Statistics for Astronomers and Physicsts: the Schechter Luminosity Function

Richard D. Gill

Mathematical Institute, University of Leiden, Netherlands

http://www.math.leidenuniv.nl/~gill

11 September, 2012

Abstract

I simulate a sample of luminosities from a population following the Schechter law, and attempt to recover its two coefficients from the data by curve fitting.

1 Introduction

According to the Schechter luminosity function, the number of stars or galaxies within a luminosity bin of fixed width at luminosity ℓ is proportional to $\ell^\alpha \exp(-\ell/L^*)$. A probabilist or statistician recognises this as an example of the probability density function of the gamma distribution. The gamma distribution with parameters β and λ has probability density $\lambda^\beta x^{\beta-1} \exp(-\lambda x)/\Gamma(\beta)$ on $x > 0$, where $\beta > 0$ and $\lambda > 0$. Thus we should simply make the translation $\beta = \alpha + 1$ and $\lambda = 1/L^*$, in order to rewrite the Schechter luminosity function as a gamma density.

A much cited value of α for field galaxies is $\alpha = -1.25$ corresponding to $\beta = -0.25 < 0$. Thus the corresponding gamma distribution does not actually exist, since $x^{\beta-1} \exp(-\lambda x)$ integrates to $+\infty$! The problem is the singularity at zero. But galaxies of luminosity close to zero can't be observed and possibly they don't even exist. We don't really believe the Schechter law continues to hold at arbitrarily small ℓ, anyway.

In the following simulation experiment I'm going to take the true value of $\alpha = -1.25$ and the true value of $L^* = 1$. It's clear that we need to introduce a lower cut-off. I'll simulate a random sample of field galaxies with luminosities larger than $0.01L^*$, and following the Schechter law. Now, `R` knows all about the gamma distribution but of course only for parameters which do indeed make it a bona fide probability distribution. I'll use the following trick to simulate luminosities from a Schechter distribution with $-1 < \alpha+1 \leq -0$. Suppose I draw a random luminosity from the gamma distribution with $\beta = \alpha + 2$ and $L^* = 1$. I will discard it immediately if $\ell < \epsilon = 0.01$ (my lower cut-off). Next, I draw a uniform random number U between 0 and 1, independently of my luminosity. If $U < \epsilon/\ell$ I

keep the luminosity, otherwise I discard it. Larger luminosities get downweighted exactly in proportion to their value!

This way I'll only keep luminosities which are larger than ϵL^*, and the relative frequency with which I observe luminosities in a bin of small fixed width at luminosity value ℓ will be proportional to $(1/\ell) \times (\ell^{\beta-1} \exp(-\ell/L^*))$, just as we want, since I take $\beta - 2 = \alpha$.

After generating my sample, I will bin the data and plot log counts against bin midpoints. According to Schechter, the number of galaxies in each bin is roughly proportional to $\ell^\alpha \exp(-\ell/L^*)$. Thus the log counts are expected to be close to a constant plus $\alpha \log \ell - \ell/L^*$. At very low values of luminosity, the plot of log counts against luminosity is dominated by the term $\alpha \log \ell$, but at high values the term $-\ell/L^*$ dominates. Back in terms of frequencies, at small luminosities we see the power law ℓ^α, at large luminosities exponential decay $\exp(-\ell/L^*)$ takes over. The literature says that L^* is the luminosity value where the transition occurs between the two behaviours. There is something in that.

The fact that log counts should be close to a constant plus $\alpha \log \ell - \ell/L^*$ suggests that we estimate L^* and α by doing a *multiple* linear regression of log counts as linear function of two predictors: ℓ and $\log \ell$. The coefficient of ℓ gives us $-1/L^*$, the coefficient of $\log \ell$ gives us α.

Of course I will actually do a *weighted* least squares (multiple linear regression) fit in order to get these coefficients.

