Project 1 - TMA4315

martin.o.berild@ntnu.no
10014

yaolin.ge@ntnu.no
10026

October 1, 2020

Exercise 1

a) The Poisson distribution is denoted

$$f(y) = \frac{\lambda^y}{y!} e^{-\lambda},$$

where y is the response of the model and λ is the exected value of y. Furthermore, we apply a canonical link between the linear predictor η, holding the M fixed effects $\boldsymbol{\beta}$ of the covariates $\mathbf{x}_p$ for $0 < p \leq M$, and the expected value of y. For the Poisson likelihood the canonical link is the log-link function as

$$\eta_i = \ln(\lambda_i) = \mathbf{x}_i^T \boldsymbol{\beta} \quad \text{or} \quad \mathbb{E}[y_i] = \lambda_i = \exp(\eta_i) = \exp(\mathbf{x}_i^T \boldsymbol{\beta}).$$

Assuming that the observation of the response y_i are independent; then, the likelihood can be written as

$$L(\beta) = \prod_{i=1}^n L_i(\beta) = \prod_{i=1}^n f(y_i; \beta) = \prod_{i=1}^n \frac{\lambda_i^{y_i}}{y_i!} \exp(-\lambda_i),$$

and thereby the log-likelihood is given by

$$\begin{aligned} \ell(\boldsymbol{\beta}) = \ln L(\boldsymbol{\beta}) &= \sum_{i=1}^n \ell_i(\boldsymbol{\beta}) = \sum_{i=1}^n [y_i \ln(\lambda_i) - \lambda_i - \ln(y_i!)] \\ &\overset{\eta_i = \ln(\lambda_i)}{=} \sum_{i=1}^n [y_i \eta_i - \exp(\eta_i) + C_i] \overset{\eta_i = \mathbf{x}_i^T \boldsymbol{\beta}}{=} \sum_{i=1}^n y_i \mathbf{x}_i^T \boldsymbol{\beta} - \sum_{i=1}^n \exp(\mathbf{x}_i^T \boldsymbol{\beta}) + C. \end{aligned}$$

The goal is now to maximize the log-likelihood by finding the optimal values of $\boldsymbol{\beta}$. This is achieved by differentiating the log-likelihood with respect to $\boldsymbol{\beta}$; then, set it equivalent to zero and solve it numerically or if possible analytically. The derivative of the log-likelihood is generally referred to as the *score function* and is written as

$$\begin{aligned} s(\boldsymbol{\beta}) = \frac{\partial \ell(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} &= \sum_{i=1}^n \frac{\partial \ell_i(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \sum_{i=1}^n \frac{\partial \ell_i(\boldsymbol{\beta})}{\partial \eta_i} \cdot \frac{\partial \eta_i}{\partial \boldsymbol{\beta}} = \sum_{i=1}^n \frac{\partial [y_i \eta_i - \exp(\eta_i) + C_i]}{\partial \eta_i} \cdot \frac{\partial [\mathbf{x}_i^T \boldsymbol{\beta}]}{\partial \boldsymbol{\beta}} \\ &= \sum_{i=1}^n [y_i - \exp(\eta_i)] \cdot \mathbf{x}_i = \sum_{i=1}^n (y_i - \lambda_i) \mathbf{x}_i. \end{aligned}$$

The expected value of the second derivative of the log-likelihood with respect to $\boldsymbol{\beta}$ is the Fisher information, Fisher matrix or expected information matrix, and can be simplified to

$$
\begin{aligned}
\mathbf{F}(\boldsymbol{\beta}) = \mathrm{Cov}[s(\boldsymbol{\beta})] = \sum_{i=1}^{n} \mathrm{Cov}[s_i(\boldsymbol{\beta})] &= \sum_{i=1}^{n} \mathbb{E}\left[(s_i(\boldsymbol{\beta}) - \mathbb{E}[s_i(\boldsymbol{\beta})]) (s_i(\boldsymbol{\beta}) - \mathbb{E}[s_i(\boldsymbol{\beta})])^T \right] \\
\overset{\mathbb{E}[s_i(\boldsymbol{\beta})]=0}{=} \sum_{i=1}^{n} \mathbb{E}\left[s_i(\boldsymbol{\beta}) s_i(\boldsymbol{\beta})^T \right] &\overset{s_i(\boldsymbol{\beta})=(Y_i-\lambda_i)\mathbf{x}_i}{=} \sum_{i=1}^{n} \mathbb{E}\left[(Y_i - \lambda_i)\mathbf{x}_i (Y_i - \lambda_i)\mathbf{x}_i^T \right] \\
= \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \mathbb{E}\left[(Y_i - \lambda_i)^2 \right] &\overset{\mathbb{E}\left[(Y_i-\lambda_i)^2\right]=\mathrm{Cov}[Y_i]=\lambda_i}{=} \sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i^T \lambda_i.
\end{aligned}
$$

b) We will now use these equations to create a R function that iteratively calculates the maximum likelihood (ML) estimates of a particular likelihood function. More spefically, the Fisher scoring algorithm for $t \geq 0$ and some starting value $\boldsymbol{\beta}^{(0)}$:

