Quantitative Methods in Linguistics – Lecture 10

Adrian Brasoveanu*

April 20, 2014

Contents

This set of lecture notes is based on Hosmer and Lemeshow (2000), Ramsey and Schafer (2002), Faraway (2006), Wright and London (2009), and Chris Manning's course materials available here `http://www-nlp.stanford.edu/manning/courses/ling289/`.

*These notes have been generated with the 'knitr' package (Xie 2013) and are based on many sources, including but not limited to: Abelson (1995), Miles and Shevlin (2001), Faraway (2004), De Veaux et al. (2005), Braun and Murdoch (2007), Gelman and Hill (2007), Baayen (2008), Johnson (2008), Wright and London (2009), Gries (2009), Kruschke (2011), Diez et al. (2013), Gries (2013).

1 Basic introduction to logistic regression

Logistic regression modeling has become, in many fields, the standard method of analysis when the response variable is categorical (binomial or multinomial).

The goal of an analysis using this method is the same as that of any regression-model building: to find the best fitting, most parsimonious and theoretically reasonable model to describe the relationship between an outcome (dependent or response) variable and a set of independent (predictor or explanatory) variables.

The most common example of modeling is the usual linear regression model where the outcome variable is assumed to be continuous.

What distinguishes a logistic regression model from the linear regression model is that the outcome variable in logistic regression is binary / dichotomous (or n-ary / polytomous).

Once this difference is accounted for, the methods employed in an analysis using logistic regression follow the same general principles used in linear regression.

For example, consider a data set discussed in Hosmer and Lemeshow (2000):

- one continuous predictor: the age of the 100 participants in the study, given in years (AGE)
- a categorical response: presence or absence of evidence of significant coronary heart disease (CHD), coded as 1 (there was evidence) and 0 (there was no evidence)

```
> chage <- read.table("chdage.dat", header = F)
> head(chage)

  V1 V2 V3
1  1 20  0
2  2 23  0
3  3 24  0
4  5 25  1
5  4 25  0
6  7 26  0

> str(chage)

'data.frame': 100 obs. of  3 variables:
 $ V1: int  1 2 3 5 4 7 6 9 8 10 ...
 $ V2: int  20 23 24 25 25 26 26 28 28 29 ...
 $ V3: int  0 0 0 1 0 0 0 0 0 0 ...

> cat(readLines("chdage.txt", n = -1), fill = 20)

Code Sheet for the Chd-Age data in Table 1.1 page 3 of
Applied Logistic Regression:Second Edition

Variable Name Values
     1 Identification Code 1-100
     2 Age Years
     3 Evidence of Coronary
Heart Disease 0 = No, 1 = Yes

> chage <- data.frame(chage$V2, chage$V3)
> names(chage) <- c("AGE", "CHD")
> head(chage)
```

```
  AGE CHD
1  20   0
2  23   0
3  24   0
4  25   1
5  25   0
6  26   0

> summary(chage)

      AGE            CHD
 Min.   :20.0   Min.   :0.00
 1st Qu.:34.8   1st Qu.:0.00
 Median :44.0   Median :0.00
 Mean   :44.4   Mean   :0.43
 3rd Qu.:55.0   3rd Qu.:1.00
 Max.   :69.0   Max.   :1.00

> write.csv(chage, "chage.csv", row.names = FALSE)
> attach(chage)
```

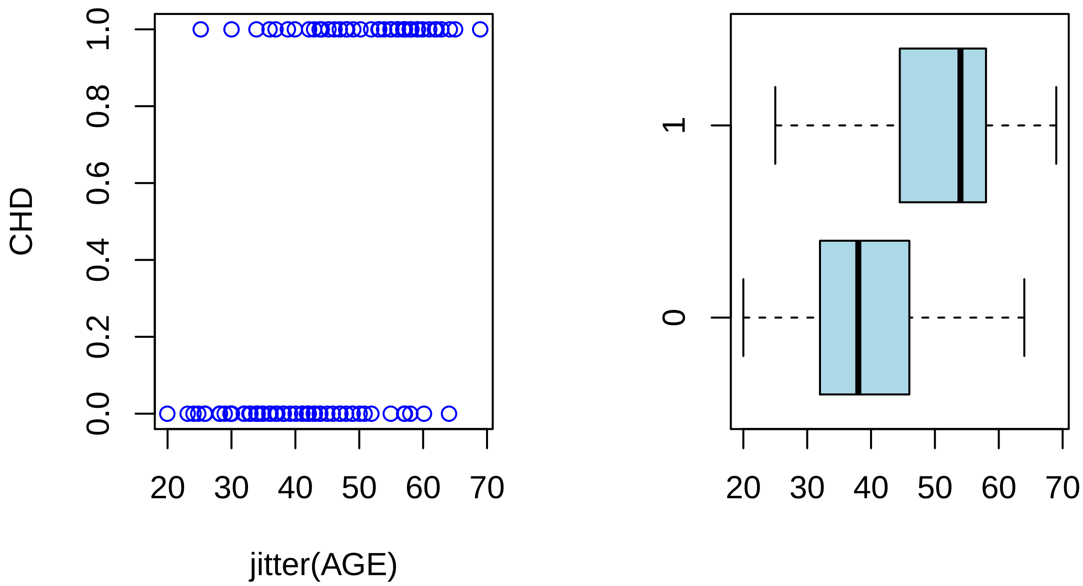
Goal: explore the relationship between age and the presence or absence of CHD in this sample and generalize this relationship from the sample to the population.

- had our outcome variable been continuous rather than binary, we probably would begin by forming a scatterplot of the outcome versus the independent variable
- we would use this scatterplot to provide an impression of the nature and strength of any relationship between the outcome and the independent variable.

A scatterplot of the data, together with a boxplot:

- all points fall on one of two parallel lines representing the absence of CHD ($y = 0$) and the presence of CHD ($y = 1$)
- there is some tendency for the individuals with no evidence of CHD to be younger than those with evidence of CHD; we can see this with a boxplot too

```
> par(mfrow = c(1, 2))
> plot(jitter(AGE), CHD, col = "blue")
> boxplot(AGE ~ CHD, col = "lightblue", horizontal = TRUE)
```




```
> par(mfrow = c(1, 1))
```

The scatterplot depicts the dichotomous nature of the outcome variable clearly, but it does not provide a clear picture of the nature of the relationship between CHD and AGE:

- the variability in CHD at all ages is large and this makes it difficult to describe the functional relationship between AGE and CHD

One common method of removing some variation while still maintaining the structure of the relationship between the outcome and the independent variable is to create intervals for the independent variable and compute the mean of the outcome variable within each group.

```
> (AGRP <- numeric(length = length(AGE)))

  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [36] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [71] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

> for (i in 1:length(AGE)) {
+     if (20 <= AGE[i] & AGE[i] < 30) {
+         AGRP[i] <- 1
+     }
+     if (30 <= AGE[i] & AGE[i] < 35) {
+         AGRP[i] <- 2
+     }
+     if (35 <= AGE[i] & AGE[i] < 40) {
+         AGRP[i] <- 3
+     }
+     if (40 <= AGE[i] & AGE[i] < 45) {
+         AGRP[i] <- 4
```

```
+     }
+     if (45 <= AGE[i] & AGE[i] < 50) {
+         AGRP[i] <- 5
+     }
+     if (50 <= AGE[i] & AGE[i] < 55) {
+         AGRP[i] <- 6
+     }
+     if (55 <= AGE[i] & AGE[i] < 60) {
+         AGRP[i] <- 7
+     }
+     if (60 <= AGE[i] & AGE[i] < 70) {
+         AGRP[i] <- 8
+     }
+ }
> chagrp <- data.frame(AGE, AGRP, CHD)
> head(chagrp)

  AGE AGRP CHD
1  20    1   0
2  23    1   0
3  24    1   0
4  25    1   1
5  25    1   0
6  26    1   0

> detach(chage)
> write.csv(chagrp, "chagrp.csv", row.names = FALSE)
> attach(chagrp)

The following object is masked _by_ .GlobalEnv:

    AGRP
```

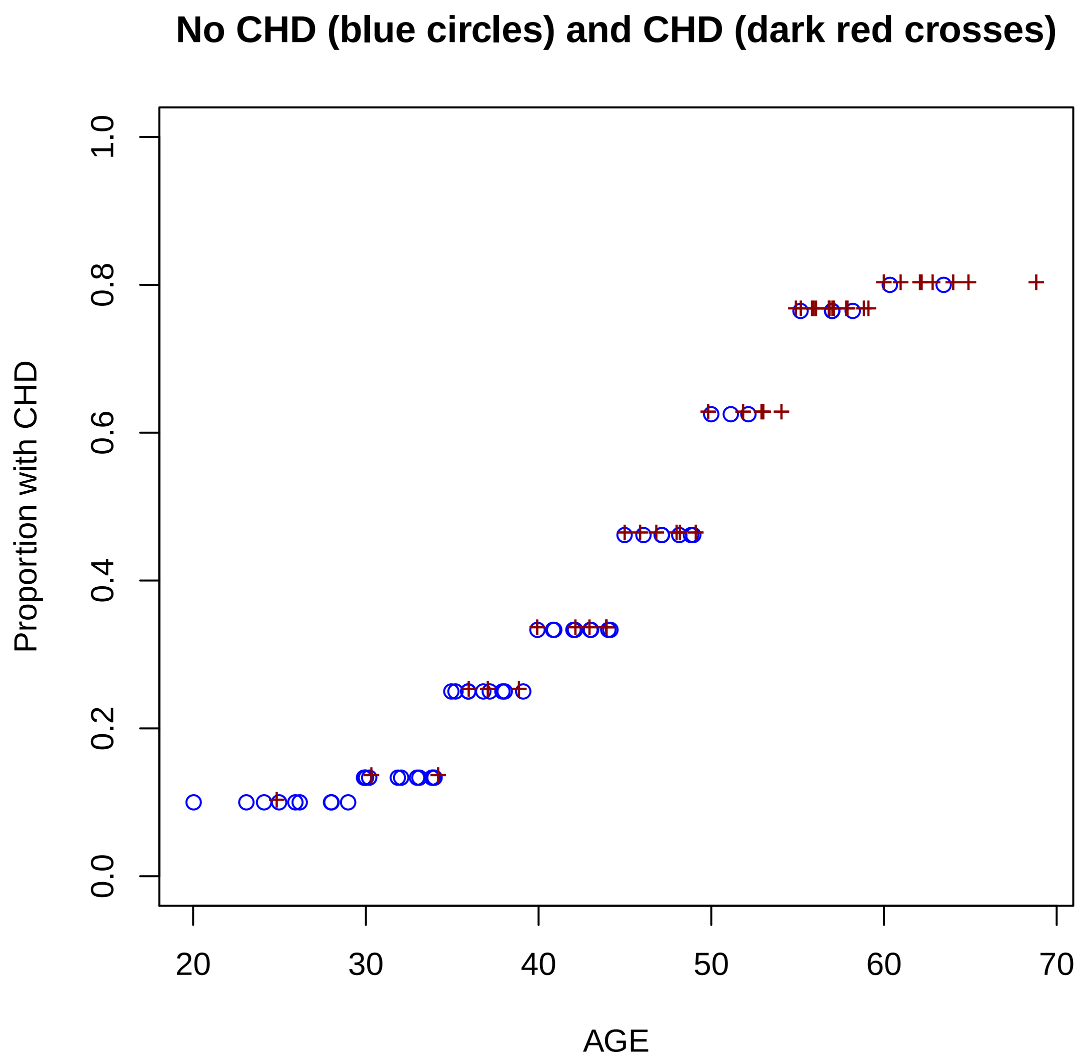
We can now compute the mean CHD, i.e., the proportion of CHD incidence, for each age group:

```
> (proportion.CHD <- numeric(length = 8))

[1] 0 0 0 0 0 0 0 0

> for (i in 1:8) {
+     proportion.CHD[i] <- mean(subset(chagrp, AGRP == i)$CHD)
+ }
> proportion.CHD

[1] 0.1000 0.1333 0.2500 0.3333 0.4615 0.6250 0.7647 0.8000
```

We plot the CHD proportions against the corresponding 8 age groups (blue circles – no CHD; dark red crosses – CHD)

```
> plot(AGE, CHD, type = "n", ylab = "Proportion with CHD", main = "No CHD (blue circles) and CHD (dark r
> for (i in 1:8) {
+     points(jitter(subset(chagrp, AGRP == i & CHD == 0)$AGE), rep(proportion.CHD[i],
+         length(subset(chagrp, AGRP == i & CHD == 0)$AGE)), col = "blue")
+     points(jitter(subset(chagrp, AGRP == i & CHD == 1)$AGE), rep(proportion.CHD[i],
+         length(subset(chagrp, AGRP == i & CHD == 1)$AGE)), pch = "+",
```

```
+           col = "darkred")
+ }
```

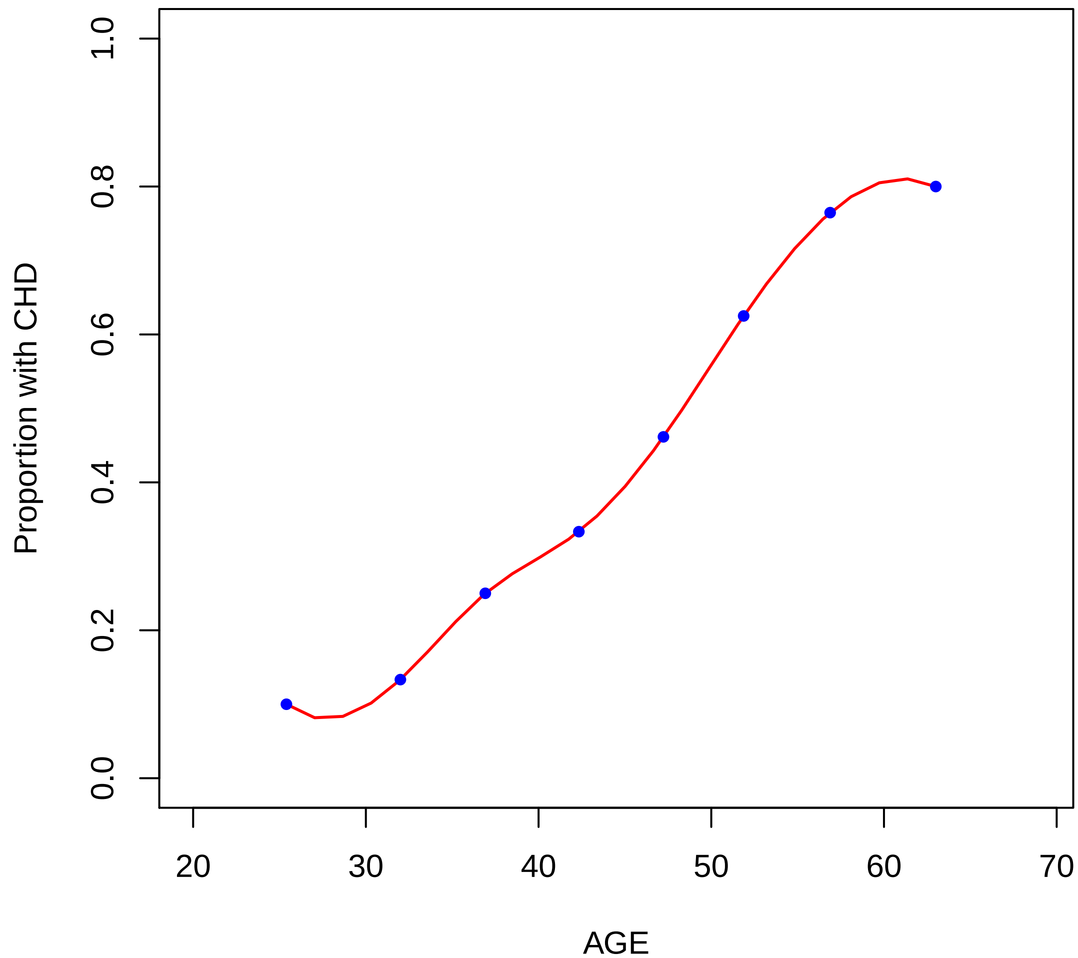


By examining the vector of CHD proportions, a clearer picture of the relationship begins to emerge: as age increases, the proportion of individuals with CHD increases.

This is particularly clear if we plot the proportion of individuals with CHD against the midpoint of each age interval.

```
> (mean.AGE <- numeric(length = 8))

[1] 0 0 0 0 0 0 0 0

> for (i in 1:8) {
+     mean.AGE[i] <- mean(subset(chagrp, AGRP == i)$AGE)
+ }
> mean.AGE

[1] 25.40 32.00 36.92 42.33 47.23 51.88 56.88 63.00
```

```
> plot(AGE, CHD, type = "n", ylab = "Proportion with CHD")
> lines(spline(mean.AGE, proportion.CHD), col = "red", lwd = 1.5)
> points(mean.AGE, proportion.CHD, pch = 20, col = "blue")
```




```
> detach(chagrp)
```

While this provides considerable insight into the relationship between CHD and AGE in this study, a functional form for this relationship needs to be described.

The plot in this figure is similar to what one might obtain if this same process of grouping and averaging were performed in a linear regression.

There are two important differences:

- The first concerns the nature of the relationship between the outcome and independent variables:
  - in any regression problem, the key quantity is the mean value of the outcome variable y given the value of the independent variable x: $E[y|x]$

- in linear regression, we assume that this mean may be expressed as an equation linear in the coefficient(s) for x (or linear in the coefficient(s) for x after some transformation of x or y), which implies that $E[y|x]$ could take any value as x ranges between $-\infty$ and $+\infty$
- but with dichotomous data, $0 \leq E[y|x] \leq 1$
- in addition, the plot shows that this mean approaches 0 and 1 gradually: the change in $E[y|x]$ per unit change in x becomes progressively smaller as the conditional mean gets closer to 0 or 1; the curve is S-shaped (sigmoidal).

