Supplement 1. Script in R for distribution modeling in *Microtus arvalis*.

```
library(maxnet)
library(raster)
library(stringr)
library(ENMeval)
library(rgeos)
library(RStoolbox)
library(ncf)
library(spdep)

base_dir <- "/var/www/rusmam/modeling/data/44/"
csv.file <- "species.csv"
bias.file <- "bias.csv"
env.file <- list.files(path="/var/www/rusmam/modeling/coverage/er",pattern = ".asc$",full.names
= T,include.dirs = F)
buffer.val <- c(2000, 3500, 5000)
Rs <- c(0.75, 1, 2, 3)
Pred.fs <- c("L", "LQ", "LQH")
env.proc <- "predictors"
max.bckg <- 10000
max.pc <- 10000
AUC.thr <- 0.6
pc.threshold <- c(0.99996)
NCores <- 2
dist.threshold <- 41000
meth.bg <- "maxent-like"

control_bckg   <- 'no'
mask_train_sample <- 'no'
result_type    <- 'cloglog'
range_buf     <- 70000
cut_buff     <- 800000
bias_range    <- 0
bias_occs     <- 2
project_res    <- 'yes'
if(project_res == 'yes')
{
 proj.file     <- list.files(path="/var/www/rusmam/modeling/coverage/moder",pattern =
".asc$",full.names = T,include.dirs = F)
}

prj       <- CRS("+proj=moll +lon_0=30 +x_0=3335846.22854 +y_0=-336410.83237
+ellps=WGS84 +datum=WGS84 +units=m +no_defs")
prj_ll      <- CRS("+proj=longlat +ellps=WGS84 +datum=WGS84 +no_defs")
meth       <- 'maxent.jar'
pdf(file = paste(base_dir, 'Rplots.pdf', sep=""))
max.par     <- 0
num.pc      <- 0
csv.dat     <- read.csv(paste(base_dir, csv.file, sep=""))
csv.spp     <- SpatialPoints(cbind(csv.dat[,2],csv.dat[,3]),prj)
bias.dat     <- read.csv(paste(base_dir, bias.file, sep=""))
```

```
bias.spp     <- SpatialPoints(cbind(bias.dat[,2],bias.dat[,3]),prj)
model.name    <- csv.dat[1,1]

if (grepl( '/er', env.file, fixed = TRUE)[1]==TRUE)
{
 options(java.parameters = "-Xmx5120m")
} else {
 options(java.parameters = "-Xmx8192m")
}
library(dismo)

killStack <- function(x){
 x <- dropLayer(x, 1:nlayers(x))
}

time_start    <- Sys.time()
cat("\n****", model.name, " distribution modelling****", date(), "\n")
write.table(paste0(model.name, " distribution modelling ", date()), file=paste(base_dir, "result",
".csv", sep=""), append = FALSE, row.names = F, col.names = T, sep = ",")
env.stack    <- lapply(env.file,raster)
env.stack    <- stack(env.stack)
crs(env.stack)  <- prj
one.ras     <- env.stack[[1]]/env.stack[[1]]

env.max.num   <- length(env.stack[[1]][!is.na(env.stack[[1]])])
csv.out     <- extract(one.ras, csv.spp)
csv.spp     <- csv.spp[which(csv.out>0)]
cat("\nНачинаем расчет\nВсего в каждом предикторе ", env.max.num, " ячеек со
значениями.")
