Lecture 13 - Dynamic N-mixture models

Heather Gaya

November 2021

Overview

We've previously seen N-mixture models across only one year or season

Overview

We've previously seen N-mixture models across only one year or season

But often we want to know how populations change over time.

Overview

We've previously seen N-mixture models across only one year or season

But often we want to know how populations change over time.

Dynamic *N*-mixture models are useful if we have repeated count data without uniquely marked individuals.

Overview

We've previously seen N-mixture models across only one year or season

But often we want to know how populations change over time.

Dynamic *N*-mixture models are useful if we have repeated count data without uniquely marked individuals.

Dynamic models are also very flexible. We can consider how detection parameters change between years, add temporal autocorrelation, account for spatial variation in density, and/or account for treatment effects.

Dynamic N-Mixtures

State model (with Poisson assumption)

$$\log(\lambda_{it}) = \beta_0 + \beta_1 x_{it1} + \beta_2 x_{it2} + \cdots$$
$$N_{it} \sim \text{Poisson}(\lambda_{it})$$

Dynamic N-Mixtures

State model (with Poisson assumption)

$$\log(\lambda_{it}) = \beta_0 + \beta_1 x_{it1} + \beta_2 x_{it2} + \cdots$$
$$N_{it} \sim \text{Poisson}(\lambda_{it})$$

Observation model

$$\text{logit}(p_{ijt}) = \alpha_0 + \alpha_1 w_{it1} + \alpha_2 w_{ijt2} + \cdots$$
$$y_{ijt} \sim \text{Binomial}(N_{it}, p_{ijt})$$

Dynamic N-Mixtures

State model (with Poisson assumption)

$$\log(\lambda_{it}) = \beta_0 + \beta_1 x_{it1} + \beta_2 x_{it2} + \cdots$$
$$N_{it} \sim \text{Poisson}(\lambda_{it})$$

Observation model

$$\text{logit}(p_{ijt}) = \alpha_0 + \alpha_1 w_{it1} + \alpha_2 w_{ijt2} + \cdots$$
$$y_{ijt} \sim \text{Binomial}(N_{it}, p_{ijt})$$

λ_{it} – Expected value of abundance at site i in year t

N_{it} – Realized value of abundance at site i in year t

p_{ijt} – Probability of detecting an individual at site i on occasion j in year t

y_{ijt} – Count data

x_1 and x_2 – site covariates

w_1 and w_2 – observation covariates

Temporal Autocorrelation

The previous model allows for independence between years.

Temporal Autocorrelation

The previous model allows for independence between years.

But we might expect that abundance at year t is related to abundance at year $t - 1$.

Temporal Autocorrelation

The previous model allows for independence between years.

But we might expect that abundance at year t is related to abundance at year $t - 1$.

Dail and Madsen (2011) suggested an expansion to the N-mixture framework that allows us to consider populations from this B.I.D.E model perspective.

Temporal Autocorrelation (Cont')

Year $k = 1$:

$$\log(\lambda_{i1}) = \beta_0 + \beta_1 x_{i1} \cdots$$
$$N_{i1} \sim \text{Poisson}(\lambda_{i1})$$

Temporal Autocorrelation (Cont')

Year $k = 1$:

$$\log(\lambda_{i1}) = \beta_0 + \beta_1 x_{i1} \cdots$$
$$N_{i1} \sim \text{Poisson}(\lambda_{i1})$$

Years $k > 1$:

$$S_{it} \sim \text{Binomial}(N_{i(t-1)}, \phi_{it})$$
$$G_{it} \sim \text{Poisson}(N_{i(t-1)} * \gamma_{it})$$
$$N_{it} = S_{it} + G_{it}$$

Temporal Autocorrelation (Cont')

Year $k = 1$:

$$\log(\lambda_{i1}) = \beta_0 + \beta_1 x_{i1} \cdots$$
$$N_{i1} \sim \text{Poisson}(\lambda_{i1})$$

