Using Approximate Bayesian Computation for parameter estimation in the cue-based retrieval model (vague priors)

Shravan Vasishth

6/25/2019

Introduction

Load interact

The following piece of code, written by Felix Engelmann and available on github, provides the main computation code for ACT-R calculations.

```
source("interACT.R")
```

Basic engine for generating predictions

Set up priors:

```
a<-2
b<-6
```

```
printcounts<-FALSE

iterate_lf <- function(values,iterations=1000){
  ## values is a scalar or vector containing an lf value or values.
  ## iterations is the number of iterations for that given value.
  ## We need multiple iterations as noise is non-zero and there will be some
  ## variability due to noise.
  maxset <- 0
  means <- NULL
  for(v in values){
    lf <<- v
    pmatr <- create_param_matrix(model_4cond, iterations)
    results <- run(pmatr)
    means2 <- compute_int_means(results)
    means2$Set <- means2$Set+maxset
    means <- bind_rows(means, means2)
  }
  means
}

## set the parameters:
reset_params()
psc <<- 0
qcf <<- 0
cuesim <<- -1
bll <<- 0.5
```

```r
## default in Engelmann et al 2019 Cog Sci paper
mp <<- 0.15
## default in Engelmann et al 2019 Cog Sci paper
mas <<- 1.5 ## could change this to a random starting value:
            ## mas <- runif(1,min=1,max=2)
            ## mas<<-sort(rnorm(50,mean=1.5,sd=0.25))
# default in Engelmann et al 2019 Cog Sci paper
ans <<- 0.2
# default in Engelmann et al 2019 Cog Sci paper
rth <<-  -1.5
dbl <<- 0
cueweighting <<- 1
```

Now generate one run (1000 iterations) with lf as the parameter to be estimated using ABC:

```r
## using a large lf value:
means <- iterate_lf(values=0.4)
## grammatical: inhibitory interference effect
means$Effect[1]
```

```
## [1] 36.884
```

```r
## ungrammatical: facilitatory interference effect
means$Effect[2]
```

```
## [1] -47.643
```

```r
# using small lf value:
means <- iterate_lf(values=0.1)
## grammatical: inhibitory interference effect
means$Effect[1]
```

```
## [1] 8.659
```

```r
## ungrammatical: facilitatory interference effect
means$Effect[2]
```

```
## [1] -11.545
```

The above runs show the expected pattern: low lf values lead to small effects, and large lf values to large effects.

Model predictions for Dillon et al data

Dillon et al 2013 data (source: Jaeger et al 2019 Bayesian reanalysis):

- Agreement -60 ms, CrI [-112, -5] ms. Implies Normal(-60,33)
- Reflexives -18 ms, CrI [-72, 36] ms. Implies Normal(-18,27)

```r
## our data from one subject in one pair of conditions (difference in means):
xbar_au<- -60
## 1 SD above and below mean
lower_au <- -93
upper_au <- -27

xbar_ru<- -18
## 1 SD above and below mean
```

```
lower_ru <- -45
upper_ru <- 9
```

Estimate lf for ungrammatical agreement data:

```
## Rejection sampling:
lower<-lower_au
upper<-upper_au

nsamp<-5000
lf_posterior<-rep(NA,nsamp)
for(i in 1:nsamp){
  ## generate *random* latency factor value each time
  latency_factor <<- rbeta(1,a,b)
## get generated effect:
generated_effect<-iterate_lf(latency_factor)$Effect[2]
if(printcounts){
print(paste("count: ",i,sep=" "))
print(paste(lower,generated_effect,upper,sep=" "))
}
  ## if generated effect is within bounds, accept
if(generated_effect>=lower & generated_effect<=upper){
  lf_posterior[i]<-latency_factor
} else {
  ## reject
  lf_posterior[i]<- -1
  }
}

rejected<-which(lf_posterior==-1)
length(lf_posterior[-rejected])/nsamp
```

```
## [1] 0.4508
```

```
quantile(lf_posterior[-rejected],probs=c(0.025,0.975))
```

```
##    2.5%   97.5%
## 0.23557 0.63633
```

```
mean(lf_posterior[-rejected])
```

