

Implementing a Bayesian Statistical Analysis with STAN

Chris Miller

Bayesian Inference

$$p(\theta|y;x) \propto L(y|\theta;x)p(\theta;x)$$

θ is a sequence of modeled unknown values: model parameters, latent variables, missing data, future predictions

y is a sequence of modeled known values

x is a sequence of un-modeled predictors

Given a likelihood model L and prior probabilities of the unknown parameters θ , one can infer the posterior probability up to a constant.

But $p(\theta|y;x)$ is not a number. It is complex and usually multidimensional and requires careful propagation of the uncertainties on θ from the model.

Solution is to “sample the posterior” in a smart but random way and use the results from these samples to define the credible intervals on the model θ .

STAN

- A computational tool developed to aid in Bayesian analyses (JAGS, BUGS, OpenBUGS).
- Requires the user to define θ and provide y and x as well as the likelihood and the prior probabilities.
- STAN uses Hamiltonian Monte Carlo, a form of MCMC to sample.
- STAN works from the command line, from within Python or from within R.
- STAN works on UNIX/LINUX, OSX, and Windows.
- STAN is well documented.

Our Mission: Determine an underlying linear relationship

The scaling relation between richness and mass of galaxy clusters: a Bayesian approach

S. Andreon^{1★} and M. A. Hurn²

¹INAF–Osservatorio Astronomico di Brera, Milano, Italy

²Department of Mathematical Sciences, University of Bath, Bath

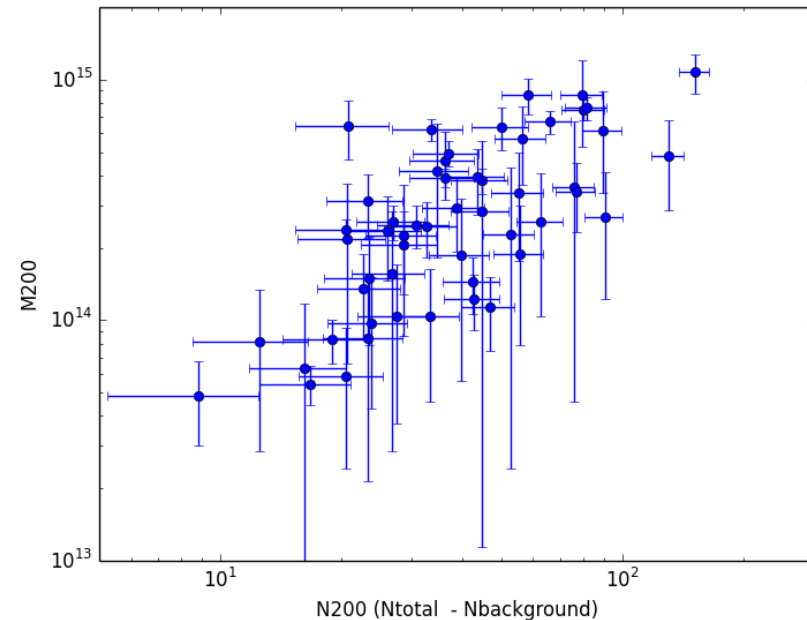
$$y_i = \alpha + \beta x_i$$

$$x = \log_{10}(M200)$$

$$y = \log_{10}(n200)$$

$$y = N(\mu, \sigma^2)$$

$$\log_{10}(M200) \approx (\alpha + 14.5 + \beta(\log(n200) - 1.5), \sigma_{scat}^2)$$



