Python and R Visualization Problems

Nane Mambreyan

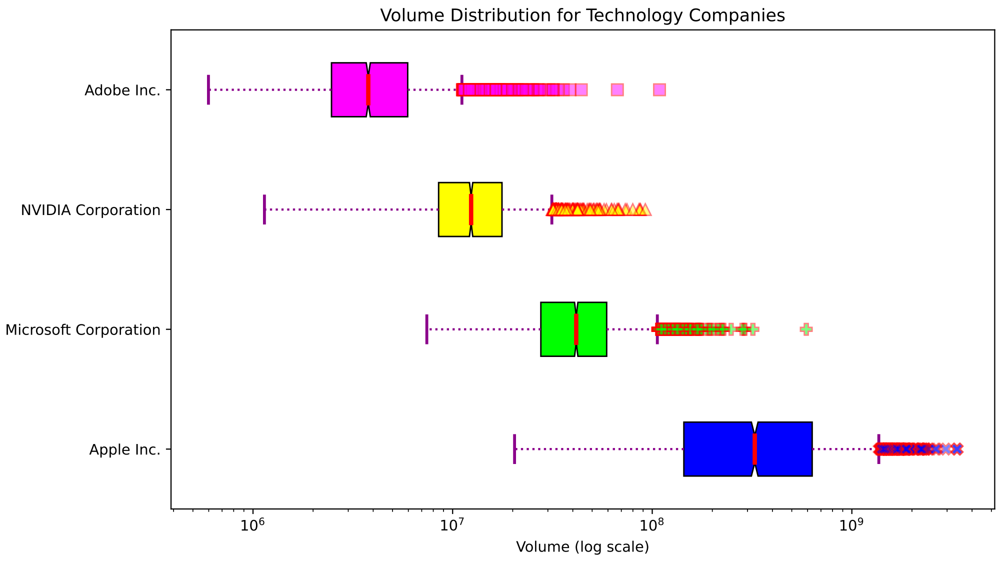
```
library(dplyr)
library(ggplot2)
library(gridExtra)
library(grid)
library(reshape2)
library(reticulate)
library(scales)
library(tidyr)
library(tidyverse)
library(ggridges)
library(moments)
library(glue)
library(VGAM)
```

```
import os
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.ticker as ticker
import pprint
```

Problem 1

```

symbol_to_name_mapping = pd.read_csv("StockIT/Technology Sector List.csv")

symbol_to_name = dict(zip(symbol_to_name_mapping['Symbol'], symbol_to_name_mapping['Name']))
# pprint.pprint(symbol_to_name)

list_of_companies = ["AAPL", "MSFT", "NVDA", "ADBE"]


# Loading data for each company into a dictionary
data_dict = {}  # Dict for storing data for each company
company_labels = []  # List for storing real names of companies

for company_symbol in list_of_companies:
    file_path = f"StockIT/Technology Companies/{company_symbol}.csv"
    # Check if the file exists
    if os.path.exists(file_path):
        company_data = pd.read_csv(file_path)

        # Drop NAs
        company_data.dropna(inplace=True)

        # Store in dict
        data_dict[company_symbol] = company_data

        # Get the real name of the company given its company_symbol
        company_labels.append(symbol_to_name.get(company_symbol))
    else:
        print("File not found")

# saving volume data separately for convenience
volume_data = [data_dict[symbol]['Volume'] for symbol in list_of_companies]
```

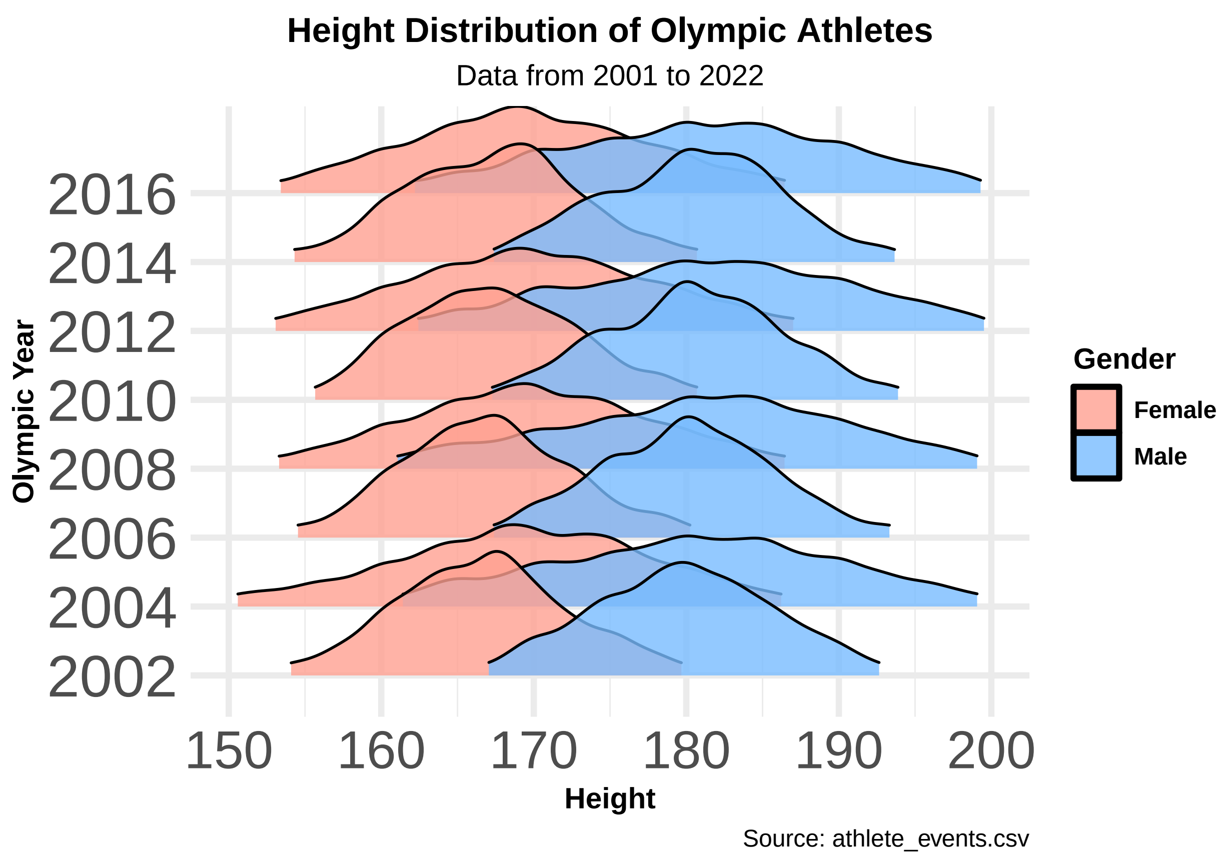
```
# initializing figure and axes
fig = plt.figure(figsize =(12, 7))
ax = fig.add_subplot(111)

# Set color of text labels and tick texts
_= plt.setp(ax.get_xticklabels(), color='black')
_= plt.setp(ax.get_yticklabels(), color='black')

# Creating boxplot of volume data with given labels
bp = ax.boxplot(volume_data, labels=company_labels, vert=False,
                patch_artist = True, notch ='True')

# setting labels,
# _= ensures that python doesn't take it as unassigned var and doesn't return None
_ = plt.title('Volume Distribution for Technology Companies', color='black')
_ = plt.xlabel('Volume (log scale)', color='black')

# to be used
colors = ['#0000FF', '#00FF00', '#FFFF00', '#FF00FF']
markers = ["X", "P", "^", "s"]

# sets the predefined colors for median lines
# and patches <=> boxes
for patch, median, color in zip(bp['boxes'], bp['medians'], colors):
    _ = patch.set_facecolor(color)
    _ = median.set(color='red', linewidth=3)


# sets colorful outliers based on the respective company's color
for flier, color, marker in zip(bp['fliers'], colors, markers):
  _ = flier.set(marker = marker, alpha = 0.5, markerfacecolor = color,
            markeredgecolor = "red", markersize = 8)


# changing color and linewidth of whiskers(lines coming out of boxes)
# and caps(line for each whisker)
for whisker, cap in zip(bp['whiskers'], bp['caps']):
    _ = whisker.set(color='#8B008B', linewidth=1.5, linestyle=':')
    _ = cap.set(color='#8B008B', linewidth=2)

# adjusts spacing for labels
plt.subplots_adjust(left=0.24)

# sets x-axis (volume) to a log scale
plt.xscale('log')

# gca <=> get current axes
# sets the aspect ratio of the current axes
plt.gca().set_aspect(0.6, adjustable='box')

plt.style.use('classic')

# shows plot
plt.show()
```



