Part 4 Section 3: Bayesian Inference – Number Density Functions

Danail Obreschkow

2025-08-29

Prerequisites

If executing for the first time, you need to install the `dftools` package via github:

```
install.packages('devtools')
library(devtools)
install_github('obreschkow/dftools')
```

Load libraries and set random seed for this chapter:

```
library(magicaxis)
library(pracma)
library(dftools)
library(cooltools)
library(foreach)
set.seed(2)
```

Definition of number density functions

A *number density function* $\lambda(x)$ is defined in such a way, that the *expected* number of events of value x, per unit dx, equals

$$\langle dN\rangle \equiv \lambda(x)dx.$$

The integral of this function then corresponds to the *expected* total number of events,

$$\langle N\rangle = \int dN = \int \lambda(x)dx.$$

NDFs $\lambda(x)$ are closely related to PDFs $\rho(x)$; the sole difference being the normalization by $\langle N\rangle$:

$$\lambda(x) = \langle N\rangle\rho(x).$$

In the context of statistical inference, the function $\lambda(x)$ is unknown; and the general problem is to determine the parameter(s) θ in a parameterized family of functions $\lambda(x|\theta)$. At first sight, this problem seems very similar to the earlier problem (part 4, section 1) of inferring the parameters θ of a parameterized PDF $\rho(x|\theta)$. However, the *a priori* unknown normalization N of the NDF acts as an additional free parameter and leads to an extra term in the likelihood function.

Before getting to the likelihood θ associated with the NDF $\lambda(x|\theta)$, we will take a relatively large detour through the example of galaxy mass functions. This example is meant to emphasize the power of a proper Bayesian approach as opposed to a more ad hoc approach.

Example: Galaxy mass functions

Definition

Faint galaxies are far more common than luminous, massive ones. In a fixed cosmic volume, the number of galaxies per unit mass approximately declines as a power law up to a cut-off scale, beyond which the number density declines almost exponentially. Formally, the number density of galaxies per unit mass is called the mass function (MF). Explicitly, the MF $\phi(M)$ is defined, such that in a cosmic volume V, the expected number of galaxies in a mass interval $[M, M + dM]$ is

$$dN = \phi(M)\,V\,dM. \tag{1}$$

In appreciation of the power law behaviour, it is often convenient to define the MF as the number density of galaxies per unit of logarithmic mass $x = \log_{10}(M/\mathrm{M}_\odot)$; i.e. in a cosmic volume V, the expected number of galaxies in an interval $[x, x + dx]$ is

$$dN = \psi(x)\,V\,dx. \tag{2}$$

The relation between $\phi(M)$ and $\psi(x)$ is thus

$$\psi(x) = \ln(10)M\phi(M). \tag{3}$$

For simplicity, this chapter will concentrate on the galaxy MF, however the concepts, formalisms and implementations are readily transferable to other objects, such as stars, star clusters, galaxy clusters and dark haloes. Likewise, the galaxy mass M is not further specified to emphasize its applicability to any choice, e.g. stellar mass, gas mass, or dynamical mass. In fact, M can be substituted for any other observable, including luminosity, absolute magnitude and even quantities from other fields of science, including multi-dimensional observables. Especially in older astronomy literature, luminosity functions (LFs) are often expressed as the number density per unit of absolute magnitude. Since there are five absolute magnitudes for two orders of magnitude in x, this introduces a rescaling of $\psi(x)$ by $2/5 = 0.4$.

Schechter function

Many analytical parametric functions have been proposed to fit observed MFs. They can all be writen as $\phi(M|\theta)$, where θ is a vector of model parameters. Perhaps the most commonly used example is the Schechter function (Schechter et al. 1976),

$$\phi_\mathrm{S}(M|\theta) = \frac{\phi_*}{M_*}\mu^\alpha e^{-\mu} \quad \Leftrightarrow \quad \psi_\mathrm{S}(x|\theta) = \ln(10)\phi_*\mu^{\alpha+1}e^{-\mu}, \tag{4}$$

where $\mu = M/M_* = 10^x\mathrm{M}_\odot/M_*$. This model depends on three parameters: the density normalisation ϕ_*, the break mass M_* and the power law slope α. The first two, are often given as $\log_{10}$-values. The non-dimensional Schechter function parameters are then

$$\theta = \Big(\log_{10}(\phi_*/\mathrm{Mpc}^{-3}), \log_{10}(\mathrm{M}_*/\mathrm{M}_\odot), \alpha\Big). \tag{5}$$

As an example, let us consider a Schechter function of exactly known true parameters

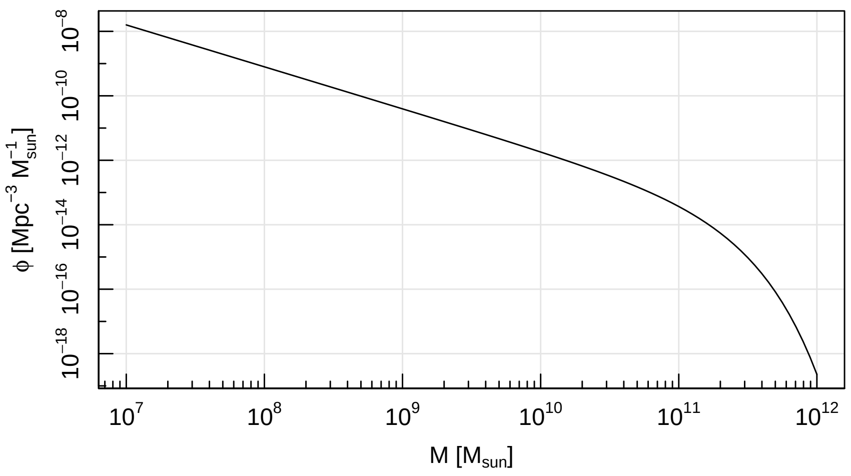
$$\bar{\theta} = (-2, 11, -1.3), \tag{6}$$

which is roughly consistent with the rounded parameters of the observed galaxy MF for baryonic matter (stars and cold gas) in the local universe.

