nature portfolio

https://doi.org/10.1038/s41559-021-01526-9

Supplementary information

The evolutionary genomics of species' responses to climate change

In the format provided by the authors and unedited

Evolutionary genomics and species' responses to climate change

Markdown

Jonás Aguirre-Liguori[1], Santiago Ramírez-Barahona, Brandon Gaut

Incorporating local adaptation into the study of climate change has proven to enrich our prediction of species persistence under rapid climate change. However, this emerging field faces challenges to the integration of evolutionary processes (*e.g.*, local adaptation, gene flow, genetic drift, genomic load) and climate change, which would have important implications for the fate of populations. Here, we show how the theory of populations genetics has been successfully combined with climate change models to predict the response of populations to emerging climate regimes. By reviewing and integrating the methods developed in previous studies, we show concrete solutions on how to estimate the impact of evolutionary processes on species' response to changing climate across the landscape. These recent advancements point towards an integrated framework for the study of the impacts of climate change and population genomics. The present document contains an extended description of the analyses reviewed in Aguirre-Liguori et al. (2021), with a step-by-step explanation on how to perform these analyses in R. All analyses are based on published data and methods.

[1] Correspondance to jonas_aguirre@hotmail.com

Table of Contents

1. General setup

1.1. Configuration

All the input data and accessory functions necessary to run the analyses are provided in the *datasets* folder that supplements this document (available at https://github.com/spiritu-santi/Climate-Change-Genomics). Any questions related to the code and data should be addressed to the first author. The code was tested in a MacBook Pro (2.6 GHz IntelCore i7) running on 'Mojave' and R version 4.03.

Step 1. Initial configuration

```
#BiocManager::install("LEA")
#install.packages("gradientForest", repos="http://R-Forge.R-
project.org",dependendices=T)
#devtools::install_github("AndiKur4/MaizePal")
#new.packages <- list.of.packages[!(list.of.packages %in%
installed.packages()[,"Package"])]
#if(length(new.packages)) install.packages(new.packages)
library(LEA); library(adegenet); library(maps); library(dismo); library(gplots); library(raster);
library(gradientForest); library(gdistance); library(geosphere);library(MaizePal)
library(tidyverse);library(ggplot2);library(igraph);library(ggridges);library(UpSetR);library(he
re);library(rasterVis)
setwd(here::here())
# create a raster mask with values set to zero, but we the same extent and resolution as
the bioclimatic variables uses.
mask <- raster("datasets/input/present/bio_1.asc") %>% replace(.,.,0)

# source functions that will be used
  source("datasets/code/climate_change_functions.R")
# create all directories that will be used
if(!dir.exists("datasets/output")){
  dir.create("datasets/output")
}
if(!dir.exists("datasets/conservation")){
  dir.create("datasets/conservation")
}
if(!dir.exists("datasets/output/admixture")){
  dir.create("datasets/output/admixture")}

if(!dir.exists("datasets/maxent")){
  dir.create("datasets/maxent")}
```

1.2. Data

1.2.1. Genomic data

All analyses are performed on an exemplar genomic dataset for *Zea mays* spp. *mexicana*[1] (hereafter *mexicana*). This dataset consists of genomic data for 23 populations (10-12 individuals per population) of *mexicana*, covering the entire geographic and environmental distribution of the subspecies. The dataset contains data for 33,454 SNPs obtained using the SNP50K Bead Chip of Illumina. The SNPs are distributed across all ten maize chromosomes, are located within or proximate to genes, and show low levels of linkage disequilibrium (LD).

1.2.2. Climate data

Ideally, when modeling the response of species to future climate change it is fundamental to consider possible sources of uncertainty in the underlying environmental data (*i.e.*, global circulation models, green-house gas emissions, modeling algorithms, and the climate data itself). However, for simplicity, here we used a single set of climatic variables modelled for the years 2050 and 2070 under one global circulation model (Community Climate System Model, CCSM) and one greenhouse gas emission model (RCP 8.5). The present and future climatic data were obtained from the WorldClim database[2,3] at a 30 arc-sec resolution.

2. Species Distribution Models and Genetic Structure

2.1. Genetic Clustering Analysis

In this section, we use the *mexicana* genomic dataset to perform analyses of genetic clustering of populations. We use the sparse non-negative matrix factorization method (snmf) implemented in the R package LEA[4] to estimate the genetic structure among *mexicana* populations and estimate the most likely number of clusters. After identifying the most likely genetic clusters, we build species distribution models for the entire species and for each cluster separately, which were then projected into the future (see).[5,6] We assign populations to genetic cluster *i* when these contained >50% of individuals with an inferred ancestry to cluster *i*. Finally, we compared the species distribution model against the stack (sum) of the cluster distribution models.

For these analyses we need:

– 1) Genomic data for populations: “datasets/input/species_input.R”

– 2) Population geographic coordinates: “datasets/coordinates.csv”

– 3) Climatic data: “datasets/input/present/.asc”, “datasets/input/year_50/.asc”, “datasets/input/year_70/.asc”

Step 1. Run admixture analyses

```
# Load genomic dataset and obtain genotypes ($tab slot).
load("datasets/input/species_input.R")
# Since adegenet shows genotypes as pairs of alleles, we need to select columns containing the frequencies of the p alleles (SNP1.G,SNP1.C; SNP2.T; SNP3.C,SNP3.A).
# The LEA package can generate the input data from different datasets (plink, lfmm, data.frame), but it is important to remove fixed and missing SNPs. Check the LEA tutorial to understand how to run the analyses and see example for input formats.
geno <- species_input$genind$tab
geno <- geno[,seq(1,ncol(geno)-1,2)]
# Obtain the coordinates and names of populations
coords <- species_input$genind$other
pops <- rownames(coords)
# In the next sections we create datasets for two clusters, but the user should be able to set the number of clusters accordingly (see Functions).
# remove fixed SNPs and set NA to 9 (needed by snmf)
fixed_SNP <- apply(geno,2,sd,na.rm=T)
geno <- geno[,-which(fixed_SNP==0)]
geno[which(is.na(geno))] <- 9
```

```
# write geno object for admixture analysis
 write.geno(geno,output.file = "datasets/output/admixture/species.geno")
# create project to run the admixture analysis and run the analysis for a K number of
groups (for more details on the method see the vignette of the LEA package)
  project = NULL
  if(FALSE){
  project = snmf("datasets/output/admixture/species.geno",
                 K = 1:10, #set number of K
                 entropy = TRUE,
                 repetitions = 1, #set number of repetitions
                 project = "new")
# The LEA package creates a file project that contains the information of the analysis and
can be loaded for future analyses
  }

warm_col=MaizePal::maize_pal("JimmyRed",4)[2]
cold_col=MaizePal::maize_pal("MaizAzul",6)[6]
project <- load.snmfProject("datasets/output/admixture/species.snmfProject")
# the Q function generates a matrix showing the ancestry proportions of each individual.
The number of columns corresponds to K, and each column contains the proportions of an
individual to each population. The highest proportion indicates the membership to a group.
i = 2
ques <- Q(object = project,K = i)
#define individual that have a higher contribution to a given genetic group (50% split) and
get the name of each individual
g1 <- which(ques[,1]>=0.5)
g2 <- which(ques[,2]>=0.5)
g1 <- rownames(geno)[ g1]
g2 <- rownames(geno)[ g2] # if you have three clusters add a new group
# we know that “Ixtlan” grows in the North (make sure that g1 belongs to G1-N).
if(length(grep("Ixtlan", g1)) ==0){
  temp <- g1
  g1 <-  g2
  g2 <-  temp
  ques  <-  ques[,c(2,1)]
rm(temp)
}
# The next code will extract the information (lon, lat, bio_1) of each individual from the
population data and add it to the data frame order inds.
order_inds <- data.frame(rownames(geno),ques, pop=NA,lon=NA,lat=NA,bio_1=NA)
names(order_inds)[1:3] <- c("inds","g1","g2")
#coords and pops were defined at the beginning.
coords <- coords[,c("longitude","latitude","bio_1")]
# For each individual, set the name of the pop, and the longitude, latitude and bio_1,
where it grows. The next loop takes the info of each population, and searches the
individuals that belong to the population (grep), finally in the rows where those individuals
are, it adds the information
for(i in 1:length(pops)){
  tpop <- pops[i] #one pop at the time
  temp <- grep(paste(tpop,"_",sep = ""),order_inds$inds) #find all inds that belong to pop i.
  order_inds[temp,"pop"] <- rownames(coords[tpop,]) #add name of pop (temp has the
```

```
index of all individuals belonging to the population
  order_inds[temp,"lon"] <- coords[tpop,"longitude"]#add longitude
  order_inds[temp,"lat"] <- coords[tpop,"latitude"]
  order_inds[temp,"bio_1"] <- coords[tpop,"bio_1"]
}

#finally we plot the ancestry proportions based on the annual mean temperature (BIO1) at
which populations grow.
my.colors <- c(warm_col,cold_col)
bio_1 <- order_inds[order(order_inds$bio_1),]
bio_1 <- as.matrix(bio_1[,c("g1","g2")])
par(mai=c(0.5,1,0.6,1))
barplot(t(bio_1),col=my.colors,border=NA,main="Individuals (ordered by mean
temperature)",ylab="Ancestry proportions (k=2)",xlab="",xaxt="n")
```

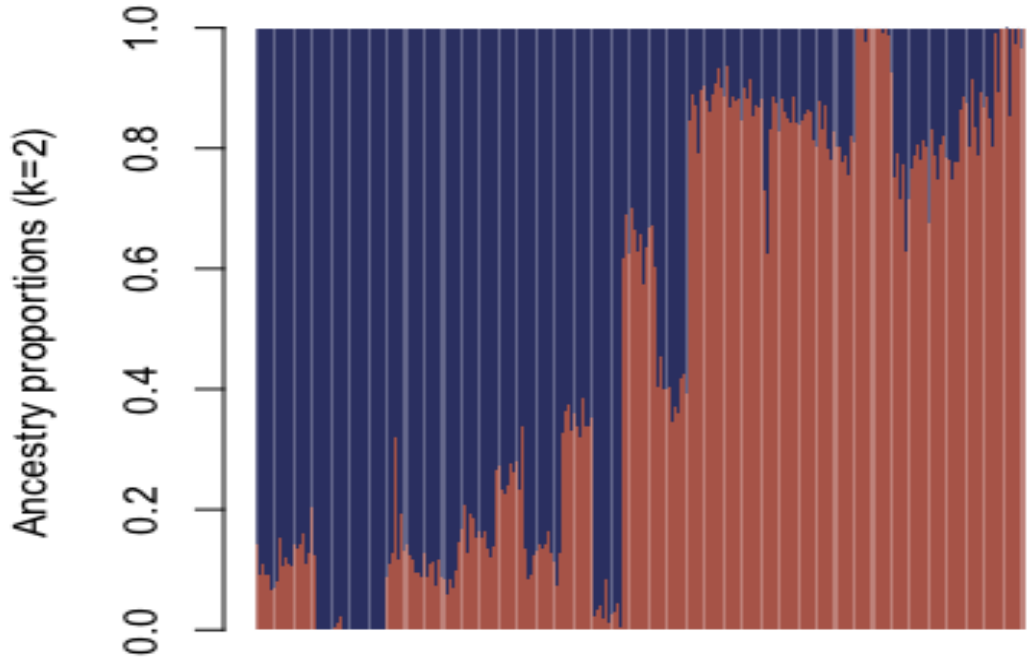


Figure S1. Ancestry proportions of mexicana individuals for the most likely genetic clusters; individuals are arranged according to the mean annual temperature where populations grow.

Step 2. Create input for Species distribution models

```
# get all individuals belonging a to genetic group (%) and select the column pop, finally,
get only the unique data
# first identify the groups with function which (if you increase the ancestry threshold above
0.5 you can remove potential admixed populations, and if you decrease the threshold you
can include admixed populations.
g1 <- order_inds[which(order_inds$g1 >= 0.5),]
g2 <- order_inds[which(order_inds$g2 >= 0.5),]
# get the names of populations
coord_g1 <- unique(g1[,"pop"])
```

```
coord_g2 <- unique(g2[,"pop"])
# repeat if you have more clusters
coord_g1 <- coords[coord_g1,c("longitude","latitude")]
coord_g2 <- coords[coord_g2,c("longitude","latitude")]
# change the names of the columns to ID, lon and lat that will be used for the input of
maxent
coord_g1 <- data.frame(ID="g1",lon= coord_g1 $longitude,lat= coord_g1$latitude)
coord_g2 <- data.frame(ID="g2",lon= coord_g2 $longitude,lat= coord_g2$latitude)
# one of the g2 populations grow within the opposite genetic group. They might be outlier
populations, so we remove them to avoid an issue with the convex hull model.
coord_g2 <- coord_g2[-which(coord_g2$lat>19.75),]
apply_buffer = FALSE # We can apply a 'buffer' around the occurrence localities; if FALSE
a convexHull is estimated.
# get all known teosintes coordinates from CONABIO and select the Mexicana group
mexicana <- read.table("datasets/input/coordinates.csv",header = T,sep = ",")
mexicana <- mexicana[grep("mexicana",mexicana$Taxa),]
names(mexicana) <- c("ID","lon","lat")
# we use convex hull models to identify populations that are putatively within the
distribution of each cluster to increase the number of populations & accuracy SDMs
# obtain the convex hull
size_buffer = 3 # Size of buffer
if(apply_buffer) ch_g1 <- buffer(SpatialPoints(coord_g1[,c("lon","lat")]),width=size_buffer)
if(!apply_buffer) ch_g1 <- convHull(coord_g1[,c("lon","lat")])
if(apply_buffer) ch_g2 <-  buffer(SpatialPoints(coord_g2[,c("lon","lat")]),size_buffer)
if(!apply_buffer) ch_g2 <- convHull(coord_g2[,c("lon","lat")])
# predict populations belonging to the models, this function searches all population from
the CONABIO dataset that grow inside the mexicana genetic groups.
if(apply_buffer) inside <- sp::over(SpatialPoints(mexicana[,2:3]),ch_g1)
if(!apply_buffer) inside <- predict(ch_g1,mexicana[,2:3])
g1 <- mexicana[which(inside==1),2:3]
g1 <- data.frame(ID="g1", g1)
#same for g2
if(apply_buffer) inside <- sp::over(SpatialPoints(mexicana[,2:3]),ch_g2)
if(!apply_buffer) inside <- predict(ch_g2,mexicana[,2:3])
g2 <- mexicana[which(inside==1),2:3]
g2 <- data.frame(ID="g2", g2)
#get coordinates for each genetic cluster (g1, g2, all)
g1 <- rbind(g1,coord_g1) #this function combines the coordinates from the conabio and
geno populations
g2 <- rbind(g2,coord_g2)
maxt <- rbind(g1,g2) # we combine all data (all the popualtions)

# create the maxent input containing each of the genetic groups and the the entire
coordinates
all <- maxt
all$ID <- "all" #we change the name from g1 and g2 to all obtain the SDM of the combined
dataset
maxt <- rbind(maxt,all) #we combine all data (g1,g2, and all)
write.table(maxt,file = "datasets/output/maxent_input.csv",row.names = F,col.names =
T,sep = ",")
```

2.2. Species Distribution Models

We implemented correlative species distribution using a set of geographic coordinates for *mexicana* retrieved from online resources[7] (CONABIO) and from sampled populations. Here, we only employed Maximum Entropy modeling (MaxEnt)[8,9] to construct SDMs, but modeling should consider possible sources of uncertainty stemming from the implementation of different modeling algorithms (*e.g.*, Maxent, GARP, MARS, GAMS). We followed best practices[10–12] to define the calibration area, fine-tuned the modeling parameters, and selected uncorrelated climatic variables. Importantly, these procedures were performed separately on each SDM. We then examined the predicted range shifts under the future climate change by projecting the resulting models onto the climate variables for the future model described above. We estimate presence/absence of populations in future models under the assumption that these would not be able to disperse into new suitable sites (*i.e.*, can only survive *in situ*); in turn, we assume that populations which are not predicted by 2050, would be absent by 2070.