- The second concerns the conditional distribution of the outcome variable:
  - in linear regression, we assume that an observation of the outcome variable may be expressed as $y = E[y|x] + \epsilon$
  - the error ϵ expresses an observation's deviation from the conditional mean
  - the most common assumption is that the error follows a normal distribution with mean 0 and some variance that is constant across the 'levels' of the independent variable
  - thus, the conditional distribution of the outcome variable given x will be normal with mean $E[y|x]$ and a variance that is constant
  - but this is not the case with a dichotomous outcome variable: the value of the outcome variable given x is $y = \pi(x) + error$
  - where $\pi(x)$, i.e., the proportion / probability of y given the value of x, is an alternative way to symbolize the conditional mean of y given x, i.e., $E[y|x]$
  - the error may assume one of two possible values: if $y = 1$, then $error = 1 - \pi(x)$ with probability $\pi(x)$; and if $y = 0$, then $error = 0 - \pi(x) = -\pi(x)$ with probability $1 - \pi(x)$
  - thus, the error has mean $\pi(x)$ and variance $\pi(x)(1 - \pi(x))$, i.e., it follows a Bernoulli (a.k.a., binomial with number of trials $n = 1$) distribution with probability $\pi(x)$

2 Generalized linear models (GLMs)

This type of data provided one of the main motivations for generalized linear models (GLMs). These models allow us to incorporate all the knowledge and techniques we have for linear models (with a continuous response), while at the same time enabling us to model non-continuous, non-normally distributed response variables.

In particular, a sub-family of GLMs can be used for binary response data while taking into account the fact that the reponse variable is categorical and the error distribution is binomial (among other things).

The crucial point:

- different functions can be used to *link* (*i*) the predicted values with (*ii*) a linear combination of the predictor variables.

Notation for this:

- η_i: this is (the deterministic part of) the model, i.e., the linear combination of the predictor variables; this linear combination can include variables multiplied by each other (i.e., interactions) and functions of these variables; remember: (generalized) linear models are linear in the coefficients / β values, not in the predictors
- μ_i: the predicted values
- the link function $g()$ connects the model and the predicted values: $g(\mu_i) = \eta_i$.

There are two key concepts needed to construct GLMs (in addition to the linear combination of predictor variables η_i that we inherit from linear models):

(1) *link functions* – we will consider three link functions:
  a. the identity function
  b. the log function
  c. the logit function (the logit should be used for the CHD∼AGE data)

(2) *error distributions* – these link functions have different error distributions associated with them, as follows:
  a. normally distributed errors are associated with the identity link
  b. Poisson distributed errors are associated with the log link
  c. binomially distributed errors are associated with the logit link

2.1 The identity link

The identity link function together with the assumption of normally distributed errors is simply the standard linear multiple regression (take a moment to think about this).

(3) The model:
  a. deterministic part (the linear combination + the link function): $\textbf{identity}(\mu_i) = \eta_i$
  b. the random / stochastic part: $y_i \sim Normal(\mu_i, \sigma^2)$

2.2 The log link

A common situation: the dependent variable is a frequency.
For example:

- how many times a child asks for help in a classroom
- the number of wide scope indefinites or modals per sentence or paragraph in a text
- the number of rice grains on any particular kitchen tile when you drop a handful of uncooked rice
- the distribution of bombs in London neighborhoods during the WW2 air raids

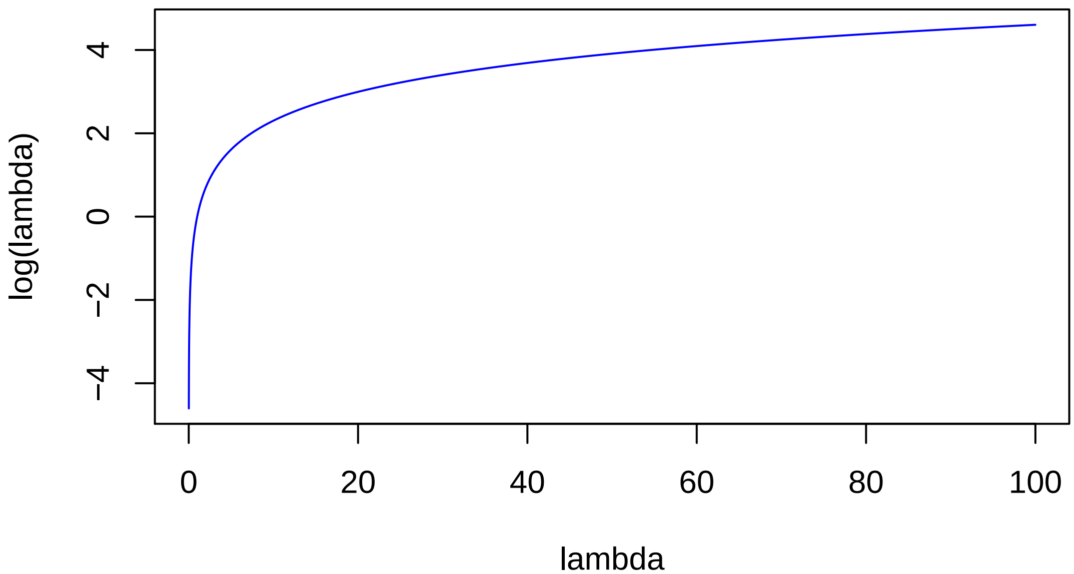
If these occurrences are independent from each other and are based only on a single probability, it is reasonable to assume that the data follow a Poisson distribution, and the log link is appropriate in this case.

(4) The model:
  a. deterministic part (the linear combination + the link function): $\log(\mu_i) = \eta_i$
  b. the random / stochastic part: $y_i \sim Poisson(\mu_i)$

The error distribution is Poisson, and the standard deviation of a Poisson distribution is the same as its mean.

(5) a. the mean of Poisson distributions is usually symbolized as λ, not μ
  b. λ is a positive real number; taking the log correctly ensures that the linear combination of predictors $\log(\lambda_i) = \log(mu_i) = \eta_i$ covers the entire real line

```
> lambda <- seq(0, 100, by = 0.01)
> plot(lambda, log(lambda), col = "blue", type = "l")
```



When a Poisson distribution is usually used, the mean rate λ is small (< 3):

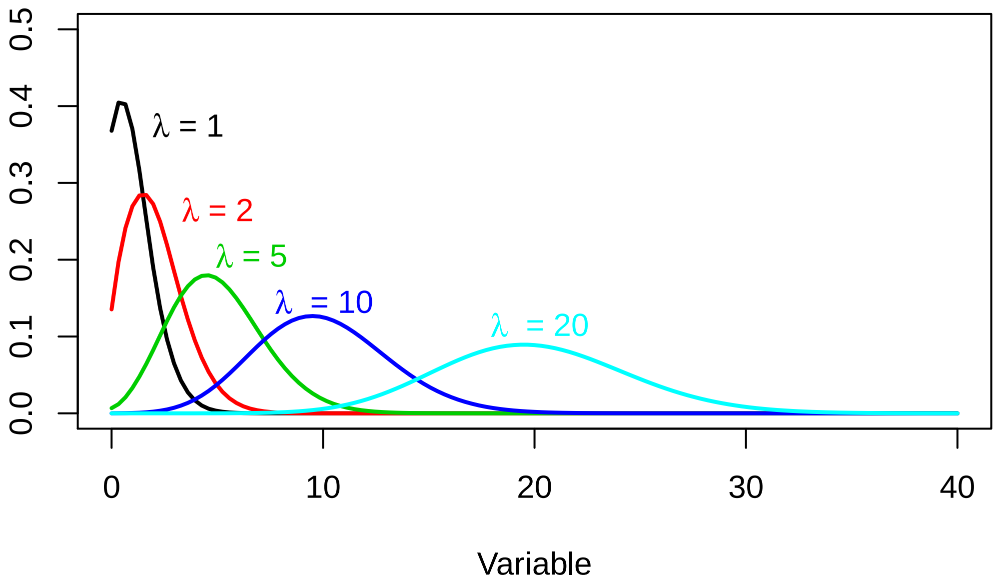
(6) it has high expected probabilities for low frequencies

(7) the expected probabilities decline as the frequencies increase (i.e., it is positively skewed)

(8) e.g., it is expected that most children ask few questions, but that some may ask lot

Examples of Poisson distributions, for $\lambda = 1, 2, 5, 10, 20$:

```
> plot(0:40, dpois(0:40, 1), xlab = "Variable", ylab = "", ylim = c(0,
+     0.5), col = "white")
> for (i in c(1, 2, 5, 10, 20)) {
+     lines(spline(0:40, dpois(0:40, i)), col = which(c(1, 2, 5, 10,
+         20) == i), lwd = 2)
+ }
> text(1, 0.37, expression(lambda, "   = 1"), pos = 4, col = 1)
> text(2.4, 0.26, expression(lambda, "   = 2"), pos = 4, col = 2)
> text(4, 0.2, expression(lambda, "   = 5"), pos = 4, col = 3)
> text(6.8, 0.14, expression(lambda, "    = 10"), pos = 4, col = 4)
> text(17, 0.11, expression(lambda, "    = 20"), pos = 4, col = 5)
```



As the value of λ reaches 10 and 20, the distribution looks more like a normal distribution, so in these situations people would often just assume normally distributed errors.

A model that involves a linear combination of predictors, the log link, and a Poisson distribution for the observations, is called a Poisson regression. The name 'log-linear model' is alternatively used, especially when we are trying to model the number of observations in a particular cell in a contingency table (Agresti 2002).

2.3 The logit link

Another common situation: a person's score is the number of correct responses out of a total (i.e., a proportion). In these situations, the logit link function can be used.

(9) The logit link function is:
   a. $\textbf{logit}(\mu_i) = \log(\frac{\mu_i}{1-\mu_i}) = \eta_i$
   b. where log is the natural logarithm
   c. the error term follows a binomial distribution

Logit stands for **log-odds**:

- the predicted value μ_i is:
  - the probability of a correct response on an item
  - the probability of heads for a coin
  - the probability that the indefinite takes wide scope in a sentence with only one other quantifier
  - etc.
- the odds: the ratio $\frac{\mu_i}{1-\mu_i}$, i.e., the probability of 'success' divided by the probability of 'failure'

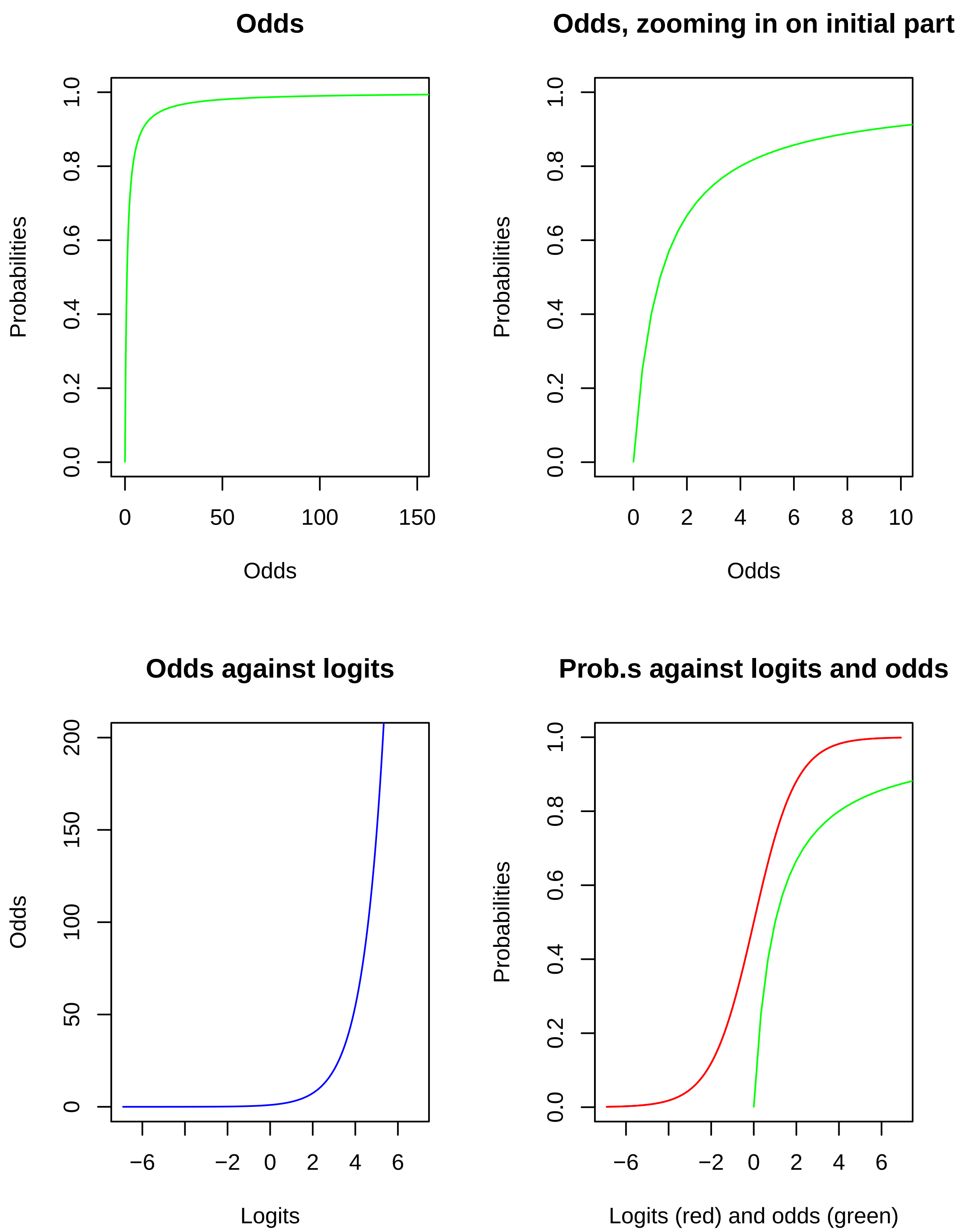
(10) The model:
   a. deterministic part (the linear combination + the link function): $\textbf{logit}(\mu_i) = \eta_i$

b. the random / stochastic part: $y_i \sim Binomial(n, \mu_i)$, where n is the total number of observations / coin flips, i.e., n is known

(11) If $n = 1$, then the observations are Bernoulli distributed: $y_i \sim Binomial(1, \mu_i)$ is equivalent to saying that $y_i \sim Bernoulli(\mu_i)$

Let's plot the corresponding odds and logits for 1000 equally spaced probabilities between 0 and 1. The logit function takes the natural logarithm of the odds, and the result is an S-shaped curve:

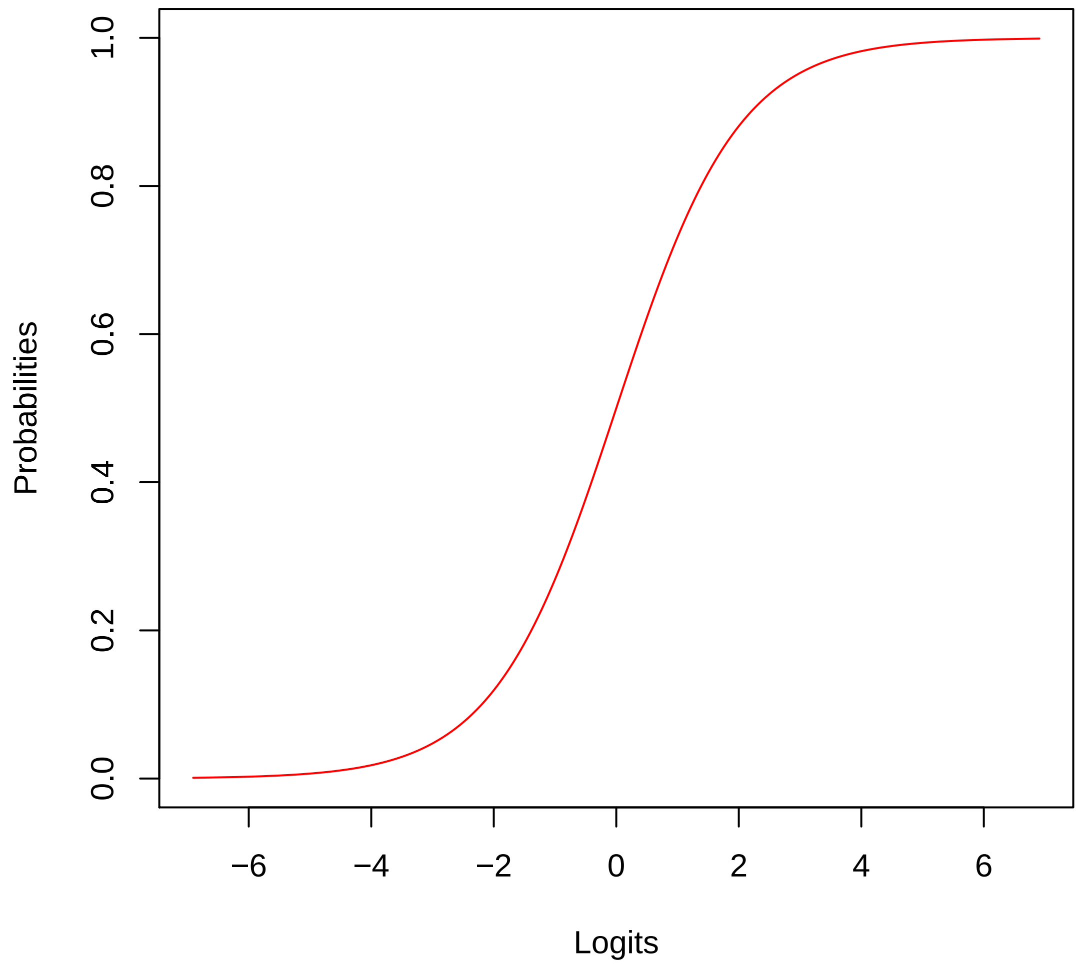
```
> y <- seq(0, 1, length.out = 1002)
> y <- y[2:1001]
> length(y)

[1] 1000

> head(y)

[1] 0.000999 0.001998 0.002997 0.003996 0.004995 0.005994

> tail(y)

[1] 0.994 0.995 0.996 0.997 0.998 0.999

> par(mfrow = c(2, 2))
> plot(y/(1 - y), y, xlab = "Odds", ylab = "Probabilities", xlim = range(-1,
+     150), col = "white", main = "Odds")
> lines(spline(y/(1 - y), y), col = "green")
> plot(y/(1 - y), y, xlab = "Odds", ylab = "Probabilities", xlim = range(-1,
+     10), col = "white", main = "Odds, zooming in on initial part")
> lines(spline(y/(1 - y), y), col = "green")
> plot(log(y/(1 - y)), y/(1 - y), xlab = "Logits", ylab = "Odds", col = "white",
+     ylim = range(0, 200), main = "Odds against logits")
> lines(spline(log(y/(1 - y)), y/(1 - y)), col = "blue")
> plot(log(y/(1 - y)), y, xlab = "Logits (red) and odds (green)", ylab = "Probabilities",
+     col = "red", type = "l", main = "Prob.s against logits and odds")
> lines(spline(log(y/(1 - y)), y), col = "red")
> lines(spline(y/(1 - y), y), col = "green")
```




```
> par(mfrow = c(1, 1))
```

