G.M. Berberich, C.F. Dormann, D. Klimetzek, M.B. Berberich, N.J. Sanders & A.M. Ellison

Appendix S2

The supplement material includes the data set, the commented R-code for all our analyses and figures and is provided as online supplementary material. All data and raw R and JAGS codes are available from the Harvard Forest Data Archive (http://harvardforest.fas.harvard.edu:8080/exist/apps/datasets/showData.html?id=hf286), dataset HF286.

Authors

Gabriele M. Berberich,[1,] Carsten F. Dormann,[2] Dietrich Klimetzek,[2] Martin B. Berberich,[3] Nathan J. Sanders,[4] and Aaron M. Ellison[5]*

[1]University Duisburg-Essen, Faculty of Biology, Department of Geology, Universitätsstr. 5, 45141 Essen, Germany

[2]Albert-Ludwigs-University of Freiburg, Faculty of Environment and Natural Resources, Department of Biometry and Environmental System Analysis, Tennenbacher Str. 4, 79085 Freiburg, Germany

[3]IT-Consulting Berberich, Am Plexer 7, 50374 Erftstadt, Germany

[4]Center for Macroecology, Evolution and Climate, Natural History Museum of Denmark, University of Copenhagen, Copenhagen 2100, Denmark

[5]Harvard University, Harvard Forest, 324 North Main Street, Petersham, Massachusetts, 01366 USA

†Corresponding author: Gabriele Berberich, email: gabriele.berberich@uni-due.de

List of files

Berberich etal_DP_Appendix S2_Supplement_AllAnalyses.pdf
Appendix_allAnalyses.html
Appendix_allAnalyses.Rmd
Appendix_allAnalyses_files
nestPlots.csv
nestRecords.csv
nestSizes.csv
augAnalysisWithoutSizes-1.png

augmentationCode-1.png
BayesBinomial-1.png
BayesPosteriorPlots-1.png
BayesvsAugmented-1.png
bubblePlot-1.png
corBayesML-1.png
nestmatrixplot-1.png
nestSizeGLMER-1.png
unnamed-chunk-12-1.png
unnamed-chunk-5-1.png
unnamed-chunk-6-1.png
unnamed-chunk-7-1.png
unnamed-chunk-8-1.png
unnamed-chunk-9-1.png

Use of files

A detailed description of how the files are to be used is given in this Appendix.

Statistical analyses accompanying: Detection probabilities for sessile organisms

G.M. Berberich, C.F. Dormann, D. Klimetzek, M.B. Berberich, N.J. Sanders & A.M. Ellison

19 June 2016

Contents

1 Introduction

This document contains the statistical analyses accompanying the paper "Detection probabilities for sessile organisms" by Berberich et al (2016). It presents the R-code and results for full reproducibility.

The analysis is carried out in three parts:

1. Evaluation whether nest size or plot-level predictors have an effect on detection.
2. The analysis of the number of nests across all plots using patch-occupancy models with data augmentation and a event-specific covariate (nest size); this results in estimates of how many nests were overlooked in total.
3. A Bayesian detection probability model across observers, based on a binomial sampling model. This model does **not** include nest sizes and is thus much simpler to implement.

4. A maximum likelihood version of the previous model. We use this model to *quickly* run the analysis for different sets of observers. It would take years to run model 1 for thousands of combinations of observers, and hence we had to resort to a maximum likelihood version. In fact, model 2 primarily serves as a link between these two models, illustrating that the maximum-likelihood model yields estimates similar to model 2, and that the main benefit of the data-augmentation approach is the incorporation of nest sizes.

2 The data

We have three data sets: plots, nests, and sizes. plots contains the misidentification-corrected recorded nests ($N_{i,s}^{obs}$) for each of the 8 observers (in columns: O1 to O8) for each of the 16 plots (in rows: Plot 1 to Plot 16). As additional columns it contains the total number of different nests recorded at each plot, which is our lower bound (N_s^{obs}) for the true number of nests at each plot ($\hat{N}_s$).

```r
plots <- read.csv("nestPlots.csv", row.names = 1)
```

The second data set, nests, is a long version of plots, in that it contains for each observer the information which nests he/she has detected (actual confirmed nests only).

```r
nests <- read.csv("nestRecords.csv", row.names = 1)
```

The third data set contains the nest sizes estimated roughly from the photographs (height, in cm, along with the variables diameters, locations and forest setting).

```r
sizes <- read.csv("nestSizes.csv", row.names = 1)  # read file in again to get all nest sizes
sizes$Height[which(sizes$Height > 100)] <- 100  # moves 1 nest to smaller size
sizes$Diameter[which(sizes$Diameter > 200)] <- 200  # moves 3 nests
nestSize <- sizes[, 3]
```

3 Effect of covariates on detection probability

3.1 Univariate exploration of predictors for detection probability

Across all observers, nest size (height or diameter) or landscape setting (location, forest type) may affect detection. Here we use a GLM to find out. First, for each nest we compute how many observers detected it. Then we relate this proportion to nest size, etc.

3.1.1 Nest size and diameter

```r
# join tables (sorted in the same way):
detnetsize <- cbind.data.frame(rowSums(nests), 8 - rowSums(nests),
    sizes)
summary(fmHeight <- glm(as.matrix(detnetsize[, 1:2]) ~ poly(Height,
    2), family = quasibinomial, data = detnetsize))
```

```
Call:
glm(formula = as.matrix(detnetsize[, 1:2]) ~ poly(Height, 2),
    family = quasibinomial, data = detnetsize)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1474  -1.4109  -0.1274   1.4317   4.1385

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        0.01676    0.10023   0.167  0.86744
poly(Height, 2)1   6.73074    1.21899   5.522 1.52e-07 ***
poly(Height, 2)2  -3.77773    1.21330  -3.114  0.00223 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 2.673435)

    Null deviance: 556.86  on 146  degrees of freedom
Residual deviance: 443.53  on 144  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4
```

```
predsHeight <- predict(fmHeight, newdata = data.frame(Height = 10:100),
    se.fit = T)
summary(fmDiameter <- glm(as.matrix(detnetsize[, 1:2]) ~ poly(Diameter,
    2), family = quasibinomial, data = detnetsize))
```

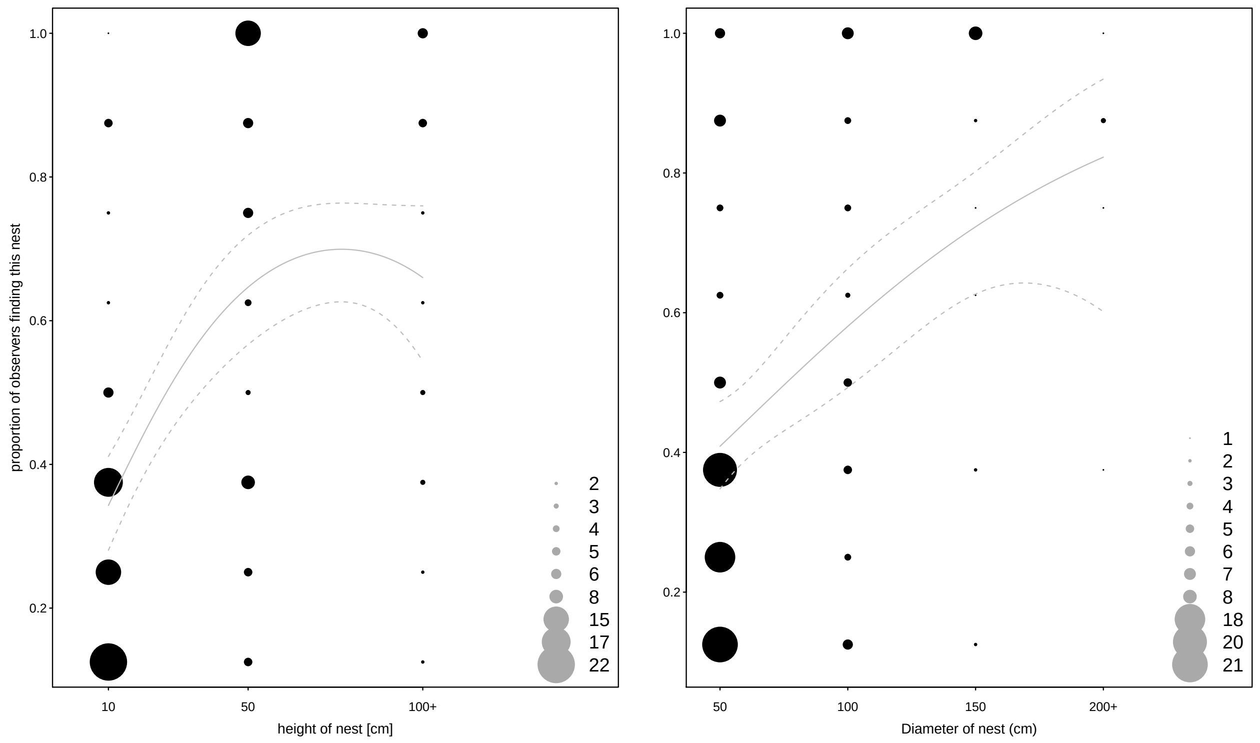
```
Call:
glm(formula = as.matrix(detnetsize[, 1:2]) ~ poly(Diameter, 2),
    family = quasibinomial, data = detnetsize)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5483  -1.1682  -0.1953   1.4989   3.7833

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)          0.03257    0.10276   0.317    0.752
poly(Diameter, 2)1   6.66866    1.37642   4.845 3.24e-06 ***
poly(Diameter, 2)2  -0.26763    1.36859  -0.196    0.845
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for quasibinomial family taken to be 2.865521)

    Null deviance: 556.86  on 146  degrees of freedom
Residual deviance: 477.70  on 144  degrees of freedom
AIC: NA

Number of Fisher Scoring iterations: 4
```

```r
predsDiameter <- predict(fmDiameter, newdata = data.frame(Diameter = 50:200),
    se.fit = T)
with(detnetsize, cor(Height, Diameter))
```

```
[1] 0.7910949
```

We can use a bubble plot to visualise this.

```r
# pdf('Fig2-2panel.pdf', width=6, height=3)

par(mfrow = c(1, 2), mar = c(4, 4, 1, 1), tcl = -0.125, mgp = c(1.25,
    0.25, 0))
cex.vec <- as.vector(table(rowSums(nests), sizes$Height))
x.vec <- rep(c(10, 50, 100), each = 8)
y.vec <- rep(seq(0.125, 1, by = 0.125), times = 3)
plot(x.vec, y.vec, las = 1, ylab = "proportion of observers finding this nest",
    xlab = "height of nest [cm]", xlim = c(0, 150), pch = 16, cex = cex.vec/5,
    axes = F, cex.lab = 0.75, font = 2)
legend("bottomright", bty = "n", pch = 16, pt.cex = sort(unique(cex.vec/5))[-1],
    legend = paste("  ", sort(unique(cex.vec))[-1]), col = "darkgrey",
    cex = 1)
axis(side = 1, at = c(10, 50, 100), cex.axis = 0.7, labels = c("10",
    "50", "100+"))
axis(side = 2, las = 1, cex.axis = 0.7)
box()
lines(10:100, plogis(predsHeight$fit), lwd = 1, col = "grey")
lines(10:100, plogis(predsHeight$fit + 2 * predsHeight$se.fit), lwd = 1,
    lty = 2, col = "grey")
lines(10:100, plogis(predsHeight$fit - 2 * predsHeight$se.fit), lwd = 1,
    lty = 2, col = "grey")

# same for diameter:
par(mar = c(4, 2, 1, 3))
cex.vec1 <- as.vector(table(rowSums(nests), detnetsize$Diameter))
x.vec1 <- rep(c(50, 100, 150, 200), each = 8)
y.vec1 <- rep(seq(0.125, 1, by = 0.125), times = 4)
plot(x.vec1, y.vec1, las = 1, ylab = "", xlab = "Diameter of nest (cm)",
    pch = 16, cex = cex.vec1/5, axes = F, xlim = c(45, 250), ylim = c(0.1,
        1), cex.lab = 0.75, font = 2)
legend("bottomright", bty = "n", pch = 16, col = "darkgrey", pt.cex = sort(unique(cex.vec1/5))[-1],
    legend = paste("  ", sort(unique(cex.vec1))[-1]), cex = 1)
axis(side = 1, at = c(50, 100, 150, 200), labels = c("50", "100",
    "150", "200+"), cex.axis = 0.7)
axis(side = 2, las = 1, cex.axis = 0.7)
box()
lines(50:200, plogis(predsDiameter$fit), lwd = 1, col = "grey")
lines(50:200, plogis(predsDiameter$fit + 2 * predsDiameter$se.fit),
    lwd = 1, lty = 2, col = "grey")
lines(50:200, plogis(predsDiameter$fit - 2 * predsDiameter$se.fit),
    lwd = 1, lty = 2, col = "grey")
```



```
# dev.off()
```

Symbol size is proportional to the number of nests of that combination of size and numbers of observers that discovered it.

Since diameter and height are highly correlated, and size is the better predictor, we shall henceforth only use height to represent size.