For these analyses we need:

– 1) Geographic coordinates for the genetic clusters:

“datasets/output/maxent_input.csv”

– 2) Climatic data: “datasets/input/present/.asc”, “datasets/input/year_50/.asc” ,

“datasets/input/year_70/.asc” – 3) Source file with SDM functions:

“datasets/code/climate_change_functions.R”

Step 3. Construct species distribution models

```
## ADD MAXENT MODELS
#indicate that we want to allow to project the model above the ecogeographic region in the future (t_ex <- TRUE); add to the outputs the pre-fix extended; if you want to define the best model settings set vals to TRUE; define genetic group
nam <- "extended"
t_ex <- TRUE
g_group = "g1"
# get a prior extension of the area that will be used to perform the SDM;
ext_tot <- raster("datasets/input/present/bio_1.asc") %>% extent()
#the function run.maxent crops the rasters based on the ecogeographic region in which populations grow. Second, it test the correlation between variables and remove those that
```

```
are correlated >cor.val. Third if do.ENMeval is TRUE it test different models to determine
which components increase the accuracy of the model (this can take very long). Finally, it
runs the maxent alogorithm and projects the model in the present. The output can be used
to project the models into other settings, as long as the same variables are used (future,
other extensions, etc)
species <- run.maxent(path_regions =
"datasets/input/official/wwf_terr_ecos.shp",path_pres =
"datasets/input/present/",path_input = "datasets/output/maxent_input.csv",genetic_group =
g_group,cor.val = 0.8,ext =ext_tot,rdp = 10000,maxent_jar =
"datasets/input/maxent.jar",visible = FALSE,patrn = "*.asc$")
# save it for future analyses
save(species,file = paste(g_group,"/model_mxnt",g_group,".R",sep = ""))
# the projections.maxent function takes the output of the maxent model, and project the
model to other settings as long as the same variables are used. Below we project the
models to the present, year 50 and year 70. If extended= TRUE it uses the buffer around
the populations to determine the potential area of migration. If it is set to true then it only
considerst that population can migrate within their ecogeographic region.
present <- projections.maxent(path_proj = "datasets/input/present/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"present",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = FALSE)
year_50 <- projections.maxent(path_proj = "datasets/input/year_50/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"year_50",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = t_ex)
year_70 <- projections.maxent(path_proj = "datasets/input/year_70/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"year_70",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = t_ex)
#combine the outputs and save
g1 <- list(present=present,year_50=year_50,year_70=year_70)
save(g1,file = paste("g1/projections_g1_",nam,".R",sep = ""))
system("mv g1 datasets/maxent/g1")

g_group <- "g2"
species <- run.maxent(path_regions =
"datasets/input/official/wwf_terr_ecos.shp",path_pres =
"datasets/input/present/",path_input = "datasets/output/maxent_input.csv",genetic_group =
g_group,cor.val = 0.8,ext =ext_tot,rdp = 10000,maxent_jar =
"datasets/input/maxent.jar",visible = FALSE,patrn = "*.asc$")
save(species,file = paste(g_group,"/model_mxnt",g_group,".R",sep = ""))
present <- projections.maxent(path_proj = "datasets/input/present/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"present",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = FALSE)
year_50 <- projections.maxent(path_proj = "datasets/input/year_50/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
```

```
"year_50",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = t_ex)
year_70 <- projections.maxent(path_proj = "datasets/input/year_70/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"year_70",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = t_ex)
g2 <- list(present=present,year_50=year_50,year_70=year_70)
save(g2,file = paste("g2/projections_g2_",nam,".R",sep = ""))
system("mv g2 datasets/maxent/g2")

g_group <- "all"
species <- run.maxent(path_regions =
"datasets/input/official/wwf_terr_ecos.shp",path_pres =
"datasets/input/present/",path_input = "datasets/output/maxent_input.csv",genetic_group =
g_group,cor.val = 0.8,ext =ext_tot,rdp = 10000,maxent_jar =
"datasets/input/maxent.jar",visible = FALSE,patrn = "*.asc$")
save(species,file = paste(g_group,"/model_mxnt",g_group,".R",sep = ""))
present <- projections.maxent(path_proj = "datasets/input/present/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"present",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = FALSE)
year_50 <- projections.maxent(path_proj = "datasets/input/year_50/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"year_50",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = t_ex)
year_70 <- projections.maxent(path_proj = "datasets/input/year_70/",ext =
ext_tot,select_var = species$select_var,myBiomodEM = species$myBiomodEM,year =
"year_70",genetic_group = species$genetic_group,polygonsOfInterest =
species$polygonsOfInterest,coords = species$coords,  pts_b = 3,patrn =
"*.asc$",extended = t_ex)
all <- list(present=present,year_50=year_50,year_70=year_70)
save(all,file = paste("all/projections_all_",nam,".R",sep = ""))
system("mv all datasets/maxent/all")
```

Step 4. Visualize Species distribution models

```
# List of all SDMs
g2 <- create.ras(cluster = "g2")
g1 <- create.ras(cluster = "g1")
all <- create.ras(cluster = "all")

exes <- lapply(c(g2,g1,all),extent)
  x.min <- exes[[which.min(lapply(exes,"[",1))]][1]
  x.max <- exes[[which.max(lapply(exes,"[",2))]][2]
  y.min <- exes[[which.min(lapply(exes,"[",3))]][3]
  y.max <- exes[[which.max(lapply(exes,"[",4))]][4]
```

```
par(mfrow=c(3,3))
par(mai=c(0.1,0.1,0.4,0.1))
for (i in 1:length(g1)){
plot(g1[[i]],col=warm_col,legend=F,xlim=c(-106,-96),ylim=c(16,24),xaxt="n",yaxt="n")
maps::map("world","Mexico",add=T)
title(names(g1)[i])
}
for (i in 1:length(g2)){
plot(g2[[i]],col=cold_col,legend=F,xlim=c(-106,-96),ylim=c(16,24),xaxt="n",yaxt="n")
maps::map("world","Mexico",add=T)
}
for (i in 1:length(all)){
plot(all[[i]],col="grey50",legend=F,xlim=c(-106,-96),ylim=c(16,24),xaxt="n",yaxt="n")
maps::map("world","Mexico",add=T)
}
```

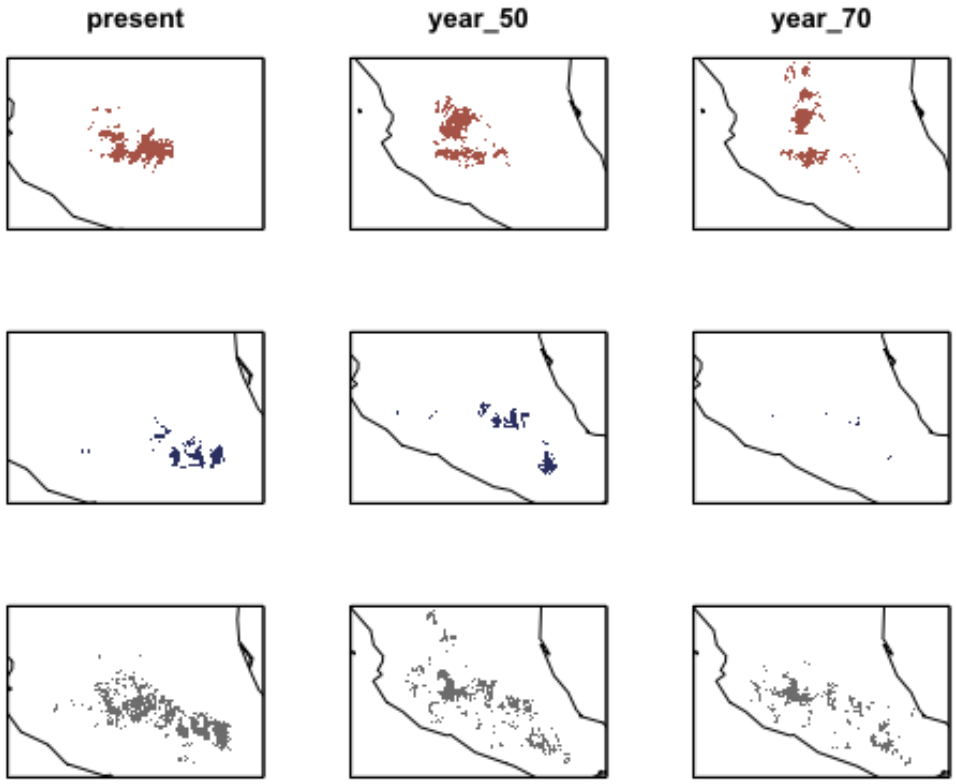


Figure S2. Present-day and future species distribution models for the G1-N (red) and G2-S (blue) clusters of mexicana, and for the whole species (grey). Once we have the present and future models, we estimate their geographic overlap to assess range stability (gains and losses).

```
# obtain the raster values for a time period
present <- lapply(list(g2,g1),"[[",1)
names(present) <- c("g2","g1")
present[[1]]@data@names <- "g2"
```

```
present[[2]]@data@names <- "g1"

ext <- extent(-106,-96,16,24)
present <- lapply(present,raster::extend,ext) %>% stack()

#arbitrarely set 2 values, one for each model so the layers can be identified, 1: g1; 2: g2, 3: overlap
present$g1[present$g1!=0] <- 1
present$g2[present$g2!=0] <- 2
present_join <- calc(present,sum,na.rm=T)
present_join[present_join<1] <- NA
# create vector with the present values. It is the sum of each model + the overlap.
tot_pres <- present_join
# get areas of the models
area <- present_join@data@values
area <- area [which(area >0)]
area <- factor(area,levels=c(1,2,3))
area <- summary(area)
pres_area <- c(G1=sum(area[c(1,3)]),G2=sum(area [c(2,3)]))
# Compare between all the model and the sum of the g1+g2 models. Set the g1+g2 to 1 and full to 2; 3 = overlap.
present_join[present_join != 0] <- 1
ext <- extent(-106,-96,16,24)
all_present <- raster::extend(all$present,ext)
all_present[all_present!=0] <- 2
full_join <- stack(present_join,all_present)
full_join <- calc(full_join,sum,na.rm=T)
full_join[full_join<1] <- NA
tot_pres <- full_join
area <- full_join@data@values
area <- area [which(area > 0)]
area <- factor(area,levels=c(1,2,3))
area <- summary(area)
pres_area <- c(pres_area,All=sum(area[2:3]))
over_pres <- area

# Do the same for 2050
fut_2050 <- lapply(list(g2,g1),"[[",2)
names(fut_2050) <- c("g2","g1")
fut_2050[[1]]@data@names <- "g2"
fut_2050[[2]]@data@names <- "g1"
ext <- extent(-106,-96,16,24)
fut_2050 <- lapply(fut_2050,raster::extend,ext) %>% stack()
fut_2050$g1[fut_2050$g1!=0] <- 1
fut_2050$g2[fut_2050$g2!=0] <- 2
future_join <- calc(fut_2050,sum,na.rm=T)
future_join[future_join<1] <- NA
tot_fut <- future_join
area <- future_join@data@values
area <- area [which(area >0)]
area <- factor(area,levels=c(1,2,3))
```

```
area <- summary(area)
future50_area <- c(G1=sum(area[c(1,3)]),G2=sum(area [c(2,3)]))

future_join[future_join != 0] <- 1
ext <- extent(-106,-96,16,24)
all_future <- raster::extend(all$year_50,ext)
all_future[all_future!=0] <- 2
full_join <- stack(future_join,all_future)
full_join <- calc(full_join,sum,na.rm=T)
full_join[full_join<1] <- NA
tot_50 <- full_join
area <- full_join@data@values
area <- area [which(area > 0)]
area <- factor(area,levels=c(1,2,3))
area <- summary(area)
future50_area <- c(future50_area,All=sum(area[2:3]))
over_50 <-  area

# Do the same for 2070
fut_2070 <- lapply(list(g2,g1),"[[",3)
names(fut_2070) <- c("g2","g1")
fut_2070[[1]]@data@names <- "g2"
fut_2070[[2]]@data@names <- "g1"
ext <- extent(-106,-96,16,24)
fut_2070 <- lapply(fut_2070,raster::extend,ext) %>% stack()
fut_2070$g1[fut_2070$g1!=0] <- 1
fut_2070$g2[fut_2070$g2!=0] <- 2
future_join <- calc(fut_2070,sum,na.rm=T)
future_join[future_join<1] <- NA
area <- future_join@data@values
area <- area [which(area >0)]
area <- factor(area,levels=c(1,2,3))
area <- summary(area)

future70_area <- c(G1=sum(area[c(1,3)]),G2=sum(area [c(2,3)]))
future_join[future_join != 0] <- 1
ext <- extent(-106,-96,16,24)
all_future <- raster::extend(all$year_70,ext)
all_future[all_future!=0] <- 2
full_join <- stack(future_join,all_future)
full_join <- calc(full_join,sum,na.rm=T)
full_join[full_join<1] <- NA
tot_70 <- full_join
area <- full_join@data@values
area <- area [which(area > 0)]
area <- factor(area,levels=c(1,2,3))
area <- summary(area)
future70_area <- c(future70_area,All=sum(area[2:3]))
over_70 <-  area
```

```
par(mfrow=c(1,3))
#plot area of the models
ylims <- c(0,signif(range(c(over_50,over_70,over_pres))[2],1))
barplot(as.vector(over_pres),main="Present-day",ylim=ylims,names=c("G1+G2\n
only","Full\n only","Overlap"),cex.names=0.8,
las=1,col=c(maize_pal("HopiBlue")[c(1,4,6)]),ylab="Area (number of pixels)")
barplot(as.vector(over_50),main="2050",ylim=ylims,names=c("G1+G2\n only","Full\n
only","Overlap"),cex.names=0.8,yaxt="n",las=1,col=c(maize_pal("HopiBlue")[c(1,4,6)]),ylab
="")
barplot(as.vector(over_70),main="2070",ylim=ylims,names=c("G1+G2\n only","Full\n
only","Overlap"),cex.names=0.8,las=1,yaxt="n",col=c(maize_pal("HopiBlue")[c(1,4,6)]),ylab
="")
```

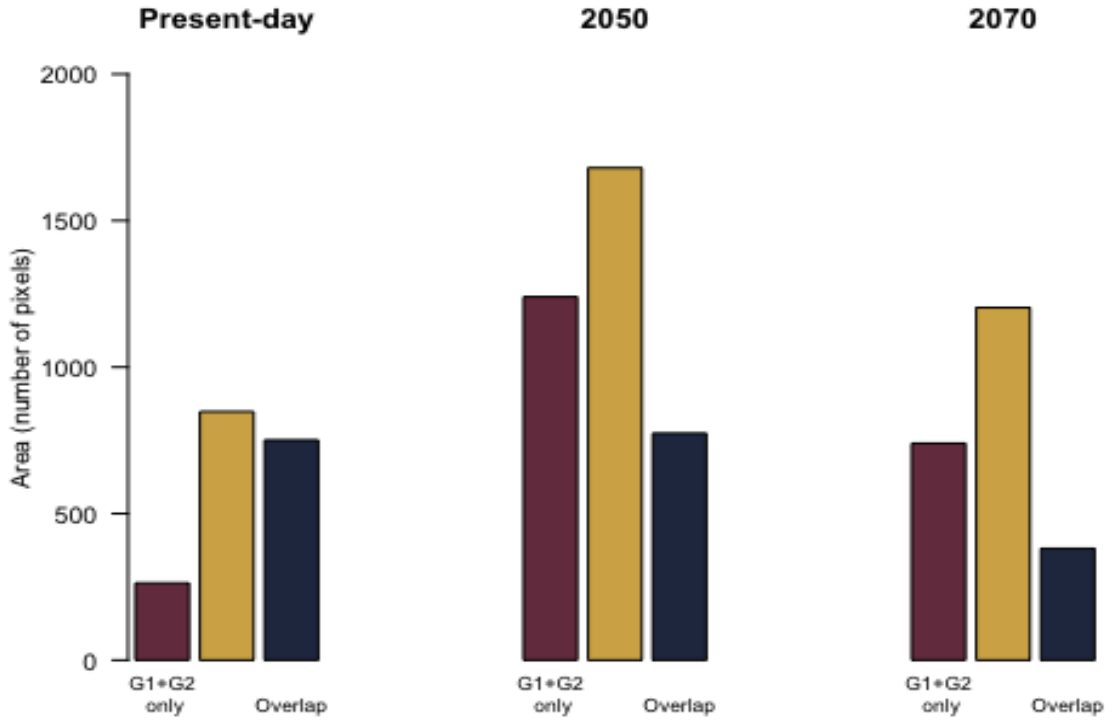


Figure S4. Future species distribution models (2050,2070) for the two genetic clusters of mexicana (left panel). Compared with the present-day models, we observe fewer areas of overlap between the full model (dark blue) and the cluster models.

Step 5. Predicting the fate of populations

```
# create a table that contains the geographic information of the fate of populations
coords <- species_input$genpop$other[,c("longitude","latitude")] # we create the coords
object again because we modified in temporal objects above
# extract the value of populations of all models to see if they will exist or not. (0 means no)
coords$vals_pres <- raster::extract(tot_pres,coords[,c("longitude","latitude")])
coords$vals_50 <- raster::extract(tot_50,coords[,c("longitude","latitude")])
coords$vals_70 <- raster::extract(tot_70,coords[,c("longitude","latitude")])
```

```
# set the environmental categories in which populations grow (depending on the genetic clusters)
coords$cluster <- NA
coords$cluster[coords$vals_pres==5]<-"cold"
coords$cluster[coords$vals_pres==3]<-"warm"
# All populations exist in the present so they have value 1
coords$vals_pres <- 1
# in year 2050 and 2070 those that become 0 will be considered extinguished, so values >1 are assumed to exist
coords$vals_50[coords$vals_50>0]<-1
coords$vals_70[coords$vals_70>0]<-1
# finally populations predicted in 2070 that were extinct in 2050 are also set to 0
coords$vals_70[coords$vals_50==0]<-0

# write the table in the conservation file for future analysis
write.csv(coords,file = "datasets/conservation/vul_sdm.csv")
```

3. Local Adaptation and Climate Change

3.1. Allele turnover across the landscape

In this section, we follow the procedures originally described in ref.[13] to employ the machine learning algorithm Gradient Forest to: 1) identify environmental variables with a strong impact on the frequency of adaptive alleles; 2) generate allele turnover functions for individual loci; and 3) estimate the genomic vulnerability (here after termed genomic offset) of populations under future climate change. Basically, these models predict how the genomic diversity of a species is distributed across the landscape, by estimating how allele frequencies of SNPs change along environmental gradients (*i.e.*, genetic turnover models)[1,14–21].