The Faraway library has a logit function – if you do not want to write up the log-odds formula.

```
> library("faraway")
> plot(logit(y), y, xlab = "Logits", ylab = "Probabilities", col = "red",
+     type = "l")
```



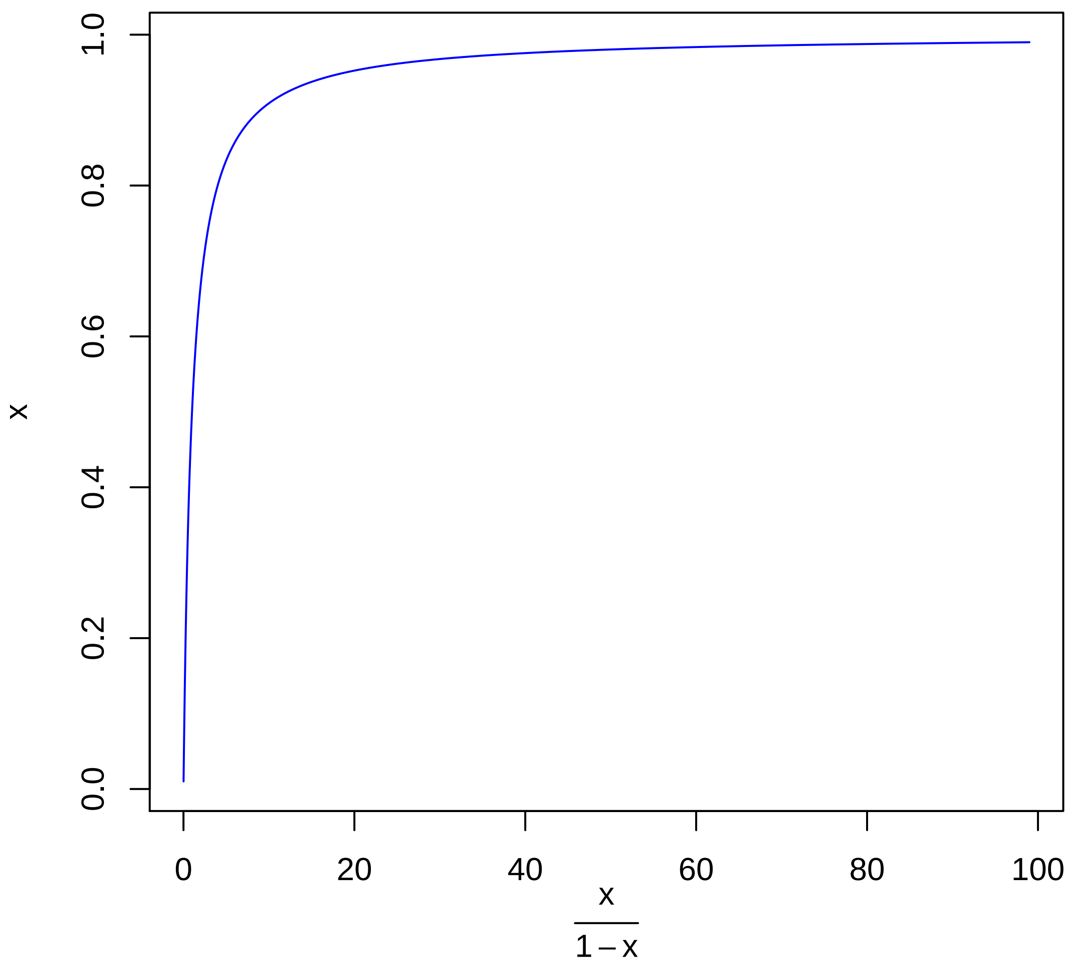
3 More about odds and log-odds, i.e., logits

3.1 More about odds

Probabilities have both a floor – at 0 – and a ceiling – at 1. The odds have no ceiling:

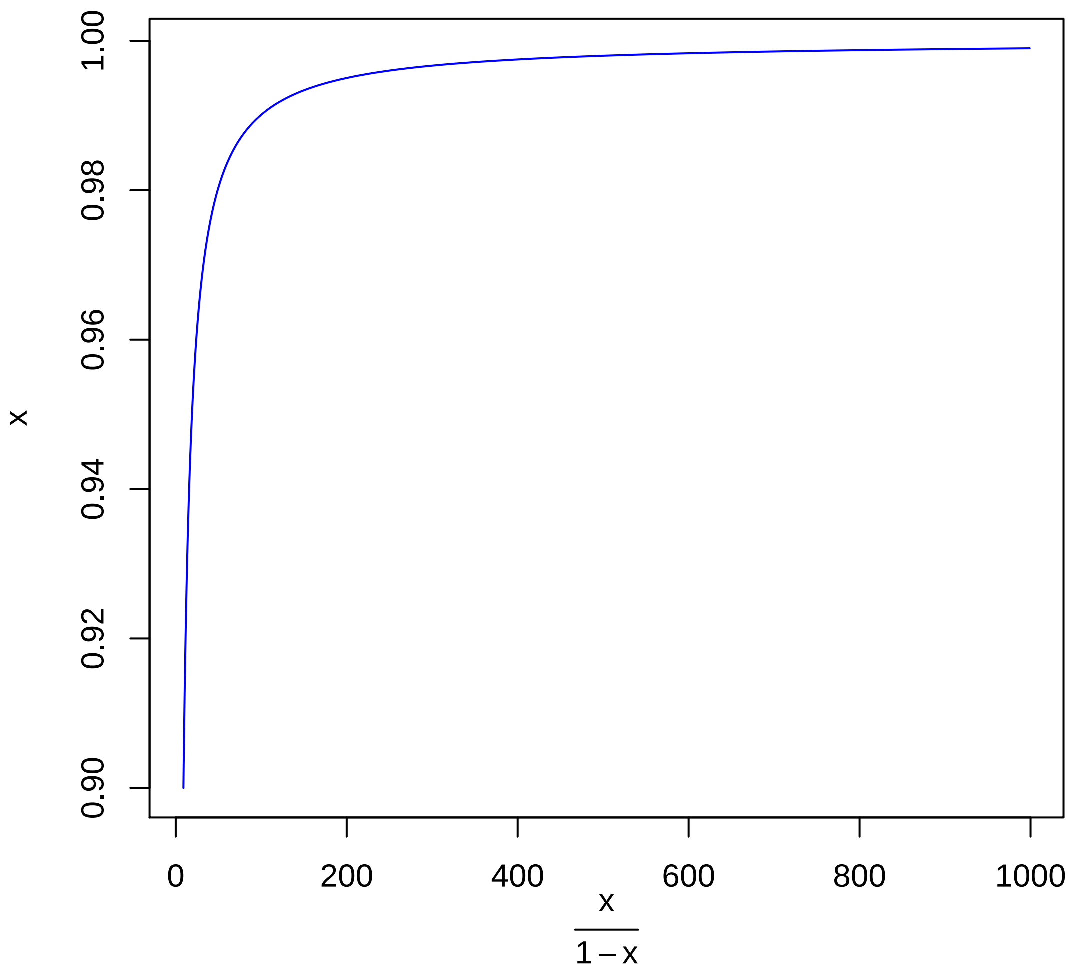
```
> x <- seq(0.01, 0.99, length.out = 10)
> round(x, 2)

 [1] 0.01 0.12 0.23 0.34 0.45 0.55 0.66 0.77 0.88 0.99

> round(1 - x, 2)
```

```
 [1] 0.99 0.88 0.77 0.66 0.55 0.45 0.34 0.23 0.12 0.01

> round(x/(1 - x), 2)

 [1]  0.01  0.13  0.29  0.51  0.80  1.24  1.97  3.39  7.41 99.00

> x <- seq(0.01, 0.99, length.out = 1000)
> plot(x/(1 - x), x, type = "l", col = "blue", xlab = expression(frac(x,
+     1 - x)))
```



As the probability gets closer to 1, the numerator of the odds becomes larger relative to the denominator and the odds become an increasingly larger number. *The odds increase greatly when the probability changes only slightly near the upper boundary of* 1.

```
> x <- seq(0.9, 0.999, length.out = 1000)
> plot(x/(1 - x), x, type = "l", col = "blue", xlab = expression(frac(x,
+     1 - x)))
```



(12) Converting between probabilities and odds:

a. Odds in terms of the probability π: $o = \frac{\pi}{1-\pi}$

b. Probability in terms of the odds o: $\pi = \frac{o}{o+1}$

Think about this for a moment and try to derive the second formula (probability in terms of odds) from the first.

Odds ratio: the ratio of two odds (which are themselves ratios):

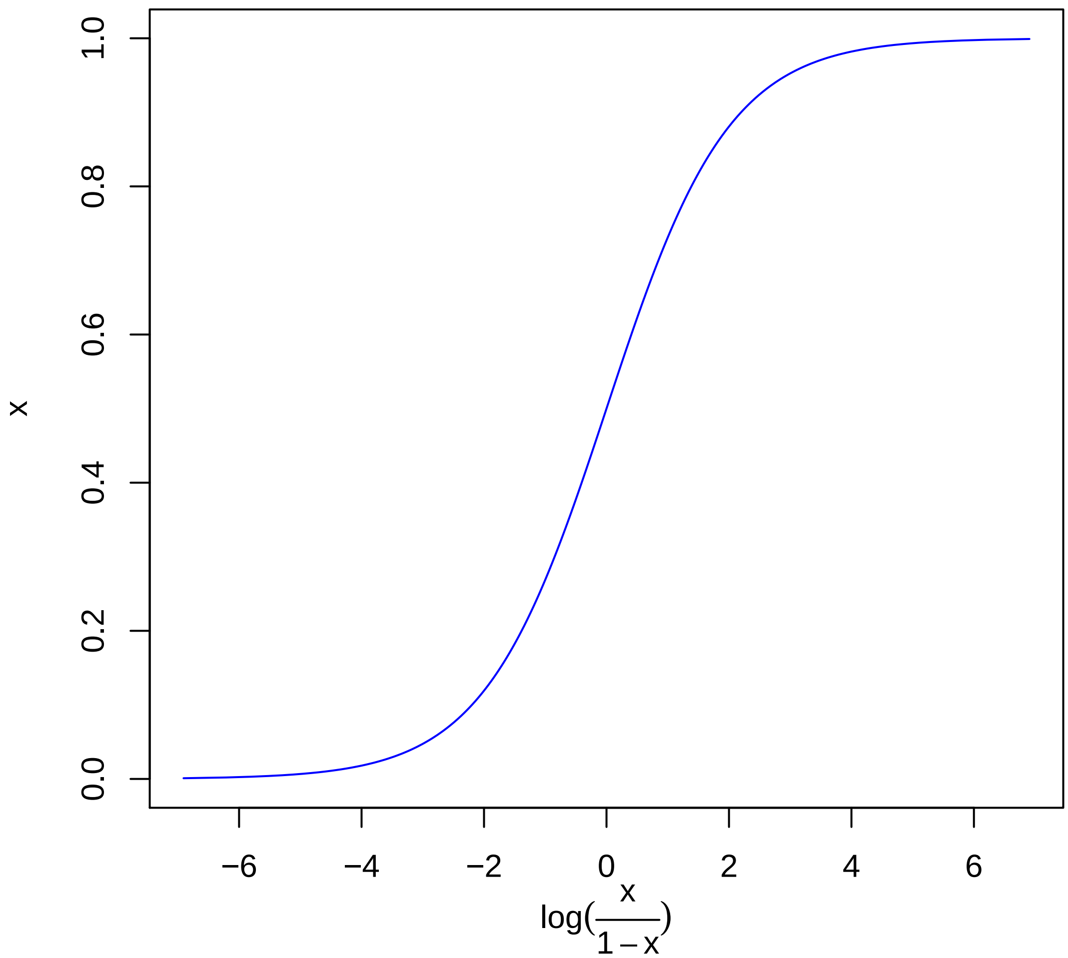
- used to measure odd changes induced by a covariate / predictor, e.g., the change in the odds that an indefinite takes wide scope if the other quantifier in the sentence is *every* vs. *each*

In sum: the odds provide a way of quantifying the likelihood of events in a way that does not have a ceiling (unlike probabilities), but that still has a floor.

3.2 More about log-odds, i.e., logits

Taking the natural log of the odds eliminates the floor. We need this if we don't want to restrict the kind of contribution predictors could make, i.e., if we want to allow predictors to be continuous / real-valued and make positive or negative contributions of arbitrary real-valued magnitudes.

```
> (x <- seq(0.01, 0.99, length.out = 10))

 [1] 0.0100 0.1189 0.2278 0.3367 0.4456 0.5544 0.6633 0.7722 0.8811 0.9900

> round(x, 2)

 [1] 0.01 0.12 0.23 0.34 0.45 0.55 0.66 0.77 0.88 0.99

> round(x/(1 - x), 2)

 [1]  0.01  0.13  0.29  0.51  0.80  1.24  1.97  3.39  7.41 99.00

> round(log(x/(1 - x)), 2)

 [1] -4.60 -2.00 -1.22 -0.68 -0.22  0.22  0.68  1.22  2.00  4.60

> cbind(round(x, 2), round(x/(1 - x), 2), round(log(x/(1 - x)), 2))

      [,1]  [,2]  [,3]
 [1,] 0.01  0.01 -4.60
 [2,] 0.12  0.13 -2.00
 [3,] 0.23  0.29 -1.22
 [4,] 0.34  0.51 -0.68
 [5,] 0.45  0.80 -0.22
 [6,] 0.55  1.24  0.22
 [7,] 0.66  1.97  0.68
 [8,] 0.77  3.39  1.22
 [9,] 0.88  7.41  2.00
[10,] 0.99 99.00  4.60

> x <- seq(0.001, 0.999, length.out = 1000)
> plot(log(x/(1 - x)), x, type = "l", col = "blue", xlab = expression(log(frac(x,
+     1 - x))))
```



That is, the logit transformation basically linearizes the non-linear relationship between the predictor x and the binary response variable y.

We obtain probabilities from logits by obtaining the odds first, then obtaining probabilities in terms of the odds:

```
> (logits <- seq(-5, 5, by = 1))

 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5

> odds <- exp(logits)
> round(odds, 2)

 [1]   0.01   0.02   0.05   0.14   0.37   1.00   2.72   7.39  20.09  54.60
[11] 148.41

> probabilities <- odds/(1 + odds)
> round(probabilities, 2)

 [1] 0.01 0.02 0.05 0.12 0.27 0.50 0.73 0.88 0.95 0.98 0.99
```

In one fell swoop:

```
> probabilities <- exp(logits)/(1 + exp(logits))
> round(probabilities, 2)

 [1] 0.01 0.02 0.05 0.12 0.27 0.50 0.73 0.88 0.95 0.98 0.99
```

Equivalently (think about why this equivalence holds for a moment):

```
> probabilities2 <- 1/(1 + exp(-logits))
> round(probabilities2, 2)

 [1] 0.01 0.02 0.05 0.12 0.27 0.50 0.73 0.88 0.95 0.98 0.99
```

The logit is symmetric around the midoint probability of 0.5:

```
> log(0.5/0.5)

[1] 0
```

- probabilities below 0.5 result in negative logits; the logit approaches $-\infty$ as the probability approaches 0
- probabilities above 0.5 result in positive logits; the logit approaches ∞ as the probability approaches 1
- the same change in probabilities translates into different changes in logits: as the probability gets closer to 0 or 1, the same change in probability translates into greater changes in logits

Consider the last point in more detail:

```
> probs <- seq(0.01, 0.99, length.out = 10)
> round(probs, 2)

 [1] 0.01 0.12 0.23 0.34 0.45 0.55 0.66 0.77 0.88 0.99
```

Differences in probability:

```
> round(probs[2:10] - probs[1:9], 2)

[1] 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11
```

Differences odds:

```
> round((probs/(1 - probs))[2:10] - (probs/(1 - probs))[1:9], 2)

[1]  0.12  0.16  0.21  0.30  0.44  0.73  1.42  4.02 91.59
```

Differences in log-odds / logits:

```
> round(log(probs/(1 - probs))[2:10] - log(probs/(1 - probs))[1:9],
+     2)

[1] 2.59 0.78 0.54 0.46 0.44 0.46 0.54 0.78 2.59
```

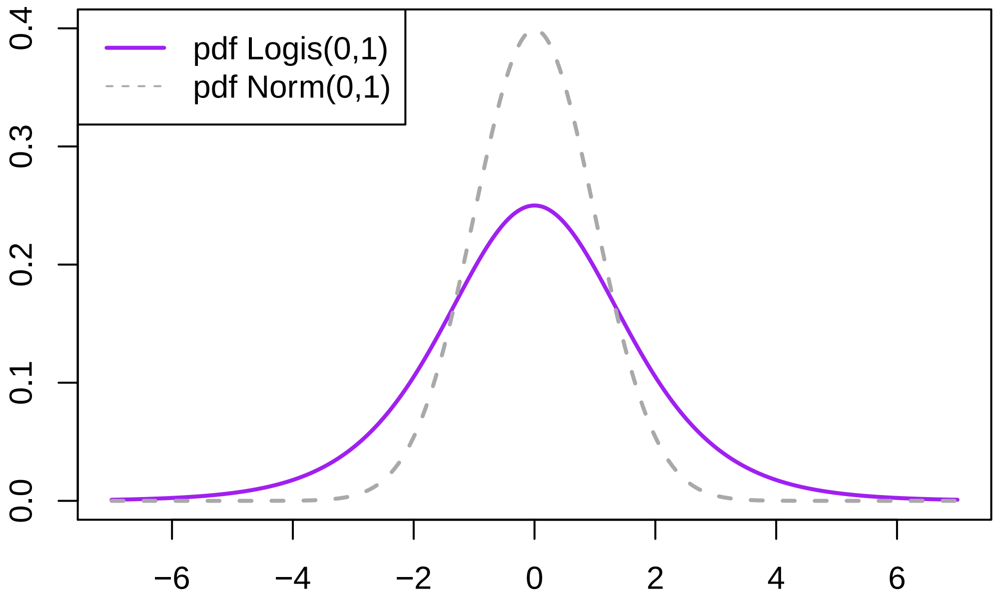
Conversely, a unit change on the logit scale results in smaller probability differences near the floor or the ceiling:

