# Incidence Function Model in R

Jari Oksanen

May 11, 2004

**Abstract**

Metapopulation dynamics predicts that species incidences in patches are a function of colonization and extinction rates. Incidence function model finds colonization and extinction rates as a function of patch size and connectivity. Patch size and connectivity are relatively easy to estimate from the data. Fitting incidence function model is a special case of generalized linear models with binomial error and logistic link function. This document explains how to fit the model in R.

## Contents

# 1 The incidence function model

The incidence function model can be defined with the following set of equation (Hanski, 1999):

$$J_i = \frac{C_i}{C_i + E_i - C_i E_i} \tag{1}$$

$$E_i = \frac{e}{A_i^x}, \quad \text{for } A \geq e^{1/x} \tag{2}$$

$$M_i = \beta S = \beta \sum_{j \neq i}^{R} \exp(-\alpha d_{ij}) p_j A_j \tag{3}$$

$$C_i = \frac{M_i^2}{M_i^2 + y^2} = \frac{S_i^2}{S_i^2 + y}, \quad \text{where } y \text{ absorps } \beta \tag{4}$$

$$J_i = \frac{S_i^2 A_i^x}{S_i^2 A_i^x + ey} = \frac{1}{1 + \frac{ey}{S_i^2 A_i^x}} = \left[1 + \frac{ey}{S_i^2 A_i^x}\right]^{-1} \tag{5}$$

Here incidence $J_i$ in patch $i$ is defined twice: in eq. 1 incidence $J_i$ is defined in terms of colonization $C_i$ and $E_i$ extinction rates, and in eq. 5 in terms of patch connectivity $S_i$ and size $A_i$, and with three parameters $x, e, y$ than can be estimated from the data. Eq. 2 defines extinction rate $E$ as a function of patch size $A$ and two estimated parameters $e, x$. Eq. 3 defines connectivity $S$ as a function of occupancies $p_j$, patch sizes $A_j$, patch distances $d_{ij}$ and a species-specific dispersion length parameter $\alpha$. Eq. 4 defines colonization rate $C_i$ as a function of patch connectivity $S_i$ and one estimated parameter $y$. Finally, substituting $C_i$ and $E_i$ in eq. 1 with eqs. 2 and 4 yields eq. 5 after some acrobacy.

# 2 Preliminaries: Data and plotting

The data are stored in a data frame called `fritty`:

```
> load("fritty.rda")
> summary(fritty)
```

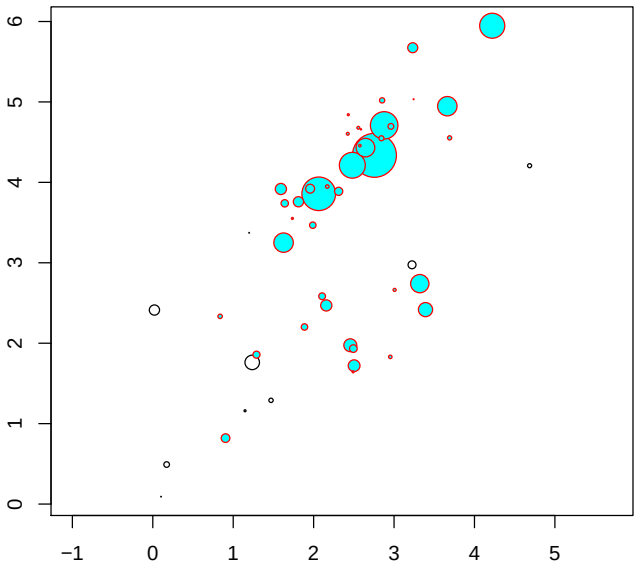
```
     x.crd            y.crd              A                p
 Min.   :0.0202   Min.   :0.0921   Min.   :0.0012   Min.   :0.00
 1st Qu.:1.6646   1st Qu.:2.2349   1st Qu.:0.0300   1st Qu.:1.00
 Median :2.4435   Median :3.6454   Median :0.1150   Median :1.00
 Mean   :2.2797   Mean   :3.3123   Mean   :0.4141   Mean   :0.82
 3rd Qu.:2.8707   3rd Qu.:4.5256   3rd Qu.:0.3225   3rd Qu.:1.00
 Max.   :4.6853   Max.   :5.9480   Max.   :4.6000   Max.   :1.00
       p2
 Min.   :0.00
 1st Qu.:0.25
 Median :1.00
 Mean   :0.74
 3rd Qu.:1.00
 Max.   :1.00
```

The data are fictitious: they were generated using simulation (section 4, page 14). However, the system resembles a *Melitaea cinxia* network: The patch areas (in ha) are the same, and the configuration of plots is similar as described by Hanski et al. (1994). The advantage of simulated data is that the real parameter values are known so that we can assess the accuracy and reliability of the fitting procedures. The following parameter values were used: $\alpha = 1$ (in km), $x = 0.41$, $y = 3.91^2$ and $e = 0.063$ (Hanski et al., 1994; Hanski, 1999). Please note that parameter $y$ appears both as squared and unsquared in eq. 4. It seems to me that the original usage was not quite consistent, and using squared values for the given $y$ resulted in more consistent fit. The remaining two variables `p` and `p2` are simulated presences in two consecutive simulation years.

The standard plotting uses equal aspect ratio (`asp=1`), scales the symbol area to patch size (`cex`), and uses different colour and shading for occupied and empty patches. The graph (Fig. 1) uses easy shortcuts utilizing the notation where empty patches are `p = 0`, and occupied are `p = 1`.

