Updated ages on the Fury and Hecla and Thule dykes and sills and age estimate for the onset of Sturtian glaciation

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Execute the code by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

Set up by closing R's graphics windows and clearing R's memory

Next we will install packages we will need and define a few functions used later in the code.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(scales)

# This function generates a median age and the plus and minus uncertainties using quantiles.This functi

age_calc <- function(agesamples, prob=0.95){
  q <- quantile(agesamples, c(0.5, 0.025, 0.975))
  boundarymedian <- q[1]
  boundaryplus <- q[3]-q[1]
  boundaryminus <- q[1]-q[2]
  return(as.vector(c(boundarymedian, boundaryplus, boundaryminus)))
}

# This function generates text describing the age and uncertainty in terms of the familiar +/- conventi

age_print <- function(params){
  params = params
  age <- paste(round(params[1],2) %>% as.character, "+", round(params[2],2) %>% as.character, "/-", rou
```

```
  return(age)
}
```

We generate the dataframe used by the code here in order to avoid the issue of co-locating a data file and this .Rmd file.

```
#all data are from Macdonald et al. (2018, Precambrian Research)
means <- data.frame('Rank'=1:5, 'Id' = c('F837A', 'F837C', 'F837B', 'F917-1', 'F840A'),
                    age = c(717.85, 717.68, 717.43, 716.94, 716.47), 'age_err95' =
                      c(0.24,0.31,0.14,0.23,0.23))
means
```

```
##   Rank     Id    age age_err95
## 1    1  F837A 717.85      0.24
## 2    2  F837C 717.68      0.31
## 3    3  F837B 717.43      0.14
## 4    4 F917-1 716.94      0.23
## 5    5  F840A 716.47      0.23
```

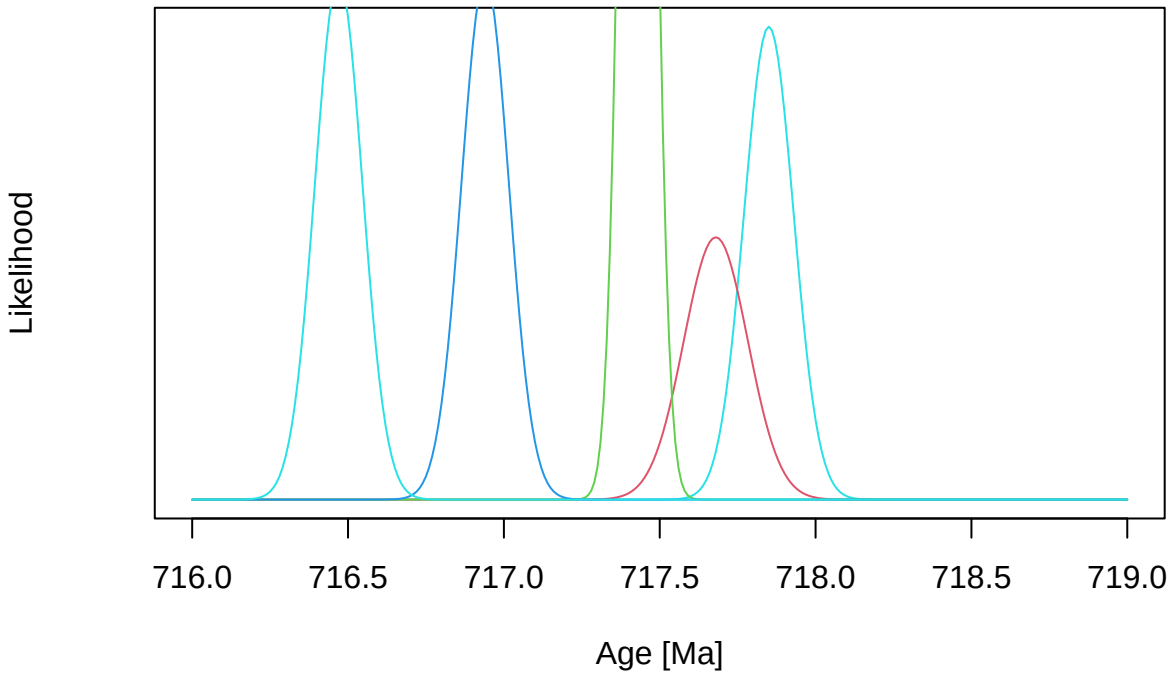
```
dur <- length(means$age)
# We need to convert 95% uncertainty to 1-sigma for the purpose of this analysis since it will be done
sd_corr <- 95.45/95
```

Now let's define the likelihood function, assuming the ages are all normal distributions.

```
ln_likelihoods <- function(t){
  ll <- matrix(0,dur,1)
  for (s in 1:dur){
      ll[s] <- dnorm(t[s],mean=means$age[s],sd=(means$age_err95[s])*sd_corr/2, log=T)
    }
  return(ll)
}
```

We will also plot the likelihood functions.

```
age_axis <- seq(716,719, by=0.01) # Changes based on data set
len <- length(age_axis)
ageAxis <- matrix(0,dur,len)

for(i in 1:len){
  ageAxis[,i] <- age_axis[i]
}


ln_ages <- matrix(0,dur,len)

for(i in 1:len) {
  ln_ages[,i] <- 10^(ln_likelihoods(ageAxis[,i]))
}

  plot(age_axis,ln_ages[1,], type="l",col=i,xlab="Age [Ma]", ylab="Likelihood", yaxt = "n")

for (i in 2:dur){

    lines(age_axis,ln_ages[i,], col=i)
  }
```



Now we will start to set up our MCMC. First we need to define the likelihood function taking into account all of the ages.