$$\boldsymbol{\beta}^{(t+1)} = \boldsymbol{\beta}^{(t)} + \mathbf{F}^{-1}(\boldsymbol{\beta}^{(t)}) \cdot S(\boldsymbol{\beta}^{(t)}),$$

is run until the convergence criterion

$$\frac{||\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}||}{||\boldsymbol{\beta}^{(t)}||} \leq \epsilon,$$

for some given $\epsilon > 0$ obtaining the ML estimates $\hat{\boldsymbol{\beta}}$. In addition, we are interested in the deviance of the model using the ML estimates. The deviance is a discrepancy measure between observed and fitted values, and is derived as

$$D = 2 \sum_{i=1}^{n} \left[\ell(\hat{\boldsymbol{\beta}}) - \ell_s \right],$$

where ℓ_s is the log-likelihood of the saturated model.

The function `myglm()` takes four input arguments `formula` (which covariates model the response), the `data`, the `start` ($\boldsymbol{\beta}^{(0)}$), and the ϵ convergence criterion. The ouput of `myglm()` is the ML estimates of the coefficient, the corresponding deviance with the staturated model and covariance matrix (Fisher information matrix $\mathbf{F}(\cdot)$). There are some comments in the code specifying the functionality of that particular line. In addition, we have made the methods `summary()`, `print()`, and `vcov()` for the class `myglm`.

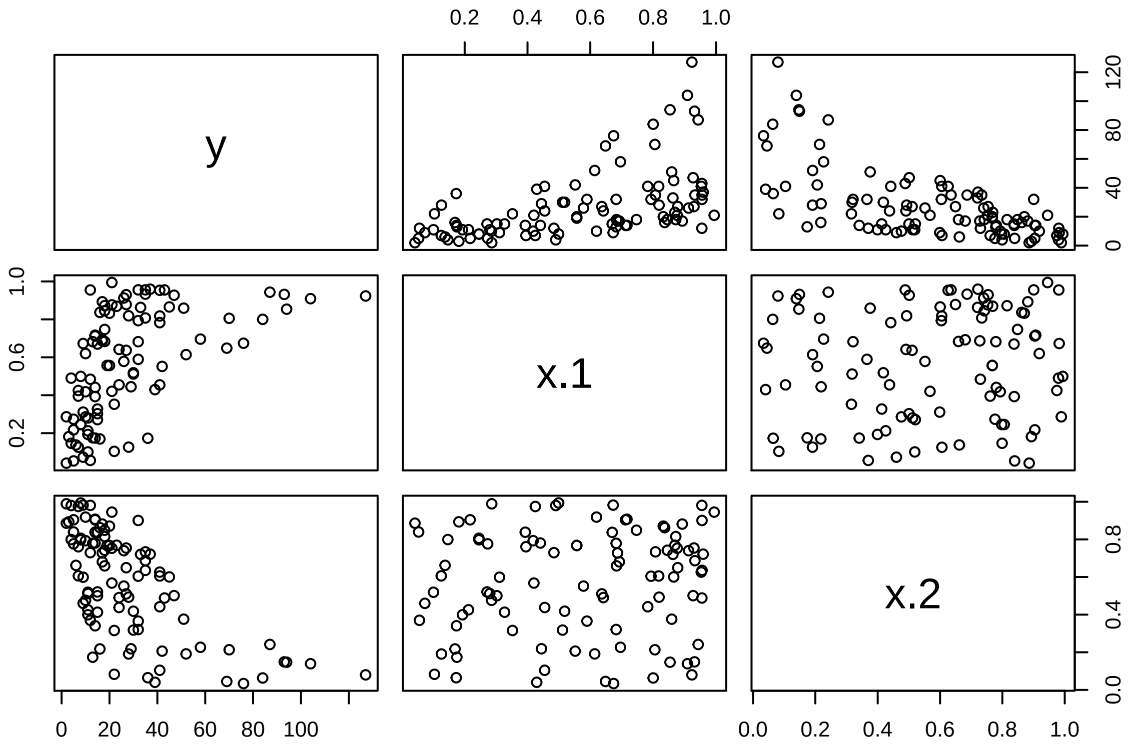
```r
myglm <- function(formula,data,start=NA,epsilon=0.01,null=TRUE){
  mf = model.frame(formula = formula, data = data)
  # splitting up covariates and response
  X = model.matrix(attr(mf, "terms"), data = mf)
  y = model.response(mf)

  # if start is not specified input
  if (anyNA(start)){
    beta = numeric(ncol(X))
  }
  new_beta = beta
  crit = 100
  count = 0
  # convergence criterion
  while (epsilon < crit){
    Sb = t(X)%*%(y-exp(X%*%beta)) # score function
    Fb = t(X)%*%as.matrix(as.data.frame(X)*exp(X%*%beta)) # fisher information
    iFb = solve(Fb) # inverse fisher information
    new_beta = beta + iFb%*%Sb # Fisher scoring iteration

    # calculating the convergence criterion
    crit = norm(as.matrix(new_beta - beta),type="2")/norm(as.matrix(beta),type="2")
    beta = new_beta
    count = count + 1
  }
  # storing the residuals, fitted values and the model in a list
  res = list(fitted.values = exp(X%*%new_beta),
             model = mf)

  # placing ML estimates in matrix
  res$coefficients = cbind(new_beta,sqrt(diag(iFb)),new_beta/sqrt(diag(iFb)),
                           p_value_glm(new_beta/sqrt(diag(iFb))))
  colnames(res$coefficients) = c("Estimate", "Std. Error","z value","Pr(>|z|))")

  # removing infinite values of log(y)
  i_rem = as.numeric(names(y[!is.infinite(log(y))]))
  # calculating the deviance
  res$residuals =  log(y[i_rem]) - X[i_rem,]%*%beta
  res$logLik = sum(-exp(X%*%beta) + y*(X%*%beta) - log(factorial(y)))
  res$logLik.sat = sum(y[i_rem]*log(y[i_rem])) - sum(y) - sum(log(factorial(y)))
  res$deviance = -2*(res$logLik-res$logLik.sat)
  if (null){
    res$null = myglm(y~1,data,null=FALSE)
  }
  res$aic = 2*length(new_beta) - 2*sum(y*(X%*%new_beta) -
                                       exp(X%*%beta) - log(factorial(y)))

  # assigning variance
  res$vcov = iFb

  # storing the function  call and formula
  res$data = data
  res$call = match.call()
  res$formula = formula
  res$df = length(y) - length(new_beta)
  res$iter = count

  # setting the class to myglm for help functions
  class(res) = "myglm"
  return(res)
}

p_value_glm <- function(z_values){
  return(2*pnorm(-abs(z_values)))
}

# print method for myglm class object
print.myglm <- function(object){
  cat("Call:\t",format(object$call),"\n")
  cat("\nCoefficients:\n")
  print(object$coefficients[,1],digits=4)
  cat("\nDegrees of Freedom:",object$null$df,"Total (i.e. Null);",
      object$df,"Residual\n")
  cat("Null Deviance:        ",object$null$deviance,"\n")
  cat("Residual Deviance:    ",object$deviance,"  AIC: ",object$aic)
```

```r
}

# summary method for myglm class object
summary.myglm <- function(object){
  cat("Call:\n")
  cat(format(object$call),"\n")
  cat("\nDeviance Residuals:\n")
  print(digits = 4,matrix(c(min(object$residuals),
                          as.numeric(quantile(object$residuals,0.25)),
                          median(object$residuals),
               as.numeric(quantile(object$residuals,0.75)),
               max(object$residuals)),
             dimnames = list(c("Min","1Q","Median","3Q","Max"),c()))[,1])
  cat("\nCoefficients:\n")
  print(object$coefficients)
  cat("---\n")
  cat("Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1\n\n")
  cat("(Dispersion parameter for poisson family taken to be 1)\n\n")
  cat("    Null deviance:", object$null$deviance," on ",
      object$null$df,"  degrees of freedom\n")
  cat("Residual deviance:", object$deviance, " on ",
      object$df, "degrees of freedom\n")
  cat("AIC:", object$aic,"\n\n")
  cat("Number of Fisher Scoring iterations: ", object$iter)
}

# a vcov method for myglm class object
vcov.myglm <- function(object){
  print(object$vcov)
}
```

