Statistiek voor astronomie en natuurkunde: the Salpeter IMF

Richard D. Gill
Mathematical Institute, University of Leiden, Netherlands
http://www.math.leidenuniv.nl/~gill

11 September, 2012

Abstract

I simulate a sample of stellar initial masses from a population following the Salpeter law, and attempt to recover the exponent of the Salpeter law from the data by curve fitting.

1 Create sample

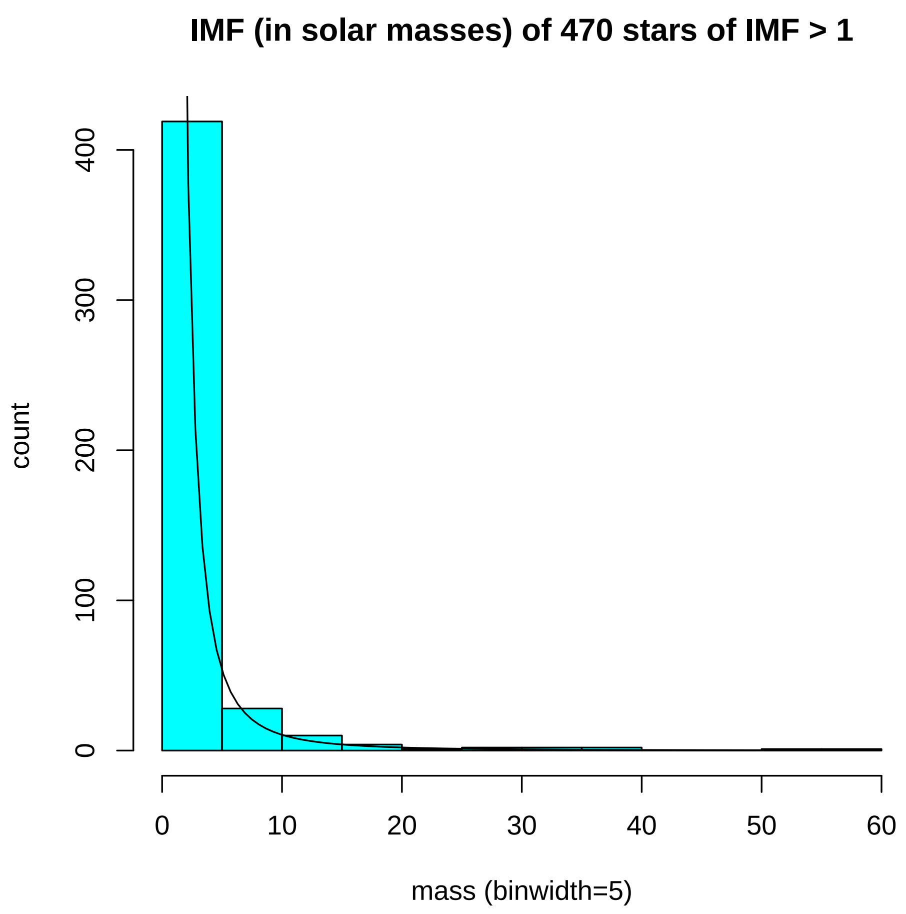
Imagine taking a random sample of stars which are just entering the main sequence (i.e., beginning hydrogen fusion). The masses of such stars are called *initial masses*, and the probability density function of the probability distribution of the initial masses of a population of such stars is called the IMF or initial mass function[1]. Let's measure mass m in multiples of 1 solar mass. Edwin Salpeter discovered in 1955 that, at least for the larger stars ($m \geq 1$), the number of stars of mass in a given range Δm appeared to decrease according to a power law with exponent $\alpha \approx 2.35$. In the language of probability theory this means that, for stars of mass $m \geq 1$, the initial mass M of a randomly sampled star is a random variable with probability density function $m^{-\alpha}$. The following `R` code generates a random sample of initial masses from this probability distribution, with the true value of α precisely equal to 2.35. Actually I took the sample size to be random (Poisson distributed with expected value 500), and discarded any masses larger than 100.

I go on to draw a histogram of the data and superimpose on this, the expected number of counts according to Salpeter.

