Example session of model selection tools and diagnostics for models fit with the `unmarked` package

Marc J. Mazerolle[*]

20 March 2023

Abstract

The `AICcmodavg` package implements model selection and multimodel inference for a wide range of model types, including those fit with the unmarked package (Fiske and Chandler, 2011). For the latter model type, the `AICcmodavg` package offers additional diagnostic tools and utility functions. Using an example of single-season single-species site analysis, this vignette illustrates these tools.

1 Introduction

Estimating demographic parameters can be challenging in animal populations afflicted by varying probabilities of detection (Williams et al., 2002; Mazerolle et al., 2007). A variety of approaches have been developed to overcome this issue, by estimating detection probabilities explicitly along with the biological parameters of interest (Buckland et al., 2001; Williams et al., 2002; MacKenzie et al., 2006; Royle and Dorazio, 2008). These methods include capture-mark-recapture models, distance sampling, site occupancy models, and N-mixture models (Williams et al., 2002; MacKenzie et al., 2006). Some of these methods are gaining popularity and have been implemented in the `unmarked` package (Fiske and Chandler, 2011). Two recent books by Kéry and Royle (2016) and Kéry and Royle (2021) are excellent introductions to the application of these methods using `unmarked`. Below, I outline an example using various tools from the `AICcmodavg` package.

2 Example using a site occupancy analysis

The basic design for site occupancy analysis consists of repeated visits conducted at a series of sites to detect a species of interest MacKenzie et al. (2006). An important assumption of these analyses is that the state of the site (occupied or not occupied) does not change between the first and last visit. The first example is drawn from a study of the invasive common reed (*Phragmites australis*, haplotype M) on the occupancy of Bullfrogs (*Lithobates catesbeianus*) in 50 wetlands in the Montréal area, Québec,

[*]Département des sciences du bois et de la forêt, Université Laval, Québec, Canada

Canada (Mazerolle, 2015). Three-minute call surveys were conducted at stations on the perimeter of the wetlands on three occasions during the 2009 Bullfrog breeding season. The number of sampling stations depended on wetland size, ranging between 4 and 11 stations (average number of sampling stations on a given visit = 8.67). Minnow traps were also used to sample adults, with a trap deployed at each sampling station during two periods of two consecutive nights. Traps were checked daily to identify and release individuals. Sites were surveyed for invasive common reed. These data are included in the `AICcmodavg` package in the `bullfrog` data frame. The help page of the data frame describes each column (see `?bullfrog`).

2.1 Preparing the data and a few diagnostics

```
> ##load package
> library(AICcmodavg)
> ##load data frame
> data(bullfrog)
```

We can start by checking the structure of the data frame:

```
> ##check data structure
> str(bullfrog)
'data.frame':        50 obs. of  23 variables:
 $ Location     : Factor w/ 50 levels "Arbo_Mc_gill",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Reed.presence: int  0 1 1 1 0 1 1 1 1 1 ...
 $ V1           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ V2           : int  0 0 0 NA 0 0 0 0 0 NA ...
 $ V3           : int  0 0 0 NA 0 0 0 0 0 NA ...
 $ V4           : int  0 0 0 0 0 1 1 1 0 1 ...
 $ V5           : int  0 0 0 NA 0 0 0 0 0 0 ...
 $ V6           : int  0 0 0 NA 0 0 0 0 0 0 ...
 $ V7           : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Effort1      : num  1.334 -0.666 -0.666 -4.666 2.334 ...
 $ Effort2      : num  1.334 -0.666 -0.666 -8.666 2.334 ...
 $ Effort3      : num  1.334 -0.666 -0.666 -8.666 2.334 ...
 $ Effort4      : num  1.334 -0.666 -0.666 -4.666 2.334 ...
 $ Effort5      : num  0.334 -0.666 -0.666 -8.666 2.334 ...
 $ Effort6      : num  0.334 -0.666 -0.666 -8.666 2.334 ...
 $ Effort7      : num  1.334 -0.666 -0.666 -4.666 2.334 ...
 $ Type1        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Type2        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Type3        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Type4        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ Type5        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Type6        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ Type7        : int  0 0 0 0 0 0 0 0 0 0 ...
> ##first rows
> head(bullfrog)
                   Location Reed.presence V1 V2 V3 V4 V5 V6 V7
1              Arbo_Mc_gill             0  0  0  0  0  0  0  0
2        Beauharnois_bassin             1  0  0  0  0  0  0  0
3        Beauharnois_chemin             1  0  0  0  0  0  0  0
```

```
4          Bois_de_liesse_elec            1  0 NA NA  0 NA NA  0
5         Bois_de_liesse_grand            0  0  0  0  0  0  0  0
6 IBoucherville_chenal_a_pinard           1  0  0  0  1  0  0  0
     Effort1    Effort2    Effort3    Effort4    Effort5    Effort6
1  1.3342857  1.3342857  1.3342857  1.3342857  0.3342857  0.3342857
2 -0.6657143 -0.6657143 -0.6657143 -0.6657143 -0.6657143 -0.6657143
3 -0.6657143 -0.6657143 -0.6657143 -0.6657143 -0.6657143 -0.6657143
4 -4.6657143 -8.6657143 -8.6657143 -4.6657143 -8.6657143 -8.6657143
5  2.3342857  2.3342857  2.3342857  2.3342857  2.3342857  2.3342857
6  0.3342857  0.3342857  0.3342857  0.3342857  0.3342857  0.3342857
     Effort7 Type1 Type2 Type3 Type4 Type5 Type6 Type7
1  1.3342857     0     1     1     0     1     1     0
2 -0.6657143     0     1     1     0     1     1     0
3 -0.6657143     0     1     1     0     1     1     0
4 -4.6657143     0     1     1     0     1     1     0
5  2.3342857     0     1     1     0     1     1     0
6  0.3342857     0     1     1     0     1     1     0
```

We will then extract the data to later format into an `unmarkedFrameOccu` object, including the detections, the site variables, and the observation variables:

```
> ##extract detections
> yObs <- bullfrog[, c("V1", "V2", "V3", "V4", "V5", "V6", "V7")]
> ##extract site variables
> siteVars <- bullfrog[, c("Location", "Reed.presence")]
> ##extract observation variables
> ##centered sampling effort on each visit
> effort <- bullfrog[, c("Effort1", "Effort2", "Effort3", "Effort4",
                        "Effort5", "Effort6", "Effort7")]
> ##survey type (0 = call survey, 1 = minnow trap)
> type <- bullfrog[, c("Type1", "Type2", "Type3", "Type4", "Type5",
                      "Type6", "Type7")]
```

Now that the variables have been extracted, we can assemble the `unmarkedFrameOccu` object. This step is important for the unmarked functions to properly retrieve the required information to run the models:

```
> ##load package
> library(unmarked)
> ##format data
> bfrogData <- unmarkedFrameOccu(y = yObs,
                                 siteCovs = siteVars,
                                 obsCovs = list(Type = type, Effort = effort))
```