We could have tried nonlinear regression, fitting the Schechter function directly to the observed counts. Perhaps you'ld like to try that, and compare results. Of course again *weighted* least squares will be called for, different weights!

Please study the code carefully and check that it does what I say. First of all we make some data.

```
> set.seed(20120910)
> eps <- 1/100
> alpha <- -1.25
> beta <- alpha+2
> data <- rgamma(100000,beta)
> data <- data[data>eps & runif(100000) < eps/data]
> N <- length(data); N

[1] 6305

> upper <- round(max(data),1) +0.1
> breaks <- seq(from=0,to=upper,by=0.1)
> nbreaks <- length(breaks)
> LuminosityBinMidpoints<-
+     (breaks[1:(nbreaks-1)]+breaks[2:nbreaks])/2
> Counts <- hist(data,breaks,plot=FALSE)$counts
> Counts
```

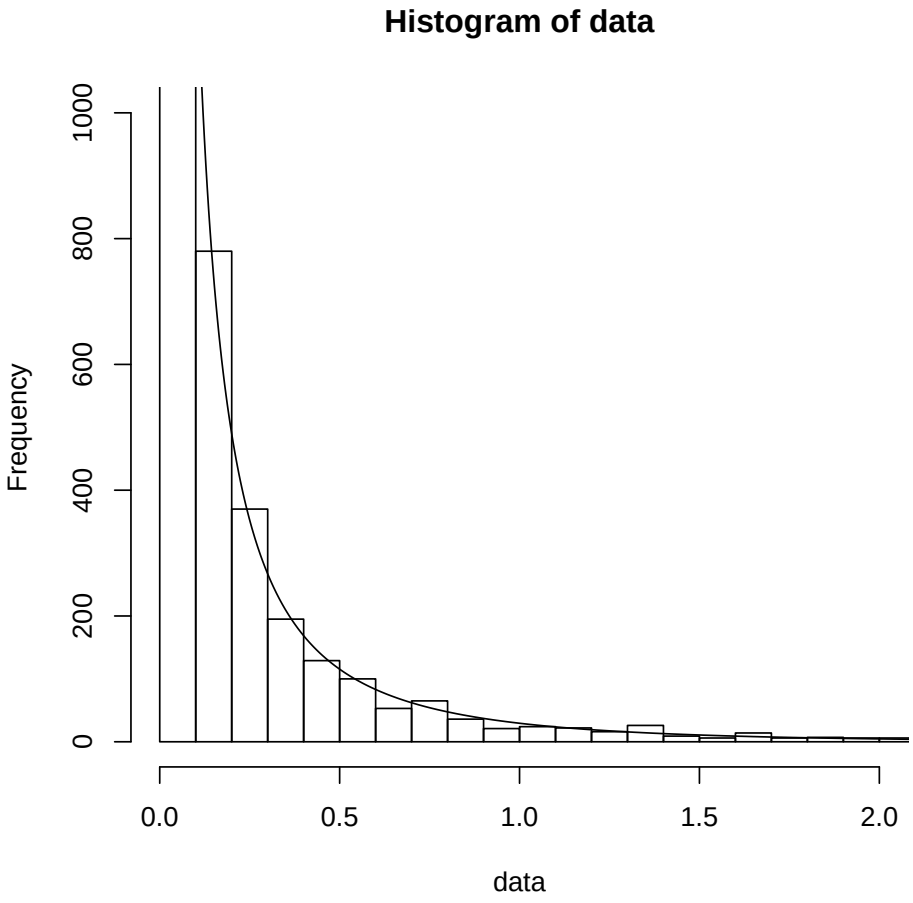
```
 [1] 4387  780  370  195  129  100   53   65   36   21   24   22   16   26    9    6
[20]    6    6    3    5    2    1    2    2    3    1    1    1    0    2    0    2
[39]    0    0    1    0    0    0    0    1
```

```
> LuminosityBinMidpoints
```

```
 [1] 0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95 1.05 1.15 1.25 1.35 1.45 1.55
[20] 1.95 2.05 2.15 2.25 2.35 2.45 2.55 2.65 2.75 2.85 2.95 3.05 3.15 3.25 3.35 3.45
[39] 3.85 3.95 4.05 4.15 4.25 4.35 4.45 4.55
```

The following picture shows that our simulation appears to be doing the right thing. I chose the proportionality constant "80" of the true Schechter function by eye.