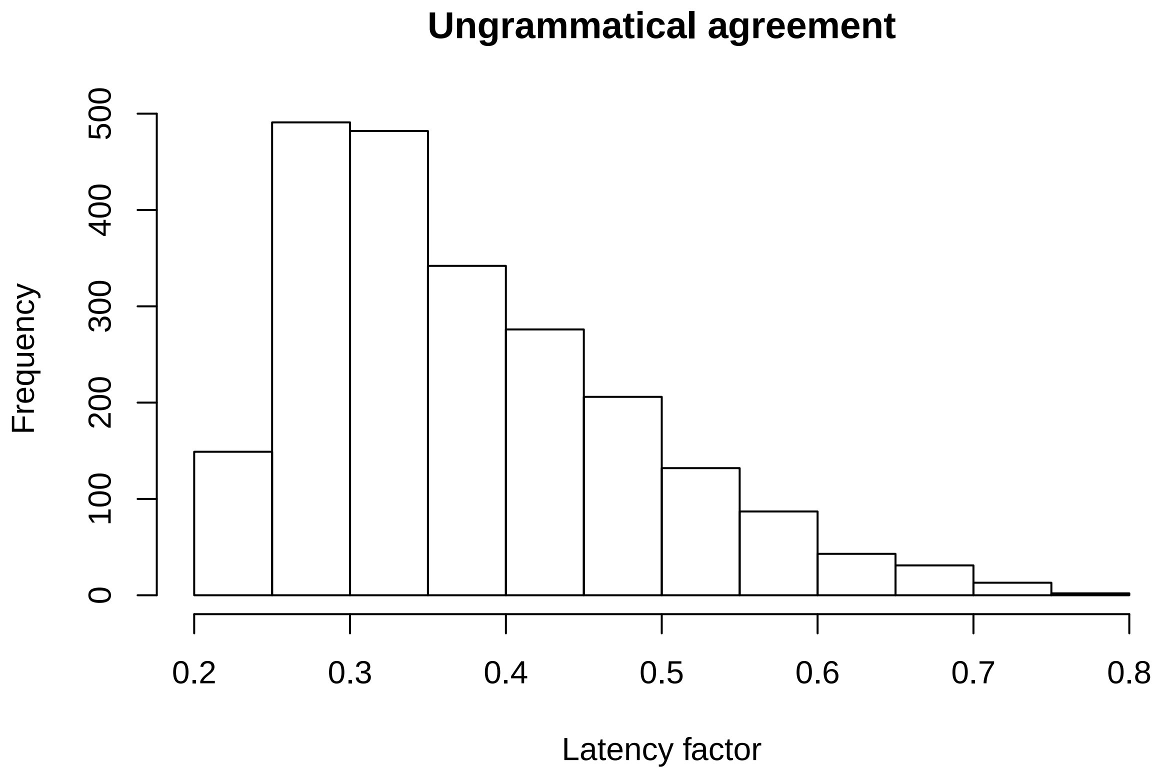
```
## [1] 0.37568
```

Save the results (if needed):

```
save(lf_posterior,file="RdaFilesVague/au_lf_D13.Rda")
```

Visualize posterior of lf:

```
hist(lf_posterior[-rejected],main="Ungrammatical agreement",
     xlab="Latency factor")
```



Estimate predicted range of effects for ungrammatical agreement using mean lf

```
load("RdaFilesVague/au_lf_D13.Rda")

lf_posterior_accepted<-lf_posterior[-which(lf_posterior==-1)]

n<-length(lf_posterior_accepted)

au_predicted_means<-rep(NA,n)
ag_predicted_means<-rep(NA,n)

for(i in 1:n){
  predictions<-iterate_lf(values=lf_posterior_accepted[i])
  ## grammatical:
  ag_predicted_means[i]<-predictions$Effect[1]
  ## ungrammatical:
  au_predicted_means[i]<-predictions$Effect[2]
}
```

Save results:

```
save(ag_predicted_means,file="RdaFilesVague/ag_predicted_means_D13.Rda")
save(au_predicted_means,file="RdaFilesVague/au_predicted_meansD13.Rda")
```

Summary of predicted RTs:

```
summary(ag_predicted_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    18.4    27.6    33.2    35.5    41.4    77.8
```

```r
summary(au_predicted_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -90.1   -49.1   -39.1   -41.7   -32.4   -20.6
```

Estimate lf for ungrammatical reflexive data:

```r
lower<-lower_ru
upper<-upper_ru

nsamp<-5000
lf_posterior<-rep(NA,nsamp)
for(i in 1:nsamp){
  ## generate *random* latency factor value each time
  latency_factor <<- rbeta(1,a,b)
## get generated effect:
generated_effect<-iterate_lf(latency_factor)$Effect[2]
if(printcounts){
print(paste("count: ",i,sep=" "))
print(paste(lower,generated_effect,upper,sep=" "))
}
  ## if generated effect is within bounds, accept
if(generated_effect>=lower & generated_effect<=upper){
  lf_posterior[i]<-latency_factor
} else {
  ## reject
  lf_posterior[i]<- -1
  }
}

rejected<-which(lf_posterior==-1)
length(lf_posterior[-rejected])/nsamp
```

```
## [1] 0.8534
```

```r
quantile(lf_posterior[-rejected],probs=c(0.025,0.975))
```