We can inspect the newly-created object:

```
> summary(bfrogData)
unmarkedFrame Object

50 sites
Maximum number of observations per site: 7
Mean number of observations per site: 6.84
Sites with at least one detection: 23
```

```
Tabulation of y observations:
   0    1 <NA>
 308   34    8

Site-level covariates:
                          Location  Reed.presence
 Arbo_Mc_gill                 : 1   Min.   :0.0
 Beauharnois_bassin           : 1   1st Qu.:0.0
 Beauharnois_chemin           : 1   Median :0.5
 Bois_de_liesse_elec          : 1   Mean   :0.5
 Bois_de_liesse_grand         : 1   3rd Qu.:1.0
 IBoucherville_chenal_a_pinard: 1   Max.   :1.0
 (Other)                      :44

Observation-level covariates:
      Type           Effort
 Min.   :0.0000   Min.   :-8.6657
 1st Qu.:0.0000   1st Qu.:-0.6657
 Median :1.0000   Median : 0.3343
 Mean   :0.5714   Mean   : 0.0000
 3rd Qu.:1.0000   3rd Qu.: 1.3343
 Max.   :1.0000   Max.   : 2.3343
```

The `detHist( )` function in `AICcmodavg` displays the frequencies for each observed detection history, and this function also works with model objects:

```
> detHist(bfrogData)
Summary of detection histories:
          0000000 0000001 0000100 0000110 0001000 0001111 0100000
Frequency      25       2       2       1       6       1       2
          0..0..0 0..1000 1000000 1000100 1001001 1010000 1011001
Frequency       1       1       3       1       1       1       1
          .000000 .000001
Frequency       1       1

Proportion of sites with at least one detection:
 0.46

Frequencies of sites with detections:
         sampled detected
Season-1      50       23
```

Also see functions `detTime( )`, `countDist( )`, and `countHist( )` to summarize time-to-detection data, distance sampling data, or count data.

2.2 Formulating and fitting the candidate models

Next, we can formulate the candidate models. In this example, I am considering four candidate models involving the probabilities of occupancy (ψ) and detection (p):

1. a null model with constant occupancy and detection probability, $\psi(.)p(.)$;
2. a model with constant occupancy but detection probability varying with survey effort (number of stations) and survey type, $\psi(.)p(\text{Effort} + \text{Type})$;

3. a model with occupancy varying with reed presence and constant detection probability, $\psi(\text{Reed})p(.)$;
4. the full model, $\psi(\text{Reed})p(\text{Effort} + \text{Type})$.

We can fit the single-season site occupancy models using the `occu( )` function of `unmarked`. Note that it is recommended to center or standardize numeric explanatory variables recorded on an interval or ratio scale to facilitate convergence of the optimization routine. Here, sampling effort was centered by subtracting the mean number of stations ($\bar{x} = 8.66$) from each value before importing the data set.

```
> ##null model
> m1 <- occu(~ 1 ~ 1, data = bfrogData)
> ##p varies with survey type and effort, occupancy is constant
> m2 <- occu(~ Type + Effort ~ 1, data = bfrogData)
> ##p constant, occupancy varies with reed presence
> m3 <- occu(~ 1 ~ Reed.presence, data = bfrogData)
> ##global model
> m4 <- occu(~ Type + Effort ~ Reed.presence, data = bfrogData)
```

2.3 Inspecting the output and checking model fit

To inspect the output, we can use `summary( )` from `unmarked` or `summaryOD( )` from the `AICcmodavg` package. With the latter, we can request the output in the form of confidence intervals around the estimates or traditional null-hypothesis testing, and we can also correct inferences for overdispersion:

```
> summary(m4)
Call:
occu(formula = ~Type + Effort ~ Reed.presence, data = bfrogData)

Occupancy (logit-scale):
              Estimate   SE     z P(>|z|)
(Intercept)       2.04 1.78  1.14   0.253
Reed.presence    -2.04 1.72 -1.19   0.235

Detection (logit-scale):
            Estimate    SE     z  P(>|z|)
(Intercept)   -1.380 0.340 -4.06 4.84e-05
Type          -1.076 0.400 -2.69 7.11e-03
Effort         0.254 0.177  1.44 1.51e-01

AIC: 213.8576
Number of sites: 50
> summaryOD(m4, out.type = "confint")
Precision unadjusted for overdispersion:

                   Estimate Std. Error Lower 95% CL Upper 95% CL
psi(Int)             2.0365     1.7825      -0.8955        4.969
psi(Reed.presence)  -2.0417     1.7191      -4.8694        0.786
p(Int)              -1.3804     0.3398      -1.9393       -0.822
p(Type)             -1.0762     0.3998      -1.7339       -0.419
p(Effort)            0.2542     0.1769      -0.0367        0.545
```

```
(c-hat = 1)
> summaryOD(m4, out.type = "nhst")
Precision and hypothesis tests unadjusted for overdispersion:

                   Estimate Std. Error z value Pr(>|z|)
psi(Int)             2.0365     1.7825   1.142  0.25325
psi(Reed.presence)  -2.0417     1.7191  -1.188  0.23499
p(Int)              -1.3804     0.3398  -4.063 4.84e-05
p(Type)             -1.0762     0.3998  -2.692  0.00711
p(Effort)            0.2542     0.1769   1.437  0.15062

(c-hat = 1)
```

We can use the above functions to inspect the output of each model, keeping an eye out for issues in model fitting, such as error or warning messages about non convergence, problems with the Hessian matrix, or abnormally large standard errors relative to the estimates. Certain diagnostic tools of the `AICcmodavg` package can be applied to a list of models. Let's create a list of models to store the output. This list will also be necessary for model selection and multimodel inference.

```
> bfrogMods <- list("null" = m1, "psidot.pTypeEffort" = m2,
                    "psiReed.pdot" = m3,
                    "psiReed.pTypeEffort" = m4)
```

We can check for the convergence of the algorithm to find the maximum likelihood estimates, with the `checkConv` function. This can be done for a single model or a list of models:

```
> ##check convergence for a single model
> checkConv(m1)
Converged:  TRUE
> ##extract values across all models
> sapply(bfrogMods, checkConv)
          null psidot.pTypeEffort psiReed.pdot psiReed.pTypeEffort
converged TRUE TRUE               TRUE         TRUE
message   NULL NULL               NULL         NULL
```

The output suggests that the algorithm converged for all four models. Next, we can compute the condition number, which is the ratio of the largest eigenvalue of the Hessian matrix to the smallest value of the same matrix. Typically, large values of the condition number (e.g., 1 000 000) can indicate that a model may be overparameterized. We can get the condition using the `extractCN( )` function:

```
> ##extract condition number of single model
> extractCN(m1)
Condition number            log10
           17.56             1.24
> ##extract condition across all models
> sapply(bfrogMods, extractCN)
```

```
       null     psidot.pTypeEffort psiReed.pdot psiReed.pTypeEffort
CN     17.55675 21.37429           520.0896     276.1269
log10  1.244444 1.329892           2.716078     2.441109
method "svd"    "svd"              "svd"        "svd"
```

