Processing and Analysis of Biological Data

Generalized Linear Models (GLM)

Øystein H. Opedal

14 Nov 2022

Introduction

Generalized linear models (GLMs) add flexibility to the linear model by allowing deviations from the usual assumption of normally distributed residuals. Briefly, a GLM consists of the familiar linear predictor of a linear model (often denoted as η),

$\eta = \beta_0 + \Sigma_j x_{ij}\beta_j + \epsilon_i$

and a link function, g, that places the predictor on a Gaussian (normally-distributed) scale.

$$y = g^{-1}(\eta)$$

Before going into details about GLMs, we need to recall some basics about the most common error distributions used in biological data analyses.

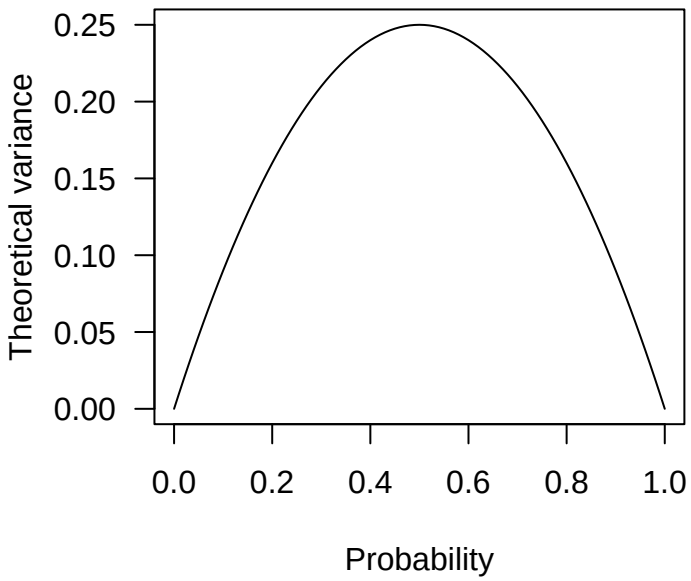
Mean-variance relations for the binomial and poission distributions

The binomial distribution has two parameters, n and p, and summarizes a set of so-called Bernoulli trials with two possible outcomes, yes (1) or no (0). When we have performed more than one trial, we can compute the proportion p of the n trials with a positive outcome.

The theoretical variance of the binomial distribution is given by

$\sigma^2 = np(1-p)$

```
#rbinom(3, 10, c(0.1, 0.5, 0.9))

x = seq(from=0, to=1, by=0.01)
v_b = x*(1-x) #Binomial variance
plot(x, v_b, type="l", xlab="Probability", ylab="Theoretical variance", las=1)
```



This is important to keep in mind, because it affects how we can compare the (proportional) variation of variables measures as proportions. If e.g. one population has a mean of 0.5, if will be expected to be much more variable than a second population with a mean of 0.1, just because there is less opportunity to vary. This is a now well-recognized problem e.g. in demography research, where the interest is often in comparing the extent of variation in life-history traits measured as proportions, such as germination or survival. One proposed solution is to scale the observed CV by the maximum based on the theoretical variance, but a perhaps simpler approach is to transform the proportional data in a way that makes them approach a normal distribution.

One previously popular but now not recommended transformation is the so-called arcsin square-root transformation,

$x' = \arcsin(\sqrt{x})$

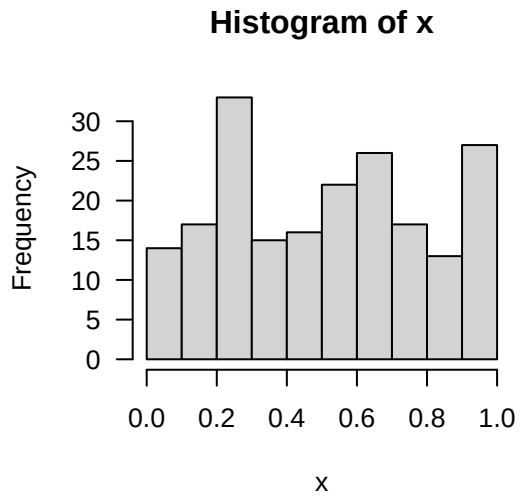
A more meaningful transformation is the logit or log odds transformation

