Spotify Analysis

Spotify, the popular music streaming service, organizes and classifies songs based on a wide range of properties (variates):

Variate	Description
`genre`	the genre of the track
`year`	the release year of the recording (note that due to vagaries of releases, re-releases, re-issues and general madness, sometimes the release years are not what you'd expect)
`bpm`	beats per minute - the tempo of the song
`energy`	the higher the value the more energetic the song
`danceability`	the higher the value the easier it is to dance to the song
`loudness`	the higher the value the louder the song
`liveness`	the higher the value the more likely the song is a live recording
`valence`	the higher the value the more positive the mood of the song
`duration`	the duration of the song (in seconds)
`acousticness`	the higher the value the more acoustic the song is
`speechiness`	the higher the value the more spoken words the song contains
`popularity`	the higher the value the more popular the song is

Available for us to study is the population of $N = 300$ Billboard Top 30 songs from 2010 - 2019 (inclusive). In addition to the song's title and artist, measurements on each of the 12 variates listed in the table above have also been recorded for each of these songs. This data is available in the `spotify.csv` file.

Consider just the `popularity` scores for the 2018 and 2019 songs. We may refer to these subpopulations as $\mathcal{P}_{2018}$ and $\mathcal{P}_{2019}$. The null hypothesis being tested by a randomization test is the following:

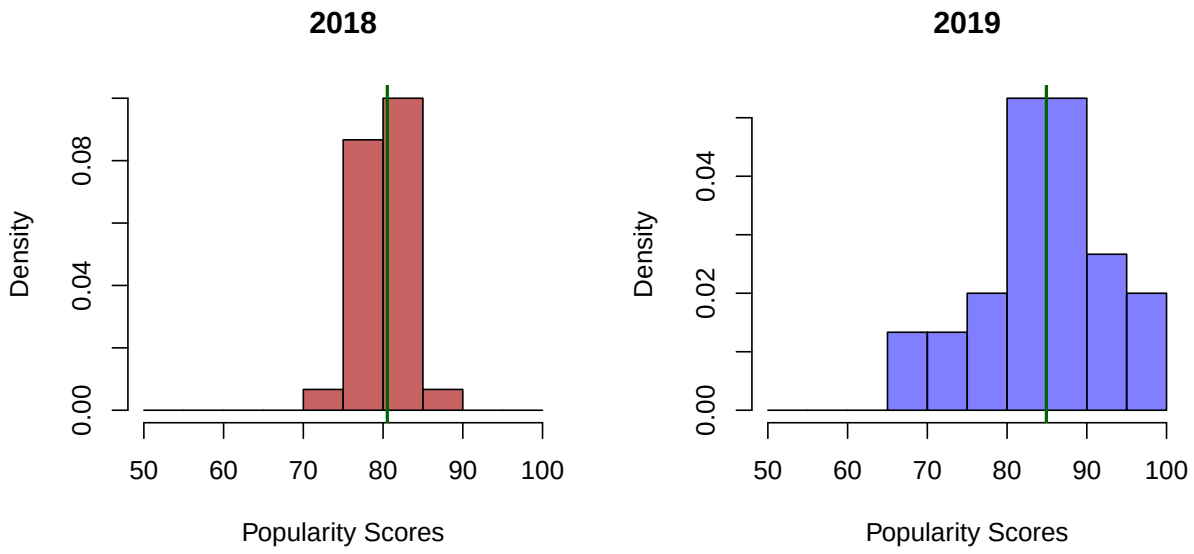
$$H_0 : \mathcal{P}_{2018} \text{ and } \mathcal{P}_{2019} \text{ are drawn from the same population of popularity scores.}$$

In other words, 2018 and 2019 popularity scores are indistinguishable.

