Summarizing parameters from Théo's project

chundra

12/18/2023

This notebook summarizes parameters from the best-fitting model for vocal rhythm frequency. The best-fitting model is a univariate (i.e., rate-homogeneous) Brownian motion model.

```
## Loading required package: ggplot2
```

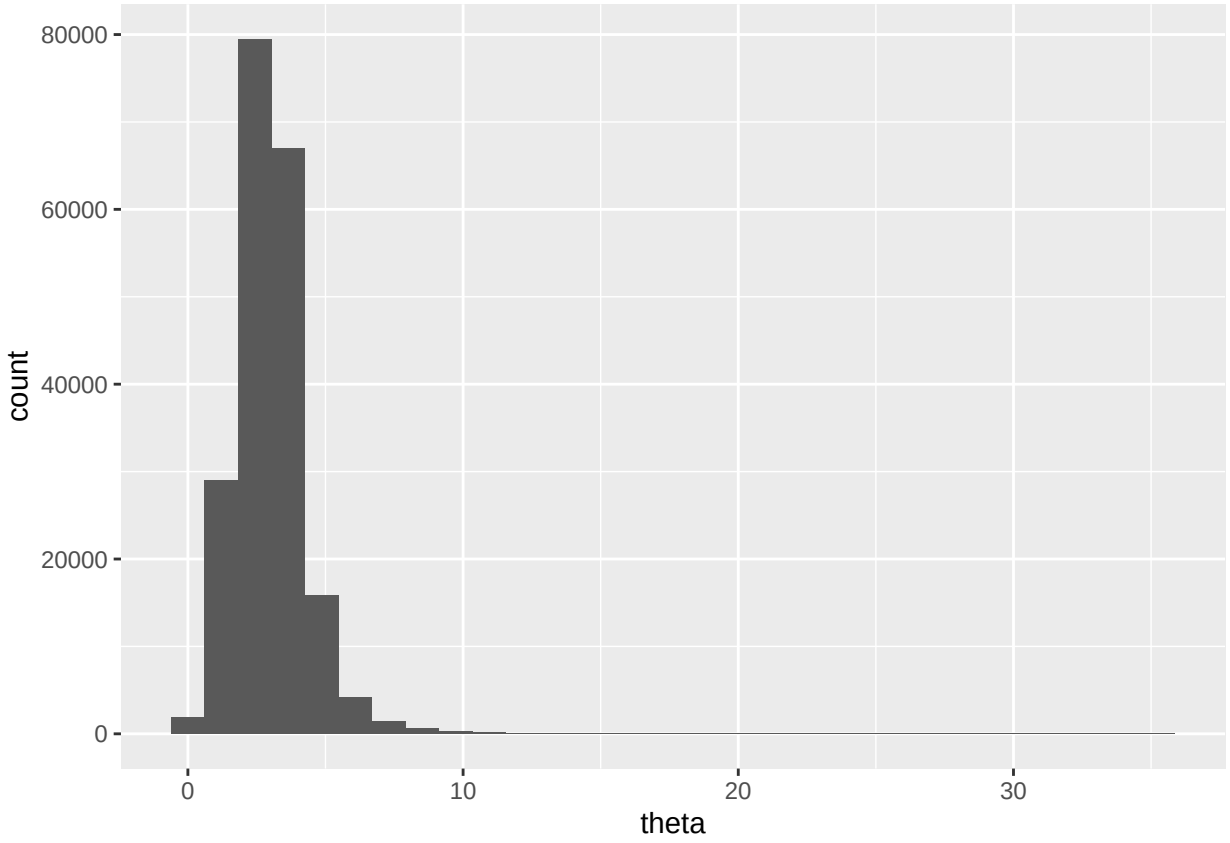
```
## Loading required package: HDInterval
```

theta

The following histogram gives the posterior distribution of θ. This parameter represents both frequency values reconstructed to the root of the phylogeny, as well as the optimal rhythm value to which the OU process reverts:

```
ggplot(data=params.df,aes(x=theta)) + geom_histogram()
```

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



The median is as follows:

```
median(exp(log.theta))
```

```
## [1] 2.914998
```

