**Supporting Information** *–* *Behavioral flexibility facilitates use of spatial and temporal refugia during extreme winter weather*

**Camera trap sampling**

We collected spatially-indexed and time-stamped deer detections from ***Snapshot Wisconsin***,a wildlife observation network powered by community scientists and camera traps (Townsend et al., 2020). Snapshot Wisconsin has two goals: to support wildlife management and increase public engagement in wildlife science. Operated by the Wisconsin Department of Natural Resources (WDNR), Snapshot Wisconsin launched in 2016 and has since generated >50 million photos from >2,000 unique camera locations.

The WDNR solicits volunteers within USPLSS quarter-townships (4.8 x 4.8 km resolution), which serve as sampling units for Snapshot Wisconsin. The WDNR prioritizes applicants from unoccupied cells. The WDNR provides each host with a Bushnell Trophy Cam (Overland Park, Kansas) with fixed settings such that the cameras record a 3-image sequence when triggered, with a 15-second gap between triggers. The hosts deploy their camera within their assigned quarter-township, typically on private land but occasionally on public land. They place their cameras along wildlife trails or water features at least 100 m from roads or buildings. Hosts mount their camera 0.75-0.9 m from the ground and 3-4.5 m from the target, positioning the camera such that it aims the target at a diagonal angle and faces north to avoid false triggers from sunrise or sunset. Bait or lures are not used. Finally, hosts are instructed to clear vegetation between the camera and the target that may obstruct detection of animals. The hosts check their cameras every 1-3 months and upload the images to a web repository. Image classification takes three forms: 1) individual classifications by camera hosts, 2) consensus-based classification by volunteers on the Zooniverse crowdsourcing platform, 3) and expert classification of subsets of images to quantify accuracy of volunteer classifications (Anhalt-Depies et al., in prep).

**Temporal refugia analysis**

*Fitting diel activity curves and estimating activity proportions*

We defined the temporal scope of the study (i.e. 15 December - 10 March in the winters of 2017-2018 and 2018-2019) and filtered cameras that were deployed and active during those time windows. We subsequently filtered photos classified as deer by date, selecting only photos from the time windows. We converted the timestamps from the photos to solar time to account for changing day length over the course of the study period (Vazquez et al., 2019). For each camera location and date, we obtained the time of night end, morning golden hour end, evening golden hour start, and night start from the suncalc R package (Thieurmel & Elmarhraoui, 2019) and converted the times to solar time using the activity R package (Rowcliffe et al., 2014; Vazquez et al., 2019).

We then used a kernel density model from the activity package to fit a diel activity curve from the timestamps for each day. We then divided the activity curve into four periods, the endpoints of which we defined based on the suncalc R package: night (defined as all times prior to night end or after night start), dawn (between night end and morning golden hour end), day (between morning golden hour end and evening golden hour start), and dusk (between evening golden hour start and night start).

For more details about fitting diel activity curves and calculating the proportion of deer activity during night, dawn, day, and dusk, please see Script 1, which simulates diel detection data and calculates the activity proportions as we did in the analysis. Actual data cannot be shared because the coordinates of volunteer cameras are private. The minimum number of deer detections in a day that we used was 373 and the maximum was 2,606.

*Beta regression*

The model structure is shown below, with *i* indexing day and *j* indexing winter:

yij ~ dbeta(αij, βij) Eqn 1

Where yij is the proportion of activity during night, dawn, day, or dusk periods, and αij and βij are shape parameters for the beta distribution. The shape parameters are modeled as follows:

αij = μij\*φ Eqn 2

βij = (1 - μij)\*φ Eqn 3

Where μij is the expected value and φ is the precision (Douma & Weedon, 2019). μij is modeled as a linear function of predictors on the logit scale.

logit(μij) = **γ\*Xij** + εj Eqn 4

Where **Xij** is the design matrix of predictors, **γ** the corresponding vector of coefficients, and εj is a random intercept for winter. We modeled εj ~ *Normal(0,* τ*)* with τ ~ *Gamma(0.1, 0.1)*. We used a *Gamma(0.1, 0.1)* prior for φ and weakly informative *~Logistic(0, 1)* priors for the coefficients **γ** (Banner et al., 2020; Hobbs & Hooten, 2015)**.** We ran the models in the R package NIMBLE with 5,000 MCMC iterations (discarding all but the final 1,000 iterations as burn-in) and 3 chains. We assessed convergence of all model parameters by visually inspecting traceplots and with the Gelman-Rubin convergence diagnostic Rhat (Brooks & Gelman, 1998), considering parameters with Rhat < 1.1 to be converged. For data and code to fit the models (actual data provided, since the observed proportions of activity during each period are removed from private camera coordinates), please see Script 2.

**Spatial refugia analysis**

We used occupancy models to make inference about daily deer activity at camera locations. For each day, we created a binary detection history (i.e. a matrix with rows representing camera locations and columns representing replicate sampling occasions, of which there were three per day). Within the detection histories, we assigned a 1 for cameras that detected a deer during a replicate sampling occasion, a 0 for cameras that did not detect a deer during a replicate sampling occasion, and NA for cameras that were not active on a particular day. Occupancy models are typically used to make inference about occurrence of a species over some longer temporal span (e.g., a breeding season). Given the extreme temporal granularity (daily), we interpret the latent state *z* as the presence or absence of deer activity at the camera location on a given day and the parameter ψ as the “probability of activity” at the camera location. The structure is such that each day is modeled via a single-season occupancy model; changes in activity state between days is modeled via dynamic weather predictors (rather than latent site extinction and colonization probabilities, as in multiseason occupancy models; MacKenzie et al., 2018). The structure of the occupancy model is provided below. First, we provide the structure of the model for the ecological process (i.e. deer activity at camera locations) with *i* indexing camera location, *j* indexing day, and *k* indexing year.

zi,j,k ~ Bernoulli(ψi,j,k) Eqn 5

logit(ψi,j,k) = **γXi,j,k** + θi + εk Eqn 6

Where zi,j,kis the latent activity state and ψi,j,k is probability of activity. We modeled ψi,j,k as a function of noncollinear predictors **Xi,j,k** and corresponding vector of coefficients **γ. Xi,j,k** consisted of four landscape predictors (Supplemental Figure S3), two weather predictors (temperature and CWSI; we omitted snow depth due to extreme collinearity with CWSI), a temperature-winter severity interaction, and landscape-weather interactions. We scaled all predictors to have a mean of zero and a standard deviation of one (Schielzeth, 2010). We used weakly informative *~Logistic(0, 1)* priors for the coefficients **γ** (Banner et al., 2020; Hobbs & Hooten, 2015). Finally, θi is a spatial random effect and εk a random effect of year. We used a penalized spline with 20 knot locations distributed across the state for the spatial random effect θi (Crainiceanu et al., 2005) and a *~Normal(0, τ)* prior for εk, with a *~Gamma(1, 2)* hyperprior for τ.

We treated the observational submodel as follows (*i*,*j,* and *k* still indexing camera, day, and year, respectively; *n* indexes replicate survey period):

yi,j,k,n ~ Bernoulli(zi,j,k,\*pi,j,k,n) Eqn 7

Where yi,j,k,n is a binary record of whether or not deer were detected, zi,j,k is the latent activity state, and pi,j,k,n is detection probability. Because of the extreme temporal granularity of our analysis, we interpret detection probability as the probability of detecting deer during a given 8-hour replicate sampling occasion, *provided deer are active at the camera location on a given day*. We allowed detection probability to vary by site, day, and sampling occasion with one covariate and a random effect:

logit(pi,j,k,n) = α\*periodi,j,k,n + εi,j,k Eqn 8

The covariate (periodi,j,k,n) was a binary indicator of whether the replicate survey period was wholly diurnal (i.e., 08:00-16:00) or contained night/crepuscular hours (i.e., 00:00-08:00 or 16:00-00:00). We used a *~Logistic(0, 1)* prior for the coefficient α and a *~Normal(0, τ)* prior for the random effect εi,j, with a *~Gamma(1, 2)* hyperprior for *τ*.

