Exercises for linear mixed effect models, part 3

Timothée Bonnet

April 15, 2019

Contents

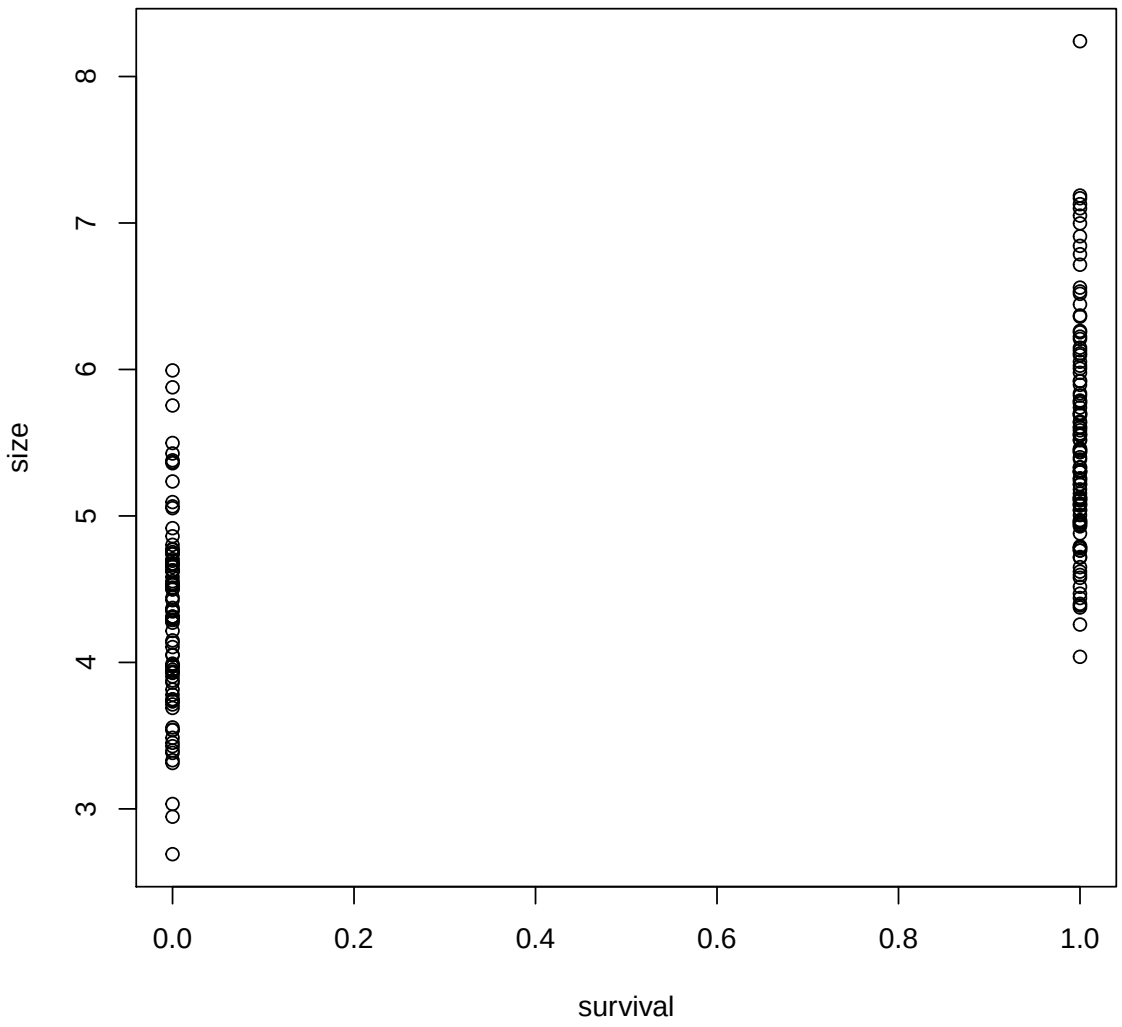
1 Logistic regression basics

* Exercise 1 Fit with glm()

Load the dataset `survivalsize.csv`. It contains fake data of individual-based measurements of body size and of survival from the time of measurement to the next year. Look at a summary of the data and plot them. Do you think size affects survival? Use the function glm() to fit a logistic regression. What should the `family` argument be? What is the direction of the effect of size on survival?

Answer of exercise 1

```
survdat <- read.csv("survivalsize.csv")
summary(survdat)

##     survival          size
##  Min.   :0.000   Min.   :2.69
##  1st Qu.:0.000   1st Qu.:4.37
##  Median :1.000   Median :4.94
##  Mean   :0.555   Mean   :4.99
##  3rd Qu.:1.000   3rd Qu.:5.57
##  Max.   :1.000   Max.   :8.24

plot(survdat)
```


size
3
4
5
6
7
8
0.0
0.2
0.4
0.6
0.8
1.0
survival

```r
glmsurvsize0 <- glm(survival ~ 1 + size, data = survdat,
                    family = "binomial")
summary(glmsurvsize0)
```

