Spotify data exploration

Kanishk Dutta

21/05/2021

For our lab exercise we will be using the Spotify Data.

First let's read in this dataset. I'll use head to ensure that this data is read in correctly and to take a look at some of the column names and the data in this csv.

```r
library('ggplot2')
```

```
## Registered S3 methods overwritten by 'tibble':
##   method     from
##   format.tbl pillar
##   print.tbl  pillar
```

```r
Songs <- read.csv('spotify_songs.csv')
head(Songs)
```

```
##                 track_id                                            track_name
## 1 6f807x0ima9a1j3VPbc7VN I Don't Care (with Justin Bieber) - Loud Luxury Remix
## 2 0r7CVbZTWZgbTCYdfa2P31                      Memories - Dillon Francis Remix
## 3 1z1Hg7Vb0AhHDiEmnDE79l                      All the Time - Don Diablo Remix
## 4 75FpbthrwQmzHlBJLuGdC7                    Call You Mine - Keanu Silva Remix
## 5 1e8PAfcKUYoKkxPhrHqw4x              Someone You Loved - Future Humans Remix
## 6 7fvUMiyapMsRRxr07cU8Ef        Beautiful People (feat. Khalid) - Jack Wins Remix
##       track_artist track_popularity         track_album_id
## 1       Ed Sheeran               66 2oCs0DGTsRO98Gh5ZSl2Cx
## 2         Maroon 5               67 63rPSO264uRjW1X5E6cWv6
## 3     Zara Larsson               70 1HoSmj2eLcsrR0vE9gThr4
## 4 The Chainsmokers               60 1nqYsOef1yKKuGOVchbsk6
## 5    Lewis Capaldi               69 7m7vv9wlQ4i0LFuJiE2zsQ
## 6       Ed Sheeran               67 2yiy9cd2QktrNvWC2EUi0k
##                                           track_album_name
## 1 I Don't Care (with Justin Bieber) [Loud Luxury Remix]
## 2                       Memories (Dillon Francis Remix)
## 3                       All the Time (Don Diablo Remix)
## 4                           Call You Mine - The Remixes
## 5               Someone You Loved (Future Humans Remix)
## 6     Beautiful People (feat. Khalid) [Jack Wins Remix]
##   track_album_release_date playlist_name            playlist_id playlist_genre
## 1               2019-06-14     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 2               2019-12-13     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 3               2019-07-05     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 4               2019-07-19     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 5               2019-03-05     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
## 6               2019-07-11     Pop Remix 37i9dQZF1DXcZDD7cfEKhW            pop
##   playlist_subgenre danceability energy key loudness mode speechiness
```