Years $k > 1$:

$$S_{it} \sim \text{Binomial}(N_{i(t-1)}, \phi_{it})$$
$$G_{it} \sim \text{Poisson}(N_{i(t-1)} * \gamma_{it})$$
$$N_{it} = S_{it} + G_{it}$$

λ_{it} – Expected value of abundance at site i in year 1

N_{it} – Realized value of abundance at site i in year t

x_1 and x_2 – site covariates

ϕ_{it} – Apparent survival at site i from year $t-1$ to t

γ_{it} – Apparent recuritment at site i from year $t-1$ to t

S_{it} – Realized number of individuals that survived/didn't emmigrate

G_{it} – Realized number of new individuals

Simulation

First lets simulate some abundance data from a landscape.

Elevation stays the same but precipitation will change each year

```
set.seed(100)
elev <- raster("Elevation.tif")
precip1 <- raster("Precipitation1.tif")
precip2 <- raster("Precipitation2.tif")
precip3 <- raster("Precipitation3.tif")
precip4 <- raster("Precipitation4.tif")
precip5 <- raster("Precipitation5.tif")
```

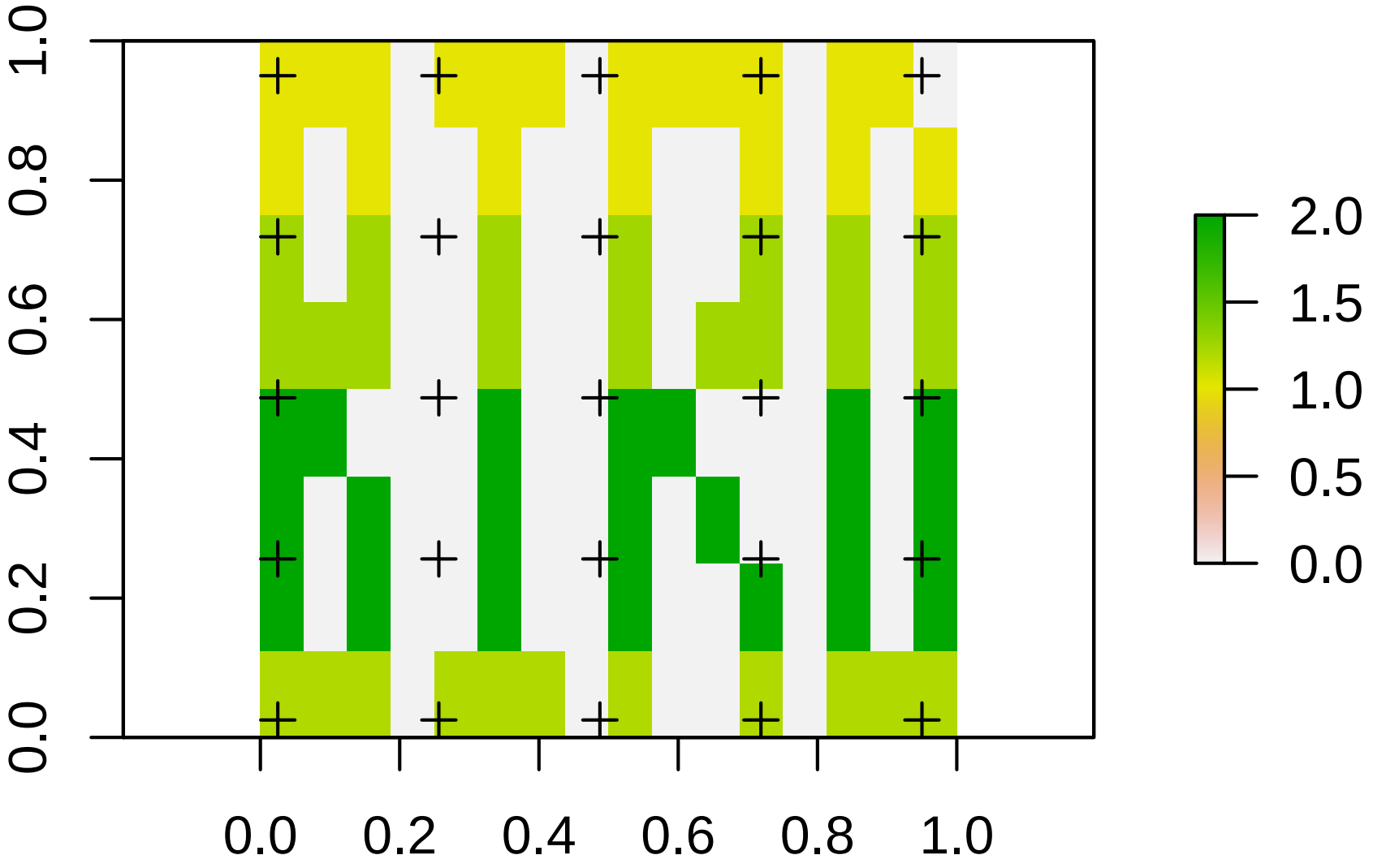
Simulation (cont.)

```
n.years <- 5
sites <- data.frame(x = rep(seq(0.025, 0.95, length.out = 5),
    5), y = rep(seq(0.025, 0.95, length.out = 5), each = 5))
n.sites <- nrow(sites)
head(sites, n = 5)
```

```
##         x     y
## 1 0.02500 0.025
## 2 0.25625 0.025
## 3 0.48750 0.025
## 4 0.71875 0.025
## 5 0.95000 0.025
```