cat("\nПрореженных точек ", model.name, ": ", length(csv.spp), "\n")
distmat <-pointDistance(csv.spp@coords, csv.spp@coords, type='Euclidean', lonlat = F, allpairs
= TRUE)
hist(distmat[-which(distmat==0)], main="Between-points distance", include.lowest = TRUE)
hist(distmat[-which(distmat==0)], main="Between-points distance (part)", xlim=c(0,50000),
include.lowest = TRUE, nclass = 500)
dist.diag <- lower.tri(distmat, diag = T)
distmat[dist.diag] <- 0
csv.out <- which(distmat < dist.threshold & distmat > 0, arr.ind=TRUE)
csv.out <- unique(csv.out[,1])
cat("\nВсего ", length(csv.out), " пар точек находятся на расстоянии меньше ",
dist.threshold/1000, " км друг от друга.\n")
if(length(csv.out) > 0)
{
 csv.spp <- csv.spp[-csv.out]
 distmat <- pointDistance(csv.spp@coords, csv.spp@coords, type='Euclidean', lonlat = F,
allpairs = TRUE)
 dist.diag <- lower.tri(distmat, diag = T)
 distmat[dist.diag] <- 0
 csv.out <- which(distmat < dist.threshold & distmat > 0, arr.ind=TRUE)
 csv.out <- unique(csv.out[,1])
 cat("\nОсталось ", length(csv.out), " пар точек на расстоянии меньше ", dist.threshold/1000, "
км друг от друга.")
```

```
 cat("\nТеперь прореженных точек ", model.name, ": ", length(csv.spp), "\n")
}

test.dat     <- bias.dat[bias.dat$Species == model.name,]
test.spp     <- SpatialPoints(cbind(test.dat[,2],test.dat[,3]),prj)
env.temp     <- one.ras
env.temp[cellFromXY(env.temp, csv.spp@coords)] <- NA
if (mask_train_sample=="yes")
{
 test.out    <- extract(env.temp, test.spp)
 test.spp    <- test.spp[which(test.out>0)]
} else {
 test.out    <- extract(one.ras, test.spp)
 test.spp    <- test.spp[which(test.out>0)]
}
if(dist.threshold < 25000) agg_coef <- 25*1.1 else agg_coef <- dist.threshold*1.1/1000
test.spp     <- SpatialPoints(gridSample(test.spp, aggregate(env.stack[[1]], agg_coef), 1), prj)
cat("\nОбъем тестовой выборки ", model.name, ": ", length(test.spp), "\n")
distmat     <- pointDistance(test.spp@coords, csv.spp@coords, type='Euclidean', lonlat = F,
allpairs = TRUE)
cat("\nВ тестовом и обучающем наборах совпадают ", length(distmat[distmat==0]), "
точек.\n")
plot(one.ras, main=paste("Train and test points"))
plot(csv.spp, pch=1, col="blue", add=T)
plot(test.spp, col="red", add=T)
legend("bottomright", legend = c("Train", "Test"), col = c("blue", "red"), pch = c(1, 3))

if(control_bckg=="yes")
{
 crop.buf    <- buffer(rbind(csv.spp, test.spp), width=cut_buff)
 env.stack    <- mask(env.stack, crop.buf)
 env.stack    <- trim(env.stack)
 one.ras     <- mask(one.ras, crop.buf)
 one.ras     <- trim(one.ras)
 env.temp    <- mask(env.temp, crop.buf)
 env.temp    <- trim(env.temp)
}

g = 0
for (i in 1:length(buffer.val))
{
 bias.buf      <- buffer(bias.spp, width=buffer.val[i])
 if(meth.bg=="maxent-like")
 {
  bias_strict.buf <- buffer(rbind(csv.spp, test.spp), width=range_buf)
  bias.buf     <- bias.buf - bias_strict.buf
  bg.ras      <- rasterize(bias.buf, env.temp, field=bias_occs, update = T, updateValue = '!NA')
  bg2.ras     <- rasterize(bias_strict.buf, env.temp, field=bias_range, update = T, updateValue =
'!NA')
  bg.ras      <- bg.ras + bg2.ras
