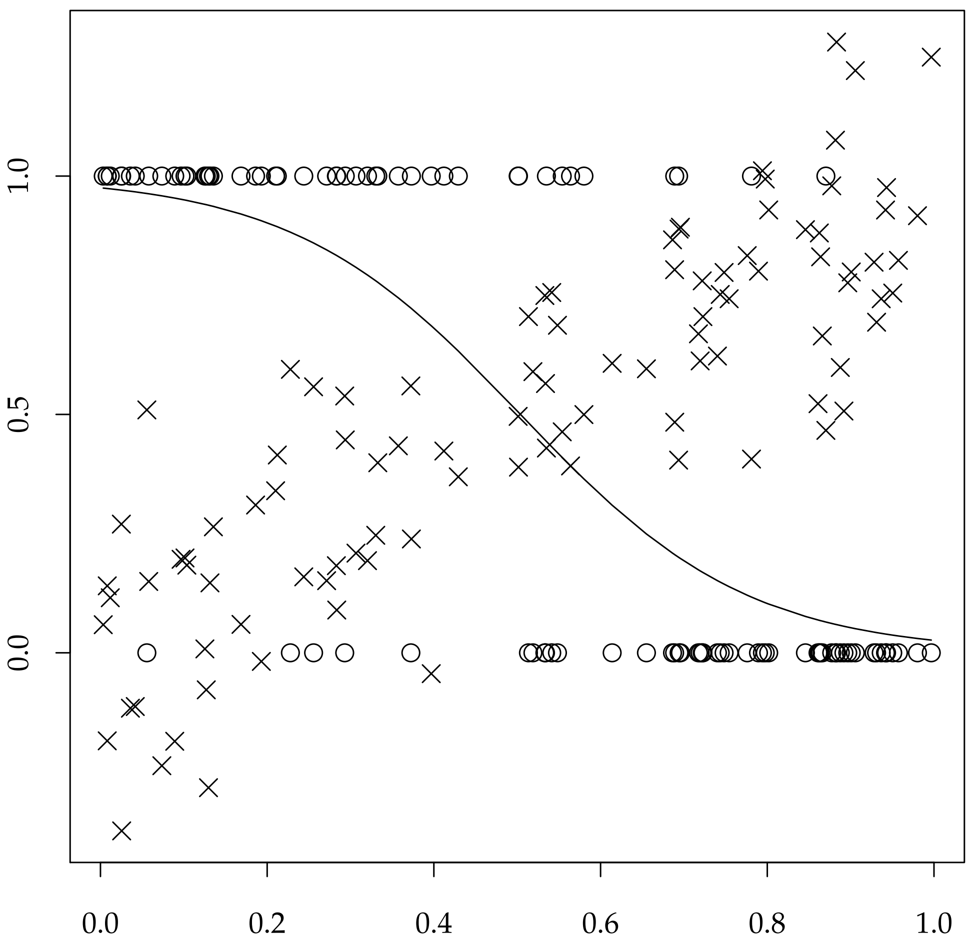
Discrete choice — 2015



Oliver Kirchkamp

Contents

On the homepage of the course I recommend a couple of good textbooks. The purpose of this handout is to give you some support during the course, but not to replace any of these textbooks. This handout will also contain numerous mistakes. If this still does not scare you off, please read on :-)

Overview Aim of the course: you should be able to apply nonlinear models / choice models

- Maximum Likelihood
- Nonlinear Regressions
- Discrete-Choice-Models
- Count Data
- Survival Models

Literature

- William Greene, Econometric Analysis, Prentice Hall, 7th Edition, 2011.
- Michael Murray; Econometrics, a Modern Introduction; Pearson 2006
- Christian Gourieroux and Alain Monfort, Statistics and Econometric Models, Vol. 1, Cambridge, 1995.
- John K. Kruschke , Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan. Academic Press, 2nd Edition, 2014.

Examples The examples in the course will be based on R. Let us compare R with alternative packages:

- R
  - gratis *and* libre
  - wide scope
  - well documented (documentation is often not gratis)
  - homogeneous structure
  - "hip"
  - Sweave allows to keep code in the paper
- SAS, STATA, EViews, TSP, SPSS,...

- more specialised
- more fragmented structure
- neither gratis *nor* libre
  - You do not want to recode everything everytime you change your workplace.
  - You do not want your code seized as a hostage by your employer.

1. Introduction

1.1. Economic relationships

Let us collect a few economic relationships

Discrete choice models, qualitative response (QR) models

number of patents	count data
labor force participation	qualitative (binary) choice
consumption	qualitative choice
opinions given on Likert scales (strongly disagree, disagree, indifferent, agree, strongly agree)	ordered choice
field of study	unordered choice

1.2. Digression: Notation

We will illustrate most of our examples with R. The input will be shown with a vertical bar on the left, and the output will be shown in a frame, like this:

```
1+1

[1] 2
```

To accomodate those of you how are already familiar with Stata, we will also (sometimes) have a look at the Stata notation. Stata input will be shown like this:

```
display 1+1
```

In any case it is strongly recommended that you try out the code yourself.

2. Estimation of Parameters

2.1. Estimations based on real world data

Let us do something that we already know. Let us run an OLS regression.
Very often we estimate parameters like this:

```
library(Ecdat)
data(Caschool)
lm (testscr ~ str,data=Caschool)


Call:
lm(formula = testscr ~ str, data = Caschool)

Coefficients:
(Intercept)          str
     698.93        -2.28
```

This is interesting if we want to learn something new about the data (or the world). However, with data from the real world we learn not much about our estimation techniques. Are they precise, efficient, consistent, etc.?

These properties can be visualised, if we first simulate a given relationship and then estimate it. We will do this with more complicated models later, but first we will use simple OLS.

2.2. Estimations based on simulated data

Assume that the true relationship is

$$y = 12 + 4.7x + e$$

where $u \sim N(0, 13)$

We want to draw 100 samples where x is taken from $N(0, 1)$. In a first step we initialise our random number gernerator:

```
set.seed(123)
```

We then store 100 randomly drawn values for x and the error term e in two vectors `x` and `e`. We also calculate our *dependent* variable y and store it in a vector `y`

```
x <- rnorm(100)
e <- rnorm (100)
y <- 12 + 4.7 * x +13 * e
```

Estimating a linear model can be done with the command `lm`. We store the estimation result in a variable `yx.lm`. This variable is actually an object that contains a lot of information regarding the regression.

```
yx.lm <- lm(y ~ x )
```

To have a brief look at this object, we can say

```
yx.lm


Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
      10.66         4.02
```

A more detailed summary can be obtained with

```
summary(yx.lm)


Call:
lm(formula = y ~ x)

Residuals:
   Min     1Q Median     3Q    Max
-24.80  -8.89  -1.14   7.55  42.78

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    10.66       1.27    8.41  3.4e-13 ***
x               4.02       1.39    2.89   0.0047 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 12.6 on 98 degrees of freedom
Multiple R-squared:  0.0786,Adjusted R-squared:  0.0692
F-statistic: 8.36 on 1 and 98 DF,  p-value: 0.00472
```

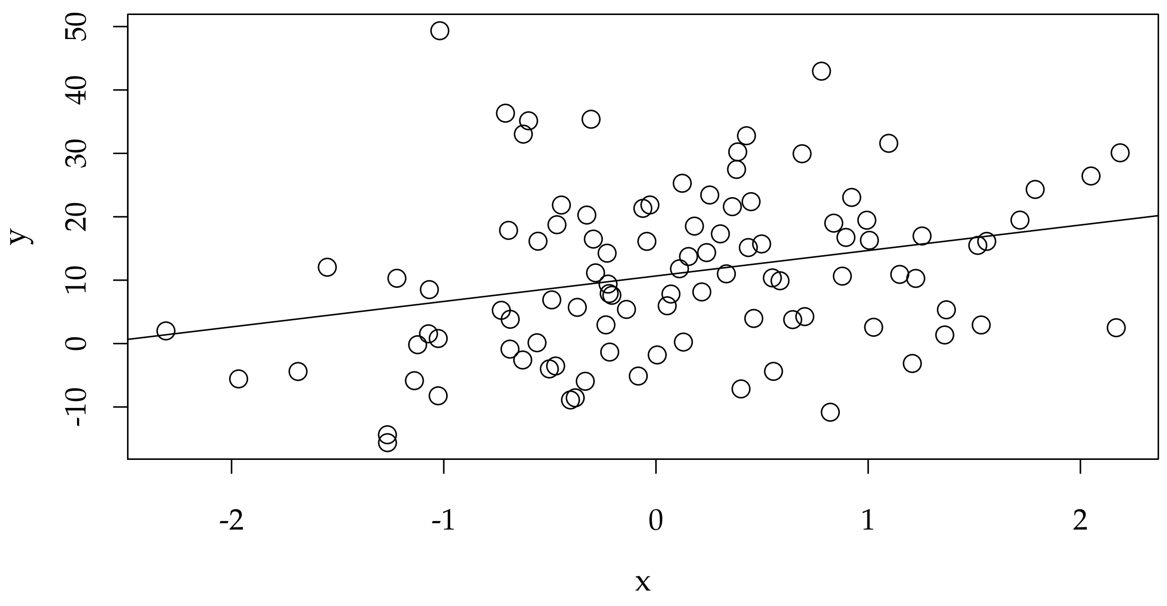
If we are interested in a single coefficient of the model (as we will be below) we use the exctractor function `coef` (the function is called "extractor function" since it extracts something from an object, here a coefficient).

```
coef(yx.lm)

(Intercept)           x
     10.664       4.018

confint(yx.lm)

            2.5 % 97.5 %
(Intercept) 8.147 13.180
x           1.261  6.775
```

There are more "extractor functions", like `AIC`, `plot`, `predict`,. . .

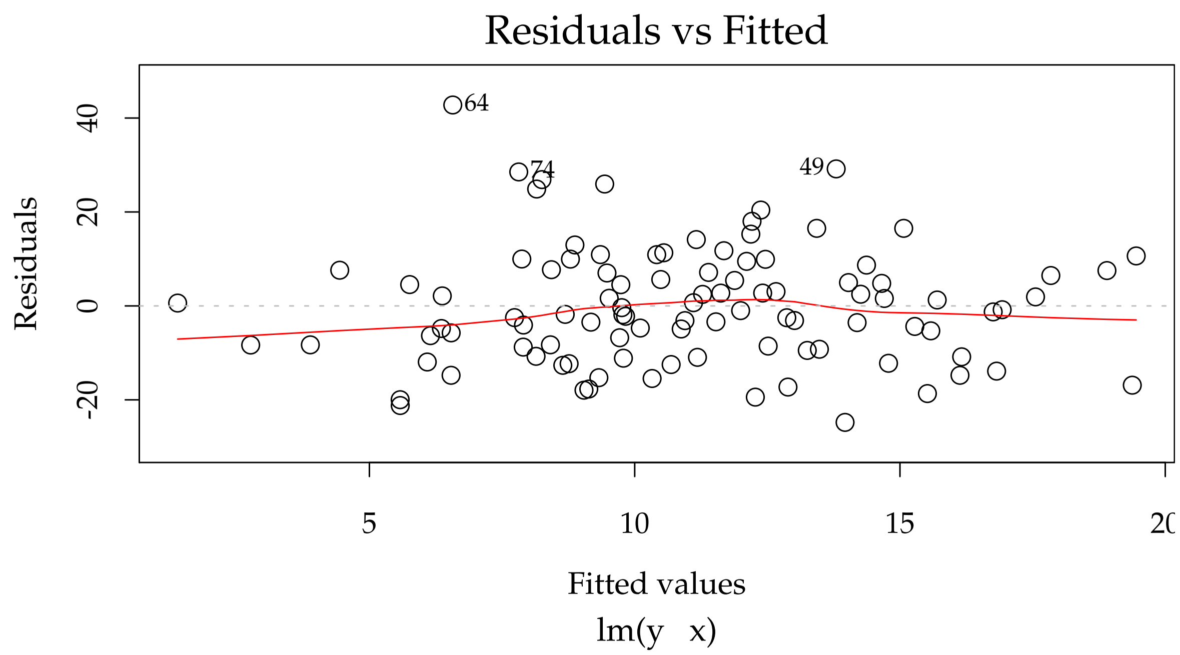
The function `plot` can give us a scatterplot of `y` over `x` (actually, the function `plot` can do a lot more, we will see this later in the course). The function `abline` draws lines, here it draws the estimated regression line.

```
plot(y ~ x)
abline(yx.lm)
```



Since we will talk about residuals below, let us also have a look at the residuals of this regression:

```
plot(yx.lm,which=1)
```



Of course, the same result can be obtained with many other packages. Here is the same exercise in Stata. In contrast to R, Stata can have only a single dataset at a time in its memory. To create a new dataset in Stata, we have to delete the old dataset and specify the number of observations in the new dataset:

```
drop _all
set obs 100
```

As above, we set the starting value for the random number generator and create our variables `x`, `e`, and `y`. Stata does not immediately provide random numbers that follow a normal distribution, but it is possible to derive them from uniformly distributed random numbers.

```
set seed 123
gen x=invnormal(uniform())
gen e=invnormal(uniform())
gen y=12 + 4.7 * x + 13 * e
```

Stata also does not allow to store estimation results in specific variables, hence the regression command is a bit shorter. As you will see, Stata implicitely assumes that we want to see a summary of the results.

```
reg y x
```

Nevertheless, not all is lost. Stata always stores some results of the last command in three lists, `ereturn`, `return`, and `sreturn`. The coefficients are stored in `ereturn`. We can get the coefficient with the following sequence of commands:

```
ereturn list
matrix coeff=e(b)
display coeff[1,1]
```

As in R, also Stata can draw a scattergram.

```
twoway scatter y x
predict yhat
twoway (line yhat x) (scatter y x)
```

We should note that the regression line in the Stata diagram is constructed in an inefficient way — Stata actually draws 99 connecting segments which go back and forth over the diagram. It is possible to fix this, though, but we leave this as an exercise.

What we have learned so far

	R	STATA
generating random numbers	`rnorm()`	`set obs...` `gen ...= invnormal( uniform() )`
plotting	`plot()`	`twoway scatter, twoway line`
OLS	`lm()`	`reg`
getting results	`<-`	`ereturn list, return list, sreturn list`

Difference between R and STATA

	R	STATA
datatypes	characters, numbers, vectors, matrices, arrays, dataframes, lists, objects	macros, scalars, matrices(1), matrices(2), dataframe
languages	1 language (object oriented)	3 languages: • for dataframes • for matrices (1) • MATA (for matrices (2))
limits	2^{31-1} vector length, Unix: 128Tb, MSWindows: ~2Gb	IC: 1 dataframe, 2047 variables, 800 mat.size, 255 string size, 1 core

Now let us do many regressions

(here we do it to simulate many estimations, in other applications we may want to estimate microbehaviour for each entity separately)

To do that, let us first write a function that handles one regression for us. Note that this function takes an argument which is not used at all in the function. The argument is just a dummy that allows to use `sapply`. Nevertheless the function returns something, namely the estimated coefficients.

```
myReg <- function (zz) {
  x <- rnorm(100)
  e <- rnorm (100)
  y <- 12 + 4.7*x +13 * e
  coef(lm(y ~ x ))
}
```

Now let us apply this function once.

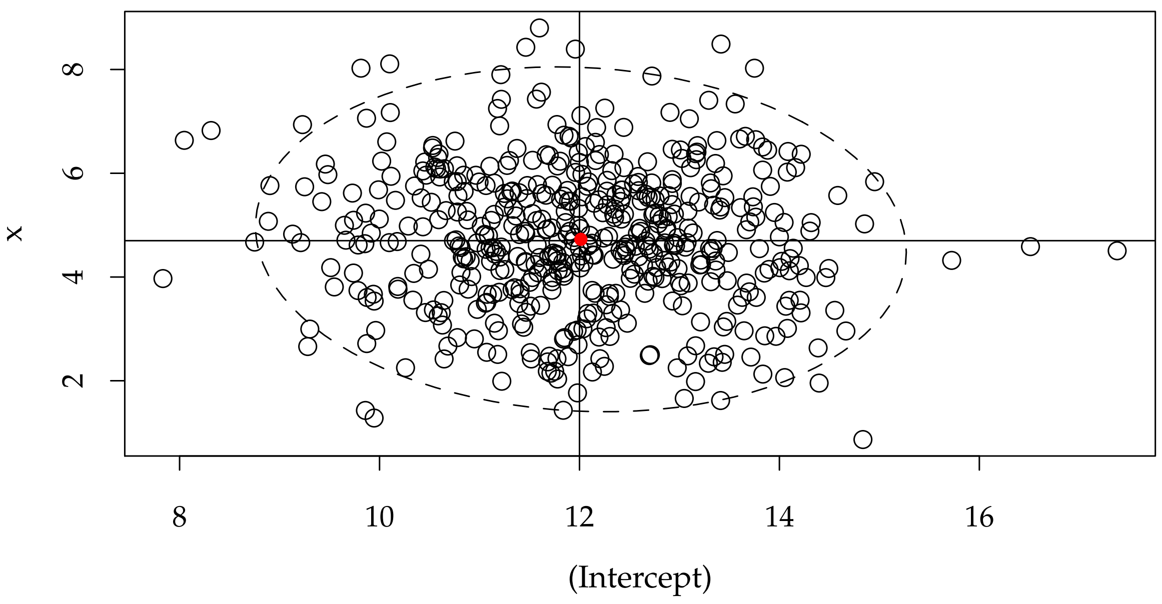
```
myReg(1)

(Intercept)            x
     11.606        4.062
```

And now we apply this function 500 times (i.e. we apply it to the list of all numbers from 1 to 500. This gives us a list of 500 estimation results which we store in `coeffs`).

```
coeffs <- sapply(1:500,myReg)
```

Next we plot this list as a scattergram. We have to transpose `coeffs` with `t` since plots expects a matrix with two columns (when we ask it to plot a matrix). Also `cov` expects a matrix with two columns.

```
plot(t(coeffs))
abline(h=4.7)
abline(v=12)
emean<-apply(coeffs,1,mean)
points(t(emean),col="red",pch=20)
library(ellipse)
lines(ellipse(cov(t(coeffs)),centre=emean),lty=2)
```



Let us briefly see, how this could be done in Stata:

```
capture program drop myReg
program define myReg
drop _all
quietly set obs 100
gen x=invnormal(uniform())
gen e=invnormal(uniform())
gen y=12 + 4.7*x +13 * e
quietly reg y x
end
capture matrix drop coeffs
set matsize 500
foreach i of numlist 1/500 {
  myReg
  matrix coeffs=nullmat(coeffs)\e(b)
}
matrix list coeffs
drop _all
svmat coeffs
twoway scatter coeffs1 coeffs2,yline(4.3) xline(12)
```

I do not know how to draw the ellipse in Stata — it is certainly possible somehow. If somebody knows, please tell me.

2.3. Digression: Accessing specific variables in specific dataframes

Often we refer in R or in Stata to a part of a dataframe.

2.3.1. Accessing certain variables from a dataframe

In R dataframes can be seen as lists. As such, a variable in a dataframe can be addressed with list notation.

Different from Stata, R has several dataframes in memory at the same time. We have to specify dataframe and variable:

```
dataframe$column
dataframe[["column"]]
dataframe[,"column"]
```

where `column` is the name of the variable. The two notations are equivalent. The first is shorter, the second is convenient for programming if `"column"` is not fixed but something that changes, an expression or a variable. So we could actually write

```
x <- "column"
dataframe[[x]]
```

But this is more advanced and you should not worry if you find this confusing.

Finally, we could also use the notation that selects (named) columns from a matrix:

```
dataframe[,x]
```

Many functions in R have a `data` option. This option allows to specify a dataframe that is first searched for names which are used as arguments for the function. E.g., in the following

```
lm (y ~ x, data=dataframe)
```

the variables `y` and `x` are first used in `dataframe`

When we repeatedly want to use different variables from always the same dataframe, `with` can be helpful:

```
with(dataframe, lm(y ~ x) )
```

`with` makes sure that names like `x` and `y` are first searched in the namespace of `dataframe`.

Sometimes we want to execute even several statements within the context of one dataframe

```
with(dataframe,{  z <- lm (x ~ y)
                  print(z)
                  z2 <- lm (log(x) ~ log(y))
                  print(z2)
                  } )
```

Again, `with` makes sure that names like `column1` and `column2` are first searched in the namespace of `with`.

If, for some reason, we want to change a dataframe, `within` is helpful:

```
dateframe2 <- within(dataframe,{ x <- x + 2
                                 y <- y - 2 * x
                                 x <- NULL } )
```

`within` first changes `x` and `y` within a copy of `dataframe` (the original remains unchanged) and then assigns this copy to `dataframe2`.

Finally, we can move a dataframe into the foreground with `attach`:

```
attach(dataframe)
lm( y ~ x)
detach(dataframe)
```

Since Stata only knows a single dataframe at a time, we have fewer options there. Dataframes are always loaded from disk with `use`

```
use dataframe
```

and then they are ready to use. Names of variables always refer to the *current* dataframe.

```
reg y x
```

Also assignments always modify the *current* dataframe

```
gen z = x + y
```

If you want to change other dataframes in Stata, you have to save the current dataframe somewhere (at least if you want to come back to it), load the other dataframe, modify it, save it back, and revert to the previous dataframe.

Hence, generating a new dataframe from a present dataframe requires some planning with Stata since the two can never coexist. Small new dataframes can first be stored in a matrix. Larger dataframes can be created as a file on disk and appended sequentially. As long as the new dataframe is certainly smaller than the present one, it is also possible to add a few columns to the present dataframe as a "temporary" store for the new dataframe.

2.3.2. Accessing subsets of a dataframe

If we want to access only a subset of a dataframe in R then use the function `subset`

```
subset(y, x>5)
```

returns the part of the vector or matrix or dataframe y where the values of x are larger then 5. Many functions also have a subset option.

```
lm(y ~ x, subset = x>5)
```

In Stata `if` does the same job:

```
keep if x>5
reg y x if x>5
```

2.3.3. Digression: Fixed-Effects, Random-Effects, Mixed-Models

$$y_j = \beta X_j + u_j \qquad \text{with } u_j \sim N(0, \sigma^2 \mathbf{I}_n)$$

Now there are $i \in \{1 \ldots M\}$ groups of data, each with $j \in \{1 \ldots n_i\}$ observations. Groups i represent one or several factors.

$$y_{ij} = \beta X_{ij} + \gamma_i Z_{ij} + u_{ij} \qquad \text{with } \gamma_i \sim N(0, \Psi), u_{ij} \sim N(0, \Sigma_i)$$

- Fixed effects: The researcher is interested in specific effects β. The researcher does not care/is ignorant about the type of distribution of β. If β is a big object, estimation can be expensive.
- Random effects: The researcher only cares about the distribution of γ. It is sufficient to estimate only the parameters of the distribution of γ (e.g. its standard deviation). This is less expensive/makes more efficient use of the data.
- Mixed model: includes a mixture of fixed and random factors

2.4. The Bayesian Approach

2.5. An alternative: The Bayesian Approach

$$\underbrace{P(\theta)}_{\text{prior}} \cdot \underbrace{P(X|\theta)}_{\text{likelihood}} \cdot \frac{1}{P(X)} = \underbrace{P(\theta|X)}_{\text{posterior}}$$

Remember the regression we did above:

```
lm (testscr ~ str,data=Caschool)


Call:
lm(formula = testscr ~ str, data = Caschool)

Coefficients:
(Intercept)          str
     698.93        -2.28
```

How can we do the same the Bayes' way?

Here we use a numerical approximation to calculate the Bayesian posterior distribution for the mean of `testscr`. We employ the Gibbs sampler `jags` (which is similar to `Bugs`).