For these analyses we need:

– 1) Genomic data: “datasets/input/gradient_forest_input.csv”

– 2) Climatic data: “datasets/input/present/.asc”, “datasets/input/year_50/.asc”, “datasets/input/year_70/.asc”

Step 6. Running Gradient Forest

```
# read the input, see above for details
gfData <- read.table("datasets/input/gradient_forest_input.csv",header =
T,sep="\t",row.names = "pop")
# first step separate data based on the category of SNPS. In the function described in
section 9, we do it for only 1 set of SNP at the time.
candidate <- gfData[,grep("candSNPs",names(gfData))]
reference <- gfData [,grep("refSNPs",names(gfData))]
# create a table with the bioclimatic information of populations
present <- gfData[,c(1,2,grep("bio",names(gfData)))]
# define the bioclimatic variables of interest
bioclimatic <- paste("bio_",1:19,sep = "")
# set the importance of the permutation distribution of each variable. Type
help(gradientForest for more details)
maxLevel <- log2(0.368*nrow(candidate)/2)
# run the algorithm (gradient forest function) for each set of SNPs
if(FALSE){ # FALSE if there is no need to run the analyses
gf_candidate <- gradientForest(cbind(present[,bioclimatic], candidate),
predictor.vars=colnames(present[,bioclimatic]),
                        response.vars=colnames(candidate), ntree=500,
                        maxLevel=maxLevel, trace=T, corr.threshold=0.50)
gf_reference <- gradientForest(cbind(present[,bioclimatic], reference),
predictor.vars=colnames(present[,bioclimatic]),
                        response.vars=colnames(reference), ntree=500,
```

```
                    maxLevel=maxLevel, trace=T, corr.threshold=0.50)
# combine the GF models into a list and save it
  gf_runs <- list(gf_reference=gf_reference,
                  gf_candidate=gf_candidate)
  if(!dir.exists("datasets/output")){
    dir.create("datasets/output")
  }
  save(gf_runs,file = "datasets/output/gf_runs.R")
}
# load the GF models
load("datasets/output/gf_runs.R")
gf_candidate <- gf_runs$gf_candidate
gf_reference <- gf_runs$gf_reference
 # Once the gradient forest model has been constructed, the importance of each variables
(variable contribution) to the model can be estimated.
# The following vector contains such information
bio_cand <- gf_candidate$overall.imp[order(gf_candidate$overall.imp,decreasing = T)]
most_cand <- names(bio_cand[1])

barplot(bio_cand,las=2,cex.names=0.8,col=c(MaizePal::maize_pal("JimmyRed",4),rep("gr
ey",15)),ylab="Weigthed importance (R-sqr)")
```

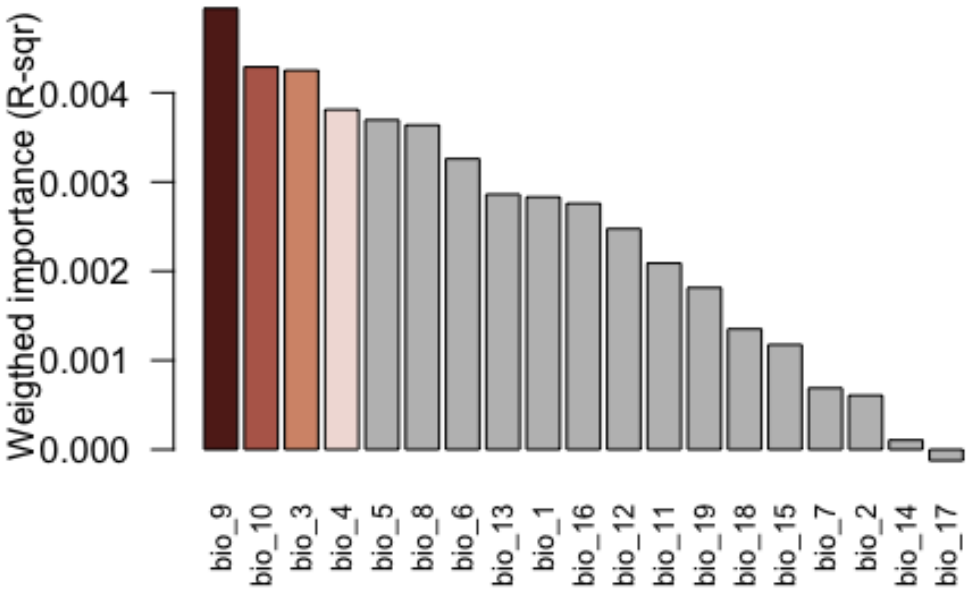


Figure S5. According to the genetic turnover model, the mean temperature of the driest quarter (bio_9) is the variable with the highest contribution. In red are the four variables with the highest contribution to the turnover model.

Step 7. Allele turnover functions across the landscape.

```
# We can extract the allele turnover as a function of a single predictor variable (in this case
bio_9). This can be done for the combined SNPs (Overall option) or individual SNPs
(Species option). Note: here we show allele turnover across the range of individual
```

```
variables, but we provide options to explore allele turnover across all variables.
temp_cand_overall <- cumimp(gf_candidate,predictor= most_cand,
type=c("Overall"),standardize = T) # al candidate SNPs
temp_cand_SNP <- cumimp(gf_candidate,predictor = most_cand,
type=c("Species"),standardize = T) #each individual candidate allele
temp_ref_overall <- cumimp(gf_reference,predictor = most_cand,
type=c("Overall"),standardize = T) #all neutral SNPs
temp_ref_SNP <- cumimp(gf_reference,predictor = most_cand,
type=c("Species"),standardize = T) #each individidual neutral SNPs
# the next code is run only to estimate the y axis limit so the final plot incorporates all data.
This is just esthetics and to automatize the code
ylim <- NULL
for(j in 1:length(temp_cand_SNP)){ #test each SNP
  ylim <- c(ylim,max(temp_cand_SNP[[j]][[2]])) # get the maximum value for a SNP
}
#same for reference SNPs, since we will plot both types of SNPs, we add the new
maximum values to those that were obtained for the candidate SNPs (we do not create a
new ylim object)
for(j in 1:length(temp_ref_SNP)){
  ylim <- c(ylim,max(temp_ref_SNP[[j]][[2]]))
}
ylim <- max(ylim)

# code to plot the overall and individual allele turnover functions across the candidate bio
par(mfrow=c(1,2))
par(mai=c(0.9,0.8,0.4,0))
plot(temp_cand_overall,type="n",ylim=c(0,ylim),mgp=c(2,0.6,0),
     ylab="Cumulative importance",xlab= "bio_9")
for(j in 1:length(temp_cand_SNP)){
  lines(temp_cand_SNP[[j]],col=adjustcolor(MaizePal::maize_pal("RubyGold")[5],alpha.f =
0.6))
}
lines(temp_cand_overall,col=MaizePal::maize_pal("RubyGold")[2],lwd=4)
par(mai=c(0.9,0.1,0.4,0.6),tcl=-0.2)
plot(temp_ref_overall,type="n",ylim=c(0,ylim),mgp=c(2,0.6,0),ylab="",xlab= "bio_9",
yaxt="n")
for(j in 1:length(temp_ref_SNP)){
  lines(temp_ref_SNP[[j]],col=adjustcolor(MaizePal::maize_pal("MaizAzul")[3],alpha.f =
0.6))
}
lines(temp_ref_overall,col=MaizePal::maize_pal("MaizAzul")[6],lwd=4)
```

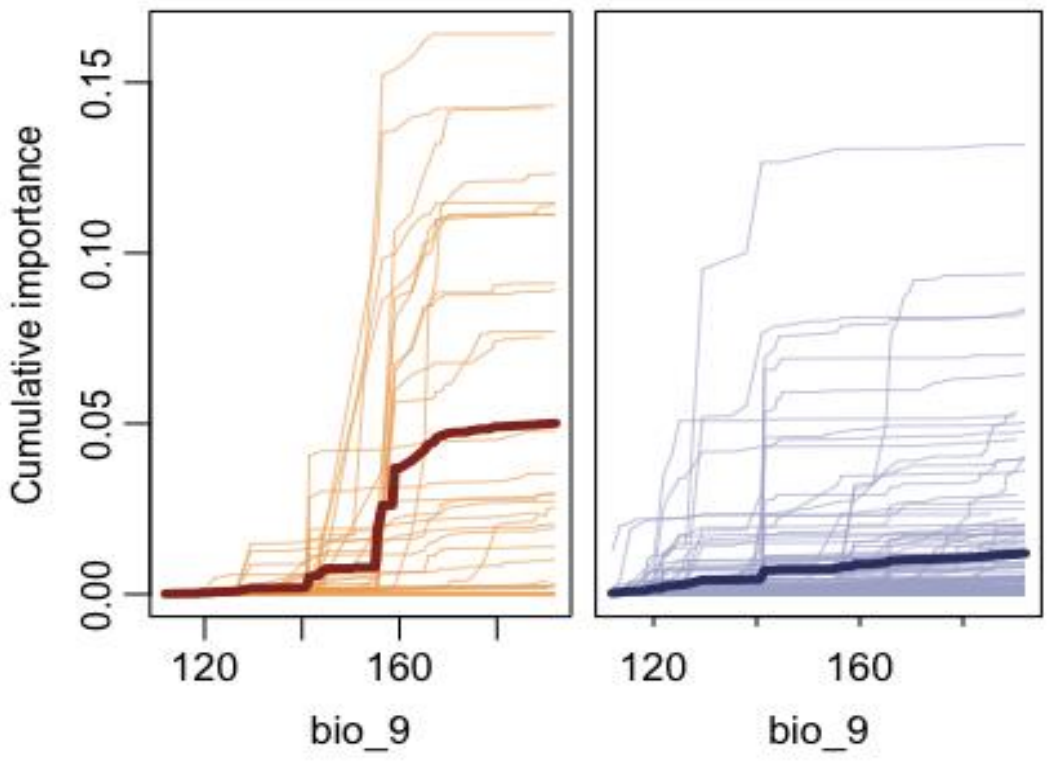


Figure S6. Allele turnover functions depicting the overall trend (bold lines) and individual trends of frequencies changes along a gradient for mean temperature of the driest quarter (bio_9): candidate SNPs (red line, left panel), reference SNPs (blue line, right panel). Light lines depict the trend for individual SNPs: candidate SNPs (light orange), reference SNPs (light blue).

Step 8. Define populations locally adapted to contrasting environments

```
# Populations can be classified based on environmental categories defined from the allele turnover functions across a given environmental gradient; here we show the patterns across bio_9.
pop_turn <- predict(gf_candidate,present[,grep("bio",names(present))])
temp <- data.frame(bio=present[,most_cand],imp=pop_turn[,most_cand]) # get the x (bio value) and y (predicted cumulative importance) values of each population
warm <- which(pop_turn[,most_cand] >= (mean(pop_turn[,most_cand]))) # identify which populations grow above the mean
cold <- which(pop_turn[,most_cand] < (mean(pop_turn[,most_cand]))) # identify which populations grow above the mean
# record the categories of populations (they are adapted to cold or warm conditions) for future analyses.
categories <- list(cold=rownames(pop_turn)[cold],warm=rownames(pop_turn)[warm]) # create a list containing the name of populations that belong to the environmental clusters. This list will be used in later analyses to classify populations.

plot(temp_cand_overall,type="n",ylim=c(0,ylim),mgp=c(2,0.6,0),
    ylab="Cumulative importance",xlab= paste("Most important variable
```

```
(",most_cand,")",sep=""),main="Candidate SNPs")
#for each individual SNP add the line, orange and lightblue indicate adaptive and
reference SNPs
for(j in 1:length(temp_cand_SNP)){
  lines(temp_cand_SNP[[j]],col=adjustcolor(MaizePal::maize_pal("RubyGold")[5],alpha.f =
0.6))
}
lines(temp_cand_overall,col=MaizePal::maize_pal("RubyGold")[2],lwd=4)
# this time we add the points of the populations depending on whether they grow in the
cold or warm adapted cluster. We plot with different colors
warm_col=MaizePal::maize_pal("JimmyRed",4)
cold_col=MaizePal::maize_pal("MaizAzul",4)
id_c <- order(temp$bio[cold])
id_ccol <- as.character(cut(1:length(id_c),length(cold_col),labels=cold_col))
id_w <- order(temp$bio[warm])
id_wcol <- as.character(cut(1:length(id_w),length(warm_col),labels=warm_col))
points(temp$bio[warm][id_w],temp$imp[warm][id_w],pch=21,bg=rev(id_wcol),cex=1.5)
points(temp$bio[cold][id_c],temp$imp[cold][id_c],pch=21,bg=id_ccol,cex=1.5)
```

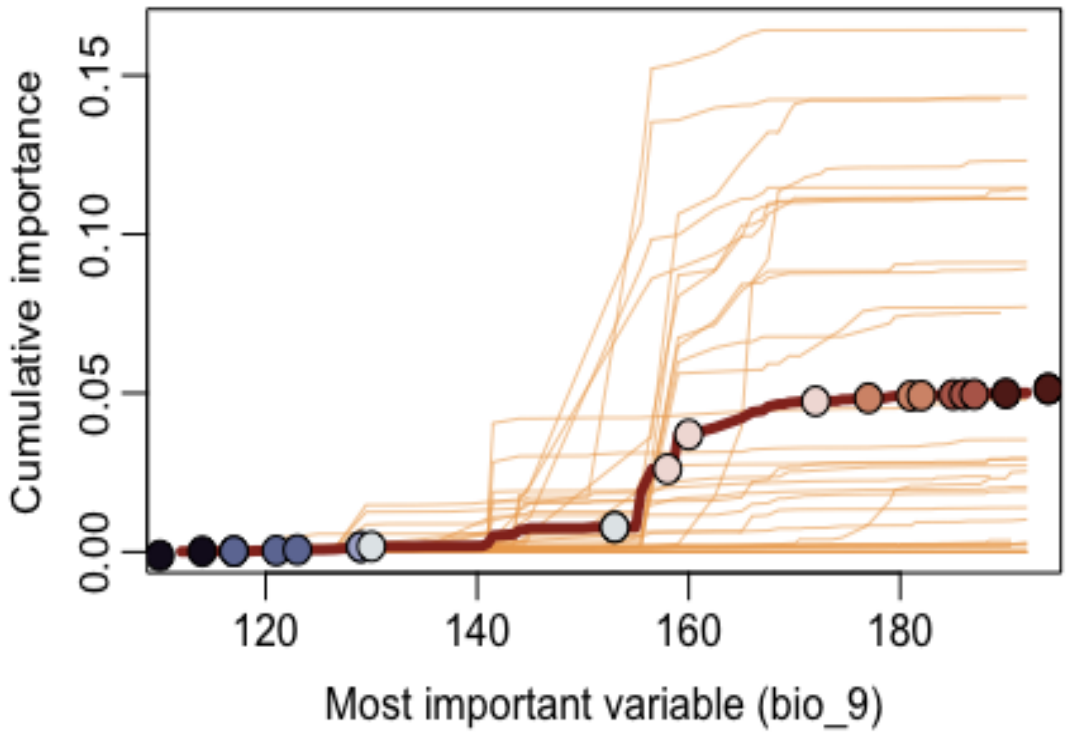


Figure S7. The overall allelic turnover function for candidate SNPs projected along bio_9 shows an 'S' pattern: a) stable frequencies below ~ 15ºC; b) a steep turnover between ~15 and 17°C; and c) stable frequencies above ~17ºC.