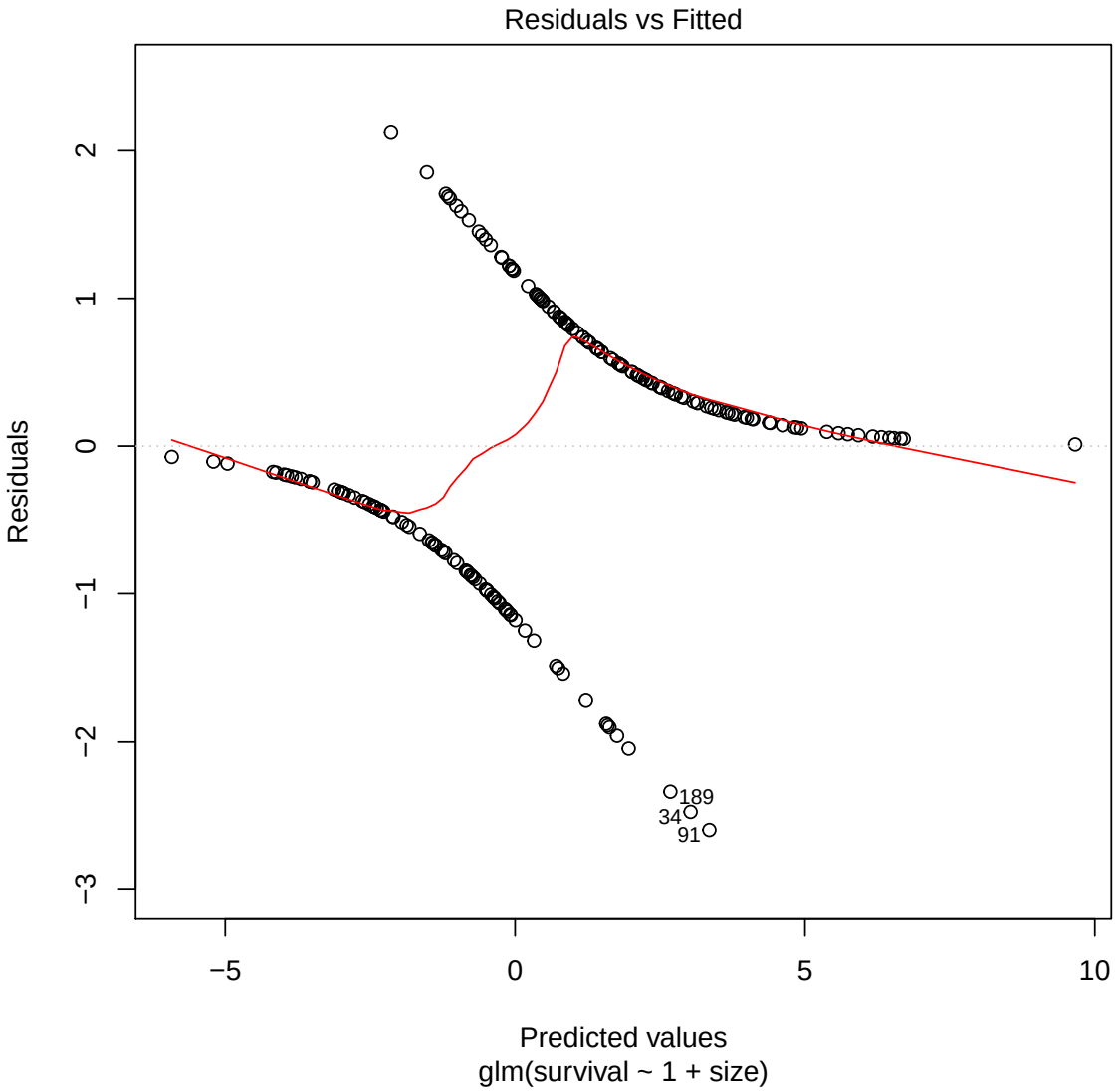
```
##
## Call:
## glm(formula = survival ~ 1 + size, family = "binomial", data = survdat)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.602  -0.606   0.108   0.641   2.122
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -13.478      1.949   -6.92  4.6e-12 ***
## size           2.808      0.402    6.99  2.7e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 274.83  on 199  degrees of freedom
## Residual deviance: 159.02  on 198  degrees of freedom
## AIC: 163
##
## Number of Fisher Scoring iterations: 6
```