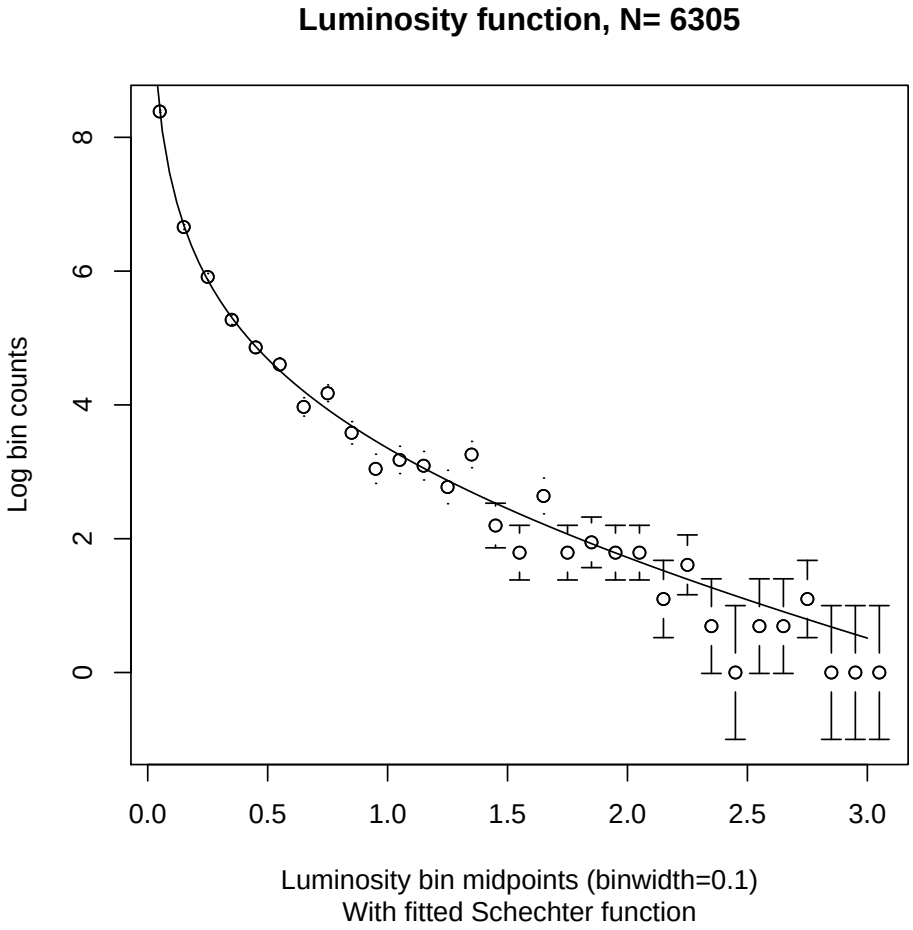
```
> xpts<-seq(from=0,to=4,length=1000)
> hist(data,breaks,ylim=c(0,1000),xlim=c(0,2))
> lines(xpts,80*exp(-xpts)*xpts^alpha)
```