Simulation (cont.)

```
plot(elev)
points(sites, pch = 3)
```



Get the Elevation and Precipitation Covariates For Each Site

```
sites$elev <- extract(elev, sites[,c("x", "y")])
sites$precip1 <- extract(precip1, sites[,c("x", "y")])
sites$precip2 <- extract(precip2, sites[,c("x", "y")])
sites$precip3 <- extract(precip3, sites[,c("x", "y")])
sites$precip4 <- extract(precip4, sites[,c("x", "y")])
sites$precip5 <- extract(precip5, sites[,c("x", "y")])
precip <- data.frame(p1 = sites$precip1,
                     p2 = sites$precip2, p3 = sites$precip3,
                     p4 = sites$precip4, p5 = sites$precip5)
precip <- as.matrix(precip)
elevation <- matrix(sites$elev)
```

Simulate Site-Specific Parameters

Simulate initial N, ϕ and γ

```
psi0 <- 3
psi1 <- -.25
phi0 <-.8
phi1 <- -3
gamma0 <- -3
gamma1 <- .2
gamma2 <- .9
phi <- gamma <- matrix(NA, nrow = n.sites, ncol = n.years)
for(i in 1:n.years){
phi[,i] <- plogis(phi0 +
          phi1*(precip[,i]-mean(precip))/sd(precip))
gamma[,i] <- exp(gamma0 +
          gamma1*(precip[,i]-mean(precip))/sd(precip) +
          gamma2*(elevation-mean(elevation))/sd(elevation))
}
```

Simulate Abundance

```
n.sites <- nrow(sites)
N <- S <- G <- array(NA, dim = c(n.sites, n.years))
lambda <- array(NA, dim = n.sites)
for (i in 1:n.sites){
  lambda[i] <- exp(psi0 +
                   psi1*(elevation[i]-mean(elevation))/sd(elevation))
  N[i,1] <- rpois(1, lambda[i])
 for (t in 2:n.years){
   S[i,t] <- rbinom(1, N[i,t-1], phi[i,t])
   G[i,t] <- rpois(1, N[i,t-1]*gamma[i])
   N[i,t] <- S[i,t] + G[i,t]
 }
}
```

