Análisis espacio-temporal de la dinámica dispersiva de *B. terrestris*

Lican Martínez

climtic.data.porcess= T # preocess climatic data and PC computation  
  
mcmc.exec= T # run mcmc chains  
  
run.pp.check.sims= T # run simulations of BT from 1997 for ppcheck  
  
run.future.BT.sims= T # run simulations ob BT from 2021 to future

library(ggplot2) #ploting  
library(GGally) # for scatter matrix plot  
library(tidyr) #datamanipulation   
library(raster) #rastermanipulation  
library(dismo) #maxentandgbiffunctions  
library(arm) #invlogit()function  
library(RStoolbox) #forrasterPCA()function  
library(corrplot)  
library(spatstat) # for density kenrels  
library(rgdal)  
library(sp)  
library(sf)  
library(rgeos)  
library(MASS) # funcion mvrnorm   
library(reshape)  
  
library(coda) # for mcmc diagnosis  
library(knitr)   
library(truncnorm)   
library(dplyr)  
library(viridis)  
library(maptools)  
library(ggExtra)  
library(ggpubr)  
library(latex2exp)  
library(cowplot)  
library(ggsn) # for sclaes, not working  
library(ggspatial)   
library(rnaturalearth)  
library(devEMF) # vector word graphics

# Introducción

El objetivo de esta parte del trabajo es parametrizar un modelo de dispersión de *B. terrestris* en Patagonia a partir de datos de ocurrencias registradas históricamente a lo largo de la región. El modelo pretende contemplar dos aspectos: la capacidad de dispersión espacial y el efecto de las condiciones ambientales en este proceso.

Para esto me basaré en una estrategia similar a la propuesta por **Cook et al. (2007)** en la que se construye una función de likelihood para los datos en función de parametros de un kernel de dispersión y un modelo de nicho ambiental. Esta función de likelihood puede ser luego utilizada para estimar las distribuciones posteriores de los parámetros mediante un enfoque bayesiano.

El abordaje consiste en discretizar los datos espaciales en celdas y tratar de conectarlos con parámetros que determinen la tasa de colonización de las mismas. Estos parámetros estarían asociados a las condiciones ambientales de una celda particular (componente de nicho ambinetal) y las distancias a las celdas que hayan sido colonizadas previamente a esa (componente de dispersión espacial).

En un escenario simplificado, podemos modelar el proceso de colonización de una celda a otra como un proceso de *Poisson*:

en el que es la probabilidad de que el tiempo que tarda una celda en ser colonizada por otra sea , y es la tasa de colonización. En este modelo podemos hacer que dependa de la distancia entre las celdas y las condiciones ambientales de la celda colonizada.

donde es la distancia euclidea entre las celdas, es un parámetro del kernel de dispersión, y es la colonizabilidad intrínseca de la celda de destino (*suitabilidad*) dada por las condiciones ambientales y los parámetros del modelo de nihco . Como una celda puede tener más de una fuente de colonización, podemos integrar las contribuciones de todas las celdas fuente y considerar la probabilidad de colonización a un tiempo

donde es la probabilidad de que la celda sea colonizada en el tiempo observado dado un conjunto de celdas colonizadas cada una a tiempo (tal que ), las cuales están a distancias de la celda .

Por otra parte, también podemos calcular la probabilidad de la no colonización de una celda por parte de otra durante un período de registro como

lo que al integrar las contribciones de todas las celdas colonizadas queda como

donde es el último año del período contemplado.

Para calcular un likelihood de todo el conjunto de datos, podemos computar el productorio de las probabilidades de colonización de todas las celdas colonizadas y de no colonización de todas las celdas sin registros

Por razones computacionales, es conveniente evaluar el *log-likelihood* ya que los productorios de una gran cantidad likelihoods pueden arrojar valores demasiado pequeños.

## Disclaimer

Este modelo supone que las celdas sin registros no han sido colonizadas. Por lo tanto, ignora la posibilidad de falsos negativos: celdas en las que la especie está presente pero no ha sido observada debido a la incompletitud del registro. Sortear este problema es más complicado de lo que uno podría suponer *a priori*, ya que para contemplar el sesgo de muestreo no alcanza con estimar un parámetro constante asociado a la probabilidad de no registrar a la especie en una celda colonizada, sino que es necesario tener en cuenta la probabilidad de colonización de las celdas con registros desde todas aquellas que se observan vacías. En la práctica, esto consistiría en construír y parametrizar un modelo oculto de dispersión, sobre el cual se acoplaría un modelo de observación de los registros. La escala de complejidad de un modelo así sería mucho mayor a la del que presento en este trabajo, por lo que, por simplicidad, de momento considero aceptable no relajar el supuesto de la no existencia de falsos negativos.

# Implementación

## Variables ambientales

global\_bioclim.dir='D:/global\_bioclim/'  
  
  
if (climtic.data.porcess) {  
   
 r=stack(paste0(global\_bioclim.dir, 'bioclim\_res2.grd'))  
   
 lonlat.crs=crs(r$bio1)  
   
 nproj="+proj=utm +zone=19H +datum=WGS84"  
   
 # patagonia no masking bioclimatic data  
 ext=c(-76, -65, -58, -25)  
 bio.patag.all=crop(x = r, y=ext)%>%  
 projectRaster(crs=nproj)  
   
 # latinoamerica bioclimatic data  
 latam.proj="+proj=utm +zone=21H +datum=WGS84"  
   
 latam.ext=c(-90, -25, -60, 15)  
 bio.latam=crop(r, y=latam.ext)  
 bio.latam.utm=projectRaster(bio.latam, crs = latam.proj)  
   
 pc.latam.latam=RStoolbox::rasterPCA(bio.latam.utm, spca=T)  
 names(pc.latam.latam$map)=paste0('PC\_', 1:19)  
   
 pc.latam.patag=raster::predict(bio.patag.all, pc.latam.latam$model, index=1:19) %>%   
 stack()  
 names(pc.latam.patag)=paste0('PC\_', 1:19)  
   
 # europe bioclimatic data and PCs  
 bio.eu=crop(x = r, y=c(-12, 30, 34, 72))  
   
 pc.eu.from.latam=raster::predict(bio.eu, pc.latam.latam$model, index=1:19) %>%   
 stack()  
 names(pc.eu.from.latam)=paste0('PC\_', 1:19)  
   
   
 saveRDS(lonlat.crs, file='./intermediate\_files/lonlat.crs.RDS')  
 saveRDS(nproj, file='./intermediate\_files/nproj.RDS')  
 saveRDS(latam.proj, file='./intermediate\_files/latam.proj.RDS')  
 saveRDS(latam.ext, file='./intermediate\_files/latam.ext.RDS')  
 saveRDS(bio.latam.utm, file='./intermediate\_files/bio.latam.utm.RDS')  
 saveRDS(pc.latam.latam, file='./intermediate\_files/pc.latam.latam.RDS')  
 saveRDS(pc.latam.patag, file='./intermediate\_files/pc.latam.patag.RDS')  
 saveRDS(bio.eu, file='./intermediate\_files/bio.eu.RDS')  
 saveRDS(pc.eu.from.latam, file='./intermediate\_files/pc.eu.from.latam.RDS')  
  
}else{  
 lonlat.crs = readRDS(file='./intermediate\_files/lonlat.crs.RDS')  
 nproj = readRDS(file='./intermediate\_files/nproj.RDS')  
 latam.proj = readRDS(file='./intermediate\_files/latam.proj.RDS')  
 latam.ext = readRDS(file='./intermediate\_files/latam.ext.RDS')  
 bio.latam.utm = readRDS(file='./intermediate\_files/bio.latam.utm.RDS')  
 pc.latam.latam = readRDS(file='./intermediate\_files/pc.latam.latam.RDS')  
 pc.latam.patag = readRDS(file='./intermediate\_files/pc.latam.patag.RDS')  
 bio.eu = readRDS(file='./intermediate\_files/bio.eu.RDS')  
 pc.eu.from.latam = readRDS(file='./intermediate\_files/pc.eu.from.latam.RDS')  
}

sum.pc=pc.latam.latam$model%>%summary()  
  
vars <- sum.pc$sdev^2  
vars <- vars/sum(vars)  
  
vars.cum=round(vars\*100, 1)  
  
pc3.cum=round(cumsum(vars)[3]\*100, 1)

Para considerar los factores ambientales se tuvieron en cuenta 19 variables bioclimáticas de la base de datos WorlClim2. Estas fueron procesadas mediante un análisis de componentes principales (PCA) utilizando los datos de todo Sudamérica, conservando las primeras 3 componentes (84.1 % de la varianza).

plot(pc.latam.latam$model,   
 main='', las=2)



#layout(matrix(1:3, ncol=3))  
#par(mar=c(4, 3, 4, 3))  
  
#for (i in 1:3) {  
# plot(pc.latam.latam$map[[i]],   
# main=paste0('PC\_', i))  
#}

load=pc.latam.latam$model$loadings[,1:3]%>%as.data.frame()  
  
load$var=rownames(load)  
  
load=pivot\_longer(load, cols = 1:3, names\_to = )  
  
  
load$name=as.factor(load$name)  
#load=load[load$name=='Comp.1',]  
  
  
load.or=data.frame(var=NULL,   
 name=NULL,   
 value=NULL)  
  
load.plot.ls=list()  
  
for (i in 1:length(levels(load$name))) {  
   
 load.i=load[load$name==levels(load$name)[i], ]  
   
 #print(load.i)  
   
 load.i$var=reorder(load.i$var, load.i$value)  
   
 #print(str(load.i))  
   
 #load.or=rbind(load.or, load.i)  
   
 p.i=ggplot(load.i)+  
 geom\_point(aes(var, value), pch=21, col='black',   
 fill='blue', alpha=0.7, size = 3)+  
 facet\_grid(.~name)+  
 theme\_bw()+  
 theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))+  
 geom\_hline(yintercept = 0)+  
 ylim(-0.5, 0.5)+   
 geom\_segment( aes(x=var, xend=var, y=0, yend=value))+  
 xlab('Variable')+ylab('Loading value')+  
 coord\_flip()  
   
 load.plot.ls[[i]]=p.i  
 #plot(p.i)  
}  
  
ggarrange(plotlist = load.plot.ls, ncol=3)



pc\_map\_list=list()  
for (i in 1:3) {  
 pc\_map\_list[[i]]=ggplot()+  
 geom\_raster(data=pc.latam.latam$map[[i]],  
 aes\_string(x='x', y='y', fill=paste0('PC\_', i)))+  
 theme\_void()+  
 #scale\_fill\_viridis(option = 'H', na.value = "white")+   
 scale\_fill\_distiller(palette = 'Spectral', na.value = 'white')+  
 coord\_equal()+  
 ggtitle(paste0('PC ', i))+  
 theme(legend.position = 'bottom',   
 plot.title = element\_text(hjust=0.5))+  
 labs(fill=NULL)  
}  
  
  
  
gg.pc.maps=ggarrange(plotlist=pc\_map\_list, ncol = 3)  
  
print(gg.pc.maps)



# european ocurrences  
bter.eu.points=read.csv('./data/bter\_eur.csv',   
 quote = "", sep='\t')  
bter.eu.points=data.frame(lon=bter.eu.points$decimalLongitude,   
 lat=bter.eu.points$decimalLatitude,   
 year=bter.eu.points$year)  
bter.eu.points=bter.eu.points[complete.cases(bter.eu.points),]  
bter.eu.points=unique(bter.eu.points)  
  
# european ocurrences rasterization  
  
## rasterization  
bter.eu.ras=rasterize(bter.eu.points[,1:2], pc.eu.from.latam$PC\_1)  
  
## set all positive to 1  
bter.eu.ras[bter.eu.ras[]>=1]=1  
  
# extract coordinates of colonized cells  
bter.eu.coor=coordinates(bter.eu.ras)[!is.na(bter.eu.ras[])&  
 bter.eu.ras[]==1, ]  
  
bter.eu=bter.eu.coor%>%as.data.frame()  
names(bter.eu)=c('lon', 'lat')

#pseudoabseces  
# heavy run  
  
  
pa.factor=10 # cantidad de background points en Europa  
  
pa.beter.eu=randomPoints(mask=pc.eu.from.latam, n=nrow(bter.eu)\*pa.factor)%>%  
 as.data.frame()%>%`colnames<-`(c('lon', 'lat'))  
pa.beter.eu$type=0  
  
bter.eu$type=1  
  
oc.pa.beter.eu=rbind(pa.beter.eu, bter.eu[,c('lon', 'lat', 'type')])  
  
eu.oc.pc=raster::extract(pc.eu.from.latam, oc.pa.beter.eu[,c('lon', 'lat')])%>%  
 cbind(oc.pa.beter.eu)  
  
prior.niche=glm(eu.oc.pc, family = binomial(link='logit'),  
 formula = type~  
 PC\_1+ I(PC\_1^2)+  
 PC\_2+ I(PC\_2^2)+   
 PC\_3+ I(PC\_3^2)  
)  
  
coef=t(prior.niche$coefficients)%>%data.frame()  
  
conf=confint(prior.niche)  
  
#SEs=(coef-t(conf[,1]))/2  
SEs=coef-t(conf[,1])  
  
# plot europe ocurrences, prior suits and expl variables  
prior.niche.europe=predict(pc.eu.from.latam, prior.niche)%>%  
 invlogit()  
  
prior.suit=predict(pc.latam.patag, prior.niche)%>%  
 invlogit()  
  
  
sd.prior.factor=5

La forma en la que estas variables se tuvieron en cuenta en la estimación de la suitabilidad ambiental -la función mencionada previamente- fue mediante la construcción de un predictor lineal de segundo orden y una función de enlace .