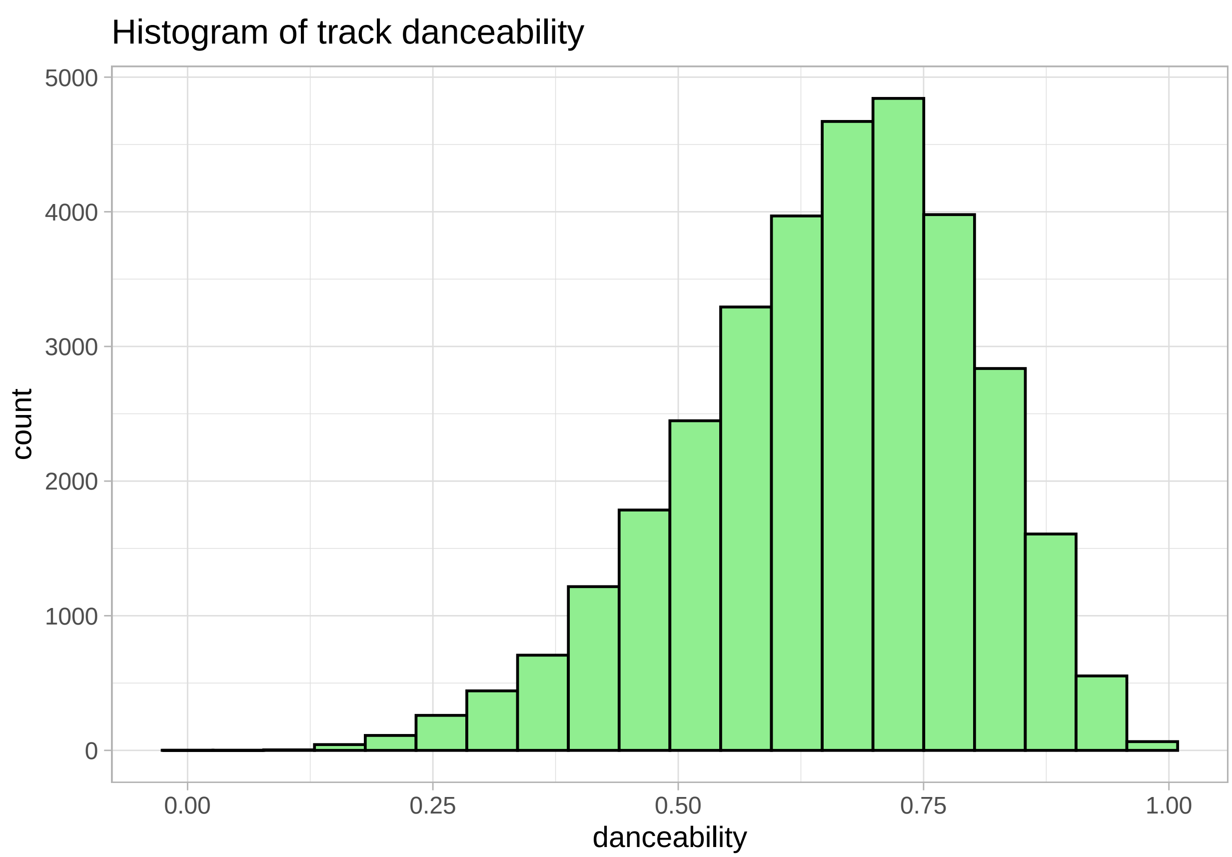
```
## 1          dance pop        0.748  0.916   6   -2.634    1       0.0583
## 2          dance pop        0.726  0.815  11   -4.969    1       0.0373
## 3          dance pop        0.675  0.931   1   -3.432    0       0.0742
## 4          dance pop        0.718  0.930   7   -3.778    1       0.1020
## 5          dance pop        0.650  0.833   1   -4.672    1       0.0359
## 6          dance pop        0.675  0.919   8   -5.385    1       0.1270
##   acousticness instrumentalness liveness valence   tempo duration_ms
## 1       0.1020         0.00e+00   0.0653   0.518 122.036      194754
## 2       0.0724         4.21e-03   0.3570   0.693  99.972      162600
## 3       0.0794         2.33e-05   0.1100   0.613 124.008      176616
## 4       0.0287         9.43e-06   0.2040   0.277 121.956      169093
## 5       0.0803         0.00e+00   0.0833   0.725 123.976      189052
## 6       0.0799         0.00e+00   0.1430   0.585 124.982      163049
```

Question 1

Continuous variable being chosen: 'danceability'

We will plot the histogram for this danceability variable below.

```
ggplot(Songs) + geom_histogram(aes(x = danceability,y=stat(count)),
fill = 'Light Green', colour = 'black', alpha=1, bins =20) +
    ggtitle('Histogram of track danceability') + theme_light()
```



1.1

Examining the histogram for the danceability variable, it is apparent that the graph is left skewed. More tracks have a higher levels of danceability rather than having low levels of danceability (which makes sense, artists will choose to make their tracks more danceable)

Examining the tails of this distribution there is a slight irregularity in the thickness at the right tail. (In normal distribution tails are to decrease uniformly). Thinking about this it does make sense as there will still be a substantial number of songs with a high amount of danceability rather than a substantial number with low danceability (left tail)

Just by looking at this histogram it would be very hard to locate the quartiles of the plot with accuracy. If we were to estimate with just the hist plot 25%: 0.5, Median: 0.65, 75%: 0.78. But these quartiles should be calculated for better accuracy or evaluated on quantile plots.

We can take a look at these quartiles by using the function below as our histogram does not give us an accurate understanding on it's own:

```
quantile(Songs$danceability)   # finding the quartiles
```

```
##    0%   25%   50%   75%  100%
## 0.000 0.563 0.672 0.761 0.983
```