(a) First we onstruct two histograms: one of the 2018 popularity scores and the other of the 2019 popularity scores, and plot them next to each in a 1×2 grid.

```
spot <- read.csv("spotify.csv", header = TRUE)
pop <- list(pop1 = spot[spot$year == 2018,],
            pop2 = spot[spot$year == 2019,])

par(mfrow = c(1,2))
hist(pop[[1]]$popularity, col = adjustcolor("firebrick", 0.7),
     freq = FALSE,
     breaks = seq(50, 100, 5),
     xlab = "Popularity Scores",
     main = "2018")
abline(v = mean(pop[[1]]$popularity), col = "darkgreen", lwd=2)
```

```r
hist(pop[[2]]$popularity, col = adjustcolor("blue", 0.5),
     freq = FALSE,
     breaks = seq(50, 100, 5),
     xlab = "Popularity Scores",
     main = "2019")
abline(v = mean(pop[[2]]$popularity), col = "darkgreen", lwd=2)
```



(b) Test the null hypothesis stated above using the discrepancy measure

$$D(\mathcal{P}_{2018}, \mathcal{P}_{2019}) = |\overline{y}_{2018} - \overline{y}_{2019}|$$

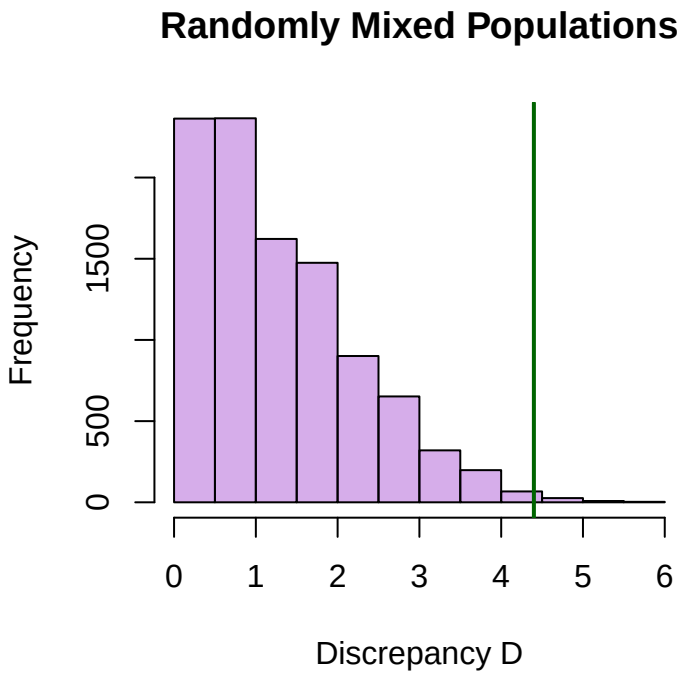
First, we calculate the observed discrepancy:

```r
D <- function(pop){
  ## First sub-population
  P1 <- pop[[1]]$popularity
  m1 <- mean(P1)

  ## Second sub-population
  P2 <- pop[[2]]$popularity
  m2 <- mean(P2)

  ## Calculate and return the Discrepancy
  abs(m1 - m2)
}
d_obs <- D(pop)
print(d_obs)
```

```
## [1] 4.4
```

Now we mix the sub-populations $M = 10,000$ times and plot a histogram of the 10,000 values of $D(\mathcal{P}^{\star}_{2018}, \mathcal{P}^{\star}_{2019})$.

```r
set.seed(341)
diffPops <- sapply(1:10000, FUN = function(...){D(mixRandomly(pop))})
hist(diffPops, breaks=20,
     main = "Randomly Mixed Populations", xlab="Discrepancy D",
     col=adjustcolor("darkorchid", 0.4))
abline(v=D(pop), col = "darkgreen", lwd=2)
```



Now we calculate the p-value:

```r
mean(diffPops >= D(pop))
```

```
## [1] 0.0057
```

With a p-value of 0.0057 we have **strong evidence** against the null hypothesis that 2018 and 2019 popularity scores are indistinguishable, as evaluated by a comparison of averages.

