NTNU – Trondheim
Norwegian University of
Science and Technology

Department of (name): Department of Mathematical Sciences

Examination paper for (course code) (course title)

ST2304 Statistical modelling for biologists and biotechnologists

Academic contact during examination: Jarle Tufto

Phone: 99705519

Examination date: May 15, 2015

Examination time (from-to): 9-13

Permitted examination support material: One handwritten yellow A4 paper, pocket calculator, “Tabeller og formler i statistikk” (Tapir forlag), K. Rottmann: Matematisk formelsamling.

Other information:

Language: English

Number of pages (front page excluded): 7

Number of pages enclosed: 0

Checked by:

_____

Date Signature

Norwegian University
of Science and Technology
Department of Mathematical Sciences

Help pages for some R functions you may need to use follow on page 6.

Problem 1 Suppose that we position a large number of insect traps along a linear transect with the objective of finding a rare insect species. Let X_i be the number of individuals of the species observed in the i'th trap after 24 hours and assume that each X_i is independent Poisson distributed with expected value equal to 0.002.

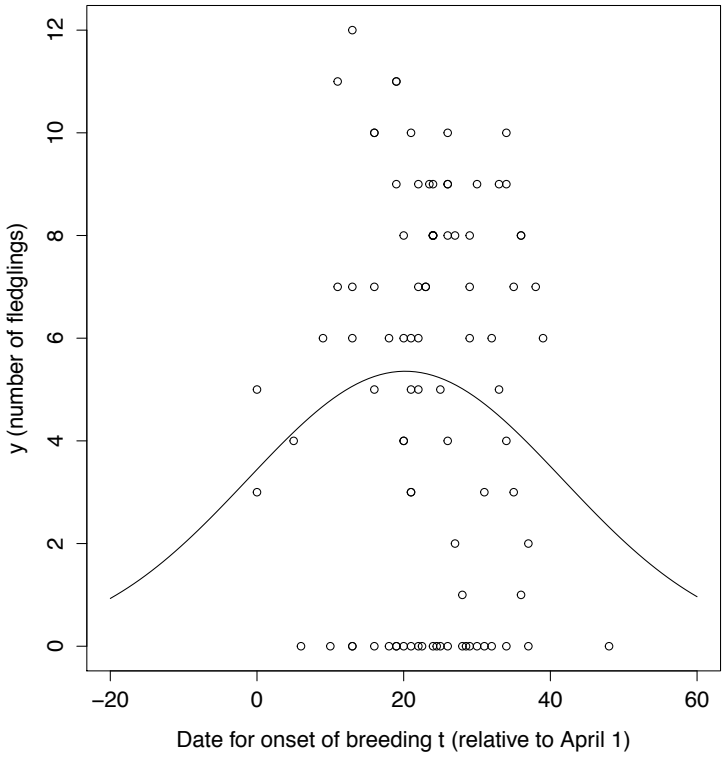
a) Explain briefly why the Poisson assumption may be reasonable and write an R expression that computes the probability that at least one individual of the species is trapped in a given trap. Write the expression such that the computed value is assigned to the variable `p.present`.

Let Y denote the number of traps that must be examined before we find a trap where the species is present.

b) What is the probability distribution of Y if we assume that the number of traps is unlimited? Write R expressions that computes the probability that Y takes a value between 18.2 and 30.5 and the probability that Y takes a value of exactly 20.

c) Write an R-expression that computes a 95%-probability interval for Y, that is, the lower and upper 2.5% quantiles of Y.

Problem 2 We wish to examine how the onset of breeding in great tits influence the number of fledgelings leaving the nest. Great tits lay at most two broods during each breeding season. We therefor register the onset of breeding t (number of days after April 1) and the number of fledglings produced y for altogether 92 breeding pairs of great tits. The observed data are as shown in the following figure.