  writeRaster(bg.ras, paste0(base_dir,'bias.asc'), format="ascii", overwrite=T, datatype='INT2S',
NAflag=-9999, prj=T)
```

```
 bg.num      <- length(bg.ras[bg.ras>0])
 if (bg.num > max.bckg) bg.num <- max.bckg
 bg       <- as.data.frame(randomPoints(bg.ras, bg.num, prob = T, lonlatCorrection = F))
 plot(one.ras, main=paste("Background points"))
 plot(bg.ras, add=T)
 bg.spp      <- SpatialPoints(cbind(bg[,1],bg[,2]),prj)
 plot(bg.spp, add=T)
 }
 if(meth.bg=="strict")
 {
 bg       <- as.data.frame(extract(env.temp, bias.buf, cellnumbers = T, na.rm=T))
 bg       <- bg[!is.na(bg$value),]
 bg.num      <- length(bg[,1])
 if (bg.num > max.bckg) bg <- bg[sample(nrow(bg), max.bckg),]
 bg       <- as.data.frame(xyFromCell(env.temp, bg$cell))
 plot(one.ras, main=paste("Background points"))
 bg.spp      <- SpatialPoints(cbind(bg[,1],bg[,2]),prj)
 plot(bg.spp, add=T)
 }
 names(bg)     <- colnames(csv.spp@coords)
 bg.num       <- length(bg[,1])
 cat("\nФоновая выборка составила ", bg.num, " точек.\n")

 if (env.proc == "pc_loc")
 {
 bckg.dat     <- bias.dat
 bckg.spp     <- SpatialPoints(cbind(bckg.dat[,2],bckg.dat[,3]),prj)
 cat("\nНепрореженных точек для PCA: ", length(bckg.spp), "\n")
 train.buf    <- buffer(bckg.spp, width=buffer.val[i])
 train.stack   <- mask(env.stack, train.buf)
 train.stack   <- crop(train.stack, train.buf)
 train.num    <- length(train.stack[[1]][!is.na(train.stack[[1]])])
 if (train.num > max.pc || is.null(max.pc) == T) ns.pc <- max.pc else ns.pc <- train.num
 cat("\nPCA будет рассчитан по ", ns.pc, " точкам. Количество слоев ", nlayers(train.stack),
"\n")
 cat("\nОбучаем PCA\n")
 train.pc     <- rasterPCA(train.stack, spca = T)
 cat("\nPCA был рассчитан по ", train.pc$model$n.obs, " точкам.\n")
 killStack(train.stack)
 train.pc$map <- 0
 var.pc      <- (train.pc$model$sdev)^2/sum((train.pc$model$sdev)^2)
 cat("\nBuffer ", buffer.val[i], " factors loadings\n")
 if(length(var.pc)>7) ml = 7 else ml = length(var.pc)
 print(round(as.data.frame(train.pc$model$loadings[,1:ml]),2))
 cat("\nPart of total variance\n")
 cat(var.pc)
 }
 if (env.proc == "pc_global")
 {
 if (bg.num > max.pc || is.null(max.pc) == T) ns.pc <- max.pc else ns.pc <- bg.num
 cat("\nPCA будет рассчитан по ", ns.pc, " точкам.")
 cat("\nОбучаем PCA\n")
```

```
 train.pc     <- rasterPCA(env.stack, nSamples = ns.pc, maskCheck=F, spca = T)
 cat("\nPCA был рассчитан по ", train.pc$model$n.obs, " точкам.")
 var.pc      <- (train.pc$model$sdev)^2/sum((train.pc$model$sdev)^2)
 cat("\nBuffer ", buffer.val[i], " factors loadings\n")
 if(length(var.pc)>7) ml = 7 else ml = length(var.pc)
 print(round(as.data.frame(train.pc$model$loadings[,1:ml]),2))
 cat("\nPart of total variance\n")
 cat(var.pc)
}
if (env.proc == "predictors")
{
 env.pc      <- env.stack
 pc.threshold   <- 1
}
gc()

for (m in 1:length(pc.threshold))
{
 g = g + 1
 cat("\n Шаг ", g)
 if (env.proc != "predictors")
 {
  num.pc      <- min(which(cumsum(var.pc)>pc.threshold[m]))
  cat("\nПрименяем PCA (разворачиваем пространство для всех растров): ", num.pc, "
компонент описывают ",pc.threshold[m] , " процентов дисперсии.")