**Figure 1** Basic plot of occupied and empty metapopulation patches.

```
> attach(fritty)
> plot(x.crd, y.crd, asp = 1, xlab = "", ylab = "", cex = sqrt(A *
+     5), pch = 21, col = p + 1, bg = 5 * p)
```



# 3 Fitting

## 3.1 Theory

Fitting of the incidence function model is based on eq. 5, where the incidence is given as a function of connectivity $S$ and patch size $A$, which both can be found from the data. The incidence function model (eq. 5) can be parameterized as a linear model for the log-odds of incidence:

$$J_i = \left[1 + \frac{ey}{S_i^2 A_i^x}\right]^{-1} \tag{6}$$

$$= \frac{1}{1 + \exp(\log(ey) - 2\log(S_i) - x\log(A_i))} \tag{7}$$

$$\log\left(\frac{J_i}{1 - J_i}\right) = -\log(ey) + 2\log(S_i) + x\log(A_i) \tag{8}$$

$$\operatorname{logit}(J_i) = \beta_0 + 2\log S + \beta_1 \log A \tag{9}$$

The final equation 9 defines a generalized linear model with logistic link function. The response variable are the presences $p_i$ of species in patches. This is clearly a binomial variate, and fitting the incidence function applied standard generalized linear model (McCullagh and Nelder, 1989; Venables and Ripley, 2002) with $\log S$ and $\log A$ as independent variables, binomial error and logistic link function. The remaining problems are deriving connectivity $S$, and separating parameters $e$ and $y$ of eqs. 5 and 6, which are combined into single parameter $\beta_0$.

This suggests the following procedure:

1. Given data are patch sizes $A_i$, patch occupancies $p_i$, and patch locations from which we find distances $d_{ij}$.
2. Use additional field observations or dirty tricks to find the strength of distance decay $\alpha$.
3. Estimate connectivity (isolation) $S$ from $A_i$, $d_{ij}$ and fixed $\alpha$ as $S_i = \sum_{j \neq i}^{R} \exp(-\alpha d_{ij}) p_j A_j$.
4. Fit GLM using $p$ as dependent variable, and $\log A$ and $\log S$ as indepent variables to get estimates of $-\log(ey)$ and $x$ in $E_i = e/A_i^x$ and $C_i = S_i^2/(S_i^2 + y)$.
5. Tear apart $e$ and $y$ (which may be painful, but is needed for simulation).

The only remaining problem is to separate estimates of $e$ and $y$ from their estimated product $\hat{\beta}_0 = -\log(\widehat{ey})$. This cannot be solved from the fitted GLM, since any pair of values of $e$ and $y$ giving the estimated $\widehat{ey}$ are just equally good. However, if we manage to fix either $e$ or $y$, the another will be found with division. For a single snapshot we may assume that the smallest plot where the species was present is of the size where extinction probability $E = 1$:

$$\widehat{ey} = \exp(-\hat{\beta}_0) \tag{10}$$

$$\tilde{e} = \min_{p \neq 0} A^{\hat{x}} \tag{11}$$

$$\tilde{y} = \widehat{ey}/\tilde{e} \tag{12}$$

We shall inspect another alternative with two consecutive surveys (section 3.4, page 8).

## 3.2 Fitting a snapshot in R

Standard function `dist` can be used to find the Euclidean distances $d$ among patches. Then we need an estimate of $\alpha$, and setting diagonal to zero means taking only $d_{j \neq i}$. Finally we multiply the columns with $A$, and get the row sums for occupied sites, so that $S = \sum_{j \neq i} \exp(-\alpha d_{ij}) p_j A_j$:

```
> d <- dist(cbind(x.crd, y.crd))
> alpha <- 1
> edis <- as.matrix(exp(-alpha * d))
> diag(edis) <- 0
> edis <- sweep(edis, 2, A, "*")
> S <- rowSums(edis[, p > 0])
```

Distances `d` $= \{d_{ij}\}$ and matrix `edis` $= \{\exp(-\alpha d_{ij})\}$ will be re-used several times in this document. Actually, it is unnecessary to explicitly set diagonal of `edis` to zero, since they are zero anyhow. Now we have the two variables $S$ and $A$ needed in fitting the incidence function model. Binomial fitting with logistic link function is waiting for us in R:

```
> mod <- glm(p ~ offset(2 * log(S)) + log(A), family = binomial)
> summary(mod)

Call:
glm(formula = p ~ offset(2 * log(S)) + log(A), family = binomial)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.365   0.133   0.279   0.446   1.563

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    0.590      0.905    0.65    0.515
log(A)         0.471      0.264    1.78    0.075

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 30.314  on 49  degrees of freedom
Residual deviance: 27.095  on 48  degrees of freedom
AIC: 31.10

Number of Fisher Scoring iterations: 5
```

The model uses fixed coefficient $\beta = 2$ for connectivity $\log S$. In R, this can be achieved using `offset`. Consequently, only two coefficients are estimated. The response variable `p` is binary (0 or 1), so we need not give the binomial denominator. The logistic function is the default link function in binomial models, and need not be specified explicitly.

The parameters of the incidence function are:

```
> beta <- coef(mod)
> (xhat <- beta[2])

 log(A)
0.47088

> (A0 <- min(A[p > 0]))

[1] 0.002

> (ey <- exp(-beta[1]))

(Intercept)
    0.55444

> (etilde <- A0^xhat)

  log(A)
0.053595

> (ytilde <- ey/etilde)

(Intercept)
     10.345
```

The real values used in simulation were $x = 0.41$, $e = 0.063$, and $y = 3.91^2 = 15.29$.

In plotting the results, it is natural to use colour to show the predicted incidence. The graph (Fig. 2) uses heat colours: the redder, the higher the incidence. R knows several other palettes that can be used.

