# R Demonstration –Logistic Regression

**Objective:** The purpose of this week's session is to demonstrate how to perform a logistic regression model in R and OpenBUGS. We will calculate a logistic regression of number of reproductive structures against plant survival.

Download the data of a *Hypericum cumulicola* study as spreadsheet text file (`hypericum_survival.txt`) from the course website and save it in your `PCB6466` folder. You may also wish to download the script `Logistic_Regression 2013.R`.

## Part I. Preparing the data

Start the R software. From the menu bar, select *File→Change dir…* and then browse to the `PCB6466` folder on the Desktop. Enter the following commands to load and attach the *Hypericum cumulicola* data. This dataset contains two continuous predictor variables (`height` and `rep_structures`) and a binary response variable (`survival`, where 0 = dead and 1 = alive). For this demo we will focus on the regression of number of reproductive structures vs. survival but feel free to explore more complex models featuring height as well.

```
orig_data <- read.table("Hypericum_survival.txt",header=T)
attach(orig_data)
```

We will constrain the data to concentrate only on the fate of reproductive individuals:

```
survival <- survival[rep_structures>0]
rep_structures <- rep_structures[rep_structures>0]
```

## Part II. Logistic regression models

The logistic model for *p* the probability of a variable as a function of x is given by:

$$\boxed{p = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}}}$$

The trick for linearizing the logistic model is a transformation known as logit (see the class website for an in depth explanation), whose errors have a binomial distribution:

$$\boxed{\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X}$$

We will now build logistic regression models to evaluate the effect of number of reproductive structures on the probability of survival. We use the function `glm` indicating `family binomial` (general linear models with binomial errors) and call its summary to inspect our estimates:

```
model1 <- glm(survival ~ rep_structures, family=binomial)
summary(model1)
```

```
Call:
glm(formula = survival ~ rep_structures, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6413  -1.2929   0.7883   0.8484   1.9309

Coefficients:
                 Estimate Std. Error z value Pr(>|z|)
(Intercept)     1.0493333  0.1183184   8.869  < 2e-16 ***
rep_structures -0.0036073  0.0006762  -5.335 9.56e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 734.95  on 569  degrees of freedom
Residual deviance: 700.61  on 568  degrees of freedom
AIC: 704.61
Number of Fisher Scoring iterations: 4
```

We now calculate a $G^2$ statistic from the deviance values to assess the overall significance of the model and get its p-value by comparing it to the Chi-square distribution. We reject the null hypothesis with p=4.65e-09.

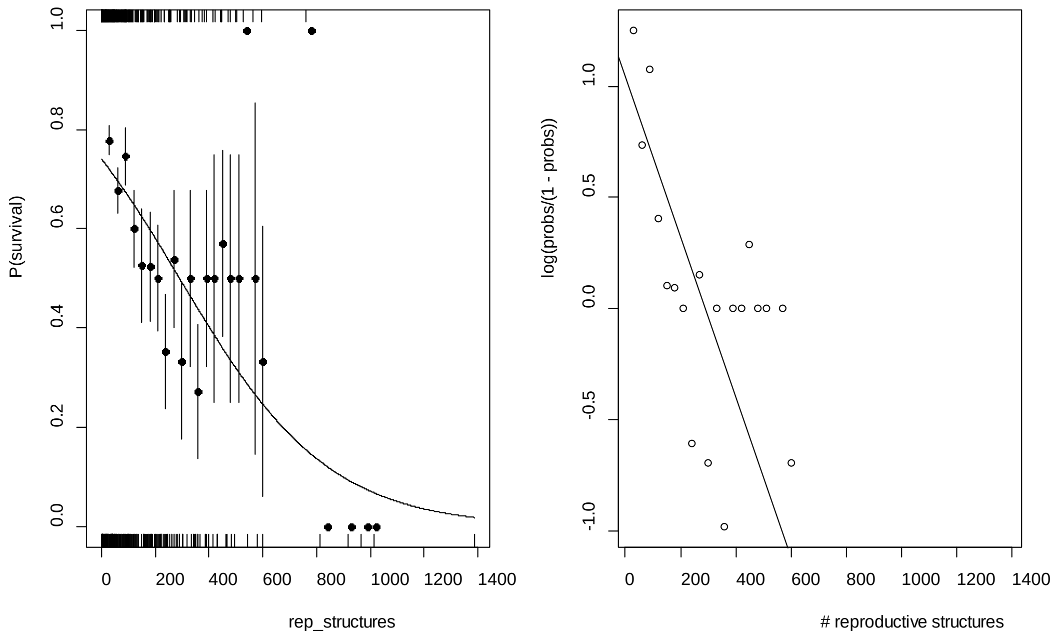
```
G_sq <- model1$null.deviance – model1$deviance
pchisq(G_sq, 1, lower.tail=F)
```

