Practicals (with solutions)

Olivier Gimenez

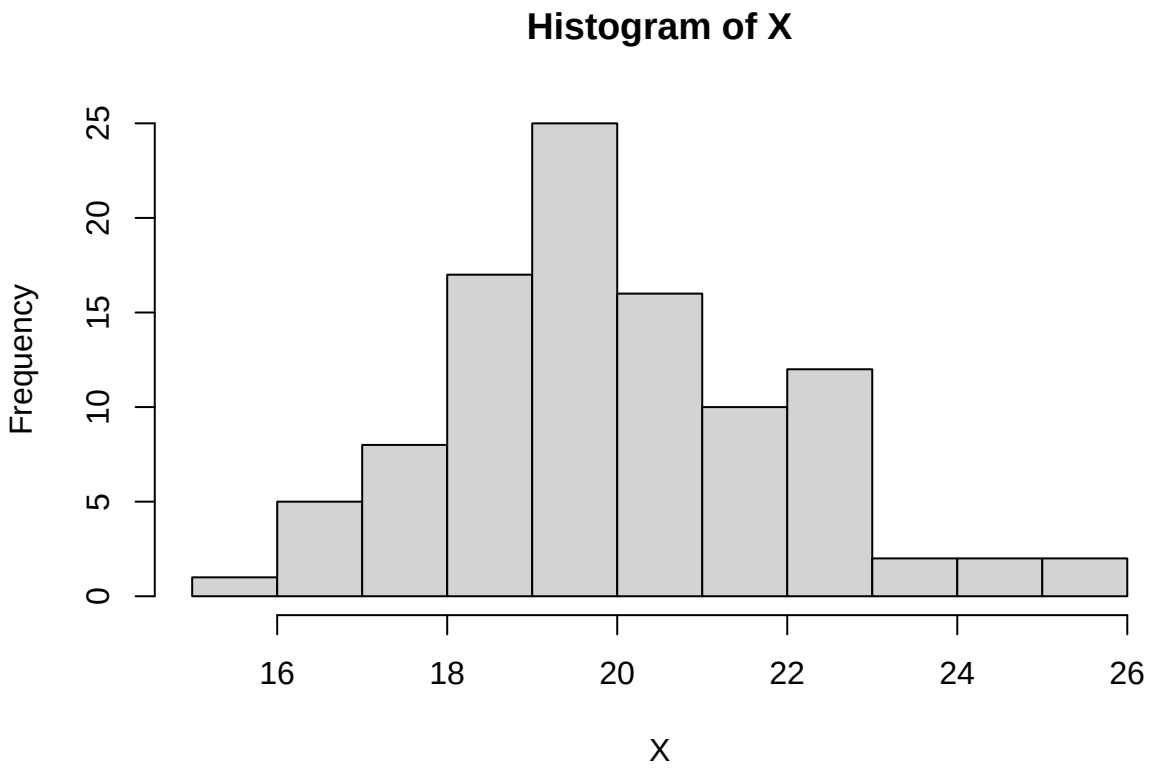
Practical #1

1. Use simulation to generate a data set of 100 covariate values from $X \sim \text{Normal}(20, \sigma = 2)$. Use a histogram to explore the distribution of X.

```
X <- rnorm(100, 20, 2)
X
```

```
##   [1] 21.11313 16.98806 18.75531 20.39310 20.69056 21.55832 24.09201 21.91313
##   [9] 22.36743 17.94493 18.38211 20.84481 19.25563 21.69480 22.03177 18.95384
##  [17] 22.99756 18.08997 20.80844 19.22738 18.67461 21.52148 19.78770 17.92379
##  [25] 19.63062 19.00796 20.37837 18.26209 21.51708 17.60400 19.03495 17.29373
##  [33] 15.59637 19.85437 21.93216 22.07678 19.88514 22.52074 19.05849 19.40907
##  [41] 20.27377 19.39220 21.48025 19.60173 20.76120 16.42910 18.56281 19.52380
##  [49] 19.23572 17.78290 20.35548 18.85290 16.89403 19.43051 18.46125 20.57020
##  [57] 19.08199 18.76050 18.60912 19.68154 23.15772 19.47189 20.20332 20.82894
##  [65] 17.93998 17.46641 22.68105 18.22159 16.89974 21.68097 20.05487 22.08072
##  [73] 19.87977 19.06121 24.50353 20.07206 22.08422 25.90894 19.95888 18.51945
##  [81] 19.75002 19.50529 18.99521 18.38603 20.22578 17.96609 19.92939 20.72435
##  [89] 20.11184 22.78767 18.38689 23.51696 18.98877 21.64547 22.25611 22.01352
##  [97] 16.60705 19.81225 22.89795 25.01177
```

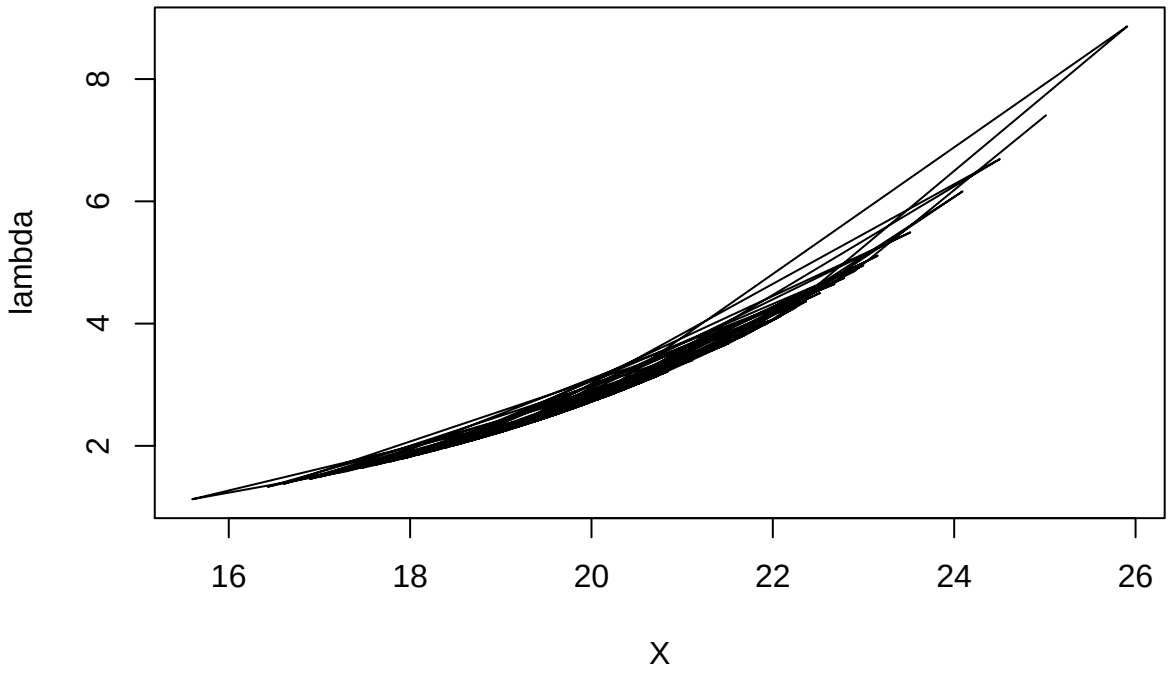
```
hist(X)
```



OK, looks normal to me, with mean 20 and some variability around it.