Simulate

Let's see what N looks like for the first 3 sites

```
N[1:3,]
```

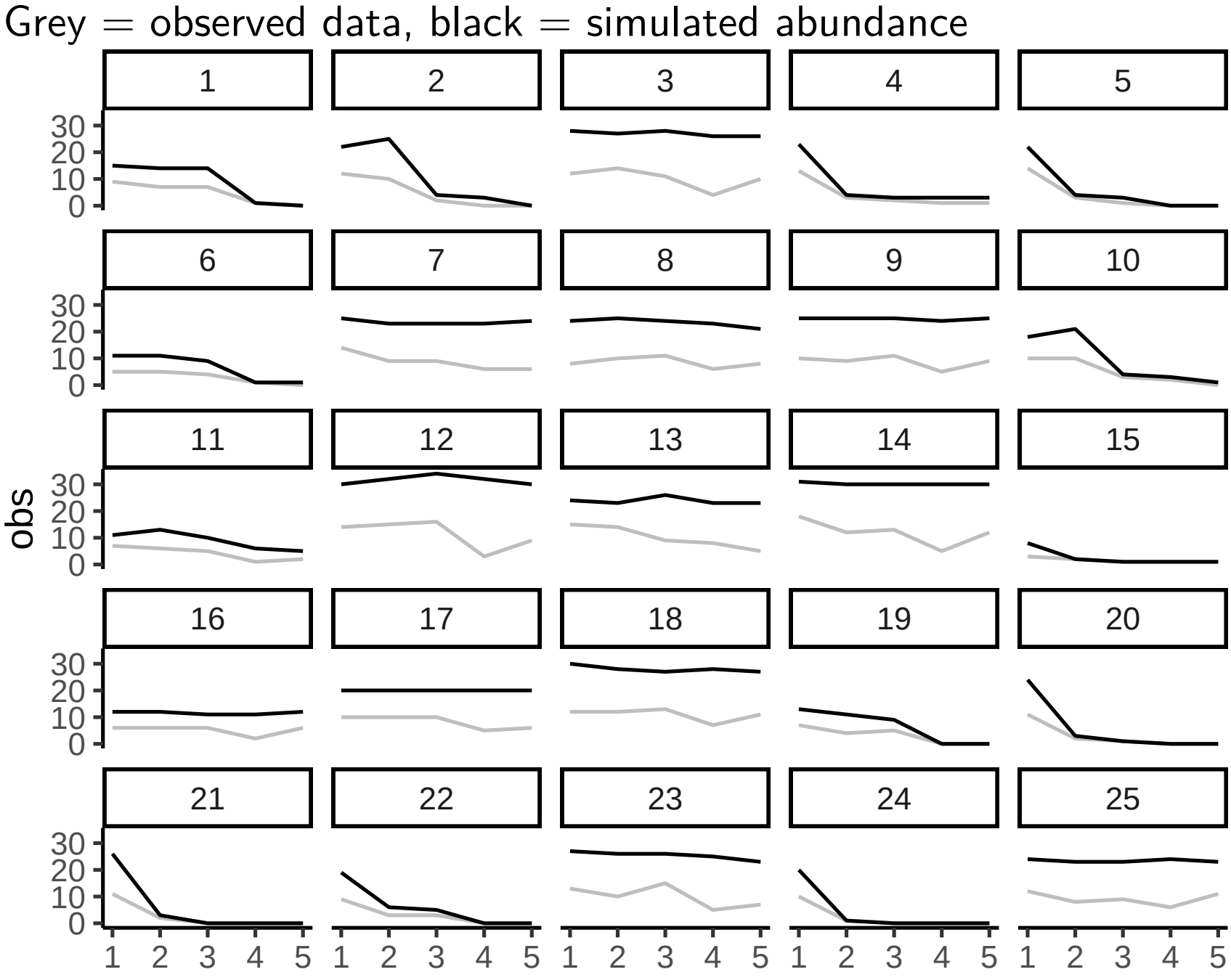
```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   15   14   14    1    0
## [2,]   22   25    4    3    0
## [3,]   28   27   28   26   26
```