Please study the code carefully and figure out why it actually does do what I say it does.

[1]Actually, astronomers typically normalize their probability densities to have total measure "mean number of objects per unit volume of space".

```
> set.seed(20120911)
>     # You will get identical results to me if you keep the random seed the same!
> alpha <- 2.35
> N <- rpois(1,500)
> N

[1] 471

> data <- 1/(runif(N)^(1/(alpha-1))); max(data)

[1] 553.9706

> data<-data[data<100]; max(data)

[1] 59.41388

> N <- length(data); N

[1] 470

> library(MASS)
> h <- 5
> truehist(data,h=h,x0=0,prob=FALSE,
+     main=paste("IMF (in solar masses) of",N,"stars of IMF > 1"),
+     xlab="mass (binwidth=5)",ylab="count")
> xpts<-seq(from=1,to=max(data),length=100)
> lines(xpts,N*h*xpts^(-alpha))
```

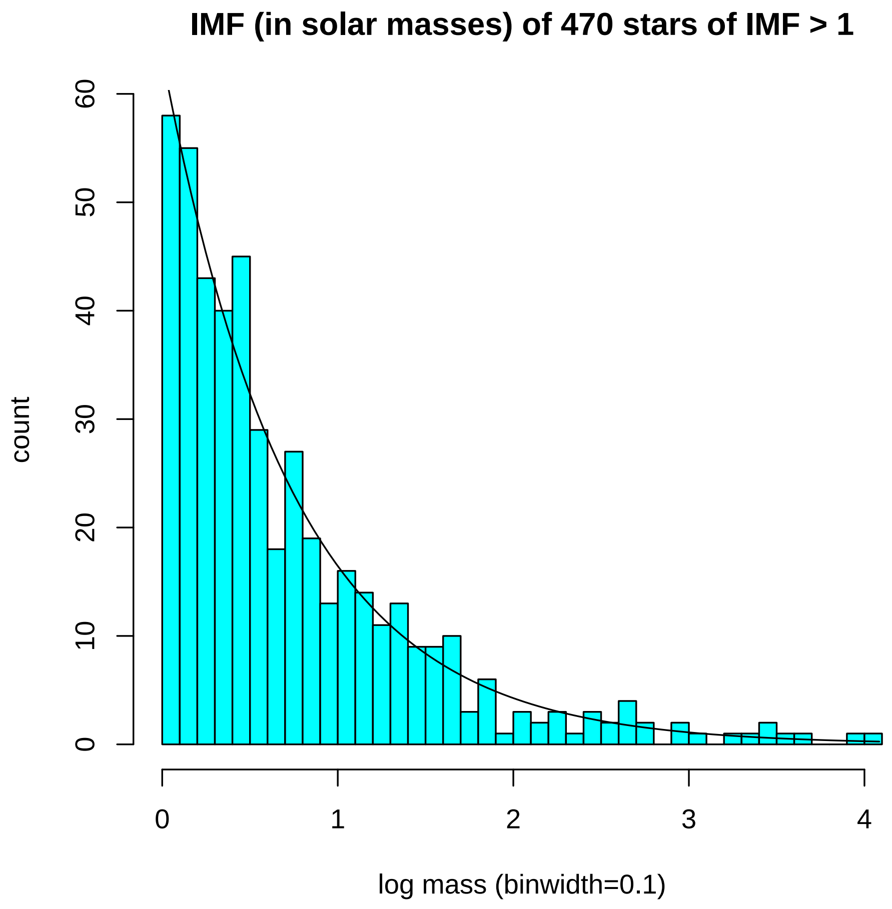


2 Logarithmic conversion

The last picture was not very informative, precisely because of the extremely rapid decrease in the frequency of stars of larger and larger initial mass. Let's redraw the picture with a logarithmic scale on the x-axis. Again, I've superimposed the theoretical curve according to Salpeter. Can you explain why it is what I say it is?

```
> logdata <- log(data)
> max(logdata)

[1] 4.084528

> min(logdata)

[1] 0.0004081704
```

```
> h <- 0.1
> truehist(logdata,h=h,x0=0,prob=FALSE,
+     main=paste("IMF (in solar masses) of",N,"stars of IMF > 1"),
+     xlab="log mass (binwidth=0.1)",ylab="count")
> xpts<-seq(from=0,to=max(logdata),length=100)
> lines(xpts,N*h*(alpha-1)*exp(-(alpha-1)*xpts))
>     # Note: alpha-1; twice!  Why??
```