(c) Now we test the null hypothesis stated above using the discrepancy measure

$$D(\mathcal{P}_{2018}, \mathcal{P}_{2019}) = \frac{|\overline{y}_{2018} - \overline{y}_{2019}|}{\sqrt{\frac{\tilde{\sigma}^2}{N_{2018}} + \frac{\tilde{\sigma}^2}{N_{2019}}}}$$

where

$$\tilde{\sigma}^2 = \frac{(N_{2018}-1)\tilde{\sigma}^2_{2018} + (N_{2019}-1)\tilde{\sigma}^2_{2019}}{(N_{2018}-1)+(N_{2019}-1)}$$

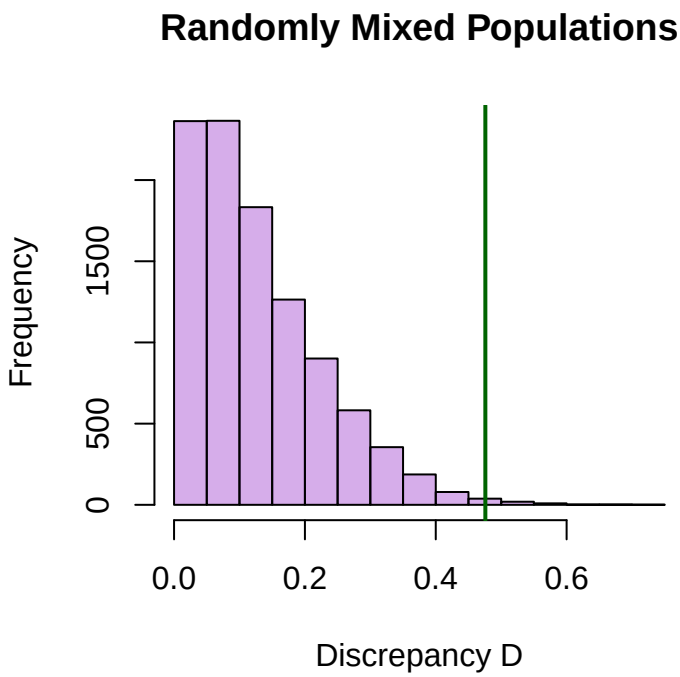
First, let's calculate the observed discrepancy:

```r
D <- function(pop){
  ## First sub-population
  P1 <- pop[[1]]$popularity
  N1 <- length(P1)
  m1 <- mean(P1)
  v1 <- var(P1)

  ## Second sub-population
  P2 <- pop[[2]]$popularity
  N2 <- length(P2)
  m2 <- mean(P2)
  v2 <- var(P2)

  ## Pool the variances
  v <- ((N1 - 1) * v1 + (N2 - 1) * v2)/(N1 + N2 - 2)

  ## Calculate and return the Discrepancy
  abs(m1 - m2) / sqrt((v^2/N1) + (v^2/N2))
}
d_obs <- D(pop)
print(d_obs)
```

```
## [1] 0.4757953
```

Now we mix the sub-populations $M = 10,000$ times and plot a histogram of the 10,000 values of $D(\mathcal{P}^{\star}_{2018}, \mathcal{P}^{\star}_{2019})$.

```r
set.seed(341)
diffPopsT <- sapply(1:10000, FUN = function(...){D(mixRandomly(pop))})
hist(diffPopsT, breaks=20,
     main = "Randomly Mixed Populations", xlab="Discrepancy D",
     col=adjustcolor("darkorchid", 0.4))
abline(v=D(pop), col = "darkgreen", lwd=2)
```



Now we calculate the p-value:

```
mean(diffPopsT >= D(pop))
```

```
## [1] 0.0053
```

With a p-value of 0.0053 we have **strong evidence** against the null hypothesis that 2018 and 2019 popularity scores are indistinguishable, as evaluated by a comparison of standard deviations.

