```
################################################################
# Script Name: neanderthal_sdm_evaluation.R
# Author: Marc Grünig
# Updated Documentation: 2025-08-08
#
# Description:
#   This script runs Species Distribution Model (SDM)
#   for Neanderthal habitat suitability during
#   the Last Interglacial (LIG) and other MIS stages.
#   It uses BIOMOD2 to train, tune, evaluate, and project
#   multiple models (GLM, GAM, RF, GBM, MAXNET) and
#   ensembles, then generates habitat suitability maps,
#   area estimates, and comparison plots.
#
# Inputs:
#   - LIG occurrence records with presence/absence and environmental
#     variables.
#   - Raster climate data for MIS stages (bio10, bio11,
#     bio16, bio17, slope, roughness).
#   - Neanderthals.shp : Study area extent.
#
# Outputs:
#   - Evaluation metrics for individual models.
#   - variable importance scores.
#   - Evaluation metrics for ensemble models.
#   - Variable importance for ensemble models.
#   - Suitable area plot.
#
# Notes:
#   - Requires: biomod2, terra, gbm, gam, randomForest, maps,
```

```
#     PresenceAbsence, sp, dismo, dplyr, tidyterra, caret,
#     ENMeval, data.table
#   - Set file paths before running.
##################################################################

#-----------------------------------------------------------
# Load required packages
#-----------------------------------------------------------
library(biomod2)
library(terra)
library(gbm)
library(gam)
library(randomForest)
library(maps)
library(PresenceAbsence)
require(sp)
library(dismo)
library(dplyr)
library(tidyterra)
library(caret)
library(ENMeval)
library(data.table)

#-----------------------------------------------------------
# Setup and load data
#-----------------------------------------------------------
path <- "/.../"  # Base data directory
set.seed(42)               # For reproducibility

myRespName <- "neandi_LIG"     # Response variable name
```

```
# Load LIG occurrence data
Training <- read.table(paste0(path, "/occurrence_data_LIG_highres.txt"))
colnames(Training) <- c("P", "x", "y", "Var1", "Var2", "Var3", "Var4", "Var6", "Var7", "year")
Training <- Training %>% mutate(P = as.numeric(P)) %>% data.frame()

# Format data for BIOMOD2
myBiomodData <- BIOMOD_FormatingData(
  resp.var = Training$P,
  expl.var = Training[, c("Var1", "Var2", "Var3", "Var4", "Var6", "Var7")],
  resp.name = myRespName,
  resp.xy = Training[, c("x", "y")]
)

# Cross-validation setup
cv <- bm_CrossValidation(
  myBiomodData,
  k = 5,
  nb.rep = 5,
  balance = "presences",
  do.full.models = TRUE
)

#-------------------------------------------------------------
# Modeling options and tuning
#-------------------------------------------------------------

myBiomodOption <- BIOMOD_ModelingOptions(
  GLM = list(type = "quadratic", interaction.level = 0, family = binomial(link = "logit"), test = "AIC"),
```

```
  GAM = list(algo = "GAM_mgcv", type = "s_smoother", k = 4, interaction.level = 0, family = binomial(link
= "logit")),
  RF  = list(nodesize = 5, ntree = 500),
  GBM = list(n.trees = 1000, n.minobsinnode = 5, cv.folds = 5)
)

# Model weights for class balancing
cal.w <- Training$P
cal.w[cal.w == 1] <- 1
cal.w[cal.w == 0] <- sum(Training$P) / length(which(Training$P == 0))
w <- round(cal.w, 3)

# Automatic model tuning
bm_tuning <- BIOMOD_Tuning(
  myBiomodData,
  models = c("GLM", "GAM", "RF", "GBM", "MAXNET"),
  metric.eval = "TSS"
)

#-------------------------------------------------------------
# Run models
#-------------------------------------------------------------
myBiomodModel <- BIOMOD_Modeling(
  bm.options = bm_tuning$models.options,
  myBiomodData,
  nb.rep = 2,
  nb.cpu = 8,
  models = c("GLM", "GAM", "RF", "GBM", "MAXNET"),
  weights = w,
  data.split.perc = 70,
```

```
  metric.eval = c("TSS", "ROC"),
  var.import = TRUE,
  modeling.id = myRespName
)

# Save evaluation and variable importance
tab_eval <- get_evaluations(myBiomodModel)