We then fit a generalized linear model to the data as follows.

```
> t2 <- t^2
> fit <- glm(y ~ t + t2, family=poisson(link="log"))
> summary(fit)

Call:
glm(formula = y ~ t + t2, family = poisson(link = "log"))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2729  -2.8407   0.2778   1.1982   2.6146

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.2361689  0.2547556   4.852 1.22e-06 ***
```

```
t            0.0437550  0.0222333   1.968   0.0491 *
t2          -0.0010827  0.0004782  -2.264   0.0236 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 360.23  on 91  degrees of freedom
Residual deviance: 353.81  on 89  degrees of freedom
AIC: 606.07

Number of Fisher Scoring iterations: 5
```

a) Let β_0, β_1 og β_2 denote the parameters of the model (with β_2 representing the regression coefficient for t^2). Write down the model in mathematical notation. Also write down the expected number of fledglings produced, $E(Y)$, as a function of onset of breeding t.

b) $E(Y)$ as a function of t is plotted in the above figure. If we reparameterize the model, this relationship can be written as

$$E(Y) = y_0 e^{-\frac{1}{2}\left(\frac{t-t_0}{\omega}\right)^2}, \tag{1}$$

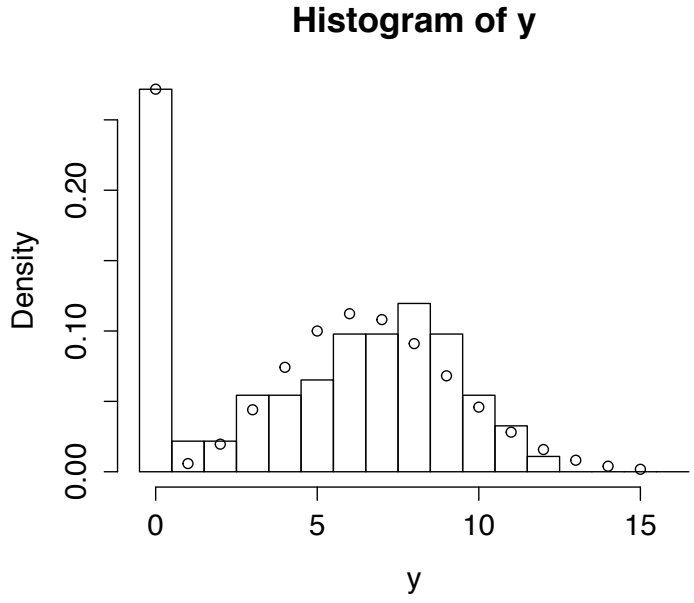
that is, a Gaussian curve (see figure) where t_0 is the optimal onset of breeding, y_0 is the expected number of fledlings given onset of breeding $t = t_0$, and ω is the "width" of the Gauss curve (number of days) (analogous to the standard deviation of a normally distributed variable). Show that the relationship between ω and β_2 in the above regression model is

$$\omega = \sqrt{-\frac{1}{2\beta_2}}. \tag{2}$$

c) Compute an estimate of ω and the associated standard error of this estimate.

d) Test if the data is overdispersed. If necessary, round the number of degrees of freedom to the nearest multiplum of 10. Based on the test result, can we trust the standard error computed in problem 2c? Suppose that we estimate the variance of Y to be inflated by a factor of 2.8 relative to the variance expected under the Poisson assumption. Use this to compute corrected standard errors for $\hat{\beta}_2$ and $\hat{\omega}$. Discuss briefly possible mechanisms that can generate overdispersion in this particular case.

I the remaining part of this problem we shall, in more detail, study the distribution of the variable y described above (the number of fledglings produced by different breeding pairs during the one breeding season). A histogram of the observed distribution of Y (represented by the bars in the histogram) is given below.



The histogram suggests that the probability that Y takes a value of zero is greater than expected from the Poisson assumption, a phenomena know as zero-inflation. To model this, we fit a model H_1 in which

$$P(Y = y) = p_0 I_0(y) + (1 - p_0)\frac{e^{-\lambda}\lambda^y}{y!}, \tag{3}$$

where $I_0(y)$ is a function of y taking a value of 1 for $y = 0$ and a value zero for $y \neq 0$. This model then represents zero-inflation occuring with probability p_0 (complete breeding failure for example as a result of the nestbox being taken over by another species such as a pied flycatcher). Conditional on zero-inflation occuring, Y takes a value of zero with probability one. Conditional on zero-inflation not occuring, Y follows a Poisson distribution with parameter λ.