```
ln_likelihood = function(params){
  sumll <- sum(ln_likelihoods(params))
  return(sumll)
}
```

Our prior function (1 if the samples obey superposition, 0 if not)

```
ln_prior <- function(params){
  for (i in seq(1,length(params)-1)){
    t_prior<- params[i]
    t_next<-params[i+1]
    if (t_prior>t_next){
      if (i==(dur-1)){
        ln_pr <- log(1)
      }else{
        t_prior <- t_next
        next
      }
    } else{
      ln_pr<-log(0)
    }

  return(ln_pr)
  }
}
```

```
# This is to test whether or not the post function is working.
ln_prior(c(753, 745, 740, 732, 730))
```

```
## [1] 0
```

Next the posterior will be calculated: the posterior = likelihood x prior *(but add them since they are the log*

values)

```
ln_post <- function(params){
  post <- ln_likelihood(params) + ln_prior(params)
  return(post)
}
```

Create the proposal function.

```
proposalfunction <- function(param){
  # The sd values for the proposal function must be calculated depending on the age type
  # We will just use 0.05 m.y. for each
  sd_vals <- rep(0.05, dur)
  means <- means[order(means$age,decreasing=TRUE),]
  return(rnorm(dur,mean = param, sd= sd_vals))
}
```

And now for the MCMC function, which will store the posterior values in chains (1 for each age)

```
run_metropolis_MCMC <- function(startvalue, iterations){
  chain = array(dim = c(iterations+1,dur))
  chain[1,] = startvalue
  for (i in 1:iterations){
    proposal = proposalfunction(chain[i,])
    probab = exp(ln_post(proposal) - ln_post(chain[i,]))
    if (runif(1) < probab){
      chain[i+1,] = proposal
    }else{
      chain[i+1,] = chain[i,]
    }
  }
  return(chain)
}
```

Now to set our start values and start the chain. Retry as needed. If it fails, it is because the start samples don't obey superposition.

```
startval <- rnorm(5, mean=means$age, sd=means$age_err95/4)

chain = run_metropolis_MCMC(startval, 30000)

burnIn = 1000
acceptance = 1-mean(duplicated(chain[-(1:burnIn),]))
```

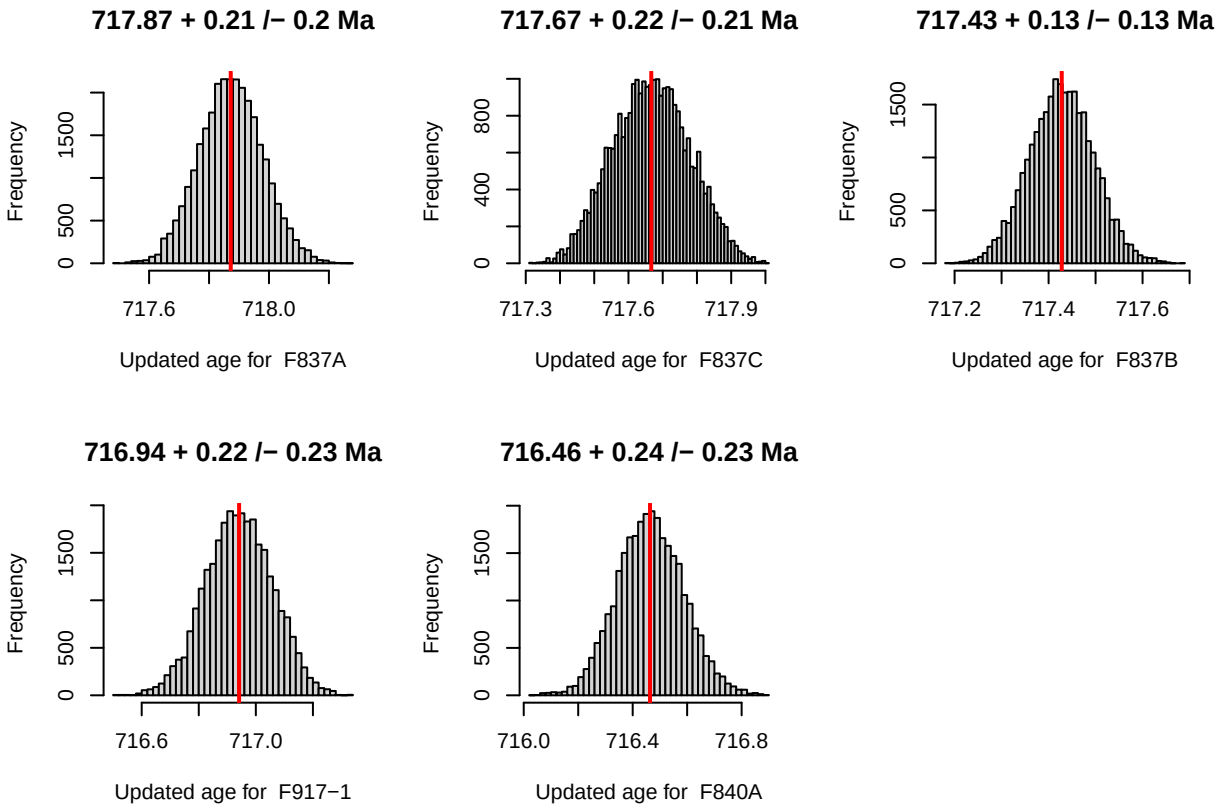
#Summary calculations of the posterior distributions:

```
  yukon_updated <- data.frame(matrix(nrow=dur,ncol=3))
  updated <- vector()
for (i in 1:dur){
  yukon_updated[i,] <- age_calc(chain[-(1:burnIn),i])
  updated[i] <- age_print(yukon_updated[i,])
}
```

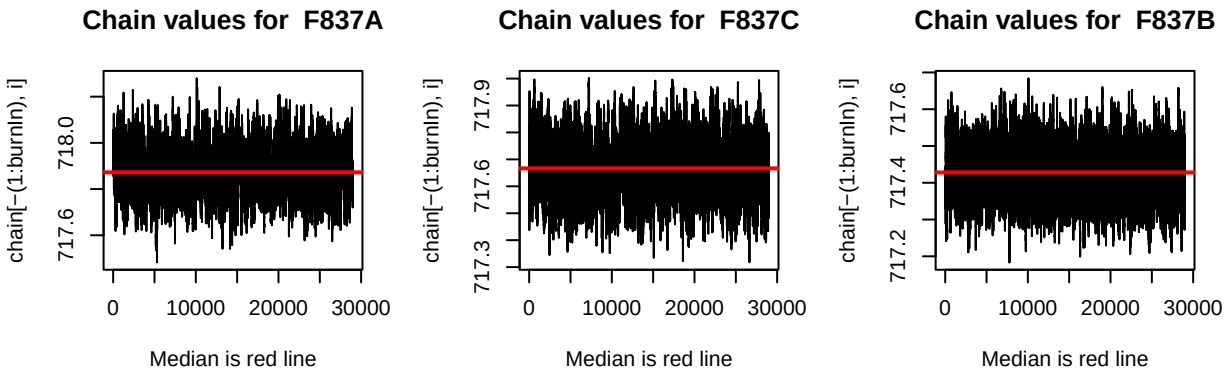
Now to plot the results

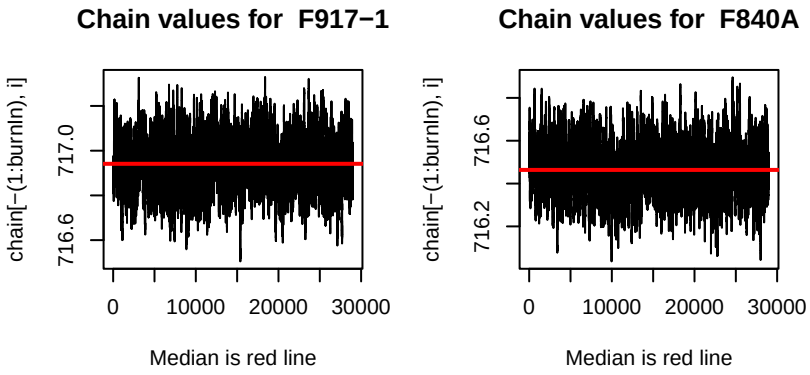
```
#histograms
par(mfrow = c(2,3))

for (i in 1:dur){
```

```
    hist(chain[-(1:burnIn),i],nclass=50, main=updated[i], xlab=paste("Updated age for ", means$Id[i]))
    abline(v = yukon_updated[i,1], col="red", lwd=2)
}

#chain plots
par(mfrow = c(2,3))
```

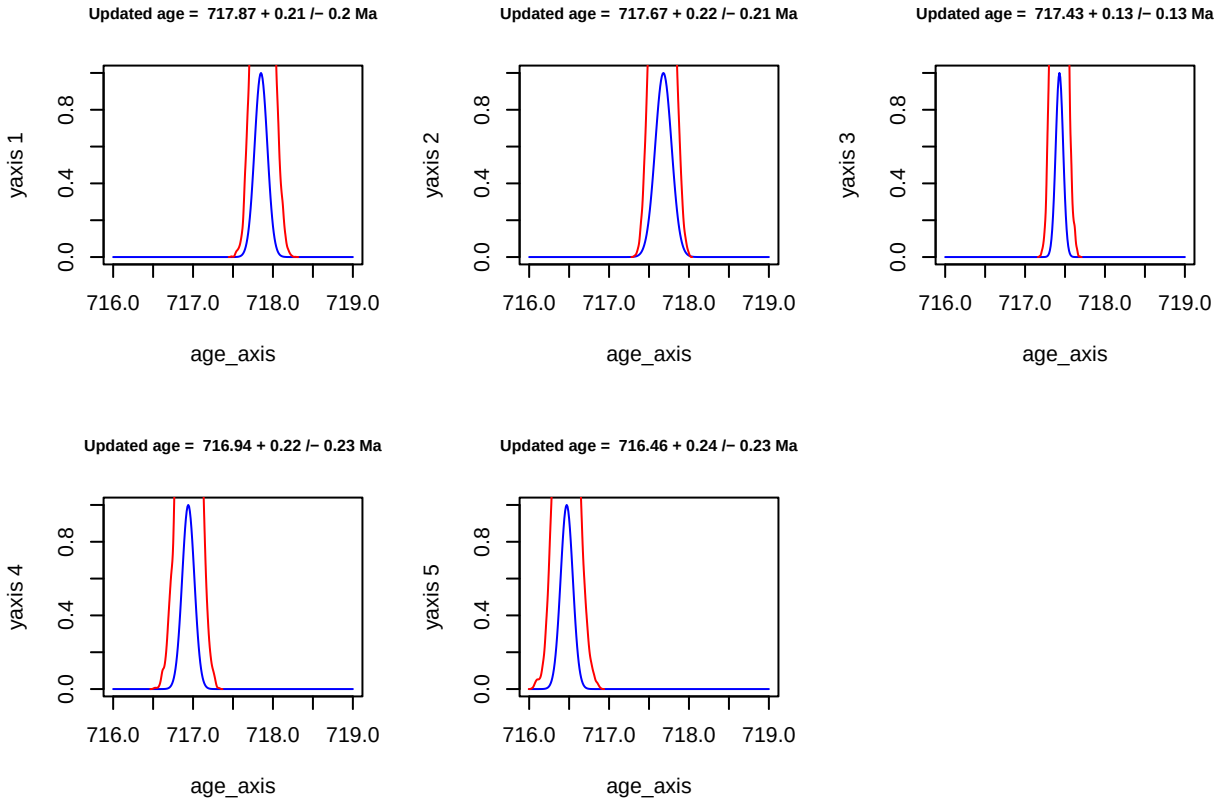



```
for (i in 1:dur){
    plot(chain[-(1:burnIn),i], type = "l", xlab="Median is red line" , main = paste("Chain values for "
    abline(h = yukon_updated[i,1], col="red", lwd=2)
}