Simulate Detection

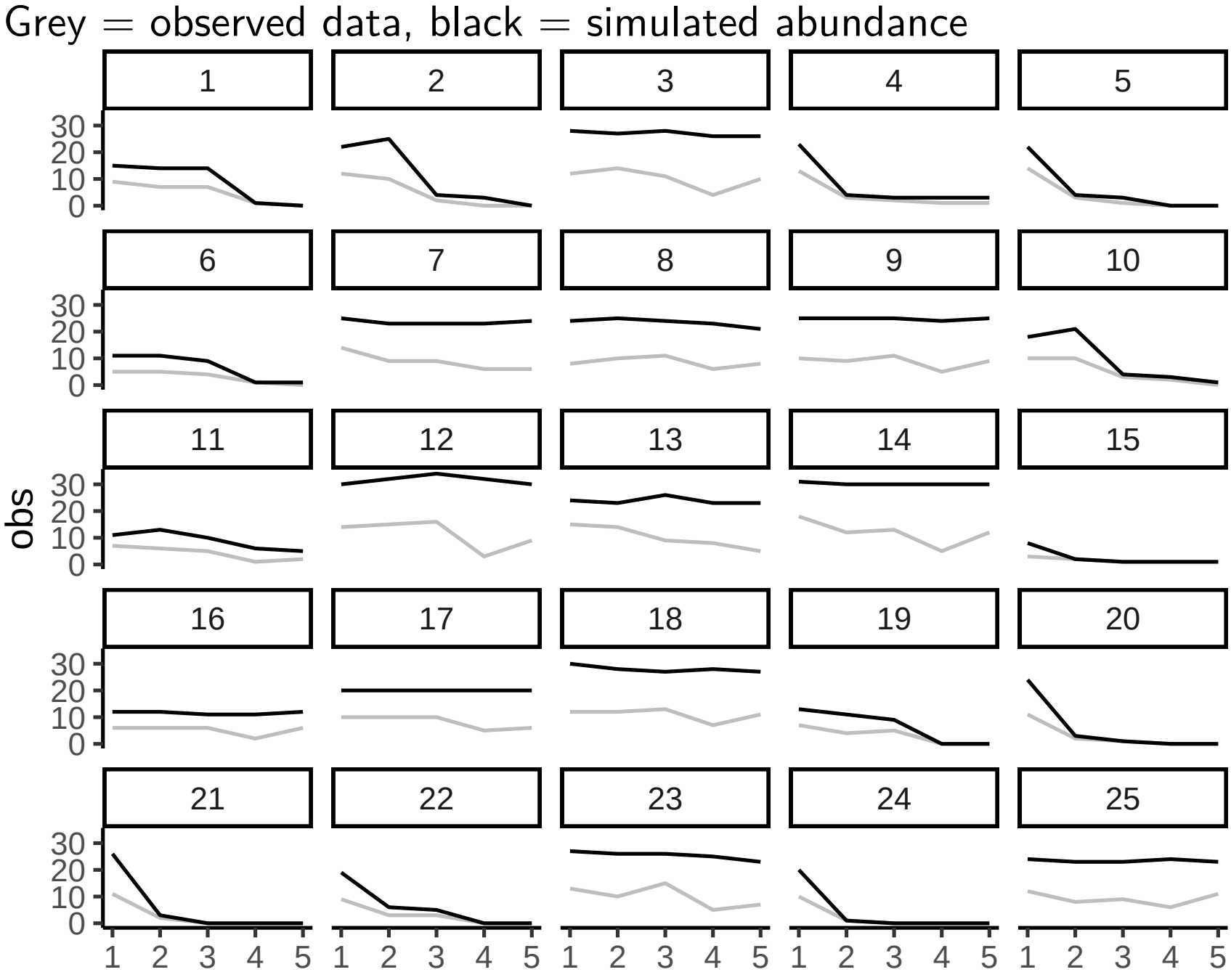
We'll use a time-varying detection for simulation but it could be anything.

```
n.visit <- 4
p <- rnorm(n.years, mean = .3, .1)
y1 <- array(NA, dim = c(n.sites, n.visit, n.years))
for (i in 1:n.sites){
  for (t in 1:n.years){
    y1[i,,t] <- rbinom(n.visit, size = N[i,t],
                       prob = p[t])
  }
}
```

