Vermeulen\_et\_al\_2024\_Ecology

Liezl M Vermeulen

2024-09-11

Check if getwd() returns the correct working directory (the folder where this .Rmd file is stored).

# Bayesian Generalised Linear Models (GLMs) for ecosystem stability

# 1 Introduction

This R markdown tutorial describes the code used to carry out the analysis for the paper “Disentangling climate and disturbance legacy effects on savanna stability”.

The objectives of this analysis are:

* Quantify savanna resistance and resilience to drought using remote sensing
* Determine the ecosystem characteristics, climate legacy, and local disturbance pressures that affect savanna drought resistance and resilience using Bayesian Generalised Linear Models (GLMs)

# 2 Packages

We begin by taking care of the packages we need for this session. This piece of code first creates a list containing the required packages and then runs them through a for-loop to check if they are installed or not. If a package is not installed yet, it will be installed with the install.packages() function. Afterwards, all the required packages are loaded with the library() function.

# Create a list with the required packages  
requiredPackages <- c("raster", "rgdal", "rts", "terra", "spdep", "brms", "ggplot2", "matrixStats", "dplyr", "tidyr", "car", "tidybayes", "effects")  
  
# Check if required packages are installed; if a package is not installed, install it; then load the packages  
for (package in requiredPackages) {  
 if (!require(package, character.only=TRUE)) {  
 install.packages(package)  
 }  
 library(package, character.only=TRUE)  
}

## Loading required package: raster

## Loading required package: sp

## Loading required package: rgdal

## Please note that rgdal will be retired during 2023,  
## plan transition to sf/stars/terra functions using GDAL and PROJ  
## at your earliest convenience.  
## See https://r-spatial.org/r/2022/04/12/evolution.html and https://github.com/r-spatial/evolution  
## rgdal: version: 1.6-5, (SVN revision 1199)  
## Geospatial Data Abstraction Library extensions to R successfully loaded  
## Loaded GDAL runtime: GDAL 3.5.2, released 2022/09/02  
## Path to GDAL shared files: C:/Users/u0142455/AppData/Local/R/win-library/4.2/rgdal/gdal  
## GDAL binary built with GEOS: TRUE   
## Loaded PROJ runtime: Rel. 8.2.1, January 1st, 2022, [PJ\_VERSION: 821]  
## Path to PROJ shared files: C:/Users/u0142455/AppData/Local/R/win-library/4.2/rgdal/proj  
## PROJ CDN enabled: FALSE  
## Linking to sp version:1.6-0  
## To mute warnings of possible GDAL/OSR exportToProj4() degradation,  
## use options("rgdal\_show\_exportToProj4\_warnings"="none") before loading sp or rgdal.

## Loading required package: rts

## Loading required package: terra

## terra 1.7.18

##   
## Attaching package: 'terra'

## The following object is masked from 'package:rgdal':  
##   
## project

## Loading required package: xts

## Loading required package: zoo

##   
## Attaching package: 'zoo'

## The following object is masked from 'package:terra':  
##   
## time<-

## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

##   
## ################################### WARNING ###################################  
## # We noticed you have dplyr installed. The dplyr lag() function breaks how #  
## # base R's lag() function is supposed to work, which breaks lag(my\_xts). #  
## # #  
## # If you call library(dplyr) later in this session, then calls to lag(my\_xts) #  
## # that you enter or source() into this session won't work correctly. #  
## # #  
## # All package code is unaffected because it is protected by the R namespace #  
## # mechanism. #  
## # #  
## # Set `options(xts.warn\_dplyr\_breaks\_lag = FALSE)` to suppress this warning. #  
## # #  
## # You can use stats::lag() to make sure you're not using dplyr::lag(), or you #  
## # can add conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop #  
## # dplyr from breaking base R's lag() function. #  
## ################################### WARNING ###################################

## rts 1.1-14 (2023-10-01)

## Loading required package: spdep

## Loading required package: spData

## To access larger datasets in this package, install the spDataLarge  
## package with: `install.packages('spDataLarge',  
## repos='https://nowosad.github.io/drat/', type='source')`

## Loading required package: sf

## Linking to GEOS 3.9.3, GDAL 3.5.2, PROJ 8.2.1; sf\_use\_s2() is TRUE

## Loading required package: brms

## Loading required package: Rcpp

## Loading 'brms' package (version 2.20.4). Useful instructions  
## can be found by typing help('brms'). A more detailed introduction  
## to the package is available through vignette('brms\_overview').

##   
## Attaching package: 'brms'

## The following object is masked from 'package:terra':  
##   
## autocor

## The following object is masked from 'package:stats':  
##   
## ar

## Loading required package: ggplot2

## Loading required package: matrixStats

## Loading required package: dplyr

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:matrixStats':  
##   
## count

## The following objects are masked from 'package:xts':  
##   
## first, last

## The following objects are masked from 'package:terra':  
##   
## intersect, union

## The following objects are masked from 'package:raster':  
##   
## intersect, select, union

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

## Loading required package: tidyr

##   
## Attaching package: 'tidyr'

## The following object is masked from 'package:terra':  
##   
## extract

## The following object is masked from 'package:raster':  
##   
## extract

## Loading required package: car

## Loading required package: carData

##   
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':  
##   
## recode

## Loading required package: tidybayes

##   
## Attaching package: 'tidybayes'

## The following objects are masked from 'package:brms':  
##   
## dstudent\_t, pstudent\_t, qstudent\_t, rstudent\_t

## Loading required package: effects

## lattice theme set by effectsTheme()  
## See ?effectsTheme for details.

# 3 Settings

Next, we make sure that we are able to import the provided data easily. Modify wd to provide the correct path to your main working directory.

## 3.1 Response metrics

The NDVI time series data for calculating resistance and resilience is loaded from wd.ndvi, while the intermediary response metric results are saved to wd.response.

## 3.2 Bayesian GLMs

The ecosystem characteristics, climate legacy and disturbance legacy input files are loaded from wd.eco, wd.dist and wd.climate respectively. Finally, save our results in wd.results.

getwd()

## [1] "C:/Users/u0142455/Documents/PhD/Processing/ch2"

wd\_ndvi <- 'C:/Users/u0142455/OneDrive - KU Leuven/PhD/Processing/ch2/data/modis/mod13/VI\_Monthly\_1Km\_v6/NDVI'  
wd\_response <- './data/response\_metrics'  
wd\_eco <- './data/ecosys\_char'  
wd\_dist <- './data/disturbance'  
wd\_climate <- './data/climate'  
  
wd\_results <- './data/results'  
  
# Load your study area shapefile  
knp <- shapefile('./data/aoi/knp.shp')

# 4 Load data

## 4.1 Response metrics

In this analysis, we use [MODIS](https://lpdaac.usgs.gov/data/get-started-data/collection-overview/missions/modis-overview/) imagery, specifically the [MOD13A3](https://lpdaac.usgs.gov/products/mod13a3v061/) product. The MOD13A3 product provides monthly composites for two Vegetation Indices (VIs) at a spatial resolution of 1 km x 1 km. The first is the Normalized Difference Vegetation Index (NDVI) and the second one is the Enhanced Vegetation Index (EVI), which has improved sensitivity over high biomass regions. We will use NDVI time series data in this session.

We load the monthly NDVI time series data from 2000 to 2022 as a Raster Stack.

# Load files NDVI raster stack  
ndvi\_files <- list.files(path = wd\_ndvi, pattern="\*.tif", full.names=TRUE, recursive=FALSE)  
ndvi\_stack <- stack(ndvi\_files)  
  
modis\_crs <- crs(ndvi\_stack)  
knp\_prj <- spTransform(knp, modis\_crs)  
ndvi\_crop <- crop(ndvi\_stack, knp\_prj)  
  
ndvi\_scaled <- stack(ndvi\_crop \* 0.0001) # apply scale factor

## 4.2 Bayesian GLMs

A wide range of data sources were used as input for the explanatory variables fo the Bayesian GLMs, representing underlying ecosystem characteristics, climate legacy and disturbance legacy. Load in the respective datasets and subset the legacy data to select relevant years to match the corresponding response variables, i.e. 2000 - 2015 for the resistance/resilience and 1986 - 2015 for the woody cover change data.

# scaling factors required for SoilGrids datasets  
carbon\_factor <- 10   
sand\_factor <- 10  
clay\_factor <- 10  
  
# Ecosystem characteristics  
soil\_carbon <- raster(file.path(wd\_eco, "soil/SoilGrids/soilCarbon\_250m\_KNP\_utm.tif")) / carbon\_factor  
soil\_sand <- raster(file.path(wd\_eco, "soil/SoilGrids/soilSand\_250m\_KNP\_utm.tif")) / sand\_factor  
soil\_clay <- raster(file.path(wd\_eco, "soil/SoilGrids/soilClay\_250m\_KNP\_utm.tif")) / clay\_factor  
geology <- shapefile(file.path(wd\_eco, "soil/basalt\_granite.shp"))  
elev <- raster(file.path(wd\_eco, "terrain/elev\_knp\_utm.tif"))  
slope <- raster(file.path(wd\_eco, "terrain/slope\_knp\_utm.tif"))  
twi <- raster(file.path(wd\_eco, "terrain/twi\_knp\_utm.tif"))  
woodyCov <- raster(file.path(wd\_eco,"woody\_cov/woodyCover\_Venter2017.tif"))  
  
# Climate legacy  
# 1981 - 2015  
drought <- stack(file.path(wd\_climate, "spi/spi12\_drought\_sev\_stack.tif"))   
drought\_sum\_2015 <- abs(sum(drought[[20:34]])) # 2000- 2015: resistance/resilience)  
drought\_sum\_full <- abs(sum(drought[[6:35]])) # 1986 - 2015: woody cover change  
  
wetness <- stack(file.path(wd\_climate, "spi/spi12\_wetness\_sev\_stack.tif"))  
wetness\_sum\_2015 <- abs(sum(wetness[[20:35]])) # 2000- 2015: resistance/resilience  
wetness\_sum\_full <- abs(sum(wetness[[6:35]])) # 1986 - 2015: woody cover change  
  
# Disturbance legacy  
elephant <- raster(file.path(wd\_dist, "elephants/elephant\_impact\_heatmap.tif")) / 100  
fire\_freq\_files <- list.files(path=paste0(wd\_dist,"/fire/output/fire\_freq"),pattern="\*.tif", full.names=TRUE, recursive=FALSE)   
fire\_freq <- stack(fire\_freq\_files)  
fire\_freq\_sum\_2015 <- sum(fire\_freq[[58:73]]) # 2000- 2015: resistance/resilience  
fire\_freq\_sum\_full <- sum(fire\_freq[[44:73]]) # 1986 - 2015: woody cover change

# 5 Response metrics

This section focuses on quantifying resistance and resilience to drought using an NDVI time series.

## 5.1 Resistance

Resistance is related to how well an ecosystem can withstand a disturbance event, i.e., how low is the magnitude of change in the NDVI time series due to a disturbance event (Lloret et al., 2007). Higher values indicate a more severe impact on the ecosystem, thus lower resistance (De Keersmaecker et al., 2014). Resistance is calculated from the NDVI anomaly time series, which is generated by removing the seasonal component, i.e., the long-term mean NDVI for a particular month of the year. The seasonal component incorporates phenological variation through time, which could potentially mask signals of stability (De Keersmaecker et al., 2015). The NDVI anomaly is calculated from a time series with observations from time to as follows:

where

where  is the seasonal component and  is the mean NDVI for all dates , across the entire time period (2000 – 2022), falling within month . Resistance to the 2015/2016 drought was subsequently calculated according to Lloret et al. (2007), which is bounded between 0 and 1:

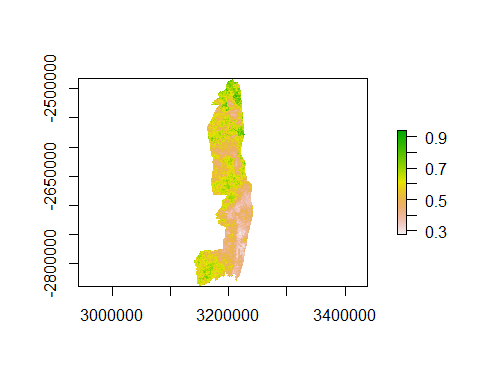
Resistance is normalised using the seasonal component to account for variations in biomass. To ease interpretation, the value was also inverted so that higher absolute values correspond to a more resistant and stable ecosystem. A single resistance value per pixel location was then obtained by computing the minimum absolute  value for the growing season of the 2015/2016 drought, namely October 2015 to March 2016.

## 5.2 Resilience

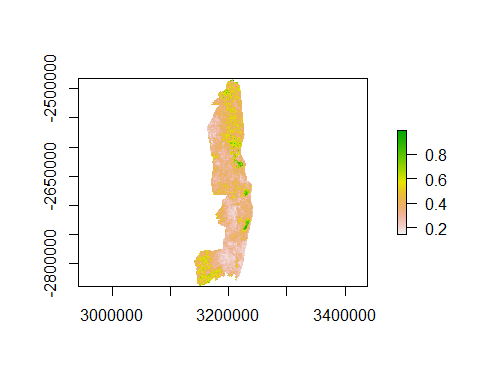
Resilience is defined as the rate of return of a system to its equilibrium state after a perturbation (Pimm, 1984). The temporal relationship between observations serves as a resilience metric and can be quantified by the temporal autocorrelation at lag-1 (), where higher  values indicate more similar subsequent anomalies, and thus, a slower return to equilibrium. Therefore, resilience can be expressed as (Dakos et al., 2012) and is bounded between 0 and 1:

where the anomaly time series is given by . Ecosystems with low resilience exhibit a slower return to equilibrium, whereas those with high resilience recover more quickly (De Keersmaecker et al., 2014). A single resilience value per pixel location was then obtained by computing the value for the three years following the drought.

# Set the disturbance period start and end. In this case, the drought occurred in the growing season of the 2015/2016 i.e. Oct 2015 - March 2016  
disturbance\_time <- c(2015,10)  
disturbance\_end\_time <- c(2016,3)  
xresponse <- function(data,thrs) {  
 r<-data  
 m<-na.approx(as.matrix(r))  
 resistance=c()  
 resilience=c()  
 variance=c()  
 for (x in 1:ncell(r)){  
   
 if(sum(is.na(m[x,]))<thrs){   
 pix=c()  
 for (i in 1:nlayers(r)){  
 pix[i]<-as.vector(m[x,i])  
 }  
 pix\_full <- append(NA,pix) # MODIS is missing the month of January 2000, add in as NA value  
 ndvi\_ts <- ts(pix\_full, start=c(2000,1), end=c(2022, 12), frequency=12)  
 pix\_m <- matrix(pix\_full[1:120], 10, byrow = TRUE) # create matrix, ordered according to the number of years   
 means <- colMeans(pix\_m, na.rm=TRUE) # calculate the mean of every time step i.e. month  
 sd <- colSds(pix\_m, na.rm=TRUE)  
 seasonality <- rep(means, 23) # duplicate for all years to get the seasonality  
   
 ## Anomaly  
 anomaly <- pix\_full - seasonality  
 anomaly\_ts <- ts(anomaly, start=c(2000,1), end=c(2022, 12), frequency=12)  
 anomaly\_norm <- anomaly / seasonality # normalised anomaly  
 anomaly\_norm\_ts <- ts(anomaly\_norm, start=c(2000,1), end=c(2022, 12), frequency=12)  
   
 # Resistance  
 # extract the NDVI naomaly time series for the disturbance period  
 disturbance\_data <- window(anomaly\_norm\_ts, start = disturbance\_time, end = disturbance\_end\_time)  
   
 # minimum NDVI anomly value within the disutrbance period  
 resist <- abs(min(disturbance\_data))   
 # date of minimum value, to calculate resilience from  
 year <- floor(time(disturbance\_data)[which.max(disturbance\_data)])  
 month <- (time(disturbance\_data)[which.min(disturbance\_data)] %% 1)\*12  
   
 if (resist < 0 || resist > 1) {  
 resistance[x] <- NA  
 } else {  
 resistance[x] <- 1 - resist  
 }  
   
 # Resilience  
 # extract NDVI anomaly post-disturbance for x years  
 recovery\_years <- 3  
 post\_disturbance\_data <- window(anomaly\_ts, start = c(as.numeric(year),as.numeric(month)+1), end = c(as.numeric(year) + recovery\_years , as.numeric(month)+1))  
   
 #acf\_post <- mean(window(acf, start = disturbance\_end\_time, end = recovery\_end\_time))  
 # calculate autocorrelation at lag-1  
 autocorrelation\_lag1 <- (acf(post\_disturbance\_data, lag.max = 1, plot = FALSE)$acf[2])  
 if (autocorrelation\_lag1 < 0) {  
 resilience[x] <- NA  
 } else {  
 resilience[x] <- 1 - autocorrelation\_lag1  
 }  
   
   
 # Variance (pre-drought)  
 anomaly\_pre <- window(anomaly\_ts, end = disturbance\_time)  
 variance[x] <- sd(as.numeric(anomaly\_pre), na.rm=TRUE)   
   
 } else {  
 resistance[x] <- NA  
 resilience[x] <- NA  
 variance[x] <- NA  
 }  
 }  
 results<-list(resistance, resilience, variance, time)  
 names(results)<-c("resistance","resilence","variance", "time")  
   
 resist\_raster<-raster(extent(r),nrow=nrow(r),ncol=ncol(r),crs=proj4string(r))  
 values(resist\_raster)<-results[[1]]  
   
 resil\_raster<-raster(extent(r),nrow=nrow(r),ncol=ncol(r),crs=proj4string(r))  
 values(resil\_raster)<-results[[2]]  
   
 var\_raster<-raster(extent(r),nrow=nrow(r),ncol=ncol(r),crs=proj4string(r))  
 values(var\_raster)<-results[[3]]  
   
 output<-list(resist\_raster,resil\_raster,var\_raster)  
 return(output)  
   
}  
  
response\_r <- xresponse(data=ndvi\_scaled, thrs=12)  
resistance <- response\_r[[1]]  
resilience <- response\_r[[2]]  
plot(mask(resistance, knp\_prj)) # resistance



plot(mask(resilience, knp\_prj)) # resilience



#writeRaster(response\_r[[1]], file.path(wd\_response, "resistance.tif"), format = "GTiff", overwrite = TRUE)  
#writeRaster((response\_r[[2]]), file.path(wd\_response, "resilience.tif"), format = "GTiff", overwrite = TRUE)  
#writeRaster(response\_r[[3]], file.path(wd\_response, "variance.tif"), format = "GTiff", overwrite = TRUE)

# 6 Bayesian GLMs

## 6.1 Pre-processing and set-up

Perform some pre-processing steps, including projecting all datasets to the same resolution and coordinate reference system and cropping to the same extent. These steps are unnecessary if your datasets already align in terms of coordinate reference system, spatial resolution and extent.

knp <- shapefile("./data/aoi/knp\_reproj.shp")  
  
# Project everything to CHIRPS spatial resolution, local crs  
chirps\_crs <- crs(drought\_sum\_2015)  
utm\_crs <- crs(fire\_freq)  
  
# Load saved response layers if necessary  
resistance <- raster(file.path(wd\_response,"new/resistance.tif"))  
resilience <- raster(file.path(wd\_response,"new/resilience.tif"))  
  
# Use one layer as the base layer  
resistance\_rp <- mask(projectRaster(resistance, res=c(1000,1000), crs=utm\_crs, method="bilinear"), knp)  
base\_r <- resistance\_rp  
  
# Reproject and crop all layers to the same crs and extent  
resilience\_rp <- mask(projectRaster(resilience, base\_r, method="bilinear"), knp)  
  
fire\_freq\_2015\_rp <- round(mask(projectRaster(fire\_freq\_sum\_2015, base\_r, method="bilinear"), knp))  
fire\_freq\_full\_rp <- round(mask(projectRaster(fire\_freq\_sum\_full, base\_r, method="bilinear"), knp))  
elephant\_rp <- mask(projectRaster(elephant, base\_r, method="bilinear"), knp)  
  
drought\_2015\_rp <- mask(projectRaster(drought\_sum\_2015, base\_r, method="bilinear"), knp)  
drought\_full\_rp <- mask(projectRaster(drought\_sum\_full, base\_r, method="bilinear"), knp)  
wetness\_2015\_rp <- mask(projectRaster(wetness\_sum\_2015, base\_r, method="bilinear"), knp)  
wetness\_full\_rp <- mask(projectRaster(wetness\_sum\_full, base\_r, method="bilinear"), knp)  
  
elev\_rp <- mask(projectRaster(elev, base\_r, method="bilinear"), knp)  
slope\_rp <- mask(projectRaster(slope, base\_r, method="bilinear"), knp)  
twi\_rp <- mask(projectRaster(twi, base\_r, method="bilinear"), knp)  
soil\_sand\_rp <- mask(projectRaster(soil\_sand, base\_r, method="bilinear"), knp)  
soil\_carbon\_rp <- mask(projectRaster(soil\_carbon, base\_r, method="bilinear"), knp)  
soil\_clay\_rp <- mask(projectRaster(soil\_clay, base\_r, method="bilinear"), knp)  
woodyCov\_rp <- mask(projectRaster(woodyCov, base\_r, method="bilinear"), knp)  
  
geology\_rp <- mask(rasterize(as(geology, "SpatVector"), rast(base\_r), field="GEOLOGY"), vect(knp))  
  
# Set up list of rasters and corresponding names  
r\_names <- c("resist" , "resil",  
 "elev", "slope", "twi", "soilSand", "soilClay", "soilCarbon",  
 "woodyCov", "geology", "fireFreq2015", "fireFreqFull", "elephant",  
 "drought2015","droughtFull", "wetness2015", "wetnessFull")  
  
r\_list <- c(rast(resistance\_rp), rast(resilience\_rp),  
 rast(elev\_rp), rast(slope\_rp), rast(twi\_rp), rast(soil\_sand\_rp),  
 rast(soil\_clay\_rp), rast(soil\_carbon\_rp), rast(woodyCov\_rp),   
 geology\_rp, rast(fire\_freq\_2015\_rp), rast(fire\_freq\_full\_rp),   
 rast(elephant\_rp), rast(drought\_2015\_rp), rast(drought\_full\_rp),   
 rast(wetness\_2015\_rp), rast(wetness\_full\_rp))  
  
names(r\_list) <- r\_names  
  
# Stack rasters  
model\_stack <- r\_list  
  
# Mask out rivers and areas within a 750m buffer  
rivers <- mtq <- st\_read("./data/rivers/buffer\_rivers\_750m.shp") %>%   
 dplyr::summarise()

## Reading layer `buffer\_rivers\_750m' from data source   
## `C:\Users\u0142455\Documents\PhD\Processing\ch2\data\rivers\buffer\_rivers\_750m.shp'   
## using driver `ESRI Shapefile'  
## Simple feature collection with 1 feature and 3 fields  
## Geometry type: MULTIPOLYGON  
## Dimension: XY  
## Bounding box: xmin: 286542.5 ymin: 7213285 xmax: 403314.8 ymax: 7470133  
## Projected CRS: WGS 84 / UTM zone 36S

mask <- st\_bbox(model\_stack) %>% # take extent of your raster  
 st\_as\_sfc() %>% # make it a sf object  
 st\_set\_crs(st\_crs(mtq)) %>% # in CRS of your polygon   
 st\_difference(mtq) %>% # intersect with the polygon object  
 st\_as\_sf() # interpret as sf (and not sfc) object  
  
model\_stack\_masked <- model\_stack %>%   
 mask(mask)  
  
# Create model data frame  
model\_df <- as.data.frame(model\_stack\_masked,xy=TRUE)  
  
# Create a copy of your original dataframe for further processing  
model\_df2 <- cbind(ID = 1:nrow(model\_df), model\_df)  
model\_df2 <- na.omit(model\_df2)  
  
# scale and center all numerical explanatory variables  
model\_df2[,6:20] <- model\_df2[,6:20] %>% mutate(across(where(is.numeric), scale))

## 6.2 Multicolilnearity analysis

First create the functions for calculating the variance inflation factor (VIF), as adapted from Zuur et al. (2010). Run this code block to generate the necessary functions.

# Functions for calculating VIF, as adapted from Zuur et al. (2010)  
  
panel.cor <- function(x, y, digits=1, prefix="", cex.cor)  
{  
 usr <- par("usr"); on.exit(par(usr))  
 par(usr = c(0, 1, 0, 1))  
 r1=cor(x,y,use="pairwise.complete.obs")  
 r <- abs(cor(x, y,use="pairwise.complete.obs"))  
   
 txt <- format(c(r1, 0.123456789), digits=digits)[1]  
 txt <- paste(prefix, txt, sep="")  
 if(missing(cex.cor)) cex <- 0.9/strwidth(txt)  
 text(0.5, 0.5, txt, cex = cex \* r)  
}  
  
panel.smooth2=function (x, y, col = par("col"), bg = NA, pch = par("pch"),  
 cex = 1, col.smooth = "red", span = 2/3, iter = 3, ...)  
{  
 points(x, y, pch = pch, col = col, bg = bg, cex = cex)  
 ok <- is.finite(x) & is.finite(y)  
 if (any(ok))  
 lines(stats::lowess(x[ok], y[ok], f = span, iter = iter),  
 col = 1, ...)  
}  
  
  
panel.lines2=function (x, y, col = par("col"), bg = NA, pch = par("pch"),  
 cex = 1, ...)  
{  
 points(x, y, pch = pch, col = col, bg = bg, cex = cex)  
 ok <- is.finite(x) & is.finite(y)  
 if (any(ok)){  
 tmp=lm(y[ok]~x[ok])  
 abline(tmp)}  
   
}  
  
  
  
  
panel.hist <- function(x, ...)  
{  
 usr <- par("usr"); on.exit(par(usr))  
 par(usr = c(usr[1:2], 0, 1.5) )  
 h <- hist(x, plot = FALSE)  
 breaks <- h$breaks; nB <- length(breaks)  
 y <- h$counts; y <- y/max(y)  
 rect(breaks[-nB], 0, breaks[-1], y, col="white", ...)  
}  
  
  
  
#VIF  
myvif <- function(mod) {  
 v <- vcov(mod)  
 assign <- attributes(model.matrix(mod))$assign  
 if (names(coefficients(mod)[1]) == "(Intercept)") {  
 v <- v[-1, -1]  
 assign <- assign[-1]  
 } else warning("No intercept: vifs may not be sensible.")  
 terms <- labels(terms(mod))  
 n.terms <- length(terms)  
 if (n.terms < 2) stop("The model contains fewer than 2 terms")  
 if (length(assign) > dim(v)[1] ) {  
 diag(tmp\_cor)<-0  
 if (any(tmp\_cor==1.0)){  
 return("Sample size is too small, 100% collinearity is present")  
 } else {  
 return("Sample size is too small")  
 }  
 }  
 R <- cov2cor(v)  
 detR <- det(R)  
 result <- matrix(0, n.terms, 3)  
 rownames(result) <- terms  
 colnames(result) <- c("GVIF", "Df", "GVIF^(1/2Df)")  
 for (term in 1:n.terms) {  
 subs <- which(assign == term)  
 result[term, 1] <- det(as.matrix(R[subs, subs])) \* det(as.matrix(R[-subs, -subs])) / detR  
 result[term, 2] <- length(subs)  
 }  
 if (all(result[, 2] == 1)) {  
 result <- data.frame(GVIF=result[, 1])  
 } else {  
 result[, 3] <- result[, 1]^(1/(2 \* result[, 2]))  
 }  
 invisible(result)  
}  
  
corvif <- function(dataz) {  
 dataz <- as.data.frame(dataz)  
 #correlation part  
 cat("Correlations of the variables\n\n")  
 tmp\_cor <- cor(dataz,use="complete.obs")  
 print(tmp\_cor)  
   
 #vif part  
 form <- formula(paste("fooy ~ ",paste(strsplit(names(dataz)," "),collapse=" + ")))  
 dataz <- data.frame(fooy=1,dataz)  
 lm\_mod <- lm(form,dataz)  
   
 cat("\n\nVariance inflation factors\n\n")  
 print(myvif(lm\_mod))  
}  
  
myvif <- function(mod) {  
 v <- vcov(mod)  
 assign <- attributes(model.matrix(mod))$assign  
 if (names(coefficients(mod)[1]) == "(Intercept)") {  
 v <- v[-1, -1]  
 assign <- assign[-1]  
 } else warning("No intercept: vifs may not be sensible.")  
 terms <- labels(terms(mod))  
 n.terms <- length(terms)  
 if (n.terms < 2) stop("The model contains fewer than 2 terms")  
 if (length(assign) > dim(v)[1] ) {  
 diag(tmp\_cor)<-0  
 if (any(tmp\_cor==1.0)){  
 return("Sample size is too small, 100% collinearity is present")  
 } else {  
 return("Sample size is too small")  
 }  
 }  
 R <- cov2cor(v)  
 detR <- det(R)  
 result <- matrix(0, n.terms, 3)  
 rownames(result) <- terms  
 colnames(result) <- c("GVIF", "Df", "GVIF^(1/2Df)")  
 for (term in 1:n.terms) {  
 subs <- which(assign == term)  
 result[term, 1] <- det(as.matrix(R[subs, subs])) \* det(as.matrix(R[-subs, -subs])) / detR  
 result[term, 2] <- length(subs)  
 }  
 if (all(result[, 2] == 1)) {  
 result <- data.frame(GVIF=result[, 1])  
 } else {  
 result[, 3] <- result[, 1]^(1/(2 \* result[, 2]))  
 }  
 invisible(result)  
}  
  
  
  
corvif <- function(dataz) {  
 dataz <- as.data.frame(dataz)  
 #correlation part  
 cat("Correlations of the variables\n\n")  
 tmp\_cor <- cor(dataz,use="complete.obs")  
 print(tmp\_cor)  
   
 #vif part  
 form <- formula(paste("fooy ~ ",paste(strsplit(names(dataz)," "),collapse=" + ")))  
 dataz <- data.frame(fooy=1,dataz)  
 lm\_mod <- lm(form,dataz)  
   
 cat("\n\nVariance inflation factors\n\n")  
 print(myvif(lm\_mod))  
}

Next iteratively compute the VIF, removing the variable with the highest VIF, until all explanatory variables have a VIF < 3. Only four iterations were necessary, leading to the exclusion of elevation, soil carbon and soil pH.

# Select covariates  
Z <- model\_df2[,c("slope", "soilSand", "soilClay", "soilCarbon", "woodyCov",   
 "fireFreq2015", "elephant",  
 "drought2015", "wetness2015")]  
  
corvif(Z)

## Correlations of the variables  
##   
## slope soilSand soilClay soilCarbon woodyCov  
## slope 1.00000000 -0.037058915 -0.16351246 0.30785746 0.221160264  
## soilSand -0.03705892 1.000000000 -0.11909863 -0.27209154 -0.002465057  
## soilClay -0.16351246 -0.119098629 1.00000000 0.07959811 -0.655963202  
## soilCarbon 0.30785746 -0.272091535 0.07959811 1.00000000 0.038815782  
## woodyCov 0.22116026 -0.002465057 -0.65596320 0.03881578 1.000000000  
## fireFreq2015 0.06083911 -0.195201953 0.19830338 0.42551886 -0.275247350  
## elephant -0.03740018 -0.361448556 -0.24923350 0.01281165 0.158995582  
## drought2015 -0.22046784 0.484557421 0.20766101 -0.19041754 -0.164327457  
## wetness2015 0.14916804 -0.252738857 -0.18105262 0.34482421 0.242919092  
## fireFreq2015 elephant drought2015 wetness2015  
## slope 0.06083911 -0.03740018 -0.22046784 0.14916804  
## soilSand -0.19520195 -0.36144856 0.48455742 -0.25273886  
## soilClay 0.19830338 -0.24923350 0.20766101 -0.18105262  
## soilCarbon 0.42551886 0.01281165 -0.19041754 0.34482421  
## woodyCov -0.27524735 0.15899558 -0.16432746 0.24291909  
## fireFreq2015 1.00000000 -0.02685100 -0.08309926 0.33011003  
## elephant -0.02685100 1.00000000 -0.35562116 -0.05320892  
## drought2015 -0.08309926 -0.35562116 1.00000000 -0.25589517  
## wetness2015 0.33011003 -0.05320892 -0.25589517 1.00000000  
##   
##   
## Variance inflation factors  
##   
## GVIF  
## slope 1.227145  
## soilSand 1.697425  
## soilClay 2.103233  
## soilCarbon 1.496455  
## woodyCov 2.070993  
## fireFreq2015 1.526208  
## elephant 1.426717  
## drought2015 1.568682  
## wetness2015 1.498845

## 6.3 Build Bayesian GLMs

Each of the response metrics served as an independent response variable in three distinct stability models. We employed Bayesian Generalised Linear Models (GLMs), a statistical approach that accommodates complex data structures and varying levels of uncertainty, to investigate the relationship between our response metrics and various environmental, climatic, and disturbance predictors. Resistance and resilience were modelled using a Beta distribution with a logit link function to accommodate its bounded nature between 0 and 1. The woody cover change response variable was modelled using a Gaussian distribution, reflecting the continuous nature of the change in woody vegetation over time. The Bayesian GLMs were developed and implemented using the *brms* package.

Note: on a 4 core CPU with 64GB RAM, each model takes ~18 minutes.

# sample 70% proportion of the full model matrix  
set.seed(2024)  
split <- rsample::initial\_split(model\_df2, prop = 0.7, strata = geology)  
train\_df <- rsample::training(split)

# Resistance model  
start\_time <- Sys.time()  
resist\_formula <- resist ~ slope + soilSand\*geology + soilClay\*geology + soilCarbon\*geology + soilClay\*woodyCov + woodyCov\*fireFreq2015 + woodyCov\*elephant + woodyCov\*drought2015 + woodyCov\*wetness2015  
resist\_model <- brm(formula = resist\_formula,   
 data = train\_df,  
 family=Beta(link="logit"),  
 warmup = 1500,   
 iter = 2000,   
 chains = 4,   
 cores = 4,  
 save\_pars = save\_pars(all = TRUE))  
end\_time <- Sys.time()  
end\_time - start\_time  
saveRDS(resist\_model, file = file.path(wd\_results,"resist\_nosa\_inter.rda"))  
  
# Resilience model  
# note: we use fireFreq2015, drought2015 and wetness2015 as these match the   
# resistance/resilience temporal period i.e. 2000 - 2015  
resil\_formula <- resil ~ slope + soilSand\*geology + soilClay\*geology + soilCarbon\*geology + soilClay\*woodyCov + woodyCov\*fireFreq2015 + woodyCov\*elephant + woodyCov\*drought2015 + woodyCov\*wetness2015  
start\_time <- Sys.time()  
resil\_model <- brm(formula = resil\_formula,   
 family=Beta(link="logit"),  
 data = train\_df,  
 warmup = 1500,   
 iter = 2000,   
 chains = 4,   
 cores = 4,  
 seed = 123,  
 save\_pars = save\_pars(all = TRUE))  
end\_time <- Sys.time()  
end\_time - start\_time  
saveRDS(resil\_model, file = file.path(wd\_results,"resil\_nosa\_inter.rda"))

## 6.4 Check for spatial autocorrelation

Spatial autocorrelation is the term used to describe the presence of systematic spatial variation in a variable and positive spatial autocorrelation, which is most often encountered in practical situations, is the tendency for areas or sites that are close together to have similar values. We will check for the presence of spatial autocorrelation using the Moran’s I test. Spatial autocorrelation is present if the *p*-value is < 0.05. In our case, spatial autocorrelation is present thus we will account for it in section 6.5 using the residual autocovariate (RAC) approach.

resist\_model <- readRDS(file=file.path(wd\_results,"resist\_nosa\_inter.rda"))  
resil\_model <- readRDS(file=file.path(wd\_results,"resil\_nosa\_inter.rda"))  
  
# extract a neighbourhood matrix from your model data x and y coordinates  
xy\_df <- distinct(as.data.frame(cbind(train\_df$ID, train\_df$x, train\_df$y)))  
names(xy\_df) <- c("ID", "x", "y")  
coordinates(xy\_df) <- ~ x + y  
knea <- knearneigh(coordinates(xy\_df), longlat = FALSE)  
W\_nb <- knn2nb(knea, sym=TRUE)  
W <- nb2mat(W\_nb, style="B", zero.policy=FALSE)  
rownames(W) <- unique(train\_df$ID)  
listW <- nb2listw(W\_nb, style="W")  
  
# extract the residulas from your Bayesian GLMs  
bres\_resist <- residuals(resist\_model, method = "posterior\_predict")[,"Estimate"]  
bres\_resil <- residuals(resil\_model, method = "posterior\_predict")[,"Estimate"]  
  
# test for spatial autocorrelation  
moran.test(bres\_resist, listW)

##   
## Moran I test under randomisation  
##   
## data: bres\_resist   
## weights: listW   
##   
## Moran I statistic standard deviate = 91.447, p-value < 2.2e-16  
## alternative hypothesis: greater  
## sample estimates:  
## Moran I statistic Expectation Variance   
## 8.427507e-01 -7.621951e-05 8.494489e-05

moran.test(bres\_resil, listW)

##   
## Moran I test under randomisation  
##   
## data: bres\_resil   
## weights: listW   
##   
## Moran I statistic standard deviate = 88.774, p-value < 2.2e-16  
## alternative hypothesis: greater  
## sample estimates:  
## Moran I statistic Expectation Variance   
## 8.180732e-01 -7.621951e-05 8.493551e-05

# all models test singificant for spatial autocorrelation

## 6.5 Check model fit and performance

Also check other performance metrics, i.e. the Bayes R-squared, leave-one-out-cross-validation information criterion (LOOIC) and the posterior predictive check plots.

# calculate Bayes R-sqaured  
bayes\_R2(resist\_model)

## Estimate Est.Error Q2.5 Q97.5  
## R2 0.5119801 0.004196373 0.5033262 0.5200442

bayes\_R2(resil\_model)

## Estimate Est.Error Q2.5 Q97.5  
## R2 0.3681858 0.005309173 0.3579572 0.3781208

# calculate LOOIC  
loo\_resist <- loo(resist\_model, save\_psis = TRUE)  
loo\_resist$estimates[3]

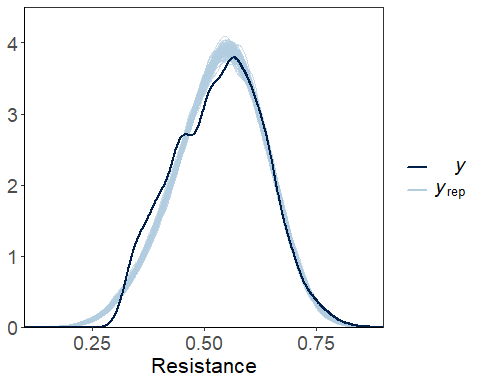
## [1] -32244.94

loo\_resil <- loo(resil\_model, save\_psis = TRUE, cores = 4)  
loo\_resil$estimates[3]

## [1] -34536.96

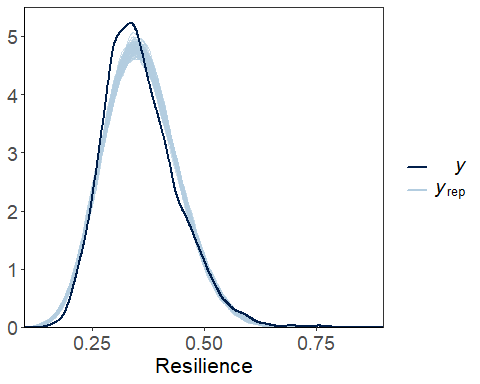
# Plot posterior predictive check plots  
pp\_resist <- pp\_check(resist\_model, ndraws = 100)  
pp\_resist + theme\_bw() +   
 labs(x = "Resistance") +  
 ylim(0,4.5)+  
 xlim(0.1,0.9)+  
 theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), panel.background = element\_blank(), axis.line = element\_line(colour = "black")) +   
 theme(axis.title=element\_text(size=16)) +  
 theme(axis.text=element\_text(size=14)) +  
 theme(legend.text=element\_text(size=14))

## Warning: Removed 12 rows containing non-finite values (`stat\_density()`).



pp\_resil <- pp\_check(resil\_model, ndraws = 100)  
pp\_resil + theme\_bw() +   
 labs(x = "Resilience") +  
 ylim(0,5.5)+  
 xlim(0.1,0.9)+  
 theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), panel.background = element\_blank(), axis.line = element\_line(colour = "black")) +   
 theme(axis.title=element\_text(size=16)) +  
 theme(axis.text=element\_text(size=14)) +  
 theme(legend.text=element\_text(size=14))

## Warning: Removed 26 rows containing non-finite values (`stat\_density()`).



## 6.6 Build Bayesian GLM RAC models

The RAC approach consists of fitting a base model, calculating an autocovariate based on the residuals of the base model of each location using a mean focal operation, updating the base model linear predictor to include the calculated autocovariate and refitting using the new RAC model.

# Resistance model  
# build RAC model   
xy <- cbind(train\_df$x, train\_df$y)  
ext(base\_r) # check extent of base raster  
# create blank raster based on extent of base raster  
rast <- raster(ncol=409, nrow = 247, ymn = 7148741.09694641, ymx = 7557741.09694641, xmn = 214375.07043509, xmx = 461375.07043509)   
res(rast) <- 1000 # set resolution  
xy\_residuals <- cbind(xy, residuals(resist\_model, method = "posterior\_predict")[,"Estimate"])  
rast[cellFromXY(rast, xy\_residuals)] <- xy\_residuals[,3]  
# calculate residuals autocovariate using focal operation  
focal\_rac\_rast <- focal(rast, w=matrix(1/9,nrow=3,ncol=3), fun = mean, na.rm = TRUE)  
plot(focal\_rac\_rast)  
focal\_rac\_vect <- terra::extract(rast(focal\_rac\_rast), vect(xy), xy=TRUE)  
names(focal\_rac\_vect) <- c("ID","RAC", "x", "y")  
train\_df$RACresist <- focal\_rac\_vect[,2]  
   
start\_time <- Sys.time()  
resist\_formula\_rac <- resist ~ slope + soilSand\*geology + soilClay\*geology + soilCarbon\*geology + soilClay\*woodyCov + woodyCov\*fireFreq2015 + woodyCov\*elephant + woodyCov\*drought2015 + woodyCov\*wetness2015 + RACresist  
resist\_model\_rac <- brm(formula = resist\_formula\_rac,   
 data = train\_df,  
 family = Beta(link="logit"),  
 warmup = 1500,   
 iter = 2000,   
 chains = 4,   
 cores = 4,  
 seed = 123,  
 save\_pars = save\_pars(all = TRUE))  
end\_time <- Sys.time()  
end\_time - start\_time  
saveRDS(resist\_model\_rac, file = file.path(wd\_results,"resist\_rac\_inter.rda"))  
  
# Resilience model  
# build RAC model   
xy <- cbind(train\_df$x, train\_df$y)  
ext(base\_r) # check extent of base raster  
# create blank raster based on extent of base raster  
rast <- raster(ncol=409, nrow = 247, ymn = 7148741.09694641, ymx = 7557741.09694641, xmn = 214375.07043509, xmx = 461375.07043509)   
res(rast) <- 1000 # set resolution  
xy\_residuals <- cbind(xy, residuals(resil\_model, method = "posterior\_predict")[,"Estimate"])  
rast[cellFromXY(rast, xy\_residuals)] <- xy\_residuals[,3]  
# calculate residuals autocovariate using focal operation  
focal\_rac\_rast <- focal(rast, w=matrix(1/9,nrow=3,ncol=3), fun = mean, na.rm = TRUE)  
plot(focal\_rac\_rast)  
focal\_rac\_vect <- terra::extract(rast(focal\_rac\_rast), vect(xy), xy=TRUE)  
names(focal\_rac\_vect) <- c("ID","RAC", "x", "y")  
train\_df$RACresil <- focal\_rac\_vect[,2]  
  
resil\_formula\_rac <- resil ~ slope + soilSand\*geology + soilClay\*geology + soilCarbon\*geology + soilClay\*woodyCov + woodyCov\*fireFreq2015 + woodyCov\*elephant + woodyCov\*drought2015 + woodyCov\*wetness2015 + RACresil  
start\_time <- Sys.time()  
resil\_model\_rac <- brm(formula = resil\_formula\_rac,   
 family=Beta(link="logit"),  
 data = train\_df,  
 warmup = 1500,   
 iter = 2000,   
 chains = 4,   
 cores= 4,  
 seed=123,  
 save\_pars = save\_pars(all = TRUE))  
end\_time <- Sys.time()  
end\_time - start\_time  
saveRDS(resil\_model\_rac, file = file.path(wd\_results,"resil\_rac\_inter.rda"))

## 6.7 Check final performance metrics

Test Moran’s I, Bayes R-squared, LOOIC and posterior predictive check plots of the new RAC models.

resist\_model\_rac <- readRDS(file=file.path(wd\_results,"resist\_rac\_inter.rda"))  
resil\_model\_rac <- readRDS(file=file.path(wd\_results,"resil\_rac\_inter.rda"))  
  
# extract the residulas from your Bayesian GLMs  
bres\_resist <- residuals(resist\_model\_rac, method = "posterior\_predict")[,"Estimate"]  
bres\_resil <- residuals(resil\_model\_rac, method = "posterior\_predict")[,"Estimate"]  
  
# test for spatial autocorrelation  
moran.test(bres\_resist, listW)

##   
## Moran I test under randomisation  
##   
## data: bres\_resist   
## weights: listW   
##   
## Moran I statistic standard deviate = 1.2714, p-value = 0.1018  
## alternative hypothesis: greater  
## sample estimates:  
## Moran I statistic Expectation Variance   
## 1.164066e-02 -7.621951e-05 8.492516e-05

moran.test(bres\_resil, listW)

##   
## Moran I test under randomisation  
##   
## data: bres\_resil   
## weights: listW   
##   
## Moran I statistic standard deviate = -2.2566, p-value = 0.988  
## alternative hypothesis: greater  
## sample estimates:  
## Moran I statistic Expectation Variance   
## -2.087261e-02 -7.621951e-05 8.493165e-05

# all models test insignificant for spatial autocorrelation  
  
# calculate Bayes R-sqaured  
bayes\_R2(resist\_model\_rac)

## Estimate Est.Error Q2.5 Q97.5  
## R2 0.9534282 0.0001649858 0.9530976 0.9537368

bayes\_R2(resil\_model\_rac)

## Estimate Est.Error Q2.5 Q97.5  
## R2 0.9193241 0.0003749372 0.9185772 0.9200165

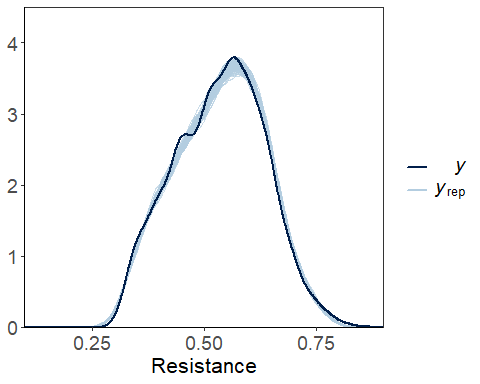
# calculate LOOIC  
loo\_resist <- loo(resist\_model\_rac, save\_psis = TRUE)  
loo\_resist$estimates[3]

## [1] -62837.47

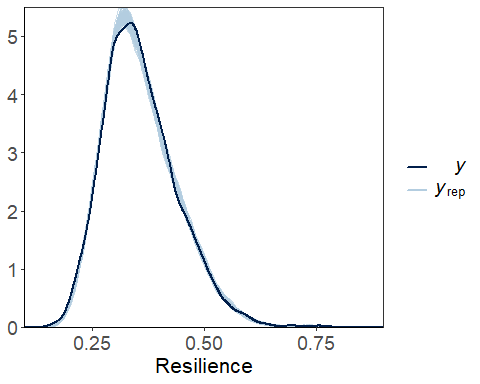
loo\_resil <- loo(resil\_model\_rac, save\_psis = TRUE, cores = 4)  
loo\_resil$estimates[3]

## [1] -61553.07

# Plot posterior predictive check plots  
pp\_resist <- pp\_check(resist\_model\_rac, ndraws = 100)  
pp\_resist + theme\_bw() +   
 labs(x = "Resistance") +  
 ylim(0,4.5)+  
 xlim(0.1,0.9)+  
 theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), panel.background = element\_blank(), axis.line = element\_line(colour = "black")) +   
 theme(axis.title=element\_text(size=16)) +  
 theme(axis.text=element\_text(size=14)) +  
 theme(legend.text=element\_text(size=14))



pp\_resil <- pp\_check(resil\_model\_rac, ndraws = 100)  
pp\_resil + theme\_bw() +   
 labs(x = "Resilience") +  
 ylim(0,5.5)+  
 xlim(0.1,0.9)+  
 theme(panel.grid.major = element\_blank(), panel.grid.minor = element\_blank(), panel.background = element\_blank(), axis.line = element\_line(colour = "black")) +   
 theme(axis.title=element\_text(size=16)) +  
 theme(axis.text=element\_text(size=14)) +  
 theme(legend.text=element\_text(size=14))



# check model estimates  
summary(resist\_model\_rac)

## Family: beta   
## Links: mu = logit; phi = identity   
## Formula: resist ~ slope + soilSand \* geology + soilClay \* geology + soilCarbon \* geology + soilClay \* woodyCov + woodyCov \* fireFreq2015 + woodyCov \* elephant + woodyCov \* drought2015 + woodyCov \* wetness2015 + RACresist   
## Data: train\_df (Number of observations: 13121)   
## Draws: 4 chains, each with iter = 2000; warmup = 1500; thin = 1;  
## total post-warmup draws = 2000  
##   
## Population-Level Effects:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS  
## Intercept 0.15 0.00 0.14 0.15 1.00 1347  
## slope 0.04 0.00 0.03 0.04 1.00 3345  
## soilSand 0.13 0.00 0.12 0.13 1.00 1664  
## geologyGranite 0.05 0.00 0.04 0.06 1.00 1447  
## soilClay -0.07 0.00 -0.07 -0.06 1.00 1101  
## soilCarbon -0.04 0.00 -0.04 -0.04 1.00 1768  
## woodyCov 0.06 0.00 0.05 0.06 1.01 2335  
## fireFreq2015 -0.08 0.00 -0.08 -0.07 1.00 2920  
## elephant -0.08 0.00 -0.09 -0.08 1.00 3347  
## drought2015 0.08 0.00 0.08 0.08 1.00 2564  
## wetness2015 0.10 0.00 0.10 0.10 1.00 2989  
## RACresist 39.68 0.12 39.44 39.91 1.00 1459  
## soilSand:geologyGranite -0.05 0.00 -0.05 -0.04 1.00 1652  
## geologyGranite:soilClay 0.03 0.00 0.02 0.03 1.01 1321  
## geologyGranite:soilCarbon 0.05 0.00 0.04 0.05 1.00 1743  
## soilClay:woodyCov 0.03 0.00 0.03 0.03 1.00 1713  
## woodyCov:fireFreq2015 0.01 0.00 0.01 0.01 1.00 2760  
## woodyCov:elephant -0.01 0.00 -0.02 -0.01 1.00 3065  
## woodyCov:drought2015 -0.01 0.00 -0.02 -0.01 1.00 3113  
## woodyCov:wetness2015 0.00 0.00 -0.00 0.00 1.00 2933  
## Tail\_ESS  
## Intercept 1405  
## slope 1642  
## soilSand 1651  
## geologyGranite 1531  
## soilClay 1262  
## soilCarbon 1788  
## woodyCov 1540  
## fireFreq2015 1494  
## elephant 1660  
## drought2015 1441  
## wetness2015 1527  
## RACresist 1362  
## soilSand:geologyGranite 1466  
## geologyGranite:soilClay 1351  
## geologyGranite:soilCarbon 1486  
## soilClay:woodyCov 1662  
## woodyCov:fireFreq2015 1732  
## woodyCov:elephant 1593  
## woodyCov:drought2015 1143  
## woodyCov:wetness2015 1626  
##   
## Family Specific Parameters:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
## phi 489.76 5.96 478.10 501.37 1.01 1311 1087  
##   
## Draws were sampled using sampling(NUTS). For each parameter, Bulk\_ESS  
## and Tail\_ESS are effective sample size measures, and Rhat is the potential  
## scale reduction factor on split chains (at convergence, Rhat = 1).

summary(resil\_model\_rac)

## Family: beta   
## Links: mu = logit; phi = identity   
## Formula: resil ~ slope + soilSand \* geology + soilClay \* geology + soilCarbon \* geology + soilClay \* woodyCov + woodyCov \* fireFreq2015 + woodyCov \* elephant + woodyCov \* drought2015 + woodyCov \* wetness2015 + RACresil   
## Data: train\_df (Number of observations: 13121)   
## Draws: 4 chains, each with iter = 2000; warmup = 1500; thin = 1;  
## total post-warmup draws = 2000  
##   
## Population-Level Effects:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS  
## Intercept -0.53 0.00 -0.53 -0.52 1.00 1361  
## slope 0.03 0.00 0.02 0.03 1.01 2978  
## soilSand 0.01 0.00 0.01 0.02 1.00 1829  
## geologyGranite -0.16 0.00 -0.16 -0.15 1.00 1424  
## soilClay 0.14 0.00 0.14 0.15 1.01 931  
## soilCarbon -0.01 0.00 -0.01 -0.01 1.00 1807  
## woodyCov 0.01 0.00 0.01 0.01 1.00 2922  
## fireFreq2015 -0.04 0.00 -0.04 -0.03 1.00 3408  
## elephant -0.03 0.00 -0.03 -0.02 1.00 3308  
## drought2015 -0.03 0.00 -0.03 -0.02 1.00 3220  
## wetness2015 0.14 0.00 0.14 0.14 1.00 3521  
## RACresil 41.84 0.15 41.54 42.13 1.01 1429  
## soilSand:geologyGranite -0.00 0.00 -0.01 0.00 1.00 1717  
## geologyGranite:soilClay -0.11 0.00 -0.12 -0.10 1.00 1091  
## geologyGranite:soilCarbon 0.04 0.00 0.03 0.04 1.00 1724  
## soilClay:woodyCov -0.01 0.00 -0.01 -0.01 1.00 1446  
## woodyCov:fireFreq2015 0.01 0.00 0.01 0.01 1.00 2899  
## woodyCov:elephant -0.02 0.00 -0.02 -0.02 1.00 3067  
## woodyCov:drought2015 -0.03 0.00 -0.03 -0.03 1.00 3223  
## woodyCov:wetness2015 -0.02 0.00 -0.02 -0.02 1.00 2980  
## Tail\_ESS  
## Intercept 1232  
## slope 1191  
## soilSand 1416  
## geologyGranite 1268  
## soilClay 948  
## soilCarbon 1259  
## woodyCov 1754  
## fireFreq2015 1557  
## elephant 1661  
## drought2015 1573  
## wetness2015 1703  
## RACresil 1237  
## soilSand:geologyGranite 1512  
## geologyGranite:soilClay 1246  
## geologyGranite:soilCarbon 1279  
## soilClay:woodyCov 1516  
## woodyCov:fireFreq2015 1640  
## woodyCov:elephant 1847  
## woodyCov:drought2015 1753  
## woodyCov:wetness2015 1804  
##   
## Family Specific Parameters:   
## Estimate Est.Error l-95% CI u-95% CI Rhat Bulk\_ESS Tail\_ESS  
## phi 414.03 4.95 404.75 423.56 1.00 1216 1371  
##   
## Draws were sampled using sampling(NUTS). For each parameter, Bulk\_ESS  
## and Tail\_ESS are effective sample size measures, and Rhat is the potential  
## scale reduction factor on split chains (at convergence, Rhat = 1).