Solutions to Practical 1

1.
MLE is y/n = 7/20 = 0.35
(a) Using results from slide 13 of lectura 1

Prior: Beta(1, 1)
Posterior: Beta(1+7, 1+20-7) = Beta(8, 14)

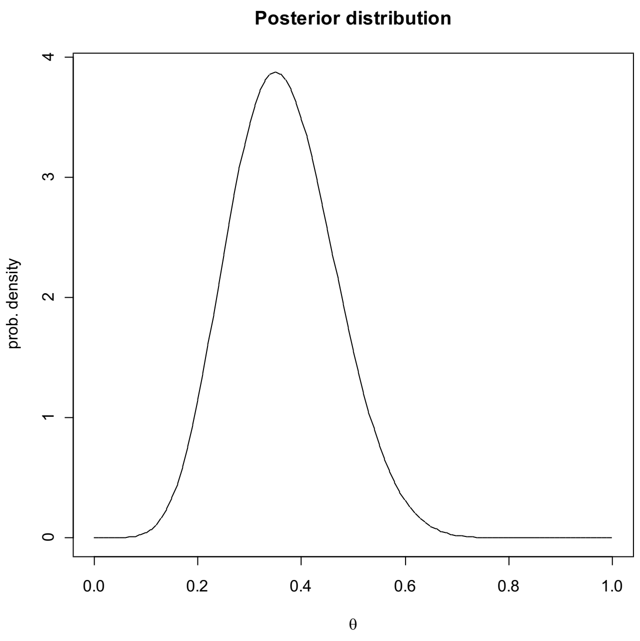
Or from first principles:

$$p(y|\theta) \propto \theta^y(1-\theta)^{n-y}$$
$$p(\theta) \propto 1$$

Hence $p(\theta|y) \propto \theta^y(1-\theta)^{n-y}$

This is a Beta distribution with parameters $y+1$ and $n-y+1$

```
> curve(dbeta(x,8,14),0,1)
```



To add labels and title:

```
> curve(dbeta(x,8,14),0,1,
xlab=expression(theta), ylab="prob. density", main="Posterior
distribution")
```

Quantiles:

```
> qtl <- c(0.025, 0.975)
> qbeta(qtl, 8,14)
[1] 0.1810716 0.5696755
```

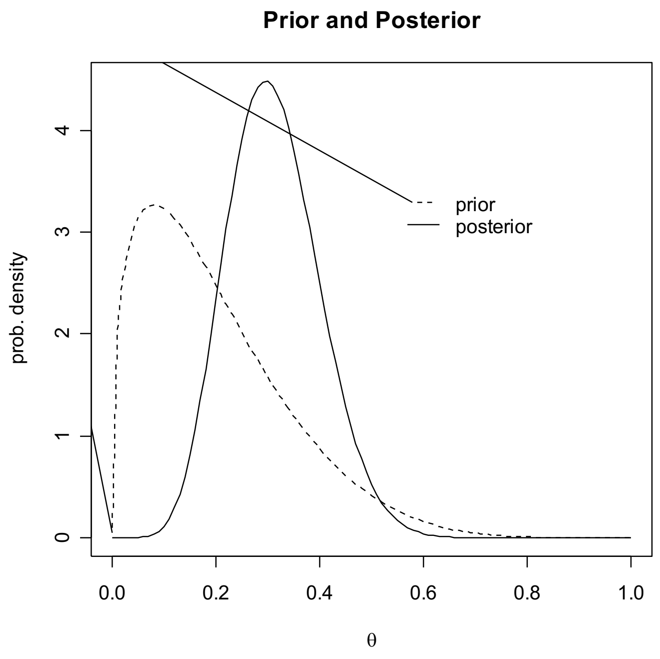
So 95% credible interval is (0.18, 0.57)

(b)

$$\mu = 0.2, \nu = 7$$
$$a = \mu\nu = 1.4$$
$$b = 7(1-0.2) = 5.6$$

Prior: Beta(1.4, 5.6)
Posterior: Beta(8.4, 18.6)

```
>
curve(dbeta(x,8.4,18.6),0,1,
xlab=expression(theta),
ylab="prob. density",
main="Prior and Posterior")
> curve(dbeta(x,1.4,5.6),0,1,
add=T, lty=2)
```