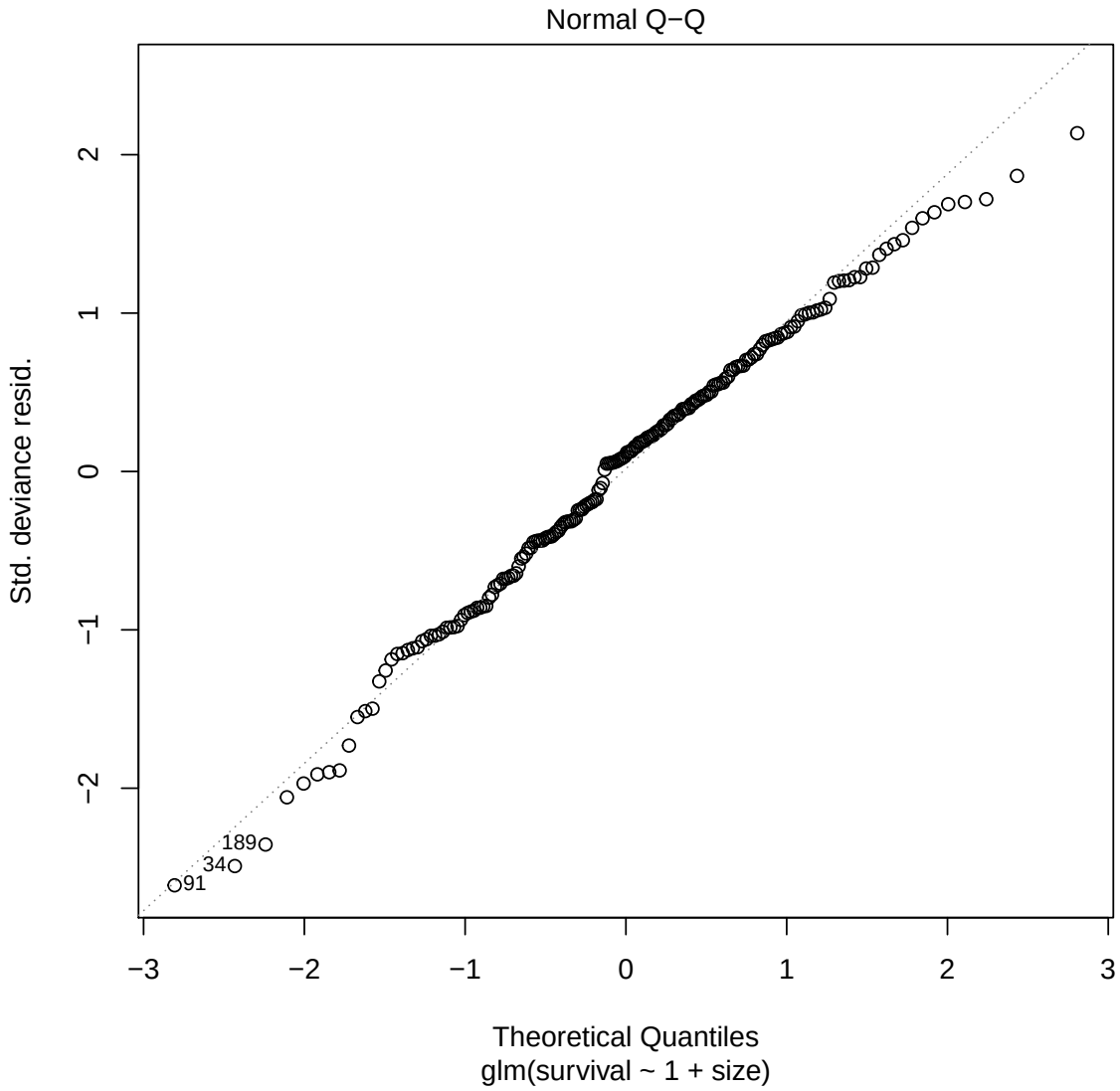
* Exercise 2 Model assumptions

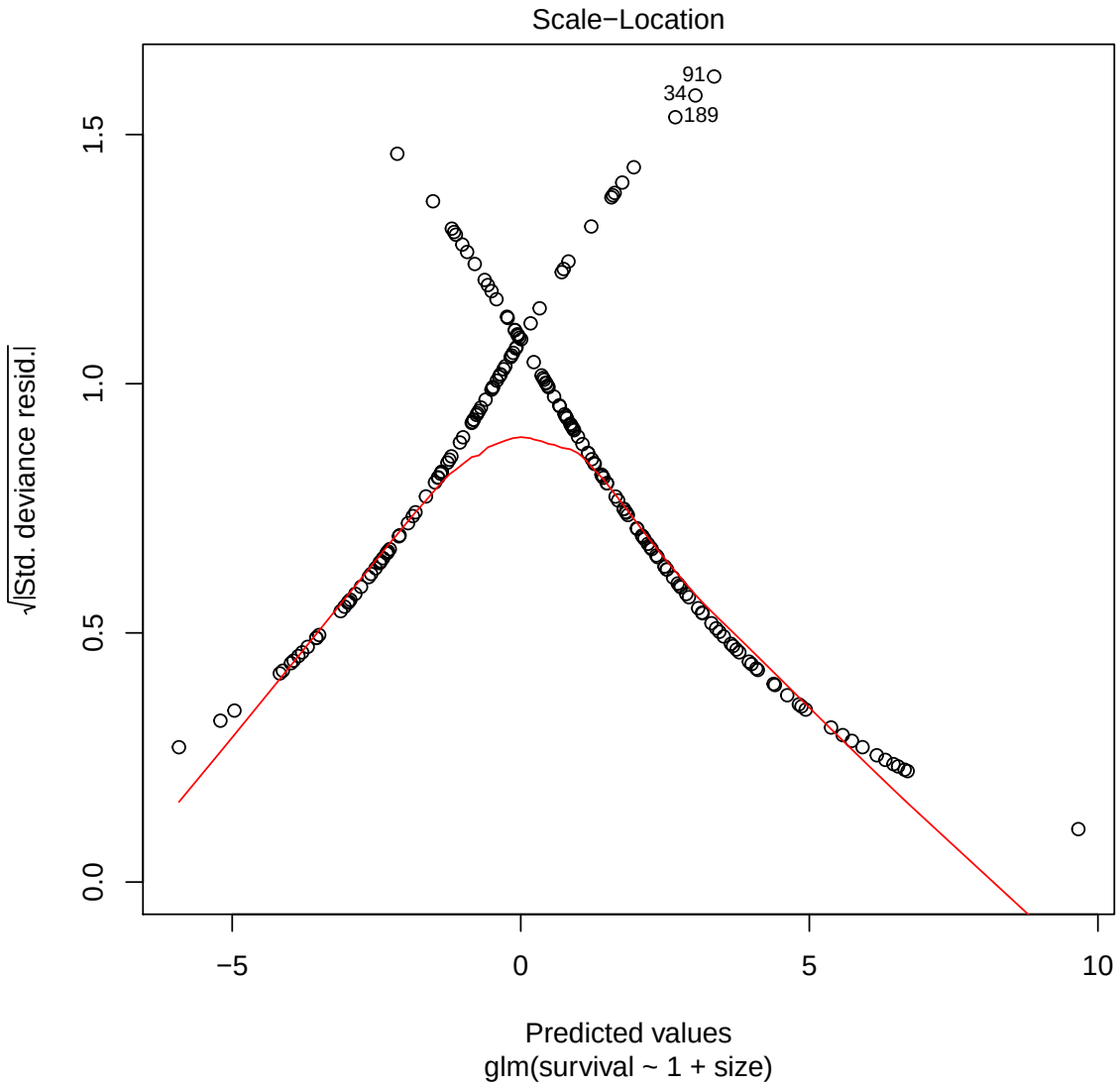
In R some model assumptions of linear models are routinely checked using plot(lm()): residual normality, independance and homogeneous variance, and legerage. If you know about these diagnostics (and what the plots should ideally look like) check them for your glm. Should you worry?