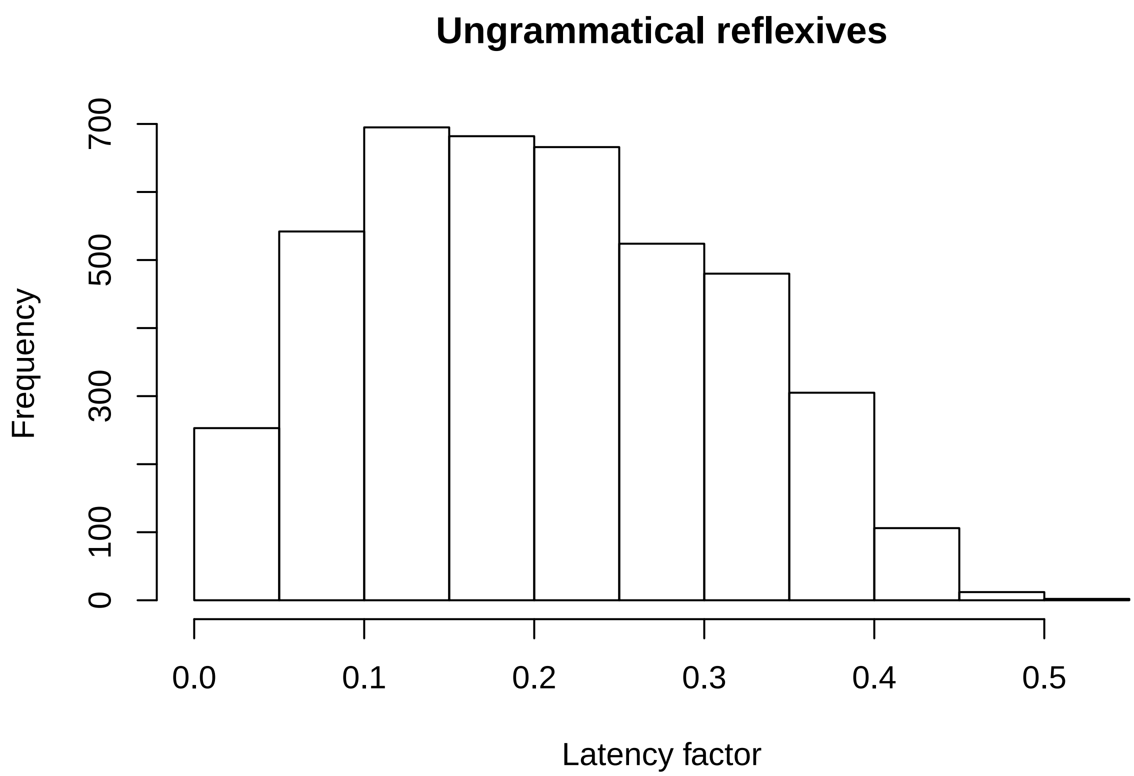
```
##     2.5%    97.5%
## 0.032547 0.403649
```

```r
mean(lf_posterior[-rejected])
```

```
## [1] 0.20363
```

```r
save(lf_posterior,file="RdaFilesVague/ru_lf_D13.Rda")
```

```r
hist(lf_posterior[-rejected],main="Ungrammatical reflexives",
     xlab="Latency factor")
```



Estimate predicted range of effects for ungrammatical reflexive using mean lf

Having estimated the LF posterior for ungrammatical conditions only, we will use it to generate predictions for *both* ungrammatical and grammatical conditions. Otherwise we may overfit to both conditions.

```
load("RdaFilesVague/ru_lf_D13.Rda")

lf_posterior_accepted<-lf_posterior[-which(lf_posterior==-1)]

n<-length(lf_posterior_accepted)

ru_predicted_means<-rep(NA,n)
rg_predicted_means<-rep(NA,n)

for(i in 1:n){
predictions<-iterate_lf(values=lf_posterior_accepted[i])
rg_predicted_means[i]<-predictions$Effect[1]
ru_predicted_means[i]<-predictions$Effect[2]
}
```

```
save(rg_predicted_means,file="RdaFilesVague/rg_predicted_meansD13.Rda")
save(ru_predicted_means,file="RdaFilesVague/ru_predicted_meansD13.Rda")
```

```
summary(rg_predicted_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.06   11.09   18.64   19.29   26.86   56.14
```

```
summary(ru_predicted_means)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -62.638 -31.412 -21.758 -22.643 -13.037  -0.162
```