2 Fitting the Schechter function

Now let's get on with the fitting the curve to the data, by weighted least squares after a logarithmic transformation of the y-axis.

```
> library(gplots)
> nbins <- min((1:length(Counts))[Counts==0]) -1
> Counts <- Counts[1:nbins]
> LuminosityBinMidpoints <- LuminosityBinMidpoints[1:nbins]
> LogCounts <- log(Counts)
> plotCI(LuminosityBinMidpoints,LogCounts,uiw=1/sqrt(Counts),
+     main=paste("Luminosity function, N=",N),
+     xlab="Luminosity bin midpoints (binwidth=0.1)",
+     ylab="Log bin counts",
+     sub="With fitted Schechter function")
> lm.data<-data.frame(y=LogCounts,
+     x1=log(LuminosityBinMidpoints),
+     x2=LuminosityBinMidpoints,
+     weights=Counts)
> lm.fit <- lm(y~x1+x2,data=lm.data,weights=lm.data$weights)
> lm.fit$coef

(Intercept)          x1          x2
   3.955781   -1.488366   -0.602138

> xpts <- seq(from=0, to = 3, length=100)
> newdata<-data.frame(x1=log(xpts),x2=xpts)
> lines(xpts,predict.lm(lm.fit,newdata))
> -1/lm.fit$coef["x2"]  # estimate of L*

      x2
1.660749

> lm.fit$coef["x1"]     # estimate of alpha

       x1
-1.488366
```



The fit looks splendid. But the estimates are really bad. Did I do something wrong? What did I do wrong???

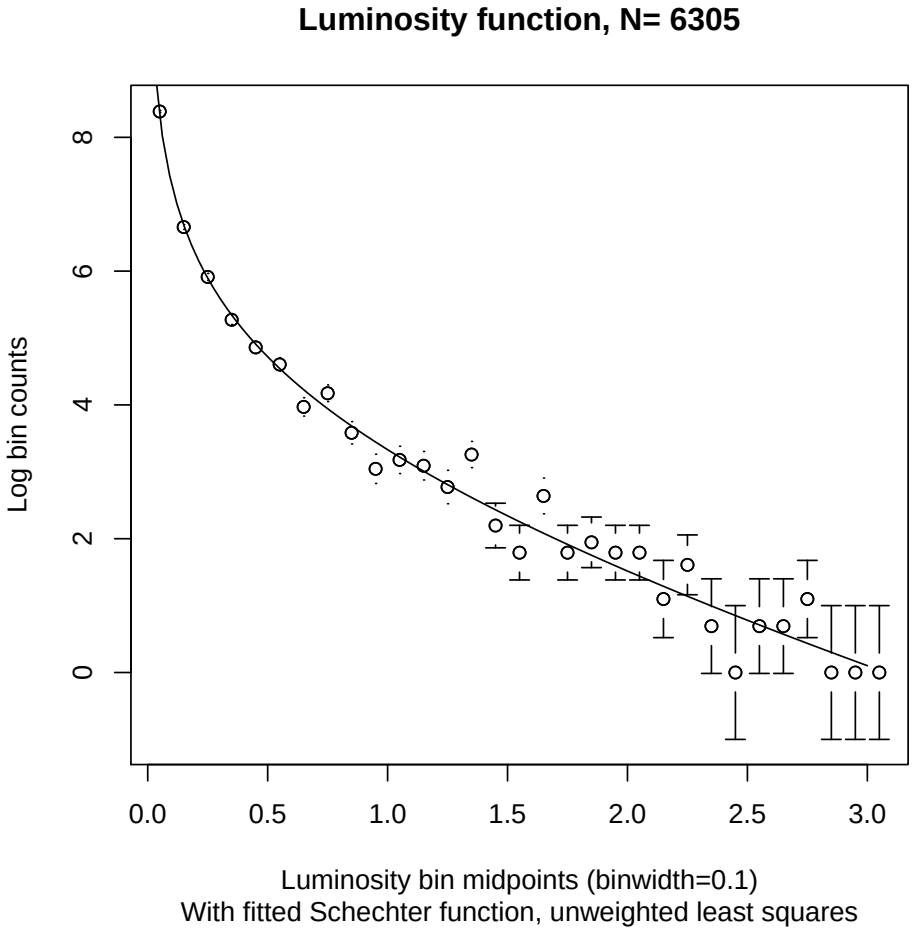
3 Fixing the fit

The data for the regression analysis (which are bin counts!) starts off with a sequence of really huge numbers. Our weighted least squares regression analysis means that these initial counts are given truely enormous weights. But for each of those bins, we are using the Schechter function at the *midpoint* of the bin as surrogate for its *average* over the bin. There is even a lower cut-off which means that the first bin really has different width to the others! I think that all this introduces a huge bias. How to fix this problem?