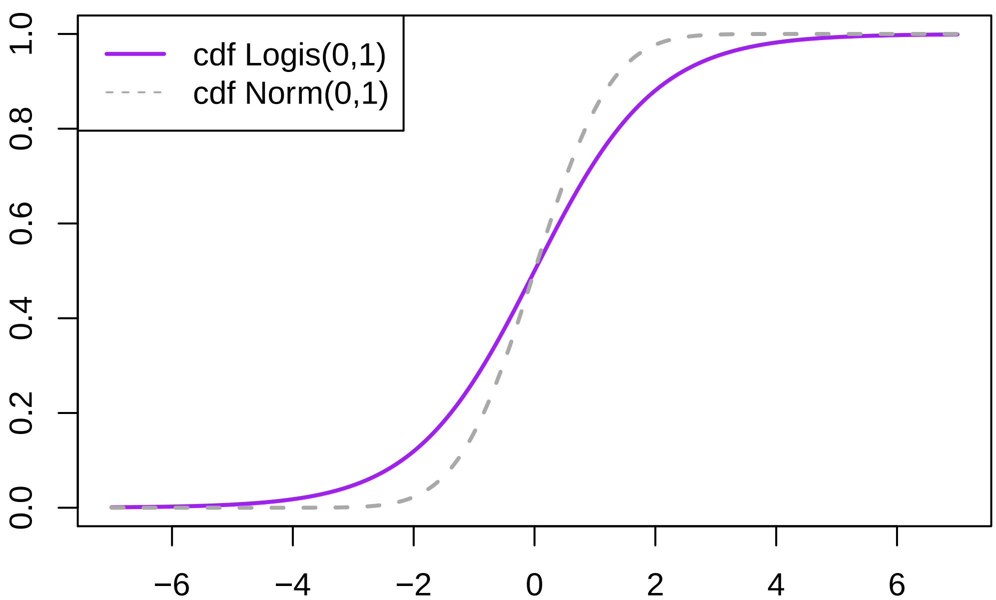
```
> (logits <- seq(-5, 5, by = 1))

 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5

> round(exp(logits)/(1 + exp(logits)), 2)

 [1] 0.01 0.02 0.05 0.12 0.27 0.50 0.73 0.88 0.95 0.98 0.99

> logits[2:11] - logits[1:10]

 [1] 1 1 1 1 1 1 1 1 1 1

> round((exp(logits)/(1 + exp(logits)))[2:11] - (exp(logits)/(1 + exp(logits)))[1:10],
+     2)

 [1] 0.01 0.03 0.07 0.15 0.23 0.23 0.15 0.07 0.03 0.01
```

4 The standard logistic distribution

We use the standard logistic distribution (mean/location=0, scale=1). It has heavier tails than the standard normal distribution.

```
> logits <- x <- seq(-7, 7, by = 0.01)
> par(mfrow = c(2, 1))
> plot(x, dlogis(x), type = "l", lwd = 2, col = "purple", main = "",
+     xlab = "", ylab = "", ylim = range(0, 0.4))
> lines(x, dnorm(x), col = "darkgrey", lwd = 2, lty = 2)
> legend(x = "topleft", legend = c("pdf Logis(0,1)", "pdf Norm(0,1)"),
+     col = c("purple", "darkgrey"), lty = c(1, 2), lwd = c(2, 1))
> plot(x, plogis(x), type = "l", lwd = 2, col = "purple", main = "",
+     xlab = "", ylab = "", )
> lines(x, pnorm(x), col = "darkgrey", lwd = 2, lty = 2)
> legend(x = "topleft", legend = c("cdf Logis(0,1)", "cdf Norm(0,1)"),
+     col = c("purple", "darkgrey"), lty = c(1, 2), lwd = c(2, 1))
```

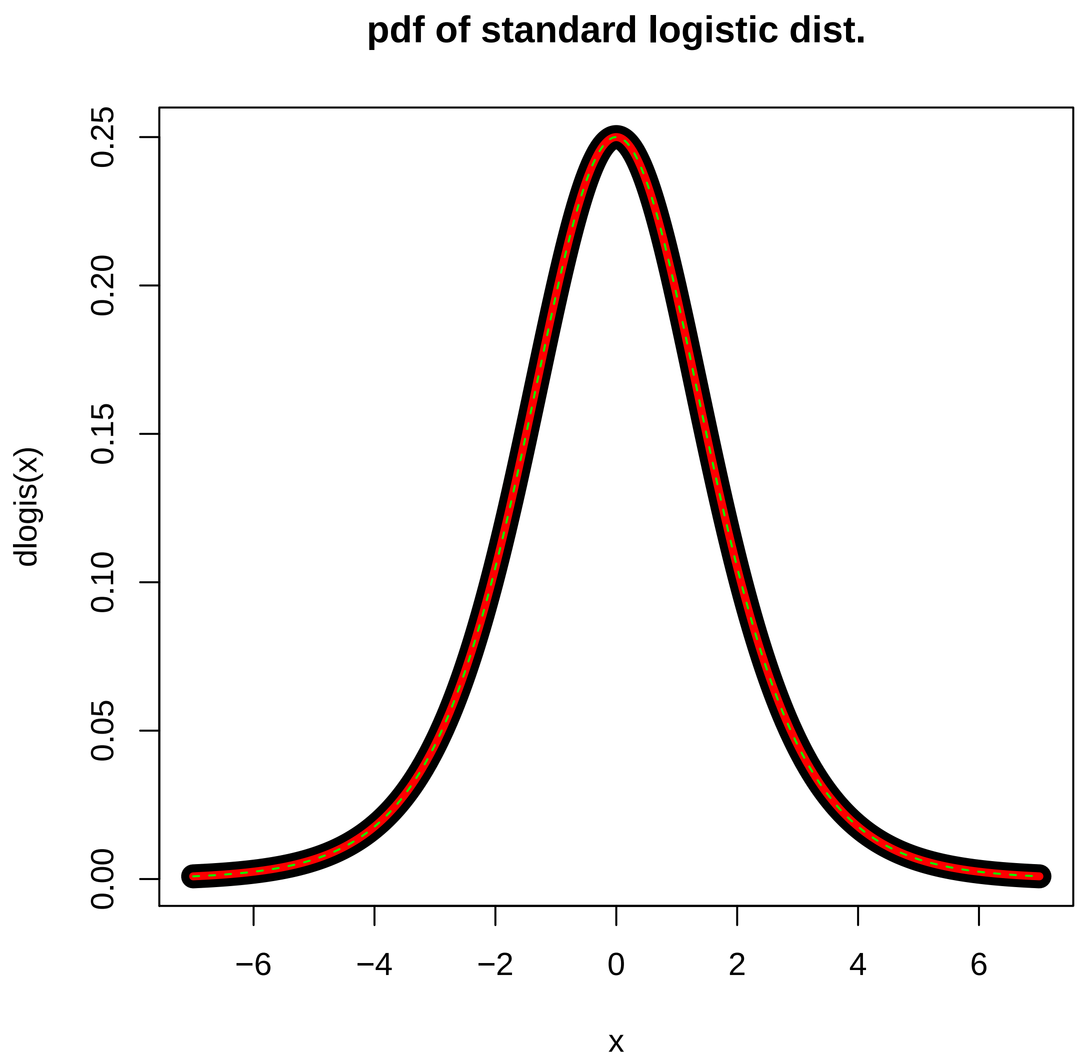

pdf Logis(0,1)
pdf Norm(0,1)
0.0
0.1
0.2
0.3
0.4
−6
−4
−2
0
2
4
6


cdf Logis(0,1)
cdf Norm(0,1)
0.0
0.2
0.4
0.6
0.8
1.0
−6
−4
−2
0
2
4
6

```
> par(mfrow = c(1, 1))
```

The standard logistic distribution has a scale of 1, but this is **not** the deviation / σ of the standard logistic distribution ($\sigma = scale \cdot \frac{\pi}{\sqrt{3}}$, where π is the numerical constant π, i.e., the ratio of a circle's circumference to its diameter).

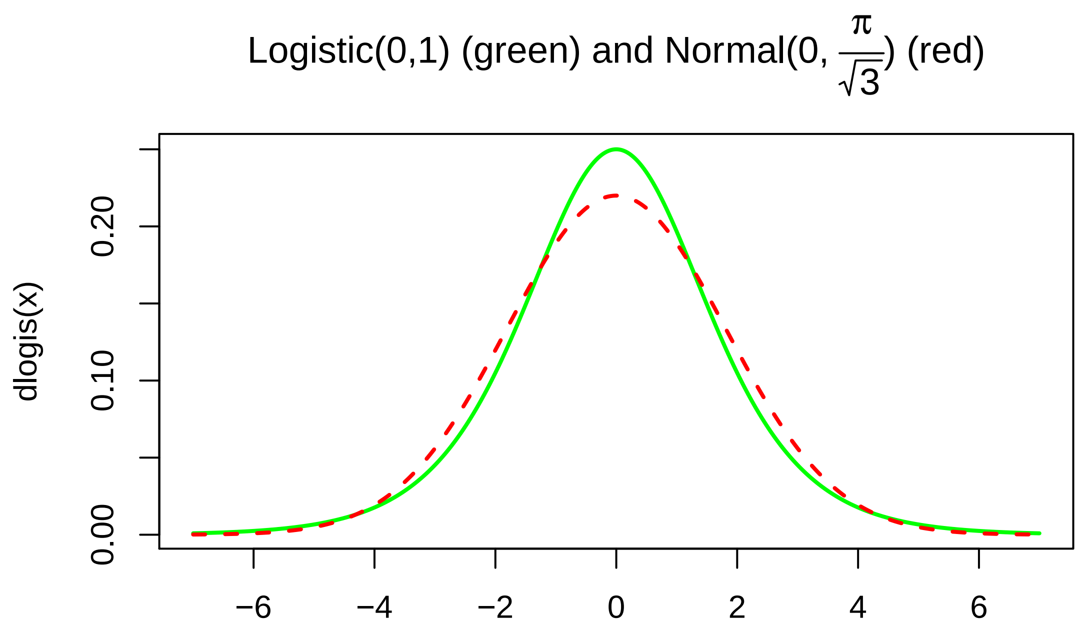
```
> plot(x, dlogis(x), type = "l", lwd = 12, col = "black", main = "pdf of standard logistic dist.")
> points(x, dlogis(x, location = 0), type = "l", lwd = 4, col = "red")
> points(x, dlogis(x, location = 0, scale = 1), type = "l", lwd = 1,
+      lty = 2, col = "green")
```

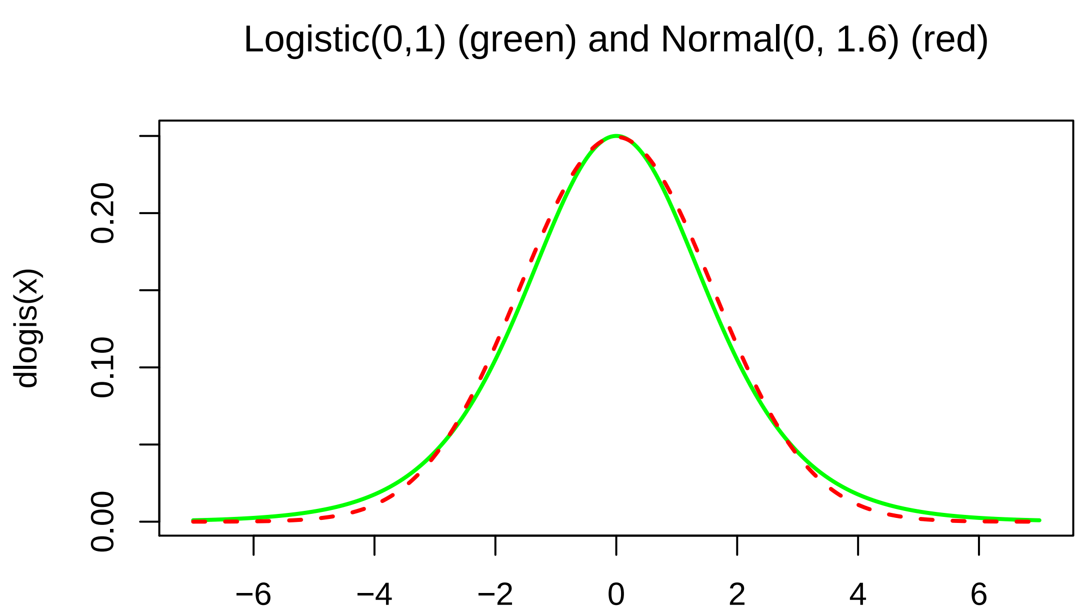


We can approximate a logistic distribution with a normal distribution with a standard deviation of $\sigma = \frac{\pi}{\sqrt{3}}$. An even better approximation is $\sigma = 1.6$ – note the fatter tails of the logistic distribution:

```
> pi/sqrt(3)

[1] 1.814
```

```
> par(mfrow = c(2, 1))
> plot(x, dlogis(x), type = "l", lwd = 2, col = "green", main = expression(paste("Logistic(0,1) (green)
+     frac(pi, sqrt(3)), ") (red)")), xlab = "")
> points(x, dnorm(x, mean = 0, sd = pi/sqrt(3)), type = "l", lwd = 2,
+     lty = 2, col = "red")
> plot(x, dlogis(x), type = "l", lwd = 2, col = "green", main = expression("Logistic(0,1) (green) and No
+     xlab = "")
> points(x, dnorm(x, mean = 0, sd = 1.6), type = "l", lwd = 2, lty = 2,
+     col = "red")
```






```
> par(mfrow = c(1, 1))
```

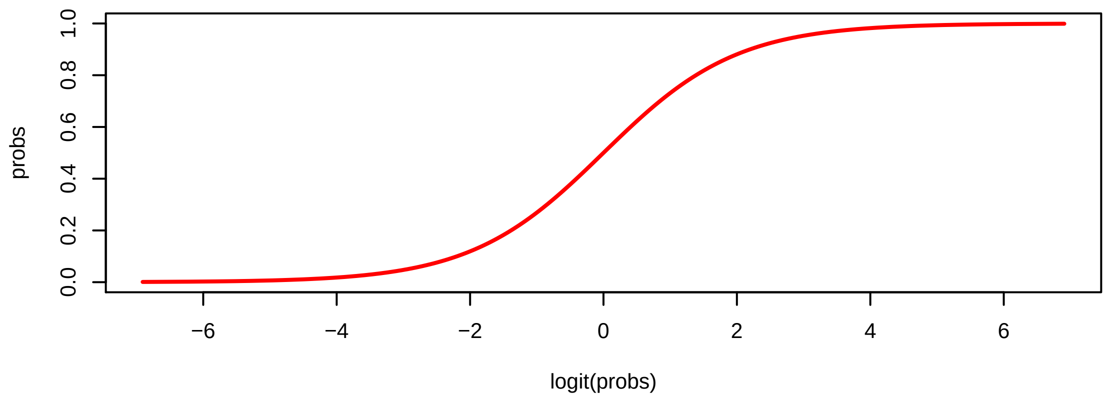
Importantly: **logit**() is the inverse function of the logistic cdf `plogis`() for the standard logistic distribution.

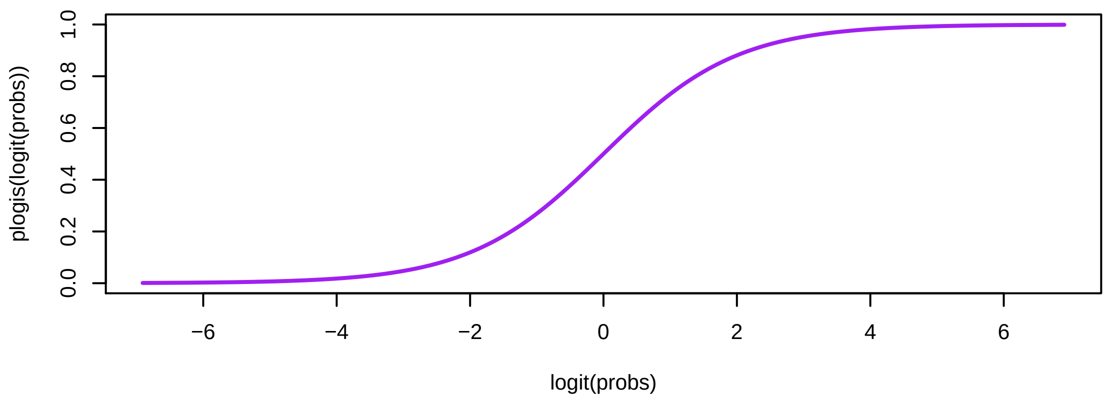
Thus, the deterministic part of logistic regression models can be formulated in either of the following 2 ways:

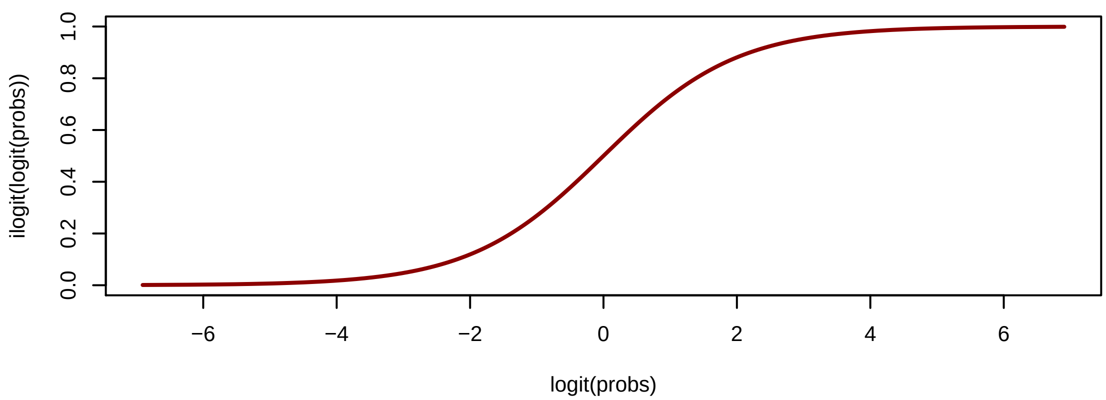
- $\textbf{logit}(\mu_i) = \eta_i$
- $\mu_i = plogis(\eta_i) = \textbf{ilogit}(\eta_i)$

where $\eta_i = X_i \cdot \beta$, i.e., η_i is the linear combination of predictors.

```
> probs <- seq(0, 1, length.out = 1002)
> probs <- probs[2:1001]
> par(mfrow = c(3, 1))
> library("faraway")
> plot(logit(probs), probs, col = "red", type = "l", lwd = 2)
> plot(logit(probs), plogis(logit(probs)), col = "purple", type = "l",
+     lwd = 2)
> plot(logit(probs), ilogit(logit(probs)), col = "darkred", type = "l",
+     lwd = 2)
```








```
> par(mfrow = c(1, 1))
```