Model predictions for Jäger et al 2019 replication data

Next, we turn to our replication data:

- Agreement: -22 [-46, 3] Implies Normal(-22,13)
- Reflexives: -23 [-48, 2] Implies Normal(-23,13)

```r
## our data from one subject in one pair of conditions (difference in means):
xbar_aurep<- -22
## 1 SD above and below mean
lower_aurep <- -35
upper_aurep <- -9

xbar_rurep<- -23
## 1 SD above and below mean
lower_rurep <- -36
upper_rurep <- -10
```

Estimate lf for ungrammatical agreement data (replication)

```r
## Rejection sampling:
lowerrep<-lower_aurep
upperrep<-upper_aurep

nsamp<-5000
lf_posterior<-rep(NA,nsamp)
for(i in 1:nsamp){
  ## generate *random* latency factor value each time
  latency_factor <<- rbeta(1,a,b)
## get generated effect:
generated_effect<-iterate_lf(latency_factor)$Effect[2]
if(printcounts){
print(paste("count: ",i,sep=" "))
print(paste(lowerrep,generated_effect,upperrep,sep=" "))
}
  ## if generated effect is within bounds, accept
if(generated_effect>=lowerrep & generated_effect<=upperrep){
  lf_posterior[i]<-latency_factor
} else {
  ## reject
  lf_posterior[i]<- -1
  }
}

rejected<-which(lf_posterior==-1)
length(lf_posterior[-rejected])/nsamp
```

```
## [1] 0.6072
```

```r
quantile(lf_posterior[-rejected],probs=c(0.025,0.975))
```

```
##     2.5%    97.5%
## 0.088215 0.322218
```

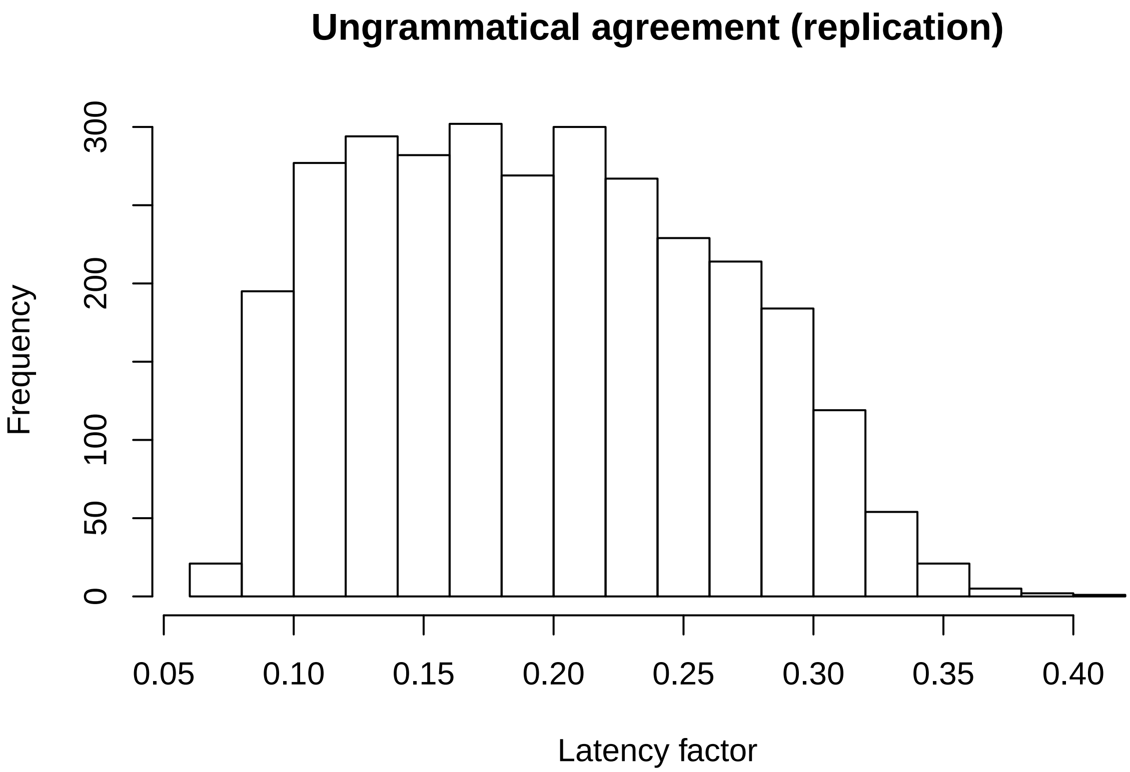
```r
mean(lf_posterior[-rejected])
```

```
## [1] 0.19459
```

```r
save(lf_posterior,file="RdaFilesVague/au_lf_D13rep.Rda")
```

