# Example run of moments++, pure two-locus modeol, 10-fold bottleneck

Gustavo V. Barroso & Aaron P. Ragsdale

2025-03-21

The goal of this example is to predict B-values over time, following a 10-fold bottleneck.

## Creating folders and files for running moments++

Builds a parameter grid that allows for fast assembly of whole chromosomes

```
Ns <- paste(1e+3, 1e+4, sep=";") # from present to past
Ts <- paste(0, 25000, sep=";") # from present (0) to past, in generations

uL <- 1e-8
uR <- 1e-8

r <- c(1e-8, 2.5e-5, 1e-3)
s <- -c(1e-5, 1e-4, 1e-3)

params <- setDT(crossing(Ns, Ts, r, s, uL, uR))

fwrite(as.data.frame(r), "output/r.csv", col.names=F)
fwrite(as.data.frame(s), "output/s.csv", col.names=F)
fwrite(params, "output/params.csv")
```

Creates directories where files will be written to

```
source ~/.bashrc

cd output

cat r.csv | while IFS=, read -r r
do
  mkdir r_$r
  cat s.csv | while IFS=, read -r s
  do
    mkdir r_$r/s_$s
  done
done

find -mindepth 2 -type d -exec cp opt.bpp {} \;

# makes directory to store purely neutral models to get pi0 predictions
mkdir demo
cp opt.bpp demo
```

Creates files that will be used as input for moments++

```
params <- fread("output/params.csv")
n_models <- nrow(params)

# writes augmented demes files specifying models (demography, u, r, s)
options(digits=16) # precision
for(i in 1:n_models) {

  Ns <- as.numeric(strsplit(params[i,]$Ns, ";")[[1]])
  Ts <- as.integer(strsplit(params[i,]$Ts, ";")[[1]])
  u <- as.numeric(params[i,]$uL)
  r <- as.numeric(params[i,]$r)
  s <- as.numeric(params[i,]$s)

  name <- paste("model_", i, sep="")
  sink(paste("output/r_", r, "/s_", s,"/", name, ".yaml", sep=""))

  cat("time_units: generations\n")
  cat("demes:\n")
  cat("  - name: X\n")
  cat("    epochs:\n")
  for(j in length(Ts):1) {
    cat("      - end_time: ")
    cat(Ts[j])
    cat("\n")
    cat("        start_size: ")
    cat(Ns[j])
    cat("\n")
  }
  cat("metadata:\n")
  cat("  - name: mutation\n")
  cat("    left_factor: 1\n")
  cat("    epochs:\n")
  cat("      - end_time: 0\n")
  cat("        rates: [")
  cat(u)
  cat("]\n")
  cat("  - name: recombination\n")
  cat("    epochs:\n")
  cat("      - end_time: 0\n")
  cat("        rates: [")
  cat(r)
  cat("]\n")
  cat("  - name: selection\n")
  cat("    start_time: .inf\n")
  cat("    epochs:\n")
  cat("      - end_time: 0\n")
  cat("        rates: [")
  cat(format(s, scientific=F))
  cat("]\n")

  sink()
}
```

```
# creates the (purely neutral) demographic model
Ns <- as.numeric(strsplit(params[i,]$Ns, ";")[[1]])
Ts <- as.numeric(strsplit(params[i,]$Ts, ";")[[1]])
u <- as.numeric(unique(params$uL))

sink(paste("output/demo/model_neutral.yaml", sep=""))
cat("time_units: generations\n")
cat("demes:\n")
cat("  - name: X\n")
cat("    epochs:\n")
for(j in length(Ts):1) {
  cat("      - end_time: ")
  cat(Ts[j])
  cat("\n")
  cat("        start_size: ")
  cat(Ns[j])
  cat("\n")
}
cat("metadata:\n")
cat("  - name: mutation\n")
cat("    left_factor: 1\n")
cat("    epochs:\n")
cat("      - end_time: 0\n")
cat("        rates: [")
cat(u)
cat("]\n")
cat("  - name: recombination\n")
cat("    epochs:\n")
cat("      - end_time: 0\n")
cat("        rates: [0]\n") #  irrelevant, we're only interested in pi0 here
cat("  - name: selection\n")
cat("    start_time: .inf\n")
cat("    epochs:\n")
cat("      - end_time: 0\n")
cat("        rates: [0]\n")
sink()
```