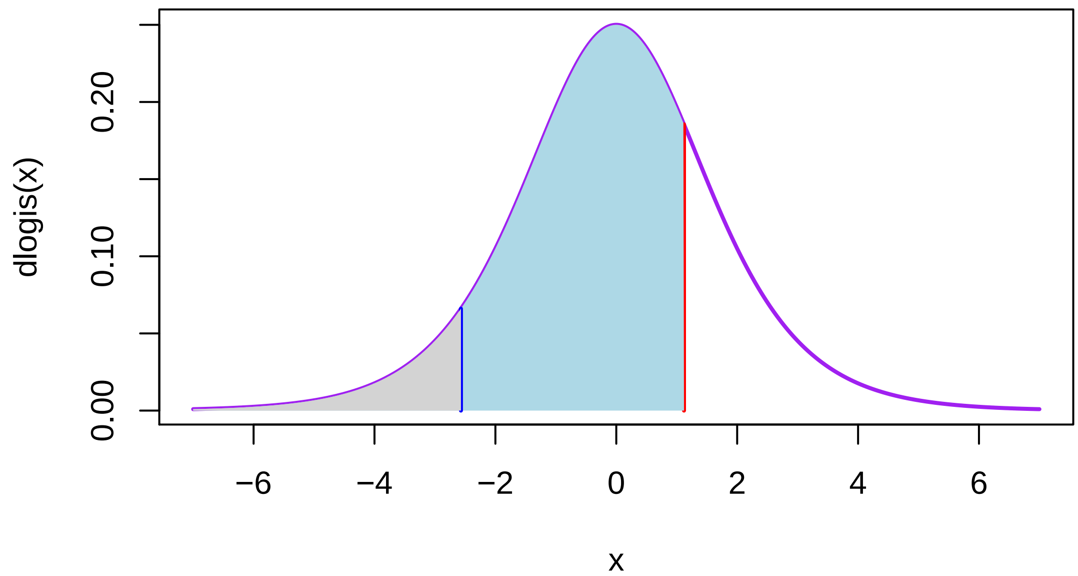
When the logit is 0, the probability is 0.5 (and the odds are 1). A positive / negative logit corresponds to a 'higher-than-chance' probability of success / failure.

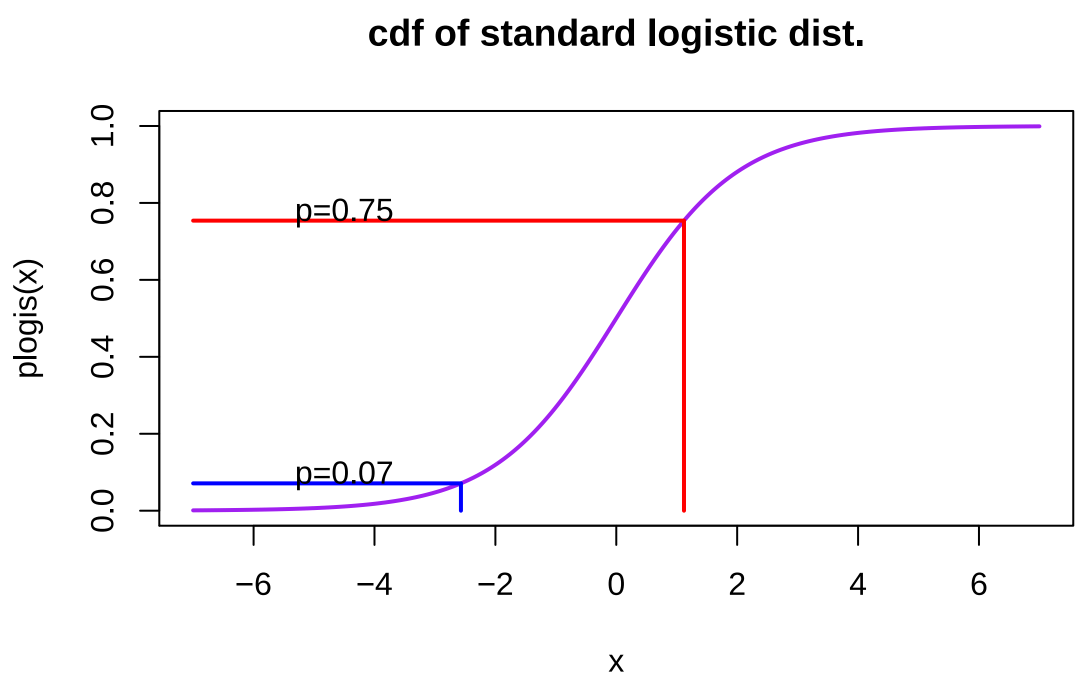
In (binomial) logistic regression:

- the mean of the logistic cdf is 0, i.e., it is fixed at 0.5 probability (chance-level probability of success / failure)
- if the linear predictor, i.e., the logit, is to the right / left of 0, then we have a 'higher-than-chance' probability of success / failure

Consider, for example, two logits of 1.12 and -2.57 and let us convert them into corresponding probabilities both wrt a pdf plot and a cdf plot

```
> logits <- x <- seq(-7, 7, by = 0.01)
> par(mfrow = c(2, 1))
> plot(x, dlogis(x), type = "l", lwd = 2, col = "purple", main = "")
> segments(1.12, 0, 1.12, dlogis(1.12), col = "red", lwd = 2)
> coord.x <- c(min(x), x[x <= 1.12], 1.12)
> coord.y <- c(0, dlogis(x[x <= 1.12]), 0)
> polygon(coord.x, coord.y, col = "lightblue", border = NA)
> segments(-2.57, 0, -2.57, dlogis(-2.57), col = "blue", lwd = 2)
> coord.x <- c(min(x), x[x <= -2.57], -2.57)
> coord.y <- c(0, dlogis(x[x <= -2.57]), 0)
> polygon(coord.x, coord.y, col = "lightgray", border = NA)
> plot(x, plogis(x), type = "l", lwd = 2, col = "purple", main = "cdf of standard logistic dist.")
> segments(1.12, 0, 1.12, plogis(1.12), col = "red", lwd = 2)
> segments(1.12, plogis(1.12), min(x), plogis(1.12), col = "red", lwd = 2)
> text(min(x) + 2.5, plogis(1.12) + 0.02, paste("p=", round(plogis(1.12),
+     2), sep = ""))
> segments(-2.57, 0, -2.57, plogis(-2.57), col = "blue", lwd = 2)
> segments(-2.57, plogis(-2.57), min(x), plogis(-2.57), col = "blue",
+     lwd = 2)
> text(min(x) + 2.5, plogis(-2.57) + 0.02, paste("p=", round(plogis(-2.57),
+     2), sep = ""))
```





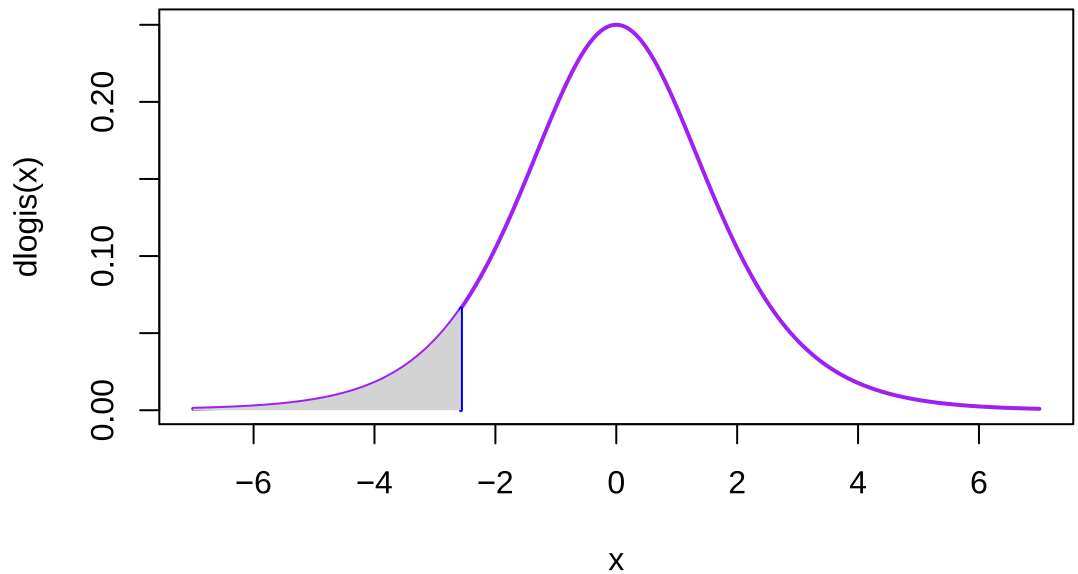

```
> par(mfrow = c(1, 1))
```

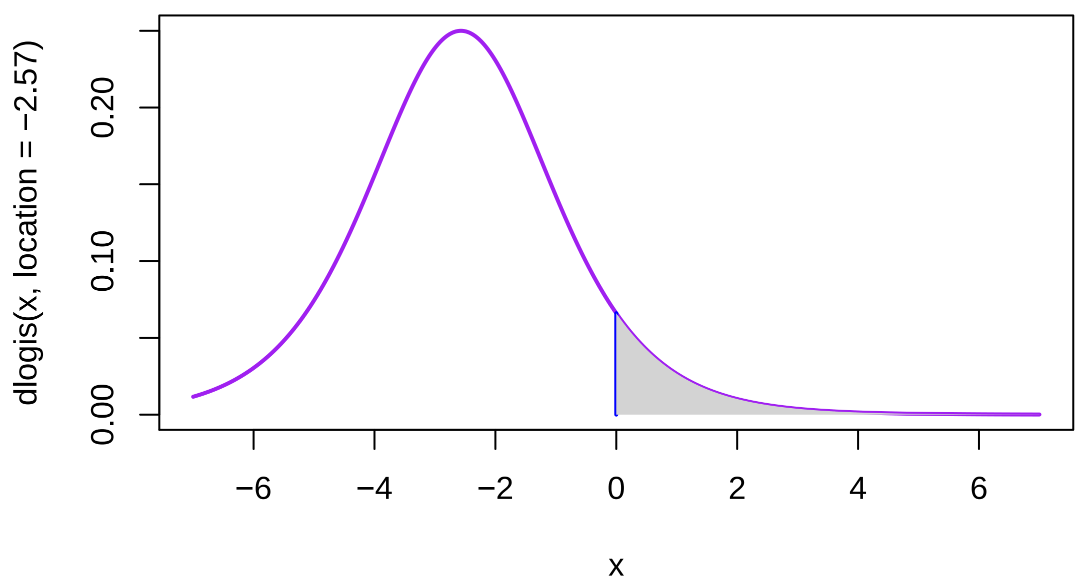
Focus now only on the pdf plot:

- we conceptualized the probability of success corresponding to a particular logit, e.g., -2.57, as the area under the pdf of the standard logistical distribution (mean=0, scale=1) whose right boundary is given by the logit

- we could alternatively conceptualize this as the area under the pdf of the logistical distribution with mean=logit=-2.57 (scale=1) that is to the right of 0

```
> par(mfrow = c(2, 1))
> plot(x, dlogis(x), type = "l", lwd = 2, col = "purple", main = "")
> segments(-2.57, 0, -2.57, dlogis(-2.57), col = "blue", lwd = 2)
> coord.x <- c(min(x), x[x <= -2.57], -2.57)
> coord.y <- c(0, dlogis(x[x <= -2.57]), 0)
> polygon(coord.x, coord.y, col = "lightgray", border = NA)
> plot(x, dlogis(x, location = -2.57), type = "l", lwd = 2, col = "purple",
+     main = "")
> segments(0, 0, 0, dlogis(0, location = -2.57), col = "blue", lwd = 2)
> coord.x <- c(0, x[x >= 0], max(x))
> coord.y <- c(0, dlogis(x[x >= 0], location = -2.57), 0)
> polygon(coord.x, coord.y, col = "lightgray", border = NA)
```

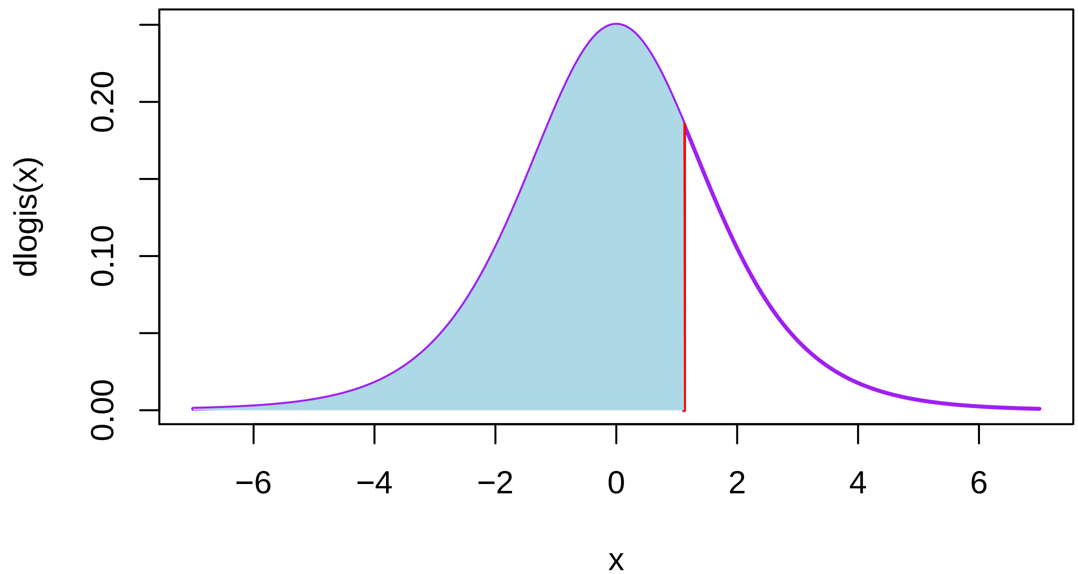


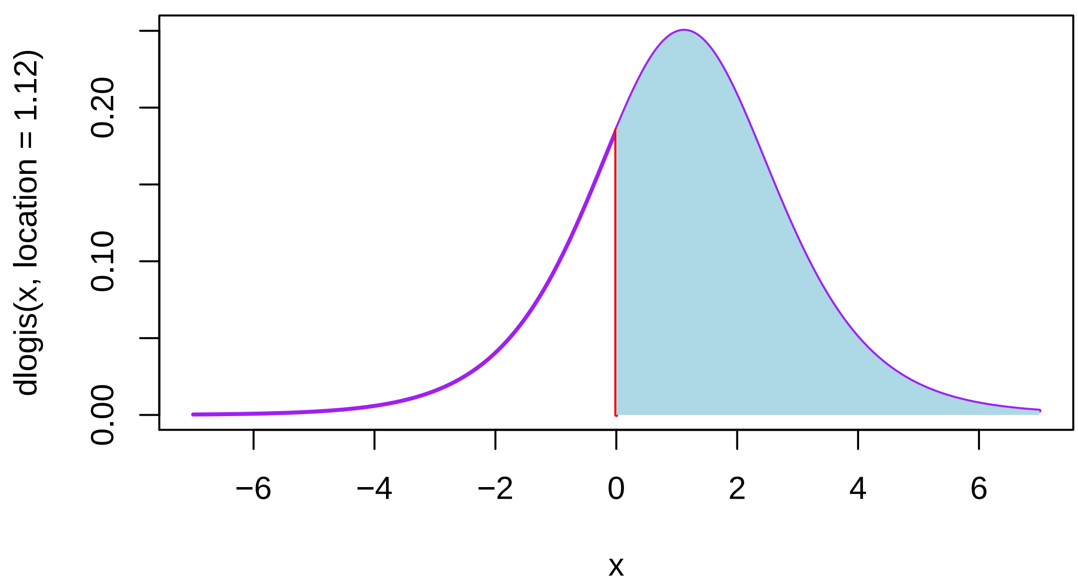



```
> par(mfrow = c(1, 1))
```

A similar reconceptualization works for the other logit, namely 1.12.

```
> par(mfrow = c(2, 1))
> plot(x, dlogis(x), type = "l", lwd = 2, col = "purple", main = "")
> segments(1.12, 0, 1.12, dlogis(1.12), col = "red", lwd = 2)
> coord.x <- c(min(x), x[x <= 1.12], 1.12)
> coord.y <- c(0, dlogis(x[x <= 1.12]), 0)
> polygon(coord.x, coord.y, col = "lightblue", border = NA)
> plot(x, dlogis(x, location = 1.12), type = "l", lwd = 2, col = "purple",
+     main = "")
> segments(0, 0, 0, dlogis(0, location = 1.12), col = "red", lwd = 2)
> coord.x <- c(0, x[x >= 0], max(x))
> coord.y <- c(0, dlogis(x[x >= 0], location = 1.12), 0)
> polygon(coord.x, coord.y, col = "lightblue", border = NA)
```






```
> par(mfrow = c(1, 1))
```

This reconceptualization will be crucial when we generalize logistic regression for binary variables to ordinal variables, i.e., the kind of responses we get in acceptability judgment tasks with a discrete rating scale (Likert scale).