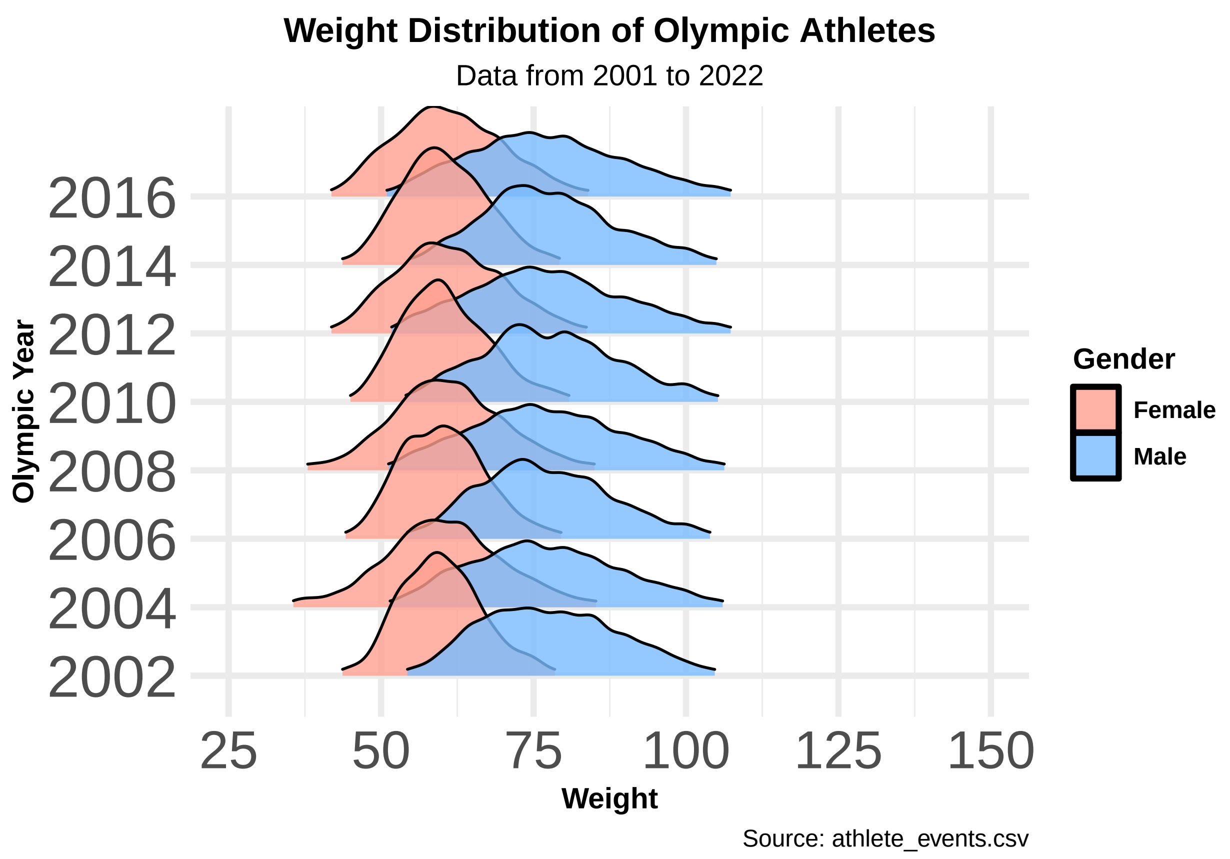
Problem 2

```
# load
df_athlete <- read.csv('athlete_events.csv')
# str(df_athlete)
```

```r
# selecting mentioned columns,
# dropping nas and filtering accordingly
filtered_df<-df_athlete %>%
  select(c(Sex, Age, Height, Weight, Year)) %>%
  drop_na() %>%
  filter(Year > 2000 & Year < 2023)



ggplot(filtered_df, aes(x = Height, y= factor(Year), fill=Sex)) +  # ridges according Years
  geom_density_ridges(alpha = 0.8, scale = 1.8,rel_min_height = 0.1) + # plotting trimmed density_ridge
  labs(title = 'Height Distribution of Olympic Athletes', fill = 'Gender',
       subtitle = 'Data from 2001 to 2022', caption = 'Source: athlete_events.csv',
       x = 'Height', y = 'Olympic Year') + # setting labs
  scale_x_continuous(breaks=seq(150, 200, 10), limits = c(150, 200)) + # changing x values
  scale_fill_manual(values=c('#FFA191','#78BCFF'), labels = c('Female', 'Male')) + #changing color
  theme_minimal() +
  theme(panel.grid.major.x = element_line(size=1.1),
        panel.grid.major.y = element_line(size=1.1),
        axis.text.x = element_text(size=20),
        axis.text.y = element_text(size=22),
        plot.title = element_text(face = 'bold', hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        plot.caption = element_text(hjust = 1),
        axis.title.x = element_text(face = 'bold'),
        axis.title.y = element_text(face = 'bold'),
        legend.title = element_text(face='bold'),
        legend.text = element_text(face='bold'),
        legend.key = element_rect(linewidth=1)) # making all non-stat changes to resemble given plot
```



```r
# same as above for weight
ggplot(filtered_df, aes(x = Weight, y= factor(Year), fill=Sex)) +
  geom_density_ridges(alpha = 0.8, scale = 1.8,rel_min_height = 0.05) +
  labs(title = 'Weight Distribution of Olympic Athletes', fill = 'Gender',
       subtitle = 'Data from 2001 to 2022', caption = 'Source: athlete_events.csv',
       x = 'Weight', y = 'Olympic Year') +
  scale_x_continuous(breaks=c(25,50,75,100,125,150), limits = c(25, 150)) +
  scale_fill_manual(values=c('#FFA191','#78BCFF'), labels = c('Female', 'Male')) +
  theme_minimal() +
  theme(panel.grid.major.x = element_line(size=1.1),
        panel.grid.major.y = element_line(size=1.1),
        axis.text.x = element_text(size=20),
        axis.text.y = element_text(size=22),
        plot.title = element_text(face = 'bold', hjust = 0.5),
        plot.subtitle = element_text(hjust = 0.5),
        plot.caption = element_text(hjust = 1),
        axis.title.x = element_text(face = 'bold'),
        axis.title.y = element_text(face = 'bold'),
        legend.title = element_text(face='bold'),
        legend.text = element_text(face='bold'),
        legend.key = element_rect(linewidth=1))
```

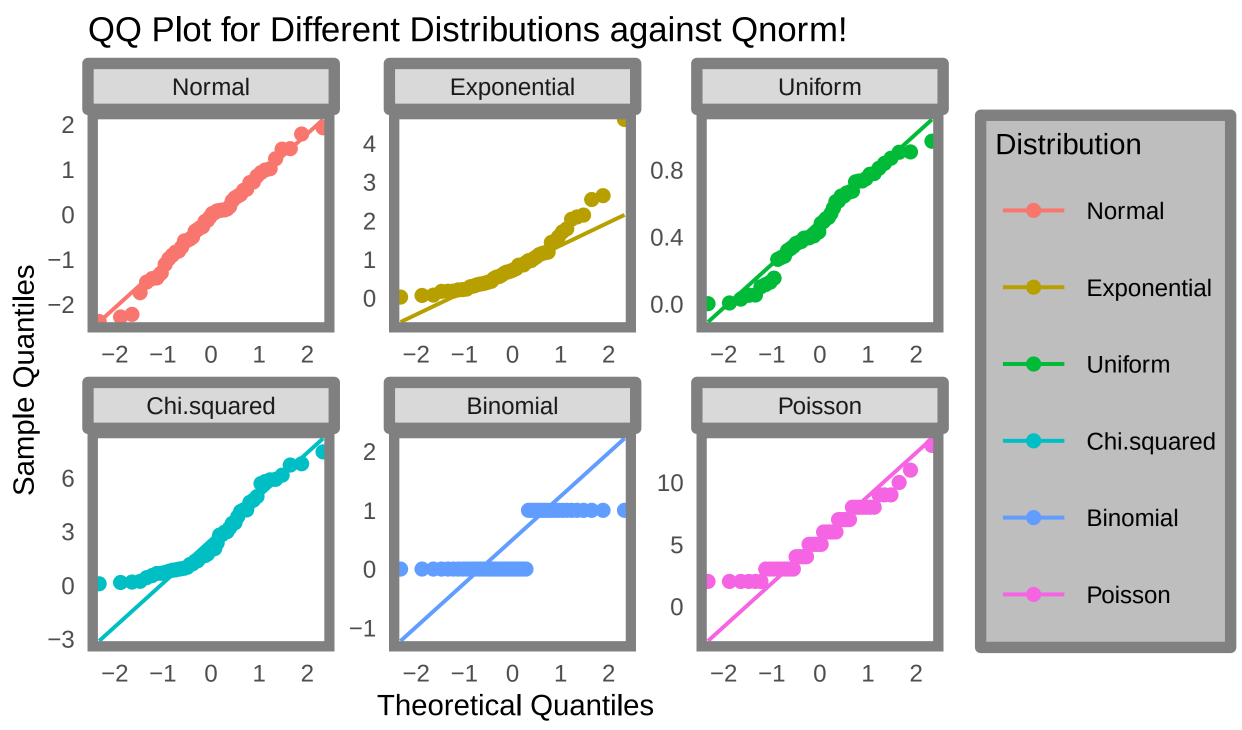


- Both the heights and the weights of females/males have approximately normal distribution

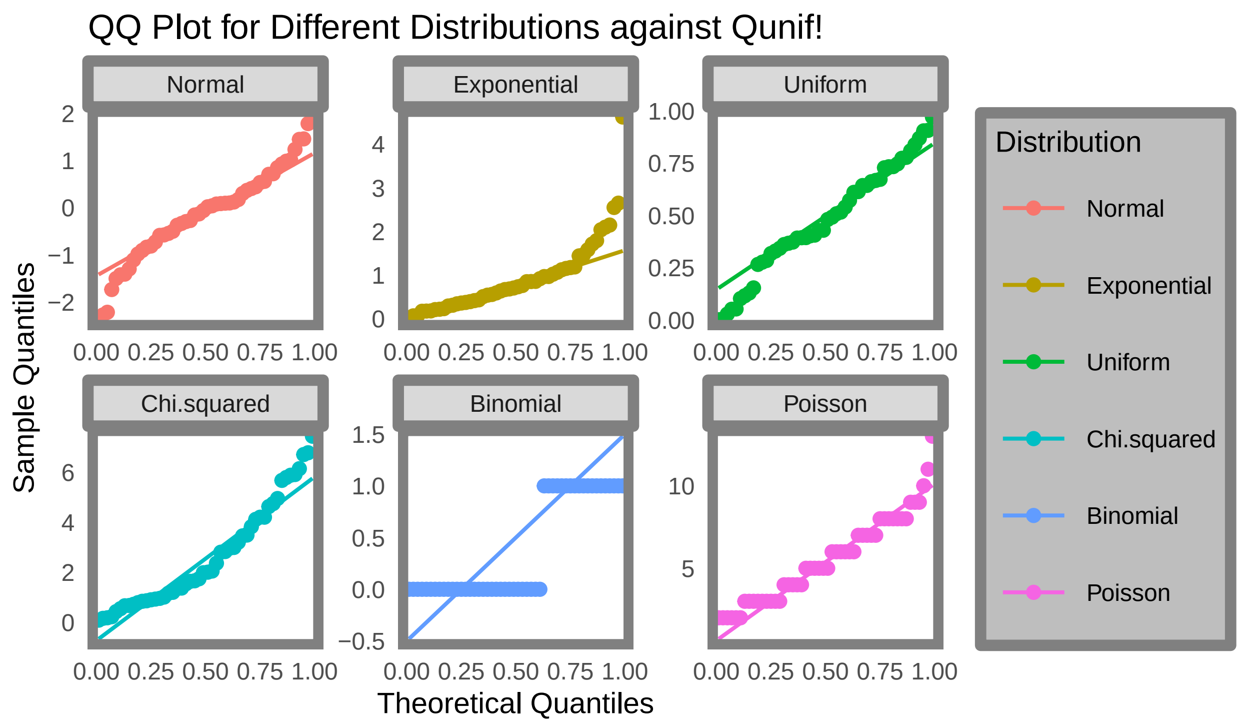
- Both the heighs and the weights of male is larger
- On average the difference between female and male heights is greater than the difference between their weigths
- The average of female heights remains almost constant throughout 2004-2016 and it is approximately 164
- The average of male heights remains almost constant throughout 2004-2016 and it is approximately 180
- The average of female weights remains almost constant throughout 2004-2016 and it is approximately 57
- The average of male weights remains almost constant throughout 2004-2016 and it is approximately 75
- The std of female and male heights is almost equal since the data is evenly spread for both genders
- The std of male weights is greater than the std of female weight since it is more spread about its mean
- Overall the pattern stays almost the same for years 2002-2016