We ran the models in the R package NIMBLE; we used 10,000 MCMC iterations (discarding all but the final 1,000 iterations as burn-in) and 3 chains. We assessed convergence of all model parameters by visually inspecting traceplots and with the Gelman-Rubin convergence diagnostic Rhat (Brooks & Gelman, 1998), considering parameters with Rhat < 1.1 to be converged. If model parameters were not converged, we extended the MCMC chains until they converged. Please see Script 3 for code to fit the model (data provided).

**Literature cited**

Banner, K. M., Irvine, K. M., & Rodhouse, T. (2020). The Use of Bayesian Priors in Ecology: The Good, The Bad, and The Not Great. *Methods in Ecology and Evolution*, *11*(8), 882–889. https://doi.org/10.1111/2041-210X.13407

Brooks, S. P., & Gelman, A. (1998). General Methods for Monitoring Convergence of Iterative Simulations. *Journal of Computational and Graphical Statistics*, *7*(4), 434–455. https://doi.org/10.1080/10618600.1998.10474787

Crainiceanu, C., Ruppert, D., & Wand, M. (2005). Bayesian Analysis for Penalized Spline Regression Using WinBUGS. *Journal of Statistical Software*, *14*(14), 1–24.

Douma, J. C., & Weedon, J. T. (2019). Analysing continuous proportions in ecology and evolution: A practical introduction to beta and Dirichlet regression. *Methods in Ecology and Evolution*, *10*(9), 1412–1430. https://doi.org/10.1111/2041-210X.13234

Hobbs, N. T., & Hooten, M. (2015). *Bayesian Models: A Statistical Primer for Ecologists*. Princeton University Press.

MacKenzie, Nichols, J., Royle, J., Pollock, K., Bailey, L., & Hines, J. (2018). *Occupancy Estimation and Modeling: Inferring Patterns and Dynamics of Species Occurrence* (2nd ed.). Academic Press.

Rowcliffe, J. M., Kays, R., Kranstauber, B., Carbone, C., & Jansen, P. A. (2014). Quantifying levels of animal activity using camera trap data. *Methods in Ecology and Evolution*, *5*(11), 1170–1179. https://doi.org/10.1111/2041-210X.12278

Schielzeth, H. (2010). Simple means to improve the interpretability of regression coefficients. *Methods in Ecology and Evolution*, *1*(2), 103–113. https://doi.org/10.1111/j.2041-210X.2010.00012.x

Thieurmel, B., & Elmarhraoui, A. (2019). *suncalc: Compute Sun Position, Sunlight Phases, Moon Position and Lunar Phase* (0.5.0) [Computer software]. https://CRAN.R-project.org/package=suncalc

Townsend, Clare, J., Liu, N., Stenglein, J. L., Anhalt-Depies, C., Deelen, T. R. V., Gilbert, N. A., Singh, A., Martin, K. J., & Zuckerberg, B. (2020). Integrating remote sensing and jurisdictional observation networks to improve the resolution of ecological management. *BioRxiv*. https://doi.org/10.1101/2020.06.08.140848

Vazquez, C., Rowcliffe, J. M., Spoelstra, K., & Jansen, P. A. (2019). Comparing diel activity patterns of wildlife across latitudes and seasons: Time transformations using day length. *Methods in Ecology and Evolution*, *10*(12), 2057–2066. https://doi.org/10.1111/2041-210X.13290

**Supplemental Figures**

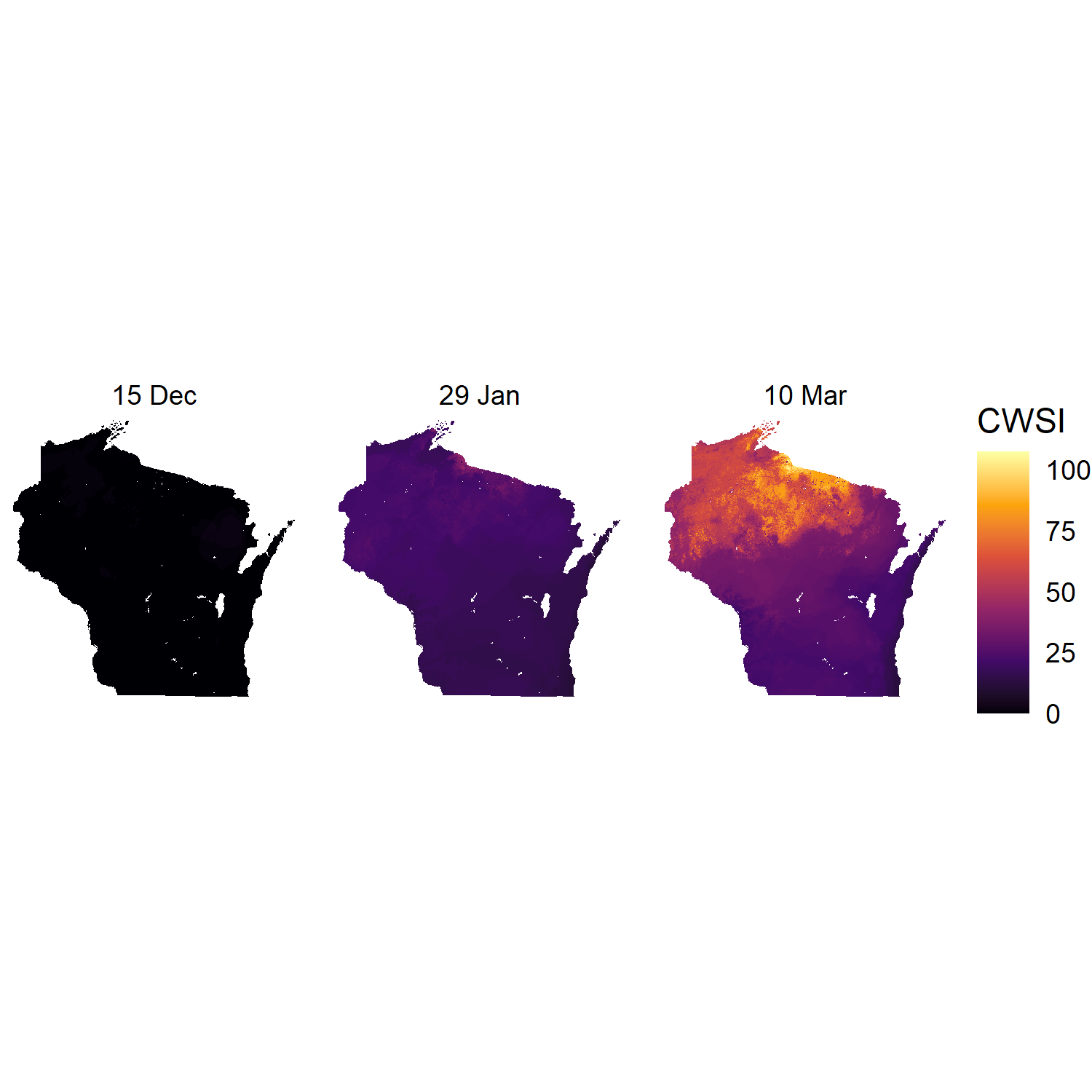


Figure S1. Accumulation of cumulative winter severity index (CWSI) values over one winter.