### 3.3 Fitting data from two surveys

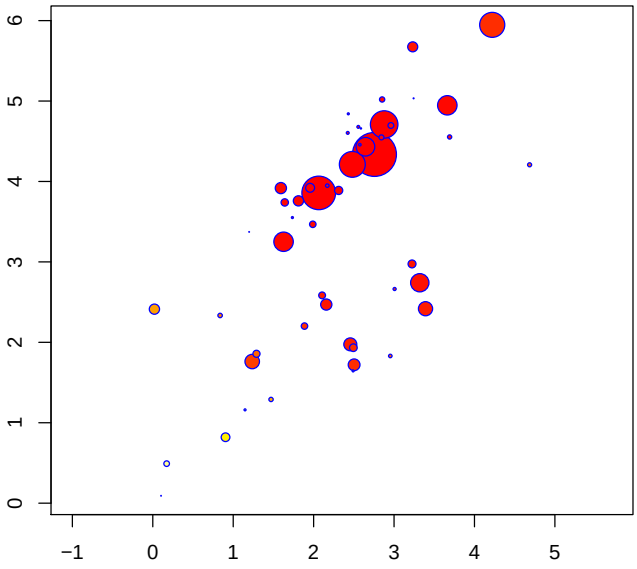
Fitting a model for two (or more) surveys is nearly as simple as fitting a snapshot. With several surveys, the response variable is the number (integer) of observed occupancies, and binomial denominator (another integer) is the number of surveys. In principle, the denominator can vary among observations, but this is hardly sensible in incidence function models. However, the observed proportional incidences must be used in weighting the patches when assessing connectivity.

In the following, we combine the two snapshots `p` and `p2` into one sum vector `P`. Distances weighted by patch sizes were already calculated above and saved to `edis`, and we need only weight these with observed frequencies when getting the connectivity. The definition of the GLM is slightly more complicated, too. When binomial denominator is larger than one, the response variable should be given as a two-column matrix, with success (occupied) and failure (empty) counts as columns.

```
> P <- p + p2
> S <- rowSums(sweep(edis, 2, P/2, "*"))
> mod2 <- glm(cbind(P, 2 - P) ~ offset(2 * log(S)) +
+     log(A), family = binomial)
> summary(mod2)
```

**Figure 2** Fitted incidences.

```
> col <- heat.colors(100)[99 * (1 - fitted(mod)) + 1]
> plot(x.crd, y.crd, asp = 1, xlab = "", ylab = "", pch = 21,
+      col = "blue", bg = col, cex = sqrt(A * 5))
```




```
Call:
glm(formula = cbind(P, 2 - P) ~ offset(2 * log(S)) + log(A),
    family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0874  -0.0432   0.4266   0.7006   2.5012

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.0701     0.5590    0.13    0.900
log(A)        0.4148     0.1692    2.45    0.014

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 59.245  on 49  degrees of freedom
Residual deviance: 53.166  on 48  degrees of freedom
AIC: 65.48
```

```
Number of Fisher Scoring iterations: 5
```

The incidence function parameters can be found in the same way as above:

```
> beta <- coef(mod2)
> xhat <- beta[2]
> ey <- exp(-beta[1])
> etilde <- min(A[P > 0])^xhat
> ytilde <- ey/etilde
> par <- c(xhat, etilde, ytilde)
> names(par) <- c("x", "e", "y")
> par

       x        e        y
 0.41482  0.06143 15.17674
```

### 3.4 Separating $e$ and $y$ with two surveys

With two consecutive surveys, it is possible to estimate the parameter $y$ from the number of transitions $T$ between two surveys. Transition is an empty patch becoming occupied, or an occupied patch becoming empty. The estimate $\tilde{y}$ can be found as the root of the equation (Hanski, 1999):

$$\sum_{i=1}^{R} \frac{1}{S_i^2 + \tilde{y}} \left[ S_i^2(1 - p_i) + \frac{\widehat{ey} p_i}{A_i^{\hat{x}}} \right] = T \tag{13}$$

After solving $\tilde{y}$, the remaining parameter can be estimated as $\tilde{e} = \widehat{ey}/\tilde{y}$. R provides function `uniroot` for solving equations. We have to write a function to be minimized. It is most practical to define the function with only the estimated parameter, and wish that the function will find other needed parameters in the local scope. The only extra parameter needed is the fork $(0, 50)$ where we search the root. It is not safe to use variable name `T`, as it may be taken as a shortcut for logical `TRUE`.

```
> f <- function(x) sum(1/(S * S + x) * (S * S * (1 -
+     pmean) + ey * pmean/A^xhat)) - Tpar
> (Tpar <- sum(p != p2))

[1] 6

> beta <- coef(mod2)
> xhat <- beta[2]
> ey <- exp(-beta[1])
> pmean <- P/2
> (sol <- uniroot(f, c(0, 50)))

$root
[1] 13.048

$f.root
[1] -7.394e-07
```

```
$iter
[1] 8

$estim.prec
[1] 6.1035e-05
```

```
> etilde <- ey/sol$root
> par2 <- c(xhat, etilde, sol$root)
> names(par2) <- c("x", "e", "y")
> rbind(par, par2)
```

```
           x        e      y
par  0.41482 0.061430 15.177
par2 0.41482 0.071452 13.048
```

## 3.5 Covariates

It is trivial to add covariates to a GLM, but it is much more difficult to understand what this means in terms of colonization and extinction probabilities. Hanski (1999) discusses in detail how to do this in a structured way, but fitting models becomes more intricate (although fitting non-linear maximum likelihood models directly is easy in R). Here I inspect the meaning of adding covariates in a GLM model.

There are no real covariates that can be used in our data, since the data really are generated with a completely defined simulation model with no covariates. Therefore I use random data in the following examples.