The data

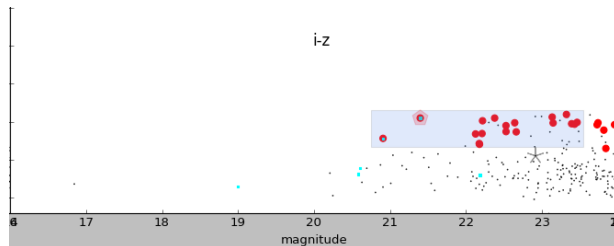
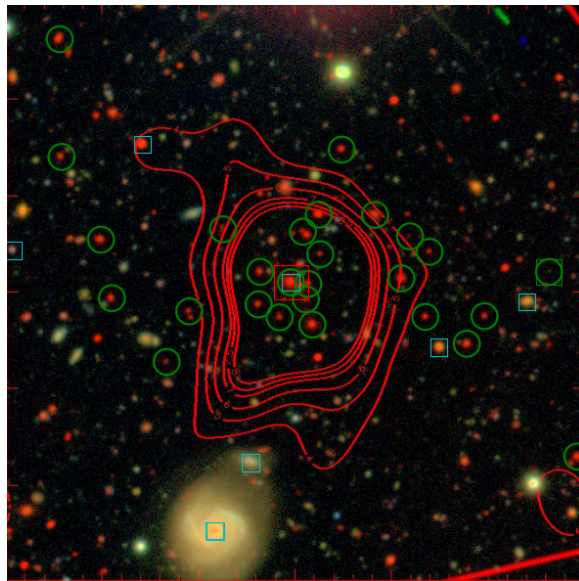
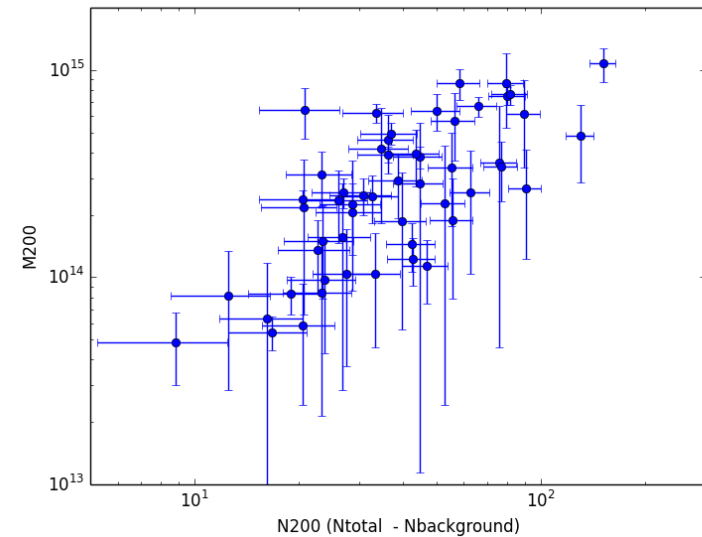
Observed M200: Mass based on a dynamical measure

Observed M200 error: Best guess from simulations

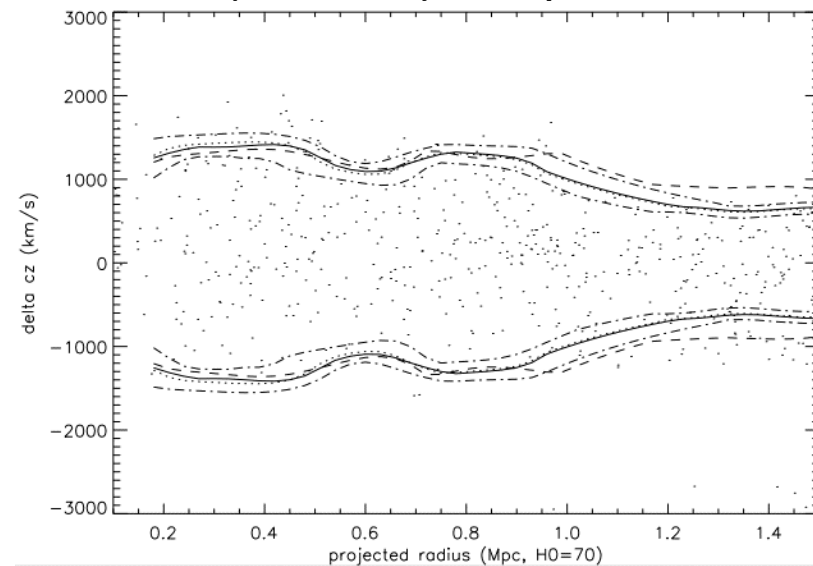
Observed NTOT: All galaxies in a radius

Observed NBKG: Estimate of the background galaxies

C: The size of the area used to measure NBKG



Escape velocity = $\sqrt{2\phi}$ to M_{200}



The parameters

M200: The true mass

True M200 error: observed error

N200: The true number count

NBKG: The true background count

α : The intercept of the linear relation

β : The slope of the linear relation

σ : The “intrinsic variation” of the linear relation (*not the error on the fit*).