#density plots
par(mfrow = c(2,3))
```






```
for (i in 1:dur){
   plot(age_axis, ln_ages[(i),]/max(ln_ages[(i),]), ylim=c(0,1), type="l", col="blue", main=paste("Upda
#dens(chain[-(1:burnIn),step], col="red", norm.comp=FALSE, add=TRUE)
lines(density(chain[-(1:burnIn),i]), col="red")
}
par(mfrow = c(1,1))
```



Now we will use the results from the two updated ages above and below the boundary to estimate the age of the boundary. To do this, we will run a MC simulation, sampling from each age distribution n times. Each time we have an age distribution that obeys stratigraphic superposition, we will simply take a random sample from a uniform distribution between the two ages (above and below).

```
#Note that we are assuming a normal distribution for these two update ages for the purpose of resamplin
upperAge <- yukon_updated[4,1] # This is the uppermost pre-glacial age
upperSd <- (yukon_updated[4,2]+yukon_updated[4,3])*sd_corr/4 # And this is its standard deviation
lowerAge <- yukon_updated[3,1] # This is the lowermost post-glacial onset age
lowerSd <- (yukon_updated[3,2]+yukon_updated[3,3])*sd_corr/4 # And this is its standard deviation


sampnum <- 500000 # number of random samples for each age
# Sample from a normal distribution of each
upper_sample <- rnorm(sampnum, mean = upperAge, sd = upperSd) # generate random samples
lower_sample <- rnorm(sampnum, mean = lowerAge, sd = lowerSd)
```

Now to test for superposition and generate a sample of viable ages for the boundary

```
n <- sum(upper_sample < lower_sample) # calculate number of samples that do not contradict superpositio
c <- 1
boundaryage <- vector(mode="numeric", length=n)