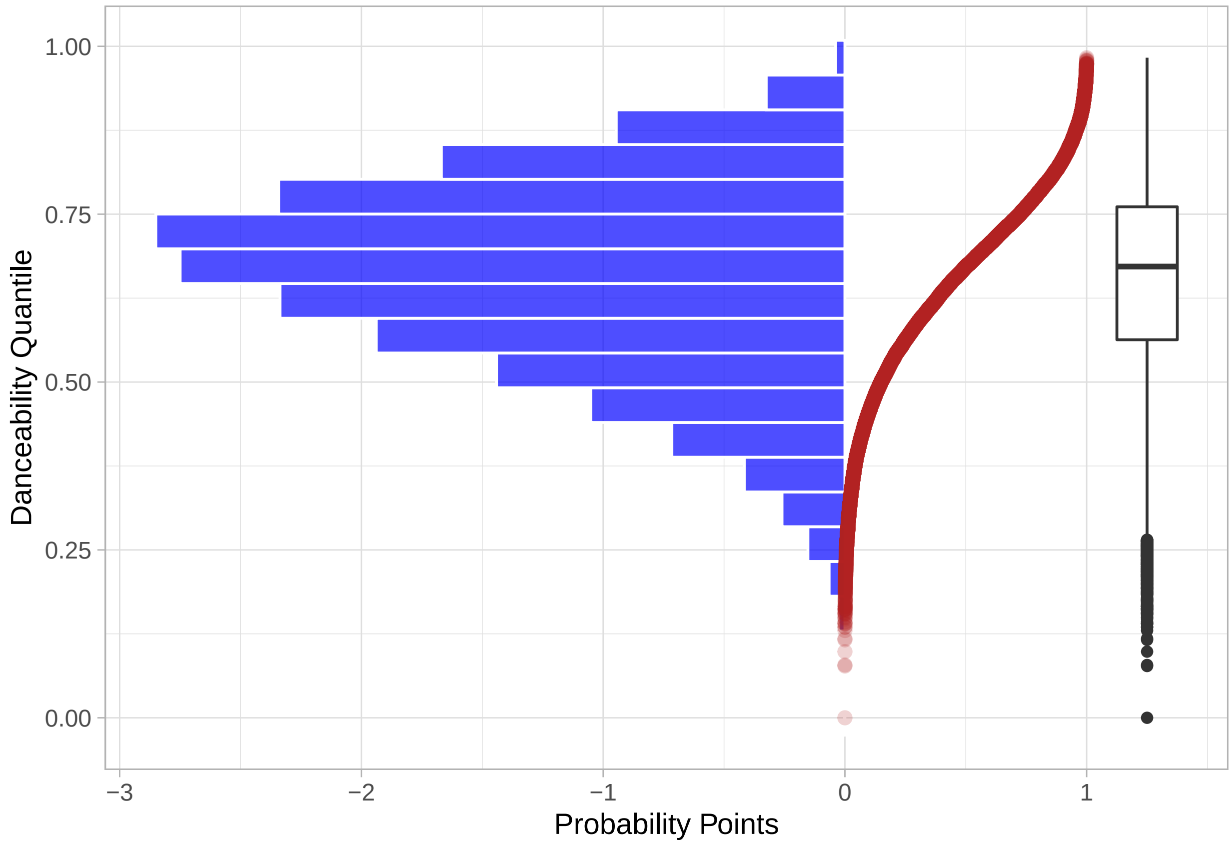
1.2

Plotting the quantile plot for the danceability variable below

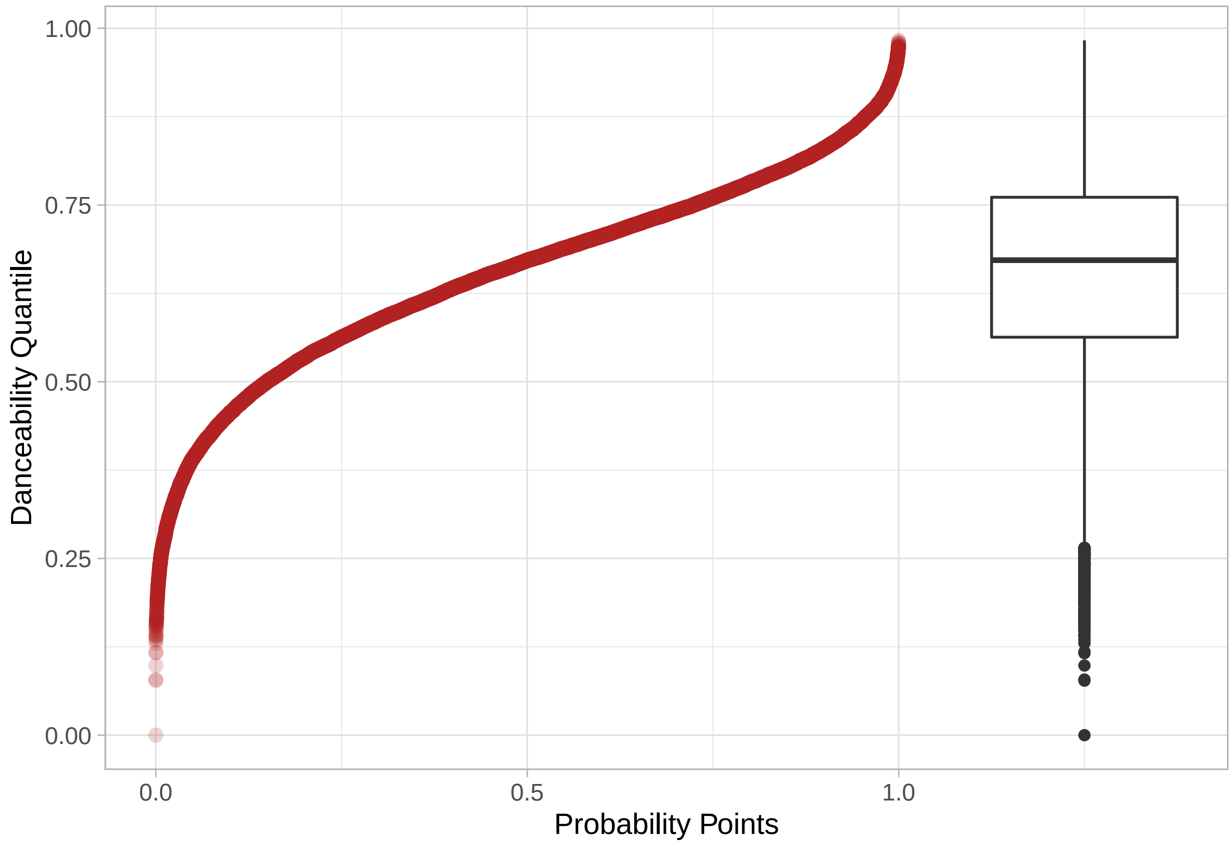
Below we have plotted the histogram with the quantile (having values sorted within prob points of 0 to 1) alongside the boxplot.