Visualize posterior of lf:

```r
hist(lf_posterior[-rejected],main="Ungrammatical agreement (replication)",xlab="Latency factor")
```



Estimate predicted range of effects for ungrammatical agreement data using mean lf (replication data)

```r
load("RdaFilesVague/au_lf_D13rep.Rda")

lf_posterior_accepted<-lf_posterior[-which(lf_posterior==-1)]

n<-length(lf_posterior_accepted)

ag_predicted_means_rep<-au_predicted_means_rep<-rep(NA,n)

for(i in 1:n){
ag_predicted_means_rep[i]<-iterate_lf(values=lf_posterior_accepted[i])$Effect[1]
au_predicted_means_rep[i]<-iterate_lf(values=lf_posterior_accepted[i])$Effect[2]
}
```

```r
save(ag_predicted_means_rep,file="RdaFilesVague/ag_predicted_meansD13rep.Rda")
save(au_predicted_means_rep,file="RdaFilesVague/au_predicted_meansD13rep.Rda")
```

Estimate lf for ungrammatical reflexive data (replication)

```r
lower<-lower_rurep
upper<-upper_rurep

nsamp<-5000
lf_posterior<-rep(NA,nsamp)
for(i in 1:nsamp){
  ## generate *random* latency factor value each time
  latency_factor <<- rbeta(1,a,b)
## get generated effect:
generated_effect<-iterate_lf(latency_factor)$Effect[2]
if(printcounts){
print(paste("count: ",i,sep=" "))
print(paste(lower,generated_effect,upper,sep=" "))
}
  ## if generated effect is within bounds, accept
if(generated_effect>=lower & generated_effect<=upper){
  lf_posterior[i]<-latency_factor
} else {
  ## reject
  lf_posterior[i]<- -1
  }
}

rejected<-which(lf_posterior==-1)
length(lf_posterior[-rejected])/nsamp
```

```
## [1] 0.602
```

```r
quantile(lf_posterior[-rejected],probs=c(0.025,0.975))
```

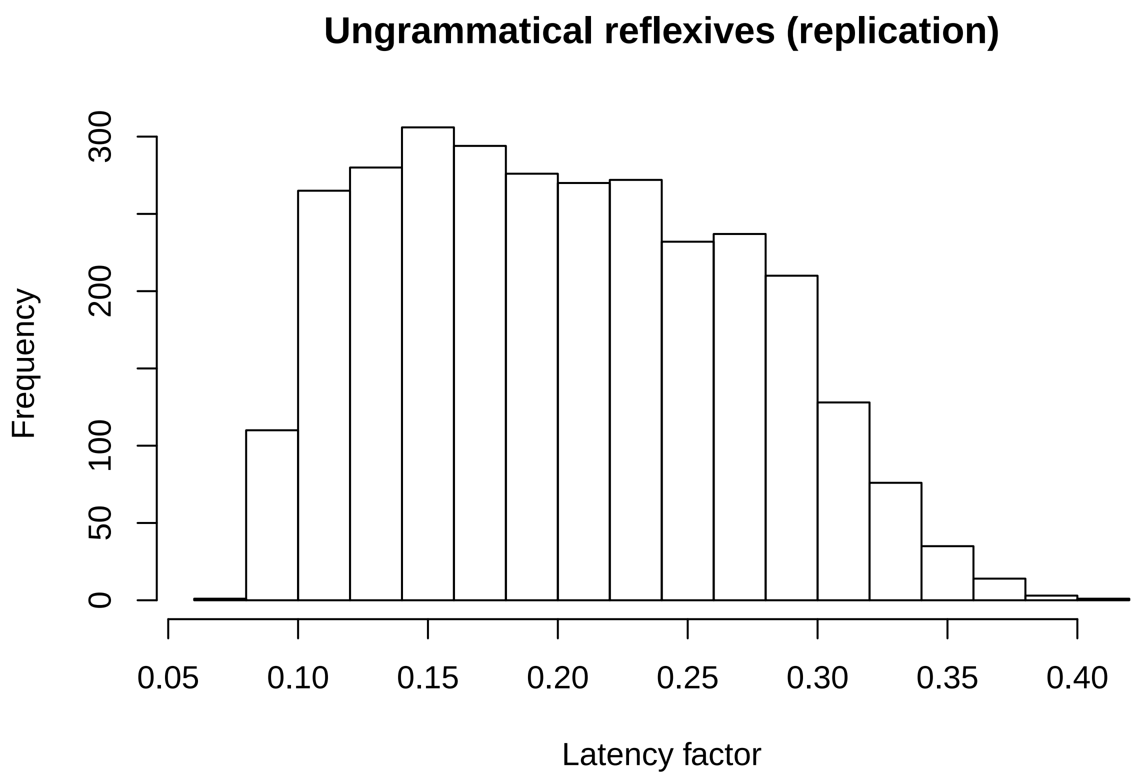
```
##     2.5%    97.5%
## 0.096733 0.330976
```

```r
mean(lf_posterior[-rejected])
```

```
## [1] 0.20264
```

```r
save(lf_posterior,file="RdaFilesVague/ru_lf_D13rep.Rda")
```

```r
hist(lf_posterior[-rejected],main="Ungrammatical reflexives (replication)",xlab="Latency factor")
```