The lower and upper bounds of the 95% highest posterior density interval are as follows:

```
hdi(exp(log.theta))
```

```
##     lower     upper
## 0.5131209 5.1486289
## attr(,"credMass")
## [1] 0.95
```

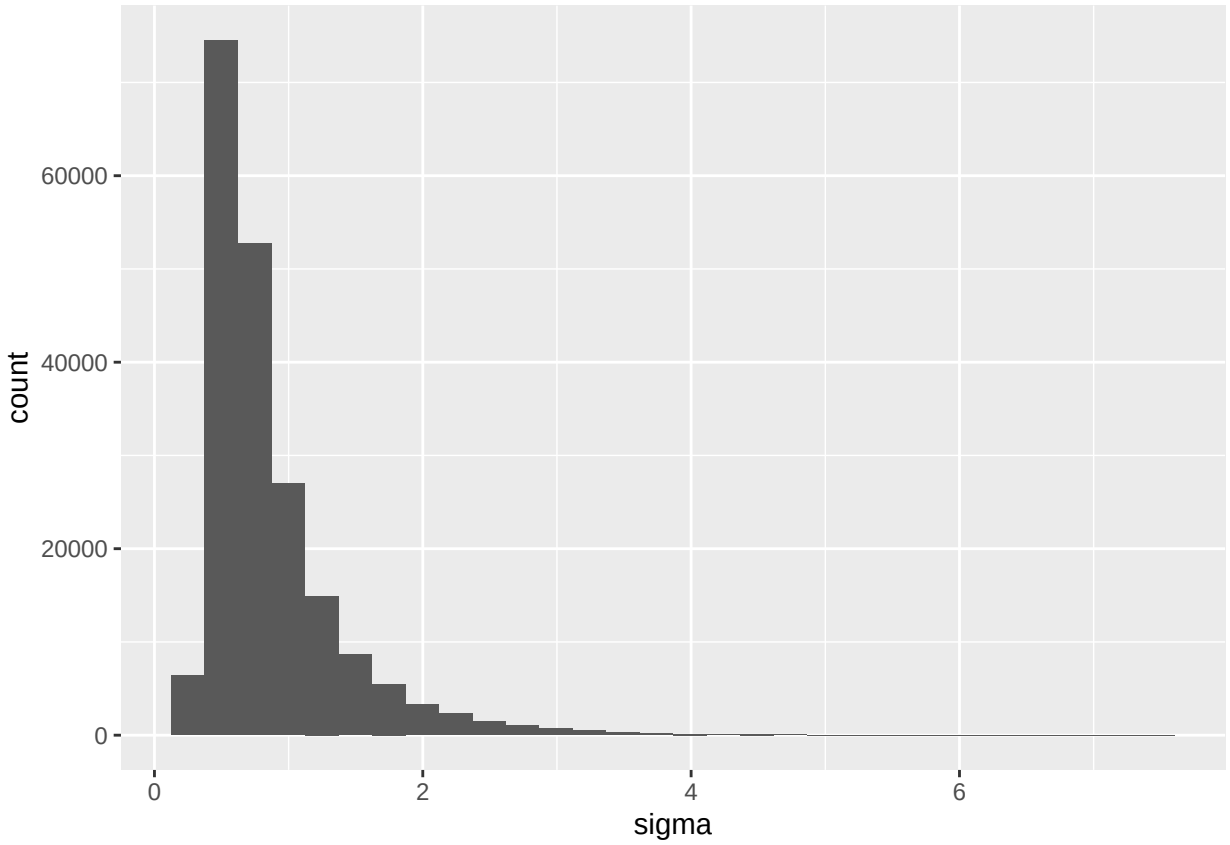
These values are more compatible with the delta frequency band (.5-4 Hz) than the theta frequency band (4-8 Hz), but it is clear that there is some overlap with theta.

sigma

The following histogram gives the posterior distribution of σ, the variance of the drift process:

```
ggplot(data=params.df,aes(x=sigma)) + geom_histogram()
```