Problem 3

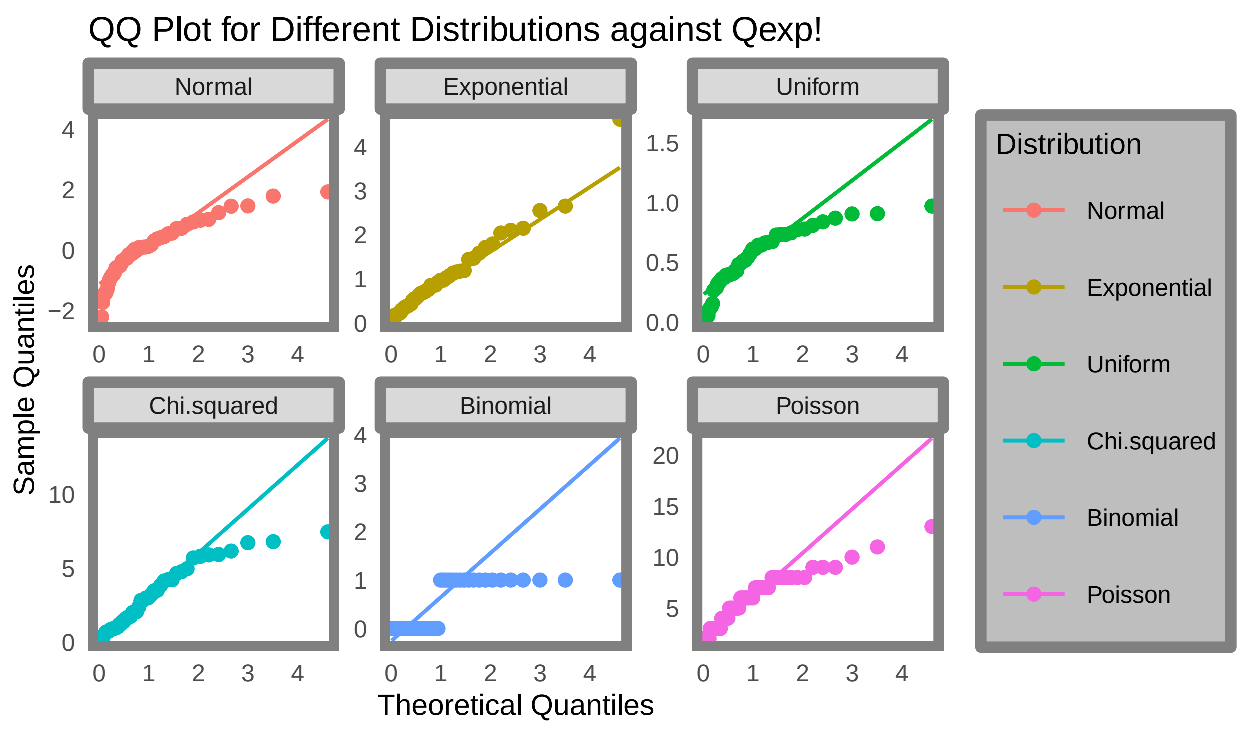
```r
# Generate random data from different distributions
n <- 50 # Sample size
normal_data <- rnorm(n)
exponential_data <- rexp(n, rate = 1)
uniform_data <- runif(n)
chi_squared_data <- rchisq(n, df = 3)
binomial_data <- rbinom(n, size = 1, prob = 0.5) # Binomial distribution
poisson_data <- rpois(n, lambda = 5) # Poisson distribution


# creating df with generated sampels and their dist names
df <- data.frame(Distribution = c(rep(c('Normal', 'Exponential', 'Uniform',
                                        'Chi.squared', 'Binomial', 'Poisson'), each = n)),
                 values = c(normal_data, exponential_data, uniform_data,
                            chi_squared_data, binomial_data, poisson_data))

# factoring, to be used later
df$Distribution <- factor(df$Distribution,
                          levels = c('Normal', 'Exponential', 'Uniform', 'Chi.squared', 'Binomial', 'Poi


ggplot(df , aes(sample=values, color = Distribution)) + # sample quantiles of sample variable
  facet_wrap(~Distribution, scale = 'free') +  # facets with respect to Dist
  geom_qq(size=2) + geom_qq_line(linewidth=0.7) +
  labs(title = 'QQ Plot for Different Distributions against Qnorm!',
       y = 'Sample Quantiles', x = 'Theoretical Quantiles') +
  theme_bw() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        panel.border = element_rect(color = '#808080', linewidth = 3.5),
        axis.ticks = element_blank(),
        axis.line.x.top = element_line(),
        legend.background = element_rect(color = '#808080', linewidth = 2, fill = 'gray'),
        legend.key = element_rect(fill = 'gray'),
        legend.key.height = unit(2, 'lines'),
        legend.key.width = unit(2, 'lines'),
        legend.position = c(1.19, 0.4979), # replacement of legend
        strip.clip = "off", # clips to make the upper rect bold separatly
        strip.background = element_rect(color = '#808080', linewidth = 2),
        plot.margin = margin(1.4, 4, 0.7, 0.2, "cm")) # replacement of plot
```



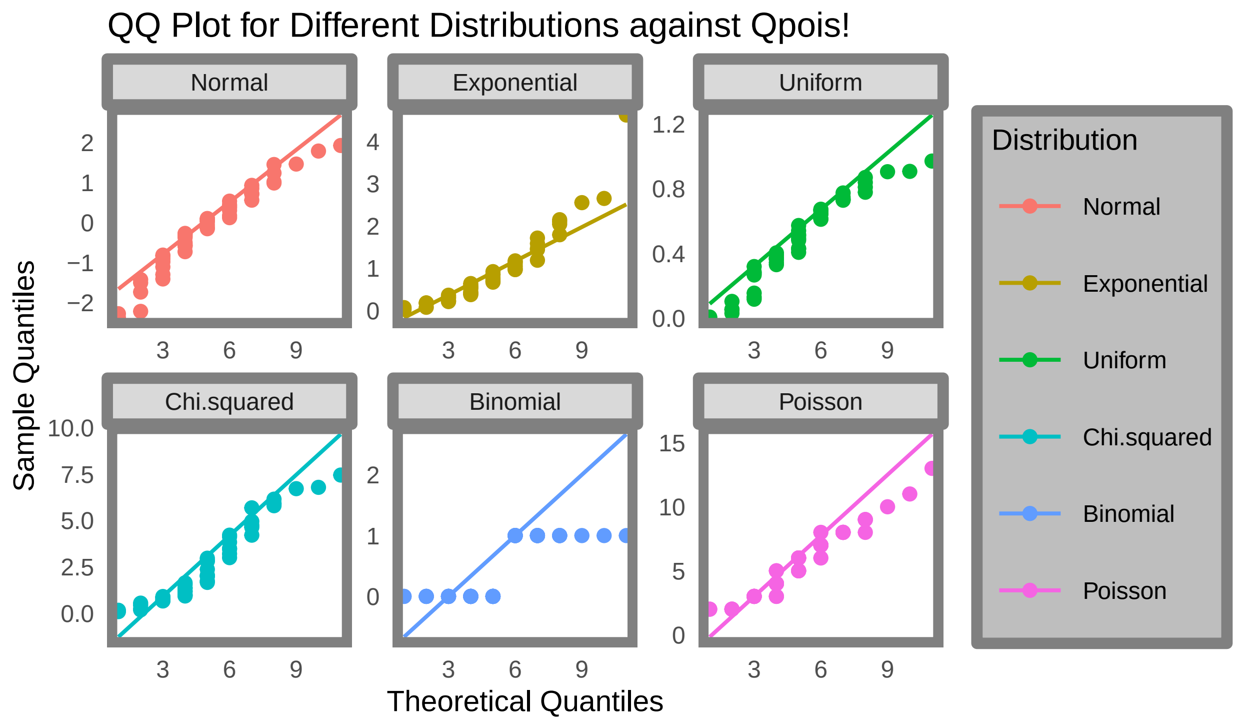
```r
# same as above but the theoretical quantiles will be that
# of unif dist, and th best fit line will have the intercept
# and slope constructed from unif's parameters
ggplot(df , aes(sample=values, color = Distribution)) +
  facet_wrap(~Distribution, scale = 'free') +
  geom_qq(distribution=stats::qunif, size=2) +
  geom_qq_line(distribution=stats::qunif, linewidth=0.7) +
  labs(title = 'QQ Plot for Different Distributions against Qunif!',
       y = 'Sample Quantiles', x = 'Theoretical Quantiles') +
  theme_bw() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        panel.border = element_rect(color = '#808080', linewidth = 3.5),
        axis.ticks = element_blank(),
        axis.line.x.top = element_line(),
        legend.background = element_rect(color = '#808080', linewidth = 2, fill = 'gray'),
        legend.key = element_rect(fill = 'gray'),
        legend.key.height = unit(2, 'lines'),
        legend.key.width = unit(2, 'lines'),
        legend.position = c(1.19, 0.4979),
        strip.clip = "off",
        strip.background = element_rect(color = '#808080', linewidth = 2),
        plot.margin = margin(1.4, 4, 0.7, 0.2, "cm"))
```



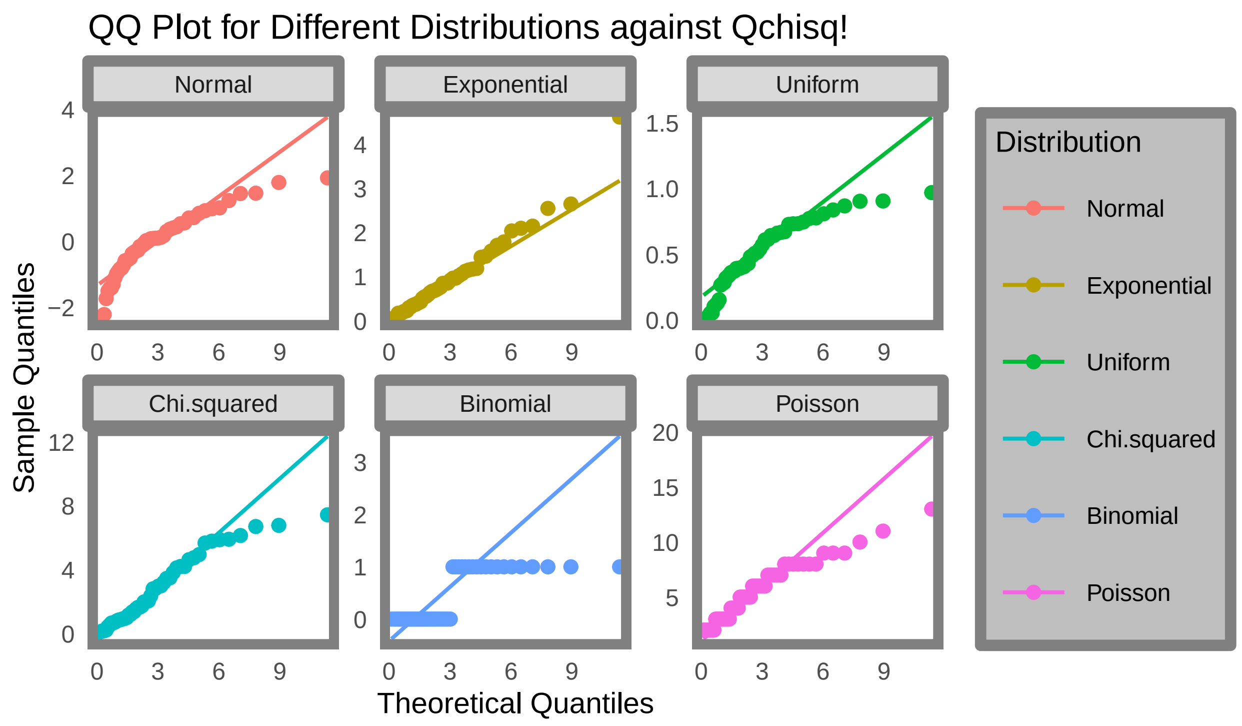
```r
# same for exponential
ggplot(df , aes(sample=values, color = Distribution)) +
  facet_wrap(~Distribution, scale = 'free') +
  geom_qq(distribution=stats::qexp, size=2) +
  geom_qq_line(distribution=stats::qexp, linewidth=0.7) +
  labs(title = 'QQ Plot for Different Distributions against Qexp!',
       y = 'Sample Quantiles', x = 'Theoretical Quantiles') +
  theme_bw() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        panel.border = element_rect(color = '#808080', linewidth = 3.5),
        axis.ticks = element_blank(),
        axis.line.x.top = element_line(),
        legend.background = element_rect(color = '#808080', linewidth = 2, fill = 'gray'),
        legend.key = element_rect(fill = 'gray'),
        legend.key.height = unit(2, 'lines'),
        legend.key.width = unit(2, 'lines'),
        legend.position = c(1.19, 0.4979),
        strip.clip = "off",
        strip.background = element_rect(color = '#808080', linewidth = 2),
        plot.margin = margin(1.4, 4, 0.7, 0.2, "cm"))
```