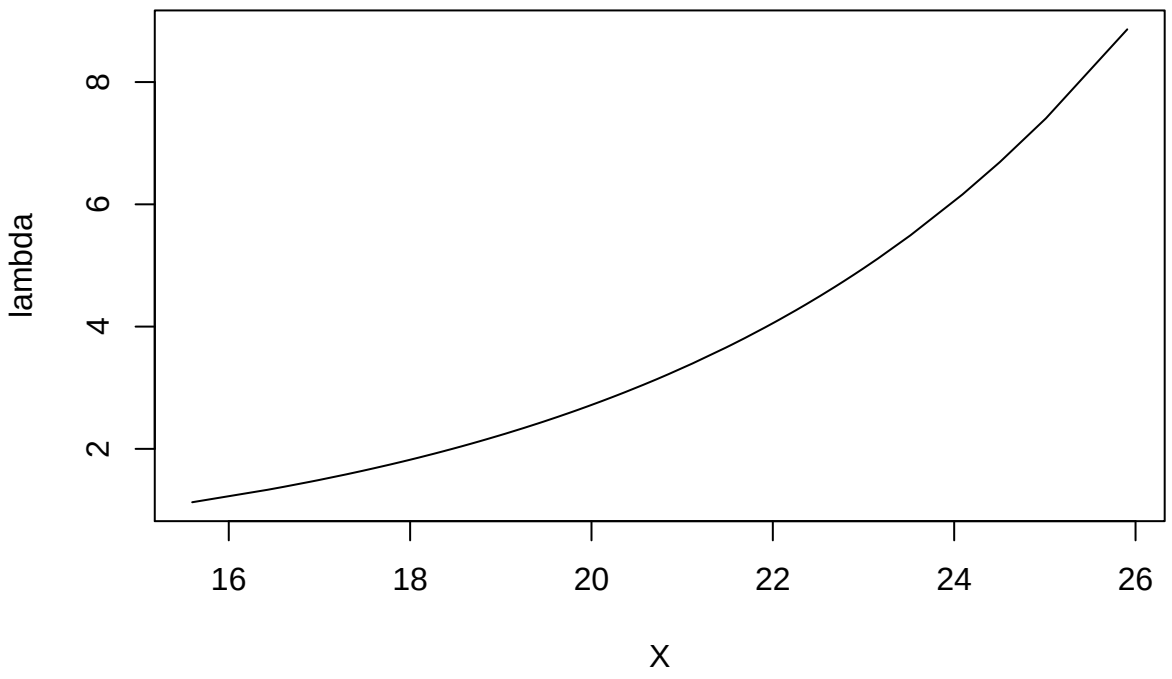
2. Generate a Poisson rate from these values according to the relationship lambda=exp(-3+0.2*X). Have a look graphically to the relationship between λ and X. Hint: You might want to order the values of X using the R function `sort()`.

```
lambda <- exp(-3 + 0.2 * X)
plot(X,lambda, type="l")
```



Something went wrong... Yes, we need to order the X values!

```
X <- sort(X)
lambda <- exp(-3 + 0.2 * X)
plot(X,lambda, type="l")
```



3. Use simulation to generate a data set of 100 response values from a Poisson distribution according to this rate $Y \sim \text{Poisson}(\lambda(X))$. Explore the distribution of Y using a bar plot (`?barplot`).

```
Y <- rpois(100, lambda)
Y
```

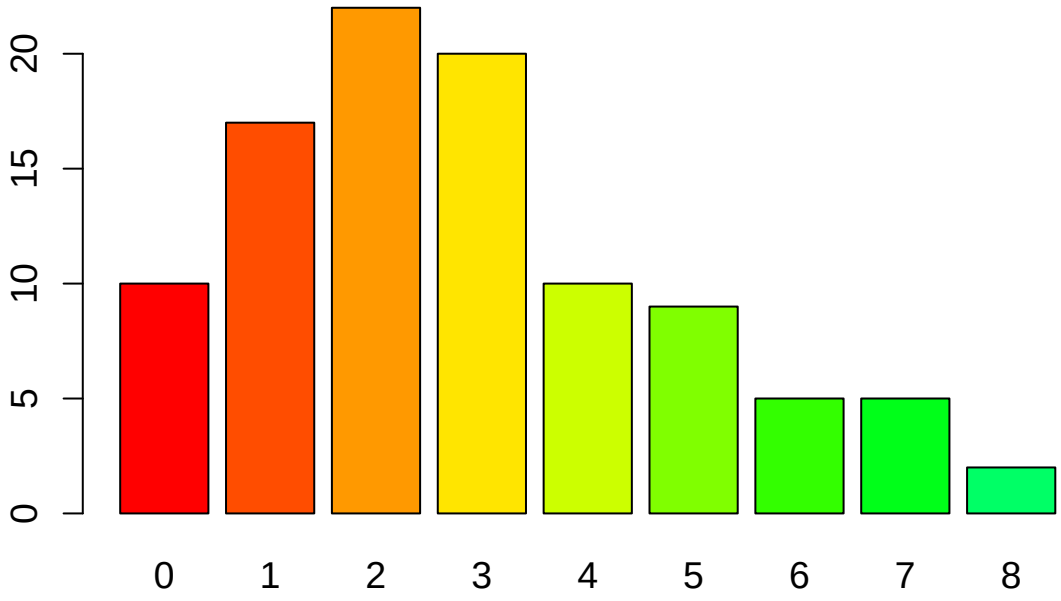
```
##   [1] 0 0 4 2 2 0 0 2 0 1 1 3 5 3 1 0 0 1 3 5 5 4 3 2 3 3 7 3 2 5 4 1 2 2 1 6 3
##  [38] 3 0 7 1 2 1 0 0 3 1 1 1 2 1 2 3 1 3 2 3 4 2 6 1 2 2 4 4 1 2 4 2 3 2 3 2 3
##  [75] 1 4 2 6 3 2 2 3 1 3 5 5 5 5 6 8 4 7 8 6 7 2 3 4 5 7
```

Explore the distribution of Y

```
tab.Y <- table(Y)
tab.Y
```

```
## Y
##  0  1  2  3  4  5  6  7  8
## 10 17 22 20 10  9  5  5  2
```

```
barplot(tab.Y,col=rainbow(20),cex.axis=1.2,cex.names=1.2,main='')
```



4. Construct a data frame containing the covariate (X) and response (Y) data. Have a look to the first and last rows.

```
dat <- data.frame(cbind(X,Y))
head(dat)
```

```
##          X Y
## 1 15.59637 0
## 2 16.42910 0
## 3 16.60705 4
## 4 16.89403 2
## 5 16.89974 2
## 6 16.98806 0
```

```
tail(dat)
```

```
##            X Y
## 95  23.15772 7
## 96  23.51696 2
## 97  24.09201 3
## 98  24.50353 4
## 99  25.01177 5
## 100 25.90894 7
```

5. Use a GLM with the appropriate structure to try and retrieve the parameters -3 and 0.2 from question 2.

```
m <- glm(Y~X, family=poisson, dat)
broom::tidy(m)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic      p.value
```

```
##   <chr>          <dbl>     <dbl>     <dbl>        <dbl>
## 1 (Intercept)   -2.20     0.589      -3.73 0.000193
## 2 X              0.160    0.0283      5.64 0.0000000173
```

It looks as though the intercept and slope estimates are close to the values -3 and 0.2 we used to simulate the data.

