DH_BIO720_DEC5

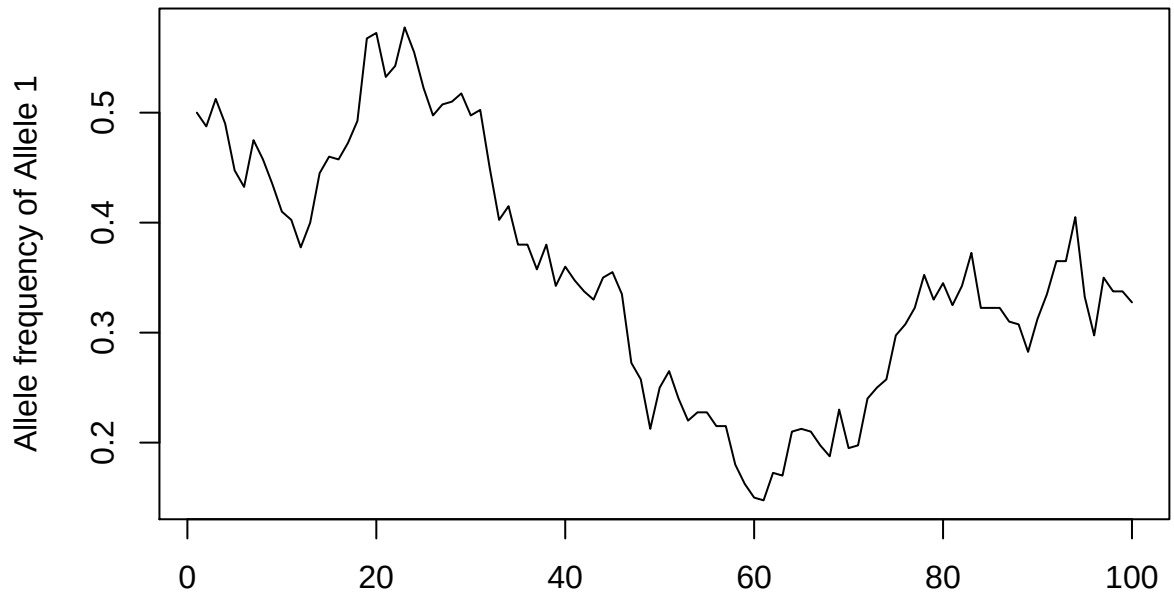
Q2.

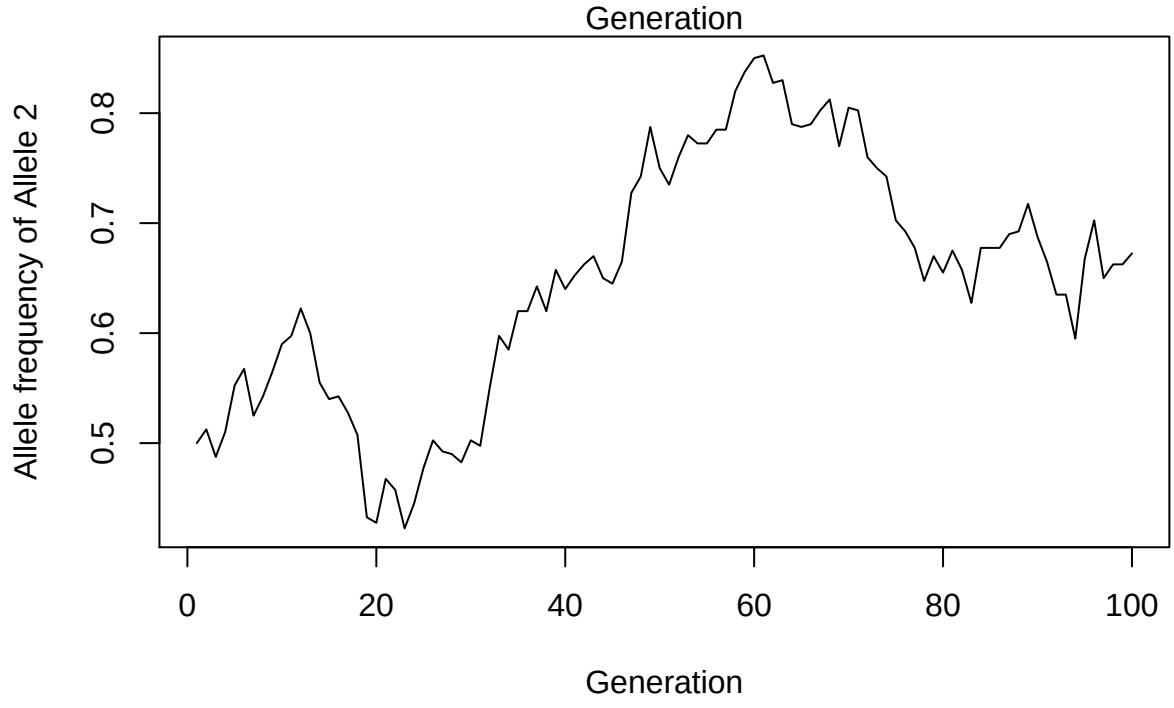
```
diploid_selection <- function(p_0 = 0.4 ,q_0 = 0.6 ,w_aa = 0.9, w_AA = 1.1 , w_Aa = 0.9, n = 100) {
  generations <- 1:n
  p <- rep(NA,n)  # a vector to store allele frequencies
  q <- rep(NA,n)
  w_bar <- rep(NA, n)

  p[1] <- p_0
  q[1]<- q_0


  for ( i in 1:(n-1)) {

        w_bar[i] <- p[i]^2 * w_AA + 2*p[i]*q[i]* w_Aa + q[i]^2 * w_aa
    p[i + 1] <- p[i]^2 * (w_AA / w_bar[i]) + p[i] * q[i] * (w_Aa / w_bar[i] )
    q[i +1] <- 1 - p[i+1]

  }

plot( generations, p, xlab = "Generation", ylab = "Allele Frequency" , pch = 20)

return(p)

}
```

Q3.

```
genetic_drift <- function (A_0=0.5, a_0 =0.5, indiv = 200, n = 100 ){
  size <- 2 * indiv
  A <- rep(NA, n)
  a <- rep(NA, n)

  A[1] <- A_0
  a[1] <- A_0


  for ( i in 1:(n-1)) {
  allele_counts <- sample(c("Allele1", "Allele2"), size, replace = TRUE, prob = c(A[i], a[i]))
  table <- table(allele_counts)
  table <- data.frame(table)
  rownames(table) <- table[,1]
  A[i +1] <- table["Allele1", 2] / size
  a[i + 1] <- table["Allele2", 2] / size


  }


generations <- 1:length(A)
plot(A ~ generations, pch = 20, type = "l",
```

```
     ylab = "Allele frequency of Allele 1",
     xlab = "Generation")

generations <- 1:length(a)
plot(a ~ generations, pch = 20, type = "l",
     ylab = "Allele frequency of Allele 2",
     xlab = "Generation")
}

genetic_drift()
```





Q4.

```r
genetic_drift2 <- function( freq= 0.5, indiv = 200, n = 100, sims = 1000) {

 A <- rep(NA, sims)


  for(t in 1:sims){

    p <- freq


    for ( i in 1:n) {
  allele_counts <- sample(c("Allele1", "Allele2"), indiv * 2, replace = TRUE, prob = c( p, 1 - p))
  p <- length(allele_counts[allele_counts == "Allele1"]) / length(allele_counts)


    }

 A[t] <- p

  }



lost_freq <- A[A == 0]
lost_freq<- length(lost_freq) / 1000
lost_freq

}
genetic_drift2()
```

```
## [1] 0.006
```

```r
genetic_drift2( freq = 0.25)
```

```
## [1] 0.119
```

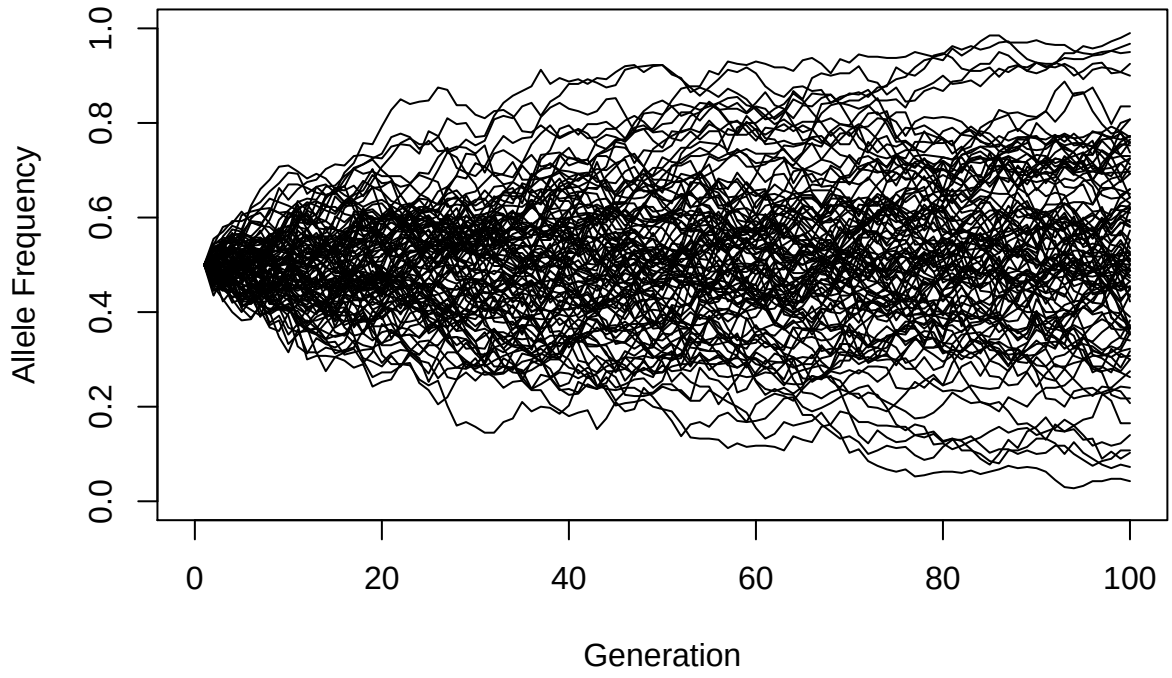
```r
genetic_drift2( freq = 0.1)
```

```
## [1] 0.438
```

Q5.

```r
plot( 0, 0, xlim= c(0, 100), ylim = c( 0,1), type = "n", xlab = "Generation", ylab ="Allele Frequency")

genetic_drift_test <- function (A_0=0.5, a_0 =0.5, indiv = 200, n = 100 ){
  size <- 2 * indiv
  A <- rep(NA, n)
  a <- rep(NA, n)

  A[1] <- A_0
  a[1] <- A_0


  for ( i in 1:(n-1)) {
  allele_counts <- sample(c("Allele1", "Allele2"), size, replace = TRUE, prob = c(A[i], a[i]))
  table <- table(allele_counts)
```

```
  table <- data.frame(table)
  rownames(table) <- table[,1]
  A[i +1] <- table["Allele1", 2] / size
  a[i + 1] <- table["Allele2", 2] / size


  }


generations <- 1:length(A)
lines(A ~ generations, pch = 20, type = "l")

}

replicate(100,genetic_drift_test())
```




```
## [[1]]
## NULL
##
## [[2]]
## NULL
##
## [[3]]
## NULL
##
## [[4]]
## NULL
##
## [[5]]
## NULL
##
## [[6]]
```

```
## NULL
##
## [[7]]
## NULL
##
## [[8]]
## NULL
##
## [[9]]
## NULL
##
## [[10]]
## NULL
##
## [[11]]
## NULL
##
## [[12]]
## NULL
##
## [[13]]
## NULL
##
## [[14]]
## NULL
##
## [[15]]
## NULL
##
## [[16]]
## NULL
##
## [[17]]
## NULL
##
## [[18]]
## NULL
##
## [[19]]
## NULL
##
## [[20]]
## NULL
##
## [[21]]
## NULL
##
## [[22]]
## NULL
##
## [[23]]
## NULL
##
## [[24]]
```

```
## NULL
##
## [[25]]
## NULL
##
## [[26]]
## NULL
##
## [[27]]
## NULL
##
## [[28]]
## NULL
##
## [[29]]
## NULL
##
## [[30]]
## NULL
##
## [[31]]
## NULL
##
## [[32]]
## NULL
##
## [[33]]
## NULL
##
## [[34]]
## NULL
##
## [[35]]
## NULL
##
## [[36]]
## NULL
##
## [[37]]
## NULL
##
## [[38]]
## NULL
##
## [[39]]
## NULL
##
## [[40]]
## NULL
##
## [[41]]
## NULL
##
## [[42]]
```

```
## NULL
##
## [[43]]
## NULL
##
## [[44]]
## NULL
##
## [[45]]
## NULL
##
## [[46]]
## NULL
##
## [[47]]
## NULL
##
## [[48]]
## NULL
##
## [[49]]
## NULL
##
## [[50]]
## NULL
##
## [[51]]
## NULL
##
## [[52]]
## NULL
##
## [[53]]
## NULL
##
## [[54]]
## NULL
##
## [[55]]
## NULL
##
## [[56]]
## NULL
##
## [[57]]
## NULL
##
## [[58]]
## NULL
##
## [[59]]
## NULL
##
## [[60]]
```

```
## NULL
##
## [[61]]
## NULL
##
## [[62]]
## NULL
##
## [[63]]
## NULL
##
## [[64]]
## NULL
##
## [[65]]
## NULL
##
## [[66]]
## NULL
##
## [[67]]
## NULL
##
## [[68]]
## NULL
##
## [[69]]
## NULL
##
## [[70]]
## NULL
##
## [[71]]
## NULL
##
## [[72]]
## NULL
##
## [[73]]
## NULL
##
## [[74]]
## NULL
##
## [[75]]
## NULL
##
## [[76]]
## NULL
##
## [[77]]
## NULL
##
## [[78]]
```

```
## NULL
##
## [[79]]
## NULL
##
## [[80]]
## NULL
##
## [[81]]
## NULL
##
## [[82]]
## NULL
##
## [[83]]
## NULL
##
## [[84]]
## NULL
##
## [[85]]
## NULL
##
## [[86]]
## NULL
##
## [[87]]
## NULL
##
## [[88]]
## NULL
##
## [[89]]
## NULL
##
## [[90]]
## NULL
##
## [[91]]
## NULL
##
## [[92]]
## NULL
##
## [[93]]
## NULL
##
## [[94]]
## NULL
##
## [[95]]
## NULL
##
## [[96]]
```

```
## NULL
##
## [[97]]
## NULL
##
## [[98]]
## NULL
##
## [[99]]
## NULL
##
## [[100]]
## NULL
```

6.