None of the models have excessively high condition numbers. Another diagnostic tool that may be useful is the highest standard errors of the estimates in each model or the number of standard errors larger than a given threshold (e.g., 25), obtained with the `checkParms( )` function:

```
> ##check highest SE in single model
> checkParms(m1)
    variable max.se n.high.se
psi psi(Int)    0.8         0
> ##check highest SE across all models
> lapply(bfrogMods, checkParms)
$null
    variable max.se n.high.se
psi psi(Int)    0.8         0

$psidot.pTypeEffort
    variable max.se n.high.se
psi psi(Int)   0.68         0

$psiReed.pdot
    variable max.se n.high.se
psi psi(Int)   3.23         0

$psiReed.pTypeEffort
    variable max.se n.high.se
psi psi(Int)   1.78         0
```

We note that the standard errors of the estimates are not exceptionally high, suggesting that there are no issues with the standard errors. We can the proceed to check the goodness of fit of the global model (Burnham and Anderson, 2002). A first assessment could be based on a comparison between the global model and the null model, using a likelihood-ratio test implemented with `anovaOD( )`. This function also adjusts inferences in case of overdispersion when the `c.hat` argument is specified:

```
> ##compare global model vs null
> anovaOD(mod.simple = m1, mod.complex = m3)
Analysis of deviance table

Simple model  (1):  psi(.)p(.)
Complex model (2):  psi(Reed.presence)p(.)

  K logLik Kdiff -2logLik Chisq Pr(>Chisq)
1 2 -109.8
2 3 -108.0     1    3.674 3.674     0.0553
```

This comparison suggests that the global model is marginally better than the null model. A more formal assessment is based on the MacKenzie and Bailey goodness of fit test, which essentially compares the observed frequencies of the detection histories to the expected frequencies based on a χ^2 statistic, where the significance of the statistic is assessed with a parametric bootstrap (MacKenzie and Bailey, 2004). This test is implemented in the `mb.gof.test( )` function. A total of 1000 to 10000 iterations are recommended (MacKenzie and Bailey, 2004). The function includes arguments to specify the number of cores on the computer (`ncores`) and whether parallel processing should be used to speed up computations (`parallel`). Because the time to complete the goodness-of-fit test depends on your hardware, saving the output in a file using `save( )` is more efficient than running the function each time.

```
> ##this takes 226 min. using 2 cores
> gof <- mb.gof.test(mod = m4, nsim = 10000, parallel = TRUE, ncores = 2)
> gof
> save(gof, file = "gofMod3.Rdata")


MacKenzie and Bailey goodness-of-fit for single-season occupancy model

Pearson chi-square table:

        Cohort Observed Expected Chi-square
0000000      0       25    24.38       0.02
0000001      0        2     3.05       0.36
0000100      0        2     0.92       1.28
0000110      0        1     0.09       9.27
0001000      0        6     3.05       2.84
0001111      0        1     0.01      94.12
0100000      0        2     0.89       1.39
1000000      0        3     3.05       0.00
1000100      0        1     0.28       1.86
1001001      0        1     0.30       1.64
1010000      0        1     0.27       1.98
1011001      0        1     0.03      30.86
0..1000      1        1     0.06      14.03
0..0..0      2        1     0.90       0.01
.000000      3        1     1.40       0.11
.000001      3        1     0.12       6.30

Chi-square statistic = 177.2475
Number of bootstrap samples = 10000
P-value = 0.2039

Quantiles of bootstrapped statistics:
   0%   25%   50%   75%  100%
   13    68   102   159 16130

Estimate of c-hat = 1.08
```

The goodness-of-fit test suggests that the model fits the data with low overdispersion ($P = 0.2039, \hat{c} = 1.08$, Fig. 1). Values of $\hat{c} > 3$ can indicate lack-of-fit in addition to overdispersion (Lebreton et al., 1992). Given that the $\hat{c}$ is very close to 1, adjusting for this overdispersion will make inferences slightly more conservative:

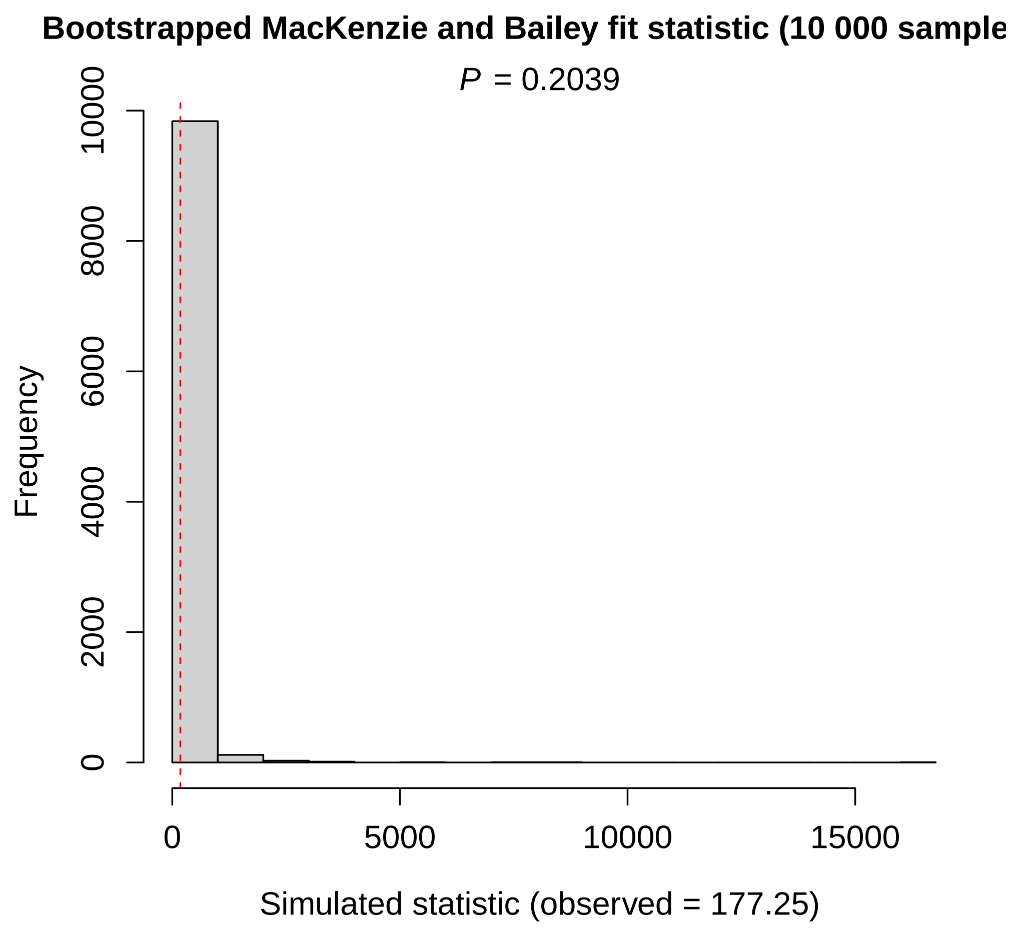

Figure 1: Distribution of chi-square values obtained from parametric bootstrapping from the global site occupancy model.

```
> ##compare inferences
> summaryOD(m3)
Precision unadjusted for overdispersion:

                   Estimate Std. Error Lower 95% CL Upper 95% CL
psi(Int)             2.5781     3.2269      -2.7297        7.886
psi(Reed.presence)  -2.5408     3.0655      -7.5831        2.501
p(Int)              -1.8344     0.2998      -2.3275       -1.341

(c-hat = 1)
> summaryOD(m3, c.hat = 1.08)
Precision adjusted for overdispersion:

                   Estimate Std. Error Lower 95% CL Upper 95% CL
psi(Int)             2.5781     3.3535      -2.9379        8.094
psi(Reed.presence)  -2.5408     3.1858      -7.7809        2.699
p(Int)              -1.8344     0.3115      -2.3468       -1.322
```

```
(c-hat = 1.08)
```

However, differences in the inferences will increase with $\hat{c}$, because the standard errors of the estimates are multiplied by $\hat{c}$.

2.4 Conducting model selection and multimodel inference

Now that we confirmed that the global model fits the data, we can proceed with model selection using `aictab( )`. The default option uses the second-order Akaike information criterion (AIC_c, Sugiura 1978; Hurvich and Tsai 1989). Other information criteria can be chosen such as AIC (Akaike, 1973) by using `second.ord = FALSE`, or their quasi-likelihood versions in the presence of overdispersion, $QAIC$ or $QAIC_c$ by modifying `c.hat` (Lebreton et al., 1992; Burnham and Anderson, 2002). Model selection with the Bayesian information criterion (BIC, Schwarz 1978) is implemented in function `bictab( )`. Below, we use the AIC_c and compare it against the $QAIC_c$:

```
> ##when no overdispersion is present
> outTab <- aictab(cand.set = bfrogMods)
> ##accounting for overdispersion
> outTabC <- aictab(cand.set = bfrogMods, c.hat = 1.08)
> outTab
Model selection based on AICc:

                    K   AICc Delta_AICc AICcWt Cum.Wt      LL
psiReed.pTypeEffort 5 215.22       0.00   0.61   0.61 -101.93
psidot.pTypeEffort  4 216.21       0.99   0.37   0.98 -103.66
psiReed.pdot        3 222.48       7.25   0.02   0.99 -107.98
null                2 223.88       8.66   0.01   1.00 -109.81
> outTabC
Model selection based on QAICc:
(c-hat estimate = 1.08)

                    K  QAICc Delta_QAICc QAICcWt Cum.Wt Quasi.LL
psiReed.pTypeEffort 6 202.71        0.00    0.55   0.55   -94.38
psidot.pTypeEffort  5 203.33        0.62    0.41   0.96   -95.98
psiReed.pdot        4 208.85        6.14    0.03   0.98   -99.98
null                3 209.88        7.17    0.02   1.00  -101.68
```

Two models emerged as being equivalent ($\Delta QAIC_c = 0.62$), with a combined Akaike weight of 0.96. Both models consisted of survey type and effort on detection probability, but varied in terms of reed presence on occupancy. We can use the evidence ratio of Akaike weights to compare the two models:

```
> ##evidence ratio between top-ranked model vs second-ranked model
> evidence(aic.table = outTabC)
Evidence ratio between models 'psiReed.pTypeEffort' and 'psidot.pTypeEffort':
1.36
```

Given that the two models are equivalent and differ only in the reed presence on occupancy, there is little evidence for a variation of bullfrog occupancy with the presence of common reed. We can formally assess the effect of reed presence across the entire model set using the model-averaging shrinkage estimator (`modavgShrink( )`, Burnham and Anderson 2002), based on the model-averaged estimate ($\hat{\bar{\beta}}$) and an unconditional 95% confidence interval (95% CI). Note that we use the `parm.type` argument in `modavgShrink( )` to specify the parameter on which reed presence appears, here `parm.type = "psi"` because reed presence is a variable on the occupancy component of the models:

```
> ##model-averaged estimate of reed presence - shrinkage estimator
> estReed <- modavgShrink(cand.set = bfrogMods,
                          parm = "Reed.presence", parm.type = "psi",
                          c.hat = 1.08)
> estReed
Multimodel inference on "psi(Reed.presence)" based on QAICc

QAICc table used to obtain model-averaged estimate with shrinkage:
        (c-hat estimate = 1.08)

                    K  QAICc Delta_QAICc QAICcWt Estimate   SE
null                3 209.88        7.17    0.02     0.00 0.00
psidot.pTypeEffort  5 203.33        0.62    0.41     0.00 0.00
psiReed.pdot        4 208.85        6.14    0.03    -2.54 3.19
psiReed.pTypeEffort 6 202.71        0.00    0.55    -2.04 1.79

Model-averaged estimate with shrinkage: -1.19
Unconditional SE: 1.75
95% Unconditional confidence interval: -4.63, 2.24
```

Although the top-ranked model included the presence of reed on occupancy, the estimate of reed presence on the occupancy of bullfrogs did not differ from 0 ($\hat{\bar{\beta}}_{\text{Reed}}$ = -1.19, 95% CI : [-4.63, 2.24]). Similarly, we can estimate the effect of survey type and sampling effort on detection probability, now specifying `parm.type = "detect"` because these two variables appear on the detection probability part of the models:

```
> estType <- modavgShrink(cand.set = bfrogMods,
                          parm = "Type", parm.type = "detect",
                          c.hat = 1.08)
> estType
Multimodel inference on "p(Type)" based on QAICc

QAICc table used to obtain model-averaged estimate with shrinkage:
        (c-hat estimate = 1.08)

                    K  QAICc Delta_QAICc QAICcWt Estimate   SE
null                3 209.88        7.17    0.02     0.00 0.00
psidot.pTypeEffort  5 203.33        0.62    0.41    -1.07 0.42
psiReed.pdot        4 208.85        6.14    0.03     0.00 0.00
psiReed.pTypeEffort 6 202.71        0.00    0.55    -1.08 0.42

Model-averaged estimate with shrinkage: -1.03
Unconditional SE: 0.46
95% Unconditional confidence interval: -1.93, -0.13
```

```
> estEffort <- modavgShrink(cand.set = bfrogMods,
                            parm = "Effort", parm.type = "detect",
                            c.hat = 1.08)
> estEffort
Multimodel inference on "p(Effort)" based on QAICc

QAICc table used to obtain model-averaged estimate with shrinkage:
        (c-hat estimate = 1.08)

                    K  QAICc Delta_QAICc QAICcWt Estimate   SE
null                3 209.88        7.17    0.02     0.00 0.00
psidot.pTypeEffort  5 203.33        0.62    0.41     0.28 0.19
psiReed.pdot        4 208.85        6.14    0.03     0.00 0.00
psiReed.pTypeEffort 6 202.71        0.00    0.55     0.25 0.18

Model-averaged estimate with shrinkage: 0.26
Unconditional SE: 0.19
95% Unconditional confidence interval: -0.12, 0.63
```

Detection probability during call surveys was higher than using minnow trapping sessions ($\hat{\bar{\beta}}_{\text{Type}} = -1.03, 95\%\, CI : [-1.93, -0.13]$). However, detection probability did not vary with the number of sampling stations ($\hat{\bar{\beta}}_{\text{Effort}} = 0.26, 95\%\, CI : [-0.12, 0.63]$).