We can plot this model in two different ways. First we plot it in its natural scale; as you can see in the y-axis, the probability of survival is bounded by its logical scale of 0 to 1, but the resulting function is not linear.

```
par(mfrow=c(1,2))
x_values_rs <- seq(0, max(rep_structures), 1)
y_values_rs <- predict(model1, list(rep_structures=x_values_rs),
type="response")
plot(rep_structures, survival,type="n",ylab="P(survival)")
rug(jitter(rep_structures[survival==0]),ticksize = 0.03,)
rug(jitter(rep_structures[survival==1]),ticksize = 0.03, side=3)
lines(x_values_rs, y_values_rs)
b <- seq(0, max(rep_structures),50)
z <- cut(rep_structures, b)
prebyden <- tapply(survival,z,sum)
tab <- table(z)
probs <-prebyden/tab
probs <- as.vector(probs)
points(b[2:length(b)],probs,pch=16,cex=1)
se2<-sqrt(probs*(1-probs)/tab)
up2 <-probs+as.vector(se2)
down2 <-probs-as.vector(se2)
for (i in 1:length(b-1)){
 lines(c(b[i+1],b[i+1]),c(up2[i],down2[i]))}
```

A different option (and easier to code) is to plot the response variable in its logit scale; where it is no longer a straightforward probability, but where it behaves linearly. We used heuristic (arbitrary) arrangements of the data to illustrate the different fits.

```
coef <-model1$coefficients
plot(b[2:length(b)],log(probs/(1-probs)),xlab="# reproductive
structures")
abline(a=coef[1],b=coef[2])
```



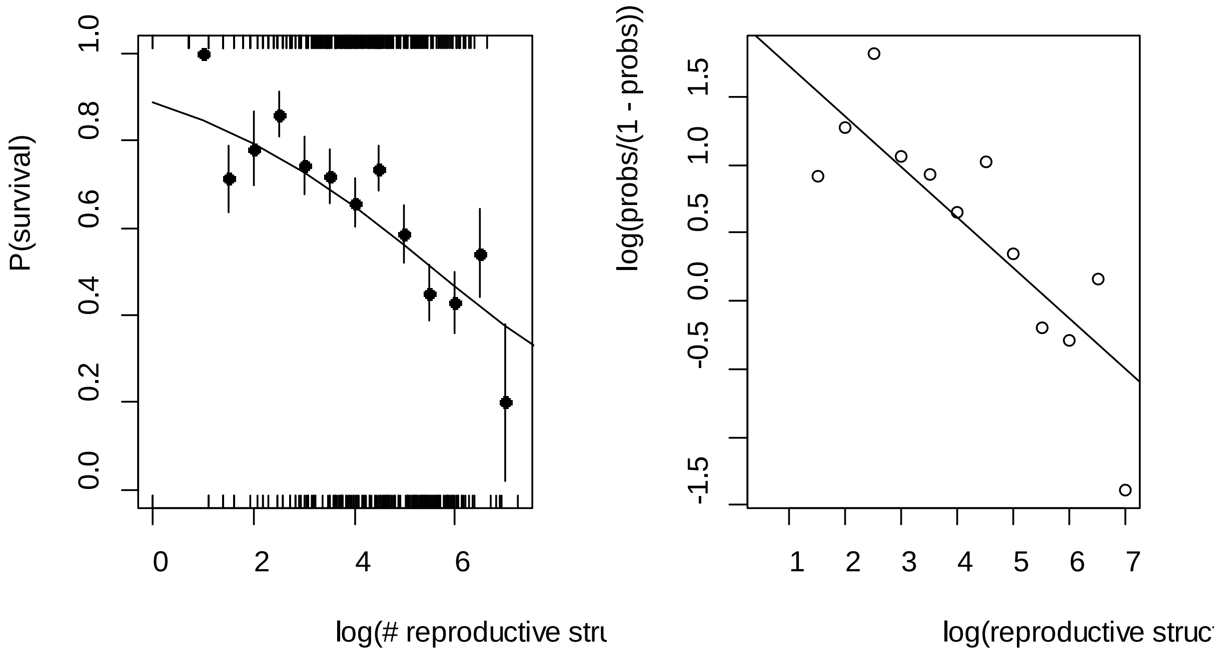
However, we are not satisfied with the distribution of the residuals in this model. The scarce information for plants with many reproductive structures has too much leverage and is pulling the model down, predicting very low survival values for plants with > 400 reproductive structures even if plants with intermediate numbers of reproductive structures have survival probabilities around 0.5. This model predicts a probability of survival of 0.32 for plants with 500 reproductive structures.

```
       rs <- 500
       coeffs <- summary(model1)$coefficients
       odds_ratio <- coeffs[1,1]  + coeffs[2,1]*rs
       prob <- 1/(1 + (1/exp(odds_ratio)))
       prob
[1] 0.31988
```

