Design and Analysis of Occupancy Studies
Part 1a
Supplemental (*unmarked* & *JAGS*)

Carl James Schwarz

Department of Statistics and Actuarial Science
Simon Fraser University
Burnaby, BC, Canada
cschwarz @ stat.sfu.ca

Single Species; Single-Season - Supplemental

Using the *unmarked* package

Single Species; Single-Season - *unmarked*

unmarked is an *R* package that is able to fit some of the simpler occupancy models

Key advantages are:

- Scripts so that analyzes can be reused.
- No more clicking
- Much easier extracting output
- Available for Macintosh, Windoze, and Linux platforms.

Key disadvantages are:

- Not all analyses can be done in *unmarked*.
- Uses S4 class system making it difficult to read documentation

Single Species; Single-Season - *unmarked*

Installation of package (available on CRAN)

- Download package file from CRAN

Open the sample program in the *Salamander* directory

Single Species; Single-Season - *unmarked* - Salamander

Basic steps in analysis:

- Input the history file and create the *UMF* object
- Fit some model using the *occu()* function.
- Model average using the *modSel* function.
- Extract results and plot results etc

Refer to sample analysis script.

Read in raw data and so some basic error checking:

```
1  input.data <- readxl::read_excel(file.path("..","salamande
2                                   sheet="CompleteData",
3                                   col_names=FALSE)  # notic
4
5  # do some basic checks on your data
6  nrow(input.data)
7  ncol(input.data)
8  head(input.data)
```

Histories is a matrix ($n_{sites} \times n_{visits}$) of 1's, 0's, or NA's.

Create the *UMF* file.
You need to create the covariates that will be used in the model fitting, including simple covariates for time dependence (!). This is a pain.

The visit covariates are a data frame that has $n_{sites} \times n_{visits}$ and a separate column for each covariate for each site-visit combination.

This has a DIFFERENT ordering than *RPresence*! In *unmarked*, the covariate values for vist_1 to visit_J are given for site_1; and then for site_2, etc.

Single Species; Single-Season - *unmarked* - Salamander

Create the visit covariates for

- Time specific effects, i.e. effects for v_1, v_2, v_3, v_4, v_5.
- First two visits have same effect; as do last 3 visit.

```
1  obs.covar <- data.frame( Site=rep(1:nrow(input.data),
2                                    each=ncol(input.data)),
3                           Visit=rep(1:ncol(input.data),
4                                    nrow(input.data)),
5                           Time =rep(c("T1","T2","T3","T4","T5")
6                                    nrow(input.data)),
7                           D =rep(c("D1","D1","D2","D2","D2"),
8                                  nrow(input.data)))
9  obs.covar[1:7,]
```

This gives:

```
> obs.covar[1:10,]
   Site Visit Time  D
1     1     1   T1 D1
2     1     2   T2 D1
3     1     3   T3 D2
4     1     4   T4 D2
5     1     5   T5 D2
6     2     1   T1 D1
7     2     2   T2 D1
8     2     3   T3 D2
9     2     4   T4 D2
10    2     5   T5 D2
```

Single Species; Single-Season - *unmarked* - Salamander

Now put together into an unmarked data frame:

```
salamander.UMF <- unmarked::unmarkedFrameOccu(
                        y = input.data,
                        obsCovs=obs.covar)
summary(salamander.UMF)
plot(salamander.UMF)
```

Single Species; Single-Season - *unmarked* - Salamander

Fitting a model *psi(.), p(.)*.
Double formula is for detection than occupancy.

```
1 mod.pdot.u <- unmarked::occu(~1 ~1 ,
2                 data=salamander.UMF,
3                 se=TRUE)
```

Note that formula DO NOT HAVE AN = SIGN.

This creates a complex object with MANY slots and extractor function

Single Species; Single-Season - *unmarked* - Salamander

Lot of interesting stuff!

- Basic statistics about your data - make sure these are correct
- Information about fitting the model - typically not of interest.
- *Beta* estimates - typically not of interest
- Estimates of occupancy and detectability.
- Posterior probability of occupancy.

Estimates for *p* and *psi* are best obtained using the *predict()* as this automatically converts to the 0-1 scale and provides standard errors.
The estimates for ψ are computed for each row in the *newdata* data frame. In this case, there are no covariates so we set up a dummy data frame

```
1 newdata <- data.frame(factor=1:1)
2 predict(mod.pdot.u, type='state', newdata=newdata)
```

```
> predict(mod.pdot.u, type='state', newdata=newdata)
  Predicted        SE     lower     upper
1 0.5946225 0.1225986 0.3512047 0.7989852
```

Ditto for estimates of p.

```
1 predict(mod.pdot.u, type='det', newdata=newdata)
```

```
> predict(mod.pdot.u, type='det', newdata=newdata)
  Predicted         SE     lower     upper
1  0.258729 0.05770192 0.1621571 0.3862968
```

Finally, the posterior probability of occupancy

```
1 bup(ranef(mod.pdot.u))
```

```
> bup(ranef(mod.pdot.u))
 [1] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.00
[14] 1.0000000 0.2471563 0.2471563 0.2471563 0.2471563 0.24
[27] 0.2471563 0.2471563 0.2471563 0.2471563 0.2471563 0.24
```

Single Species; Single-Season - *unmarked* - Salamander

Fit a model where detection varies across visits.
We set up the visit covariate dataframe earlier. Notice the use of the user-defined variable *Time* to represent the visits.

```
1  mod.pt.u <- unmarked::occu(~Time ~1 ,
2              data=salamander.UMF,
3              se=TRUE)
```

Single Species; Single-Season - *unmarked* - Salamander

Fit a model where $p_1 = p_2; p_3 = p_4 = p_5$.
We created a data frame for the D covariate earlier.

```
1  mod.pcustom.u <- unmarked::occu(~D ~1 ,
2                 data=salamander.UMF,
3                 se=TRUE)
```

Single Species; Single-Season - *unmarked* - Salamander

Model averaging:

```
1  models.u <-unmarked::fitList(
2              mod.pdot.u,
3              mod.pt.u,
4              mod.pcustom.u)
5  aic.table.u <- unmarked::modSel(models.u)
6  aic.table.u
```

```
> aic.table.u
                nPars    AIC delta AICwt cumltvWt
mod.pcustom.u       3 162.82  0.00 0.760     0.76
mod.pdot.u          2 165.76  2.94 0.175     0.93
mod.pt.u            6 167.71  4.90 0.066     1.00
```

Model averaged results for ψ.

```
1 predict(models.u, type="state")[1,]
```

```
> predict(models.u, type="state")[1,]
  Predicted        SE     lower     upper
1 0.5849543 0.119358 0.3498164 0.7868192
```

Summary:

- Don't put = signs in formula.
- Formula order is DETECTION then OCCUPANCY.
- Creating visit covariates is a bit tricky
- Same output as in *PRESENCE* and *MARK*.

Single Species; Single-Season - Missing Values

Missing Completely at Random:

- Bad weather
- Equipment breakdown

Modify histories in input files as follows:

- *PRESENCE*: Use a "-" (hyphen) to represent a missing value.
- *MARK*: Use a "." (period) to represent a missing value.
- *RPresence*: Use NA to represent a missing value
- *RMark*: Use a "." (period) to represent a missing value in the *ch* variable in the data.frame.
- *unmarked*: Use NA to represent a missing value
- *JAGS*: Need to delete the missing values

Do NOT use a 0 to represent a missing value!