An ad hoc solution is to omit the first few counts from the data. Dropping the first three produced a decent value for α though L^* was quite far off. Another ad hoc solution is simply to drop the weights (from the least squares fit)! Take a look at this:

```
> plotCI(LuminosityBinMidpoints,LogCounts,uiw=1/sqrt(Counts),
+     main=paste("Luminosity function, N=",N),
```

```
+     xlab="Luminosity bin midpoints (binwidth=0.1)",
+     ylab="Log bin counts",
+     sub="With fitted Schechter function, unweighted least squares")
> lm.data<-data.frame(y=LogCounts,
+     x1=log(LuminosityBinMidpoints),
+     x2=LuminosityBinMidpoints)
> lm.fit <- lm(y~x1+x2,data=lm.data)
> lm.fit$coef

(Intercept)          x1          x2
  4.1826695  -1.3889046  -0.8516252

> xpts <- seq(from=0, to = 3, length=100)
> newdata<-data.frame(x1=log(xpts),x2=xpts)
> lines(xpts,predict.lm(lm.fit,newdata))
> -1/lm.fit$coef["x2"]  # estimate of L*

      x2
1.174225

> lm.fit$coef["x1"]     # estimate of alpha

       x1
-1.388905
```



Not bad!

But *maybe* (but I don't think so) we were just unlucky with our sample. We had better repeat this a few times before drawing any strong conclusions. Be my guest!

Note that in the real world, we do not know the truth. We will easily make beautiful graphics and convince ourselves that we have got a beautiful fit, when in reality we are far off.

4 Fixing the fit, second attempt

It is clear that binning and curve fitting is not the right thing to do when there are such huge differences in counts. *If* we are going to do binning and curve fitting, we had better move to log luminosities. But will the density still be linear in (transformations of) the parameters?

The answer is yes. Linearity is preserved. And this is not a coincidence.

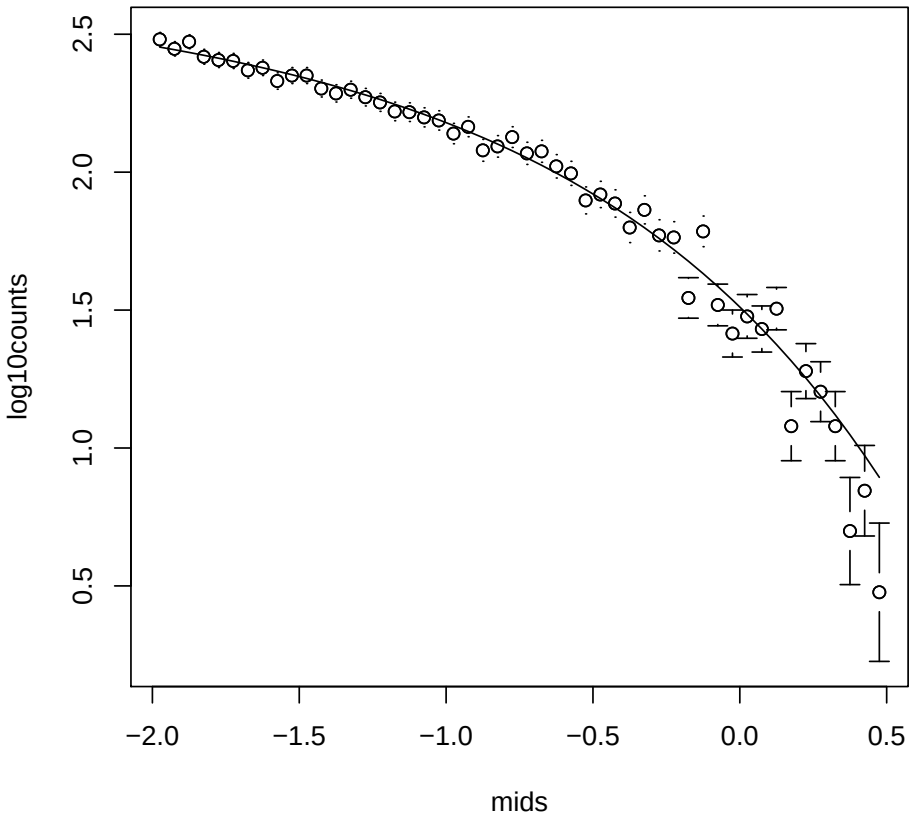
If X has a probability density proportional to $x^\alpha \exp(-x/L^*)$, then it is easy to check that $Y = \log(X)$ has probability density proportional to $e^y \cdot (e^y)^\alpha \exp(-e^y/L^*) = e^{y(\alpha+1)-e^y/L^*}$.