6. Increase sample size (multiply by 1000) and comment on the parameter estimates.

```
N <- 100000
X <- sort(rnorm(N, 20, 2))
lambda <- exp(-3 + 0.2 * X)
Y <- rpois(N, lambda)
dat <- data.frame(cbind(X,Y))
m <- glm(Y~X, family=poisson, dat)
summary(m)
```

```
##
## Call:
## glm(formula = Y ~ X, family = poisson, data = dat)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7527  -0.7972  -0.1043   0.5811   4.0992
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.9664589  0.0192720  -153.9   <2e-16 ***
## X            0.1982977  0.0009228   214.9   <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: 155809  on 99999  degrees of freedom
## Residual deviance: 109742  on 99998  degrees of freedom
## AIC: 374078
##
## Number of Fisher Scoring iterations: 5
```

With increasing sample size, the estimates get much closer to the values we used to generate the data (i.e. the truth). That's an appealing property of maximum likelihood estimates.

7. Overdispersion. We would like to do the same exercise with an overdispersed Poisson distribution. To do so, we need to generate data from a quasi-Poisson distribution. Interestingly, the negative-binomial (NB) distribution allows relaxing the Poisson assumption $E(Y) = V(Y) = \lambda$. There are several parameterizations for the NB distribution. Here, we will use $W \sim NB(\lambda, \phi)$ where λ is the expected value as in the Poisson distribution and ϕ is the overdispersion parameter. From the NB properties, we have that the mean of W is λ while its variance is $\lambda + \lambda^2/\phi$. Therefore, a small value of ϕ means a large deviation from a Poisson distribution (the variance of W is much larger than λ, which would also be the mean for a Poisson distribution), while as ϕ gets larger the NB looks more and more like a Poisson distribution (the term λ^2/ϕ tends to 0, and the mean and variance looks more and more alike). In R, we will specify:

```
lambda = 2
phi = 5
n = 1000

# simulate the response
w_nb <- rnbinom(n,size=phi,mu=lambda)

# recover the NB parameters from its mean and variance
mean(w_nb) # lambda
```

```
## [1] 1.871
```

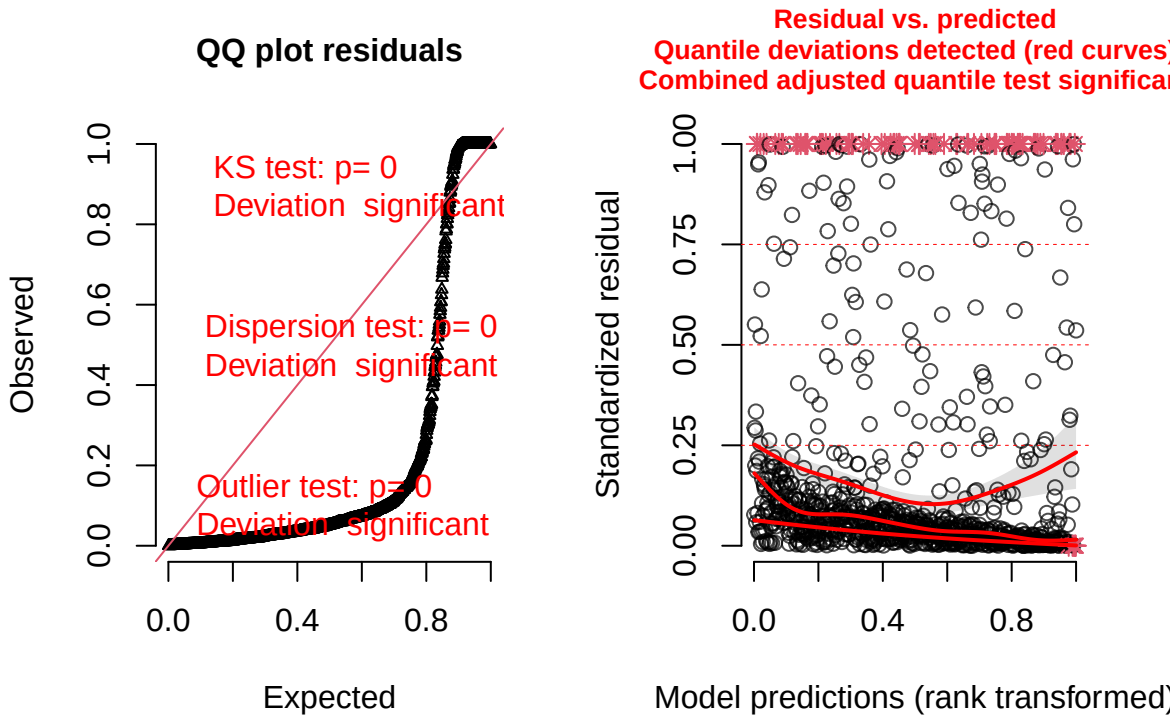
```
mean(w_nb)^2/(var(w_nb)-mean(w_nb)) # phi
```

```
## [1] 4.079582
```

Adopt the same approach as above to simulate data from a Poisson distribution with overdispersion parameter $\phi = 0.1$. Inspect the residuals. Do they look ok to you? If not, fit a GLM with an appropriate structure. Have a look to the relationship between the response and the predictor for both models, without and with the overdispersion parameter (use the `visreg` from the package `visreg`). Is there any difference?

```
N <- 1000
X <- sort(rnorm(N, 20, 2))
lambda <- exp(-3 + 0.2 * X)
phi <- 0.1
Y <- rnbinom(N,size=phi,mu=lambda)
dat <- data.frame(cbind(X,Y))
m <- glm(Y~X, family=poisson, dat)
DHARMa::simulateResiduals(m, plot = TRUE)
```

```
## DHARMa:plot used testOutliers with type = binomial for computational reasons (nObs > 500). Note that
```