Below that is also the quantile plot with just the boxplot for a less cluttered view.

```
ggplot(Songs) +
geom_histogram(aes(y=danceability, x = -..density..), alpha = .7, colour = 'white', bins=20,fill = 'blu
geom_point(aes(y=sort(danceability), x=ppoints(danceability) ), colour='firebrick', alpha = .2, size =2
geom_boxplot(aes(y=danceability,x = 1.25), width=.25)  +
  labs(x='Probability Points', y='Danceability Quantile') + theme_light()
```




```r
ggplot(Songs) +
geom_point(aes(y=sort(danceability), x=ppoints(danceability) ),colour='firebrick', alpha = .2, size =2 )
  geom_boxplot(aes(y=danceability,x = 1.25), width=.25)  +
  labs(x='Probability Points', y='Danceability Quantile') + theme_light()
```



Examining the quantile plot we see that it is slightly left-skewed. This is because there is a sparsity in points at the bottom (of y-axis) and towards the top the points are flatter (with a slight steepness at the top, thus only slightly left-skewed)

It is much easier to estimate where the quartiles are using the plot as we have the prob points on our x-axis to aid us.

```
The Median is at about 0.66, the first quartile at about 0.55 and the third
quartile at approx 0.76.
```

1.3

Below we utilize the fourmoments function to calculate the fourmoments for the danceability variable from the spotify data.

```
# adding the fourmoments function which we will use to learn more about our data
library('moments')
fourmoments <- function(rv){
  c('Mean' = mean(rv),
    'Variance' = var(rv),
    'Skewness' = skewness(rv),
    'Kurtosis' = kurtosis(rv))
}

fourmoments(Songs$danceability)
```

```
##        Mean    Variance    Skewness    Kurtosis
##  0.65484952  0.02104975 -0.50446539  3.01001783
```

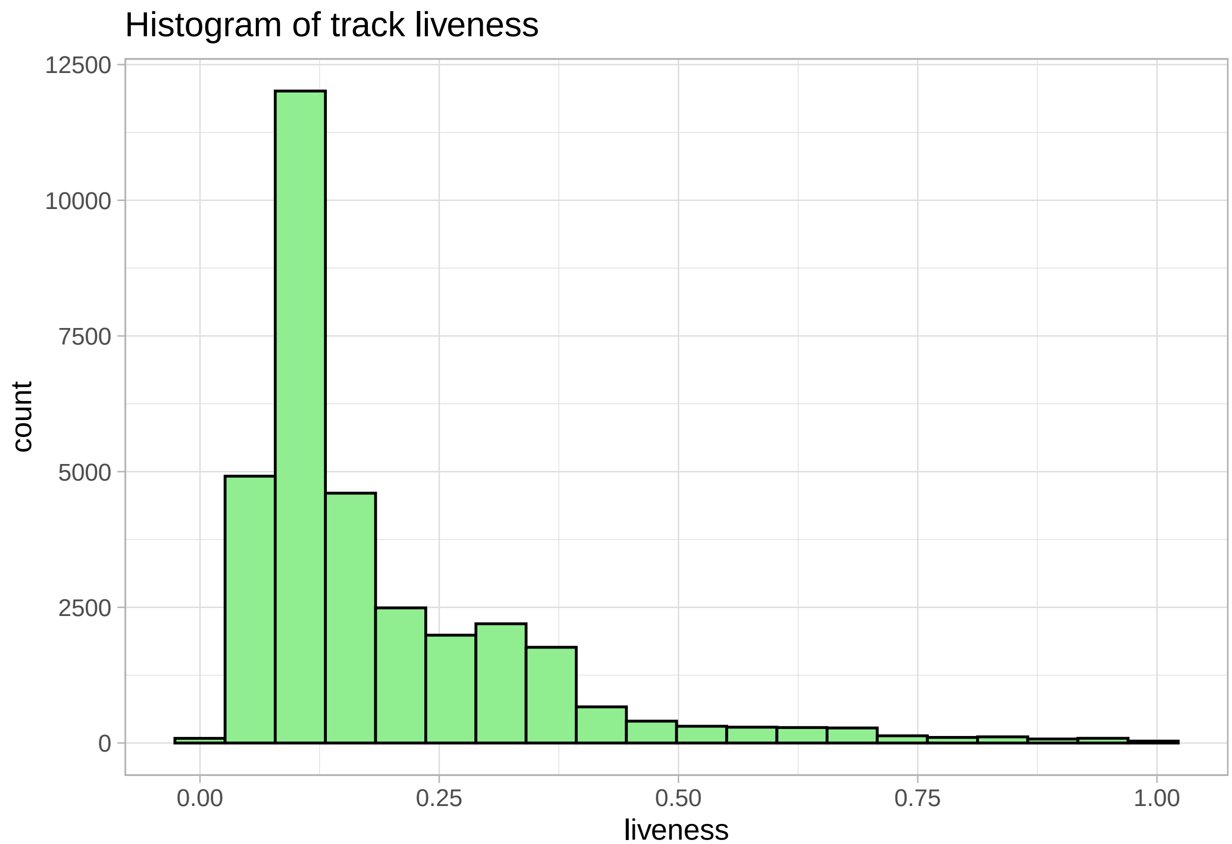
Examining the skewness of this variable we have a value of -0.5044, which is indicating that this is a left skewed distribution.