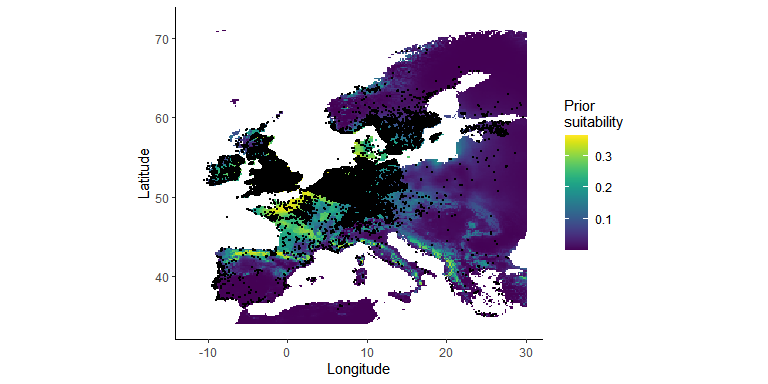
donde es el factor de suitabilidad y los son parámetros a estimar a partir de los datos. Esta estimación sería equivalente a parametrizar el nicho climático de *B. terrestris*. Como la estrategia utilizada para la estimación de los parámetros es Bayesiana, es posible utilizar información previa sobre el nicho de esta especie. En este caso, es razonable considerar que la suitabilidad ambiental para la colonización por parte de *B. terrestris* en latinoamérica sea relativamente similar al nicho climático ocupado en su región de origen. Teniendo en cuenta esto, se utilizaron datos de ocurrencias de Europa para hacer una estimación aproximada de los valores *a priori* de cada mediante la aplicación de Modelos Lineales Generalizados (GLM). Para obtener las variables explicativas de este GLM se extrapoló el modelo de PCA construido con las variables bioclimáticas en Latinoamérica a la región europea.

#layout(matrix(1:3, ncol=3))  
#par(mar=c(3, 3, 3, 2))  
  
#for (i in 1:3) {  
# plot(pc.eu.from.latam[[i]],   
# main=paste0('PC\_', i))  
#}  
  
pc\_map\_list.eu=list()  
for (i in 1:3) {  
 pc\_map\_list.eu[[i]]=ggplot()+  
 geom\_raster(data=pc.eu.from.latam[[i]],  
 aes\_string(x='x', y='y', fill=paste0('PC\_', i)))+  
 theme\_void()+  
 #scale\_fill\_viridis(option = 'H', na.value = "white")+   
 scale\_fill\_distiller(palette = 'Spectral', na.value = 'white')+   
 coord\_equal()+  
 ggtitle(paste0('PC ', i))+  
 theme(legend.position = 'bottom',   
 plot.title = element\_text(hjust=0.5))+  
 labs(fill=NULL)  
}  
  
  
  
gg.pc.maps.eu=ggarrange(plotlist=pc\_map\_list.eu, ncol = 3)  
  
print(gg.pc.maps.eu)



De estas capas se extrajeron los valores correspondientes a las 18851 celdas con registros de *B. terrestris* , y a una cantidad 10 veces mayor de puntos muestreados aleatoriamente a modo de *background*. El predictor lineal del GLM se construyó de la misma manera que el planteado arriba para el modelo de suitabilidad, utilizando como variable respuesta el tipo de dato (*1 = ocurrecia, 0 = background*) y una función de enlace *logit*.

ggplot(mapping = aes(y=lat, x=lon))+  
 geom\_raster(data=prior.niche.europe, alpha=1,  
 aes(fill=layer, x=x, y=y)) +  
 geom\_point(data=bter.eu, lwd=0.001, col='black')+  
 theme\_classic()+  
 scale\_fill\_viridis(option = 'D', na.value = "white", direction = 1)+   
 #scale\_fill\_distiller(palette = 'Spectral', na.value = 'white')+  
 #scale\_fill\_gradientn(colours = terrain.colors(7), na.value = "white")+  
 xlim(extent(prior.niche.europe)[1:2])+   
 coord\_equal()+  
 labs(fill='Prior  
suitability')+xlab('Longitude')+ylab('Latitude')



Los puntos negros corresponden a las coordenadas de las 18851 celdas con registros de *B. terrestris*.

confint=as.data.frame(conf)  
confint$Estimate=t(coef[1,])  
  
confint=confint[,c(3, 1, 2)]  
  
library(knitr)  
kable(round(confint, 2), caption = 'GLM estimates')

GLM estimates

|  | Estimate | 2.5 % | 97.5 % |
| --- | --- | --- | --- |
| (Intercept) | -11.04 | -11.48 | -10.60 |
| PC\_1 | -2.57 | -2.73 | -2.41 |
| I(PC\_1^2) | -0.26 | -0.27 | -0.25 |
| PC\_2 | -2.47 | -2.56 | -2.39 |
| I(PC\_2^2) | -0.40 | -0.42 | -0.39 |
| PC\_3 | 0.76 | 0.69 | 0.84 |
| I(PC\_3^2) | -0.44 | -0.47 | -0.42 |

Para moderar el grado de informatividad de las previas para estos parámetros, cada una de ellas se definió como una distribución normal con media igual al estimador puntual del GLM y desvío estándar igual a 5 veces la distancia entre el estimador y el límite inferior del intervalo de confianza de 95 % del mismo (percentil del 2.5 %).

## Kernel de dispersión

El modelo utilizado para el kernel de dispersión -la función presentada previamente- es el siguiente:

Donde es la distancia entre celdas y es un parámetro a ajustar con los datos.

## Datos

#setwd('./data')  
  
#### Abrahamovich y Diaz ####  
  
diaz<-read.csv('./data/external sources/diaz.txt', sep='\t')  
diaz$sp<-'Bombus dahlbomii'  
diaz$db='diaz'  
  
#### morales ####  
mor<-read.csv('./data/external sources/Morales\_data.txt', sep='\t')  
  
bd.mor<-subset(mor, B.dahlbomii>=1)  
bd.mor<-bd.mor[,c("Longitude..W...decimal.degrees.",   
 "Latitude..S...decimal.degrees.")]  
bd.mor$year<-2011  
bd.mor$sp<-'Bombus dahlbomii'  
  
br.mor<-subset(mor, B.ruderatus>=1)  
br.mor<-br.mor[,c("Longitude..W...decimal.degrees.",   
 "Latitude..S...decimal.degrees.")]  
br.mor$year<-2011  
br.mor$sp<-'Bombus ruderatus'  
  
bt.mor<-subset(mor, B.terrestris>=1)  
bt.mor<-bt.mor[,c("Longitude..W...decimal.degrees.",   
 "Latitude..S...decimal.degrees.")]  
bt.mor$year<-2011  
bt.mor$sp<-'Bombus terrestris'  
  
morales<-rbind(bd.mor, br.mor, bt.mor)  
colnames(morales)<-c('lon', 'lat', 'year', 'sp')  
morales$db='Morales'  
  
#### idigbio ####  
  
idb<-read.csv('./data/external sources/occurrence\_raw\_idigbio.csv')  
idb2<-data.frame(lon=idb$dwc.decimalLongitude,  
 lat=idb$dwc.decimalLatitude,  
 year=idb$dwc.year,   
 sp=paste(idb$dwc.genus, idb$dwc.specificEpithet),   
 db='idb')  
  
#### SCAN ####  
  
scan<-read.csv('./data/external sources/occurrences\_scan.csv')  
scan2<-data.frame(lon=scan$decimalLongitude,  
 lat=scan$decimalLatitude,  
 year=scan$year,   
 sp=paste(scan$genus, scan$specificEpithet),   
 db='scan')  
scan2<-subset(scan2, sp!=' ')  
  
  
  
#### GBIF ####  
#setwd('./external sources/gbif manual/')  
  
# Bombus dahlbomii  
# DOI 10.15468/dl.d4neky  
bd0<-read.csv('./data/external sources/gbif manual/dahlbomii/0316392-200613084148143.csv',   
 sep='\t')  
#bd0<-read.csv('./dahlbomii/0138260-200613084148143.csv', sep='\t')  
  
bd<-data.frame(lon=bd0$decimalLongitude,   
 lat=bd0$decimalLatitude,   
 year=bd0$year,   
 sp='Bombus dahlbomii')  
  
bd<-bd[complete.cases(bd),]  
  
  
# Bombus terrestris  
# DOI 10.15468/dl.ymkhwc  
  
bt0<-read.csv('./data/external sources/gbif manual/terrestris/0316394-200613084148143.csv',   
 sep='\t')  
  
bt<-data.frame(lon=bt0$decimalLongitude,   
 lat=bt0$decimalLatitude,   
 year=bt0$year,   
 sp='Bombus terrestris')  
bt<-bt[complete.cases(bt),]  
  
# Bombus ruderatus  
# DOI 10.15468/dl.urxqrj  
  
br0<-read.csv('./data/external sources/gbif manual/ruderatus/0316281-200613084148143.csv',   
 sep='\t')  
  
br<-data.frame(lon=br0$decimalLongitude,   
 lat=br0$decimalLatitude,   
 year=br0$year,   
 sp='Bombus ruderatus')  
br<-br[complete.cases(br),]  
  
  
  
# three species  
gbif<-rbind(bd, bt, br)  
gbif$db='gbif'  
  
  
#### schmid ####  
  
sch=read.csv('./data/external sources/schmith hempel 2014/sp.three.csv',   
 sep='\t')  
  
sch=sch[sch$n>=1, c(3, 2, 1, 5)]  
sch[,1:2]=sch[,1:2]\*(-1)  
sch$db='sch'  
  
colnames(sch)=colnames(diaz)  
  
# add introduction site from scmith hempel (Quillota, centarl Chile)  
sch=rbind(sch, list(lon=-71.14, lat=-32.52, year=1998,   
 sp='Bombus terrestris', db='sch'))  
  
  
  
#sch=sch[rep(row.names(sch), sch$n),c(3, 2, 1, 5)]  
#rownames(sch)=NULL  
  
  
  
#### all ####  
  
bind<-rbind(diaz, morales, idb2, scan2, gbif, sch)  
  
oc.pat.plot=ggplot(bind)+  
 geom\_point(aes(lon, lat), alpha=0.3)+  
 geom\_vline(xintercept = c(-60, -76))+  
 geom\_point(data=bind[bind$lon>=68&bind$lon<=76,],   
 aes(lon, lat), alpha=0.2, col='red')+  
 coord\_equal()  
  
# espejar las antipodas  
bind[bind$lon>=68&bind$lon<=76,c('lon', 'lat')]=  
 (bind[bind$lon>=68&bind$lon<=76,c('lon', 'lat')])\*(-1)  
  
# filtrar todo el resto (europa, oceania, etc..)  
bind=bind[bind$lon<=-60&bind$lon>=-76,]  
  
## Highly probably false ocurrences   
lat.north.cut=-25  
  
n.removed=nrow(bind[!bind$lat<=lat.north.cut,])  
  
bind=bind[bind$lat<=lat.north.cut,]  
  
bind<-unique(bind) # remove duplications  
  
#bind=unique(bind[,1:4])  
  
cnt.raw=count(bind, db, sp)  
  
write.csv(row.names = F, bind, './bind\_mor.diaz.scan.idb.gbif.csv')  
  
  
#setwd('../')

## importation and formatting ####  
#all=read.csv("./data/bind\_mor.diaz.scan.idb.gbif.csv")  
  
all=bind  
  
# dataframe formatting  
oc=all[all$sp=='Bombus terrestris'&  
 !is.na(all$year),]  
  
# introduction data  
int.site=c('Copiapó', 'Ovalle', 'San Felipe',   
 'Quillota', 'Limache', 'Santiago', 'Los Ángeles')  
int.lat=c(-27.2, -30.36, -32.45, -32.5,   
 -32.59, -33.27, -37.28)  
int.long=c(-70.19, -71.12, -70.43, -71.1,   
 -71.17, -70.40, -72.21)  
  
int.df=data.frame(lon=int.long,   
 lat=int.lat,   
 year=1997,  
 sp='Bombus terrestris',   
 db='Montalva\_2011',  
 site=int.site)  
  
  
oc=rbind(oc, int.df[,1:5])  
  
ll=oc[,c('lon', 'lat')]  
ll=SpatialPoints(ll)  
  