5 The logistic regression for the CHD∼AGE data

```
> chage <- read.csv("chage.csv")
> head(chage)

  AGE CHD
1  20   0
2  23   0
3  24   0
4  25   1
5  25   0
6  26   0

> m1 <- glm(CHD ~ AGE, family = binomial, data = chage)
> summary(m1)


Call:
glm(formula = CHD ~ AGE, family = binomial, data = chage)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.972  -0.846  -0.458   0.825   2.286

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.3095     1.1337   -4.68  2.8e-06 ***
AGE           0.1109     0.0241    4.61  4.0e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 136.66  on 99  degrees of freedom
Residual deviance: 107.35  on 98  degrees of freedom
AIC: 111.4

Number of Fisher Scoring iterations: 4

> round(summary(m1)$coef, 2)

            Estimate Std. Error z value Pr(>|z|)
(Intercept)    -5.31       1.13   -4.68        0
AGE             0.11       0.02    4.61        0
```

The logits:

```
> round(predict(m1), 2)

    1     2     3     4     5     6     7     8     9    10    11    12
-3.09 -2.76 -2.65 -2.54 -2.54 -2.43 -2.43 -2.20 -2.20 -2.09 -1.98 -1.98
   13    14    15    16    17    18    19    20    21    22    23    24
-1.98 -1.98 -1.98 -1.98 -1.76 -1.76 -1.65 -1.65 -1.54 -1.54 -1.54 -1.54
   25    26    27    28    29    30    31    32    33    34    35    36
```

```
-1.54 -1.43 -1.43 -1.32 -1.32 -1.32 -1.21 -1.21 -1.21 -1.09 -1.09 -0.98
   37    38    39    40    41    42    43    44    45    46    47    48
-0.98 -0.87 -0.87 -0.76 -0.76 -0.65 -0.65 -0.65 -0.65 -0.54 -0.54 -0.54
   49    50    51    52    53    54    55    56    57    58    59    60
-0.43 -0.43 -0.43 -0.43 -0.32 -0.32 -0.21 -0.21 -0.10 -0.10 -0.10  0.01
   61    62    63    64    65    66    67    68    69    70    71    72
 0.01  0.01  0.13  0.13  0.13  0.24  0.24  0.35  0.46  0.46  0.57  0.57
   73    74    75    76    77    78    79    80    81    82    83    84
 0.68  0.79  0.79  0.79  0.90  0.90  0.90  1.01  1.01  1.01  1.01  1.01
   85    86    87    88    89    90    91    92    93    94    95    96
 1.01  1.12  1.12  1.12  1.23  1.23  1.35  1.35  1.46  1.57  1.57  1.68
   97    98    99   100
 1.79  1.79  1.90  2.34
```

The probabilities:

```
> round(1/(1 + exp(-predict(m1))), 2)

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
0.04 0.06 0.07 0.07 0.07 0.08 0.08 0.10 0.10 0.11 0.12 0.12 0.12 0.12 0.12
  16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
0.12 0.15 0.15 0.16 0.16 0.18 0.18 0.18 0.18 0.18 0.19 0.19 0.21 0.21 0.21
  31   32   33   34   35   36   37   38   39   40   41   42   43   44   45
0.23 0.23 0.23 0.25 0.25 0.27 0.27 0.29 0.29 0.32 0.32 0.34 0.34 0.34 0.34
  46   47   48   49   50   51   52   53   54   55   56   57   58   59   60
0.37 0.37 0.37 0.39 0.39 0.39 0.39 0.42 0.42 0.45 0.45 0.48 0.48 0.48 0.50
  61   62   63   64   65   66   67   68   69   70   71   72   73   74   75
0.50 0.50 0.53 0.53 0.53 0.56 0.56 0.59 0.61 0.61 0.64 0.64 0.66 0.69 0.69
  76   77   78   79   80   81   82   83   84   85   86   87   88   89   90
0.69 0.71 0.71 0.71 0.73 0.73 0.73 0.73 0.73 0.73 0.75 0.75 0.75 0.77 0.77
  91   92   93   94   95   96   97   98   99  100
0.79 0.79 0.81 0.83 0.83 0.84 0.86 0.86 0.87 0.91

> round(predict(m1, type = "response"), 2)

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
0.04 0.06 0.07 0.07 0.07 0.08 0.08 0.10 0.10 0.11 0.12 0.12 0.12 0.12 0.12
  16   17   18   19   20   21   22   23   24   25   26   27   28   29   30
0.12 0.15 0.15 0.16 0.16 0.18 0.18 0.18 0.18 0.18 0.19 0.19 0.21 0.21 0.21
  31   32   33   34   35   36   37   38   39   40   41   42   43   44   45
0.23 0.23 0.23 0.25 0.25 0.27 0.27 0.29 0.29 0.32 0.32 0.34 0.34 0.34 0.34
  46   47   48   49   50   51   52   53   54   55   56   57   58   59   60
0.37 0.37 0.37 0.39 0.39 0.39 0.39 0.42 0.42 0.45 0.45 0.48 0.48 0.48 0.50
  61   62   63   64   65   66   67   68   69   70   71   72   73   74   75
0.50 0.50 0.53 0.53 0.53 0.56 0.56 0.59 0.61 0.61 0.64 0.64 0.66 0.69 0.69
  76   77   78   79   80   81   82   83   84   85   86   87   88   89   90
0.69 0.71 0.71 0.71 0.73 0.73 0.73 0.73 0.73 0.73 0.75 0.75 0.75 0.77 0.77
  91   92   93   94   95   96   97   98   99  100
0.79 0.79 0.81 0.83 0.83 0.84 0.86 0.86 0.87 0.91

> round(predict(m1, type = "response"), 2) == round(1/(1 + exp(-predict(m1))),
+     2)

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
  16   17   18   19   20   21   22   23   24   25   26   27   28   29   30 
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE 
  31   32   33   34   35   36   37   38   39   40   41   42   43   44   45 
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE 
  46   47   48   49   50   51   52   53   54   55   56   57   58   59   60 
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE 
  61   62   63   64   65   66   67   68   69   70   71   72   73   74   75 
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE 
  76   77   78   79   80   81   82   83   84   85   86   87   88   89   90 
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE 
  91   92   93   94   95   96   97   98   99  100 
TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE 
```

Let's compare the predicted logits and the corresponding probabilities:

```
> cbind(round(predict(m1), 2), round(predict(m1, type = "response"),
+     2))

     [,1] [,2]
1   -3.09 0.04
2   -2.76 0.06
3   -2.65 0.07
4   -2.54 0.07
5   -2.54 0.07
6   -2.43 0.08
7   -2.43 0.08
8   -2.20 0.10
9   -2.20 0.10
10  -2.09 0.11
11  -1.98 0.12
12  -1.98 0.12
13  -1.98 0.12
14  -1.98 0.12
15  -1.98 0.12
16  -1.98 0.12
17  -1.76 0.15
18  -1.76 0.15
19  -1.65 0.16
20  -1.65 0.16
21  -1.54 0.18
22  -1.54 0.18
23  -1.54 0.18
24  -1.54 0.18
25  -1.54 0.18
26  -1.43 0.19
27  -1.43 0.19
28  -1.32 0.21
29  -1.32 0.21
30  -1.32 0.21
31  -1.21 0.23
32  -1.21 0.23
33  -1.21 0.23
34  -1.09 0.25
35  -1.09 0.25
```

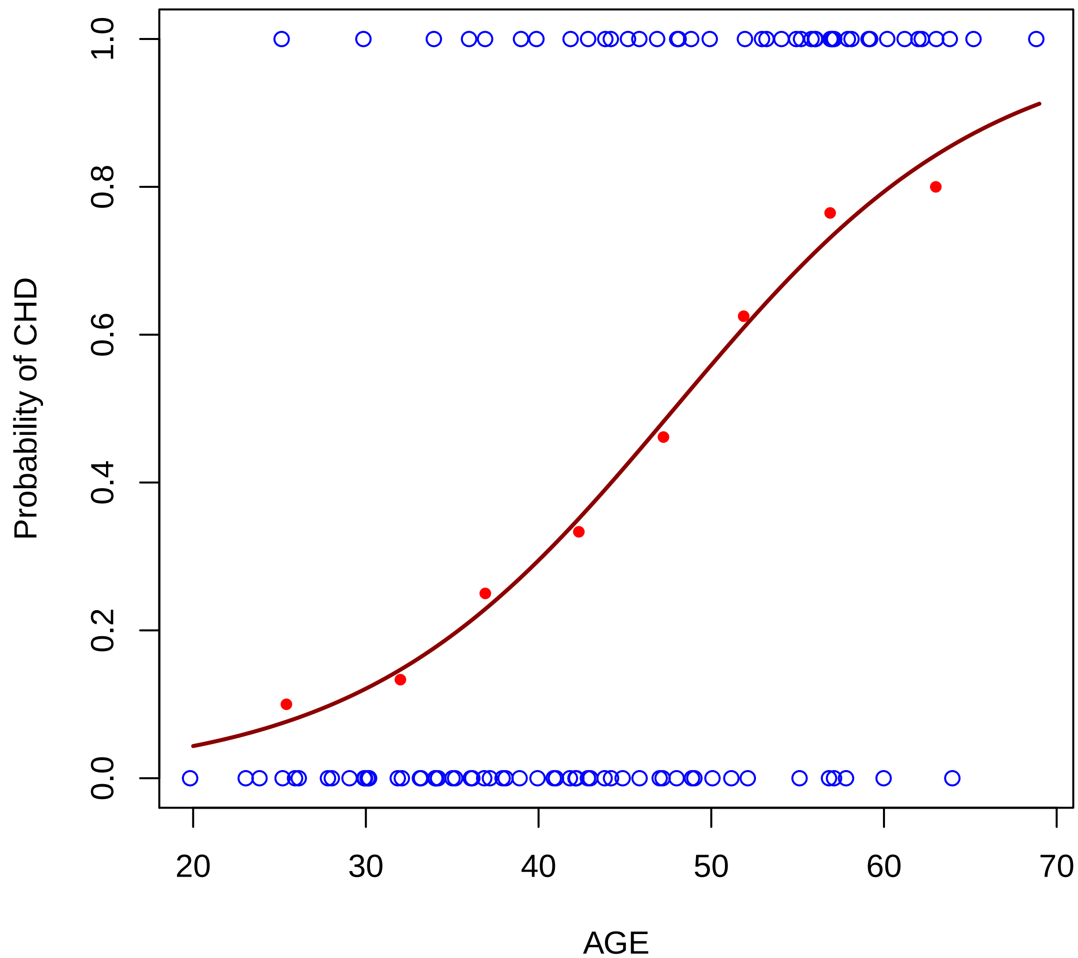
```
36  -0.98 0.27
37  -0.98 0.27
38  -0.87 0.29
39  -0.87 0.29
40  -0.76 0.32
41  -0.76 0.32
42  -0.65 0.34
43  -0.65 0.34
44  -0.65 0.34
45  -0.65 0.34
46  -0.54 0.37
47  -0.54 0.37
48  -0.54 0.37
49  -0.43 0.39
50  -0.43 0.39
51  -0.43 0.39
52  -0.43 0.39
53  -0.32 0.42
54  -0.32 0.42
55  -0.21 0.45
56  -0.21 0.45
57  -0.10 0.48
58  -0.10 0.48
59  -0.10 0.48
60   0.01 0.50
61   0.01 0.50
62   0.01 0.50
63   0.13 0.53
64   0.13 0.53
65   0.13 0.53
66   0.24 0.56
67   0.24 0.56
68   0.35 0.59
69   0.46 0.61
70   0.46 0.61
71   0.57 0.64
72   0.57 0.64
73   0.68 0.66
74   0.79 0.69
75   0.79 0.69
76   0.79 0.69
77   0.90 0.71
78   0.90 0.71
79   0.90 0.71
80   1.01 0.73
81   1.01 0.73
82   1.01 0.73
83   1.01 0.73
84   1.01 0.73
85   1.01 0.73
86   1.12 0.75
87   1.12 0.75
88   1.12 0.75
```

```
89   1.23 0.77
90   1.23 0.77
91   1.35 0.79
92   1.35 0.79
93   1.46 0.81
94   1.57 0.83
95   1.57 0.83
96   1.68 0.84
97   1.79 0.86
98   1.79 0.86
99   1.90 0.87
100  2.34 0.91
```

Rule of thumb: divide the logit by 4 and you get the approximate shift in probability relative to chance, i.e., relative to 0.5 probability.

We plot the probability of CHD against AGE, and the points corresponding to the proportions for the grouped data that we started with:

```
> attach(chage)
> plot(AGE, CHD, xlab = "AGE", ylab = "Probability of CHD", type = "n")
> points(jitter(AGE), CHD, col = "blue")
> lines(spline(AGE, predict(m1, type = "response")), col = "darkred",
+     lwd = 2)
> chagrp <- read.csv("chagrp.csv")
> head(chagrp)

  AGE AGRP CHD
1  20    1   0
2  23    1   0
3  24    1   0
4  25    1   1
5  25    1   0
6  26    1   0

> detach(chage)
> attach(chagrp)

The following object is masked _by_ .GlobalEnv:

    AGRP

> proportion.CHD <- numeric(length = 8)
> for (i in 1:8) {
+     proportion.CHD[i] <- mean(subset(chagrp, AGRP == i)$CHD)
+ }
> proportion.CHD

[1] 0.1000 0.1333 0.2500 0.3333 0.4615 0.6250 0.7647 0.8000

> mean.AGE <- numeric(length = 8)
> for (i in 1:8) {
+     mean.AGE[i] <- mean(subset(chagrp, AGRP == i)$AGE)
+ }
> mean.AGE

[1] 25.40 32.00 36.92 42.33 47.23 51.88 56.88 63.00

> points(mean.AGE, proportion.CHD, pch = 20, col = "red")
```




```
> detach(chagrp)
```

6 A couple of simple examples of GLMs

```
> webreg <- "http://www.sagepub.co.uk//wrightandlondon//"
```

6.1 Example 1: Associations with test score

The dataset: the hypothetical values received by 20 children from a standardized intelligence test that is distributed according to a standard normal $Normal(0, 1^2)$.

```
> glmexample <- read.table(paste(webreg, "glmexample.dat", sep = ""),
+     header = T)
```

There is an extra variable in the dataset that we're not interested in, so we remove it:

```
> glmexample <- glmexample[, -5]
>
> head(glmexample)

   test social books math detent
1 -1.75  -2.90     0    0      1
2 -1.18  -0.89     0    0      1
3 -0.97   0.30     0    1      1
4 -0.73  -1.44     1    1      0
5 -0.62  -1.63     0    2      0
6 -0.59  -1.49     0    2      0

> str(glmexample)

'data.frame': 20 obs. of  5 variables:
 $ test  : num  -1.75 -1.18 -0.97 -0.73 -0.62 -0.59 -0.21 -0.13 -0.12 0.07 ...
 $ social: num  -2.9 -0.89 0.3 -1.44 -1.63 -1.49 -1.45 0.79 1.25 -0.7 ...
 $ books : int  0 0 0 1 0 0 0 0 1 1 ...
 $ math  : int  0 0 1 1 2 2 2 1 1 6 ...
 $ detent: int  1 1 1 0 0 0 1 0 1 1 ...
```

We want to see how the test scores predict:

- scores from a scale of socializability (ratio variable; linear regression)
- the number of books read (count variable; Poisson regression)
- the number correct out of 10 a math quiz (binomial variable; logistic regression)
- whether the child received detention during the previous year (Bernoulli variable; logistic regression)

```
> attach(glmexample)
```

6.1.1 Model 1: Simple linear regression

(13) a. response: social scores
     b. predictor: intelligence scores

This can be done with `glm()` or `lm()`, but we use `glm()` for illustration.
Defaults for `glm()`:

- residuals are normally distributed
- the link function is the identity function

```
> socreg <- glm(social ~ test)
> summary(socreg)
```

```
Call:
glm(formula = social ~ test)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8598  -0.8897  -0.0874   1.0834   1.5773

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.222      0.266   -0.84   0.4139
test           0.874      0.246    3.55   0.0023 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.363)

    Null deviance: 41.710  on 19  degrees of freedom
Residual deviance: 24.531  on 18  degrees of freedom
AIC: 66.84

Number of Fisher Scoring iterations: 2
```

We see a positive and significant relationship between TEST and SOCIAL.
The output is a little different than the `lm()` function:

```
> socreg.lm <- lm(social ~ test)
> summary(socreg.lm)


Call:
lm(formula = social ~ test)

Residuals:
    Min      1Q  Median      3Q     Max
-1.8598 -0.8897 -0.0874  1.0834  1.5773

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.222      0.266   -0.84   0.4139
test           0.874      0.246    3.55   0.0023 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.17 on 18 degrees of freedom
Multiple R-squared:  0.412,Adjusted R-squared:  0.379
F-statistic: 12.6 on 1 and 18 DF,  p-value: 0.00229
```

The dispersion parameter (i.e., variance of the errors) is not usually mentioned with the standard regression because it is allowed to vary:

```
> summary(socreg)$dispersion

[1] 1.363
```

With the other GLMs, it can be more important because the standard deviation/variance is often assumed to be a function of the mean.