In terms of the thickness of the tails, we examine the Kurtosis which is of a value of 3.01. The Kurtosis of a normal is 3, thus this being slightly higher this distribution is leptokurtic, having slightly thicker tails than normal.

Question 2

Below we have chosen the variable ‘liveness’ which produces a right skewed distribution. Below is the histogram to confirm this

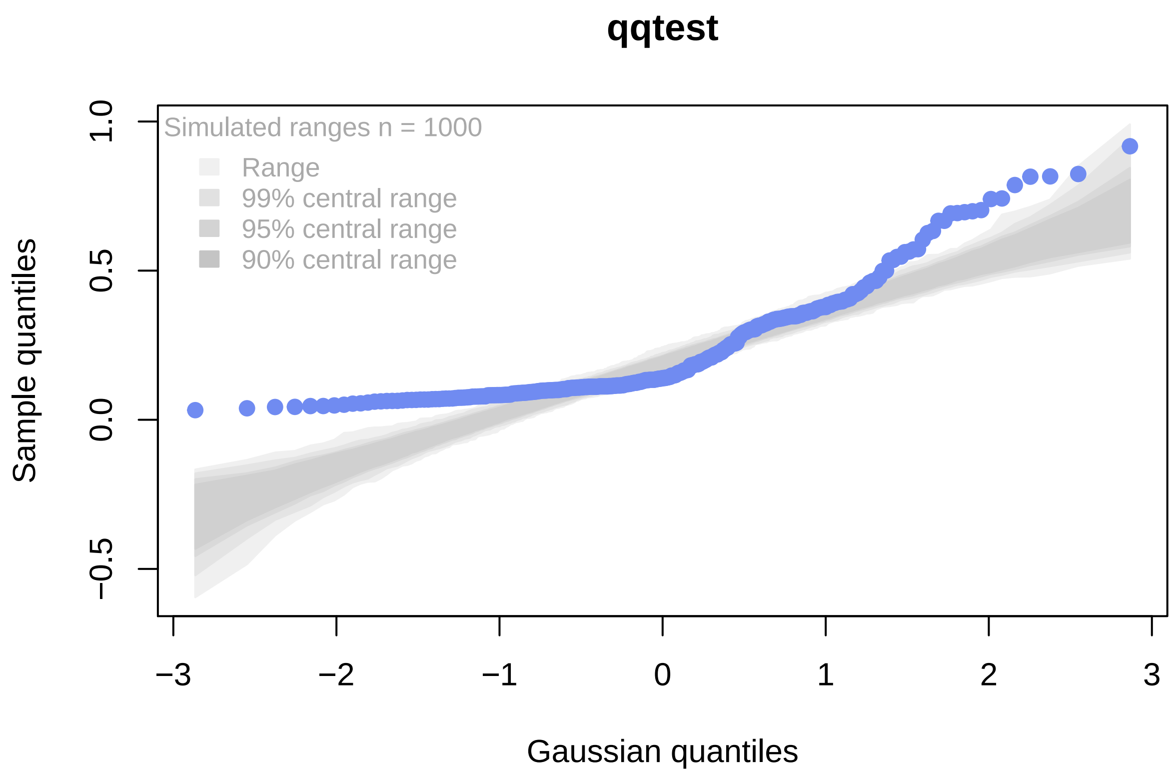
```
ggplot(Songs) + geom_histogram(aes(x = liveness,y=stat(count)),
fill = 'Light Green', colour = 'black', alpha=1, bins =20) +
    ggtitle('Histogram of track liveness') + theme_light()
```



Now to use qqtest to evaluate how this variable fits within different types of distributions.

Firstly checking the normal

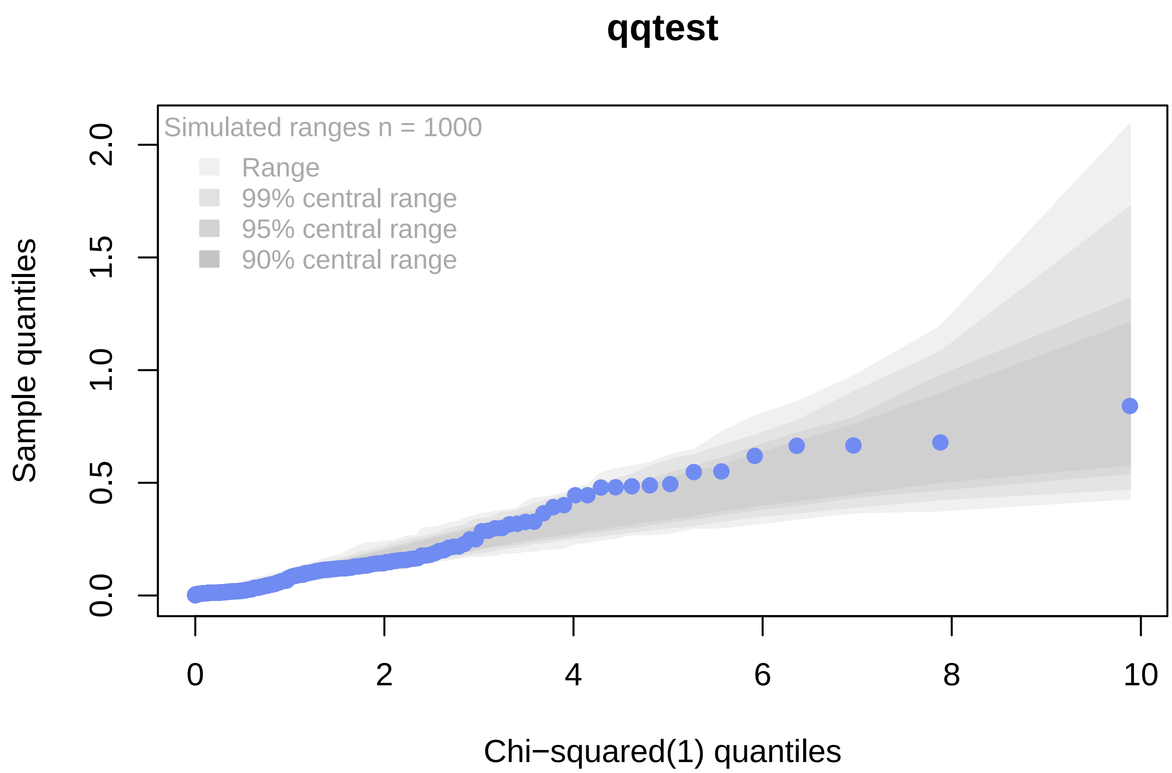
```
library('qqtest')
qqtest(tail(Songs$liveness,300),dist = 'normal')
```



Evaluating the normal distribution it is hard to say that this of normal dist, at both tails the points are out of the tolerable range.