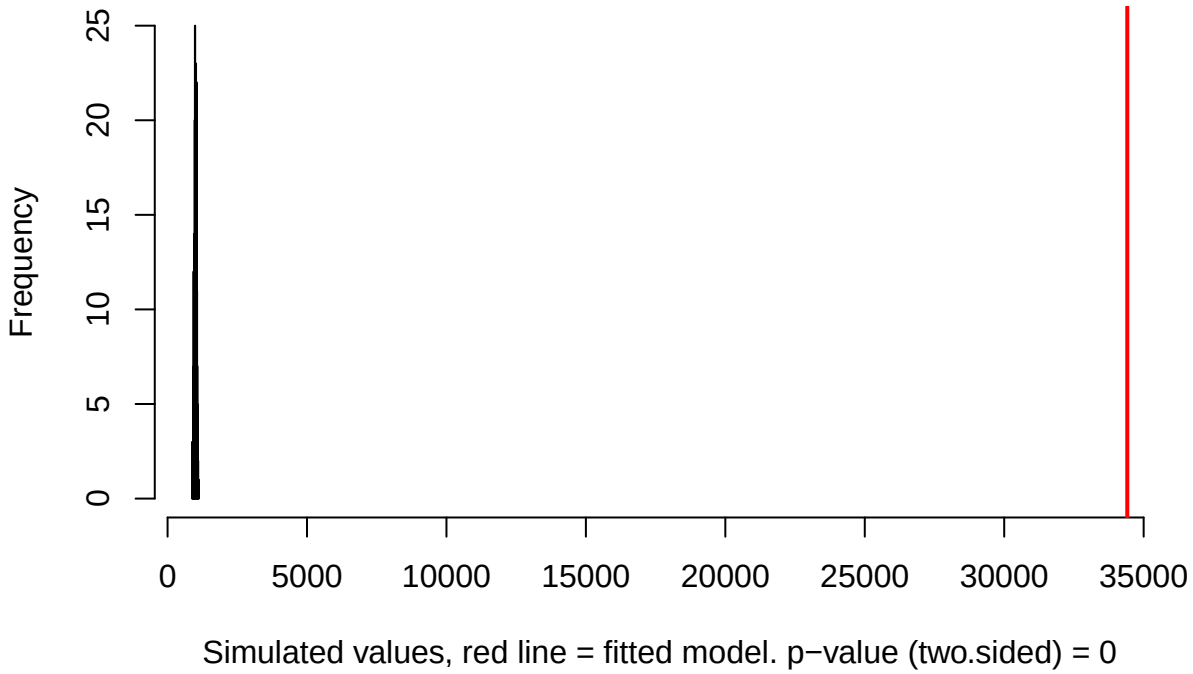



```
## Object of Class DHARMa with simulated residuals based on 250 simulations with refit = FALSE . See ?DH
##
## Scaled residual values: 0.07744324 0.2935717 0.5506392 0.1996207 0.2868503 0.3338296 0.05843678 0.18
```

Something is weird in the residuals, clearly the Poisson distribution does not seem to be appropriate for these data. Is it an overdispersion issue?

```
simres <- DHARMa::simulateResiduals(m, refit = TRUE)
DHARMa::testOverdispersion(simres)
```

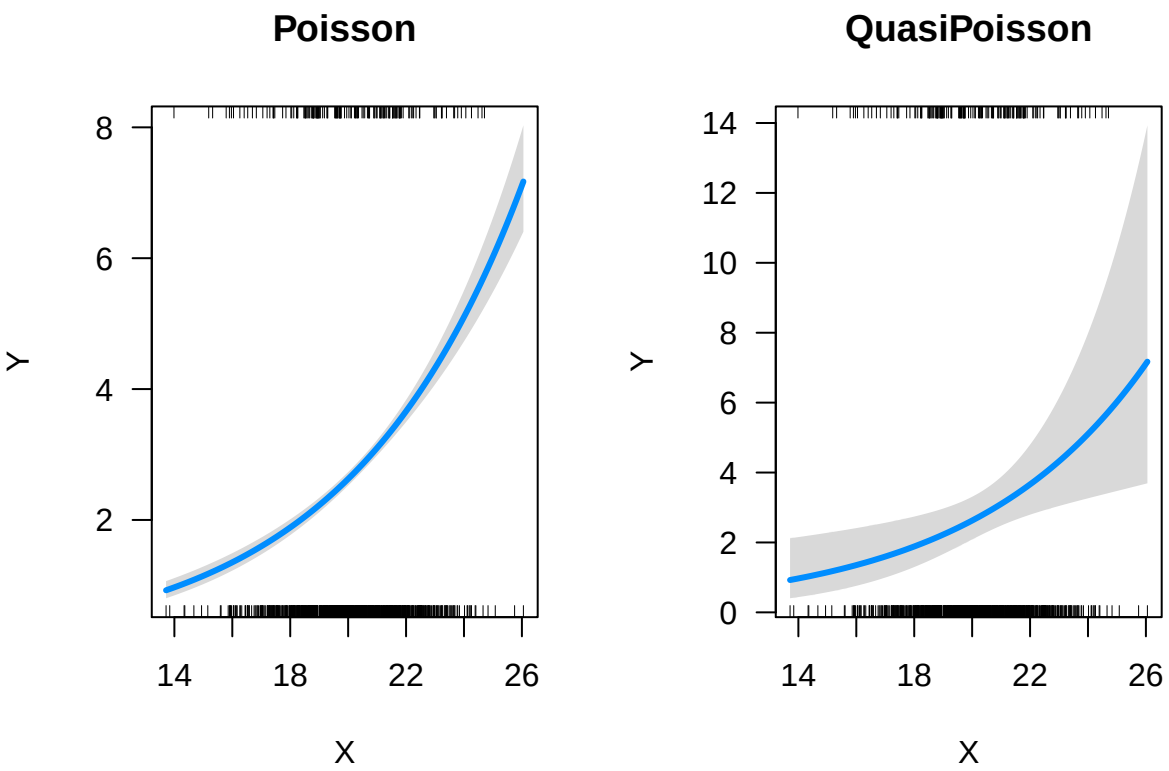
```
## testOverdispersion is deprecated, switch your code to using the testDispersion function
```




```
##
##  DHARMa nonparametric dispersion test via mean deviance residual fitted
##  vs. simulated-refitted
##
## data:  simulationOutput
## dispersion = 34.482, p-value < 2.2e-16
## alternative hypothesis: two.sided
```

Yes, overdispersion is significant. Let's fit a model with overdispersion (quasiPoisson distribution).

```
m_quasi <- glm(Y~X, family=quasipoisson, dat)
```



The uncertainty is bigger when overdispersion is accounted for. Overdispersion is often related to pseudoreplication. In brief, it means that there is some dependence among the statistical units (e.g., individuals). As a consequence, while you might think you have, say, a hundred individuals, it is actually (much) less in a statistical sense because some of them bring the same information due to their dependence (ressemblance) with each other. In turn, the confidence intervals are narrower than they should be. This issue is addressed by accounting for overdispersion with a quasiPoisson distribution, therefore explaining the wider confidence intervals.

Practical #2

1. Relationship between tree diameter and height and LMM. You have data on 1000 trees from 10 plots, with 4 up to 392 trees per plot and several measurements taken on each tree.

```
# dbh is diameter at breast height (diameter of the trunk)
trees <- read.table("trees.txt", header=TRUE)
str(trees)
```

```
## 'data.frame':    1000 obs. of  6 variables:
##  $ plot  : int  2 4 5 2 4 4 1 5 1 2 ...
##  $ dbh   : num  38.9 26.1 42.7 20.7 21.8 ...
##  $ height: num  37.8 38.1 50.2 30.1 34 21.9 18.2 35.8 35.5 41 ...
##  $ sex   : chr  "female" "female" "female" "female" ...
##  $ dead  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dbh.c : num  13.85 1.05 17.66 -4.28 -3.17 ...
```

```
head(trees)
```