c)

To test the `myglm()`function, we generate $n = 100$ observations of two covariates from a uniform distribution between zero and one $x_p \sim \text{U}[0,1]$ for $p = 1,2$. Furthermore, we use the log-link function on a poisson likelihood function with $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2) = (3, 2, -2)$ to obtain realization of y. A realization is obtained in the code below and a visualization is given in the corresponding figure.

```r
x = matrix(c(runif(100),runif(100)),ncol = 2)
y = rpois(100,lambda = exp(3  + 2*x[,1] - 2*x[,2]))
df = data.frame(y=y,x=x)
plot(df)
```



Then, we fit the model using our custom `myglm()`function:

```
myglm1 = myglm(y~x, data = df)
myglm1 #print
```

```
## Call:     myglm(formula = y ~ x, data = df)
##
## Coefficients:
## (Intercept)           x1           x2
##       3.056        2.008       -2.087
##
## Degrees of Freedom: 99 Total (i.e. Null); 97 Residual
## Null Deviance:          1681.428
## Residual Deviance:      83.7171    AIC:  572.1076
```

and compare it with the estimates obtained from the **R** function `glm()`:

```
glm1 = glm(y~x,family = poisson)
glm1 #print
```

```
##
## Call:  glm(formula = y ~ x, family = poisson)
##
## Coefficients:
## (Intercept)           x1            x2
##       3.056         2.008        -2.087
##
## Degrees of Freedom: 99 Total (i.e. Null);  97 Residual
## Null Deviance:       1681
## Residual Deviance: 83.72     AIC: 572.1
```