# geo projection  
# set starting long-lat crs  
crs(ll)=CRS("+proj=longlat +datum=WGS84")  
  
# new CRS  
#nproj="+proj=utm +zone=19H +datum=WGS84"  
  
ll.proj=spTransform(ll, CRS(nproj))  
  
ll.proj.df=as.data.frame(ll.proj)  
   
  
bt=cbind(ll.proj.df, year=oc[,3])  
  
#mp=bt[,c('lon', 'lat', 'year')]

Los datos utilizados para ajustar el modelo de dispersión consistieron en ocurrencias de *B. terrestris* que contaran con información de sus coordenadas y fecha de registro, y los valores de las componentes principales 1, 2 y 3 en esas coordenadas (*después daré mas detalles sobre estos datos y sobre su obtención*). 0 ocurrencias fueron removidas del análisis ya que se encontraban en latitudes anómalamente altas (al norte de -25 grados). Con el objetivo de representar de la manera más fidedigna posible las distancias geográficas entre las coordenadas de los registros se realizó, para todas las capas de información (tanto ocurrencias como variables climáticas), una proyección de Mercator Transversal (UTM, centrada en la zona 19) para pasar del sistema de referencia polar longitud-latitud a uno euclideo.

mp=bt[,c('lon', 'lat', 'year')]  
  
y=as.factor(mp$year)  
  
cell.size=50000 # meters  
x.n.cells=round(abs(max(mp$lon)-min(mp$lon))/cell.size)  
y.n.cells=round(abs(max(mp$lat)-min(mp$lat))/cell.size)  
  
  
r=raster(ncol=x.n.cells, nrow=y.n.cells, ext=extent(ll.proj))  
r[]=1:length(r)  
  
rp=rasterize(x=as.matrix(mp[,1:2]),   
 y=r)  
rp[rp>=1]=1  
  
# rasterizacion por anio  
s=list()  
for (i in 1:length(levels(y))) {  
 #print(i)  
 mp.i=mp[mp$year<=as.integer(levels(y)[i]),]  
   
 rp.i=rasterize(x=as.matrix(mp.i[,1:2]),   
 y=r)  
 rp.i[rp.i>=1]=1  
   
 s[[i]]=rp.i  
}  
s=stack(s)  
names(s)=levels(y)%>%as.character()  
  
  
# p/c/celda se queda con el 1er anio con registros  
C=c()  
#yl=1:nlayers(s)  
yl=levels(y)%>%as.character()%>%as.integer()  
  
for (i in 1:length(rp)) {  
 #print(i)  
 c.i=s[[1:nlayers(s)]][i]  
 if (all(is.na(c.i))) {  
 C[i]=NA  
 }else{  
 C[i]=yl[!is.na(c.i)]%>%min()  
 }  
}  
  
rp.y=rp  
rp.y[]=C  
  
## data visualization Figure A ####  
  
points=coordinates(rp.y)%>%as.data.frame()  
points$Year=rp.y[]  
points$Year=as.factor(points$Year)  
points=points[points$y<=extent(pc.latam.patag)[4],]  
  
ras.plot=rp.y#%>%crop(y=extent(pc.latam.patag))  
ras.plot[]=raster::extract(prior.suit, coordinates(ras.plot))  
  
arg=raster::getData("GADM",country=c("Argentina"),level=0)  
chl=raster::getData("GADM",country=c("Chile"),level=0)  
  
arg.p=spTransform(arg, crs(nproj))%>%  
 raster::crop(y=extent(ll.proj))%>%fortify()  
  
chl.p=spTransform(chl, crs(nproj))%>%gSimplify(tol=1000)%>%  
 raster::crop(y=extent(ll.proj))%>%fortify()  
  
  
  
cell.coor=coordinates(rp.y)  
  
df=data.frame(i=1:length(C), Ti=C,   
 xi=cell.coor[,1], yi=cell.coor[,2])  
df$c=NA  
df$c[is.na(df$Ti)]=0  
df$c[!is.na(df$Ti)]=1  
  
### add enviromental variables ####  
cov.expl.pcs=raster::extract(pc.latam.patag[[1:3]], df[,c('xi', 'yi')])  
  
# biocliamtic not used, only PCs  
#cov.expl.bio=raster::extract(expl.bio.pat.all[[c(1, 12)]], df[,c('xi', 'yi')])  
  
df=cbind(df, cov.expl.pcs)  
  
df=df[!is.na(df[,6:ncol(df)])%>%  
 as.matrix()%>%  
 rowSums()>=1, ]

#data.plot=  
p1=ggplot()+  
 geom\_raster(data=rp.y, aes(x, y, fill=layer), alpha=0.9)+  
 #theme(legend.position = "top", legend.direction = 'horizontal')+   
 coord\_equal()+  
 #scale\_fill\_viridis(option = 'A', na.value = "white")+   
 scale\_fill\_distiller(palette = 'YlOrRd', na.value = 'white')+  
 theme\_void()+  
 geom\_point(data=points[!is.na(points$Year),], aes(x, y), size=3, col='black', pch=0)+  
 labs(fill = "Year of first  
detection")+  
 annotation\_scale()+  
 geom\_polygon(data=arg.p, aes(long, lat, group=group), alpha=0, col='black')+  
 geom\_polygon(data=chl.p, aes(long, lat, group=group), alpha=0, col='black')  
  
print(p1)



En el gráfico se muestran los valores de suitabiidad *a priori* en forma de raster para ilustrar la escala de las celdas. Las celdas con puntos son aquellas en las que se registraron ocurrencias, y el color de los mismos indica la fecha más temprana de registro.

## Parametrización

Para estimar todos los parámetros se programó un algoritmo de Markov Chain Monte Carlo (MCMC) que recorra el espacio multidimensional de parámetros, generando de esta forma una muestra de la distribución posterior conjunta (multivariada). Este algoritmo propone, en cada iteración de la cadena, un nuevo punto en este espacio multivariado y calcula un valor proporcional a la probabilidad posterior conjunta en el mismo, multiplicando el valor de la previa por el likelihood. Es un valor proporcional, y no absoluto, porque no se calcula la probabilidad total de los datos (ya que esto requeriría integrar el likelihood a lo largo de todo el espacio de parámetros, lo que en este caso es computacionalmente inviable). Sin embargo, esto no significa un problema, ya que lo que interesa es el cociente entre este valor calculado en el nuevo punto propuesto y el valor en el punto inmediatamente anterior en la cadena. Este cociente se utiliza para evaluar un algoritmo de aceptación-rechazo del tipo *Metropolis-Hastings*. De esta forma, cuando el número de iteraciones es grande, el conjunto de puntos de la cadena se aproxima muy bien a la distribución posterior conjunta.

# functions definition ####  
  
likelihood.func=function(data, pars, cov.names){  
   
 # list of parameters of climatic covariables  
 cov.pars=pars$cov.pars  
   
 ## distances matrix  
 data$i=1:nrow(data)  
   
 d=dist(data[,c('xi', 'yi')], upper = T)%>%  
 as.matrix()  
   
 # splitting colonized cells from un-colonized  
 data\_1=data[data$c==1,]  
   
 data\_0=data[data$c==0,]  
   
 # distances matrix of colonized cells  
 d1=dist(data\_1[,c('xi', 'yi')], upper = T)%>%  
 as.matrix()  
   
   
 # environmental suitability computation  
 # each row corresponds to a single cell  
   
 ## matrix of linear and quadratic coefficients  
 betas=cov.pars[2:length(cov.pars)]%>%  
 unlist()%>%  
 rep(nrow(data))%>%  
 matrix(byrow = T, nrow=nrow(data))  
   
 ## matrix of environmental variables  
 Xs=data[,rep(cov.names, rep(length(terms), length(cov.names)))]%>%  
 as.matrix()  
   
 ## matriz de exponentes a los cuales elevar las variable ambientales  
 ## terms es un vector c(1, 2), por lineal y cuadratico  
 ## tomado del scope general (asi funciona... no tocar)  
 exponents=rep(terms, length(cov.names))%>%  
 rep(nrow(data))%>%  
 matrix(byrow = T, nrow=nrow(data))  
   
 ## suitability components matrix (each linear and quadratic contribution)  
 suit.mat=betas\*(Xs^exponents)  
   
 ## vector of integrated linear predictions (one elemnt per cell)  
 suit.vec.linear=rowSums(suit.mat)+pars$cov.pars$b0  
   
 ## vector of integrated suitabilities (one elemnt per cell)  
 suit.vec.invlogit=invlogit(suit.vec.linear)  
   
   
   
   
 # matriz de componente espacial de flujos entre celdas dado un valor de gamma  
 ## corresponde a la imagen de f() en la formula del texto principal  
 ## aca phi es usado solo como la componente de dispersion (no es equivalente al phi del texto)  
   
 phi.mat.1 = d1^(-2) \* exp(pars$gamma)  
   
 # computes for all cells the phi value but only for propagules coming from colonized cells   
 #(computation economy)  
 phi.mat.all = matrix(NA, ncol=ncol(d), nrow=nrow(d))  
 phi.mat.all[,data$c==1] = d[,data$c==1]^(-2) \* exp(pars$gamma)  
   
   
 # list of likelihoods for all cells i \in C (ie. colonized group)  
 Lik\_vec\_1=c()  
   
 for (i in 1:nrow(data\_1)) {  
   
 # vector filtro (boolean) de celdas colonizadas antes que la celda i  
 J=data\_1$Ti<data\_1$Ti[i]  
   
 d.i=d1[i,J] # distances vector (row of dist matrix)  
   
 Ts=data\_1[J, c('Ti')]  
   
 T.i=data\_1[i, c('Ti')]-Ts # vector de deltas T  
   
 # computation of rate phi in cell i  
 phi.i\_d=phi.mat.1[i, J] # dispersion component  
 phi.i=phi.i\_d \* suit.vec.invlogit[data\_1$i[i]] # both components product  
   
   
 # likelihoods for all contributing cells j (given a Poisson process)  
 Li=sum(phi.i)\*exp(-1\*sum(phi.i\*T.i))   
   
 Lik\_vec\_1[i]=Li  
 }  
   
   
 Lik\_vec\_1[Lik\_vec\_1<=0]=1e-100 # to avoid log() errors  
   
 joint\_log.lik\_1=sum(log(Lik\_vec\_1), na.rm = T)  
   
   
 Lik\_vec\_0=c()  
   
 for (i in 1:nrow(data\_0)) {  
 #print(i)  
   
 J=data$c==1  
   
 row.index.0=data\_0$i[i]  
   
 d.i=d[row.index.0,J] # distances vector  
   
 T.i=max(data$Ti, na.rm = T)-data[J, c('Ti')] # vector de deltas T  
   
 #suit.i=data\_0[i, 'suit']  
   
 # computation of rate phi in cell i  
   
 phi.i\_d=phi.mat.all[row.index.0, J]  
   
 # debug  
 if (all(phi.i\_d == phi.i\_d)) {  
 #print('ok')  
 }else{  
 #print(matrix(c(phi.i\_d, phi.i\_d2), ncol=2))  
 }  
   
   
 phi.i=phi.i\_d \* suit.vec.invlogit[data\_0$i[i]]   
   
   
 Li= exp(-1\*sum(phi.i\*T.i))   
   
 Lik\_vec\_0[i]=Li  
   
 }  
   
 Lik\_vec\_0[Lik\_vec\_0<=0]=1e-100 # to avoid log() errors  
   
   
   
 joint\_log.lik\_0=sum(log(Lik\_vec\_0), na.rm = T)  
   
 joint\_1n0\_log.lik = joint\_log.lik\_1 + joint\_log.lik\_0  
   
 return(joint\_1n0\_log.lik)  
}  
  
  
  
# obsolete, not used (uses only colonized cells)  
likelihood.func\_col.only=function(data, pars, cov.names){  
   
 cov.pars=pars$cov.pars  
   
 ## distances matrix  
 data$i=1:nrow(data)  
   
 #print('sd')  
 d=dist(data[,c('xi', 'yi')], upper = T)  
 #print(d)  
   
 d=as.matrix(d)  
   
 # splitting colonized cells from uncolonized  
 data\_1=data[data$c==1,]  
   
 data\_0=data[data$c==0,]  
   
 # ditances matrix of colonized cells  
   
 d1=dist(data\_1[,c('xi', 'yi')], upper = T)%>%as.matrix()  
   