  env.pc      <- predict(env.stack, train.pc$model, index=1:num.pc)
  gc()
 }
 cat("\nСчитаем модель\n")

 mod       <- ENMevaluate(csv.spp@coords, env.pc, bg = bg, occs.testing = test.spp@coords,
tune.args = list(fc = Pred.fs, rm = Rs), partitions = "testing", algorithm = meth, doClamp = F,
other.settings = list(abs.auc.diff = T, pred.type = 'raw', validation.bg = 'full'), parallel = T,
numCores = NCores, overlap = F)
 list.preds    <- sapply(mod@variable.importance, function(x){
  ll <- which(x[,3] > 5)
  return(ll)
 })
 list.preds    <- unique(sort(unname(unlist(list.preds))))
 cat(length(list.preds), " переменных вносят вклад в модель.\n")
 mod       <- ENMevaluate(csv.spp@coords, env.pc[[list.preds]], bg = bg, occs.testing =
test.spp@coords, tune.args = list(fc = Pred.fs, rm = Rs), partitions = "testing", algorithm = meth,
doClamp = F, other.settings = list(abs.auc.diff = T, pred.type = 'raw', validation.bg = 'full'),
parallel = T, numCores = NCores, overlap = T)
 aicc.test    <- aic.maxent(as.data.frame(extract(mod@predictions, test.spp)),
mod@results$ncoef, mod@predictions)
 gc()
 print(mod@variable.importance)
 print(mod@results)
 print(aicc.test)
 print(mod@overlap)
```

```
  write.table(paste("Buffer =", buffer.val[i], ", PC.cut =", pc.threshold[m], sep=" "),
file=paste(base_dir, "result", ".csv", sep=""), append = TRUE, row.names = F, col.names = T,
sep = ",")
  write.table(cbind(mod@results, aicc.test), file=paste(base_dir, "result", ".csv", sep=""), append
= TRUE, row.names = F, col.names = T, sep = ",")
  if (is.na(min(mod@results,"AICc"])) == T)
  {
   max.par      <- max(max.par, max(mod@results,'ncoef']))
   cat("\nСреди моделей есть излишне параметризованные с ", max(mod@results,'ncoef']), "
параметрами.\n")
  }
  killStack(mod@predictions)
  list_fin.preds    <- which(mod@variable.importance[[which(aicc.test$delta.AICc == 0)[1]]][,3]
> 2)
  list_fin.preds    <- sort(unname(list_fin.preds))
  cat(length(list_fin.preds), " переменных вносят вклад в конечную модель.\n")
  mod         <- ENMevaluate(csv.spp@coords, env.pc[[list.preds]][[list_fin.preds]], bg = bg,
occs.testing = test.spp@coords, tune.args = list(fc =
levels(mod@results[,"fc"])[mod@results[which(aicc.test$delta.AICc == 0)[1],"fc"]], rm =
as.numeric(levels(mod@results[,"rm"])[mod@results[which(aicc.test$delta.AICc ==
0)[1],"rm"]])), partitions = "testing", algorithm = meth, doClamp = T, other.settings =
list(abs.auc.diff = T, pred.type = result_type, validation.bg = 'full'), parallel = T, numCores =
NCores, overlap = F)
  gc()
  candidate.ras    <- mod@predictions[[1]]

  if (exists("result.ras"))
  {
   result.ras     <- stack(result.ras, candidate.ras)
  }
  else
  {
   result.ras     <- candidate.ras
  }

  if (!exists("result"))
  {
   result       <- matrix(0,nrow=(length(buffer.val)*length(pc.threshold)), ncol=14);
   colnames(result)  <- c("Buffer","Npc","PCcut", "Bg
points","Features","R","train.AUC","or.10p","AICc.train","Pars", "AICc.test", "min.pres",
"10p.pres", "MTSS")