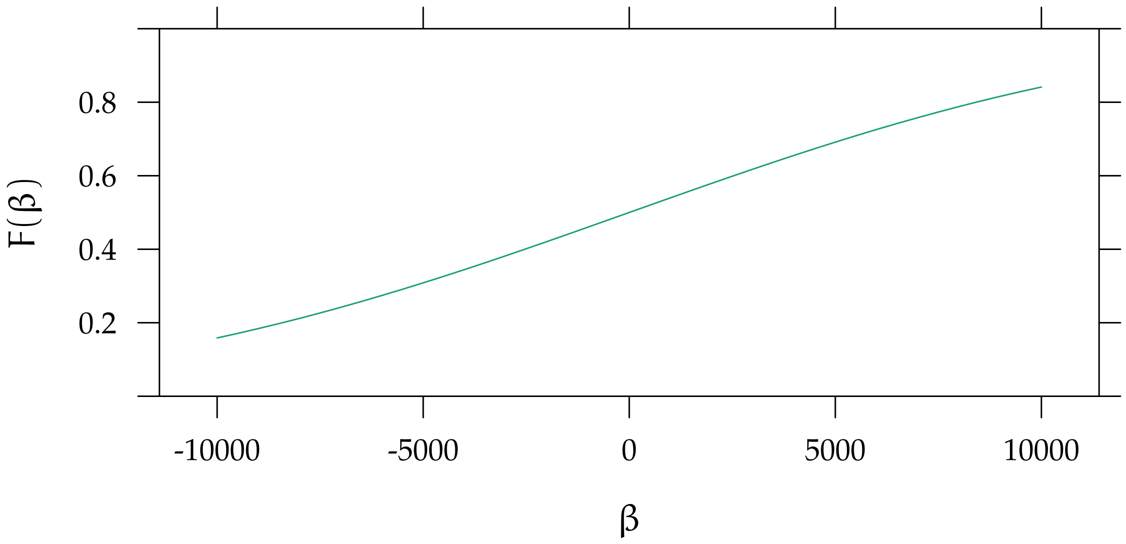
The first lines specify the stochastic process (`y[i] ~ dnorm(mu,tau)`), the next lines specify the priors. Here we use "uninformed priors", `mu ~ dnorm (0,.0001)` means that `mu` could take almost any value. The precision of the normal distribution (`0.0001`) is very small.

```
library(runjags)
modelX <- 'model {
 for (i in 1:length(y)) {
        y[i] ~ dnorm(beta0+beta1*x[i],tau)
    }
    beta0   ~ dnorm (0,.0001)
    beta1   ~ dnorm (0,.0001)
    tau  ~ dgamma(.01,.01)
    sd  <- sqrt(1/tau)
  }
}'
bayesX<-run.jags(model=modelX,data=list(y=Caschool$testscr,x=Caschool$str),
                 monitor=c("beta0","beta1","sd"))

Calling the simulation using the parallel method...
Following the progress of chain 1 (the program will wait for all chains to finish before c
Welcome to JAGS 3.4.0 on Sat Feb 28 08:57:10 2015
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
. . Reading data file data.txt
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 1673
. Reading parameter file inits1.txt
. Initializing model
. Adapting 1000
ERROR: Model is not in adaptive mode
. Updating 4000
-------------------------------------------------| 4000
************************************************** 100%
. . . . Updating 10000
-------------------------------------------------| 10000
************************************************** 100%
. . . Updating 0
. Deleting model
.
All chains have finished
Simulation complete.  Reading coda files...
Coda files loaded successfully
Calculating the Gelman-Rubin statistic for 3 variables....
```

```
The Gelman-Rubin statistic is below 1.05 for all parameters
Finished running the simulation
```

Digression: an uninformed prior for β, `beta ~ dnorm(0,.0001)`

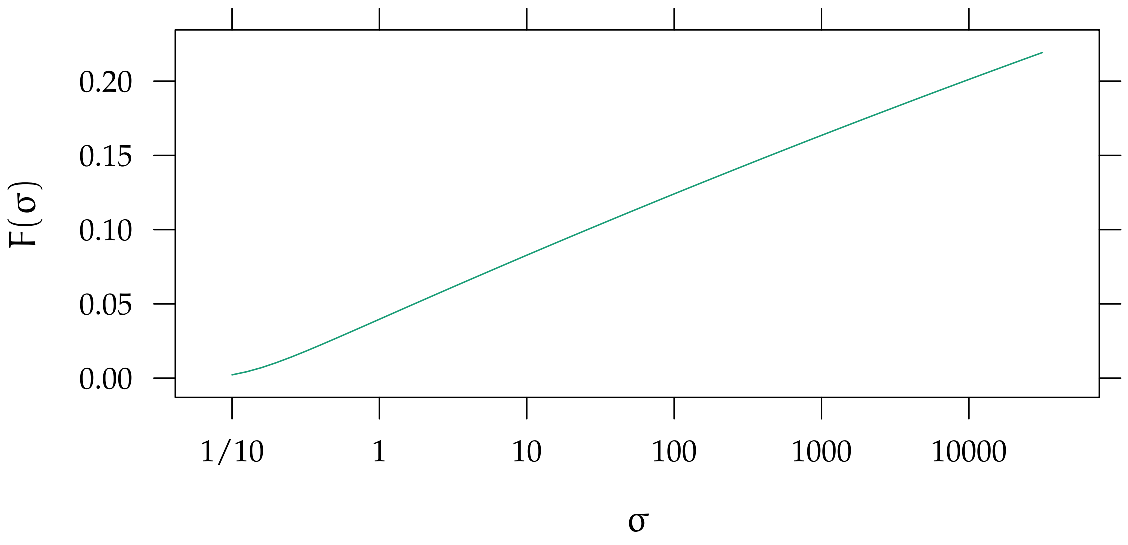
```
precision<-.0001
x<-seq(-10000,10000,500)
xyplot(pnorm(x,0,1/precision) ~ x,type="l",xlab="$\\beta$",ylab="$F(\\beta)$",ylim=c(0,1))
```



The prior distribution for μ (i.e. `dnorm(0,.0001)`) assigns (more or less) the same a-priori probability to any reasonable value of μ.

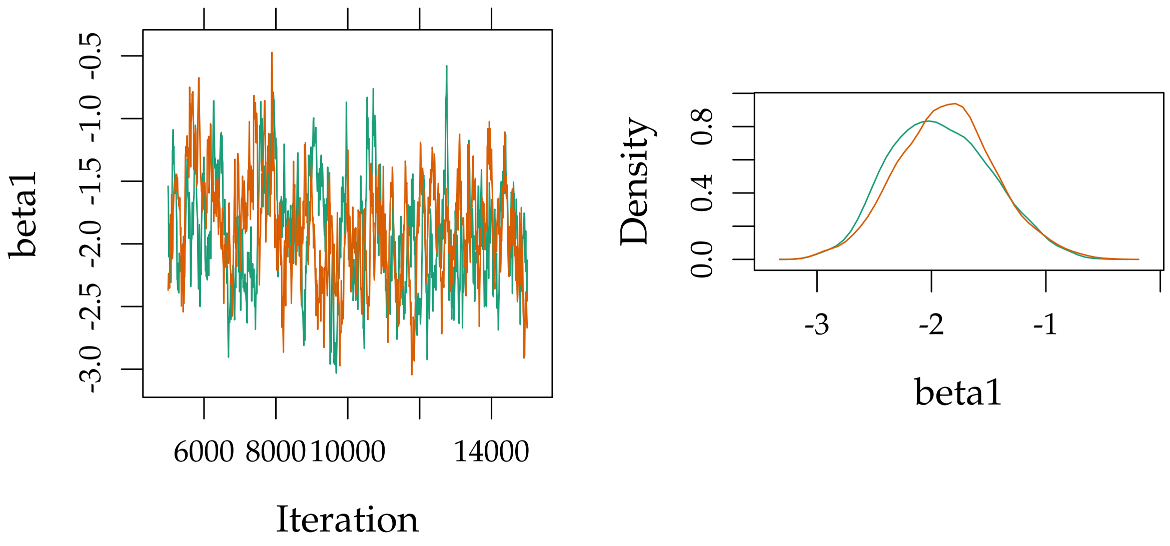
Digression: an uninformed prior for $\sigma = 1/\sqrt{\tau}$, `tau~dgamma(.01,.01)`:

```
s<-10^seq(-1,4.5,.1)
x<-1-pgamma(1/s^2,.01,.01)
xyplot(x ~ s, scales=list(x=list(log=T)), xscale.components = xscale.components.fractions,xlab
```

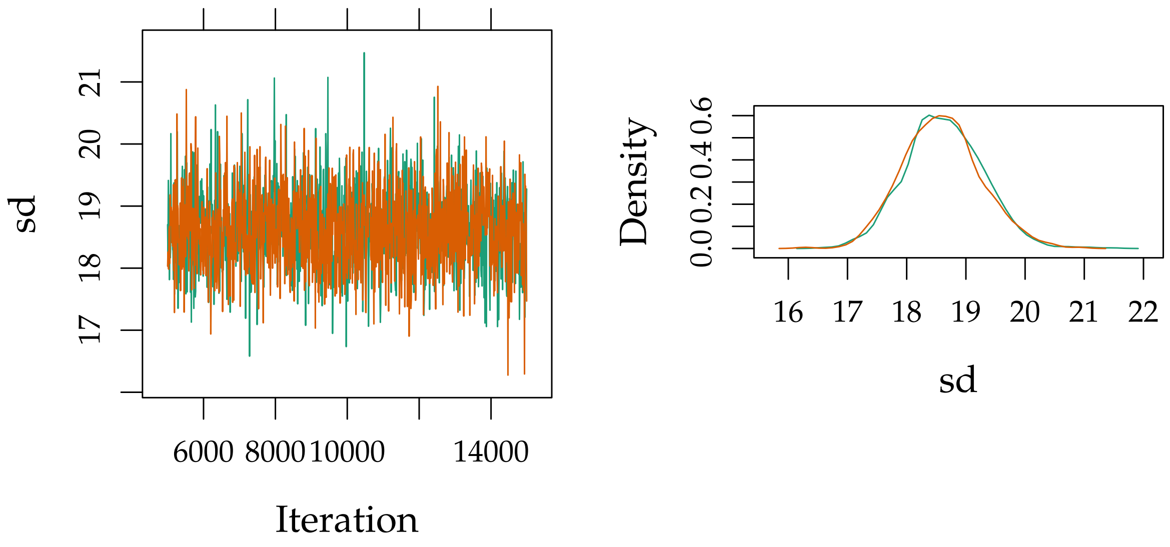


JAGS gives us now a posterior distribution for μ and for the standard deviation of `testscr`.

```
plot(bayesX,var="beta1",type=c("trace","density"))

Producing 2 plots for 1 variables to the active graphics device
(see ?runjagsclass for options to this S3 method)
```



```
plot(bayesX,var="sd",type=c("trace","density"))

Producing 2 plots for 1 variables to the active graphics device
(see ?runjagsclass for options to this S3 method)
```



```
summary(bayesX)


Iterations = 5001:15000
Thinning interval = 1
Number of chains = 2
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

         Mean     SD Naive SE Time-series SE
beta0 691.257 8.4566 0.059797       0.780009
beta1  -1.893 0.4291 0.003034       0.039816
sd     18.626 0.6516 0.004607       0.004644

2. Quantiles for each variable:

        2.5%     25%    50%     75%   97.5%
beta0 674.15 685.620 691.43 697.338 706.870
beta1  -2.68  -2.201  -1.90  -1.605  -1.021
sd     17.40  18.178  18.61  19.053  19.951
```

Comparison with the frequentist approach: The "credible interval" which can be obtained from the last line of the summary is very similar to the confidence interval from the frequentist approach.

Credible interval:

```
           2.5%      97.5%
beta0 674.14990 706.870050
```

```
beta1  -2.67978  -1.021076
sd     17.40420  19.950805
```

Confidence interval:

```
                2.5 %     97.5 %
(Intercept) 680.32313 717.542779
str          -3.22298  -1.336637
```

Also the estimated mean and its standard deviation are very similar to mean and standard error of the mean from the frequentist approach.

Priors: uninformed / mildly informed / informed Are "uninformed" priors reasonable?

Example:
You measure the eye colour of your fellow students. You sample 5 students and they all have blue eyes.

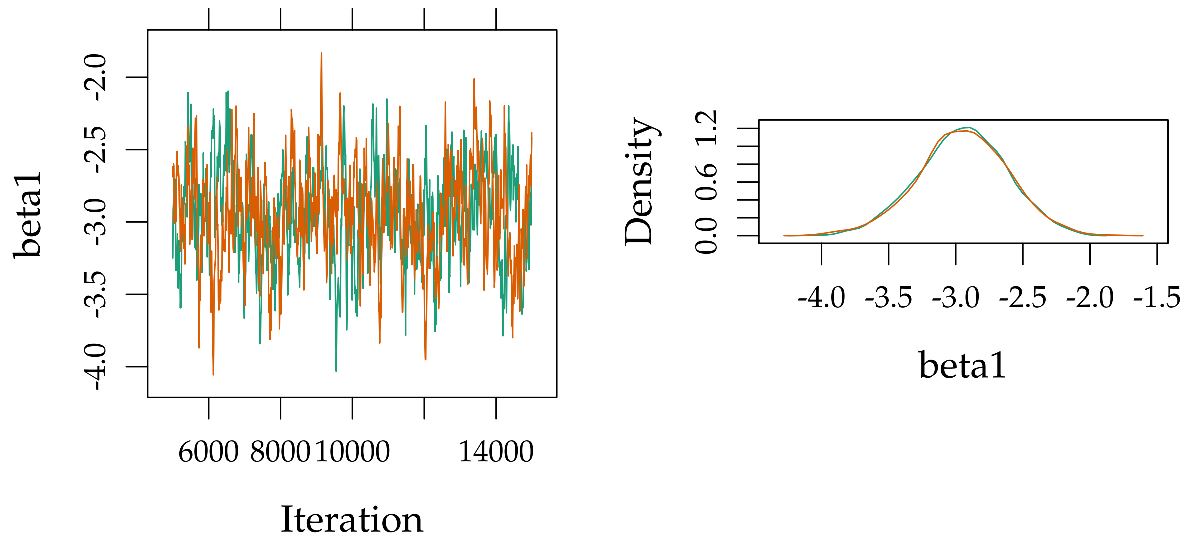
$\rightarrow$ 100% of your sample has blue eyes. You have no variance. How many of the remaining students will have blue eyes? Can you give a confidence interval?

Informed priors Above we used (similar to the frequentist approach) an "uninformed prior". Here we will assume that we already know something. Actually, we will pretend that we already did a similar study. That study gave us results of similar precision but with a different mean. Here we pretend that our prior distribution for μ is `dnorm(664,1)`. Everything else remains the same.

```
library(runjags)
modelI <- 'model {
 for (i in 1:length(y)) {
        y[i] ~ dnorm(beta0+beta1*x[i],tau)
    }
    beta0   ~ dnorm (0,.0001)
    beta1   ~ dnorm (-4,4) #<-- here is the information!
    tau  ~ dgamma(.01,.01)
    sd  <- sqrt(1/tau)
  }
}'
bayesI<-run.jags(model=modelI,data=list(y=Caschool$testscr,x=Caschool$str),
                 monitor=c("beta0","beta1","sd"))

Calling the simulation using the parallel method...
Following the progress of chain 1 (the program will wait for all chains to finish before co
Welcome to JAGS 3.4.0 on Sat Feb 28 08:57:17 2015
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
```

```
. . Reading data file data.txt
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 1675
. Reading parameter file inits1.txt
. Initializing model
. Adapting 1000
ERROR: Model is not in adaptive mode
. Updating 4000
-------------------------------------------------| 4000
************************************************** 100%
. . . . Updating 10000
-------------------------------------------------| 10000
************************************************** 100%
. . . Updating 0
. Deleting model
.
All chains have finished
Simulation complete.  Reading coda files...
Coda files loaded successfully
Calculating the Gelman-Rubin statistic for 3 variables....
The Gelman-Rubin statistic is below 1.05 for all parameters
Finished running the simulation
```

```
plot(bayesI,var="beta1",type=c("trace","density"))
```



```
summary(bayesX)$quantiles[,c("2.5%","97.5%")]

           2.5%      97.5%
beta0 674.14990 706.870050
```

```
beta1  -2.67978  -1.021076
sd     17.40420  19.950805

summary(bayesI)$quantiles[,c("2.5%","97.5%")]

            2.5%      97.5%
beta0 699.280950 725.140350
beta1  -3.613074  -2.309257
sd     17.411895  19.954908
```

We see that the informed prior has shifted the posterior away from the previous results. The new results are now somewhere between the ones we got with an uninformed prior and the new prior.

Comparison: Frequentist versus Bayesian approach

Frequentist: Null Hypothesis Significance Testing (Ronald A. Fisher, Statistical Methods for Research Workers, 1925, p. 43)

- $X \leftarrow \theta$, X is random, θ is fixed.
- Confidence intervals and p-values are easy to calculate.
- Interpretation of confidence intervals and p-values is awkward.
- p-values depend on the intention of the researcher.
- We can test "Null-hypotheses" (but where do these Null-hypotheses come from).
- Not good at accumulating knowledge.
- More restrictive modelling.

Bayesian: (Thomas Bayes, 1702-1761; Metropolis et al., "Equations of State Calculations by Fast Computing Machines". Journal of Chemical Physics, 1953.)

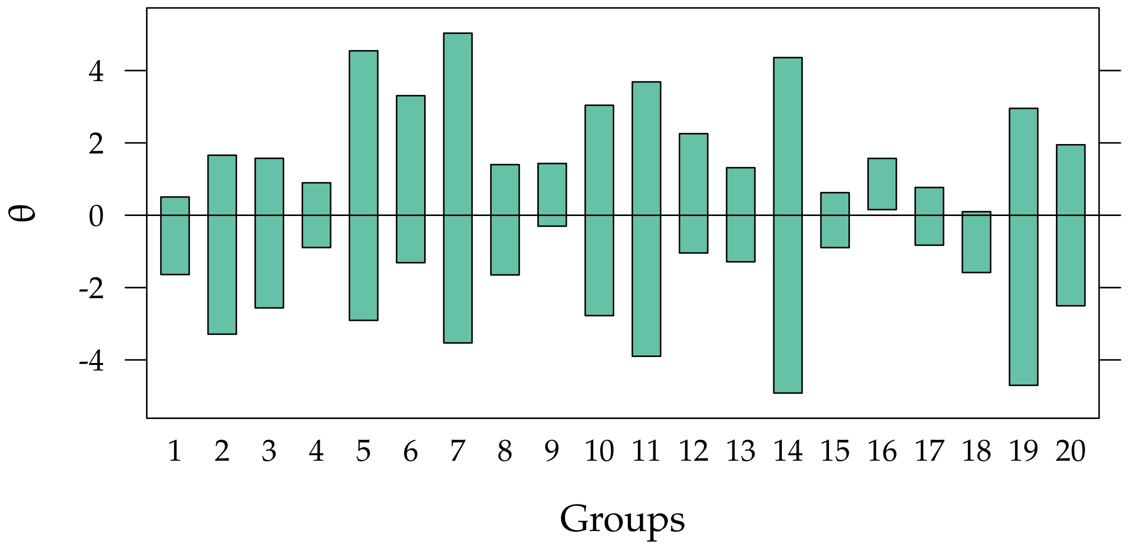
- $X \rightarrow \theta$, X is fixed, θ is random.
- Requires more computational effort.
- "Credible intervals" are easier to interpret.
- Can work with "uninformed priors" (similar results as with frequentist statistics)
- Efficient at accumulating knowledge.

- Flexible modelling.

Most people are still used to the frequentist approach. Although the Bayesian approach might have clear advantages it is important that we are able to understand research that is done in the context of the frequentist approach.

How the intention of the researcher affects p-values Example: Multiple testing (this is not the only example).

Assume a researcher obtains the following confidence intervals for different groups ($H_0 : \theta = 0$)



A researcher who a priori only suspects group 16 to have $\theta \neq 0$ will find (correctly) a significant effect.

A researcher who does not have this a priori hypothesis, but who justs carries out 20 independent tests, must correct for multiple testing and will find no significant effect. After all, it is not surprising to find in 5% of all samples a 95% confidence interval which does not include the Null-hypothetical value.

3. Binary choice

3.1. Example

3.1.1. A binary dependent variable

A teacher has 100 students and devotes a random effort $x \in [0, 1]$ to each of them.

Student qualification `qual` depends on the effort and a random component e which is $N(0, \sigma)$.

Students pass the final exam ($Y = 1$) when their qualification is larger than a critical value `crit`.

$$Y = 1 \Leftrightarrow x + e > \texttt{crit}$$

True relationship:

$$\begin{aligned} \Pr(Y = 1|x) &= F(x, \beta) \\ \Pr(Y = 0|x) &= 1 - F(x, \beta) \end{aligned}$$

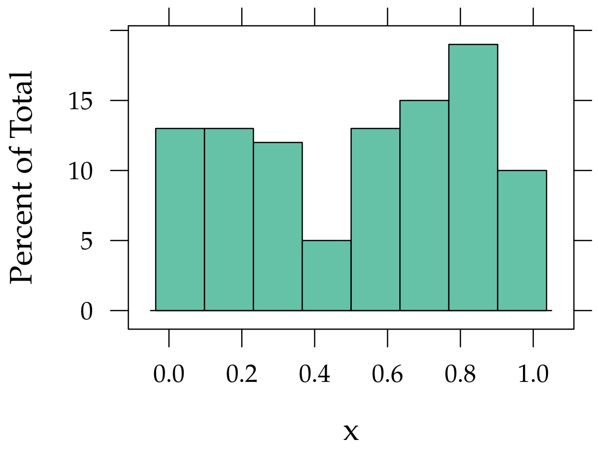
The teacher knows the structure of the process $F()$, but not its parameters β. The teacher is interested in knowing the critical effort that must be invested in the student such that, net of the random component, the student passes.

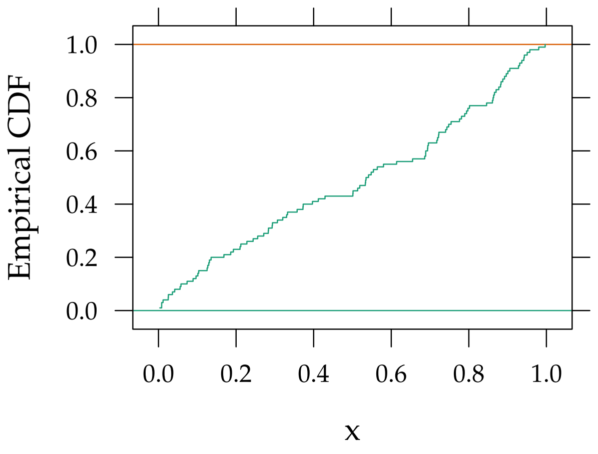
Let us first define a few variables, in particular our vector of random efforts x.

```
N <- 100
sigma <- .1
crit <- .3
set.seed(453)
x <- sort(runif(N))
```

Using a `sorted` vector `x` will help us in our plots later. Below we see a histogram of x. An alternative way to look at the distribution of x is the cumulative distribution function.