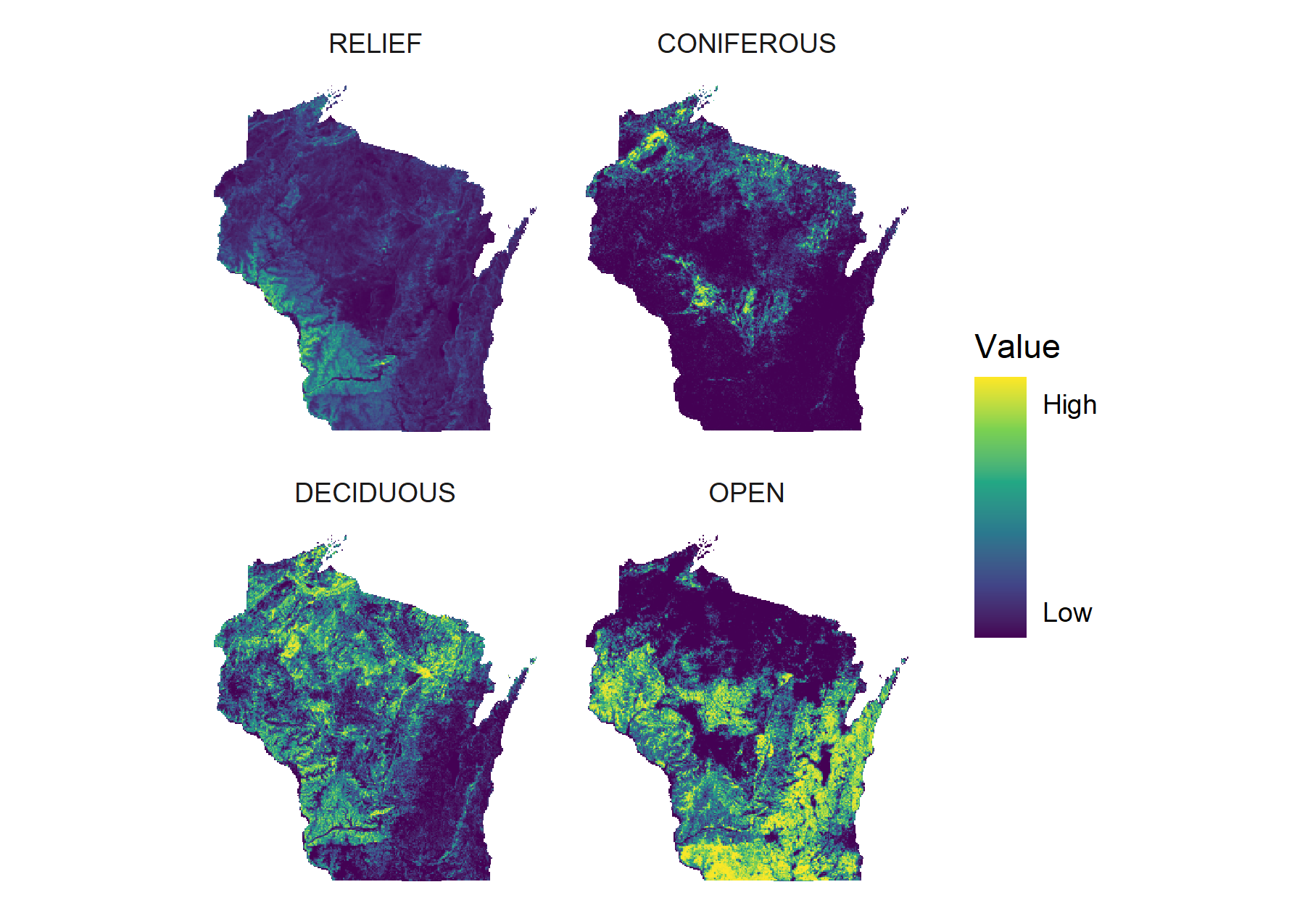


Figure S2. The four landscape predictors that we hypothesized would be relevant to winter deer spatial activity. Here, all predictors are scaled to fall in the 0-1 range for visualization.

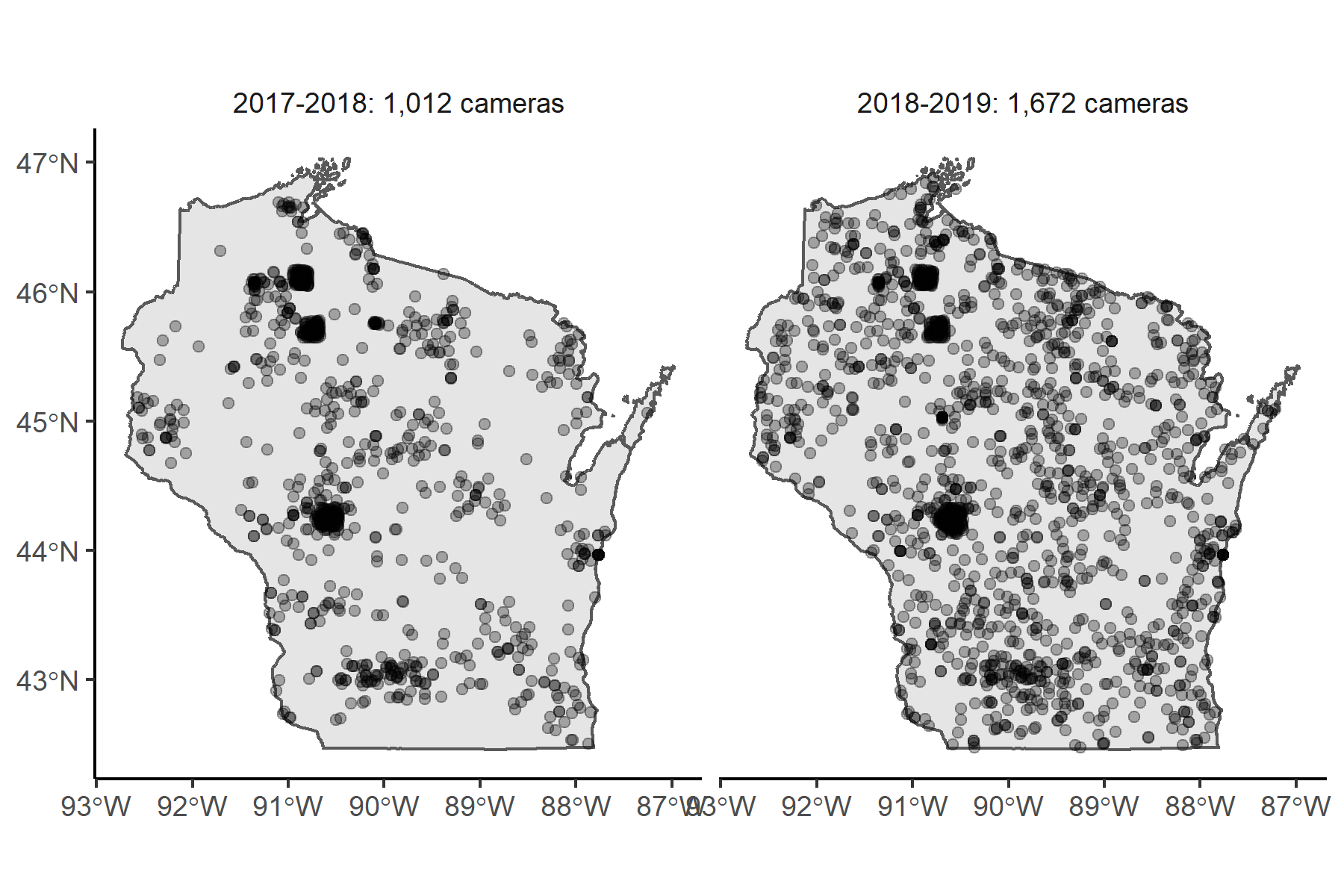
****

Figure S3. Locations of Snapshot Wisconsin cameras during the two winters included in the analysis.