Single Species; Single-Season - Missing Values - *unmarked*

Run *unmarked* in the usual way – refer to *R* code.
Things to watch out for: Reading in '.' or '-' from file for missing values.

```
input.data <- readxl::read_excel("Blue_Ridge_pg99.xls",
                                 sheet="SomeMissingValues",
                                 na='-',
                                 col_names=FALSE)  # notice
```

Any covariates (e.g. Time) will be ignored for missing values

Single Species; Single-Season - Model Assessment - *unmarked*

Method 1. Use a chi-square test from MacKenzie, D. I., Bailey, L. L. (2004) that compares observed vs. expected frequency and computes a chi-square statistic.
This doesn't have a "nice" distribution with smallish samples, so need to do a parametric bootstrap goodness of fit.

Implemented in the *mb.chisq()* function in the *AICcmodavg* package.

```
1  library(AICcmodavg)
2  #  Mackenzie Bailey Goodness of fit test
3  mod.pcustom.u.chi   = mb.chisq(mod.pcustom.u)
4  mod.pcustom.u.chi
5
6  mod.pcustom.u.boot = mb.gof.test(mod.pcustom.u, nsim = 100)
7
8  print(mod.pcustom.u.boot, digit.vals=4, digits.chisq=4)
```

Single Species; Single-Season - Model Assessment - *unmarked*

This gives

```
MacKenzie and Bailey goodness-of-fit for single-season occu

Pearson chi-square table:

      Cohort Observed Expected Chi-square
00000      0       21    21.00       0.00
00001      0        2     2.41       0.07
00010      0        2     2.41       0.07
...

Chi-square statistic = 42.311
```

Single Species; Single-Season - Model Assessment - *unmarked*

We need to find the distribution of the gof statistic and see how extreme it is.

```
Number of bootstrap samples = 100
P-value = 0.1

Quantiles of bootstrapped statistics:
  0%  25%  50%  75% 100%
   8   18   23   31  104

Estimate of c-hat = 1.59
```

Usually $\widehat{c}$ below about 2 are ignored.

Single Species; Single-Season - Model Assessment - *unmarked*

Method 2. Create your own statistics and do parametric bootstrapping.

```
fitstats <- function(fm) {
    observed <- getY(fm@data)
    expected <- fitted(fm)
    resids <- residuals(fm)
    sse <- sum(resids^2)
    chisq <- sum((observed - expected)^2 / expected)
    freeTuke <- sum((sqrt(observed) - sqrt(expected))^2)
    out <- c(SSE=sse, Chisq=chisq, freemanTukey=freeTuke)
    return(out)
    }

mod.pcustom.u.pboot <- unmarked::parboot(mod.pcustom.u,
          fitstats, nsim=99, report=100
```

Single Species; Single-Season - Model Assessment - *unmarked*

This gives

```
> mod.pcustom.u.pboot@t0
         SSE         Chisq freemanTukey
    24.85043    164.98712     35.41157
> head(mod.pcustom.u.pboot@t.star)
          SSE    Chisq freemanTukey
[1,] 29.10684 158.9999     40.62576
[2,] 24.01709 165.9999     34.27657
[3,] 20.06838 172.0047     29.71141
[4,] 25.19658 163.9998     35.55930
[5,] 27.03846 162.0001     38.14598
[6,] 30.01709 157.0000     41.44367
...
```

Single Species; Single-Season - Model Assessment - *unmarked*

And then we compute the p-value

```
> # How extreme is t0 relative to the bootstrap values.
> # Small probabilities indicate lack of fit
> colMeans(mod.pcustom.u.pboot@t.star >
                    matrix(mod.pcustom.u.pboot@t0, byrow=TRU
+                                                ncol=length(mod.pcust
+                                                nrow=nrow(mod.pcustom
          SSE         Chisq freemanTukey
    0.4343434    0.5555556    0.4444444
```

No evidence of lack of fit.

Single Species; Single Season

Covariates with *unmarked*

Mahoenui giant weta (*Deinacrida mahoenui*) is endemic to New Zealand and under stress from rats and other predators.
72 circular plots (3 m radius, primarily prickly gorse plants) were surveyed for weta.
Each plot surveyed 3-5 times.
Covariates to be considered:

- Observer. Three different observers and not every plot surveyed by each observer.
- Browse. Was each site browsed by goats, yes or no.

Getting the data into *unmarked*.

1. Capture History

Get $n_{sites} \times n_{visit}$ data.frame (or matrix) of 1, 0, or NAs

```
input.history <- readxl::read_excel(
     "Weta_pg116.xls",
      sheet="detection_histories",
      na="-",
      col_names=FALSE)  # notice no column names in row 1
```

Detection histories include many missing values. Are these MCAR?

Getting the data into *unmarked*.

2. Site Covariates

Get $n_{sites} \times n_{site-covariates}$ data.frame of site covariates.

- Continuous covariates occupy 1 column
- Categorical covariates can either be alpha-numeric code or a set of indicator variables.

With modern software, the former is preferred.

Getting the data into *unmarked*.

2. Site Covariates

```
site_covar <- readxl::read_excel("Weta_pg116.xls",
                                 sheet="site_covar",
                                 na="-",
                                 col_names=TRUE)  # notice

# Create an alternate site level covariate that is a categ
# than indicator variables
site_covar$BrowCat <-
   paste(c("","B")[1+unlist(site_covar[,1])],
         c("","N")[1+unlist(site_covar[,2])], sep="")
xtabs(~BrowCat, data=site_covar,exclude=NULL, na.action=na
colSums(site_covar[,1:2])

head(site_covar)
```

Getting the data into *unmarked*.

2. Site Covariates

```
> head(site_covar)
  Browsed Unbrowsed BrowCat
1       1         0       B
2       1         0       B
3       1         0       B
4       0         1       N
5       1         0       B
6       0         1       N
```

Browse covariate entered as two indicator variables (this has implications later on) or as a categorical variable.
No missing values allowed in site-level covariates.

Getting the data into *unmarked*.

3. Visit Covariates

Get $n_{visit-covariates}$ sets of $n_{sites} \times n_{visits}$ data.frame of visit covariates.

- Continuous covariates have values in each cell
- Categorical covariates can either be alpha-numeric codes or a set of indicator variables.

With modern software, the former is preferred.

Getting the data into *unmarked*.

3. Visit Covariates

```
# Get the individual covariates.
obs1 <- readxl::read_excel("Weta_pg116.xls",
                           sheet="Obs1",
                           na="-",
                           col_names=FALSE)
....

# Create an alternate site level covariate that is a categ
# than indicator variables
obs <- obs1*1 + obs2*2 + obs3*3
```

Notice how I created a single categorical covariate for observer number (preferred)

Getting the data into *unmarked*.