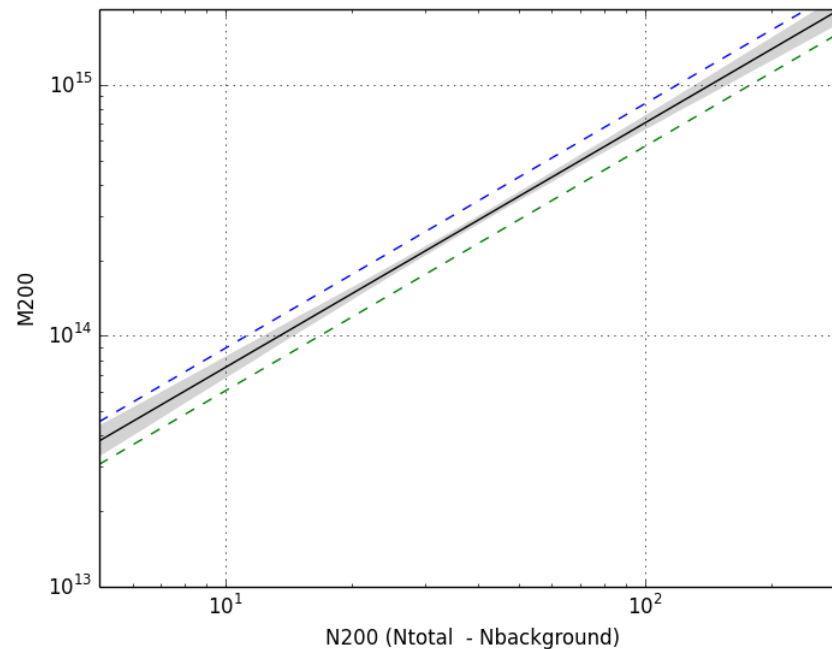
$$y_i = \alpha + \beta x_i$$

$$x_i = \log_{10}(n200)$$

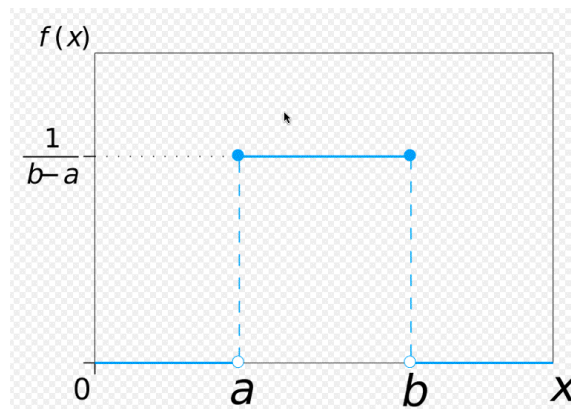
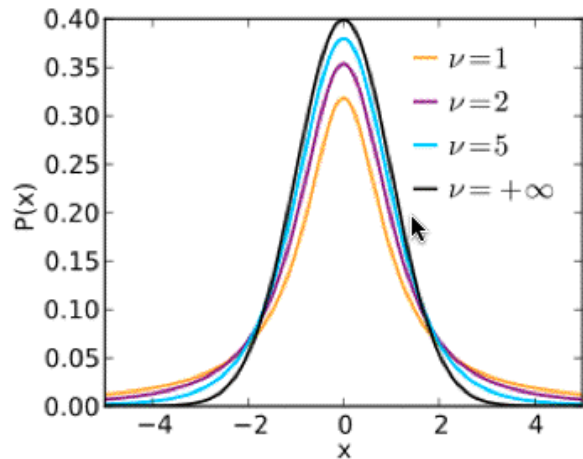
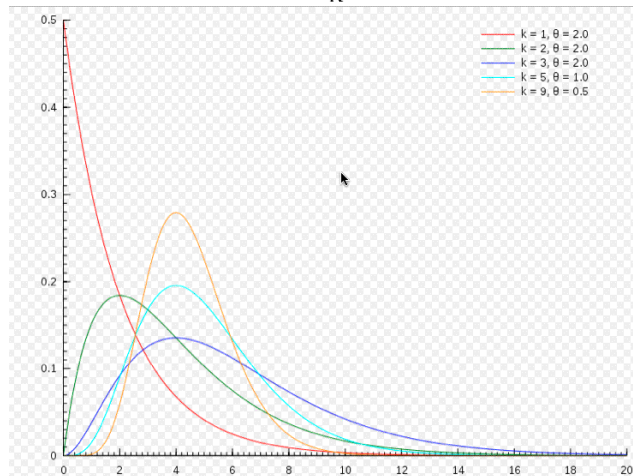
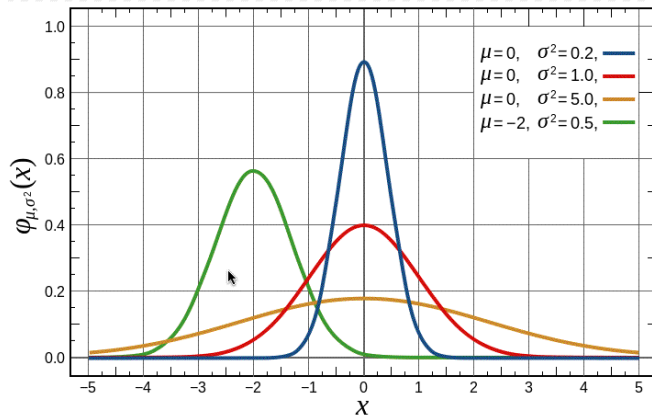
$$y_i = \log_{10}(M200)$$