Thus expected log counts in log luminosity bins of equal widths are roughly equal to a constant plus $(\alpha+1)y + (-1/L^*)e^y$. We just multiply the previous density by the absolute value of the Jacobian of the inverse transformation, which is a function of the data only. If the log density of the data can be written as something linear in transformed parameters, the same is true after any transformation of the data.

Since I had a lower cut-off of 0.01 I'll take logarithms to base 10 instead of natural logarithms. Letting $Z = Y/\log(10)$ the density of Z becomes proportional to $10^{z(\alpha+1)-10^z/L^*}$, and the expected log (base 10) counts in log-base-10 luminosity bins of equal widths are roughly equal to a constant plus $z(\alpha+1) - 10^z/L^*$.

Thus we should perform linear regression on logarithmic (base 10) counts with regressors z and 10^z, whose coefficients are estimators of $\alpha+1$ and $-1/L^*$ respectively. Take a look at the results:

```
> logdata <- log(data)
> log10data<-logdata/log(10)
> counts<-hist(log10data,breaks=seq(from=-2,to=1,by=0.05),plot=FALSE)$counts
> mids<-hist(log10data,breaks=seq(from=-2,to=1,by=0.05),plot=FALSE)$mids
> counts <- counts[1:50]
> mids <- mids[1:50]
> logcounts<-log(counts)
> log10counts <- logcounts/log(10)
> coefs<-lm(log10counts~mids+exp(mids),weights=counts)$coef
> plotCI(mids,log10counts,uiw=1/(sqrt(counts)*log(10)))
> lines(mids,coefs[1]+coefs[2]*mids+coefs[3]*exp(mids))
> coefs

(Intercept)        mids   exp(mids)
 2.48694238 -0.05154688 -0.97563440

> alpha

[1] -1.25
```



Not too bad, I suppose. But not too good either. There must be better ways to do this...