3. Visit Covariates

These must be stacked into a vector of length $n_{sites} \times n_{visits}$ where values for site 1 appear first (for all visit), then for site 2, etc. **The order differs from RPresence**.

```
1 survey.covar.u <- data.frame(
2       Site=rep(1:nrow(input.history), each=ncol(input.histo
3       Visit=rep(1:ncol(input.history), nrow(input.history))
4       Time =rep(c("T1","T2","T3","T4","T5"), nrow(input.hist
5       obs1 =as.vector(t(as.matrix(obs1))),
6       obs2 =as.vector(t(as.matrix(obs2))),
7       obs3 =as.vector(t(as.matrix(obs3))),
8       obs  =as.character(as.vector(t(as.matrix(obs)))), str
9 head(survey.cov.)
```

Notice how *obs* was forced to be alphanumeric so *unmarked* will not treat it as a continuous variable.

Single Species; Single-Season - Covariates - *unmarked*

Getting the data into *unmarked*.

3. Visit Covariates

```
> survey.covar.u[1:10,]
   Site Visit Time obs1 obs2 obs3  obs
1     1     1   T1    1    0    0    1
2     1     2   T2    0    0    1    3
3     1     3   T3    0    1    0    2
4     1     4   T4    0    0    1    3
5     1     5   T5   NA   NA   NA <NA>
6     2     1   T1    1    0    0    1
....
```

Final column is a categorical covariate with values of "1", "2","3" (be sure that these are alpha-number codes, otherwise *unmarked* will treat as continuous variables.:

Single Species; Single-Season - Covariates - *unmarked*

Getting the data into *unmarked*.

Finally, create the *UMF* object.

```
weta.UMF <- unmarked::unmarkedFrameOccu(
                        y = input.history,
                        siteCovs=site_covar,
                        obsCovs=survey.covar.u
                        )
```

Single Species; Single-Season - Covariates - *unmarked*

Fit the $\psi(*), p(*)$ model and look at estimates.

```
mod.pdot.u <- unmarked::occu(~1 ~1 ,
      data=weta.UMF, se=TRUE)

# get estimates of occpancy
newdata <- data.frame(factor=1:1)
predict(mod.pdot.u, type='state', newdata=newdata)

# look at the estimated probability of detection. It gives
predict(mod.pdot.u, type='det', newdata=newdata)
```

Fit the $\psi(*), p(*)$ model and look at estimates.

```
> predict(mod.pdot.u, type='state', newdata=newdata)
  Predicted         SE     lower     upper
1 0.6166237 0.08854806 0.4356335 0.7701904

> predict(mod.pdot.u, type='det', newdata=newdata)
  Predicted         SE    lower     upper
1 0.3493752 0.05373287 0.252544 0.4604619...
```

Fitting a model where occupancy varies by browse.
It doesn't make sense to model occupancy as a function of observer - why?

There are several (equivalent) ways to do this.

Models where occupancy varies by browse.

- Cell-means approach.

$$logit(\psi) = \alpha_1(unbrowsed) + \alpha_2(browsed)$$

This requires a design matrix with initial columns of 0/1's to indicate UNbrowsed, and second column of 0/1's to indicate Browsed.
α_1 is interpreted as the logit (occupancy) for the unbrowsed sites and α_2 is interpreted as the logit(occupancy) for the browsed sites directly.

Only useful for models with a single categorical covariate for a parameter.

Models where occupancy varies by browse.

- Cell-effects approach.

$$\textit{logit}(\psi) = \alpha_1 + \alpha_2(\textit{browse})$$

This requires a design matrix with an initial column of 1's (for α_1) and a second column of 0/1's to indicate if browsed (for α_2).
α_1 is interpreted as the logit(occupancy) for the (baseline) of unbrowsed and α_2 is the difference in logits between unbrowsed and browsed.

Can be used with any number of categorical covariates. Trick is figuring out which is the reference class used (corresponding to the α_1 term)

Cell-means approach:

```
mod.pdot.psiB.1.u <- unmarked::occu(
     ~1 ~-1+Browsed+Unbrowsed,
     data=weta.UMF, se=TRUE)
```

or (prefered)

```
mod.pdot.psiB.2.u <- unmarked::occu(
     ~1 ~-1+BrowCat,
     data=weta.UMF, se=TRUE)
```

No need for you to define indicator variables with the latter model.

Single Species; Single-Season - Covariates - *unmarked*

Cell-means approach: Results (same for ALL models)

```
> newdata <- data.frame(Browsed=c(1,0), Unbrowsed=c(0,1))
> cbind(newdata,predict(mod.pdot.psiB.1.u, type='state', ne
  Browsed Unbrowsed Predicted        SE     lower     upper
1       1         0 0.7593382 0.1198171 0.4660501 0.919392
2       0         1 0.4809820 0.1078825 0.2843231 0.683715
```

```
> newdata <- data.frame(BrowCat=c("B","N"))
> cbind(newdata,predict(mod.pdot.psiB.2.u, type='state', ne
  BrowCat Predicted        SE     lower     upper
1       B 0.7593680 0.1198254 0.4660415 0.9194190
2       N 0.4809969 0.1078859 0.2843301 0.6837338
```

Probability of Occupancy for Browsed areas is 0.75; that of unbrowwed areas 0.48.

Cell-means approach:
Look at design matrix

```
> coef(mod.pdot.psiB.1.u)
  psi(Browsed) psi(Unbrowsed)          p(Int)
    1.14905452    -0.07610881    -0.62220870
```

Cell-means approach:

$$
\begin{aligned}
logit(\psi_{unbrowsed}) &= \alpha_1(0) + \alpha_2(1) = -0.076 \\
\psi_{unbrowsed} &= 1/(1 + exp(-(-0.076)) = 0.4810 \\
logit(\psi_{browsed} &= \alpha_1(1) + \alpha_2(0) = 1.1493 \\
\psi_{browsed} &= 1/(1 + exp(-(1.1493)) = 0.7594.
\end{aligned}
$$

Cell-means approach: Estimate the odds ratio

```
1 mod.pdot.psiB.1.u.oddsratio.browse <-
2       exp( sum(c(1,-1,0)*coef(mod.pdot.psiB.1.u)))
3 mod.pdot.psiB.1.u.oddsratio.browse
```

```
> mod.pdot.psiB.1.u.oddsratio.browse
[1] 3.404722
```

Odds of Occupancy for Browsed areas is 3.41x that of unbrowsed areas.

Cell-effects approach:

```
mod.pdot.psiB.2.u <- unmarked::occu(
     ~1 ~Browse
     data=weta.UMF, se=TRUE)
```

or (prefered)

```
mod.pdot.psiB.2.u <- unmarked::occu(
     ~1 ~BrowCat,
      data=weta.UMF, se=TRUE)
```

No need for you to define indicator variables with the latter model.

Single Species; Single-Season - Covariates - *unmarked*

Cell-effects approach:
Look at design matrix

```
> coef(mod.pdot.psiB.2.u)
     psi(Int) psi(BrowCatN)          p(Int)
    1.1492178    -1.2252670    -0.6222472
```

Cell-effects approach:

$$
\begin{aligned}
logit(\psi_{unbrowsed}) &= \alpha_1(1) + \alpha_2(0) = -0.076 \\
\psi_{unbrowsed} &= 1/(1 + exp(-(-0.076)) = 0.4810 \\
logit(\psi_{browsed} &= \alpha_1(1) + \alpha_2(1) = -0.076 + 1.23 = 1.1493 \\
\psi_{browsed} &= 1/(1 + exp(-(1.1493)) = 0.7594.
\end{aligned}
$$

Cell-effects approach: Estimate the odds ratio

```
1  mod.pdot.psiB.2.u.oddsratio.browse <-
2        exp( sum(c(0,-1,0)*coef(mod.pdot.psiB.2.u)))
```

```
> mod.pdot.psiB.2.u.oddsratio.browse
[1] 3.405075
```

Odds of Occupancy for Browsed areas is 3.41x that of unbrowsed areas.