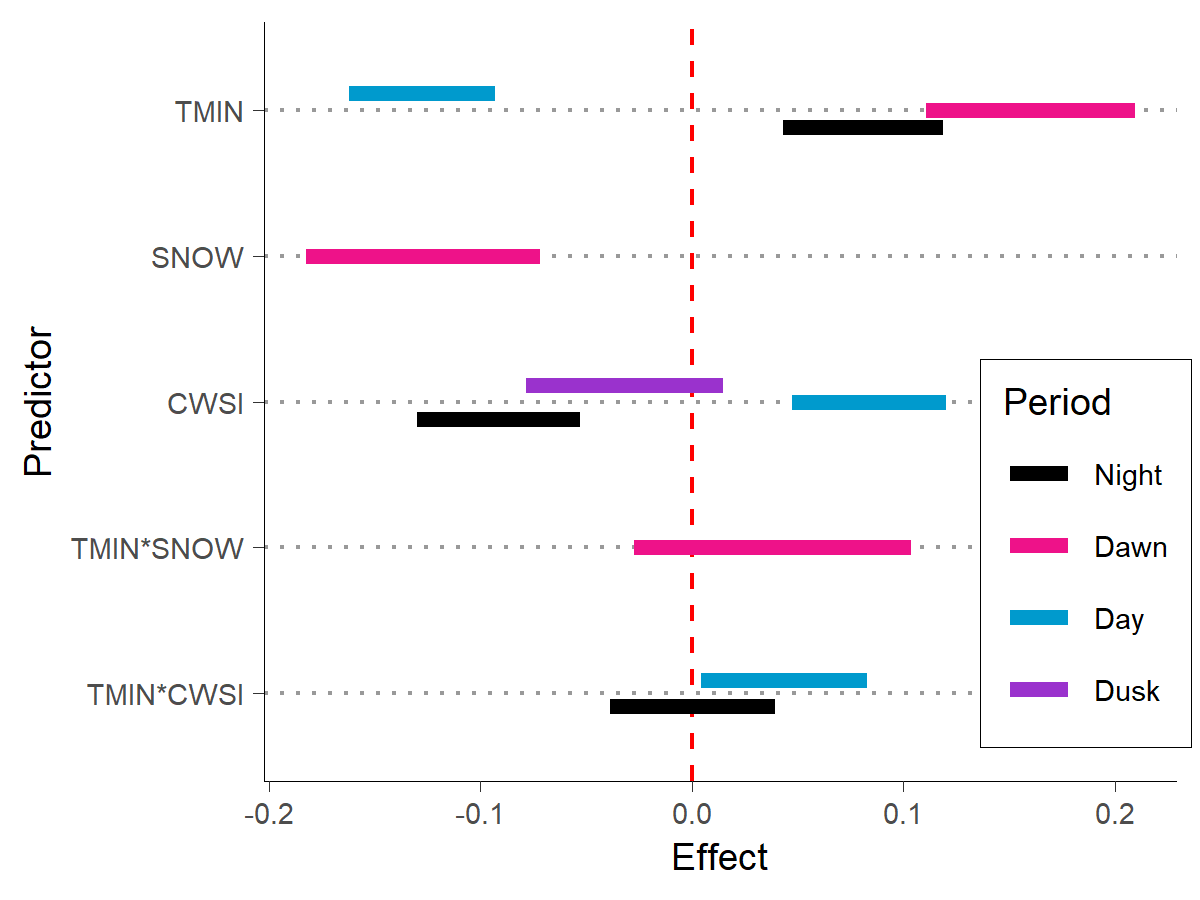


Fig. S4. Effects of the three weather predictors (TMIN, minimum temperature; SNOW, snow depth; CWSI, cumulative winter severity index) and their interactions on the proportions of night, dawn, day, and dusk activity from the top-ranked model for each period (see Table 1 of the main text). The bars represent the 95% credible intervals of the coefficients and the colors represent the four periods.

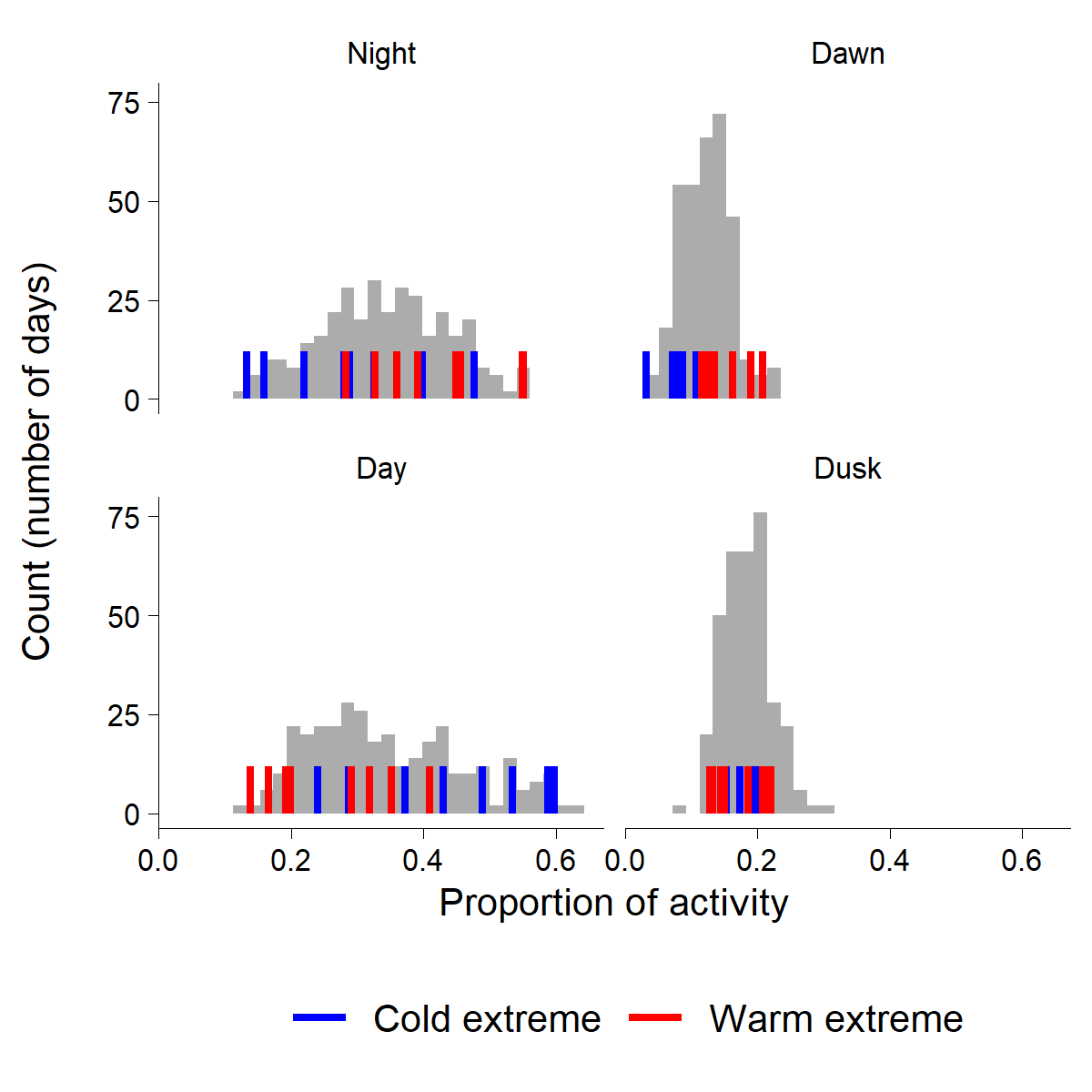


Figure S5. Proportion of activity during night, dawn, day, and dusk periods for the coldest and warmest 5% of days (colored lines) overlaid on the distribution of activity proportions for all days (gray histograms).