```
 #intercept, slope, sample size(number of observations, residual standard error ( stochastic variation)

stat_pwr <- function (sample_size = 20, intercept = 0.5, slope = 0.1, sd = 2 ){
    length.out <- sample_size
  x <- seq(from = 1, to = 10, length.out = 20)
  y_deterministic <- intercept + slope*x
  y_simulated <- rnorm(length(x), mean = y_deterministic, sd = 2)

  mod_sim <- lm(y_simulated ~ x)
  p_val_slope <- summary(mod_sim)$coef[2,4]
  p_val_slope
}
```

```
#confirm it is doing the same thing

set.seed(720)
stat_pwr()
```

```
## [1] 0.3620625
```

```
stat_pwr()
```

```
## [1] 0.2179742
```

```
set.seed(720)
x <- seq(from =1, to = 10, length.out =  20)
intercept <- 0.5
slope <- 0.1

 y_deterministic <- intercept + slope*x

 y_simulated <- rnorm(length(x), mean = y_deterministic, sd = 2)

 mod_sim <- lm(y_simulated ~ x)
 p_val_slope <- summary(mod_sim)$coef[2,4]
 p_val_slope
```

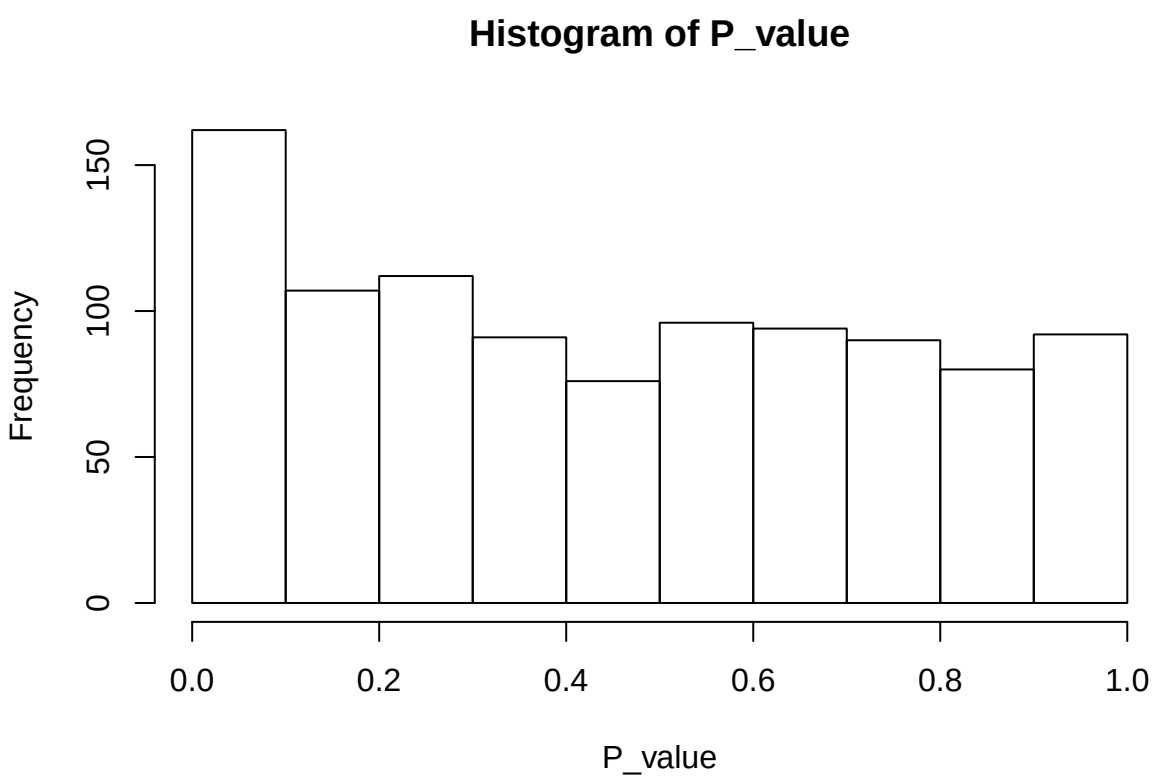
```
## [1] 0.3620625
```

```
x <- seq(from =1, to = 10, length.out =  20)
intercept <- 0.5
```

```r
slope <- 0.1

 y_deterministic <- intercept + slope*x

 y_simulated <- rnorm(length(x), mean = y_deterministic, sd = 2)

 mod_sim <- lm(y_simulated ~ x)
 p_val_slope <- summary(mod_sim)$coef[2,4]
 p_val_slope
```

```
## [1] 0.2179742
```

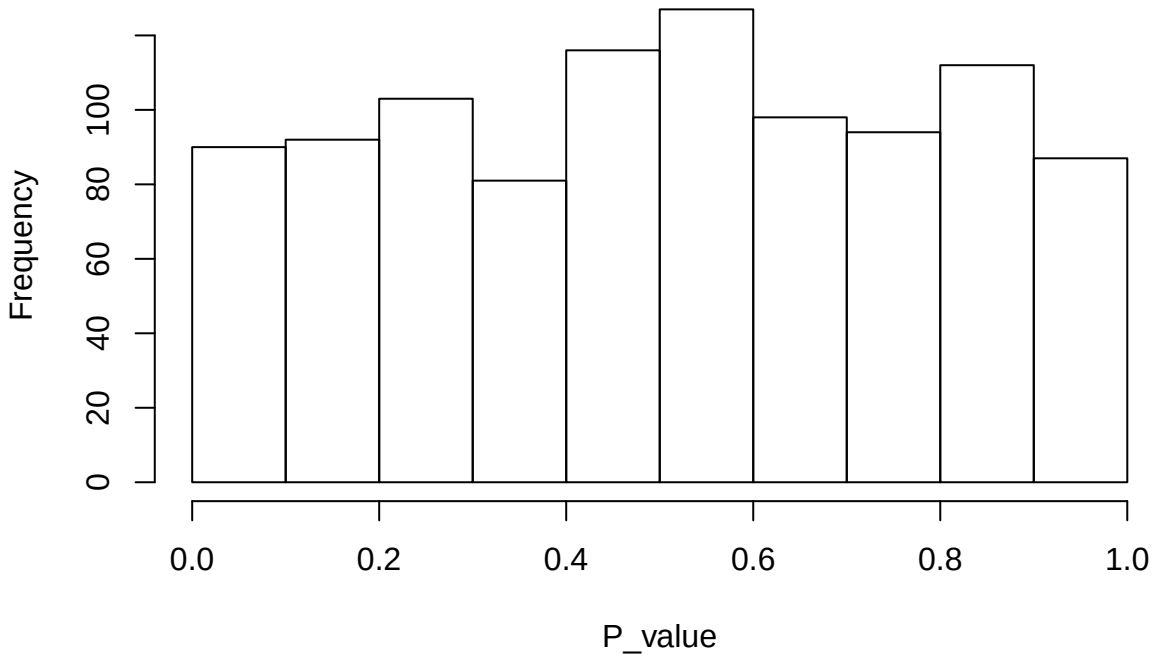
```r