write.table(tab_eval, file = paste0(path, "/LIG_neanderthals/eval_sdm_LIG.csv"))

tab_varimp <- get_variables_importance(myBiomodModel)
write.table(tab_varimp, file = paste0(path, "/LIG_neanderthals/varimp_sdm_LIG.csv"))

#-------------------------------------------------------------
# Visualization of model performance
#-------------------------------------------------------------

bm_PlotEvalMean(bm.out = myBiomodModel, group.by = 'run')
bm_PlotEvalBoxplot(bm.out = myBiomodModel, group.by = c('full.name', 'algo'))
bm_PlotVarImpBoxplot(bm.out = myBiomodModel, group.by = c('expl.var', 'full.name', 'algo'))

# Response curves
mods <- get_built_models(myBiomodModel, run = 'RUN1')
bm_PlotResponseCurves(
  bm.out = myBiomodModel,
  models.chosen = mods,
  fixed.var = 'median'
)

#-------------------------------------------------------------
# Ensemble modeling
```

```
#-----------------------------------------------------------

myBiomodEM <- BIOMOD_EnsembleModeling(
  bm.mod = myBiomodModel,
  models.chosen = 'all',
  em.by = 'all',
  em.algo = c('EMmean'),
  metric.select = c('TSS'),
  metric.select.thresh = c(0.4),
  metric.eval = c('TSS', 'ROC'),
  var.import = 3,
  seed.val = 42
)

# Save ensemble evaluation and importance
tab_eval <- get_evaluations(myBiomodEM)
write.table(tab_eval, file = paste0(path, "/LIG_neanderthals/eval_sdm_LIG_ensemble.csv"))

tab_varimp <- get_variables_importance(myBiomodEM)
write.table(tab_varimp, file = paste0(path, "/LIG_neanderthals/varimp_sdm_LIG_ensemble.csv"))

#-----------------------------------------------------------
# Projections for LIG
#-----------------------------------------------------------

ext <- vect(paste0(path, "/Neanderthals.shp"))

# Prepare DEM and slope
dem <- rast(paste0(path, "/dem_30arcmin.tif")) %>% terra::mask(ext)
r.mask <- terra::aggregate(dem, fact = 10, na.rm = TRUE)
```

```
Eu.slope.0ky <- terra::terrain(dem, "slope") %>% terra::aggregate(fact = 10, na.rm = TRUE)


stages <- c("LIG")

for (k in stages) {

  Var1 <- rast(paste0(fl_path, "/climate_data/downscaled/bioclim_v1/MIS_periods/", k,
"_bio10GAM_ds.tif"))

  Var2 <- rast(paste0(fl_path, "/climate_data/downscaled/bioclim_v1/MIS_periods/", k,
"_bio11GAM_ds.tif"))

  Var3 <- rast(paste0(fl_path, "/climate_data/downscaled/bioclim_v1/MIS_periods/", k,
"_bio16GAM_ds.tif"))

  Var4 <- rast(paste0(fl_path, "/climate_data/downscaled/bioclim_v1/MIS_periods/", k,
"_bio17GAM_ds.tif"))

  Var6 <- resample(r.mask, Var1)

  Var7 <- resample(Eu.slope.0ky, Var1)


  env_stack <- c(Var1, Var2, Var3, Var4, Var6, Var7)

  names(env_stack) <- c("Var1", "Var2", "Var3", "Var4", "Var6", "Var7")


  BIOMOD_EnsembleForecasting(

    bm.em = myBiomodEM,

    new.env = env_stack,

    proj.name = k,

    nb.cpu = 8,

    metric.binary = 'TSS',

    models.chosen = 'all'

  )

}


#-------------------------------------------------------------

# Calculate suitable area

#-------------------------------------------------------------

area_df <- list()
```

```
r <- 0
for (y in stages) {
  r <- r + 1
  proj <- rast(paste0(path, "/LIG_neanderthals/neandi.LIG/proj_", y, "/proj_", y,
"_neandi.LIG_ensemble_TSSbin.tif"))
  area_df[[r]] <- sum(values(proj), na.rm = TRUE)
}
area_df <- do.call(rbind, area_df) %>%
  cbind(years = stages) %>%
  data.table() %>%
  arrange(years) %>%
  rename("suitable_area" = V1, "MIS_stage" = years)

# Plot suitable area over time
p1 <- ggplot(area_df, aes(x = as.factor(MIS_stage), y = suitable_area)) + geom_point()
ggsave(p1, paste0(path, "/results/suitable_area_over_time_MIS.png"))



### end ------------------------------------------------------------------------



#################################################################
# Script Name: neanderthal_connectivity.R
# Author: Marc Grünig
# Updated Documentation: 2025-08-08
#
# Description:
#   This script calculates connectivity between Neanderthal
#   presence points during the Last Interglacial (LIG) using
```