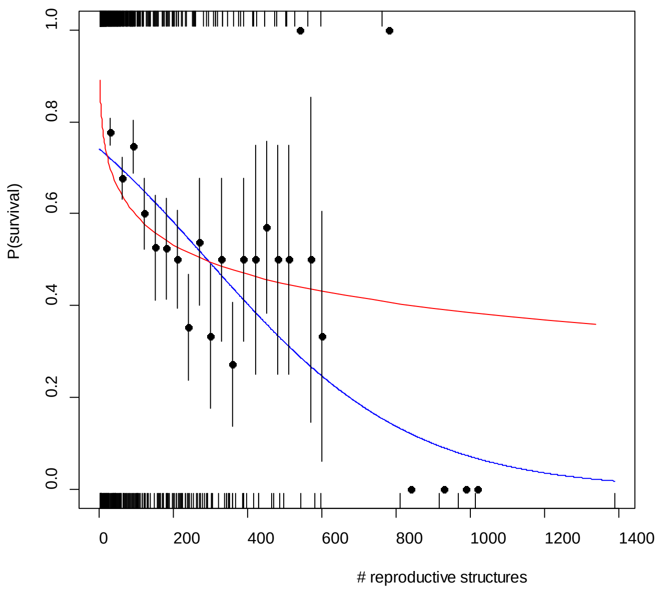
We will now try a model where the number of reproductive structures has been log-transformed and see if that improves the fit and assumptions (we include only the output, but the R script features the code written in exactly the same way as for model 1):

```
rep <- log(rep_structures)
model2 <-glm(survival ~ rep, family=binomial)
summary(model2)
```

```
Call:
glm(formula = survival ~ rep, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.1012  -1.2201   0.7246   0.9424   1.3358

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.09096    0.27783   7.526 5.23e-14 ***
rep         -0.37016    0.06466  -5.724 1.04e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 734.95  on 569  degrees of freedom
Residual deviance: 698.51  on 568  degrees of freedom
AIC: 702.51

Number of Fisher Scoring iterations: 4
```



The new model has better residuals and allows us to conclude that the decline in survival with number of reproductive structures is larger for plants with fewer reproductive structures, and then decreases with increasing number of reproductive structures. We can visualize the differences between the two models even better by plotting them together (code included in R script, model 1 in blue, model 2 in red):



The transformed model predicts a probability of survival of 0.45 for plants with 500 reproductive structures. This value is more consistent with the overall data. We also calculate the AIC for both models and decide that model 2 is better to describe the variation in survival probability.

```
       rs <- log(500)
       coeffs <- summary(model2)$coefficients
       odds_ratio <- coeffs[1,1]  + coeffs[2,1]*rs
       prob <- 1/(1 + (1/exp(odds_ratio)))
       prob
[1] 0.4478278

AIC(model1,model2)
       df      AIC
model1  2 704.6129
model2  2 702.5136
```

We can also use the `predict` function to produce an estimate of the survival probability of a plant with 500 flowering structures:

```
new_data <- data.frame(list(rep=log(500)))
predict(model2, newdata=new_data, type="response")
        1
0.4478278
```

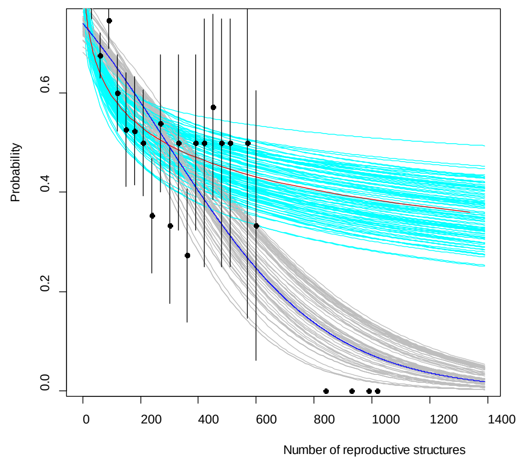
## Part III. Logistic regression with a Bayesian approach