Vi estimate p_0 and λ by programming the negative log-likelihood function for the model described above as a function named `lnL`. We then fit the model numerically as follows in R. The fitted model is represented by the circles in the above figure.

```
> fit <- optim(c(.2,5),lnL,y=y,hessian=TRUE)
> fit
$par
[1] 0.2709197 6.7380696

$value
[1] 213.6592

$counts
function gradient
      57       NA

$convergence
[1] 0

$message
NULL

$hessian
            [,1]       [,2]
[1,] 463.7016481 -0.4010448
[2,]  -0.4010448  9.8763866

> solve(fit$hessian)
             [,1]         [,2]
[1,] 2.156635e-03 8.757323e-05
[2,] 8.757323e-05 1.012552e-01
```

e) What is the maximum likelihood estimates of p_0 and λ and their approximate standard errors?

f) Write down a mathematical expression for the log likelihood function for the model without zero-inflation, H_0, and compute the maximum log likelihood under this model. You do not need to derive the MLE of λ if you remember this. The following sample statistics are given: $n = 92$, $\sum_{i=1}^{n} y_i = 452$, and $\sum_{i=1}^{n} \ln(y_i!) = 452$. Can you reject H_0 in favour of H_1?

g) Program the function `lnL` that we have used above to fit model H_1. This can be done for example by using conditonal element selection, see the helppage of `ifelse`, or by using logical vectors as indices. It may also be adviseable to use the `dpois` function.

```
Poisson                 package:stats                 R Documentation

The Poisson Distribution

Description:

     Density, distribution function, quantile function and random
     generation for the Poisson distribution with parameter 'lambda'.

Usage:

     dpois(x, lambda, log = FALSE)
     ppois(q, lambda, lower.tail = TRUE, log.p = FALSE)
     qpois(p, lambda, lower.tail = TRUE, log.p = FALSE)
     rpois(n, lambda)

Arguments:

       x: vector of (non-negative integer) quantiles.

       q: vector of quantiles.

       p: vector of probabilities.

       n: number of random values to return.

  lambda: vector of (non-negative) means.

log, log.p: logical; if TRUE, probabilities p are given as log(p).

lower.tail: logical; if TRUE (default), probabilities are P[X <= x],
          otherwise, P[X > x].

Details:

     The Poisson distribution has density

                       p(x) = lambda^x exp(-lambda)/x!

     for x = 0, 1, 2, ... .  The mean and variance are E(X) = Var(X) =
     lambda.

     If an element of 'x' is not integer, the result of 'dpois' is
     zero, with a warning.  p(x) is computed using Loader's algorithm,
     see the reference in 'dbinom'.

     The quantile is right continuous: 'qpois(p, lambda)' is the
     smallest integer x such that P(X <= x) >= p.

     Setting 'lower.tail = FALSE' allows to get much more precise
     results when the default, 'lower.tail = TRUE' would return 1, see
     the example below.

Value:

     'dpois' gives the (log) density, 'ppois' gives the (log)
     distribution function, 'qpois' gives the quantile function, and
     'rpois' generates random deviates.

     Invalid 'lambda' will result in return value 'NaN', with a
     warning.

     The length of the result is determined by 'n' for 'rpois', and is
     the maximum of the lengths of the numerical arguments for the
     other functions.

     The numerical arguments other than 'n' are recycled to the length
     of the result.  Only the first elements of the logical arguments
     are used.

Source:

     'dpois' uses C code contributed by Catherine Loader (see
     'dbinom').

     'ppois' uses 'pgamma'.

     'qpois' uses the Cornish-Fisher Expansion to include a skewness
     correction to a normal approximation, followed by a search.

     'rpois' uses

     Ahrens, J. H. and Dieter, U. (1982).  Computer generation of
     Poisson deviates from modified normal distributions.  _ACM
     Transactions on Mathematical Software_, *8*, 163-179.

See Also:

     Distributions for other standard distributions, including 'dbinom'
     for the binomial and 'dnbinom' for the negative binomial
     distribution.

     'poisson.test'.

Examples:

     require(graphics)

     -log(dpois(0:7, lambda = 1) * gamma(1+ 0:7)) # == 1
     Ni <- rpois(50, lambda = 4); table(factor(Ni, 0:max(Ni)))

     1 - ppois(10*(15:25), lambda = 100)  # becomes 0 (cancellation)
         ppois(10*(15:25), lambda = 100, lower.tail = FALSE)  # no cancellation

     par(mfrow = c(2, 1))
     x <- seq(-0.01, 5, 0.01)
     plot(x, ppois(x, 1), type = "s", ylab = "F(x)", main = "Poisson(1) CDF")
     plot(x, pbinom(x, 100, 0.01), type = "s", ylab = "F(x)",
          main = "Binomial(100, 0.01) CDF")
--------------------------------------------------------------------------------