2.5 Plotting results

The next step is to create plots to illustrate the results. Because no single model encompassed all the support, we can make predictions based on the entire set of models. This is implemented in `modavgPred( )`. The approach is similar to `predict( )` which uses a `newdata` argument to specify a new data set to make predictions. To obtain model-averaged predictions with `modavgPred( )`, you must supply a data frame with the `newdata` argument. This data frame must include values for every variable appearing in the component of the candidate models for which predictions are desired (e.g., occupancy or detection probability). A further requirement is that each variable must be of the same class, and factors must use the same reference level than in the original analysis. To facilitate the identification of each variable to include in predictions, the `extractX( )` function summarizes the variables appearing at least once in a given component of the model, as well as their class:

```
> ##variables on psi
> extractX(cand.set = bfrogMods, parm.type = "psi")
Predictors appearing in candidate models:
Reed.presence

Structure of predictors in siteCovs:
 $ Reed.presence: int  0 1 1 1 0 1 1 1 1 1 ...
> ##variables on p
> extractX(cand.set = bfrogMods, parm.type = "detect")
Predictors appearing in candidate models:
Type    Effort

Structure of predictors in obsCovs:
 $ Type  : int  0 1 1 0 1 1 0 0 1 1 ...
 $ Effort: num  1.334 1.334 1.334 1.334 0.334 ...
```

We see that reed presence is the only variable appearing in models on occupancy, whereas the survey type and sampling effort appear in models on detection probability.

To make predictions of occupancy according to reed presence, we could create the following data frame:

```
> reedFrame <- data.frame(Reed.presence = c(0, 1))
```

We would then compute the model-averaged predictions of occupancy in the presence or absence of reed:

```
> outReed <- modavgPred(cand.set = bfrogMods, newdata = reedFrame,
                        parm.type = "psi", c.hat = 1.08)
> outReed
Model-averaged predictions on the response scale
based on entire model set and 95% confidence interval:

  mod.avg.pred uncond.se lower.CL upper.CL
1        0.810     0.196    0.156    0.992
2        0.586     0.189    0.227    0.876
```

Before building the plot, it can be useful to store the predictions in the data frame and take advantage of the `data` argument in `plot( )`:

```
> ##store predictions and confidence intervals in data frame
> reedFrame$fit <- outReed$mod.avg.pred
> reedFrame$low95 <- outReed$lower.CL
> reedFrame$upp95 <- outReed$upper.CL
```

We can now create the plot:

```
> ##create plot
> xvals <- c(0.2, 0.4)
> plot(fit ~ xvals, data = reedFrame,
       ylab = "Probability of occupancy",
       xlab = "Presence of reed",
       ylim = c(0, 1),
       cex = 1.2, cex.axis = 1.2, cex.lab = 1.2,
       xlim = c(0, 0.6),
       xaxt = "n")
> #add x axis
> axis(side = 1, at = xvals,
       labels = c("absent", "present"),
       cex.axis = 1.2)
> ##add error bars
> segments(x0 = xvals, y0 = reedFrame$low95,
           x1 = xvals, y1 = reedFrame$upp95)
```

Unsurprisingly, Figure 2 shows that there is no difference in occupancy in the presence or absence of common reed in the wetland, consistent with the results of the model-averaged β estimate of reed presence shown earlier.

We can use a similar strategy to compute predictions for detection probability for the two survey types. Recall that survey type was coded as 0 (call survey) or 1 (minnow

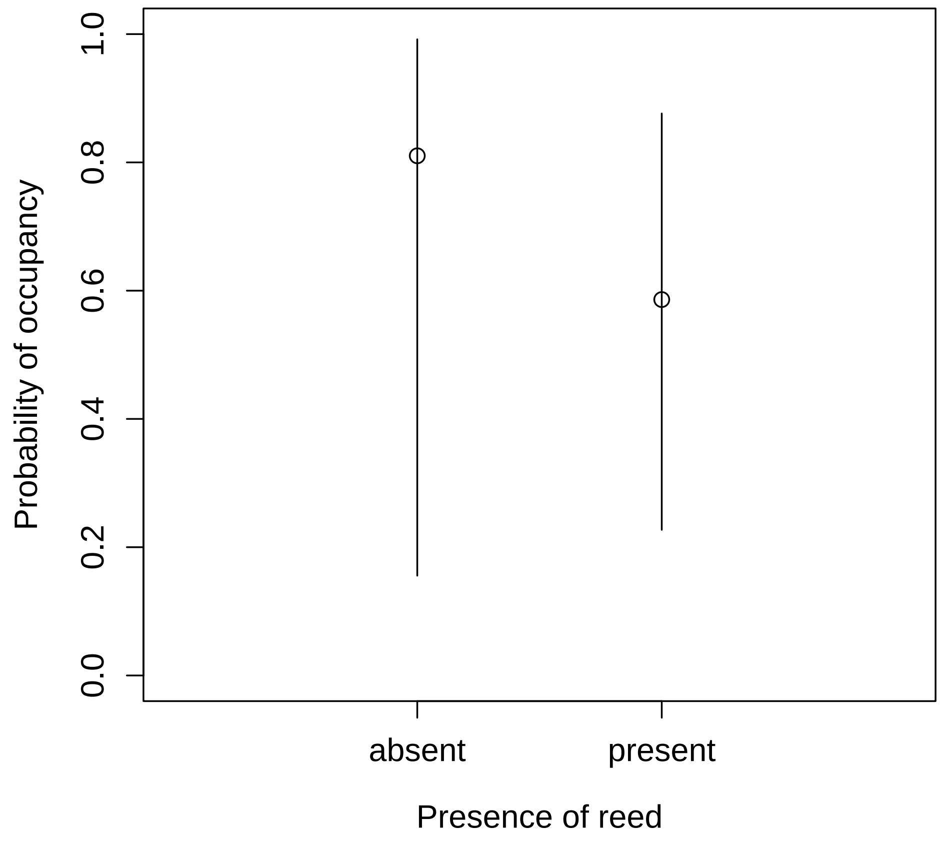

Figure 2: Occupancy probability of bullfrogs (*Lithobates catesbeianus*) in wetlands with and without invasive common reed.