Odds of Occupancy for Browsed areas is 3.41x that of unbrowsed areas.
Standard errors for the log(odds ratio) can also be found (contact me) and is 0.72.
Then the 95% c.i. for the odds ratio is found as
$(\exp(1.24 - 2 \times 0.72), \exp(1.24 + 2 \times 0.72) = (0.82, 14.6)$.
Note that this covers the value of 1, so there isn't very strong evidence of a browse effect.

The ΔAIC is also within 2 units of the model with no browse effect, so the evidence for a browse effect is minimal.

Which approach is better?

- For a single covariate, it makes no difference.
- For more than one covariate, use the cell effects approach where a factor with m levels has $m - 1$ indicator variables and columns in the design matrix. Otherwise you can end up with a design matrix that is not full rank.

Try fitting a model where detectability also depends on browse status of the site. i.e. $\psi(browse), p(browse)$.

Try fitting a model where detectability also depends on browse status of the site. i.e. $\psi(browse), p(browse)$.

```
1  mod.pB.psiB.u <- unmarked::occu(
2        ~BrowCat ~BrowCat,
3        data=weta.UMF, se=TRUE)
```

Model $\psi(browse), p(browse)$.
Estimated occupancy by browse:

```
> newdata <- data.frame(BrowCat=c("B","N"))
> cbind(newdata,predict(mod.pB.psiB.u, type='state', newdat
  BrowCat Predicted        SE     lower     upper
1       B 0.7563981 0.1277022 0.4439049 0.9235363
2       N 0.4840246 0.1191192 0.2691862 0.7049346
```

Model $\psi(browse), p(browse)$.
Estimated detection by browse:

```
> newdata <- data.frame(BrowCat=c("B","N"))
> cbind(newdata,predict(mod.pB.psiB.u, type='det', newdata=
  BrowCat Predicted         SE     lower     upper
1       B 0.3519078 0.06891004 0.2309670 0.4953830
2       N 0.3451077 0.08602485 0.1999464 0.5263258
```

Try fitting a model where detectability depends on the visit $\psi(browse), p(t)$.

Model $\psi(browse), p(t)$.

```
1  mod.pt.psiB.u <- unmarked::occu(
2        ~Time ~BrowCat,
3        data=weta.UMF, se=TRUE)
```

Single Species; Single-Season - Covariates - *unmarked*

Model $\psi(browse), p(t)$.

```
> newdata <- data.frame(BrowCat=c("B","N"))
> cbind(newdata,predict(mod.pt.psiB.u, type='state', newdat
  BrowCat Predicted        SE     lower     upper
1       B 0.7699156 0.1232913 0.4610099 0.9290342
2       N 0.4931714 0.1116875 0.2884153 0.7002442

> newdata <- expand.grid(Time=c("T1","T2","T3","T4","T5"))
> cbind(newdata,predict(mod.pt.psiB.u, type='det', newdata=
  Time Predicted          SE      lower     upper
1   T1 0.3520566 0.09838300 0.18918319 0.5585560
2   T2 0.3175290 0.08921528 0.17192419 0.5104358
3   T3 0.1694829 0.06707747 0.07424059 0.3417987
4   T4 0.3115818 0.08815495 0.16822723 0.5031935
5   T5 0.5917250 0.10433923 0.38334101 0.7716398
```

Try fitting a model where detectability depends on the observer, but NOT on time $\psi(browse), p(observer)$. Hint: 3 observers need 2 NEW indicator columns. What does the intercept now mean?

Single Species; Single-Season - Covariates - *unmarked*

Model $\psi(\textit{browse}), p(\textit{observer})$.

```
1  mod.p0.psiB.u <- unmarked::occu(
2        ~obs ~BrowCat,
3        data=weta.UMF, se=TRUE)
```

Model $\psi(\textit{browse}), p(\textit{observer})$.

```
> newdata <- data.frame(BrowCat=c("B","N"))
> cbind(newdata,predict(mod.p0.psiB.u, type='state', newdat
  BrowCat Predicted        SE     lower     upper
1       B 0.7535927 0.1164150 0.4723052 0.9126656
2       N 0.4846725 0.1084303 0.2865485 0.6877343

> newdata <- expand.grid(obs=c("1",'2','3'))
> cbind(newdata,predict(mod.p0.psiB.u, type='det', newdata=
  obs Predicted          SE     lower     upper
1   1 0.2235216 0.06273691 0.1241470 0.3689348
2   2 0.3786085 0.07998866 0.2383337 0.5426262
3   3 0.4462474 0.08081043 0.2980095 0.6047043
```

Joint effects of covariates.
Suppose that detectability depended both on occasion effects and observer effects. There are two types of models:

- Additive models. Observers vary among themselves, but are consistent among occasions. For example, one observer has a lower (and consistent) detectability in all occasions even though the detectability varies over occasions. Notation is $p(t + obs)$. Append columns for each covariate.
- Interaction models. Observers are not consistent over occasions. In some days, observer 1 is worst; on other days observer 2 is worst, etc. Notation is $p(t * obs)$. Append columns and then append multiplication of columns.

This is easily done in *unmarked* without having to physically create the extra columns using standard modelling notation of *R*.

Fit the model: $\psi(browse), p(observer + time)$.

Single Species; Single-Season - Covariates - *unmarked*

Model: $\psi(browse), p(observer + time)$.

```
1  mod.pOpV.psiB.u <- unmarked::occu(
2       ~obs+Time ~BrowCat,
3       data=weta.UMF, se=TRUE)
```

Single Species; Single-Season - Covariates - *unmarked*

Model: $\psi(\textit{browse}), p(\textit{observer} + \textit{time})$.

```
> newdata <- data.frame(BrowCat=c("B","N"))
> cbind(newdata,predict(mod.pOpV.psiB.u, type='state', newd
  BrowCat Predicted        SE     lower     upper
1       B 0.7673286 0.1206506 0.4672431 0.9253797
2       N 0.5059973 0.1148964 0.2938184 0.7160377
...
> newdata <- expand.grid(obs=c("1",'2','3'), Time=c("T1","T
> cbind(newdata,predict(mod.pOpV.psiB.u, type='det', newdat
   obs Time  Predicted         SE      lower     upper
1    1   T1 0.21421538 0.08831907 0.08882487 0.4325798
2    2   T1 0.36049060 0.12278182 0.16560616 0.6155310
3    3   T1 0.44282792 0.11677497 0.23915456 0.6677296
4    1   T2 0.18929552 0.08154684 0.07613362 0.3981664
5    2   T2 0.32560546 0.10240877 0.16216606 0.5463518
...
```

Single Species; Single-Season - Covariates - *unmarked*

Fit the following models:

- $\psi(browse), p(observer + time + browse)$.
- $\psi(*), p(observer + time + browse)$.
- $\psi(*), p(observer + time)$.