Estimate predicted range of effects for ungrammatical reflexive data using mean lf (replication data)

```
load("RdaFilesVague/ru_lf_D13rep.Rda")

lf_posterior_accepted<-lf_posterior[-which(lf_posterior==-1)]

n<-length(lf_posterior_accepted)

rg_predicted_means_rep<-ru_predicted_means_rep<-rep(NA,n)

for(i in 1:n){
rg_predicted_means_rep[i]<-iterate_lf(values=lf_posterior_accepted[i])$Effect[1]
ru_predicted_means_rep[i]<-iterate_lf(values=lf_posterior_accepted[i])$Effect[2]
}
```

```
save(rg_predicted_means_rep,file="RdaFilesVague/rg_predicted_meansD13rep.Rda")
save(ru_predicted_means_rep,file="RdaFilesVague/ru_predicted_meansD13rep.Rda")
```

Validating the ABC method on gold standard (known lf) values

The estimates of lf for the Dillon et al ungrammatical agreement and reflexives conditions are computed above. Using the Dillon et al data, we first computed the means of the posteriors of the latency factor from the two comparisons (reflexives and agreement). Then we check if the ABC algorithm recovers the true lf value used to generate the predictions from the model.

Estimates of the interference effect based on the posterior distributions of the latency factor

```
## this is the posterior distribution of lf based on the Dillon reflexive data:
load("RdaFilesVague/ru_lf_D13.Rda")
## extract mean and lower and upper 95% credible interval bounds:
mean_ru_lf_D13<-mean(lf_posterior[-which(lf_posterior==-1)])
lower_ru_lf_D13<-quantile(lf_posterior[-which(lf_posterior==-1)],prob=0.025)
upper_ru_lf_D13<-quantile(lf_posterior[-which(lf_posterior==-1)],prob=0.975)
## fit three models, but we will use only the model based on means:
means <- iterate_lf(mean_ru_lf_D13)
lower <- iterate_lf(lower_ru_lf_D13)
upper <- iterate_lf(upper_ru_lf_D13)
ru_lf_est_D13<-c(lower$Effect[2],
                 means$Effect[2],
                 upper$Effect[2])
```

Reflexive effect: Estimating lf

First we use the small reflexives estimate as our effect to recover the lf parameter estimate.

```
## our data from one subject in one pair of conditions (difference in means):
xbar<-ru_lf_est_D13[2]
## distance in SE units of upper bound from mean
abs(ru_lf_est_D13[3]-ru_lf_est_D13[1])/2
```

```
## [1] 19.327
```

```
## lower bound distance:
abs(ru_lf_est_D13[2]-ru_lf_est_D13[1])/2
```

```
## [1] 8.953
```

```
## take the bigger value
se<-abs(ru_lf_est_D13[3]-ru_lf_est_D13[1])/2
## set acceptable bounds of generated effect:
lower_bound <- xbar-se
upper_bound <- xbar+se
```

Now, we check whether the algorithm can recover the true value of lf that generated the data.

The true mean of lf here was `{r round(mean_ru_lf_D13,4)}`.

```
## Rejection sampling:
nsamp<-5000
lf_posterior_au<-rep(NA,nsamp)
for(i in 1:nsamp){
  ## generate random latency factor value each time:
  latency_factor <<- rbeta(1,a,b)
## get generated effect:
generated_effect<-iterate_lf(latency_factor)$Effect[2]
if(printcounts){
print(paste("count: ",i,sep=" "))
print(paste(lower_bound,generated_effect,upper_bound,sep=" "))
}
```

```r
  ## if generated effect is within bounds, accept
if(generated_effect>=lower_bound & generated_effect<=upper_bound){
  if(printcounts){print("accept")}
  lf_posterior[i]<-latency_factor
} else {
  ## reject
  if(printcounts){print("reject")}
  lf_posterior[i]<- -1
  }
}

rejected<-which(lf_posterior==-1)
length(lf_posterior[-rejected])/nsamp
```

```
## [1] 0.789
```

```r
quantile(lf_posterior[-rejected],probs=c(0.025,0.975))
```