Step 9. Estimate Genomic Offset

```
# We create matrices of present and future climate data that will be used to extrapolate the
functions constructed with the Gradient Forest analysis across the geographic landscape.
```

```
Briefly, the function converts any given number of enviromental raster layers into a data frame.
turn_score <- data.frame(gfData[,c("X","Y",most_cand)],temp)
present_mat <- convert_env_trns(path = "datasets/input/present/")
future_mat_50 <- convert_env_trns(path = "datasets/input/year_50/") # you can change the layers to have other periods. Here it is 2050-RCP4.5. We used this so the change is not so drastic, and we can test the code.
future_mat_70 <- convert_env_trns(path = "datasets/input/year_70/") #Here it is 2070-RCP8.5, that will be harsher.
#predict allelic data across the landscape
pred_paSNPs <- predict(gf_candidate,present_mat[grep("bio",names(present_mat))])
pred_paSNPs_future_50 <-
predict(gf_candidate,future_mat_50[grep("bio",names(future_mat_50))])
pred_paSNPs_future_70 <-
predict(gf_candidate,future_mat_70[grep("bio",names(future_mat_70))])
# finally we estimate the Euclidian distance between the two matrices, this is the genetic offset; the euclidian_distance function is also an accessory function
euclidian_50 <-
euclidian_distance(proj_fut=pred_paSNPs_future_50,pred_pres=pred_paSNPs)
euclidian_70 <-
euclidian_distance(proj_fut=pred_paSNPs_future_70,pred_pres=pred_paSNPs)
# create a raster layer that contains the genetic offset of each pixel. We add this information in the mask raster created at the beginning
offset_ras_50 <- mask
offset_ras_50[present_mat$cell]<- euclidian_50  #the present_mat$cell contains the cell of each pixel in the distribution of the species
offset_ras_70 <- mask
offset_ras_70[present_mat$cell]<- euclidian_70
#obtain the genetic offset of the know populations
genetic_off_50 <- raster::extract(offset_ras_50,present[,1:2])
genetic_off_70 <- raster::extract(offset_ras_70,present[,1:2])
#create the tables that contain the coordinates of populations and the genetic offset
pop_vul_50 <- data.frame(present[,1:2], genetic_off_50)
pop_vul_70 <- data.frame(present[,1:2], genetic_off_70)
# Add the category of the genetic offset to a table for future analyses.
pop_vul_50$temp <- NA
pop_vul_50$temp[warm]<-"warm"
pop_vul_50$temp[cold]<-"cold"
pop_vul_50$temp <- factor(pop_vul_50$temp,levels = c("cold","warm"))
pop_vul_70$temp <- NA
pop_vul_70$temp[warm]<-"warm"
pop_vul_70$temp[cold]<-"cold"
pop_vul_70$temp <- factor(pop_vul_70$temp,levels = c("cold","warm"))
# get max offset
max_val <- max(c(pop_vul_50$genetic_off_50,pop_vul_70$genetic_off_70))
# create a table that the genetic offset of each cell in the map
vulnerable_areas <- as.data.frame(offset_ras_50)
vulnerable_areas <- data.frame(cell=1:nrow(vulnerable_areas),vul=vulnerable_areas[,1])
#extract only the cells and values of areas where there is a predicted vulnerability
vulnerable_areas_50 <- vulnerable_areas[which(vulnerable_areas$vul>0),]
# same for 2070
```

```
vulnerable_areas <- as.data.frame(offset_ras_70)
vulnerable_areas <- data.frame(cell=1:nrow(vulnerable_areas),vul=vulnerable_areas[,1])
vulnerable_areas_70 <- vulnerable_areas[which(vulnerable_areas$vul>0),]
# create object that contains the different outputs of GF, these will be used for other
analyses in the next sections
GO_objects <- list(pred_paSNPs=pred_paSNPs, #predicted genetic space in the present,
and the two future (next two)
                   pred_paSNPs_future_50=pred_paSNPs_future_50,
                   pred_paSNPs_future_70=pred_paSNPs_future_70,
                   present_mat=present_mat, #environmental values of all the cells in the
present
                   vulnerable_areas_50=vulnerable_areas_50, #cell that have a predicted
genetic offset in the future (same for the next line)
                   vulnerable_areas_70=vulnerable_areas_70,
                   genetic_off_50=genetic_off_50, #genetic offset of populations in the present
and the future
                   genetic_off_70=genetic_off_70,
                   pop_vul_50=pop_vul_50, #table containing containing coordinates and genetic
offset
                   pop_vul_70=pop_vul_70,
                   present=present, #table containing bioclim data of popualtions
                   temp_cand_overall=temp_cand_overall, # turnover functions
                   gf_candidate=gf_candidate, # gradient forest model
                   offset_ras_50=offset_ras_50, #raster with the offset
                   offset_ras_70=offset_ras_70,
                   gfData=gfData, #input data
                   bio_cand=most_cand, #bio climatic variable with the strongest contribution
                   categories=categories,
                   turn_score=turn_score) #categories of populations

# save the outputs from GF for futher analyses
save(GO_objects,file = "datasets/output/GO_objects.R")
#finally write the information to the conservation file
vul_gen_off <-
cbind(pop_vul_50[c("X","Y","genetic_off_50")],pop_vul_70[c("genetic_off_70")])
# create dir if it does not exist
write.csv(vul_gen_off,file = "datasets/conservation/vul_fg.csv",row.names = T) # create a
data frame indicating all areas that have a genetic offset abvove the threshold

gen_off_stack <- stack(offset_ras_50,offset_ras_70)
names(gen_off_stack) <- paste(c("Year_2050","Year_2070"))
#plot genetic offset and the known populations according to the environmental cluster in
which they grow
rasterVis::levelplot(gen_off_stack,margin=FALSE, colorkey=list(space="bottom"),
xlab=NULL, ylab=NULL, scales=list(draw=FALSE),main = "Genomic
offset",par.settings=rasterVis::rasterTheme(rev(maize_pal("MaizMorado",type="continuous
"))))
```

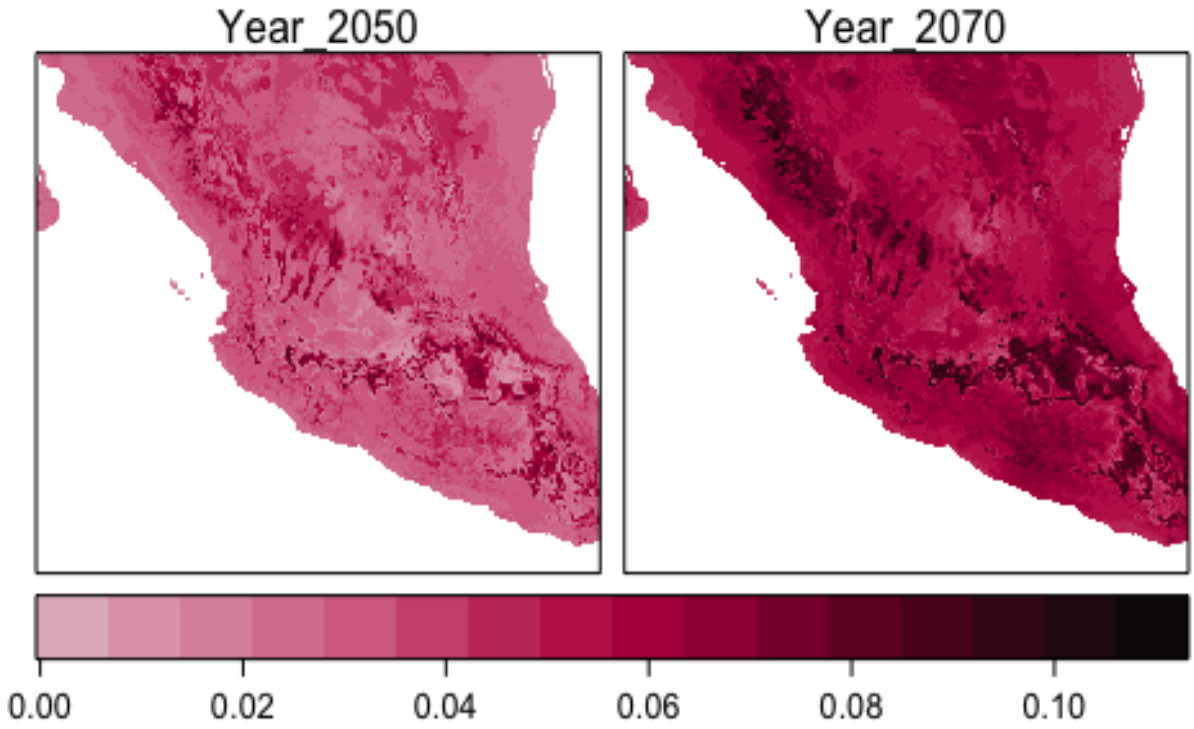


Figure S8. Genomic offset across the landscape for two different time periods: 2050 and 2070. Genomic offset is a measure of vulnerability to climate change and is proportional to the amount of change in allele frequencies needed to maintain the current gene-environment relationships.

```
#plot the distribution of genetic offset according to the environment in which populations grow and estimate if they are significantly different
warm_col=MaizePal::maize_pal("JimmyRed",4)[2]
cold_col=MaizePal::maize_pal("MaizAzul",6)[2]
par(mfrow=c(1,2))
par(mai=c(0.6,0.8,0.6,0))
plot(pop_vul_50$genetic_off_50 ~ pop_vul_50$temp, ylab="Genetic offset", xlab=
most_cand, col=c(cold_col,warm_col),main="2050",ylim=c(0,max_val))
# now, let's plot according to the North/South distribution. Again, here the order is increasing and positive, so the order is N->S
par(mai=c(0.6,0.1,0.6,0.6),tcl=-0.2)
#test the significance
fit <- aov(pop_vul_50$genetic_off ~pop_vul_50$temp)
p <- summary(fit)
#add legend
plot(pop_vul_70$genetic_off ~ pop_vul_70$temp, ylab="Genetic offset", xlab=
bio_cand,col=c(cold_col,warm_col),main="2070",ylim=c(0,max_val),yaxt="n")

fit <- aov(pop_vul_70$genetic_off ~ pop_vul_70$temp)
p2 <- summary(fit)
```

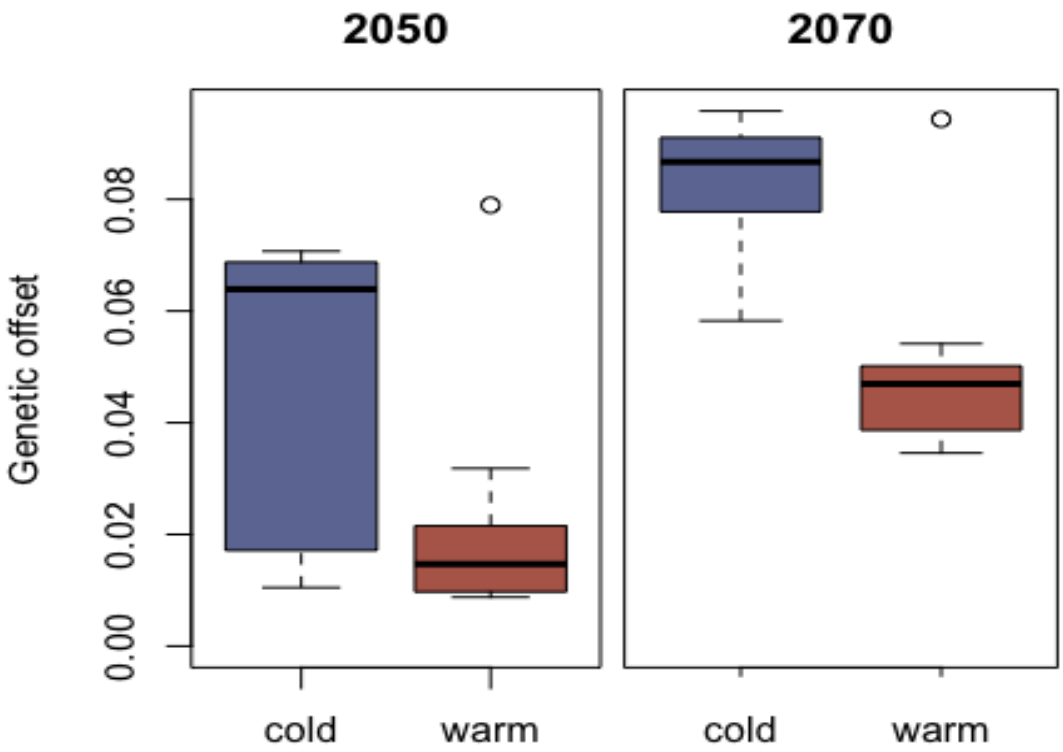


Figure S9. Genomic offset of populations estimated for two different time periods: 2050 and 2070. Here, we obtained the genomic offset values for sample population grouped according to environmental conditions (i.e., mean temperature of the driest quarter).

4. Adaptive and mal-adaptive gene flow

4.1. Estimating gene flow between populations

In this section we use *Fastsimcoal2*[22,23] to estimate effective population sizes (*Ne*) and gene flow (*m*) between pairs of populations. *Fastsimcoal2* is a powerful tool that can be used to model complex evolutionary scenarios, by estimating population parameters that maximize the composite Likelihood based on the observed Site Frequency Spectrum (SFS). For each pair of populations we produce a file with the two-dimensional folded SFS and proceed to run the evolutionary models in *Fastsimcoal2* (see the *Fastsimcoal2* manual for details on naming input files). We build simple models of population divergence with gene flow, using the priors from ref.[1]. The next two chunks of code (Steps 10, 11) need command lines from a UNIX based environment, and *Fastsimcoal2* needs to be compiled prior to performing the analyses (see Manual).

For these analyses we need:

– 1) Genomic offset output (Step 9): ""“datasets/output/GO_objects.R”

– 2) Genomic data: “datasets/input/species_input.R”

– 3) Input for the coalescent simulations: “datasets/input/input.par”, “datasets/input/input.tpl”

– 4) Functions to manipulate SFS: “functions_SFS.R”

Step 10. Format the SFS

```
# We use functions from Liu S, et al. (2018) Molecular Ecology, 27: 4725-4743 to run SFS
# load genomic dataset and obtain genotypes
source("datasets/code/functions_SFS.R")
# load genomic dataset and obtain genotypes
load("datasets/input/species_input.R")
load("datasets/output/GO_objects.R")
categories <- GO_objects$categories
# first we need to generate the genotype data, using the Minor allele frequencies (q allele)
# obtain genotypes and coordinates
geno <- species_input$genind$tab
coords <- species_input$genind$other[,c("longitude","latitude")]
# set categories and colors
coords$temperature <- NA # add category
coords$cols <- NA # add category
coords[categories$cold,c("temperature","cols")] <- data.frame("cold",cold_col)
```

```
coords[categories$warm,c("temperature","cols")] <- data.frame("warm",warm_col)

#select the q allele. Since adegenet shows genotypes in pairs of alleles, we need to select the odd columns with the function (seq), you need to check that you have two alleles per SNP, if not you would need to extract only the q alleles. See more detailed explanation in section 2-admixture.
geno <- geno[,seq(2,ncol(geno),2)]
#create a vector indicating to which populations individuals belong to
pop <- as.character(species_input$genind$pop)
geno <- t(geno)
#create mygt input, it is a list with a vector containing populations and another one containing the q alleles
mygt <- list(popmap=pop,genotype=geno)
#get names of populations
pop <- unique(pop)
#obtain all the populations pairs
cbm <- combn(1:length(pop),m = 2)
#create a directory that will have all the inputs, one per pair of populations
if(FALSE){ #if this has been done, set to FALSE so it does not do it every time
  if(!exists("datasets/output/SFS_inputs")) dir.create("datasets/output/SFS_inputs")
  #the next loop takes each pair of population, and obtains the genotype of each pair of populations, plots the SFS, the two dimensional SFS and writes the SFS in the fastsimcoal input
  for(i in 1:ncol(cbm)){
    cat(i,"\r")
    #select the two pairs of populations
    pop1 <- pop[cbm[1,i]]
    pop2 <- pop[cbm[2,i]]
    pop_c <- unlist(lapply(strsplit(pop,"_"),"[",1))
    pop1_lab <- pop_c[cbm[1,i]]
    pop2_lab <- pop_c[cbm[2,i]]
    #create the name of the pairs of populations
    path <- paste(pop1,"_",pop2,sep = "")
    #create a dir where the input will be written
    dir.create(paste("datasets/output/SFS_inputs/",path,sep = ""))

    # from the mygt object extract the info of the individual pops
    # now extract the 2-dimensional SFS
    mysfs2<-gt2sfs.raw(mygt, c(pop1,pop2))
    #if you don't have the ancestral allele fold the SFS
    mysfs2 <- fold.sfs(sfs = mysfs2)
    # write the sfs
    # write the names of the input. Since we need to automatize the code, we need to put a generic name "input" and fastsimcoal2 needs that the termination has specific characteristics, depending on the type of SFS (number of pops, folded or unfolded, etc, see manual)
    input <- paste("datasets/output/SFS_inputs/",path,"/input_jointMAFpop1_0.obs",sep = "") # here it is a joint (2 pops), with folded "MAF"
    write.2D.fsc(sfs = mysfs2,f.output = input)
    #map the two populations of interest, and color as a function of the category in which they grow, indicate the adaptive score of both, and whether it needs to adapt
```

```
  # the next code only works if you have linux system  ; it copies the fs program and the input files in each of the directories that were created.

    system(paste("cp datasets/input/fs datasets/output/SFS_inputs/",path,sep = ""))
    system(paste("cp datasets/input/input.* datasets/output/SFS_inputs/",path,sep=""))

  }
  }
```

Step 11. Run Fastsimcoal2

```
# this script was written in a Rscript.R code, and run using slurm  # this will only work on UNIX based systems
if(FALSE){
  setwd("SFS_inputs")
  # get all the names of the files 300 containing the pairs of populations input, for the mexicana dataset
  files <- list.files()
  # set number of replicates
  replicates <- 1:5
  # if you want particular populations you can indicate them with grep
  #files <- files[-c(grep("Cocotitlan_Cocotitlan",files),
  #  grep("SMH573_Acambaro",files))]

  # the next loop runs fastsimcoal for different replicates
  for(i in 1:length(files)){
    # name of the file
    run <- files[i]
    print(run)
    #indicate replicate running
    print(paste("missing",length(files)-i))
    # move to the file
    setwd(run)
    for(j in replicates){
      print(paste("running replicate",j, "out of",length(replicates)))
      #create a replicate
      rep <- paste("replicate_",j,sep = "")
      dir.create(rep)
      # copy the fs program (obtained from fastsimcoal download)
      system(paste("cp ../../fs",rep))
      # copy the input models indicating the model to run and the priors
      system(paste("cp ../../input*",rep))
      # copy the SFS
      system(paste("cp input_jointMAFpop1_0.obs ",rep,sep = ""))
      # get into the replicate file
      setwd(rep)
      # allow SFS to be executable
      system("chmod 777 fs")
      # run the analysis, see option in the manual or fs -h option
      system("./fs -t input.tpl  -n 100000 -L 40 -M 0.001 -e input.est -q --removeZeroSFS --core 20 -m   > out_input.txt")
```

```
# return to the pair of populations file
      setwd("..")
    }
# return to the SFS file
    setwd("..")
  }
}
```

4.2. Assessing the type of gene flow

We combine all replicate runs and select the 5% of simulations with the highest composite likelihoods. Next, we estimate the effective population size of populations (*Ne*) and the effective migration between populations (*Nem*). Previously, we categorized populations into warm-adapted and cold-adapted populations (based on the allele turnover functions). We use this information as a proxy to discriminate between likely adaptive (from warm to cold populations) and mal-adaptive gene flow (from cold to warm populations).