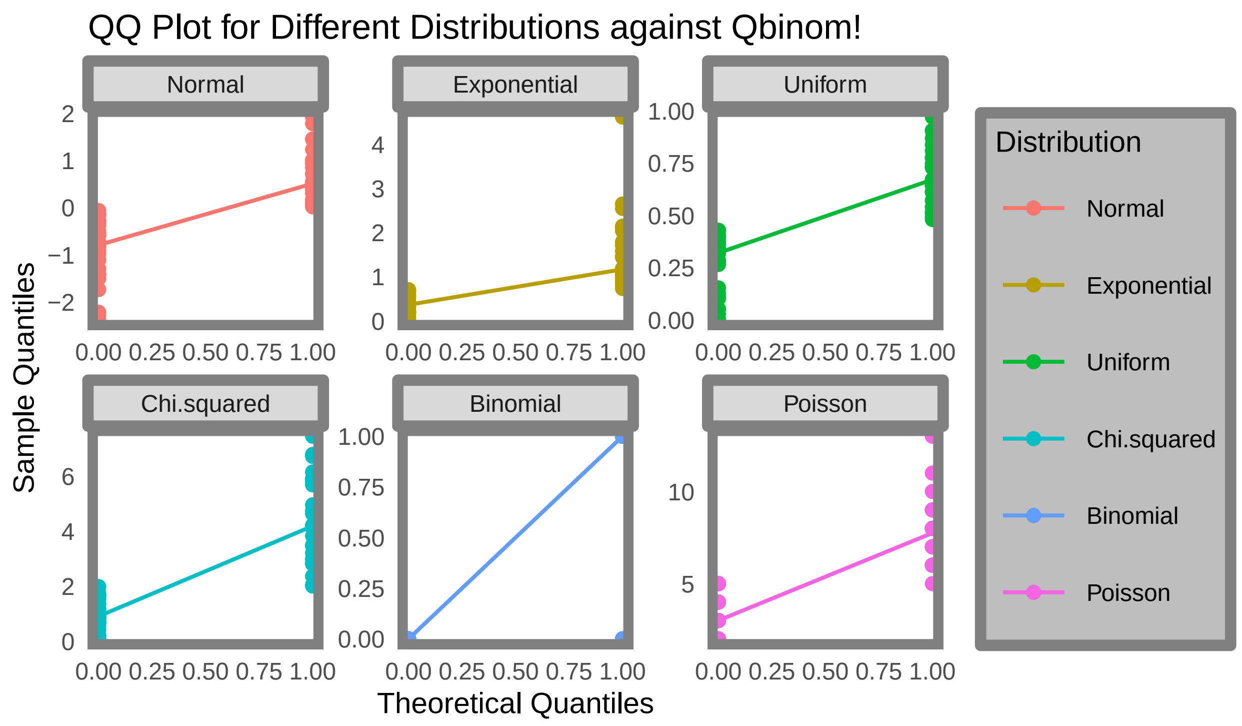
```r
# same for poisson
ggplot(df , aes(sample=values, color = Distribution)) +
  facet_wrap(~Distribution, scale = 'free') +
  geom_qq(distribution=stats::qpois, dparams = list(lambda = 5), size=2) +
  geom_qq_line(distribution=stats::qpois, dparams = list(lambda = 5), linewidth=0.7) +
  labs(title = 'QQ Plot for Different Distributions against Qpois!',
       y = 'Sample Quantiles', x = 'Theoretical Quantiles') +
  theme_bw() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        panel.border = element_rect(color = '#808080', linewidth = 3.5),
        axis.ticks = element_blank(),
        axis.line.x.top = element_line(),
        legend.background = element_rect(color = '#808080', linewidth = 2, fill = 'gray'),
        legend.key = element_rect(fill = 'gray'),
        legend.key.height = unit(2, 'lines'),
        legend.key.width = unit(2, 'lines'),
        legend.position = c(1.19, 0.4979),
        strip.clip = "off",
        strip.background = element_rect(color = '#808080', linewidth = 2),
        plot.margin = margin(1.4, 4, 0.7, 0.2, "cm"))
```



```
# same for chi squared
ggplot(df , aes(sample=values, color = Distribution)) +
  facet_wrap(~Distribution, scale = 'free') +
  geom_qq(distribution=stats::qchisq, dparams = list(df = 3), size=2) +
  geom_qq_line(distribution=stats::qchisq, dparams = list(df = 3), linewidth=0.7) +
  labs(title = 'QQ Plot for Different Distributions against Qchisq!',
       y = 'Sample Quantiles', x = 'Theoretical Quantiles') +
  theme_bw() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        panel.border = element_rect(color = '#808080', linewidth = 3.5),
        axis.ticks = element_blank(),
        axis.line.x.top = element_line(),
        legend.background = element_rect(color = '#808080', linewidth = 2, fill = 'gray'),
        legend.key = element_rect(fill = 'gray'),
        legend.key.height = unit(2, 'lines'),
        legend.key.width = unit(2, 'lines'),
        legend.position = c(1.19, 0.4979),
        strip.clip = "off",
        strip.background = element_rect(color = '#808080', linewidth = 2),
        plot.margin = margin(1.4, 4, 0.7, 0.2, "cm"))
```



```r
# same for binomial
ggplot(df , aes(sample=values, color = Distribution)) +
  facet_wrap(~Distribution, scale = 'free') +
  geom_qq(distribution=stats::qbinom, dparams = list(size = 1, prob = 0.5), size=2) +
  geom_qq_line(distribution=stats::qbinom, dparams = list(size = 1, prob = 0.5), linewidth=0.7) +
  labs(title = 'QQ Plot for Different Distributions against Qbinom!',
       y = 'Sample Quantiles', x = 'Theoretical Quantiles') +
  theme_bw() +
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        panel.border = element_rect(color = '#808080', linewidth = 3.5),
        axis.ticks = element_blank(),
        axis.line.x.top = element_line(),
        legend.background = element_rect(color = '#808080', linewidth = 2, fill = 'gray'),
        legend.key = element_rect(fill = 'gray'),
        legend.key.height = unit(2, 'lines'),
        legend.key.width = unit(2, 'lines'),
        legend.position = c(1.19, 0.4979),
        strip.clip = "off",
        strip.background = element_rect(color = '#808080', linewidth = 2),
        plot.margin = margin(1.4, 4, 0.7, 0.2, "cm"))
```



- We see that the qqline fits the qq_plot data best when the line is constructed by the parameters of the dist, agains which sample quantiles are ploted.

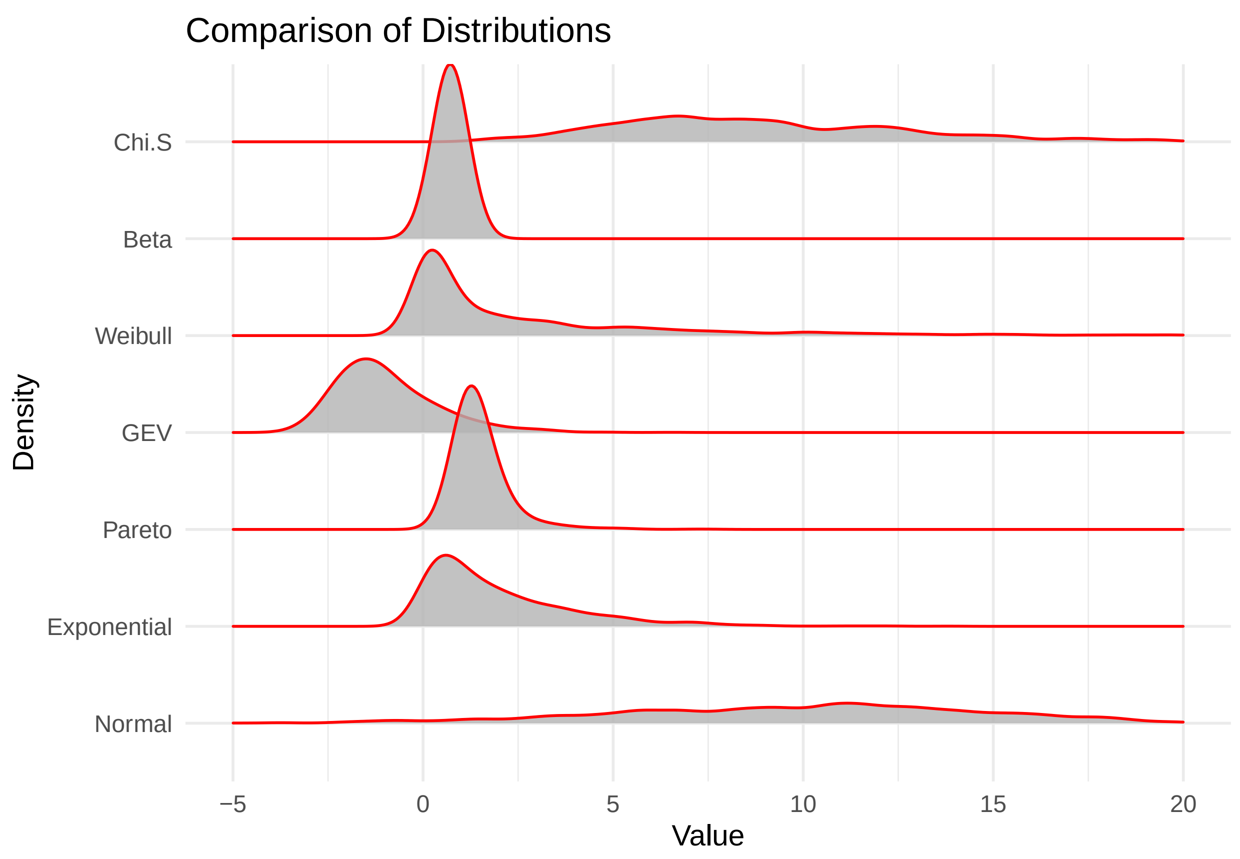
- default geom_qq and geom_qq_line work with normal dist
- when specifying the line params, the mean becomes intercept and the std slope
- in case of normal it is bisector
- so we plot points with coordinates (theoretical quantile, sample quantile), if the sample comes from the distribution of theoretical quantiles we will have (a,b) coordinates where a = b
- the qq_line is constructed by connecting the first and last points with above mentioned coords
- Hence for each case we have one plot that obviously fits the line