$$y = N(\mu, \sigma^2)$$

$$\log_{10}(M200) \approx (\alpha + 14.5 + \beta(\log(n200) - 1.5), \sigma_{scat}^2)$$



The Priors



M200: Normal, comes from the linear model

True M200 error: Gamma, small values

N200: Uniform, > 0

NBKG: Uniform, > 0

α : Normal, mean 0, large

β : Student T, 1 degree-of-freedom

σ : Gamma, small values

Observed M200: Normal, true mass with error

Observed M200 error: Gamma, small values

Observed NTOT: Poisson

Observed NBKG: Poisson

BEWARE!

Univariate Continuous Distributions

Name	JAGS	OpenBUGS	Stan
Beta	dbeta(a, b)	dbeta(a, b)	beta(a, b)
Chi-Square	dchisqr(k)	dchisq(k)	chi_square(k)
<i>Double Exponential</i>	ddexp(mu, tau)	ddexp(mu, tau)	double_exponential(mu, 1.0 / tau)
Exponential	dexp(lambda)	dexp(lambda)	exponential(lambda)
F	df(n, m)	df(n, m, 0.0, 1.0)	
<i>Flat</i>		dflat()	Note ^[1]
Gamma	dgamma(r, lambda)	dgamma(r, mu)	gamma(r, lambda)
Generalized Extreme Value		dgev(mu, sigma, eta)	
Generalized F		df(n, m, mu, tau)	
Generalized Gamma	dgen.gamma(r, lambda, b)	dggamma(r, lambda, b)	
Generalized Pareto		dgpar(mu, sigma, eta)	
<i>Generic Loglikelihood</i>		dloglik(lambda)	Note ^[2]
<i>Logistic</i>	dlogis(mu, tau)		logistic(mu, 1.0 / tau)
Non-central Chi-square	dnchisqr(k, delta)		
<i>Log-Normal</i>		dlnorm(mu, tau)	log_normal(mu, pow(tau, -0.5))
<i>Normal</i>	dnorm(mu, tau)	dnorm(mu, tau)	normal(mu, pow(tau, -0.5))
<i>Pareto</i>	dpar(alpha, c)	dpar(alpha, c)	pareto(pow(c, alpha), alpha)
<i>Student-t</i>	dt(mu, tau, k)	dt(mu, tau, k)	student_t(k, mu, pow(tau, -0.5))
Uniform	dunif(a, b)	dunif(a, b)	uniform(a, b)

<http://www.jrnold.me/blog/jagsopenbugs-to-stan-distributions.html>

The Answer

see Figure 2 in Andreon and Hurn 2010