Step 12. Estimate population parameters

```
# The following chunk of code requires the tidyverse and the igraph libraries
# get the files with the pairs of populations
load("datasets/output/GO_objects.R")
path_t <- "datasets/SFS_inputs_runned/"
files <- list.files(path=path_t)
replicates <- 5
#get names of populations
populations <- GO_objects$categories %>% unlist()
names(populations) <- NULL
# the next loop will obtain the fastsimcoal outputs and combine them into a tibble after
obtain the top 5% parameters that maximize the Likelihood
for(i in 1:length(files)){
  # select a pair of populations file
  run <- files[i]
  print(run)
  for(j in 1:replicates){
    if(j==1){ #if it is the first replicate create the object output that will contain the data
      output <- read.csv(paste(path_t,"/",run,"/replicate_",j,"/","input/input.brent_lhoods",sep
= ""),sep = "\t")
      }else{#if not only add the new rows
      output <-
rbind(output,read.csv(paste(path_t,run,"/replicate_",j,"/","input/input.brent_lhoods",sep =
""),sep = "\t"))
    }
```

```
  }
  #remove any line that can have Estimated Likelihoods with NA
  cl <- which(is.na(output$MaxEstLhood))
  if(length(cl)!=0){
    output <- output[-cl,]
  }
  #order based on the likelihood and plot likelihood
  output <- output[order(output$MaxEstLhood),]
  #select the rows with the 5% top likelihoods and indicate the cut
  qt <- quantile(output$MaxEstLhood,probs = 0.95, na.rm = T)
  #select runs with the top likelihood
  output <- output[output$MaxEstLhood > qt,]
  #read the joint SFS to get the name of populations, extract the first column and row to get
it
  sfs <- read.table(paste(path_t,run,"input_jointMAFpop1_0.obs",sep="/"),header = T,skip =
1)
  pop1 <- rownames(sfs)[1]; pop1 <- sub("^d","",pop1); pop1 <- sub("_0","",pop1)
  pop2 <- colnames(sfs)[1]; pop2 <- sub("^d","",pop2); pop2 <- sub("_0","",pop2)
  spop1 <- coords[pop1,"temperature"]
  spop2 <- coords[pop2,"temperature"]

  # estimate Nem (Ne*m) between both populations, remember that it is a coalescent
process, and therefore we model 1 migrant moving backwards in time
  #mig21 means: migration from 1 to 2; or pop1 to pop2
  #mig12 means: migration from 2 to 1; or pop2 to pop1
  output$MIG21 <- output$MIG21*output$POP1 #pop1 is sending migrants
  output$MIG12 <- output$MIG12*output$POP2
  #build the output, indicating the names of the populations
  output <- output[,c("POP1","POP2","MIG21","MIG12")] #select the columns
  #for Ne indicate the populations
  names(output)[1:2]<-c(pop1,pop2)
  #for Nem indicate the sense of migration with TO
  names(output)[3] <- paste(pop1,"TO",pop2,sep = "")
  names(output)[4] <- paste(pop2,"TO",pop1,sep = "")
  # Create a temporary tibble with the data and combine it into a master tibble
  # Two vector with population names: from target to source
  source <-
c(rep(colnames(output)[1],dim(output)[1]),rep(colnames(output)[2],dim(output)[1]))
  target <- rev(source)
  # arrange the two vectors of Ne into one, ordered according to the target vector.
  ne <- c(output[,2],output[,1])
  # arrange the two vectors of Nem into one, ordered according to the target vector.
  nem <- c(output[,4],output[,3])
  # create vectore of target population adaptive scores
  score <- c(rep(spop2,dim(output)[1]),rep(spop1,dim(output)[1]))
  temp <-
tibble(Target=target,Source=source,Ne_Target=ne,Nem=nem,Score_source=score)
  if(!exists("master")) master <- temp else master <- bind_rows(master,temp)
}
master <- master %>%
mutate(Turn_source=GO_objects$turn_score$imp[match(master$Target,rownames(GO_o
```

```
bjects$turn_score))])
if (TRUE){
ids_w <- which(rownames(GO_objects$turn_score) %in% GO_objects$categories$warm)
tresh <- min(GO_objects$turn_score$imp[ids_w],na.rm=T)
# First, generate new column with Adaptive/Maladptive categories according to the
adaptive score
master <- master %>% mutate(Type=case_when(master$Turn_source < tresh ~
"Maladaptive",
                                   master$Turn_source >= tresh ~  "Adaptive")) %>%
  mutate(Type=as_factor(Type),Target=as_factor(Target),Source=as_factor(Source))
# Second, group_by pair of populations to summarise the parameters. Here we are using
the median (0.5 quantile), but this
# can be customized to any quantile of interest (e.g., 0.05)
# We are also creating a column with colors and alpha transparency according to the
adaptive score of populations (this can also be customized)
warm_col=MaizePal::maize_pal("JimmyRed",4)[2]
cold_col=MaizePal::maize_pal("MaizAzul",6)[2]
master_sum <- master %>% group_by(Target,Source) %>%
summarise(Ne_Target=quantile(Ne_Target,probs = 0.5, na.rm = T),
                                    Nem=quantile(Nem,probs = 0.5, na.rm = T),
Score_source=first(Score_source),Turn_source=first(Turn_source),Type=first(Type)) %>%
  mutate(Cols_adapt=case_when(Type=="Adaptive" ~
warm_col,Type=="Maladaptive"~cold_col)) %>%
  mutate(Alpha=case_when(Nem < 1 ~ 0.2,Nem >= 1 ~ 0.9)) %>%
rename(SourcePop=Target,TargetPop=Source)

save(master,file = "datasets/output/GeneFlow_FULL.Rdata") # individual results
save(master_sum,file = "datasets/output/GeneFlow_SUM.Rdata") # summarized
master_sum <- master_sum %>% ungroup() %>% group_by(TargetPop)%>%
pivot_wider(names_from = Score_source,values_from = Nem) %>%
summarise(WarmFlow=mean(warm,na.rm=T),ColdFlow=mean(cold,na.rm=T))
master_sum <- master_sum %>% mutate(Flow_prop=case_when(WarmFlow > ColdFlow
~ "Adaptive",WarmFlow < ColdFlow ~ "Maladaptive")) %>% ungroup()
  write.csv(master_sum,file = "datasets/conservation/vul_flow.csv")
}

load("datasets/output/GeneFlow_FULL.Rdata")
warm_col=MaizePal::maize_pal("JimmyRed",4)[2]
cold_col=MaizePal::maize_pal("MaizAzul",6)[2]
levels(master$Target)[13] -> pop
master %>% group_by(Target,Source) %>%
  mutate(Cols_adapt=case_when(Type=="Adaptive" ~ cold_col,Type=="Maladaptive"~
warm_col)) %>%
  mutate(Alpha=case_when(Nem < 1 ~ 0.2,Nem >= 1 ~ 0.9)) %>%
rename(SourcePop=Target,TargetPop=Source) %>%
  filter(TargetPop==pop) %>%
  ggplot(aes(x=Nem,y=fct_reorder(SourcePop, Turn_source),fill=Cols_adapt)) +
  geom_density_ridges_gradient(scale = 2.5, rel_min_height = 0.001,bandwidth = 0.05) +
  #scale_fill_gradientn(colours = MaizePal::maize_pal("HighlandMAGIC")) +
  #labs(title =paste('Gene flow into ',meta_target$SourcePop,"
(",round(meta_target$Score_source,2),")",sep="")) +
```

```
ylab("") +
geom_vline(xintercept = 1,linetype="dashed",size=1) +
theme(panel.grid=element_blank(), panel.grid.major.y=element_line(colour="grey80"),
  panel.border = element_rect(colour="grey50",fill=NA), legend.position="none",
  panel.spacing = unit(0.1, "lines"), strip.text.x = element_text(size = 8)
)
```

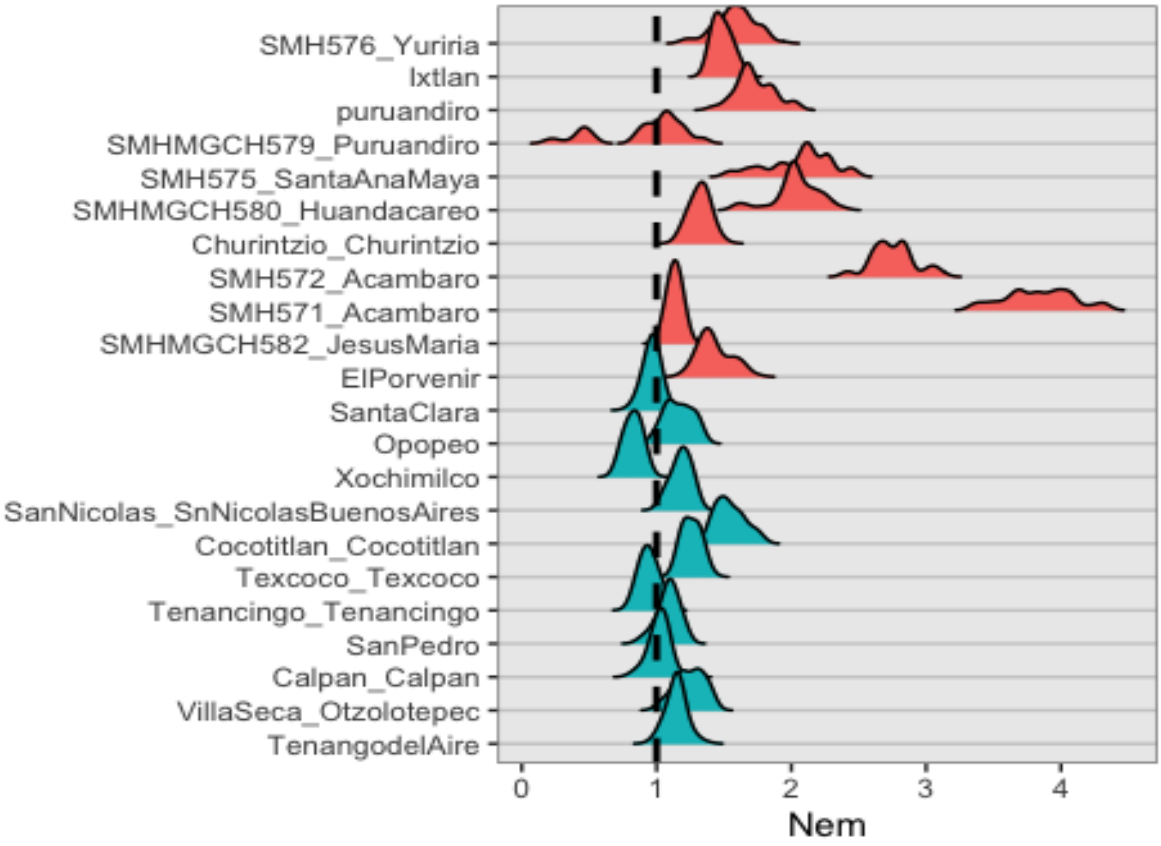


Figure S10. Distribution of gene flow estimates (Nem) into the Acámbaro population (target population), with colors depicting the provenance of the source populations: N-G1 cluster (red) and NG-2 cluster (blue).

5. Dispersion

5.1. Identify potential areas of dispersion

In this section, we combine estimates of migration and genetic diversity to study the movement of populations across the landscape. Importantly, all methods are based on allele turnover models and estimates of genetic offsets. Here we follow refs.[18–20] to identify potential areas of settlement of populations (forward genetic offest *sensu* refs.[19]). This approach aims at identifying areas where a target population would have the lowest possible genetic offset. Here, we use the 10% percentile of genetic offset values to define the threshold of genetic offset allowing population viability.

For these analyses we need:

– 1) Genomic offset output (Step 9): ""“datasets/output/GO_objects.R”. – 2) Climatic data: “datasets/input/present/.asc”, “datasets/input/year_50/.asc” , “datasets/input/year_70/.asc”

Step 13. Identify potential settlement sites

```
# first identify 10% of genetic offset as a threshold.
load("datasets/output/GO_objects.R")
gen_off <-
c(GO_objects$vulnerable_areas_50$vul)#,GO_objects$vulnerable_areas_70$vul)
model_gf <- GO_objects$gf_candidate
cords_pop <- GO_objects$gfData[,c("X","Y")]
qt <- quantile(gen_off,0.1)
# read the environmental raster layers and save them as a stack object (for present and
future layers)
present <- stack(list.files("datasets/input/present/",pattern = ".asc$",full.names = T))
future_50 <- stack(list.files("datasets/input/year_50/",pattern = ".asc$",full.names = T))
future_70 <- stack(list.files("datasets/input/year_70/",pattern = ".asc$",full.names = T))
qt_cost=0.9
  km_dist=10
  future=future_50
pops <- rownames(GO_objects$gfData) # get names of populations
    #create a list that will contain all the information of potential migration areas in the future
for each population
    matrix_list <- vector("list",length(pops))
    names(matrix_list) <- pops
    # create a table that will have summary data
    matrix_table <- data.frame(pop=pops,min=NA,median=NA,num_sites=NA,row.names =
"pop")
```

```
# Estimate populations genetic offset between their current location and all the future potential locations, and based on circuit theory estimates the cost of migrating to all potential areas. We run the code for one population, but it can be made into a loop.
 i=11
    # get coordinates of population i
    coord_temp <- cords_pop[i,]
    sp::coordinates(coord_temp) <- c("X", "Y")
    # create a buffer around the population considering the potential areas of migration (M in BAM model)
    pts_b <- rgeos::gBuffer(coord_temp, width=3) %>% as('SpatialPolygonsDataFrame')
    # get the present and future rasters and crop based on the buffer polygon
    matrix_future <- raster::crop(future, pts_b) %>% raster::mask(.,pts_b)%>% stack() %>% rasterToPoints() %>% data.frame()
    coords_furure <- matrix_future[,c("x","y")]
    # predict allelic frequencies in present and future
    matrix_future <- predict(model_gf,matrix_future[,grep("bio",names(matrix_future))])
    matrix_present <- data.frame(raster::extract(present,coord_temp@coords))
    matrix_present<- predict(model_gf,matrix_present[,grep("bio",names(matrix_present))])
    # create a matrix that contains the present bio values of population i repeated as many times as the potential future migration areas (this allows resting matrices for calculating the Euclidian distance.
    matrix_present <- rbind(matrix_present, matrix_present[rep(1, nrow(matrix_future)-1), ])

    #obtain genetic offset between the current location of a population and the future potential areas of migration to obtain the future genetic offset (as in Gougherty et al. 2020)
    gen_off <- (matrix_future-matrix_present)^2
    gen_off <- sqrt(rowSums(gen_off))
    gen_off <- data.frame(coords_furure,gen_off)
    gen_off <- gen_off[which(gen_off$gen_off<=qt),]
    #if there are no migration areas, set that the population will become extinct
    if(nrow(gen_off)==0){
      matrix_list[[i]] <- "Extinction"
      next
    }
    # get the geographic distances to all the areas where the population could migrate
    distances <- distGeo(cords_pop[i,c("X","Y")],gen_off[,c("x","y")])/1000
    gen_off$geo <- distances
```

5.2. Use resistance to estimate dispersal routes

We employed the approach of ref.[1] to estimate movement of populations across the landscape using Circuit Theory[24]. This approach involves three basic steps: (1) a landscape feature is transformed into a transition (resistance) matrix; (2) geographic distances between two sets of points; and (3) estimate the dispersal

route that has the lowest associated transition costs. Here, we transform the allele turnover functions along bio_9 into a transition matrix to estimate populations' dispersal costs across the landscape, but it is possible to estimate a transition matrix using multiple variables (*i.e.*, turnover functions). For each population, we estimate the minimum geographic distance of dispersal and the extent of the potential settlement area within a given geographic distance (*i.e.*, 20 km).