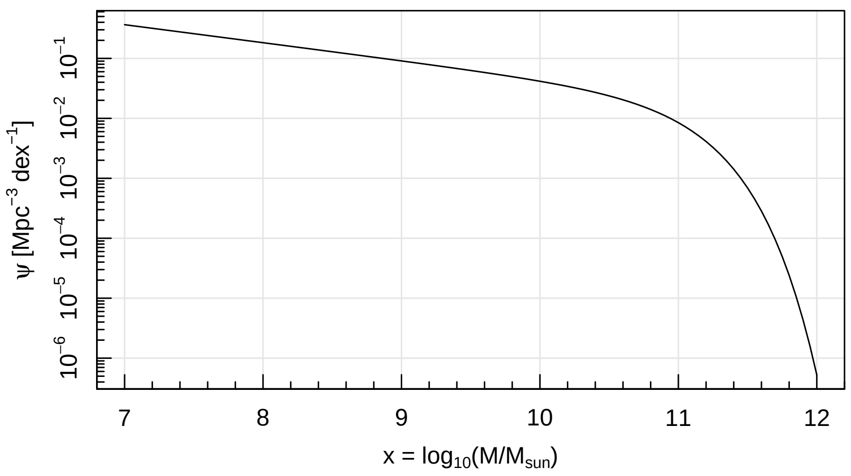
Let us plot $\phi_\mathrm{S}(M)$ and $\psi_\mathrm{S}(x)$ in **R**:

```
# linear mass function
phi.schechter = function(M,p) {
  # p[1] = log10(phi*/Mpc^-3), p[2] = log10(M*/Msun), p[3] = alpha
  mu = M/10^p[2]
  return(10^(p[1]-p[2])*mu^p[3]*exp(-mu)) # [Msun^-3 Msun^-1]
}
```

```r
# log10-based mass function
psi.schechter = function(x,p) {
  mu = 10^(x-p[2])
  return(log(10)*10^p[1]*mu^(p[3]+1)*exp(-mu)) # [Msun^-3 dex^-1]
}

p.true = c(-2,11,-1.3)

magcurve(phi.schechter(x,p.true),xlim=10^c(7,12),log='xy',
        xlab=expression('M [M' ['sun']*']'),
        ylab=expression(phi~'[Mpc'^'-3'~'M' ['sun']^'-1'*']'))
```



```r
magcurve(psi.schechter(x,p.true),xlim=c(7,12),log='y',
        xlab=expression('x = log' [10]*'(M/M' ['sun']*')'),
        ylab=expression(psi~'[Mpc'^'-3'~'dex'^'-1'*']'))
```

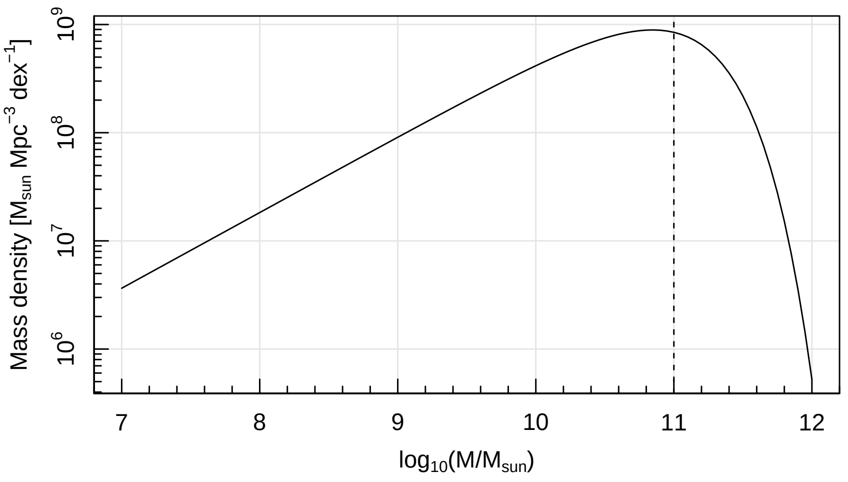


In the second plot, the term dex^{-1} in the units of the y-axis is a dimensionless term, which could be dropped. We have only included it as a reminder that ψ is a NDF is per unit of the log-mass x.

We see that $\phi_S(M)$ and $\psi_S(x)$ have different slopes due to the factor M that relates them.

If we are interested in the mass density rather than the number density of galaxies at a certain mass, the MF must be multiplied by M:

```r
magcurve(psi.schechter(x,p.true)*10^x,xlim=c(7,12),log='y',
        xlab=expression('log' [10]*'(M/M' ['sun']*')'),
        ylab=expression('Mass density [M' ['sun']~'Mpc'^'-3'~'dex'^'-1'*']'))
abline(v=p.true[2],lty=2)
```



This plot reveals that galaxies of masses close to M_* (dashed vertical line) are the ones that contribute most to the total mass density. In fact, the function peaks exactly at $M = M_*$ if $\alpha = -1$.

The Schechter function exhibits interesting analytical properties. It can be integrated analytically to get the total number density n of the galaxies per cosmic volume:

$$n = \int_0^\infty \phi(M)dM = \phi_* \int_0^\infty \mu^\alpha e^{-\mu} d\mu = \phi_* \Gamma(\alpha + 1), \tag{7}$$

where Γ is the gamma function, implemented as **gamma(x)** in **R**. Likewise, the total mass density ρ per unit volume is

$$\rho = \int_0^\infty \phi(M)MdM = \phi_* M_* \int_0^\infty \mu^{\alpha+1} e^{-\mu} d\mu = \phi_* M_* \Gamma(\alpha + 2). \tag{8}$$

From these expressions it follows that the total number density only converges if $\alpha > -1$ and the total mass density only converges if $\alpha > -2$. In our example only the total mass density is finite and equal to:

```
rho = 10^p.true[1]*10^p.true[2]*gamma(p.true[3]+2)
sprintf('Total mass density = %.3e Msun/Mpc^3',rho)
```

```
## [1] "Total mass density = 1.298e+09 Msun/Mpc^3"
```

Let us verify this solution using a numerical evaluation of the integral $\int_0^\infty \phi(M)MdM$:

```
integrand = function(M) phi.schechter(M,p.true)*M
rho = integrate(integrand,0,1e15)$value # 1e15 is a sufficient upper bound
sprintf('Total mass density = %.3e Msun/Mpc^3',rho)
```

```
## [1] "Total mass density = 1.298e+09 Msun/Mpc^3"
```

This solution is indeed identical to the gamma function solution.

The Schechter function is the best-known MF model that captures the truncated power law behaviour. We will use the Schechter function for most illustrations, but the formalism and implementations remain applicable to any MF model, including quasi non-parametric (‘stepwise’) models, discretised into a custom number of bins or vertices.

Fitting galaxy mass functions

Inference problem

Galaxy MF parameters θ can be conveniently used to compare observations to theoretical predictions with the aim to constrain models of galaxy evolution. Thus, both observations and simulations are frequently faced with the task to determine θ based on a finite sample of galaxies. Especially in the case of galaxy surveys, this inference problem also has to deal with the fact that not all galaxies are equally represented in the sample.

Mathematically, the problem can be stated as follows: In a galaxy survey, we detect N galaxies $i = 1, ..., N$ with masses $M_i = 10^{x_i} \mathrm{M}_\odot$. For the time being, we assume that the mass measurements are exact, but an extended treatment of mass uncertainties can be found in Obreschkow et al. (2018, see https://arxiv.org/abs/1712.00149). We assume that galaxies of log-mass x can be detected in a volume $V(x)$. This volume can be thought of as a sharp detection limit: a galaxy of log-mass x is detected if and only if it lies inside this volume. In the presence of more complicated selection functions, $V(x)$ will have a slightly different meaning, but its use will remain the same. Thus, we normally refer to $V(x)$ as the *effective volume*.

Objective: Infer the parameters θ of a MF, given N data points x_i and an effective volume $V(x)$.

Mock data

Before discussing the inference problem, let us first investigate what the data $\{x_i\}$ normally look like. To do so, we model a galaxy survey that is purely sensitivity limited. For a constant mass-to-light ratio and a non-evolving, static, flat universe this implies that a galaxy of mass M is detectable out to a maximum distance proportional to $M^{1/2}$. Hence the effective volume scales as $V \propto M^{3/2} \propto 10^{1.5x}$. As an example, we here choose a proportionality constant, such that

$$V(x) = 10^{1.5(x-8)}\ \text{Mpc}^3 \tag{9}$$

Given $V(x)$ and $\psi(x|\theta)$, the *expected number of detected galaxies per unit log-mass x* is

$$\lambda(x|\theta) = V(x)\psi(x|\theta). \tag{10}$$

The total expected number of detected galaxies thus becomes $\langle N\rangle = \int \lambda(x|\bar{\theta})dx$.