Construct the AIC table.

```
models.u <-unmarked::fitList(
              mod.pdot.u,
              mod.pdot.psiB.1.u,
              mod.pB.psiB.u,
              mod.pt.psiB.u,
              mod.pO.psiB.u,
              mod.pOpV.psiB.u,
              mod.pOpVpB.psiB.u,
              mod.pOpVpB.psi.u,
              mod.pOpV.psi.u)
aic.table.u <- unmarked::modSel(models.u)
aic.table.u
```

What do you conclude from the AIC table?

```
> aic.table.u
                   nPars    AIC delta  AICwt cumltvWt
mod.pOpV.psiB.u        9 257.60  0.00 0.3370     0.34
mod.pOpV.psi.u         8 258.55  0.95 0.2099     0.55
mod.pOpVpB.psi.u       9 259.41  1.82 0.1360     0.68
mod.pt.psiB.u          7 259.44  1.84 0.1345     0.82
mod.pOpVpB.psiB.u     10 259.60  2.00 0.1240     0.94
mod.pO.psiB.u          5 262.04  4.44 0.0366     0.98
mod.pdot.psiB.1.u      3 264.26  6.66 0.0120     0.99
mod.pdot.u             2 265.79  8.19 0.0056     1.00
mod.pB.psiB.u          4 266.26  8.66 0.0044     1.00
```

Model averaging of the ψ values.

```
1 newdata <- data.frame(BrowCat=c("B","N"), Browsed=c(1,0), U
2 cbind(newdata,predict(models.u, type="state", newdata=newda
```

```
> cbind(newdata,predict(models.u, type="state", newdata=new
  BrowCat Browsed Unbrowsed Predicted        SE     lower
1       B       1         0 0.7274000 0.127004 0.4564418 0.
2       N       0         1 0.5551937 0.132542 0.3434075 0.
```

Single Species; Single-Season - Supplemental

Using the *JAGS* package - a Bayesian solution

Single Species; Single-Season - *JAGS*

JAGS is an program to do Bayesian inference using MCMC.
It is usually called using the *R2Jags* package from *R*.
Why go Bayesian?

- Allows for more complex models than likelihood based methods
- Natural way to model random effect such as spatial autocorrelation
- Natural way to answer questions such a "What is belief (probability) that occupancy is declining over time"

Key disadvantages are:

- NOT FOR THE FAINT HEARTED!
- Goodness-of-fit is difficult to do.
- Model selection is not well developed.

Single Species; Single-Season - *JAGS*

Installation of *JAGS* and *R2Jags* package.

- Search and download *JAGS* from web.
- Download package file from CRAN

Open the sample program in the *Salamander* directory

Single Species; Single-Season - *JAGS* - Salamander

Basic steps in analysis:

- Define the *JAGS* program.
- Input the history file and create the required data vectors.
- Call *JAGS* using the *jags()* function.
- Extract information from the MCMC output.

Refer to sample analysis script.

Read in raw data and so some basic error checking:

```
1  input.data <- readxl::read_excel(file.path("..","salamande
2                                   sheet="CompleteData",
3                                   col_names=FALSE)  # notic
4
5  # do some basic checks on your data
6  nrow(input.data)
7  ncol(input.data)
8  head(input.data)
```

Histories is a matrix ($n_{sites} \times n_{visits}$) of 1's, 0's, or NA's.

Single Species; Single-Season - *JAGS* - Salamander

Create the following data vectors:

- History - a $n_{sites} \times n_{visits}$ vector of 1's and 0's for each site-visit. Drop all site-visits not done.
- Site - vector that matches *History* that identifies the site number.
- Visit - vector that matches *History* that identifies the visit.

```
1 History <- as.vector(unlist(input.data))  # stacks the col
2 Site    <- rep(1:nrow(input.data), ncol(input.data))
3 Visit   <- rep(1:ncol(input.data), each=nrow(input.data))
4 cbind(Site, Visit, History)[1:10,]
```

This gives:

```
> cbind(Site, Visit, History)[1:10,]
      Site Visit History
 [1,]    1     1       0
 [2,]    2     1       0
 [3,]    3     1       0
 [4,]    4     1       1
 [5,]    5     1       0
 [6,]    6     1       0
 [7,]    7     1       0
 [8,]    8     1       0
 [9,]    9     1       0
[10,]   10     1       1
```

Single Species; Single-Season - *JAGS* - Salamander

JAGS program uses a latent (hidden) STATE variables $z[i]$ that is 1 or 0 if the site i is occupied or not.

If at least one detection is made, then $z[i]$ is known to be 1.
If no detections are made, the actual value of $z[i]$ is unknown.

The state (occupied or not) follows a Bernoulli distribution depending on the occupancy probability ψ

```
1     for(i in 1:Nsites){
2        z[i] ~  dbern(psi)
3     }
```

Single Species; Single-Season - *JAGS* - Salamander

JAGS program models the observed data (detect or not) depending on the (latent) state space

If $z[i] = 0$, then $Pr(detect) = 0$
If $z[i] = 1$, then $Pr(detect) = p$

History[j] Bernoulii(z[i] * p) for each visit.

```
1     for(i in 1:Nsites.visits){
2        p.detect[i] <- z[Site[i]]*p
3        History[i] ~ dbern(p.detect[i])
4     }
```

Single Species; Single-Season - *JAGS* - Salamander

Add prior distirbuiton:

```
1     psi ~ dbeta(1,1)
2     p   ~ dbeta(1,1)
```

Define derived variables:

```
1      # number of occupied sites
2     occ.sites <- sum(z[1:Nsites])
3
4     # belief that psi is above some value
5     prob.psi.greater.50 <- ifelse( psi > 0.5, 1, 0)
```

Define the data, initialize the system, and which parameters should be monitored?

```
1   data.list <- list("Nsites","Nvisits","Nsites.visits",
2                     "History", "Site", "Visit") # or
3   ....
4   # Next create the list of parameters to monitor.
5   # The deviance is automatically monitored.
6
7   monitor.list <- c("z","p", "psi", "occ.sites", "prob.psi.gr
```

Call to *JAGS*

```
results <- R2jags::jags(
      data       =data.list,    # list of data variables
      inits      =init.list,    # list/function for initial
      parameters=monitor.list,# list of parameters to moni
      model.file="model.txt",  # file with bugs model
      n.chains=3,
      n.iter  =5000,           # total iterations INCLUDING
      n.burnin=2000,           # number of burning iteratio
      n.thin=2,                # how much to thin
      DIC=TRUE,                # is DIC to be computed?
      working.dir=getwd()     # store results in current wo
      )
```

Whew!