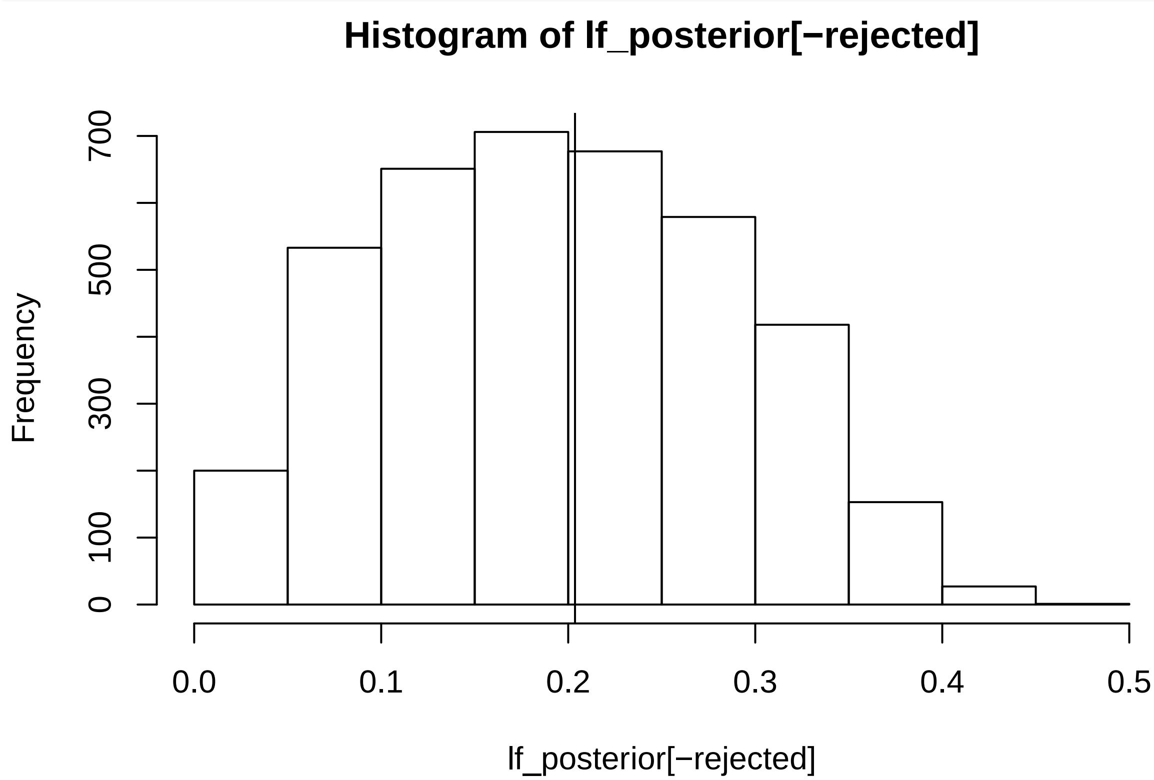
```
##     2.5%    97.5%
## 0.036139 0.368041
```

```r
mean(lf_posterior[-rejected])
```

```
## [1] 0.19448
```

Visualize posterior of lf:

```r
## posterior:
hist(lf_posterior[-rejected])
abline(v=mean_ru_lf_D13)
```



The true value is within the 95% credible interval.

Agreement effect: Estimating lf

```
load("RdaFilesVague/au_lf_D13.Rda")
mean_au_lf_D13<-mean(lf_posterior[-which(lf_posterior==-1)])
lower_au_lf_D13<-quantile(lf_posterior[-which(lf_posterior==-1)],prob=0.025)
upper_au_lf_D13<-quantile(lf_posterior[-which(lf_posterior==-1)],prob=0.975)
means <- iterate_lf(mean_au_lf_D13)
lower <- iterate_lf(lower_au_lf_D13)
upper <- iterate_lf(upper_au_lf_D13)
au_lf_est_D13<-c(lower$Effect[2],
                 means$Effect[2],
                 upper$Effect[2])
```

```
## our data from one subject in one pair of conditions (difference in means):
xbar<-au_lf_est_D13[1]
## distance in SE units of upper bound from mean
abs(au_lf_est_D13[3]-au_lf_est_D13[1])/2
```

```
## [1] 20.636
```

```
## lower bound distance:
abs(au_lf_est_D13[2]-au_lf_est_D13[1])/2
```

```
## [1] 9.987
```

```
## take the bigger value
se<-abs(au_lf_est_D13[3]-au_lf_est_D13[1])/2
## set acceptable bounds of generated effect:
lower_bound <- xbar-se
upper_bound <- xbar+se
```

Now, we check whether the algorithm can recover the true value of lf that generated the data.