   result       <- as.data.frame(result, row.names = NULL, optional = F)
  }
  result[g,]      <- c(buffer.val[i], num.pc, pc.threshold[m], bg.num,
levels(mod@results[,"fc"])[mod@results[1,"fc"]],
levels(mod@results[,"rm"])[mod@results[1,"rm"]],
mod@results[1,c("auc.train","or.10p","AICc","ncoef")], aicc.test[1,"AICc"], 0, 0, 0)

  if (env.proc != "predictors")
  {
  if (!exists("pc.model")) pc.model <- list()
  pc.model[[g]]    <- train.pc$model
```

```
  if (!exists("pc.nums")) pc.nums <- list()
  pc.nums[[g]]     <- num.pc
  }

  if (!exists("models")) models <- list()
  models[[g]]     <- mod@models[[1]]

  if (!exists("importance")) importance <- list()
  importance[[g]]   <- mod@variable.importance[[1]]

  result[g,"MTSS"]   <- eval(parse(text = eval(expression(paste0("mod@models$",
mod@results[1,c("tune.args")], "@results['",
"Maximum.training.sensitivity.plus.specificity.Cloglog.threshold", "',]")))))
  result[g,"10p.pres"] <- eval(parse(text = eval(expression(paste0("mod@models$",
mod@results[1,c("tune.args")], "@results['",
"X10.percentile.training.presence.Cloglog.threshold", "',]")))))
  result[g,"min.pres"] <- eval(parse(text = eval(expression(paste0("mod@models$",
mod@results[1,c("tune.args")], "@results['", "Minimum.training.presence.Cloglog.threshold",
"',]")))))

  pal         <- colorRampPalette(c("blue","cyan","green","yellow","orange","red"))
  plot(result.ras[[g]], col=pal(12), main=paste("Buffer =", buffer.val[i], ", PC.cut =",
pc.threshold[m], sep=" "))
  plot(bias.buf, add=T)
  plot(csv.spp, add=T)
 }
}

killStack(env.stack)
if(project_res == 'yes')
{
 cat("Preparing rasters for projection\n")
 proj.stack    <- lapply(proj.file,raster)
 proj.stack    <- stack(proj.stack)
 crs(proj.stack)  <- prj
 if (env.proc != "predictors")
 {
  proj.pc     <- predict(proj.stack, pc.model[[which(result$AICc.test ==
min(result$AICc.test))[1]]], index=1:pc.nums[[which(result$AICc.test ==
min(result$AICc.test))[1]]])
  gc()
 } else {
  proj.pc     <- proj.stack
 }
 project.ras    <- predict(proj.pc, models[[which(result$AICc.test == min(result$AICc.test))[1]]])
 killStack(proj.stack)
 killStack(proj.pc)
}

cat("\nVariables contribution\n")
importance[[which(result$AICc.test == min(result$AICc.test))[1]]]
response(models[[which(result$AICc.test == min(result$AICc.test))[1]]])
```

```
result
cat("\n")
result[which(result$AICc.test == min(result$AICc.test))[1],]
write.table(result[which(result$AICc.test == min(result$AICc.test))[1],], file=paste(base_dir,
"result", ".csv", sep=""), append = TRUE, row.names = F, col.names = T, sep = ",")

crs(result.ras)  <- prj
plot(result.ras[[which(result$AICc.test == min(result$AICc.test))[1]]], col=pal(12), main="Final
map")
if(project_res == 'yes') plot(project.ras, col=pal(12), main="Projected map")

if(g > 1)
{
 sd.ras     <- calc(result.ras, sd)
 plot(sd.ras, main="SD of the maps set")
 sd_ll.ras   <- projectRaster(trim(sd.ras), crs=prj_ll)
 writeRaster(sd_ll.ras, paste0(base_dir, model.name,'_sd.tif'), format="GTiff", overwrite=T,