- The following is an R function that may be used to automate bootstrap-t confidence interval calculation:

```
bootstrap_t_interval <- function(S, a, confidence, B, D) {
    ## Inputs: S = an n element array containing the variate values in the
    ## sample a = a scalar-valued function that calculates the attribute a()
    ## of interest confidence = a value in (0,1) indicating the confidence
    ## level B = a numeric value representing the outer bootstrap count of
    ## replicates (used to calculate the lower and upper limits) D = a
    ## numeric value representing the inner bootstrap count of replicates
    ## (used to estimate the standard deviation of the sample attribute for
    ## each (outer) bootstrap sample)

    Pstar <- S
    aPstar <- a(Pstar)
    sampleSize <- length(S)
    ## get (outer) bootstrap values
    bVals <- sapply(1:B, FUN = function(b) {
        Sstar <- sample(Pstar, sampleSize, replace = TRUE)
        aSstar <- a(Sstar)
        ## get (inner) bootstrap values to estimate the SD
        Pstarstar <- Sstar
        SD_aSstar <- sd(sapply(1:D, FUN = function(d) {
            Sstarstar <- sample(Pstarstar, sampleSize, replace = TRUE)
            ## return the attribute value
            a(Sstarstar)
        }))
        z <- (aSstar - aPstar)/SD_aSstar
        ## Return the two values
        c(aSstar = aSstar, z = z)
    })
    SDhat <- sd(bVals["aSstar", ])
    zVals <- bVals["z", ]
    ## Now use these zVals to get the lower and upper c values.
    cValues <- quantile(zVals, probs = c((1 - confidence)/2, (confidence +
        1)/2), na.rm = TRUE)
    cLower <- min(cValues)
    cUpper <- max(cValues)
    interval <- c(lower = aPstar - cUpper * SDhat, middle = aPstar, upper = aPstar -
        cLower * SDhat)
    interval
}
```

We will be dealing with three quartile-based attributes defined in terms of

- the first quartile: $Q_1(y_i)$, i.e., the 25th percentile of the values $\{y_1, \ldots, y_N\}$
- the median: $Q_2(y_i)$, i.e., the 50th percentile of the values $\{y_1, \ldots, y_N\}$
- the third quartile: $Q_3(y_i)$, i.e., the 75th percentile of the values $\{y_1, \ldots, y_N\}$

The attributes of interest are respectively measures of center, spread, and skewness:

- The median: $M(\mathcal{P}) = Q_2(\mathcal{P})$
- The median absolute deviation: $MAD(\mathcal{P}) = Q_2(|y_i - Q_2(\mathcal{P})|)$
- Bowley's skewness coefficient: $BSKEW(\mathcal{P}) = \frac{Q_3(\mathcal{P}) - 2Q_2(\mathcal{P}) + Q_1(\mathcal{P})}{Q_3(\mathcal{P}) - Q_1(\mathcal{P})}$

The following are some R functions that help to calculate these, given an array `y`:

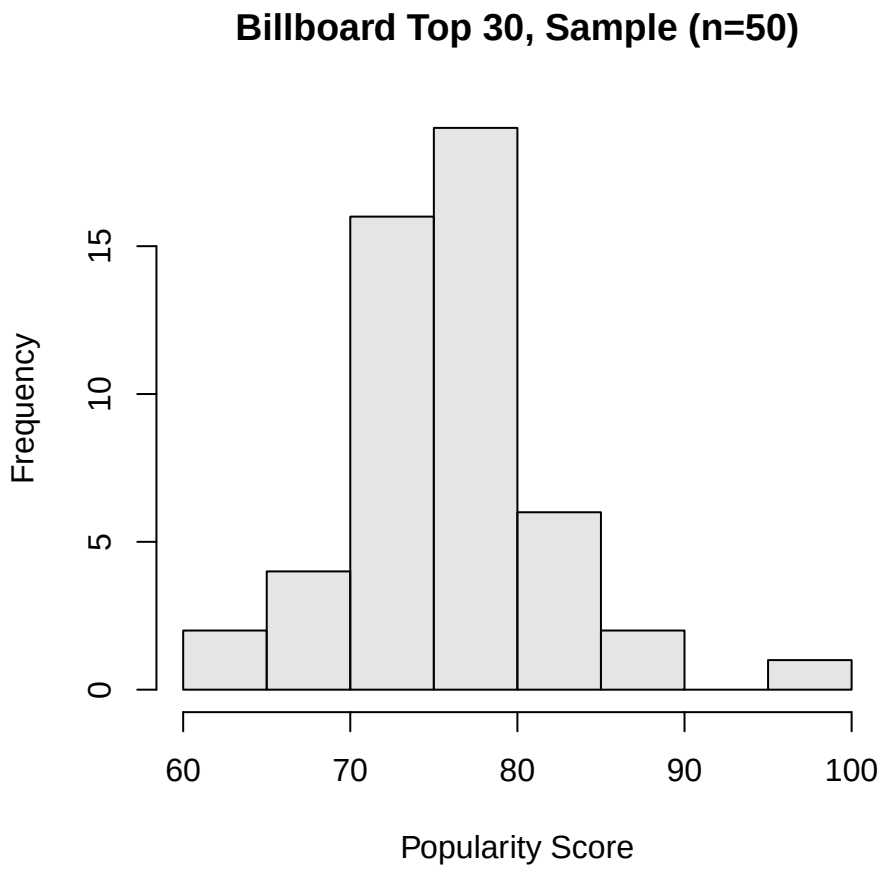
```
mad <- function(y){
  return( median(abs(y - median(y))) )
}
```

```
bskew <- function(y){
  Q1 <- quantile(y, 0.25)
  Q2 <- quantile(y, 0.50)
  Q3 <- quantile(y, 0.75)
  return( as.numeric((Q3 - 2*Q2 + Q1)/(Q3 - Q1)))
}
```

