MATH5743M: Statistical Learning - Lecture 6

Dr Seppo Virtanen, s.virtanen@leeds.ac.uk

Semester 2: 28 Feb 2022

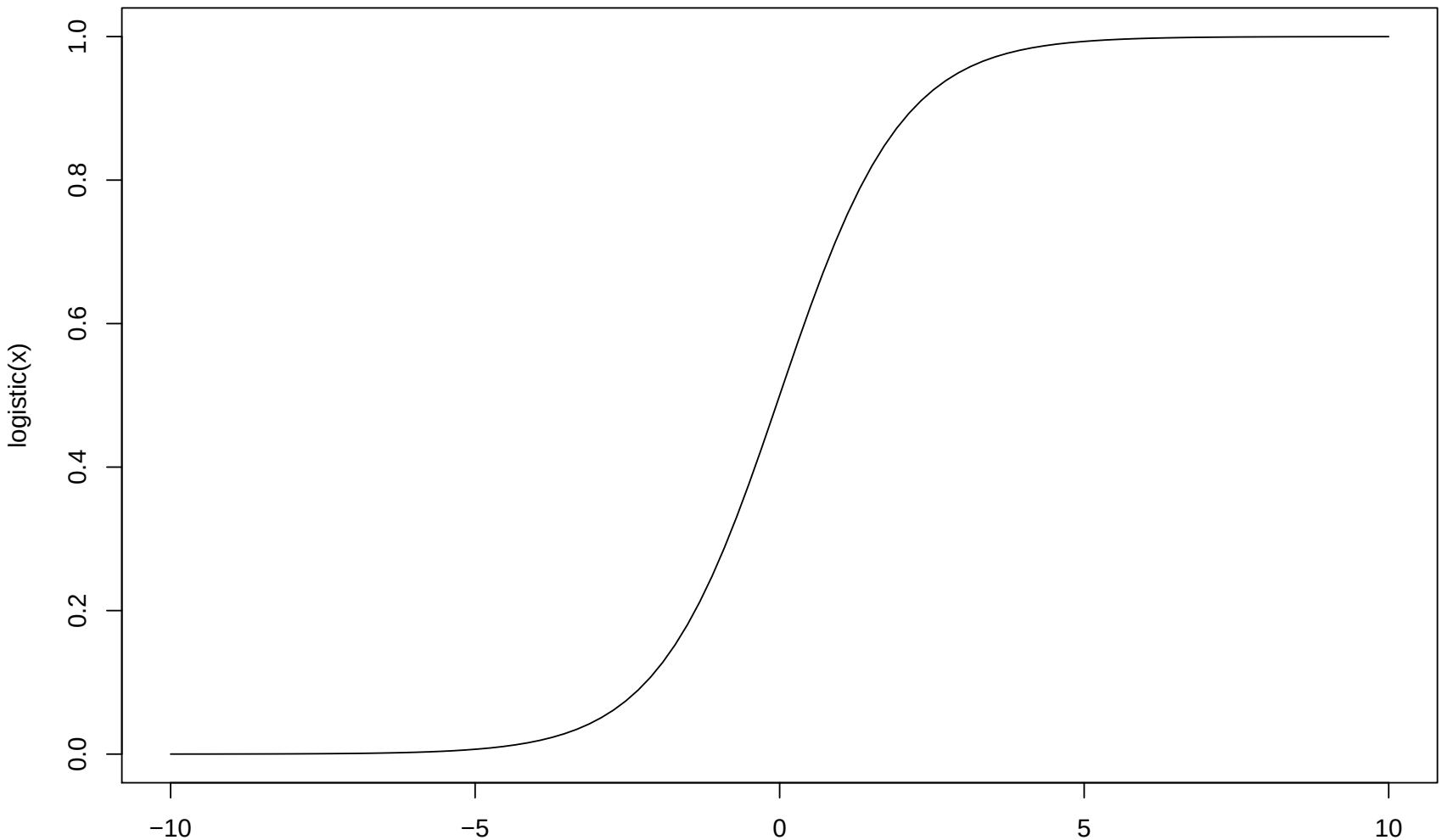
Outline

- Logistic regression: The output can only take two possible values. Model the probability that an output belongs to one of two possible classes
- Family of statistical methods known as the Generalised Linear Model, which gives the tool **glm** its name.