Futhermore, we can compare their `summary()` outputs:

```r
summary(myglm1) #summary
```

```
## Call:
## myglm(formula = y ~ x, data = df)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.87317 -0.16469 -0.01941  0.08845  0.54346
##
## Coefficients:
##              Estimate Std. Error   z value     Pr(>|z|))
## (Intercept)  3.055743 0.06279138  48.66501  0.000000e+00
## x1           2.007926 0.07818707  25.68104 1.903623e-145
## x2          -2.086875 0.06744819 -30.94041 3.419692e-210
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1681.428  on  99   degrees of freedom
## Residual deviance: 83.7171  on  97 degrees of freedom
## AIC: 572.1076
##
## Number of Fisher Scoring iterations:  71
```

```r
summary(glm1)
```

```
##
## Call:
## glm(formula = y ~ x, family = poisson)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.90141  -0.70689  -0.08642   0.44171   2.27024
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  3.05573    0.06286   48.61   <2e-16 ***
## x1           2.00791    0.07826   25.66   <2e-16 ***
## x2          -2.08684    0.06754  -30.90   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1681.428  on 99  degrees of freedom
## Residual deviance:   83.717  on 97  degrees of freedom
## AIC: 572.11
##
## Number of Fisher Scoring iterations: 4
```

and the vcov():

```r
vcov(myglm1) #covariance matrix
```

```
##              (Intercept)            x1            x2
## (Intercept)  0.003942758 -0.0041387667 -0.0016255747
## x1          -0.004138767  0.0061132182 -0.0003664619
## x2          -0.001625575 -0.0003664619  0.0045492585
```

```r
vcov(glm1)
```

```
##              (Intercept)            x1            x2
## (Intercept)  0.003950946 -0.0041442593 -0.0016333733
## x1          -0.004144259  0.0061253586 -0.0003680182
## x2          -0.001633373 -0.0003680182  0.0045620178
```

We observe that the output from `myglm()` function is very similar to the output from the *R* function `glm()`.

Exercise 2

First, we load the data:

```r
load(url("https://www.math.ntnu.no/emner/TMA4315/2020h/hoge-veluwe.Rdata"))
head(data)
```

```
##    y  t
## 1  6 16
## 2  7 16
## 3 10 17
## 4  7 16
## 5  9  9
## 6  7 18
```

and observe that it consists of two variables y_i, number of fledlings leaving nest i, and t_i, the date the fledlings left nest i. We want to fit a model assuming that the number of fledlings, y, is Poisson distributed where the expected number of fledlings λ_i at date t_i follow a Gaussian function as

$$\mu_i = \lambda_i = E[Y_i] = \lambda_0 \exp\left\{-\frac{(t_i-\theta)^2}{2\omega^2}\right\} = h(\eta_i) = \exp\{\eta_i\}.$$

a)

The λ_0 parameter can be interpreted as the maximum production rate of fledlings or simply the scale of the expected value. θ sets the date of the maximum production rate of fledlings or can be viewed as a location parameter, and lastly, ω is a shape parameter controlling the width of the Gaussian function and the slope of the bell-curve.

b) Generally the linear predictor η_i in a generalized linear model is an addition of the effects of the covariates β_p. Using the log-link as shown in **Exercise 1**, we obtain the linear predictor:

$$\begin{aligned}\eta_i &= \beta_0 + \beta_1 \cdot t_i + \beta_2 \cdot t_i^2 = \ln(\mu_i) \\ &= \ln(\lambda_i) = \ln(\lambda_0) - \frac{(t_i-\theta)^2}{2\omega^2} \\ &= \ln(\lambda_0) - \frac{1}{2\omega^2}\cdot t_i^2 + \frac{\theta}{\omega^2}\cdot t_i - \frac{\theta^2}{2\omega^2}.\end{aligned}$$

Here, we see that the linear predictor using the specified expectation λ can be expressed as a additive linear predictor that is linked to the mean of the response through the log-link function. The model is therefore a GLM with a Poisson likelihood.

In the above reparametrization, we observe that the relationship between the parameters $(\beta_0, \beta_1, \beta_2)$ and $(\lambda_0, \theta, \omega)$ is

$$\beta_0 = \ln(\lambda_0) - \frac{\theta^2}{2\omega^2} \;\; ; \;\; \beta_1 = \frac{\theta}{\omega^2} \;\; ; \;\; \beta_2 = -\frac{1}{2\omega^2}.$$

Assuming that $(\beta_0, \beta_1, \beta_2)$ are known, we can find the unknown parameters by reformulating the above expression as

$$\omega = \frac{1}{\sqrt{-2\beta_2}} \;\; ; \;\; \theta = -\frac{\beta_1}{2\beta_2} \;\; ; \;\; \lambda_0 = \exp\left\{\beta_0 - \frac{\beta_1^2}{2\beta_2}\right\}.$$

c)

Using $(\beta_0, \beta_1, \beta_2)$ the model can be fit with the `myglm()` function created in **Exercise 1**. The quadratic terms of t is specified in the formula by using `I(t^2)`, s.t. the formula is `y~1 + t + I(t^2)`. The resulting fit of the model is:

```
res = myglm(y~t+I(t^2),data=data)
res
```

```
## Call:     myglm(formula = y ~ t + I(t^2), data = data)
##
## Coefficients:
## (Intercept)           t      I(t^2)
##    1.420157    0.085181   -0.003299
##
## Degrees of Freedom: 134 Total (i.e. Null); 132 Residual
## Null Deviance:         300.1075
## Residual Deviance:     277.4613   AIC:  740.6677
```

d)