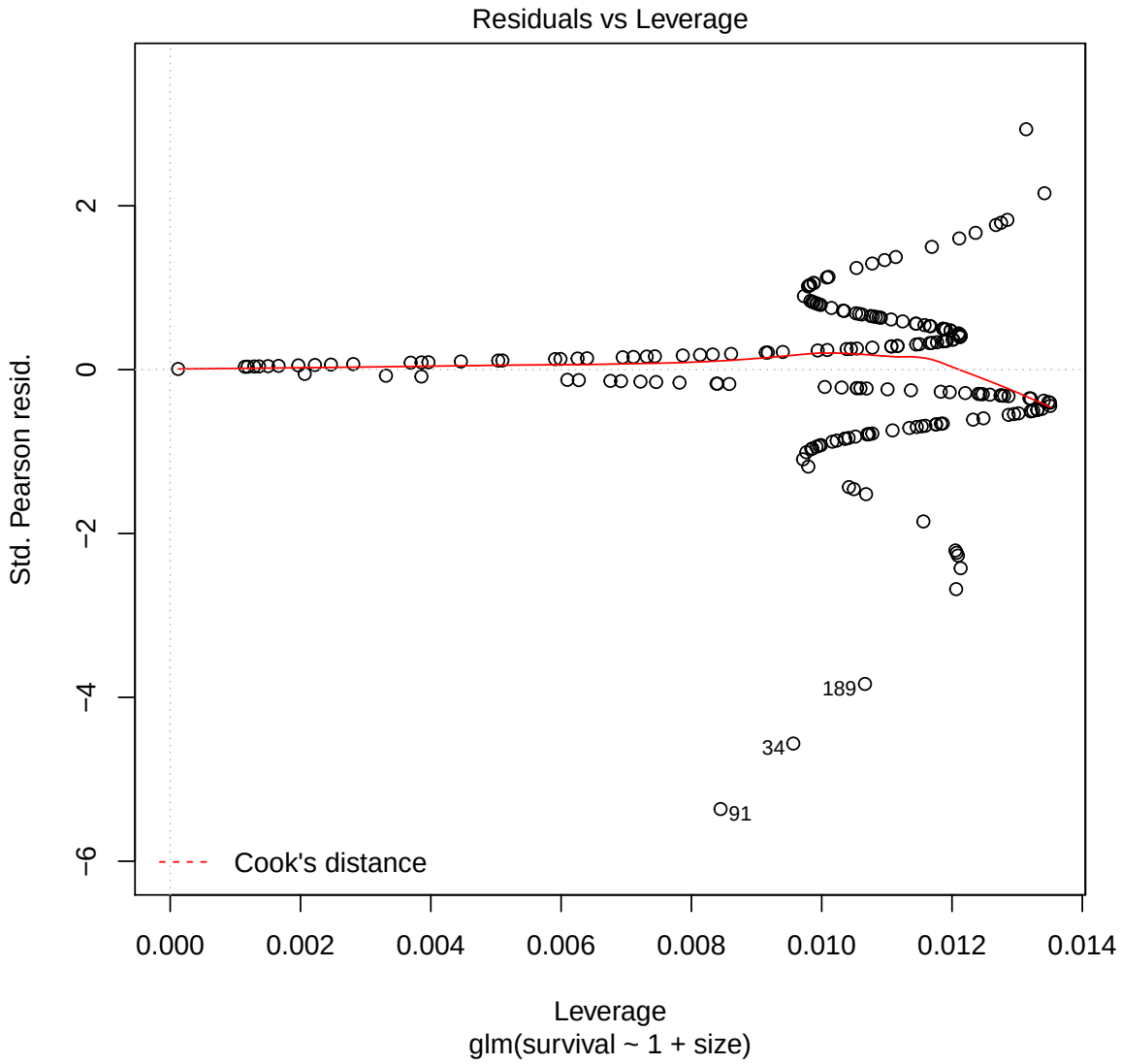
Answer of exercise 2

```r
plot(glmsurvsize0)
```


Residuals vs Fitted
Residuals
Predicted values
glm(survival ~ 1 + size)
189
34
91
2
1
0
−1
−2
−3
−5
0
5
10


Normal Q−Q
Std. deviance resid.
Theoretical Quantiles
glm(survival ~ 1 + size)
189
34
91
−3
−2
−1
0
1
2
3


Scale−Location
91
34
189
√|Std. deviance resid.|
0.0
0.5
1.0
1.5
−5
0
5
10
Predicted values
glm(survival ~ 1 + size)



No. A binary logistic regression makes much less stringent assumptions than a linear model. The one thing you can easily check if that data are binary. The assumptions that may be violated (non-independance and no measurement error) would generally not be visible on these plots. The definition of a logistic regression means that the residuals will generally not follow a normal distribution, but will be dependent on the predictors, and will have changing variance; that is perfectly okay.

** Exercise 3 Making sense of model estimates

Consider the model estimates for the intercept and slope. How to recover the mean survival from these two numbers? You will need to use the back-transformation inverse logit: probability $= \frac{1}{1+\exp(-y)}$ where y is a model prediction on the logit-scale. You can also use the function `predict()`.

How much does the ratio of surviving/dying probabilities changes for an individual of size 6 compared to an individual of size 5?

Answer of exercise 3

```
summary(glmsurvsize0)

##
## Call:
## glm(formula = survival ~ 1 + size, family = "binomial", data = survdat)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.602  -0.606   0.108   0.641   2.122
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -13.478      1.949   -6.92  4.6e-12 ***
## size           2.808      0.402    6.99  2.7e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 274.83  on 199  degrees of freedom
## Residual deviance: 159.02  on 198  degrees of freedom
## AIC: 163
##
## Number of Fisher Scoring iterations: 6
```

The data mean survival is:

```
mean(survdat$survival)

## [1] 0.555
```

You CANNOT recover the mean data survival by back-transforming the predicted mean on the logit-scale, because the back-transformation is not linear:

```
meanlogit <- mean(coefficients(glmsurvsize0)[1]+
coefficients(glmsurvsize0)[2]*survdat$size)

1/(1+exp(-meanlogit))

## [1] 0.6311
```

You must back-transform individual data points and THEN calculate the mean:

```
preddat <- coefficients(glmsurvsize0)[1]+
coefficients(glmsurvsize0)[2]*(survdat$size)
mean(1/(1+exp(-preddat)))

## [1] 0.555
```

The function predict with `type` argument set to `response` can help with the back-transformation:

```
mean(predict(glmsurvsize0, type = "response"))

## [1] 0.555
```

The increase in the odd survive with an increase of one unit in size is an odd-ratio. It is simply the exponential of the slope:

```
exp(coefficients(glmsurvsize0)[2])

##  size
## 16.57
```