for ( i in 1:sampnum ) {
  if (upper_sample[i] < lower_sample[i]) {
   boundaryage[c] <- runif(1, min=upper_sample[i], max=lower_sample[i])
  c <- c + 1
  }
}
# To save space, we'll now eliminate the original random samples
```

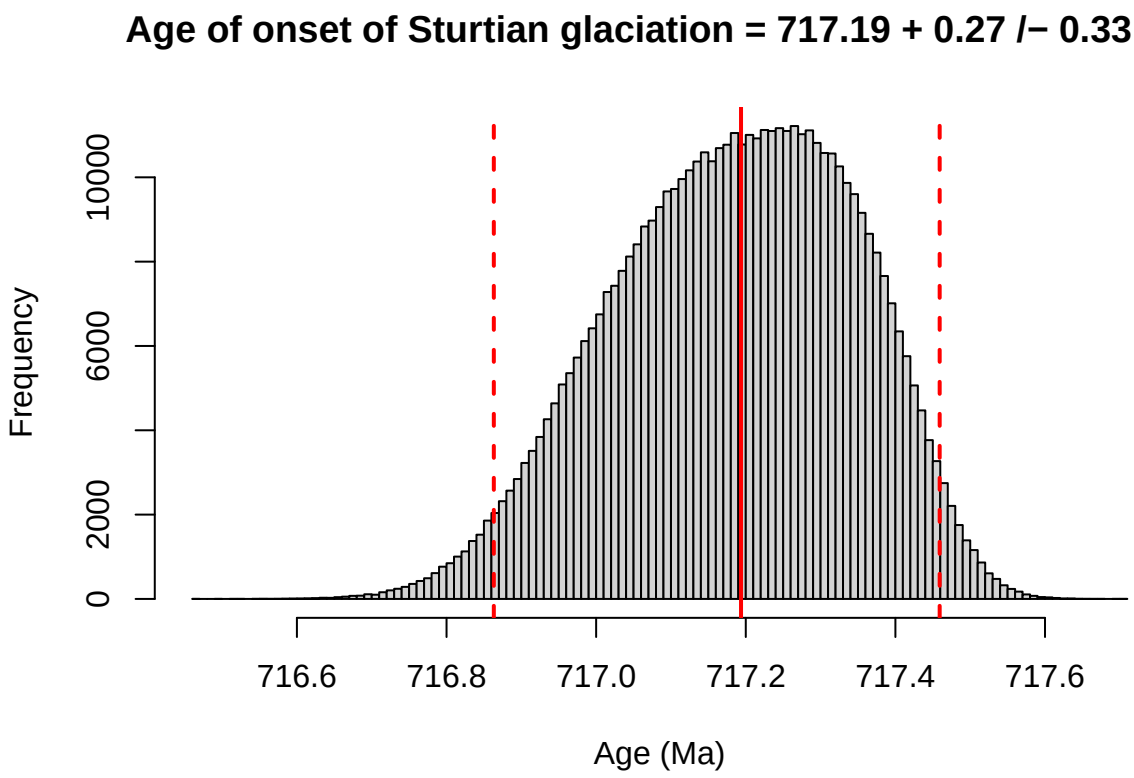
```
rm(upper_sample, lower_sample)
```

And now we calculate the summary statistics

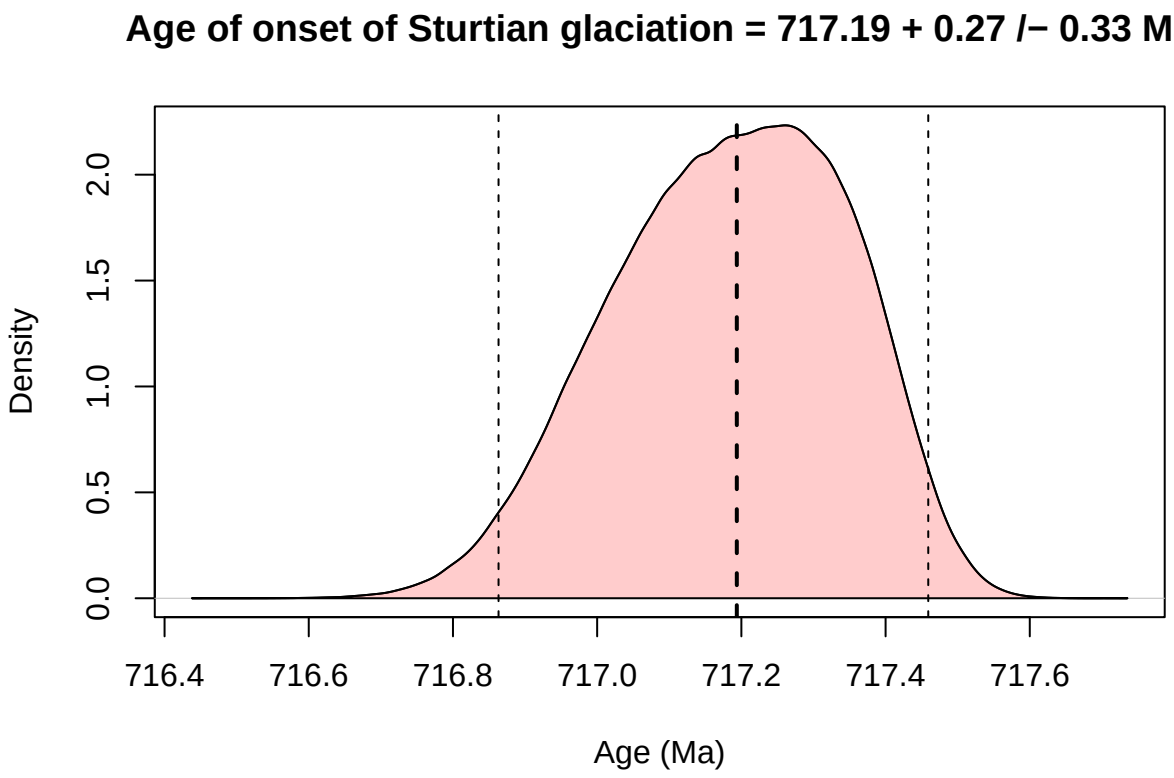
```
# First, extract the median and the 2.5% and 97.5% quantiles
q <- quantile(boundaryage, c(0.5, 0.025, 0.975))
boundary_median <- q[1]
boundaryminus <- q[1]-q[2]
boundaryplus <- q[3]-q[1]
boundary <- paste(round(boundary_median,2) %>% as.character, "+", round(boundaryplus,2) %>% as.characte
```

And we plot the results

```
hist(boundaryage, nclass=100, xlab="Age (Ma)", main=paste("Age of onset of Sturtian glaciation =", boun
abline(v=q[1], col="red", lwd=2, lty=1)
abline(v=q[2], col="red", lwd=2, lty=2)
abline(v=q[3], col="red", lwd=2, lty=2)
```




```
plot(density(boundaryage), xlab="Age (Ma)", main=paste("Age of onset of Sturtian glaciation =", boundary
polygon(density(boundaryage), col=alpha("red", 0.2))
abline(v = boundary_median, col="black", lwd=2, lty=2)
abline(v = q[2], col="black", lwd=1, lty=2)
abline(v = q[3], col="black", lwd=1, lty=2)
```



Now to update the ages of the dykes and a sills. This will be done without using MCMC, because it is not in fact required here—it can be done by a Monte Carlo method where we simply eliminate combinations of samples that disobey crosscutting relationships.

We will start by updating the age of the dykes and sills in each location using cross-cutting constraints

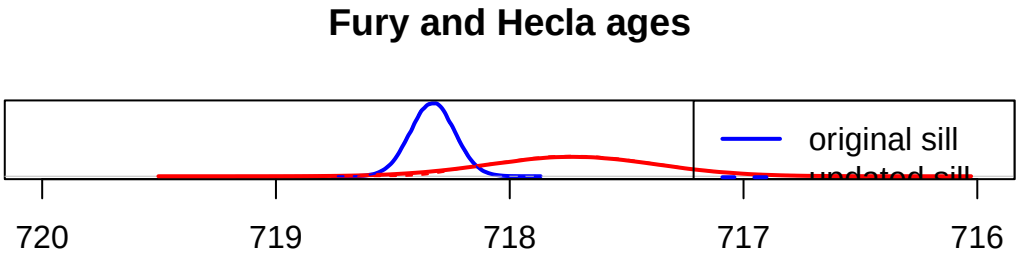
```
# First for the Fury and Hecla basin
FHsill_mean <- 718.33
FHsill_sd <- 0.19/2
FHdyke_mean <- 717.73
FHdyke_sd <- 0.72/2
sampnum <- 300000 # number of random samples for each age
# Sample from a normal distribution of each
FHsill_sample <- rnorm(sampnum, mean = FHsill_mean, sd = FHsill_sd) # generate random samples
FHdyke_sample <- rnorm(sampnum, mean = FHdyke_mean, sd = FHdyke_sd)