3.1.2 Location and forest type

All but three nests were recorded in spruce forest (one in pine, one in beech), and hence we would not expect to be able to detect effects of forest type. Similarly, location has several levels (13), but 95/147 data points are from fully surrounded by forest, rather than moss, thistle etc.

```
anova(glm(as.matrix(detnetsize[, 1:2]) ~ Forest, family = quasibinomial,
    data = detnetsize), test = "F")
```

```
Analysis of Deviance Table

Model: quasibinomial, link: logit

Response: as.matrix(detnetsize[, 1:2])

Terms added sequentially (first to last)

       Df Deviance Resid. Df Resid. Dev      F Pr(>F)
NULL                     138     530.50
Forest  2    7.509       136     522.99 1.1827 0.3096
```

```
anova(glm(as.matrix(detnetsize[, 1:2]) ~ Location, family = quasibinomial,
    data = detnetsize), test = "F")
```

```
Analysis of Deviance Table

Model: quasibinomial, link: logit

Response: as.matrix(detnetsize[, 1:2])

Terms added sequentially (first to last)

         Df Deviance Resid. Df Resid. Dev      F Pr(>F)
NULL                       146     556.86
Location 12   56.021       134     500.84 1.5007 0.1312
```

Neither location nor forest type adds significantly to explaining variation in detection, and both are hence omitted from further analyses.

3.2 Nest size into detection rate analysis

For each nest, the probability of observing it depends on (a) the detection rate of the observer, (b) the size of the nest, and (c) plot characterisitcs. As shown in the last section, we have not recorded any useful measures of plot characateristics, so we leave out point (c) here.

We try two different models: fmm1 with an observer-specific detection curve, and fmm2 with the same detection curve for all observers, but an observer-specific intercept. The latter model will use fewer degrees of freedom. As this turns out to be the more appropriate model for our data, we plot these results.

```
# reformat data for analysis: all observers underneath each other:
part1 <- stack(nests)
colnames(part1) <- c("detected", "observer")
part2 <- do.call("rbind", replicate(8, sizes, simplify = FALSE))
dats <- cbind(part1, part2)
# head(dats)
library(lme4)
# fit a model with variable effect of nest height for each
# observer:
summary(fmm1 <- glmer(detected ~ (poly(Height, 2) | observer), family = binomial,
    data = dats))
```

```
Generalized linear mixed model fit by maximum likelihood
  (Laplace Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: detected ~ (poly(Height, 2) | observer)
   Data: dats

     AIC      BIC   logLik deviance df.resid
  1525.7   1561.2   -755.8   1511.7     1169

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.6253 -0.7663  0.6153  0.7934  1.8713
```

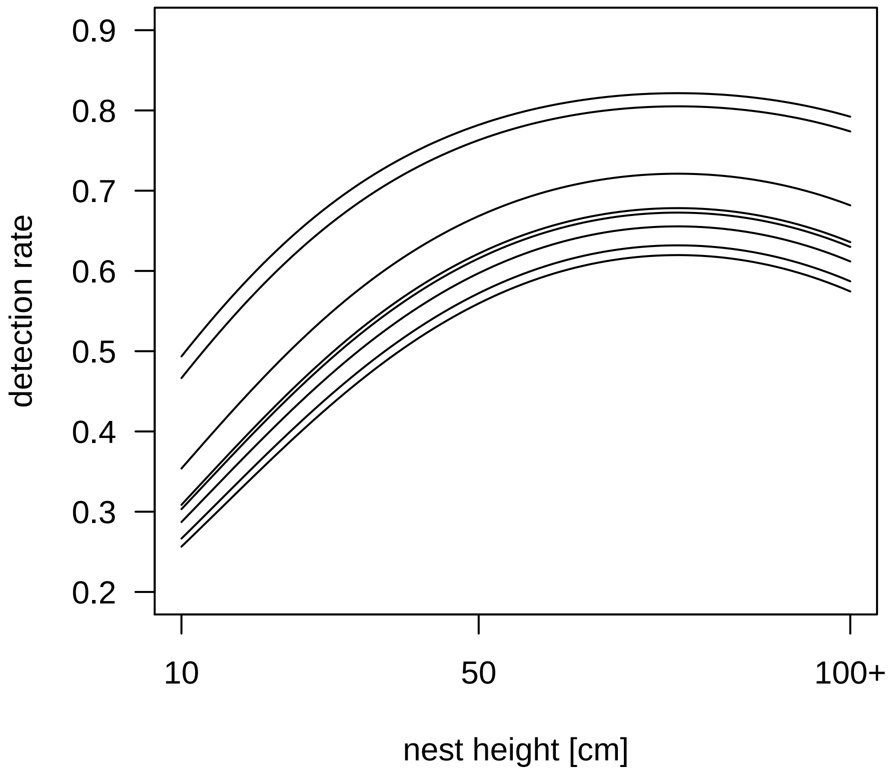
```
Random effects:
 Groups   Name             Variance Std.Dev. Corr
 observer (Intercept)        1.212   1.101
          poly(Height, 2)1 379.454  19.480   -0.99
          poly(Height, 2)2 129.448  11.378    1.00 -0.99
Number of obs: 1176, groups:  observer, 8

Fixed effects:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.028      0.480    2.14   0.0323 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(fmm2 <- glmer(detected ~ poly(Height, 2) + (1 | observer),
    family = binomial, data = dats))
```

```
Generalized linear mixed model fit by maximum likelihood
  (Laplace Approximation) [glmerMod]
 Family: binomial  ( logit )
Formula: detected ~ poly(Height, 2) + (1 | observer)
   Data: dats

     AIC      BIC   logLik deviance df.resid
  1498.2   1518.5   -745.1   1490.2     1172

Scaled residuals:
    Min      1Q  Median      3Q     Max
-1.9525 -0.7399  0.5122  0.8212  1.7025

Random effects:
 Groups   Name        Variance Std.Dev.
 observer (Intercept) 0.1622   0.4028
Number of obs: 1176, groups:  observer, 8

Fixed effects:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)       0.01972    0.15548   0.127    0.899
poly(Height, 2)1  19.76902    2.17022   9.109  < 2e-16 ***
poly(Height, 2)2 -11.09957    2.14672  -5.170 2.34e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Correlation of Fixed Effects:
            (Intr) p(H,2)1
ply(Hgh,2)1  0.000
ply(Hgh,2)2  0.003 -0.004
```

```
plot(10:100, plogis(predict(fmm2, newdata = data.frame(Height = 10:100,
    observer = "O1"))), type = "l", las = 1, xlim = c(10, 100), ylim = c(0.2,
    0.9), ylab = "detection rate", xlab = "nest height [cm]", axes = F)
axis(1, las = 1, at = c(10, 50, 100), labels = c("10", "50", "100+"))
axis(2, las = 1)
```

```
box()
for (j in 2:8) {
    lines(10:100, plogis(predict(fmm2, newdata = data.frame(Height = 10:100,
        observer = paste0("O", j)))), type = "l", las = 1)
}
```



All the analyses above are fine for analysing the correlation between detection of nests and various attributes, but they do not tell us anything about how many nests we have *not* seen. To answer that question, we turn to a very different approach. The above analyses have been useful, however, in guiding us which covariates to include in the following step.

4 Data augmented patch-occupancy model with event-specific covariate

Our data provide the following challenges:

1. We have eight observers sampling the same plots, but each has a different detection rate (due to experience, eye sight, ...).
2. We have shown that small ant nests are easier to overlook than large ones. Thus, each event ("ant nest") has a covariate affecting its detection (nest size, which we simplify to the values "small"=0 and "large"=1, for 10 cm and others, respectively).
3. We may have some nests that none of our eight observers discovered. For those we obviously also do not know the size.

Typical patch-occupancy data assume constant detection rates ("repeated within-season visits") and focus on detection and occurrence of the (typically) animal at each plot. Instead, we want to estimate how many nests were not recorded *at all*. As ant nests (similar to trees, but in contrast to animals) don't move, we can safely assume that occurrence (psi) is 1 if any observer has observed a nest.

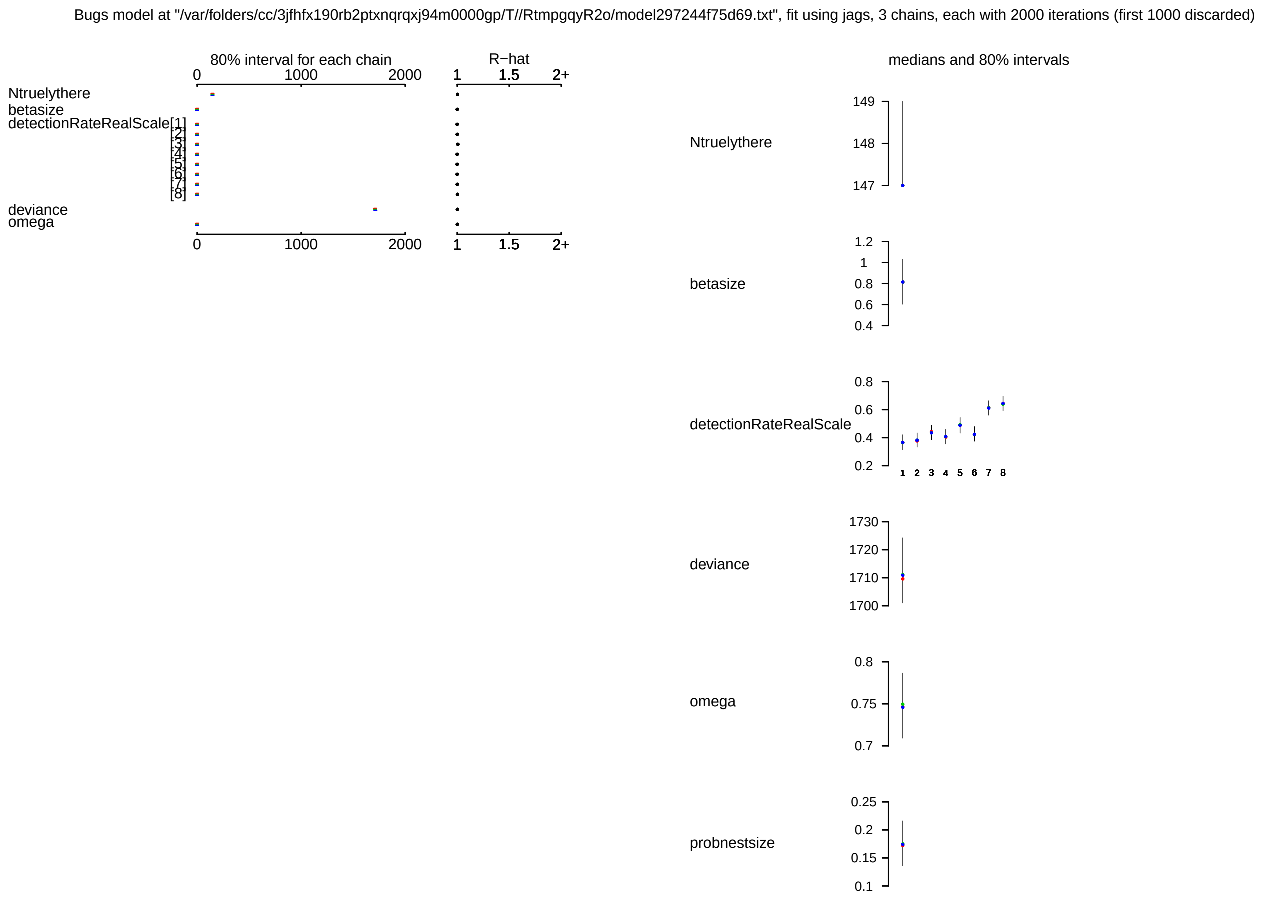
We can handle the "overlooked nests"-issue by adding NA-records to our data set nests, which are then guessed (estimated) during the modelling procedure. This is called "data-augmentation", which feels a bit like Bayesian magic, but isn't. What the model does is to estimate for N^{aug} nests which were not observed, how likely it is that they are there, but were not observed. This can be achieved by realising that also the unobserved nests (and their sizes) are drawn from the same data model that we fit to the observed data. The main tuning parameter on top of a simpler patch-occupancy model is the number of nests we assume to be missing. (In the specific case, we shall assume $N^{aug} = 50$ overlooked nests, but the results do not change if we assume 20 or 200 instead.)

```
library(R2jags)  # load access to JAGS
# augment the matrix with some unobserved nests:
Nunobserved <- 50
augnests <- rbind(as.matrix(nests), matrix(0, Nunobserved, 8))
jags.data <- list(Y = augnests, N = NROW(augnests), J = NCOL(augnests),
    nestsize = c(ifelse(sizes$Height < 70, 0, 1), rep(NA, Nunobserved)))  # categorise nest size into s

augAnalysis <- function() {
    # the classical patch-occupancy model: loop through nests,
    # observed plus augmented
    for (i in 1:N) {
        w[i] ~ dbern(omega)  # realised nest probability

        nestsize[i] ~ dbern(probnestsize)  # either nest size 0 (small) or 1 (large)
        for (j in 1:J) {
            # loop through observers
            Y[i, j] ~ dbern(P[i, j] * w[i])  # compute detection based on the members in the set and th
            logit(P[i, j]) <- detectrate[j] + betasize * nestsize[i]  # nestsize effect on detection
        }
    }

    # Priors and constraints:
    for (j in 1:J) {
        detectrate[j] ~ dnorm(0, 0.01)  # flat but informative prior centred on p=0.5
        # (note: this is at logit-scale, thus mu=0 -> p=0.5);
        # curve(plogis(dnorm(x, 0, 10)), -20, 20)
    }

    omega ~ dunif(0, 1)
    probnestsize ~ dbeta(1, 1)
    betasize ~ dnorm(0, 0.01)

    # derived parameters:
    Ntruelythere <- sum(w)  # number of nests across all plots
    for (j in 1:J) {
        # back-transformed detection rate per observer
        detectionRateRealScale[j] <- exp(detectrate[j])/(1 + exp(detectrate[j]))
    }

}  # end of function

# inits<-function() list (w=c(rep(1, NROW(inventedData)), rep(0,
```

```r
# Nunobserved)), betasize=rnorm(1),
# detectrate=rnorm(n=NCOL(inventedAugnests),1))

parms <- c("omega", "Ntruelythere", "detectionRateRealScale", "betasize",
    "probnestsize")
ni <- 2000
nb <- ni/2
nc <- 3
nt <- 3  # 8000 will do for final estimation!

inits <- function() list(w = c(rep(1, NROW(nests)), rep(0, Nunobserved)),
    betasize = rnorm(1), detectrate = rnorm(n = NCOL(augnests), 1))

# call JAGS
system.time(augJags <- jags(jags.data, inits, parms, model.file = augAnalysis,
    n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, working.directory = getwd()))
```

```
module glm loaded


Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 1723
   Unobserved stochastic nodes: 258
   Total graph size: 4507

Initializing model


   user  system elapsed
 10.979   0.066  11.161
```

```r
plot(augJags)
```



```
augJags
```

```
Inference for Bugs model at "/var/folders/cc/3jfhfx190rb2ptxnqrqxj94m0000gp/T//RtmpgqyR2o/model297244f75
 3 chains, each with 2000 iterations (first 1000 discarded), n.thin = 3
 n.sims = 1002 iterations saved
                          mu.vect sd.vect    2.5%      25%
Ntruelythere              147.627   0.811  147.000  147.000
betasize                    0.817   0.165    0.493    0.708
detectionRateRealScale[1]   0.366   0.042    0.285    0.339
detectionRateRealScale[2]   0.381   0.039    0.314    0.353
detectionRateRealScale[3]   0.437   0.041    0.352    0.411
detectionRateRealScale[4]   0.407   0.040    0.332    0.380
detectionRateRealScale[5]   0.488   0.044    0.401    0.457
detectionRateRealScale[6]   0.426   0.041    0.347    0.397
detectionRateRealScale[7]   0.613   0.041    0.530    0.585
detectionRateRealScale[8]   0.644   0.040    0.565    0.617
omega                       0.748   0.031    0.679    0.727
probnestsize                0.175   0.031    0.119    0.155
deviance                 1711.737   9.690 1699.031 1704.103
                              50%      75%    97.5%  Rhat n.eff
Ntruelythere              147.000  148.000  150.000 1.004   630
betasize                    0.814    0.928    1.131 1.001  1000
detectionRateRealScale[1]   0.365    0.393    0.446 1.000  1000
detectionRateRealScale[2]   0.381    0.408    0.462 1.001  1000
```

```
detectionRateRealScale[3]     0.437    0.463    0.519 1.007   400
detectionRateRealScale[4]     0.406    0.434    0.489 1.000  1000
detectionRateRealScale[5]     0.489    0.517    0.574 1.000  1000
detectionRateRealScale[6]     0.424    0.452    0.513 1.000  1000
detectionRateRealScale[7]     0.613    0.642    0.693 1.001  1000
detectionRateRealScale[8]     0.642    0.672    0.723 1.003   790
omega                         0.747    0.769    0.807 1.002   800
probnestsize                  0.174    0.194    0.238 1.000  1000
deviance                   1710.662 1717.094 1736.772 1.002   710

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 46.9 and DIC = 1758.6
DIC is an estimate of expected predictive error (lower deviance is better).
```