trapping). Two variables appeared on detection probability. To plot detection probability across survey types, we need to hold the other variable constant. We can hold numeric variables constant at their mean (0 if variable was centered or standardized), whereas for binary variables or factors, we must choose a level for predictions.

```
> ##vary Type, hold Effort constant at its mean
> typeFrame <- data.frame(Type = c(0, 1), Effort = 0)
> ##model-averaged predictions
> outType <- modavgPred(cand.set = bfrogMods, newdata = typeFrame,
                        parm.type = "detect", c.hat = 1.08)
> outType
Model-averaged predictions on the response scale
based on entire model set and 95% confidence interval:

  mod.avg.pred uncond.se lower.CL upper.CL
1        0.196     0.057    0.107    0.332
2        0.081     0.031    0.038    0.162
```

We now plot predictions across survey type:

```
> ##store predictions and confidence intervals in data frame
> typeFrame$fit <- outType$mod.avg.pred
> typeFrame$low95 <- outType$lower.CL
> typeFrame$upp95 <- outType$upper.CL
> ##create plot
> xvals <- c(0.2, 0.4)
> plot(fit ~ xvals, data = typeFrame,
       ylab = "Detection probability",
       xlab = "Survey type",
       ylim = c(0, 1),
       cex = 1.2, cex.axis = 1.2, cex.lab = 1.2,
       xlim = c(0, 0.6),
       xaxt = "n")
> #add x axis
> axis(side = 1, at = xvals,
       labels = c("call survey", "minnow trapping"),
       cex.axis = 1.2)
> ##add error bars
> segments(x0 = xvals, y0 = typeFrame$low95,
           x1 = xvals, y1 = typeFrame$upp95)
```

Figure 3 shows that detection probability is higher during call surveys than during minnow trapping periods. We can create the plot for the second explanatory variable (sampling effort) appearing on detection probability.

We note that sampling effort is a numeric variable relating to the number of stations sampled on each visit. To make predictions across sampling effort, we can provide a series of values within the range of values observed in the original data set. Thirty or forty values usually suffice to plot the predicted curve. Sampling effort was recorded as the number of sampling stations and this variable was centered using the average number of sampling stations per visit ($\bar{x} = 8.67$). Working from the standardized variable, we can determine the minimum and maximum values to plot.

```
> ##extract centered values of sampling effort
> effort <- bfrogData@obsCovs$Effort
> ##create a series of 30 values to plot
> Effort.cent <- seq(from = min(effort), to = max(effort),
                     length.out = 30)
> ##back-transform values to original scale of variable
> Effort.mean <- 8.67 #mean of original variable see ?bullfrog
> Effort.orig <- Effort.cent + Effort.mean
```

We can then assemble the variables in a data frame and make model-averaged predictions:

```
> ##note that all variables on the parameter must appear here
> pred.dataEffort <- data.frame(Effort.orig = Effort.orig,
                                Effort = Effort.cent, #centered variable
                                Type = 1)
> #Recall that \texttt{Type} was coded 1 (minnow trap) or 0 (call survey)
>
> ##compute model-averaged predictions with modavgPred on probability scale
```

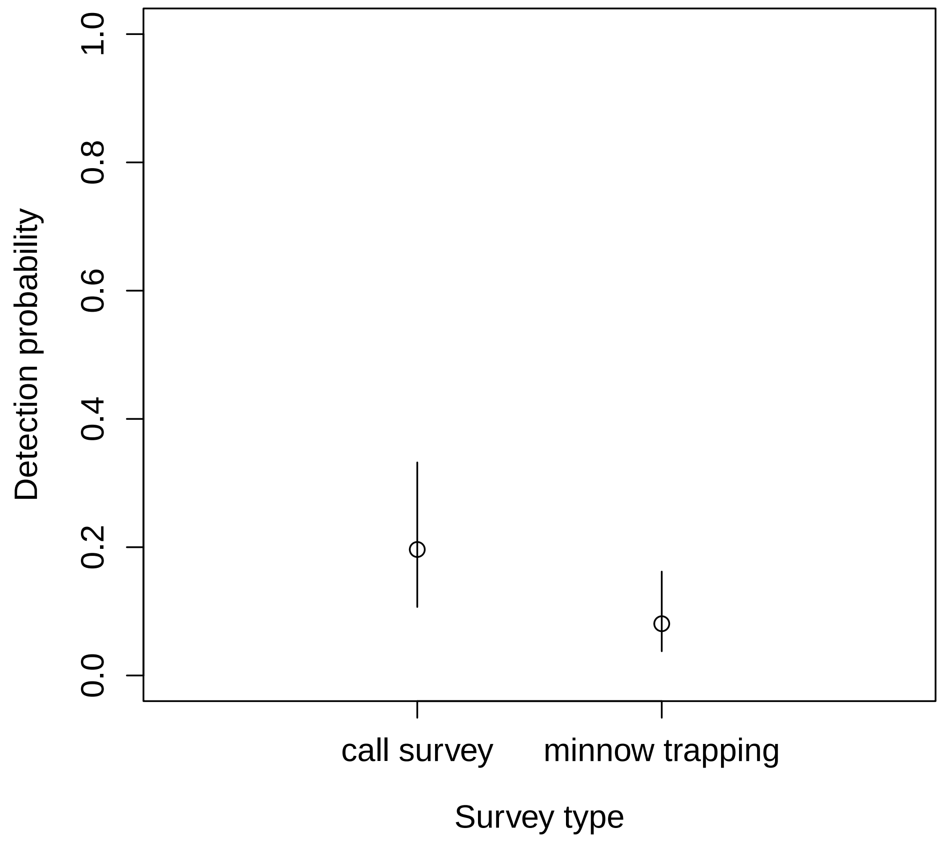

Figure 3: Detection probability of bullfrogs (*Lithobates catesbeianus*) in wetlands during call surveys and minnow trapping surveys.

```
> out.predsEffort <- modavgPred(cand.set = bfrogMods,
                                newdata = pred.dataEffort, parm.type = "detect",
                                type = "response", c.hat = 1.08)
```

Next, we add the predictions and confidence limits to the data frame containing the variables used for predictions:

```
> ##add predictions to data set to keep everything in the same place
> pred.dataEffort$fit <- out.predsEffort$mod.avg.pred
> pred.dataEffort$se.fit <- out.predsEffort$uncond.se
> pred.dataEffort$low95 <- out.predsEffort$lower.CL
> pred.dataEffort$upp95 <- out.predsEffort$upper.CL
```

We can now plot the predictions:

```
> ##create plot
>
> ##plot
> plot(fit ~ Effort.orig,
       ylab = "Detection probability",
       xlab = "Sampling effort",
       ylim = c(0, 1),
       type = "l",
       cex = 1.2, cex.lab = 1.2, cex.axis = 1.2,
       data = pred.dataEffort)
> ##add 95% CI around predictions
> lines(low95 ~ Effort.orig, data = pred.dataEffort,
        lty = "dashed")
> lines(upp95 ~ Effort.orig, data = pred.dataEffort,
        lty = "dashed")
```