# now to select those samples that obey cross-cutting constraint
n <- sum(FHsill_sample > FHdyke_sample) # calculate number of samples that do not contradict superposit
c <- 1
FHsill_new <- vector(mode="numeric", length=n)
FHdyke_new <- vector(mode="numeric", length=n)
for ( i in 1:sampnum ) {
  if (FHsill_sample[i] > FHdyke_sample[i]) {
    FHsill_new[c] <- FHsill_sample[i]
    FHdyke_new[c] <- FHdyke_sample[i]
    c <- c + 1
  }
}

# And finally to calculate the summary statistics
# And calculate the summary statistics and generate a print label for the figures at the same time
```

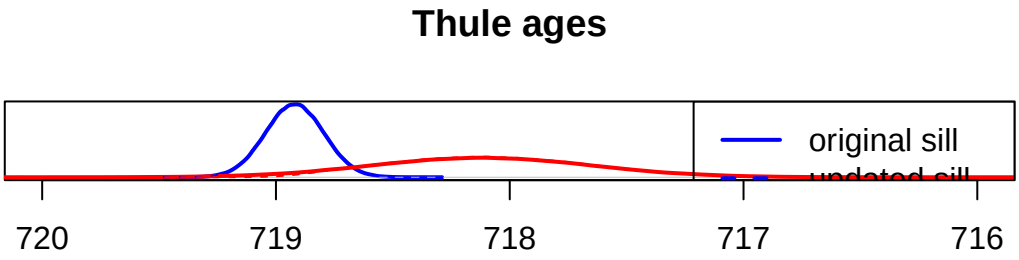
```
FHsill_age <- age_calc(FHsill_new)
FHsill_label <- age_print(FHsill_age)
FHsill_label <- paste("Fury & Hecla sill: ", FHsill_label)

FHdyke_age <- age_calc(FHdyke_new)
FHdyke_label <- age_print(FHdyke_age)
FHdyke_label <- paste("Fury & Hecla dyke: ", FHdyke_label)
```

Now for the Thule basin

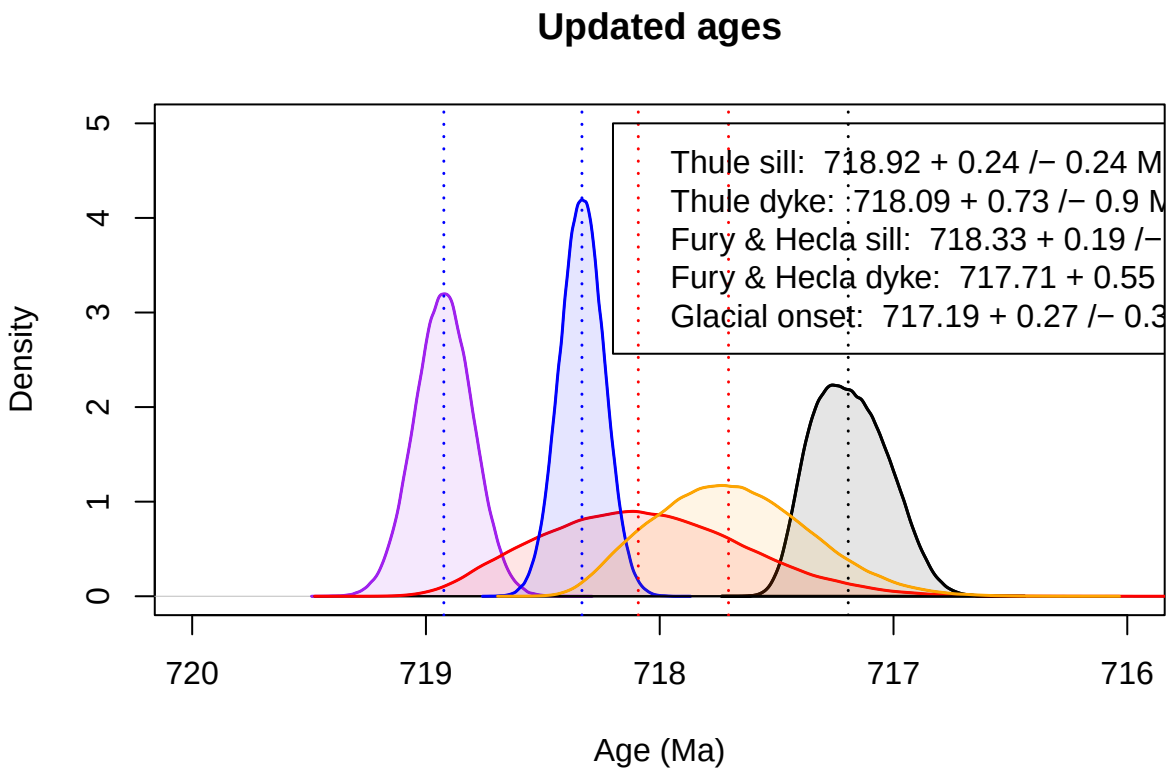
```
# First for the Fury and Hecla basin
THsill_mean <- 718.92
THsill_sd <- 0.25/2
THdyke_mean <- 718.12
THdyke_sd <- 0.94/2
sampnum <- 300000 # number of random samples for each age
# Sample from a normal distribution of each
THsill_sample <- rnorm(sampnum, mean = THsill_mean, sd = THsill_sd) # generate random samples
THdyke_sample <- rnorm(sampnum, mean = THdyke_mean, sd = THdyke_sd)