Logistic function

The logistic function, $\phi(x)$, is a *sigmoidal* function $\phi : \mathbb{R} \to [0, 1]$,

$$\phi(x) = \frac{1}{1 + \exp(-x)} \tag{1}$$



The logistic-linear model in one-dimension

Given binary outputs Y and inputs X, the probability that $Y = 1$ is

$$P(Y = 1 \mid X = x) = \phi(\beta_0 + \beta_1 x) \tag{2}$$

and for $Y = 0$ we have

$$P(Y = 0 \mid X = x) = 1 - P(Y = 1 \mid X = x) = 1 - \phi(\beta_0 + \beta_1 x), \tag{3}$$

because Y can only take two values. We assume outputs follow the Bernoulli distribution.

The likelihood

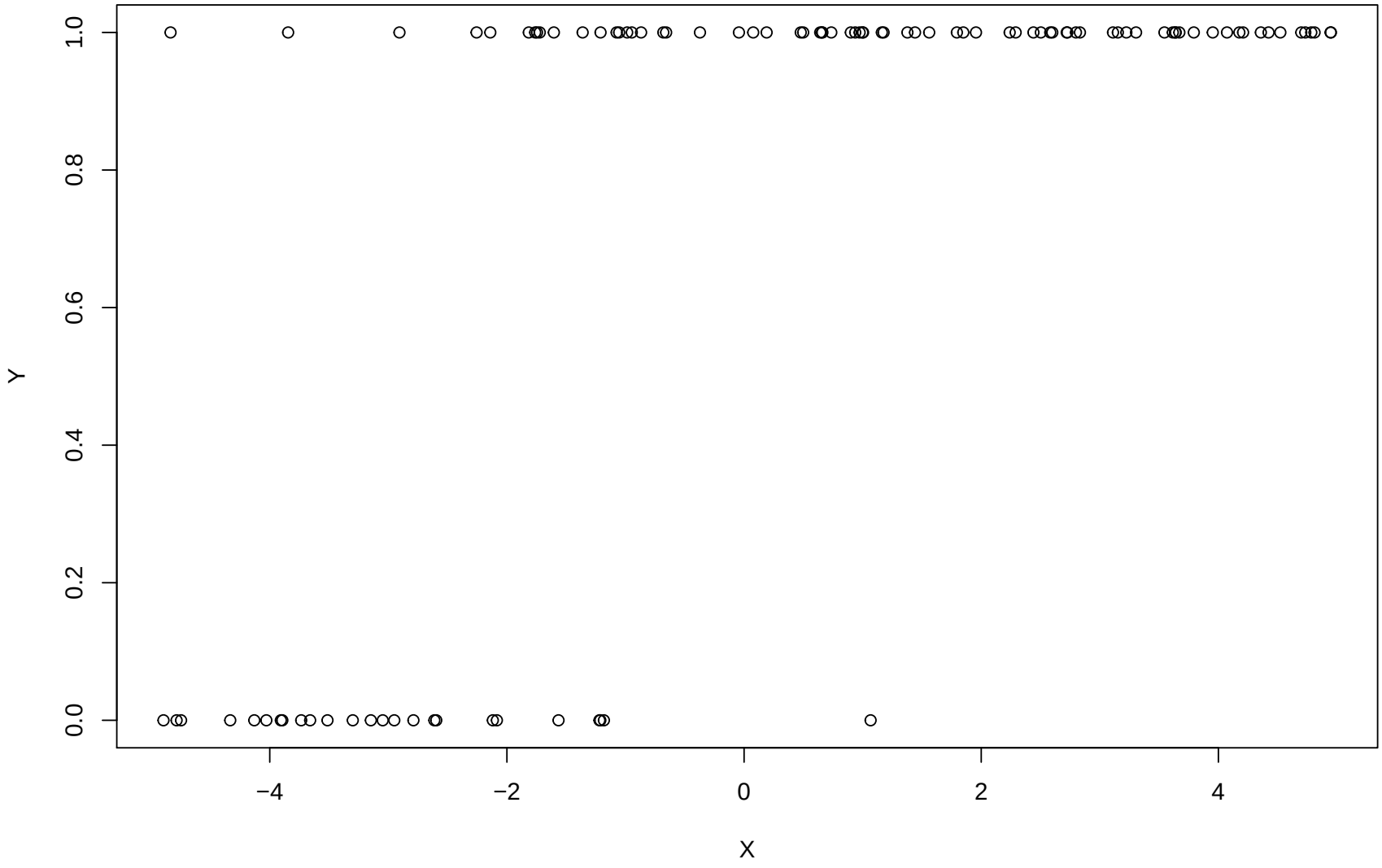
$$\log \mathcal{L}(\beta_0, \beta_1) = \sum_{i=1}^{n} Y_i \log \phi(\beta_0 + \beta_1 X_i) + (1 - Y_i) \log \left(1 - \phi(\beta_0 + \beta_1 X_i)\right). \tag{4}$$