Problem 4

4.1

```
# num of data points in each sample
sample_size <- 1000

# generate random samples from different distributions
# and store in df
```

```r
df <- data.frame(
  Normal = rnorm(sample_size, 10, 5),
  Exponential = rexp(sample_size,0.5),
  Pareto = rpareto(sample_size, shape=3),
  GEV = rgev(sample_size, -1.5),
  Weibull = rweibull(sample_size, shape = 0.5, scale = 2),
  Beta = rbeta(sample_size, 5, 2),
  Chi.S = rchisq(sample_size, df = 9)
  )






# reshape the data, give colnames for easier plotting
df_long <- stack(df)
colnames(df_long) <- c("Value", "Distribution")



# plot density for each above generated samples
ggplot(df_long, aes(x = Value, y=Distribution )) + # ridges with respect to Dist
  geom_density_ridges(alpha = 0.8, color='red') +
  xlim(c(-5,20)) +
  labs(x = "Value", y = "Density", title = "Comparison of Distributions") + # give labs
  theme_minimal()
```



- Exponential and Pareto are usually heavy tailed, which can be a good starting reference point for further visual analysis.

- Just visually we can see that Exponential, Pareto, Gev, Weibull have heavy tails especially compared to normal which has less density so it is most likely thin tailed. Then comparing Normal and Chi.S we see that again as we approach to tails of the density plot, Chi.S has higher density than normal, so even though at first sight it seems thin tailed similar to normal, it is most likely heavy tailed (at least compared to visualized normal).
- As we approach to the tails of Exponential, Pareto, Gev, Weibull we see that they have more data in tails with higher density value. This pattern also suggests that most probably they are heavy tailed
- The next visualized dist. that doesn't have obvious heavy/light tails is Beta. Which is somewhat similar to that of Pareto. But observing Pareto's and Beta's right tails we see more or less ligher tails for Beta, since it has less density values near the tails than Pareto.
- Since visual inferences cannot be very reliable, the claims are further justified with the values of respective kurtosis below.

4.2

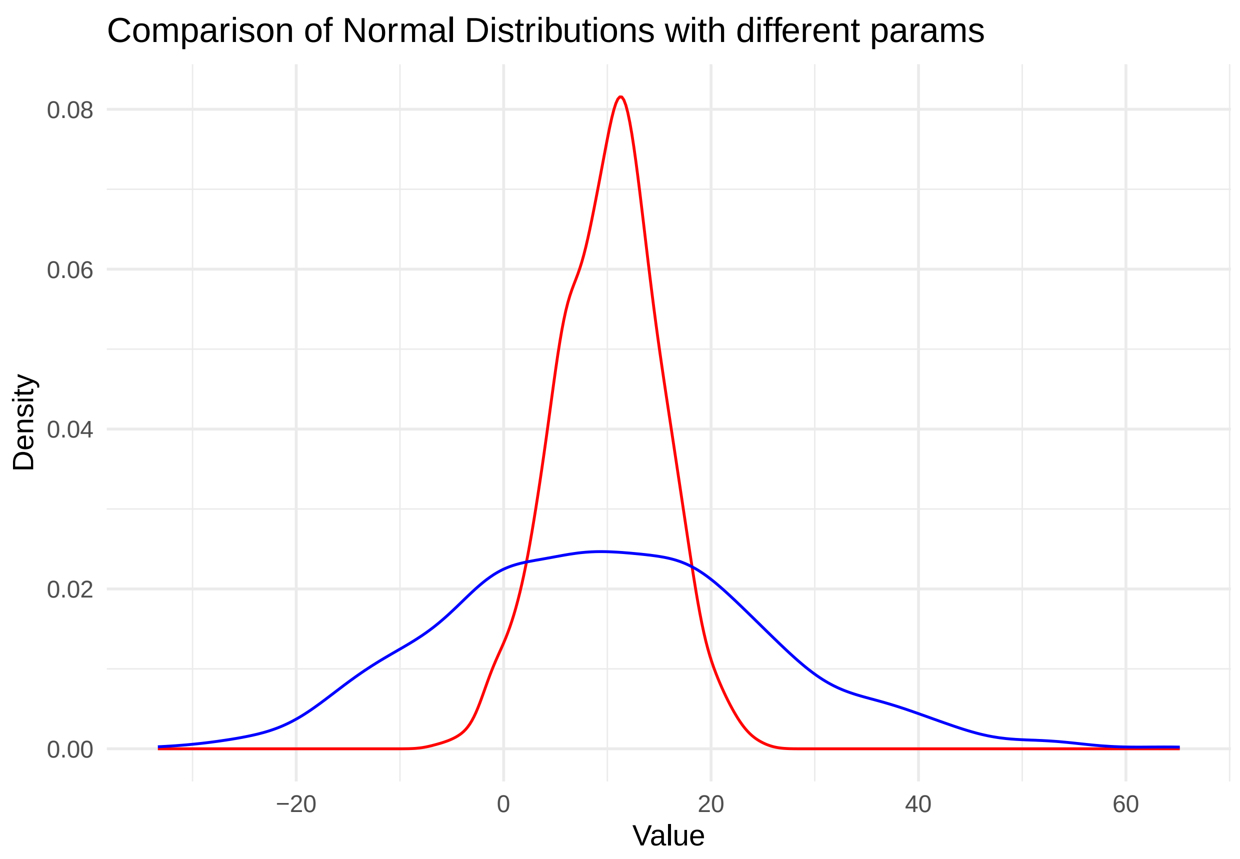
```r
# calculate mean, variance, kurtosis, and skewness for each dist
stat_measures <- df_long %>%
  group_by(Distribution) %>%
  summarize(
    Mean = mean(Value),
    Variance = var(Value),
    Kurtosis = kurtosis(Value),
    Skewness = skewness(Value)
  )


# stat analysis for skewness
for (dist in stat_measures$Distribution) {
  if (stat_measures[stat_measures$Distribution==dist, 'Skewness'] < 0){
    print(glue('Visualized {dist} is left-skewed'))
  } else {
    print(glue('Visualized {dist} is right-skewed'))
  }
}
```

```
## Visualized Normal is left-skewed
## Visualized Exponential is right-skewed
## Visualized Pareto is right-skewed
## Visualized GEV is right-skewed
## Visualized Weibull is right-skewed
## Visualized Beta is left-skewed
## Visualized Chi.S is right-skewed
```

```
# for kurtosis
for (dist in stat_measures$Distribution) {
  if (stat_measures[stat_measures$Distribution==dist, 'Kurtosis'] < 3){
    print(glue('Visualized {dist} has light tails'))
  } else {
    print(glue('Visualized {dist} has heavy tails'))
  }
}
```

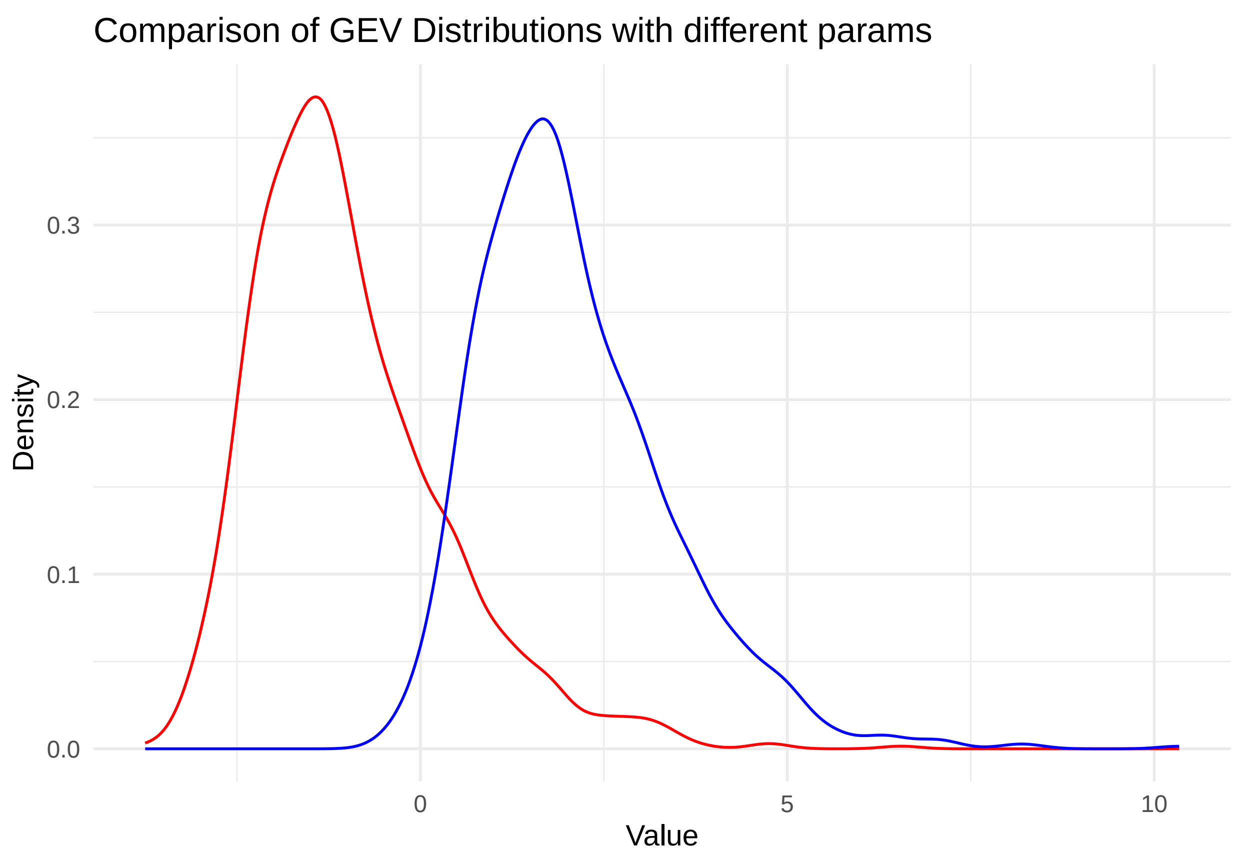
```
## Visualized Normal has light tails
## Visualized Exponential has heavy tails
## Visualized Pareto has heavy tails
## Visualized GEV has heavy tails
## Visualized Weibull has heavy tails
## Visualized Beta has light tails
## Visualized Chi.S has heavy tails
```

```
# comparing questionable distributions with same dist, different params
# taking reference dist
normal_ref = rnorm(sample_size,10,15)

# plotting both original and reference to make inferences
ggplot() +
  geom_density(aes(df_long[df_long$Distribution == "Normal", "Value"]),alpha = 0.8, color='red') +
  geom_density(aes(normal_ref),alpha = 0.8, color='blue') + # 2 densities at once
  labs(x = "Value", y = "Density", title = "Comparison of Normal Distributions with different params")
  theme_minimal()
```