datatype='FLT4S', NAflag=-9999, prj=F)
 calc.niche.overlap(result.ras, "D")
}

bckg.spp     <- SpatialPointsDataFrame(coords = cbind(test.dat[,2],test.dat[,3]), data =
as.data.frame(test.dat[,4]), proj4string = prj)
suit       <- cbind(as.data.frame(extract(result.ras[[which(result$AICc.test ==
min(result$AICc.test))]], bckg.spp)), as.data.frame(test.dat[,4]))
colnames(suit)  <- c("Suit","ID")
suit       <- suit[order(suit$Suit),]

occ.dat     <- extract(result.ras[[which(result$AICc.test == min(result$AICc.test))]], csv.spp)
occ.dat     <- occ.dat[!is.na(occ.dat)]
hist(occ.dat, main="Suitability in initial dataset points", xlim=c(0,1), include.lowest = TRUE,
breaks = c(0, seq(0.067, 1.1, 0.067)))
boxplot(occ.dat, main = "Suitability in initial dataset points")
cat("\nOutliers: ")
boxplot.stats(occ.dat)$out

csv.spp     <- SpatialPointsDataFrame(csv.spp,
as.data.frame(extract(result.ras[[which(result$AICc.test == min(result$AICc.test))]], csv.spp)))
csv.spp     <- csv.spp[!is.na(csv.spp@data[,1]),]
csv.spp     <- spTransform(csv.spp, prj_ll)
distmat     <- pointDistance(csv.spp@coords, csv.spp@coords, type='Euclidean', lonlat = T,
allpairs = TRUE)
maxdist     <- 2/3000 * max(distmat)
spline.corr   <- spline.correlog(x = csv.spp@coords[,1], y = csv.spp@coords[,2], z =
csv.spp@data[,1], latlon = T, na.rm = T, xmax = maxdist, resamp = 100, type = "boot")

plot (spline.corr)
neigh      <- dnearneigh(x = csv.spp@coords, d1 = 0, d2 = 100, longlat = T)
wts       <- nb2listw(neighbours = neigh, style = 'W', zero.policy = T)
mor.norm     <- moran.test(x = csv.spp@data[,1], listw = wts, na.action = na.omit, randomisation
= F, zero.policy = T)
```

```
mor.norm
mor.norm$estimate[1]
cat("p value: ", mor.norm$p.value, "\n")
write.table(as.matrix(c(mor.norm$estimate,setNames(mor.norm$p.value, 'p-value'))),
file=paste(base_dir, "result", ".csv", sep=""), append = TRUE, row.names = T, col.names = F,
sep = ",")
write.table(suit, file=paste(base_dir, "result", ".csv", sep=""), append = TRUE, row.names = F,
col.names = T, sep = ",")

result_ll.ras <- projectRaster(trim(result.ras[[which(result$AICc.test ==
min(result$AICc.test))[1]]]), crs=prj_ll)
writeRaster(result_ll.ras, paste0(base_dir, model.name,'.tif'), format="GTiff", overwrite=T,
datatype='FLT4S', NAflag=-9999, prj=F)
if(project_res == 'yes')
{
 project_ll.ras <- projectRaster(trim(project.ras), crs=prj_ll)
 writeRaster(project_ll.ras, paste0(base_dir, model.name,'_prj.tif'), format="GTiff", overwrite=T,
datatype='FLT4S', NAflag=-9999, prj=F)
}

if (max.par > 0) cat("\nНекоторые модели с числом параметров, превышающим число точек
встреч, были удалены. Максимальное число параметров достигало ", max.par, ".\n")

time_diff    <- difftime(Sys.time(), time_start, units='mins')
if (length(mod@results[mod@results[,"auc.train"]>AUC.thr,"AICc"])==0) cat("Все модели
имели AUC ниже ", AUC.thr, ". Это было причиной прекращения анализа.\n")
cat ("\nTime elapsed:", round(time_diff,2), "minutes\n")
```