Let us now assume that the galaxy population is perfectly described by a Schechter function of true parameters $\bar{\theta}$. In this case $\langle N\rangle$ can be computed as

```
V = function(x) 10^(1.5*(x-8)) # [Mpc^3]
lambda = function(x) V(x)*psi.schechter(x,p.true)
N.expected = integrate(lambda,7,14)$value
sprintf('Expected number of galaxies = %.2f',N.expected)
```

```
## [1] "Expected number of galaxies = 290.35"
```

Note that we used 7 and 14 as integration boundaries instead of $\pm\infty$ to simplify the numerical integration. This is justified since virtually all detectable objects fall in this smaller range.

To produce a mock survey, we first fix the *actual* number of galaxies N, which can generally differ from $\langle N\rangle$ due to shot noise. Assuming that different galaxies are uncorrelated, the PDF for N, is a Poisson distribution of mean $\langle N\rangle$. We pick N using:

```
N = rpois(1,N.expected)
sprintf('Actual number of galaxies = %d',N)
```

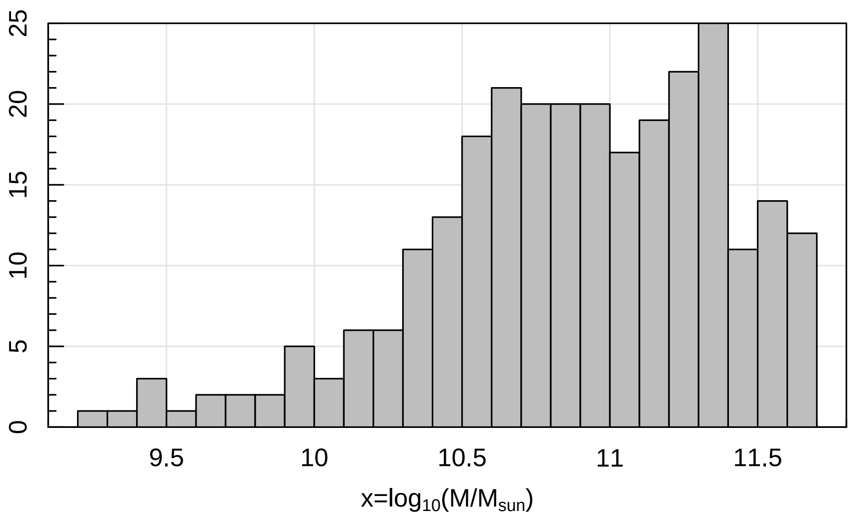
```
## [1] "Actual number of galaxies = 275"
```

To draw N random galaxy masses from the non-normalised distribution $\lambda(x|\bar{\theta})$, we must first compute the normalised PDF, then evaluate the corresponding CDF, then invert the CDF to form the quantile function and finally feed with this quantile function with uniform random numbers. This corresponds to the family of functions **d/p/q/r**, such as **dnorm/pnorm/qnorm/rnorm** (see part 3 for examples). In **R**, this family of functions can be generated from a non-normalised distribution **fun(x)** using the routine **dpqr** of the **cooltools** package.

Here, we only need the random number generator, which we call `rsurvey`. Using this function, we then draw and plot N log-masses x_i, stored in the vector `data.x`:

```
rsurvey = dpqr(lambda,7,14)$r
data.x = rsurvey(N)
maghist(data.x,30,xlab=expression('x=log' [10]*'(M/M' ['sun']*')'),col='grey',yaxs='i')
```

```
## Summary of used sample:

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   9.249  10.568  10.912  10.872  11.267  11.694
## Pop Std Dev: 0.49753

## MAD: 0.51805

## Half 16-84 Quan (1s): 0.47394

## Half 02-98 Quan (2s): 0.97961

## Using 275 out of 275
```



Basic 1/V-method with bins

The 1/V-method (also known as 1/V$_{\text{max}}$-method or 1/V$_{\text{eff}}$-method) is a simple and commonly used, but inaccurate way of recovering a MF from a data sample. Its standard implementation discretises the x-axis (log-mass axis) into discrete bins j. These bins are normally equally spaced with a fixed width Δx and mid-points x_j. For each bin j, we count the number of galaxies n_j, whose mass is contained in the interval $(x_j - \Delta x/2, x_j + \Delta x/2]$. From these counts, the MF-value in each bin can be estimated as

$$\psi_j = \frac{n_j}{V_j \Delta x}, \tag{11}$$

where V_j are the effective volumes of the bins. Often they are approximated as $V_j = V(x_j)$, but smarter methods exist. (Mathematically the best solution is to set V_j equal to the *harmonic* mean of the volumes $V(x_i)$ of the galaxies that go into the j-th bin, but we won't bother doing this here.)

Uncertainties on ψ_j can be estimated as

$$\sigma_j = \frac{\sqrt{n_j}}{V_j \Delta x}. \tag{12}$$

We can then fit a MF model, such as a Schechter function, using a χ^2-minimisation, where

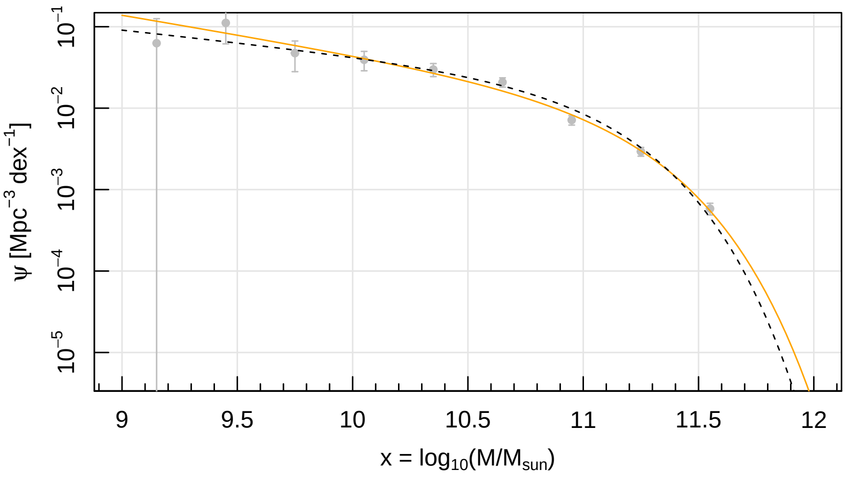
$$\chi^2(\theta) = \sum_j \frac{(\psi_j - \psi(x_j|\theta))^2}{\sigma_j^2}. \tag{13}$$

A major caveat are empty bins, where $n_j = 0$ and hence $\sigma_j = 0$, making the χ^2 ill-defined. We therefore simply reject empty bins from the sum in the χ^2. Clearly, this is an odd procedure, which indicates the limits of this non-Bayesian approach.