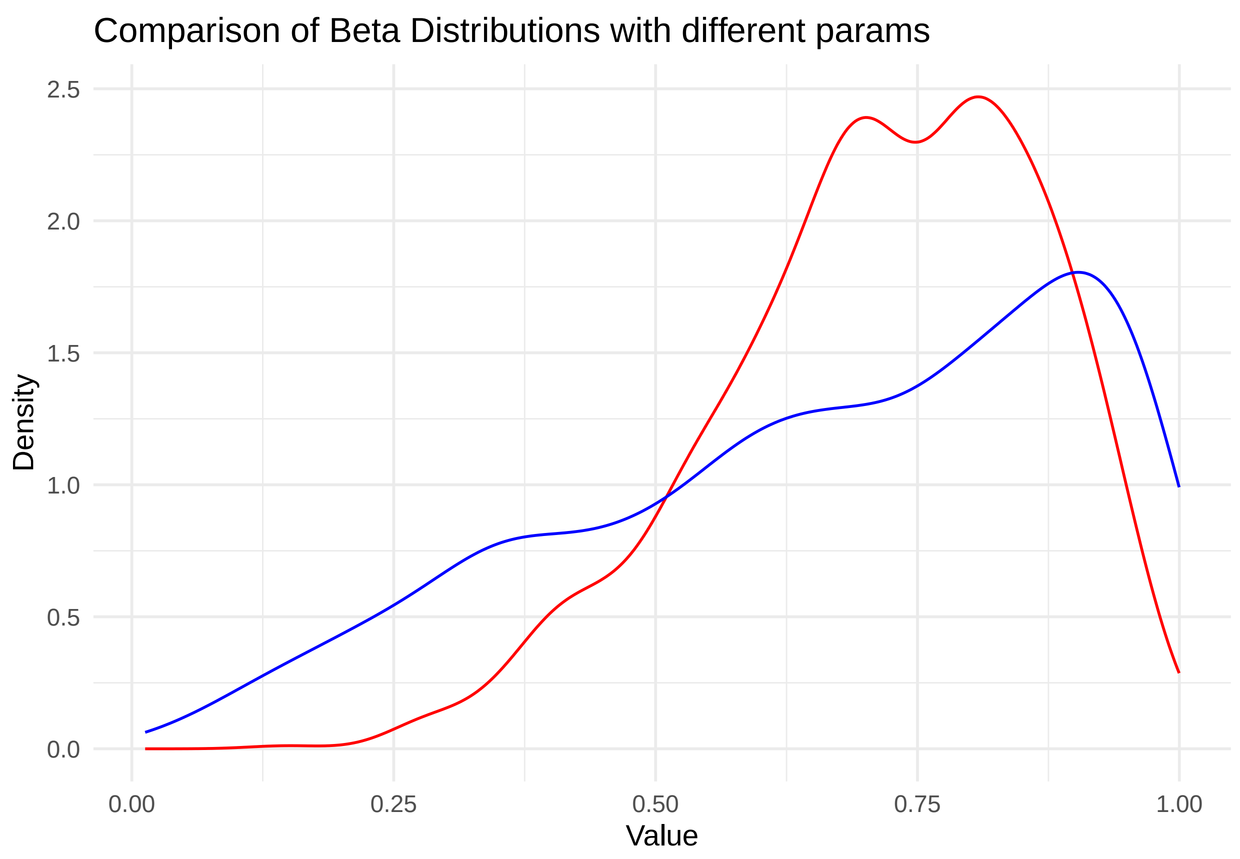
As we approach to the tails the reference(blue) normal gets higher density values than original(red). So the original was light tailed

```r
# taking reference dist
g_ref = rgev(sample_size, 1.5)

# plotting both original and reference to make inferences
ggplot() +
  geom_density(aes(df_long[df_long$Distribution == "GEV", "Value"]),alpha = 0.8, color='red') +
  geom_density(aes(g_ref),alpha = 0.8, color='blue') + # 2 densities at once
  labs(x = "Value", y = "Density", title = "Comparison of GEV Distributions with different params") +
  theme_minimal()
```



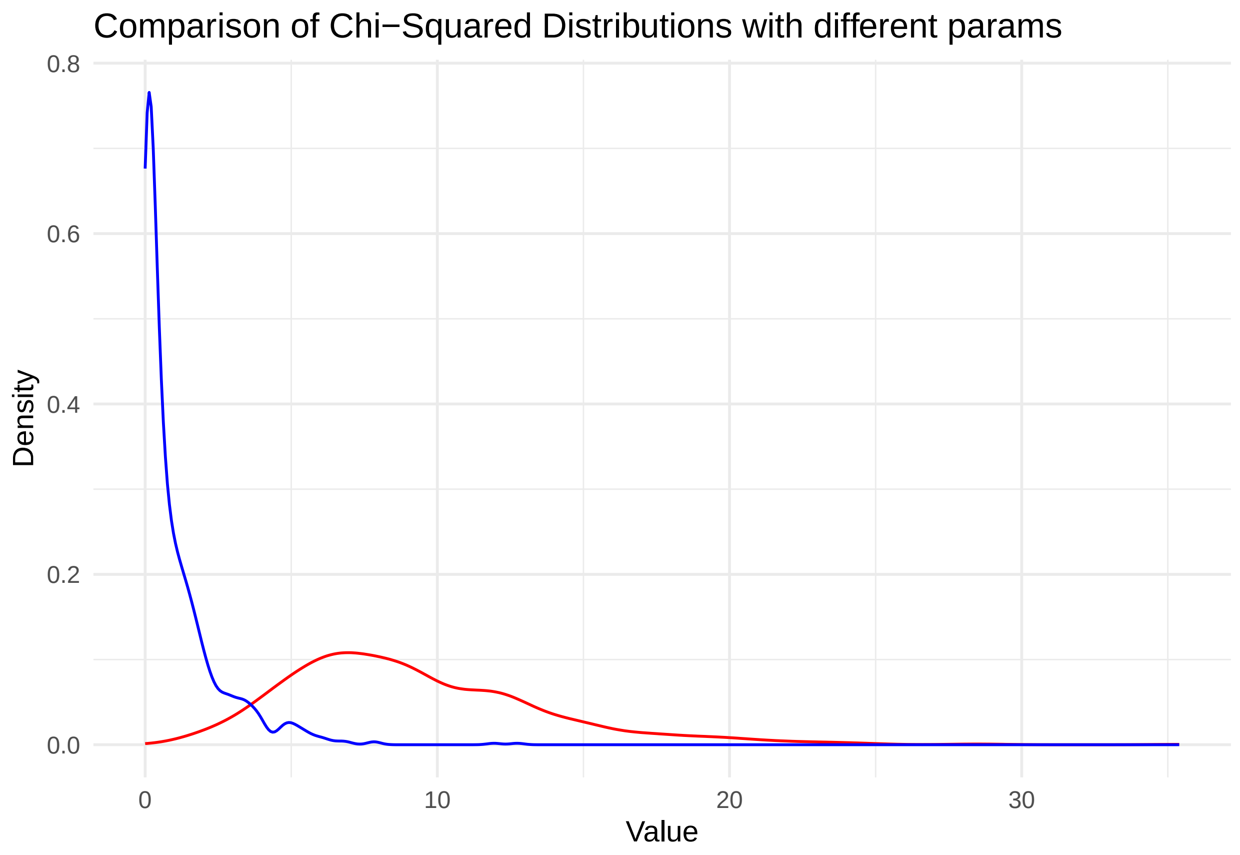
We conclude that GEV was heavy tailed in original since it has greater density near tails

```r
# taking reference dist
b_ref = rbeta(sample_size, 2, 1)

# comparing with original visually
ggplot() +
  geom_density(aes(df_long[df_long$Distribution == "Beta", "Value"]),alpha = 0.8, color='red') +
  geom_density(aes(b_ref),alpha = 0.8, color='blue') +
  labs(x = "Value", y = "Density", title = "Comparison of Beta Distributions with different params") +
  theme_minimal()
```



As we approach to the tails the red line(original) gets lower than the referenced Betra dist. So it supports to the claim that in original Beta was light tailed

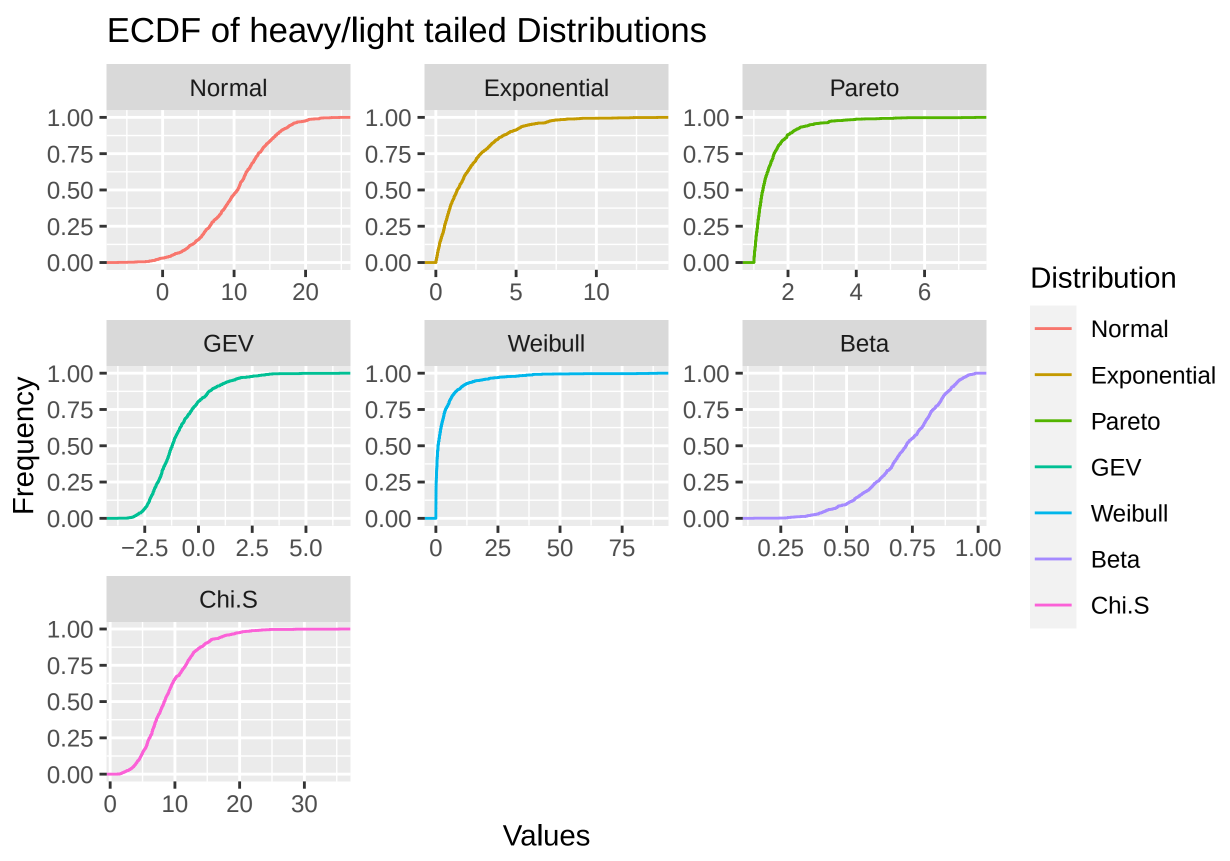
```r
# taking reference dist
chi_ref = rchisq(sample_size, df = 1)

# comparing with original visually
ggplot() +
  geom_density(aes(df_long[df_long$Distribution == "Chi.S", "Value"]),alpha = 0.8, color='red') +
  geom_density(aes(chi_ref),alpha = 0.8, color='blue') +
  labs(x = "Value", y = "Density", title = "Comparison of Chi-Squared Distributions with different para
  theme_minimal()
```