## Running moments++

For each model, get expected summary statistics over time (T=500 generations)

```
#!/bin/bash

cd output
nr=$(wc -l r.csv | cut -d ' ' -f 1)

for r in r_*
do
  cd $r
  for s in s_*
  do
    cd $s
```

```
    for f in model_*.yaml
    do
      # this exploits knowledge of the .yaml files specifying the models
      Na=$(sed '6q;d' $f | cut -d : -f 2 | cut -d ' ' -f 2)
      N1=$(sed '8q;d' $f | cut -d : -f 2 | cut -d ' ' -f 2)
      Ne=$Na

      # this shows how to be a bit more general, and choose the order of
      # (1-2p) factors based on the largest of the two population sizes
      if [[ $N1 -gt $Na ]]
      then
        Ne=$N1
      fi

      # an example of how to handle arbitrary values of Ne*s
      x=$(sed '23q;d' $f | cut -d : -f 2 | cut -d [ -f 2 | cut -d ] -f 1)
      s=$(echo "scale=10; $x" | bc)
      a=$(echo "scale=10; $Ne * $s" | bc)

      if [[ $(echo "$a < -1" | bc) == 1 ]]
      then
        b=${a%.*}
        Ns=$(($b * -2))

        # choosing an appropriate order of (1-2p) factors based on Ne*s
          if [[ $Ns -lt 30 ]]
          then
            momentspp params=opt.bpp F=$f O=70 > /dev/null
        elif [[ $Ns -lt 70 ]]
        then
          momentspp params=opt.bpp F=$f O=150 > /dev/null
        elif [[ $Ns -lt 150 ]]
        then
          momentspp params=opt.bpp F=$f O=300 > /dev/null
        else
          momentspp params=opt.bpp F=$f O=400 > /dev/null
        fi
        else
          momentspp params=opt.bpp F=$f O=10 > /dev/null
      fi
    done
    cd ..
  done
  cd ..
done

# get pi0 from neutral model following the same demography
cd demo
for f in *yaml
do
  momentspp params=opt.bpp F=$f O=2
done
```

## Assembling lookup table

```
params <- fread("output/params.csv")
n_models <- nrow(params)

lookup_r <- unique(params$r)
lookup_s <- unique(params$s)
t <- max(as.numeric(strsplit(as.character(unique(params$Ts)), ";")[[1]]))
sampling_times <- seq(from=t, to=0, by=-500) # time_steps = 500 in opt.bpp

# Het stats over time (include steady-state and present-time)
hrs <- as.data.frame(matrix(nrow=n_models, ncol=length(sampling_times)))
hls <- as.data.frame(matrix(nrow=n_models, ncol=length(sampling_times)))

names(hrs) <- as.character(sampling_times)
names(hls) <- as.character(sampling_times)

hrs$Order <- 0
hls$Order <- 0

for(i in 1:length(lookup_r)) {
  for(j in 1:length(lookup_s)) {
    fnames <- list.files(paste("output/r_", lookup_r[i],
                               "/s_", lookup_s[j], sep=""),
                         pattern="*_e_*.*_time.txt", full.names=TRUE)

    idx <- 0
    hets <- NULL
    for(l in 1:length(fnames)) {
      moms <- fread(fnames[l], header=F)
      str_sides <- strsplit(fnames[l], "_O_")[[1]]
      idx <- as.numeric(strsplit(str_sides[1], "model_")[[1]][2])
      order <- as.numeric(strsplit(str_sides[2], "_e")[[1]][1])
      hets <- rbind.data.frame(hets, moms)
    }
    left <- filter(hets, V1=="Hl_0_0")
    right <- filter(hets, V1=="Hr_0_0")

    # multiple epochs may overlap their output in their last/first time-point
    left <- left[!duplicated(left$V4)]
    right <- right[!duplicated(right$V4)]

    hls[idx,] <- c(t(dplyr::select(left, V3)), order)
    hrs[idx,] <- c(t(dplyr::select(right, V3)), order)
  }
}

lookup_hl <- cbind.data.frame(params, hls)
lookup_hr <- cbind.data.frame(params, hrs)
lookup_tbl <- pivot_longer(lookup_hr, cols=as.character(sampling_times),
                           names_to="Generation", values_to="Hr")
lookup_tbl$Hr <- as.numeric(lookup_tbl$Hr)
lookup_tbl_hl <- pivot_longer(lookup_hl, cols=as.character(sampling_times),
```

```
                              names_to="Generation", values_to="Hl")
lookup_tbl$Hl <- as.numeric(lookup_tbl_hl$Hl)
lookup_tbl$Generation <- as.numeric(lookup_tbl$Generation)

# get pi0 from demography
demo_models <- crossing(unique(params$Ns), unique(params$Ts))
names(demo_models) <- c("Ns", "Ts")
num_models <- nrow(demo_models)
pi0 <- as.data.frame(matrix(nrow=num_models, ncol=length(sampling_times)))
names(pi0) <- as.character(sampling_times)

fnames <- list.files("output/demo", pattern="*_e_*.*_time.txt", full.names=TRUE)
hets <- NULL
for(l in 1:length(fnames)) {
  moms <- fread(fnames[l])
  str_sides <- strsplit(fnames[l], "_0_")[[1]]
  order <- as.numeric(strsplit(str_sides[2], "_e")[[1]][1])
  hets <- rbind.data.frame(hets, moms)
}

right <- filter(hets, V1=="Hr_0_0")
right <- right[!duplicated(right$V4)]
pi0[1,] <- t(dplyr::select(filter(right, V1=="Hr_0_0"), V3))

demo_pi0 <- cbind.data.frame(demo_models, pi0)
m_pi0 <- pivot_longer(demo_pi0,
                      cols=as.character(sampling_times),
                      names_to="Generation", values_to="pi0")
m_pi0$Generation <- as.numeric(m_pi0$Generation)
m_pi0 <- dplyr::select(m_pi0, -c(Ns, Ts))

lookup_tbl <- left_join(lookup_tbl, m_pi0, by=c("Generation"))
lookup_tbl$B <- lookup_tbl$Hr / lookup_tbl$pi0
lookup_tbl$piN_pi0 <- lookup_tbl$Hl / lookup_tbl$pi0
lookup_tbl$piN_piS <- lookup_tbl$Hl / lookup_tbl$Hr

lookup_tbl$Generation <- rep(sampling_times, nrow(lookup_tbl) /
                             length(sampling_times))

fwrite(lookup_tbl, "lookup_tbl.csv.gz", compress="gzip")
head(lookup_tbl, n=5)
```

```
## # A tibble: 5 x 14
##   Ns      Ts        r     s    uL    uR Order Generation      Hr      Hl     pi0
##   <chr>   <chr> <dbl> <dbl> <dbl> <dbl> <dbl>      <dbl>   <dbl>   <dbl>   <dbl>
## 1 1000;1~ 0;25~  1e-8 -1e-3  1e-8  1e-8    70      25000 2.00e-4 9.99e-6 2.00e-4
## 2 1000;1~ 0;25~  1e-8 -1e-3  1e-8  1e-8    70      24500 1.60e-4 8.90e-6 1.60e-4
## 3 1000;1~ 0;25~  1e-8 -1e-3  1e-8  1e-8    70      24000 1.29e-4 8.73e-6 1.29e-4
## 4 1000;1~ 0;25~  1e-8 -1e-3  1e-8  1e-8    70      23500 1.05e-4 8.79e-6 1.05e-4
## 5 1000;1~ 0;25~  1e-8 -1e-3  1e-8  1e-8    70      23000 8.62e-5 8.89e-6 8.62e-5
## # i 3 more variables: B <dbl>, piN_pi0 <dbl>, piN_piS <dbl>
```