Let us illustrate the $1/V$-method at the example of the previously generated data stored in the vector `data.x`.

```
# make bins
n.bins = 10 # user-selected number of bins
x.min = 9
x.max = 12
dx = (x.max-x.min)/n.bins
```

```
x.mid = seq(x.min+dx/2,by=dx,length=n.bins) # vector of bin mid-points

# count the number of galaxies in each bin
ix = ceiling((data.x-x.min)/(x.max-x.min)*n.bins)
counts = foreach(i=1:n.bins,.combine='c') %do% sum(ix==i)

# compute the MF value and uncertainties in each bin
psi.obs = counts/V(x.mid)/dx
sigma = sqrt(counts)/V(x.mid)/dx

# fit Schechter function parameters
ok = counts>0
chi2 = function(p) sum((psi.obs[ok]-psi.schechter(x.mid[ok],p))^2/sigma[ok]^2)
p.best = optim(p.true,chi2)$par

# plot data, fit and input model
magplot(x.mid[ok],psi.obs[ok],pch=20,col='grey',log='y',xlim=c(9,12),ylim=c(5e-6,0.1),
        xlab=expression('x = log' [10]*'(M/M' ['sun']*')'),
        ylab=expression(psi~'[Mpc'^'-3'~'dex'^'-1'*']'))
magerr(x.mid[ok],psi.obs[ok],ylo=sigma[ok],col='grey')
curve(psi.schechter(x,p.best),col='orange',add=T)
curve(psi.schechter(x,p.true),lty=2,add=T)
```

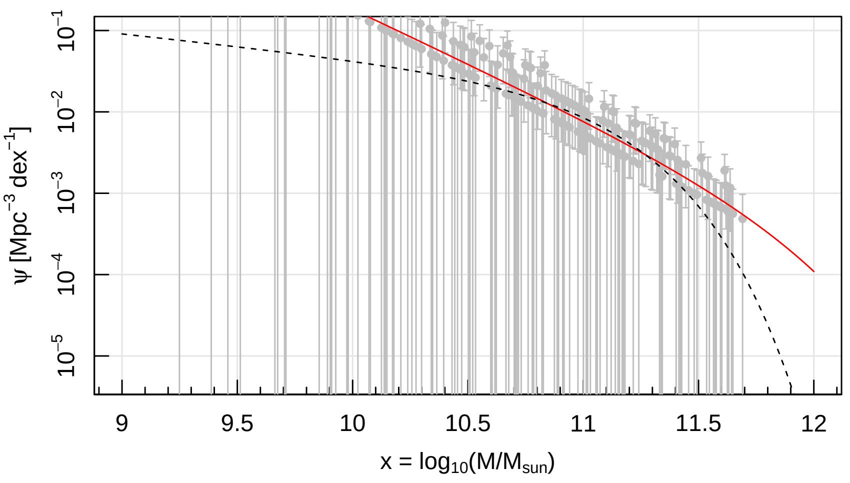


```
# save data for later use
x.mid.10 = x.mid
counts.10 = counts
```

The fitted MF (yellow) agrees 'reasonably' well with the input model (dashed line), in the sense that the deviation is approximately consistent with the error bars of the data points (grey).

This method has two major drawbacks: it uses arbitrary bins and the χ^2-minimisation is not the Bayesian approach, as can be seen from the problem of dealing with empty bins. In fact, one would hope that making the bins smaller would lead to better results, as normally the case in numerical analysis problems. However, as $\Delta x \to 0$, the number of empty bins becomes infinite and the finite number of non-empty bins have diverging values ψ_j with diverging uncertainties σ_j. Hence, the basic $1/V$-method is bound to break down.

To illustrate this failure of $1/V$-method, let us run the same code again, but with 500 mass bins. As can be seen in the plot, this leads to a fit (red) which is much worse.



Maximum Likelihood method with bins

The crucial difference between the $1/V$-method discussed before and the Bayesian approach is that the former relies on backwards modelling (i.e. reconstructing the MF from the source counts), whereas the latter reverts the problem to forward modelling (i.e. predicting the source counts from a MF model). An important consequence of this difference is that the Bayesian approach can correctly handle empty bins, which will ultimately allow us to get rid of the bins all together.

As always, we start by writing down the likelihood function, i.e. the probability of the observed data given a model. In the present case, this is the probability of observing the source counts $\{n_j\}$ given the parameters θ of a MF model (e.g. a Schechter function). Firstly, we note that the predicted number of galaxies detected in the mass bin j is

$$\lambda_j(\theta) = \int_{x_j-\Delta x/2}^{x_j+\Delta x/2} \lambda(x|\theta)dx \approx \lambda(x_j|\theta)\Delta x, \tag{14}$$

where the error of the approximation vanishes asymptotically as $\Delta x \to 0$.

Given the expected source counts $\lambda_j(\theta)$, the partial likelihood $\mathcal{L}_j(\theta)$ of detecting n_j galaxies in bin j can be modelled via a Poisson distribution,

$$\mathcal{L}_j(\theta) = \frac{e^{-\lambda_j(\theta)}\lambda_j(\theta)^{n_j}}{\Gamma(n_j+1)}. \tag{15}$$

As in earlier discussions, the total likelihood function $\mathcal{L} = \prod_j \mathcal{L}_j$ is conveniently written as logarithm. This yields a log-likelihood akin to the photon statistics of Cash (1979):

$$\ell(\theta) = \ln \mathbf{L}(\theta) = \sum_j \big(n_j \ln \lambda_j(\theta) - \lambda_j(\theta) - \ln\Gamma(n_j+1)\big). \tag{16}$$

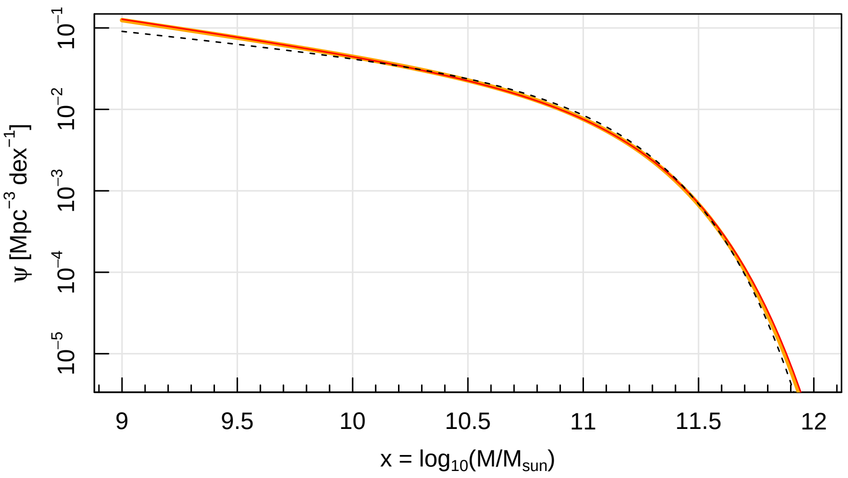
The last term in the sum does not depend on the model parameters an can hence be dropped if we are interested in the maximum and the shape of the likelihood function. The resulting effective log-likelihood (Eq. (16) without the last term) constitutes the core of numerous MF articles in the scientific literature since its introduction for parametric (Sandage 1979) and non-parametric (Efstathiou 1988) MFs.