The results show that large nests have a higher chance of being detected (notice the estimate for betasize of 0.819 at the link scale, representing the effect of going from small to large nests). Furthermore, we get per-observer observation estimates (at the real scale) between 0.365 and 0.642, i.e. almost a factor of 2. And, finally, we get an estimate of the total number of nests across all plots as 147.7 (95%-confidence interval up to 150), i.e. 1 to 3 nests overlooked.

Since we do not have any covariates at the plot level, we can distribute the overlooked nests across plots proportional to the number of nests observed there.

```r
quants <- quantile(augJags$BUGSoutput$sims.list$Ntruelythere - 147,
    c(0.025, 0.5, 0.975))
# So the number of nests per plot are:
estimated <- matrix(plots$Nmin, ncol = 3, nrow = 16, byrow = F) +
    matrix(quants, ncol = 3, nrow = 16, byrow = T)/matrix(plots$Nmin,
        ncol = 3, nrow = 16, byrow = F)
# in line with reason, but against maths, we assume for plot 4
# that 0/0=0, and get:
estimated[4, ] <- 0
colnames(estimated) <- c("lower CI", "median", "upper CI")
round(estimated, 2)
```

```
      lower CI median upper CI
 [1,]       24     24    24.12
 [2,]       17     17    17.18
 [3,]        8      8     8.38
 [4,]        0      0     0.00
 [5,]       18     18    18.17
 [6,]        3      3     4.00
 [7,]        1      1     4.00
 [8,]        4      4     4.75
 [9,]       10     10    10.30
[10,]       20     20    20.15
[11,]        1      1     4.00
[12,]       11     11    11.27
[13,]        4      4     4.75
[14,]        3      3     4.00
[15,]       12     12    12.25
```

```
[16,]        11      11     11.27
```

Essentially this indicates that with our eight observers, we have good faith of not having overlooked any nest!

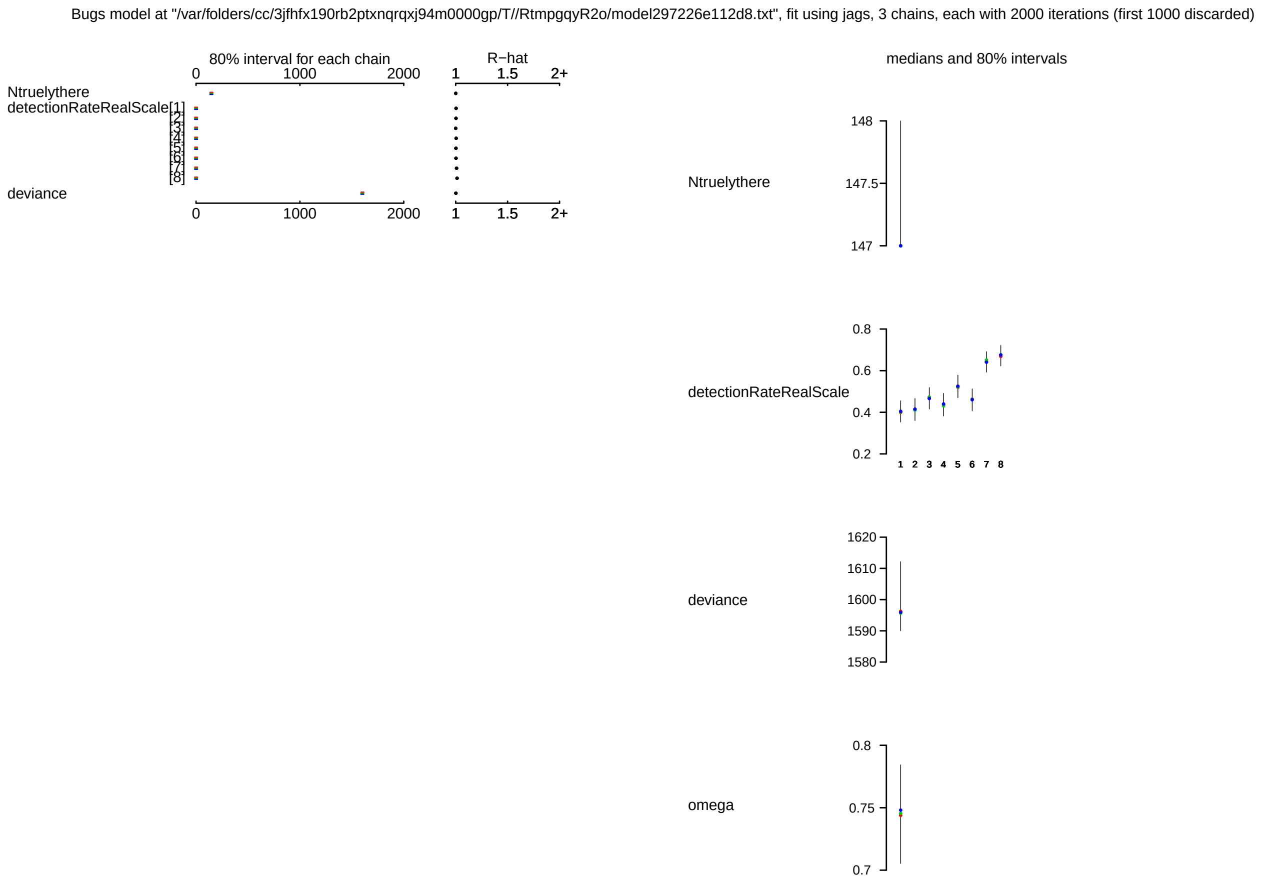
One advantage of the above data-augmentation approach is that it allows us to model each nest separately and thereby include a covariate for the nest. If we omit the effect of nest size, the results are as follows:

```r
augAnalysis2 <- function() {
    # the classical patch-occupancy model: loop through nests,
    # observed plus augmented
    for (i in 1:N) {
        w[i] ~ dbern(omega)  # realised nest probability

        for (j in 1:J) {
            # loop through observers
            Y[i, j] ~ dbern(P[i, j] * w[i])  # compute detection based on the members in the set and th
            logit(P[i, j]) <- detectrate[j]  #+ betasize*nestsize[i] # no nestsize effect on detection
        }
    }

    # Priors and constraints:
    for (j in 1:J) {
        detectrate[j] ~ dnorm(0, 0.01)  # flat but informative prior centred on p=0.5
        # (note: this is at logit-scale, thus mu=0 -> p=0.5);
        # curve(plogis(dnorm(x, 0, 10)), -20, 20)
    }

    omega ~ dunif(0, 1)

    # derived parameters:
    Ntruelythere <- sum(w)  # number of nests across all plots
    for (j in 1:J) {
        # back-transformed detection rate per observer
        detectionRateRealScale[j] <- exp(detectrate[j])/(1 + exp(detectrate[j]))
    }

}  # end of function

# inits<-function() list (w=c(rep(1, NROW(inventedData)), rep(0,
# Nunobserved)), betasize=rnorm(1),
# detectrate=rnorm(n=NCOL(inventedAugnests),1))

parms <- c("omega", "Ntruelythere", "detectionRateRealScale")
inits <- function() list(w = c(rep(1, NROW(nests)), rep(0, Nunobserved)),
    detectrate = rnorm(n = NCOL(augnests), 1))

# call JAGS
system.time(augJags2 <- jags(jags.data, inits, parms, model.file = augAnalysis2,
    n.chains = nc, n.thin = nt, n.iter = ni, n.burnin = nb, working.directory = getwd()))
```

```
Warning in jags.model(model.file, data = data, inits =
init.values, n.chains = n.chains, : Unused variable "nestsize" in
data
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 1576
   Unobserved stochastic nodes: 206
   Total graph size: 3428

Initializing model


   user  system elapsed
  4.685   0.068   4.965
```

```
plot(augJags2)
```



```
augJags2
```

```
Inference for Bugs model at "/var/folders/cc/3jfhfx190rb2ptxnqrqxj94m0000gp/T//RtmpgqyR2o/model297226e1
 3 chains, each with 2000 iterations (first 1000 discarded), n.thin = 3
 n.sims = 1002 iterations saved
                           mu.vect sd.vect    2.5%      25%
Ntruelythere               147.460   0.726  147.000  147.000
detectionRateRealScale[1]    0.403   0.040    0.328    0.376
```

```
detectionRateRealScale[2]    0.413   0.041    0.334    0.386
detectionRateRealScale[3]    0.467   0.041    0.389    0.438
detectionRateRealScale[4]    0.436   0.042    0.354    0.408
detectionRateRealScale[5]    0.523   0.042    0.441    0.494
detectionRateRealScale[6]    0.461   0.041    0.387    0.435
detectionRateRealScale[7]    0.643   0.038    0.563    0.618
detectionRateRealScale[8]    0.672   0.038    0.596    0.647
omega                        0.745   0.031    0.686    0.724
deviance                  1599.104   9.295 1588.527 1592.232
                               50%      75%    97.5%  Rhat n.eff
Ntruelythere               147.000  148.000  149.000 1.001  1000
detectionRateRealScale[1]    0.403    0.431    0.482 1.004   480
detectionRateRealScale[2]    0.413    0.441    0.491 1.003   540
detectionRateRealScale[3]    0.469    0.496    0.548 1.000  1000
detectionRateRealScale[4]    0.436    0.466    0.513 1.004   480
detectionRateRealScale[5]    0.523    0.551    0.603 1.003   630
detectionRateRealScale[6]    0.461    0.488    0.542 1.003  1000
detectionRateRealScale[7]    0.644    0.669    0.715 1.009   220
detectionRateRealScale[8]    0.674    0.698    0.743 1.013   170
omega                        0.746    0.766    0.802 1.003   530
deviance                  1596.090 1604.094 1622.298 1.002  1000

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 43.2 and DIC = 1642.3
DIC is an estimate of expected predictive error (lower deviance is better).
```

That means that detection probabilities for the observers are slightly higher across the board, but the estimated number of nests hardly changes. In other words, while nest size affects detection probability, it does not greatly bias estimates of the total number of nests.