Note:

- we represented the linear model with an error term ϵ s.t. $\epsilon \sim Normal(0, \sigma)$
- the dispersion value 1.36 is the estimate of σ^2 – and it is the residual sum of squares (24.531) divided by its degrees of freedom (18)

```
> sum(residuals(socreg)^2)/18

[1] 1.363
```

The estimate for σ (i.e., the residual standard error in the `socreg.lm` output) is the square root of the dispersion:

```
> sqrt(summary(socreg)$dispersion)

[1] 1.167

> summary(socreg.lm)$sigma

[1] 1.167
```

The residual sum of squares (listed as the deviance measure) and the coefficient estimates are the same as the ones for the `lm` function:

```
> socreg$deviance

[1] 24.53

> deviance(socreg)

[1] 24.53

> sum(residuals(socreg)^2)

[1] 24.53

> sum(residuals(socreg.lm)^2)

[1] 24.53

> summary(socreg)$coef

            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.2224     0.2658 -0.8365 0.413851
test          0.8743     0.2463  3.5504 0.002286

> summary(socreg.lm)$coef

            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.2224     0.2658 -0.8365 0.413851
test          0.8743     0.2463  3.5504 0.002286
```

Statistics like R^2 are not printed, but can be easily calculated:

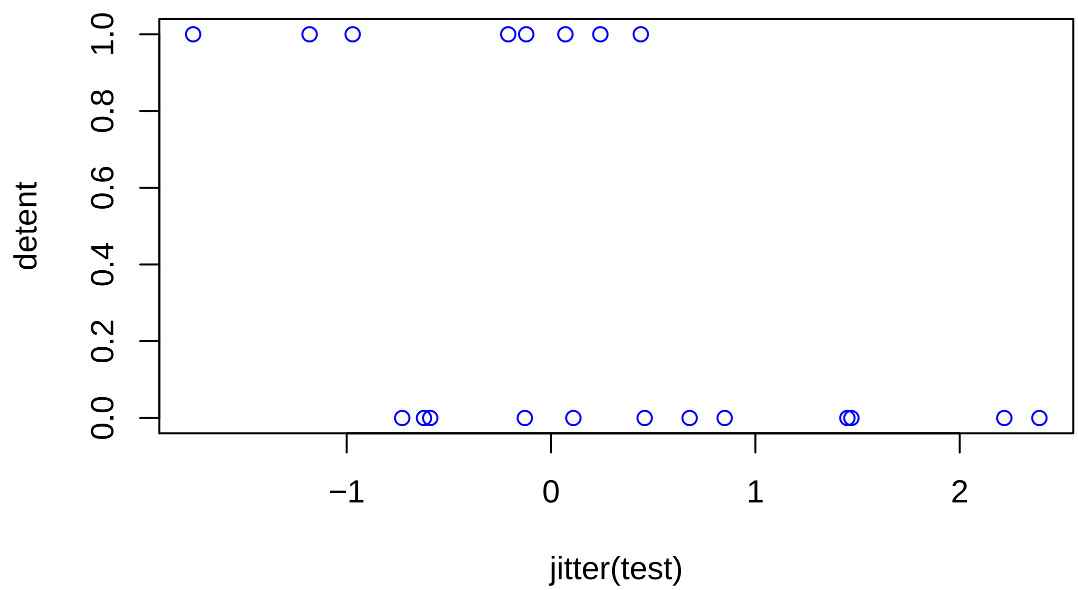
```
> socreg$deviance

[1] 24.53

> glm(social ~ 1)$deviance

[1] 41.71

> (glm(social ~ 1)$deviance - socreg$deviance)/glm(social ~ 1)$deviance

[1] 0.4119

> summary(socreg.lm)$r.squared

[1] 0.4119
```

6.1.2 Model 2: A logistic regression with a Bernoulli response

The Bernoulli response variable is a binary (two-outcome) variable, e.g., a coin flip, and we only have one observation for the coin (single coin flip).

We regress detention (Bernoulli response: either there was detention or not) on test scores:

```
> data.frame(detent, test)

   detent  test
1       1 -1.75
2       1 -1.18
3       1 -0.97
4       0 -0.73
5       0 -0.62
6       0 -0.59
7       1 -0.21
8       0 -0.13
9       1 -0.12
10      1  0.07
11      0  0.11
12      1  0.24
13      1  0.44
14      0  0.46
15      0  0.68
16      0  0.85
17      0  1.45
18      0  1.47
19      0  2.22
20      0  2.39

> plot(jitter(test), detent, col = "blue")
```




```
> detreg <- glm(detent ~ test, binomial)
> summary(detreg)


Call:
glm(formula = detent ~ test, family = binomial)

Deviance Residuals:
   Min       1Q   Median       3Q      Max
-1.459  -0.850  -0.350   0.903   1.589

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.338      0.531   -0.64    0.524
test          -1.343      0.706   -1.90    0.057 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 26.920  on 19  degrees of freedom
Residual deviance: 21.185  on 18  degrees of freedom
AIC: 25.18

Number of Fisher Scoring iterations: 5
```

We observe the following:

- there is a negative relationship between test score and detention

- the test statistic: $z = -1.9, p = .06$
- note that the test statistic is **not** t on 18 dof.s, although the results would be pretty much the same ($t(18) = 1.90, p = .06$)
- the reason we use the standard normal distribution and not a t-distribution with an appropriate number of dof.s is that, for logistic regression (same for Poisson regression), we do not need to separately estimate the variance of the residuals; once we estimate the mean, the variance is deterministically obtained for binomial / Bernoulli or Poisson distributions
- similarly, we will use χ^2 distributions (with only 1 dof parameter) for model comparison and not F distributions (with two dof parameters)

6.1.3 Model 3: A logistic regression with a binomial response (multiple coin flips)

We regress the math scores (number of correct answers out of 10) on test scores:

R has different ways to run regressions with proportions. One way is to enter the proportions as a two column matrix:

- the first column is the number of correct answers
- the second column is the number of incorrect answers

This is useful in case people have answered different numbers of questions.

```
> (x <- cbind(math, 10 - math))

      math
 [1,]    0 10
 [2,]    0 10
 [3,]    1  9
 [4,]    1  9
 [5,]    2  8
 [6,]    2  8
 [7,]    2  8
 [8,]    1  9
 [9,]    1  9
[10,]    6  4
[11,]    7  3
[12,]    7  3
[13,]    6  4
[14,]    7  3
[15,]    9  1
[16,]    9  1
[17,]   10  0
[18,]    9  1
[19,]   10  0
[20,]   10  0

> mathreg <- glm(x ~ test, binomial)
> summary(mathreg)


Call:
glm(formula = x ~ test, family = binomial)
```

```
Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.794  -0.670   0.212   0.712   1.321

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.322      0.202   -1.59     0.11
test           2.703      0.404    6.69  2.2e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 138.151  on 19  degrees of freedom
Residual deviance:  14.885  on 18  degrees of freedom
AIC: 52.8

Number of Fisher Scoring iterations: 5
```

The model shows that test scores are a significant predictor for math scores.

6.1.4 Model 4: Poisson regression

We regress the number of read books on test scores:

```
> bookreg <- glm(books ~ test, poisson)
> summary(bookreg)


Call:
glm(formula = books ~ test, family = poisson)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.481  -0.704  -0.273   0.282   1.185

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.405      0.311   -1.30     0.19
test           1.130      0.173    6.52  7.1e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 62.779  on 19  degrees of freedom
Residual deviance: 11.063  on 18  degrees of freedom
AIC: 47.27

Number of Fisher Scoring iterations: 5
```

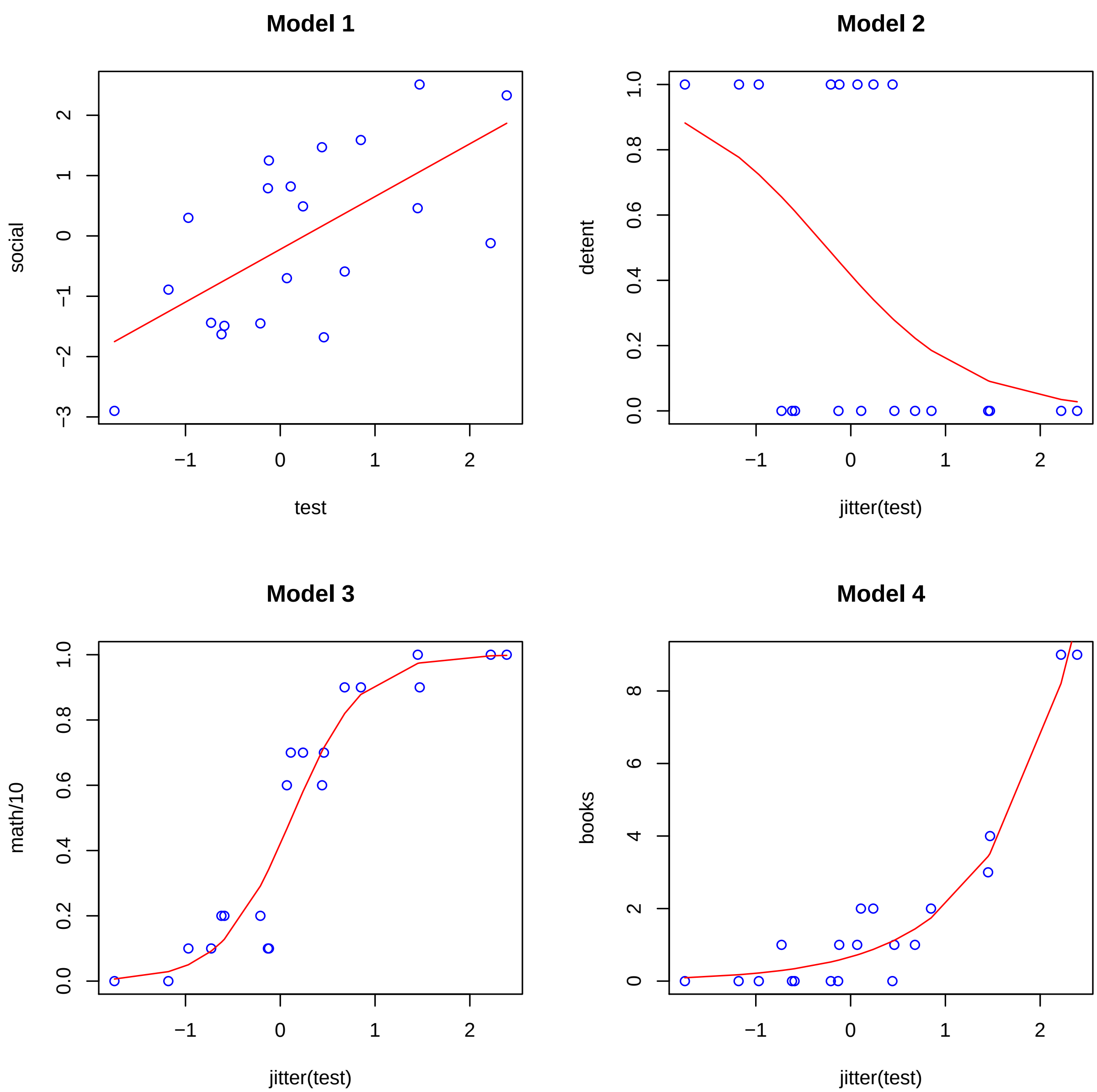
The model shows that test scores are a significant predictor for the number of read books.

6.2 Visualization

GLMs are difficult to conceptualize given that they model responses in terms of logits (for binomial / Bernoulli variables) and logs (for Poisson variables). Data visualization is all the more important to see the patterns in the data.

We draw the four graphs corresponding to the four models just discussed. The commnad `predict(glm.object, type="response")` gives us the predicted response values μ_i on the y axis:

```
> par(mfrow = c(2, 2))
> plot(test, social, col = "blue", main = "Model 1")
> lines(test, predict(socreg, type = "response"), col = "red")
> plot(jitter(test), detent, col = "blue", main = "Model 2")
> lines(test, predict(detreg, type = "response"), col = "red")
> plot(jitter(test), math/10, col = "blue", main = "Model 3")
> lines(test, predict(mathreg, type = "response"), col = "red")
> plot(jitter(test), books, col = "blue", main = "Model 4")
> lines(test, predict(bookreg, type = "response"), col = "red")
```




```
> par(mfrow = c(1, 1))
```

7 Model comparison

We often want to compare different GLMs. For example, we could be interested in whether including the social scores in addition to test scores improves the predictions for detention.

```
> detreg <- glm(detent ~ test, binomial)
> summary(detreg)


Call:
```

```
glm(formula = detent ~ test, family = binomial)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.459  -0.850  -0.350   0.903   1.589

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.338      0.531   -0.64    0.524
test          -1.343      0.706   -1.90    0.057 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 26.920  on 19  degrees of freedom
Residual deviance: 21.185  on 18  degrees of freedom
AIC: 25.18

Number of Fisher Scoring iterations: 5

> det2 <- glm(detent ~ test + social, binomial)
> summary(det2)


Call:
glm(formula = detent ~ test + social, family = binomial)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.430  -0.873  -0.243   0.687   1.601

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.290      0.560   -0.52     0.60
test          -2.224      1.133   -1.96     0.05 *
social         0.730      0.597    1.22     0.22
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 26.920  on 19  degrees of freedom
Residual deviance: 19.412  on 17  degrees of freedom
AIC: 25.41

Number of Fisher Scoring iterations: 5
```

ANOVA shows that the main effect of the additional term is non-significant:

```
> anova(detreg, det2, test = "Chi")

Analysis of Deviance Table
```

```
Model 1: detent ~ test
Model 2: detent ~ test + social
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        18       21.2
2        17       19.4  1     1.77     0.18
```

We specified `test="Chi"`; had we said `test="F"`, R would print a warning that the F test is not appropriate in this circumstance:

```
> anova(detreg, det2, test = "F")

Warning:  using F test with a 'binomial' family is inappropriate

Analysis of Deviance Table

Model 1: detent ~ test
Model 2: detent ~ test + social
  Resid. Df Resid. Dev Df Deviance    F Pr(>F)
1        18       21.2
2        17       19.4  1     1.77 1.77   0.18
```

7.1 Deviance and log-likelihood ratios

A log-likelihood ration (LRT) is the log of the likelihood ratio between two models, i.e., `log(likelihood ratio)`.

(14) The likelihood ratio Λ (capital lambda):

a. $\Lambda = \frac{\text{maximum likelihood for model } H_0}{\text{maximum likelihood for more complex model}}$

b. $\log(\Lambda) = \log(\frac{\text{maximum likelihood for model } H_0}{\text{maximum likelihood for more complex model}})$
$= \log(\text{maximum likelihood for model } H_0) - \log(\text{maximum likelihood for more complex model})$

(15) The test statistic for the likelihood ratio: $G^2 = -2 \times \log(\Lambda)$

a. G^2 will take a minimum value of 0 when the likelihood of the two models is identical

b. G^2 will take higher values as the more complex model becomes more likely

(16) Residual deviance: the difference in G^2 between the saturated model that has a separate parameter for each response value (20 parameters in this case – since we have 20 responses) and the fitted model.

- since the saturated model has a parameter / predictor for each response value, it captures (basically memorizes) the response values perfectly

(17) Degrees of freedom: the change in the number of estimated parameters.

We want the deviance to be as small as possible – just as we wanted the OLS error, i.e., the (mean) sum of squared residuals, to be as small as possible.

```
> summary(detreg)


Call:
glm(formula = detent ~ test, family = binomial)

Deviance Residuals:
```

```
   Min      1Q  Median      3Q     Max
-1.459  -0.850  -0.350   0.903   1.589

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.338      0.531   -0.64    0.524
test          -1.343      0.706   -1.90    0.057 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 26.920  on 19  degrees of freedom
Residual deviance: 21.185  on 18  degrees of freedom
AIC: 25.18

Number of Fisher Scoring iterations: 5

> summary(det2)


Call:
glm(formula = detent ~ test + social, family = binomial)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-1.430  -0.873  -0.243   0.687   1.601

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.290      0.560   -0.52     0.60
test          -2.224      1.133   -1.96     0.05 *
social         0.730      0.597    1.22     0.22
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 26.920  on 19  degrees of freedom
Residual deviance: 19.412  on 17  degrees of freedom
AIC: 25.41

Number of Fisher Scoring iterations: 5
```

(18) Null deviance: the difference in G^2 between the saturated model and the intercept/mean-only model.

```
> summary(glm(detent ~ 1, binomial))


Call:
glm(formula = detent ~ 1, family = binomial)
```

```
Deviance Residuals:
   Min      1Q  Median      3Q     Max  
 -1.01   -1.01   -1.01    1.35    1.35  

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -0.405      0.456   -0.89     0.37

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 26.92  on 19  degrees of freedom
Residual deviance: 26.92  on 19  degrees of freedom
AIC: 28.92

Number of Fisher Scoring iterations: 4
```

The difference in deviances can be tested against the χ^2 distribution for significance:

- we extract the deviances

```
> deviance(detreg)

[1] 21.18

> deviance(det2)

[1] 19.41

> (deviance_difference <- deviance(detreg) - deviance(det2))

[1] 1.773
```

- we extract the degrees of freedom

```
> df.residual(detreg)

[1] 18

> df.residual(det2)

[1] 17

> (df_difference <- df.residual(detreg) - df.residual(det2))

[1] 1
```

- we compute the p-value

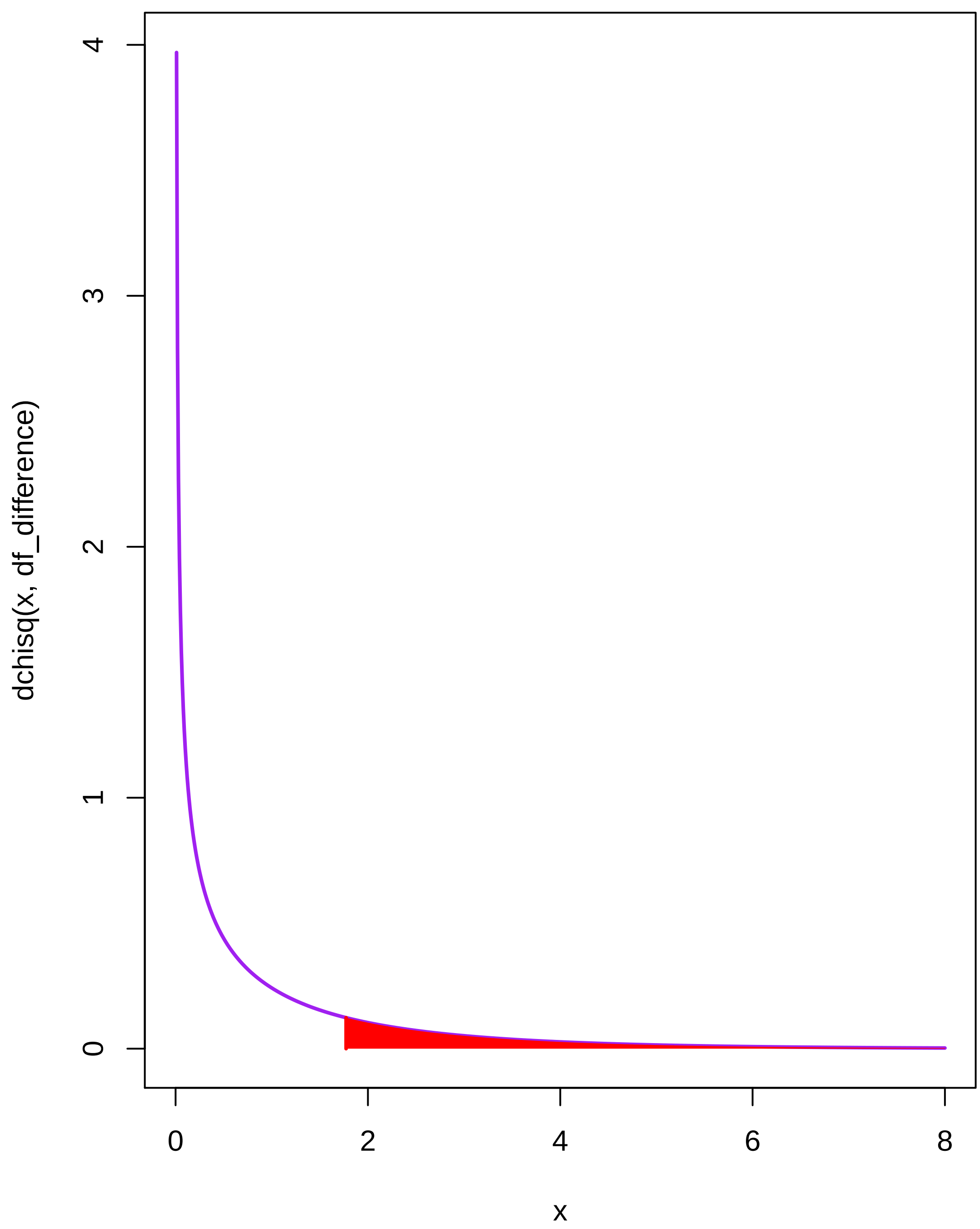
```
> 1 - pchisq(deviance_difference, df_difference)

[1] 0.183
```

The reduction in deviance is not significant at the .05 level. Or, in one go:

```
> anova(detreg, det2, test = "Chi")

Analysis of Deviance Table

Model 1: detent ~ test
Model 2: detent ~ test + social
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        18       21.2
2        17       19.4  1     1.77     0.18
```

The p-value is the probability of a χ^2 value at least as extreme as the observed one:

```
> pchisq(deviance_difference, df_difference)

[1] 0.817

> x <- seq(0, 8, by = 0.01)
> plot(x, dchisq(x, df_difference), type = "l", lwd = 2, col = "purple",
+     main = "")
> segments(deviance_difference, 0, deviance_difference, dchisq(deviance_difference,
+     df_difference), col = "red", lwd = 2)
> coord.x <- c(deviance_difference, x[x >= deviance_difference], max(x))
> coord.y <- c(0, dchisq(x[x >= deviance_difference], df_difference),
+     0)
> polygon(coord.x, coord.y, col = "red", border = NA)
```



7.2 Background on likelihood functions and maximum likelihood estimates (MLEs)

When we think of the Bernoulli pmf as a function with θ fixed that gives the probabilities of the possible outcomes, we have a probability distribution function: the arguments are possible outcomes and the values are their probabilities.