A caveat in Eq. (16) are bins that are predicted to have zero counts, $\lambda_j(\theta) = 0$. For these bins, the factor $\ln \lambda_j(\theta)$ is undefined. However, this issue can be resolved by considering the limits $\lambda_j \to 0$: (a) if galaxies are observed in this bin ($n_j > 0$), then $\lim_{\lambda_j\to+0}(n_j \ln \lambda_j) = -\infty$, which will lead to a strict rejection of the model parameters θ in the sense that they will have zero probability in the posterior, irrespective of the prior; (b) if no galaxies are observed in the bin ($n_j = 0$) then $\lim_{\lambda_j\to+0} 0 \ln \lambda_j = 0$. It is hence correct to convene that $0 \ln 0 = 0$. Programmatically, both cases can be dealt with by replacing $\ln \lambda_j(\theta)$ for $\ln \lambda_j(\theta) + \epsilon$, where $\epsilon \ll 1$ is a small enough number not to have any effect if $\lambda_j(\theta) > 0$, but avoids infinities of $\lambda_j(\theta) = 0$. In **R**, the smallest working choice is $\epsilon = 10^{-323}$.

If for some reason we decided to keep the parameter-independent last term in eq. (16), a second caveat is worth pointing out: The gamma function $\Gamma(n_j+1)$ can get too large to be handled with standard floating point ranges. In **R**, any value $n_j \geq 170$ exceeds the floating point range and will thus generate infinite numbers (`Inf`), even if $\ln\Gamma(n_j+1)$ is a perfectly reasonable number. To avoid this issue, most modern programming languages include the natural logarithm of the gamma function as a separate function. In **R** this function, called **lgamma**, is part of the base functions (see part 2 of this course). Other common languages also provide this function: `lgamma` in C, `gammaln` in MATLAB and SciPy, `log_gamma` in Fortran 2008 and later.

In the present case, there is no need to keep the constant $\ln\Gamma(n_j+1)$. Upon dropping this term, the likelihood of the Schechter function parameters is maximised as follows:

```
# define log-likelihood function
log.likelihood = function(p,x.mid,counts) {
  dx = x.mid[2]-x.mid[1]
  lambda = psi.schechter(x.mid,p)*V(x.mid)*dx # vector of expected counts in each bin
  sum(counts*log(lambda+1e-323)-lambda) # without parameter-independent term
}

# find maximum likelihood solution
fit.10 = optim(p.true,log.likelihood,x.mid=x.mid.10,counts=counts.10,
               control=list(fnscale=-1))
fit.500 = optim(p.true,log.likelihood,x.mid=x.mid.500,counts=counts.500,
                control=list(fnscale=-1))

# plot data, fit and input model
magplot(NA,log='y',xlim=c(9,12),ylim=c(5e-6,0.1),
        xlab=expression('x = log' [10]*'(M/M' ['sun']*')'),
        ylab=expression(psi~'[Mpc'^'-3'~'dex'^'-1'*']'))
curve(psi.schechter(x,fit.10$par),col='orange',add=T,lwd=3)
curve(psi.schechter(x,fit.500$par),col='red',add=T)
curve(psi.schechter(x,p.true),lty=2,add=T)
```



The two coloured lines correspond to the same number of bins as before (10 for yellow, 500 for red). They both provide a reasonably good fit to the input model (dashed line). This clearly demonstrates that, irrespective of the number of bins, we obtain a sensible Schechter function fit when using the MLE. Moreover, the two fits are almost identical as expected for large enough numbers of bins.

ML Method without bins

In order to make the bins disappear, we first note that in the limit $\Delta x \to 0$ the approximation in Eq. (14) becomes exact: $\lambda_j(\theta) = \lambda(x_j)\Delta x$. In this limit, the log-likelihood of Eq. (16) becomes

$$\ell(\theta) = \sum_j \Big(n_j \ln\lambda(x_j|\theta) + n_j \ln\Delta x - \lambda(x_j|\theta)\Delta x - \ln\Gamma(n_j+1)\Big). \tag{17}$$

Let us now drop the terms that do not depend on θ, i.e. the terms $n_j \ln \Delta x$ and $\ln \Gamma(n_j+1)$ and rewrite the remaining *effective* log-likelihood as

$$\ell(\theta) = \Delta x \sum_j \left(\frac{n_j}{\Delta x} \ln \lambda(x_j|\theta) - \lambda(x_j|\theta) \right). \tag{18}$$

If $\Delta x \to dx$, the ratio $n_j/\Delta x$ becomes a sum of Dirac delta-functions, $n_j/\Delta x \to \sum_i \delta_{\rm D}(x_j - x_i)$, where i iterates over the galaxies in the survey. In this limit, the sum becomes an integral,

$$\ell(\theta) = \int \left(\sum_i \delta_{\rm D}(x - x_i) \ln \lambda(x|\theta) - \lambda(x|\theta) \right) dx \tag{19}$$

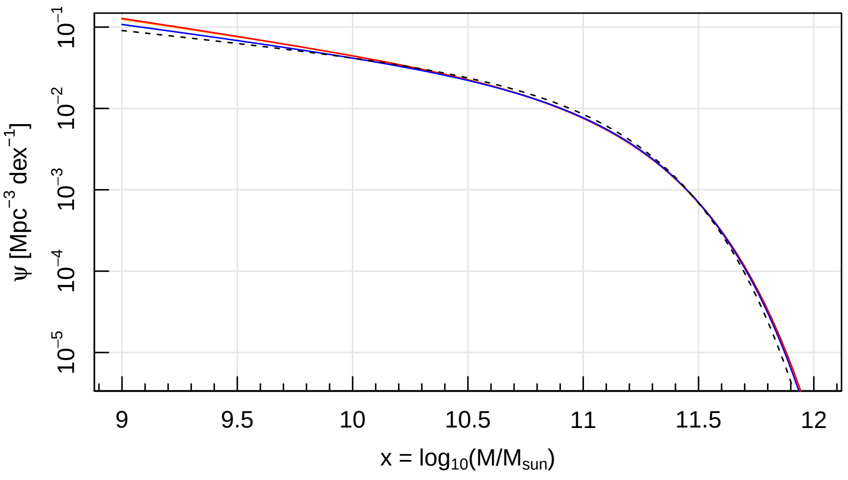
and finally,

$$\ell(\theta) = \sum_i \ln \lambda(x_i|\theta) - \int \lambda(x|\theta) dx \tag{20}$$

This equation, where $\lambda(x|\theta)$ is the expected number density per unit x for parameters θ, is the universal effective log-likelihood of the bin-free Bayesian inference problem. It is not specific to galaxy MFs, but generically applies to fitting any NDF $\lambda(x|\theta)$.

Let us run this in **R**:

```
# define log-likelihood function
lamb = function(x,p) psi.schechter(x,p)*V(x)
log.likelihood = function(p,data.x) sum(log(lamb(data.x,p)))-integrate(lamb,7,14,p=p)$value

# find maximum likelihood solution
fit.inf = optim(p.true,log.likelihood,data.x=data.x,hessian=T,control=list(fnscale=-1))

# plot data, fit and input model
magplot(NA,log='y',xlim=c(9,12),ylim=c(5e-6,0.1),
        xlab=expression('x = log' [10]*'(M/M' ['sun']*')'),
        ylab=expression(psi~'[Mpc'^'-3'~'dex'^'-1'*']'))
curve(psi.schechter(x,fit.10$par),col='orange',add=T)
curve(psi.schechter(x,fit.500$par),col='red',add=T)
curve(psi.schechter(x,fit.inf$par),col='blue',add=T)
curve(psi.schechter(x,p.true),lty=2,add=T)
```



As expected, the solution of the bin-free ML approach (blue) stays very similar to the previous solutions.