Unlike linear regression, logistic regression does not have a general analytic solution, so we will need to learn the appropriate parameters numerically.

Example

```r
#Define number of data points
N = 100
#Define the logistic function phi
phi <- function(b0, b1, x) 1/(1+exp(-b0-b1*x))
#Generate random values of X between -5 and 5
X = runif(n=N, min=-5, max=5)
#Calculate the probability for each Y to be 1 based on X
P = phi(b0=2, b1= 1, x=X)
#Assign 1's or 0's to Y based on P
Y = as.numeric(runif(N) < P)
```

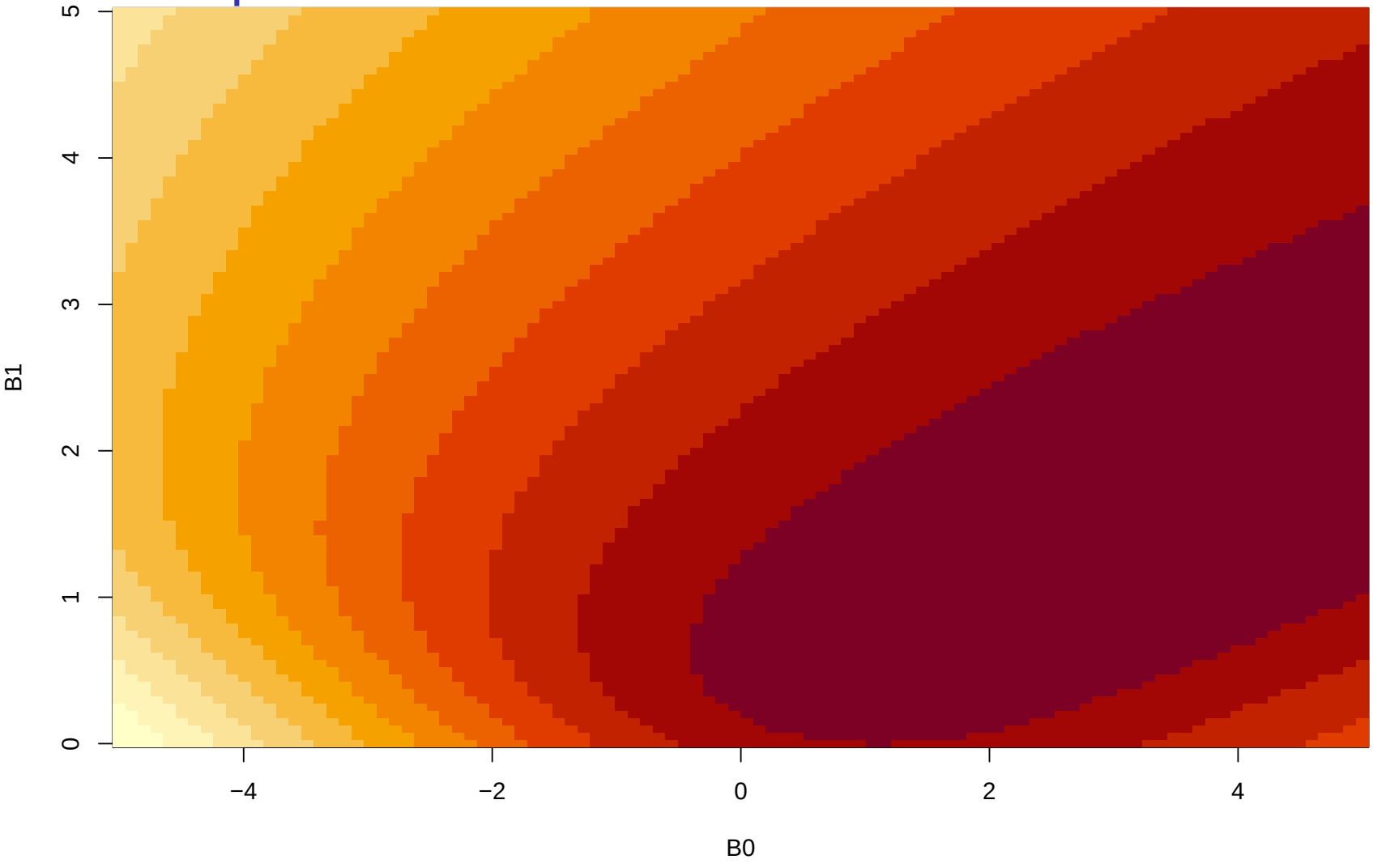
Example



Numerical optimisation

```
#define the possible values of beta0 and beta1 to check.
#Note that it is a good idea to make the number of each di
B0 = seq(-5, 5, length.out=100)
B1 =seq(0, 5, length.out=101)
#define an empty matrix of log-likelihood values
LL = matrix(NA, 100,101)
#for every pair of B0 and B1, evalaute the log-likelihood
for (i in 1:100){
  for (j in 1:101){
    LL[i,j] = sum(Y*log(phi(B0[i],B1[j],X)) +
                  (1-Y)*log(1-phi(B0[i], B1[j],X)))
  }
}
```

Numerical optimisation




```
## [1] "MLE estimate for beta 0 is 2.27272727272727"

## [1] "MLE estimate for beta 1 is 1.05"
```

Use optim in R

```
#Define the negative log-likelihood function
negLL <- function(param){-sum(Y*log(phi(param[1],param[2],X
                        +(1-Y)*log(1-phi(param[1],param[2],X)
start_guess = c(0, 1)
optimResults = optim(par=start_guess, fn=negLL)
print(paste("MLE estimate for beta 0 is", optimResults$par
```

```
## [1] "MLE estimate for beta 0 is 2.33047088889263"
```

```
print(paste("MLE estimate for beta 1 is", optimResults$par
```

```
## [1] "MLE estimate for beta 1 is 1.063542779293"
```

Generalised Linear Model (GLM)