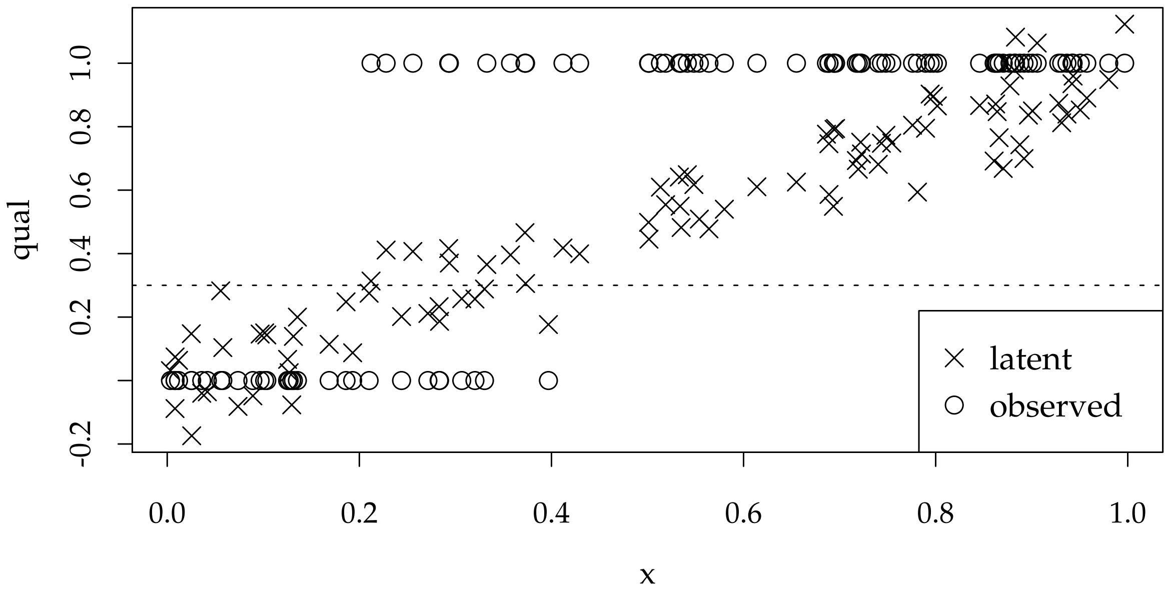
```
histogram(x)
ecdfplot(x)
```





Let us next add some random noise e to the effort x of the teacher and determine the (unobservable) qualification `qual` as well as the (observable) exam result y:

```
e <- sigma * rnorm(N)
qual <- x+e
plot (x,qual,pch=4)
abline(h=crit,lty="dotted")
y <- ifelse(qual>crit,1,0)
points(y ~ x, pch=1)
legend("bottomright",c("latent","observed"),pch=c(4,1))
```



3.1.2. Try to estimate the relationship with OLS

$$\begin{aligned}\Pr(Y=1|x) &= F(x,\beta)\\ \Pr(Y=0|x) &= 1-F(x,\beta)\end{aligned}$$

With OLS we might be tempted to use

$$F(x,\beta)=x'\beta$$

and then estimate

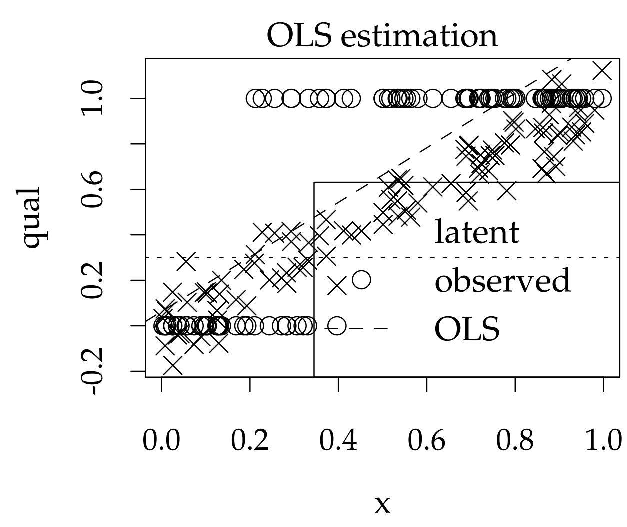
$$y=x'\beta+u$$

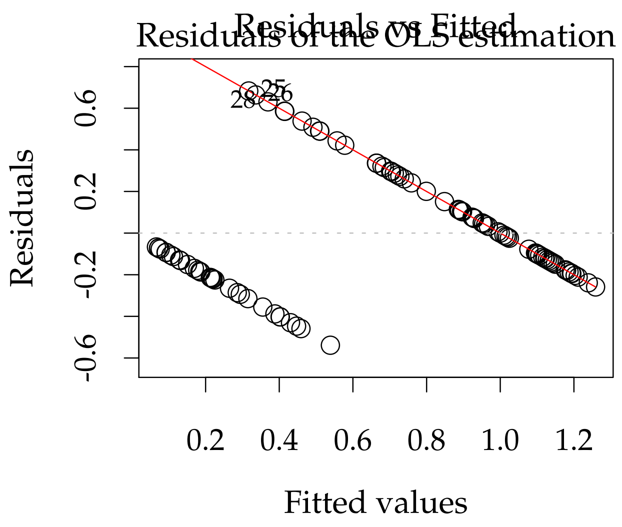
Why is this problematic?

- $E(u|X)\neq 0$ for most X
- predictions will not always look like probabilities

```
or <- lm ( y ~ x)
```

```
plot (qual ~ x,pch=4,main="OLS estimation")
abline(h=crit,lty="dotted")
points(y ~ x, pch=1)
abline(or,lty=2)
legend("bottomright",c("latent","observed","OLS"),pch=c(4,1,-1),lty=c(0,0,2))
plot(or,which=1,main="Residuals of the OLS estimation")
```





We see two problems with OLS.

- $E(u|X) \neq 0$ for most X
- $x'\beta$ does not stay in $[0, 1]$, predictions do not always look like probabilities

Let us start with the second problem:

3.1.3. A problem with OLS

How can we make sure that $x'\beta$ stays in $[0, 1]$?

$$\begin{aligned} \Pr(Y = 1|x) &= F(x, \beta) \\ \Pr(Y = 0|x) &= 1 - F(x, \beta) \end{aligned}$$

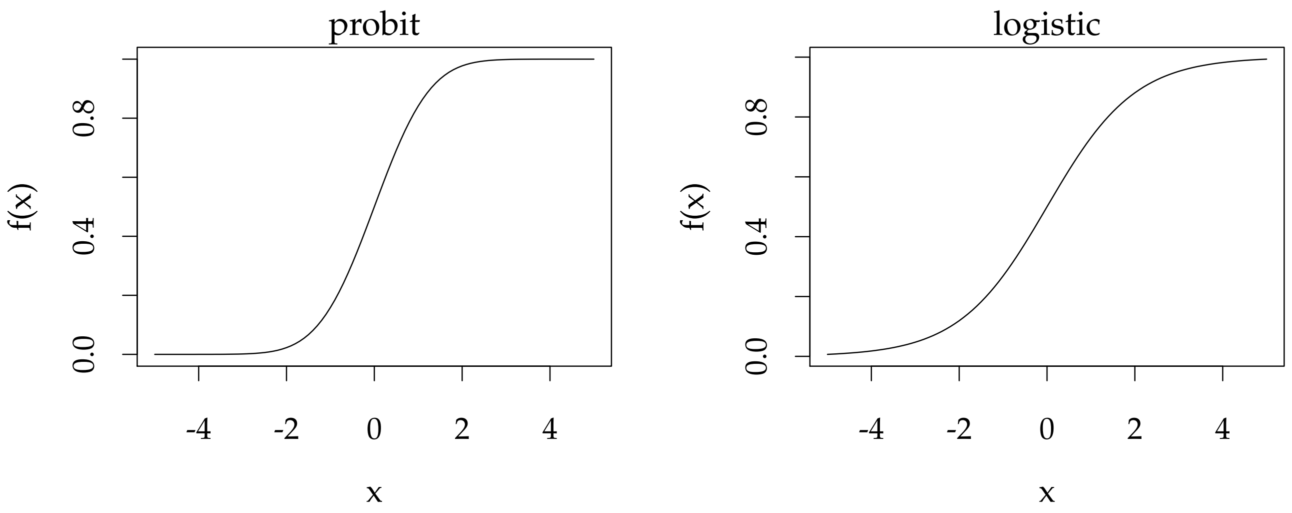
Trick, find a monotonic $\hat{F}$ such that $F(x, \beta) = \hat{F}(x'\beta)$ with

$$\begin{aligned} \lim_{x'\beta\to+\infty} \Pr(Y = 1|x) &= 1 \\ \lim_{x'\beta\to-\infty} \Pr(Y = 1|x) &= 0 \end{aligned}$$

Any continuous distribution function would do.

A common distribution function is, e.g. the standard normal distribution. In R we call this function `pnorm`. Another possible function is the logistic function $\frac{e^x}{1+e^x}$

```
plot(pnorm,-5,5,main="probit",ylab="f(x)")
plot(function(x) {exp(x)/(1+exp(x))},-5,5,main="logistic",ylab="f(x)")
```

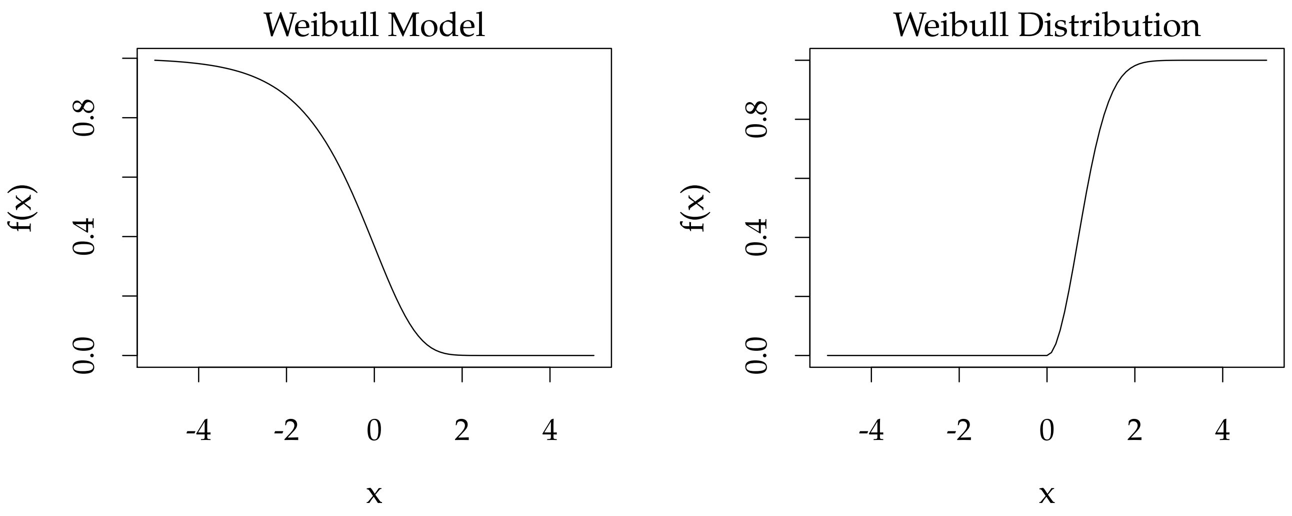


Since the logistic function has a couple of convenient properties, we will use it frequently and, hence, it also has a special name in R, we call it `plogis`. Instead of writing down the complicated formula, we could have said

```
plot(plogis,-5,5,main="logistic",ylab="f(x)")
```

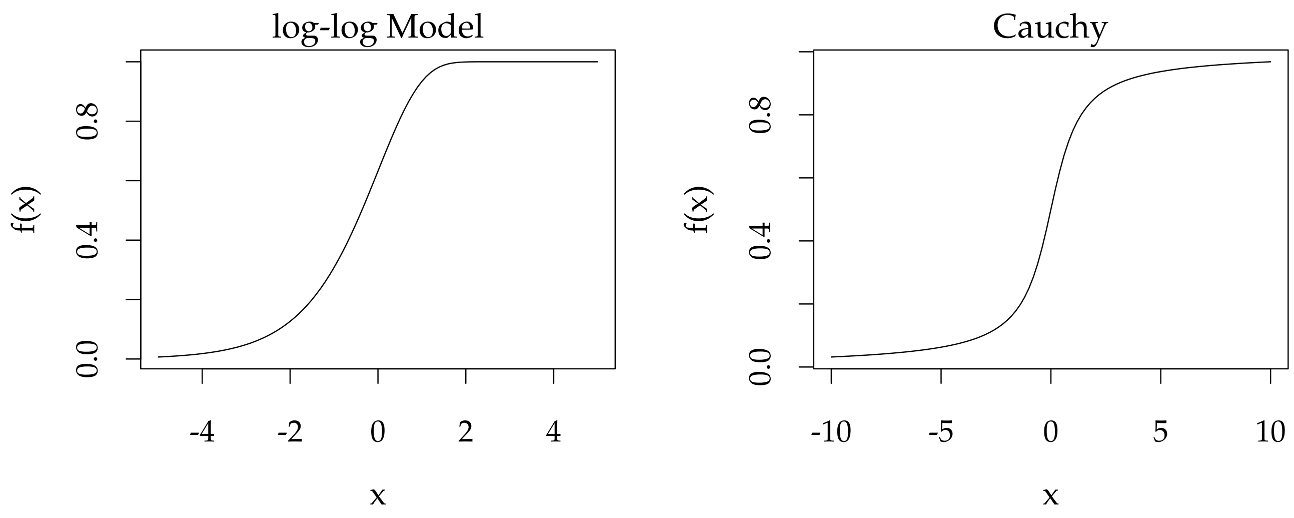
Less common functions are the Weibull model and the the Weibull distribution…

```
plot(function(x) {exp(-exp(x))},-5,5,main="Weibull Model",ylab="f(x)")
plot(function(x) {pweibull(x,shape=2)},-5,5, main="Weibull Distribution",ylab="f(x)")
```



…the log-log Model or the Cauchy distribution …

```
plot(function(x) {1-exp(-exp(x))},-5,5,main="log-log Model",ylab="f(x)")
plot(function(x) {(atan(x)+pi/2)/pi},-10,10,main="Cauchy",ylab="f(x)")
```



which is again so convenient that it has a special name in R, `cauchy`

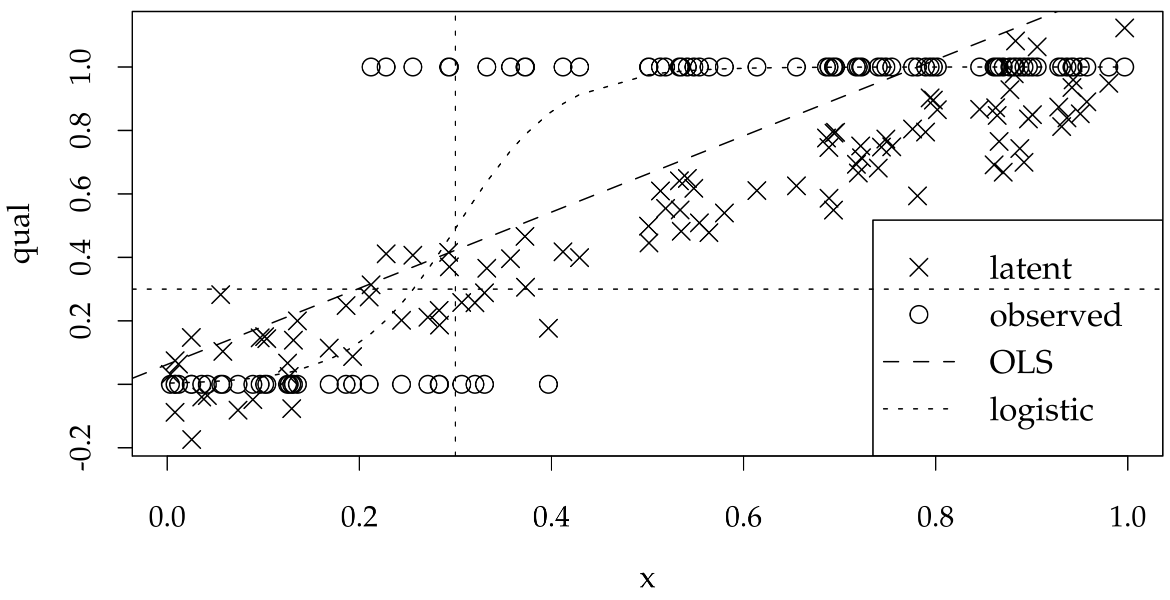
```
plot(pcauchy,-10,10,main="Cauchy",ylab="f(x)")
```

So let us apply, e.g. the logistic function to our problem. This is no longer done with `lm`, now we need a generalised linear model. These models are provided in R in a special library `survival`

```
(lr <- glm( y ~ x,family=binomial(link="logit")))


Call:  glm(formula = y ~ x, family = binomial(link = "logit"))

Coefficients:
(Intercept)            x
      -5.58        18.48

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      125
Residual Deviance: 30   AIC: 34
```

Let us compare, in a graph, the OLS model and the logistic model:

```
plot (qual ~ x,pch=4,)
abline(h=crit,v=crit,lty="dotted")
points(y ~ x, pch=1)
abline(or,lty=2)
lines(fitted(lr) ~ x, lty=3)
legend("bottomright",c("latent","observed","OLS","logistic"),
       pch=c(4,1,-1,-1),lty=c(0,0,2,3))
```



The `glm` function is rather flexible. We used to do OLS with `lm`, but we can also do this with `glm`. Let us first do OLS with `lm`

```
lm( y ~ x)


Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
     0.0626       1.2001
```

Now we do the same with `glm`

```
glm( y ~ x)


Call:  glm(formula = y ~ x)

Coefficients:
(Intercept)            x
     0.0626       1.2001

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:     21.8
Residual Deviance: 7.5  AIC: 30.7
```

The estimated coefficients are the same. But what happened to the long list of parameters we had previously in `glm` when we estimated the logistic model. Actually, `glm` always has *default* values for its parameters. We could as well have said

```
glm( y ~ x,family=gaussian(link="identity"))


Call:  glm(formula = y ~ x, family = gaussian(link = "identity"))

Coefficients:
(Intercept)            x
     0.0626       1.2001

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:     21.8
Residual Deviance: 7.5  AIC: 30.7
```

and obtain the same result. The *family* argument indicates the type of the distribution of the dependent variable.

3.2. Families and links

Remember that we could write the OLS model with a standard normal distributed error term

$$Y = \beta' X + \epsilon \text{ and } \epsilon \sim N(0, \sigma^2)$$

also as

$$Y \sim \underbrace{N}_{\text{family}} \left(\underbrace{\beta' X}_{\text{link}}, \sigma^2 \right)$$

Similarly, with the logistic model we said

$$\Pr(Y = 1|x) = F(X'\beta)$$

but we can also write

$$Y \sim \underbrace{\text{binom}}_{\text{family}} \left(\underbrace{F}_{\text{link}} (X'\beta) \right)$$

3.2.1. A list of families and link functions

family	link (F())
gaussian (N)	identity, log, inverse
binomial	logit, probit, cauchit, log, cloglog
gamma	identity, log, inverse
poisson	identity, log, sqrt
inverse.gaussian	identity, log, inverse, $1/\mu^2$
quasi	logit, probit, log, cloglog, identity, inverse, sqrt, $1/\mu^2$

3.2.2. The teacher's threshold

Let us come back to our teacher. She knows that

$$\Pr(Y = 1|x) = F(x, \beta)$$

She wants to know the x^* such that

$$\Pr(Y = 1|x^*) = F(x^*, \beta) = \frac{1}{2}$$

With $F(x, \beta) = F(x'\beta)$ and $F()$ the logistic function we have $F(0) = \frac{1}{2}$ or $x^{*\prime}\beta = 0$

With $F(x, \beta) = x'\beta$ (the OLS approach) we have $x^{*\prime}\beta = \frac{1}{2}$

In any case we need to know the estimated β. This can be extracted from the entire estimation result (that we stored in `lr` with the function `coef`.

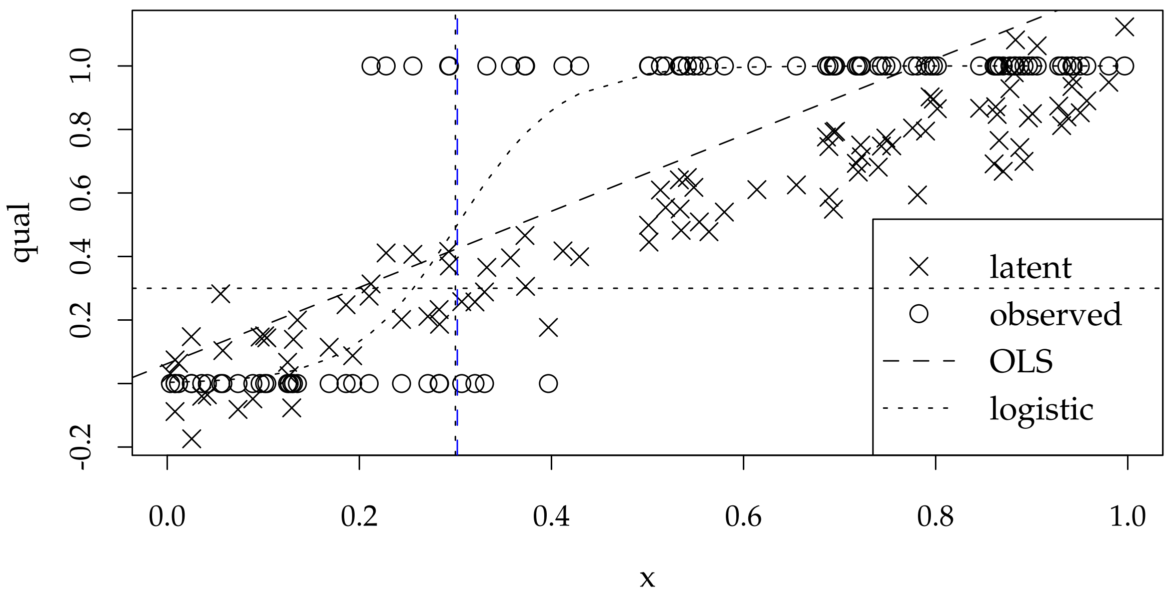
```
coef(lr)

(Intercept)            x
     -5.581       18.483
```

Hence, we can calculate the critical value as follows:

```
mycrit <- 0 - coef(lr)["(Intercept)"] / coef(lr)["x"]
mycrit

(Intercept)
      0.302
```

We could insert this value as a vertical line in our plot as follows:

```
abline(v=mycrit,lty="dashed",col="blue")
```



3.3. Marginal effects

3.3.1. Marginal effects with OLS

$$E[y|x] = E[x'\beta + \epsilon] = x'\beta$$

$$\frac{\partial E[y|x]}{\partial x} = \beta$$

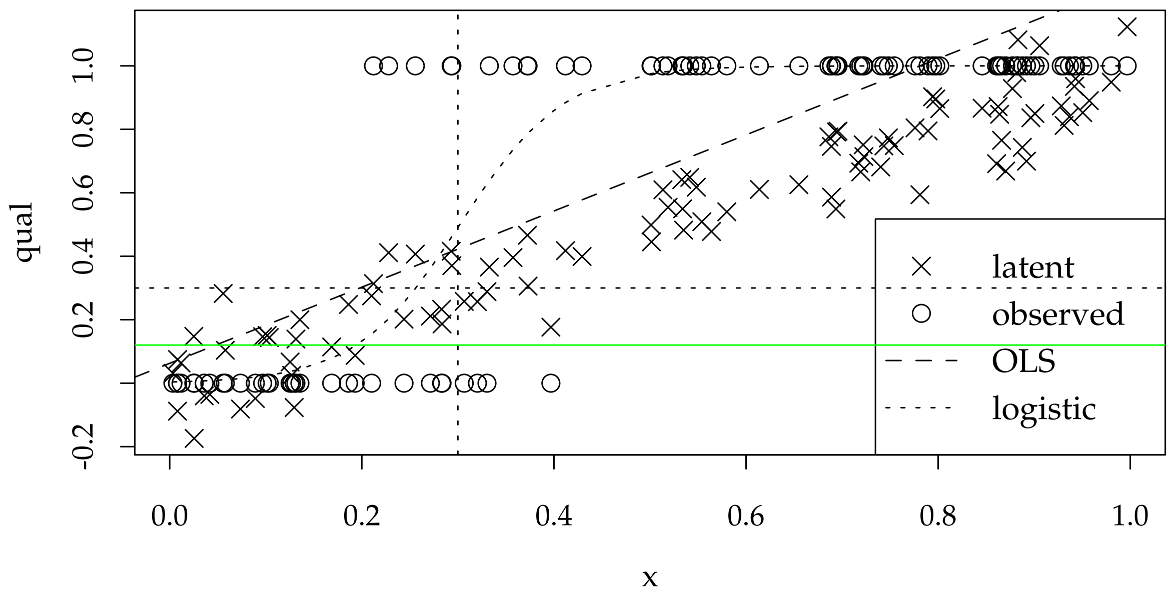
Here the marginal effect is simple to determine, it is just β.
The marginal effect does not depend on x.

```
coef(or)

(Intercept)           x
    0.06257     1.20008
```

If we want to show the marginal effect in our diagram, we can say (after rescaling the effect so that it fits into our diagram)

```
myScale=.1
```

```
abline(h=coef(or)["x"]*myScale,col="green")
```



3.3.2. Marginal effects with binary choice models

$$\Pr(Y = 1|x) = F(x'\beta)$$

or, in other words

$$E[y|x] = F(x'\beta)$$

hence

$$\frac{\partial E[y|x]}{\partial x} = f(x'\beta)\cdot\beta$$

since f is the density of the logistic distribution this expression can be simplified:

$$E(y|x) = F(x'\beta) = L(x'\beta) = \frac{e^{x'\beta}}{1+e^{x'\beta}}$$

generally the derivative $d\frac{a}{1+a}\Big/ da$ is $\frac{1}{(1+a)^2}$, thus, the marginal effect is

$$\begin{aligned}\frac{\partial E[y|x]}{\partial x} &= f(x'\beta)\cdot\beta = \frac{dL(x'\beta)}{dx}\cdot\beta = \frac{e^{x'\beta}}{(1+e^{x'\beta})^2}\cdot\beta = \\ &= L(x'\beta)(1-L(x'\beta))\cdot\beta\end{aligned}$$

(let us check:)

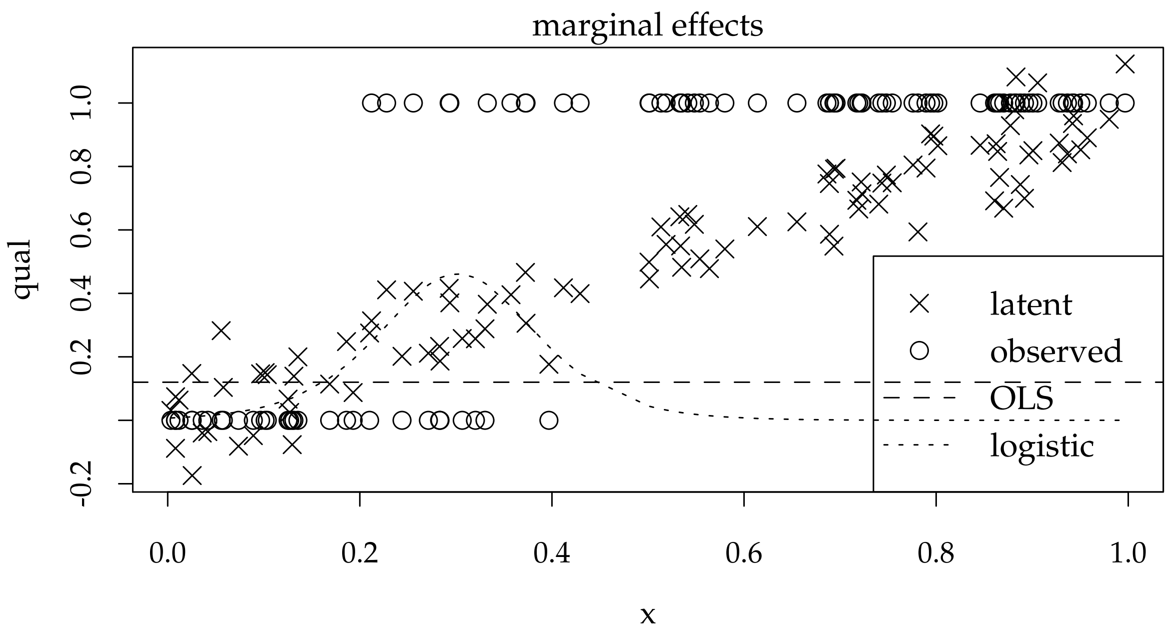
$$\begin{aligned}L(x'\beta)(1-L(x'\beta))\cdot\beta &= \frac{e^{x'\beta}}{(1+e^{x'\beta})}\cdot\left(1-\frac{e^{x'\beta}}{(1+e^{x'\beta})}\right)\cdot\beta = \\ &= \frac{e^{x'\beta}}{(1+e^{x'\beta})}\cdot\left(\frac{1+e^{x'\beta}}{(1+e^{x'\beta})}-\frac{e^{x'\beta}}{(1+e^{x'\beta})}\right)\cdot\beta \\ &= \frac{e^{x'\beta}}{(1+e^{x'\beta})}\cdot\left(\frac{1}{(1+e^{x'\beta})}\right)\cdot\beta \\ &= \frac{e^{x'\beta}}{(1+e^{x'\beta})^2}\cdot\beta\end{aligned}$$

We can calculate the marginal effects as follows

```
lmarginal <- plogis(predict(lr)) * (1-plogis(predict(lr))) * coef(lr)["x"]
```

and show them in our diagram:

```
plot (qual ~ x,pch=4,main="marginal effects")
points(y ~ x, pch=1)
abline(h=coef(or)["x"]*myScale,lty=2)
lines(lmarginal * myScale ~ x,lty=3)
legend("bottomright",c("latent","observed","OLS","logistic"),pch=c(4,1,-1,-1),lty=c(0,0,2,3))
```



3.3.3. The marginal effect is not constant

$$\frac{\partial E[y|x]}{\partial x} = L(x'\beta)(1 - L(x'\beta)) \cdot \beta$$

It is highest for the critical value and smaller for the extreme values of x. If our teacher is interested in increasing the number of students who pass the test, then she might concentrate her efforts on the students in the middle, since there the expected marginal return is highest.