Next, we implement a Bayesian analysis of both models 1 and 2. We use the function `library (R2OpenBUGS)` to connect with OpenBUGS. To define the model below we provide three sets of data: `n` the number plants, `x` their reproductive structures (non or log-transformed), and `y` their survival. The following lines of code describe model 1, the uninformed prior distributions, and give the initial conditions and MCMC simulation parameters. We use this model to compare with the frequentist approach and to calculate the predicted survival for plants with 500 flowering structures. To evaluate model 2 we just change `rep_structures` for `rep` and 500 for log(500). You may notice from the output that the estimates for the parameters are not identical as usual. You can minimize this by drastically increasing the thinning rate, but that will also dramatically increase the time it takes to run! Also notice that the mean of `prob_est` coincides with the predictions of the frequentist approach, but now we have predicted distributions and not simply parameters.

```
##non-transformed (model 1)
library(R2OpenBUGS)
n <- NROW(rep_structures)
x <- rep_structures
y <-survival

# Write model
Logmodel<-function()
 {
  for( i in 1 : n ) {
    y[i] ~ dbern( mu[i] )
    mu[i] <- 1/(1+exp(-( b0 + inprod( b1 , x[i] ))))
                    }
  b0 ~ dnorm( 0 , 1.0E-12 )
  b1 ~ dnorm( 0 , 1.0E-12 )
      prob_est <- 1/(1 + (1/exp(b0 + b1*500)))
      }
write.model(Logmodel, "Logmodel.txt")

# Bundle data
win.data <- list("x","y","n")

# Inits function
inits <- function(){ list(b0=0.1, b1=0.1)}

# Parameters to estimate
params <- c("b0","b1","prob_est")

# MCMC settings
nc = 3
ni=10000
nb=1000
nt=1

# Start Gibbs sampler
out <- bugs(data = win.data, inits = inits, parameters = params, model
= "logmodel.txt",
n.thin = nt, n.chains = nc, n.burnin = nb, n.iter = ni, codaPkg=T)

library(coda)
reg.coda<-read.bugs(out)
results<-summary(reg.coda)
results
```

We now visualize the differences between the two models by calculating and plotting the Bayesian credibility intervals (gray and cyan) together with the frequentist models (model 1 in blue, model 2 in red):

```
estim1 <- estim2 <- array(0,c(140,100))
x <- seq(1,1400,10)
s1 <- round(runif(100)*5000,0)
s2 <- round(runif(100)*5000,0)
for(i in 1:100){
for(j in 1:140){
estim1[j,i]   <-   1/(1   +   (1/exp(as.numeric(reg.coda1[s1[i],1][1])   +
as.numeric(reg.coda1[s1[i],2][1])*x[j])))
estim2[j,i]   <-   1/(1   +   (1/exp(as.numeric(reg.coda2[s2[i],1][1])   +
as.numeric(reg.coda2[s2[i],2][1])*log(x[j]))))
                }
                        }
plot(x_values_rs,  y_values_rs,  col="blue",  type="l",xlab  ="Number  of
reproductive structures", ylab ="Probability")
for (i in 1:100){
lines(exp(log(seq(1,1400,10))),estim1[,i],type="l",col="gray")
lines(seq(1,1400,10),estim2[,i],type="l",col="cyan")
}
points(exp(x_values_lrs), y_values_lrs, col="red",type="l")
points(x_values_rs, y_values_rs, col="blue", type="l")
b <- seq(0, max(rep_structures),30)
z <- cut(rep_structures, b)
prebyden <- tapply(survival,z,sum)
tab <- table(z)
probs <-prebyden/tab
probs <- as.vector(probs)
points(b[2:length(b)],probs,pch=16,cex=1)
se2<-sqrt(probs*(1-probs)/tab)
up2 <-probs+as.vector(se2)
down2 <-probs-as.vector(se2)
for (i in 1:length(b-1)){
 lines(c(b[i+1],b[i+1]),c(up2[i],down2[i]))}
```



As always, we finish our session by detaching the data we attached at the beginning of our R session:

```
detach(orig_data)
```