To determine if there is evidence of a quadratic effect of t, we can investigate the p-value of the effect of the quadratic term. The p-value is found by computing the significance of the quadratic effect compared with the null model. It is already implemented in the `myglm` class and is extracted using the `summary()` method for the class as

```
summary(res)
```

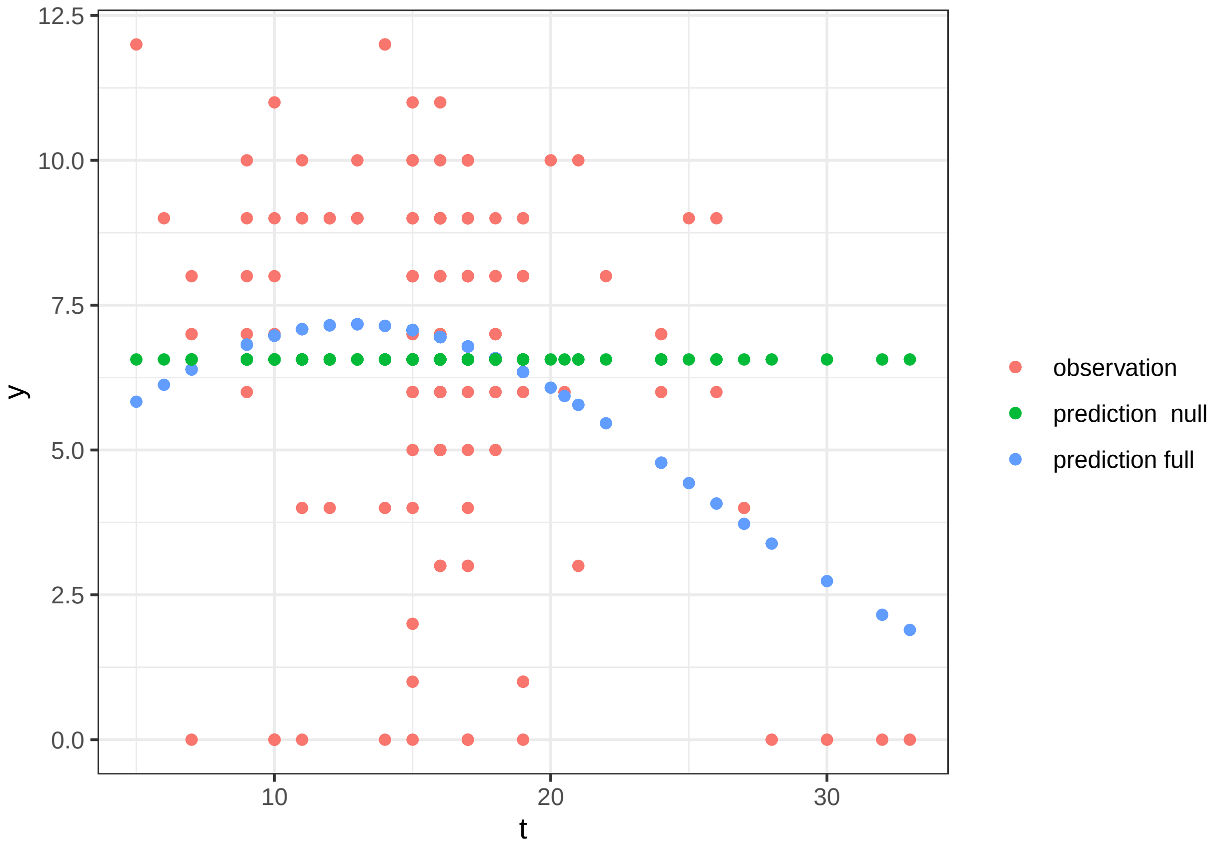
```
## Call:
## myglm(formula = y ~ t + I(t^2), data = data)
##
## Deviance Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9557 -0.1278  0.1408  0.2823  0.7922
##
## Coefficients:
##                 Estimate  Std. Error   z value    Pr(>|z|))
## (Intercept)  1.420156794 0.281865068  5.038428 4.693714e-07
## t            0.085181410 0.034002196  2.505174 1.223913e-02
## I(t^2)      -0.003298583 0.001018477 -3.238742 1.200580e-03
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 300.1075  on  134   degrees of freedom
## Residual deviance: 277.4613  on  132 degrees of freedom
## AIC: 740.6677
##
## Number of Fisher Scoring iterations:  8
```

Here, we observe that the p-value is 0.0012006 which is less than a significant level of for example $\alpha = 0.05$ and thereby the quadratic effect is significant.

e)