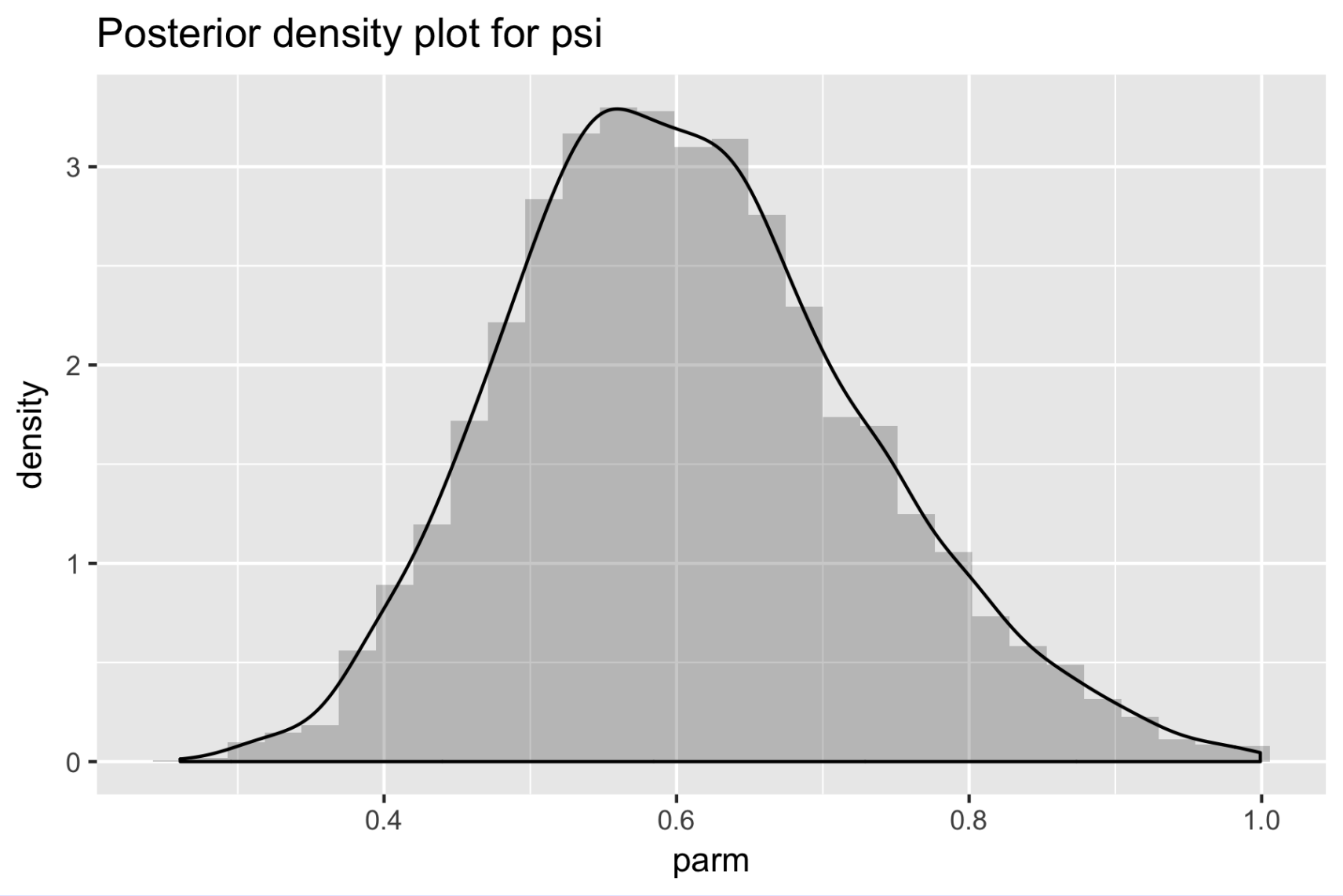
Single Species; Single-Season - *JAGS* - Salamander

MCMC output is long and complex! It will drive you crazy!

```
> results$BUGSoutput$summary[,c("mean", "sd")]
                           mean          sd
occ.sites             23.9548889  3.93020466
p                      0.2605346  0.05551244
prob.psi.greater.50    0.8124444  0.39040023
psi                    0.6075471  0.12102044
z[1]                   1.0000000  0.00000000
z[2]                   1.0000000  0.00000000
....
z[15]                  0.2693333  0.44366274
z[16]                  0.2864444  0.45214982
```

Single Species; Single-Season - *JAGS* - Salamander

MCMC output is long and complex! It will drive you crazy!
Posterior distribution of ψ.



What is $Pr(\psi > 0.50)$?

Single Species; Single-Season - *JAGS* - Salamander

Summary:

- Yikes!
- Don't bother for simple cases, except for easy way to find posterior beliefs.

If you are keen, I'll go over more complex models with you in *JAGS*.

Single Species; Single-Season - Missing Values

Missing Completely at Random:

- Bad weather
- Equipment breakdown

Modify histories in input files as follows:

- *PRESENCE*: Use a "-" (hyphen) to represent a missing value.
- *MARK*: Use a "." (period) to represent a missing value.
- *RPresence*: Use NA to represent a missing value
- *RMark*: Use a "." (period) to represent a missing value in the *ch* variable in the data.frame.
- *unmarked*: Use NA to represent a missing value
- *JAGS*: Need to delete the missing values

Do NOT use a 0 to represent a missing value!

Single Species; Single-Season - Missing Values - *JAGS*

You must remove any missing values from the input vector to *JAGS*.

```
input.data <- readxl::read_excel("Blue_Ridge_pg99.xls",
                                 sheet="SomeMissingValues",
                                 na='-',
                                 col_names=FALSE)  # notice
...
# Remove misstng values in History, Site, Visit, and any o
no.visit <- is.na(History)

Site    <- Site    [ !no.visit]
Visit   <- Visit   [ !no.visit]
History<- History[ !no.visit]
```

Any covariates (e.g. Time) will be ignored for missing values

Common practice is to compute Bayesian posterior predictive checks.
Beyond the scope of this workshop.

Single Species; Single-Season - Covariates - *JAGS*

- In theory, any model can also be fit using *JAGS*, but you must code them "manually" yourself.

Single Species; Single-Season - Covariates - *JAGS*

Mahoenui giant weta (*Deinacrida mahoenui*) is endemic to New Zealand and under stress from rats and other predators.
72 circular plots (3 m radius, primarily prickly gorse plants) were surveyed for weta.
Each plot surveyed 3-5 times.
Covariates to be considered:

- Observer. Three different observers and not every plot surveyed by each observer.
- Browse. Was each site browsed by goats, yes or no.

Getting the data into *JAGS*.

1. Capture History

Get $n_{sites} \times n_{visit}$ data.frame (or matrix) of 1, 0, or NAs

```
input.history <- readxl::read_excel(
     "Weta_pg116.xls",
      sheet="detection_histories",
      na="-",
      col_names=FALSE)  # notice no column names in row 1
```

Detection histories include many missing values. Are these MCAR?

Getting the data into *JAGS*.

2. Site Covariates

Get $n_{sites} \times n_{site-covariates}$ data.frame of site covariates.

- Continuous covariates occupy 1 column
- Categorical covariates can either be alpha-numeric code or a set of indicator variables.

With modern software, the former is preferred.

Single Species; Single-Season - Covariates - *JAGS*

Getting the data into *JAGS*.

2. Site Covariates

```
site_covar <- readxl::read_excel("Weta_pg116.xls",
                                 sheet="site_covar",
                                 na="-",
                                 col_names=TRUE)  # notice

# Create an alternate site level covariate that is a categ
# than indicator variables
site_covar$BrowCat <-
   paste(c("","B")[1+unlist(site_covar[,1])],
         c("","N")[1+unlist(site_covar[,2])], sep="")
xtabs(~BrowCat, data=site_covar,exclude=NULL, na.action=na
colSums(site_covar[,1:2])

head(site_covar)
```

Getting the data into *JAGS*.