The first case concerns adding a class covariate with two levels:

```
> Class <- factor(sample(c("A", "B"), length(p), replace = TRUE))
> modc <- glm(p ~ offset(2 * log(S)) + log(A) + Class -
+     1, family = binomial)
> coef(modc)
```

```
 log(A)  ClassA  ClassB
0.50603 0.56676 1.01305
```

Term `-1` in the model formula usually removes the intercept (which cannot be done), but with class covariate it just parametrizes the model so that the coefficients give fitted averages of factor levels. Now these factor coefficients are the estimates of $-\log(\widehat{ey})$ for each factor level, and we should break them separately for the estimate of $e$ and $y$. Parameter $e$ is site-specific and influences extinction probability (eq. 2), but parameter $y$ influences the connectivity (eq. 3). It is natural to think that $y$ should be constant for both factor levels, but the differences should be shown in $e$. This cannot be done easily, except for two surveys (section 3.4). Further, this fitting assumes that another site specific variable $x$ is constant and independent of the covariate. We can relax this assumption with the following model:

```
> modc <- glm(p ~ offset(2 * log(S)) + Class/log(A) -
+     1, family = binomial)
> coef(modc)
```

```
      ClassA        ClassB ClassA:log(A) ClassB:log(A)
   -0.16864       2.81093       0.20055       0.95425
```

This fits a nested model and the coefficients give separate estimates of both $\hat{x}$ and $-\log(\widehat{ey})$ for factor levels. There still remains the problem of separating $e$ and $y$.

The model is just as simple to define with a continuous covariate:

```
> vec <- runif(length(p))
> modv <- glm(p ~ offset(2 * log(S)) + log(A) + vec,
+     family = binomial)
```

However, now $-\log(\widehat{ey})$ is dependent on the value of the continuous covariate, and typically is different for all observations:

```
> (b <- coef(modv))

(Intercept)      log(A)         vec
   -0.20779     0.47955     1.58994

> ey <- exp(-(b[1] + b[3] * vec))
```

Separating $e$ and $y$ may again be impossible or painful.

The interaction term with $\log(A)$ should be defined as:

```
> modv <- glm(p ~ offset(2 * log(S)) + log(A) * vec,
+     family = binomial)
```

Now the estimate of $\hat{x}$ is dependent on the covariate, too:

```
> (b <- coef(modv))

(Intercept)      log(A)         vec  log(A):vec
   0.534362    0.689839    0.040317   -0.450945

> xhat <- b[2] + b[4] * vec
> ey <- exp(-(b[1] + b[3] * vec))
```

An alternative is to keep the intercept $-\log(\widehat{ey})$ constant, but let $\hat{x}$ vary with the covariates. I show below how this can be done for continuous and factor covariates, but analyse in more detail only the latter case:

```
> modx <- glm(p ~ offset(2 * log(S)) + log(A) * vec -
+     vec, family = binomial)
> coef(modx)

(Intercept)      log(A)  log(A):vec
    0.55489     0.69441    -0.46030

> modx <- glm(p ~ offset(2 * log(S)) + Class/log(A) -
+     Class, family = binomial)
> (b <- coef(modx))

  (Intercept) ClassA:log(A) ClassB:log(A)
      0.58606       0.41923       0.49246
```

```
> A0 <- min(A[p > 0])
> (etilde <- A0^b[2:3])

ClassA:log(A) ClassB:log(A)
     0.073876      0.046866

> (ytilde <- exp(-b[1])/etilde)

ClassA:log(A) ClassB:log(A)
       7.5331       11.8746
```

This fits a model where $\hat{x}$, $\tilde{e}$ and $\tilde{y}$ are dependent on the covariate, but $\tilde{e} \times \tilde{y} = \widehat{ey}$ is independent of the covariate. This is simpler, but not very as realistic as the previous more complete models.

The problems in model fitting concern only expressing the site effects with primitive parameters $e$ and $y$. The fitted models and their confidence limits may be appropriate, although we are unable to translate the GLM parametrization into coefficients $e$ and $y$. We can assess the "significance" of covariates in the usual way:

```
> anova(modc, test = "Chisq")

Analysis of Deviance Table

Model: binomial, link: logit

Response: p

Terms added sequentially (first to last)


             Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                           50       31.8
Class         2      1.5        48       30.3       0.5
Class:log(A)  2      5.1        46       25.2       0.1
```

Since we used a random covariate, its effect should be non-significant.

## 3.6 Estimating $\alpha$

We have analysed incidence function as a model with three estimated parameters $x$, $e$, and $y$. However, there is also a fourth parameter, $\alpha$ of eq. 3. Parameter $\alpha$ is specific to species, and it must be estimated separately. After estimating $\alpha$, it is fixed, and the other parameters are estimated conditionally to the fixed $\alpha$.

Parameter $\alpha$ is the inverse of average dispersal length, and typically it is estimated by studying the dispersal patterns. Hanski (1999) discusses various ways of estimating $\alpha$ before fitting the incidence function model. However, we can estimate $\alpha$ from the snapshot of site occupancies. This may be very inaccurate and misleading, but can be used as a last resort in lack of biological observations on the origin of occupied patches. In principle, fitting is extremely simple: we try with different values of $\alpha$ and select the value giving the best

fitting incidence function model. This means that we have to recalculate connectivities $S_i$ for each value of $\alpha$ making the procedure so long that it is best to write a separate function for the task. The criterion value of the function is the deviance of the fitted GLM, and we can use R function `optimize` to find the best value of $\alpha$:

```
> alphascan <- function(alpha, d, A, p) {
+     edis <- as.matrix(exp(-alpha * d))
+     diag(edis) <- 0
+     edis <- sweep(edis, 2, A, "*")
+     S <- rowSums(edis[, p > 0])
+     mod <- glm(p ~ offset(2 * log(S)) + log(A), family = binomial)
+     deviance(mod)
+ }
> (sol <- optimize(alphascan, c(0.1, 5), d = d, p = p,
+     A = A))

$minimum
[1] 1.5491

$objective
[1] 26.222
```