Although the model is non-linear, it includes a *latent* linear function: for example, $\beta_0 + \beta_1 x$. GLMs are a wide class of models where a non-linear transformation (such as the logistic function) is applied to an underlying, latent linear function of the inputs in order to determine the probability of the outputs.

Generalised Linear Model (GLM)

glm {stats} R Documentation

Fitting Generalized Linear Models

Description

`glm` is used to fit generalized linear models, specified by giving a symbolic description of the linear predictor and a description of the error distribution.

Usage

```
glm(formula, family = gaussian, data, weights, subset,
    na.action, start = NULL, etastart, mustart, offset,
    control = list(...), model = TRUE, method = "glm.fit",
    x = FALSE, y = TRUE, contrasts = NULL, ...)

glm.fit(x, y, weights = rep(1, nobs),
        start = NULL, etastart = NULL, mustart = NULL,
        offset = rep(0, nobs), family = gaussian(),
        control = list(), intercept = TRUE)

## S3 method for class 'glm'
weights(object, type = c("prior", "working"), ...)
```

Arguments

`formula`	an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
`family`	a description of the error distribution and link function to be used in the model. For `glm` this can be a character string naming a family function, a family function or the result of a call to a family function. For `glm.fit` only the third option is supported. (See `family` for details of family functions.)
`data`	an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `glm` is called.
`weights`	an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The 'factory-fresh' default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
`start`	starting values for the parameters in the linear predictor.