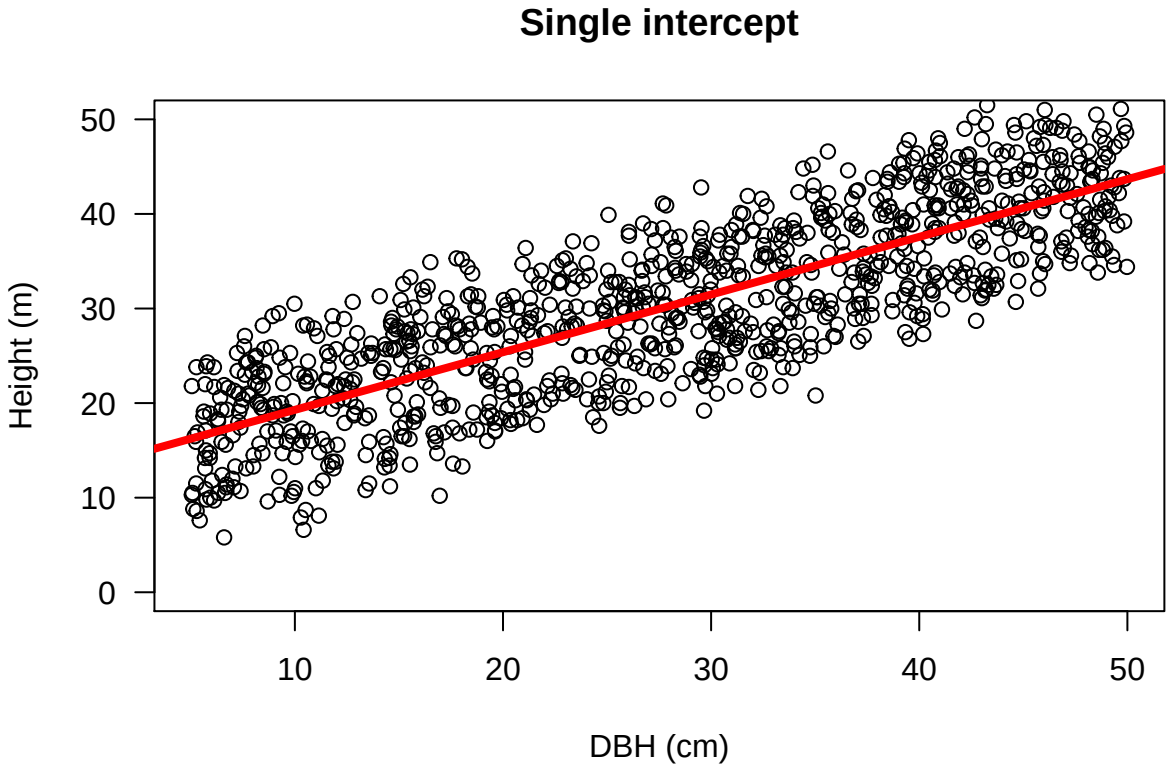
```
##   plot   dbh height    sex dead  dbh.c
## 1    2 38.85   37.8 female    0  13.85
## 2    4 26.05   38.1 female    0   1.05
## 3    5 42.66   50.2 female    0  17.66
## 4    2 20.72   30.1 female    0  -4.28
## 5    4 21.83   34.0 female    0  -3.17
## 6    4  8.23   21.9   male    0 -16.77
```

Fit a linear model with height as the response variable and dbh as a predictor. Plot the data on a graph, and add the regression line (use `abline`).

```
lm.simple <- lm(height ~ dbh, data = trees)
summary(lm.simple)
```

```
##
## Call:
## lm(formula = height ~ dbh, data = trees)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7384  -4.7652   0.4759   4.2931  13.5282
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.18767    0.41476   31.80   <2e-16 ***
## dbh          0.60967    0.01351   45.14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.549 on 998 degrees of freedom
## Multiple R-squared:  0.6712, Adjusted R-squared:  0.6709
## F-statistic:  2038 on 1 and 998 DF,  p-value: < 2.2e-16
```

```
plot(height ~ dbh, data=trees, las=1, xlab="DBH (cm)", ylab="Height (m)", ylim = c(0, 50), main = "Sing
abline(lm(height ~ dbh, data=trees), lwd=4, col="red")
```



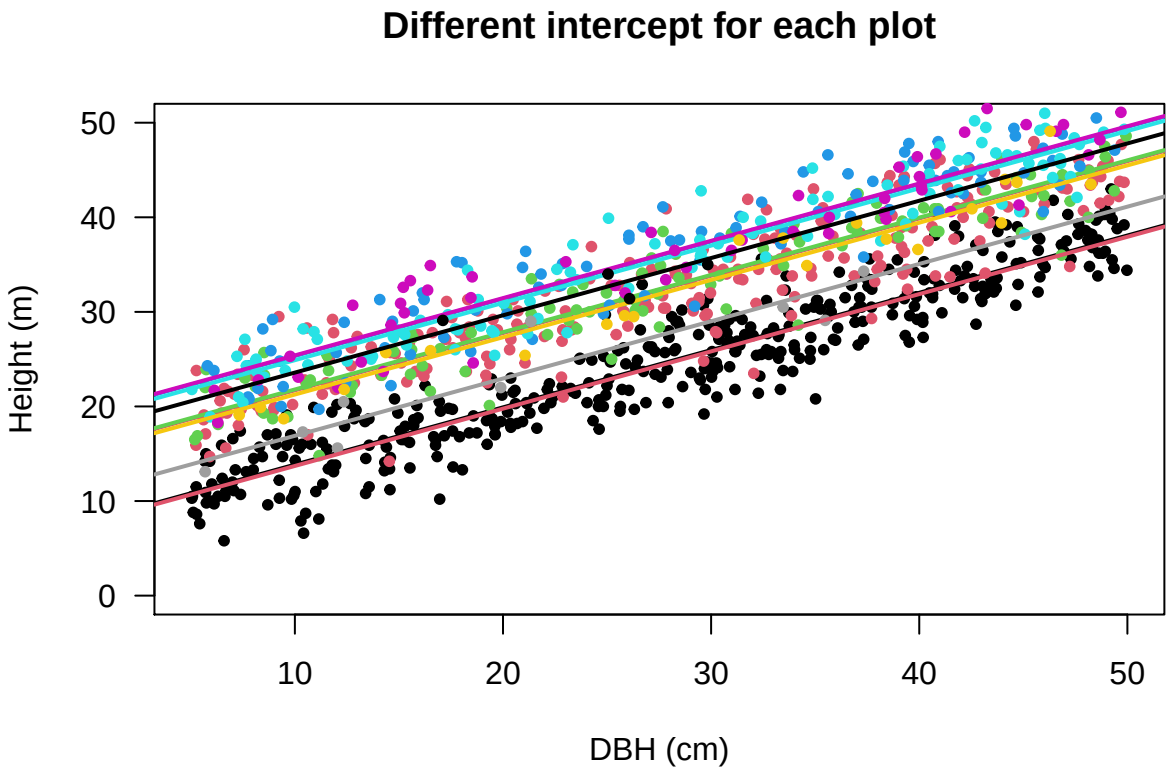
Now we have only one intercept. What if allometry varies among plots? Fit a linear model with a different intercept for each plot, and inspect the results with `broom::tidy`. Try and represent the 20 regression lines on the same graph.

```
lm.interc <- lm(height ~ factor(plot) + dbh, data = trees)
broom::tidy(lm.interc)
```

```
## # A tibble: 11 x 5
##    term           estimate std.error statistic   p.value
##    <chr>             <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)       7.79    0.243      32.0   5.94e-155
##  2 factor(plot)2     7.86    0.245      32.1   1.96e-155
##  3 factor(plot)3     7.95    0.319      24.9   1.70e-106
##  4 factor(plot)4    11.5     0.334      34.4   7.30e-171
##  5 factor(plot)5    11.0     0.321      34.4   1.97e-171
##  6 factor(plot)6    11.6     0.430      26.9   5.07e-120
##  7 factor(plot)7     7.41    0.632      11.7   7.97e- 30
##  8 factor(plot)8     3.05    0.974       3.13  1.79e-  3
##  9 factor(plot)9     9.73    1.45        6.71  3.23e- 11
## 10 factor(plot)10   -0.138   0.925      -0.149 8.81e-  1
## 11 dbh               0.606   0.00704    86.0   0.
```

```
plot(trees$dbh[trees$plot==1], trees$height[trees$plot==1],
     pch=20, las=1, xlab="DBH (cm)", ylab="Height (m)", col=1,
     ylim=c(0,50), main = "Different intercept for each plot")
abline(a=coef(lm.interc)[1], b=coef(lm.interc)[11], col=1, lwd=2) # plot 1
for(i in 2:10){
  points(trees$dbh[trees$plot==i], trees$height[trees$plot==i], pch=20, col=i)
```

```
  abline(a=coef(lm.interc)[1] + coef(lm.interc)[i], b=coef(lm.interc)[11], col=i, lwd=2) # plot 2-10
}
```



How many parameters are estimated in the model pooling all plots together vs. the model considering a different intercept for each plot?