We scanned the interval $\alpha = 0.1 \ldots 5$, and got the estimate $\hat{\alpha} = 1.549$ with deviance 26.22. The real value used in simulation was $\alpha = 1$ with deviance 27.1. The difference of deviances is 0.8736 which has $p = 0.35$ in Chi-square distribution with one degree of freedom. Fig. 3 gives the profile deviance for $\alpha$: the deviance obtained with different levels of $\alpha$. The deviance is approximately distributed as Chi-squared, and we can use the profile deviance for assessing the confidence intervals of $\alpha$. More detailed discussion can be found consulting R documentation of functions `profile` and `confint`.

## 3.7 Confidence intervals of estimates

The standard errors of GLM estimates are directly available:

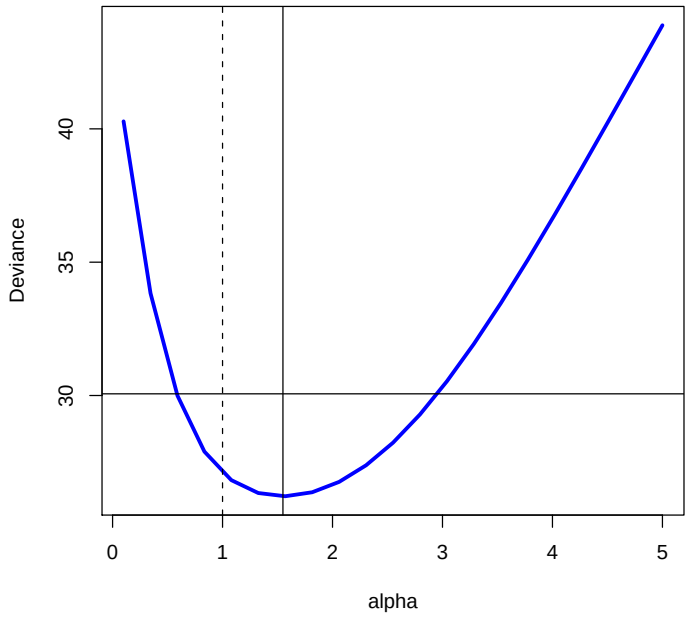
```
> (tmp <- summary(mod2)$coefficients)

            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.07010    0.55898 0.12541 0.900202
log(A)       0.41482    0.16915 2.45239 0.014191
```

Approximate 95 % confidence intervals are the parameter estimate $\pm 2$ times standard error:

```
> tmp[1, 1] + c(-2, 2) * tmp[1, 2]

[1] -1.0479  1.1881

> tmp[2, 1] + c(-2, 2) * tmp[2, 2]

[1] 0.076522 0.753124
```

**Figure 3** Profile deviance of $\alpha$. The horizontal line limits the deviance which deviates less than the critical level corresponding to $p = 0.05$ in Chi-square distribution with one *d.f.* and belong to the 95 % confidence interval of $\alpha$. The vertical lines show the best fitting (solid) and real values (dotted line) of $\alpha$.

```
> nseq <- 21
> alpha <- seq(0.1, 5, length = nseq)
> prof <- numeric(nseq)
> for (i in 1:nseq) prof[i] <- alphascan(alpha[i], d = d,
+     A = A, p = p)
> plot(alpha, prof, ylab = "Deviance", type = "l", col = "blue",
+     lwd = 3)
> abline(v = sol$minimum)
> abline(v = 1, lty = 2)
> abline(h = sol$objective + qchisq(0.95, 1))
```



or more cryptically but efficiently using `outer`:

```
> tmp[, 1] + t(outer(c(-2, 2), tmp[, 2]))
```

```
                 [,1]    [,2]
(Intercept) -1.047863 1.18806
log(A)       0.076522 0.75312
```

These confidence limits are based on the assumption of asymptotic normality of parameter estimates. More robust estimates are based on profile deviance, and can be found using function `confint.glm` (and implicitly `profile.glm`) of the `MASS` library (Venables and Ripley, 2002):

```
> library(MASS)
> confint(mod2)
```

```
Waiting for profiling to be done...
               2.5 % 97.5 %
(Intercept) -0.95580 1.2501
log(A)       0.08543 0.7561
```

We already looked at the profile deviance for $\alpha$ (section 3.6). However, if we estimate $\alpha$ from the data, the standard errors will be biased in GLM, since they are based on fixed $\alpha$.

The standard errors are for the GLM parameters $\hat{x}$ (`log(A)`) and $-\log(\widehat{ey})$ (`(Intercept)`). There is no easy way of estimating the standard errors of real parameters $e$ and $y$ from the standard error of their product.

Standard function `predict` can be used to find the standard errors and confidence limits of fitted incidences.

# 4 Simulation

For simulation, the incidence function model must be expressed in terms of colonization and extinction probabilities (eq. 1) for each site. Therefore parameters $e$ and $y$ must be separated from each other after fitting the model. The following pseudocode describes the simulation:

- Set $A_i$, $d_{ij}$, $\alpha$, $e$, $y$, $x$, $p_i(t) \in \{0, 1\}$
- $E_i \leftarrow eA_i^{-x}$
- for $t$ in Time:
  1. $S_i(t) \leftarrow \sum_{j \neq i} \exp(-\alpha d_{ij}) A_j \times p_j(t)$
  2. $C_i(t) \leftarrow \dfrac{S_i^2(t)}{S_i^2(t) + y}$
  3. for $i$ in Patches:
     (a) if $p_i(t) \in \{0\}$: fill at probability $C_i(t)$
     (b) if $p_i(t) \in \{1\}$: kill at probability $[1 - C_i(t)]E_i$

Parameters indexed with time ($t$) change in each simulation step. This means that extinction rate $E_i$ and $\sum_{j \neq i} \exp(-\alpha d_{ij}) A_j$ remain constant and can be solved once before simulation, but $S_i(t)$, $C_i(t)$ and $p(t)$ change. The fixed parameters are the observed patch sizes $A_i$, patch distances $d_{ij}$, species dispersion parameter $\alpha$, incidence function parameters $e, y, x$, and starting values of occupancies $p_i$.