Inspect Our Simulation



Inspect Our Simulation



Setup in Unmarked

```
umf <- unmarkedFramePCO(y = y_matrix,
         siteCovs = data.frame(elevation),
         yearlySiteCovs = list(precip=precip),
         numPrimary = 5)
```

Run in Unmarked

This may take awhile.

```
m1 <- pcountOpen(~scale(elevation),
      ~scale(elevation)+scale(precip),
      ~scale(precip),
      ~1, umf, K=40, mixture = "P",
      dynamics = "autoreg")
```

Order of equations is lambda, apparent recruitment (gamma), apparent survival, detection

Unmarked Output

```
m1
```

```
##
## Call:
## pcountOpen(lambdaformula = ~scale(elevation), gammaformula = ~scale(elevation) +
##     scale(precip), omegaformula = ~scale(precip), pformula = ~1,
##     data = umf, mixture = "P", K = 40, dynamics = "autoreg")
##
## Abundance:
##                  Estimate     SE     z   P(>|z|)
## (Intercept)         2.969 0.1036 28.65 1.55e-180
## scale(elevation)   -0.224 0.0549 -4.07  4.74e-05
##
## Recruitment:
##                  Estimate     SE     z P(>|z|)
## (Intercept)       -0.5120 0.1904 -2.69 0.00716
## scale(elevation)  -0.0585 0.0637 -0.92 0.35783
## scale(precip)     -0.3381 0.1628 -2.08 0.03777
##
## Apparent Survival:
##                Estimate   SE      z P(>|z|)
## (Intercept)      -4.46 8.52 -0.524   0.601
## scale(precip)    -0.10 4.78 -0.021   0.983
##
## Detection:
##  Estimate    SE    z P(>|z|)
##    -0.378 0.151 -2.5  0.0123
##
## AIC: 1906.147
```

Extract real estimates for sites

```
re <- ranef(m1)
```

Let's check site 1 each year and compare with our simulation

```
round(bup(re, stat="mean")[1,], digits = 2)
```

```
## [1] 16.39 11.82 11.01  1.79  0.07
```

```
confint(re, level=0.95)[1,,] # 95% CI
```

```
##       [,1] [,2] [,3] [,4] [,5]
## 2.5%    13    9    8    1    0
## 97.5%   21   16   14    4    1
```

```
N[1,]
```

```
## [1] 15 14 14  1  0
```

JAGS

```
model {
for (i in 1:n.sites) {
  log(lambda[i]) <- psi0 + psi1*elevation[i]

  N[i,1] ~dpois(lambda[i])

  for (t in 2:n.years){
    logit(phi[i,t]) <- phi0 + phi1*precip[i,t]
    log(gamma[i,t]) <- gamma0 + gamma1*elevation[i]+gamma2*precip[i,t]

    S[i,t] ~ dbin(phi[i,t], N[i,t-1])
    G[i,t] ~ dpois(N[i,t-1]*gamma[i,t])

    N[i,t] <- S[i,t] + G[i,t]
  } #end t
  for (t in 1:n.years){
  for (j in 1:n.visit){
    y1[i,j,t] ~ dbin(p[t], N[i,t])
    } #end j
  }#end t again
  } #end i

  for(t in 1:n.years){
     p[t] ~ dunif(0,1)
  }
  gamma0 ~ dunif(-5,5)
  gamma1 ~ dunif(-5,5)
  gamma2 ~ dunif(-5,5)
  psi0 ~ dunif(-5,5)
  psi1 ~ dunif(-5,5)
  phi0 ~ dnorm(0,.3)
  phi1 ~ dnorm(0,.3)
}
```

Send to JAGS

```
jd <- list(n.sites =n.sites, n.visit = n.visit,
           n.years = n.years, y1 = y1,
           elevation = as.vector(scale(elevation)),
           precip = (precip - mean(precip))/sd(precip))
params = c("p", "psi0", "psi1",
           "phi0", "phi1",
           "gamma0", "gamma1", "gamma2")
```

What do we do about initial values?