Figure 1: Help summary for glm command

Generalised Linear Model (GLM)

The *family* argument is new: this specifies the type of GLM we will be using, and the default option of 'Gaussian' gives us standard linear regression.

```
#Make a data frame with the inputs and outputs
mydata = data.frame(X=X, Y=Y)
#Specify a glm
myglm = glm(Y ~ X, family=binomial, data=mydata)
```

Generalised Linear Model (GLM)

```
summary(myglm)
```

```
##
## Call:
## glm(formula = Y ~ X, family = binomial, data = mydata)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.64438  -0.05895   0.10137   0.31567   2.39678
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   2.3306     0.5582   4.175 2.98e-05 ***
## X             1.0637     0.2347   4.532 5.84e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
##
## (Dispersion parameter for binomial family taken to be 1)
```

Confidence intervals in logistic regression

The model summary above shows that **glm** returns much of the same information to us for logistic regression as it does for linear regression. This includes both estimates for the model parameters (the coefficients) and also standard errors for these estimates. Recall from weeks 3-4 that we can use these standard errors to obtain confidence intervals for the parameters - intervals within which we can be confident (to a given percentage value) that the true parameter lies.

Confidence intervals in *linear* regression

In linear regression the standard errors were calculated *exactly* (remember that we derived the analytical expressions for parameter estimation).

Therefore we were able to be very precise in our calculation of the confidence interval, taking note of how many data points we had, and therefore how many degrees of freedom, and finding the required critical value from the t-distribution.

Confidence intervals in logistic regression

In logistic regression by contrast, the standard errors are approximated by assuming that the likelihood is normally distributed. This means that the confidence intervals themselves will be approximate, and we perform a lower precision calculation. Instead of defining the interval based on the number of degrees of freedom and the t-distribution, we assume that the data is *large* and use the appropriate value from the normal distribution (sometimes called the z-distribution) instead (this is the same as the t-distribution for $n \to \infty$). For example, we can construct a 95% CI as:

$$95\%\ \mathrm{CI}: \ \mathrm{Estimate} \pm \mathrm{Standard\ Error} \times z_{2.5\%}$$

Where $z_{2.5\%} = 1.96$ (look this up on the normal distribution table)

Example

```r
#Get critical value of z_2.5% (should be 1.96)
zc = qnorm(0.975)

#Extract estimate and standard error of coefficient from m
estimate = summary(myglm)$coefficients[2,1]
standard_error = summary(myglm)$coefficients[2,2]

#Calculate and print the lower and upper boundaries of the
CI_min = estimate - zc*standard_error; CI_max = estimate +
print(CI_min); print(CI_max)
```

```
## [1] 0.6036656
```

```
## [1] 1.523638
```

Model selection in logistic regression

- Fit the models of interest and record the AIC values. Lower values of AIC indicate that the model will be more useful for prediction
- Perform cross-validation to directly test how well each model predicts new data. Remember to repeat the data splitting many times to check your results. Compare these to the values you get from AIC.

Other Generalised Linear Models

A GLM is any model where the probability of the OUTPUT is a function of some linear combination of the INPUTS. That is:

$$P(Y = y \mid X_1 = x_1, \ldots, X_n = x_n) = f(\beta_0 + \beta_1 x_1 + \ldots + \beta_n x_n) \quad (5)$$

For example, in linear regression, $f()$ was the normal distribution density function, and in logistic regression $f()$ was the logistic function $\phi()$.

- binomial(link = "logit")
- gaussian(link = "identity")
- Gamma(link = "inverse")
- inverse.gaussian(link = "1/mu^2")
- poisson(link = "log")
- quasi(link = "identity", variance = "constant")
- quasibinomial(link = "logit")
- quasipoisson(link = "log")