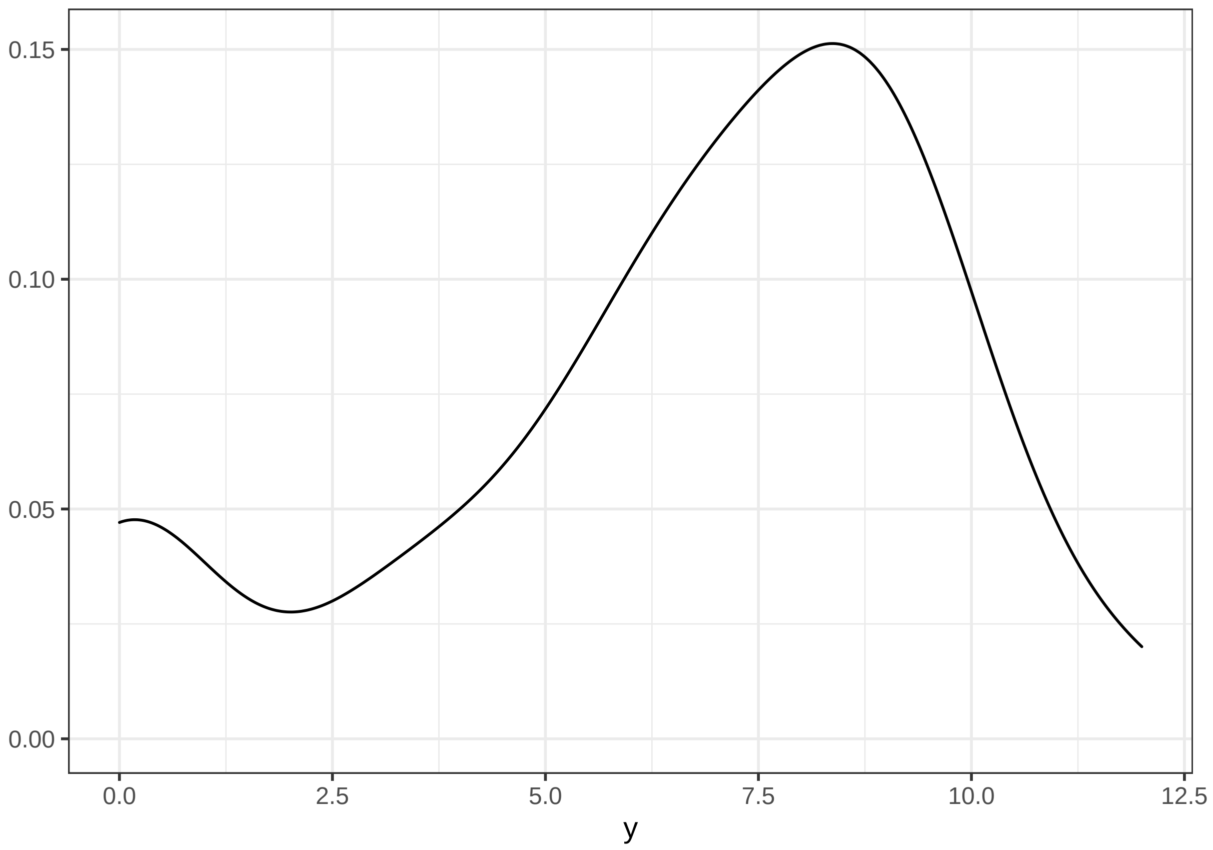
Considering the observed deviance, 277.461337, and the deviance of the null model, 300.107511, in the summary output above there is no large difference in their value. This can also be viewed in the p-values of the different effects, where the intercept is more significant than the other effects. These facts tend towards a poor goodness-of-fit. In the first figure below presenting the ovservation and the predicted values of both the null and our proposed model, we can also observe this poor fit of both the null and fully parametric model.

```
ggplot() +
  geom_point(data = data, aes(x=t,y=y, color = "observation")) +
  geom_point(data = data.frame(t = data$t, y = res$fitted.values),
             aes(x=t,y=y,color = "prediction full")) +
  geom_point(data = data.frame(t = data$t, y = res$null$fitted.values),
             aes(x=t,y=y,color = "prediction  null")) +
  labs(color = "") +
  theme_bw()
```



Lastly, in the figure below showing the density of y we observe a significant bimodality caused by zero-inflation in the data. The poisson model used in this exercise does not allow for such a bimodality and the assumption about the response being Poisson distributed is therefore faulty given our current choice of linear predictor.

```
ggplot() +
  geom_density(data = data.frame(y = data$y), aes(x=y)) +
  labs(y="") +
  theme_bw()
```



f)

In exercise **2b)**, we found the relationship between $\boldsymbol{\beta}$ and $\boldsymbol{Z} = (\lambda_0, \theta, \omega)$ to be $Z_k = f_{Z_k}(\beta_0, \beta_1, \beta_2)$ for the corresponding element $Z_k \in \boldsymbol{Z}$ where $k = 1, 2, 3$. Using this relationship and the ML estimates of $\hat{\boldsymbol{\beta}}$ the ML estimates of $(\hat{\lambda_0}, \hat{\theta}, \hat{\omega})$ are:

```
omega = 1/sqrt(-2*res$coefficients[3,1])
theta = - res$coefficients[2,1] /(2*res$coefficients[3,1])
lambda0 = exp(res$coefficients[1,1] - res$coefficients[2,1]^2/(2*res$coefficients[3,1]))
cat("lambda0 =", lambda0, "\ntheta =",theta, "\nomega =", omega)
```

```
## lambda0 = 12.42864
## theta = 12.91182
## omega = 12.31179
```

Furhtermore, the standard deviations of the estimates of $(\hat{\lambda_0}, \hat{\theta}, \hat{\omega})$ can be found wtih the delta method as

$$\text{Var}(Z_k) = \sum_{i=1}^{M}\sum_{j=1}^{M} \frac{\partial f_{Z_k}}{\partial \beta_i}\frac{\partial f_{Z_k}}{\partial \beta_j}\text{Cov}(\beta_i, \beta_j),$$

where M is the number of fixed effects $\boldsymbol{\beta}$ and $\text{Cov}(\beta_i, \beta_j)$ is the covariance between β_i and β_j such that if $i = j$ we have $\text{Cov}(\beta_i, \beta_i) = \text{Var}(\beta_i)$.

