {  
 rc1 <- colorRampPalette(colors = rev(RColorBrewer::brewer.pal(n = 3, name = 'Reds')),  
 space = "Lab")(abs(min(rasVals\_Pt)))  
 rc2 <- colorRampPalette(colors = RColorBrewer::brewer.pal(n = 3, name = 'Blues'),  
 space = "Lab")(max(rasVals\_Pt))  
 rasCols\_Pt <- c(rc1, rc2)  
 } else if (max(rasVals\_Pt) < 0 & min(rasVals\_Pt) < 0) {  
 rasCols\_Pt <- colorRampPalette(colors = rev(RColorBrewer::brewer.pal(n = 3, name = 'Reds')),  
 space = "Lab")(abs(min(rasVals)))  
 } else if (max(rasVals\_Pt) > 0 & min(rasVals\_Pt) > 0) {  
 rasColsPt <- colorRampPalette(colors = RColorBrewer::brewer.pal(n = 3, name = 'Blues'),  
 space = "Lab")(max(rasVals\_Pt))  
 }  
 legendPal\_Pt <- colorNumeric(rev(rasCols\_Pt), rasVals\_Pt, na.color='transparent')  
 rasPal\_Pt <- colorNumeric(rasCols\_Pt, rasVals\_Pt, na.color='transparent')  
 #Create map  
 m <- leaflet() %>% addProviderTiles(providers$Esri.WorldTopoMap)   
m %>%  
 leaflet::addLegend("bottomright", pal = legendPal\_Pt, title = "MESS Values",  
 values = rasVals\_Pt, layerId = 'xfer',  
 labFormat = reverseLabel(2, reverse\_order=TRUE)) %>%  
 # map model prediction raster and transferring polygon  
 clearMarkers() %>% clearShapes() %>% removeImage('xferRas') %>%  
 addRasterImage(xferMess\_Pt, colors = rasPal\_Pt , opacity = 0.9,  
 layerId = 'xferRas', group = 'xfer', method = "ngb") %>%  
 ##add transferring polygon: this we need to fix for now please replace bgExt\_Pt for the name of your transferring polygon.  
 addPolygons(data = bgExt\_Pt, fill = FALSE,  
 weight = 4, color = "blue", group = 'xfer')

## ESPACE analysis for *Africa truncatus* and *Americas truncatus* (Aft\_Amt)

### Environmental space

Performing and plotting principal component analysis to reduce dimensionality of environmental space for *Africa truncatus* & *Americas truncatus*. PCA done for occs.

# Determine the variables to use  
pcaSel\_Aft\_Amt <- c('bio01', 'bio02', 'bio03', 'bio04', 'bio05', 'bio06', 'bio07', 'bio08', 'bio09', 'bio10', 'bio11', 'bio12', 'bio13', 'bio14', 'bio15', 'bio16', 'bio17', 'bio18', 'bio19')  
# Run the pca  
espace\_pca\_Aft\_Amt <- espace\_pca(  
 sp.name1 = "Africa truncatus",  
 sp.name2 = "Americas truncatus",   
 occs.z1 = occs\_Aft[,pcaSel\_Aft\_Amt],  
 occs.z2 = occs\_Amt[,pcaSel\_Aft\_Amt],  
 bgPts.z1 = bgEnvsVals\_Aft[,pcaSel\_Aft\_Amt],  
 bgPts.z2 = bgEnvsVals\_Amt[,pcaSel\_Aft\_Amt])  
  
## Generate plots  
# PCA Scatter Plot  
if ("occs" == "occs") {  
 x <- espace\_pca\_Aft\_Amt$scores[espace\_pca\_Aft\_Amt$scores$bg == 'sp', ]  
 x.f <- factor(x$sp)  
} else if ("occs" == "occsBg") {  
 x <- espace\_pca\_Aft\_Amt$scores[espace\_pca\_Aft\_Amt$scores$sp == 'bg', ]  
 x.f <- factor(x$bg)  
}  
ade4::s.class(x, x.f, xax = 1, yax = 2,  
 col = c("red", "blue"), cstar = 0, cpoint = 0.1)  
title(xlab = paste0("PC", 1), ylab = paste0("PC", 2))  
# PCA Correlation circle  
ade4::s.corcircle(espace\_pca\_Aft\_Amt$co, xax = 1, yax = 2,  
 lab = pcaSel\_Aft\_Amt, full = FALSE, box = TRUE)  
 title(xlab = paste0("PC", 1),  
 ylab = paste0("PC", 2))  
# PCA screeplot  
screeplot(espace\_pca\_Aft\_Amt, main = NULL)  
# Print PCA summary of results  
summary(espace\_pca\_Aft\_Amt)

### Environmental space

Calculating the part of environmental space more densly populated by species & the availability of environmental conditions in the background for *Africa truncatus* & *Americas truncatus*

# Create density grid  
espace\_occDens\_Aft\_Amt <- espace\_occDens(  
 sp.name1 = "Africa truncatus",  
 sp.name2 = "Americas truncatus",   
 pca = espace\_pca\_Aft\_Amt)   
# Plots  
graphics::par(mfrow = c(1,2))  
ecospat::ecospat.plot.niche(espace\_occDens\_Aft\_Amt[["Africa truncatus"]],   
 title = "Africa truncatus")  
ecospat::ecospat.plot.niche(espace\_occDens\_Aft\_Amt[["Americas truncatus"]],   
 title = "Americas truncatus")

### Environmental space

Evaluating niche overlap between *Africa truncatus* & *Americas truncatus* for which the occurrence density grid was computed. Running equivalence test (FALSE) and similarity test TRUE

## Run tests   
espace\_nicheOv\_Aft\_Amt <- espace\_nicheOv(  
 z1 = espace\_occDens\_Aft\_Amt[["Africa truncatus"]],  
 z2 = espace\_occDens\_Aft\_Amt[["Americas truncatus"]],  
 iter = 100,  
 similarity = TRUE)   
# Plots  
layout(matrix(c(1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 3, 3), 4, 3, byrow = FALSE))  
  
ecospat::ecospat.plot.niche.dyn(  
 espace\_occDens\_Aft\_Amt[["Africa truncatus"]],  
 espace\_occDens\_Aft\_Amt[["Americas truncatus"]],  
 0.5,  
 title = "Africa truncatus\_Americas truncatus",  
 col.unf = "blue",  
 col.exp = "red",  
 col.stab = "purple",  
 colZ1 = "blue",  
 colZ2 = "red",  
 transparency = 25  
)  
# Plot  
ecospat::ecospat.plot.overlap.test(espace\_nicheOv\_Aft\_Amt$simil,  
 "D", "Similarity test")