Step 14. Create the transition matrix

```
# first we obtain the turnvover function of the adaptive alleles across the candidate bio
    costs <- GO_objects$temp_cand_overall
    costs <- data.frame(bio=costs$x,turnover=costs$y)
    bio_cand <- GO_objects$bio_cand
    # get turnover functions of populations with function predict and relativize by the max
value
    # obtain the current candidate raster layer. this will be used as the migration layer
    temp_pres <- raster::crop(present, pts_b) %>% raster::mask(.,pts_b) %>% stack()
    # get the environmental values of all cells
    matrix_pres <- rasterToPoints(temp_pres) %>% data.frame()
    coords_pres <- matrix_pres[,c("x","y")]
    # predict the allelic turnover across the landscape
    turn_pres <- predict(model_gf,matrix_pres[,grep("bio",names(matrix_pres))])
    # get the value for the candidate value, and create a table that has all the bio values
and the GF predicted values
    turn_costs <- data.frame(bio_cand=matrix_pres[,bio_cand],turn=turn_pres[,bio_cand])
    turn_costs <- turn_costs[!duplicated(turn_costs$bio_cand),]

    # get the allelic value for the population
    pop_turn<- predict(model_gf,GO_objects$gf_candidate$X[i,])
    pop_turn <- pop_turn[,bio_cand]
    pop <- pops[i]
    coords_temp <- cords_pop[i,]
    # for the candidate variable, get the raster layer (migration matrix)
    mig <- temp_pres[[bio_cand]]
    # the next loop takes each value of the migration matrix (turn cost biocand values) and
transforms it to a migration cost based on the differential between the observed value of
the pop and the predicted change across the turnover function
    for(j in 1:nrow(turn_costs)){
  #pop_turn is the value of the pop at its bio_cand distribution; turn_costs[j,"turn"] is the
value of the turnover at bio_cand[j]; the differential is the cost of migration by moving from
1 to 2; we take absolute value because there is no order
      set_cost <- abs(pop_turn- turn_costs[j,"turn"])
      # the migration matrix at the bio_cand value j is transformed by the cost created
      mig[mig==turn_costs[j,"bio_cand"]]<- set_cost #for each grid it sets a cost
    }
    mig[mig<0]<-NA
    # transform migration cost from 0 to 1
    mig <- mig/max(values(mig),na.rm = T)
```

```
    # the least cost path model uses the reciprocal cost, so we transform it to 1-mig and
multiply by 1000, cost will go from 0 to 1000, with 1000 being lower migration costs and
plot
    cost_mig <- 1-mig
    cost_mig <- cost_mig*1000

    # create transition object, based on the costs of migration
    tr <- transition(cost_mig,transitionFunction = mean,directions = 8)
    # getall the possible coordinates where the population could settle in the future
(remove all 0 areas, not exist)
    future_matrix <- gen_off[,c("x","y")]

    # estimate the cost of migration between the focal population and all the potential
areas where it could migrate to; and transform it to a vector
    cost <- costDistance(tr,fromCoords = as.matrix(coord_temp@coords),#the coord of the
populations
                         toCoords = as.matrix(future_matrix[,c("x","y")]))# all the coord values of
settlement
    cost <- as.vector(cost)
    # create table cotaining the coordinates of potential migration and costs
    future_matrix <- data.frame(future_matrix,cost)
    future_matrix <- na.omit(future_matrix)

    # if migration is not possible (because there are no future matrix values) the matrix will
have no rows, indicate the population will go extinct
    if(nrow(future_matrix)==0){
      matrix_list[[i]] <- "Extinction"
      next
    }
    #get the distance in (km) to each potential area of migration and add it to the
future_matrix migration table
    distances <- distGeo(coord_temp@coords[,c("X","Y")],future_matrix[,c("x","y")])/1000
    future_matrix <- data.frame(future_matrix,distances=distances)
    # order the table by increasing distance and add a color with warmer colors indicating
lower migration potential (it will need to migrate more).
    temp <- future_matrix[order(future_matrix$distances),]
    # get the top cost and lower migration distance of populations
    qt_cst <- quantile(future_matrix$cost,qt_cost)
    qt_dist <-quantile(future_matrix$distances,1-qt_cost)
    # if any migration is at a low migration distance but a high cost, remove them, since it
is because migration resistance is high
    condition <- which(future_matrix$cost>qt_cst & future_matrix$distances<qt_dist)
    if(length(condition)>0){
      future_matrix <- future_matrix[-condition,]
    }
        matrix_list[[i]]<- future_matrix
    #add summary statistics to the table
    matrix_table[i,"min"]<-min(future_matrix$distances,na.rm=T)
    matrix_table[i,"median"]<-median(future_matrix$distances,na.rm=T)
    matrix_table[i,"num_sites"]<-length(which(future_matrix$distances<km_dist)) #number
of areas that are suitable below km_dist Kms
```

```
  # } # commented out as part of the loop
   # create an output and return it
   output <- list(matrix_list=matrix_list,matrix_table=matrix_table)

     mig %>% rasterToPoints() %>% data.frame() -> mig_p
      mig_p$col <- NA
      # order by costs to add colors depending of the genetic offset
     mig_p <- mig_p[order(mig_p$bio_9),]
     colores = c("grey90","grey70","grey45","grey20","grey10")
     mig_p$col <- as.character(cut(1:nrow(mig_p),length(colores),labels=colores))
      colores = maize_pal("MaizMorado")
     temp$col <- rev(as.character(cut(1:nrow(temp),length(colores),labels=colores)))
    # plot migration matrix transformed from bio cand to costs of migration

plot(extent(pts_b),type="n",main=rownames(cords_pop)[i],xlab="",xaxt="n",ylab="",yaxt="n
")
     maps::map("world","mexico",add=T)
     points(mig_p[,c("x","y")],col=alpha(colour = mig_p$col,alpha = 0.4),pch=15,cex=0.5)
     points(temp[,1:2],pch=15,col=temp$col,cex=0.5)

points(coord_temp,pch=21,col=maize_pal("HighlandMAGIC")[6],bg="white",lwd=2,cex=1.5
)
```

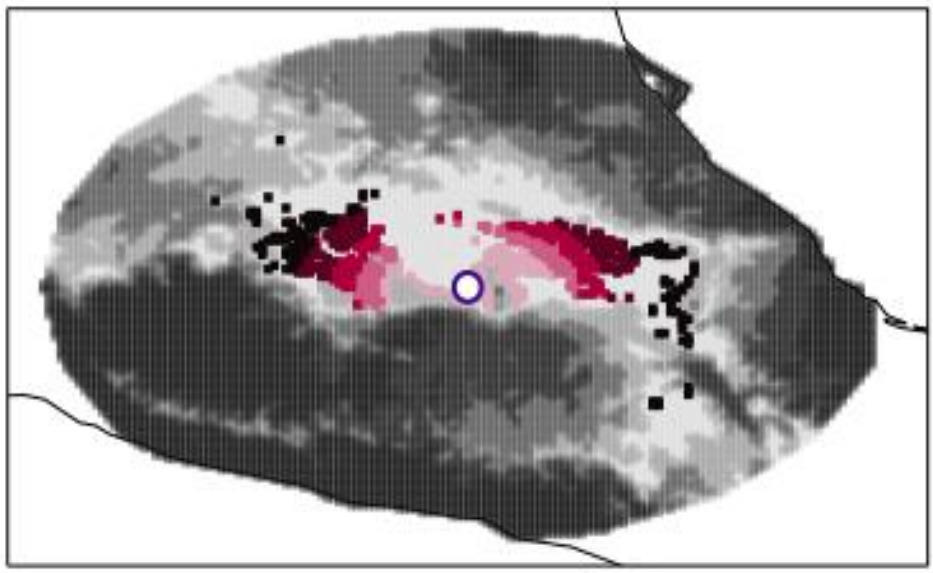


Figure S11. The distribution of migration costs encountered by the target population names Cocotitlán (purple circle) in the future if it were to migrate into potential settlement sites (based on the 10% percentile of genetic offset). Lighter colors indicate lower costs to migration, indicating low resistance for migration.

```
# We have created a function that estimates the potential migration of populations for
different time periods and for different adapted clusters. Create the function object and
execute.
# we create a function so this can be run for different time periods:
```

```
"climate_change_functions.R"

# we perform these runs for the 2050 and 2070 layers
  migration_50 <- run.migration(model_gf =
model_gf,cords_pop=cords_pop,present=present,future=future_50,qt = qt,qt_cost =
0.9,km_dist = 20)
  migration_70 <- run.migration(model_gf =
model_gf,cords_pop=cords_pop,present=present,future=future_70,qt = qt,qt_cost =
0.9,km_dist = 20)

  # save the data
migrations <- list(migration_50=migration_50,migration_70=migration_70)
save(migrations ,file = "datasets/output/migrations.R")

categories <- GO_objects$categories
warm_col=MaizePal::maize_pal("JimmyRed",4)
cold_col=MaizePal::maize_pal("MaizAzul",4)
load(file = "datasets/output/migrations.R")
migrations$migration_50  -> migration_50
migrations$migration_70  -> migration_70
migration_50$matrix_table$category <- NA
migration_50$matrix_table$col <- NA
migration_50$matrix_table[categories$cold,c("category","col")] <-
data.frame("cold",cold_col[2])
migration_50$matrix_table[categories$warm,c("category","col")] <-
data.frame("warm",warm_col[2])
migration_70$matrix_table$category <- NA
migration_70$matrix_table$col <- NA
migration_70$matrix_table[categories$cold,c("category","col")] <-
data.frame("cold",cold_col[2])
migration_70$matrix_table[categories$warm,c("category","col")] <-
data.frame("warm",warm_col[2])
migration_50$matrix_table <-
migration_50$matrix_table[order(migration_50$matrix_table$category),]
migration_70$matrix_table <-
migration_70$matrix_table[order(migration_70$matrix_table$category),]
# load migration objects
load(file = "datasets/output/migrations.R")
migration_50 <- migrations$migration_50
migration_70 <- migrations$migration_70
  GO_objects$present$col <- NA
  GO_objects$present <- GO_objects$present[order(GO_objects$present$bio_9),]
  cold <- which(GO_objects$present$bio_9< median(GO_objects$present$bio_9))
  id_ccol <-
as.character(cut(1:dim(GO_objects$present[cold,])[1],length(cold_col),labels=cold_col))
  id_wcol <- as.character(cut(1:dim(GO_objects$present[-
cold,])[1],length(warm_col),labels=warm_col))
  GO_objects$present$col <- c(id_ccol,rev(id_wcol))
  # add colors
  dodo_50 <-
match(rownames(migration_50$matrix_table),rownames(GO_objects$present))
```

```
dodo_70 <-
match(rownames(migration_70$matrix_table),rownames(GO_objects$present))
  migration_50$matrix_table$col <- GO_objects$present$col[dodo_50]
  migration_70$matrix_table$col <- GO_objects$present$col[dodo_70]
  migration_50$matrix_table$bio_9 <- GO_objects$present$bio_9[dodo_50]
  migration_70$matrix_table$bio_9 <- GO_objects$present$bio_9[dodo_70]
migration_50$matrix_table <-
migration_50$matrix_table[order(migration_50$matrix_table$bio_9),]
migration_70$matrix_table <-
migration_70$matrix_table[order(migration_70$matrix_table$bio_9),]

  #get tables and add names to define periods
  mig_50 <- migration_50$matrix_table; names(mig_50)<-
paste("year_50_",names(mig_50),sep = "")
  mig_70 <- migration_70$matrix_table; names(mig_70)<-
paste("year_70_",names(mig_70),sep = "")
  #combines tables and remove the color column
  mig_vul <- data.frame(mig_50,mig_70)
  mig_vul <- mig_vul[-grep("col",names(mig_vul))]
  #write the vulnerability table
  write.csv(mig_vul,file = "datasets/conservation/vul_mig.csv")

  # plot the different summary statistics
par(mfrow=c(2,2))
barplot(migration_50$matrix_table$min,col=migration_50$matrix_table$col,ylab="Minimu
m distance (km
)",names=rownames(migration_50$matrix_table),las=2,main="2050",xaxt="n")
barplot(migration_50$matrix_table$num_sites,col=migration_50$matrix_table$col,ylab="N
o. Suitable
sites",names=rownames(migration_50$matrix_table),las=2,main="2050",xaxt="n")

barplot(migration_70$matrix_table$min,col=migration_70$matrix_table$col,ylab="Minimu
m distance (km)",names=rownames(migration_70$matrix_table),las=2,main="2070")
barplot(migration_70$matrix_table$num_sites,col=migration_70$matrix_table$col,ylab="N
o. Suitable sites",names=rownames(migration_70$matrix_table),las=2,main="2070")
```

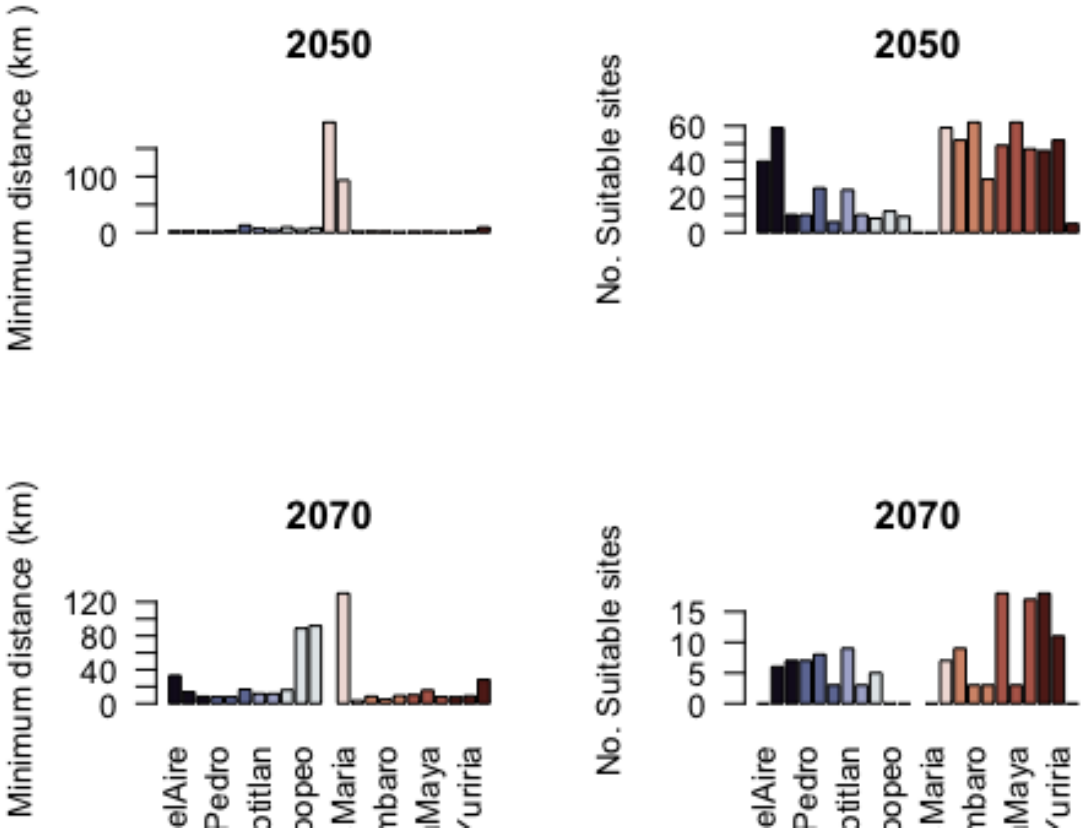


Figure S12. The estimated minimum migration distance and number of pontential settelement sites within a given distance for each population (left and right panels, respectively) for two time periods: 2050 and 2070. Colors correspond to the mean temperature of the driest quarter (bio_9) extracted for each population for the present-day.

6. Genomic load

6.1. Estimate Genomic load

A reduction in effective population sizes lead to increased levels of genetic drift and in consequence reduce the efficiency of selection leading to accumulation of deleterious mutations within populations (*i.e.*, genomic load) and ensuing loss of fitness[25], specially under rapidly changing climate. We estimate genomic load within populations and then assess these estimates as a function of the distance of populations to the ecological niche centroid[26,27], which is theoretically positively correlated with effective population density[11,28,29]. More specifically, we estimate genomic load as the proportion of non-synonymous to synonymous mutations

(pnFn/psFs), controlled by the frequency of the corresponding SNPs within populations[30].

For this analysis, we need:

– 1) Genomic data: “datasets/input/species_input.R”

– 2) SNPs annotations based on published datasets: “datasets/anotation_SNPs.csv”

– 3) Climatic data: “datasets/input/present/.asc”, “datasets/input/year_50/.asc” , “datasets/input/year_70/.asc”

Step 15 Estimate levels of load

```
################## load ############################
load("datasets/input/species_input.R")
# this takes some time, so if you have run it, you can change it to FALSE
if(FALSE){
  #read the tables with the annotations. Basically, we need information whether the SNP is S, NS or other
  snp <- read.table("datasets/anotation_SNPs.csv",header=T,sep="\t")
  # change all "." to "_" so that alleles can be considered in other sections (the names of loci have to be the same than from the genind section)
  snp$Original.Name <- gsub(".","_",snp$Original.Name,fixed = T)
  # change rownames to the SNPs
  snp <- data.frame(snp,row.names = "Original.Name")
  #get the names of SNPs from the genind object
  loc <- names(species_input$genpop$all.names)
  # get all shared SNPs and annotated SNPs and obtain the table containing them
  loc <- intersect(loc,rownames(snp))
  snp <- snp[loc,]
  #print type of SNPs
  print(summary(snp$Polymorphism.type))
  #get all SNPs that are synonymous and non synonymous rest
  NS_snp <- snp[which(snp$Polymorphism.type=="nonsynonymous"|snp$Polymorphism.type=="stopgain"|snp$Polymorphism.type=="stoploss"),]
  S_snp <- snp[which(snp$Polymorphism.type=="synonymous"),]

  # get allelic frequencies of the populations
  frecuencies <- mex_input$genpop$tab
  #the next loop searches all columns where the NS SNPs are found with the function grep
  list_NS_SNP <- NULL #create null vector that will held all coordinates
  for(i in 1:nrow(NS_snp)){
    s <- paste(rownames(NS_snp)[i],".",sep = "") #add a "." so it considers the alleles, and does not generate errors if the pattern is found in multple SNPs
    s <- grep(s,colnames(frecuencies),fixed = T) #search the pattern, by putting fixed we are sure that the loci ends before the alleles are inticated (it ends with ",").
    list_NS_SNP <- c(list_NS_SNP,s)
  }
```

```
  # same for synonymous SNPs
  list_S_SNP <- NULL
  for(i in 1:nrow(S_snp)){
    s <- paste(rownames(S_snp)[i],".",sep = "")
    s <- grep(s,colnames(frecuencies),fixed = T)
    list_S_SNP <- c(list_S_SNP,s)
  }
  # create a table containing the names of the populations, and add the columns indicating
different proportions of S and NS SNPs
  prop <-
data.frame(pop=rownames(frecuencies),prop_NS=NA,prop_S=NA,tot_NS=NA,tot_S=NA,
mean_NS=NA,mean_S=NA)
  prop$pop<- as.vector(prop$pop) #change from factor to vector

  # get the allelic frequencies of NS SNPs across populations
  # get genotypes
  frecuencies_NS <- frecuencies[,list_NS_SNP]
  #get P and Q, in adegenet each SNP has the alleles coded in two columns, so all odds
are allele p, and all pairs are allele q
  P <- frecuencies_NS[,seq(1,ncol(frecuencies_NS)-1,2)]
  Q <- frecuencies_NS[,seq(2,ncol(frecuencies_NS),2)]
  # get the frequency
  freq_NS <- Q/(P+Q)
  #the next code estimates the frequency of the type of SNPs, Q are the derived alleles so
we do it based on Q
  for(i in 1:nrow(Q)){
    # get for pop1 the number of SNPs that are NS, and divide by the entire numbr of SNPs
    prop[i,"prop_NS"] <- length(which(Q[i,]>0))/ncol(Q)
    prop[i,"tot_NS"]<-length(which(Q[i,]>0))
    #
    prop[i,"mean_NS"] <- mean(freq_NS[i,which(Q[i,]>0)],na.rm=T)
  }

  # same for synomymous SNPs
  frecuencies_S <- frecuencies[,list_S_SNP]
  P <- frecuencies_S[,seq(1,ncol(frecuencies_S)-1,2)]
  Q <- frecuencies_S[,seq(2,ncol(frecuencies_S),2)]
  freq_S <- Q/(P+Q)

  for(i in 1:nrow(Q)){
    prop[i,"prop_S"] <- length(which(Q[i,]>0))/ncol(Q)
    prop[i,"tot_S"]<-length(which(Q[i,]>0))
    prop[i,"mean_S"] <- mean(freq_S[i,which(Q[i,]>0)],na.rm=T)
  }
  # estimate genomic load summaries:
  #pns/ps:
  prop$pns_ps <- prop$tot_NS/(prop$tot_NS+prop$tot_S)
  #pnfn/psfs, the difference here is that we multiply by the frequency of the SNP
  prop$pnfn_psfs <- prop$prop_NS*prop$mean_NS/prop$prop_S*prop$mean_S
```

```
  # name by population
  prop <- data.frame(prop,row.names = "pop")
  #load categories to determine adaption based on adaptive score
  load("datasets/output/category_pops.R")

  # create variable score and add them based on the pops
  prop$score <- NA
  prop[rownames(sfs_categories$score),"score"]<-sfs_categories$score$score

  #set type of adaptation based on the adaptive score
  prop$temperature <- NA
  prop[which(prop$score<0.5),"temperature"] <- "cold"
  prop[which(prop$score>0.5),"temperature"] <- "warm"

  # add the coordinates to the table
  prop <- data.frame(mex_input$genpop$other[,c("longitude","latitude")],prop)
  #save the table of genomic load

  write.csv(prop,file = "datasets/output/genomic_load.csv",row.names = T)
} #this curly bracket closes the if(TRUE) condition above
```

6.2. Genomic load across the landscape

Here we model changes in genomic load within populations as a function of the distance of populations to the niche centroid, where populations farther away from the niche centroid inhabit more extreme (and less suitable) environments. We analyze each genetic cluster separately, estimating a niche centroid for each cluster and the distances of populations to the centroid of their corresponding cluster. We use Principal Component Analyses to reduce the dimensionality of the ecological space and perform all calculations on the resulting principal components (see ref.[27]).

Step 16. Distances to the niche centroids

```
# We calculate the distance to the niche centroid by estimating the multidimensional
euclidian distance between each population's environmental distribution and their niche
centroid
# first we obtain the coordinates of the environmental clusters identified in the SDM section
# read genetic load data
warm_col=MaizePal::maize_pal("JimmyRed",4)[2]
cold_col=MaizePal::maize_pal("MaizAzul",4)[2]
load_temp <- read.csv("datasets/output/genomic_load.csv",row.names = "X")
coords_mxt <- read.csv("datasets/output/maxent_input.csv")
# we also get the populations from the GF model & separate by environmental categories
coords_pop <- GO_objects$gfData[,c("X","Y")]
```

```
categories <- GO_objects$categories
coords_pop$cat <- NA
coords_pop[categories$cold,"cat"]<-"cold"
coords_pop[categories$warm,"cat"]<-"warm"
model <- GO_objects$gf_candidate # get turnover model for candidate SNPs
present <- stack(list.files(path = "Genetic_offset/present/",pattern = ".asc",full.names = T))
# get climate data
#get northern cluster (g1) and combine to the warm populations
g1 <- data.frame(coords_mxt[which(coords_mxt$ID=="g1"),c(2:3)],pops="g1");
names(g1)[1:2]<-c("X","Y")
warm <- data.frame(coords_pop[coords_pop$cat=="warm",c(1:2)])
warm$pops <- rownames(warm)
warm <- rbind(warm,g1)
warm <- warm[-which(duplicated(warm[,1:2])),] # remove duplicated records
raster::extract(present,warm[,c("X","Y")]) %>% data.frame() -> bios
warm_ras <- predict(model,bios) # predict turnover model for records
# obtain first 6 PCs
warm_ras <- prcomp(warm_ras,center = T,scale. = T)
warm_ras <- data.frame(warm_ras$x[,1:6])
# calculate centroid (mean of PCs)
niche_centroid <- colMeans(warm_ras[,grep("PC",names(warm_ras))])
# calucalte distance to the niche centroid (euclidian distance)
warm_ras$niche_centroid <- mapply(FUN = function(x,y) (x-
y)^2,x=warm_ras[,grep("PC",names(warm_ras))],y=niche_centroid) %>%  rowSums()
%>% sqrt()
#obtain sampled populations
warm <- data.frame(pops=warm$pops,niche_centroid=warm_ras$niche_centroid)
warm <- data.frame(warm[warm$pops!="g1",],row.names = "pops")
# add load data
load_warm <- data.frame(warm,load_temp[rownames(warm),])
# population El porvenir is an environmental outlier. Actually it grows closer to the g2
cluster. We remove it
load_warm <- load_warm[!rownames(load_warm)=="ElPorvenir",]
load_warm$col <- warm_col

g2 <- data.frame(coords_mxt[which(coords_mxt$ID=="g2"),c(2:3)],pops="g2");
names(g2)[1:2]<-c("X","Y")
cold <- data.frame(coords_pop[coords_pop$cat=="cold",c(1:2)])
cold$pops <- rownames(cold)
cold <- rbind(cold,g2)
cold <- cold[-which(duplicated(cold[,1:2])),]
raster::extract(present,cold[,c("X","Y")]) %>% data.frame() -> bios
cold_ras <- predict(model,bios)
cold_ras <- prcomp(cold_ras,center = T,scale. = T)
cold_ras <- data.frame(cold_ras$x[,1:6])
niche_centroid <- colMeans(cold_ras[,grep("PC",names(cold_ras))])
cold_ras$niche_centroid <- mapply(FUN = function(x,y) (x-
y)^2,x=cold_ras[,grep("PC",names(cold_ras))],y=niche_centroid) %>%  rowSums() %>%
sqrt()
cold <- data.frame(pops=cold$pops,niche_centroid=cold_ras$niche_centroid)
cold <- data.frame(cold[cold$pops!="g2",],row.names = "pops")
```

```
load_cold <- data.frame(cold,load_temp[rownames(cold),])
load_cold$col <- cold_col
# combine the two genetic clusters & plot
tot_load <- rbind(load_cold,load_warm)

# plot load based on the adaptive clusters
par(mfrow=c(1,2))
plot(load_temp$pnfn_psfs~factor(load_temp$category),ylab="Genomic
load",xlab="",col=c(cold_col,warm_col))
p <- summary(aov(load_temp$pnfn_psfs~load_temp$category))
p <- p[[1]][[5]][[1]]
# plot distance to the niche centroid
ylims=range(c(load_warm$niche_centroid,load_cold$niche_centroid))
ylims <- ylims + c(-.01,0.01)
plot(load_warm$niche_centroid,load_warm$pnfn_psfs,bg=adjustcolor(warm_col,alpha.f =
0.7),pch=21,ylab="Genomic load",xlab="DNC")
fit <- lm(load_warm$pnfn_psfs~load_warm$niche_centroid)
sfit <- summary(fit)
if(sfit$coefficients[2,4]<0.05){
  abline(fit)
}else{
  abline(fit,lty=2,col=warm_col)
}
points(load_cold$niche_centroid,load_cold$pnfn_psfs,bg=adjustcolor(cold_col,alpha.f =
0.7),pch=21)
fit <- lm(load_cold$pnfn_psfs~load_cold$niche_centroid)
sfit <- summary(fit)
if(sfit$coefficients[2,4]<0.05){
  abline(fit)
}else{
  abline(fit,lty=2,col=cold_col)
}
```

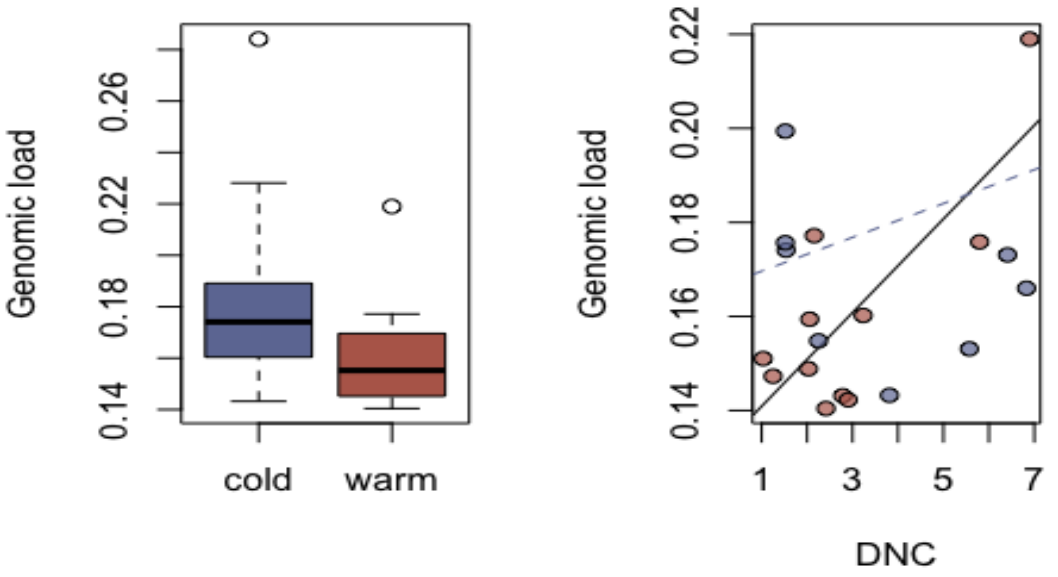


Figure S13. The genomic load estimated for the two clusters of populations of mexicana and its relationship with the distance to the ecological niche centroids (DNC) of the clusters: Northern (warm) and Southern (cold) (clusters in red and blue, respectively).

Step 17. Change in distances to the niche centroid

```
# Since there is a positive correlation between niche suitability and genomic load, we analyze how the distance to the niche centroid will change across the landscape to approximate changes in genetic load.
# get stacks of present and future bio variables
present <- stack(list.files(path = "datasets/input/present/",pattern = ".asc",full.names = T))
year_50 <- stack(list.files(path = "datasets/input/year_50/",pattern = ".asc",full.names = T))
year_70 <- stack(list.files(path = "datasets/input/year_70/",pattern = ".asc",full.names = T))
#get populations belonging to g1 (North) and predict GF turnover values
warm_pres <-
predict(model,data.frame(raster::extract(present,coords_mxt[coords_mxt$ID=="g1",c(2:3)])
))
warm_50 <-
predict(model,data.frame(raster::extract(year_50,coords_mxt[coords_mxt$ID=="g1",c(2:3)]
)))
warm_70 <-
predict(model,data.frame(raster::extract(year_70,coords_mxt[coords_mxt$ID=="g1",c(2:3)]
)))

# We perform a PCA using the turnover values, and then predict for the future
pca <- prcomp(warm_pres[,grep("bio_",names(warm_pres))],center = T,scale. = T)
warm_pc <- data.frame(warm_pres,pca$x[,1:6])
warm_pc$type <- "present"
niche_centroid <- colMeans(warm_pc[,grep("PC",names(warm_pc))])
```

```
# Predict for the future
warm_50_pc <- predict(pca, newdata = warm_50[,grep("bio_",names(warm_50))])[,1:6]
%>% cbind(warm_50,.)
warm_50_pc$type <- "fut_2050"
warm_70_pc <- predict(pca, newdata = warm_70[,grep("bio_",names(warm_70))])[,1:6]
%>% cbind(warm_70,.)
warm_70_pc$type <- "fut_2070"
pca_all_w <- rbind(warm_pc,warm_50_pc,warm_70_pc)
pca_all_w$niche_centroid <- mapply(FUN = function(x,y) (x-
y)^2,x=pca_all_w[,grep("PC",names(pca_all_w))],y=niche_centroid) %>%  rowSums()
%>% sqrt()
pca_all_w %>% mutate(Color=case_when(type == "present" ~ maize_pal("GlassGem")[4],
                                     type == "fut_2050" ~ maize_pal("GlassGem")[1],
                                     type == "fut_2070" ~ maize_pal("GlassGem")[2],)) -> pca_all_w
pca_all_w$type <- factor(pca_all_w$type, levels = c("present","fut_2050","fut_2070"))
cold_pres <-
predict(model,data.frame(raster::extract(present,coords_mxt[coords_mxt$ID=="g2",c(2:3)])
))
cold_50 <-
predict(model,data.frame(raster::extract(year_50,coords_mxt[coords_mxt$ID=="g2",c(2:3)]
)))
cold_70 <-
predict(model,data.frame(raster::extract(year_70,coords_mxt[coords_mxt$ID=="g2",c(2:3)]
)))
# repeat for other cluster
pca <- prcomp(cold_pres[,grep("bio_",names(cold_pres))],center = T,scale. = T)
cold_pc <- data.frame(cold_pres,pca$x[,1:6])
cold_pc$type <- "present"
niche_centroid <- colMeans(cold_pc[,grep("PC",names(cold_pc))])
cold_50_pc <- predict(pca, newdata = cold_50[,grep("bio_",names(cold_50))])[,1:6] %>%
cbind(cold_50,.)
cold_50_pc$type <- "fut_2050"
cold_70_pc <- predict(pca, newdata = cold_70[,grep("bio_",names(cold_70))])[,1:6] %>%
cbind(cold_70,.)
cold_70_pc$type <- "fut_2070"
pca_all_c <- rbind(cold_pc,cold_50_pc,cold_70_pc)
pca_all_c$niche_centroid <- mapply(FUN = function(x,y) (x-
y)^2,x=pca_all_c[,grep("PC",names(pca_all_c))],y=niche_centroid) %>%  rowSums() %>%
sqrt()
pca_all_c %>% mutate(Color=case_when(type == "present" ~ maize_pal("GlassGem")[4],
                                     type == "fut_2050" ~ maize_pal("GlassGem")[1],
                                     type == "fut_2070" ~ maize_pal("GlassGem")[2],)) -> pca_all_c
pca_all_c$type <- factor(pca_all_c$type, levels = c("present","fut_2050","fut_2070"))
# write vulnerability data
vul_load <- load_temp[,c("pnfn_psfs","category","col")]
write.csv(vul_load,file = "datasets/conservation/vul_load.csv")par(mfrow=c(1,2))
par(mai=c(1,1,1,0.2))
ylims=range(c(pca_all_w$niche_centroid,pca_all_c$niche_centroid))
plot(pca_all_w$niche_centroid~factor(pca_all_w$type),col=maize_pal("JimmyRed",4)[3:1],
ylab="cDNC",xlab="",las=2,ylim=ylims,main="Northern Cluster (N-G1)",pch=21,cex=0.4)
par(mai=c(1,0.2,1,1))
```

```
plot(pca_all_c$niche_centroid~factor(pca_all_c$type),col=maize_pal("MaizAzul",4)[3:1],yla
b="",xlab="",las=2,ylim=ylims,pch=21,cex=0.4,main="Southern Cluster (S-G2)")
```

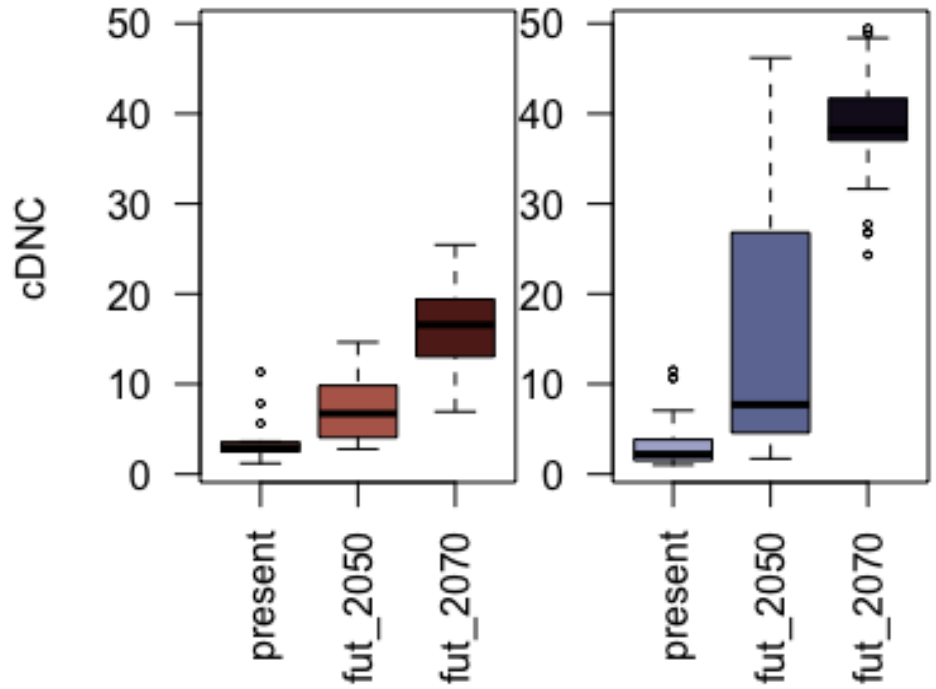


Figure S14. Change in populations distance to the niche centroid (cDNC) for two time periods (2050, 2070) for the two genetic clusters: Northern (red) and Southern (blue).

7. Evaluating all evolutionary processes

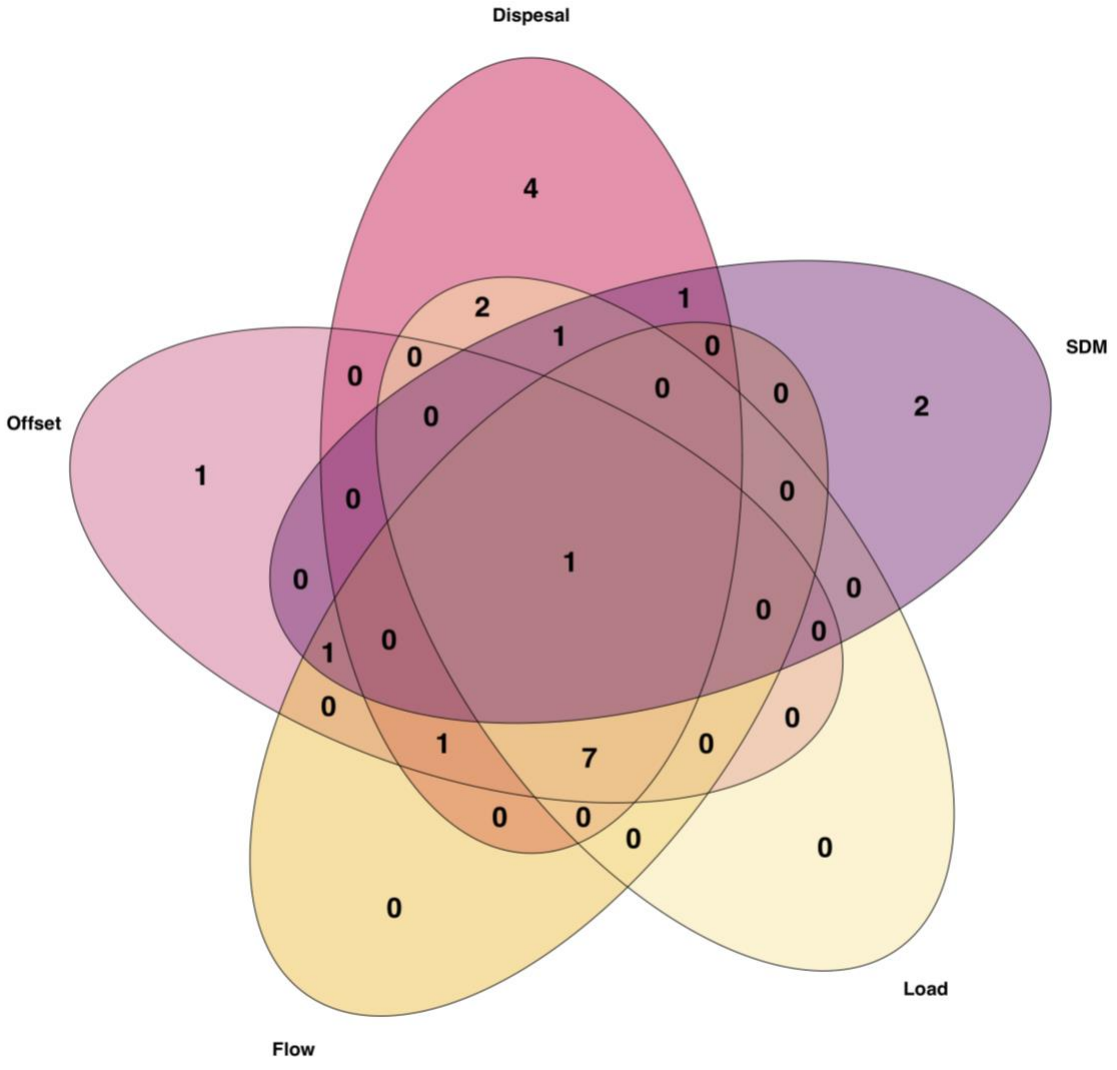
7.1. An integrated framework

Here we use population-level estimators to assess how each evolutionary processess will affect the response of species to climate change. We apply vulnerability threshold for each process and, using the basic idea behind the BAM diagram of ecological niche modeling, we use the intersection among processes (*i.e.*, low dispersal, high genetic offset) to predict different possible outcomes for population survival under climate change.

Step 18. Estimating and ploting intersections

```
# SDMs: whether the population is predicted in the future (1) or not (0)
vul_sdm <- read.csv("datasets/conservation/vul_sdm.csv",row.names = "X")
pop <- rownames(vul_sdm)
sdm <- rownames(vul_sdm)[which(vul_sdm$vals_70 == 1)]


# Genetic offset:  higher values indicate higher vulnerability
vul_offset <- read.csv("datasets/conservation/vul_fg.csv",row.names = "X.1")
rownames(vul_offset)[19] <- "puruandiro"
vul_offset <- vul_offset[pop,]
vul <- median(vul_offset$genetic_off_70,na.rm=T)
offset <- rownames(vul_offset)[which(vul_offset$genetic_off_70 < vul)]

# Gene flow: the levels of adaptive and maladaptive gene flow
vul_flow <- read.csv("datasets/conservation/vul_flow.csv",row.names = "TargetPop")
vul_flow <- vul_flow[pop,]
flow <- rownames(vul_flow)[which(vul_flow$Flow_prop=="Adaptive")]


# Migration: migration potential (approximated by minimum distance to future sites)
vul_migration <- read.csv("datasets/conservation/vul_mig.csv",row.names = "X")
vul_migration <- vul_migration[pop,]
migration <- rownames(vul_migration)[which(vul_migration$year_70_num_sites > 0 &
!is.na(vul_migration$year_70_num_sites))]


# Genomic load: estimated genetic load
vul_load <- read.csv("datasets/conservation/vul_load.csv",row.names = "X")
vul_load <- vul_load[pop,]
vul <- median(vul_load$pnfn_psfs,na.rm=T) # we use the median as a threshold
load <- rownames(vul_load)[which(vul_load$pnfn_psfs < vul)]
all <- list(Dispesal=migration,Offset=offset,Flow=flow,Load=load,SDM=sdm)

# plot the venn diagram
```

```
VennDiagram::venn.diagram(
        x = all, filename = 'datasets/output/venn_diagram.png',
        output=TRUE, imagetype="png" ,  resolution = 300, compression = "lzw", lwd = 0.6,
        lty = 1, fill = maize_pal("HighlandMAGIC",5), cex = 1.2, fontface = "bold",fontfamily =
"sans", cat.cex = 0.8, cat.fontface = "bold", cat.default.pos = "outer", cat.fontfamily =
"sans",margin=0.2
)
```



s

Figure S15. Venn diagram depicting the number of populations meeting either of the following criteria: 1) predicted by species distribution models (Presence); 2) a genetic offset below the median (Low Offset); 3) a higher proportion of adaptive to maladaptive gene flow (Adaptive Flow); 4) potential dispersal within short distances (Dispersal); and 5) genetic load below the median (Low Load).

References

1. Aguirre-Liguori, J. A., Ramírez-Barahona, S., Tiffin, P. & Eguiarte, L. E. Climate change is predicted to disrupt patterns of local adaptation in wild and cultivated maize. *Proceedings of the Royal Society B: Biological Sciences* **286**, 20190486 (2019).
2. Fick, S. E. & Hijmans, R. J. WorldClim 2: new 1-km spatial resolution climate surfaces for global land areas. *International Journal of Climatology* **37**, 4302–4315 (2017).
3. Hijmans, R., Cameron, S., Parra, J., Jones, P. & Jarvis, A. Very high-resolution interpolated climate surfaces for global land areas. *International Journal of Climatology* **25**, 1965–1978 (2005).
4. Frichot, E. & François, O. LEA: An R package for landscape and ecological association studies. *Methods in Ecology and Evolution* **6**, 925–929 (2015).
5. Hällfors, M. H. *et al.* Addressing potential local adaptation in species distribution models: Implications for conservation under climate change. *Ecological Applications* **26**, 1154–1169 (2016).
6. Ikeda, H. & Setoguchi, H. Phylogeography and refugia of the Japanese endemic alpine plant, Phyllodoce nipponica Makino (Ericaceae). *Journal of Biogeography* **34**, 169–176 (2007).
7. Sanchez-Gonzalez, J. D. J., De La Cruz Larios, L., Ruíz-Corral, J. A. & Miranda-Medrano, R. Base de datos de la distribución histórica de teocintle. Producto del programa del monitoreo de las poblaciones de teocintle (Zea spp.), aprovechamiento y estrategias de conservación en México. (2017).
8. Phillips, S. & Dudík, M. Modeling of species distributions with axent: new extensions and comprehensive evaluation. *Ecography* **31**, 161–175 (2008).
9. Phillips, S. J., Anderson, R. & Schapire, R. Maximum entropy modeling of species geographic distributions. *Ecological Modelling* **190**, 231–259 (2006).
10. Foden, W. B. *et al.* Climate change vulnerability assessment of species. *Wiley Interdisciplinary Reviews: Climate Change* **10**, 1–36 (2019).
11. Peterson, A. T. *et al. Ecological niches and geographic distributions*. 328 (Princeton University Press, 2011).
12. Peterson, A. T. & Soberón, J. Species distribution modeling and ecological niche modeling - getting the concepts right. **10**, 102–107 (2012).
13. Fitzpatrick, M. & Keller, S. Ecological genomics meets community-level modelling of biodiversity: mapping the genomic landscape of current and future environmental adaptation. *Ecology Letters* **18**, 1–16 (2015).
14. Bay, R. A. *et al.* Genomic signals of selection predict climate-driven population declines in a migratory bird. *Science* **359**, 83–86 (2018).
15. Capblancq, T., Fitzpatrick, M. C., Bay, R. A., Exposito-Alonso, M. & Keller, S. R. Genomic Prediction of (Mal)Adaptation across Current and Future Climatic Landscapes. *Annual Review of Ecology, Evolution, and Systematics* **51**, 245–269 (2020).
16. Exposito-Alonso, M., Burbano, H. A., Bossdorf, O., Nielsen, R. & Weigel, D. Natural selection on the Arabidopsis thaliana genome in present and future climates. *Nature* **573**, 126–129 (2019).
17. Fitzpatrick, M., Chhatre, V., Soolanayakanahally, R. & Keller, S. Experimental support for genomic prediction of climate maladaptation using the machine learning approach Gradient Forests. 1–22 (2020).
18. Gougherty, A. V., Keller, S. R., Chhatre, V. E. & Fitzpatrick, M. C. Future climate change promotes novel gene-climate associations in balsam poplar (Populus balsamifera L.), a forest tree species. *biorxiv* (2020) doi:10.1101/2020.02.28.961060.
19. Gougherty, A. V., Keller, S. R. & Fitzpatrick, M. C. Maladaptation, migration and extirpation fuel climate change risk in a forest tree species. *Nature Climate Change* (2021) doi:10.1038/s41558-020-00968-6.

20. Rhoné, B. *et al.* Pearl millet genomic vulnerability to climate change in West Africa highlights the need for regional collaboration. *Nature Communications* **11**, 5274 (2020).
21. Ruegg, K. *et al.* Ecological genomics predicts climate vulnerability in an endangered southwestern songbird. *Ecology Letters* **21**, 1085–1096 (2018).
22. Excoffier, L. & Foll, M. fastsimcoal: A continuous-time coalescent simulator of genomic diversity under arbitrarily complex evolutionary scenarios. *Bioinformatics* **27**, 1332–1334 (2011).
23. Excoffier, L., Dupanloup, I., Huerta-Sánchez, E., Sousa, V. C. & Foll, M. Robust Demographic Inference from Genomic and SNP Data. *PLoS Genetics* **9**, e1003905 (2013).
24. McRae, B. H., Dickson, B. G., Keitt, T. H. & Shah, V. B. Using circuit theory to model connectivity in ecology, evolution, and conservation. *Ecology* **89**, 2712–2724 (2008).
25. Frankham, R. Genetics and extinction. *Biological Conservation* **126**, 131–140 (2005).
26. Eckert, C. G., Samis, K. E. & Lougheed, S. C. Genetic variation across species' geographical ranges: The central-marginal hypothesis and beyond. *Molecular Ecology* **17**, 1170–1188 (2008).
27. Lira-Noriega, A. & Manthey, J. D. Relationship of genetic diversity and niche centrality: A survey and analysis. *Evolution* **68**, 1082–1093 (2014).
28. Yañez-Arenas, C., Martínez-Meyer, E., Mandujano, S. & Rojas-Soto, O. Modelling geographic patterns of population density of the white-tailed deer in central Mexico by implementing ecological niche theory. *Oikos* **121**, 2081–2089 (2012).
29. Yañez-Arenas, C., Guevara, R., Martínez-Meyer, E., Mandujano, S. & Lobo, J. M. Predicting species' abundances from occurrence data: Effects of sample size and bias. *Ecological Modelling* **294**, 36–41 (2014).
30. Willi, Y., Fracassetti, M., Zoller, S. & Van Buskirk, J. Accumulation of mutational load at the edges of a species range. *Molecular Biology and Evolution* **35**, 781–791 (2018).