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



The scale of drift has the following median

```
median(sigma)
```

```
## [1] 0.6960472
```

and HPD:

```
hdi(sigma)
```

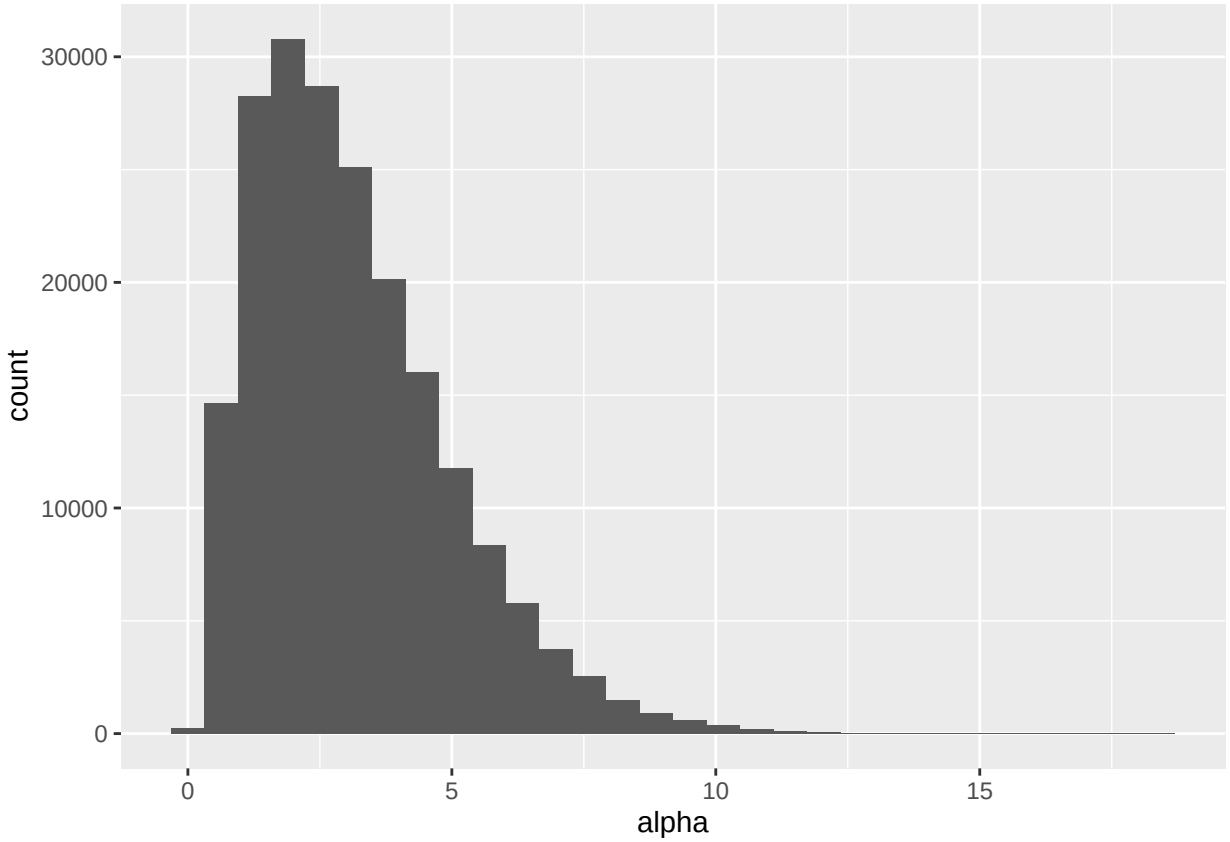
```
##     lower     upper
## 0.2949979 1.9085910
## attr(,"credMass")
## [1] 0.95
```

alpha

The following histogram gives the posterior distribution of α, the strength of selection to the optimal value:

```
ggplot(data=params.df,aes(x=alpha)) + geom_histogram()
```