2. Site Covariates

```
> head(site_covar)
  Browsed Unbrowsed BrowCat
1       1         0       B
2       1         0       B
3       1         0       B
4       0         1       N
5       1         0       B
6       0         1       N
```

Browse covariate entered as two indicator variables (this has implications later on) or as a categorical variable.
No missing values allowed in site-level covariates.

Getting the data into *JAGS*.

3. Visit Covariates

Get $n_{visit-covariates}$ sets of $n_{sites} \times n_{visits}$ data.frame of visit covariates.

- Continuous covariates have values in each cell
- Categorical covariates can either be alpha-numeric codes or a set of indicator variables.

With modern software, the former is preferred.

Getting the data into *JAGS*.

3. Visit Covariates

```
# Get the individual covariates.
obs1 <- readxl::read_excel("Weta_pg116.xls",
                           sheet="Obs1",
                           na="-",
                           col_names=FALSE)
....

# Create an alternate site level covariate that is a categ
# than indicator variables
obs <- obs1*1 + obs2*2 + obs3*3
```

Notice how I created a single categorical covariate for observer number (preferred)

Getting the data into *JAGS*.

3. Visit Covariates

These must be stacked into a vector of length $n_{sites} \times n_{visits}$ where values for site 1 appear first (for all visit), then for site 2, etc. **The actual order actually doesn't matter that much in JAGS**.

Be sure to add the site-level covariates to the visit covariates.

```
survey.cov <- data.frame(
                History=as.vector(unlist(input.history)),
                Site   =rep(1:nrow(input.history) , ncol(inp
                Visit  =rep(1:ncol(input.history), each=nrow
                obs1   =as.vector(unlist(obs1)),
                 ....
                Obs    =as.character(as.vector(unlist(obs)))
                site_covar,
                stringsAsFactors=FALSE)
head(survey.cov)
```

Single Species; Single-Season - Covariates - *JAGS*

Getting the data into *JAGS*.

3. Visit Covariates

```
> head(survey.cov)
  History Site Visit obs1 obs2 obs3 Obs Browsed Unbrowsed F
1       0    1     1    1    0    0   1       1         0
2       0    2     1    1    0    0   1       1         0
3       0    3     1    1    0    0   1       1         0
4       0    4     1    1    0    0   1       0         1
5       0    5     1    1    0    0   1       1         0
6       0    6     1    1    0    0   1       0         1
```

Notice that History is stored with the survey covariates.

Getting the data into *JAGS*.

3. Visit Covariates

You must remove all missing visits (i.e. not done).

Check that no missing visit covariates for remaining visits.

```
1 survey.cov <- na.omit(survey.cov)
```

No missing values allowed for covariates in observation process.

Single Species; Single-Season - Covariates - *JAGS*

You need to create the design matrix for the ψ and p "manually" using the *model.matrix()* function or *R*.

Example, creating the design matrix for a $p(t)$ model.

```
# In this case, for time effects only
covar.p <- model.matrix( ~as.factor(Visit),
                         data=survey.cov)
covar.p[1:10,]
```

Single Species; Single-Season - Covariates - *JAGS*

Example, creating the design matrix for a $p(t)$ model.

```
> covar.p <- model.matrix( ~as.factor(Visit), data=survey.c
> cbind(Site, Visit, History, covar.p)[c(1:10, (1:10)+Nsite
    Site Visit History (Intercept) as.factor(Visit)2 as.fac
1      1     1       0           1                    0
2      2     1       0           1                    0
...
9      9     1       0           1                    0
10    10     1       1           1                    0
92    20     2       0           1                    1
98    26     2       0           1                    1
...
```

Single Species; Single-Season - Covariates - *JAGS*

You need to create the design matrix for the ψ and p "manually" using the *model.matrix()* function or *R*.

Example, creating the design matrix for a *psi*(*Browse*) model.

```
1  covar.psi <- model.matrix( ~as.factor(BrowCat),
2                data=site_covar)
3  covar.psi[c(1:10),]
```

Single Species; Single-Season - Covariates - *JAGS*

Example, creating the design matrix for a *psi*(*Browse*) model.

```
> cbind(site_covar, covar.psi)[1:10,]
   Browsed Unbrowsed BrowCat (Int) as.factor(BrowCat)N
1        1         0       B             1                    0
2        1         0       B             1                    0
3        1         0       B             1                    0
4        0         1       N             1                    1
5        1         0       B             1                    0
...
```

Single Species; Single-Season - Covariates - *JAGS*

You need to create the design matrix for predictions of ψ "manually". Here, I just obtained the unique elements of the design matrix.

```
> pred.covar.psi <- unique(covar.psi)
> pred.covar.psi
  (Intercept) as.factor(BrowCat)N
1           1                   0
4           1                   1
```

Need to figure out what each row estimates?

Single Species; Single-Season - Covariates - *JAGS*

Create the data list for *JAGS*.

```
data.list <- list("Nsites","Nvisits","Nsites.visits",
                  "History", "Site", "Visit",
                  "Ncovar.p","covar.p",
                  "Ncovar.psi", "covar.psi",
                  "Npred.psi", "pred.covar.psi") # or
```

Single Species; Single-Season - Covariates - *JAGS*

Create the initial values for *JAGS*.

```
1  init.list <- list(
2        list(z=init.z, beta.p=rep(0, Ncovar.p), beta.psi=rep
3        list(z=init.z, beta.p=rep(0, Ncovar.p), beta.psi=rep
4        list(z=init.z, beta.p=rep(0, Ncovar.p), beta.psi=rep
5        )  # end of list of lists of initial values
```

There is a separate set of β terms for ψ and p.

Single Species; Single-Season - Covariates - *JAGS*

Create the monitoring list for *JAGS*.

```
monitor.list <- c("z",
                  "beta.p","p.detect",
                  "beta.psi", "psi",
                  "occ.sites", "prob.psi.greater.50",
                  "pred.psi") # parameters to monitor
```

There is a separate set of β terms for ψ and p.

Single Species; Single-Season - Covariates - *JAGS*

Relevant code fragments from *JAGS*.
State portion.

```
1     # estimate the occupancy probabilities
2     for(i in 1:Nsites){
3        logit(psi[i]) = inprod(beta.psi[1:Ncovar.psi],
4                               covar.psi[i, 1:Ncovar.psi])
5     }
6
7     # set up the state model, i.e. is the site actually occ
8     for(i in 1:Nsites){
9        z[i] ~  dbern(psi[i])
10    }
```

Similar to simple models with nocovariates except we use the model matrix to estimate the ψ.

Single Species; Single-Season - Covariates - *JAGS*

Relevant code fragments from *JAGS*.

Observation process

```
1     # the observation model.
2     for(j in 1:Nsites.visits){
3        logit(p.detect[j]) <- inprod(beta.p[1:Ncovar.p],
4                                     covar.p[j, 1:Ncovar.p])
5        p.z[j] <- z[Site[j]]*p.detect[j]
6        History[j] ~ dbern(p.z[j])
7     }
```

Similar to simple models with nocovariates except we use the model matrix to estimate the p.

Single Species; Single-Season - Covariates - *JAGS*

Relevant code fragments from *JAGS*.