```
#   circuit theory (passage function in gdistance).
#   It:
#     1. Loads SDM habitat suitability projections.
#     2. Converts them into resistance surfaces.
#     3. Builds a transition matrix for movement.
#     4. Computes passage probabilities between all pairs of
#        presence points.
#     5. Aggregates and saves the average connectivity layer.
#
# Inputs:
#   - Presence/absence dataset with coordinates and environmental variables.
#   - Ensemble habitat suitability raster for target MIS stage(s).
#
# Outputs:
#   - Raster of averaged passage probabilities across all point pairs.
#
# Notes:
#   - Requires: gdistance, raster, terra, tidyverse, sp.
#   - This version is set to run for LIG only but can be
#     adapted for MIS2–MIS5 by updating `mis_stages`.
#   - Runtime can be long for many points due to all-pair
#     calculations.
##################################################################



#-----------------------------------------------------------
# Load required packages
#-----------------------------------------------------------
library(gdistance)
library(raster)
```

```
library(sp)
library(terra)
library(tidyverse)

#------------------------------------------------------------
# File path setup
#------------------------------------------------------------

# path <- "/.../"

#------------------------------------------------------------
# Load presence/absence data
#------------------------------------------------------------

Training <- read.table(paste0(path, "/occurrence_data_LIG_highres.txt"))
colnames(Training) <- c("P", "x", "y", "Var1", "Var2", "Var3",
                        "Var4", "Var6", "Var7", "MIS")

# Keep only presence records
pres <- Training[Training$P == 1, ]

#------------------------------------------------------------
# Define target MIS stage(s) for analysis
#------------------------------------------------------------

mis_stages <- c("LIG")

#------------------------------------------------------------
# Loop over each MIS stage
#------------------------------------------------------------
```

```
for (mis in mis_stages) {

  message("Processing stage: ", mis)

  # Load habitat suitability raster (ensemble model)
  proj <- rast(paste0(path, "/LIG_neanderthals/neandi.LIG/proj_",
                      mis, "/proj_", mis, "_neandi.LIG_ensemble.tif"))
  crs_proj <- crs(proj)

  #------------------------------------------------------------
  # Create resistance layer
  #------------------------------------------------------------
  # Aggregate for speed (reduce resolution)
  resist <- raster(aggregate(proj, fact = 5))

  # Replace NA with 0 (no movement possible)
  resist[is.na(resist)] <- 0

  # Build transition matrix for cost surface
  asp_tr <- transition(resist, transitionFunction = mean, directions = 4) %>%
    geoCorrection(type = "c", multpl = FALSE)

  crs(asp_tr) <- "+proj=longlat +datum=WGS84 +no_defs"

  #------------------------------------------------------------
  # Compute passage probabilities
  #------------------------------------------------------------
  reps <- 1               # Number of replicate runs
  rep_layers <- list()    # Store replicates
```

```
for (r in 1:reps) {
  message("  Replicate: ", r)

  # Use all presence points (sampling possible here)
  pres_stage_sample <- pres

  # Create all point pair combinations
  Pj_combn <- combn(nrow(pres_stage_sample), 2) %>%
    t() %>%
    as.matrix()

  # Store results for each point pair
  passages <- list()

  message("  Calculating passage probabilities...")
  system.time({
    for (i in 1:nrow(Pj_combn)) {

      # Origin and goal points
      locations <- SpatialPoints(
        rbind(
          pres_stage_sample[Pj_combn[i, 1], 2:3],
          pres_stage_sample[Pj_combn[i, 2], 2:3]
        ),
        crs(resist)
      )

      # Try to compute passage; handle errors gracefully
      passages[[i]] <- tryCatch({
```

```
    passage(
      asp_tr,
      origin = locations[1],
      goal = locations[2],
      theta = 0.001
    )
  }, error = function(e) {
    message("    Error at pair ", i, ": ", e$message)
    return(rast())  # Empty placeholder
  })

  # Progress updates every 100 iterations
  if (i %% 100 == 0) {
    message("    ", round((i / nrow(Pj_combn)) * 100, 0), "% complete")
  }

  # Clean temp files and free memory
  removeTmpFiles(h = 0.1)
  gc()
 }
})

# Keep only valid raster outputs
is_raster <- function(x) inherits(x, "Raster")
filtered_list <- Filter(is_raster, passages)

# Average passage rasters for this replicate
passages_stack <- stack(filtered_list)
passages_overlay <- sum(passages_stack) / nrow(Pj_combn)
plot(passages_overlay, main = paste("Passage overlay -", mis))
```

```
  # Save replicate result
  rep_layers[[r]] <- passages_overlay
 }

 #-------------------------------------------------------
 # Average across replicates and save final result
 #-------------------------------------------------------
 layers <- rast(stack(rep_layers))
 layers_calc <- app(layers, "mean")
 plot(layers_calc, main = paste("Mean passage probability -", mis))

 writeRaster(layers_calc,
          paste0(path, "/results/gdistance_", mis, "_all_records.tif"),
          overwrite = TRUE)
}

### end ----------------------------------------------------------------
```