## Plotting

```
lookup_tbl <- fread("lookup_tbl.csv.gz")

scaleFUN <- function(x) sprintf("%.5f", x)

# raising B-values to 1000th power
p1 <- ggplot(data=dplyr::filter(lookup_tbl, s==-1e-5),
             aes(x=Generation, y=B^1000, color=as.factor(r))) +
  theme_bw() + geom_line(linewidth=1.25) + facet_wrap(~s) +
  scale_y_log10(labels=scaleFUN) +
  labs(title=NULL, x=NULL, y="B") +
  theme(axis.title=element_text(size=16),
        axis.text=element_text(size=14),
        axis.text.x=element_blank(),
        axis.text.y=element_text(size=14),
        strip.text.x=element_text(size=13),
        strip.text.y=element_text(size=13),
        legend.position="none")

p2 <- ggplot(data=dplyr::filter(lookup_tbl, s==-1e-4),
             aes(x=Generation, y=B^1000, color=as.factor(r))) +
  theme_bw() + geom_line(linewidth=1.25) + facet_wrap(~s) +
  scale_y_log10(labels=scaleFUN) +
  labs(title=NULL, x=NULL, y="B") +
  theme(axis.title=element_text(size=16),
        axis.text=element_text(size=14),
        axis.text.x=element_blank(),
        axis.text.y=element_text(size=14),
        strip.text.x=element_text(size=13),
        strip.text.y=element_text(size=13),
        legend.position="none")

p3 <- ggplot(data=dplyr::filter(lookup_tbl, s==-1e-3),
             aes(x=Generation, y=B^1000, color=as.factor(r))) +
  theme_bw() + geom_line(linewidth=1.25) + facet_wrap(~s) +
  scale_x_continuous(breaks=seq(0, 25000, 5000)) +
  scale_y_log10(labels=scaleFUN) +
  labs(title=NULL, x="Generations ago", y="B") +
  theme(axis.title=element_text(size=16),
        axis.text=element_text(size=14),
        axis.text.x=element_text(size=14),
        axis.text.y=element_text(size=14),
        strip.text.x=element_text(size=13),
        strip.text.y=element_text(size=13),
        legend.position="bottom")

cp1 <- plot_grid(p1, p2, p3, ncol=1, rel_heights=c(1, 1, 1.35), align='v')
save_plot("two-locus_time_bottleneck.pdf", cp1, base_height=15, base_width=12)
```