Quantiles for informative prior:

```
> qbeta(qtl,8.4,18.6)
[1] 0.1546589 0.4940403
```

95% credible interval: (0.15, 0.49)

Now with prior derived from 50 prior observations

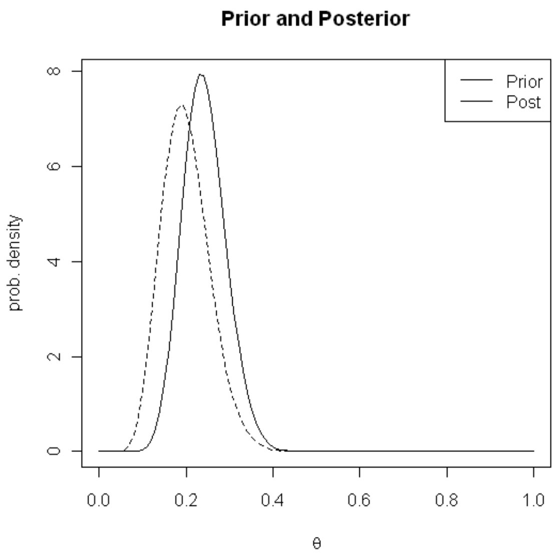
Now if $\mu = 0.2, \nu = 52$
$a = \mu\nu = 0.2 * 52 = 10.4$
$b = \nu\,(1-\mu) = 52 * 0.8 = 41.6$

So Posterior is $\theta|y \sim Beta(17.4, 54.6)$

95% credible interval is

(0.151, 0.346)

So posterior is sharper and credible interval is smaller because our prior is sharper and data has less influence



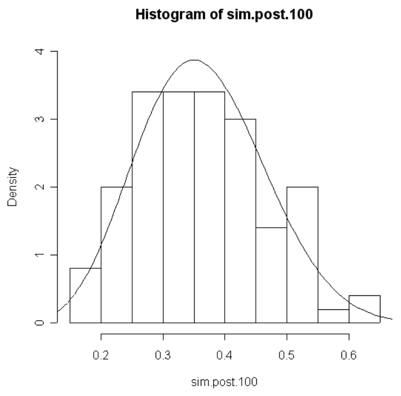
Solutions to Practical 2

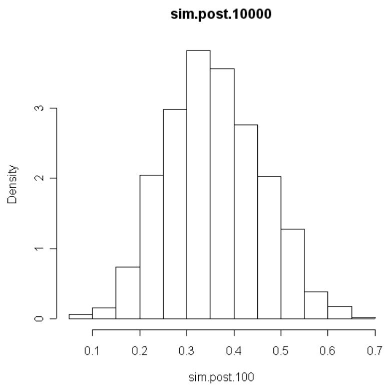
Sampling from standard distributions in R

(a) Posterior distribution is:

$$\theta \mid y \sim Beta(\alpha + y, \beta + n - y)$$
$$\theta \mid y \sim Beta(1+7, 1+20-7) = Beta(8,14)$$

So $a = 8, b = 14$



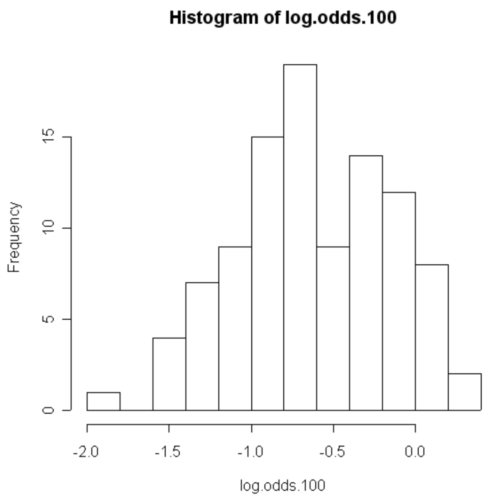


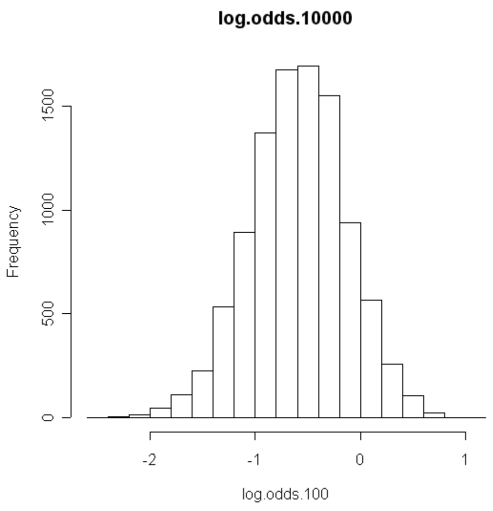
(b) Credible intervals:

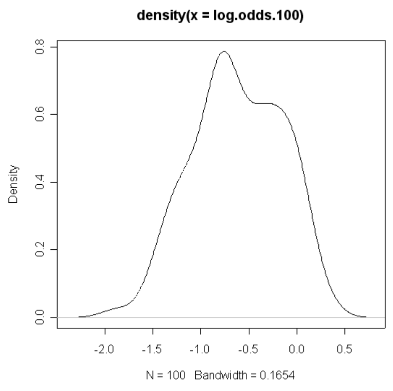
	2.5%	97.5%
Simulation (100)	0.1856	0.5730
Simulation (10,000)	0.1809	0.5713
Theoretical	0.1811	0.5697

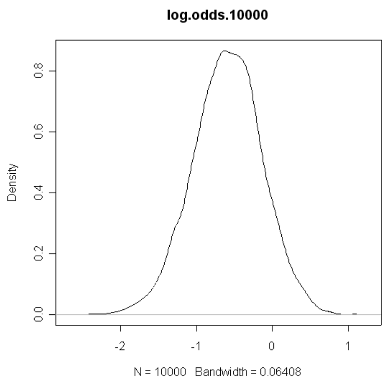
Not surprisingly 10,000 values mirror theoretical values better

(c) Log odds









95% credible interval	**2.5%**	**97.5%**
100 simulation values	-1.456	0.093
10,000 simulation values	-1.510	0.287

So results are closer to theoretical results when more simulations have been carried out.

Solutions to Practical 4

1. *Regression models with DIC*

R code for the regressions:

```
eg2 <- data.frame(dget("prac 7 ex 1 data.txt"))
attach(eg2)
# scatterplot matrix plus smoother
pairs(eg2, panel=panel.smooth)
# centring x-vars
x1 <- x1-mean(x1)
x2 <- x2-mean(x2)
x3 <- x3-mean(x3)
# linear regression on x1, x2, x3
lm1 <- lm(y~x1+x2+x3)
summary(lm1)
AIC(lm1)
# linear regression on x1, x2
lm2 <- update(lm1, .~.-x3)
summary(lm2)
AIC(lm2)
# linear regression on x2, x3
lm3 <- update(lm1, .~.-x1)
summary(lm3)
AIC(lm3)
# linear regression on x1, x3
lm4 <- update(lm1, .~.-x2)
summary(lm4)
AIC(lm4)
```

parameter	95% credible interval
β_1	9.68, 10.69
β_2	-5.21, -4.80
β_3	-0.14, 0.09

WinBUGS results from running 20,000 with burn-in of 1,000.

		"Classical" models		Bayesian models	
Model	*x*-variables	No. of parameters	*AIC*	*pD*	*DIC*
1	x_1, x_2, x_3	5	751.9	4.9	751.9
2	x_1, x_2	4	750.0	4.0	750.2
3	x_2, x_3	4	1039.2	4.0	1039.3
4	x_3, x_1	4	1075.4	4.0	1075.5

Stats for model 1:

node	**mean**	**sd**	**MC error**	**2.5%**	**median**	**97.5%**
beta0	-98.92	1.021	0.00926	-100.9	-98.92	-96.91
beta1	10.19	0.2545	0.00222	9.686	10.18	10.69
beta2	-5.008	0.1042	0.001003	-5.213	-5.008	-4.802
beta3	-0.02284	0.05869	5.572E-4	-0.1371	-0.02325	0.09094
tau	0.009839	0.001416	1.307E-5	0.00727	0.009768	0.01277

With informative prior: note how the prior dominates (cf stats for first model above with non-informative priors) …

node	**mean**	**sd**	**MC error**	**2.5%**	**median**	**97.5%**
beta0	-98.94	4.089	0.04677	-107.0	-98.9	-90.97
beta1	0.5901	0.2566	0.003216	0.09668	0.5902	1.098
beta2	-4.516	0.4144	0.00471	-5.328	-4.516	-3.695
beta3	-0.344	0.2319	0.002803	-0.7942	-0.345	0.1024
tau	6.167E-4	9.286E-5	1.181E-6	4.499E-4	6.12E-4	8.091E-4

DIC is now 1028.0 (with *pD* = 3.9).

2. Logistic regression

(a) *R* code and output for binomial logistic regression:

```
> attach(rats)
> resp <- cbind(y,n-y)
> logr1 <- glm(resp~t, family=binomial)
> summary(logr1)

Call:
glm(formula = resp ~ t, family = binomial)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.5943  -0.7968   0.2571   1.3268   2.4509

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.2574     0.2035   6.177 6.53e-10 ***
t             0.9259     0.3331   2.779  0.00544 **
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 92.567  on 31  degrees of freedom
Residual deviance: 84.432  on 30  degrees of freedom
AIC: 133.11

Number of Fisher Scoring iterations: 4
```

(1) Sig. Level of treatment effect = 0.005

(2) Resid. Deviance = 84.43 on 30 d.f.

(3) *AIC* = 133.1

(b) *Bayesian analysis*

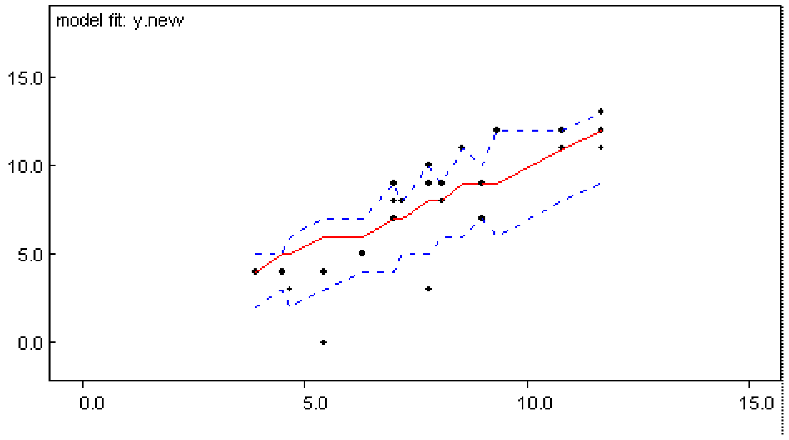
(1) posterior mean of β_1 = 0.94, and posterior SD = 0.34

(2) 95% credible interval = 0.30, 1.62

(3) *DIC* = 133.1

Treatment effect seems to be important.

Model checking for the simple logistic regression model



x-axis is **np**

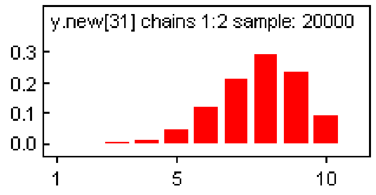
Blue lines are 95% credible interval for **y.new**

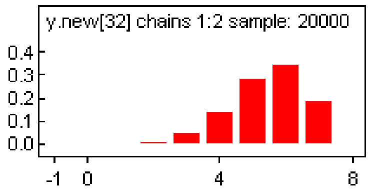
Red is the median value of **y.new**

Black dots are the data points, **y**

This graph illustrate that the model is not capturing all of the variability in the data. If it were we would not expect so many data points to be on the boundaries or outside the credible interval for **y.new**.

If we look at the distribution of **y.new** – in particular for data points 31 and 32 where y_{31}=3 and y_{32}=0. The posterior mean for **np** these two points are np_{31} = 7.787 and np_{32} = 5.451. These are the two data points below the credible interval in the plot above. The distribution of **y.new** for these two points are as below with 95% credible intervals of (5,10) for i=31 and (3,7) for i=32. As shown in the picture of above, the data are far outside the credible intervals. There are many other points where the data are on the boundary of the credible interval.





node	mean	sd	MC error	2.5%	median	97.5%
np[31]	7.787	0.3523	0.003826	7.065	7.804	8.433
np[32]	5.451	0.2466	0.002678	4.946	5.463	5.903
y.new[31]	7.785	1.359	0.01025	5.0	8.0	10.0
y.new[32]	5.468	1.116	0.008065	3.0	6.0	7.0

Extra modelling with random effects

Random effects model -

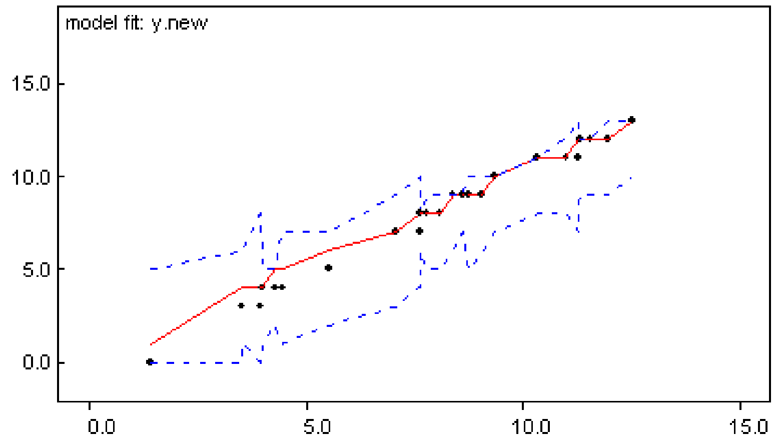
(1) posterior mean of β_1 = 1.11, and posterior SD = 0.73

(2) 95% credible interval = -0.29, 2.60

(3) *DIC* = 98.3

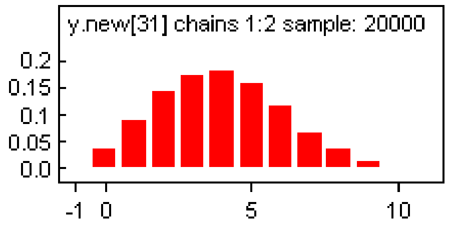
Treatment effect no longer important; prefer the random effects model (much lower *DIC*).

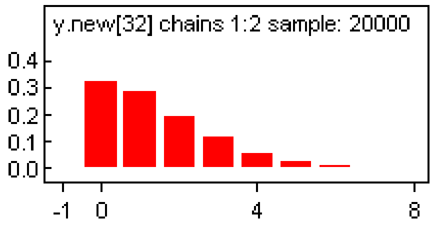
Model checking for the random effects model



In these graphs we see that there are less points are on the boundary of the credible interval. In particular by comparing the graphs on the right for the two models we see that by adding the random effects the means for the data values (**np**) have shifted left and right. The model does seem to be accounting for the overdispersion better as the data points are included in the credible interval.. Note that although there are still a number of data points on the boundary, the median values are also on the boundary. These are data points where $\mathbf{y_i = n_i}$.

node	mean	sd	MC error	2.5%	median	97.5%
np[31]	3.929	1.407	0.01147	1.399	3.874	6.767
np[32]	1.406	0.9549	0.01205	0.1154	1.223	3.646
y.new[31]	3.94	2.043	0.01564	0.0	4.0	8.0
y.new[32]	1.401	1.377	0.01348	0.0	1.0	5.0





The data y_{31} = 3 and y_{32} = 0 are now contained within the credible interval of **y.new** in each case.

Solutions to Practical 5

Sampling from a posterior using Metropolis-Hastings

This is a challenging question, but if you code this up and try to understand the output, it should give you a stronger understanding of several of the main concepts in the course.

The task is to take 10 data points, and to find the posterior distribution of the mean of a Gaussian using the data. The first thing to do is to simulate some data. Note that this is not part of the analysis – imagine that the data is the output of an experiment. You need 10 points from a normal distribution. You can choose the mean and standard deviation. The standard deviation you will treat as fixed and known, and the mean is what you will try to infer back from the data. I've chosen the mean to be zero (this is what we will try and infer) and the standard deviation to be 1. You can simulate 10 points in R using

```
> n = 10

> fixed_sd = 1

> y = rnorm(n,0,fixed_sd)
```

Now you need to choose a prior distribution on the mean. This can be whatever you want, since we are not really analysing data – we are simply trying out a method. The important thing (that is not mentioned in the question, but maybe should have been) is that the prior is a normal distribution – this is the conjugate prior and thus yields a normal distribution as the posterior. See lecture 5 for the details of this. Here I will use a prior that designed to be quite uninformative – I chose a normal distribution with mean 0 and standard deviation 10. In R I will set these parameters as follows.

```
> prior_mean = 0

> prior_sd = 10
```

The first task is to compute the true posterior. You are given the posterior (when using a conjugate normal prior) in lecture 5. The posterior parameters are found using

```
> posterior_mean = ( prior_sd^(-2)*prior_mean + n*fixed_sd^(-
2)*mean(y) ) / ( prior_sd^(-2) + n*fixed_sd^(-2) )

> posterior_sd = sqrt( 1 / ( prior_sd^(-2) + n*fixed_sd^(-2) )
)
```

Note that here I have calculated the posterior variance, then taken the square root to get the standard deviation. The only reason I have done this is that R takes the standard deviation as an argument when dealing with normal distributions, rather than the variance.

Then the posterior expectation is the expectation of the normal distribution with these parameters, which is just given by

```
> posterior_mean
```

This completes part 1. However, before we move on, let's examine your answer. Is it close to zero? You might hope it would be, since this is the "true' mean that we are trying to infer. However, this will depend on the 10 points you simulated. It is possible that it is not near zero. At this point you might also examine the maximum likelihood estimate of the mean (which you can find be taking the sample mean of the data). If you chose an uninformative prior like I did, you will see that the maximum likelihood estimate is quite close to the posterior expectation. What you are really looking for is for the posterior you have found to include zero in it main region of probability mass. Look at your posterior variance. Remember that 95% of the posterior probability mass will be within two standard deviations of the posterior mean. If you find the posterior standard deviation using

```
> posterior_sd
```

Unless you have chosen a different prior to me, you should see that zero falls into this region. If you don't understand this at the moment, hopefully the next part of the question will help visualise things.

In part 2 you simply need to simulate from the normal distribution that has the parameters you just calculated. You would like many points in order to get a good representation of this distribution – I used 10,000. To do the simulation you use

```
> monte_carlo_points =
rnorm(10000,posterior_mean,posterior_sd)
```

Then, for part 2, we need the estimate of the posterior expectation from these points. To do this take their sample mean using

```
> mean(monte_carlo_points)
```

To visualise the posterior we could find a histogram of the points. Use

```
> hist(monte_carlo_points)
```

You should see that zero falls within the main region of posterior probability mass.

Now on to part 3, which is the most challenging. You need to code up the Metropolis-Hastings algorithm, which is in lecture 8. You have all of the ingredients you need to code this up, apart from q, the proposal distribution. You are free to choose this to be whatever you like. I suggest choosing this to be a normal distribution, with mean the current point of the chain, and some suitable variance. Note that since the normal distribution is symmetrical, the qs in the acceptance ratio cancel out – this is the special case of Metropolis-Hastings called the Metropolis algorithm.

For the variance all that you need is that it is suitable for exploring the posterior distribution. In this case from the previous parts of the question you already know what the posterior distribution looks like. You should be able to see that a variance of 100 would be unnecessarily big, and a variance of 0.001 would be too small. Let's choose 2 (anything in this order of magnitude is fine – I have chosen 2 just so you can spot it in the code below).

Before giving the complete code, note that finding the acceptance probability involves evaluating the prior, which is a normal distribution, and the likelihood, which is a product of ten normal distributions (one for each data point). For this latter evaluation we use the R command "prod".

Now here is the code. You can put this in an R source file and run it.

```
num_mcmc = 10000

initial_point = 0

proposal_sd = 2

points = matrix(0,num_mcmc)

points[1] = initial_point
```

```
for(i in 2:num_mcmc)
{
    proposed_point = rnorm(1,points[i-1],proposal_sd)
    u = runif(1)
    prior_at_proposed =
dnorm(proposed_point,prior_mean,prior_sd)
    prior_at_old = dnorm(points[i-1],prior_mean,prior_sd)
    llhd_at_proposed = prod(dnorm(y,proposed_point,fixed_sd))
    llhd_at_old = prod(dnorm(y,points[i-1],fixed_sd))
    accept_prob =
min(1,prior_at_proposed*llhd_at_proposed/(prior_at_old*llhd_at
_old))
    if (u<accept_prob)
    {
        points[i] = proposed_point
    }
    else
    {
        points[i] = points[i-1]
    }
}
```

Check that it looks as through the chain has converged through a trace plot

```
> plot(points)
```

To find the posterior expectation call

```
> mean(points)
```