One disadvantage of the above analysis is that we essentially treat the data as if they were from one large plot, rather than from 16 different plots. If we wanted to estimate abundance of red wood ant nests at each plot, rather than across all plots, we have to employ an abundance model. The approach of Royle (2004: *N*-mixture models for estimating population size from spatially replicated counts. Biometrics 60:108) is in principle suitable for such data, except that in his case the observation matrix is very sparse (has many 0s), while ours is dense (0s only in few observer-plot combinations):

```
plots
```

```
        O1 O2 O3 O4 O5 O6 O7 O8 Nmin
Plot 1  12 14 12  9 10 11 12 13   24
Plot 2   1  5  5  7  7  7  9 12   17
Plot 3   4  5  2  5  4  5  7  6    8
Plot 4   0  0  0  0  0  0  0  0    0
Plot 5   9  7  7  6 13  8 14 14   18
Plot 6   0  0  0  0  0  0  3  2    3
Plot 7   0  0  1  0  0  0  1  0    1
Plot 8   2  2  2  3  4  2  3  3    4
Plot 9   3  3  3  4  3  5  5  7   10
Plot 10  6  5 10  8  9  8  9 10   20
Plot 11  0  0  0  1  0  0  0  0    1
Plot 12  7  6  9  7  9  6 10  8   11
Plot 13  1  1  0  1  1  3  3  3    4
Plot 14  2  2  1  2  2  2  3  3    3
Plot 15  5  5  8  5  8  4  7 10   12
Plot 16  7  6  9  7  7  7  9  8   11
```

In this case it seems more intuitive to model abundance of nests per plot as a binomial random variate, rather than the more involved mixture of N Poisson distributions. This is what our next model does.

```r
library(unmarked)
obs <- matrix(as.character(1:8), 16, 8, byrow = T)
sitecovs <- data.frame(X = as.factor(1:16))
antsumf <- unmarkedFramePCount(y = as.matrix(plots[, -9]), obsCovs = list(observer = obs),
    siteCovs = sitecovs)
(fit <- pcount(~observer ~ X, data = antsumf, K = 50, mixture = "P"))
```

```
Call:
pcount(formula = ~observer ~ X, data = antsumf, K = 50, mixture = "P")

Abundance:
            Estimate      SE       z  P(>|z|)
(Intercept)    3.760   0.269 13.9927 1.73e-44
X2            -0.530   0.289 -1.8360 6.64e-02
X3            -0.889   0.327 -2.7162 6.60e-03
X4           -15.401 351.669 -0.0438 9.65e-01
X5            -0.156   0.259 -0.6035 5.46e-01
X6            -2.497   0.607 -4.1172 3.83e-05
X7            -3.589   1.015 -3.5378 4.03e-04
X8            -1.500   0.415 -3.6128 3.03e-04
X9            -1.027   0.344 -2.9879 2.81e-03
X10           -0.352   0.275 -1.2818 2.00e-01
X11           -3.674   1.025 -3.5846 3.38e-04
X12           -0.402   0.279 -1.4398 1.50e-01
X13           -1.931   0.494 -3.9078 9.31e-05
X14           -1.719   0.456 -3.7726 1.62e-04
X15           -0.569   0.293 -1.9400 5.24e-02
X16           -0.438   0.282 -1.5501 1.21e-01

Detection:
            Estimate    SE      z  P(>|z|)
(Intercept)  -1.3132 0.326 -4.034 5.49e-05
observer2     0.0425 0.206  0.206 8.37e-01
observer3     0.2033 0.202  1.004 3.15e-01
observer4     0.1246 0.204  0.611 5.41e-01
observer5     0.3519 0.200  1.757 7.90e-02
observer6     0.1839 0.203  0.907 3.64e-01
observer7     0.6556 0.202  3.248 1.16e-03
observer8     0.7189 0.203  3.534 4.10e-04

AIC: 449.1964
```

```
ests <- fit@estimates@estimates$det@estimates
# observer probabilities:
plogis(c(observer1 = unname(ests[1]), ests[2:8] + ests[1]))
```

```
observer1 observer2 observer3 observer4 observer5 observer6
0.2119490 0.2191383 0.2478838 0.2335077 0.2766223 0.2442902
observer7 observer8
0.3412760 0.3556458
```

```
# site estimates:
estplot <- fit@estimates@estimates$state@estimates
round(exp(c(X1 = unname(estplot[1]), estplot[2:16] + estplot[1])),
    2)
```

```
   X1    X2    X3    X4    X5    X6    X7    X8    X9   X10
42.97 25.29 17.66  0.00 36.74  3.54  1.19  9.58 15.38 30.20
  X11   X12   X13   X14   X15   X16
 1.09 28.75  6.23  7.70 24.32 27.73
```

5 Bayesian plot-level detection model

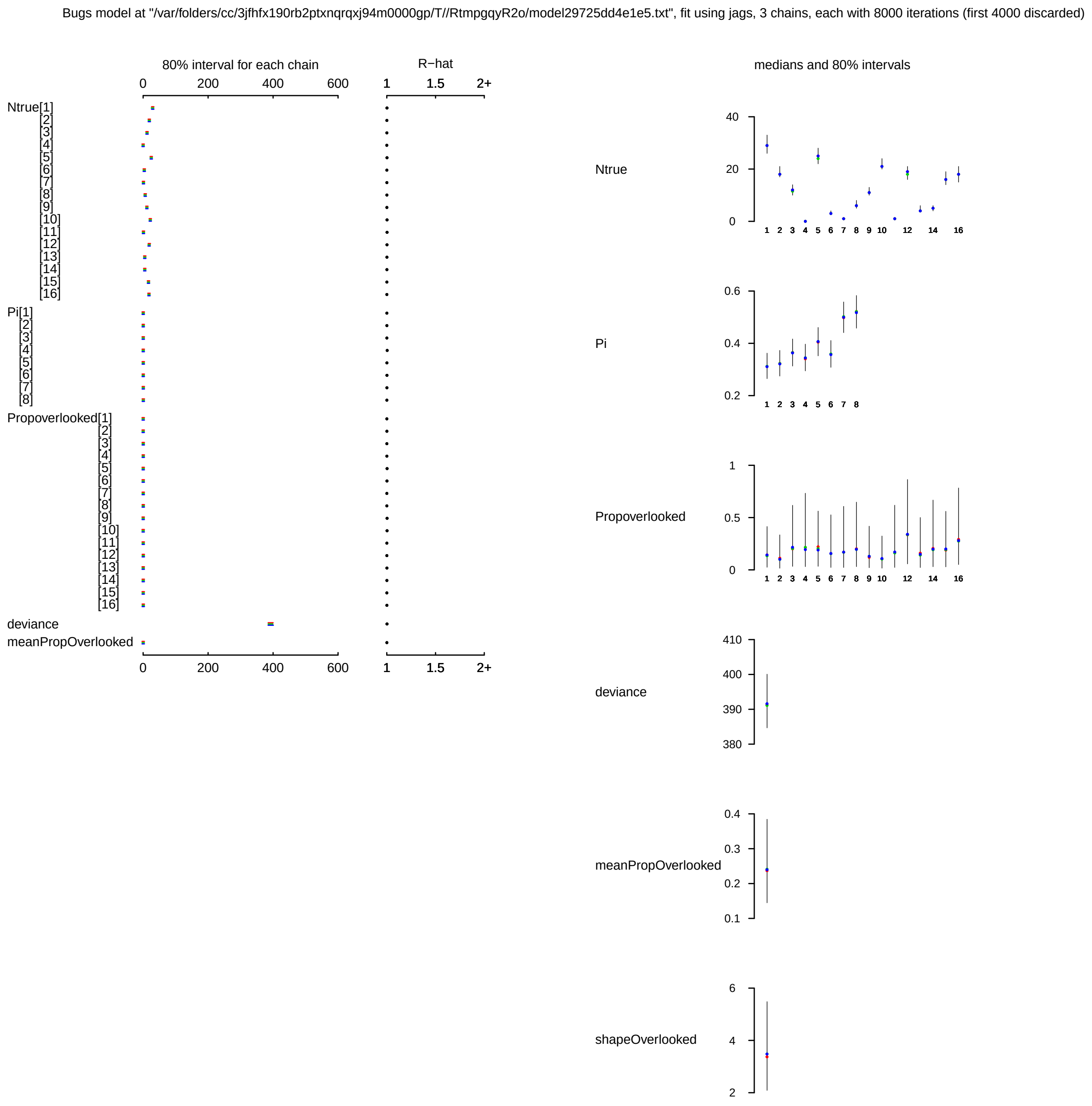
Our model of the plots data as displayed above assumes that the number of nests observed in a plot s by observer i is a draw from a binomial distribution with a estimated population size $\hat{N}_s^{\text{true}}$ for each plot s, and an estimated observation probability $\hat{P}_i$ for observer i. Since we have eight observers and 16 plots, we can estimate both $\hat{P}_i$ and $\hat{N}_s$. For this Bayesian model we need to choose priors for detection probabilities and $\hat{N}_s$. The latter has a lower bound at $N_s^{\text{min}} = N_s^{\text{obs}}$, as there cannot be fewer nests than observed.

```
detectBinom <- function() {
    # the detection model loop through observers loop through plots
    for (i in 1:8) {
        for (j in 1:16) {
            Nobs[j, i] ~ dbin(Pi[i], Ntrue[j])
        }
    }

    # Priors and constraints:
    for (j in 1:16) {
        # Ntrue must be an integer greater or equal Nobs:
        Ntrue[j] ~ dpois(Nmin[j] * (1 + Propoverlooked[j]))
        T(Nmin[j], )
        # overlooked nests modelled as proportion of the number observed:
        Propoverlooked[j] ~ dexp(shapeOverlooked)
    }
    shapeOverlooked ~ dgamma(1, 1)
    # uninformative prior on detection:
    for (i in 1:8) {
        Pi[i] ~ dbeta(1, 1)  # flat line
    }
    # compute another value of interest:
    meanPropOverlooked <- mean(Propoverlooked)
}  # end of function
```

```r
jags.data <- list(Nobs = plots[, -9], Nmin = plots[, 9])
parametersBinom <- c("Ntrue", "Pi", "Propoverlooked", "shapeOverlooked",
    "meanPropOverlooked")
ni <- 8000
nb <- ni/2
nc <- 3
nt <- 3
# call JAGS
system.time(antdetectBinom <- jags(jags.data, inits = NULL, parametersBinom,
    model.file = detectBinom, n.chains = nc, n.thin = nt, n.iter = ni,
    n.burnin = nb, working.directory = getwd()))
```

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 128
   Unobserved stochastic nodes: 41
   Total graph size: 253

Initializing model


   user  system elapsed
  5.762   0.039   5.828
```

```r
plot(antdetectBinom)
```



```
(antdetectBinom)
```

```
Inference for Bugs model at "/var/folders/cc/3jfhfx190rb2ptxnqrqxj94m0000gp/T//RtmpgqyR2o/model29725dd4
 3 chains, each with 8000 iterations (first 4000 discarded), n.thin = 3
 n.sims = 4002 iterations saved
                   mu.vect sd.vect    2.5%     25%     50%
Ntrue[1]            29.077   2.893  24.000  27.000  29.000
Ntrue[2]            18.482   1.492  17.000  17.000  18.000
Ntrue[3]            11.670   1.586   9.000  11.000  12.000
Ntrue[4]             0.000   0.000   0.000   0.000   0.000
Ntrue[5]            24.779   2.557  20.000  23.000  25.000
Ntrue[6]             3.133   0.360   3.000   3.000   3.000
```

```
Ntrue[7]             1.044   0.206   1.000   1.000   1.000
Ntrue[8]             6.158   1.061   4.000   5.000   6.000
Ntrue[9]            11.124   1.164  10.000  10.000  11.000
Ntrue[10]           21.704   1.691  20.000  20.000  21.000
Ntrue[11]            1.023   0.156   1.000   1.000   1.000
Ntrue[12]           18.695   2.161  15.000  17.000  19.000
Ntrue[13]            4.572   0.741   4.000   4.000   4.000
Ntrue[14]            4.883   0.939   3.000   4.000   5.000
Ntrue[15]           16.254   1.951  13.000  15.000  16.000
Ntrue[16]           18.006   2.143  14.000  17.000  18.000
Pi[1]                0.313   0.038   0.241   0.287   0.311
Pi[2]                0.323   0.038   0.252   0.297   0.322
Pi[3]                0.364   0.041   0.287   0.337   0.364
Pi[4]                0.345   0.040   0.271   0.317   0.343
Pi[5]                0.406   0.042   0.322   0.377   0.406
Pi[6]                0.359   0.040   0.280   0.331   0.358
Pi[7]                0.499   0.047   0.407   0.467   0.500
Pi[8]                0.520   0.048   0.425   0.488   0.520
Propoverlooked[1]    0.186   0.164   0.006   0.060   0.141
Propoverlooked[2]    0.145   0.139   0.004   0.044   0.104
Propoverlooked[3]    0.278   0.258   0.008   0.085   0.205
Propoverlooked[4]    0.315   0.347   0.008   0.084   0.208
Propoverlooked[5]    0.259   0.213   0.010   0.092   0.203
Propoverlooked[6]    0.231   0.250   0.005   0.064   0.156
Propoverlooked[7]    0.263   0.294   0.005   0.068   0.168
Propoverlooked[8]    0.282   0.276   0.009   0.085   0.198
Propoverlooked[9]    0.181   0.174   0.005   0.053   0.127
Propoverlooked[10]   0.144   0.138   0.004   0.043   0.105
Propoverlooked[11]   0.257   0.274   0.006   0.067   0.167
Propoverlooked[12]   0.410   0.329   0.016   0.157   0.338
Propoverlooked[13]   0.215   0.214   0.006   0.062   0.149
Propoverlooked[14]   0.285   0.277   0.007   0.082   0.198
Propoverlooked[15]   0.255   0.228   0.007   0.081   0.192
Propoverlooked[16]   0.360   0.302   0.014   0.124   0.284
meanPropOverlooked   0.254   0.097   0.111   0.184   0.239
shapeOverlooked      3.654   1.385   1.590   2.648   3.448
deviance           391.993   6.101 381.722 387.700 391.340
                       75%   97.5%  Rhat n.eff
Ntrue[1]            31.000  36.000 1.002  1400
Ntrue[2]            19.000  22.000 1.001  4000
Ntrue[3]            13.000  15.000 1.001  4000
Ntrue[4]             0.000   0.000 1.000     1
Ntrue[5]            26.000  30.000 1.002  1200
Ntrue[6]             3.000   4.000 1.001  4000
Ntrue[7]             1.000   2.000 1.001  4000
Ntrue[8]             7.000   8.000 1.001  4000
Ntrue[9]            12.000  14.000 1.001  4000
Ntrue[10]           23.000  26.000 1.003   980
Ntrue[11]            1.000   1.000 1.003  4000
Ntrue[12]           20.000  23.000 1.002  1300
Ntrue[13]            5.000   6.000 1.002  2100
Ntrue[14]            5.000   7.000 1.001  2500
Ntrue[15]           17.000  20.975 1.001  4000
Ntrue[16]           19.000  23.000 1.001  4000
```

