Appendix S4

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

# Model for each mountain range per time period

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

#

# written for JAGS

#

# DATA -----------------------------------------

# n.el = number of elevational bins

# J = number of species

# y[n.el, ] = number of detections of each species

# Y[n.el] = total number of detections at each elevation

# delta[J] = pr(detect an individual if present)

# LAMBDA[J] = total number of detections of each species

# distAway[n.el,J] = number of meters from bin i to last detection

# NOTE: 0 if inside interpolated range, positive if outside interpolated range

# interpPatchy[J] = proportion of bins without detections in interpolated range

#

# NOTE: distAway and interpPatchy were z-transformed to a standard normal

# distribution (mean=0, sd=1) for improved MCMC convergence

#

# PARAMETERS -----------------------------------

# lambda[n.el,J] = true relative abundance of each species (unobserved)

# beta[3] = slopes (all species together)

# a[J] = intercepts (species-specific)

# alpha = average intercept across all species

# sigma = standard deviation in intercept across all species

# Z[n.el,J] = true presence (1) or absence (0) of species

# psi[n.el,J] = pr(Z=1) based on sampling uncertainty

model{

# Likelihood

for(i in 1:n.el) {

# species pool probabilities

p[i,1:J] <- lambda[i,1:J]\*delta[1:J]\*Z[i,1:J]

# detections from a multinomial distribution

# with probabilities p (divide by sum(p) to enforce [0,1] & sum = 1)

y[i,] ~ dmulti(p[i,1:J]/sum(p[i,1:J]), Y[i])

# occupancy and sampling effects

for(j in 1:J) {

Z[i,j] ~ dbern(psi[i,j])

logit(psi[i,j]) <- a[j] +

beta[1]\*distAway[i,j] +

beta[2]\*interpPatchy[j] +

beta[3]\*distAway[i,j]\*interpPatchy[j]

}

}

# Prior distributions

alpha ~ dnorm(0, 0.01)

tau <- 1/(sigma\*sigma) # JAGS uses precision = 1/variance

sigma ~ dnorm(0, 0.01) T(0, )

beta[1] ~ dnorm(0, 0.01)

beta[2] ~ dnorm(0, 0.01)

beta[3] ~ dnorm(0, 0.01)

for(j in 1:J) {

a[j] ~ dnorm(alpha, tau)

# lambda prior for each bin = Normal(mean=total abundance, sd=100)[0,]

for(i in 1:n.el) {

lambda[i,j] ~ dnorm(LAMBDA[j], 0.0001) T(0, )

}

}

}