# Feb 22, MCMC

Stephen R. Proulx

2/1/2021

## Today's objectives:

- See the Metropolis algorithm working
- Observe what "chains" might look like (and their problems)
- Learn the `ulam` syntax

### Using `ulam` and `stan` on the cluster

There is an issue with the way the cluster is set up that we have a fix for, but sometimes requires restarting `R` again. The step is to first run the following:

```
dotR <- file.path(Sys.getenv("HOME"), ".R")
if (!file.exists(dotR)) dir.create(dotR)
M <- file.path(dotR, "Makevars")
if (!file.exists(M)) file.create(M)
cat("\nCXX14FLAGS=-O3 -march=native -mtune=native -fPIC",
    "CXX14=g++", # or clang++ but you may need a version postfix
    file = M, sep = "\n", append = TRUE)
```

And then go to Session->Restart R. You will need to run these lines of code again if the server is restarted (like when you login and it says "launching...")

### The Metropolis algorithm

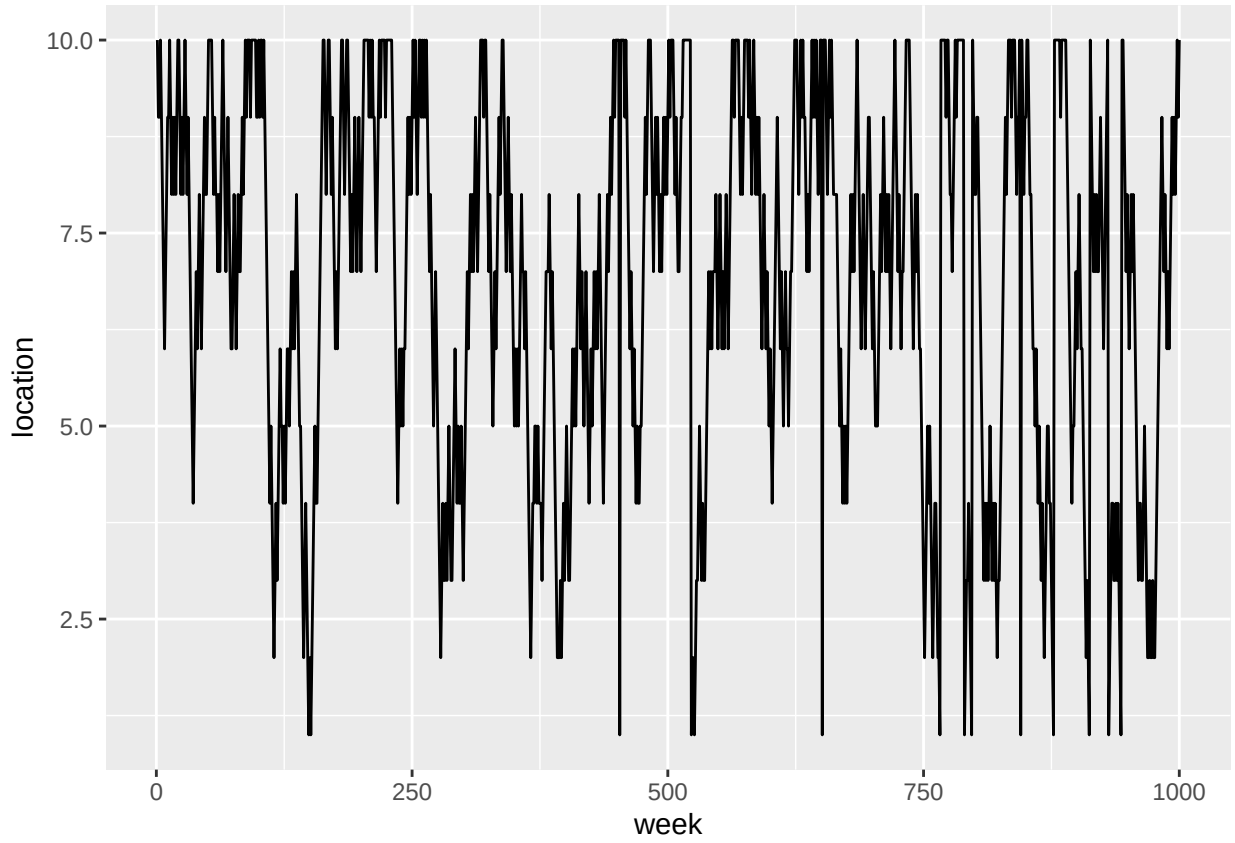
Here is the code from the book to run a simple implementation of the algorithm. It requires a function that we are trying to measure, which is the way that population size varies by island.

In this scenario, the population of the island is just proportional to it's position.

```
# tibble version
num_weeks <- 1e5
positions<- tibble(week=seq(num_weeks),location=rep(0,num_weeks))
current <- 10
for ( i in 1:num_weeks ) {
  ## record current position
    positions$location[i] <- current
  ## flip coin to generate proposal
    proposal <- current + sample( c(-1,1) , size=1 )
  ## now make sure he loops around the archipelago
```

```
    if ( proposal < 1 ) proposal <- 10
    if ( proposal > 10 ) proposal <- 1
  ## move? Here we set the probability of moving by comparing the population size of the current island
    prob_move <- proposal/current
    current <- ifelse( runif(1) < prob_move , proposal , current )
}
```

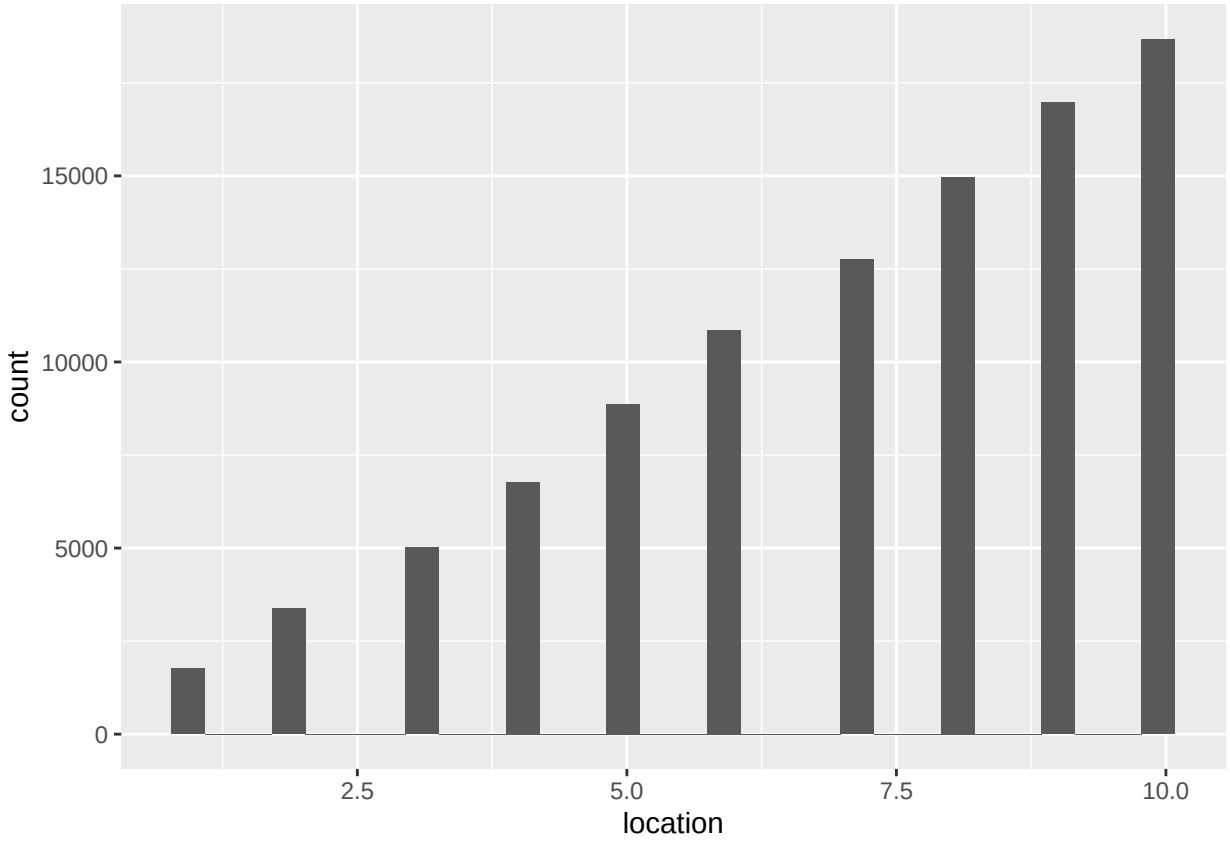
```
ggplot(data=head(positions,1000), aes(x=week,y=location))+
  geom_line()
```




We can alternatively plot the histogram of locations visited. This is what we are actually after, the density of the funtion that we are trying to match.

```
ggplot(data=positions, aes(x=location))+
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




**A more difficult archipelago to study**

Now we will set the island size explicitly. Here it has two high density islands that have low density islands between them.

```
popsize=tibble(location=seq(1,14),size=c(0.1,1,10,100,10,1,0.1,0.1,0.1,1,10,100,10,0.1))
```

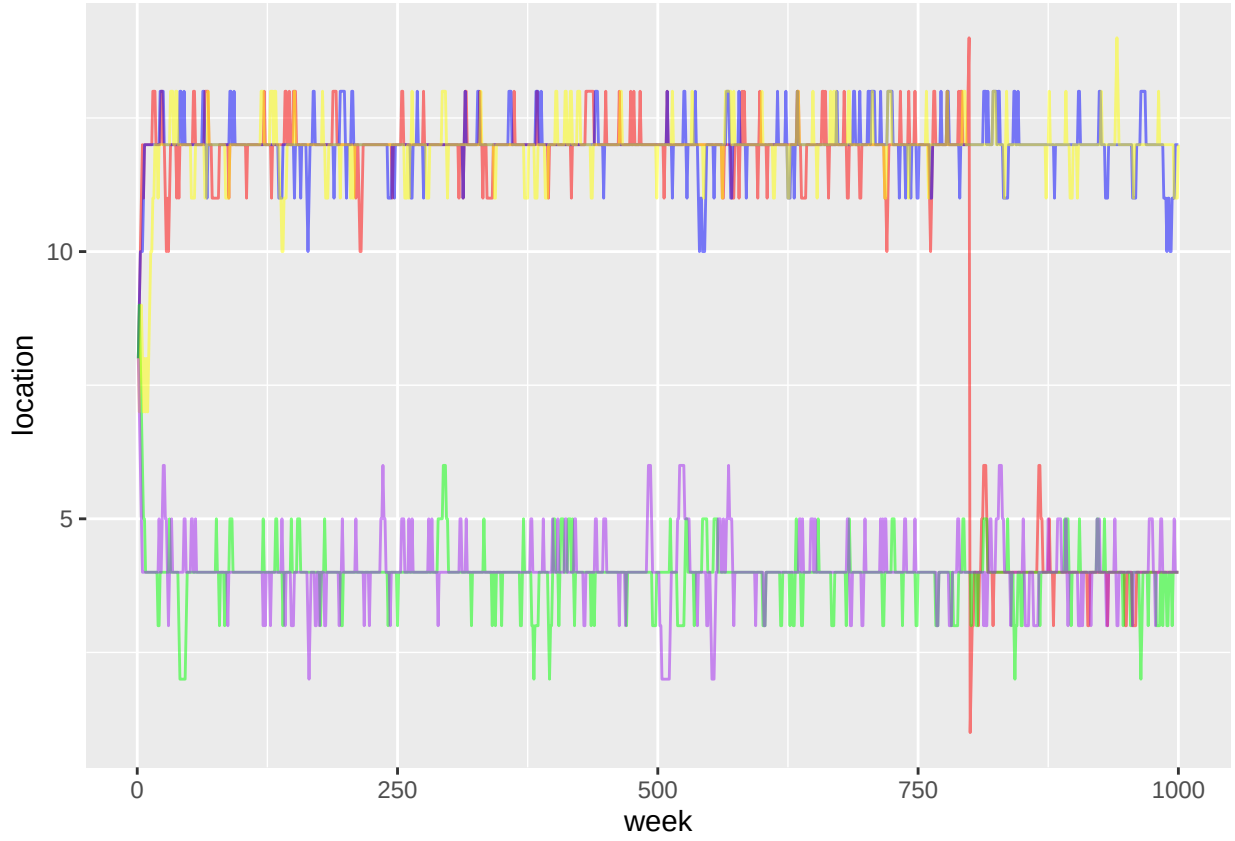
And we will run multiple Markov chains

```
num_weeks <- 1e3
reps=5
chains=list()

for(j in 1:reps){
positions<- tibble(week=seq(num_weeks),location=rep(0,num_weeks))
current <- 8
for ( i in 1:num_weeks ) {
  ## record current position
    positions$location[i] <- current
  ## flip coin to generate proposal
    proposal <- current + sample( c(-1,1) , size=1 )
  ## now make sure he loops around the archipelago
    if ( proposal < 1 ) proposal <- 14
    if ( proposal > 14 ) proposal <- 1
  ## move?
```

```
    prob_move <- popsize[popsize$location==proposal,]$size/popsize[popsize$location==current,]$size
    current <- ifelse( runif(1) < prob_move , proposal , current )
}

chains[[j]]  <- positions
}
```

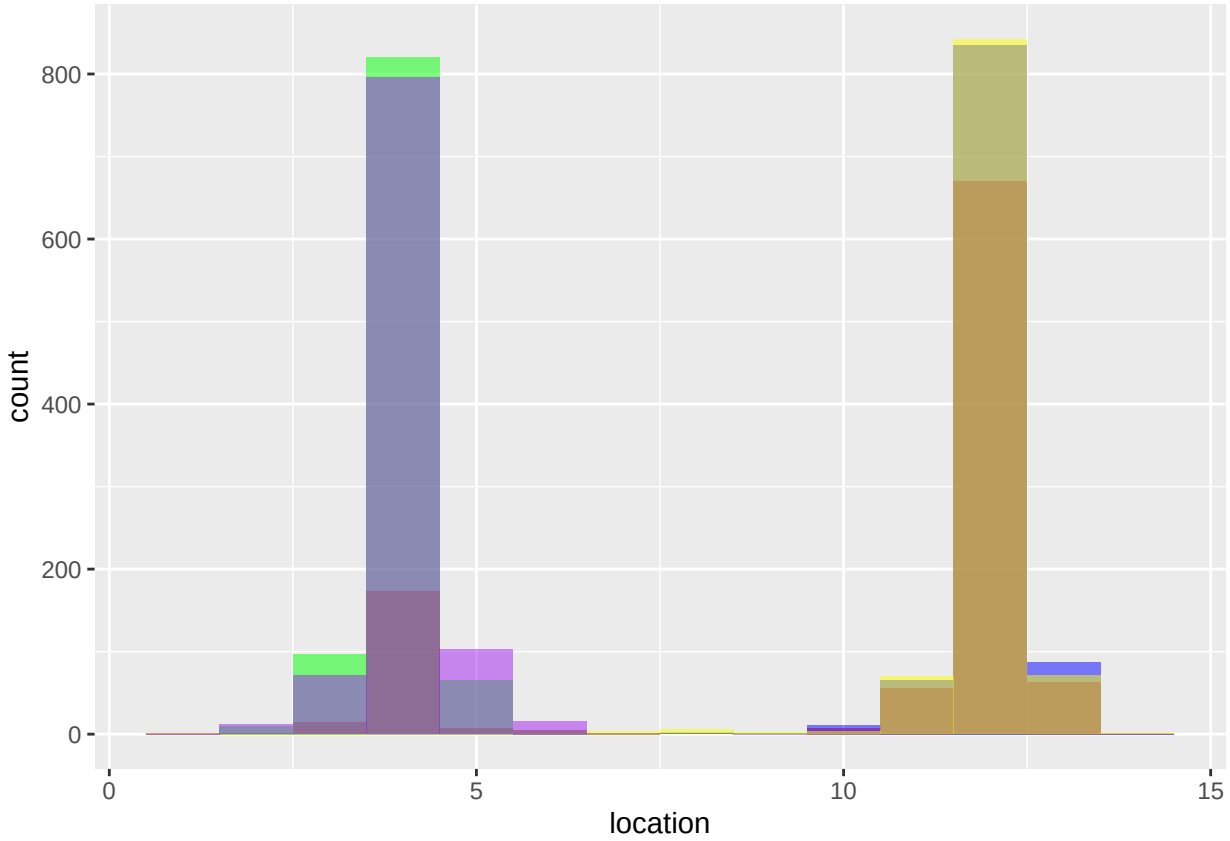
We can plot each of these paths and see how they all compare with each other.

```
ggplot(data=chains[[1]] , aes(x=week,y=location))+
  geom_line(color="red",alpha=0.5)+
  geom_line(data=chains[[2]],color="blue",alpha=0.5)+
  geom_line(data=chains[[3]],color="green",alpha=0.5)+
  geom_line(data=chains[[4]],color="yellow",alpha=0.5)+
  geom_line(data=chains[[5]],color="purple",alpha=0.5)
```




Or the histograms:

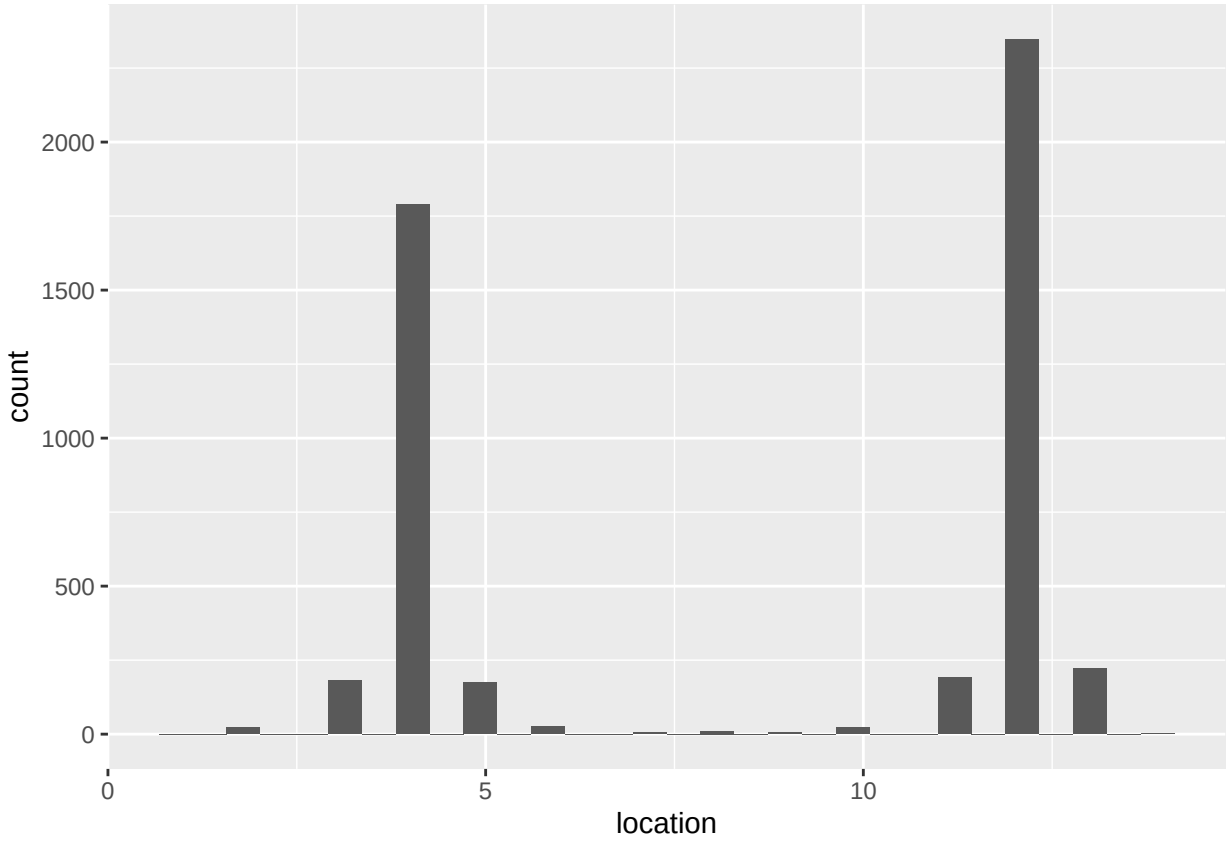
```
ggplot(data=chains[[1]] , aes(x=location))+
  geom_histogram(bins = 14,fill="red",alpha=0.5)+
  geom_histogram(data=chains[[2]],bins = 14,fill="blue",alpha=0.5)+
  geom_histogram(data=chains[[3]],bins = 14,fill="green",alpha=0.5)+
  geom_histogram(data=chains[[4]],bins = 14,fill="yellow",alpha=0.5)+
  geom_histogram(data=chains[[5]],bins = 14,fill="purple",alpha=0.5)
```




And now we'll put the chains together and summarize them:

```
combined_chains=chains[[1]]
for(i in 2:5){
  combined_chains=bind_rows(combined_chains,chains[[i]])
}

ggplot(data=combined_chains , aes(x=location))+
  geom_histogram( )
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




Did combining the chains solve our problem? Not entirely.

#### Try your own surface

Make up your own "population size surface". You can make it longer or shorter, just make sure to change the part of the program that tracks what to do when we get a proposal above the number of islands in your list.

## `ulam` syntax

`ulam` uses syntax that is based on the same principles as `qaup`. The main difference is that we can specify how many separate chains we want to run and how many samples to generate from that chain.

Let's redo our analyisis from last time.

First the data

```
data(rugged)
d <- as_tibble(rugged)


# make log version of outcome and  extract countries with GDP data
d <- d %>%
  mutate(log_gdp = log(rgdppc_2000))%>%
  drop_na(rgdppc_2000) %>%
  mutate(log_gdp_std=log_gdp/mean(log_gdp), rugged_std = rugged/max(rugged)) %>%
```

```
  mutate(rugged_std=rugged_std - mean(rugged_std))%>%
  select(log_gdp,rugged,log_gdp_std,rugged_std,country, cont_africa)
```

And now the `quap` model:

```
m.q.7.5 <-
  quap(alist(
    log_gdp  ~ dnorm( mu , sigma ),
    mu <- anA*(1-cont_africa) + aA* cont_africa  + brnA*rugged_std*(1-cont_africa) + brA *rugged_std* co
   anA  ~ dnorm(1,1),
    aA  ~ dnorm(1,1),
    brnA  ~ dnorm(0,1),
    brA  ~ dnorm(0,1),
    sigma ~ dexp(1)
  ), data=d )
```

And the `ulam` version

```
m.u.7.5 <-
  ulam(alist(
    log_gdp  ~ dnorm( mu , sigma ),
    mu <- anA*(1-cont_africa) + aA* cont_africa  + brnA*rugged_std*(1-cont_africa) + brA *rugged_std* co
   anA  ~ dnorm(1,1),
    aA  ~ dnorm(1,1),
    brnA  ~ dnorm(0,1),
    brA  ~ dnorm(0,1),
    sigma ~ dexp(1)
  ), data=d ,  chains=4, cores=4 , iter=3000 )
```

```
## Trying to compile a simple C file

## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Library/Frameworks/R.frame
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
##                 ^
##                 ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/3.6/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

Check the summaries of the output.

```
precis(m.u.7.5)
```

```
##             mean         sd      5.5%      94.5%    n_eff     Rhat4
## anA    8.8895860 0.08659603  8.752666  9.0278550 8108.519 0.9995293
## aA     7.3970808 0.13930466  7.174108  7.6234958 7642.947 0.9996245
## brnA  -0.9962747 0.42878610 -1.677510 -0.3058857 8855.588 1.0002384
## brA    0.7528313 0.55339119 -0.134944  1.6361344 8346.650 0.9998544
## sigma  0.9520969 0.05277652  0.870589  1.0389228 7478.804 1.0001566
```

```
precis(m.q.7.5)
```

```
##             mean         sd       5.5%      94.5%
## anA    8.8920287 0.08519796  8.7558659  9.0281915
## aA     7.3996920 0.13451438  7.1847120  7.6146720
## brnA  -1.0047547 0.43011500 -1.6921616 -0.3173479
## brA    0.7582566 0.54841334 -0.1182138  1.6347271
## sigma  0.9352938 0.05101991  0.8537541  1.0168334
```

They are more or less the same, we have two new columns, `n_eff` and `Rhat`.

`n_eff` is a measure of how many independnet samples fromthe posterior we have. It is an attempt to correct for correlations in the data. If `n_eff` is small compared to the number of iterations you ran, there may be a problem. Here we ran 3000*4 iterations, but half of them were in the warmup phase and not included. So we expect something like 6000 samples.

`Rhat` is a measure of how much agreement between chains there is, and is supposed to tell you that the Markov chains have "converged". When this number is above 1 then there may be a problem. In practice it is often between 1 and 1.1.