```
Pi[1]                 0.338   0.387 1.001  4000
Pi[2]                 0.349   0.401 1.001  4000
Pi[3]                 0.391   0.445 1.001  4000
Pi[4]                 0.371   0.422 1.004   640
Pi[5]                 0.434   0.490 1.001  3100
Pi[6]                 0.386   0.439 1.002  1700
Pi[7]                 0.530   0.592 1.001  2600
Pi[8]                 0.554   0.614 1.001  4000
Propoverlooked[1]     0.267   0.619 1.001  4000
Propoverlooked[2]     0.201   0.506 1.001  2800
Propoverlooked[3]     0.391   0.956 1.001  4000
Propoverlooked[4]     0.418   1.263 1.001  4000
Propoverlooked[5]     0.373   0.792 1.003   840
Propoverlooked[6]     0.306   0.904 1.002  2200
Propoverlooked[7]     0.359   1.056 1.001  4000
Propoverlooked[8]     0.394   1.021 1.001  4000
Propoverlooked[9]     0.256   0.654 1.003   880
Propoverlooked[10]    0.202   0.508 1.003  2200
Propoverlooked[11]    0.352   0.973 1.001  4000
Propoverlooked[12]    0.585   1.198 1.001  4000
Propoverlooked[13]    0.301   0.803 1.001  2500
Propoverlooked[14]    0.403   1.041 1.001  3300
Propoverlooked[15]    0.368   0.841 1.001  4000
Propoverlooked[16]    0.516   1.103 1.001  4000
meanPropOverlooked    0.307   0.488 1.001  4000
shapeOverlooked       4.444   7.003 1.002  2100
deviance            395.603 405.636 1.002  1300

For each parameter, n.eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor (at convergence, Rhat=1).

DIC info (using the rule, pD = var(deviance)/2)
pD = 18.6 and DIC = 410.6
DIC is an estimate of expected predictive error (lower deviance is better).
```

```
# estimated number of nests:
sum(antdetectBinom$BUGSoutput$mean$Ntrue)
```

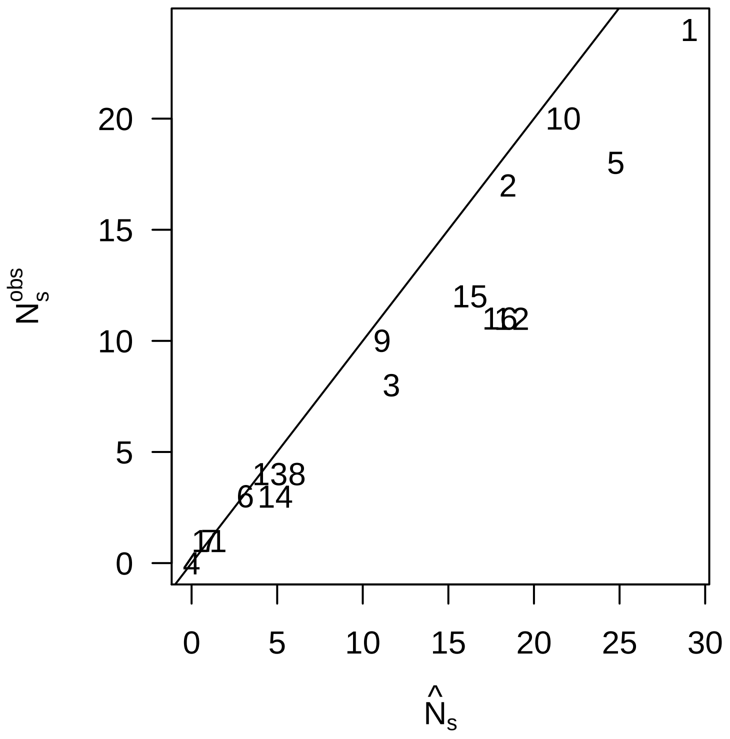
```
[1] 190.6059
```

The key findings are that we overlook, according to this model, around 26% of nests (estimating the total as roughly 191 ± 18); and that the detection rates vary between 0.312 and 0.520 among observers. These values look rather different to the previous data-augmentation model. This difference can be attributed to two changes in the model structure:

1. The current model has fewer data points because it aggregates all nests within a plot ($16 \cdot 8 = 128$ vs. 147 for the data augmentation).
2. The current model joinedly estimates detection rates and true number of nests, $P(N_{i,s}^{obs} \,|\, \hat{P}_i, \hat{N}_s)$, rather than conditionally $P(N_{i,s}^{obs} \,|\, \hat{N}_{i,s})$ for each nest as in the data-augmentation model. The reason is that we have no way to estimate, for the aggregated data, the probability of a nest being present and thus estimate at plot- rather than nest-level.

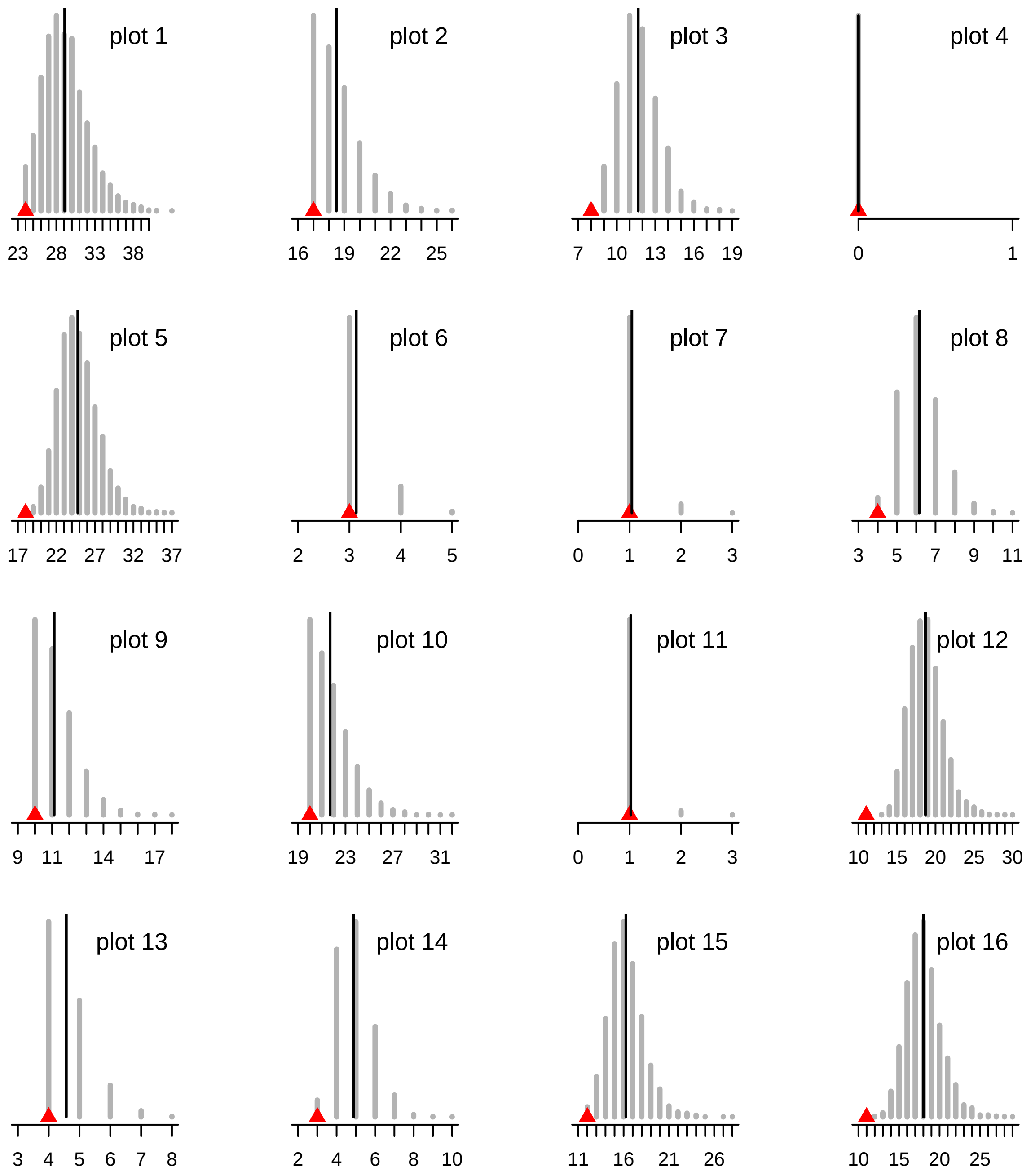
We can plot the estimated (x-axis) and observed number of nests per plot (y-axis):

```
par(mar = c(4, 5, 1, 1))
plot(antdetectBinom$BUGSoutput$mean$Ntrue, plots[, 9], type = "n",
    ylab = expression(N[s]^{
        obs
    }), xlab = expression(hat(N)[s]), las = 1)
abline(0, 1)
text(antdetectBinom$BUGSoutput$mean$Ntrue, plots[, 9], cex = 1)
```



Clearly the correction applied is larger in plots with more ant nests, or, phrased differently, the adjustment is proportional. We can plot the posteriors and their mean:

```
par(mar = c(3, 4, 1, 1), mfrow = c(4, 4))
for (i in 1:16) {
    dens <- antdetectBinom$BUGSoutput$sims.list$Ntrue[, i]
    tdens <- table(dens)
    plot(as.numeric(names(tdens)), tdens/length(dens), las = 1, col = "grey70",
        type = "h", lwd = 3, ylab = "", xlab = "", xlim = c(min(max(0,
            plots[i, 9] - 1), as.numeric(names(tdens))), max(c(1,
            as.numeric(names(tdens))))), ylim = c(0, max(tdens/length(dens))),
        axes = F)
    axis(1, at = 0:40)
    points(plots[i, 9], 0, pch = 17, cex = 1.5, col = "red", xpd = T)
    legend("topright", bty = "n", legend = paste("plot", i), cex = 1.2)
    lines(rep(antdetectBinom$BUGSoutput$mean$Ntrue[i], 2), c(0, 1),
        col = "black", lwd = 1.5)
}
```



Histograms are Bayesian posteriors for $\hat{N}_s$, with mean estimate indicated by the vertical black line, and the red triangle indicates N_s^{obs} for each plot.

6 Plot-level N-mixture estimation (not mentioned in the main document)

Royle (2004) proposed to view observed abundances as mixtures of Poisson distributions, thus extending the patch-occupancy idea to abundances. As is the case for the previous Bayesian plot-level analysis, this approach cannot accommodate nest traits and is thus at the plot level.

```
library(unmarked)
obs <- matrix(as.character(1:8), 16, 8, byrow = T)
sitecovs <- data.frame(X = as.factor(1:16))
antsumf <- unmarkedFramePCount(y = as.matrix(plots[, -9]), obsCovs = list(observer = obs),
    siteCovs = sitecovs)
(fit <- pcount(~observer ~ X, data = antsumf, K = 50, mixture = "P"))
```

```
Call:
pcount(formula = ~observer ~ X, data = antsumf, K = 50, mixture = "P")

Abundance:
            Estimate      SE       z  P(>|z|)
(Intercept)    3.760   0.269 13.9927 1.73e-44
X2            -0.530   0.289 -1.8360 6.64e-02
X3            -0.889   0.327 -2.7162 6.60e-03
X4           -15.401 351.669 -0.0438 9.65e-01
X5            -0.156   0.259 -0.6035 5.46e-01
X6            -2.497   0.607 -4.1172 3.83e-05
X7            -3.589   1.015 -3.5378 4.03e-04
X8            -1.500   0.415 -3.6128 3.03e-04
X9            -1.027   0.344 -2.9879 2.81e-03
X10           -0.352   0.275 -1.2818 2.00e-01
X11           -3.674   1.025 -3.5846 3.38e-04
X12           -0.402   0.279 -1.4398 1.50e-01
X13           -1.931   0.494 -3.9078 9.31e-05
X14           -1.719   0.456 -3.7726 1.62e-04
X15           -0.569   0.293 -1.9400 5.24e-02
X16           -0.438   0.282 -1.5501 1.21e-01

Detection:
            Estimate    SE      z  P(>|z|)
(Intercept)  -1.3132 0.326 -4.034 5.49e-05
observer2     0.0425 0.206  0.206 8.37e-01
observer3     0.2033 0.202  1.004 3.15e-01
observer4     0.1246 0.204  0.611 5.41e-01
observer5     0.3519 0.200  1.757 7.90e-02
observer6     0.1839 0.203  0.907 3.64e-01
observer7     0.6556 0.202  3.248 1.16e-03
observer8     0.7189 0.203  3.534 4.10e-04

AIC: 449.1964
```

```
ests <- fit@estimates@estimates$det@estimates
# observer probabilities:
plogis(c(observer1 = unname(ests[1]), ests[2:8] + ests[1]))
```

```
observer1 observer2 observer3 observer4 observer5 observer6
0.2119490 0.2191383 0.2478838 0.2335077 0.2766223 0.2442902
observer7 observer8
0.3412760 0.3556458
```

```
# site estimates:
estplot <- fit@estimates@estimates$state@estimates
round(exp(c(X1 = unname(estplot[1]), estplot[2:16] + estplot[1])),
    2)
```

```
   X1    X2    X3    X4    X5    X6    X7    X8    X9   X10
42.97 25.29 17.66  0.00 36.74  3.54  1.19  9.58 15.38 30.20
  X11   X12   X13   X14   X15   X16
 1.09 28.75  6.23  7.70 24.32 27.73
```

Observer rates are estimated lower, and number of nests accordingly higher, than in the Bayesian model. Otherwise the model confirms the results of the previous models.

7 Plot-level maximum likelihood estimation

We can dispense with the Bayesian nature of the previous step as we are using uninformative priors only, and thus in this case use maximum likelihood to estimate the model parameters (observation rates and true number of nests per plot). Again we assume that the number of nests detected by observer i at plot s is a draw from a binomial distribution, i.e. $N_{i,s}^{obs} \sim Binom(size = \hat{N}_s, p = \hat{P}_i)$. Thus, across the 16 plots and 8 observers we have to optimise 24 values to find the maximum likelihood fit for the 128 data points in plots.