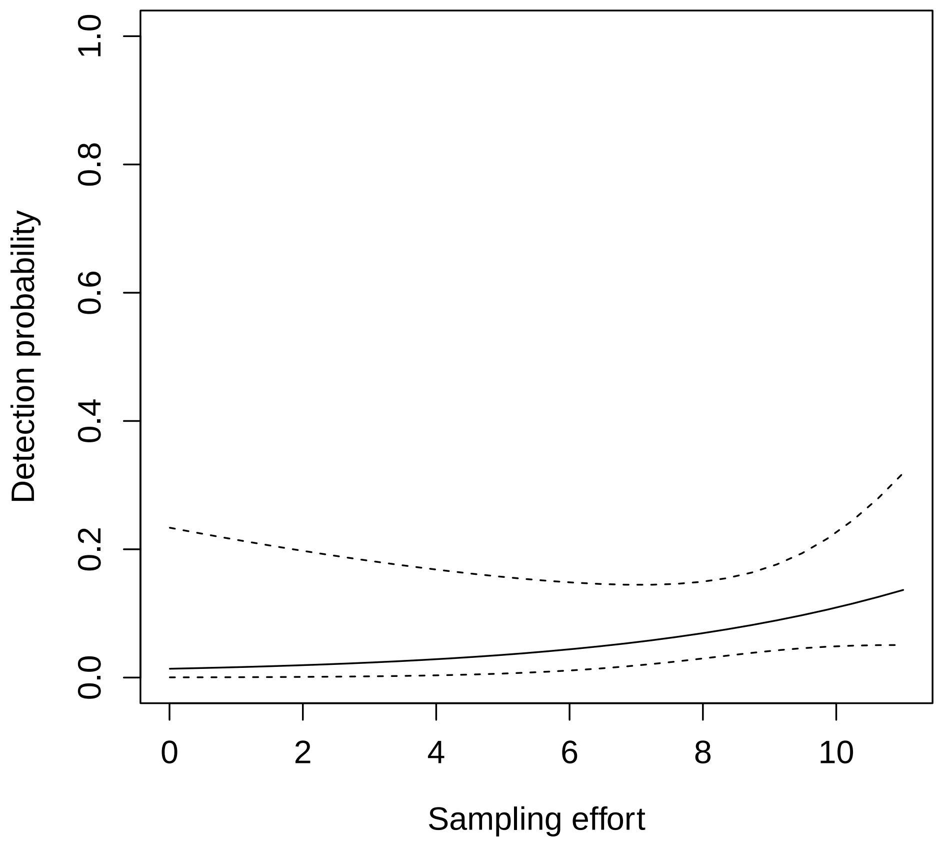

Figure 4: Detection probability of bullfrogs (*Lithobates catesbeianus*) in wetlands during call surveys and minnow trapping surveys.

Figure 4 does not suggest large changes in detection probability within the range of values of sampling effort observed during the study.

3 Additional methods to export output

For users familiar with `Markdown` or LaTeX, the *AICcmodavg* package offers a number of utility functions to convert output to LaTeX tables. You must load the `xtable` package first to enable this functionality. For example, we can extract the results of a single model:

```
> library(xtable)
> xtable(summaryOD(m4))
```

```
% latex table generated in R 4.4.3 by xtable 1.8-4 package
% Wed Mar  5 17:25:26 2025
\begin{table}[ht]
\centering
\begin{tabular}{lrrrr}
  \hline
 & estimate & se & lowlim & upplim \\
  \hline
psi(Int) & 2.04 & 1.78 & -0.90 & 4.97 \\
  psi(Reed.presence) & -2.04 & 1.72 & -4.87 & 0.79 \\
  p(Int) & -1.38 & 0.34 & -1.94 & -0.82 \\
  p(Type) & -1.08 & 0.40 & -1.73 & -0.42 \\
  p(Effort) & 0.25 & 0.18 & -0.04 & 0.55 \\
   \hline
\end{tabular}
\end{table}
```

We can format the output the model selection table:

```
> xtable(outTabC)
```

```
% latex table generated in R 4.4.3 by xtable 1.8-4 package
% Wed Mar  5 17:25:26 2025
\begin{table}[ht]
\centering
\begin{tabular}{lrrrrrr}
  \hline
 & Model & K & QAICc & Delta QAICc & QAICc weight & log-Likelihood \\
  \hline
4 & psiReed.pTypeEffort &   6 & 202.71 & 0.00 & 0.55 & -94.38 \\
  2 & psidot.pTypeEffort &   5 & 203.33 & 0.62 & 0.41 & -95.98 \\
  3 & psiReed.pdot &   4 & 208.85 & 6.14 & 0.03 & -99.98 \\
  1 & null &   3 & 209.88 & 7.17 & 0.02 & -101.68 \\
   \hline
\end{tabular}
\end{table}
```

We can also format the results of multimodel inference to a LaTeX table:

```
> xtable(estReed)
```

```
% latex table generated in R 4.4.3 by xtable 1.8-4 package
% Wed Mar  5 17:25:26 2025
\begin{table}[ht]
\centering
\begin{tabular}{lrrrr}
```

```
  \hline
 & Model-averaged beta estimate & Unconditional SE & 95\% lower limit & 95\% upper limit \\
  \hline
psi(Reed.presence) & -1.19 & 1.75 & -4.63 & 2.24 \\
   \hline
\end{tabular}
\end{table}
```

Other possibilities include the detection history summary:

```
> xtable(detHist(m3))
% latex table generated in R 4.4.3 by xtable 1.8-4 package
% Wed Mar  5 17:25:26 2025
\begin{table}[ht]
\centering
\begin{tabular}{lrr}
  \hline
 & sampled & detected \\
  \hline
Season-1 & 50 & 23 \\
   \hline
\end{tabular}
\end{table}
```

We can also obtain the LaTeXversion of the chi-square table when checking model fit:

```
> xtable(mb.chisq(m3))
% latex table generated in R 4.4.3 by xtable 1.8-4 package
% Wed Mar  5 17:25:26 2025
\begin{table}[ht]
\centering
\begin{tabular}{lrrrrr}
  \hline
 & Detection history & Cohort & Observed & Expected & Chi-square \\
  \hline
0000000 & 0000000 &   0 & 25.00 & 24.10 & 0.03 \\
  0000001 & 0000001 &   0 & 2.00 & 1.92 & 0.00 \\
  0000100 & 0000100 &   0 & 2.00 & 1.92 & 0.00 \\
  0000110 & 0000110 &   0 & 1.00 & 0.31 & 1.57 \\
  0001000 & 0001000 &   0 & 6.00 & 1.92 & 8.66 \\
  0001111 & 0001111 &   0 & 1.00 & 0.01 & 125.80 \\
  0100000 & 0100000 &   0 & 2.00 & 1.92 & 0.00 \\
  1000000 & 1000000 &   0 & 3.00 & 1.92 & 0.61 \\
  1000100 & 1000100 &   0 & 1.00 & 0.31 & 1.57 \\
  1001001 & 1001001 &   0 & 1.00 & 0.05 & 18.46 \\
  1010000 & 1010000 &   0 & 1.00 & 0.31 & 1.57 \\
  1011001 & 1011001 &   0 & 1.00 & 0.01 & 125.80 \\
  0..1000 & 0..1000 &   1 & 1.00 & 0.04 & 23.83 \\
  0..0..0 & 0..0..0 &   2 & 1.00 & 0.82 & 0.04 \\
  .000000 & .000000 &   3 & 1.00 & 1.40 & 0.11 \\
  .000001 & .000001 &   3 & 1.00 & 0.07 & 13.02 \\
   \hline
\end{tabular}
\end{table}
```