You can check this by comparing two predictions

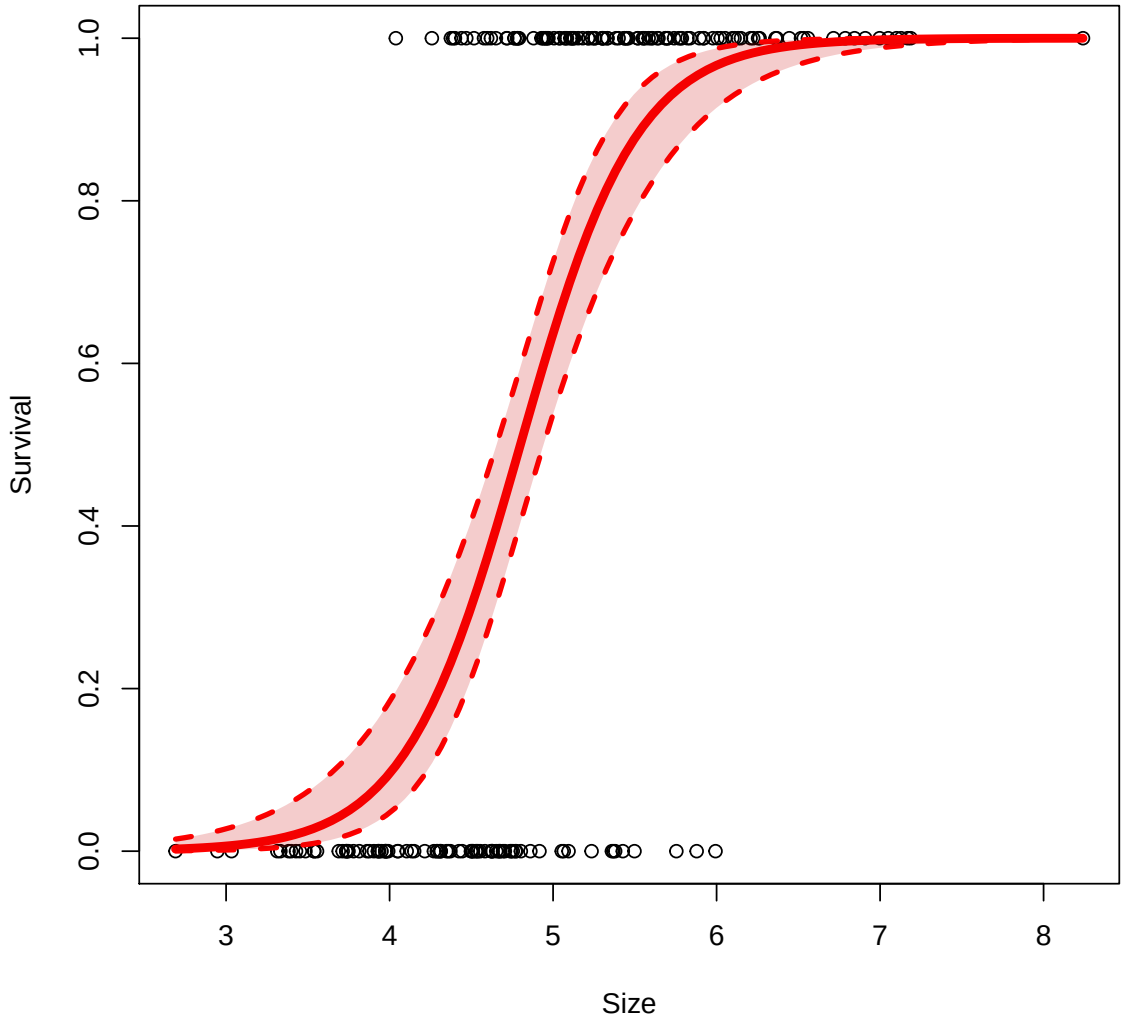
```
survprob5 <- 1/(1+exp(-(coefficients(glmsurvsize0)[1]+
                          5*coefficients(glmsurvsize0)[2])))
survprob6 <- 1/(1+exp(-(coefficients(glmsurvsize0)[1]+
                          6*coefficients(glmsurvsize0)[2])))

#definition of odd-ratio:
(survprob6/(1-survprob6))/(survprob5/(1-survprob5))

## (Intercept)
##       16.57
```

** Exercise 4 Visualize results

Use the function `predict()` to make a graph of the relationship between survival and size, with a confidence interval.

Answer of exercise 4

```r
plot(survdat$size, survdat$survival, xlab="Size", ylab="Survival")

newdat <- data.frame(size=seq(min(survdat$size),max(survdat$size),
                              length.out = 100))
newdat <- cbind(newdat, predict(glmsurvsize0, newdata = newdat,
                                type = "link", se.fit = TRUE))
newdat$ciu <- 1/(1+exp(-(newdat$fit + 1.96*newdat$se.fit)))
newdat$cil <- 1/(1+exp(-(newdat$fit - 1.96*newdat$se.fit)))
newdat$prob <- 1/(1+exp(-(newdat$fit)))

lines(x = newdat$size, newdat$prob, lwd=5, col='red')
#confidence interval with lines
lines(x = newdat$size, newdat$ciu, lwd=3, col='red', lty=2)
lines(x = newdat$size, newdat$cil, lwd=3, col='red', lty=2)

#or with polygon:
polygon(x=c(newdat$size,rev(newdat$size)),
        y=c(newdat$ciu,rev(newdat$cil)),
        col = rgb(0.8,0,0,0.2), border =NA)
```