An R implementation of a single simulation step is:

```
> metastep <- function(p, edis, E, y) {
+     p <- p > 0
+     if (any(p)) {
+         S <- rowSums(edis[, p, drop = FALSE])
+         C <- S^2/(S^2 + y)
+         cond <- ifelse(p, (1 - C) * E, C)
+         p <- ifelse(runif(length(p)) < cond, !p, p)
+     }
+     as.numeric(p)
+ }
```

Using current values of model parameters and previously calculated matrix `edis`, a single step can be run using:

```
> tmp <- p
> par
```

```
       x        e        y
 0.41482  0.06143 15.17674
```

```
> E <- pmin(par[2]/A^par[1], 1)
> tmp <- metastep(tmp, edis, E, par[3])
```

A simulation run of 100 timesteps is:

```
> occup <- matrix(0, nrow = length(p), ncol = 100 + 1)
> occup[, 1] <- p
> for (t in 1:100) occup[, t + 1] <- metastep(occup[,
+     t], edis, E, par[3])
```

Figure 4 demonstrates how to plot the simulation history, figure 5 compares simulated incidences against snapshot fitting.

# 5 Metacommunity capacity

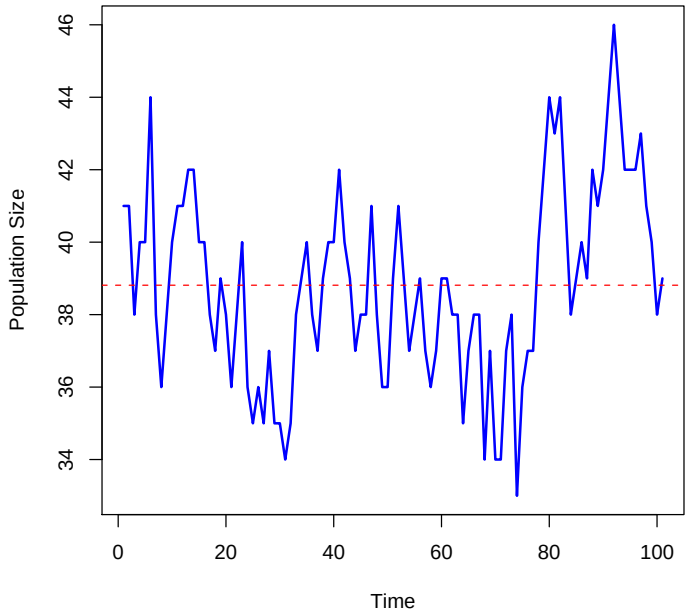
Metapopulation capacity describes the ability of a patch network to sustain a metapopulation (Hanski and Ovaskainen, 2000). It is defined as the largest ("leading") eigenvalue $\lambda_M$ of matrix $\mathbf{M}$ with elements

$$m_{ij} = \begin{cases} A_i \exp(-\alpha d_{ij}) A_j & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{14}$$

Here the latter part $\exp(-\alpha d_{ij}) A_j$ contains the familiar component of the connectivity (eq. 3), or the rate at which patch $j$ colonizes patch $i$. The first

**Figure 4** Simulated population size.

```
> plot(colSums(occup), type = "l", col = "blue", lwd = 2,
+     xlab = "Time", ylab = "Population Size")
> abline(h = mean(colSums(occup)), col = "red", lty = 2)
```




component, $A_i$, is the expected lifetime of colonized patch assuming that $e = 1$ and $x = 1$ in $E = e/A^x$ (eq. 4). By clever selection of unit of patch size, we can always make $e = 1$, and it makes sense to assume that extinction rate is linearly related to $A$ so that $x = 1$. However, the absolute value of $\lambda_M$ will be dependent on the units of measurement, both for the patch sizes $A_i$ and $A_j$ (which might be different!) and for distance $d$. This means that we cannot say when absolute values of $\lambda_M$ are "large" or "small", but we can compare different patch networks of the same species, or the effect of changes in patch networks.

Metapopulation capacity is almost trivial to compute in R using the old distance matrix `d` and using $\alpha = 1$:

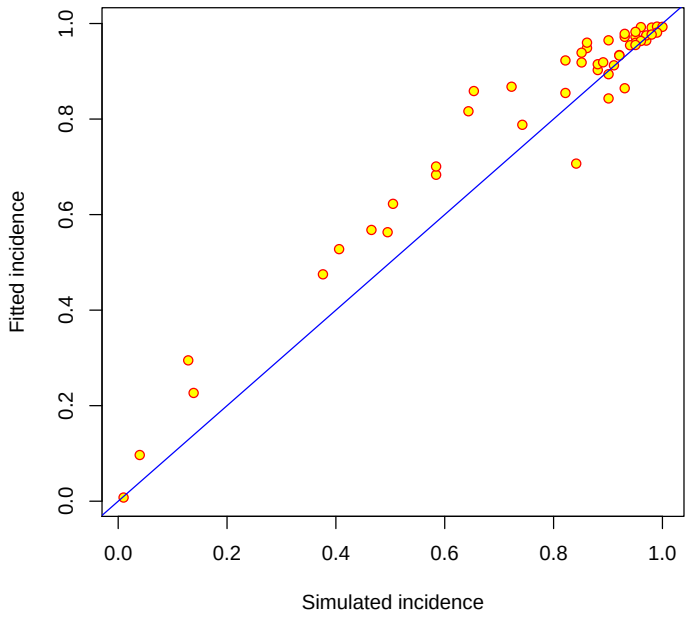
```
> alpha <- 1
> M <- outer(A, A) * as.matrix(exp(-alpha * d))
> tmp <- eigen(M)
```