Initial values are tricky because the model is recursive. Luckily, we have a fun loop we can use to help.

```
getInits <- function(counts, sites, years) {
  nSites <- sites
  nYears <- years
  N <- array(NA_integer_, c(nSites, nYears))
  G <- S <- array(NA_integer_, c(nSites,nYears))
  for(i in 1:nSites){
    N[i,1:nYears] <- max(counts[i,,], na.rm=TRUE)+2
  }
    S[,1] <- G[,1] <- NA
    for(t in 2:nYears) {
      S[,t] <- rbinom(nSites, size=N[,t-1], 0.6)
      G[,t] <- N[,t]-S[,t]
    }
  N.r <- N
  N.r[,2:nYears] <- NA
  return(list(S=S, G=G, N = N.r))
}
```

Create Initial Values

```
inits <- getInits(y1, n.sites, n.years)
ji <- function() {
  list(psi0 = runif(1),psi1 = runif(1),
    phi0 = runif(1), phi1 = runif(1),
    gamma0 = runif(1), gamma1 = runif(1),
    gamma2 = runif(1), S = inits$S,
    G= inits$G, N = inits$N
    )
}
```

Run JAGS

```
jags.post <- jags.basic(data=jd, inits = ji,
               parameters.to.save=params,
               model.file="dynamic_n.txt",
               n.chains=3, n.adapt=100, n.burnin=2000,
               n.iter=3000, parallel=TRUE)
```

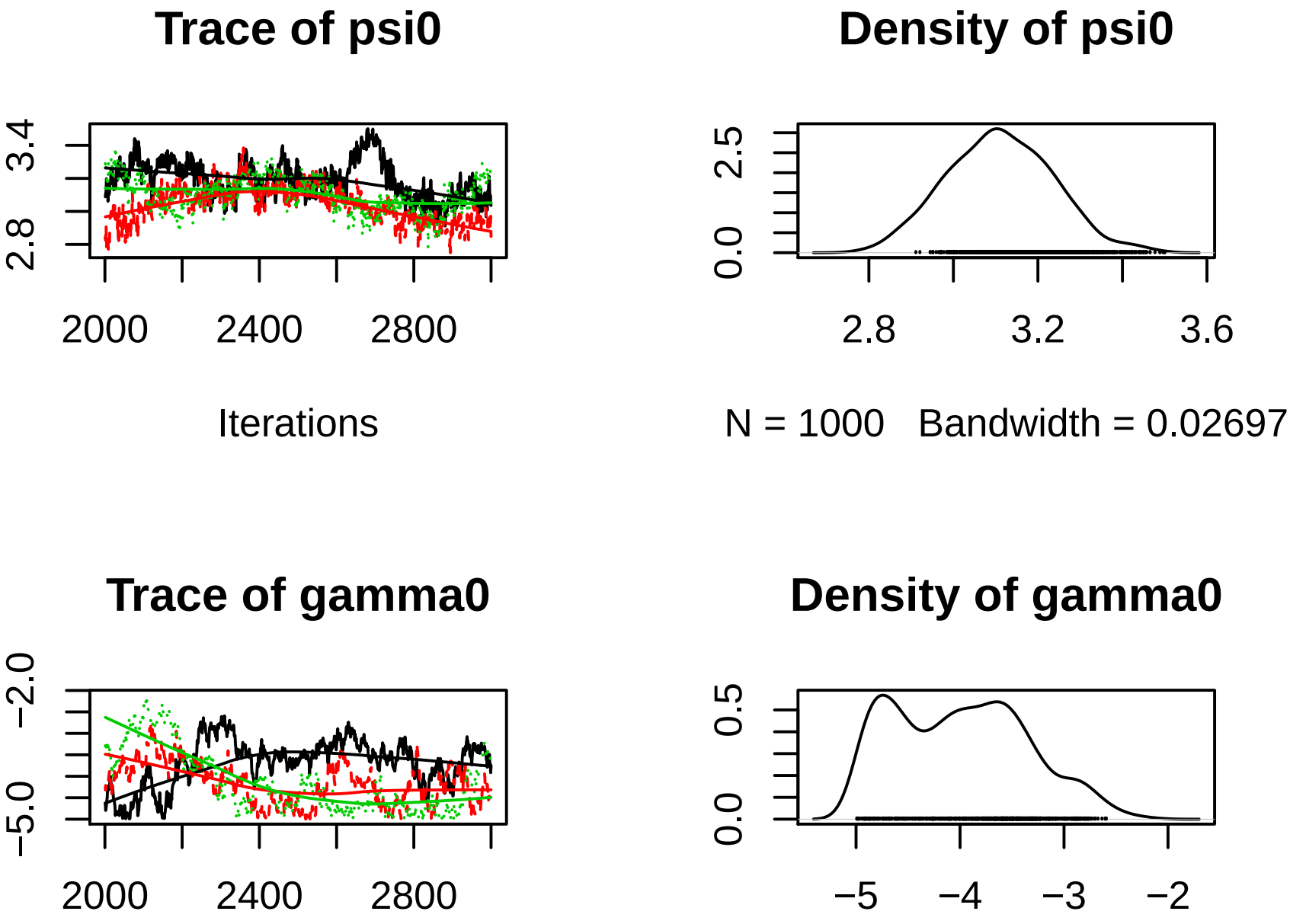
JAGS Output

```
summary(jags.post)$quantiles[-1,]
```