# now to select those samples that obey cross-cutting constraint
n <- sum(THsill_sample > THdyke_sample) # calculate number of samples that do not contradict superposit
c <- 1
THsill_new <- vector(mode="numeric", length=n)
THdyke_new <- vector(mode="numeric", length=n)
for ( i in 1:sampnum ) {
  if (THsill_sample[i] > THdyke_sample[i]) {
    THsill_new[c] <- THsill_sample[i]
    THdyke_new[c] <- THdyke_sample[i]
    c <- c + 1
  }
}

# And calculate the summary statistics and generate a print label for the figures at the same time
THsill_age <- age_calc(THsill_new)
THsill_label <- age_print(THsill_age)
THsill_label <- paste("Thule sill: ", THsill_label)

THdyke_age <- age_calc(THdyke_new)
THdyke_label <- age_print(THdyke_age)
THdyke_label <- paste("Thule dyke: ", THdyke_label)
```

Now we'll plot these up to compare them.





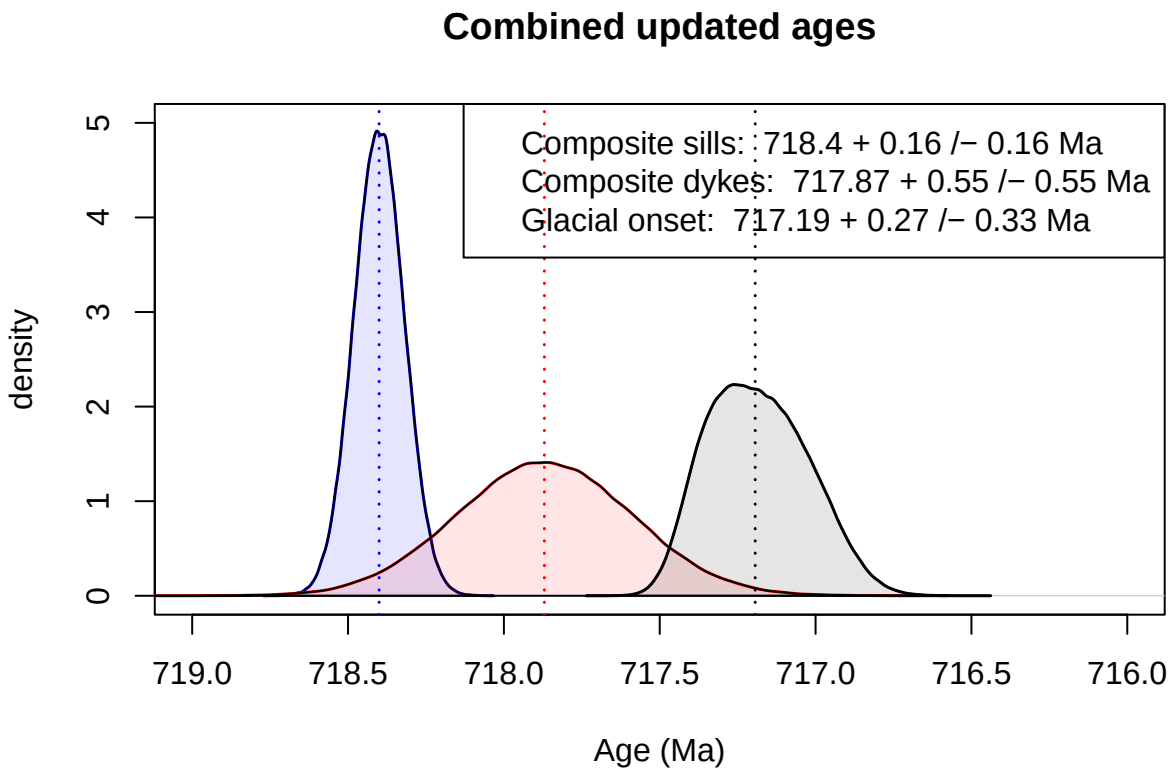

```
par(mfrow=c(1,1))
# first, let's make some labels for the legend
boundary_label <- paste("Glacial onset: ", boundary)

# first, we'll plot the sills


plot(density(boundaryage), col=alpha("black"), lwd=1.5, xlim=c(720, 716), ylim=c(0,5), main="Updated ag
polygon(density(boundaryage), col=alpha("black", 0.1), lwd=1.5)
polygon(density(THsill_new), col=alpha("purple", 0.1))
lines(density(THsill_new), col="purple", lwd=1.5)
polygon(density(THdyke_new), col=alpha("red", 0.1))
lines(density(THdyke_new), col="red", lwd=1.5)
polygon(density(FHsill_new), col=alpha("blue", 0.1))
lines(density(FHsill_new), col="blue", lwd=1.5)
polygon(density(FHdyke_new), col=alpha("orange", 0.1))
lines(density(FHdyke_new), col="orange", lwd=1.5)