The answer is 3 (intercept, slope, residual variance) vs. 12 (10 intercepts, 1 slope, 1 residual variance).

Mixed models enable us to make a compromise between the two models by accounting for plot-to-plot variability.

Recall that:

$$\begin{aligned} y_{ij} &\sim N(\mu_{ij}, \sigma^2) \\ \mu ij &= a_j + bx_i \\ a_j &\sim N\left(\mu_a, \tau^2\right) \end{aligned}$$

How does this GLMM formulation translate into in our example? In our example, it translates into:

$$\begin{aligned} \text{Height}_{ij} &\sim N(\mu_{ij}, \sigma^2) \\ \mu_{ij} &= a_j + b\ DBH_i \\ a_j &\sim N\left(\mu_a, \tau^2\right) \\ j &= 1, \ldots, 10 \text{ (plots)} \end{aligned}$$

We have the gradient:

- **complete pooling**: Single overall intercept.
  - `lm (height ~ dbh)`
- **no pooling**: One *independent* intercept for each plot.

  - `lm (height ~ dbh + factor(plot))`

- **partial pooling**: Inter-related intercepts.
  - `lmer(height ~ dbh + (1 | plot))`

Fit model with plot as a random effect. Comment on the outputs.

```
library(lme4)
```

```
## Loading required package: Matrix
```

```
mixed <- lmer(height ~ dbh + (1|plot), data = trees)
summary(mixed)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: height ~ dbh + (1 | plot)
##    Data: trees
##
## REML criterion at convergence: 5007.6
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.84491 -0.65574 -0.02247  0.69295  3.09733
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  plot     (Intercept) 19.834   4.454
##  Residual              8.325   2.885
## Number of obs: 1000, groups:  plot, 10
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) 14.79816    1.43742   10.29
## dbh          0.60566    0.00704   86.03
##
## Correlation of Fixed Effects:
##     (Intr)
## dbh -0.135
```

The following are the important pieces:

```
Random effects:
 Groups   Name        Variance Std.Dev.
 plot     (Intercept) 19.834   4.454
 Residual              8.325   2.885
Number of obs: 1000, groups:  plot, 10
```

This tells us the variance and standard deviation (Std.Dev. = sqrt(Variance)) of our random intercept on `plot` (τ^2) and of the residual variation `Residual` (σ^2). We are also told how many rows of data we have `Number of obs: 1000` and how many random intercept groups we have `plot, 10`.

```
Fixed effects:
            Estimate Std. Error t value
(Intercept) 14.79816    1.43742   10.29
dbh          0.60566    0.00704   86.03
```

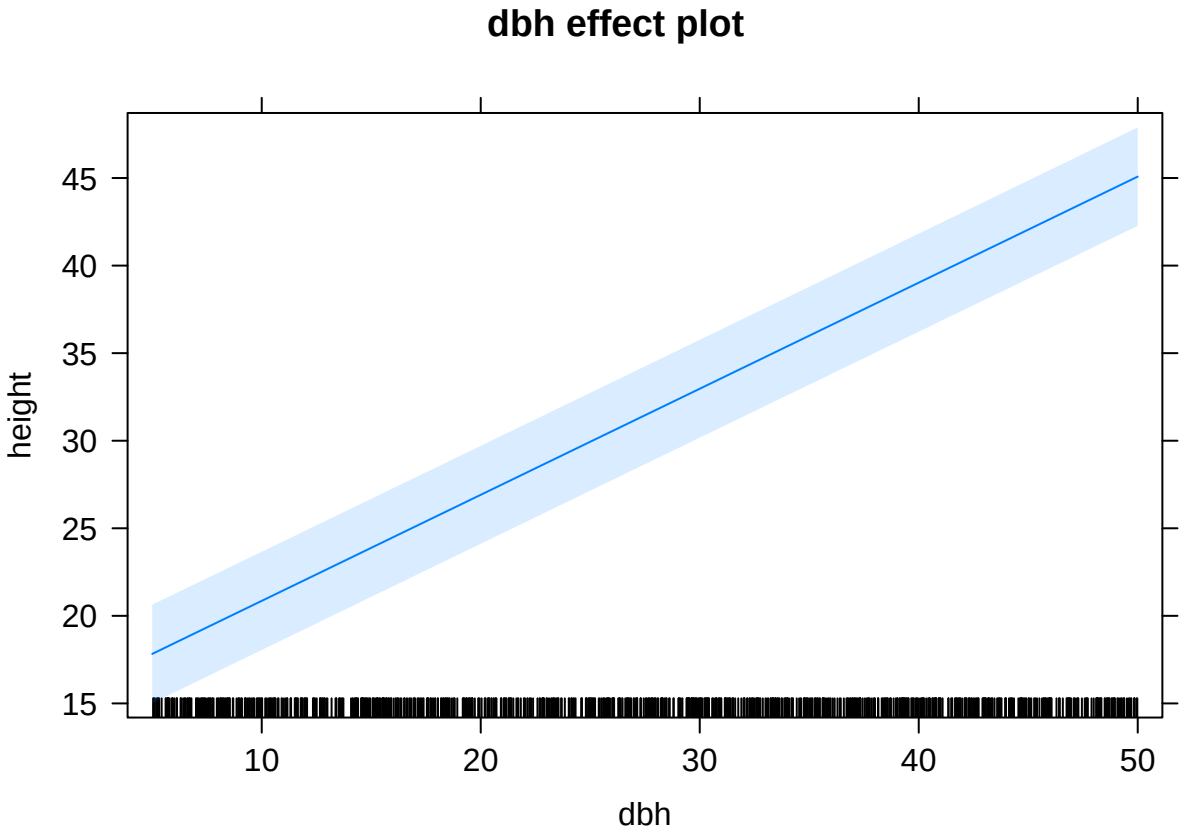
This tells us our fixed effect estimates and their standard errors. The row with (Intercept) has to do with μ_a, while that with dbh has to do with b. Assuming normality, a 95% confidence interval on those coefficients can be obtained with their estimate +/- 1.96 the standard error. In other words approximately estimate +/- 2*SE.

Visualise the effect using `allEffects` and/or `visreg`.