2 Classification

** Exercise 5 What populations are at risk?

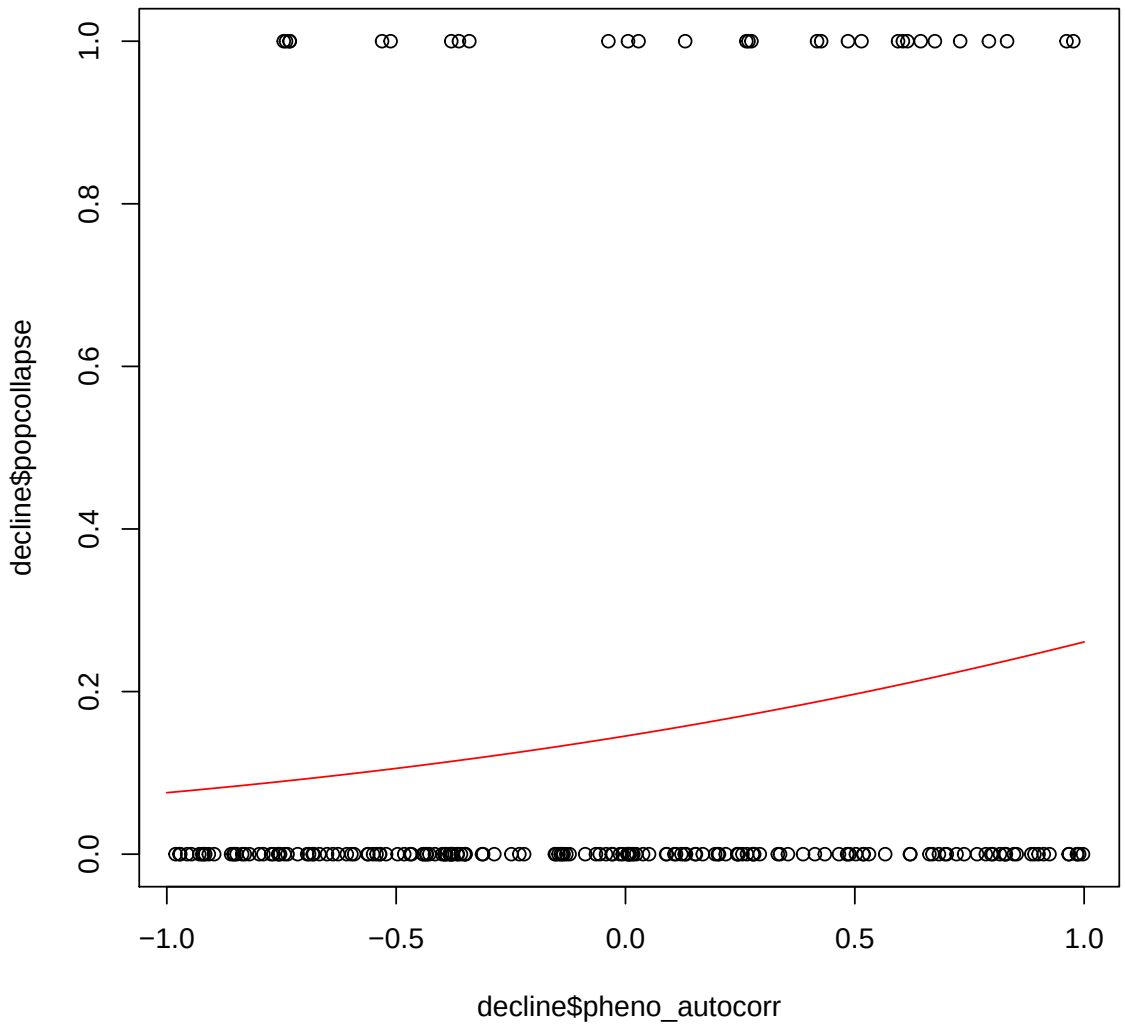
Researchers are currently trying to predict animal population collapses in advance, by finding metrics that changed before historic populations collapsed. Let's imagine we have measured the time-autocorrelation in phenotypes (what that means does not matter) in 200 populations, some of which have collapsed in the past decades. We want to konw whether autocorrelation is related to collapse, and whether it could be used to predict which population will collapse in the future.

Load the dataset `decline.csv`, it contains fake data of past collapses and past auto-

correlations in populations, as well as future collapses (yes, I am a fortune teller). Fit a logistic regression of `popcollapse` as a function of `atcor`. Does `atcor` increase the probability of population collapse? Visualize the relationship. Does it make sense to define a threshold of autocorrelation and allocate conservation resources to those populations that are at risk based on our metric? For instance, what do you think about conserving all populations above an autocorrelation of 0.5? How many future at-risk populations would be missed? How many populations that are not at risk would receive unecessary resources? How could you improve the prediction?

Answer of exercise 5

```
decline <- read.csv("decline.csv")
summary(glmcoll0 <- glm(popcollapse~pheno_autocorr,
                        data = decline, family = "binomial"))

##
## Call:
## glm(formula = popcollapse ~ pheno_autocorr, family = "binomial",
##     data = decline)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -0.777  -0.611  -0.497  -0.429   2.196
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -1.772      0.206   -8.61   <2e-16 ***
## pheno_autocorr    0.732      0.353    2.07    0.038 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 169.08  on 199  degrees of freedom
## Residual deviance: 164.64  on 198  degrees of freedom
## AIC: 168.6
##
## Number of Fisher Scoring iterations: 4

newdat <- data.frame(pheno_autocorr=seq(-1,1,length.out = 100))
newdat <- cbind(newdat, pred = predict(glmcoll0,
                          type = "response", newdata = newdat))

plot(decline$pheno_autocorr, decline$popcollapse)
lines(newdat$pheno_autocorr, newdat$pred, col="red")
```



Using a threshold of 0.5 would be correct for 5 at-risk populations:

```
sum(decline$futurecollapse[decline$pheno_autocorr>=0.5],
    na.rm=TRUE)

## [1] 5
```

and 116 safe populations:

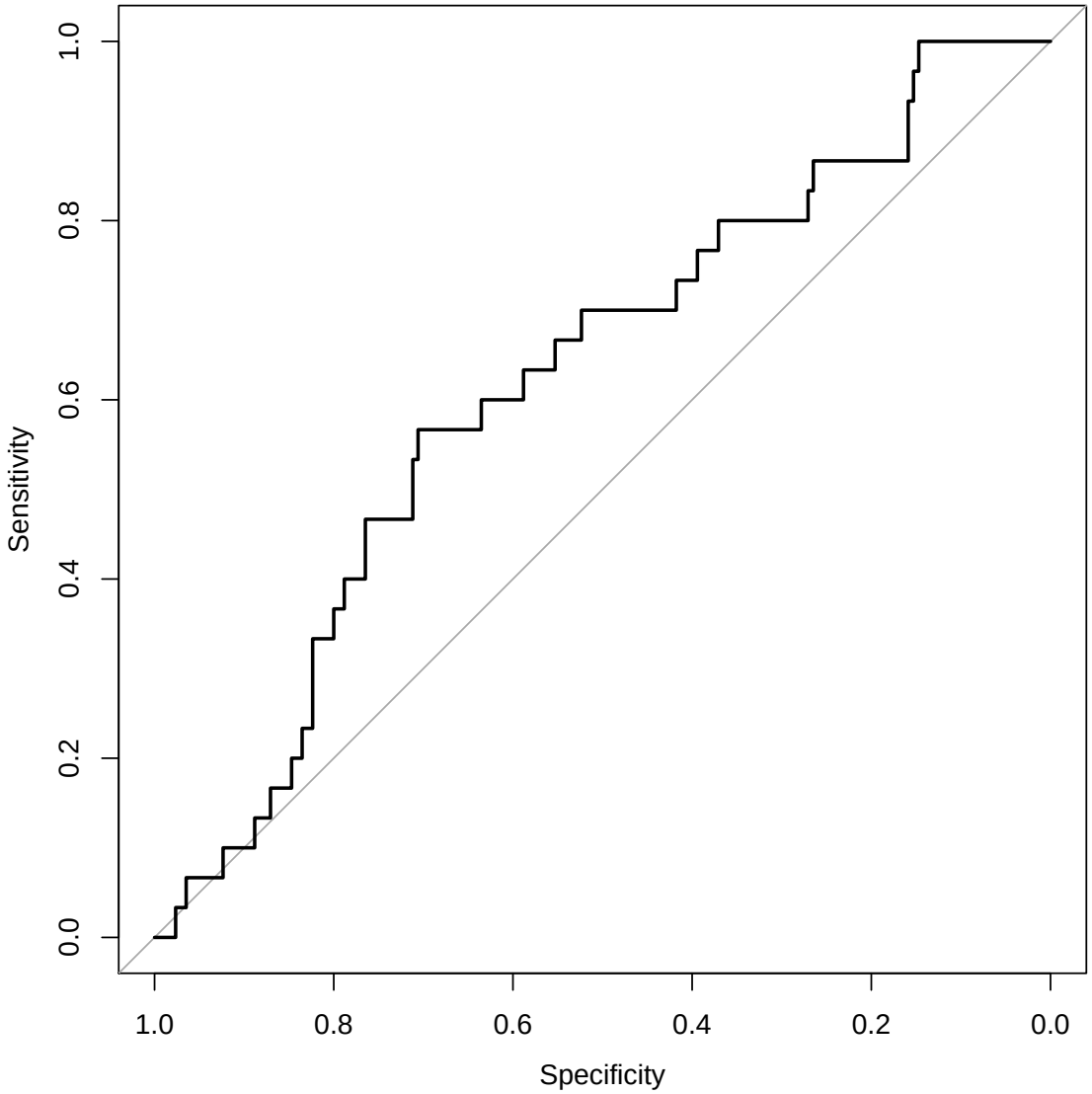
```
sum(1-decline$futurecollapse[decline$pheno_autocorr<0.5],
    na.rm=TRUE)

## [1] 116
```

But would be incorrect for 19 at-risk populations and 30 safe populations:

```
sum(decline$futurecollapse[decline$pheno_autocorr<0.5],
    na.rm=TRUE)

## [1] 19

sum(1-decline$futurecollapse[decline$pheno_autocorr>=0.5],
    na.rm=TRUE)

## [1] 30
```

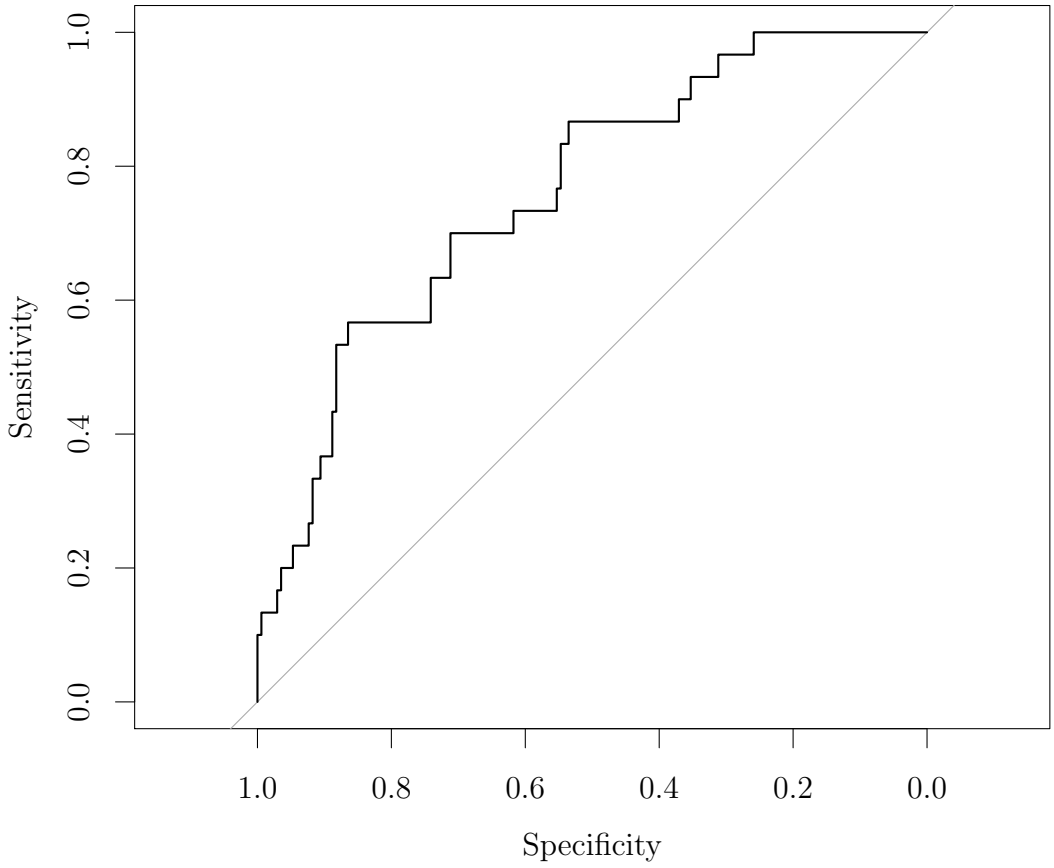
As could have been expected, this is a pretty bad performance. The result can be obtained more formally looking at a Receiver Operating Characteristic plot:

```
library(pROC) #need to install this package!

## Type 'citation("pROC")' for a citation.
##
## Attaching package:  'pROC'
## The following objects are masked from 'package:stats':
##
##    cov, smooth, var

plot.roc(predictor = predict(glmcoll0, type = "response"),
         x = decline$popcollapse)
```