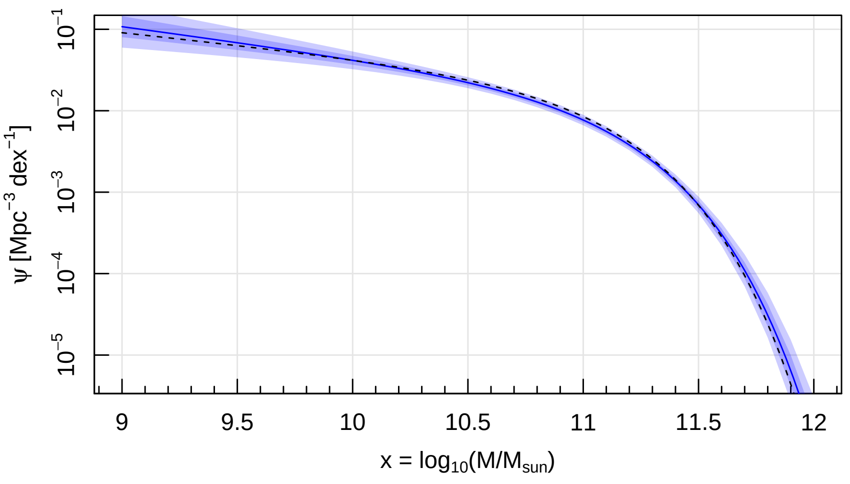
Uncertainty ranges

How can we determine the measurement uncertainties on the MF? The covariance matrix Σ_θ of the model parameters θ can be determined using the Laplace approximation, i.e. by inverting the negative Hessian of the log-likelihood at the maximum point $\hat{\theta}$. To linear order, the statistical uncertainty of any point on the fitted MF can then be computed via (see part 4, section 2),

$$\sigma^2_\psi(x) = \mathbf{J}\Sigma_\theta\mathbf{J}^T, \text{ where } \mathbf{J} = \frac{\partial\psi(x|\hat{\theta})}{\partial\theta}. \tag{21}$$

Let us do this in **R** and plot the 68% and 95% uncertainty ranges as light shades.

```r
x.plot = seq(9,12,by=0.1)
f = function(p) log10(psi.schechter(x.plot,p))
J = jacobian(f,fit.inf$par)
C = solve(-fit.inf$hessian)
C.psi = J%*%C%*%t(J)
sd.psi = sqrt(diag(C.psi))

magplot(NA,log='y',xlim=c(9,12),ylim=c(5e-6,0.1),
        xlab=expression('x = log' [10]*'(M/M' ['sun']*')'),
        ylab=expression(psi~'[Mpc'^'-3'~'dex'^'-1'*']'))
psi.top = psi.schechter(x.plot,fit.inf$par)*10^(sd.psi*2)
psi.bottom = psi.schechter(x.plot,fit.inf$par)/10^(sd.psi*2)
polygon(c(x.plot,rev(x.plot)),c(psi.top,rev(psi.bottom)),col='#0000ff33',border=NA)
psi.top = psi.schechter(x.plot,fit.inf$par)*10^sd.psi
psi.bottom = psi.schechter(x.plot,fit.inf$par)/10^sd.psi
polygon(c(x.plot,rev(x.plot)),c(psi.top,rev(psi.bottom)),col='#0000ff33',border=NA)
curve(psi.schechter(x,fit.inf$par),col='blue',add=T)
curve(psi.schechter(x,p.true),lty=2,add=T)
```

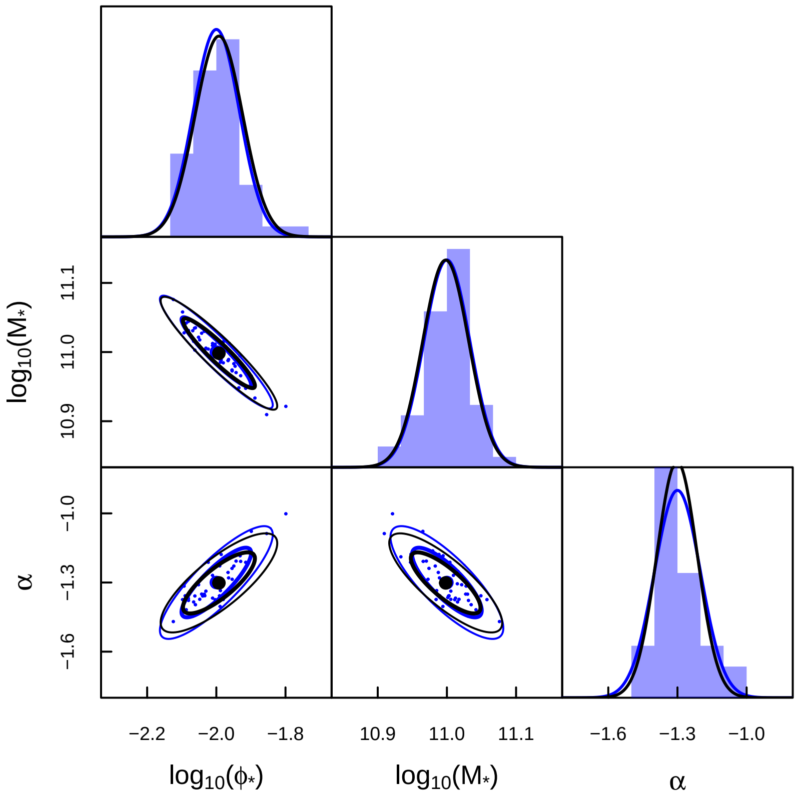


The fit is indeed consistent with the true input model at any point on the Schechter function.

Testing many mock surveys

Let us now repeat the mock survey 50 times, drawing a new random set of galaxies from $\lambda(x|\bar{\theta})$ each time. The aim is to check if the distribution of the MLE solutions (blue) matches the expected uncertainties from the Laplace approximation (black).

```r
n.iterations = 50
p.best = array(0,c(n.iterations,3))
c.mean = array(0,c(3,3))
for(i in 1:n.iterations) {
  # We could use data.x = rsurvey(rpois(1,N.expected)), but to speed things up
  # let's use the equivalent function from dftools
  data.x = dfmockdata(veff=V, shot.noise=T, seed=i)$x
  fit = optim(p.true,log.likelihood,data.x=data.x,hessian=T,control=list(fnscale=-1))
  p.best[i,] = fit$par
  c.mean = c.mean+solve(-fit$hessian)
}
c.mean = c.mean/n.iterations
names = c(expression('log'[10]*'('*phi['*']*')'),
          expression('log'[10]*'(M'['*']*')'),expression(alpha))
dfplotcov(list(p.best,list(p.true,c.mean)), cloud.alpha=1, names=names,
           text.size.numbers=0.6,text.size.labels=0.8,
           text.offset.numbers=c(0.6,-1.2))
```



As we can see this comparison works out very favourably: The Schechter parameters fitted to the 50 random survey realisations (little blue dots) have means (big blue dots), which line up nicely with the input parameters $\bar{\theta}$ (big black dots). Moreover, the measured covariance of these 50 realizations, represented by the blue ellipses, is consistent with the mean covariance of the Laplace approximations (black ellipses).