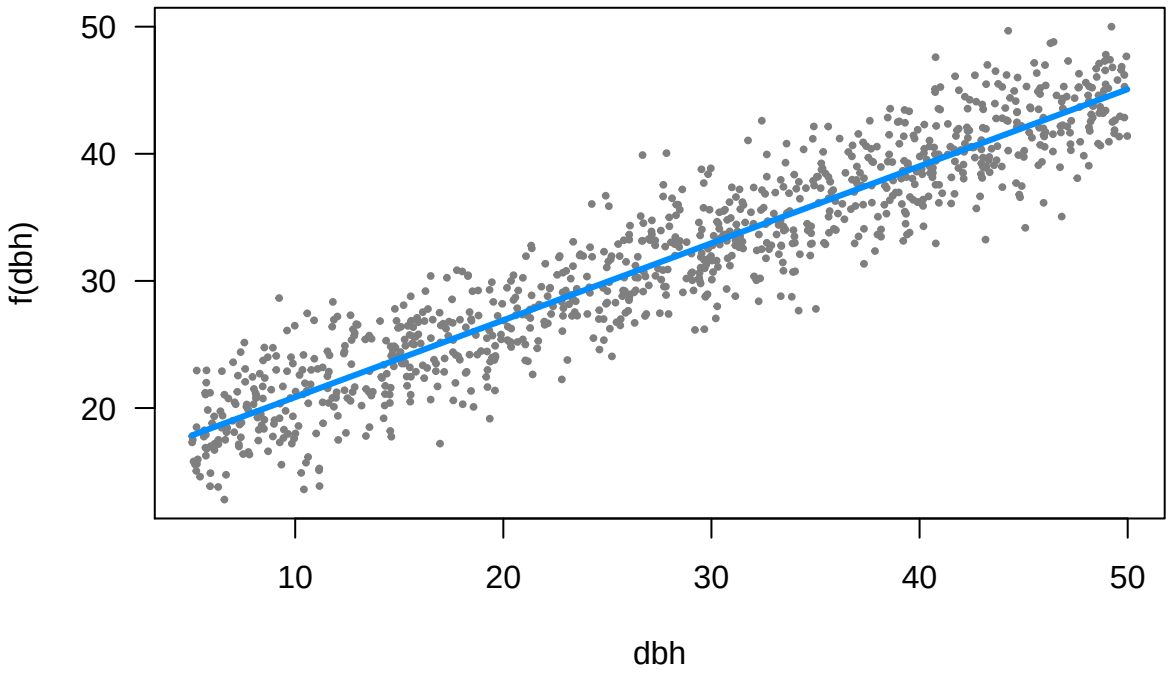
```
effects::allEffects(mixed)
```

```
##  model: height ~ dbh
##
##  dbh effect
## dbh
##        5       20       30       40       50
## 17.82644 26.91126 32.96781 39.02436 45.08091
```

```
plot(effects::allEffects(mixed))
```




```
visreg::visreg(mixed,xvar='dbh')
```

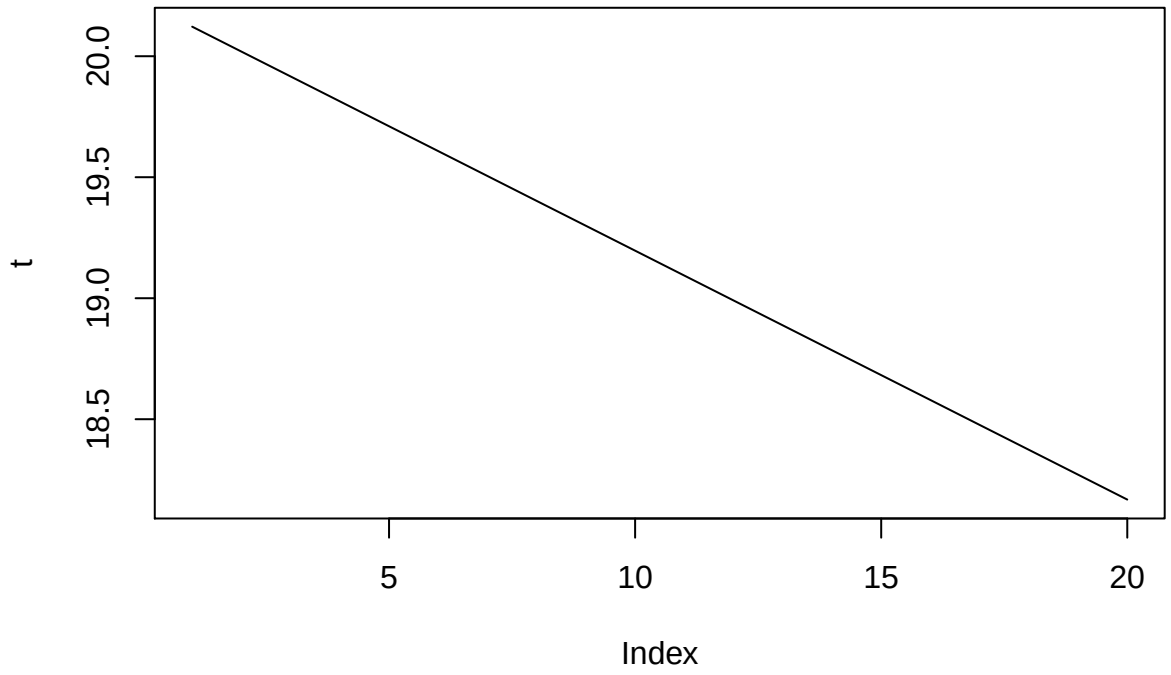


2. Longitudinal study on coral reef and GLMM. A survey of a coral reef uses 10 predefined linear transects covered by divers once every week. The response variable of interest is the abundance of a particular species of anemone as a function of water temperature. Counts of anemones are recorded at 20 regular line segments along the transect. The following piece of code will generate a data set with realistic properties according to the above design. Make sure you understand what it is doing. You might want to explain the script to the colleague next to you.