In practice, however, we have a fixed data set and we want to know what its probability is according to the Bern function relative to *different* possible biases, i.e., different thetas.

In this case, we have a function that takes different thetas and returns the probability of the fixed (actual) data set for those thetas.

This is not a probability distribution function: we do not assign probabilities to the different thetas.

We call this the Bern likelihood function to indicate that it is a function of theta; the formula is the same, but the functions are different.

Using lambda notation, we have the following:

(19) prob. dist. function: $\lambda y.Bern(y, \theta)$

- note that θ is contextually provided here, i.e., it is a free variable, while y is bound

(20) likelihood function: $\lambda \theta.Bern(y, \theta)$

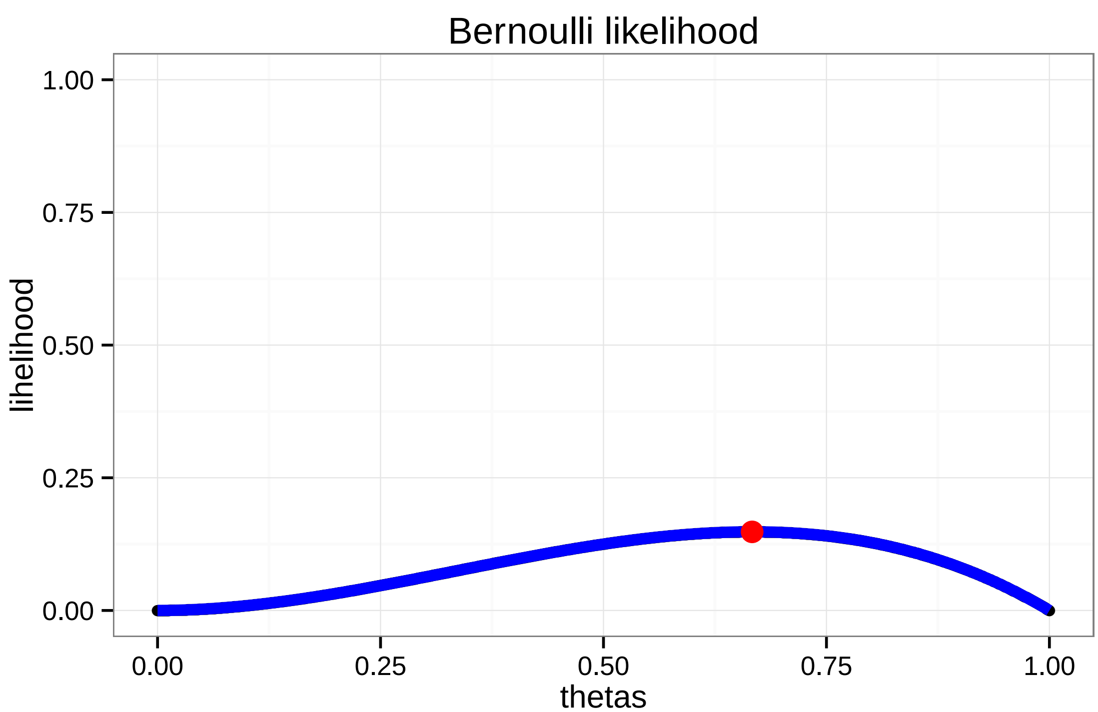
- note that y is contextually provided here, i.e., it is a free variable, while θ is bound

```
> y <- c(1, 1, 0)
> Bern <- function(y, theta) {
+     prod(theta^y * (1 - theta)^(1 - y))
+ }
> theta <- 0.4
> Bern(y, theta)

[1] 0.096

> theta * theta * (1 - theta)

[1] 0.096

> theta <- 0.3
> Bern(y, theta)

[1] 0.063

> theta * theta * (1 - theta)

[1] 0.063

> theta <- 0.5
> Bern(y, theta)

[1] 0.125

> theta * theta * (1 - theta)

[1] 0.125

> theta <- 0.6
> Bern(y, theta)

[1] 0.144

> theta * theta * (1 - theta)

[1] 0.144
```

Given a fixed, contextually provided data set y, the θ that maximizes the value of the likelihood function is called the *Maximum Likelihood Estimate (MLE)*.

In our case, this is the highest point of the curve plotted below:

```
> thetas <- seq(0, 1, length.out = 1000)
> thetaLikelihoods <- vector(length = length(thetas))
> for (i in 1:length(thetas)) {
+     thetaLikelihoods[i] <- Bern(y, thetas[i])
+ }
> MLE <- sum(y)/length(y)
> MLE

[1] 0.6667

> which(thetaLikelihoods == max(thetaLikelihoods))

[1] 667

> thetas[which(thetaLikelihoods == max(thetaLikelihoods))]

[1] 0.6667

> library("ggplot2")
> qplot(thetas, thetaLikelihoods, ylim = range(0, 1), xlab = "thetas",
+     ylab = "lihelihood", main = "Bernoulli likelihood") + geom_line(size = 2,
+     col = "blue") + geom_point(aes(MLE, Bern(y, MLE)), col = "red",
+     size = 4) + theme_bw() + theme(legend.position = "none")
```

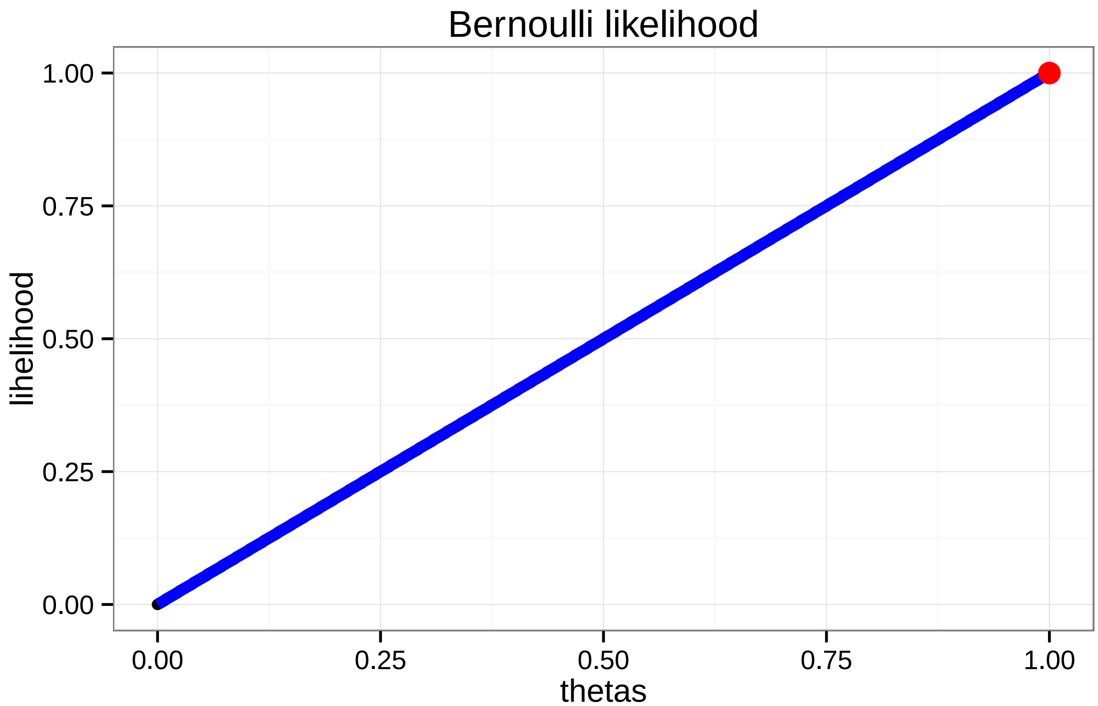


Note that the Bernoulli likelihood is not a prob. density function – it does not integrate to 1:

```
> BernLike <- function(theta) {
+     as.numeric(Bern(y, theta))
+ }
```

```
> BernLike(MLE)

[1] 0.1481

> integrate(Vectorize(BernLike), lower = 0, upper = 1)

0.08333 with absolute error < 9.3e-16
```

```
> y <- 1
> thetas <- seq(0, 1, length.out = 1000)
> thetaLikelihoods <- vector(length = length(thetas))
> for (i in 1:length(thetas)) {
+     thetaLikelihoods[i] <- Bern(y, thetas[i])
+ }
> MLE <- sum(y)/length(y)
> MLE

[1] 1

> which(thetaLikelihoods == max(thetaLikelihoods))

[1] 1000

> thetas[which(thetaLikelihoods == max(thetaLikelihoods))]

[1] 1

> library("ggplot2")
> qplot(thetas, thetaLikelihoods, ylim = range(0, 1), xlab = "thetas",
+     ylab = "lihelihood", main = "Bernoulli likelihood") + geom_line(size = 2,
+     col = "blue") + geom_point(aes(MLE, Bern(y, MLE)), col = "red",
+     size = 4) + theme_bw() + theme(legend.position = "none")
```




```
> integrate(Vectorize(BernLike), lower = 0, upper = 1)

0.5 with absolute error < 5.6e-15
```

7.3 Evaluating the interaction model

The interaction also fails to significantly improve the fit of the model:

```
> det3 <- glm(detent ~ test * social, binomial)
> anova(det2, det3, test = "Chi")

Analysis of Deviance Table

Model 1: detent ~ test + social
Model 2: detent ~ test * social
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        17       19.4
2        16       19.3  1    0.122     0.73

> anova(detreg, det3, test = "Chi")

Analysis of Deviance Table

Model 1: detent ~ test
Model 2: detent ~ test * social
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1        18       21.2
2        16       19.3  2      1.9     0.39
```

7.4 Adding polynomial functions as additional predictors

We can include polynomial functions within the `glm` function (recall that 'linear' is in terms of the coefficients β, not in terms of the predictors X).

For example, we might want to check whether adding a quadratic test term (the 2 in the `poly` function) significantly improves the prediction of the social scores.

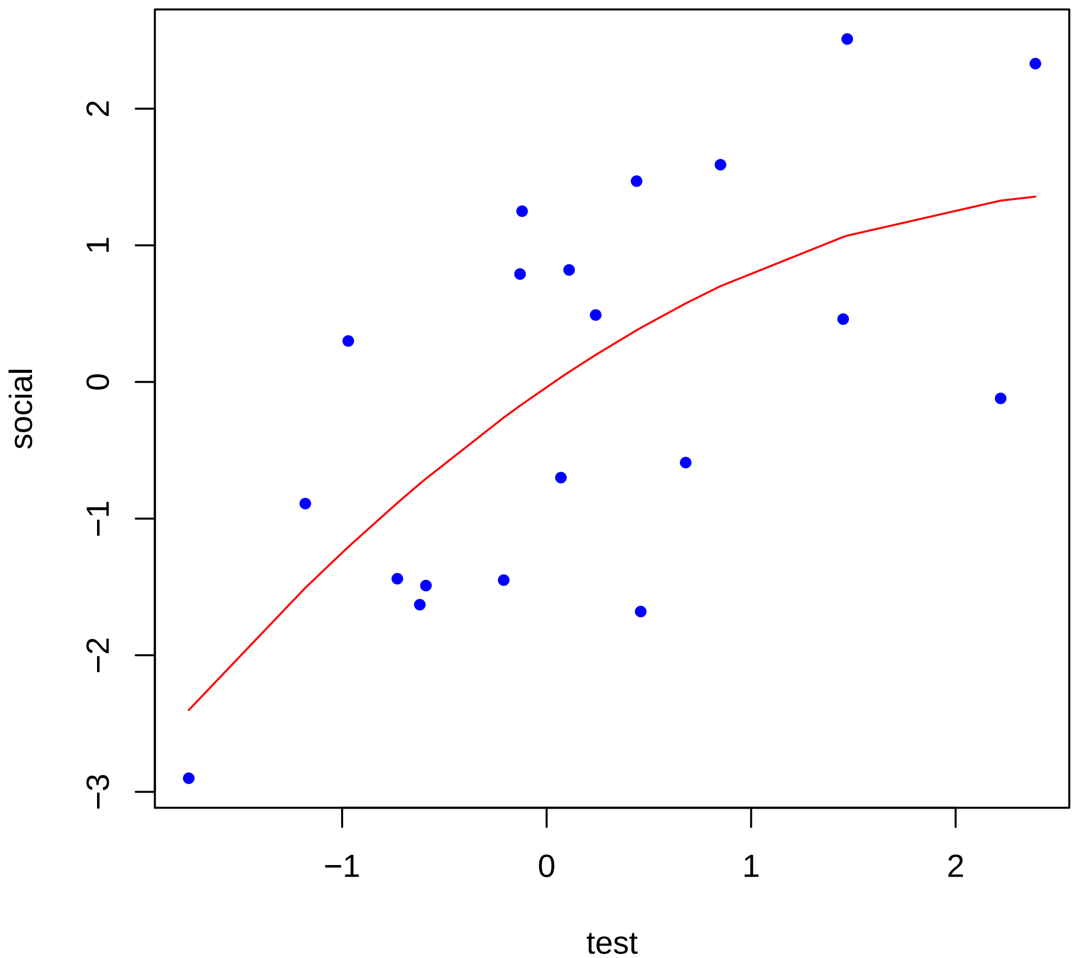
```
> socialpoly <- glm(social ~ poly(test, 2))
```

It does not:

```
> summary(socialpoly)


Call:
glm(formula = social ~ poly(test, 2))

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-2.076  -0.808  -0.103   0.966   1.507

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      -0.044      0.262   -0.17   0.8684
poly(test, 2)1    4.145      1.169    3.54   0.0025 **
poly(test, 2)2   -1.133      1.169   -0.97   0.3461
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 1.367)

    Null deviance: 41.710  on 19  degrees of freedom
Residual deviance: 23.247  on 17  degrees of freedom
AIC: 67.77

Number of Fisher Scoring iterations: 2
```

The curve does not deviate much from the linear:

```
> plot(test, social, col = "blue", pch = 20)
> lines(test, predict(socialpoly, type = "response"), col = "red", )
```




```
> detach(glmexample)
```

References

Abelson, R.P. (1995). *Statistics as Principled Argument*. L. Erlbaum Associates.

Agresti, A. (2002). *Categorical Data Analysis*. Wiley.

Baayen, R. Harald (2008). *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*. Cambridge University Press.

Braun, J. and D.J. Murdoch (2007). *A First Course in Statistical Programming with R*. Cambridge University Press.

De Veaux, R.D. et al. (2005). *Stats: Data and Models*. Pearson Education, Limited.

Diez, D. et al. (2013). *OpenIntro Statistics: Second Edition*. CreateSpace Independent Publishing Platform. URL: http://www.openintro.org/stat/textbook.php.

Faraway, J.J. (2004). *Linear Models With R*. Chapman & Hall Texts in Statistical Science Series. Chapman & Hall/CRC.

Faraway, J.J. (2006). *Extending the Linear Model With R*. Chapman & Hall Texts in Statistical Science Series. Chapman & Hall/CRC.

Gelman, A. and J. Hill (2007). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press.

Gries, S.T. (2009). *Quantitative Corpus Linguistics with R: A Practical Introduction*. Taylor & Francis.

— (2013). *Statistics for Linguistics with R: A Practical Introduction, 2nd Edition*. Mouton De Gruyter.

Hosmer, D.W. and S. Lemeshow (2000). *Applied Logistic Regression*. 2nd ed. Wiley.

Johnson, K. (2008). *Quantitative methods in linguistics*. Blackwell Pub.

Kruschke, John K. (2011). *Doing Bayesian Data Analysis: A Tutorial with R and BUGS*. Academic Press/Elsevier.

Miles, J. and M. Shevlin (2001). *Applying Regression and Correlation: A Guide for Students and Researchers*. SAGE Publications.

Ramsey, Fred L. and Daniel W. Schafer (2002). *The statistical sleuth: A course in methods of data analysis*. 2nd ed. Duxbury Press.

Wright, D.B. and K. London (2009). *Modern regression techniques using R: A practical guide for students and researchers*. SAGE.

Xie, Yihui (2013). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC.