AlleFreqPower

Katie Lotterhos

4/28/2019

setwd("/Users/lotterhos/Documents/GitHub/LifeCyclePower/src")

FST function

This function returns FST for a haploid sample with 2 alleles. It's not designed for pool-seq data but just used here as an approximation.

```
WC_FST_FiniteSample_Haploids_2AllelesB_MCW<-function(AllCounts){
  #Input a matrix of the counts of each allele (columns) in each population (rows)
  #returns vector instead of list of Fst values, according to Weir

  n_pops<-dim(AllCounts)[1]
  r<-n_pops
  counts1 <- AllCounts[,1]
  sample_sizes <- rowSums(AllCounts)
  n_ave <- mean(as.numeric(sample_sizes))
  n_c = (n_pops*n_ave - sum(sample_sizes^2)/(n_pops*n_ave))/(n_pops-1)
  p_freqs = counts1/sample_sizes
  p_ave = sum(sample_sizes*p_freqs)/(n_ave*n_pops)
  #note: this differs slightly from mean(p_freqs) in R
  He <- 2*p_ave*(1-p_ave)

  s2 = sum(sample_sizes*(p_freqs - p_ave)^2)/((n_pops-1)*n_ave)
  #note: this differs slightly from var(p_freqs) in R

  T1 <- s2 - 1/(n_ave-1)*(p_ave*(1-p_ave) -(s2*(r-1)/r))
  T2 <- (n_c - 1)*(p_ave*(1-p_ave))/(n_ave-1) + (1 + (r-1)*(n_ave-n_c)/(n_ave-1))*s2/r

  FST <- T1/T2

  return(c(He,p_ave, FST, T1, T2))
}
```

Allele frequency functions

This function takes the following as input:

- `p0` true allele frequency at time 0
- `p1` true allele frequency at time 1
- `numind` is number of individual sampled at time 0 and again at time 1
- `numreads` is the number of reads

It then uses a random binomial model to samples alleles from individuals based on `numind` and `p0` (as would be done sampling individuals from the population), and then from that frequency, sample chromosomes from individuals based on `numreads` (as would be done during library prep and sequencing).

The function returns the following:

- `p0` the true initial allele frequency
- `p1` the true final allele frequency

- `p0.samp2` the observed initial allele frequency after sequencing
- `p1.samp2` the observed final allele frequency after sequencing
- `dp.true` the true change in allele frequency
- `dp.1` the observed change in allele frequency after sampling individuals but before sequencing
- `dp.2` *the observed change in allele frequency after sampling individuals and sequencing*
- `FST.0` The true FST between the intial and final sample based on true allele frequency
- `FST.2` The observed FST between the intial and final sample based on sampled allele frequencies (p0.samp2 and p1.samp2)
- `He.2` The observed He between the intial and final sample based on sampled allele frequencies

Figure out power as function of intial allele frequency

This code evaluates power for different initial parameters.

For a given number of individuals sampled and read depth, this code evaluates the power to detect an allele frequency change based on 1000 replicate datasets for each level of initial minor allele frequency (from 0.01 to 0.5) and for a given level of allele frequency change (from 0.01 to 0.4).

For each level, an observed allele frequency change is calculated as described above. Then, a null distribution is created based on the possible distribution of allele frequencies that could be observed based on sampling and the null hypothesis of no change in allele frequency between the timepoints. Finally, the power of the design is calculated as the proportion of times the observed allele frequency change is outside the 95% quantiles of the null distribution.

- This code takes a while to run, so I just run it once and the file is saved, and then loaded in the next chunk below.*

```
numreads_levels = c(50, 100, 200, 300, 400)
numind_levels <- c(100, 1000, 10000)
```

```
dp <- seq(0.01, 0.4, 0.01)
power<- c()
dat <- data.frame()
length(p0)

k <- 0

for (a in 1: length(numreads_levels)){
  numreads=numreads_levels[a]
  for (b in 1: length(numind_levels)){
    numind = numind_levels[b]
    for (i in 1:length(dp)){
    print(i)
      for (p_0 in seq(0.01, 0.5, 0.01)){
        k <- k+1
        p0 <- rep(p_0, 1000)
        delta_p <- dp[i]
        p1 <- p0 + delta_p

        x <- data.frame(t(mapply(allelefreqchange.pooled, p0,p1,numind=numind ,numreads=numreads)))
        x <- sapply( x, as.numeric )
        x <- data.frame(x)
    #head(x)
        null <- data.frame(t(mapply(allelefreqchange.pooled, p0+0.00001,p0,numind=numind ,numreads=numre
        null <- sapply( null, as.numeric )
        null <- data.frame(null)
    #head(null)

        cutoff <- quantile(null$dp.2,0.95, type=1)
        power <- sum(x$dp.2 > cutoff)/length(x$dp.2)
        dat0 <- data.frame(numind, numreads, p_0, delta_p, power)
        dat <- rbind(dat, dat0)
  }
    }
  }
}

head(dat)
saveRDS(dat,file = "PowerTable.RDS")
```

```
dat_full <- readRDS("PowerTable.RDS")
str(dat_full)
```

```
## 'data.frame':    30000 obs. of  5 variables:
##  $ numind  : num  100 100 100 100 100 100 100 100 100 100 ...
##  $ numreads: num  50 50 50 50 50 50 50 50 50 50 ...
##  $ p_0     : num  0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
##  $ delta_p : num  0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ...
##  $ power   : num  0.081 0.063 0.037 0.065 0.056 0.067 0.065 0.07 0.059 0.072 ...
```

```
levels(factor(dat_full$numind))
```

```
## [1] "100"   "1000"  "10000"
```

```
levels(factor(dat_full$numreads))
```

```
## [1] "50"  "100" "200" "300" "400"
```

```
makeplot <- function(n_ind, n_reads){
  dat1 <- dat_full[dat_full$numind==n_ind & dat_full$numreads==n_reads,]
e <- ggplot(dat1, aes(delta_p, power))
#e + geom_point()#, x, y, alpha, color, fill, shape, size, stroke
#e + geom_point(aes(color=p_0))
#e + geom_point(aes(color=p_0)) + theme_bw()
#e + geom_point(aes(color=p_0)) + theme_classic()
#e + geom_point(aes(color=p_0)) + theme_classic() + labs(x="Change in allele frequency", y="Power")
#e + geom_point(aes(color=p_0)) + theme_classic() + labs(x="Change in allele frequency", y="Power") + g
#e + geom_point(aes(color=p_0)) + theme_classic() + labs(x="Change in allele frequency", y="Power", col
#e + geom_point(aes(color=p_0)) + theme_classic() + labs(x="Change in allele frequency", y="Power", col

e + geom_point(aes(color=p_0)) + theme_classic() + labs(x="Change in allele frequency", y="Power", colo
#+ geom_vline(xintercept=0.1, color="grey")
}
```

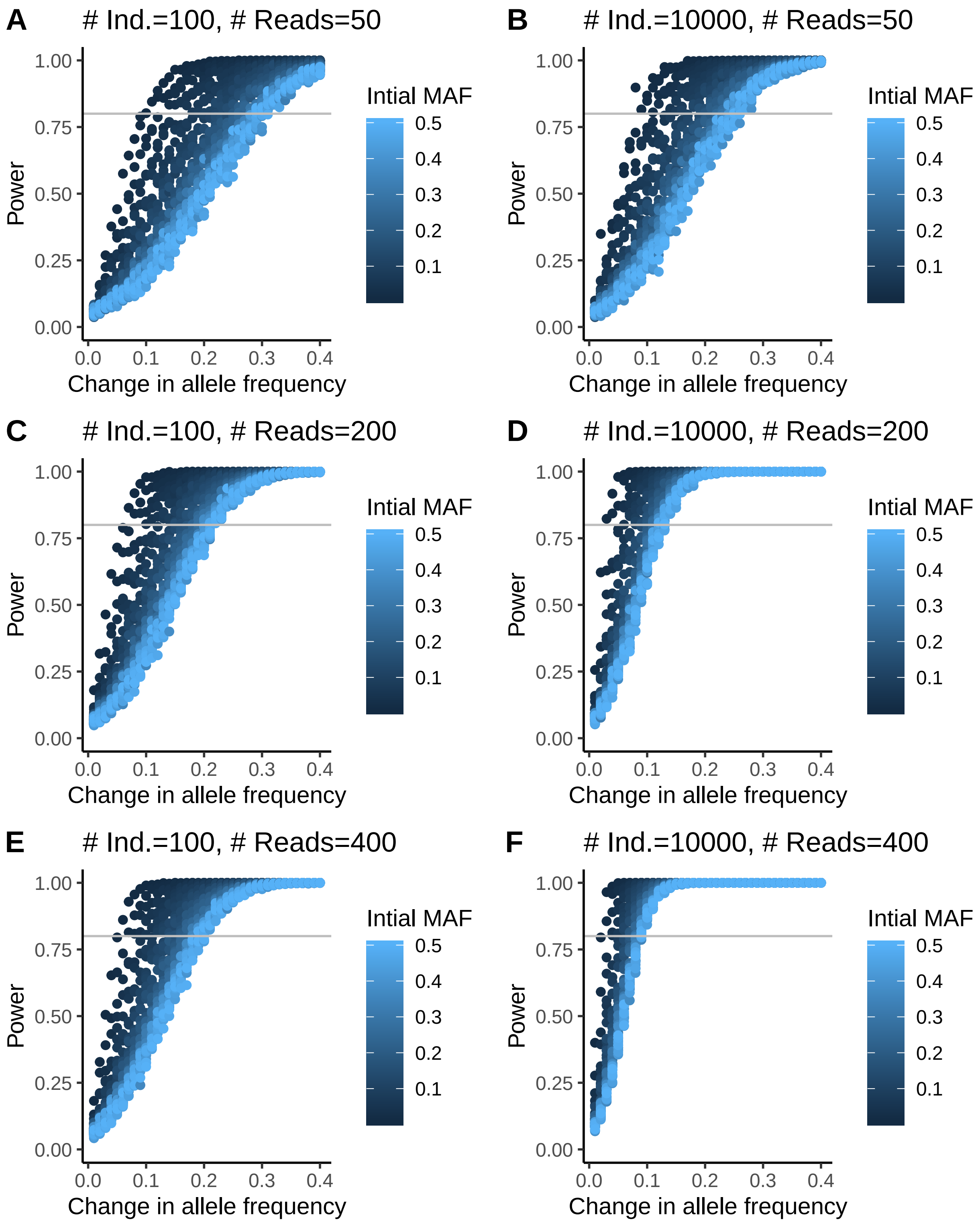
```
#for(a in 1:length(numreads_levels)){
#  for (b in 1:length(numind_levels)){
#    makeplot( numind_levels[b], numreads_levels[a])
#  }
#}

numind_levels
```

```
## [1]   100  1000 10000
```

```r
numreads_levels
```

```
## [1]  50 100 200 300 400
```

```r
# http://www.sthda.com/english/articles/24-ggpubr-publication-ready-plots/81-ggplot2-easy-way-to-mix-mu



i100_r50 <-   makeplot(100, 50)
i10000_r50 <-   makeplot(10000, 50)

i100_r200 <-  makeplot(100, 200)
i10000_r200 <-  makeplot(10000, 200)

i100_r400 <-  makeplot(100, 400)
i10000_r400 <-   makeplot(10000, 400)

  ggarrange(i100_r50, i10000_r50,
            i100_r200, i10000_r200,
            i100_r400, i10000_r400,
          labels = LETTERS[1:6],
          ncol = 2, nrow = 3)
```


A
# Ind.=100, # Reads=50
B
# Ind.=10000, # Reads=50
C
# Ind.=100, # Reads=200
D
# Ind.=10000, # Reads=200
E
# Ind.=100, # Reads=400
F
# Ind.=10000, # Reads=400
Power
1.00
0.75
0.50
0.25
0.00
0.0
0.1
0.2
0.3
0.4
Change in allele frequency
Intial MAF
0.5
0.4
0.3
0.2
0.1