Now assume, somebody is interested in a somehow aggregate measure of the marginal effect. Where do we evaluate the marginal effect?

- We can evaluate the marginal effect at the sample mean. In this case the sample mean is `mean(x)`, hence $x'\beta$ is

```
xbmean <- coef(lr) %*% c(1,mean(x))
xbmean
```

```
      [,1]
[1,] 3.928
```

hence, the marginal effect $L(x'\beta)(1 - L(x'\beta)) \cdot \beta$ is

```
plogis(xbmean)*(1-plogis(xbmean))*coef(lr)["x"]
```

```
       [,1]
[1,] 0.3499
```

- We can also evaluate the marginal effect for all observations and then use the average. Remember that `lmarginal` still contains all the marginal effects, so we can simply take the mean.

```
  mean(lmarginal)


[1] 0.8586
```

As we see, the result depends heavily on the method. The first method gave us the marginal effect on the *average subject*, the second the *average marginal effect*.

Things are slightly different if the X is a dummy variable. It makes no sense to look at derivatives. Instead, we use

$$\Pr(Y=1|\bar{x}_{-d}, d=1) - \Pr(Y=1|\bar{x}_{-d}, d=0)$$

3.3.4. The normal link function

$$\begin{aligned}\Pr(Y=1|x) &= F(x,\beta) = \Phi(x'\beta)\\ \Pr(Y=0|x) &= 1-F(x,\beta) = 1-\Phi(x'\beta)\end{aligned}$$

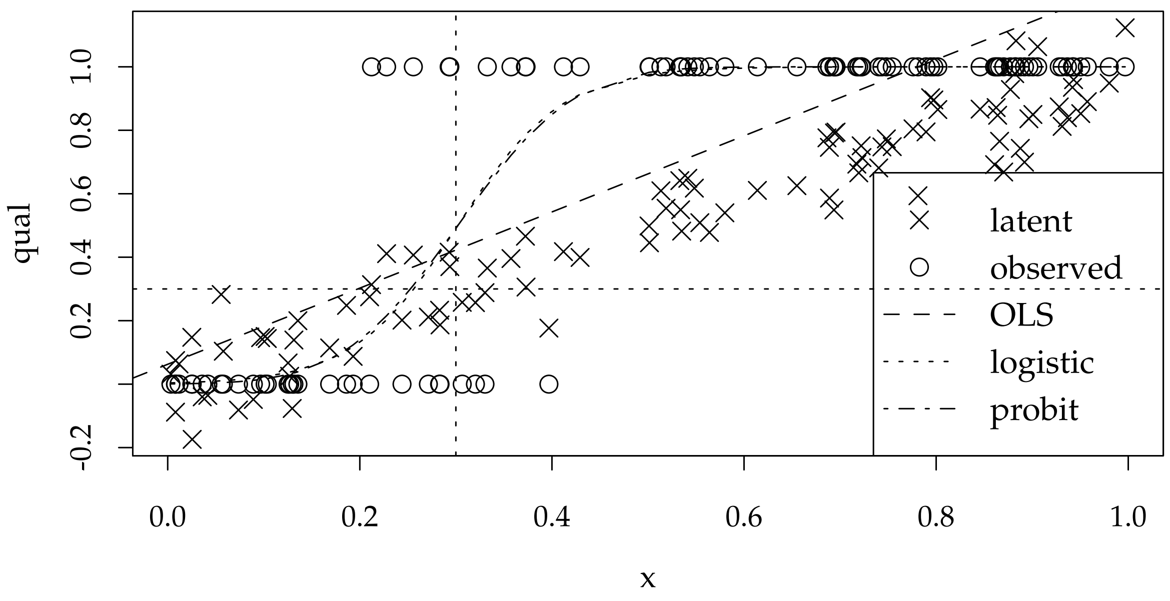
Or

$$Y \sim \underbrace{\text{binom}}_{\text{family}}\left(\underbrace{\Phi}_{\text{link}}(X'\beta)\right)$$

```
(pr <- glm( y ~ x,family=binomial(link="probit")))


Call:  glm(formula = y ~ x, family = binomial(link = "probit"))

Coefficients:
(Intercept)            x
       -3.2         10.6

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:      125
Residual Deviance: 29.4   AIC: 33.4
```

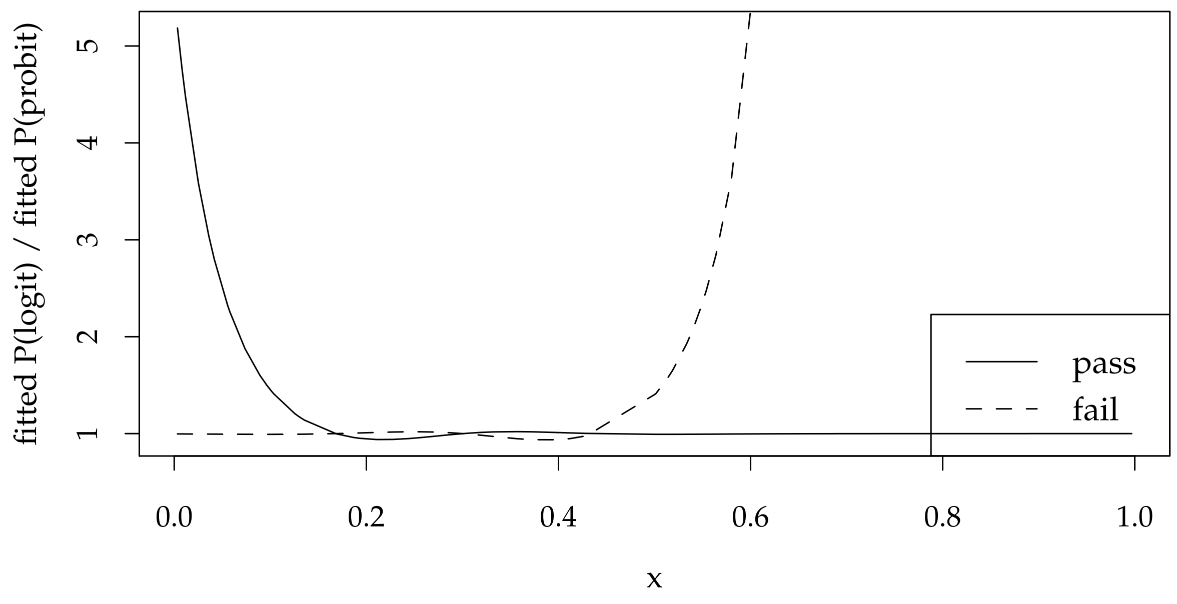
Here is the well known figure again, now with the estimation result for the normal link function added:

```
plot (qual ~ x,pch=4)
points(y ~ x, pch=1)
abline(h=crit,v=crit,lty="dotted")
abline(or,lty=2)
lines(fitted(lr) ~ x, lty=3)
lines(fitted(pr) ~ x, lty=4)
legend("bottomright",c("latent","observed","OLS","logistic","probit"),pch=c(4,1,-1,-1,-1),
```



We see that the logistic model and the probit model does not seem to be far apart. Note, however, that for very small or very large probabilities the distribution does matter. To see that we have a look at the ratio of the predicted values:

```
lp <- fitted(lr)/fitted(pr)
lp1 <- (1-fitted(lr))/(1-fitted(pr))
plot (lp ~ x,type="l",ylab="fitted P(logit) / fitted P(probit)")
lines (lp1 ~ x,lty=2)
legend("bottomright",c("pass","fail"),lty=c(1,2))
```



To predict probabilities for extreme values (probabilitites close to zero or close to one) the choice of the link function is essential. (Since, usually, we do not know much about the "correct" link function, we can not say too much about ratios of probabilities for extreme values).

3.3.5. Marginal effects with the normal link function

$$\begin{aligned}\Pr(Y=1|x) &= F(x,\beta) = \Phi(x'\beta)\\ \Pr(Y=0|x) &= 1-F(x,\beta) = 1-\Phi(x'\beta)\end{aligned}$$

$$E(y|x) = F(x'\beta) = \Phi(x'\beta)$$

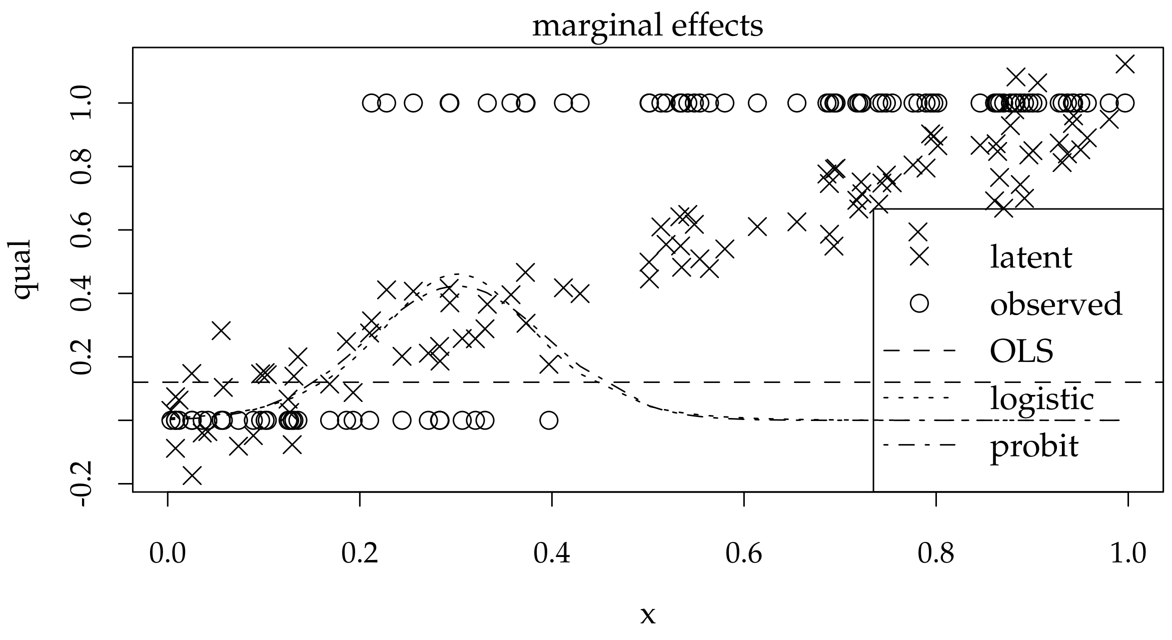
Marginal effects are:

$$\frac{\partial E[y|x]}{\partial x} = f(x'\beta)\cdot\beta = \phi(x'\beta)\cdot\beta$$

```
pmarginal=dnorm(pr$linear.predictors)*coef(pr)["x"]
```

Now let us draw the familiar figure again, now with the marginal effects of the probit model added:

```
plot (qual ~ x,pch=4,main="marginal effects")
points(y ~ x, pch=1)
abline(h=coef(or)["x"]*myScale,lty=2)
lines(lmarginal*myScale ~ x,lty=3)
lines(pmarginal*myScale ~ x,lty=4)
legend("bottomright",c("latent","observed","OLS","logistic","probit"),pch=c(4,1,-1,-1,-1),lty=
```



Again, we have different options to calculate aggregate marginal effects.

- at the sample mean? Remember, the sample mean is in `xbmean`, hence, the marginal effect there is

```
dnorm(xbmean)*coef(pr)["x"]
```

```
         [,1]
[1,] 0.001887
```

- We can also evaluate the marginal effect for all observations and then use the average. `pmarginal` still contains all the marginal effects, so we can simply take the mean.

```
mean(pmarginal)
```

```
[1] 0.8421
```

As with the logistic regression, also with the probit regression the result depends heavily on the method.

3.3.6. Marginal effects and interactions

linear model

$$\begin{aligned}
y &= \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \epsilon \\
E[y|x] &= \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 \\
\frac{\partial E[y|x]}{\partial x_1} &= \beta_1 + \beta_{12} x_2 \\
\frac{\partial^2 E[y|x]}{\partial x_1 \partial x_2} &= \beta_{12}
\end{aligned}$$

probit model

$$\begin{aligned}
E[y|x] &= \Phi(\underbrace{\beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2}_{u}) \\
\frac{\partial E[y|x]}{\partial x_1} &= (\beta_1 + \beta_{12} x_2) \cdot \phi(u) \\
\frac{\partial^2 E[y|x]}{\partial x_1 \partial x_2} &= \beta_{12} \cdot \phi(u) + (\beta_1 + \beta_{12} x_2) \cdot \frac{\partial}{\partial x_2} \phi(u) \\
&= \beta_{12} \cdot \phi(u) + (\beta_1 + \beta_{12} x_2) \cdot (\beta_2 + \beta_{12} x_1) \cdot \phi'(u)
\end{aligned}$$

Note: The marginal effect is *not* $\beta_{12}\phi(u)$

Note 2: Assume $\beta_{12} = 0$

$$\begin{aligned}
\frac{\partial^2 E[y|x]}{\partial x_1 \partial x_2} &= \beta_{12} \cdot \phi(u) + (\beta_1 + \beta_{12} x_2) \cdot (\beta_2 + \beta_{12} x_1) \cdot \phi'(u) \\
&= \beta_1 \beta_2 \phi'(u)
\end{aligned}$$

i.e. even if the estimated interaction on the level of the latent variable is zero, the marginal effect could be (significantly) different from zero.

- Obviously: Significance of the coefficient β_{12} has nothing to do with significance of the marginal effect.

- Should we study the model (the interactions) on the level of the latent variable, or on the level of y?

Interactions of three expressions

$$
\begin{aligned}
E[y|x] &= \Phi(\beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_{12} x_1 x_2 + \beta_{13} x_1 x_3 + \beta_{23} x_2 x_3) \\
\frac{\partial^3 E[y|x]}{\partial x_1 \partial x_2 \partial x_3} &= \phi''(u) \cdot (\beta_3 + x_1\beta_{13} + x_2\beta_{23} + x_1x_2\beta_{123}) \cdot \\
&\qquad\qquad (\beta_2 + x_1\beta_{12} + x_3\beta_{23} + x_1x_3\beta_{123}) \cdot \\
&\qquad\qquad (\beta_1 + x_2\beta_{12} + x_3\beta_{13} + x_2x_3\beta_{123}) + \\
& \phi'(u) \cdot (\beta_{12} + x_3\beta_{123}) \cdot (\beta_3 + x_1\beta_{13} + x_2\beta_{23} + x_1x_2\beta_{123}) + \\
& \phi'(u) \cdot (\beta_{13} + x_2\beta_{123}) \cdot (\beta_2 + x_1\beta_{12} + x_3\beta_{23} + x_1x_3\beta_{123}) + \\
& \phi'(u) \cdot (\beta_{23} + x_1\beta_{123}) \cdot (\beta_1 + x_2\beta_{12} + x_3\beta_{13} + x_2x_3\beta_{123}) + \\
& \phi(u) \cdot \beta_{123}
\end{aligned}
$$

3.3.7. Critical values

Let us determine critical values more systematically with a function:

```
invreg <- function (reg,y=0) {
  gc <- coef(reg)
  as.vector((y - gc[1])/gc[2])
}
```

This function has a first argument `reg` which is a regression object. The second argument, y, is optional. We need this, since sometimes (for the OLS model) the critical value is determined by $x'\beta = 1/2$ and sometimes (for the logit and the probit model) the critical value is determined by $x'\beta = 0$. We use `as.vector` to remove names from the result. These names would otherwise complicate the names lateron. Let us now apply this function to our three estimates:

```
invreg(lr)

[1] 0.302

invreg(pr)

[1] 0.3021

invreg(or,y=.5)

[1] 0.3645
```

3.4. Estimated values as random numbers

I am looking here at the critical value since this is a number that we can compare accross the different regression methods. The coefficient themselves are rather different.

We have to keep in mind that estimated coefficients as well as estimated critical values are *random* — they depend on the particular sample we draw from the population. To make that clear, let us do the following Monte Carlo exercise. We want to draw several times a random sample and compare the three methods.

Let us first write a function that does the above exercise once:

```
logreg <- function () {
  x <- sort(runif(N))
  e <- sigma * rnorm(N)
  qual <- x+e
  y <- ifelse(qual>crit,1,0)
  r <- lm ( y ~ x)
  rt<-invreg(r,.5)
  lr <- glm( y ~ x,family=binomial(link="logit"))
  lt<-invreg(lr)
  pr <- glm( y ~ x,family=binomial(link="probit"))
  pt<-invreg(pr)
  c(OLS=rt,probit=pt,logit=lt,meanY=mean(y))
}
```

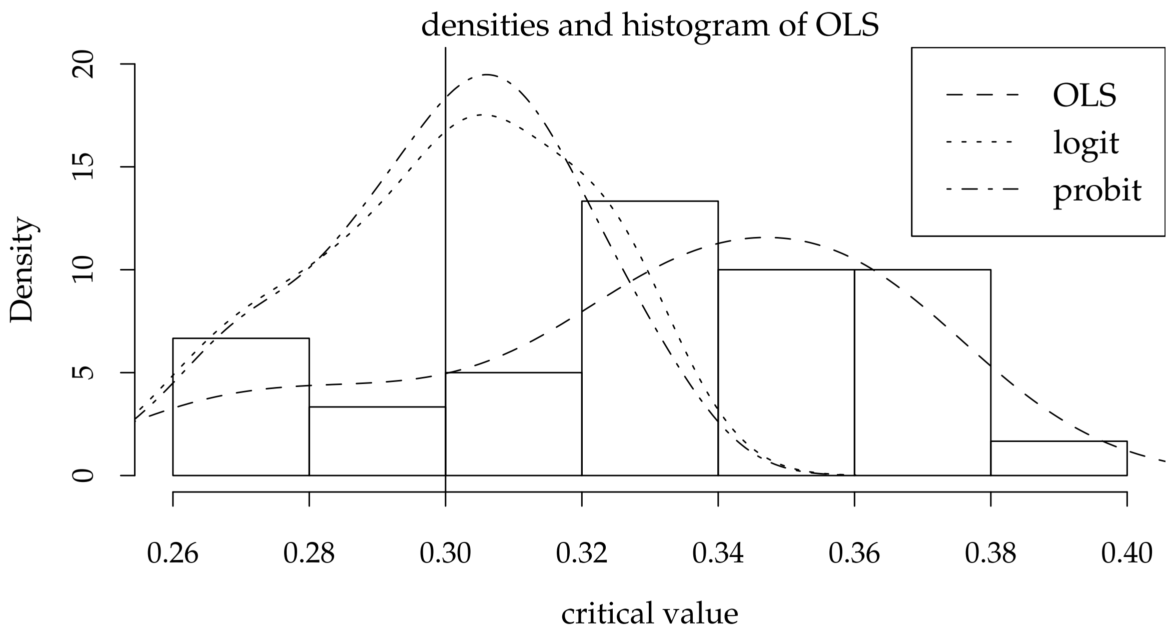
```
N <- 100
sigma <- .1
crit <- .3
logreg()

   OLS probit  logit  meanY
0.3023 0.2704 0.2650 0.7000
```

Now use the function a few times

```
coeffs <- t(sapply(1:30,function(x) {logreg()}))
```

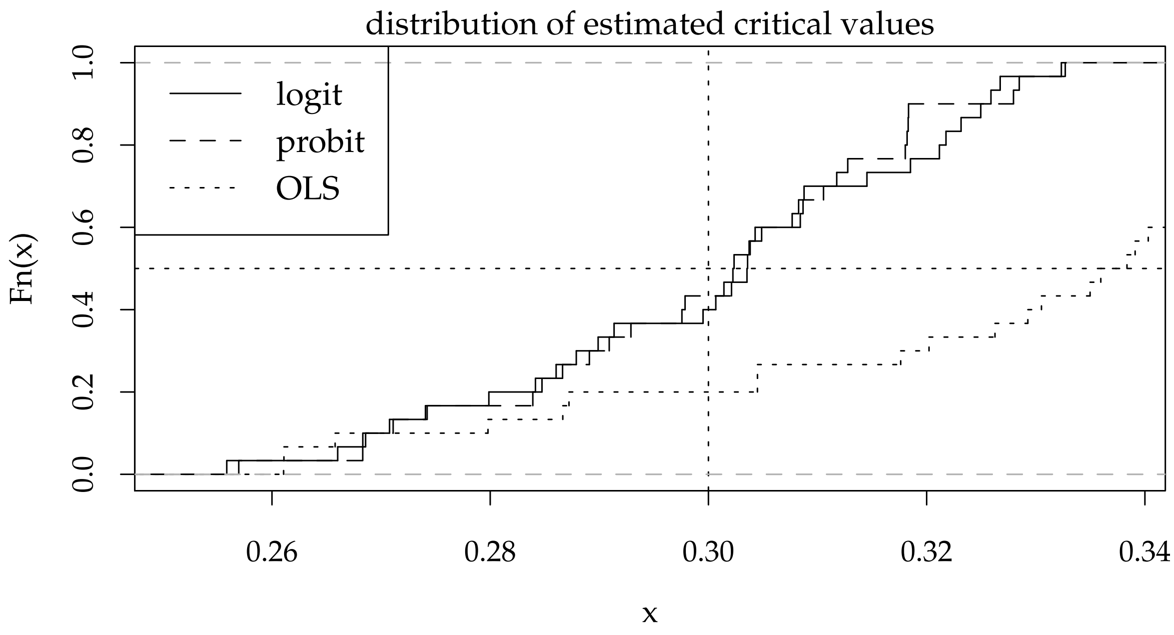
Below we show estimated densities for the three methods. To convince the reader that densities are related to the empirical distribution we also show the histogram, but only for the OLS case.

```
hist(coeffs[,"OLS"],freq=F,ylim=c(0,20),xlab="critical value",main="densities and histogram o
lines(density(coeffs[,"OLS"]),lty=2)
lines(density(coeffs[,"logit"]),lty=3)
lines(density(coeffs[,"probit"]),lty=4)
abline(v=.3)
legend("topright",c("OLS","logit","probit"),lty=c(2,3,4))
```