Adding another predictor helps improve the prediction:

```
glmcoll1 <- glm(popcollapse~pheno_autocorr+variance,
                data = decline, family = "binomial")

plot.roc(predictor = predict(glmcoll1, type = "response"),
         x = decline$popcollapse)
```



3 Repeatability

** Exercise 6 Repeatability, on what scale?

Load the dataset `runaway.csv`. It contains fake data of a behavioural experiment: you played the trumpet to an animal and see whether it an animal and see whether they ran away. 500 individuals were tested twice a year during 5 years. You measured distance between you and the animal. You would like to know whether individuals behave consistently that is, have a personality and are repeatable. We start by fitting a linear mixed model using the package lme4 (accounting for differences among years, and for the effect of distance):

```
roodat <- read.csv("runaway.csv")
lmm0 <- lmer(RunAway ~ 1 + distance +(1|individual) + (1|year),
             data = roodat)
VarCorr(lmm0)

##  Groups     Name        Std.Dev.
##  individual (Intercept) 0.0663
##  year       (Intercept) 0.0159
##  Residual               0.4139
```

This model suggests a repeatability of:

```
(0.0663^2)/(0.0663^2+0.0159^2+0.4139^2)

## [1] 0.02498
```

That is not a lot, and we wonder whether the small number is due to the lack of fit of the model. Fortunately it is possible to fit a logistic mixed model by changing `lmer` to `glmer` and specifying `family='binomial'`. If you do so, what repeatibility do you find? Where does the difference come from? What is right?

Answer of exercise 6

```
summary(repmod <- glmer(RunAway ~ 1 + distance +(1|individual) +
                          (1|year), data = roodat, family = "binomial"))

## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: binomial  ( logit )
## Formula: RunAway ~ 1 + distance + (1 | individual) + (1 | year)
##    Data: roodat
##
##      AIC      BIC   logLik deviance df.resid
##     5354     5380    -2673     5346     4996
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -2.447  0.430  0.503  0.555  0.771
##
## Random effects:
##  Groups     Name        Variance Std.Dev.
##  individual (Intercept) 0.13740  0.3707
##  year       (Intercept) 0.00513  0.0716
## Number of obs: 5000, groups:  individual, 500; year, 5
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   1.7448     0.1189   14.67   <2e-16 ***
## distance     -0.1621     0.0347   -4.67    3e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr)
## distance -0.903

VarCorr(repmod)

##  Groups     Name        Std.Dev.
##  individual (Intercept) 0.3707
##  year       (Intercept) 0.0716
```

So the repeatability on the logit scale is:

```
0.3707^2/(0.0716^2+0.3707^2)

## [1] 0.964
```

That is much much higher than the LMM estimate (about 2% versus about 96%).

This huge difference comes from two things: first, the logit transformation is not linear, and may distort variation. We can get a taste for that by back-transforming random numbers generated with the random effect variances and fixed effect coefficients. For instance, when `distance=3`, the repeatability on the back-transformed scale is approximately:

```
var(1/(1+exp(-(rnorm(1000000, 1.7448-0.1621*3, 0.3707))))) /
  var(1/(1+exp(-(rnorm(1000000, 1.7448-0.1621*3, 0.0716+0.3707)))))

## [1] 0.7036
```

That is a bit less, but not really small.

Also, notice that the repeatability is not the same for all values of `distance`:

```
var(1/(1+exp(-(rnorm(1000000, 1.7448-0.1621*0, 0.3707))))) /
  var(1/(1+exp(-(rnorm(1000000, 1.7448-0.1621*0, 0.0716+0.3707)))))

## [1] 0.6888

var(1/(1+exp(-(rnorm(1000000, 1.7448-0.1621*6, 0.3707))))) /
  var(1/(1+exp(-(rnorm(1000000, 1.7448-0.1621*6, 0.0716+0.3707)))))

## [1] 0.7109
```

The other important reason why repeatability is so much smaller in the LMM is that our GLMM does not explicitly measures the residual variation! For a logistic regression residual variance is not defined on the transformed scale, instead a Bernouilli random process generates deviations between a model prediction and data. The variation in this process is only a function of the expected probability.

To see what it does, lets draw random Bernouilli numbers around our expected back-transformed values before calculating the total variance:

```
var(1/(1+exp(-(rnorm(1000000, 1.7448-0.1621*3, 0.3707))))) /
  var(sapply(X = rnorm(100000, 1.7448-0.1621*0, 0.0716+0.3707),
        FUN=function(x){rbinom(n = 1, size = 1, prob = 1/(1+exp(-(x))))
                        }))

## [1] 0.03119
```

So, about 3% of the data variation is explained by the individual random effect! That is not a perfectly exact calculation though, because it is only for the value of `distance=3`. To be exact we should integrate individual differences over the range of possible predictor values (that is a bit complicated).

The package QGglmm offers a proper back-transformation of variance components and repeatability on the data scale (denoted $h^2.obs$ below):

```r
library(QGglmm)
QGglmm::QGparams(predict = predict(repmod),var.a = 0.3707^2,
                 var.p = 0.0716^2+0.3707^2, model = "binom1.logit")

## [1] "Computing observed mean..."
## [1] "Computing variances..."
## [1] "Computing Psi..."
##   mean.obs var.obs var.a.obs  h2.obs
## 1   0.7671  0.1787  0.004095 0.02292
```