```
Nobs <- plots[, -9]  # remove Nmin
# define the maximum likelihood based on the 24 parameters:
toopt <- function(parms, Nobs = plots[, -9]) {
    Ntrue <- exp(parms[1:16])  # ensure positive values
    Pi <- plogis(parms[17:24])  # ensure values in [0,1]
    Nsest <- tcrossprod(Ntrue, Pi)
    -sum(dpois(as.matrix(Nobs), lambda = as.matrix(Nsest), log = T))
}
# use Nmin as start values for Ntrue and 0.5 for detection rates:
Nmin <- plots[, 9]
(op <- optim(par = c(log(Nmin + 1), rep(0.5, 8)), fn = toopt, method = "BFGS",
    hessian = F))
```

```
$par
 [1]   3.2723471   2.7100414   2.3773348 -13.9086521   3.0964573
 [6]   0.3491841  -0.5671277   1.7842722   2.2362588   2.9141346
[11]  -1.2602092   2.8668821   1.3047004   1.5729595   2.6909924
[16]   2.8340923  -0.6153202  -0.5634996  -0.3626869  -0.4619341
[21]  -0.1688646  -0.3873130   0.2611425   0.3585991

$value
[1] 198.443

$counts
function gradient
      91       43

$convergence
[1] 0
```

```
$message
NULL
```

```
# image(op$hessian) # after setting hessian=T reveals that there
# is no correlation between estimates; therefore we can compute
# errors for Nhat as sum of independent plots estimated detProb
# per observer:
round(plogis(op$par[17:24]), 2)
```

```
[1] 0.35 0.36 0.41 0.39 0.46 0.40 0.56 0.59
```

```
# estimated number of nests
round(bestguessNtrue <- exp(op$par[1:16]))
```

```
 [1] 26 15 11  0 22  1  1  6  9 18  0 18  4  5 15 17
```

```
# compare to Nmin:
plots[, 9]
```

```
 [1] 24 17  8  0 18  3  1  4 10 20  1 11  4  3 12 11
```

```
# estimated number of nests:
sum(bestguessNtrue)
```

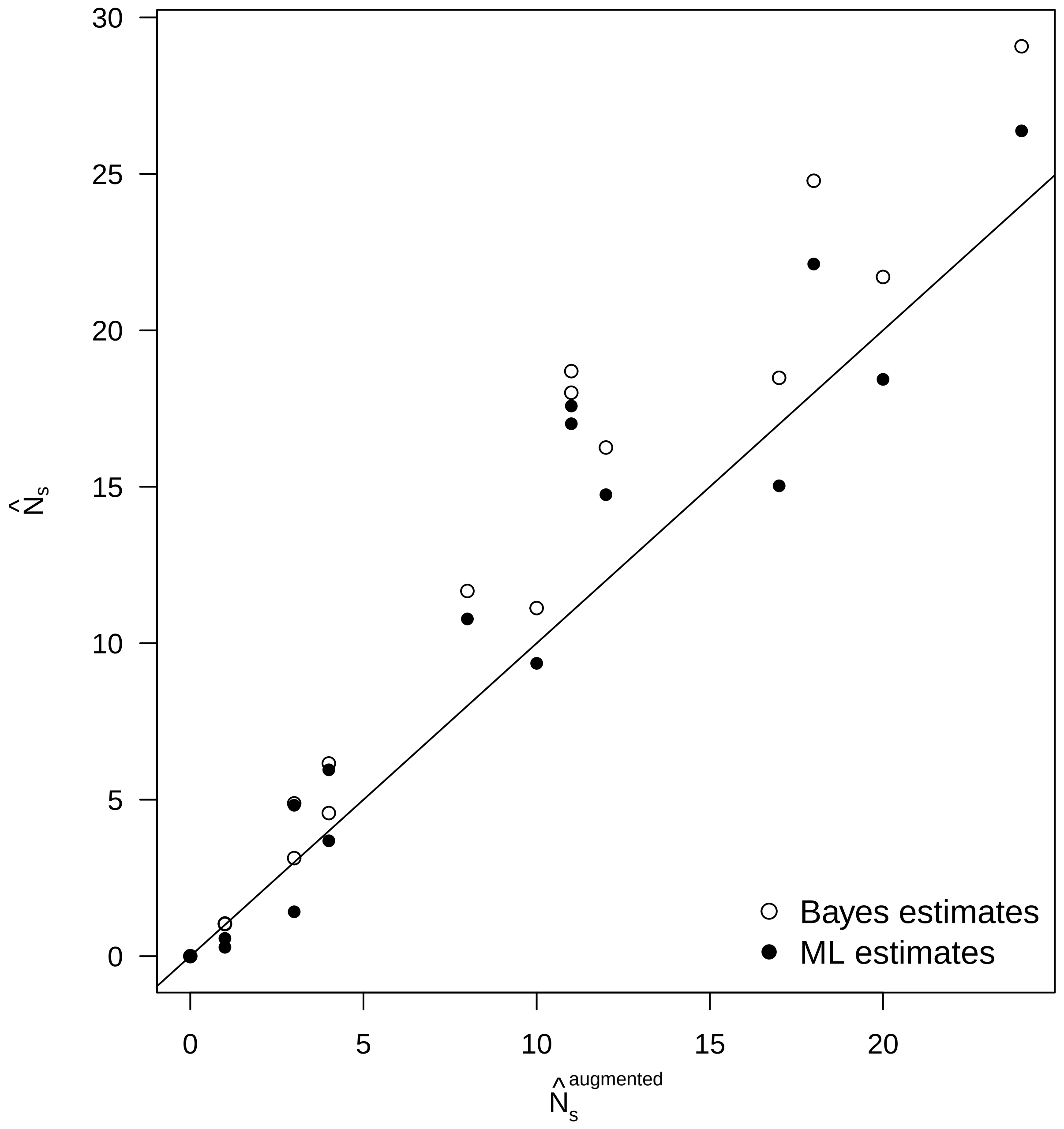
```
[1] 168.1645
```

Maximum-likelihood estimates are somewhat higher than N_s^{obs} (= Nmin), but in plots 2, 6, 9 and 11 they are lower (than the observed number of nests). But the results are extremely similar to the Bayesian plot-level analysis and also compare well to the data-augmentation results (which are essentially identical to N_s^{obs}):

```
cor(cbind(mlEst = bestguessNtrue, BayesEst = antdetectBinom$BUGSoutput$mean$Ntrue,
    dataAug = estimated[, 2]))
```

```
             mlEst  BayesEst   dataAug
mlEst    1.0000000 0.9963630 0.9517471
BayesEst 0.9963630 1.0000000 0.9717419
dataAug  0.9517471 0.9717419 1.0000000
```

```
par(mar = c(5, 5, 1, 1))
plot(estimated[, 2], antdetectBinom$BUGSoutput$mean$Ntrue, xlab = expression(hat(N)[s]^{
    augmented
}), ylab = expression(hat(N)[s]), las = 1)
points(estimated[, 2], bestguessNtrue, pch = 16)
abline(0, 1)
legend("bottomright", pch = c(1, 16), legend = c("Bayes estimates",
    "ML estimates"), bty = "n", cex = 1.2)
```



The figure shows estimates from the site-level Bayesian and maximum likelihood approach (y-axis) against the nest-level data-augmentation results (x-axis). Line gives perfect accordance (1:1).

We think that we can thus use the site-level maximum likelihood approach in lieu of the Bayesian plot-level model, particularly when in the next step we re-run the analysis many times for different combinations of observers. This takes only seconds using the maximum likelihood approach, but would take many hours with the Bayesian plot-level model.

Computing confidence intervals or standard errors for the maximum likelihood estimates is a bit involved, as asymptotic errors (based on the Hessian matrix) are very unreliable for such small data sets. We thus use bootstrapping instead.

```
# draw, for each plot, with replacement from the nests data set:
plotNames <- substring(rownames(nests), 3, 4)
```

```r
n.bs <- 1000  # number of bootstraps
detProb.bs <- matrix(NA, nrow = n.bs, ncol = 24)
for (n in 1:n.bs) {
    plots.bs <- matrix(0, nrow = 16, ncol = 8)
    rownames(plots.bs)[c(1:3, 5:16)] <- unique(plotNames)
    rownames(plots.bs)[4] <- "04"
    for (i in unique(plotNames)) {
        thisPlot <- nests[which(plotNames == i), ]
        plotBS <- thisPlot[sample(nrow(thisPlot), nrow(thisPlot),
            replace = T), ]
        plots.bs[rownames(plots.bs) == i, ] <- colSums(plotBS)
    }
    (op.bs <- optim(par = c(log(Nmin + 1), rep(0.5, 8)), fn = toopt,
        Nobs = plots.bs, method = "BFGS"))
    detProb.bs[n, 17:24] <- plogis(op.bs$par[17:24])  # detection rates
    detProb.bs[n, 1:16] <- exp(op.bs$par[1:16])  # nest estimates
}
# observation rate estimates:
round(colMeans(detProb.bs[, 17:24]), 2)
```

```
[1] 0.34 0.36 0.40 0.38 0.45 0.39 0.55 0.58
```

```r
round(apply(detProb.bs[, 17:24], 2, sd), 3)
```

```
[1] 0.059 0.059 0.066 0.064 0.073 0.064 0.095 0.110
```

```r
round(apply(detProb.bs[, 17:24], 2, quantile, c(0.025, 0.975)), 3)
```

```
       [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
2.5%  0.246 0.263 0.301 0.279 0.342 0.299 0.418 0.434
97.5% 0.481 0.496 0.564 0.532 0.635 0.548 0.794 0.866
```

```r
# nest number estimates:
round(colMeans(detProb.bs[, 1:16]), 2)
```

```
 [1] 27.58 15.92 11.28  0.00 23.27  1.50  0.59  6.15  9.25 19.95
[11]  0.30 18.49  3.79  4.99 15.63 17.77
```

```r
round(apply(detProb.bs[, 1:16], 2, sd), 3)
```

```
 [1] 6.177 3.322 2.692 0.000 4.343 0.300 0.086 1.897 2.700 3.853
[11] 0.043 3.317 1.401 1.491 3.772 3.425
```

```r
round(apply(detProb.bs[, 1:16], 2, quantile, c(0.025, 0.975)), 3)
```

```
        [,1]   [,2]   [,3] [,4]   [,5]  [,6]  [,7]   [,8]   [,9]
2.5%  19.428  9.375  6.316    0 14.924 0.948 0.409  2.804  4.769
97.5% 42.517 22.337 16.642    0 31.418 2.100 0.744 10.256 15.143
       [,10] [,11]  [,12] [,13] [,14]  [,15]  [,16]
2.5%  12.183 0.205 11.779 1.720 2.148  8.393 11.046
97.5% 27.105 0.372 24.678 7.025 7.921 22.960 24.311
```

8 Simulating more (and fewer) observers

8.1 How many nests would we have estimated with fewer observers?

It is easy to simulate fewer observers by simply randomly drawing the desired number of observers from the data and repeating the (maximum-likelihood) analysis (the Bayesian would take quite a long time, since we have to repeat this random drawing many times).