The next diagram shows the cumulative distribution:

```
plot(ecdf(coeffs[,"logit"]), do.points=F, verticals=T,  main="distribution of estimated cri
abline(h=.5,v=crit,lty="dotted")
lines(ecdf(coeffs[,"probit"]), do.points=F, verticals=T, lty=2)
lines(ecdf(coeffs[,"OLS"]), do.points=F, verticals=T,lty=3)
legend("topleft",c("logit","probit","OLS"),lty=1:3)
```

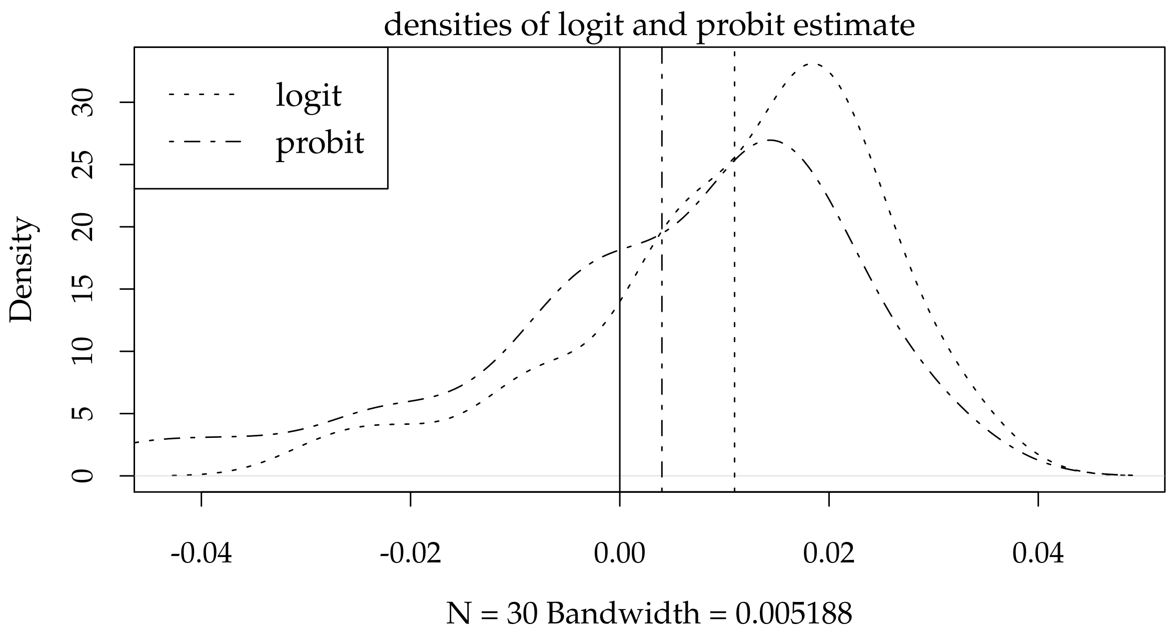


The simulation shows that distributions for the logistic and the probit model are centered around the theoretical value of 0.3 in the example. We also see that OLS seems to give a biased estimate. This should not be surprising. We know that the assumptions of the OLS model are not satisfied.

Above we mentioned that the choice of the link function is essential when we want to predict very small or very large probabilities. Similarly, the link function is essential for our estimation if what we observe contains mainly one type of an event. In the following example we observe mainly successes. We choose a larger sample size of $N = 1000$ to make sure that each sample contains at least some failures.

```
set.seed(123)
N <- 1000
sigma <- .1
crit <- 0
coeffs <- t(sapply(1:30,function(x) {logreg()}))
summary(coeffs[,"meanY"])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.948   0.955   0.958   0.958   0.961   0.973
```

```
plot(density(coeffs[,"logit"]),lty=3,main="densities of logit and probit estimate")
lines(density(coeffs[,"probit"]),lty=4)
abline(v=c(crit,mean(coeffs[,"logit"]),mean(coeffs[,"probit"])),lty=c(1,3,4))
legend("topleft",c("logit","probit"),lty=3:4)
```



We see that now our estimates depend on the link function. The probit estimate (which uses the correct link function for this specific model) gives on average an estimate close to the correct value, the logit estimate (which does not use the correct link function here) turns out to be too large on average.

The same exercise can be done with Stata:

```
drop _all
local N=100
local sigma=.1
local crit=.3
quietly set obs ‘N’
```

```
gen x=uniform()
gen e=‘sigma’ * invnormal(uniform())
gen qual = x + e
gen y = 0
replace y = 1 if qual > ‘crit’
quietly reg y x

capture program drop invreg
program define invreg, rclass
   syntax ,[y(real 0)]
   matrix coeff=e(b)
   return scalar crit=(‘y’-coeff[1,2])/coeff[1,1]
end
```

`invreg` is now a function that returns a scalar with the estimated critical value. Here is an example:

```
invreg,y(.5)
return list
```

As above with the R code also here we write a function that returns three estimates. Here we return a very small dataset with a single observation. This observation will then be appended to a larger file:

```
capture program drop logreg
program define logreg,rclass
   drop _all
   local N=100
   local sigma=.1
   local crit=.3
   quietly set obs ‘N’
   gen x=uniform()
   gen e=‘sigma’ * invnormal(uniform())
   gen qual = x + e
   gen y = 0
   replace y = 1 if qual > ‘crit’
   quietly reg y x
   invreg,y(.5)
   local rt = r(crit)
   quietly logit y x
   invreg
   local lt = r(crit)
   quietly probit y x
   invreg
   local pt = r(crit)
   drop _all
   set obs 1
   gen rt = ‘rt’
   gen lt = ‘lt’
   gen pt = ‘pt’
end
```

Now we can build our dataset:

```
logreg
save ex01,replace
forvalues i = 1/29 {
  logreg
  append using ex01
  save ex01,replace
}
```

and plot the results. First look at densities:

```
use ex01,clear
sum
cumul rt,gen(crt)
cumul lt,gen(clt)
cumul pt,gen(cpt)
scatter crt rt ||scatter clt lt ||scatter cpt pt
showgraph
```

next empirical cumulative distributions

```
list
twoway (histogram rt)
showgraph
twoway (histogram lt)
showgraph
twoway (histogram pt)
showgraph
*
cumul rt,gen(crt)
cumul lt,gen(clt)
cumul pt,gen(cpt)
scatter crt rt,c(J) ms(i) sort ||
      scatter clt lt,c(J) ms(i) sort ||
      scatter cpt pt,c(J) ms(i) sort yline(.5) xline(.3)
showgraph
```

3.5. Theoretical background

Does it make sense to use this procedure?

What we did so far:

$$\begin{aligned}\Pr(Y=1|x) &= F(x,\beta)\\ \Pr(Y=0|x) &= 1-F(x,\beta)\end{aligned}$$

$$F(x,\beta)=x'\beta$$

- Latent regression $y^l = x'\beta + u$
- Index function $y = \begin{cases}1 & \text{if } y^l > 0\\ 0 & \text{otherwise}\end{cases}$

Note: we can normalise the variance of u to any number — we only have to adjust β.

$y^l/\sigma = x'\beta/\sigma + u/\sigma$ is the same model.

Note that we have to take this into account when we compare two estimates with different variance for u (like logit coefficients are $\sqrt{\pi^2/3}$ times larger than their probit equivalent).

3.5.1. Latent variables

- Latent variable $y^l = x'\beta + u$
- Index function $y = \begin{cases} 1 & \text{if } y^l > 0 \\ 0 & \text{otherwise} \end{cases}$
- If u follows a logistic distribution with variance $\pi^2/3$

$$\boxed{\begin{array}{l} L(x) = e^x/(1+e^x), \\ l(x) = dL(x)/dx = e^x/(1+e^x)^2, \\ \text{thus } \int_{-\infty}^{+\infty} x^2 l(x)\,dx = \pi^2/3 \end{array}}$$

- or u follows a normal distribution with variance 1

then the probability that $y = 1$ is

$$\Pr(y^l > 0|x) = \Pr(x'\beta + u > 0|x) = \Pr(u < x'\beta|x) = F(x'\beta)$$

$$\begin{aligned} \Pr(Y=1|x) &= F(x,\beta) \\ \Pr(Y=0|x) &= 1 - F(x,\beta) \end{aligned}$$

$$F(x,\beta) = x'\beta$$

What the ML estimator actually does:

$$\Pr(\forall i : Y_i = y_i|X) = \prod_{y_i=1} F(x_i'\beta) \cdot \prod_{y_i=0} \left(1 - F(x_i'\beta)\right)$$

thus, the likelihood function is

$$L(\beta|\{y,X\}) = \prod_i \left(F(x_i'\beta)\right)^{y_i} \left(1 - F(x_i'\beta)\right)^{1-y_i}$$

take logs

$$\ln L = \sum_i \left(y_i \ln F(x_i'\beta) + (1-y_i)\ln\left(1-F(x_i'\beta)\right)\right)$$

take the first derivative

$$\frac{d\ln L}{d\beta} = \sum_i \left(\frac{y_i f(x_i'\beta)}{F(x_i'\beta)} + (1-y_i)\frac{-f(x_i'\beta)}{1-F(x_i'\beta)}\right) x_i = 0$$

in the logit case, e.g.

$$\begin{aligned}
\frac{d\ln L}{d\beta} &= \sum_i \left(\frac{y_i f(x_i'\beta)}{F(x_i'\beta)} + (1-y_i)\frac{-f(x_i'\beta)}{1-F(x_i'\beta)}\right) x_i \\
&= \sum_i \left(\frac{y_i}{1+e^{x_i'\beta}} - \frac{(1-y_i)e^{x_i'\beta}}{\left(1+e^{x_i'\beta}\right)^2\left(1-\frac{e^{x_i'\beta}}{1+e^{x_i'\beta}}\right)}\right) x_i \\
&= \sum_i \left(\frac{y_i}{1+e^{x_i'\beta}} - \frac{(1-y_i)e^{x_i'\beta}}{\left(1+e^{x_i'\beta}\right)^{2}\left(\frac{1+\cancel{e^{x_i'\beta}}-\cancel{e^{x_i'\beta}}}{\cancel{1+e^{x_i'\beta}}}\right)}\right) x_i
\end{aligned}$$

$$\begin{aligned}
&= \sum_i \left(\frac{y_i}{1+e^{x_i'\beta}} - \frac{(1-y_i)e^{x_i'\beta}}{1+e^{x_i'\beta}}\right) x_i \\
&= \sum_i \left(\frac{y_i - e^{x_i'\beta} + y_i e^{x_i'\beta}}{1+e^{x_i'\beta}}\right) x_i
\end{aligned}$$

$$\begin{aligned}
&= \sum_i \left(\frac{y_i(1+e^{x_i'\beta}) - e^{x_i'\beta}}{1+e^{x_i'\beta}}\right) x_i \\
&= \sum_i \left(y_i - \frac{e^{x_i'\beta}}{1+e^{x_i'\beta}}\right) x_i = \sum_i \left(y_i - F(x_i'\beta)\right) x_i = 0
\end{aligned}$$

For the logit and the probit model the Hessian is always negative definite. The numerical optimisation is well behaved.

3.5.2. Misspecification with OLS

If the true model is

$$y = X_1\beta_1 + X_2\beta_2 + u$$

and $X_2\beta_2$ is omitted then the expected estimate for β_1 is

$$E[\hat{\beta}_1] = \beta_1 + \left(X_1'X_1\right)^{-1} X_1'X_2\beta_2$$

which is β_1 if

- $\beta_2 = 0$
- or $X_1'X_2 = 0$, i.e. X_1 and X_2 are orthogonal.

With binary choice models only the first holds.

3.5.3. Goodness of fit

- log-likelihood function

$$\ln L = \ln L(\beta|\{y, X\}) = \ln \prod_i \left(F(x_i'\beta)\right)^{y_i} \left(1 - F(x_i'\beta)\right)^{1-y_i}$$

- log-likelihood with only a constant term

$$\ln L_0 = \ln L(\beta_0|\{y, X\}) = \ln \prod_i (\beta_0)^{y_i} (1 - \beta_0)^{1-y_i}$$

Mac Fadden's likelihood ratio index (similar to R^2, Pseudo R^2 in Stata):

$$LRI = 1 - \frac{\ln L}{\ln L_0}$$

More parameters (K) will give a better fit. We introduce a penalty for "overfitting".

3.5.4. Criteria for model selection

Akaike "An Information Criterion": $AIC = -2\ln L + 2K$
with OLS:

$$AIC = n \cdot \ln\left(\frac{\hat{e}'\hat{e}}{n}\right) + 2K$$

Bayesian information criterion (BIC): $BIC = -2\ln L + K \ln N$
with OLS:

$$BIC = n \cdot \ln\left(\frac{\hat{e}'\hat{e}}{n}\right) + K \ln N$$

We will prefer the model with the lower AIC or BIC.

3.6. Example II

Labour market participation Let us look at an example from labour market participation. The dataset `Participation` contains 872 observations from Switzerland. The variable `lfp` is a binary variable which is "yes" for a participating individual and "no" otherwise. The variable `lnnlinc` describes the nonlabour income. A hypothesis might be that the higher the nonlabour income, the less likely is it that the individual participates.

```
data(Participation)
(reg<-glm ( lfp ~ lnnlinc,family=binomial(link="logit"),data=Participation))


Call:  glm(formula = lfp ~ lnnlinc, family = binomial(link = "logit"),
    data = Participation)

Coefficients:
(Intercept)      lnnlinc
      9.628       -0.917

Degrees of Freedom: 871 Total (i.e. Null);  870 Residual
Null Deviance:      1200
Residual Deviance: 1180  AIC: 1180
```

Indeed, we see that the coefficient of `lnnlinc` is negative. We also see the AIC. To also calculate Mac Fadden's likelihood ratio index we regress on a constant:

```
(reg0<-glm ( lfp ~ 1,family=binomial(link="logit"), data=Participation))


Call:  glm(formula = lfp ~ 1, family = binomial(link = "logit"), data = Participation)

Coefficients:
(Intercept)
     -0.161

Degrees of Freedom: 871 Total (i.e. Null);  871 Residual
Null Deviance:      1200
Residual Deviance: 1200  AIC: 1210

1-logLik(reg)/logLik(reg0)

'log Lik.' 0.02265 (df=2)
```

3.6.1. Comparing models

The likelihood ratio:

$$2\log\frac{L_l}{L_s} \sim \chi^2_{l-s}$$

where L_s and L_l are likelihoods of a small model (with s parameters) and a large model (with l parameters), respectively.

```
pchisq (2 * (logLik(reg) - logLik(reg0))[1] , 1 , lower.tail=FALSE)

[1] 1.786e-07
```

Apparently, `lnnlinc` significantly improves the fit of the model. We can obtain the same output more conveniently with

```
anova(reg0,reg,test="Chisq")

Analysis of Deviance Table

Model 1: lfp ~ 1
Model 2: lfp ~ lnnlinc
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       871       1203
2       870       1176  1     27.2  1.8e-07 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

or even with

```
anova(reg,test="Chisq")

Analysis of Deviance Table

Model: binomial, link: logit

Response: lfp

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                      871       1203
lnnlinc  1     27.2       870       1176  1.8e-07 ***
---
Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
```

Now let us add more parameters to the model:

```
reg4<-glm(lfp ~ lnnlinc + age + educ + nyc + noc ,
          family=binomial(link="logit"),data=Participation)
summary(reg4)


Call:
glm(formula = lfp ~ lnnlinc + age + educ + nyc + noc, family = binomial(link = "logit"),
    data = Participation)
```

```
Deviance Residuals:
   Min       1Q   Median       3Q      Max
-1.940   -1.043   -0.607    1.115    2.465

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  12.4332     2.1440    5.80  6.7e-09 ***
lnnlinc      -0.8941     0.2041   -4.38  1.2e-05 ***
age          -0.5640     0.0889   -6.34  2.3e-10 ***
educ         -0.0458     0.0260   -1.77    0.077 .
nyc          -1.2210     0.1758   -6.95  3.8e-12 ***
noc          -0.0163     0.0723   -0.23    0.821
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1203.2  on 871  degrees of freedom
Residual deviance: 1098.9  on 866  degrees of freedom
AIC: 1111

Number of Fisher Scoring iterations: 4
```

We see that the parameter educ is not really significant. We can use the likelihood ratio to test whether educ contributes significantly to the model:

```
reg3 <- update ( reg4 , ~ . - educ)
anova(reg3,reg4,test="Chisq")

Analysis of Deviance Table

Model 1: lfp ~ lnnlinc + age + nyc + noc
Model 2: lfp ~ lnnlinc + age + educ + nyc + noc
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1       867       1102
2       866       1099  1     3.13    0.077 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In this exercise both p-values are similar. Generally, this need not be the case.

3.6.2. Confidence intervals

As with linear models we can calculate confidence intervals for our estimated coefficients (and as many people prefer confidence intervals over p-values, we might want to do this). The MASS library contains a more precise method to determine confidence intervals for generalised linear models.

```
library(MASS)
confint(reg4)

                 2.5 %    97.5 %
(Intercept)  8.33310 16.742834
lnnlinc     -1.30377 -0.503512
age         -0.74088 -0.392000
educ        -0.09698  0.004941
nyc         -1.57629 -0.886273
noc         -0.15861  0.125231
```

To be able to do the same exercise in Stata, we load the `foreign` library which provides an interface to data formats of other packages, including Stata's, and save the dataset in Stata's format.

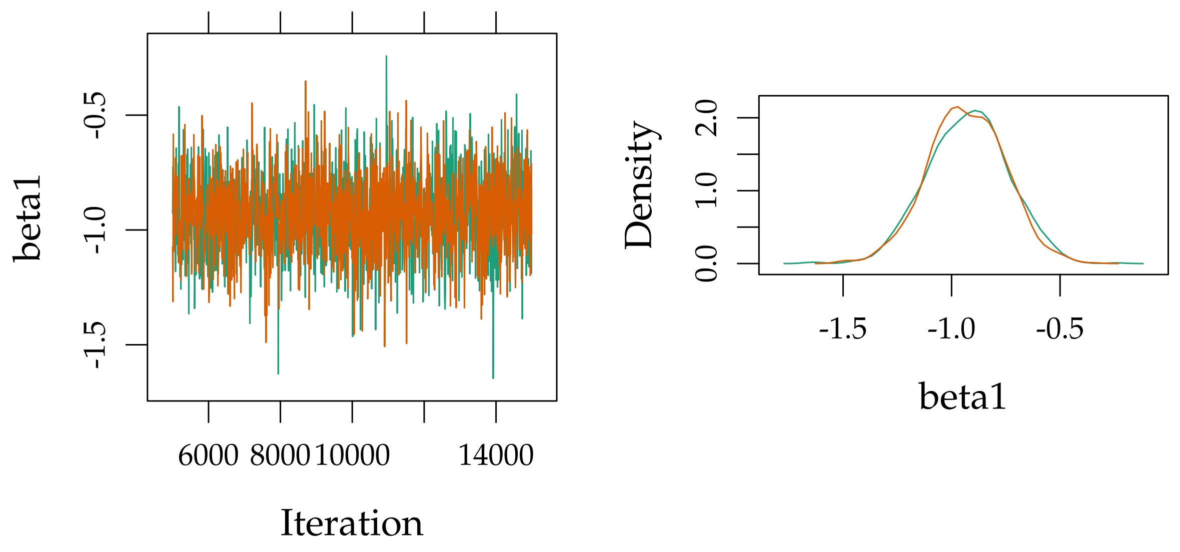
```
library(foreign)
write.dta(Participation,file="participation.dta")
```

```
clear
use participation
replace lfp=lfp-1
logit lfp lnnlinc
estat ic
```

3.6.3. Bayesian discrete choice:

```
library(runjags)
modelL <- 'model {
 for (i in 1:length(y)) {
        y[i] ~ dbern(p[i])
        logit(p[i]) <- beta0+beta1*x[i]
    }
    beta0   ~ dnorm (0,.0001)
    beta1   ~ dnorm (0,.0001)
  }
}'
bayesL<-run.jags(model=modelL,modules="glm",
       data=list(y=ifelse(Participation$lfp=="yes",1,0),
                  x=Participation$lnnlinc),
       monitor=c("beta0","beta1"))

Calling the simulation using the parallel method...
Following the progress of chain 1 (the program will wait for all chains to finish before c
Welcome to JAGS 3.4.0 on Sat Feb 21 18:42:29 2015
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
. Loading module: glm: ok
. . Reading data file data.txt
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 4358
. Reading parameter file inits1.txt
. Initializing model
. Adapting 1000
ERROR: Model is not in adaptive mode
. Updating 4000
-------------------------------------------------| 4000
************************************************** 100%
. . . Updating 10000
-------------------------------------------------| 10000
************************************************** 100%
. . . Updating 0
. Deleting model
.
All chains have finished
Simulation complete.  Reading coda files...
Coda files loaded successfully
Calculating the Gelman-Rubin statistic for 2 variables....
The Gelman-Rubin statistic is below 1.05 for all parameters
Finished running the simulation
```

```
plot(bayesL,var="beta1",type=c("trace","density"))
```



```
summary(bayesL)$quantiles[,c("2.5%","97.5%")]

        2.5%  97.5%
beta0  5.862 13.615
beta1 -1.289 -0.564
```

```
confint(reg)

              2.5 %  97.5 %
(Intercept)  5.839 13.5788
lnnlinc     -1.287 -0.5619
```

3.6.4. Odds ratios

$$\Pr(Y=1|x)=F(x'\beta)\quad\text{with}\quad F(x'\beta)=\frac{e^{x'\beta}}{1+e^{x'\beta}}$$

Let us define the odds $o=\frac{p}{1-p}$, then

$$\log(o)=\log\frac{p}{1-p}=\log\frac{\frac{e^{x'\beta}}{1+e^{x'\beta}}}{1-\frac{e^{x'\beta}}{1+e^{x'\beta}}}=\log\frac{\frac{e^{x'\beta}}{1+e^{x'\beta}}}{\frac{1+e^{x'\beta}-e^{x'\beta}}{1+e^{x'\beta}}}=$$

$$\log\frac{e^{x'\beta}}{1+e^{x'\beta}-e^{x'\beta}}=\log e^{x'\beta}=x'\beta=\beta_0+\beta_1x_1+\beta_2x_2+\ldots$$

or

$$o=e^{\beta_0}\cdot e^{\beta_1x_1}\cdot e^{\beta_2x_2}\ldots$$

i.e. an increase of x_i by one unit increases the odds o by a factor of e^{β_i}.

In particular if x_i is a binary variable ($x_i\in\{0,1\}$)

$$\begin{aligned}\frac{\frac{P(Y=1|x_i=1)}{1-P(Y=1|x_i=1)}}{\frac{P(Y=1|x_i=0)}{1-P(Y=1|x_i=0)}} &= \frac{o(Y=1|x_i=1)}{o(Y=1|x_i=0)}\equiv o_{x_i}\\ &= \frac{e^{\beta_0+\beta_1x_1+\beta_2x_2+\ldots+\beta_i+\ldots}}{e^{\beta_0+\beta_1x_1+\beta_2x_2+\ldots+0+\ldots}}\\ &= e^{\beta_i}\end{aligned}$$

Note: The odds-ratio is a ratio of a ratio of probabilities. This is not necessarily intuitive.