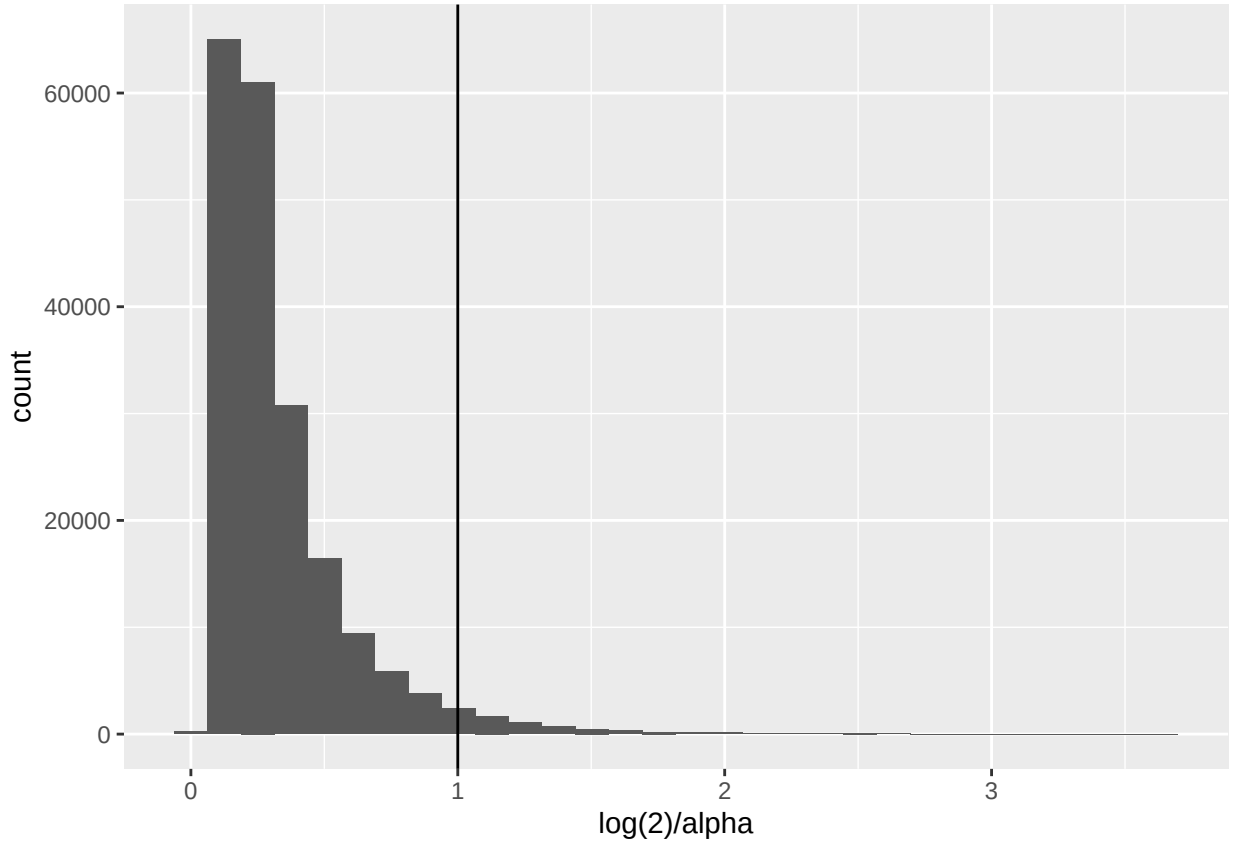
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



An easier way to interpret this parameter is to transform it to the phylogenetic half-life (Grabowski et al. 2023), $\frac{\ln 2}{\alpha}$. This is interpreted as the average time for a trait to evolve halfway from an ancestral state toward a new optimum, indicating how long it will take before adaptation to a new regime is more influential than constraints from the ancestral state. If half-life values are greater than the height of the phylogeny (1 in our case, as the tree length is scaled to unit height), the process increasingly resembles Brownian motion and involves a slower adaptation speed:

```
ggplot(data=params.df,aes(x=log(2)/alpha)) + geom_histogram() + geom_vline(xintercept=1)
```

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



The following proportion of half-life values are greater than 1 (the height of the tree):

```
length(which(log(2)/params.df$alpha > 1))/length(params.df$alpha)
```

```
## [1] 0.03102
```

There is more support for a faster adaptation scenario (ca. 97% posterior support vs. ca. 3% posterior support).

simulation

The following function simulates a OU process using draws from the posterior distribution of parameters. The process is logged at intervals of .001 units. We simulate the OU process using the Euler-Maruyama method:

```
set.seed(1234)

sim.OU <- function(theta, alpha, sigma, total.time=1, interval=.001) {
  i = interval
  x <- c(log(theta))
  x.old <- log(theta)
  while (i < total.time) {
    x.new <- x.old - alpha*(x.old - log(theta))*interval + rnorm(1,0,sigma)*sqrt(interval)
    x <- c(x,x.new)
```

```
    x.old <- x.new
    i <- i + interval
  }
  return(exp(x))
}
```