Function `eigen` finds all eigenvalues, but we need only the first one and the associated squared eigenvector.

```
> lambda.M <- tmp$value[1]
> lambda.vec <- tmp$vector[, 1]^2
```

**Figure 5** Simulated incidences against fitted incidence.

```
> plot(rowMeans(occup), fitted(mod), pch = 21, col = "red",
+     bg = "yellow", xlab = "Simulated incidence", ylab = "Fitted incidence")
> abline(0, 1, col = "blue")
```




The estimated metapopulation capacity is $\lambda_M = 13.07$. However, we have no idea if this is large or small, since changing the units of $A$ would change these values.

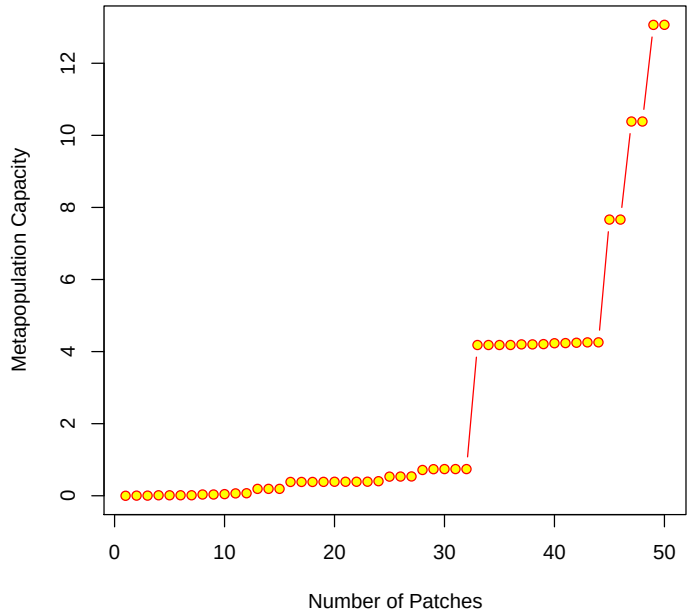
The following code studies the change in metapopulation capacity when we remove patches in random order. The data are (or may be) ordered, so we have to use `sample` to order them randomly, but we will use the same ordering in all analyses, and we can reuse matrix `M` calculated previously. The resulting capacity profile is shown in Fig 6.

```
> N <- length(A)
> take <- sample(N)
> tmp <- M[take, take]
> cap <- numeric(N)
> for (i in 1:N) cap[i] <- eigen(tmp[i:N, i:N])$value[1]
```

The analysis (Fig 6) is based only one random ordering, and different runs would yield different capacity profiles. Metapopulation capacity seems to drop at abrupt steps: The removal of an important patch decreases capcity sharply, but many other patches hardly influence the capacity. The critical capacity for survival can be assessed with simulation.

**Figure 6** Metapopulation capacity as a function of number of randomly selected patches.

```
> plot(N:1, cap, xlab = "Number of Patches", ylab = "Metapopulation Capacity",
+      type = "b", col = "red", pch = 21, bg = "yellow")
```




The `eigen` function produces normalized eigenvectors in R. This means that the sum of squared values is unity for each eigenvector. We saved above these squared values in vector `lambda.vec`, and these give directly the proportional contribution of each patch to the total capacity:

```
> round(lambda.vec, 3)

 [1] 0.380 0.177 0.157 0.173 0.005 0.014 0.011 0.066 0.004 0.000
[11] 0.001 0.000 0.000 0.001 0.002 0.002 0.000 0.001 0.002 0.000
[21] 0.002 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
[31] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
[41] 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
```

The contributions distribute very unevenly. The most important patch contributes 38 % to the total capacity and five most important patches contribute 95.3 % of the total capacity. The metapopulation capacities of the network of five best patches and all other patches are:

```
> take <- rev(order(lambda.vec))
> eigen(M[take[1:5], take[1:5]])$value[1]
```

```
[1] 12.452
```

```
> eigen(M[take[6:N], take[6:N]])$value[1]
```

```
[1] 0.68921
```

Figure 7 continues the simulation of 5 separately for the network of five best and the remaining 45 poorer patches. The steps in 7 look complicated, but they just divide the original data into two subsets, and repeat the analysis of Fig. 5 for each.

# 6 About this document

This document was created automatically using the `Sweave` tool in R. The basic document was formatted in LaTeX, but the original document contained only R input, and the output and graphics were added when processing the source file with `Sweave` (this also implies that all code was tested for syntactical correctness). If you change or replace the source file `fritty.rda`, output will be adjusted to the new data (however, I have not yet debugged the source file so that some discrepancies may be left). In addition, many analyses were based on simulation, and their output will be different each time you generate a new version of the document.