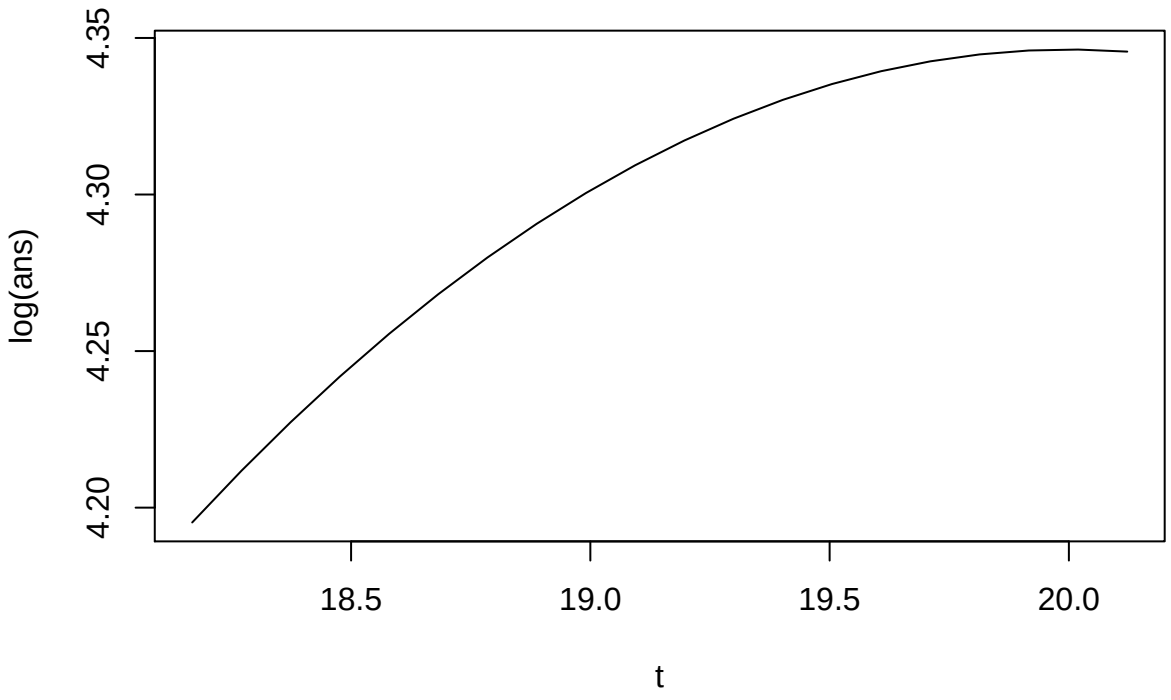
```
transects <- 10
data <- NULL
for (tr in 1:transects){
  ref <- rnorm(1,0,.5) # random effect (intercept)
  t <- runif(1, 18,22) + runif(1,-.2,0.2)*1:20 # water temperature gradient
  ans <- exp(ref -14 + 1.8 * t - 0.045 * t^2) # Anemone gradient (expected response)
  an <- rpois(20, ans) # Actual observations: counts on 20 segments of the current transect
  data <- rbind(data,cbind(rep(tr, 20), t, an))
}
```

str(data) To try and make sense of the code of others, it is always good to plot and/or run small sections of the code.

```
ref <- rnorm(1,0,.5) # random effect (intercept)
t <- runif(1, 18,22) + runif(1,-.2,0.2)*1:20 # water temperature gradient
plot(t,type='l')
```




```
ans <- exp(ref -14 + 1.8 * t - 0.045 * t^2) # Anemone gradient (expected response)
plot(t,log(ans),type='l')
```



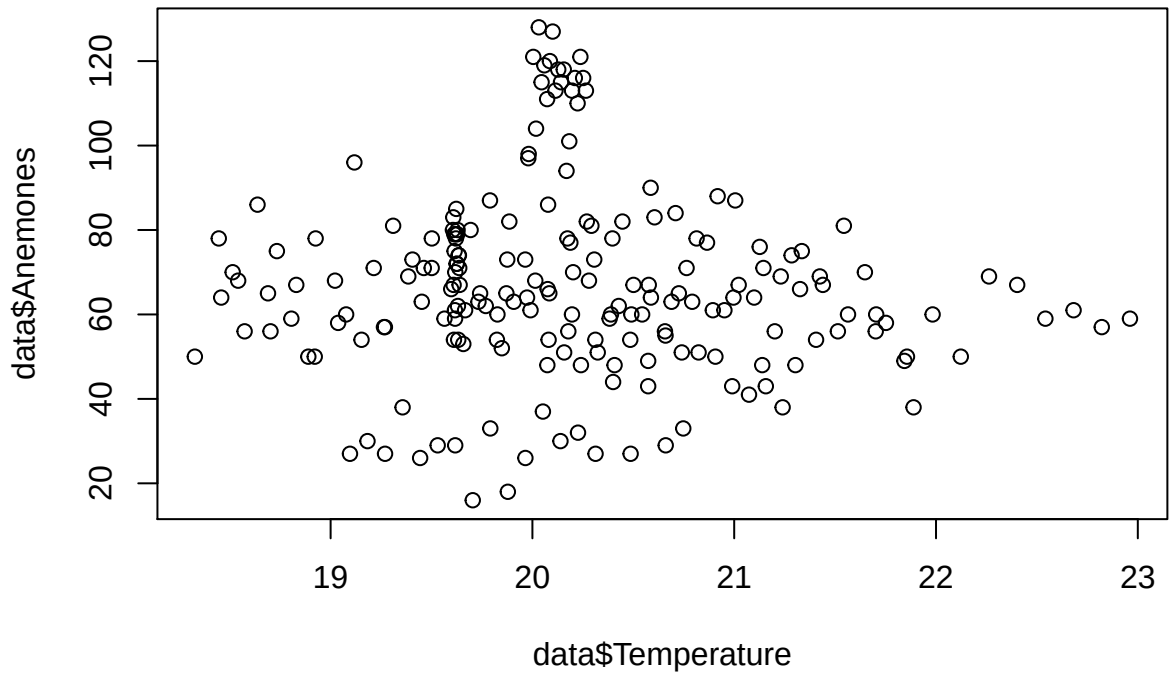
Generate a data set using the anemone code and fit to it the following mixed effects models:

- m1: GLM with temperature
- m2: GLM with quadratic formula in temperature
- m3: GLMM with temperature and random intercept

- m4: GLMM with quadratic temperature and random intercept

Carry out model selection from this set of models. Inspect the temperature effect with `visreg`.

```
data <- data.frame(Transect=data[,1],Temperature=data[,2],Anemones=data[,3])
plot(data$Temperature, data$Anemones)
```




```
library(lme4)
m1 <- glm(Anemones ~ Temperature, data=data, family=poisson)
m2 <- glm(Anemones ~ Temperature + I(Temperature^2), data=data, family=poisson)
m3 <- glmer(Anemones ~ Temperature + (1 | Transect), data=data, family=poisson)
m4 <- glmer(Anemones ~ Temperature + I(Temperature^2) + (1 | Transect), data=data, family=poisson)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00612482 (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly unide
##  - Rescale variables?;Model is nearly unidentifiable: large eigenvalue ratio
##  - Rescale variables?
```

```
AIC(m1,m2,m3,m4)
```

```
##    df      AIC
## m1  2 2667.187
## m2  3 2632.702
## m3  3 1462.475
## m4  4 1438.617
```

There seems to be issues of convergence, and the warning message is quite clear in that we should standardize our Temperature covariate. Let's do it then:

```
data$Temp <- (data$Temperature - mean(data$Temperature))/sd(data$Temperature)
head(data)
```

```
##   Transect Temperature Anemones      Temp
## 1        1    18.45788       64 -1.961158
## 2        1    18.57377       56 -1.831337
## 3        1    18.68965       65 -1.701516
## 4        1    18.80553       59 -1.571695
## 5        1    18.92141       50 -1.441874
## 6        1    19.03729       58 -1.312053
```

A column Temp has been added to our dataframe.

```
library(lme4)
m1 <- glm(Anemones ~ Temp, data=data, family=poisson)
m2 <- glm(Anemones ~ Temp + I(Temp^2), data=data, family=poisson)
m3 <- glmer(Anemones ~ Temp + (1 | Transect), data=data, family=poisson)
m4 <- glmer(Anemones ~ Temp + I(Temp^2) + (1 | Transect), data=data, family=poisson)
AIC(m1,m2,m3,m4)
```

```
##    df      AIC
## m1  2 2667.187
## m2  3 2632.702
## m3  3 1462.475
## m4  4 1438.617
```

The model ranking is the same, and we do no longer get warnings. Let's have a look to the relationship between the number of anemones and water temperature.

```
visreg::visreg(m4,xvar='Temp')
```