We can simulate 500 trajectories:

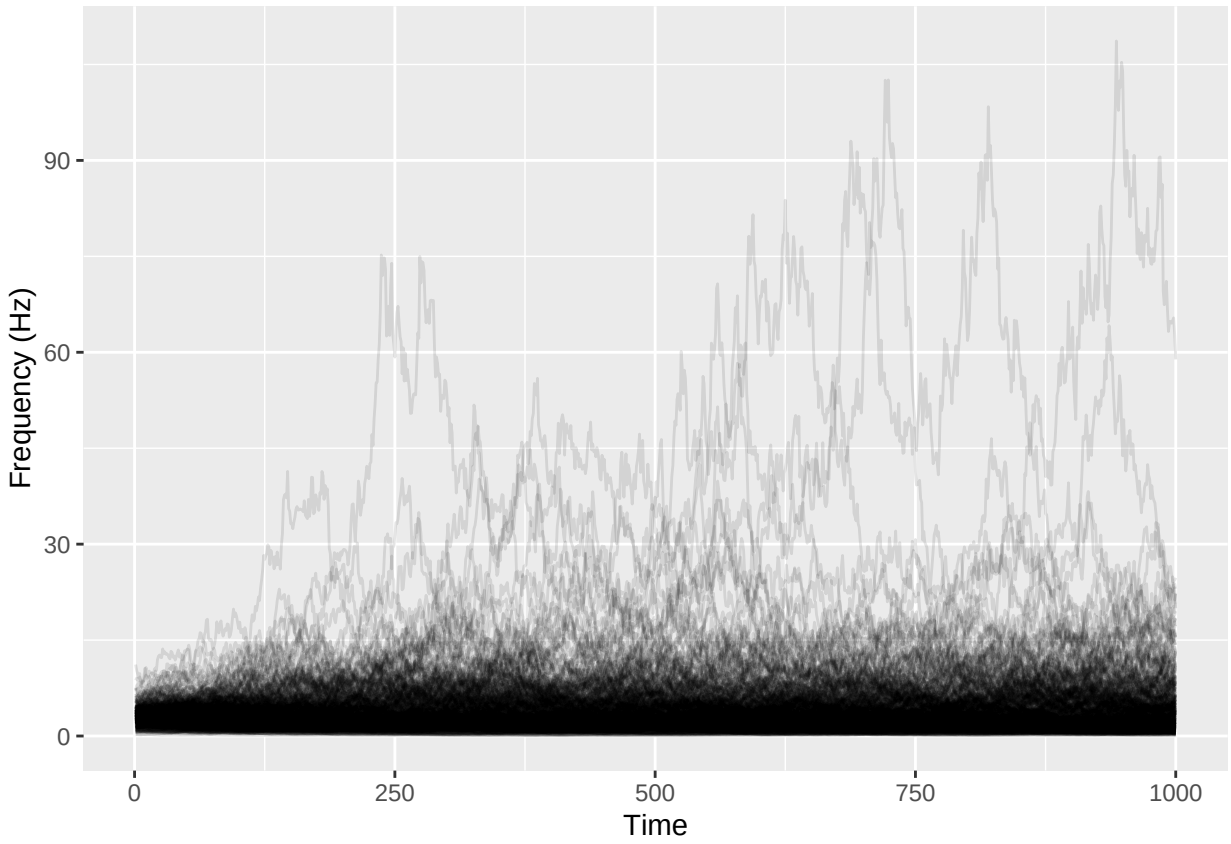
```
time <- c()
value <- c()
iter <- c()


for (i in 1:500) {
  j <- sample(1:nrow(params.df),1)
  theta.i <- params.df[j,]$theta
  alpha.i <- params.df[j,]$alpha
  sigma.i <- params.df[j,]$sigma
  x <- sim.OU(theta.i,alpha.i,sigma)
  N <- length(x)
  time <- c(time,c(1:N))
  value <- c(value,x)
  iter <- c(iter,rep(i,N))
}
```