For $\hat{\theta}$ the variance can be calculated using the delta method as

$$\begin{aligned}
\text{Var}(\theta) &= \left(\frac{\partial\theta}{\partial\beta_1}\right)^2 \text{Var}(\beta_1) + \left(\frac{\partial\theta}{\partial\beta_2}\right)^2 \text{Var}(\beta_2) + 2\frac{\partial\theta}{\partial\beta_1}\frac{\partial\theta}{\partial\beta_2}\text{Cov}(\beta_1,\beta_2) \\
&= \left(-\frac{1}{2\beta_2}\right)^2 \text{Var}(\beta_1) + \left(\frac{\beta_1}{2\beta_2^2}\right)^2 \text{Var}(\beta_2) + 2\frac{-1}{2\beta_2}\frac{\beta_1}{2\beta_2^2}\text{Cov}(\beta_1,\beta_2) \\
&= \frac{1}{4\beta_2^2}\text{Var}(\beta_1) + \frac{\beta_1^2}{4\beta_2^4}\text{Var}(\beta_2) - \frac{\beta_1}{2\beta_2^3}\text{Cov}(\beta_1,\beta_2),
\end{aligned}$$

and using the ML estimates of $\boldsymbol{\beta}$ we get:

```
sd_theta = sqrt(1/(4*res$coefficients[3,1]^2)*res$vcov[2,2] +
                res$coefficients[2,1]^2/(4*res$coefficients[3,1]^4)*res$vcov[3,3] -
                res$coefficients[2,1]/(2*res$coefficients[3,1]^3)*res$vcov[2,3])
sd_theta
```

```
## [1] 1.605586
```

Next for $\hat{\omega}$ which is simply dependent on just β_2, we get:

$$\text{Var}(\omega) = \left(\frac{\partial\omega}{\partial\beta_2}\right)^2 \text{Var}(\beta_2) = \left(\frac{1}{2\sqrt{2}(-\beta_2)^{\frac{3}{2}}}\right)^2 \text{Var}(\beta_2) = \frac{1}{8\cdot(-\beta_2)^3}\text{Var}(\beta_2),$$

resulting in the standard deviation of

```
sd_omega = sqrt(1/(8*(-res$coefficients[3,1])^(3))*res$vcov[3,3])
sd_omega
```

```
## [1] 1.900706
```

Lastly, the variance of $\hat{\lambda_0}$ is calcuated by

$$\begin{aligned}
\text{Var}(\lambda_0) = &\left(\frac{\partial\lambda_0}{\partial\beta_0}\right)^2 \text{Var}(\beta_0) + \left(\frac{\partial\lambda_0}{\partial\beta_1}\right)^2 \text{Var}(\beta_1) + \left(\frac{\partial\lambda_0}{\partial\beta_2}\right)^2 \text{Var}(\beta_2) + \ldots \\
&2\frac{\partial\lambda_0}{\partial\beta_0}\frac{\partial\lambda_0}{\partial\beta_1}\text{Cov}(\beta_0,\beta_1) + 2\frac{\partial\lambda_0}{\partial\beta_0}\frac{\partial\lambda_0}{\partial\beta_2}\text{Cov}(\beta_0,\beta_2) + 2\frac{\partial\lambda_0}{\partial\beta_1}\frac{\partial\lambda_0}{\partial\beta_2}\text{Cov}(\beta_1,\beta_2) \\
= &\exp\left[2\left(\beta_0 - \frac{\beta_1^2}{2\beta_2}\right)\right]\cdot\left[\text{Var}(\beta_0) + \frac{\beta_1^2}{\beta_2^2}\text{Var}(\beta_1) + \frac{1}{4}\frac{\beta_1^4}{\beta_2^4}\text{Var}(\beta_2) - \ldots\right. \\
&\left. 2\frac{\beta_1}{\beta_2}\text{Cov}(\beta_0,\beta_1) + \frac{\beta_1^2}{\beta_2^2}\text{Cov}(\beta_0,\beta_2) - \frac{\beta_1^3}{\beta_2^3}\text{Cov}(\beta_1,\beta_2)\right].
\end{aligned}$$