Now evaluating the chi-squared plot

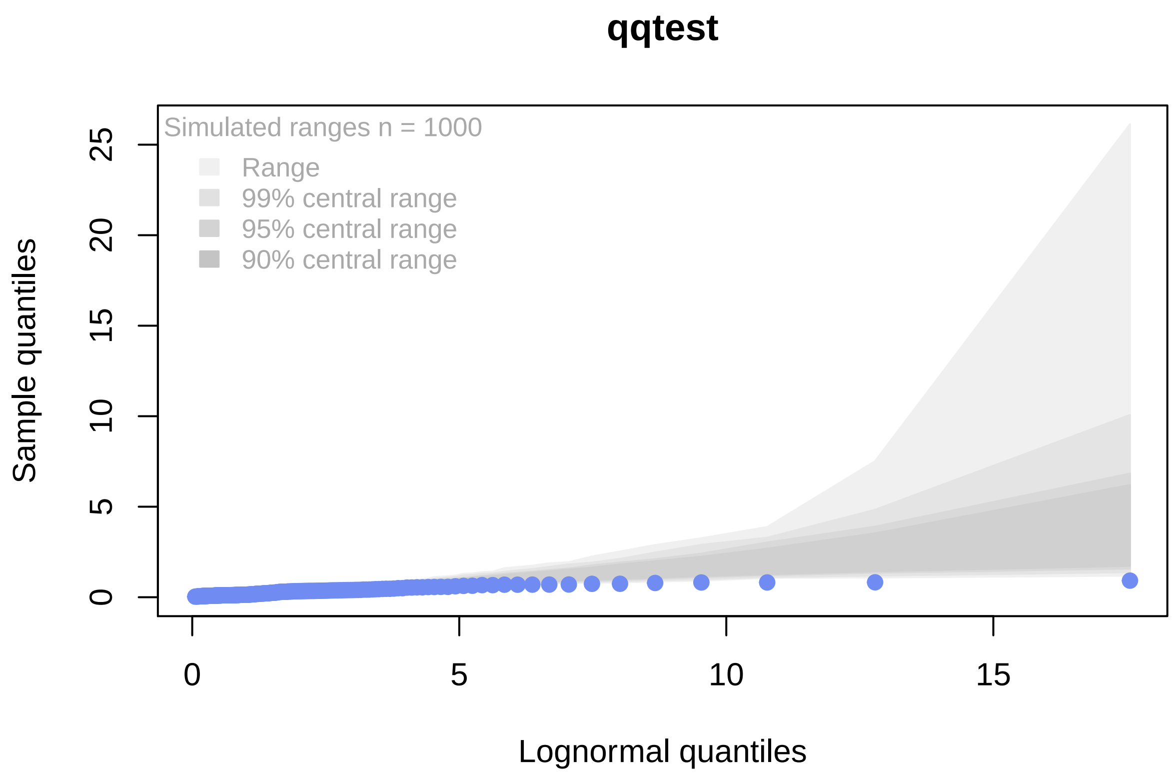
```
library('qqtest')
qqtest((tail(Songs$liveness,300))^2,dist = 'chi-squared')
```



Chi squared seems like a good fit, all of the points from our dataset are within the ranges of the function. (With furthest point bleeding just at edge)

Let's continue to eval the other functions, log-normal below

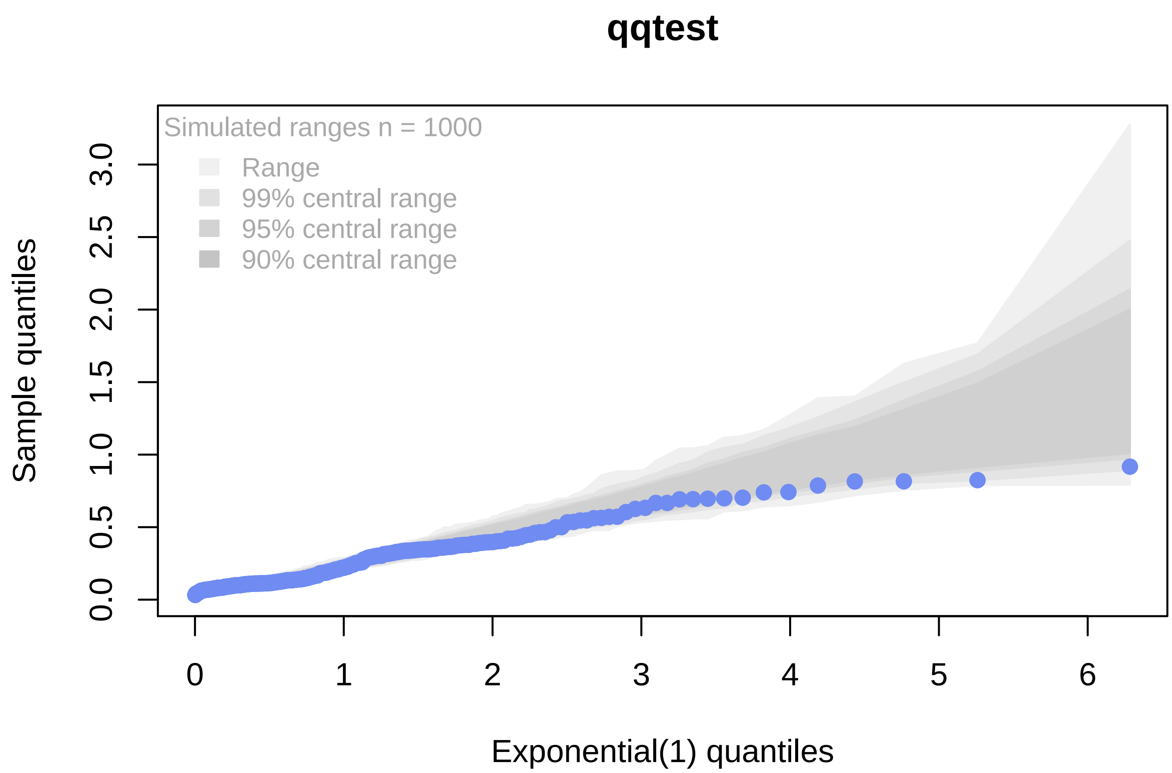
```
library('qqtest')
qqtest(tail(Songs$liveness,300),dist = 'log-normal')
```



Log-normal is not a great fit for our data with many datapoints outside of the ranges.