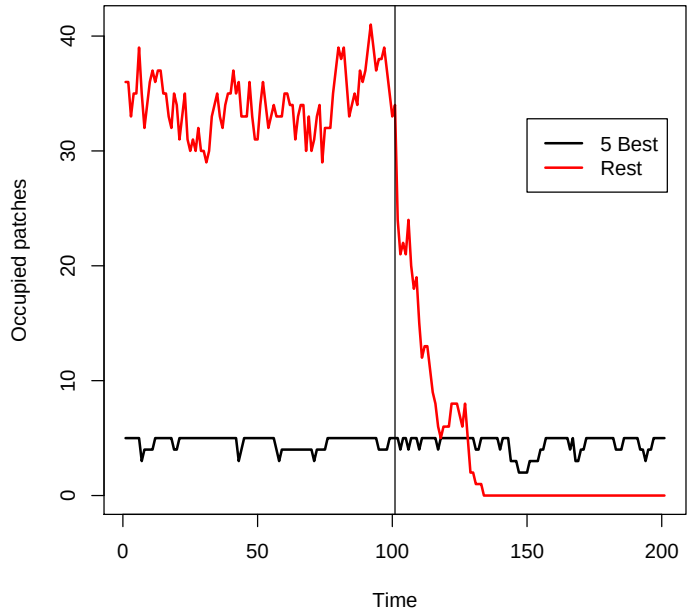
# A Appendix: Data set

```
> fritty
```

```
      x.crd   y.crd      A p p2
1  2.756198 4.33625 4.6000 1  1
2  2.063587 3.85860 2.7000 1  1
3  2.876978 4.70939 1.8000 1  1
4  2.481195 4.21329 1.6000 1  1
5  4.220646 5.94800 1.5000 1  1
6  3.662137 4.94735 0.9000 1  0
7  1.626086 3.25092 0.9000 1  1
8  2.644281 4.43233 0.8300 1  1
9  3.320014 2.74123 0.8000 1  1
10 1.236315 1.76146 0.5000 0  0
11 3.391192 2.41816 0.4800 1  1
12 2.455908 1.97514 0.4000 1  1
13 2.503299 1.72068 0.3300 1  1
14 2.157133 2.47000 0.3000 1  1
15 1.592676 3.91793 0.3000 1  1
16 1.811007 3.75921 0.2500 1  1
17 0.020215 2.41252 0.2500 0  0
18 3.232148 5.67419 0.2400 1  1
19 1.956237 3.92153 0.1900 1  1
20 0.904536 0.81989 0.1800 1  1
21 2.311231 3.88864 0.1600 1  1
```

**Figure 7** Simulation results separately for the sites with highest metapopulation capacity ("5 Best") and other 45 sites ("Rest"). For 100 first time steps, the simulation is identical to that in Fig. 5, but is displayed separately for the subset of patches, but after that (vertical line), the simulation was continued separately for the subsets.

```
> best <- matrix(0, nrow = 5, ncol = 101)
> rest <- matrix(0, N - 5, ncol = 101)
> best[, 1] <- occup[take[1:5], 101]
> rest[, 1] <- occup[take[6:N], 101]
> i <- take[1:5]
> for (t in 1:100) best[, t + 1] <- metastep(best[, t],
+     edis[i, i], E[i], par[3])
> i <- take[6:N]
> for (t in 1:100) rest[, t + 1] <- metastep(rest[, t],
+     edis[i, i], E[i], par[3])
> bestline <- c(colSums(occup[1:5, ]), colSums(best[,
+     -1]))
> restline <- c(colSums(occup[6:N, ]), colSums(rest[,
+     -1]))
> matplot(1:201, cbind(bestline, restline), xlab = "Time",
+     ylab = "Occupied patches", type = "l", lwd = 2,
+     lty = 1)
> abline(v = 101)
> legend(150, 0.8 * max(restline), c("5 Best", "Rest"),
+     lty = 1, col = 1:2, lwd = 2)
```



```
22 3.223386 2.97488 0.1500 0  0
23 2.493993 1.93321 0.1400 1  1
24 1.640751 3.73955 0.1300 1  1
25 1.289309 1.85776 0.1200 1  1
26 2.105513 2.58391 0.1100 1  1
27 1.886316 2.20186 0.1000 1  1
28 1.989523 3.46758 0.1000 1  0
29 2.960682 4.69625 0.0800 1  1
30 0.172328 0.49156 0.0750 0  0
31 2.851745 5.01974 0.0660 1  1
32 2.843055 4.54838 0.0600 1  1
33 0.837135 2.33413 0.0480 1  0
34 1.470259 1.28972 0.0450 0  0
35 3.690926 4.55332 0.0430 1  1
36 4.685348 4.20734 0.0400 0  0
37 2.953328 1.83039 0.0300 1  1
38 2.169728 3.94790 0.0300 1  1
39 3.006825 2.66304 0.0230 1  1
40 2.424058 4.60411 0.0200 1  1
41 2.555626 4.67892 0.0200 1  1
42 2.575048 4.45705 0.0150 1  1
43 2.490998 1.64436 0.0100 1  0
44 1.146973 1.16018 0.0100 0  0
45 2.431145 4.84117 0.0100 1  1
46 1.736047 3.55126 0.0100 1  1
47 2.587934 4.65960 0.0040 1  1
48 3.242839 5.03358 0.0020 1  0
49 0.102708 0.09210 0.0012 0  0
50 1.198278 3.37319 0.0012 0  1
```

# References


Hanski, I. 1999. Metapopulation Ecology. Oxford UP.

Hanski, I., M. Kuussaari, and M. Nieminen. 1994. Metapopulation structure and migration in buttefly *Melitaea cinxia*. Ecology **75**:747–762.

Hanski, I. and O. Ovaskainen. 2000. The metapopulation capacity of a fragmented landscape. Nature **404**:755–758.

McCullagh, P. and J. A. Nelder. 1989. Generalized linear models. 2nd edition, Chapman and Hall.

Venables, B. and B. D. Ripley. 2002. Modern applied statistics with S. 4th edition, Springer.