The standard deviation of λ_0 is therefore:

```
sd_lambda_0 = sqrt(
  exp(
    2*(res$coefficients[1,1] - res$coefficients[2,1]^2/(2*res$coefficients[3,1])))*
    (res$vcov[1,1] +
       res$coefficients[2,1]^2/res$coefficients[3,1]^2*res$vcov[2,2] +
       res$coefficients[2,1]^4/res$coefficients[3,1]^4*res$vcov[3,3]/4 -
       2*res$coefficients[2,1]/res$coefficients[3,1]*res$vcov[1,2] +
       res$coefficients[2,1]^2/res$coefficients[3,1]^2*res$vcov[1,3] -
       res$coefficients[2,1]^3/res$coefficients[3,1]^3*res$vcov[2,3])
  )
sd_lambda_0
```

```
## [1] 3.460543
```

To sum up the resulting estimates are

```r
matrix(c(lambda0,theta,omega,sd_lambda_0,sd_theta,sd_omega),
       nrow=3,ncol=2,dimnames = list(c("lambda0","theta","omega"),
                                     c("Estimate","Std. Error")))
```

```
##         Estimate Std. Error
## lambda0 12.42864   3.460543
## theta   12.91182   1.605586
## omega   12.31179   1.900706
```

g)

Assume that the t_i for $i = 1, \dots, n$ are idependent samples of a Gaussian distribution with mean θ and variance σ^2. The sample mean is μ and the sample variance is s_t^2. To determine if the mean breeding date is significantly different from the estimated optimal breeding date we perform the hypothesis test:

$$H_0 : \mu = \theta \qquad \text{vs.} \qquad H_1 : \mu - \theta \neq 0.$$

The resulting test statistics can be found by

$$z_t = \frac{\mu - \theta}{\sqrt{\frac{s_t^2 + \text{Var}[\theta]}{n}}},$$

and by inputting the respective values we get:

```r
z_t = (mean(data$t)-theta)/(var(data$t) + sd_theta^2)/sqrt(length(data$t))
z_t
```

```
## [1] 0.01023106
```

Observe that the test statistic z_t is 0.0102311 which is between a critical value of $\pm z_{0.025} = \pm 1.96$ and thus the null hypothesis holds.

Exercise 3

In the code below we have implemented a parametric bootstrap algorithm specifically for the model in exercise **2**. We use the results obtained in **2c** to generate 1000 realizations of y_i given the ML estimates of $\hat{\boldsymbol{\beta}}$, then re-fitting 1000 models to these realizations to obtain a bootstrap estimate of the standard deviation of $\boldsymbol{\beta}$. We have also made a print function corresponding to the class pboot that compares the fisher information and the bootstrap estimates.

```r
pboot <- function(mod,start=NA,epsilon=4e-3,trials=1000){
  mf = model.frame(formula = mod$formula, data = mod$data)
  # splitting up covariates and response
  X = model.matrix(attr(mf, "terms"), data = mf)
  y = model.response(mf)
  # if start is not specified input
  if (anyNA(start)){
    start = numeric(ncol(X))
  }
  betas = matrix(NA,nrow = trials, ncol = ncol(X))
  for (i in seq(trials)){
    # generating realization of y from estimate model
    tmp_data = data.frame(t=mod$data$t,y = rpois(length(mod$data$y),
                                                lambda = exp(X%*%mod$coefficients[,1])))
    new_beta = beta = start
    crit = 100
    mf = model.frame(formula = mod$formula, data = tmp_data)
    # splitting up covariates and response
    y = model.response(mf)
    # convergence criterion
    while (epsilon < crit){
      Sb = t(X)%*%(y-exp(X%*%beta)) # score function
      Fb = t(X)%*%as.matrix(as.data.frame(X)*exp(X%*%beta)) # fisher information
      iFb = solve(Fb) # inverse fisher information
      new_beta = beta + iFb%*%Sb # Fisher scoring iteration

      # calculating the convergence criterion
      crit = norm(as.matrix(new_beta - beta),type="2")/norm(as.matrix(beta),type="2")
      beta = new_beta
    }
    betas[i,] = new_beta
  }
  # mean and standard deviation
  res = list(coefficients=cbind(colMeans(betas),
                                apply(betas, 2, sd)))
  res$glm = mod
  res$call = match.call()
  rownames(res$coefficients) = rownames(res$glm$coefficients[,c(1,2)])
  colnames(res$coefficients) = colnames(res$glm$coefficients[,c(1,2)])
  class(res) = "pboot"
  return(res)
}


print.pboot <- function(object){
  cat("Call:\t",format(object$call),"\n")
  cat("\nParametric bootstrapping:\n")
  print(object$coefficients,digits=4)
  cat("\nFisher information:\n")
  print(object$glm$coefficients[,c(1,2)],digits=4)
}
```

Running the parametric bootstrap algorithm on the model fitted in exercise **2c** gives the following estimates:

```r
res_boot = pboot(res)
res_boot
```

```
## Call:     pboot(mod = res)
##
## Parametric bootstrapping:
##              Estimate Std. Error
## (Intercept)  1.391896   0.287508
## t            0.088571   0.034751
## I(t^2)      -0.003407   0.001034
##
## Fisher information:
##              Estimate Std. Error
## (Intercept)  1.420157   0.281865
## t            0.085181   0.034002
## I(t^2)      -0.003299   0.001018
```

Here, we observe that the standard deviations from the two methods are very similar which tells us that this asymptotical approach is a good approximation.