Note 2: Only if $P(Y=1|x_i=1)$ and $P(Y=1|x_i=0)$ are close to zero we have

$$\frac{\frac{P(Y=1|x_i=1)}{1-P(Y=1|x_i=1)}}{\frac{P(Y=1|x_i=0)}{1-P(Y=1|x_i=0)}}\approx\frac{P(Y=1|x_i=1)}{P(Y=1|x_i=0)}$$

i.e. the *odds-ratio* approximates the *risk-ratio*. The risk-ratio migh be more intuitive.
E.g. $P(Y=1|x_i=1)=0.4$ and $P(Y=1|x_i=0)=0.2$ then

$$\frac{\frac{P(Y=1|x_i=1)}{1-P(Y=1|x_i=1)}}{\frac{P(Y=1|x_i=0)}{1-P(Y=1|x_i=0)}}=\frac{\frac{0.4}{0.6}}{\frac{0.2}{0.8}}=\frac{1}{6}$$

```
options(scipen=5)
exp(coef(reg4))

(Intercept)      lnnlinc          age         educ          nyc          noc
251003.4538       0.4090       0.5690       0.9552       0.2949       0.9838
```

Confidence intervals can be calculated similarly:

```
exp(confint(reg4))

                 2.5 %         97.5 %
(Intercept) 4159.2921 18677567.2442
lnnlinc        0.2715        0.6044
age            0.4767        0.6757
educ           0.9076        1.0050
nyc            0.2067        0.4122
noc            0.8533        1.1334
```

Note: This interpretation is possible for the logistic link function only.

3.6.5. Odds ratios with interactions

Assume $P(Y=1|x)=F(\beta_1x_1+\beta_2x_2+\beta_{12}x_1x_2)$ with F the logistic function. Then

$$o_{x_1}=\frac{o(Y=1|x_1=1)}{o(Y=1|x_1=0)}=\frac{e^{\beta_1+\beta_2x_2+\beta_{12}x_2}}{e^{\beta_2x_2}}=e^{\beta_1+\beta_{12}x_2}$$

hence

$$\frac{o_{x_1|x_2=1}}{o_{x_1|x_2=0}}=\frac{e^{\beta_1+\beta_{12}}}{e^{\beta_1}}=e^{\beta_{12}}$$

i.e. $e^{\beta_{12}}$ is a ratio of two odds ratios (which, in turn, are ratios of ratios of probabilities).

4. Tobit

4.1. Motivation

Censored variables We call a variable "censored" when we can only observe the latent variable in a certain range.

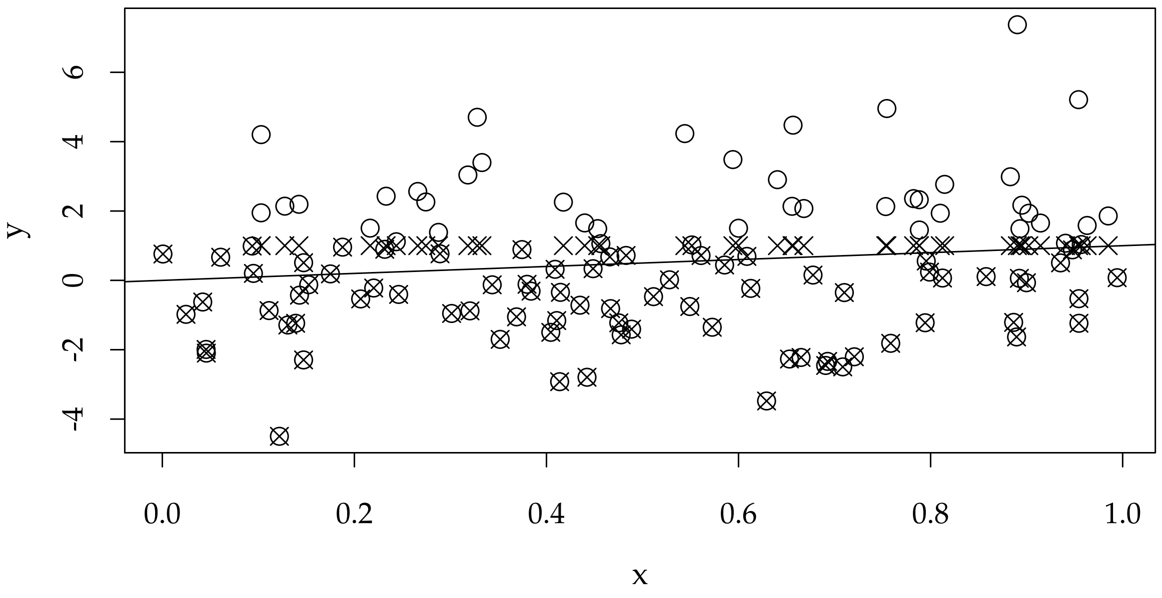
- hours worked (negative amounts can not be observed)
- income (only when relevant for social incurance)
- bids in an auction (often only the second highest bid is observed)
- effort in a tournament/war of attrition (only the second highest effort is observed)

4.2. Example

Let us generate a simple censored variable:

```
library(survival)
set.seed(123)
n <- 120
sd <- 2
x <- sort(runif(n))
y <- x + sd*rnorm(n)
ycens <- y
ycens[y>=1] <- 1
```

```
plot(y ~ x)
points(ycens ~ x,pch=4)
abline(a=0,b=1)
```



As with the logit model we can try to use simple OLS to estimate the relationship:

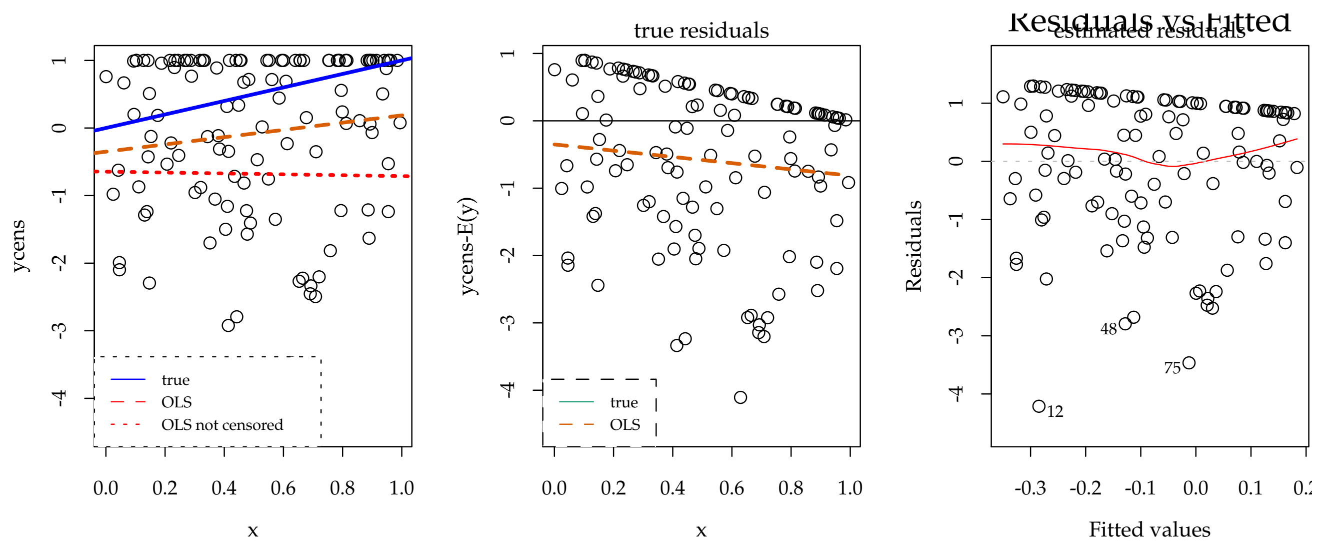
```
olsall<-lm(ycens ~ x)
summary(olsall)


Call:
lm(formula = ycens ~ x)

Residuals:
   Min     1Q Median     3Q    Max
-4.212 -0.705  0.448  0.987  1.295

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.350      0.231   -1.52     0.13
x              0.537      0.395    1.36     0.18

Residual standard error: 1.23 on 118 degrees of freedom
Multiple R-squared:  0.0154,Adjusted R-squared:  0.00707
F-statistic: 1.85 on 1 and 118 DF,  p-value: 0.177
```

This result is not too convincing, perhaps we should only look at the uncensored observations:

```
olsrel<-lm(ycens ~ x,subset=ycens<1)
summary(olsrel)


Call:
lm(formula = ycens ~ x, subset = ycens < 1)

Residuals:
   Min     1Q Median     3Q    Max
-3.842 -0.667  0.210  0.853  1.642

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.6461     0.2637   -2.45    0.017 *
x            -0.0662     0.4852   -0.14    0.892
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.18 on 75 degrees of freedom
Multiple R-squared:  0.000248,Adjusted R-squared:  -0.0131
F-statistic: 0.0186 on 1 and 75 DF,  p-value: 0.892
```

The left part of the following picture shows the true relation as well as the two estimates:

```
plot(ycens ~ x)
abline(a=0,b=1,lwd=3,col="blue")
abline(olsall,lty=2,col=2,lwd=3)
abline(olsrel,lty=3,col="red",lwd=3)
legend("bottomleft",c("true","OLS","OLS not censored"),lty=1:3,
       col=c("blue","red","red"),bg="white",cex=.7)
#
plot(ycens-x ~ x,main="true residuals",ylab="ycens-E(y)")
abline(h=0)
lines(predict(olsall) -x  ~ x,lty=2,col=2,lwd=3)
legend("bottomleft",c("true","OLS"),col=1:2,lty=1:2,bg="white",cex=.7)
#
plot(olsall,which=1,main="estimated residuals")
```



The graph in the middle shows the true residuals, the graph on the right shows the estimated residuals. While the estimated residuals underestimate the problem, it is still visible:

$E(u|X) \neq 0$ for most X.

4.3. Solution

Interval regression: As in the logistic case we use maximum likelihood. The procedure is called "interval regression" since the dependent variable is now an interval.

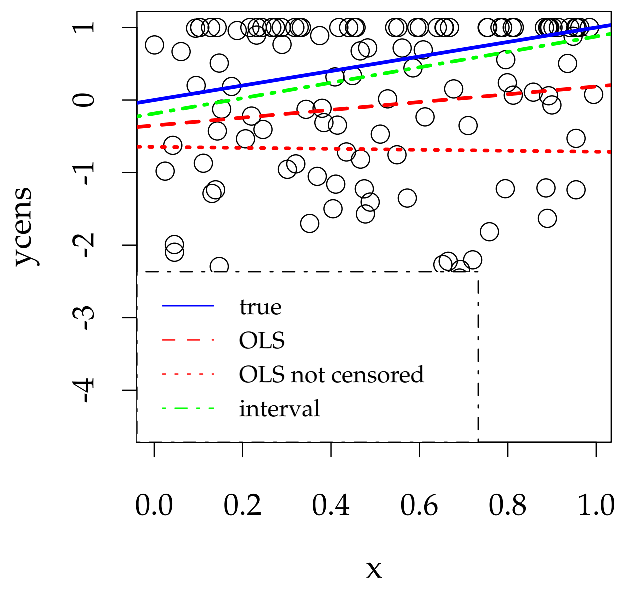
- $[x, x]$ — known observation
- $[x, \infty]$ — observation is larger than x
- $[\infty, x]$ — observation is smaller than x
- $[x, y]$ — observation is between x and y

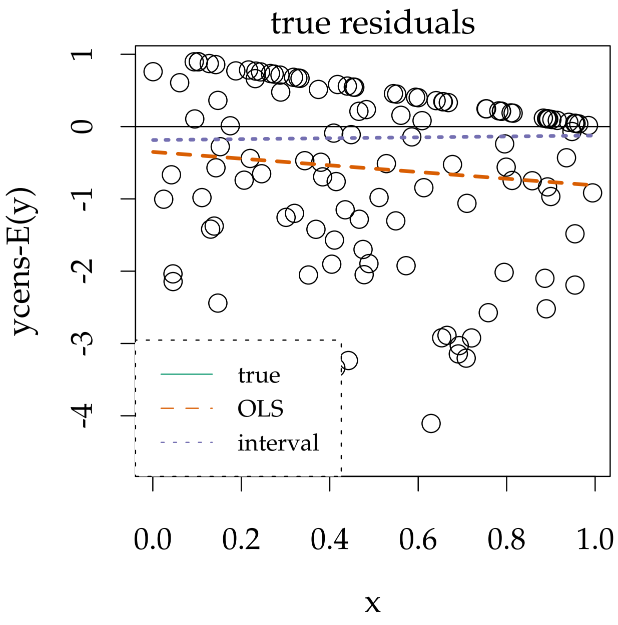
In Stata and in R we use missings (in Stata `.` and in R `NA` to indicate ∞.

```
ymin<-ycens
ymax<-ycens
ymax[ycens==1]<-NA
(intreg <- survreg(Surv(ymin,ymax,type="interval2") ~ x,dist='gaussian'))

Call:
survreg(formula = Surv(ymin, ymax, type = "interval2") ~ x, dist = "gaussian")

Coefficients:
(Intercept)           x
     -0.186       1.062

Scale= 1.739

Loglik(model)= -187.2   Loglik(intercept only)= -188.8
Chisq= 3.21 on 1 degrees of freedom, p= 0.073
n= 120
```

The estimated β is not perfect, but much better then the naïve OLS estimates above. Let us show the estimated regression line in a graph:

```
plot(ycens ~ x)
abline(a=0,b=1,lwd=3,col="blue")
abline(olsall,olsrel,lty=2,col="red",lwd=3)
abline(olsrel,lty=3,col="red",lwd=3)
abline(intreg,lty=4,col="green",lwd=3)
legend("bottomleft",c("true","OLS","OLS not censored","interval"),lty=1:4,col=c("blue","red","
#
plot(ycens-x ~ x,main="true residuals",ylab="ycens-E(y)")
abline(h=0)
lines(predict(olsall) -x  ~ x,lty=2,col=2,lwd=3)
lines(predict(intreg) -x   ~ x,lty=3,col=3,lwd=3)
legend("bottomleft",c("true","OLS","interval"),col=1:3,lty=1:3,bg="white",cex=.7)
```





The graph on the right side of the figure shows again the true residuals, i.e. $y - E(y)$. We should note two things:

- The OLS estimate typically *underestimates* the relationship. Reason: The extreme values are missing (censored)
- The Interval regression *can* overestimate. It is not necessarily very stable.

Of course, the same can be done in Stata:

```
drop _all
set obs 1000
gen x=uniform()
gen y=x+.2*invnorm(uniform())
reg y x
gen ycens=y
replace ycens=1 if y>=1
reg ycens x
reg ycens x if ycens<1
gen ymin=ycens
gen ymax=ycens
replace ymax=. if ycens==1
list ycens ymin ymax in 1/100
intreg ymin ymax x
```

4.4. Theoretical background

4.4.1. The maximum likelihood method

Remember the maximum likelihood approach for the logistic model:

$$L = \prod_{y_i=1} F(x_i'\beta) \cdot \prod_{y_i=0} \left(1 - F(x_i'\beta)\right)$$

The interval regression is quite similar (now y_i is an interval)

$$L = \prod_{i \in C_0} f(x_i\beta - y) \cdot \prod_{i \in C_{-1}} (1 - F(x_i\beta - y)) \cdot \prod_{i \in C_{+1}} F(x_i\beta - y)$$

	C_0	not censored
where	C_{-1}	censored from below
	C_{+1}	censored from above

$$\dot{y} = x_i'\beta + \epsilon \qquad -\epsilon = x_i'\beta - \dot{y}$$

4.5. Bayesian censored model

We need (JAGS) notation for interval-censored data:

`Y ~ dinterval(t, c)`

$$Y = \begin{cases} 0 & \text{if } t \leq c[1] \\ m & \text{if } c[m] < t \leq c[m+1] \quad \text{for } 1 \leq m < M \\ M & \text{if } c[M] < t \end{cases}$$

Here our data is censored from above, i.e.

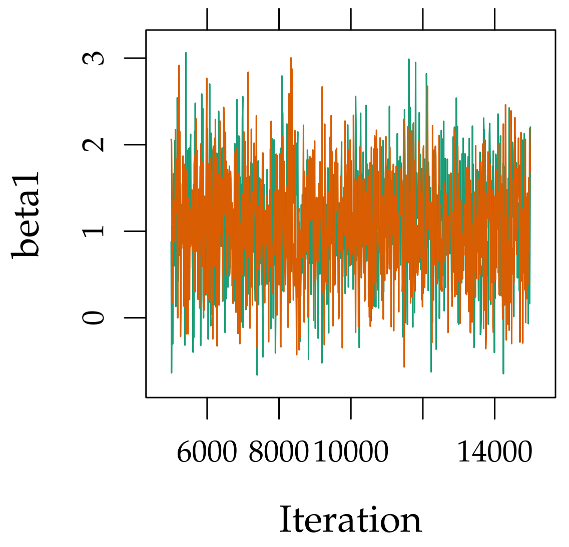
$$Y = \begin{cases} 0 & \text{if } t \leq 1 \\ 1 & \text{if } 1 < t \end{cases}$$

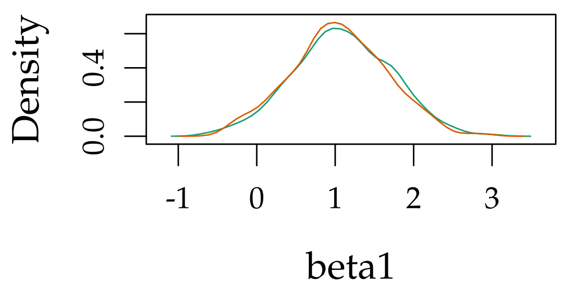
One complication with the censored model is that the censored observations are unknown, so JAGS will fit random values. Unless we help JAGS a little, the initial values will be inconsistent, i.e. JAGS will randomise values for y which are not in the right interval.

```
library(runjags)
is.censored <- ycens>=1
yc       <- ifelse(is.censored,NA,ycens)
yInit    <- ifelse(is.censored,2,NA) #<- must resolve initial uncertainty about censored y
dataList<-list(y=yc,x=x,is.censored=as.numeric(is.censored))
initList<-list(beta0=.5,beta1=.5,tau=1,y=yInit)
```

```
modelC <- 'model {
 for (i in 1:length(y)) {
       y[i] ~ dnorm(beta0+beta1*x[i],tau)
       is.censored[i] ~ dinterval(y[i],1)
    }
    beta0    ~ dnorm (0,.0001)
    beta1    ~ dnorm (0,.0001)
    tau   ~ dgamma(.01,.01)
    sd <- 1/sqrt(tau)
  }
}'
bayesC<-run.jags(model=modelC,data=dataList,
       inits=list(initList,initList),monitor=c("beta0","beta1","sd"))

Calling the simulation using the parallel method...
Following the progress of chain 1 (the program will wait for all chains to finish before cont
Welcome to JAGS 3.4.0 on Sat Feb 21 18:43:01 2015
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
. . Reading data file data.txt
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 609
. Reading parameter file inits1.txt
. Initializing model
. Adapting 1000
-------------------------------------------------| 1000
++++++++++++++++++++++++++++++++++++++++++++++++++ 100%
Adaptation successful
. Updating 4000
-------------------------------------------------| 4000
************************************************** 100%
. . . . Updating 10000
-------------------------------------------------| 10000
************************************************** 100%
. . . Updating 0
. Deleting model
.
All chains have finished
Simulation complete.  Reading coda files...
Coda files loaded successfully
Calculating the Gelman-Rubin statistic for 3 variables....
The Gelman-Rubin statistic is below 1.05 for all parameters
Finished running the simulation
```

```
plot(bayesC,var="beta1",type=c("trace","density"))

Producing 2 plots for 1 variables to the active graphics device
(see ?runjagsclass for options to this S3 method)
```





```
summary(bayesC)$quantiles[,c("2.5%","97.5%")]

         2.5%  97.5%
beta0 -0.8813 0.5136
beta1 -0.1282 2.3281
sd     1.5058 2.1302

confint(intreg)

                 2.5 % 97.5 %
(Intercept) -0.8528 0.4808
x           -0.1049 2.2296
```

5. Multinomial (polytomous) logit

5.1. Motivation and background

Multinomial logit

- choices are mutually exclusive
- choices are exhaustive
- choices are finite

Problems

- one can map problems that do not look mutually exclusive or not exhaustive into a problem that is

  E.g.: heating modes: gas / oil / wood / electricity

  What about households which use, e.g., gas + electricity $\rightarrow$

  - introduce an additional category
  - ask for 'primary source of heating'

  Some households do not use any of the above:

  - introduce an additional category

- Using discrete choice models for metric variables
  - E.g.: consumption of goods which follow a non-linear tariff (telephone, electricity)

5.1.1. Random utility models

Can we tell a story like in the logit/probit case?

A latent variable model (random utility model):

$$\begin{aligned} \eta_1 &= x'\beta_1 + \xi_1 \\ \eta_2 &= x'\beta_2 + \xi_2 \\ \eta_3 &= x'\beta_3 + \xi_3 \\ &\vdots \end{aligned}$$

The decision maker chooses alternative k if $\eta_k \geq \eta_j$ for all j

Note: these models are equivalent to their affine transformations.

5.1.2. Normalisations

- We often normalise the constant part of one of the equations to zero.
- If ξ_j are i.i.d. we often normalise their variance to a convenient value.

  (this implies that different distributions for ξ will lead to different scales for coefficients — logit coefficients will be $\pi/\sqrt{6}$ times larger than probit.)

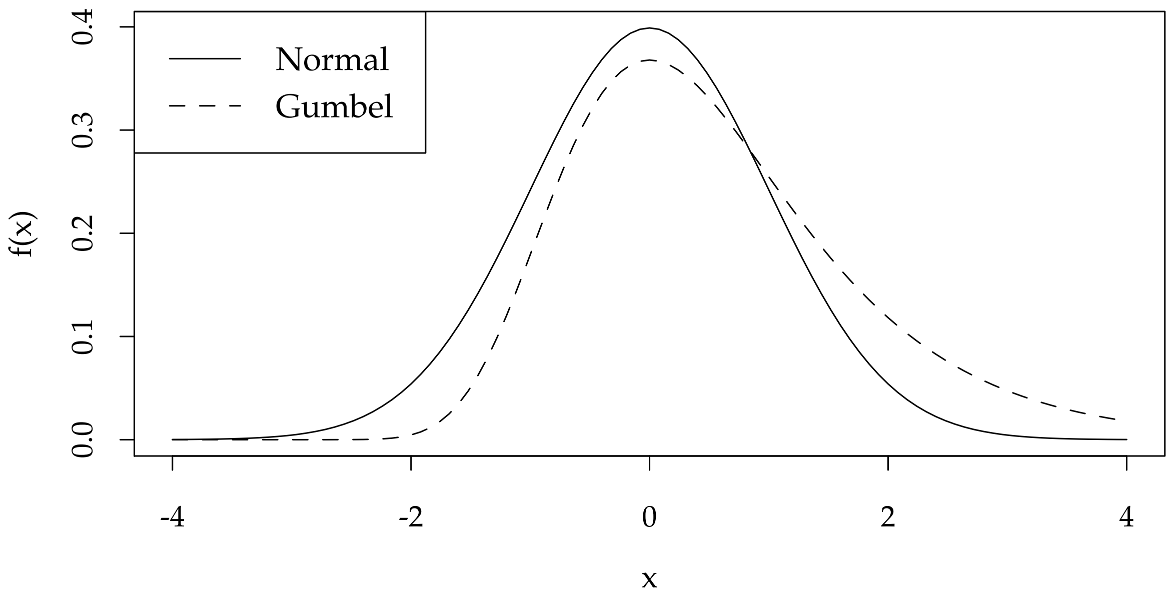
5.1.3. Differences

Let us look at the differences between two alternatives:

$$v_{kj} = \eta_k - \eta_j = x'(\beta_k - \beta_j) + \xi_k - \xi_j$$

- $\xi \sim N(0,1)$: $\xi_k - \xi_j$ has variance 2 and covariance 1 (for $k \neq j$)

```r
dgumbel<-function(x) exp(-exp(-x)-x)
plot(dnorm,-4,4,ylab="f(x)")
curve(dgumbel,add=TRUE,lty=2)
legend("topleft",c("Normal","Gumbel"),lty=1:2)
```



- $\xi \sim$ Gumbel $\left(F_{\text{Gumbel}}(\xi) = e^{-e^{-\xi}}\right)$ then
  - the difference ν_{ki} follows a logistic distribution

$$\Pr(y=k|\xi_k) = \prod_{j\neq k} F_{\text{Gumbel}}(x'(\beta_k - \beta_j) + \xi_k)$$

average over ξ_k

$$\Pr(y=k) = \int f_{\text{Gumbel}}(\xi_k) \prod_{j\neq k} F_{\text{Gumbel}}(x'(\beta_k - \beta_j) + \xi_k)\, d\xi_k$$

$$\Pr(y=k) = \frac{e^{x'\beta_k}}{\sum_{i=1}^m e^{x'\beta_i}}$$

  - we get the following multinomial logit (McFadden)

$$\begin{aligned}
\Pr(y=1) &= \frac{e^{x'\beta_1}}{\sum_{k=1}^m e^{x'\beta_k}} \\
\Pr(y=2) &= \frac{e^{x'\beta_2}}{\sum_{k=1}^m e^{x'\beta_k}} \\
\Pr(y=3) &= \frac{e^{x'\beta_3}}{\sum_{k=1}^m e^{x'\beta_k}} \\
&\vdots
\end{aligned}$$

- $0 < \Pr(y = k) < 1$
- $\sum_k \Pr(y = k) = 1$

$\uparrow$ β_k are not identified

Normalise:

$$\begin{aligned}
\Pr(y = 1) &= \frac{1}{1 + \sum_{k=2}^{m} e^{x'\beta_k}} \\
\Pr(y = 2) &= \frac{e^{x'\beta_2}}{1 + \sum_{k=2}^{m} e^{x'\beta_k}} \\
\Pr(y = 3) &= \frac{e^{x'\beta_3}}{1 + \sum_{k=2}^{m} e^{x'\beta_k}} \\
&\vdots
\end{aligned}$$

the odds ratios are:

$$\frac{\Pr(y = k)}{\Pr(y = 1)} = e^{x'\beta_k}$$

That is a strong assumption on the error terms.