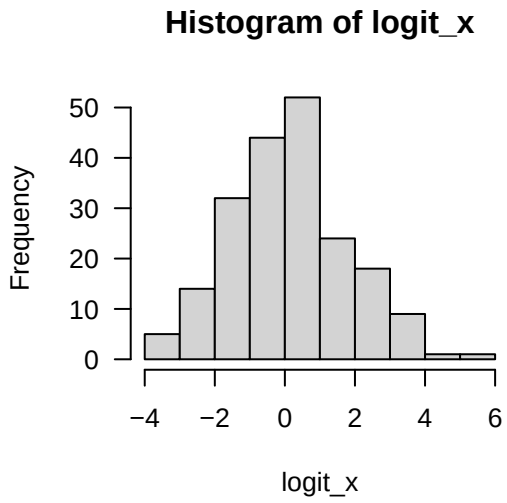
$logit(x) = log(\frac{x}{1-x})$

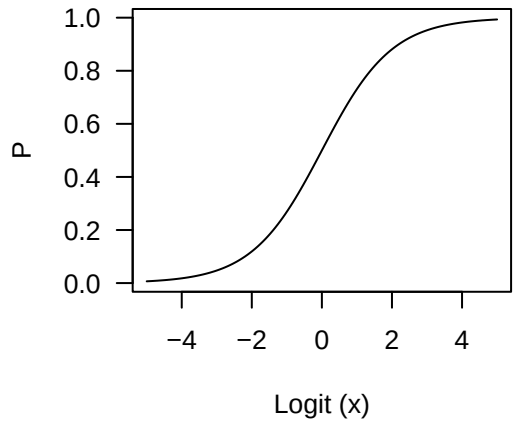
In the following example we are using some functions. This may be a good time to visit the Appendix introducing some basics on writing functions. Note that when functions are given on a single line, we can skip the special curvy brackets (`{`).

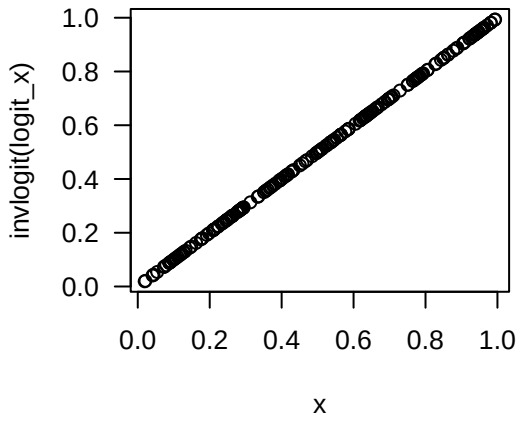
```
logit = function(x) log(x/(1-x))
invlogit = function(x) 1/(1+exp(-x))

x = runif(200)
logit_x = logit(x)

par(mfrow=c(2,2))
hist(x, las=1)
hist(logit_x, las=1)
```

```
xx = seq(-5, 5, 0.01)
plot(xx, invlogit(xx), type="l", las=1,
     xlab="Logit (x)",
     ylab="P")
plot(x, invlogit(logit_x), las=1)
```



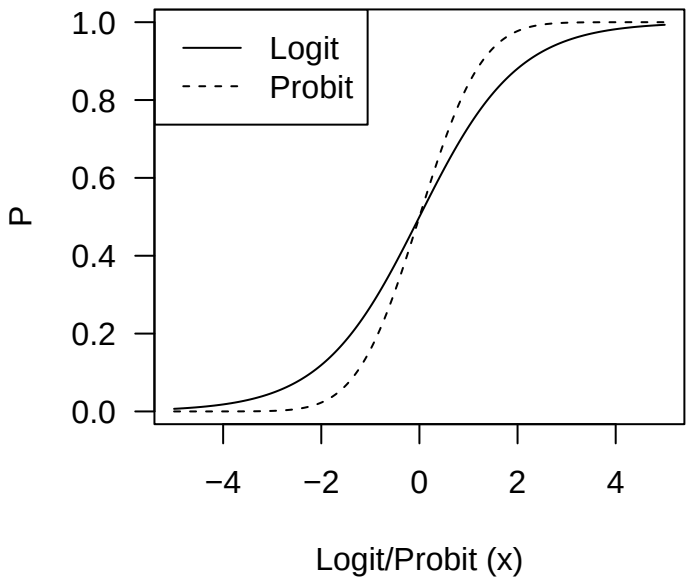






The logit transformation is the most common *link function* in a Generalized Linear Model with binomial errors. A similar alternative is the so-called probit link, which corresponds to the quantile distribution of the standard normal distribution (which is why we can use the `pnorm` function to compute the inverse).

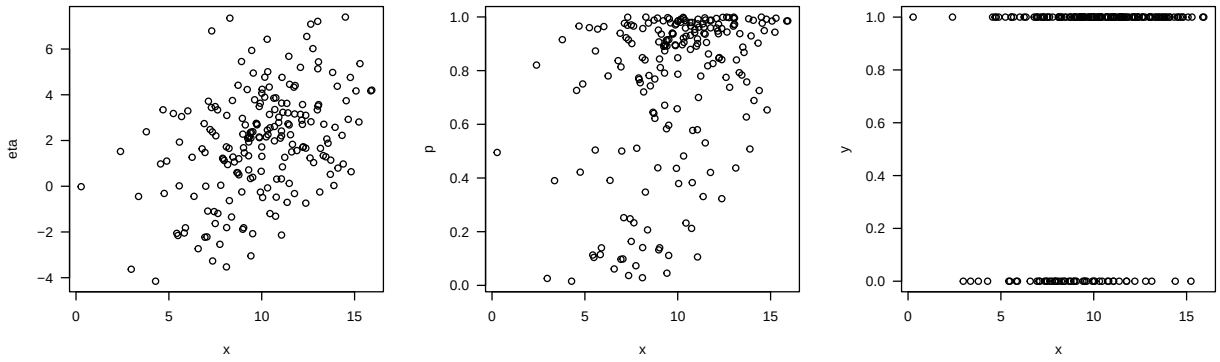
```
plot(xx, invlogit(xx), type="l", las=1,
     xlab="Logit/Probit (x)",
     ylab="P")
lines(xx, pnorm(xx), lty=2)
legend("topleft", legend=c("Logit", "Probit"),
       lty=c(1,2))
```



Logistic regression

As we have already seen, a logistic regression (GLM with binomial errors) is well suited for analysing binary data (or proportions).

```
x = rnorm(200, 10, 3)
eta = -2 + 0.4*x + rnorm(200, 0, 2)
p = invlogit(eta)
y = rbinom(200, 1, p)

par(mfrow=c(1,3))
plot(x, eta, las=1)
plot(x, p, las=1)
plot(x, y, las=1)
```



Above, we simulated data by first formulating a linear predictor η, then transforming the predicted values into probabilities (through the inverse logit transformation), and finally binarizing the data by sampling from the binomial distribution. The last step adds additional uncertainty by accounting for the stochasticity of the observation process.

```
m = glm(y~x, family=binomial(link="logit"))
summary(m)
```

```
##
## Call:
## glm(formula = y ~ x, family = binomial(link = "logit"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.16801   0.04514   0.63804   0.77260   1.52780
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.85048    0.59785  -1.423  0.15486
## x            0.20351    0.06195   3.285  0.00102 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 224.93  on 199  degrees of freedom
## Residual deviance: 213.31  on 198  degrees of freedom
## AIC: 217.31
##
## Number of Fisher Scoring iterations: 4
```

The summary table includes, as always, a lot of information. The first thing to be aware is that when we are fitting a GLM, we obtain the parameter estimates on the link scale (here logit). Note that the parameter estimates are not too far from those we used to define the linear predictor η when we simulated the data. These values are meaningful as such, and if the predictor variable has units of *mm*, the slopes have units of *log odds* mm^{-1}.

To interpret the results biologically and to represent them in graphs, it can be useful to backtransform the predicted values to the probability scale. For example, we can ask how much the probability changes for a

standard deviation increase in the predictor variable (though note that this is no longer a linear transform, so the consequences of increasing and decreasing the predictor by 1 standard deviation may be different).

Recall from the previous section that a log odds of 0 corresponds to a probability of 0.5 ($log(\frac{0.5}{1-0.5}) = log(1) = 0$). If we solve the model equation (the linear predictor) for 0, we can thus obtain the predictor value corresponding to a probability of 0.5, which is often a relevant benchmark.

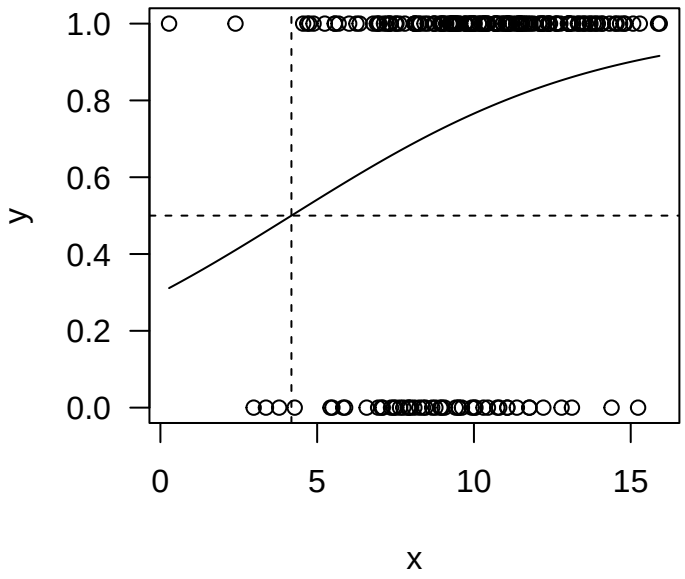
$0 = \beta_0 + \beta_x$

$\frac{-\beta_0}{\beta_x} = x$

EXERCISE: Replicate the plot below. To produce a regression line, we define some new x-values that spans the data range along the y-axis, then obtain predicted values $\hat{y}$ (using the model coefficients), and finally transform these values to the probability scale to obtain the predicted probabilities $\hat{p}$.

```
coefs = summary(m)$coef

x_pred = seq(from=min(x), to=max(x), by=0.01)
y_hat = coefs[1,1] + coefs[2,1]*x_pred
p_hat = invlogit(y_hat)
```

Compute the value of x corresponding to a probability of 0.5, and add lines to the plot to illustrate the results.



The GLM summary table does not provide an r^2 value, because the normal r^2 does not work for logistic regression. There are however several 'Pseudo-r^2' available, typically based on comparing the likelihood of the model to that of a null model (a similar model but with only an intercept). The `MuMIn` package provides one such measure.

```
library(MuMIn)
r.squaredGLMM(m)
```

```
## Warning: 'r.squaredGLMM' now calculates a revised statistic. See the help page.
```

```
## Warning: the null model is correct only if all variables used by the original
## model remain unchanged.
```

```
##                    R2m        R2c
## theoretical 0.08968801 0.08968801
## delta       0.05729288 0.05729288
```

Another possibility for logistic regression is to compute the coefficient of discrimination, or Tjur's D. Indeed, there are several ways to evaluate the performance of a statistical model, and in a logistic regression it makes sense to ask how well the model discriminates between true positives and negatives in the data.

The coefficent of discrimination is defined as $D = \bar{\hat{\pi}_1} - \bar{\hat{\pi}_0}$, and is computed by comparing the predicted probability for those data points that are successes or positives (1s; $\bar{\hat{\pi}_1}$) to the predicted probability for those data points that are failures or negatives (0s; $\bar{\hat{\pi}_1}$).

```
y_hat = coefs[1,1] + coefs[2,1]*x
p_hat = invlogit(y_hat)

mean(p_hat[which(y==1)]) - mean(p_hat[which(y==0)])
```

```
## [1] 0.05766512
```

Some final notes on fitting binomial GLM's. There are three ways to formulate these models in R. In the example above, the data were 0's and 1's, and we could specify the model simply as

```
glm(y ~ x, family=binomial(link="logit"))
```

When each observation is based on more than one trial, we can formulate the model in two ways. The first is

```
glm(y ~ x, family=binomial(link="logit"), weights=n)
```

where `y` is the proportion of successes, and `n` is the number of trials. The second method is to fit a two-column matrix as response variable, where the first colomn is the number of successes, and the second column is the number of failures, i.e. `y = cbind(successes, failures)`. The model formula is then

```
glm(cbind(successes, failures) ~ x, family=binomial(link="logit"))
```

Data exercise: seed germination

The following data are from a study investigating patterns of seed dormancy in a plant. In this plant species, seeds needs to stay in dry conditions for a specific amount of time before they are ready to germinate, a process known as 'after-ripening'. once the seeds are 'ripe' they will normally germinate when exposed to favourable (moist) conditions.

After different durations of after-ripening, the seeds were sown on moist soil and their germination success (proportion of seeds germinated) recorded. The seeds came from four different populations, and were weighed (in mg) prior to sowing.

The variables in the data are as follows:

- `pop` = Population ID
- `mother` = Maternal plant ID

- `crossID` = Unique cross identifier
- `blocktray` = Sowing tray (experimental block)
- `timetosowing` = Time in days from seed dispersal to watering
- `MCseed` = Population-mean-centered seed mass in mg
- `nseed` = Number of seeds sown
- `germ2` = Proportion of seeds germinated

Analyse the data to estimate the pattern of germination success in response to variation in the duration of after-ripening. Are the patterns similar in different populations? Are there other factors affecting germination success? Produce relevant summary statistics, parameter estimates, and graphs.

```
dat = read.csv("datasets/dormancy/dormancy.csv")
names(dat)
```

```
## [1] "pop"          "mother"       "crossID"      "blocktray"    "timetosowing"
## [6] "MCseed"       "nseed"        "germ2"
```

As a suggested start, the following lines fit a simple model to data from one population using two different methods. Note that the model fit is the same (the log Likelihood of the two models is identical).

```
subdat = dat[dat$pop=="CC",]

germ = subdat$germ2 * subdat$nseed #Successes
notgerm = subdat$nseed - germ #Failures

mod1 = glm(cbind(germ, notgerm) ~ timetosowing, "binomial", data=subdat)
mod2 = glm(germ2 ~ timetosowing, "binomial", weights=nseed, data=subdat)
logLik(mod1) == logLik(mod2)
```

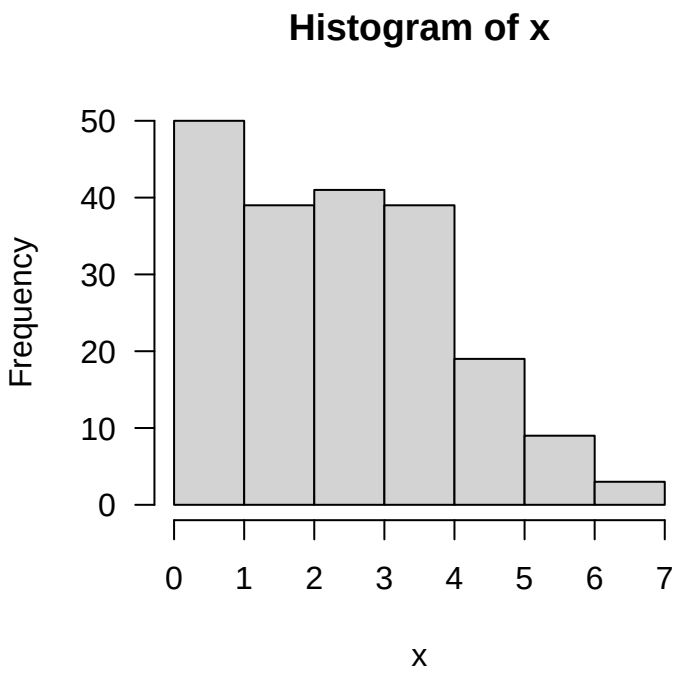
```
## [1] TRUE
```

Can you use the fitted models to estimate the duration of after-ripening required for the expected germination rate to be 0.5?

8. Generalized linear models II: Poisson and negative-binomial regression

A second very common data type in biology is count data, which occurs when we have counted something. In ecological studies we often count individuals or species, and in evolutionary biology we often count e.g. the number of offspring. For such data, the data distribution is often skewed, and the variance tends to increase with the mean. The Poisson distribution is tailored for such data.

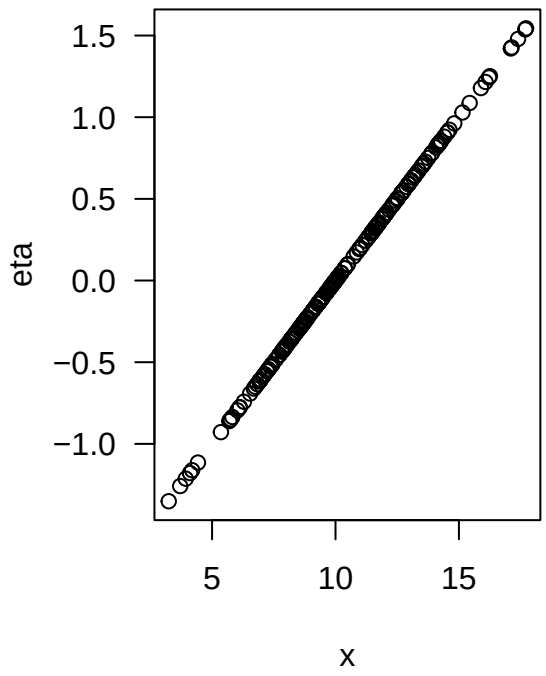
```
x = rpois(200, 3)
hist(x, las=1)
```

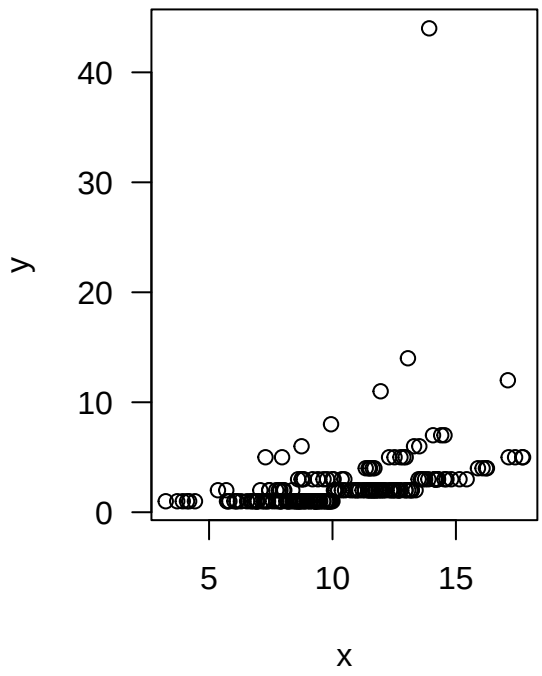


The Poisson distribution has a single parameter λ that determines both the mean and the variance. Thus, the variance increases linearly with the mean.

The distribution of count data can sometimes by normalized through a log-transformation, and the log is indeed the link function of a Poisson regression model. The alternative method of log-transforming the data and then fitting a Gaussian model is problematic when there are zeros in the data. Adding a constant (e.g. 0.5 or 1) is sometimes an option, but is generally not recommended. A better option is to analyze the data in a GLM framework with Poisson-distributed errors and a log link function.

```
x = rnorm(200, 10, 3)
eta = -2 + 0.2*x
y = ceiling(exp(eta + rpois(200, 0.3)))

par(mfrow=c(1,2))
plot(x, eta, las=1)
plot(x, y, las=1)
```






```
m = glm(y~x, family="poisson")
summary(m)
```

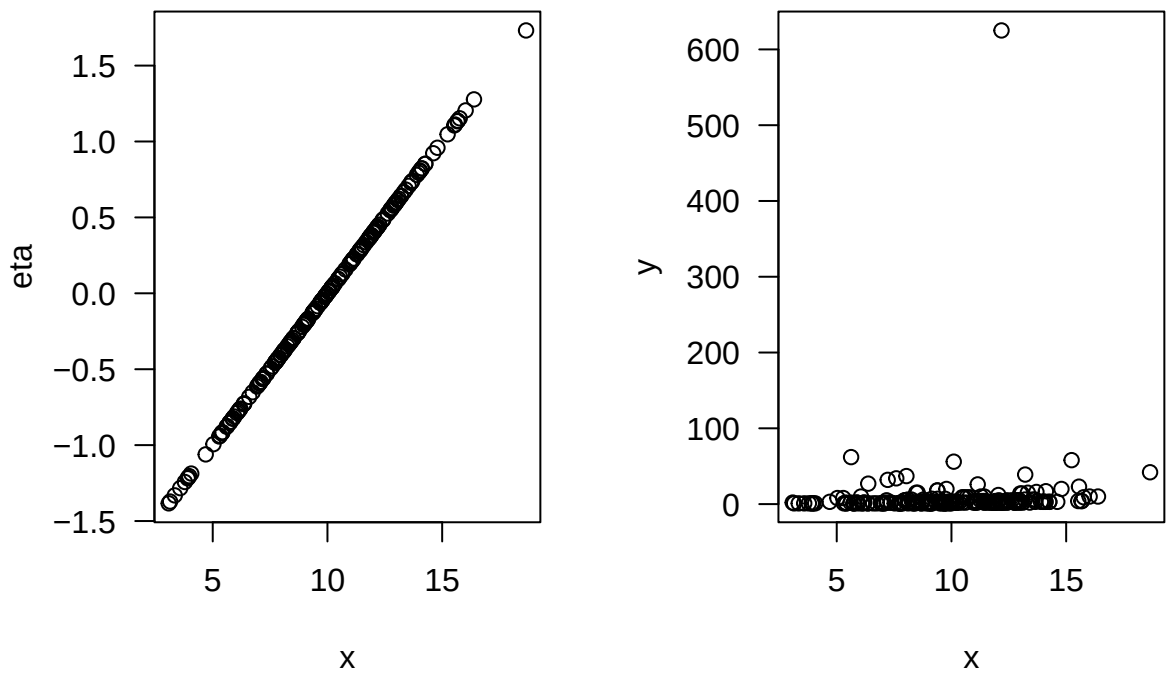
```
##
## Call:
## glm(formula = y ~ x, family = "poisson")
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.0571  -0.7021  -0.4626   0.2138  11.3900
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.89148    0.18111  -4.922 8.56e-07 ***
## x            0.16483    0.01483  11.113  < 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: 391.05  on 199  degrees of freedom
## Residual deviance: 267.69  on 198  degrees of freedom
## AIC: 790.86
##
## Number of Fisher Scoring iterations: 5
```

When interpreting Poisson regressions, there are several things to keep in mind. First, as in all GLMs, the parameters are reported on the link scale, here log. Recall that the link scale has useful proportional properties, so that the slope can be interpreted roughly as the proportional change in y per unit change in x.

Second, recall that the Poisson distribution has a single parameter λ derermining both the mean and the variance. In real count data the variance often increase disproportionally compared to the mean, a phenomenon called *overdispersion*. Biologically, this occurs because we tend to observe multiple entities together. For example, in a survey of common eiders, we will often see either no eiders, or a lot of eiders.

We can quantify overdispersion in the data based on the fitted model by calculating the ratio of the residual deviance to the residual degrees of freedom. In the model above there is no serious overdispersion, because the residual deviance is only a little larger than the residual degrees of freedom. Let us construct an example with more serious overdispersion.

```
x = rnorm(200, 10, 3)
eta = -2 + 0.2*x
y = ceiling(exp(eta + rnbinom(200, 1, mu=.8)))

par(mfrow=c(1,2))
plot(x, eta, las=1)
plot(x, y, las=1)
```




```
m = glm(y~x, family="poisson")
summary(m)
```

```
##
```

```
## Call:
## glm(formula = y ~ x, family = "poisson")
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.335  -2.931  -2.111  -1.270  61.495
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.500430   0.093107   5.375 7.67e-08 ***
## x           0.158445   0.007978  19.860  < 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: 6052.9  on 199  degrees of freedom
## Residual deviance: 5651.3  on 198  degrees of freedom
## AIC: 6260.1
##
## Number of Fisher Scoring iterations: 7
```

Here the overdispersion is serious, and we can not trust the model estimates. In this case, we need an alternative link function that allows the variance to increase more than the mean. The negative binomial distribution is a good option.

```
library(MASS)
m = glm.nb(y~x)
summary(m)
```

```
##
## Call:
## glm.nb(formula = y ~ x, init.theta = 0.6407200349, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.1351  -0.9986  -0.6551  -0.4863   7.8163
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.37996    0.31802   1.195    0.232
## x            0.16974    0.03056   5.554 2.78e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(0.6407) family taken to be 1)
##
##     Null deviance: 247.59  on 199  degrees of freedom
## Residual deviance: 218.00  on 198  degrees of freedom
## AIC: 1228.6
##
## Number of Fisher Scoring iterations: 1
##
```

```
##
##                Theta:  0.6407
##            Std. Err.:  0.0595
##
##  2 x log-likelihood:  -1222.6260
```

Note that the parameter estimates are similar but the estimated standard error is much higher.

8. Generalized linear models III: Hurdle models