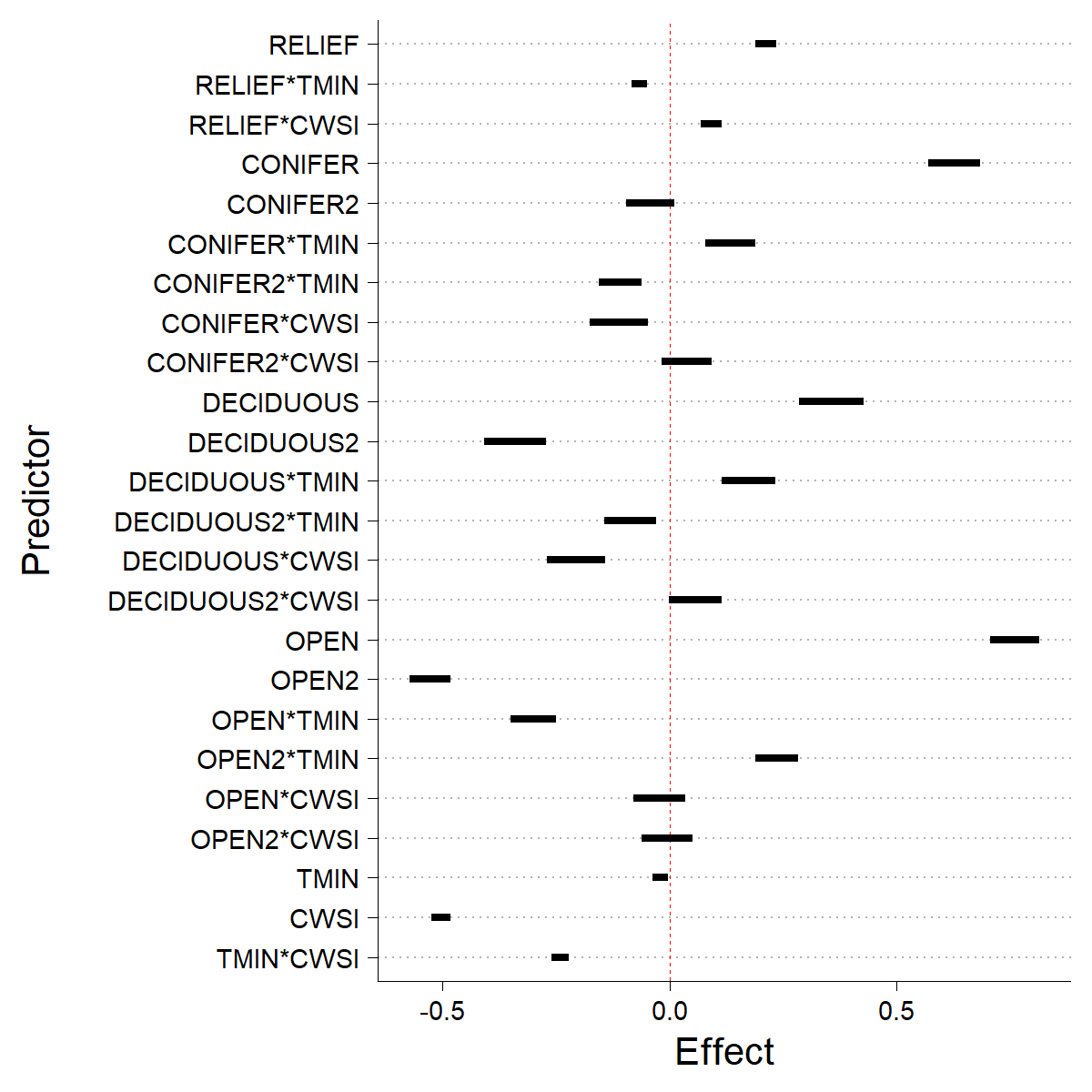


Figure S6. Effects of landscape predictors, weather predictors, and their interactions on daily spatial activity of deer. Landscape-scale predictors were measured within a 1-km radius of camera locations. Each bar represents the 95% credible interval for the coefficient, and the red dashed line indicates no relationship between the predictor and activity.

***Script 1 - Calculate proportions of activity during night, dawn, day, and dusk***

Code and data available at <https://github.com/n-a-gilbert/whiplash_supplement>

# 24 November 2020

# Snapshot Wisconsin data cannot be published to protect volunteer privacy

# specifically, lat-longs associated with cameras cannot be published

# So here's code to simulate time-of-day detection data,

# fit it with a nonparametric kernel density model,

# and calculate the proportion of activity within night, dawn, day, dusk

library(here)

library(sf)

library(tidyverse)

library(lubridate)

library(suncalc)

library(activity)

setwd(here::here("data"))

# shapefile of Wisconsin...for generating random points

wi <- sf::st\_read("WI\_state\_outline.shp") %>%

sf::st\_transform(crs = 4326)

# how many random points to simulate

npoints <- 10

# how many detections per site to simulate

ndets <- 10

# generate random points within Wisconsin

random\_points <- sf::st\_sample(wi, size = npoints) %>%

sf::st\_as\_sf(.) %>%

dplyr::mutate(XCOORD = sf::st\_coordinates(.)[,1],

YCOORD = sf::st\_coordinates(.)[,2]) %>%

sf::st\_drop\_geometry(.)

# sanity check - do we get random points showing up within Wisconsin?

ggplot() +

geom\_sf(data = wi, aes(geometry = geometry)) +

geom\_point(data = random\_points, aes(x = XCOORD, y = YCOORD))

# start date

start <- lubridate::ymd("2017-12-15")

# end date

end <- lubridate::ymd("2017-12-20")

dates <- seq(from = start, to = end, by = 1)

# create a dataframe with the unique points & dates of survey

points\_dates <- as\_tibble(random\_points[rep(1:nrow(random\_points),

times = length(dates)),]) %>%

arrange(XCOORD, YCOORD) %>%

add\_column(date = rep(dates, npoints))

# create a dataframe with the points and simulated detections

random\_detections <- as\_tibble(random\_points[rep(1:nrow(random\_points),

times = ndets),]) %>%

arrange(XCOORD, YCOORD) %>%

# here, we're simulating detection date-times, 10 per site

add\_column(detection = as\_datetime(start)

+ runif(ndets\*npoints, 0, as.numeric(difftime(end,

start,

unit = "sec")))) %>%

mutate(date = as\_date(detection))

# get times of relevant times of day for defining periods for each site

suntimes <- suncalc::getSunlightTimes(data = dplyr::select(points\_dates,

date,

lat = YCOORD,

lon = XCOORD),

keep = c("nightEnd",

"goldenHourEnd",

"goldenHour",

"night")) %>%

mutate\_at(vars(nightEnd:night), function(x) with\_tz(x, "America/Chicago")) %>%

rename(YCOORD = lat,

XCOORD = lon) %>%

# add on the simulated detections

full\_join(random\_detections) %>%

filter(!is.na(detection)) %>%

# convert everything to radians

mutate(detection = activity::solartime(detection, YCOORD, XCOORD, tz = 6,

format = "%Y-%m-%d %H:%M:%S")$solar,

nightEnd = activity::solartime(nightEnd, YCOORD, XCOORD, tz = 6,

format = "%Y-%m-%d %H:%M:%S")$solar,

goldenHourEnd = activity::solartime(goldenHourEnd, YCOORD, XCOORD, tz = 6,

format = "%Y-%m-%d %H:%M:%S")$solar,

goldenHour = activity::solartime(goldenHour, YCOORD, XCOORD, tz = 6,

format = "%Y-%m-%d %H:%M:%S")$solar,

night = activity::solartime(night, YCOORD, XCOORD, tz = 6,

format = "%Y-%m-%d %H:%M:%S")$solar)

# empty list for results

period\_proportions <- list()

for(i in 1:(length(dates) - 1)){

current\_date <- dates[i]

# fit activity model to simulated detection data

act <- activity::fitact(filter(suntimes,

date == current\_date) %>% pull(detection))

# grab just the probability density function of the activity model

pdf <- act@pdf

# points at which PDF is evaluated

xx <- pdf[,1]

# binwdith

dx <- xx[2L] - xx[1L]

# probability density for each bin

yy <- pdf[,2]