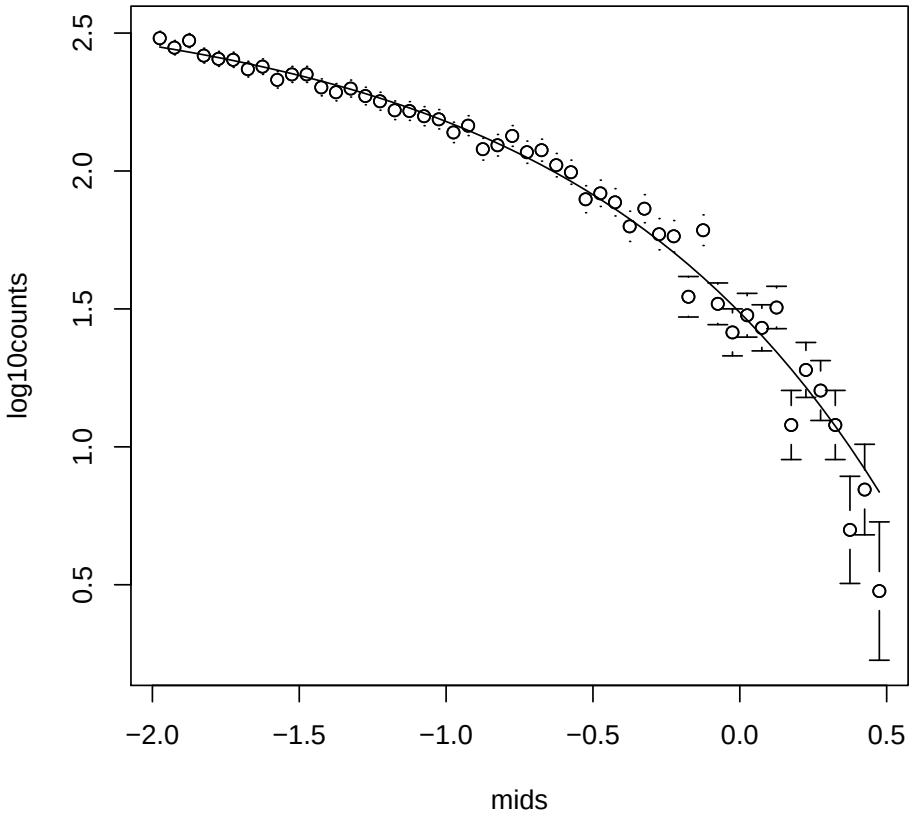
5 Fixing the fit, third attempt

Clearly we should prefer weighted least squares to least squares, but perhaps so far our choice of weights was unfortunate. The counts themselves are subject to random variation. If a count is untypically large then we are going to give that same observation larger weight, and vice versa. Thus we probably are biasing our estimates, making the fitted curve fit the *coincidentally* relatively larger values better than the *coincidentally* relatively smaller values, to a larger degree than we ought.

A solution is to take the weights for the weighted least squares regression from the estimated counts from an earlier, e.g., unweighted, fit. Take a look at this.

```
> logdata <- log(data)
> log10data<-logdata/log(10)
> counts<-hist(log10data,breaks=seq(from=-2,to=1,by=0.05),plot=FALSE)$counts
```

```
> mids<-hist(log10data,breaks=seq(from=-2,to=1,by=0.05),plot=FALSE)$mids
> counts <- counts[1:50]
> mids <- mids[1:50]
> logcounts<-log(counts)
> log10counts <- logcounts/log(10)
> fits <- lm(log10counts~mids+exp(mids))$fitted
> coefs <- lm(log10counts~mids+exp(mids), weights=10^fits)$coef
> plotCI(mids,log10counts,uiw=1/(sqrt(counts)*log(10)))
> lines(mids,coefs[1]+coefs[2]*mids+coefs[3]*exp(mids))
> coefs

(Intercept)        mids   exp(mids)
 2.53356111 -0.03090173 -1.04646084

> alpha

[1] -1.25
```



6 Literature

Chapter 4 of Feigelson and Babu.

Wikipedia article "Luminosity function (astronomy)".

Note: In the real world, we certainly don't just go and get a random sample of luminosities of galaxies. We can't see very faint objects at all. Moreover, something which seems faint to us might just be a very bright object, very far away. Varying distances means that to get actual luminosities we have to convert observed magnitudes according to the distance of the object concerned. We have to estimate this distance from its redshift. For this we need to take a lot of assumptions from cosmology and astrophysics.

The way objects are observed means that there certainly are cut-offs and selection biases. We have to be take them into account in the statistical analysis, and in order to do this we have to model them.