 ## colonization functions  
   
 # colonization rate from a cell at a distance d (vectorization is compatible)  
 # obsolete function  
 fd=function(d, l){ # f(d\_ij)  
 y=l^2\*d^(-2\*l) # power law dispersal kernel  
 return(y)  
 }  
   
 # colonizability of a cell given covariables values  
 # obsolete function  
 fa=function(x, cov.names, cov.pars){  
 y\_=c()  
   
 # non intercept parameters  
 non.int.pars=cov.pars[2:length(cov.pars)]%>%  
 unlist()  
   
 env=x[,cov.names]  
 env=rep(env, rep(length(terms), length(env)))%>%unlist()  
   
 exponents=rep(terms, length(cov.names))  
   
 y\_=non.int.pars\*(env^exponents)  
   
 y=sum(y\_)+cov.pars$b0  
 out=arm::invlogit(y)  
 return(out)  
 }  
   
   
 # matrix aproach for enviromental suitability computation  
   
 betas=cov.pars[2:length(cov.pars)]%>%  
 unlist()%>%  
 rep(nrow(data))%>%  
 matrix(byrow = T, nrow=nrow(data))  
   
 Xs=data[,rep(cov.names, rep(length(terms), length(cov.names)))]%>%  
 as.matrix()  
   
 exponents=rep(terms, length(cov.names))%>%  
 rep(nrow(data))%>%  
 matrix(byrow = T, nrow=nrow(data))  
   
 suit.mat=betas\*(Xs^exponents)  
   
 suit.vec.linear=rowSums(suit.mat)+pars$cov.pars$b0  
   
 suit.vec.invlogit=invlogit(suit.vec.linear)  
   
   
   
 # matrix aproach for colonization rates computation  
   
 phi.mat.1 = d1^(-2) \* exp(pars$gamma)  
   
 # computes for all cells the phi value but only from colonized cells   
 #(computation economy)  
 phi.mat.all = matrix(NA, ncol=ncol(d), nrow=nrow(d))  
 phi.mat.all[,data$c==1] = d[,data$c==1]^(-2) \* exp(pars$gamma)  
   
   
 # list of likelihoods for all cells i \in C  
 Lik\_vec\_1=c()  
   
 for (i in 1:nrow(data\_1)) {  
   
 J=data\_1$Ti<data\_1$Ti[i]  
   
 d.i=d1[i,J] # distances vector (row of dist matrix)  
   
 Ts=data\_1[J, c('Ti')]  
   
 T.i=data\_1[i, c('Ti')]-Ts # vector de deltas T  
   
 #suit.i=data\_1[i, 'suit']  
   
 #if (is.na(suit.i)) {message('na suit')}  
   
 # computation of rate phi in cell i  
   
   
 phi.i\_d=phi.mat.1[i, J]  
   
   
 phi.i=phi.i\_d \* suit.vec.invlogit[data\_1$i[i]] # \* phi.i\_a   
   
 # debug  
 if (any(is.na(phi.i))) { message('na phi.i') }  
   
 Li=sum(phi.i)\*exp(-1\*sum(phi.i\*T.i)) # likelihoods for all cell j (assuming Poisson process)  
   
 Lik\_vec\_1[i]=Li  
 }  
   
 Lik\_vec\_1[Lik\_vec\_1<=0]=1e-100  
   
 joint\_log.lik\_1=sum(log(Lik\_vec\_1), na.rm = T)  
   
 joint\_1n0\_log.lik = joint\_log.lik\_1  
   
 return(joint\_1n0\_log.lik)  
}  
  
# mcmc chains  
mc=function(data, init.pars, sd.jump, cov.names,  
 likelihood.func, niter,   
 priors.func, prior.means, prior.st.errs,   
 multivar){  
   
 parnames=unlist(init.pars)%>%names()  
   
 prior.func.vec=unlist(priors.func)  
 prior.means.vec=unlist(prior.means)  
 prior.st.errs.vec=unlist(prior.st.errs)  
   
 joint.prior=function(pars.vec, prior.func.vec,   
 prior.means.vec=prior.means.vec,   
 prior.st.errs.vec=prior.st.errs.vec){  
 # vector of priors for each parameter value  
 p.vec=c()  
   
 pars.vec=unlist(pars.vec)  
   
 for (i in 1:length(parnames)) {  
   
 f=prior.func.vec[[parnames[i]]]  
 p.vec[i]=f(x=pars.vec[[parnames[i]]],   
 mean=prior.means.vec[[parnames[i]]],   
 se=prior.st.errs.vec[[parnames[i]]])  
 }  
   
 # joint posterior (sum since log values are returned from prior functions)  
 jp=sum(p.vec)  
 return(jp)  
 }  
   
 # empty matrix to fill with parameters sample  
 sims=matrix(NA, ncol = length(parnames), nrow = niter)  
 colnames(sims)=parnames  
   
 # empty vector to fill with computed posteriors   
 # (not actually posteriors since its not divided by data's probability)  
 pos=numeric(niter)  
   
 # set initial values of parameters  
 sims[1,]=unlist(init.pars)  
   
 # compute initial posterior  
 # likelihood  
 lik.1=likelihood.func(pars=init.pars, data=data, cov.names = cov.names)  
   
 # joint prior  
 prior.1=joint.prior(pars.vec=init.pars, prior.func.vec = prior.func.vec,   
 prior.means.vec=prior.means.vec,   
 prior.st.errs.vec=prior.st.errs.vec)  
   
 # joint posterior (sum since log values are used for both likelihoods and priors)  
 pos[1]=lik.1+prior.1  
   
 # vector of jump standar deviations for each parameter  
 sd.jump.vec=unlist(sd.jump)  
   
 for (i in 2:niter) {  
 message(paste0('iter ', i))  
   
 # first set parameter values and posterior of the iteration to i-1 values  
 sims[i,]=sims[i-1,]  
 pos[i]=pos[i-1]  
   
 # then run accept-reject algorithm switching one parameter at the time   
 # (one dimentional jumping)  
   
 new.par.set=sims[i-1,]  
 for (j in 1:length(parnames)) {  
   
 # choose a new value for parameter j  
 new.par.j=rnorm(1, sims[i-1, j], sd = sd.jump.vec[j])  
   
 # define the new parameter set  
 #new.par.set=sims[i-1,]  
 new.par.set[j]=new.par.j  
   
 new.pars=relist(new.par.set, skeleton = init.pars)  
   
   
 if (multivar) {  
 if (j==length(parnames)) { # multivariate jump  
   
 # new likelihood  
 new.lik=likelihood.func(pars=new.pars, data=data, cov.names = cov.names)  
   
 # new prior  
 new.prior=joint.prior(pars.vec = new.pars,   
 prior.func.vec = prior.func.vec,   
 prior.means.vec=prior.means.vec,   
 prior.st.errs.vec=prior.st.errs.vec)  
   
 # evaluate the posterior  
 new.pos=new.lik+new.prior  
 #print(new.pos)  
   
 # probability of jumping to new value  
 prob.jump=min(1, exp(new.pos-pos[i-1]))  
 #print(prob.jump)  
   
   
 jump=rbinom(1, 1, prob.jump)  
 #print(jump)  
   
 # execute the jump  
 if (jump==1) {  
 sims[i,]=new.par.set  
 pos[i]=new.pos  
 }  
 }  
 }else{ # jumps in each parameter iteratively  
   
 # new likelihood  
 new.lik=likelihood.func(pars=new.pars, data=data, cov.names = cov.names)  
   
 # new prior  
 new.prior=joint.prior(pars.vec = new.pars,   
 prior.func.vec = prior.func.vec,   
 prior.means.vec=prior.means.vec,   
 prior.st.errs.vec=prior.st.errs.vec)  
   
 # evaluate the posterior  
 new.pos=new.lik+new.prior  
 #print(new.pos)  
   
 # probability of jumping to new value  
 prob.jump=min(1, exp(new.pos-pos[i-1]))  
 #print(prob.jump)  
   
   
 jump=rbinom(1, 1, prob.jump)  
 #print(jump)  
   
 # execute the jump  
 if (jump==1) {  
 sims[i, j]=new.par.j  
 pos[i]=new.pos  
 }  
 }  
 }  
 }  
 # define output  
 #samples.df=as.data.frame(sims)  
   
 ls.out=list(posteriors=pos, samples=as.data.frame(sims))  
 return(ls.out)  
}

## init values ####  
  
cov.names=c('PC\_1', 'PC\_2', 'PC\_3')  
  
## terms of polinomic coefficients (3 = quadratic)  
order=2  
  
terms=1:order  
  
## list of params by covariable  
# te values will be modified when running chains  
cov.pars=list()  
for (i in 1:length(cov.names)) {  
 cov.pars.i=list()  
   
 for (j in 1:length(terms)) {  
 cov.pars.i[[j]]=0 # setting init parameters values to 0  
 }  
 names(cov.pars.i)=paste0('b', terms)  
   
 cov.pars[[i]]=cov.pars.i  
}  
names(cov.pars)=cov.names  
cov.pars$b0=0  
cov.pars=cov.pars[c(length(cov.pars), 1:(length(cov.pars)-1))]  
  
lambda=0  
gamma=0  
  
pars=list(gamma, cov.pars)  
names(pars)=c('gamma', 'cov.pars')

## priors ####  
  
  
  
priors=pars  
  
# means and SEs  
prior.means=relist(coef, cov.pars)  
prior.means=list(#lambda=0.5,   
 gamma=0, cov.pars=prior.means)  
  
prior.st.errs=relist(SEs\*sd.prior.factor, cov.pars)  
prior.st.errs=list(#lambda=2,   
 gamma=15, cov.pars=prior.st.errs)  
  
# functions  
  
#priors$lambda=function(x, mean, se){  
# p=dtruncnorm(x=x, a=0, b=Inf, mean = mean, sd=se)  
 #p=1  
# return(log(p))  
#}  
  
priors$gamma=function(x, mean, se){  
 p=dnorm(x=x, mean = mean, sd=se, log = T)  
 #p=1  
 return(p)  
}  
  
priors$cov.pars$b0=function(x, mean, se){  
 p=dnorm(x=x, mean = mean,   
 sd=se, log = T)  
 #p=1  
 return(p)  
}  
  
# priors of cov.pars are defined below  
  
## jumps ####  
  
jumps=pars  
  
jumps$gamma=0.1  
#jumps$lambda=0.01  
jumps$cov.pars$b0=1.7  
# the rest of non-intercept cov.pars were defined all at once with the priors   
  
  
# for the terms  
for (i in 1:length(cov.names)+1) {  
 for (j in 1:length(terms)) {  
   
 #jumps$cov.pars[[i]][j]=0.03  
 jumps$cov.pars[[i]][j]=0.02  
   
 if (j==2) {  
 jumps$cov.pars[[i]][j]=jumps$cov.pars[[i]][[j]]/2  
 }  
   
 pars$cov.pars[[i]][j]=relist(coef, cov.pars)[[i]][j]  
   
 priors$cov.pars[[i]][[j]]=function(x, mean, se){  
   
 p=dnorm(x=x, mean = mean,   
 sd=se, log = T)  
 #p=1  
 return(p)  
 }  
 }  
}  
  
  
jumps$cov.pars$PC\_1$b1=jumps$cov.pars$PC\_1$b1\*9  
jumps$cov.pars$PC\_1$b2=jumps$cov.pars$PC\_1$b2\*2  
jumps$cov.pars$PC\_2$b1=jumps$cov.pars$PC\_2$b1\*4  
jumps$cov.pars$PC\_2$b2=jumps$cov.pars$PC\_2$b2\*2  
jumps$cov.pars$PC\_3$b1=jumps$cov.pars$PC\_3$b1\*3  
jumps$cov.pars$PC\_3$b2=jumps$cov.pars$PC\_3$b2\*3

Niter=30000  
n.chains=3  
  
# min-max runif range to shuffle from central init.pars values (defined above)  
init.shuffle=5  
  
# only colonized cells  
col.only=F  
  
# Numeric index saved locally to identify different runs  
  
run.index=readRDS('./intermediate\_files/run.index.RDS')  
  
# a prior rds file path. To be modified if new MCMC is run  
chain.list.file.path=paste0('./intermediate\_files/chain.list',   
 '\_prior.factor.', sd.prior.factor,  
 '\_pa.factor.', pa.factor,   
 '\_Niter.', Niter,   
 '\_nchains.', n.chains,   
 '\_idx.', run.index,   
 '\_col.only-', as.character(col.only),   
 '.RDS')