Real-world considerations

In dealing with real galaxies there are several major complications, which can all be addressed in a Bayesian framework. If you ever need a solution to these issues, you might find it useful to read my article by Obreschkow et al. (2018, see https://arxiv.org/abs/1712.00149) and use the associated **dffit** routine in the **dftools** package. Some of the most prominent real-world problems (dealt with by **dftools**) are:

- Mass measurement errors are, of course, subject to some measurement uncertainty. Further complicating the situation, different galaxies usually have different measurement uncertainties, for instance due to different photon counts, different sky noise or different distance errors. It is important to appreciate that the effect of such measurement uncertainties does *not* disappear if the size of the sample increases to infinity. Instead, they systematically bias the model parameters θ. In the case of galaxy MFs (and LFs), the main component of this bias is the *Eddington bias*: a galaxy of an observed luminosity L is more likely to have a true luminosity smaller than L than it is to have a true luminosity larger than L. This is because the declining LF makes it more likely that the observed value L is the result of a galaxy with a luminosity $< L$ scattered upwards than the result of a galaxy with $> L$ scattered downwards. Overall, Eddington bias tends to smooth out the LF (or MF) and artificially increase the number of the most luminous objects.
- Most galaxy surveys do not have sharp sensitivity limits. There is no well-defined maximum volume

inside which all galaxies of a fixed mass are detected, while outside none are detected. Instead, the fraction of detected sources (true positives) decreases gradually when reaching the detection limit, while the fraction of wrong detections (false positives) increases; and the detection probability depends on other quantities than mass or luminosity (e.g. on the inclination of a galaxy).

Summary

At the example of galaxy mass functions, we have shown that the *general* log-likelihood function of NDFs reads

$$\ell(\theta) = \sum_i \ln \lambda(x_i|\theta) - \int \lambda(x|\theta)dx, \tag{22}$$

where $\{x_1, ..., x_N\}$ is a set of N independent samples drawn from an NDF $\lambda(x|\bar{\theta})$ (with $\bar{\theta}$ denoting the true population parameters).

The log-likelihood for NDFs contains the log-likelihood for PDFs as a particular case. In fact, PDFs are normalised to unity, thus they can be seen as NDFs, whose norm $\int \lambda(x|\theta)dx$ is always unity, irrespective of θ. Hence, the integral term in Eq. (22) can be dropped; and we end up with the known $\ell(\theta) = \sum \ln \rho(x_i|\theta)$ (see part 4, sections 1 & 2).

The entire formalism presented so far is straightforward to generalize to D-dimensional NDFs, defined, such that the *expected* number of events per unit dx is

$$\langle dN \rangle = \lambda(\mathbf{x})d^D x,$$

where $\mathbf{x}$ is a D-dimensional vector. In this case the log-likelihood generalises to

$$\ell(\theta) = \sum_i \ln \lambda(\mathbf{x}_i|\theta) - \int \lambda(\mathbf{x}|\theta)d^D x, \tag{23}$$

where $\mathbf{x}_i$ are N data points, each being a D-dimensional vector. For instance, if we construct the distribution of galaxies in the mass-size plane ($D = 2$), the first dimension could represent the mass and the second could represent the half-mass radius.

Example: Fitting an Navarro–Frenk–White profile

This section is a bonus part to this chapter, which presents an example of fitting a 2D NDF model to data in an astrophysical context.

Problem statement

In astronomy, dark matter is regularly found in the form of spherical haloes, whose radial density profiles are well approximated by the Navarro–Frenk–White (NFW) profile. It is usually written in the form

$$\rho(r) = \frac{\rho_0}{\frac{r}{R_s}\left(1 + \frac{r}{R_s}\right)^2},$$

where r is the radius from the centre, ρ_0 is a normalising density, and R_s is the scale radius in the same units as r. These parameters can vary from halo to halo.

Since the integrated mass of the NFW profile is actually divergent to infinity, we usually consider the mass out to a reference radius R_{200}, where $R_{200} = cR_s$, and c is called the concentration parameter. The mass out to a given radius (R_{max}) can be computed as

$$M = \int_0^{R_{max}} 4\pi r^2 \rho r = 4\pi\rho_0 R_s^3 \left[\ln\left(\frac{R_s + R_{max}}{R_s}\right) - \frac{R_{max}}{R_s + R_{max}}\right]$$

In this cosmology the critical mass density of the Universe is $1.3 \times 10^{11} \mathrm{M}_\odot\,\mathrm{Mpc}^{-3}$ (sometimes written in terms of energy, but you can convert between these two forms using $E = mc^2$), and we define R_{200} as the radius containing an average density equal to 200 times the critical mass density of the Universe.

The data file “../data/NFW.csv” contains the Cartesian coordinates (in units kpc) for a particular spherical halo with 2×10^4 simulated particles, where the mass of each of these particles is $10^8 \mathrm{M}_\odot$. Using these data find the mass within R_{200} (M_{200}) and concentration (c) of the halo.

Initialization of the problem

Define physical constants:

```r
mpart = 1e8 # [Msun] particle mass
rho.crit = 130 # [Msun/kpc^3] closure mass density of the universe
```

Load data:

```r
fn = '../data/NFW.csv'
d = read.csv(fn) # [kpc] coordinates
n = dim(d)[1] # number of particles
```

Determine radii relative to centre of mass:

```r
cm = colSums(d)/n # [kpc] centre of mass coordinates
r = sqrt(colSums((t(d)-cm)^2)) # [kpc] radii
Rmax = max(r) # maximum radius
```

Method 1: First determine R_{200}, then fit for c

Determine R_{200}:

```r
r = sort(r)
V = 4*pi/3*r^3 # [kpc^3] vector of enclosed spherical volumes
M = seq(n)*mpart # [Msun] vector of enclosed total mass
density.ratio = M/V/rho.crit # [units of rho.crit] vector of enclosed mean density
f = approxfun(r,density.ratio,rule=2)
R200 = uniroot(function(r) f(r)-200,c(0,1e3))$root
cat(sprintf('R200 = %.1f kpc\n',R200))
```

```
## R200 = 208.8 kpc
```

Convert to M_{200}:

```r
M200 = sum(r<=R200)*mpart
cat(sprintf('M200 = %.3e Msol\n',M200))
```

```
## M200 = 9.917e+11 Msol
```

Given a fixed truncation radius R_{200}, the total NFW mass is converged and the NFW profile can be normalised to a probability density function (PDF) per unit of radius. To do so, it suffices to normalise the mass differentials $dM(r) = 4\pi r^2 \rho_{\mathrm{NFW}}(r)dr$ by the total mass $M_{200} = 4\pi\rho_0 R_s^3[\ln(1+c) - c/(1+c)]$:

$$\mathrm{PDF}(r|c) = \frac{1}{M_{200}}\frac{dM}{dr} = \frac{x}{R_s\,(1+x)^2\left[\ln(1+c) - c/(1+c)\right]}, \tag{24}$$

where $R_s = R_{200}/c$ and $x \equiv r/R_s$. This PDF has indeed units of 1/length and satisfies $\int_0^{R_{200}} \mathrm{PDF}(\mathrm{r}|\mathrm{c})dr = 1\ \forall c > 0$. Let's now fit this PDF to all particles inside the (fixed) radius R_{200}, which is achieved by maximising the log-likelihood function