Finally, we can evaluate the exponential dist.

```
library('qqtest')
qqtest(tail(Songs$liveness,300),dist = 'exponential')
```



The exponential (along with chi-squared) also has all of our data-points in the ranges. Both have slight deviations from the central range, we will continue using the exponential.

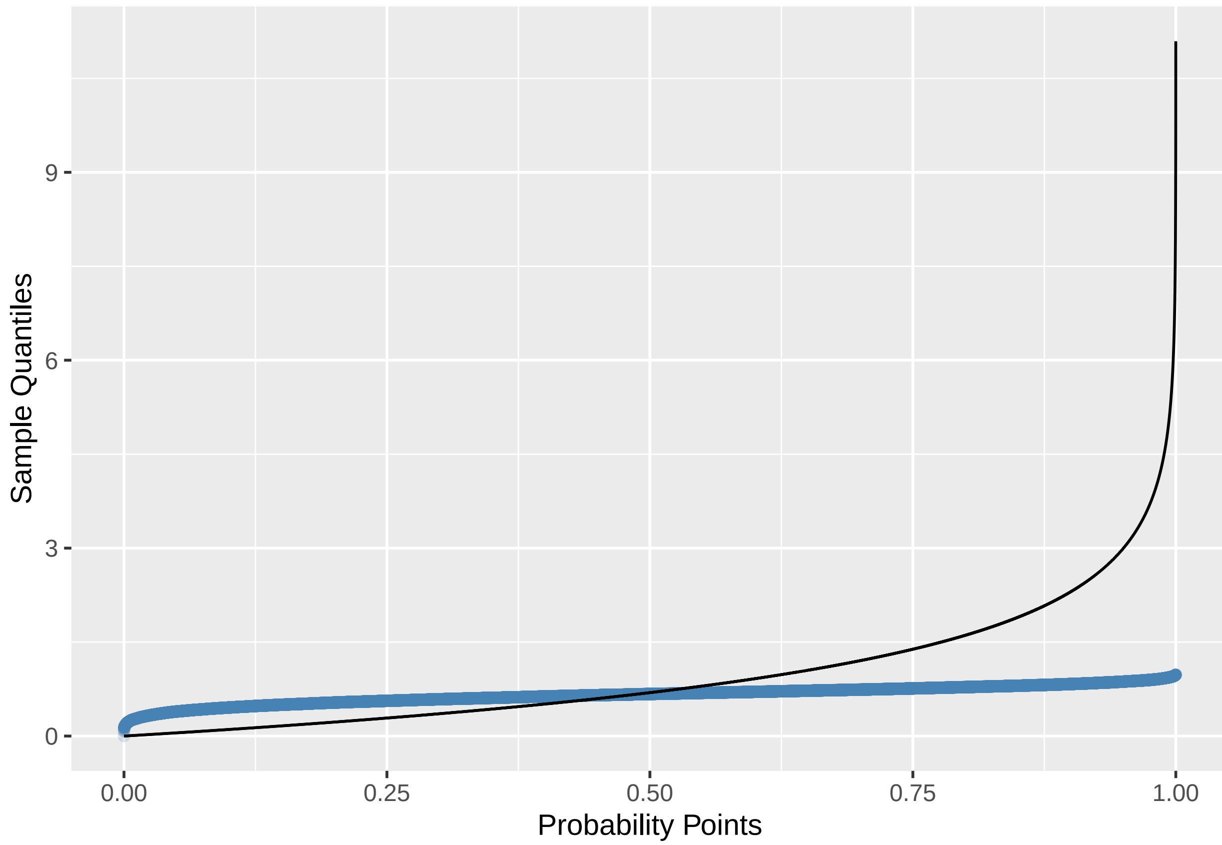
Below we will plot our ideal quantile with the sample quantile. We use qexp below.

```
library('plotly')
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
series <- ((Songs$danceability))
probs    <- ppoints(series)
q.exp <- qexp(probs)
q.sample <- sort((Songs$danceability))


ggplot() +
  geom_point(aes(y = q.sample, x= probs) ,colour = 'steelblue', alpha = .2) +
```

```r
geom_line(aes(y = q.exp, x = probs),colour = "black") +
  labs(x = 'Probability Points', y= 'Sample Quantiles')
```



Comparing the ideal line with the sample quantile line, they follow a relativiley close relation until prob 0.6 (approx) after having a high level of deviation as the sample does not follow the ideal exponential. This is in line with what we saw from the qqtest as the data points were scarce and not in the central range towards the end of the exponential quantiles.