Geometric               package:stats                 R Documentation

The Geometric Distribution

Description:

     Density, distribution function, quantile function and random
     generation for the geometric distribution with parameter 'prob'.

Usage:

     dgeom(x, prob, log = FALSE)
     pgeom(q, prob, lower.tail = TRUE, log.p = FALSE)
     qgeom(p, prob, lower.tail = TRUE, log.p = FALSE)
     rgeom(n, prob)

Arguments:

    x, q: vector of quantiles representing the number of failures in a
          sequence of Bernoulli trials before success occurs.

       p: vector of probabilities.

       n: number of observations. If 'length(n) > 1', the length is
          taken to be the number required.

    prob: probability of success in each trial. '0 < prob <= 1'.

log, log.p: logical; if TRUE, probabilities p are given as log(p).

lower.tail: logical; if TRUE (default), probabilities are P[X <= x],
          otherwise, P[X > x].

Details:

     The geometric distribution with 'prob' = p has density

                              p(x) = p (1-p)^x

     for x = 0, 1, 2, ..., 0 < p <= 1.

     If an element of 'x' is not integer, the result of 'dgeom' is
     zero, with a warning.

     The quantile is defined as the smallest value x such that F(x) >=
     p, where F is the distribution function.

Value:

     'dgeom' gives the density, 'pgeom' gives the distribution
     function, 'qgeom' gives the quantile function, and 'rgeom'
     generates random deviates.

     Invalid 'prob' will result in return value 'NaN', with a warning.

     The length of the result is determined by 'n' for 'rgeom', and is
```

```
     the maximum of the lengths of the numerical arguments for the
     other functions.

     The numerical arguments other than 'n' are recycled to the length
     of the result.  Only the first elements of the logical arguments
     are used.

Source:

     'dgeom' computes via 'dbinom', using code contributed by Catherine
     Loader (see 'dbinom').

     'pgeom' and 'qgeom' are based on the closed-form formulae.

     'rgeom' uses the derivation as an exponential mixture of Poissons,
     see

     Devroye, L. (1986) _Non-Uniform Random Variate Generation._
     Springer-Verlag, New York. Page 480.

See Also:

     Distributions for other standard distributions, including
     'dnbinom' for the negative binomial which generalizes the
     geometric distribution.

Examples:

     qgeom((1:9)/10, prob = .2)
     Ni <- rgeom(20, prob = 1/4); table(factor(Ni, 0:max(Ni)))
--------------------------------------------------------------------------
ifelse                  package:base                   R Documentation

Conditional Element Selection

Description:

     'ifelse' returns a value with the same shape as 'test' which is
     filled with elements selected from either 'yes' or 'no' depending
     on whether the element of 'test' is 'TRUE' or 'FALSE'.

Usage:

     ifelse(test, yes, no)

Arguments:

    test: an object which can be coerced to logical mode.

     yes: return values for true elements of 'test'.

      no: return values for false elements of 'test'.

Details:

     If 'yes' or 'no' are too short, their elements are recycled.
     'yes' will be evaluated if and only if any element of 'test' is
     true, and analogously for 'no'.

     Missing values in 'test' give missing values in the result.

Value:

     A vector of the same length and attributes (including dimensions
     and '"class"') as 'test' and data values from the values of 'yes'
     or 'no'.  The mode of the answer will be coerced from logical to
     accommodate first any values taken from 'yes' and then any values
     taken from 'no'.

Warning:

     The mode of the result may depend on the value of 'test' (see the
     examples), and the class attribute (see 'oldClass') of the result
     is taken from 'test' and may be inappropriate for the values
     selected from 'yes' and 'no'.

     Sometimes it is better to use a construction such as

       (tmp <- yes; tmp[!test] <- no[!test]; tmp)
     , possibly extended to handle missing values in 'test'.

References:

     Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S
     Language_.  Wadsworth & Brooks/Cole.

See Also:

     'if'.

Examples:

     x <- c(6:-4)
     sqrt(x)  #- gives warning
     sqrt(ifelse(x >= 0, x, NA))  # no warning

     ## Note: the following also gives the warning !
     ifelse(x >= 0, sqrt(x), NA)

     ## example of different return modes:
     yes <- 1:3
     no <- pi^(0:3)
     typeof(ifelse(NA, yes, no))    # logical
     typeof(ifelse(TRUE, yes, no))  # integer
     typeof(ifelse(FALSE, yes, no)) # double
```