Derived variables.

```
1     # derived variables
2     # number of occupied sites
3     occ.sites <- sum(z[1:Nsites])
4
5     # predicted psi values for values of covariates
6     for(i in 1:Npred.psi){
7        logit(pred.psi[i]) = inprod(beta.psi[1:Ncovar.psi],
8                                    pred.covar.psi[i, 1:Ncova
9     }
10
11    # belief that psi is above some value
12    prob.psi.greater.50 <- ifelse( pred.psi > 0.5, 1, 0)
```

You could also estimate specific values of p in a similar way

Single Species; Single-Season - Covariates - *JAGS*

Run *JAGS* in the usual way

```
results <- R2jags::jags(
      data       =data.list,    # list of data variables
      inits      =init.list,    # list/function for initial
      parameters=monitor.list,# list of parameters to moni
      model.file="model.txt",  # file with bugs model
      n.chains=3,
      n.iter  =5000,           # total iterations INCLUDING
      n.burnin=2000,           # number of burning iteratio
      n.thin=2,                # how much to thin
      DIC=TRUE,                # is DIC to be computed?
      working.dir=getwd()     # store results in current wo
      )
```

Single Species; Single-Season - Covariates - *JAGS*

Selected output from *JAGS*.
Beta coefficients

```
> results$BUGSoutput$summary[,c("mean", "sd")]
                                mean           sd
beta.p[1]                 -0.8771668  0.43429329
beta.p[2]                 -0.1563713  0.52607463
beta.p[3]                 -0.9883317  0.59402437
beta.p[4]                 -0.1423077  0.53232519
beta.p[5]                  0.9859274  0.52514302

beta.psi[1]                7.5989301  5.55585448
beta.psi[2]               -7.2591188  5.37612311
```

Single Species; Single-Season - Covariates - *JAGS*

Selected output from *JAGS*.
Estimates of actual *p* and *psi* for observations and sites.

```
> results$BUGSoutput$summary[,c("mean", "sd")]
                                mean           sd
p.detect[1]                0.3011515  0.08965039
......
p.detect[54]               0.2699550  0.08024488
...
psi[1]                     0.9303305  0.11455006
psi[2]                     0.9303305  0.11455006
...
```

Single Species; Single-Season - Covariates - *JAGS*

Selected output from *JAGS*.
Estimates of predicted ψ

```
> results$BUGSoutput$summary[,c("mean", "sd")]
                                mean          sd
pred.psi[1]                0.9303305  0.11455006
pred.psi[2]                0.5621752  0.13933476
prob.psi.greater.50[1]     0.9980000  0.04468158
prob.psi.greater.50[2]     0.6448889  0.47860016
```

Single Species; Single-Season - Covariates - *JAGS*

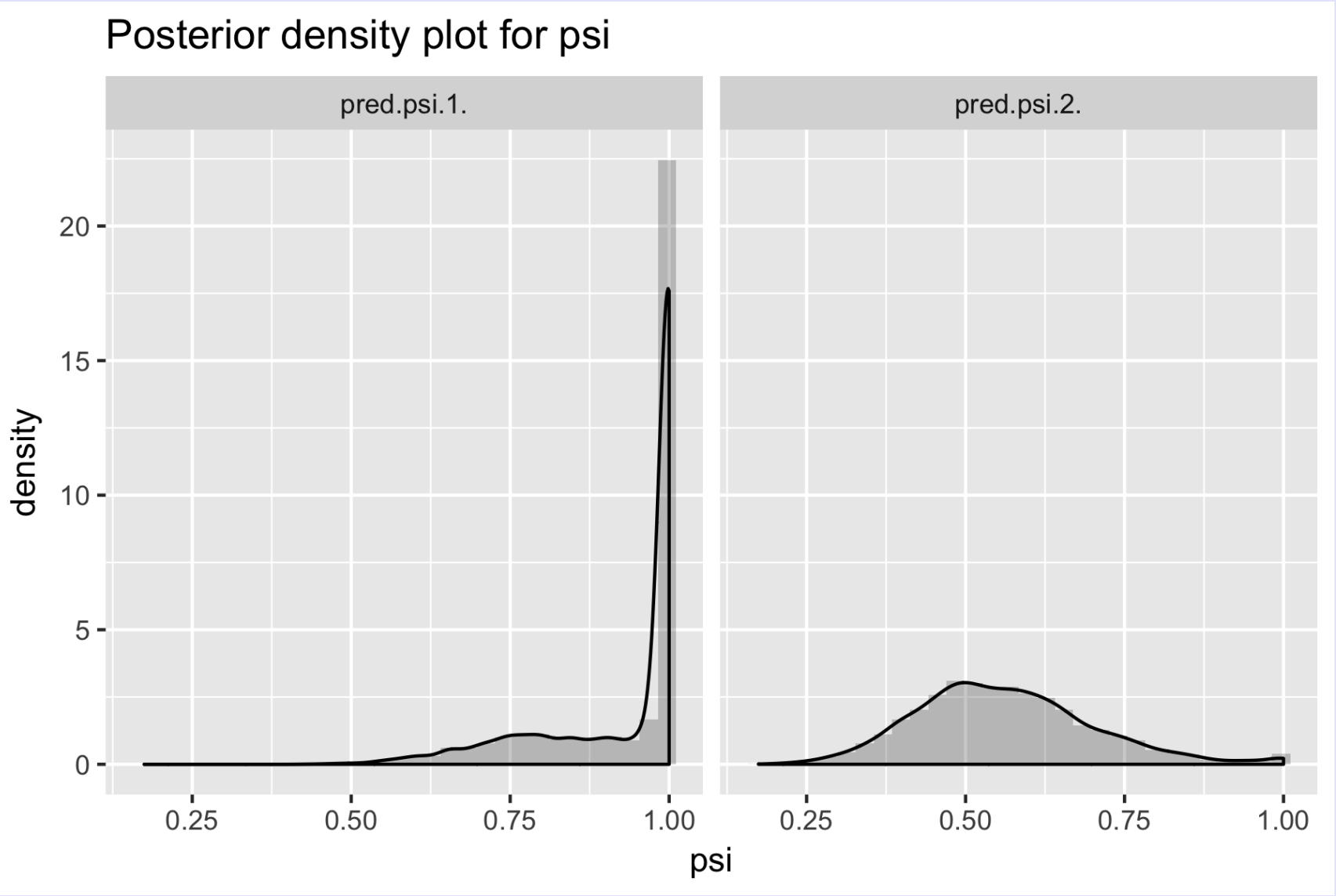
Selected output from *JAGS*.
Posterior probability of occupancy for each site

```
> results$BUGSoutput$summary[,c("mean", "sd")]
                                  mean           sd
z[1]                         0.8382222  0.36828775
z[2]                         0.8375556  0.36889902
z[3]                         1.0000000  0.00000000
z[4]                         0.3124444  0.46354144
z[5]                         0.8362222  0.37011493
...
```

Single Species; Single-Season - Covariates - *JAGS*

Selected output from *JAGS*.

Posterior density of predicted ψ



Summary:

- In theory, any model can be fit, but tedious and you are responsible for creating design matrices.
- Able to get posterior beliefs about ψ (e.g. belief that above a threshold; estimate of actual occupancy of selected sites) easily.
- Model selection can be based on DIC, but there are theoretical difficultiies.
- Model averaging VERY COMPLEX and likely not feasible for mere mortals.