Here we will deal with the Billboard Top 30 data again, and in particular the `popularity` score of the $N = 300$ songs. In this part we will take a sample of size $n = 50$ from the population and for a variety of attributes use this sample data to calculate 95% confidence intervals for the true population value of each of the attributes. We use three bootstrap approaches to calculated this intervals.

(a) We take a sample of size $n = 50$ from this population and plot the `popularity` scores from this sample.

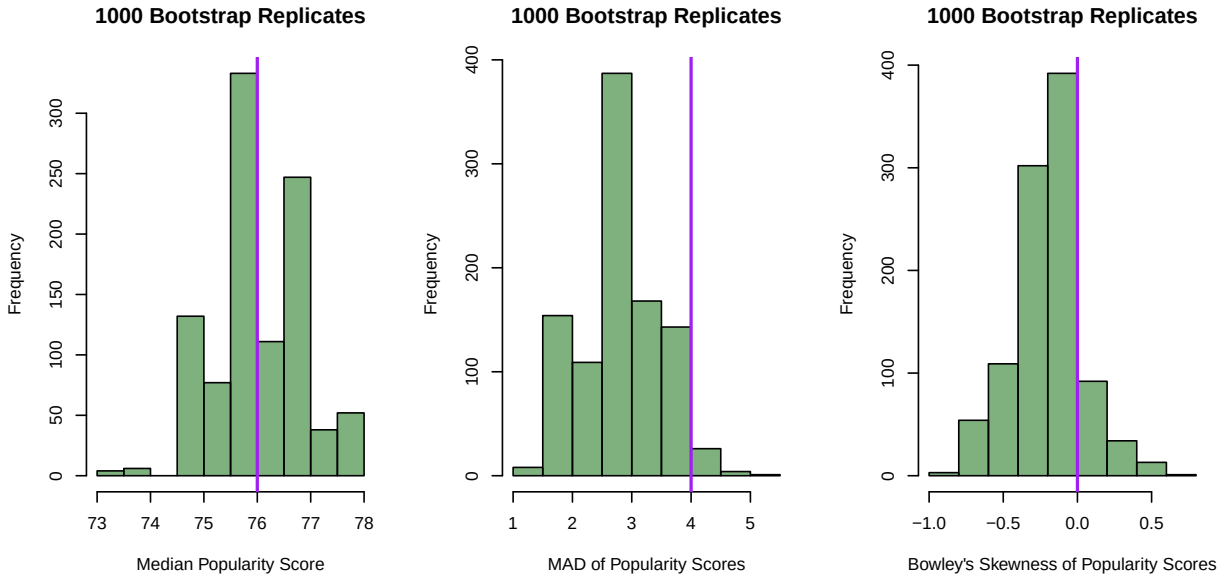
```
# Take the sample
spot <- read.csv("spotify.csv", header = TRUE)
set.seed(341)
N <- dim(spot)[1]
n <- 50
samp.indx <- sample(N, n)
S <- spot[samp.indx, "popularity"]

# Make the plot
hist(S, xlab = "Popularity Score", main = "Billboard Top 30, Sample (n=50)",
     col = adjustcolor("grey", 0.4))
```



(b) By resampling $\mathcal{S}$ with replacement, we construct $B = 1000$ bootstrap samples $S_1^\star, S_2^\star, \ldots, S_{1000}^\star$ and construct histograms of the median, MAD, and Bowley's skewness calculated on each of those samples. Include vertical lines that indicate the median, MAD, and Bowley's skewness in the population.

```
# Get the bootstrap samples
B <- 1000
Sstar <- sapply(1:B, FUN = function(b) {
    sample(S, n, replace = TRUE)
})

# Calculate the median on all of these bootstrap samples
med_star <- apply(X = Sstar, MARGIN = 2, FUN = median)

# Calculate the MAD on all of these bootstrap samples
mad_star <- apply(X = Sstar, MARGIN = 2, FUN = mad)

# Calculate Bowley's skewness on all of these bootstrap samples
bskew_star <- apply(X = Sstar, MARGIN = 2, FUN = bskew)

# Construct histograms for each of these
par(mfrow = c(1,3))
hist(med_star, col = adjustcolor("darkgreen", 0.5), xlab = "Median Popularity Score",
```

```
    main = "1000 Bootstrap Replicates")
abline(v = median(spot$popularity), col = "purple", lwd = 2)

hist(mad_star, col = adjustcolor("darkgreen", 0.5), xlab = "MAD of Popularity Scores",
    main = "1000 Bootstrap Replicates")
abline(v = mad(spot$popularity), col = "purple", lwd = 2)

hist(bskew_star, col = adjustcolor("darkgreen", 0.5), xlab = "Bowley's Skewness of Popularity Scores",
    main = "1000 Bootstrap Replicates")
abline(v = bskew(spot$popularity), col = "purple", lwd = 2)
```



(c) We calculate naive normal theory, quantile method, and bootstrap-t 90% confidence intervals for the population median. For the bootstrap-t interval use $B = 1000$ and $D = 100$.

```
# Naive normal theory
median(S) + qnorm(0.95) * c(-1, 1) * sd(med_star)
```

```
## [1] 74.61612 77.38388
```

```
# Quantile method
c(quantile(med_star, 0.05), quantile(med_star, 0.95))
```

```
##  5% 95%
##  75  78
```

```
# Bootstrap-t
bootstrap_t_interval(S = S, a = median, confidence = 0.90, B = 1000, D = 100)
```

```
##    lower   middle    upper
## 74.00927 76.00000 77.13058
```

(d) We calculate naive normal theory, quantile method, and bootstrap-t 90% confidence intervals for the population MAD. For the bootstrap-t interval use $B = 1000$ and $D = 100$.

```
# Naive normal theory
mad(S) + qnorm(0.95) * c(-1, 1) * sd(mad_star)
```

```
## [1] 1.879404 4.120596
```

```
# Quantile method
c(quantile(mad_star, 0.05), quantile(mad_star, 0.95))
```

```
##  5% 95%
##   2   4
```

```
# Bootstrap-t
bootstrap_t_interval(S = S, a = mad, confidence = 0.90, B = 1000, D = 100)
```

```
##    lower   middle    upper
## 1.851587 3.000000 4.271365
```

(e) We calculate naive normal theory, quantile method, and bootstrap-t 90% confidence intervals for Bowley's population skewness. For the bootstrap-t interval use $B = 1000$ and $D = 100$.

```
# Naive normal theory
bskew(S) + qnorm(0.95) * c(-1, 1) * sd(bskew_star)
```

```
## [1] -0.5455296  0.2378373
```

```
# Quantile method
c(quantile(bskew_star, 0.05), quantile(bskew_star, 0.95))
```

```
##   5%  95%
## -0.6  0.2
```

```
# Bootstrap-t
bootstrap_t_interval(S = S, a = bskew, confidence = 0.90, B = 1000, D = 100)
```

```
##      lower     middle      upper
## -0.5020611 -0.1538462  0.1823933
```