Table 1 presents the different `xtable` methods useful with objects created with the `AICcmodavg` package. A number of options are available to suppress certain columns in the table, in addition to including captions or table labels:

```
> #add caption, suppress log-likelihood, and include cumulative Akaike weight
> print(xtable(outTabC,
               caption = "Model selection accounting for overdispersion in the bullfrog data.'
               include.LL = FALSE, include.Cum.Wt = TRUE),
        caption.placement = "top", include.rownames = FALSE)
```

```
% latex table generated in R 4.4.3 by xtable 1.8-4 package
% Wed Mar  5 17:25:26 2025
\begin{table}[ht]
\centering
\caption{Model selection accounting for overdispersion in the bullfrog data.}
\begin{tabular}{rrrrrr}
  \hline
Model & K & QAICc & Delta QAICc & QAICc weight & Cumulative weight \\
  \hline
psiReed.pTypeEffort &   6 & 202.71 & 0.00 & 0.55 & 1.08 \\
  psidot.pTypeEffort &   5 & 203.33 & 0.62 & 0.41 & 1.08 \\
  psiReed.pdot &   4 & 208.85 & 6.14 & 0.03 & 1.08 \\
  null &   3 & 209.88 & 7.17 & 0.02 & 1.08 \\
   \hline
\end{tabular}
\end{table}
```

References

Akaike, H. 1973. Second International Symposium on Information Theory, chapter Information theory as an extension of the maximum likelihood principle, pages 267–281. Akadémiai Kiadó, Budapest, Hungary.

Buckland, S. T., D. R. Anderson, K. P. Burnham, J. L. Laake, D. L. Borchers, and L. Thomas. 2001. Introduction to distance sampling: estimating abundance of biological populations. Oxford University Press, New York, USA.

Burnham, K. P. and D. R. Anderson. 2002. Model selection and multimodel inference: a practical information-theoretic approach, second edition. Springer-Verlag, New York, USA.

Fiske, I. and R. Chandler. 2011. `unmarked`: an `R` package for fitting hierarchical models of wildlife occurrence and abundance. Journal of Statistical Software 43:1–23.

Hurvich, C. M. and C.-L. Tsai. 1989. Regression and time series model selection in small samples. Biometrika 76:297–307.

Kéry, M. and J. A. Royle. 2016. Applied hierarchical modeling in ecology: analysis of distribution, abundance and species richness in R and BUGS. Volume 1: Prelude and static models. Academic Press, San Diego, California, USA.

Kéry, M. and J. A. Royle. 2021. Applied hierarchical modeling in ecology: analysis of distribution, abundance and species richness in R and BUGS. Volume 2: Dynamic and advanced models. Academic Press, San Diego, California, USA.

Lebreton, J.-D., K. P. Burnham, J. Clobert, and D. R. Anderson. 1992. Modeling survival and testing biological hypotheses using marked animals: a unified approach with case-studies. Ecological Monographs 62:67–118.

MacKenzie, D. I. and L. L. Bailey. 2004. Assessing the fit of site-occupancy models. Journal of Agricultural, Biological, and Environmental Statistics 9:300–318.

MacKenzie, D. I., J. D. Nichols, J. A. Royle, K. H. Pollock, L. L. Bailey, and J. E. Hines. 2006. Occupancy estimation and modeling: inferring patterns and dynamics of species occurrence. Academic Press, New York, USA.

Mazerolle, M. J. 2015. Estimating detectability and biological parameters of interest with the use of the R environment. Journal of Herpetology 49:541–559.

Mazerolle, M. J., L. L. Bailey, W. L. Kendall, J. A. Royle, S. J. Converse, and J. D. Nichols. 2007. Making great leaps forward: accounting for detectability in herpetological field studies. Journal of Herpetology 41:672–689.

Royle, J. A. and R. M. Dorazio. 2008. Hierarchical modeling and inference in ecology: the analysis of data from populations, metapopulations and communities. Academic Press, New York, USA.

Schwarz, G. 1978. Estimating the dimension of a model. Annals of Statistics 6:461–464.

Sugiura, N. 1978. Further analysis of the data by Akaike's information criterion and the finite corrections. Communications in Statistics: Theory and Methods A7:13–26.

Williams, B. K., J. D. Nichols, and M. J. Conroy. 2002. Analysis and management of animal populations. Academic Press, New York, USA.

Table 1: Methods of the `xtable` package extended for objects created with the `AICcmodavg` package.

`AICcmodavg` function producing result	Additional arguments supplied to `xtable`
`aictab`	`nice.names = TRUE`, `include.AICc = TRUE`, `include.LL = TRUE`, `include.Cum.Wt = FALSE`
`bictab`	`nice.names = TRUE`, `include.BIC = TRUE`, `include.LL = TRUE`, `include.Cum.Wt = FALSE`
`boot.wt`	`nice.names = TRUE`, `include.AICc = TRUE`, `include.AICcWt = FALSE`
`dictab`	`nice.names = TRUE`, `include.DIC = TRUE`, `include.Cum.Wt = FALSE`
`ictab`	`nice.names = TRUE`, `include.IC = TRUE`, `include.LL = TRUE`, `include.Cum.Wt = FALSE`
`countDist`	`nice.names = TRUE`, `table.countDist = "distance"`
`countHist`	`nice.names = TRUE`, `table.countHist = "count"`
`detHist`	`nice.names = TRUE`, `table.detHist = "freq"`
`detTime`	`nice.names = TRUE`, `table.detTime = "freq"`
`anovaOD`	`nice.names = TRUE`, `include.BIC = TRUE`, `include.LL = TRUE`, `include.Cum.Wt = FALSE`
`summaryOD`	`nice.names = TRUE`
`mb.chisq`	`nice.names = TRUE`, `include.detection.histories = TRUE`
`checkParms`	`nice.names = TRUE`, `include.variable = TRUE`, `include.max.se = TRUE`, `include.n.high.se = TRUE`
`modavg`	`nice.names = TRUE`, `print.table = FALSE`
`modavgCustom`	`nice.names = TRUE`, `print.table = FALSE`
`modavgEffect`	`nice.names = TRUE`, `print.table = FALSE`
`modavgIC`	`nice.names = TRUE`, `print.table = FALSE`
`modavgShrink`	`nice.names = TRUE`, `print.table = FALSE`
`modavgPred`	`nice.names = TRUE`
`multComp`	`nice.names = TRUE`, `print.table = FALSE`