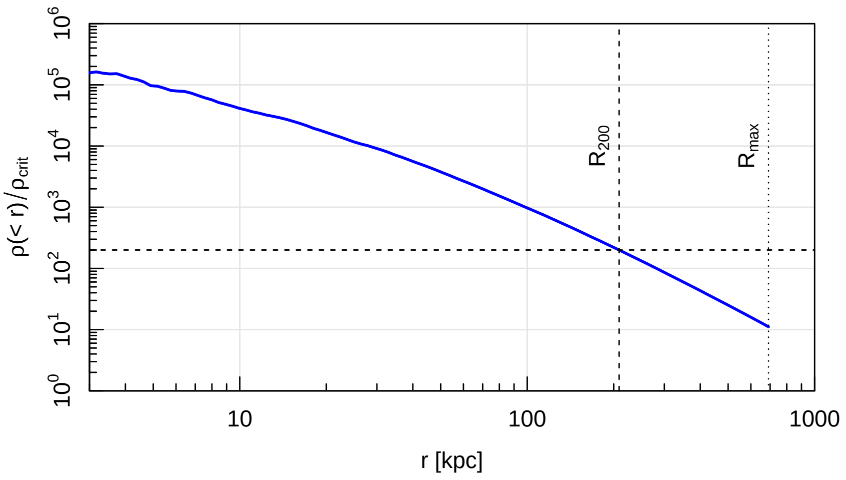
$$\ell(c) = \sum_i \ln \mathrm{PDF}(r_i|c), \tag{25}$$

where the sum goes over all particles with $r_i \leq R_{200}$. Maximising this likelihood in **R** gives:

```r
nfw.pdf = function(r,c) {
  # r = radius [kpc]=
  Rs = R200/c # [kpc] scale radius
  x = r/Rs # [-] normalised radius
  return(x/(1+x)^2/(log(1+c)-c/(1+c))/Rs) # [1/kpc] 1D probability density
}
log.likelihood = function(p) sum(log(nfw.pdf(r[r<=R200],p)))
fit = optimise(log.likelihood,c(1,10),maximum=TRUE)
cat(sprintf('MLE concentration c = %.3f\n',fit$minimum))
```

The density profile of the halo is visualised below. This plot also shows that the density profile is sampled out to a radius $R_{\max} \approx 691$kpc – a feature, which we will use in the following method.

```r
magcurve(f,3,Rmax,xlim=c(3,1e3),ylim=c(1,1e6),lwd=2,log='xy',xaxs='i',yaxs='i',
         xlab='r [kpc]',ylab=expression(rho*"(< r)"/rho [crit]),col='blue')
abline(h=200,v=c(R200,Rmax),lty=c(2,2,3))
text(R200*0.85,1e4,expression(R [200]),srt=90)
text(Rmax*0.85,1e4,expression(R [max]),srt=90)
```



Method 2: Fitting for R_{200} and c simultaneously

Motivation: Method 1 has two drawbacks:

- Fitting the PDF implicitly assumes that the 20,000 particles represent a random sample from a population PDF. This, in turn, means that the measured value of R_{200} is itself a statistical observable. Since R_{200} and c are correlated, an isolated ML estimation of c isn't strictly correct.
- It seems odd to ignore the particles at radii between R_{200} and $R_{\max}$, which could further constrain the NFW profile. (at least purely statistically speaking; see last comment on next page)

A Bayesian solution to both issues is to maximise the full log-likelihood $\ell(\rho_0, R_s)$, accounting for all particles. In other words, we are not just fitting a PDF, but a *number* density function (NDF) with a normalisation that is itself set by the model parameters. As discussed in the lecture (chapter 12), the log-likelihood in this case reads (up to a constant)

$$\ell(\rho_0, R_s) = \sum_i \ln \mathrm{NDF}(r_i|\rho_0, R_s) - \int_0^{R_{\max}} \mathrm{NDF}(r|\rho_0, R_s)dr, \tag{26}$$

where $\mathrm{NDF}(r|\rho_0, R_s)$ is the expected number of particles per unit radius at radius r. Explicitly,

$$\mathrm{NDF}(r|\rho_0, R_s) = \frac{4\pi r^2 \rho_0}{x(1+x)^2 m}, \tag{27}$$

where $x \equiv r/R_s$ and m is the mass of a simulation-particle. As expected, this function has units of 1/length. Since only particles with $r \leq R_{\max}$ are provided, the NDF should be zero for $r > R_{\max}$, but this has already been accounted for by the upper limit of the integral in the log-likelihood.

Since we are interested in c and R_{200} rather than ρ_0 and R_s, we express the latter pair as a function of the former pair in the following function.

```r
nfw.ndf = function(r,p) {
  # r = radius [kpc]
  c = p[1] # concentration
  R200 = p[2] # [kpc]
  Rs = R200/c # [kpc]
  rho0 = 200/3*rho.crit*c^3/(log(1+c)-c/(1+c)) # [Msun/kpc^3]
  x = r/Rs # [-] normalised radius
  rho = 4*pi*r^2*rho0/x/(1+x)^2/mpart # [1/kpc] 1D number density
  return(rho)
}
```

The MLE solution for c and R_{200} then reads

```r
log.likelihood = function(p) sum(log(nfw.ndf(r,p)))-integrate(nfw.ndf,0,Rmax,p=p)$value
fit = optim(c(5,200),log.likelihood,hessian=TRUE,
            control=list(reltol=1e-12,parscale=c(1,100),fnscale=-1))
sd = sqrt(diag(solve(-fit$hessian)))
cat(sprintf('MLE concentration c    = %.3f\u00B1%.3f\n',fit$par[1],sd[1]))
```

```
## MLE concentration c    = 5.008±0.111
```

```r
cat(sprintf('MLE virial radius R200 = %.1f\u00B1%.1f kpc\n',fit$par[2],sd[2]))
```

```
## MLE virial radius R200 = 209.3±0.8 kpc
```

```r
cat(sprintf('MLE virial mass M200   = %.3e\u00B1%.3e Msol\n',
            200*rho.crit*4*pi/3*fit$par[2]^3,200*rho.crit*4*pi*fit$par[2]^2*sd[2]))
```

```
## MLE virial mass M200   = 9.989e+11±1.152e+10 Msol
```

These halo data were actually drawn from a spherical NFW model with $c = 5$ and $M_{200} = 10^{12}\mathrm{M}_{\mathrm{sol}}$. The MLE solution is consistent with this input.

Some further thoughts

- In this particular example, the results of methods 1 and 2 are statistically consistent, albeit conceptually quite different.
- One could argue that even the centre of mass is a statistical observable that depends on the random sample of 20,000 particles, which would further modify the likelihood of method 2. An alternative quick fix would be to re-sample the 20,000 particles (e.g. using bootstrapping) and redetermine the CM, R_{200} and c at each iteration, following Method 1.
- Flat priors on the NFW parameters (ρ_0, R_s, R_{200} or c) are not very physical, since these parameters are all bound to positive numbers. Probably logarithmic priors would be more objective in the sense of Jeffreys.
- In method 2, we used all particles out to R_{max}, which lies significantly beyond the virial radius. In real cosmological scenarios (rather than analytically generated data or controlled simulations), one would *not* expect the density profiles to be settled beyond the virial radius. It might therefore still make sense to limit the considered particles to R_{200} (or even less) when using method 2.