# add the median lines
abline(v = THsill_age[1], col="blue", lwd=1.5, lty=3)
abline(v = THdyke_age[1], col="red", lwd=1.5, lty=3)
abline(v = FHsill_age[1], col="blue", lwd=1.5, lty=3)
abline(v = FHdyke_age[1], col="red", lwd=1.5, lty=3)
abline(v = boundary_median[1], col="black", lwd=1.5, lty=3)
legend(x=718.2, y=5, c(THsill_label, THdyke_label, FHsill_label, FHdyke_label, boundary_label))
```



Now we'll compared the combined ages (weighed means) of the dykes and sills with the onset of the boundary

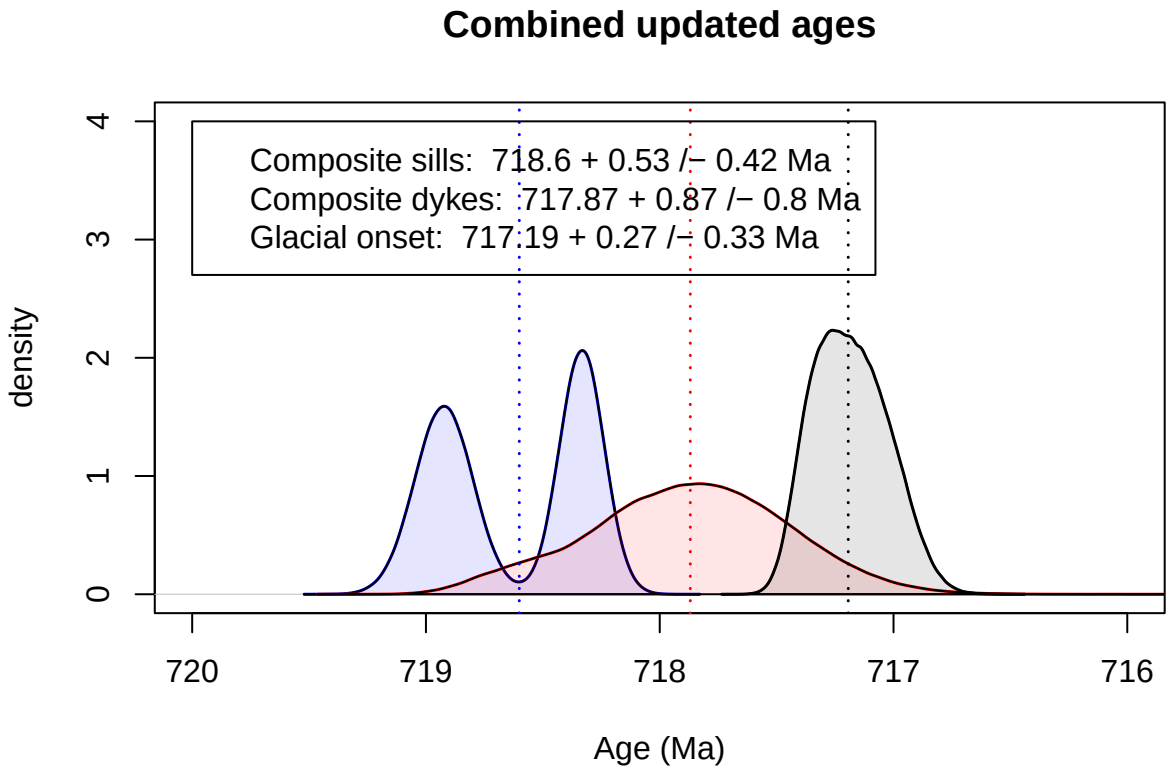
```
combined_sills_mean <- 718.40
combined_sills_sd <- 0.16*sd_corr/2
combined_dykes_mean <- 717.87
combined_dykes_sd <- 0.56*sd_corr/2

sills_distribution <- rnorm(200000, mean = combined_sills_mean, sd = combined_sills_sd)
dykes_distribution <- rnorm(200000, mean = combined_dykes_mean, sd = combined_dykes_sd)
sills_age <- age_calc(sills_distribution)
sills_label <- age_print(sills_age)
dykes_age <- age_calc(dykes_distribution)
dykes_label <- age_print(dykes_age)
sills_label <- paste("Composite sills: ", sills_label)
dykes_label <- paste("Composite dykes: ", dykes_label)

plot(density(sills_distribution), col="blue", xlim=c(719, 716), ylim=c(0,5), lwd=1.5, main="Combined up
polygon(density(sills_distribution), col=alpha("blue", 0.1))

lines(density(dykes_distribution), col="red", lwd=1.5)
polygon(density(dykes_distribution), col=alpha("red", 0.1))


lines(density(boundaryage), col="black", lwd=1.5)
polygon(density(boundaryage), col=alpha("black", 0.1))

abline(v = sills_age[1], col="blue", lwd=1.5, lty=3)
abline(v = dykes_age[1], col="red", lwd=1.5, lty=3)
abline(v = boundary_median, col="black", lwd=1.5, lty=3)
```

```
legend(x='topright', c(sills_label, dykes_label, boundary_label))
```



Now we'll with combined ages where we combined the updated sill and dyke ages

```
sills_age <- age_calc(c(THsill_new, FHsill_new))
sills_label <- age_print(sills_age)
dykes_age <- age_calc(c(THdyke_new, FHdyke_new))
dykes_label <- age_print(dykes_age)
sills_label <- paste("Composite sills: ", sills_label)
dykes_label <- paste("Composite dykes: ", dykes_label)

plot(density(c(THsill_new, FHsill_new)), col="blue", xlim=c(720, 716), ylim=c(0,4), lwd=1.5, main="Combi
polygon(density(c(THsill_new, FHsill_new)), col=alpha("blue", 0.1))

lines(density(c(THdyke_new, FHdyke_new)), col="red", lwd=1.5)
polygon(density(c(THdyke_new, FHdyke_new)), col=alpha("red", 0.1))


lines(density(boundaryage), col="black", lwd=1.5)
polygon(density(boundaryage), col=alpha("black", 0.1))

abline(v = sills_age[1], col="blue", lwd=1.5, lty=3)
abline(v = dykes_age[1], col="red", lwd=1.5, lty=3)
abline(v = boundary_median, col="black", lwd=1.5, lty=3)

legend(x=720, y=4, c(sills_label, dykes_label, boundary_label))
```


Combined updated ages
Composite sills: 718.6 + 0.53 /− 0.42 Ma
Composite dykes: 717.87 + 0.87 /− 0.8 Ma
Glacial onset: 717.19 + 0.27 /− 0.33 Ma
density
0
1
2
3
4
720
719
718
717
716
Age (Ma)