We make a data frame:

```
sim.df <- data.frame(time=time,value=value,iter=iter)
```

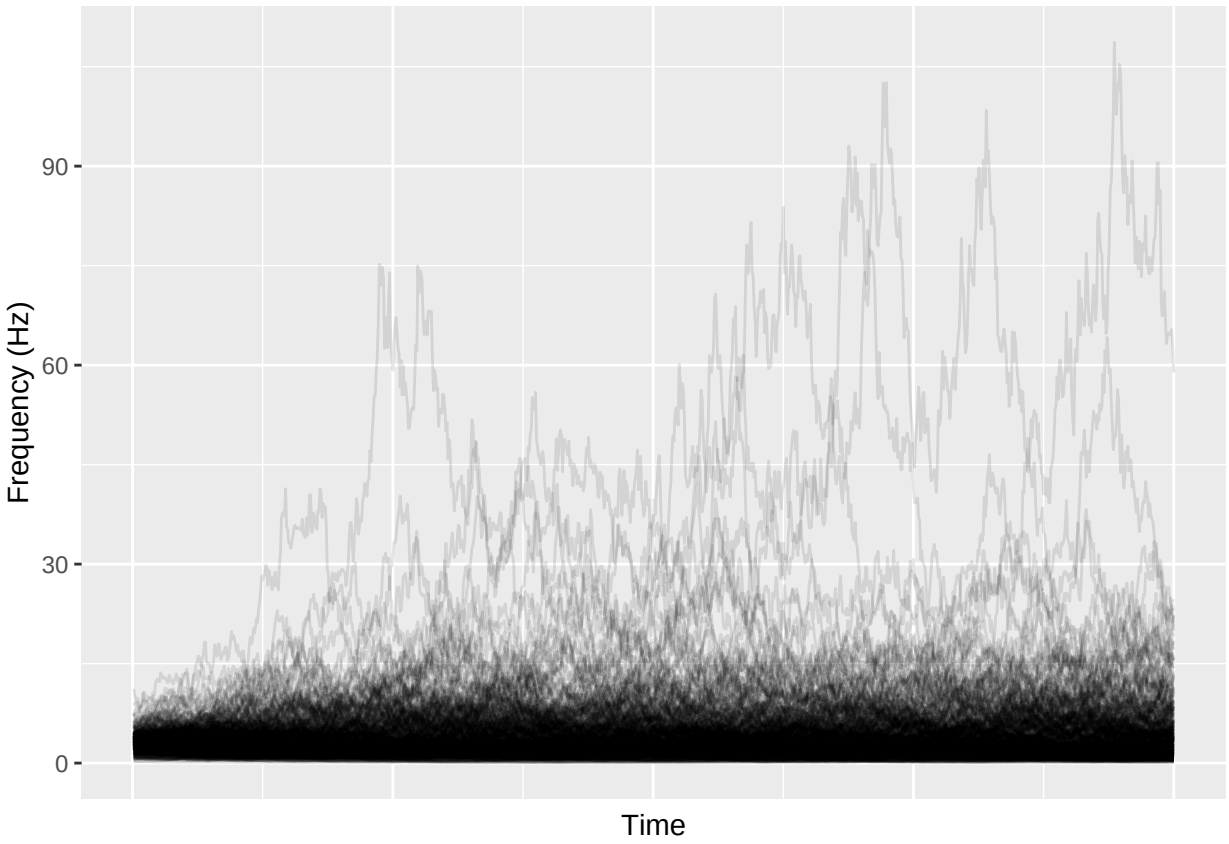
Plotting trajectories, with time units:

```
ggplot() + geom_line(data=sim.df,aes(x = time, y = value, group = iter),alpha=.1) + #theme(labs(y='Freq
  xlab('Time') + ylab('Frequency (Hz)')
```



Without time units:

```
ggplot() + geom_line(data=sim.df,aes(x = time, y = value, group = iter),alpha=.1) + #theme(labs(y='Freq
  xlab('Time') + ylab('Frequency (Hz)') +
  theme(axis.text.x = element_blank(),axis.ticks.x = element_blank())
```


90
60
30
0
Frequency (Hz)
Time