3 Bin and count

We saw that logarithmic conversion on the x-axis changes a power law into an exponential; the coefficient in the exponent is one one less than the power in the power law. This suggests taking logarithms of counts of binned log masses, plotting them against log masses, and fitting a straight line. But if we don't watch out, some counts will be zero, and logarithms of zero are embarrassing. The following code computes bin midpoints and bin counts for log mass, with binwidth 0.25; all counts from the first zero are discarded. I use the standard

R function hist to do the counting. Normally we use it to see a histogram but in this case I suppress the printing and just use it for its useful side effect.

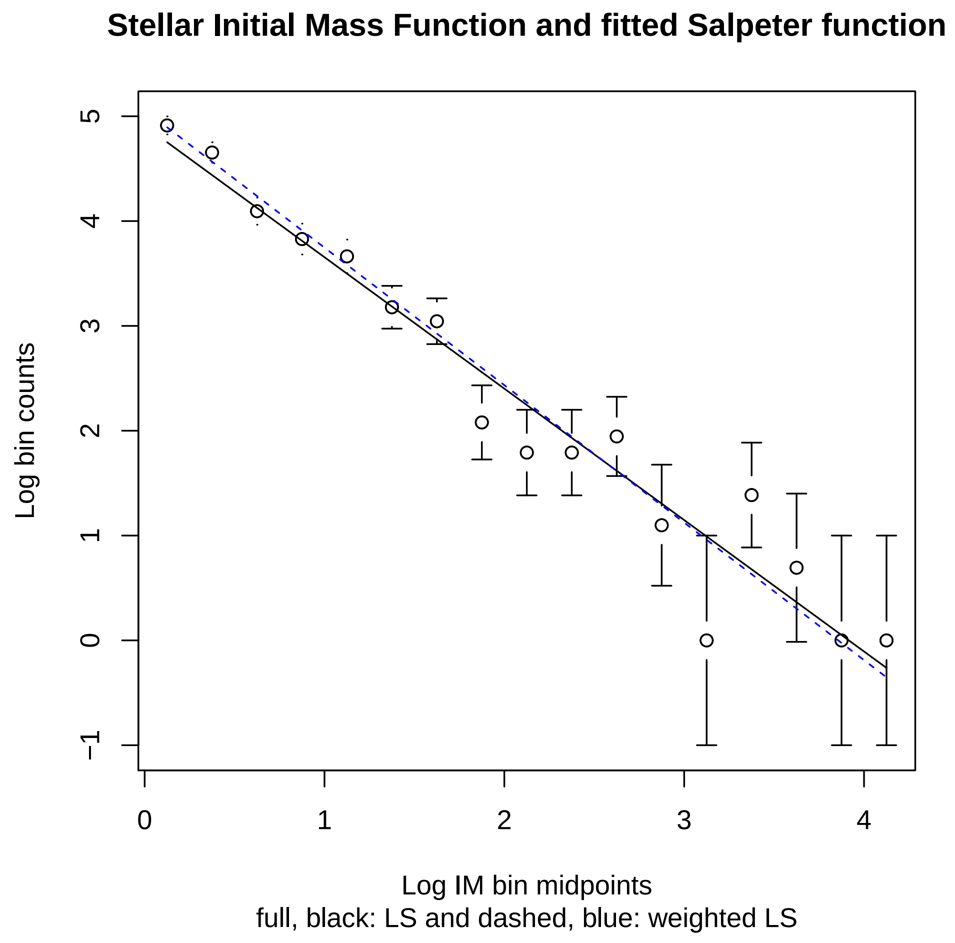
```
> breaks <- seq(from=0,to=7,by=0.25)
> nbins <- length(breaks)-1
> LogMassBinMidpoints <-(breaks[1:nbins]+breaks[(1:nbins)+1])/2
> Counts <- hist(logdata,breaks,plot=FALSE)$counts
> Counts

 [1] 136 105  60  46  39  24  21   8   6   6   7   3   1   4   2
[16]   1   1   0   0   0   0   0   0   0   0   0   0   0

> firstzero <- min((1:(nbins+1))[c(Counts,0)==0])
> Counts <-Counts[1:(firstzero-1)]
> LogMassBinMidpoints <- LogMassBinMidpoints[1:(firstzero-1)]
> LogMassBinMidpoints

 [1] 0.125 0.375 0.625 0.875 1.125 1.375 1.625 1.875 2.125 2.375
[11] 2.625 2.875 3.125 3.375 3.625 3.875 4.125

> Counts

 [1] 136 105  60  46  39  24  21   8   6   6   7   3   1   4   2
[16]   1   1

> LogCounts <- log(Counts)
```