5.1.4. Indenpendence from irrelevant alternatives — IIA

E.g. in a model where the dependent is choice of travel mode, an unobservable might be a personal preference for/against means of mass transportation (tube/train). But then these choices/error terms are correlated.

$\rightarrow$ logit can represent systematic variation of choices (explained by observed characteristics) but *not* individual (unobserved) variation of choices.

The log-likelihood:

$$\ln L = \sum_i \sum_{k=1}^{m} I_k(y_i) \ln \Pr(y_i = k)$$

with $I_k(y_i) = \begin{cases} 1 & \text{if } y_i = i \\ 0 & \text{otherwise} \end{cases}$

this function $\ln L$ is globally concave in β (McFadden, 1974)

5.2. Example

Let us first create individual characteristics, x1, x2, x3.

```
N<-100
sd<-10
ex <- cbind(x1=runif(N),x2=runif(N))
head(ex)

          x1     x2
[1,] 0.28758 0.6000
[2,] 0.78831 0.3328
[3,] 0.40898 0.4886
[4,] 0.88302 0.9545
[5,] 0.94047 0.4829
[6,] 0.04556 0.8904
```

The following matrix determines how individual characteristics translate into preferences for three choices:

```
mat<-rbind(c(400,0),
          c(250,200),
          c(100,300))
mat

     [,1] [,2]
[1,]  400    0
[2,]  250  200
[3,]  100  300
```

```
latent<-(ex %*% t(mat)) +  sd *
                    cbind(rnorm(N),rnorm(N),rnorm(N))
head(latent)

       [,1]  [,2]  [,3]
[1,] 107.93 213.9 201.6
[2,] 317.89 276.8 171.2
[3,] 161.12 197.3 178.1
[4,] 349.73 417.1 364.1
[5,] 366.67 327.6 234.5
[6,]  17.77 184.7 275.0

max.col(latent)

 [1] 2 1 2 2 1 3 2 1 2 1 2 1 1 3 3 1 3 3 3 1 1 1 1 1 1 2 1 1 2 3 1 2 2 2 3 2 2 3
[39]  3 3
 [ reached getOption("max.print") -- omitted 60 entries ]
```

```
choice <- max.col(latent)
library(nnet)
(est<-multinom(choice ~ x1 + x2,as.data.frame(ex)))

# weights:  12 (6 variable)
initial  value 109.861229
iter  10 value 18.323213
iter  20 value 16.923568
iter  30 value 16.881715
iter  40 value 16.880637
iter  50 value 16.880332
iter  60 value 16.880044
iter  70 value 16.879931
final  value 16.879896
converged
Call:
multinom(formula = choice ~ x1 + x2, data = as.data.frame(ex))

Coefficients:
  (Intercept)     x1    x2
2      0.9445 -25.81 31.72
3      0.1557 -58.60 52.67

Residual Deviance: 33.76
AIC: 45.76
```

Note that the estimated coefficients are not the matrix of coefficients `mat` that we employed above. However, they are a projection. We are expecting this:

```
mat

     [,1] [,2]
[1,]  400    0
[2,]  250  200
[3,]  100  300
```

but we got that:

```
coef(est)

  (Intercept)     x1    x2
2      0.9445 -25.81 31.72
3      0.1557 -58.60 52.67
```

The estimator normalises the first category to zero

```
mat

     [,1] [,2]
[1,]  400    0
[2,]  250  200
[3,]  100  300
```

```
mat - cbind(c(1,1,1)) %*% mat[1,]

     [,1] [,2]
[1,]    0    0
[2,] -150  200
[3,] -300  300
```

and sets the variance to one:

```
(mat -  cbind(c(1,1,1)) %*% mat[1,])*pi / sqrt(6) / 10

       [,1]  [,2]
[1,]   0.00  0.00
[2,] -19.24 25.65
[3,] -38.48 38.48
```

To access estimation results we can use similar commands as with `lm`:

```
coef(est)

  (Intercept)     x1    x2
2      0.9445 -25.81 31.72
3      0.1557 -58.60 52.67

confint(est)

, , 2

              2.5 % 97.5 %
(Intercept)  -3.099  4.988
x1          -43.459 -8.153
x2           11.420 52.021

, , 3

              2.5 %  97.5 %
(Intercept)  -4.745   5.056
x1          -89.593 -27.602
x2           26.237  79.094
```

To do the same exercise in Stata, we have to export the dataset:

```
library(foreign)
write.dta(as.data.frame(cbind(choice,ex)),file="multinom.dta")
```

```
use multinom.dta,clear
desc
mlogit choice x1 x2 x3,baseoutcome(1)
```

5.3. Bayesian multinomial

```
modelM <- 'model {
 for (i in 1:length(y)) {
        for (j in 1:3) { # three different choices
          exb[i,j] <- exp(inprod(beta[,j],ex[i,]))
        }
        y[i] ~ dcat(exb[i,1:3])
    }
    for (k in 1:K) {
        beta[k,1] <- 0 # identifying restriction
    }
    for (j in 2:3) {
      for (k in 1:K) {
         beta[k,j] ~ dnorm(0,.0001)
      }
    }
}'
dataList<-list(y=choice,ex=cbind(1,ex),K=dim(ex)[2]+1)
bayesM <-run.jags(model=modelM,data=dataList,monitor=c("beta"))

Calling the simulation using the parallel method...
Following the progress of chain 1 (the program will wait for all chains to finish before co
Welcome to JAGS 3.4.0 on Sat Feb 21 18:43:08 2015
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
. . Reading data file data.txt
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 1116
. Reading parameter file inits1.txt
. Initializing model
. Adapting 1000
-------------------------------------------------| 1000
++++++++++++++++++++++++++++++++++++++++++++++++++ 100%
Adaptation successful
. Updating 4000
-------------------------------------------------| 4000
************************************************** 100%
. . Updating 10000
-------------------------------------------------| 10000
************************************************** 100%
. . . Updating 0
. Deleting model
.
All chains have finished
Simulation complete.  Reading coda files...
Coda files loaded successfully
*WARNING* The monitored variables 'beta[1,1]', 'beta[2,1]' and
'beta[3,1]' appear to be non-stochastic; they will not be included in
```

```
the convergence diagnostic
Calculating the Gelman-Rubin statistic for 9 variables....
The Gelman-Rubin statistic is below 1.05 for all parameters
Finished running the simulation
```

```
summary(bayesM)$quantiles[-c(1,2,3),c("2.5%","50%","97.5%")]

              2.5%       50%    97.5%
beta[1,2]   -2.649   1.18089    5.994
beta[2,2]  -49.098 -30.36380  -16.752
beta[3,2]   21.521  37.43575   55.481
beta[1,3]   -4.808   0.08833    5.854
beta[2,3] -113.850 -69.31255  -44.593
beta[3,3]   40.716  62.54645   92.264

confint(est)

, , 2

              2.5 % 97.5 %
(Intercept)  -3.099  4.988
x1          -43.459 -8.153
x2           11.420 52.021

, , 3

              2.5 %  97.5 %
(Intercept)  -4.745   5.056
x1          -89.593 -27.602
x2           26.237  79.094
```

6. Ordered probit

6.1. Model

We observe whether latent variables $x'\beta$ are in an interval

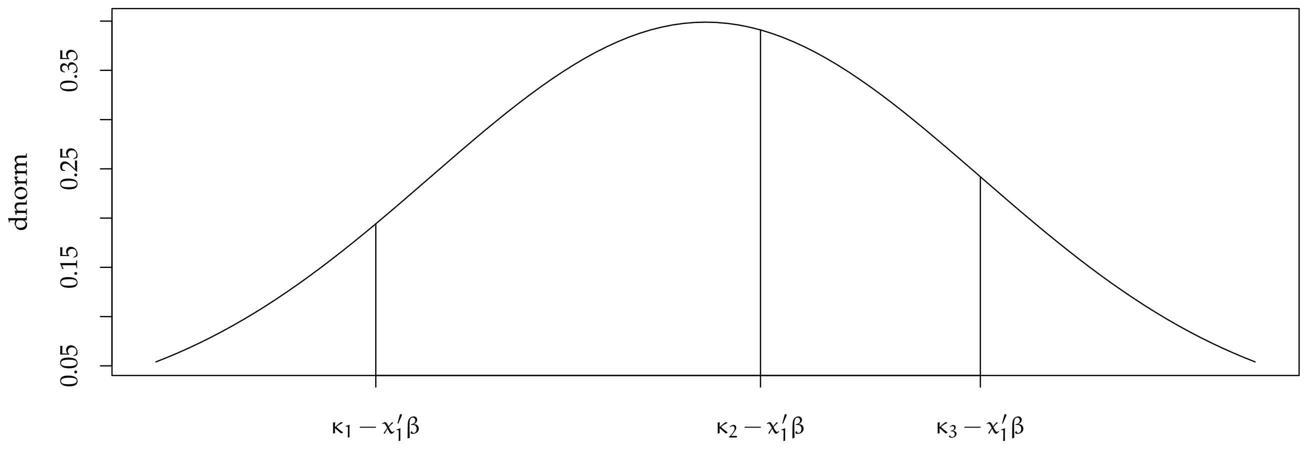
$$\begin{array}{rcl}
\Pr(y_i = 1) & = & \Pr(\kappa_0 < x_i'\beta + u \leq \kappa_1) \\
\Pr(y_i = 2) & = & \Pr(\kappa_1 < x_i'\beta + u \leq \kappa_2) \\
\Pr(y_i = 3) & = & \Pr(\kappa_2 < x_i'\beta + u \leq \kappa_3) \\
 & \vdots &
\end{array}$$

or (solving for u)

$$\begin{aligned}
\Pr(y_i = 1) &= \Pr(\kappa_0 - x_i'\beta < u \leq \kappa_1 - x_i'\beta) \\
\Pr(y_i = 2) &= \Pr(\kappa_1 - x_i'\beta < u \leq \kappa_2 - x_i'\beta) \\
\Pr(y_i = 3) &= \Pr(\kappa_2 - x_i'\beta < u \leq \kappa_3 - x_i'\beta) \\
&\vdots
\end{aligned}$$

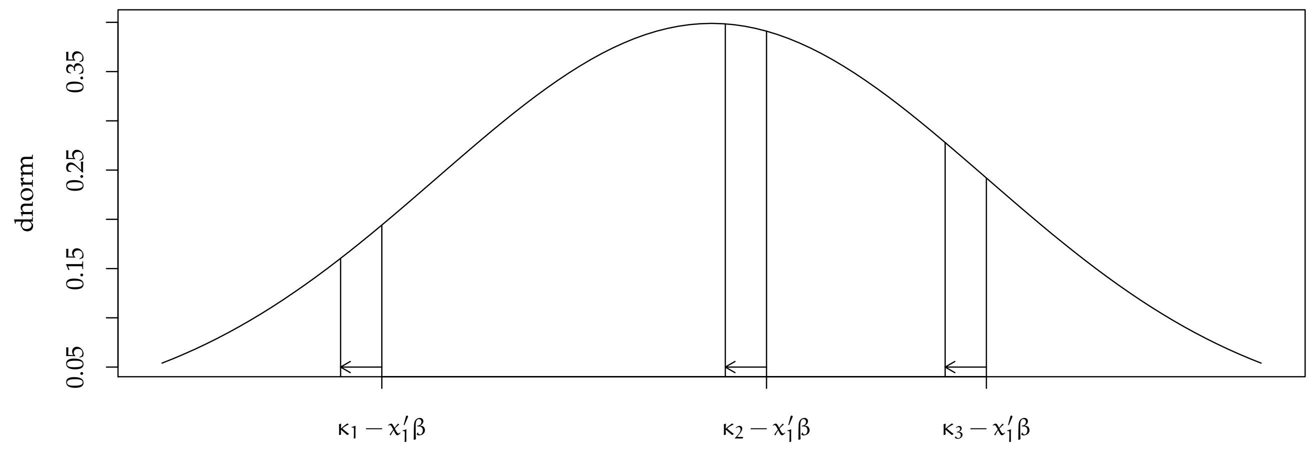
The u can follow any (standard) distribution (logistic, normal, ...)

```
plot(dnorm,-2,2,xaxt="n",xlab=NA)
kappas<-c(-1.2,.2,1)
for(i in 1:length(kappas)) {x<-kappas[i];lines(c(x,x),c(0,dnorm(x)))}
axis(1,kappas,sapply(1:length(kappas),function(d) sprintf("$\\kappa_%d - x_1'\\beta$",d)))
```



Marginal effects:

```
plot(dnorm,-2,2,xaxt="n",xlab=NA)
kappas<-c(-1.2,.2,1)
for(i in 1:length(kappas)) {
    x<-kappas[i];lines(c(x,x),c(0,dnorm(x)))
    y<-kappas[i]-.15;lines(c(y,y),c(0,dnorm(y)))
    arrows(x,.05,y,.05,length=.05)
}
axis(1,kappas,sapply(1:length(kappas),function(d) sprintf("$\\kappa_%d - x_1'\\beta$",d)))
```



6.1.1. The maximum likelihood problem

$$\begin{aligned}
\Pr(y_i = 1) &= \Pr(\kappa_0 - x_i'\beta < u \le \kappa_1 - x_i'\beta) \\
\Pr(y_i = 2) &= \Pr(\kappa_1 - x_i'\beta < u \le \kappa_2 - x_i'\beta) \\
\Pr(y_i = 3) &= \Pr(\kappa_2 - x_i'\beta < u \le \kappa_3 - x_i'\beta) \\
&\vdots
\end{aligned}$$

$$\ln L = \sum_i \sum_{k=1}^{m} I_k(y_i) \ln \Pr(y_i = k)$$

with $I_k(y_i) = \begin{cases} 1 & \text{if } y_i = i \\ 0 & \text{otherwise} \end{cases}$

6.2. Illustration — the Fair data

As an illustration, let us look at a dataset on extramarital affairs, collected by Ray Fair. Two variables from the dataset are

- `ym` number of years married
- `rate` self rating of mariage (unhappy=1...5=happy)

Does the rating of marriage change over time? A naïve approach would be to use OLS and to explain `rate` as a linear function of `ym`.

```
library(MASS)
library(Ecdat)
data(Fair)
lm(rate ~ ym,data=Fair)


Call:
lm(formula = rate ~ ym, data = Fair)

Coefficients:
(Intercept)           ym
     4.3255      -0.0481
```

This approach would assume that all ratings are equidistant. More appropriate is, perhaps, an ordered logistic model. . .

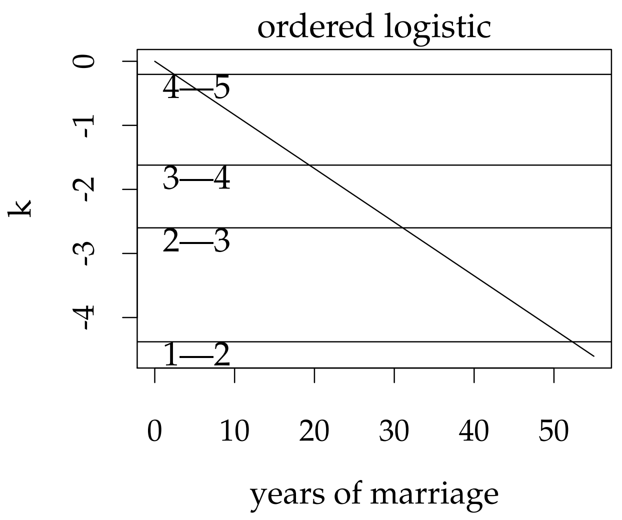
```
(estL<-polr(factor(rate) ~ ym,data=Fair))

Call:
polr(formula = factor(rate) ~ ym, data = Fair)

Coefficients:
      ym
-0.08371

Intercepts:
    1|2     2|3     3|4     4|5
-4.3787 -2.5997 -1.6208 -0.2043

Residual Deviance: 1597.27
AIC: 1607.27
```

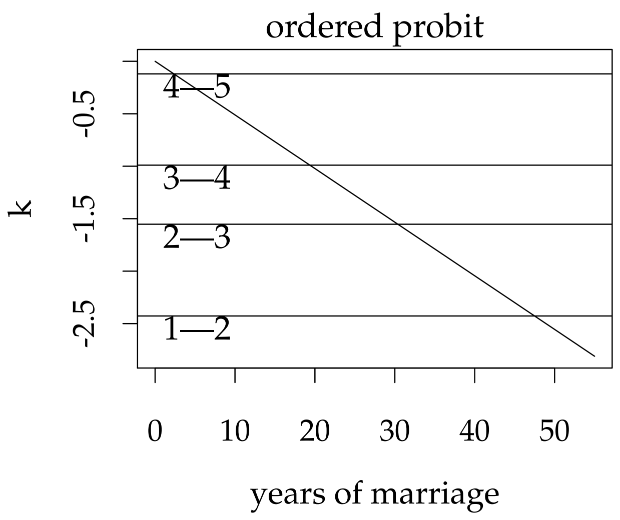
. . . or an ordered probit:

```
(estP<-polr(factor(rate) ~ ym,data=Fair,method="probit"))

Call:
polr(formula = factor(rate) ~ ym, data = Fair, method = "probit")

Coefficients:
      ym
-0.05111

Intercepts:
    1|2     2|3     3|4     4|5
-2.4272 -1.5529 -0.9901 -0.1198

Residual Deviance: 1594.99
AIC: 1604.99
```

The following graph illustrates the estimated thresholds κ_i:

```
probFig <- function (est,main) {
  plot(function(x) {x * est$coef},0,55,ylab=expression(kappa),xlab="years of marriage",main=ma
  for (a in est$zeta) {
    abline(h=a)
    lab=names(est$zeta)[which(est$zeta==a)]
    text(1,a,labels=lab,adj=c(0,1))
  }
}
probFig(estL,main="ordered logistic")
probFig(estP,main="ordered probit")
```

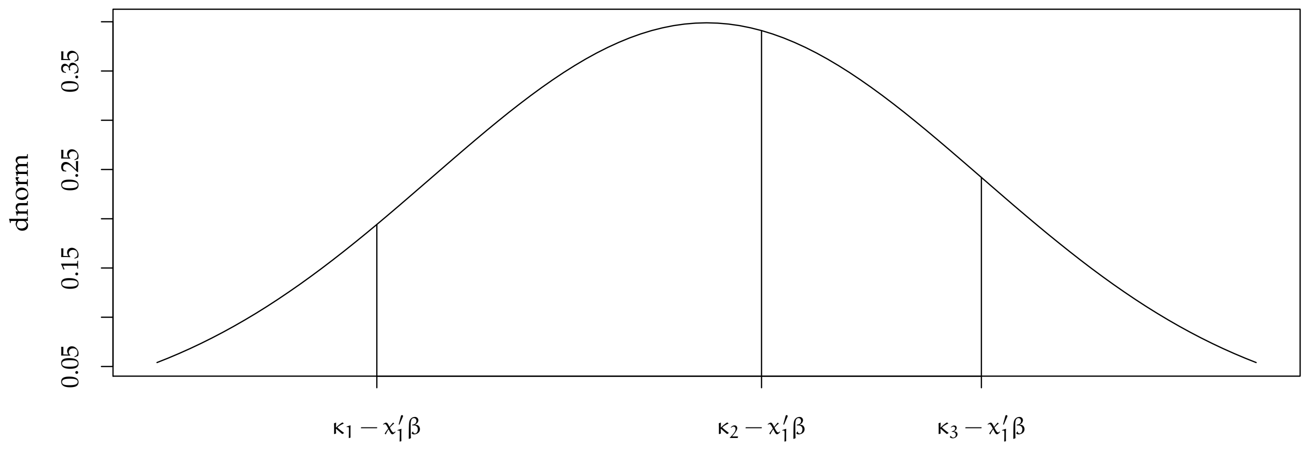




To do the same exercise in Stata, we have to export the dataset:

```
library(foreign)
write.dta(Fair,file="Fair.dta")
```

```
use Fair.dta,clear
desc
ologit rate ym
```



- Dependent variable `y[i]`
- Latent variable `t[i]`
- Independent variable `x[i]`
- Parameters `beta, kappa[j]`

```
model0 <- 'model {
 for (i in 1:length(y)) {
    y[i] ~ dinterval(t[i],kappa)
    t[i] ~ dnorm(beta*x[i],1)
 }
 for (j in 1:K) {
    kappa0[j] ~ dnorm(0,.0001)
 }
 kappa[1:4] <- sort(kappa0)
 beta ~ dnorm(0,.0001)
}'
dataList<-list(y=Fair$rate-1,x=Fair$ym,K=max(Fair$rate)-1)
initList<-with(dataList,list(t=y+1/2,kappa0=1:K))
bayes0 <-run.jags(model=model0,data=dataList,inits=list(initList,initList),
                  monitor=c("beta","kappa"))

Calling the simulation using the parallel method...
Following the progress of chain 1 (the program will wait for all chains to finish before co
Welcome to JAGS 3.4.0 on Sat Feb 21 18:43:29 2015
JAGS is free software and comes with ABSOLUTELY NO WARRANTY
Loading module: basemod: ok
Loading module: bugs: ok
. . Reading data file data.txt
. Compiling model graph
   Resolving undeclared variables
   Allocating nodes
   Graph Size: 1822
. Reading parameter file inits1.txt
. Initializing model
. Adapting 1000
-------------------------------------------------| 1000
++++++++++++++++++++++++++++++++++++++++++++++++++ 100%
Adaptation successful
. Updating 4000
-------------------------------------------------| 4000
************************************************** 100%
. . . Updating 10000
-------------------------------------------------| 10000
************************************************** 100%
. . . Updating 0
. Deleting model
.
All chains have finished
```

```
Simulation complete.  Reading coda files...
Coda files loaded successfully
Calculating the Gelman-Rubin statistic for 5 variables....
Convergence may have failed for this run for 5 parameters after 10000
iterations (multi-variate psrf = 1.296)
Finished running the simulation
```

```
summary(bayesO)$quantiles[,c("2.5%","50%","97.5%")]

             2.5%      50%     97.5%
beta     -0.06787 -0.04768  0.002842
kappa[1] -2.73985 -2.37272 -1.774619
kappa[2] -1.74902 -1.49953 -0.866708
kappa[3] -1.16440 -0.95374 -0.291193
kappa[4] -0.29059 -0.09269  0.476098

estP

Call:
polr(formula = factor(rate) ~ ym, data = Fair, method = "probit")

Coefficients:
      ym
-0.05111

Intercepts:
    1|2     2|3     3|4     4|5
-2.4272 -1.5529 -0.9901 -0.1198

Residual Deviance: 1594.99
AIC: 1604.99
```