The true mean of lf here was `{r round(mean_au_lf_D13,4)}`.

```
## Rejection sampling:
nsamp<-5000
lf_posterior_au<-rep(NA,nsamp)
for(i in 1:nsamp){
  ## generate random latency factor value each time:
  latency_factor <<- rbeta(1,a,b)
## get generated effect:
generated_effect<-iterate_lf(latency_factor)$Effect[2]
if(printcounts){
print(paste("count: ",i,sep=" "))
print(paste(lower_bound,generated_effect,upper_bound,sep=" "))
}
  ## if generated effect is within bounds, accept
if(generated_effect>=lower_bound & generated_effect<=upper_bound){
  if(printcounts){print("accept")}
  lf_posterior[i]<-latency_factor
} else {
  ## reject
  if(printcounts){print("reject")}
  lf_posterior[i]<- -1
  }
}
```

```r
rejected<-which(lf_posterior==-1)
length(lf_posterior[-rejected])/nsamp
```

```
## [1] 0.816
```

```r
quantile(lf_posterior[-rejected],probs=c(0.025,0.975))
```

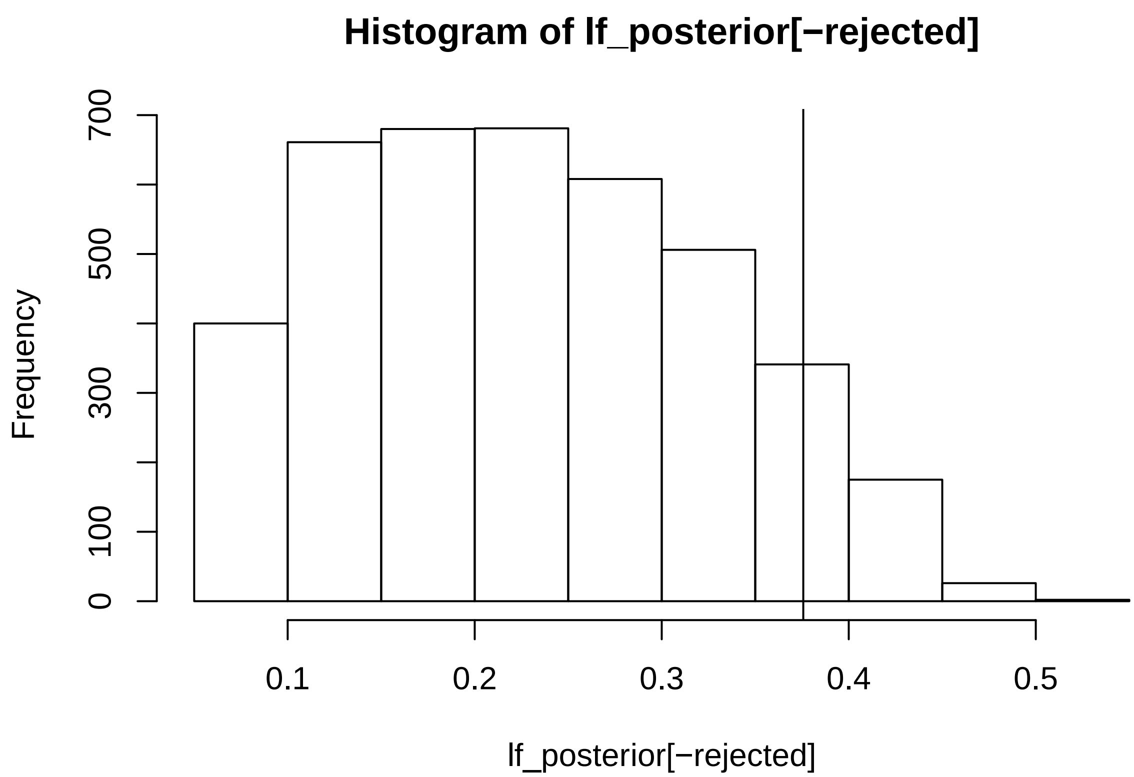
```
##     2.5%    97.5%
## 0.072014 0.419112
```

```r
mean(lf_posterior[-rejected])
```

```
## [1] 0.22859
```

Visualize posterior of lf:

```r
## posterior:
hist(lf_posterior[-rejected])
abline(v=mean_au_lf_D13)
```



The true lf value lies within the 95% credible interval of the posterior distribution.

This shows that the ABC approach can in principle generate a posterior distribution of the lf parameter that contains the true value.