# get the mean period endpoints for points, since activity model

# pools detection data from all points

period\_times <- filter(suntimes, date == current\_date) %>%

dplyr::select(nightEnd:night) %>%

summarise\_all(mean)

# proportion dawn-active

# dawn is the time between night end and end of morning "golden hour"

dawn <- sum(yy[(xx > period\_times$nightEnd & xx < period\_times$goldenHourEnd)])\*dx

# proportion dusk-active

# dusk is time between start of evening "golden hour" and night start

dusk <- sum(yy[(xx > period\_times$goldenHour & xx < period\_times$night)])\*dx

#proportion diurnal

#day is between end of morning golden hour and start of evening golden hour

diurnal <- sum(yy[(xx > period\_times$goldenHourEnd & xx < period\_times$goldenHour)])\*dx

#proportion nocturnal

# night is between midnight and night end plus nigth start until midnight

nocturnal <- sum(yy[(xx > 0 & xx < period\_times$nightEnd) | (xx > period\_times$night)])\*dx

period\_proportions[i] <- list(tibble(

date = current\_date,

pdawn = dawn,

pdusk = dusk,

pdiurnal = diurnal,

pnocturnal = nocturnal))

}

# unpack list into a dataframe

# each day of survey is a row

# columns for proportions of activity within dawn, dusk, day, and night periods

( period\_proportions\_df <- bind\_rows(period\_proportions) )

***Script 2 - beta regression for activity proportions***

Code and data available at <https://github.com/n-a-gilbert/whiplash_supplement>

# code to run beta regression analysis looking at how proportion of deer activity

# during night, dawn, day, & dusk periods varies as a function of weather predictors

# These models are written in NIMBLE, a language derived from BUGS.

# Unlike previous programs (WinBUGS, JAGS), NIMBLE models are programmable objects

# in R, but are compiled in C++ for speed. Therefore, prior to installing NIMBLE,

# you must have Rtools installed so the code can be compiled.

# Please see the NIMBLE website for instructions: https://r-nimble.org/download

library(here)

library(tidyverse)

library(nimble)

library(coda)

setwd(here::here("data"))

# season: indicates which winter (1 = 2017-2018, 2 = 2018-2019)

# date: date

# pdawn: proportion of activity occuring during dawn period

# pdusk: proportion of activity occuring during dusk period

# pdiurnal: proportion of activity occurring during day period

# pnocturnal: proportion of activity occuring during night period

# CWSI: cumulative winter severity index, statewide average

# TMIN: daily minimum temperature, statewide average

# SNOW: daily snow depth, statewide average

dat <- readr::read\_csv("activity\_proportions\_predictors\_v01.csv") %>%

rename(pdawn = Dawn, pdusk = Dusk, pdiurnal = Day, pnocturnal = Night)

# inspect

glimpse(dat)

# reformat data so each year of data has a separate column

act <- dat %>%

dplyr::mutate\_at(vars(CWSI:SNOW), function(x) as.vector(scale(x))) %>%

group\_by(season) %>%

mutate(id = row\_number()) %>%

dplyr::select(-date) %>%

pivot\_wider(names\_from = season, values\_from = pdawn:SNOW)

# following?

glimpse(act)

# this is a key that will help us track through the different models

key <- tibble(

index = 1:20,

response = sort(rep(c("pnocturnal", "pdawn", "pdiurnal", "pdusk"), 5)), # which period

name = rep(paste0("mod", 1:5), 4),

npreds = rep(c(2, 2, 1, 1, 1), 4), # how many predictors

pred1 = rep(c("TMIN", "TMIN", "TMIN", "SNOW", "CWSI"), 4), # which predictor?

pred2 = rep(c("CWSI", "SNOW", NA, NA, NA), 4)) # which predictor? if there's more than 1 in model

# list where we'll send the results

result\_list <- list()

for(i in 1:nrow(key)){

# slightly separate model structure for 1 versus 2 predictors

if(key[[i, "npreds"]] == 1){

# package up data for nimble model

data <- list(response = unname(as.matrix(dplyr::select(act,

starts\_with(key[[i, "response"]])))),

pred1 = unname(as.matrix(dplyr::select(act,

starts\_with(key[[i, "pred1"]])))))

# package up constants; for indexing purposes

constants <- list(pdays = nrow(data$response),

nseason = ncol(data$response))

# here's the code for the nimble model

modelCode <- nimble::nimbleCode( {

# likelihood

for(i in 1:pdays){

for(j in 1:nseason){

response[i, j] ~ dbeta(alpha[i,j], beta[i,j])

# link function

alpha[i,j] <- mu[i,j]\*phi

beta[i,j] <- (1 - mu[i,j])\*phi

# linear predictor with random effect of year

logit(mu[i,j]) <- b1\*pred1[i,j] + eps[j]

}}

# priors

# random effect of year; gives each season a different intercept

for(j in 1:nseason){

eps[j] ~ dnorm(0, tau)

}

tau ~ dgamma(0.1, 0.1)

phi ~ dgamma(0.1, 0.1)

# weakly informative prior for predictor coefficient

b1 ~ dlogis(0, 1)

}) # end of model code

# parameters we're gonna monitor

keepers <- c("b1",

"tau",

"eps")

# initial values

inits <- function(){

list(phi = rgamma(1, 0.1, 0.1),

b1 = runif(1, -1, 1),

tau = rgamma(1, 0.1, 0.1),

eps = rnorm(2, 0, 1))

}

# getting things ready to go

mod <- nimble::nimbleModel(code = modelCode,

constants = constants,

data = data,

inits = inits())

conf <- configureMCMC(mod,

enableWAIC = TRUE)

conf$addMonitors(keepers)

rmcmc <- buildMCMC(conf)

cmodel <- compileNimble(mod)

cmcmc <- compileNimble(rmcmc, project = cmodel)

# run it

out <- runMCMC(cmcmc,

niter = 5000,

nburnin = 4000,

nchains = 3,

samples = TRUE,

WAIC = TRUE)

# put results from all chains into one array

samples <- do.call(rbind, out$samples)

# convert to mcmc for calculating rhat

samples\_mcmc <- as.mcmc(lapply(out$samples, mcmc))

# assess convergence with rhat

rhat <- as\_tibble(coda::gelman.diag(samples\_mcmc)$psrf,

rownames = "parameter") %>%

rename(rhat = `Point est.`) %>%

dplyr::select(parameter, rhat) %>%

dplyr::filter(grepl("^b", parameter)) %>%

add\_column(pred = key[[i, "pred1"]])

# package up the result as a dataframe

result\_list[[i]] <- as\_tibble(samples) %>%

dplyr::select(., starts\_with("b")) %>%

pivot\_longer(cols = 1,

names\_to = "parameter",

values\_to = "values") %>%

group\_by(parameter) %>%

summarise(mean = mean(values),

sd = sd(values),

lower = quantile(values, c(0.025)),

upper = quantile(values, c(0.975))) %>%

full\_join(rhat) %>%

add\_column(waic = out$WAIC,

response = key[[i, "response"]],

index = key[[i, "index"]]) %>%

dplyr::select(index, response, parameter, pred, mean:rhat, waic)

# same thing over again, but for models with multiple predictors

} else {

data <- list(response = unname(as.matrix(dplyr::select(act,

starts\_with(key[[i, "response"]])))),

pred1 = unname(as.matrix(dplyr::select(act,

starts\_with(key[[i, "pred1"]])))),

pred2 = unname(as.matrix(dplyr::select(act,

starts\_with(key[[i, "pred2"]])))))

constants <- list(pdays = nrow(data$response),

nseason = ncol(data$response))

modelCode <- nimble::nimbleCode( {

for(i in 1:pdays){

for(j in 1:nseason){

response[i, j] ~ dbeta(alpha[i,j], beta[i,j])

alpha[i,j] <- mu[i,j]\*phi

beta[i,j] <- (1 - mu[i,j])\*phi

logit(mu[i,j]) <- b1\*pred1[i,j] + b2\*pred2[i, j] + b3\*pred1[i,j]\*pred2[i, j] + eps[j]