```
k <- 3
Y <- Nobs[, c(1, 3, 5)]
tooptk <- function(parms, k) {
    Ntrue <- exp(parms[1:16])  # ensure positive values
    Pi <- plogis(parms[17:(17 + k - 1)])  # ensure values in [0,1]
    estY <- tcrossprod(Ntrue, Pi)
    -sum(dpois(as.matrix(Y), lambda = as.matrix(estY), log = T))
}
(opk <- optim(par = c(log(Nmin + 1), rep(0, k)), fn = tooptk, k = k,
    control = list(maxit = 20000)))
```

```
$par
 [1]  4.2022158  3.2148259  2.9396221 -1.6024937  4.0601943
 [6] -2.4845207  0.6931455  2.7263550  2.8637990  3.8470075
[11] -2.5610937  3.8868345  1.3011575  2.2898649  3.7009533
[16]  3.7855818 -1.7505154 -1.5653785 -1.4278722

$value
[1] 67.59848

$counts
function gradient
   10566       NA

$convergence
[1] 0

$message
NULL
```

```
round(exp(opk$par[1:16]), 1)  # estimated number of nests
```

```
 [1] 66.8 24.9 18.9  0.2 58.0  0.1  2.0 15.3 17.5 46.9  0.1 48.8
[13]  3.7  9.9 40.5 44.1
```

```
round(plogis(opk$par[17:(17 + k - 1)]), 2)  # estimated detProb per observer
```

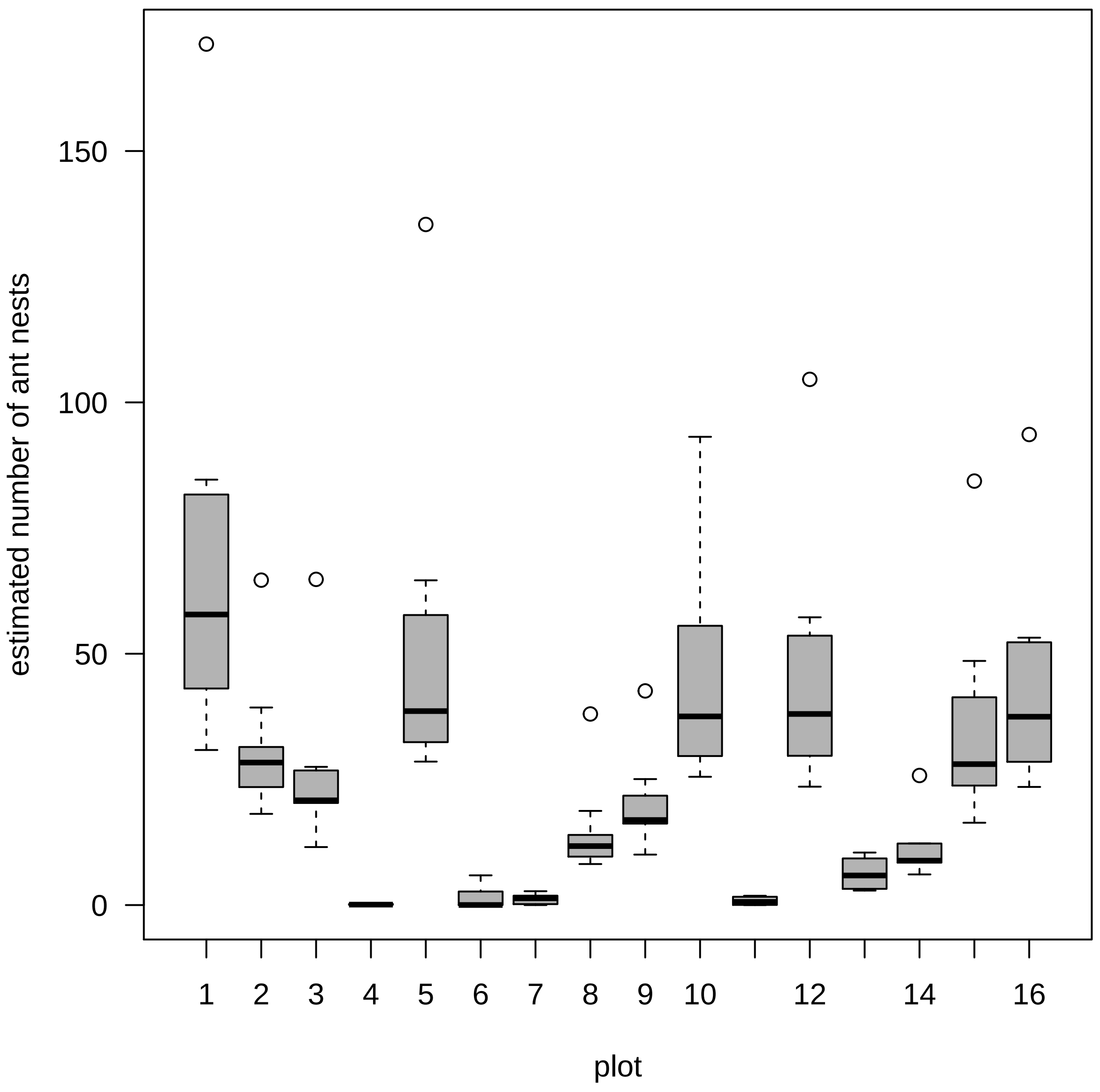
```
[1] 0.15 0.17 0.19
```

```
# this can be looped through 1000 times to get error bars for the
# estimates, always drawing a random set of observers; and then we
# repeat this with k=2 to k=8; here the example for k=3:
Nreps <- 10  # only for illustration the value is set low; set this to something larger (takes .3s per
```

```r
k <- 3
trueNmat3 <- matrix(NA, ncol = 16, nrow = Nreps)
for (i in 1:Nreps) {
    # choose a random set of observers:
    useTheseObs <- sample(8, k)
    Y <- Nobs[, useTheseObs]
    # compute the trueY based on this:
    opk <- optim(par = c(log(Nmin + 1), rep(0, k)), fn = tooptk, k = 3,
        control = list(maxit = 50000))
    if (opk$convergence != 0)
        stop("not converged!")
    trueNmat3[i, ] <- exp(opk$par[1:16])
    cat(i, " ")
}
```

```
1  2  3  4  5  6  7  8  9  10
```

```r
boxplot(trueNmat3, las = 1, ylab = "estimated number of ant nests",
    xlab = "plot", col = "grey70")
```



Note that these values are *much* higher when only few observers are used for estimation. Let us briefly try and understand why.

The estimated values of the observed $N_{i,s}$ are the cross-product of the estimated number of nests in a plot, $\hat{N}_s$ and each observers detection probability, P_i. If we estimate high values for $\hat{N}_s$, we can 'compensate' this by low values of P_i. If variability is high (because we have only few observers), we have little to go on for estimating either value. There is now the choice between 'many nest, poor detection rates' and 'few nests, high detection'. As it works out, in the binomial probability mass function, it is easier to accommodate highly variable data with 'many nests, poor detection'. Thus, the poorer the data (read: the fewer observers), the more the estimation will overestimate the true number of nests.

Repeating this for different values of k will lead to Fig. 3 in the main text. We do not provide the code here, but it is near-trivial to adapt the above for any value of k.

8.2 How does overall detection rate change with the number of observers like ours?

As detailed in the main paper, we require two different probabilities to simulate *more* observers: (a) the detection rate P_i of each observer i (which we have from either the maximum-likelihood estimation or the Bayesian analysis); and (b) the probability that a second observer discovers a **new** (complementary) nest, P_c. We compute P_c for each pair of observers, yielding a matrix from which to sample when simulating more observers (or indeed fewer).

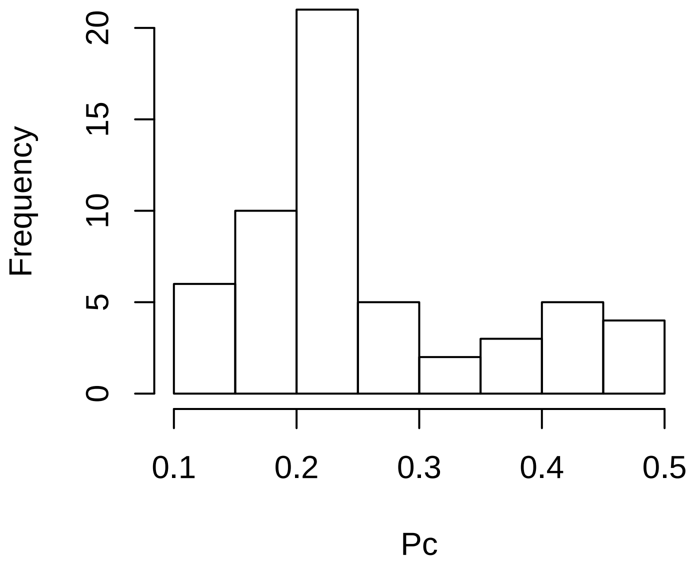
```
# add overlooked species according to jack1 estimate (intermediate
# between Chao and jack2)
nestsAll <- rbind(as.matrix(nests), matrix(0, nrow = 25, ncol = 8))
# nests which second observers found but first overlooked
# (quantifying complementarity):
Pc.mat <- matrix(0, 8, 8)
colnames(Pc.mat) <- rownames(Pc.mat) <- colnames(nests)
for (i in 1:8) {
    for (j in 1:8) {
        tt <- table(nestsAll[, i], nestsAll[, j])
        Pc.mat[i, j] <- tt[1, 2]/sum(tt[1, ])  # proportion of 0s turned into 1s
    }
}
round(Pc.mat, 3)  # note that this matrix is (obviously) not symmetric!
```

```
      01    02    03    04    05    06    07    08
01 0.000 0.159 0.265 0.257 0.283 0.239 0.442 0.496
02 0.144 0.000 0.207 0.225 0.243 0.216 0.450 0.441
03 0.194 0.146 0.000 0.204 0.223 0.223 0.350 0.447
04 0.215 0.196 0.234 0.000 0.290 0.262 0.393 0.477
05 0.147 0.116 0.158 0.200 0.000 0.137 0.358 0.411
06 0.173 0.163 0.231 0.240 0.212 0.000 0.423 0.452
07 0.182 0.208 0.130 0.156 0.208 0.221 0.000 0.351
08 0.219 0.151 0.219 0.233 0.233 0.219 0.315 0.000
```

```
# mean Pc value:
mean(c(Pc.mat[lower.tri(Pc.mat)], Pc.mat[upper.tri(Pc.mat)]))  # 0.2565
```

```
[1] 0.2568836
```

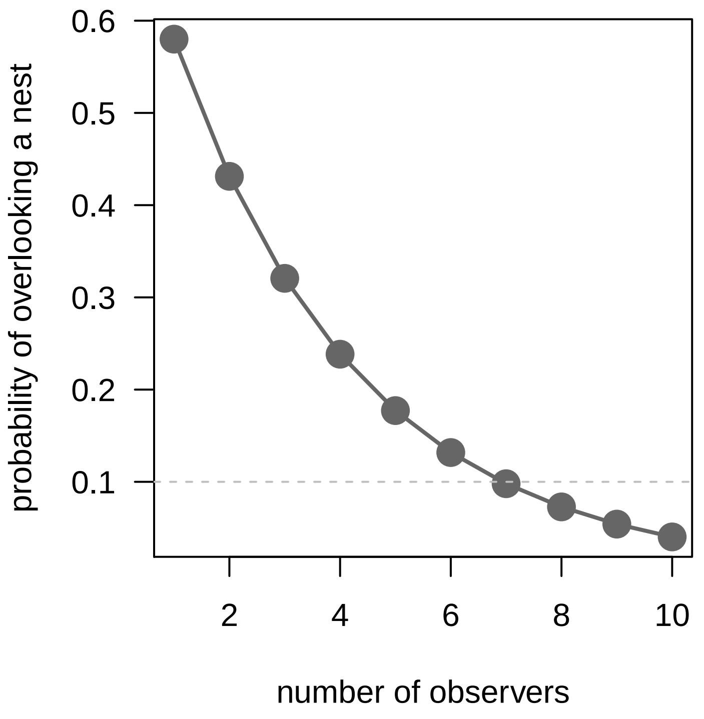
```
hist(c(Pc.mat[lower.tri(Pc.mat)], Pc.mat[upper.tri(Pc.mat)]), main = "",
    xlab = "Pc")
```