#run 1000 times and generate histogram of p values

 P_value  <- (replicate(1000, stat_pwr()))
 hist(P_value )
```




```r
 #what proportion of times is P value less than 0.05

 rep_proportion <- P_value[P_value < 0.05]
 rep_proportion <- length(rep_proportion) / 1000
 rep_proportion
```

```
## [1] 0.107
```

```r
#redo above but with slope at 0

P_value <- (replicate(1000, stat_pwr(slope = 0)))
hist(P_value)
```




```
rep_proportion <- P_value[P_value < 0.05]
 rep_proportion <- length(rep_proportion) / 100
 rep_proportion
```

```
## [1] 0.48
```

```
sample_size <- seq(10,100,5)
freq <- rep(0, length(sample_size))
for (i in 1:length(sample_size)) {
  p_values <- rep(0, 100)
  for (x in 1:100) {
    p_values[x] <- stat_pwr(intercept = 0.5, slope = 0.1, sample_size[i], sd = 1.5)
  }
  freq[i] <- length(p_values[p_values < 0.05])/length(p_values)
}

freq
```

```
##  [1] 0.08 0.14 0.10 0.08 0.07 0.05 0.09 0.13 0.04 0.04 0.07 0.11 0.12 0.02
## [15] 0.14 0.06 0.14 0.09 0.07
```

```
#plot to get better look
sample_size <- seq(10,100, 5)

#can't figure out using plot() function. try ggplot


#freq <- as.data.frame(freq)
#rownames(freq) <- sample_size
#ggplot(freq, aes(x = sample_size, y = freq)) + geom_bar(stat="identity") +
  #labs(x = "Sample size", y = "Frequency of p-val less than 0.05")
```

```
#should see p-value less than 0.05 increase with sample size
```