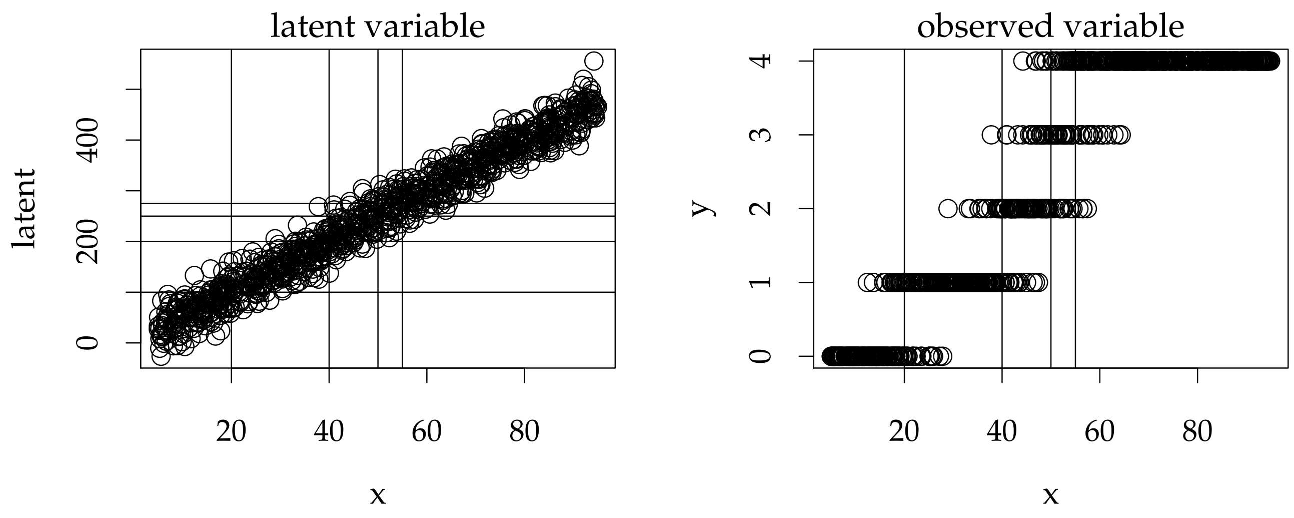
6.3. Illustration II — a simulated dataset

In the following we generate a latent variable `latent`. The problem is that we can not observe `latent`. We only see y which indicates a range of possible values for `latent`. These ranges are defined by `cuts`.

```
set.seed(123)
N <-1000
sd<-25
b <-5
x <-runif(N,min=5,max=95)
latent<-b*x+sd*rnorm(N)
cuts<-sort(c(100,200,250,275))
y <- sapply(latent,function(x) sum(x>cuts))
```

The following graph illustrates the problem. On the left we have the latent variable on the vertical axis. Our observed variable is shown in the right diagram. This is the data we use in our regression.

```
plot(latent ~ x,main="latent variable")
abline(h=cuts)
abline(v=cuts/b)
plot (y ~ x,main="observed variable")
abline(v=cuts/b)
```



polr estimates the ordered model. By default the logistic method is used.

```
polr(factor(y) ~ x)

Call:
polr(formula = factor(y) ~ x)

Coefficients:
     x
0.3657

Intercepts:
   0|1    1|2    2|3    3|4
 7.324 14.652 18.034 19.852

Residual Deviance: 748.40
AIC: 758.40
```

Is this the value we should expect? Remember that the logistic distribution has $\sigma = \pi/\sqrt{3}$. As the coefficient for x we should, hence, expect

```
b/sd*pi/sqrt(3)

[1] 0.3628
```

and as intercepts

```
cuts/sd*pi/sqrt(3)

[1]  7.255 14.510 18.138 19.952
```

which is both close to the estimated coefficients.
The standard normal distribution has $\sigma = 1$, hence we get different results with the probit method:

```
polr(factor(y) ~ x,method="probit")

Call:
polr(formula = factor(y) ~ x, method = "probit")

Coefficients:
     x
0.2037

Intercepts:
   0|1    1|2    2|3    3|4
 4.103  8.161 10.050 11.069

Residual Deviance: 744.04
AIC: 754.04
```

As the coefficient for x we should expect

```
b/sd

[1] 0.2
```

and as intercepts

```
cuts/sd

[1]  4  8 10 11
```

7. Count data

7.1. Model

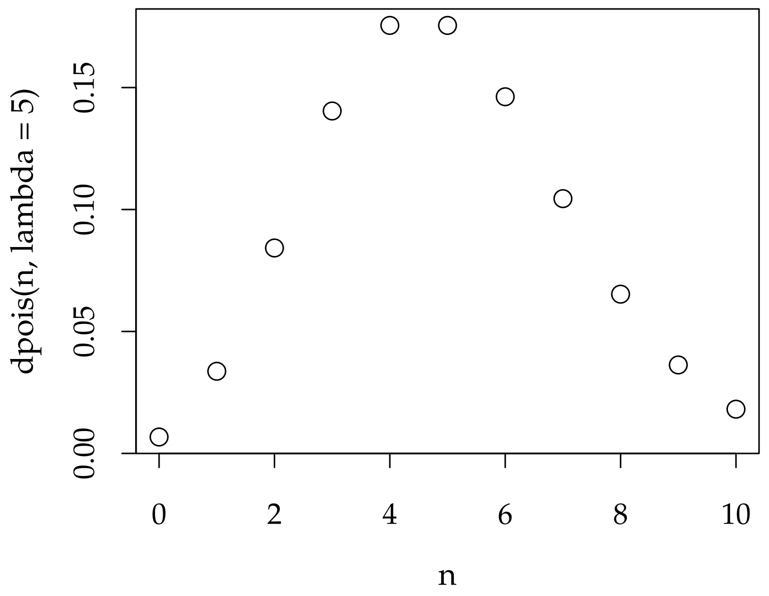
Poisson process (birth-only process):
be $N(t)$ the number of arrivals until time t

$$\Pr\left(N(t+\tau) - N(t) = y\right) = \frac{e^{-\lambda\tau}(\lambda\tau)^y}{y!}$$

λ = expected number of arrivals per unit of time
Poisson distribution:

$$\Pr(Y_i = y_i | x_i) = \frac{e^{-\lambda_i}\lambda_i^{y_i}}{y_i!}$$

```
n<-0:10;
plot(n,dpois(n,lambda=5),t="p")
```



Link function It is convenient to assume

$$\ln \lambda_i = x_i'\beta$$

$$\Pr(Y_i = y_i | x_i) = \frac{e^{-e^{x_i'\beta}} \left(e^{x_i'\beta}\right)^{y_i}}{y_i!}$$

$$\ln L = \sum_{i=1}^{n} -e^{x_i'\beta} + y_i x_i'\beta - \ln y_i!$$

$$\frac{\partial \ln L}{\partial \beta} = \sum_{i=1}^{n} \left(y_i - e^{x_i'\beta}\right) x_i = 0$$

The Hessian is negative definite and Newton's method converges quickly

7.2. Illustration — the Fair data

The dataset collected by Ray Fair contains another interesting variable:

- `nbaffairs` number of affairs in past year

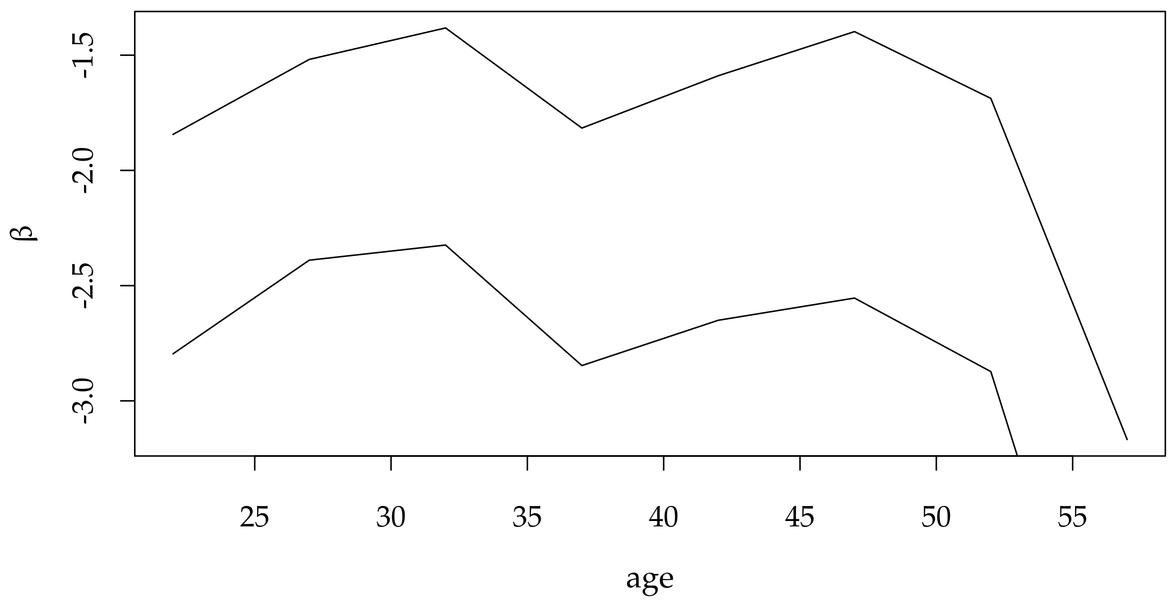
We will assume that this variable follows a Poisson distribution.

```
library(Ecdat)
library(MASS)
data(Fair)
table(Fair$nbaffairs)


  0   1   2   3   7  12
451  34  17  19  42  38
```

```
(est<-glm(nbaffairs ~ sex + rate + ym + age + child +
    religious + occupation,family=poisson(link=log),
        data=Fair))


Call:  glm(formula = nbaffairs ~ sex + rate + ym + age + child + religious +
    occupation, family = poisson(link = log), data = Fair)

Coefficients:
(Intercept)      sexmale         rate           ym          age      childyes
    2.55999      0.05835     -0.41038      0.11695     -0.03302      -0.00256
  religious   occupation
   -0.35479      0.07211

Degrees of Freedom: 600 Total (i.e. Null);  593 Residual
Null Deviance:     2930
Residual Deviance: 2360  AIC: 2870
```

While it is tempting to include each and every variable, not all explanatory variables seem to have a linear impact.

```
est2<-glm(nbaffairs ~ sex + rate + ym + factor(age) + child +
    religious + occupation,family=poisson(link=log),
        data=Fair)
xx<-confint(est2)[grep("age",names(coef(est2))),]
xx<-cbind(xx,as.numeric(sub("factor\\(age\\)","",rownames(xx))))
plot(xx[,2] ~ xx[,3],xlab='age',ylab='$\\beta$',t='l')
lines(xx[,1] ~ xx[,3])
```



In this context we should mention the `dropterm` function:

```
dropterm(est,sort=TRUE)

Single term deletions

Model:
nbaffairs ~ sex + rate + ym + age + child + religious + occupation
           Df Deviance  AIC
child       1     2360 2868
sex         1     2360 2868
<none>            2360 2870
occupation  1     2370 2878
age         1     2393 2901
ym          1     2479 2987
religious   1     2493 3001
rate        1     2573 3081
```

Let us, hence, look at a simpler model:

```
est2<-glm(nbaffairs ~ rate + religious + ym ,
    family=poisson(link=log),data=Fair)
summary(est2)


Call:
glm(formula = nbaffairs ~ rate + religious + ym, family = poisson(link = log),
    data = Fair)

Deviance Residuals:
   Min      1Q  Median      3Q     Max
-4.204  -1.598  -1.129  -0.745   7.408
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.17408    0.14449    15.1   <2e-16 ***
rate        -0.40219    0.02734   -14.7   <2e-16 ***
religious   -0.36378    0.03074   -11.8   <2e-16 ***
ym           0.07563    0.00683    11.1   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 2925.5  on 600  degrees of freedom
Residual deviance: 2401.6  on 597  degrees of freedom
AIC: 2904

Number of Fisher Scoring iterations: 6
```

As with other models, we have the usual extractor functions:

```
confint(est2)

               2.5 %  97.5 %
(Intercept)  1.88862  2.4551
rate        -0.45568 -0.3485
religious   -0.42418 -0.3037
ym           0.06231  0.0891
```

Of course, the same can be done with Stata.

```
use Fair.dta,clear
desc
poisson nbaffars rate ym child occupatn
```

8. Doing maximum likelihood

Motivation In this course we cover a few applications for the ML method. Many more are conceivable. You might find yourself in a situation where you know the likelihood function but there is not ready-made R or Stata function that implements this function. Don't worry! It is quite straightforward to write your own likelihood maximisers.

As an example, we will look again at the logistic model:

$$L(\beta|\{y,X\}) = \prod_i (F(\theta_i))^{y_i} (1-F(\theta_i))^{1-y_i}$$

or, in logs,

$$\ln L = \sum_i (y_i \ln F(\theta_i) + (1-y_i) \ln (1-F(\theta)))$$

with $\theta_i = x_i'\beta$

Let us first create a few random numbers:

```
set.seed(123)
N<-100
sd<-.2
x <- runif(N)
lat<- x-.5+sd*rnorm(N)
y<-ifelse(lat>0,1,0)
```

If F is the logistic distribution we need a couple of logs $\ln F()$ and $\ln(1-F())$. Luckily, the `plogis` function already implements logs and $1-F()$.

Compare

```
1-plogis(3)

[1] 0.04743
```

and

```
plogis(3,lower.tail=FALSE)

[1] 0.04743
```

This is more than a simplification. What, if our θ becomes so large, that rounding errors come in the way of our calculations? Compare

```
1-plogis(38)

[1] 0
```

with

```
plogis(38,lower.tail=FALSE)

[1] 3.139e-17
```

Both expressions should be the same, but, due to rounding, the are not. This is crucial when we take logs:

```
log(1-plogis(38))

[1] -Inf
```

```
log(plogis(38,lower.tail=FALSE))

[1] -38
```

or, simpler

```
plogis(38,lower.tail=FALSE,log=TRUE)

[1] -38
```

We exploit this, when we write our log-likelihood function:

```
logli=function(beta) {
  n=length(x)
  onevec=rep(1,n)
  theta=x*beta[2]+onevec*beta[1]
  -(sum(y*(plogis(theta,log.p=TRUE))+
    sum((1-y)*plogis(theta,lower.tail=FALSE,log.p=TRUE)))
}
```

This function takes as an argument a vector of coefficients β and returns the log-likelihood:

```
logli(c(-2,6))

[1] 50.88
```

Finding the maximum is now easy. We just give a starting value and let R do the rest (We can ask for more detail by setting `trace` to a larger value):

```
optim(c(-2,6),logli,control=list(trace=0))

$par
[1] -4.095  7.778

$value
[1] 39.46

$counts
function gradient
      55       NA

$convergence
[1] 0

$message
NULL
```

This is the result that we also get with the built in function `glm`

```
glm( y ~ x,family=binomial(link="logit"))


Call:  glm(formula = y ~ x, family = binomial(link = "logit"))

Coefficients:
(Intercept)            x
```

```
     -4.10        7.78

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:     138
Residual Deviance: 78.9  AIC: 82.9
```

Let us export the dataset for Stata, so that we can repeat the exercise there, too:

```
library(foreign)
write.dta(as.data.frame(cbind(x,y)),file="ml.dta")
```

```
use ml
list in 1/10
capture program drop logli
program logli
   args todo b lnf
   tempvar theta
   mleval ‘theta’ = ‘b’
   mlsum ‘lnf’=$ML_y1*ln(exp(‘theta’)/(1+exp(‘theta’)))+
         (1-$ML_y1) * ln(exp(-‘theta’)/(1+exp(-‘theta’)))
end
ml model d0 logli (y = x)
ml check
ml search
ml maximize
logit y x
```

9. What have we learned

Discrete Choice

	R	Stata
binomial choice (logit, probit)	`glm`	`logit`
multinomial choice	`multinom`	`mlogit`
ordered choice	`polr`	`ologit`
count data	`glm`	`poisson`
censored data	`survreg`	`intreg`

Different Methods

- OLS is not suitable for these problems, estimator is biased
- ML approach
  - consistency
  - goodness of fit
  - marginal effects

On the programming side

- datastructures in STATA and R
- libraries in R and extra Matrix languages in STATA
- plotting (plot, hist, abline / twoway scatter, ...)
- creating data (rnorm, runif, ecdf / gen, invnorm, uniform, cumul)
- estimation commands (lm, glm, fitted / reg, logit, probit, predict)
- concatenating things (c,cbind,rbind / "\", append)
- writing functions (function / program)

A. Exercises

A.1. Coconut island

You receive a dataset from a hypothetical friend (`coconut.csv` from the attached set of files)[1]. This friend has visited several tropical islands and has recorded demand for coconuts (quantity) and prices (price) as well as an index for the island (name).

- Run an OLS to explain prices as a function of the demand for coconuts. Draw a diagram where you describe quantity and price on each island. Draw into this diagram the regression line.
- Now run an OLS separately for each island. Plot, for each island separately, the intercept and the slope of the regression. Draw, into the same diagram, the result of the previous regression.
- Do you see any structure in the estimated coefficients? If so, which?
- What is the highest values of the intercept for those islands where you could estimate a regression line
- You can read the file in R with the function `read.table`

  In STATA you read the file with `insheet`

[1]You can extract the files that are bundled with this PDF document with most PDF viewers (including Acrobat Reader) and with pdftk. The icon in Acrobat is either a paper-clip or a drawing-pin.

- You may find it clumsy to run the separate regression for all the islands. In R have a look at the manual entry for the `lm` command. The `subset` option allows you to include only a part of a dataset, the `data` option allows to refer to a specific dataset, without always refering to its name.

  In STATA the `if` qualifier in a `reg` command allows you include only part of a dataset.

A.2. Mosquitos

A biologist is trying to kill mosquitos with a poisonous drug. The attached dataset `mosquito.csv` contails the following information: The id of the mosquito is coded in the first column, the amount of poison in the second column, the status of the mosquito is shown in the third column. The fourth column, finally, contains the id of the research assistant who administered the poison.

- Estimate a probit model that explains survival of the mosquito as a function of the amount of poison.

  What is the critical amount of poison such that the surivial probability is just 1/2?

- Draw a diagram with the marginal effect on the vertical axis and the amount of poison on the horizontal axis.

- Our biologist remembers that two of her assistants joined the team later. As a result, their animals were treated for a shorter time. Unfortunately, our biologist forgot who these assistants were.

  To help her, draw in one diagram for each of the assistants the marginal effects. Are they all similar? Can you identify the two who are different?

A.3. Sellers

Attached you find the dataset `sellers.csv` from a hypothetical experiment where 60 subjects repeatedly had the opportunity to sell an object at a specified price.

The dataset contains 2000 observations. Each line contains one observation. Columns are separated by commas.

The first columns contains an “S” if the subject was sold, otherwise an “H” The second column contains the price. The third column contains the subject id (a number between 1 and 60)

- Read the data and inspect it carefully for coding mistakes. Which mistakes do you find? Unfortunately, coding mistakes are very frequent in most of the data that you will receive. Developing strategies to detect coding mistakes and to eliminate them ex post is a major part of this exercise.

- Assume that all subjects are alike. Estimate the decision of the subjects as a logit model and determine the threshold between sell and hold.
- Now assume that subjects are different. Estimate for each subject the threshold. Plot the cumulative distribution function of thresholds.

Help for R

- When you read files with `read.table` R tries to guess what is the best format to store the different columns of your data. Data that does not look like number is coded as factors, i.e. as a categorical variable. If this is not what you have in mind the option `read.table(...,colClasses="character")` might help to store data as character strings.
- Once you have character strings the function `substr` may help you to access parts of these strings.
- The function `table` helps you to get a quick overview over different levels of a variable. Check whether a variable that is supposed to take only values "S" and "H" really takes only these values. If not, use the `substr` function to extract those parts of the data you can use and drop the rest.

  You may want to write those parts of the data that you can use to another file (with `write.table` and read it again.

Help for Stata

- You can read the data with the `insheet` command.
- Once you have character strings the function `substr` may help you to access parts of these strings.
- The functions `tab` helps you to get a quick overview over different levels of a variable. Check whether a variable that is supposed to take only values "S" and "H" really takes only these values. If not, use the `substr` may function to extract those parts of the data you can use and drop the rest.

  You may want to write those parts of the data that you can use to another file (with `outsheet`) and read it again.

  If you get strange errors, issue the command `set trace on` and then execute your code again. You will obtain more information about when the error occurs.

  If you get a strange error that involves an `if` qualifier then have a look at the help page for `if`. Note the difference between == and =.

A.4. Intermediate Problems

The dataset `inter1.csv` contains three variables, the independent variables `x1` and `x2` and the dependent `y`.

1. Use a logistic regression to estimate the relationship.
2. You use this model to predict outcomes. Your boss is not interested in probabilities but needs a clear classification. You assume that a predicted probability larger than 1/2 corresponds to a success and anything less than 1/2 to a failure. Using the model you estimated above, how many events are wrongly classified as failures and how many are wrongly classified as successes?
3. You (rightly) find it inappropriate to use the same data to estimate and to test the prediction quality. Take only the first half of the dataset to estimate the model and use the second half to test the quality of the predictions. How many events are now wrongly classified as failures and how many are wrongly classified as successes
4. What is the p-value of `x2` in the first estimation? Should you keep `x2` in the model equation at all? Estimate the model without `x2`. What is the proportion of correctly predicted values for `y`? Did using the p-value help in selecting a good model?
5. Draw a diagram with `x1` and `x2` at the two axes. Use different colors or different symbols to indicate the different values of `y`. For each of the above estimations, draw a line that separates the predicted successes from the failures. What is the problem in this exercise?

A.5. Tobit

Consider the following linear relationship

$$y = 5.3x - 2.6 + \epsilon$$

Assume that y is only observable when $y \in (20, 25)$. All smaller values are coded as 20, all larger values are coded as 25. The standard deviation of ϵ is 17.

- Take a sample of size 30 and calculate the coefficient of x with the help of an interval regression and with the help of OLS.
- Repeat this process 100 times and draw the cumulative distribution of the coefficients.

A.6. Auctions

The attached dataset `auctions.csv` could be from a hypothetical experiment on bidding behaviour in auctions. You receive a dataset from a hypothetical experiment on bidding behaviour in auctions. In the experiment 2000 bidders are grouped into pairs of 2 bidders. In each group they bid for one item. Each bidder has a valuation for the item, though, the valuation is typically different for different bidders. In each group bidders will be called bidder 1 and bidder 2. Which is called bidder 1 and which is bidder 2 is entirely random. $v1$ and $v2$ are the valuations for bidders 1 and 2. The auction is an English auction, i.e. in each auction the price is increasing until one of the two bidders stops bidding. At this stage the other bidder wins the object. `b` contains the winning bid. wi is the identity of the winning bidder (1 or 2).

We assume that the bid is a linear function of the valuation. Explain why we have a censoring problem here.

Read the data and estimate a linear bidding function.

Several observations for one observation It is typical, but a bit inconvenient that you find information for both bidders in the same record.

Rearrange the data

wide

v1	v2	b	wi
54.35	15.56	83.42	2
75.73	99.85	38.22	1
44.43	8.61	7.50	2
⋮	⋮	⋮	⋮

→

long

id	v	b	wi
1	54.35	83.42	2
2	15.56	83.42	2
1	75.73	38.22	1
2	99.85	38.22	1

Similar problems which require reshaping

`personId`	`income2003`	`income2004`	`income2005`
⋮	⋮	⋮	⋮

↓

`personID`	`year`	`income`
⋮	⋮	⋮

`year`	`patentsFirmA`	`patentsFirmB`	`patentsFirmC`
⋮	⋮	⋮	⋮

↓

`year`	`Firm`	`patents`
⋮	⋮	⋮

Both Stata and R have a powerful `reshape` command that supports you in these conversions.

A.7. Outdoor

Attached you find the dataset `outdoor.csv` which could be from a hypothetical study on choice of favourite outdoor activities. The dataset contains descriptive variables `x1`, `x2`, `x3`, and `x4` for 1000 participants of the study. Each participant spends 20 days in a holiday resort. Participants have been asked on each of the 20 consecutive days for their preferred activity from a set of five activities $\{1,2,3,4,5\}$. The chosen activity for each of the 20 days is coded as the value of the variable `v1`, `v2`, ..., `v20`.

1. Reshape the data such that each record contains only one participant on one day.

2. Use a multinomial logit to estimate the determinants of the favourite activity. Independent variables are `x1`, `x2`, `x3`, and `x4` as well as the time already spent in the holiday resort. In R you would need the commands

   ```
   library(nnet)
   multinom(v ~ x1 + x2 + x3 + x4 + time)
   ```

   In Stata you would say

   ```
   mlogit v x1 x2 x3 x4 time
   ```

   In any case the multinomial logit estimates coefficients $\beta_2, \beta_3, \ldots$ if probabilities of events $y = 1$, or $y = 2$, or $y = 3$,... are given by

$$\begin{aligned}
\Pr(y=1) &= \frac{1}{1+\sum_{k=2}^{m} e^{x'\beta_k}} \\
\Pr(y=2) &= \frac{e^{x'\beta_2}}{1+\sum_{k=2}^{m} e^{x'\beta_k}} \\
\Pr(y=3) &= \frac{e^{x'\beta_3}}{1+\sum_{k=2}^{m} e^{x'\beta_k}} \\
&\vdots
\end{aligned}$$

3. Can you say that an increase in `x3` increases the relative preference of activity 5 compared with activity 4? Explain your answer.

4. What is the impact of an increase in `x3` by one unit on the relative probabilities to choose 5 rather than 4?
5. Can you say that an increase in the length of the stay increases the relative preference of activity 2 compared with activity 1? Explain your answer.