}}

for(j in 1:nseason){

eps[j] ~ dnorm(0, tau)

}

tau ~ dgamma(0.1, 0.1)

phi ~ dgamma(0.1, 0.1)

b1 ~ dlogis(0, 1)

b2 ~ dlogis(0, 1)

b3 ~ dlogis(0, 1)

})

keepers <- c("b1",

"b2",

"b3",

"tau",

"eps")

inits <- function(){

list(phi = rgamma(1, 0.1, 0.1),

b1 = runif(1, -1, 1),

b2 = runif(1, -1, 1),

b3 = runif(1, -1, 1),

tau = rgamma(1, 0.1, 0.1),

eps = rnorm(2, 0, 1))

}

mod <- nimble::nimbleModel(code = modelCode,

constants = constants,

data = data,

inits = inits())

conf <- configureMCMC(mod,

enableWAIC = TRUE)

conf$addMonitors(keepers)

rmcmc <- buildMCMC(conf)

cmodel <- compileNimble(mod)

cmcmc <- compileNimble(rmcmc, project = cmodel)

out <- runMCMC(cmcmc,

niter = 5000,

nburnin = 4000,

nchains = 3,

samples = TRUE,

WAIC = TRUE)

samples <- do.call(rbind, out$samples)

samples\_mcmc <- as.mcmc(lapply(out$samples, mcmc))

rhat <- as\_tibble(coda::gelman.diag(samples\_mcmc)$psrf,

rownames = "parameter") %>%

rename(rhat = `Point est.`) %>%

dplyr::select(parameter, rhat) %>%

dplyr::filter(grepl("^b", parameter)) %>%

add\_column(pred = c(key[[i, "pred1"]],

key[[i, "pred2"]],

paste(key[[i, "pred1"]], key[[i, "pred2"]], sep = "\*")))

result\_list[[i]] <- as\_tibble(samples) %>%

dplyr::select(., starts\_with("b")) %>%

pivot\_longer(cols = 1:3,

names\_to = "parameter",

values\_to = "values") %>%

group\_by(parameter) %>%

summarise(mean = mean(values),

sd = sd(values),

lower = quantile(values, c(0.025)),

upper = quantile(values, c(0.975))) %>%

full\_join(rhat) %>%

add\_column(waic = out$WAIC,

response = key[[i, "response"]],

index = key[[i, "index"]]) %>%

dplyr::select(index, response, parameter, pred, mean:rhat, waic)

}

}

# one sleek data frame of results

result\_df <- bind\_rows(result\_list)

# what is the top-ranked model for each period?

full\_join(key, result\_df) %>%

dplyr::select(index, response, npreds, pred1, pred2, waic) %>%

dplyr::mutate(pred3 = hablar::if\_else\_(!is.na(pred2),

paste(pred1, pred2, sep = "\*"),

NA)) %>%

dplyr::select(index:pred2, pred3, waic) %>%

distinct(.) %>%

group\_by(response) %>%

arrange(response, waic) %>%

slice(1)

# look at effects of predictors/interactions from top model

full\_join(key, result\_df) %>%

dplyr::mutate(pred3 = hablar::if\_else\_(!is.na(pred2),

paste(pred1, pred2, sep = "\*"),

NA)) %>%

# dplyr::select(index:pred2, pred3, waic) %>%

# distinct(.) %>%

dplyr::select(response, pred1, pred2, pred3, pred, mean, lower, upper, waic) %>%

group\_by(response) %>%

arrange(response, waic) %>%

filter(waic == min(waic)) %>%

pivot\_longer(pred1:pred3, names\_to = "junk", values\_to = "name") %>%

filter(name == pred) %>%

filter(!is.na(name)) %>%

ggplot(aes(x = mean, y = name, color = response)) +

geom\_errorbar(aes(xmin = lower, xmax = upper), size = 2,

position = position\_dodge(width = 0.4),

width = 0) +

geom\_vline(xintercept = 0, color = "red")

# save results if you want

# setwd(here::here("results"))

# write\_csv(result\_df, "beta\_regression\_result\_summary\_all.csv")

***Script 3 - example occupancy model for habitat use analysis***

Code and data available at <https://github.com/n-a-gilbert/whiplash_supplement>

library(here)

library(tidyverse)

library(nimble)

library(parallel)

library(coda)

setwd(here::here("data"))

load("whiplashOccModelDataConstants\_v02.RData")

#### DATA ####

# the list named "data" contains 5 matrices

str(data)

# the first is called "covariate"

# these are the static landscape predictors, measured within a 1100-m buffer of camera locations

# rows are sites, columns are the different predictors

# they are, in order: aspect, deciduous, deciduous^2, elevSD, open, open^2, and shelter

# aspect is the proportion of south-facing slopes, calculated from a digital elevation model

# deciduous is the proportion of deciduous forest in the landscape

# deciduous^2 is just deciduous squared (for quadratic effect)

# elevSD is topographic relief, calculated as the standard deviation of elevation

# open is the proportion of "open" (cropland and grassland) habitat

# open^2 is just open squared

# shelter is the proportion of shelter habitat, defined as coniferous, mixed forest and wooded wetlands

# all predictors are already scaled

# inspect distribution of predictors

data$covariate %>%

as\_tibble() %>%

setNames(., c("relief", "deciduous", "deciduous2", "open", "open2", "conifer", "conifer2")) %>%

mutate(id = row\_number()) %>%

pivot\_longer(relief:conifer2, names\_to = "predictor", values\_to = "value") %>%

ggplot(., aes(x = value)) +

geom\_histogram() +

facet\_wrap(~predictor, scales = "free")

# the second matrix contains daily minimum temperature (tmin) and cumulative winter severity index (cwsi)

# rows are sites, columns days, matrix slices are tmin/cwsi

# columns 1-86 are the winter of 2017-2018, 87-172 are 2018-2019

# values are already scaled

# example: this shows tmin for one site over the course of the first winter

plot(data$weather[1,1:86,1,1])

# here's average tmin of all sites over the course of the first winter

plot(apply(data$weather[,,1,1], 2, mean)[1:86])

# here's average cwsi of all sites over the course of the winter - notice how it accumulates

plot(apply(data$weather[,,1,2], 2, mean)[1:86])

# the fourth matrix, "zmat" is for the spatial random effect.

# it provides, for each camera location (row), the distance to 20 knot locations (columns)

# values are already scaled

# the fifth matrix, "period", indicates for each camera location (row), day(column),

# whether a replicate survey period is 00:00-08:00 hrs (0),

# 08:00-16:00 hrs (1), or 16:00-00:00 hrs(0)

str(data$period)

# for one site on one day

plot(data$period[1,1,,1])

# finally, the sixth matrix, "y", is the deer detection record

# rows are sites, columns days, 3rd matrix slices are replicate survey periods, 4th matrix slice is year

# values can be 0 (deer not detected), 1 (deer detected), or NA (camera not active)

str(data$y)

# here's a detection history for a single site. this camera was active only in the first winter

data$y[1,,,]

#### CONSTANTS ####

# the object "constants" is a list of constants in the model

# these are used for indexing in the model

str(constants)

# nsite - number of camera locations

# nday - total number of days

# nsurvey - number of replicate surveys in one day

# nyear - number of years

# nknots - number of knots for spatial random effect

# nweather - number of weather predictors

# wstart - index for where weather predictors start in linear predictor of model

# wend - index for where weather predictors end in linear predictor of model

# tw - index for tmin\*cwsi interaction in linear predictor

# tintstart - index for where tmin\*landscape interactions start in linear predictor

# tintend - index for where tmin\*landscape interactions end in linear predictor

# wintstart - index for where cwsi\*landscape interactions start in linear predictor

# wintend - index for where cwsi\*landscape interactions end in linear predictor

# npred - total number of predictors in linear predictor

# .......................................................................

#### MODEL CODE ####

# .......................................................................

myCode <- nimbleCode({

# .............................................................

# PRIORS

# .............................................................

# priors for the predictor coefficients

# a logistic prior is just a normal distribution with fatter tails

# and more robust to logistic transformations

# a logistic(0,1) prior is weakly informative

# with mean = 0, if a predictor isn't related to the response, the coefficient will shrink to 0

for(i in 1:npred){

b[i] ~ dlogis(0, 1)

}

# priors for the spatial random effect penalized spline

# gamma is just a normal linear predictor

# sd\_bs is the precision

for(i in 1:nknots){

gamma[i] ~ dnorm(0, sd\_bs[i])

sd\_bs[i] ~ dgamma(1, 2)

}

# prior for the period predictor in the observation submodel

a1 ~ dlogis(0, 1)

# prior for random intercept for year

for(k in 1:nyear){

eps\_psi[k] ~ dnorm(0, sd\_eps\_psi)

}

# and precision

sd\_eps\_psi ~ dgamma(1, 2)

# prior for site/survey random effect

for(i in 1:nsite){

for(j in 1:nday){

for(k in 1:nyear){

eps\_p[i, j, k] ~ dnorm(0, sd\_eps\_p)

}

}

}

# and its precision

sd\_eps\_p ~ dgamma(1, 2)

# .............................................................

# LIKELIHOOD

# .............................................................

# state model - ecological process

for(i in 1:nsite){

for(j in 1:nday){

for(k in 1:nyear){

z[i, j, k] ~ dbern(psi[i, j, k])

logit(psi[i, j, k]) <- inprod(covariate[i,1:nland], b[1:nland]) + # landscape patterns

inprod(weather[i, j, k, 1:nweather], b[wstart:wend]) + # tmin & wsi

b[tw]\*weather[i,j, k, 1]\*weather[i, j, k, 2] + # tmin\*wsi interaction

inprod(covariate[i, 1:nland]\*weather[i, j, k, 1], b[tintstart:tintend]) + # landscape\*tmin interactions

inprod(covariate[i, 1:nland]\*weather[i, j, k, 2], b[wintstart:wintend]) + # landscape\*wsi interactions

inprod(zmat[i, 1:nknots], gamma[1:nknots]) + # spatial random effect

eps\_psi[k] # random intercept for year

# detection submodel - observation process

for(m in 1:nsurvey){

y[i, j, m, k] ~ dbern(z[i, j, k]\*p[i, j, m, k])

logit(p[i, j, m, k]) <- a1\*period[i, j, m, k] + eps\_p[i, j, k]

} # nsurvey

} # nyear

} # nday

} # nsite

}) # model

# initial values for running model

inits <- function() {list(p = array(0.5, dim = c(constants$nsite,

constants$nday,

constants$nsurvey,

constants$nyear)),

z = array(1, dim = c(constants$nsite, constants$nday, constants$nyear)),

y = array(1, dim = c(constants$nsite, constants$nday, constants$nsurvey, constants$nyear)),

b = runif(constants$npred, -1, 1),

a1 = runif(1, -1, 1),

gamma = runif(constants$nknots, -1, 1),

sd\_bs = runif(constants$nknots, 0, 2),

eps\_p = array(rnorm(constants$nsite\*constants$nday\*constants$nyear, 0, 2),

dim = c(constants$nsite, constants$nday, constants$nyear)),

sd\_eps\_p = runif(1, 0, 2),

eps\_psi = rnorm(constants$nyear, 0, 2),

sd\_eps\_psi = runif(1, 0, 2))}

# parameters to monitor

keepers <- c("b", "a1", "gamma", "eps\_psi")

# OPTION 1 - NOT RECOMMENDED - RUN CHAINS ONE AFTER ANOTHER

# this is a big model and takes a long time to run - better to

# do option #2 and run in parallel

# model <- nimbleModel(code = myCode,

# constants = constants,

# data = data,

# inits = inits())

# model$initializeInfo()

# Cmodel <- compileNimble(model)

# modelConf <- configureMCMC(model)

# modelConf$addMonitors(keepers)

# modelMCMC <- buildMCMC(modelConf)

# CmodelMCMC <- compileNimble(modelMCMC, project = model)

#

# out1 <- runMCMC(CmodelMCMC,

# nburnin = 9000,

# niter = 10000,

# nchains = 3)

# OPTION 2 (RECOMMENDED) - RUN CHAINS IN PARALLEL

# again, this is a big model, and especially if

# running chains in parallel, make sure you're equipped for the task

# I'm running this on a lab server

# Intel 4116 CPU @ 2.10 GHz, 768 GB RAM, 48 cores

# With that, it takes roughly 20 hours to run 3 chains for 10,000 iterations

# but, a laptop will probably choke trying to run this sucker

nc <- 3

nb <- 9000

ni <- nb + 1000

start <- Sys.time()

# makes a cluster with a node for each chain to run

cl <- makeCluster(nc)

parallel::clusterExport(cl, c("myCode",

"inits",

"data",

"constants",

"keepers",

"nb",

"ni"))

for(j in seq\_along(cl)){

set.seed(j)

init <- inits()

clusterExport(cl[j], "init")

}

out <- clusterEvalQ(cl, {

library(nimble)

library(coda)

model <- nimbleModel(code = myCode,

name = "myCode",

constants = constants,

data = data,

inits = init)

Cmodel <- compileNimble(model)

modelConf <- configureMCMC(model)

modelConf$addMonitors(keepers)

modelMCMC <- buildMCMC(modelConf)

CmodelMCMC <- compileNimble(modelMCMC, project = model)

out1 <- runMCMC(CmodelMCMC,

nburnin = nb,

niter = ni)

return(as.mcmc(out1))

})

# you'll want to shut down the cluster once you're done, but only do so

# once you're sure model is coverged and you don't need to extend chains

# stopCluster(cl)

end <- Sys.time()

end - start

# key for parameter names

key <- tibble(

parameter = c("a1",

paste0("b", "[", 1:constants$npred, "]"),

paste0("gamma[", 1:constants$nknots, "]")),

name = c("period",

"relief", "deciduous", "deciduous2", "open", "open2", "conifer", "conifer2",

"tmin", "wsi", "tmin\*wsi",

"relief\*tmin", "deciduous\*tmin", "deciduous2\*tmin", "open\*tmin", "open2\*tmin", "conifer\*tmin", "conifer2\*tmin",

"relief\*wsi", "deciduous\*wsi", "deciduous2\*wsi", "open\*wsi", "open2\*wsi", "conifer\*wsi", "conifer2\*wsi",

paste0("spline", 1:constants$nknots)))

outmcmc <- as.mcmc(out)

# calculate rhat, should be < 1.1 to be considered converged

rhat <- as\_tibble(coda::gelman.diag(outmcmc[,1:47])[[1]],

rownames = "parameter") %>%

setNames(., c("parameter", "rhat", "upper")) %>%

dplyr::select(-upper)

samps <- do.call(rbind, out)

# results bundled up nicely

res <- as.data.frame(samps[, 1:47]) %>%

pivot\_longer(`a1`:`gamma[20]`, names\_to = "parameter", values\_to = "value") %>%

group\_by(parameter) %>%

mutate(mean = mean(value),

lower = quantile(value, c(0.025)),

upper = quantile(value, c(0.975))) %>%

dplyr::select(-value) %>%

distinct(.) %>%

full\_join(key) %>%

dplyr::select(parameter, name, mean:upper) %>%

full\_join(rhat)

# If model is not converged, you can extend the chains

start <- Sys.time()

out2 <- clusterEvalQ(cl, {

outt2 <- runMCMC(CmodelMCMC,

niter = 10000,

nburnin = 9000)

return(as.mcmc(outt2))

})

end <- Sys.time()

end - start

# stop that cluster once you're done...

# stopCluster(cl)