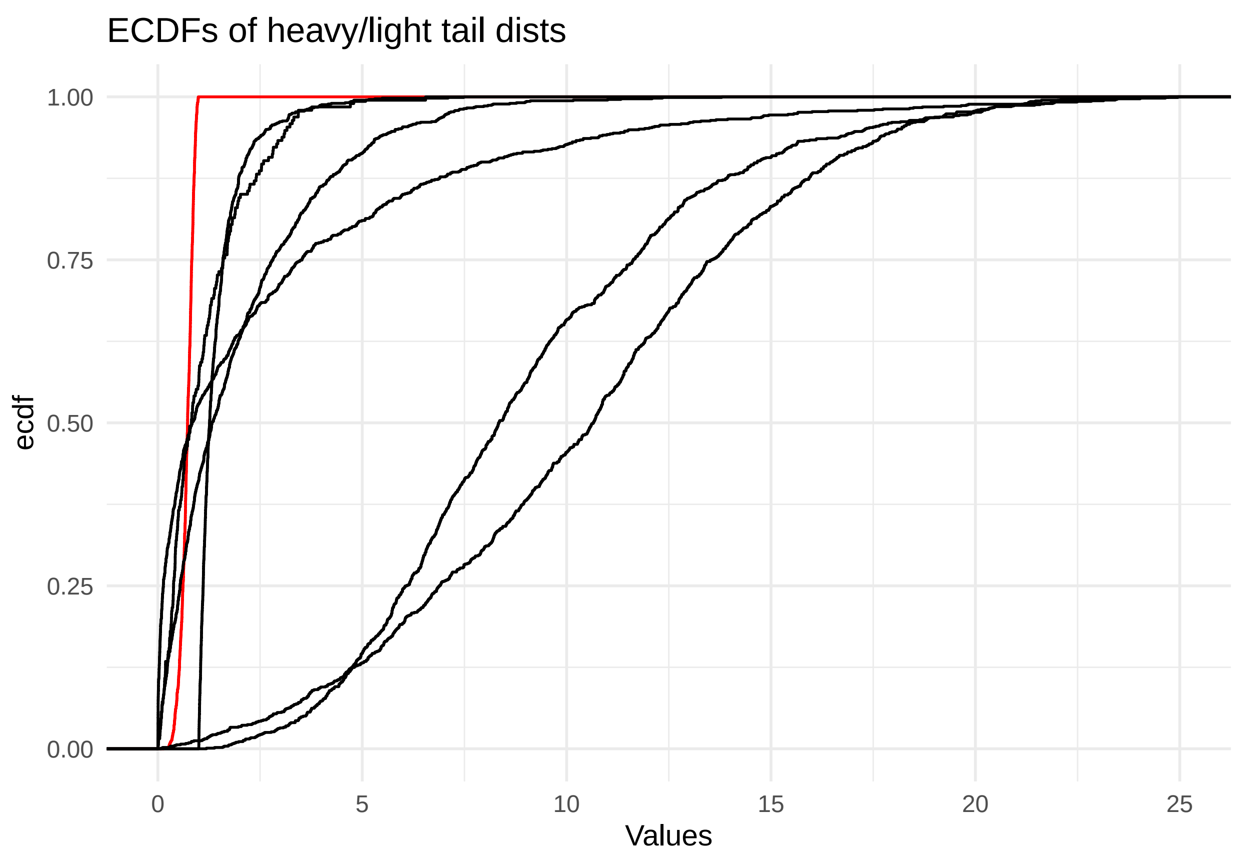
As we see as both density lines appraoch to their tails, blue(referenced) plot gets little density values , which means that in original Chi.S distribution had heavy tails.

4.3

```r
#ploting ecdfs with different facets
ggplot(df_long, aes(x = Value, color = Distribution)) +
  facet_wrap(~Distribution, scale='free') +
  stat_ecdf() +
  labs(x = "Values", y = "Frequency", title = "ECDF of heavy/light tailed Distributions") +
  theme_gray()
```



```r
# plotting all in one
ggplot() +
  stat_ecdf(aes(df_long[df_long$Distribution == "Beta", "Value"]), color = 'red', linedith = 5) +
  stat_ecdf(aes(df_long[df_long$Distribution == "Normal", "Value"])) +
  stat_ecdf(aes(df_long[df_long$Distribution == "Exponential", "Value"])) +
  stat_ecdf(aes(df_long[df_long$Distribution == "GEV", "Value"])) +
  stat_ecdf(aes(df_long[df_long$Distribution == "Weibull", "Value"])) +
  stat_ecdf(aes(df_long[df_long$Distribution == "Pareto", "Value"])) +
  stat_ecdf(aes(df_long[df_long$Distribution == "Chi.S", "Value"])) +
  xlim(c(0,25)) +
  labs(title = 'ECDFs of heavy/light tail dists', x = 'Values') +
  theme_minimal()
```

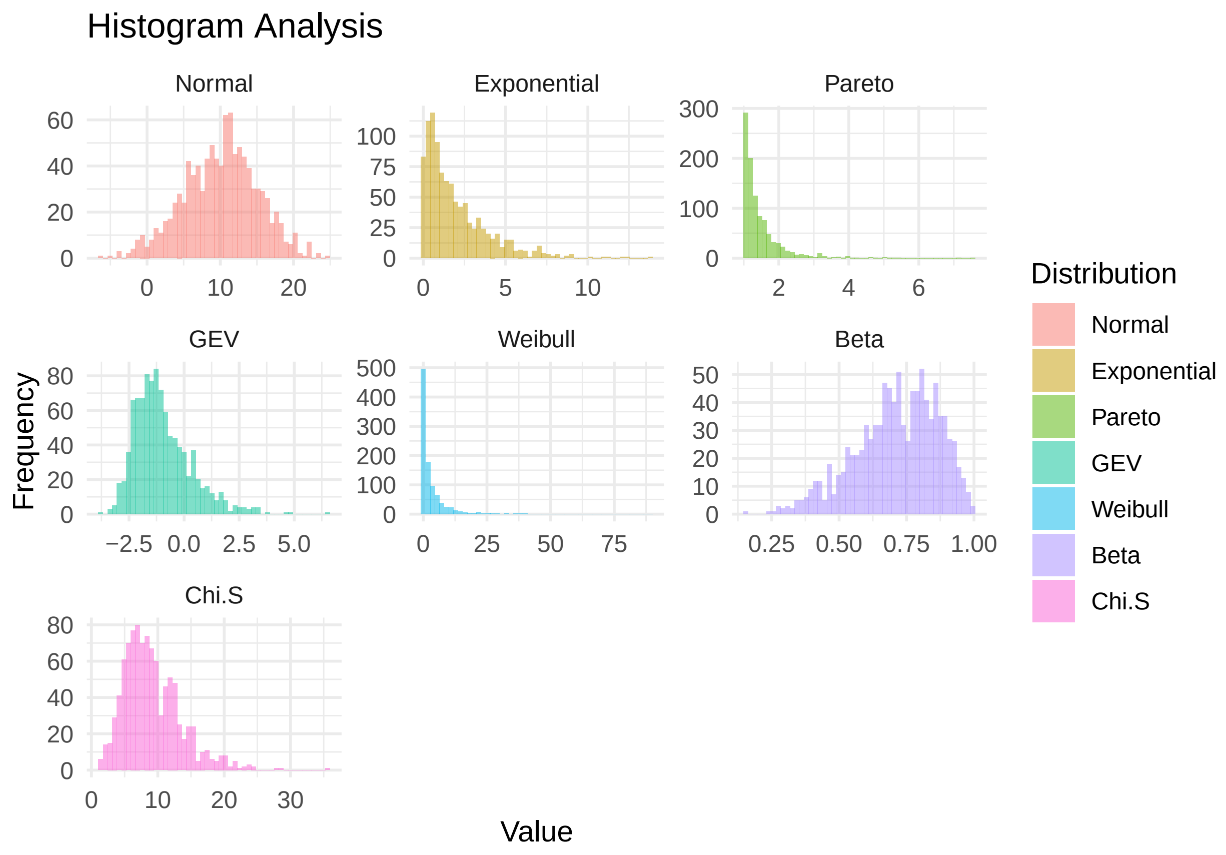


- To sum up For light tailed distributions, the ecdf typically increases quickly and approaches 1 gradually as x increases. In the plot, there's a steep rise in the beginning, indicating that most observations are clustered near the lower values.(especially for Beta) The ecdf tends to reach high values relatively quickly for increasing values of x, indicating that extreme values occur less frequently.

- For heavy tailed distributions, the ecdf increases slowly initially and then rise rapidly, indicating that extreme values are more frequent than expected in a light tailed distribution. (more visible in faceted plot) There is gradual increase at first, followed by a sharp increase, indicating the presence of outliers or extreme values.