```
# note that this is the complement to the value computed above in
# jointly.mat!!
```

We now have a matrix with values representing the probabilities of discovering new nests overlooked by one previous observer. The chance of k observers to **all** overlooking a nest is $(1 - P_c)^k(1 - P_i)$. We turn this into a little function and plot it for k from 1 to 10.

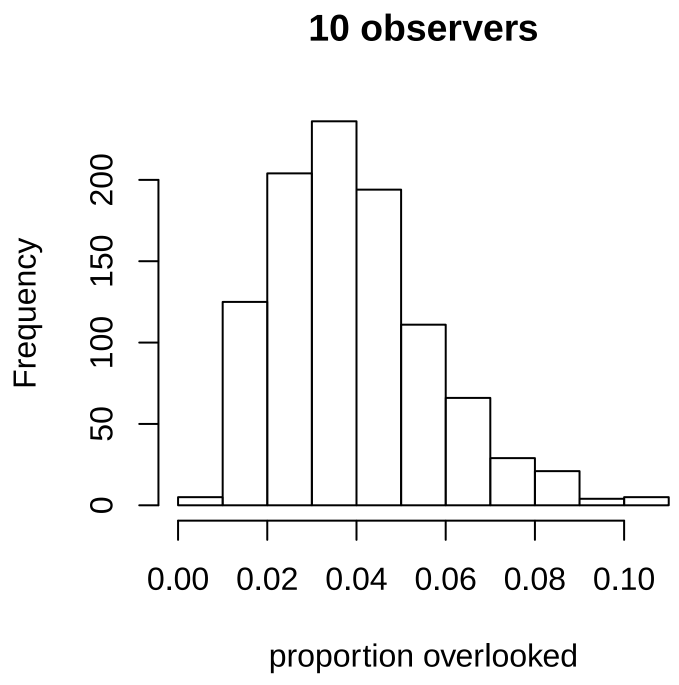
```
overlooked <- function(k, Pc = 0.2565, Pi = 0.42) {
    # Pd is mean of Bayesian estimates returns the probability of
    # having overlooked nests
    (1 - Pc)^(k - 1) * (1 - Pi)
}
par(mar = c(5, 5, 1, 1))
plot(1:10, overlooked(1:10), type = "o", pch = 16, cex = 2, las = 1,
    ylab = "probability of overlooking a nest", xlab = "number of observers",
    lwd = 2, col = "grey40")
abline(h = 0.1, col = "grey", lty = 2)
```



Clearly, and obviously, the more observers we have, the lower is the chance of overlooking a nest. With 8 observers we cross to below 10% overlooked nests.

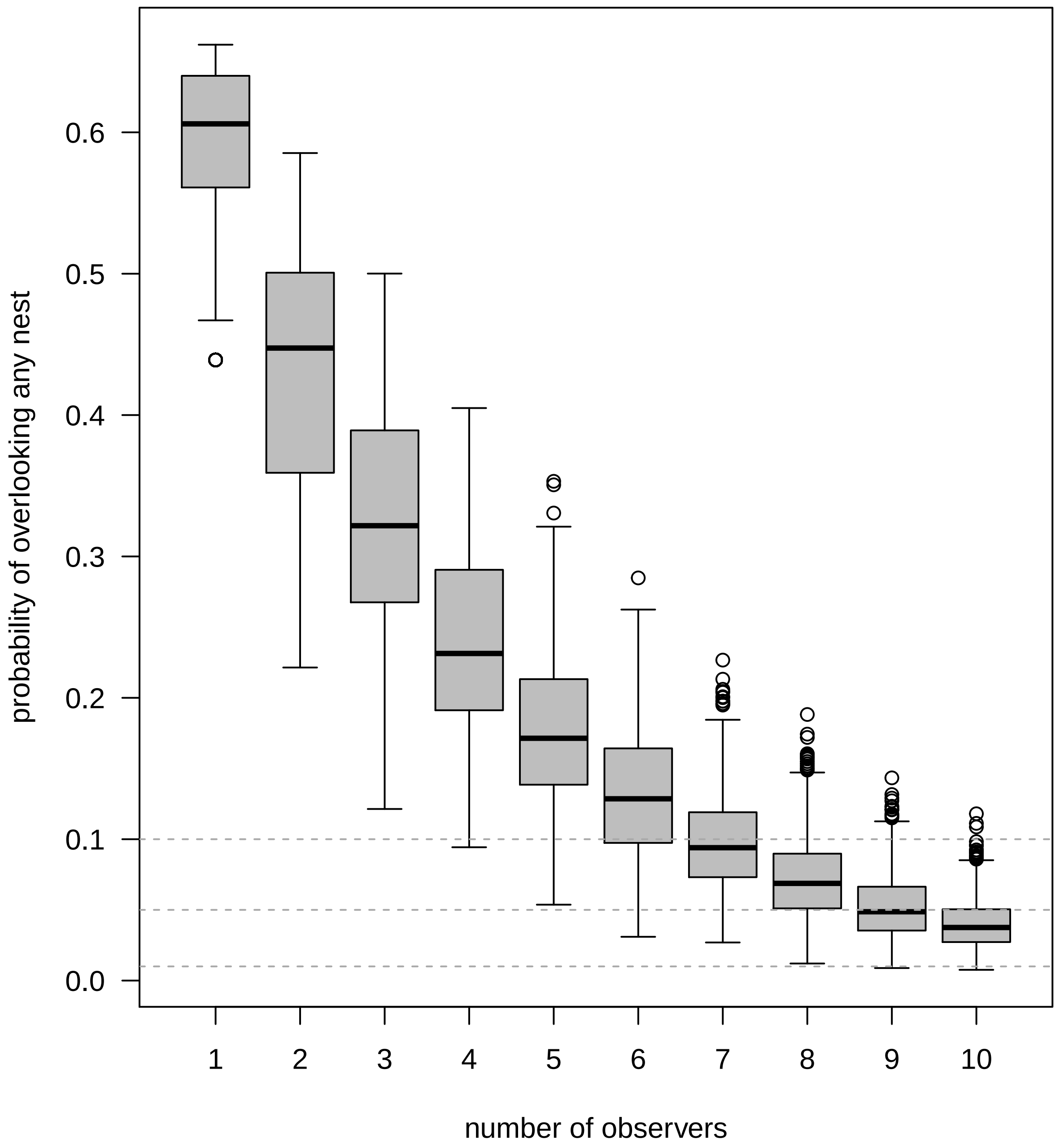
This plot ignores the variation around the detection and complementarity rates. So we now open the function to bootstrapping (i.e. sampling with replacement) P_i and P_c and run it on the real data (P_i from maximum likelihood).

```r
overlookedBS <- function(k, Pc, Pi) {
    # recursive problem!
    if (k == 1)
        return((1 - sample(Pi, 1)))  # for one observer: 1-detection probability
    # for two or more observers:
    (1 - sample(Pc, 1)) * overlookedBS(k - 1, Pc, Pi)
}
# run a test for 10 observers, 1000 repetitions:
hist(replicate(1000, overlookedBS(10, Pc = c(Pc.mat[lower.tri(Pc.mat)],
    Pc.mat[upper.tri(Pc.mat)]), Pi = plogis(op$par[17:24]))), main = "10 observers",
    xlab = "proportion overlooked")
```



So we can now compute the overlooking probability for 1 to 10 (or more) observers.

```
simuObservers <- sapply(1:10, function(x) replicate(1000, overlookedBS(x,
    Pc = c(Pc.mat[lower.tri(Pc.mat)], Pc.mat[upper.tri(Pc.mat)]),
    Pi = c(0.338, 0.35, 0.394, 0.37, 0.439, 0.388, 0.533, 0.561))))
par(mar = c(5, 5, 1, 1))
boxplot(simuObservers, las = 1, whisklty = 1, col = "grey", ylab = "probability of overlooking any nest
    xlab = "number of observers")
abline(h = c(0.1, 0.05, 0.01), col = "darkgrey", lty = 2)  # targets for accuracy
```



The black lines are the same values as in the previous dot-plot, but now we also get an estimate of the uncertainty around this value.

9 Non-parametric omission error analysis

For completeness, we also include the more traditional way to estimate overlooked nests, building on coarse approximations of the ratio of rare and common events. Each observer's records are a sample of the true nests at each plot. We can use non-parametric richness estimators to predict the number of nests across all plots. This is akin to having multiple recordings of a community and estimating the total number of species in the pool. The nests-data have to be transposed before analysis to have "species" (i.e. nest locations) as columns.

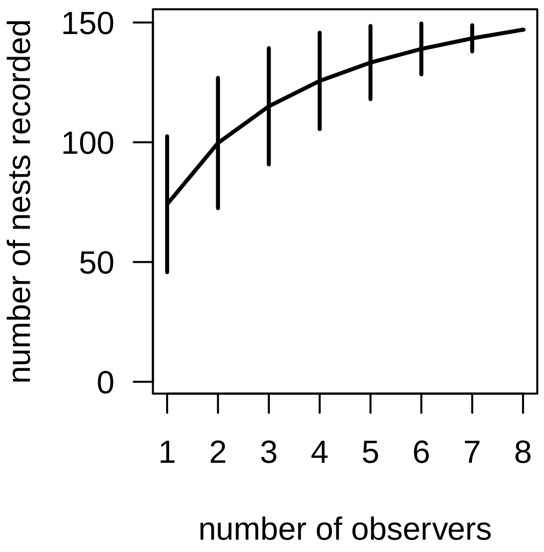
```r
library(vegan)
```

```r
specpool(t(nests))
```

```
    Species     chao  chao.se   jack1 jack1.se    jack2     boot
All     147 163.7244 8.247339 172.375 11.88289 179.9821 159.8692
     boot.se n
All 8.652339 8
```

As boot is frequently reported as underestimating the true richness, we shall only use Chao's S, jack1 S and jack2 S as estimators of the true number of nests across all plots. Thus, we have sampled 147/164*100% = 90% (Chao), 85% (jack1) or 82% (jack2) of all nests, which suggests a high sampling coverage:

```r
par(mar = c(4, 4, 1, 1))
plot(specaccum(t(nests)), xlab = "number of observers", ylab = "number of nests recorded",
    las = 1, lwd = 2)
```

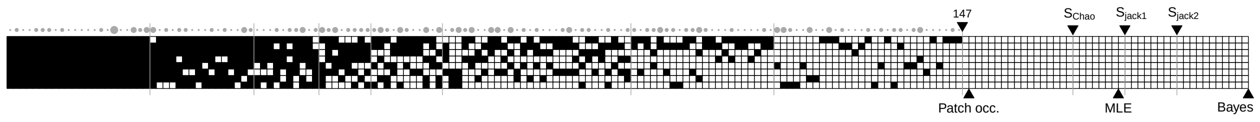


To visualise which ant nests were detected by who, we can plot the nests data as a matrix.

```r
library(bipartite)
```

```r
sizes <- read.csv("nestSizes.csv", row.names = 1)  # read file in again to get all nest sizes
nestSize <- sizes$Height
par(oma = c(0, 0, 0, 0.1))  # create space for label to the right
# overlooked species according to jack2 estimate (see further
# below):
nests2 <- rbind(as.matrix(nests), matrix(0, nrow = 44, ncol = 8))
visweb(t(nests2), labsize = 5, prednames = F, preynames = F, clear = F)
# here comes the symbol/col for nest size:
points(((1:147) - 0.5)[order(rowSums(nests), decreasing = TRUE)],
    rep(9, 147), pch = 16, col = "darkgrey", cex = 1.3 * ifelse(nestSize ==
        10, 0.3, ifelse(nestSize == 50, 0.6, ifelse(nestSize == 100,
        0.9, 1.2))))
```

```r
# lines for number of observers/nest:
lines(c(22, 22), c(-1, 10), col = "darkgrey")
lines(c(22 + 16, 22 + 16), c(-1, 10), col = "darkgrey")
lines(c(22 + 16 + 10, 22 + 16 + 10), c(-1, 10), col = "darkgrey")
lines(c(22 + 16 + 10 + 8, 22 + 16 + 10 + 8), c(-1, 10), col = "darkgrey")
lines(c(22 + 16 + 10 + 8 + 11, 22 + 16 + 10 + 8 + 11), c(-1, 10),
    col = "darkgrey")
lines(c(22 + 16 + 10 + 8 + 11 + 29, 22 + 16 + 10 + 8 + 11 + 29), c(-1,
    10), col = "darkgrey")
lines(c(22 + 16 + 10 + 8 + 11 + 29 + 22, 22 + 16 + 10 + 8 + 11 + 29 +
    22), c(-1, 10), col = "darkgrey")
lines(c(22 + 16 + 10 + 8 + 11 + 29 + 22 + 29, 22 + 16 + 10 + 8 + 11 +
    29 + 22 + 29), c(-1, 10), col = "darkgrey")
# indicator for the 147 detected nests:
points(147, 9.7, pch = 25, cex = 1.5, bg = "black")
text(147, 11.5, "147", cex = 1)  #expression(N[min])
# indicators for S_est and uncertainty:
lines(c(164, 164), c(-1, 8.5), col = "darkgrey")  # S_chao
points(164, 9.2, pch = 25, cex = 1.5, bg = "black")
text(164 + 1, 11.5, expression(S[Chao]), cex = 1.2)
lines(c(172, 172), c(-1, 8.5), col = "darkgrey")  # S_jack1
points(172, 9.2, pch = 25, cex = 1.5, bg = "black")
text(172 + 1, 11.5, expression(S[jack1]), cex = 1.2)
lines(c(180, 180), c(-1, 8.5), col = "darkgrey")  # S_jack1
points(180, 9.2, pch = 25, cex = 1.5, bg = "black")
text(180 + 1, 11.5, expression(S[jack2]), cex = 1.2)
# add MLE:
points(171, -1, pch = 24, cex = 1.5, bg = "black")
text(171, -3, "MLE", cex = 1.2)
points(191, -1, pch = 24, cex = 1.5, bg = "black")
text(191 - 2, -3, "Bayes", cex = 1.2)
points(148, -1, pch = 24, cex = 1.5, bg = "black")
text(148, -3, "Patch occ.", cex = 1.2)
```



Dots above the nest indicate the nest's size. Empty cells to the right were unrecorded by all observers. The best guess for $\hat{N}_{\text{true}}$ of the different methods is indicated by a triangle alongside the method.

Platform, session and package information:

```r
sessionInfo()
```

```
R version 3.2.3 (2015-12-10)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.10.5 (Yosemite)

locale:
[1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
```

```
attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
 [1] unmarked_0.11-0 Rcpp_0.12.5     reshape_0.8.5
 [4] lme4_1.1-12     Matrix_1.2-6    truncnorm_1.0-7
 [7] R2jags_0.5-7    rjags_4-6       coda_0.18-1
[10] bipartite_2.06  sna_2.3-2       vegan_2.3-5
[13] lattice_0.20-33 permute_0.9-0

loaded via a namespace (and not attached):
 [1] formatR_1.4      nloptr_1.0.4     plyr_1.8.4
 [4] tools_3.2.3      boot_1.3-18      digest_0.6.9
 [7] evaluate_0.9     nlme_3.1-128     mgcv_1.8-12
[10] igraph_1.0.1     yaml_2.1.13      parallel_3.2.3
[13] spam_1.3-0       raster_2.4-18    stringr_1.0.0
[16] cluster_2.0.4    knitr_1.13       fields_8.4-1
[19] maps_3.1.0       grid_3.2.3       rmarkdown_0.9.6
[22] sp_1.1-1         minqa_1.2.4      magrittr_1.5
[25] codetools_0.2-14 htmltools_0.3.5  R2WinBUGS_2.1-21
[28] MASS_7.3-45      splines_3.2.3    abind_1.4-3
[31] stringi_1.1.1
```