	2.5%	25%	50%	75%	97.5%
gamma0	-4.96	-4.59	-4.00	-3.53	-2.72
gamma1	0.44	1.02	1.28	1.58	2.17
gamma2	-1.92	-0.77	-0.28	0.05	0.70
p[1]	0.27	0.32	0.35	0.38	0.43
p[2]	0.31	0.35	0.37	0.39	0.44
p[3]	0.37	0.42	0.45	0.48	0.53
p[4]	0.15	0.18	0.19	0.21	0.26
p[5]	0.32	0.37	0.42	0.49	0.60
phi0	-0.18	0.03	0.28	0.59	0.85
phi1	-2.32	-1.86	-1.67	-1.49	-1.26
psi0	2.87	3.02	3.11	3.20	3.37
psi1	-0.34	-0.26	-0.23	-0.19	-0.13

Trace Plot

```
plot(jags.post[,c("psi0","gamma0"),])
```



Compare with Truth

Pretty close to the simulated values

	Simulated	JAGS	lower	upper
gamma0	-3.00	-4.00	-4.96	-2.72
gamma1	0.20	1.28	0.44	2.17
gamma2	0.90	-0.28	-1.92	0.70
p[1]	0.36	0.35	0.27	0.43
p[2]	0.35	0.37	0.31	0.44
p[3]	0.36	0.45	0.37	0.53
p[4]	0.13	0.19	0.15	0.26
p[5]	0.24	0.42	0.32	0.60
phi0	0.80	0.28	-0.18	0.85
phi1	-3.00	-1.67	-2.32	-1.26
psi0	3.00	3.11	2.87	3.37
psi1	-0.25	-0.23	-0.34	-0.13

Assignment

1. Fit the unmarked model but allow p to vary by year. Hint: Look at the unmarked help page. Compare the AIC score for the model with time varying detection and the one we ran in class. Did unmarked correctly identify the true model?

2. Using the JAGS model, compare the true abundance and estimated abundance of sites 1, 2, and 3. Did the JAGS model capture the true value? Be sure to plot means and CIs for your JAGS model output.