4 Fit straight line

OK, now we are ready to fit a straight line through these log counts of binned log masses. I will do this in two ways. First of all by ordinary least squares, i.e., I determine the straight line which minimizes the sum of squares of the vertical distances of the points from the line. Now the statistical variation in the counts depends on their mean expected value. In fact we expect Poisson statistics, thus variances equal to means, thus standard deviations equal to square root of mean. When we take logarithms (derivative of $\log(x)$ is $1/x$), propagation of error tells us that standard deviations of log counts are (approximately) equal to 1 divided by the square root of the corresponding mean.

Therefore we should get a better fit by weighting the squared deviations with the inverses of the actually observed counts. Look at the two fitted lines, and see if you agree – by looking at the picture – that the weighted least squares fit does a better job.

```
> fit.lm <- lm(LogCounts~LogMassBinMidpoints)
> fit.weighted.lm <- lm(LogCounts~LogMassBinMidpoints,weights=Counts)
> library(gplots)
> plotCI(LogMassBinMidpoints,LogCounts,uiw=1/sqrt(Counts),xlab="Log IM bin midpoints
+     main="Stellar Initial Mass Function and fitted Salpeter function",
+     sub="full, black: LS and dashed, blue: weighted LS")
> lm.data<-data.frame(y=LogCounts,x=LogMassBinMidpoints,weights=Counts)
> lines(LogMassBinMidpoints,predict.lm(fit.lm,lm.data))
> lines(LogMassBinMidpoints,predict.lm(fit.weighted.lm,lm.data),lty=2,col="blue")
> fit.lm$coef

        (Intercept) LogMassBinMidpoints
           4.908933           -1.253680

> fit.weighted.lm$coef

        (Intercept) LogMassBinMidpoints
           5.058487           -1.311709
```



5 Conclusions

And what about the results? Ordinary least squares gave an estimate of α of 2.25 (remember, we have to increase the slope by one), weighted least squares gave 2.31.

Since the truth was $\alpha = 2.35$ this seems to favour weighted least squares. But what if we repeat the experiment, with a new random sample from *exactly the same population*?

Well: just go ahead and do that! Do it 100 times (or 1000 times if you prefer) and take a look at the *sampling distribution* of the two estimators of α. What do you see?

Here is my prediction: weighted least squares is substantially better than ordinary least squares. Both estimators are systematically biased downwards, ordinary least squares more than weighted least squares; and the variance of ordinary least squares is larger than that of weighted least squares.

This naive curve fitting method suffers from one obvious defect: the arbitrariness of the choice of bin-width. Both too large and too small will lead to disastrous results. Is there an "optimal" choice and can that be made in an objective fashion?

Anyway, are there perhaps better estimators altogether, based on quite different principles?

6 Literature

Chapter 4 of Feigelson and Babu.

Wikipedia article "Initial mass function".

Note: In the real world, we usually don't just go and get a random sample of initial masses of new born stars. We typically take a sample of all (any) stars in some region, and use astrophysical theory of stellar evolution to predict their initial masses from various observable features of those stars as we see them now. We'll also have to take account of various selection biases, which might well mean that the initial masses of the stars we observe actually come from a different distribution from that of all stars!