# run if eval=T in chunk parameters  
  
  
if (col.only) {  
 lik.f=likelihood.func\_col.only  
}else{  
   
 lik.f=likelihood.func  
}  
  
if (mcmc.exec==T) {  
   
   
   
 chain.ls=list()  
 for (ch in 1:n.chains) {  
   
 # define a new random starting points between -10 and 10  
 pars0=(unlist(pars)+runif(1, -init.shuffle, init.shuffle))%>%  
 relist(skeleton = pars)  
   
 sim.ch=mc(data=df, init.pars = pars0,  
 sd.jump = jumps, likelihood.func = lik.f,  
 priors.func = priors, cov.names = cov.names,   
 prior.means=prior.means, prior.st.errs=prior.st.errs,   
 niter = Niter, multivar=F)  
   
 sam.ch=sim.ch$samples  
 sam.ch$chain=ch  
 sam.ch$iter=1:Niter  
   
 chain.ls[[ch]]=sam.ch  
 }  
   
 run.index=run.index+1  
 saveRDS(run.index, file = './intermediate\_files/run.index.RDS')  
   
 # update rds file path  
 chain.list.file.path=paste0('./intermediate\_files/chain.list',   
 '\_prior.factor.', sd.prior.factor,  
 '\_pa.factor.', pa.factor,   
 '\_Niter.', Niter,   
 '\_nchains.', n.chains,   
 '\_idx.', run.index,   
 '\_col.only-', as.character(col.only),   
 '.RDS')  
   
 # export RDS chain.list  
 saveRDS(chain.ls, file=chain.list.file.path)  
   
}

# import last chain list file  
  
  
#chain.ls=readRDS('./intermediate\_files/chain.list\_20000.iters\_3.chs\_41.RDS')  
chain.ls=readRDS(chain.list.file.path)  
  
# convergence check  
  
burnin=3000  
  
# delete burnin  
iter.chain.vars.out=function(x){  
 x1=x[-(1:burnin),!colnames(x)%in%c('iter', 'chain')]  
 return(x1)  
}  
  
sam.mat=chain.ls[1:length(chain.ls)]%>%  
 lapply(iter.chain.vars.out)%>%  
 lapply(as.matrix)%>%  
 lapply(mcmc)  
  
sam.coda=mcmc.list(sam.mat)  
  
#gelman.plot(sam.coda)  
Neff=effectiveSize(sam.coda)  
rej.rate=rejectionRate(sam.coda)  
#traceplot(sam.coda)  
summ=summary(sam.coda)  
summ.st=summ$statistics%>%as.data.frame()%>%round(2)  
summ.q=summ$quantiles%>%as.data.frame()%>%round(2)  
#autocorr.plot(sam.coda)  
  
summ.mcmc=data.frame(`Prior\_m`=unlist(prior.means),   
 `Prior\_sd`=unlist(prior.st.errs),  
 `Jumps\_sd`=unlist(jumps),  
 `Neff`=round(Neff),  
 `Reject.rate`=round(rej.rate, 3),   
 Post\_m=summ.st$Mean,   
 q\_2.5=summ.q$`2.5%`,   
 q\_97.5=summ.q$`97.5%`)  
if (n.chains>=2) {  
 rhat=gelman.diag(sam.coda)  
 summ.mcmc$Rhat=rhat$psrf[,1]  
}  
  
  
chain.df=do.call('rbind', chain.ls)  
  
  
sam=gather(chain.df, par, value,   
 names(chain.df)[!names(chain.df)%in%c('iter', 'chain')])  
  
chain.file.path=paste0(chain.list.file.path, 'dataframe.csv')  
  
# export chains in tabular format  
write.csv(sam, chain.file.path, row.names = F)

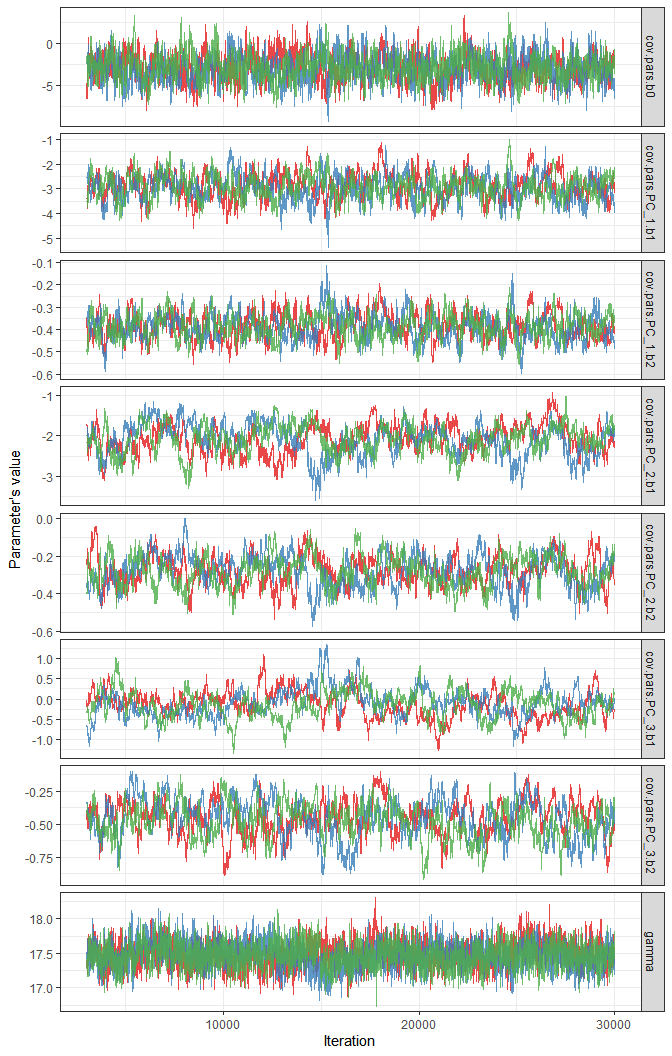
En este caso se corrieron 3 cadenas en paralelo, cada una con una longitud de 30000 iteraciones, descartando las primeras 3000 para no considerar el periodo de acercamiento de las cadenas a su estado estable (*burnin*). Los valores iniciales de cada parámetro en las diferentes cadenas se definieron aleatoriamente con probabilidad uniforme entre -5 y 5. La muestra utilizada corresponde a la unión de las 3 cadenas (descartando de cada una el intervalo de *burnin*).

kable(round(summ.mcmc, 3))

|  | Prior\_m | Prior\_sd | Jumps\_sd | Neff | Reject.rate | Post\_m | q\_2.5 | q\_97.5 | Rhat |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| gamma | 0.000 | 15.000 | 0.10 | 1200 | 0.381 | 17.48 | 17.14 | 17.83 | 1.043 |
| cov.pars.b0 | -11.038 | 2.232 | 1.70 | 737 | 0.725 | -2.90 | -5.83 | 0.39 | 1.005 |
| cov.pars.PC\_1.b1 | -2.572 | 0.813 | 0.18 | 384 | 0.765 | -2.91 | -3.89 | -1.91 | 1.022 |
| cov.pars.PC\_1.b2 | -0.259 | 0.068 | 0.02 | 386 | 0.785 | -0.39 | -0.49 | -0.27 | 1.019 |
| cov.pars.PC\_2.b1 | -2.470 | 0.428 | 0.08 | 181 | 0.792 | -2.08 | -2.85 | -1.43 | 1.114 |
| cov.pars.PC\_2.b2 | -0.404 | 0.071 | 0.02 | 228 | 0.799 | -0.28 | -0.44 | -0.12 | 1.015 |
| cov.pars.PC\_3.b1 | 0.763 | 0.376 | 0.06 | 122 | 0.801 | -0.13 | -0.77 | 0.53 | 1.088 |
| cov.pars.PC\_3.b2 | -0.445 | 0.118 | 0.03 | 198 | 0.804 | -0.46 | -0.75 | -0.21 | 1.059 |

La convergencia de las cadenas se evaluó manualmente mediante el análisis visual de los *trace-plots*:

sam=read.csv(chain.file.path)  
sam$chain=as.factor(sam$chain)  
  
sam.b=sam[sam$iter>burnin,]  
  
chain.plot=ggplot(sam.b)+  
 geom\_line(aes(iter, value, col=chain),   
 alpha=0.8, size=0.05, show.legend = F)+  
 facet\_grid(par~., scales = 'free\_y')+  
 xlab('Iteration')+  
 ylab('Parameter\'s value')+  
 #xlim(c(5000, 5500))+  
 theme\_bw()+  
 scale\_color\_brewer(palette = 'Set1')  
  
print(chain.plot)



sam.w=reshape(sam.b, idvar = c("iter", 'chain'),   
 timevar = "par", direction = "wide")  
  
names=names(sam.w)  
names.c=gsub(pattern = 'value.', replacement = '', x = names)%>%  
 gsub(pattern = 'cov.pars.', replacement = '')  
names(sam.w)=names.c

Evaluación del grado de correlación entre parámetros

En cuanto a las previas utilizadas, estas han sido la principal contribución a las distribuciones posteriores de algunos de los parámetros de los términos lineales y cuadráticos asociados a las variables ambientales (), pero no para todos; así como tampoco para .

prior.dists.plots=list()  
  
  
for (i in 1:length(unlist(pars))) {  
   
 mean=unlist(prior.means)[i]  
 sd=unlist(prior.st.errs)[i]  
 func=unlist(priors)[[i]]  
   
 x=seq(mean-5\*sd, mean+5\*sd, by=sd/1000)  
   
 d=func(x=x, mean=mean, se=sd)%>%exp()  
   
 prior.df.i=data.frame(x=x, d=d, par=names(unlist(pars))[i])  
   
   
 if (names(unlist(pars))[i]=='gamma') {  
 plot=ggplot(prior.df.i)+  
 #geom\_ribbon(aes(x, ymax=d, ymin=0), alpha=0.1)+  
 theme\_bw()+  
 xlab(expression(gamma))+  
 geom\_density(data=sam.w,  
 aes\_string(names(sam.w)[i+2]),   
 fill='grey', alpha=0.6)+  
 geom\_line(aes(x, d), col='red', size=0.7)+  
 ylab('')+  
 xlim(c(15, 20))  
 }else if(F){  
 plot=ggplot(prior.df.i)+  
 #geom\_ribbon(aes(x, ymax=d, ymin=0), alpha=0.1)+  
 theme\_bw()+  
 xlab(names(sam.w)[i+2])+  
 geom\_density(data=sam.w,  
 aes\_string(names(sam.w)[i+2]),   
 fill='grey', alpha=0.6)+  
 geom\_line(aes(x, d), col='red', size=0.7)+  
 xlim(c(0, 1.2))+ylab('')  
 }else{  
 plot=ggplot(prior.df.i)+  
 #geom\_ribbon(aes(x, ymax=d, ymin=0), alpha=0.1)+  
 theme\_bw()+  
 xlab(names(sam.w)[i+2])+  
 geom\_density(data=sam.w,  
 aes\_string(names(sam.w)[i+2]),   
 fill='grey', alpha=0.6)+  
 geom\_line(aes(x, d), col='red', size=0.7)+  
 #xlim(c(5, 20))+  
 ylab('')  
 }  
 prior.dists.plots[[i]]=plot  
}  
  
  
arr=ggarrange(plotlist=prior.dists.plots)  
  
# Annotate the figure by adding a common labels  
prior.post=annotate\_figure(arr, left = text\_grob("Density", rot = 90))  
  
print(prior.post)



Las líneas rojas corresponden a las densidades de probabilidad *a priori*, y las curvas grises son las densidades posteriores.

## Visualización del kernel de dispersión

En un proceso de Poisson, la probabilidad de no observar presencias en una celda dada una tasa de colonización es

donde es la tasa de colonización y es el intervalo de tiempo considerado. Considerando un período de un año, y una suitabilidad ambiental , la probabilidad de colonización desde una celda a una distancia se puede calcular como

Teniendo en cuenta esto, se tomó toda la muestra de la posterior conjunta para calcular la probabilidad de colonización de una celda a otra, luego de un año de dispersión, a a diferentes distancias.

pc1y=function(l=1, gam, d){  
 phi=l^2 \* d^(-2\*l) \*exp(gam)  
   
 p0=exp(-1\*phi)  
 p1=1-p0  
   
 return(p1)  
}  
  
#ker.plot.sample=length(gammas)  
ker.plot.sample=1000  
  
# vectors of distances and parameters  
dists=seq(100, 50000, 100)  
  
#lambdas=sam.w$lambda  
gammas=sam.w$gamma  
  
# matrix of colonization at 1 year probability  
p=matrix(NA, ncol = ker.plot.sample, nrow = length(dists))  
  
  
  
for (i in 1:length(dists)) {  
   
 for (j in 1:ker.plot.sample) {  
   
 f=pc1y(#l=lambdas[j],   
 d=dists[i], gam=gammas[j])  
 p[i,j]=f  
   
 }  
}  
  
# put matrix values in dataframe  
  
col.prob.dist=data.frame(dist=NA,   
 p\_2.5=NA,   
 p\_97.5=NA,   
 p\_median=NA)  
  
cred.int=c(0.01, 0.99)  
cred.int.level=100\*(cred.int[2]-cred.int[1])  
  
for (i in 1:length(dists)) {  
   
 df.dist=data.frame(dist=dists[i],   
 p\_2.5=quantile(p[i,], cred.int[1]),   
 p\_97.5=quantile(p[i,], cred.int[2]),   
 p\_median=quantile(p[i,], 0.5))  
   
 col.prob.dist=rbind(col.prob.dist,   
 df.dist)  
}  
  
  
col.prob.dist=col.prob.dist[-1,]

ker.plot=ggplot(col.prob.dist[col.prob.dist$dist<=50000,])+  
 geom\_line(aes(dist, p\_median))+  
 geom\_ribbon(aes(x = dist, ymin=p\_2.5, ymax=p\_97.5), alpha=0.3)+  
 theme\_bw()+xlab('Distance (m)')+ylab('Probability')#+  
 #ggtitle(TeX("$P = 1-e^{-\\lambda^2d^{-2\\lambda}e^{\\gamma}}$"))+  
 #ggtitle(TeX("$P = 1-e^{-\\frac{e^\\gamma}{d^2}$"))  
  
print(ker.plot)



El área en gris denota el intervalo de credibilidad del 98 %.

p.decay=0.1  
  
gam = mean(sam.w$gamma)  
  
d = sqrt( -1\*exp(gam) / log(1-p.decay) )

La probabilidad media de colonización decae por debajo de 0.1 a partir de 1.9212964^{4} metros de distancia.

## Extrapolación del modelo de suitabilidad a Sudamérica

Con los valores medios de las posteriores de los parámetros asociados a variables climáticas se estimó la suitabilidad actualizada a lo largo de todo el continente sudamericano.

post.suit.latam=mean(sam.w$b0)+  
 mean(sam.w$PC\_1.b1)\*pc.latam.latam$map$PC\_1+  
 mean(sam.w$PC\_1.b2)\*pc.latam.latam$map$PC\_1^2+  
 mean(sam.w$PC\_2.b1)\*pc.latam.latam$map$PC\_2+  
 mean(sam.w$PC\_2.b2)\*pc.latam.latam$map$PC\_2^2+  
 mean(sam.w$PC\_3.b1)\*pc.latam.latam$map$PC\_3+  
 mean(sam.w$PC\_3.b2)\*pc.latam.latam$map$PC\_3^2  
  
post.suit.latam=invlogit(post.suit.latam)  
  
prior.suit.latam=predict(pc.latam.latam$map, prior.niche)%>%invlogit()  
  
latam.suit=list(Prior=prior.suit.latam, Posterior=post.suit.latam)%>%  
 stack()  
  
prior.suit.gg=ggplot(mapping = aes(y=lat, x=lon))+  
 geom\_raster(data=latam.suit$Prior, alpha=0.9,  
 aes(x=x, y=y, fill=Prior))+  
 theme\_void()+  
 scale\_fill\_viridis(option = 'viridis', na.value = "white")+   
 coord\_equal()+  
 labs(fill='Suitability')+  
 labs(fill='Adecuabilidad  
ambiental')+  
 ggtitle('Prior')+  
 theme(legend.position = 'bottom')+  
 annotation\_scale()  
  
post.suit.gg=ggplot(mapping = aes(y=lat, x=lon))+  
 geom\_raster(data=latam.suit$Posterior, alpha=0.9,  
 aes(x=x, y=y, fill=Posterior), show.legend = T)+  
 theme\_void()+  
 scale\_fill\_viridis(option = 'viridis', na.value = "white")+   
 coord\_equal()+  
 labs(fill='')+  
 ggtitle('Posterior')+  
 theme(legend.position = 'bottom')+  
 labs(fill='Adecuabilidad  
ambiental'  
)  
  
gg.suit=ggarrange(plotlist=list(prior.suit.gg, post.suit.gg)  
 #,widths = c(5, 7)+2.5  
 )  
print(gg.suit)



# *Para hacer de acá en adelante*

Mi idea es usar este modelo de dispersión que estimé para hacer las simualciones del objetivo 2 de la tesina (además de que la estimación del kernel de dispersión *B. terrestris* ya me parece algo interesante de por sí). Para esto voy a partir de estados iniciales sencillos sembrando colonias en el centro de chile y con esas series temporales simuladas hacer comparaciones con el estado actual. Probablemente la comparación la haga entre las medias (y/o alguna otra métrica) de PC\_1, 2 y 3 del conjunto de celdas colonizadas en la simulación al tiempo y el conjunto de celdas con registros reales de BT. Esto me permitiría responder a la pregunta del obj2 de si se espera que la evolución del patrón de ocupación de nicho de BT siga evolucionando o si ya está ocupandolo todo. Por otra parte me va a paermitir hacer predicciones en el espacio geográfico, *i.e* predecir el curso de la invasión a futuro. El último gráfico usa solo la media de las posteriores para hacer esa proyección, pero mi idea es en todos los casos hacer las simulaciones con una muestra de la posterior para tener intervalos de certiducmbre para las predicciones de invasibilidad.

# Análisis prospectivo de la invasión de B. terrestris

sim.extent=c(-1.5e6, 1.6e6, -6.5e6, -2.8e06)%>%extent()

# ver tema del mapa esta roto por ahora  
paises=c("argentina", "bolivia", "brazil",  
 "chile", "colombia", "ecuador",  
 "guyana", "paraguay", "peru",  
 "suriname", "uruguay", "venezuela")  
  
latam.map=ne\_countries(country = paises, scale = 'medium')  
crs(latam.map)=lonlat.crs  
  
latam.map=spTransform(latam.map, latam.proj)%>%  
 crop(y=sim.extent)  
#plot(latam.map)  
  
  
sim.extent=extent(latam.map)

## Simulaciones desde 1997 ppcheck

# El condiiconal ppcheck.base no es importante  
# es algo sucio que arme para evitar correr las simulaciones a cada render  
  
ppcheck.base=!run.pp.check.sims  
  
if (ppcheck.base) {  
 n.years=30  
   
 n.sims=100  
   
 sim.cell.size=50000  
   
   
 from.1997=T  
   
 posterior.sample=T  
   
 if (posterior.sample) {  
 post.type.message='una muestra aleatoria'  
 }else{  
 post.type.message='los valores medios'  
 }  
   
 run.simulations=F  
   
}else{  
 n.years=30  
   
 n.sims=100  
   
 sim.cell.size=50000  
   
   
 from.1997=T  
   
 posterior.sample=T  
   
 if (posterior.sample) {  
 post.type.message='una muestra aleatoria'  
 }else{  
 post.type.message='los valores medios'  
 }  
 run.simulations=T  
}

Con una muestra aleatoria de las posteriores de todos los parámetros se realizaron 100 simulaciones de dispersión de *B. terrestris* a partir de un conjunto de celdas colonizadas por la introducción comercial de colonias en un conjunto de coordenadas **cita montalva**. Las simulaciones se realizaron en series de 30 años partiendo de 1997, con celdas de un tamaño de ~50 .

Año a año se calculó, para cada celda, la probabilidad de ser colonizada dado un conjunto de celdas colonizadas en años anteriores y unas condiciones ambientales determinadas en dicha celda. Con este valor de probabilidad se determinó la colonización o no mediante un proceso de Bernoulli. Para proyectar las coordenadas de las celdas a un espacio euclideo y calcular las distancias con el menor error posible se realizó una proyección UTM (+proj=utm +zone=21H +datum=WGS84).

# list of simulations matrices  
sim.list=list()  
  
# template from climatic latam data  
sim.raster.0=pc.latam.latam$map$PC\_1%>%crop(sim.extent)  
sim.raster.0[!is.na(sim.raster.0[])]=0  
  
## new raster with choosed cell.sizes  
  
# number of horizontal and vertical cells given cell size and raster extent  
x.n.cells.sim = abs( sim.extent[1]-sim.extent[2] ) / sim.cell.size  
y.n.cells.sim = abs( sim.extent[3]-sim.extent[4] ) / sim.cell.size  
  
# build raster  
sim.raster.seed=raster(ext=sim.extent,   
 nrows=y.n.cells.sim,   
 ncols=x.n.cells.sim,   
 crs=crs(sim.raster.0)) #  
  
# transfer values from template to new raster  
sim.raster.seed[]=raster::extract(sim.raster.0, coordinates(sim.raster.seed))  
  
# reproject current BT ocurrences to UTM 21 (latam proj)  
  
  
if (from.1997) {  
 start.dist=as.matrix(int.df[,1:2])%>%  
 SpatialPoints()%>%  
 `projection<-`(lonlat.crs)%>%  
 spTransform(latam.proj)  
   
 start.dist=as.matrix(mp[mp$year==1997,1:2])%>%  
 SpatialPoints()%>%  
 `projection<-`(nproj)%>%  
 spTransform(latam.proj)  
   
}else{  
 start.dist=as.matrix(mp[,1:2])%>%  
 SpatialPoints()%>%  
 `projection<-`(nproj)%>%  
 spTransform(latam.proj)  
}  
  
  
# current ocurrences rasterization  
col.seed=rasterize(x=start.dist,   
 y=sim.raster.seed)  
col.seed[col.seed>=1]=1  
col.seed[is.na(col.seed)]=0  
  
## mount current occurrences on seed raster space  
sim.raster.seed=sim.raster.seed+col.seed  
  
# define continent filter  
continent.filter=!is.na(sim.raster.seed[])  
  
all.cells.index=1:length(sim.raster.seed[])  
continent.index=all.cells.index[continent.filter]  
  
# distances matrix  
sim.coor=coordinates(sim.raster.seed)  
  
d=dist(sim.coor[continent.filter,])%>%  
 as.matrix()  
  
# fil simulation matrix with seed data (to preserve NAs)  
sim.mat=matrix(rep(sim.raster.seed[continent.filter], n.years),   
 byrow=T, nrow=n.years)  
  
sim.mat.ls=list()  
  
suit.list=list()  
  
  
# env variables  
sim.env.ls=list()  
  
for (i in 1:nlayers(pc.latam.latam$map)) {  
 sim.env.ls[[i]]=sim.raster.seed  
   
 sim.env.ls[[i]][]=raster::extract(pc.latam.latam$map[[i]],   
 coordinates(sim.raster.seed))  
 #sim.env.ls[[i]][]=vals  
}  
  
sim.env=stack(sim.env.ls)  
names(sim.env)=names(pc.latam.latam$map)

sim.mat.ls.filename.past=paste0('./intermediate\_files/past\_sim.mat.ls\_y.',   
 n.years, '\_n.', n.sims, '\_s.',   
 sim.cell.size/1000, '.kms.RDS')  
  
if (run.simulations) {  
 for (k in 1:n.sims) {  
 message(paste('sim', k))  
   
 if (posterior.sample) {  
   
 # sample one point in MCMC  
 k.chain=sample(1:n.chains, 1)  
 k.iter=sample( (burnin+1):Niter, 1)  
   
 # assembly parameters list  
 pars.df=sam.w  
 rownames(pars.df)=NULL  
   
 sim.pars.vec=pars.df[pars.df$chain==k.chain&  
 pars.df$iter==k.iter,  
 3:10]%>%as.numeric()  
   
 }else{ # eman posterior values  
 sim.pars.vec=as.matrix(pars.df[,3:10])%>%colMeans()  
 }  
   
   
 sim.pars.k=relist(sim.pars.vec, pars)  
   
 ## compute suitability raster layer   
   
 suit.terms=pars$cov.pars  
   
   
 # computes each linear term except intercept  
 for (i in (1:length(cov.names))+1 ) {  
   
 for (j in 1:length(terms)) {  
   
 # suit contribution of i PC and j term  
 suit.ij = (sim.env[[i-1]])^j \* as.numeric(sim.pars.k$cov.pars[[i]][[j]])  
   
 suit.terms[[i]][[j]]=suit.ij  
   
 }  
 }  
 # add intercept  
 suit.terms$b0=sim.pars.k$cov.pars$b0%>%as.numeric()  
   
 suit.terms.un=unlist(suit.terms)  
   
 # integrate terms  
 suit.terms.sum=suit.terms.un[2:7]%>%stack()%>%sum()  
 suit.k=suit.terms.sum+suit.terms$b0  
 suit.k=invlogit(suit.k)  
   
 suit.list[[k]]=suit.k  
   
 sim.mat.k=sim.mat  
   
 for (i in 2:n.years) {  
 message(paste('year', i))  
   
 colonized = (1:ncol(sim.mat.k))[sim.mat.k[i-1,]==1]  
 uncolonized = (1:ncol(sim.mat.k))[sim.mat.k[i-1,]==0]  
   
   
 for (j in 1:ncol(sim.mat.k)) {  
 if ( !is.na(sim.mat.k[i-1,j]) ) {  
   
 if (is.na(sim.mat.k[i,j])) {  
 #print(sim.mat[i,j])  
 }  
   
   
 if (sim.mat.k[i-1,j]==0) {  
   
 # calculates amount of years since donor cells were colonized  
 T\_j=matrix(sim.mat.k[ 1:i-1 , sim.mat.k[i-1,]==1 ], nrow=i-1)%>%  
 colSums(na.rm = F)  
   
 col.times=matrix(sim.mat.k[ 1:i-1 , ], nrow=i-1)%>%  
 colSums(na.rm = F)  
   
 T\_j2=col.times[colonized]  
   
 # flux from each cell to focal cell given gamma and distances  
 flux\_j= exp(sim.pars.k$gamma) / (d[j, colonized]^2)  
   
 # integration across donor cells  
 flux.time\_j=flux\_j\*T\_j  
 phi\_j= sum(flux.time\_j) \* suit.k[continent.index[j]]  
   
 # debug  
 if (is.na(phi\_j)) {  
 #print(c(k, i, j))  
 }  
   
 # colonization probability  
 p0=exp(-1\*phi\_j)  
 p1=1-p0  
 sim.mat[i,j]=rbinom(1, 1, p1)  
   
 }else{  
 sim.mat.k[i,j]=1  
 }  
 }  
 }  
 }  
   
 # store k simaulation amtrix in list  
 sim.mat.ls[[k]]=sim.mat.k  
 }  
   
 saveRDS(object = sim.mat.ls, file=sim.mat.ls.filename.past)  
}  
  
  
  
#saveRDS(object = sim.mat.ls, file='intermediate\_files/sim.mat.ls.2000.RDS')

#save.image()  
  
  
  
sim.mat.ls=readRDS(file=sim.mat.ls.filename.past)  
  
# empty list to fill with each temporal series of rasters (in a RasterStack fmt)  
raster.sim.ls=list()  
  
for (i in 1:n.sims) {  
   
 # empty list to fill with each year colonization state of i simluation  
 raster.sim.i=list()  
   
 for (j in 1:n.years) {  
   
 # template  
 ras=sim.raster.seed  
   
 # fill with values of simulation i at year j  
 ras[continent.index]=sim.mat.ls[[i]][j, ]  
   
 raster.sim.i[[j]]=ras  
 }  
   
 names(raster.sim.i)=paste('Year', 1:n.years)  
   
 # stack and sotre  
 raster.sim.ls[[i]]=stack(raster.sim.i)  
   
}  
  
names(raster.sim.ls)=paste('Sim.', 1:n.sims)  
  
  
#plot(raster.sim.ls$`Sim. 1`)

current\_year=1996  
  
col.y.mat=matrix(NA, ncol=ncol(sim.mat.ls[[1]]),   
 nrow = n.sims)  
  
for (i in 1:n.sims) {  
   
 mat.i=sim.mat.ls[[i]]  
   
 ys=1:n.years  
   
 col.ys=c()  
   
 for (j in 1:ncol(mat.i)) {  
   
   
 # si fue colonizada  
 if ( sum(mat.i[,j])>=1 ) {  
 #print('col')  
   
 # todos los anios con celda colonizada  
 col.ys.vec.j = ys[ mat.i[, j]==1 ]  
 # primer anio de detección en la simulacion  
 col.year.min.j=min(col.ys.vec.j)  
 #print(col.year.min.j)  
 }else{ # si no  
 #print('nocol')  
 col.year.min.j=NA  
 }  
   
 col.ys[j]=col.year.min.j  
   
 }  
 col.y.mat[i, ]=col.ys  
}  
  
col.y.mat=col.y.mat+current\_year  
  
  
col.year.summary=data.frame(mean=1:ncol(col.y.mat),   
 min=NA,   
 max=NA,   
 median=NA,   
 q\_2.5=NA,   
 q\_97.5=NA)  
  
for (i in 1:nrow(col.year.summary)) {  
   
 col.year.summary$mean[i]=mean( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$min[i]=min( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$max[i]=max( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$median[i]=median( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$q\_2.5[i]=quantile( col.y.mat[,i], 0.025, na.rm = T)%>%suppressWarnings()  
 col.year.summary$q\_97.5[i]=quantile( col.y.mat[,i], 0.975, na.rm = T)%>%suppressWarnings()  
   
}  
  
  
col.summary.raster=list()  
  
for (k in 1:ncol(col.year.summary)) {  
 # template  
 summ.ras=sim.raster.seed  
 summ.ras[continent.index]=col.year.summary[,k]  
 #print(class(summ.ras))  
   
 col.summary.raster[[k]]=summ.ras  
}  
  
col.summary.raster=stack(col.summary.raster)  
names(col.summary.raster)=colnames(col.year.summary)  
  
# transformation to dataframe  
col.summary.raster.df=fortify(col.summary.raster)  
  
col.summary.raster.df=reshape::melt.data.frame(col.summary.raster.df,   
 id.vars=1:2)  
  
col.summary.raster.df$value=round(col.summary.raster.df$value)  
  
decades=seq(1995, 2030, 5)  
col.summary.raster.df$dec=cut(col.summary.raster.df$value,   
 decades,   
 labels=paste0('<', decades[-1]))  
  
col.summary.raster.df$value=as.ordered(col.summary.raster.df$value)

Para cada celda con ocurrencias de *Bombus terrestris* se evaluaron los años de colonización registrados en las simulaciones para evaluar el grado de fidelidad del modelo con los datos.

ll.proj=spTransform(ll, CRS(latam.proj))  
  
ll.proj.df=as.data.frame(ll.proj)  
   
  
bt=cbind(ll.proj.df, year=oc[,3])  
  
col.hist=bt[,c('lon', 'lat', 'year')]  
  
#cell.size=sim.cell.size # meters  
#x.n.cells=round(abs(max(col.hist$lon)-min(col.hist$lon))/cell.size)  
#y.n.cells=round(abs(max(col.hist$lat)-min(col.hist$lat))/cell.size)  
  
  
#r=raster(ncol=x.n.cells, nrow=y.n.cells, ext=extent(ll.proj))  
#r[]=1:length(r)  
  
rp=rasterize(x=as.matrix(col.hist[,1:2]),   
 y=col.summary.raster$mean)  
  
rp[rp[]>=1]=1  
  
# rasterizacion por anio  
s=list()  
for (i in 1:length(levels(y))) {  
 #print(i)  
 mp.i=col.hist[col.hist$year<=as.integer(levels(y)[i]),]  
   
 rp.i=rasterize(x=as.matrix(mp.i[,1:2]),   
 y=col.summary.raster$mean)  
 rp.i[rp.i[]>=1]=1  
   
 s[[i]]=rp.i  
}  
s=stack(s)  
names(s)=levels(y)%>%as.character()  
  
  
# p/c/celda se queda con el 1er anio con registros  
C=c()  
#yl=1:nlayers(s)  
yl=levels(y)%>%as.character()%>%as.integer()  
  
for (i in 1:length(rp)) {  
 #print(i)  
 c.i=s[[1:nlayers(s)]][i]  
 if (all(is.na(c.i))) {  
 C[i]=NA  
 }else{  
 C[i]=yl[!is.na(c.i)]%>%min()  
 }  
}  
  
rp.y=rp  
rp.y[]=C  
  
## data visualization Figure A ####  
  
points=coordinates(rp.y)%>%as.data.frame()  
points$Year=rp.y[]  
points$Year=as.factor(points$Year)  
points=points[points$y<=extent(pc.latam.patag)[4],]  
  
  
#current.points=points[,1:2]%>%  
 #SpatialPoints()%>%  
 #`projection<-`(nproj)%>%  
 #spTransform(latam.proj)%>%  
 #as.data.frame()  
  
current.points=points  
  
current.points$year=as.character(points$Year)%>%as.numeric()  
  
inv.years.current=raster::extract(col.summary.raster, current.points[,1:2])%>%  
 as.data.frame()  
inv.years.current$obs.year=current.points$year%>%as.character()%>%as.numeric()  
  
  
sim.year.ppc=pivot\_longer(inv.years.current, cols = colnames(inv.years.current)[1:6])  
  
colnames(sim.year.ppc)=c('obs.year', 'sim.summary', 'sim.year')  
  
pp.plot=ggplot(sim.year.ppc[!sim.year.ppc$sim.summary%in%c('max', 'min', 'mean'  
 ),],   
 aes(obs.year, sim.year))+  
 geom\_point(aes(obs.year, sim.year, col=sim.summary)  
 , show.legend = F  
 ,size=2, alpha=0.3  
 )+  
 facet\_grid(.~sim.summary)+  
 geom\_abline(slope = 1, intercept = 0, col='black')+  
 #geom\_smooth(method = 'lm', col=sim.summary, formula = sim.year~obs.year+sim.summary)+  
 theme\_bw()+  
 xlab('Observed colonization year')+  
 ylab('Colonization year in simulations')+  
 xlim(1996, 1996+n.years)+ylim(1996, 1996+n.years)  
  
print(pp.plot)



ggplot(sim.year.ppc[!sim.year.ppc$sim.summary%in%c('max', 'min', 'mean'  
 ),],   
 aes(obs.year, sim.year))+  
 geom\_point(aes(obs.year, sim.year, pch=sim.summary)  
 #, col=sim.summary  
 , col='grey'  
 , show.legend = T  
 ,size=1.5, alpha=0.9  
 )+  
 geom\_point(data=sim.year.ppc[sim.year.ppc$sim.summary=='median',],   
 aes(obs.year, sim.year)  
 #, col=sim.summary  
 , col='black'  
 , show.legend = F  
 ,size=2, alpha=0.9  
 )+  
 #facet\_grid(.~'sim.summary')+  
 geom\_abline(slope = 1, intercept = 0, col='black')+  
 #geom\_smooth(method = 'lm', col=sim.summary, formula = sim.year~obs.year+sim.summary)+  
 theme\_bw()+  
 xlab('Observed colonization year')+  
 ylab('Colonization year in simulations')+  
 xlim(1996, 1996+n.years)+ylim(1996, 1996+n.years)+  
 labs(pch=NULL)



Este gráfico muestra el año de colonización registrado en las simulaciones (media y percentiles 2.5 y 97.5) contra el año en que se registró la colonización en los datos reales.

## simulaciones a futuro desde 2021

# El condiiconal future.base no es importante  
# es algo sucio que arme para evitar correr las simulaciones a cada render  
  
future.base=!run.future.BT.sims  
  
if (future.base) {  
 n.years=20  
   
 n.sims=300  
   
 sim.cell.size=25000  
   
 from.1997=F  
  
 run.simulations=F  
}else{  
   
 n.years=20  
   
 n.sims=300  
   
 sim.cell.size=25000  
   
 from.1997=F  
  
 run.simulations=T  
}

Con una muestra aleatoria de las posteriores de todos los parámetros se realizaron 300 simulaciones de dispersión de *B. terrestris* a partir de un conjunto de celdas colonizadas registradas hasta 2021. Las simulaciones se realizaron en series de 20 años partiendo de 2021, con celdas de un tamaño de ~25 .

# list of simulations matrices  
sim.list=list()  
  
# template from climatic latam data  
sim.raster.0=pc.latam.latam$map$PC\_1%>%crop(sim.extent)  
sim.raster.0[!is.na(sim.raster.0[])]=0  
  
## new raster with choosed cell.sizes  
  
# number of horizontal and vertical cells given cell size and raster extent  
x.n.cells.sim = abs( sim.extent[1]-sim.extent[2] ) / sim.cell.size  
y.n.cells.sim = abs( sim.extent[3]-sim.extent[4] ) / sim.cell.size  
  
# build raster  
sim.raster.seed=raster(ext=sim.extent,   
 nrows=y.n.cells.sim,   
 ncols=x.n.cells.sim,   
 crs=crs(sim.raster.0)) #  
  
# transfer values from template to new raster  
sim.raster.seed[]=raster::extract(sim.raster.0, coordinates(sim.raster.seed))  
  
# reproject current BT ocurrences to UTM 21 (latam proj)  
  
  
if (from.1997) {  
 start.dist=as.matrix(int.df[,1:2])%>%  
 SpatialPoints()%>%  
 `projection<-`(lonlat.crs)%>%  
 spTransform(latam.proj)  
   
}else{  
 start.dist=as.matrix(mp[,1:2])%>%  
 SpatialPoints()%>%  
 `projection<-`(nproj)%>%  
 spTransform(latam.proj)  
}  
  
sim.env.ls=list()  
  
for (i in 1:nlayers(pc.latam.latam$map)) {  
 sim.env.ls[[i]]=sim.raster.seed  
   
 sim.env.ls[[i]][]=raster::extract(pc.latam.latam$map[[i]],   
 coordinates(sim.raster.seed))  
 #sim.env.ls[[i]][]=vals  
}  
  
sim.env=stack(sim.env.ls)  
names(sim.env)=names(pc.latam.latam$map)  
  
  
# current ocurrences rasterization  
col.seed=rasterize(x=start.dist,   
 y=sim.raster.seed)  
col.seed[col.seed>=1]=1  
col.seed[is.na(col.seed)]=0  
  
## mount current occurrences on seed raster space  
sim.raster.seed=sim.raster.seed+col.seed  
  
# define continent filter  
continent.filter=!is.na(sim.raster.seed[])  
  
all.cells.index=1:length(sim.raster.seed[])  
continent.index=all.cells.index[continent.filter]  
  
# distances matrix  
sim.coor=coordinates(sim.raster.seed)  
  
d=dist(sim.coor[continent.filter,])%>%  
 as.matrix()  
  
# fil simulation matrix with seed data (to preserve NAs)  
sim.mat=matrix(rep(sim.raster.seed[continent.filter], n.years),   
 byrow=T, nrow=n.years)  
  
sim.mat.ls=list()  
  
suit.list=list()

sim.mat.ls.filename.future=paste0('./intermediate\_files/future\_sim.mat.ls\_y.',   
 n.years, '\_n.', n.sims, '\_s.', sim.cell.size/1000, '.kms.RDS')  
  
if (run.simulations) {  
 for (k in 1:n.sims) {  
 message(paste('sim', k))  
   
 if (posterior.sample) {  
   
 # sample one point in MCMC  
 k.chain=sample(1:n.chains, 1)  
 k.iter=sample( (burnin+1):Niter, 1)  
   
 # assembly parameters list  
 pars.df=sam.w  
 rownames(pars.df)=NULL  
   
 sim.pars.vec=pars.df[pars.df$chain==k.chain&  
 pars.df$iter==k.iter,  
 3:10]%>%as.numeric()  
   
 }else{ # eman posterior values  
 sim.pars.vec=as.matrix(pars.df[,3:10])%>%colMeans()  
 }  
   
   
 sim.pars.k=relist(sim.pars.vec, pars)  
   
 ## compute suitability raster layer   
   
 suit.terms=pars$cov.pars  
   
   
 # computes each linear term except intercept  
 for (i in (1:length(cov.names))+1 ) {  
   
 for (j in 1:length(terms)) {  
   
 # suit contribution of i PC and j term  
 suit.ij = (sim.env[[i-1]])^j \* as.numeric(sim.pars.k$cov.pars[[i]][[j]])  
   
 suit.terms[[i]][[j]]=suit.ij  
   
 }  
 }  
 # add intercept  
 suit.terms$b0=sim.pars.k$cov.pars$b0%>%as.numeric()  
   
 suit.terms.un=unlist(suit.terms)  
   
 # integrate terms  
 suit.terms.sum=suit.terms.un[2:7]%>%stack()%>%sum()  
 suit.k=suit.terms.sum+suit.terms$b0  
 suit.k=invlogit(suit.k)  
   
 suit.list[[k]]=suit.k  
   
 sim.mat.k=sim.mat  
   
 for (i in 2:n.years) {  
 message(paste('year', i))  
   
 colonized = (1:ncol(sim.mat.k))[sim.mat.k[i-1,]==1]  
 uncolonized = (1:ncol(sim.mat.k))[sim.mat.k[i-1,]==0]  
   
   
 for (j in 1:ncol(sim.mat.k)) {  
 if ( !is.na(sim.mat.k[i-1,j]) ) {  
   
 if (is.na(sim.mat.k[i,j])) {  
 #print(sim.mat[i,j])  
 }  
   
   
 if (sim.mat.k[i-1,j]==0) {  
   
 # calculates amount of years since donor cells were colonized  
 T\_j=matrix(sim.mat.k[ 1:i-1 , sim.mat.k[i-1,]==1 ], nrow=i-1)%>%  
 colSums(na.rm = F)  
   
 col.times=matrix(sim.mat.k[ 1:i-1 , ], nrow=i-1)%>%  
 colSums(na.rm = F)  
 T\_j2=col.times[colonized]  
   
 # flux from each cell to focal cell given gamma and distances  
 flux\_j= exp(sim.pars.k$gamma) / (d[j, colonized]^2)  
   
 # integrationa across donor cells  
 flux.time\_j=flux\_j\*T\_j  
 phi\_j= sum(flux.time\_j) \* suit.k[continent.index[j]]  
   
 # debug  
 if (is.na(phi\_j)) {  
 #print(c(k, i, j))  
 }  
   
 # colonization probability  
 p0=exp(-1\*phi\_j)  
 p1=1-p0  
 sim.mat[i,j]=rbinom(1, 1, p1)  
   
 }else{  
 sim.mat.k[i,j]=1  
 }  
 }  
 }  
 }  
   
 # store k simaulation amtrix in list  
 sim.mat.ls[[k]]=sim.mat.k  
 }  
  
#save.image()  
  
 saveRDS(object = sim.mat.ls, file=sim.mat.ls.filename.future)  
}  
  
#saveRDS(object = sim.mat.ls, file='intermediate\_files/sim.mat.ls.2000.RDS')

#save.image()  
  
  
  
sim.mat.ls=readRDS(file=sim.mat.ls.filename.future)  
  
# empty list to fill with each temporal series of rasters (in a RasterStack fmt)  
raster.sim.ls=list()  
  
for (i in 1:n.sims) {  
   
 # empty list to fill with each year colonization state of i simluation  
 raster.sim.i=list()  
   
 for (j in 1:n.years) {  
   
 # template  
 ras=sim.raster.seed  
   
 # fill with values of simulation i at year j  
 ras[continent.index]=sim.mat.ls[[i]][j, ]  
   
 raster.sim.i[[j]]=ras  
 }  
   
 names(raster.sim.i)=paste('Year', 1:n.years)  
   
 # stack and sotre  
 raster.sim.ls[[i]]=stack(raster.sim.i)  
   
}  
  
names(raster.sim.ls)=paste('Sim.', 1:n.sims)  
  
  
#plot(raster.sim.ls$`Sim. 1`)

current\_year=max(oc$year, na.rm = T)-1  
  
col.y.mat=matrix(NA, ncol=ncol(sim.mat.ls[[1]]),   
 nrow = n.sims)  
  
for (i in 1:n.sims) {  
   
 mat.i=sim.mat.ls[[i]]  
   
 ys=1:n.years  
   
 col.ys=c()  
   
 for (j in 1:ncol(mat.i)) {  
   
   
 # si fue colonizada  
 if ( sum(mat.i[,j])>=1 ) {  
 #print('col')  
   
 # todos los anios con celda colonizada  
 col.ys.vec.j = ys[ mat.i[, j]==1 ]  
 # primer anio de detección en la simulacion  
 col.year.min.j=min(col.ys.vec.j)  
 #print(col.year.min.j)  
 }else{ # si no  
 #print('nocol')  
 col.year.min.j=Inf  
 }  
   
 col.ys[j]=col.year.min.j  
   
 }  
 col.y.mat[i, ]=col.ys  
}  
  
col.y.mat=col.y.mat+current\_year  
  
  
col.year.summary=data.frame(mean=1:ncol(col.y.mat),   
 min=NA,   
 max=NA,   
 median=NA,   
 q\_2.5=NA,   
 q\_97.5=NA)  
  
for (i in 1:nrow(col.year.summary)) {  
   
 col.year.summary$mean[i]=mean( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$min[i]=min( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$max[i]=max( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$median[i]=median( col.y.mat[,i], na.rm = T)%>%suppressWarnings()  
 col.year.summary$q\_2.5[i]=quantile( col.y.mat[,i], 0.025, na.rm = T)%>%suppressWarnings()  
 col.year.summary$q\_97.5[i]=quantile( col.y.mat[,i], 0.975, na.rm = T)%>%suppressWarnings()  
   
}  
  
  
col.summary.raster=list()  
  
for (k in 1:ncol(col.year.summary)) {  
 # template  
 summ.ras=sim.raster.seed  
 summ.ras[continent.index]=col.year.summary[,k]  
 #print(class(summ.ras))  
   
 col.summary.raster[[k]]=summ.ras  
}  
  
col.summary.raster=stack(col.summary.raster)  
names(col.summary.raster)=colnames(col.year.summary)  
  
# transformation to dataframe  
col.summary.raster.df=fortify(col.summary.raster)  
  
col.summary.raster.df=reshape::melt.data.frame(col.summary.raster.df,   
 id.vars=1:2)  
  
col.summary.raster.df$value=round(col.summary.raster.df$value)  
  
decades=seq(2020, 2100, 5)  
col.summary.raster.df$dec=cut(col.summary.raster.df$value,   
 decades,   
 labels=paste0('<', decades[-1]))  
  
col.summary.raster.df$value=as.ordered(col.summary.raster.df$value)

#col.summary.raster.df[col.summary.raster.df==Inf]=NA  
  
col.summary.raster.df$variable=relevel(col.summary.raster.df$variable, 'q\_2.5')  
  
formatter1000 <- function(){  
 function(x)x/1000  
}  
  
  
col.maps=ggplot(col.summary.raster.df[!col.summary.raster.df$variable%in%  
 c('mean',   
 'min', 'max'),])+  
 geom\_raster(aes(x, y, fill=dec))+   
 #scale\_fill\_brewer(palette = "OrRd")+  
 #scale\_fill\_continuous(guide = guide\_colourbar(direction = "horizontal",nbin = 30))+  
 facet\_grid(.~variable)+  
 #scale\_fill\_viridis()+  
 #theme(legend.position = 'bottom')+  
 #scale\_fill\_viridis\_d('Year', direction = -1,option = 'D')+   
 scale\_fill\_brewer(palette = 'Spectral', na.value = 'white')+  
 geom\_polygon(data=latam.map,   
 aes(long, lat, group=group),  
 alpha=0, col='black')+ # Mapa  
 theme\_classic()+ # estilo del gráfico (que no haya grilla por ej.)  
 coord\_equal()+ # que no deforme horizontal ni verticalmente las ecalas  
 xlab('Longitude in km')+  
 ylab('Latitude in km')+  
 #xlim(c(-2500000, 3300000))+  
 #scale\_x\_continuous(labels=function(x)x/1000, n.breaks = 5)+  
 #theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))+  
 #ylim(c(-6500000, sim.extent[4]))+  
 #scale\_y\_continuous(labels=function(x)x/1000, n.breaks = 5)+  
 #ggtitle('Year of colonization')+  
 #annotation\_scale()+  
 theme(axis.text.x = element\_text(angle = 90))+  
 scale\_y\_continuous(labels=formatter1000())+  
 scale\_x\_continuous(labels=formatter1000())  
  
#col.maps=col.maps+geom\_point(data = current.points[!is.na(current.points$year),1:2],aes(x, y), size=0.8, alpha=0.7, col='red')  
  
print(col.maps)



col.maps.mean=ggplot(col.summary.raster.df[col.summary.raster.df$variable%in%  
 c('median'),])+  
 geom\_raster(aes(x, y, fill=dec))+   
 #scale\_fill\_brewer(palette = "OrRd")+  
 #scale\_fill\_continuous(guide = guide\_colourbar(direction = "horizontal",nbin = 30))+  
 #facet\_grid(.~variable)+  
 #scale\_fill\_viridis()+  
 #theme(legend.position = 'bottom')+   
 scale\_fill\_brewer(palette = 'Spectral', na.value = 'white')+  
 geom\_polygon(data=latam.map,   
 aes(long, lat, group=group),  
 alpha=0, col='black')+ # Mapa  
 theme\_classic()+ # estilo del gráfico (que no haya grilla por ej.)  
 coord\_equal()+ # que no deforme horizontal ni verticalmente las ecalas  
 xlab('Longitude in km')+  
 ylab('Latitude in km')+  
 xlim(c(sim.extent[1], sim.extent[2]))+  
 #scale\_x\_continuous(labels=function(x)x/1000, n.breaks = 5)+  
 #theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1))+  
 ylim(c(sim.extent[3], sim.extent[4]))+  
 #scale\_y\_continuous(labels=function(x)x/1000, n.breaks = 5)+  
 ggtitle('Year of colonization')+  
 annotation\_scale()  
  
  
print(col.maps.mean)



### Evaluación a nivel de nicho climático (en pausa por ahora)

Esquema de lo que quiero hacer (grafico para la diapo)

Grafico posta

Esquemas para la diapo

### re mapeo from env to geo

# Estimación de efectos en el uso de nicho de B. dahlbomii

## climatic data pre-processing