4.4

```r
# plotting histograms for each dist with separate facet
ggplot(df_long, aes(x = Value, fill = Distribution)) +
  facet_wrap(~Distribution, scale='free') +
  geom_histogram(position = "identity", alpha = 0.5, bins = 50) +
  labs(x = "Value", y = "Frequency", title = "Histogram Analysis") +
  theme_minimal()
```

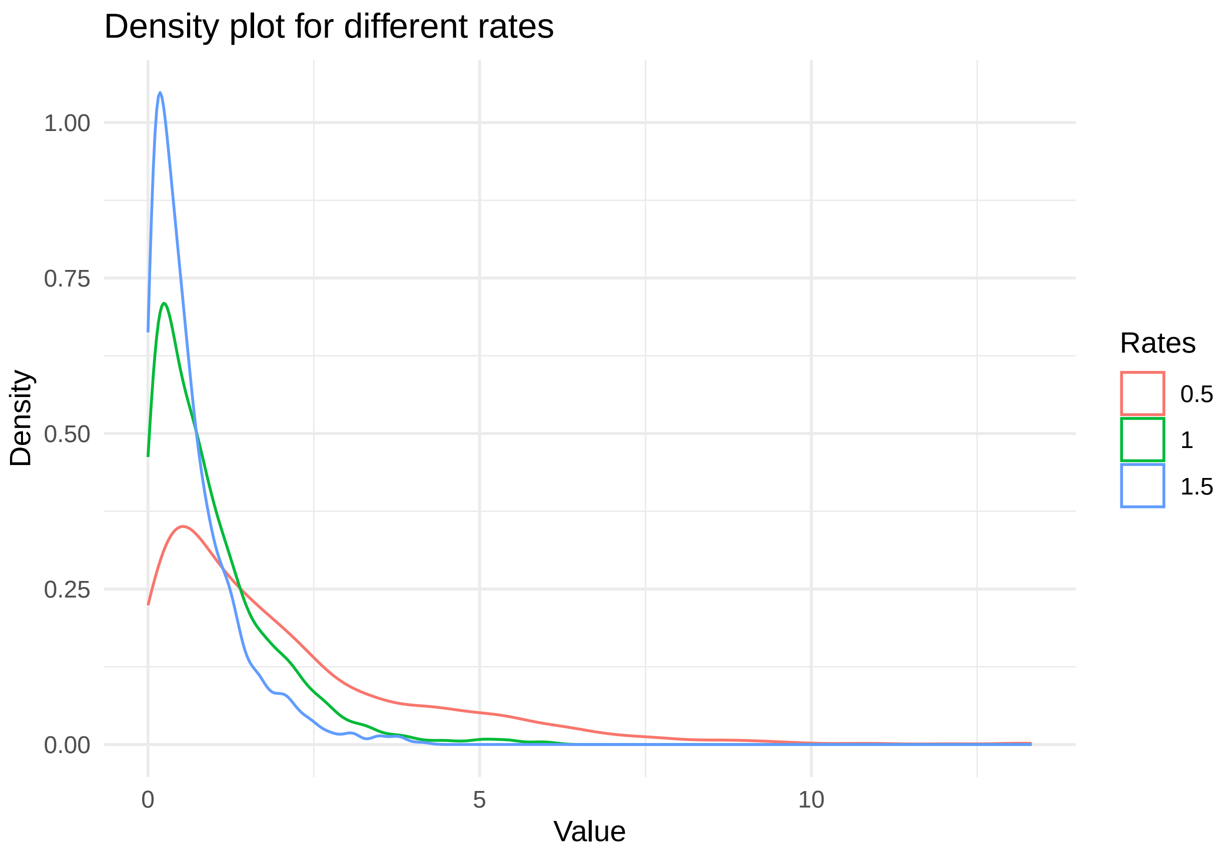


- The histogram plot is further support for the fact that depicted Beta and Normal dist-s are thin tailed and all others are heavy tailed.

- There is slower decay in frequencies towards the tails(for heavy tailed). This slow decrease indicates that extreme values are more probable than in light tailed.
- There is rapid decrease in frequencies towards the tails. This rapid decrease indicates that extreme values are less probable.
- All the depicted heavy tailed dist-s have longer spread tails
- The tails light tailed dist-s are relatively short and do not extend far from the central part of the data.

4.5

```r
# generating data for exponential distributions with different rate parameters
rate_values <- c(0.5, 1, 1.5)  # Different rate parameters
sample_size <- 1000
data_list <- lapply(rate_values, function(rate) {
  rexp(sample_size, rate = rate)
})

# combining the data into df
combined_data <- data.frame(
  value = unlist(data_list),
  rate = rep(rate_values, each = sample_size)
)

# creating a density plot with different colors for each rate
ggplot(combined_data, aes(x = value, color = as.factor(rate))) +
  geom_density(alpha = 0.2, linewidth = 0.5) +
  labs(title = "Density plot for different rates",
       x = "Value",
       y = "Density", color='Rates') +
  theme_minimal()
```



- As the rate increases, the distribution becomes more concentrated around lower values, leading to a higher peak in the density plot. In general, exponential distribution: higher rate values => faster decay of the distribution, and thus the large part of the data is concentrated near the origin. (This is because in pdf we have e^-rate…)

- lower rate => slower decrease in probability density funciton, resulting in a thick tailed behavior as already discussed above. Conversely, higher rate => faster decrease in pdf, resulting in a thin tailed dist.
- taking rate = 1 as reference, rate = 1.5 is light tailed, rate = 0.5 heavy tailed

4.6

Heavy tailed dists: Display wider spreads with longer and thicker tails. Outliers and extreme values are more likely Light tailed dists: Show narrower spreads with shorter and thinner tails. Extreme values are less likely to occur, and data is concentrated around the mean.

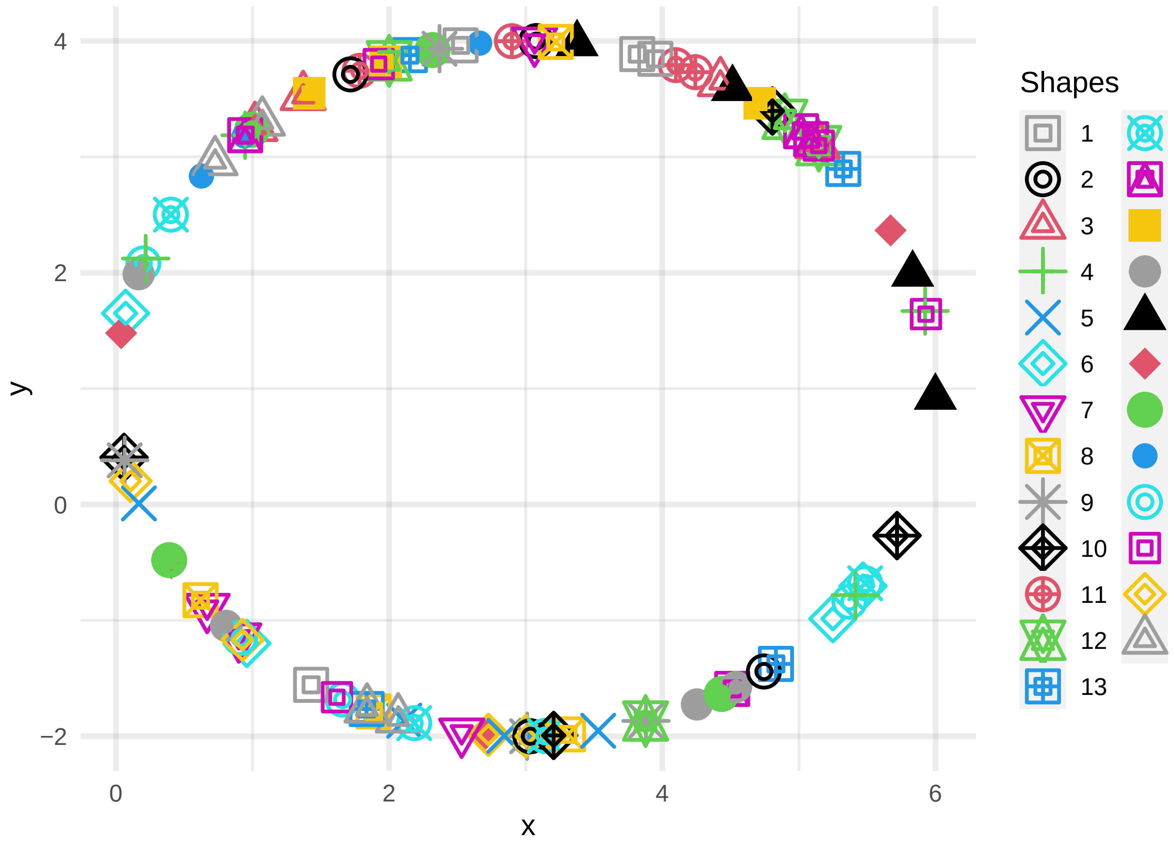
Measures like kurtosis are higher (>3) in heavy- tailed dists reflecting the presence of outliers and the dist's heavy tailed nature, while light tailed dists have lower kurtosis(<3), indicating a thinner tail and less presence of outliers.

Heavy tailed dists have tails that decline more slowly resulting in a higher frequency of extreme values, while light tailed dists have tails that decline more rapidly making extreme values less likely.

In light tailed dists the ecdf shows a relatively rapid increase particularly in the tails. while in heavy tailed dists, the ecdf often has a slow and steady increase particularly in the tails.

Problem 5

```r
# Having the formula of the circle, I get Y, generate random numbers
# for X, plug in to get Y

# Generate data points for the upper half of the circle
x_upper <- runif(50, 0, 6)
y_upper <- sqrt(9 - ((x_upper - 3) ^ 2)) + 1

# Generate data points for the lower half of the circle
x_lower <- runif(50, 0, 6)
y_lower <- -sqrt(9 - ((x_lower - 3) ^ 2)) + 1

# Combine the upper and lower halves into one vector to form a full circle
x <- c(x_upper, x_lower)
y <- c(y_upper, y_lower)


# Generating dataframe, which has x,y columns and shape column/feature
# Each point in df with (x,y) coordinates will correspond to one of 25 classes
# I classify the points randomly so I use sample function which shuffles
# points from 0 to 24 (25 distinct classes) and assigns one for the point

circle_data <- data.frame(x, y, shape = sample(0:24))

# Categorizing points using factor() to use shape column for color and shape parameters later
# Since the 1 shape cooresponds to circle, but i need it to be square, I dont do 1:25
# but rather 0:24, and then change their names inside legend

circle_data$shape <- sample(factor(circle_data$shape))


p1 <- ggplot(circle_data, aes(x = x, y = y, color = shape, shape = shape)) + # mapping color and shape
  geom_point( size = 5, stroke = 1) +
  geom_point( size = 2, stroke = 1) +  # Second point with smaller size for double lined elements
  scale_shape_manual(values = c(0:24), name='Shapes', labels = c(as.numeric(1:25))) +
  # changing the names of shapes to be 1-25
  scale_color_manual(values=8:32,  name='Shapes', labels = c(as.numeric(1:25))) +
  # if we didn't specify the same name, we would end up with separate legends for color and shape
  theme(panel.border = element_blank(), axis.ticks = element_blank(),
        panel.background = element_blank(), legend.background = element_blank(),
      panel.grid = element_line(color  = rgb(0.5, 0.5, 0.5, alpha = 0.15), size = 1))
  # removing the all spines, axis ticks, background, adding grids with right color and
  # transparency, removing the background of legend


p1
```

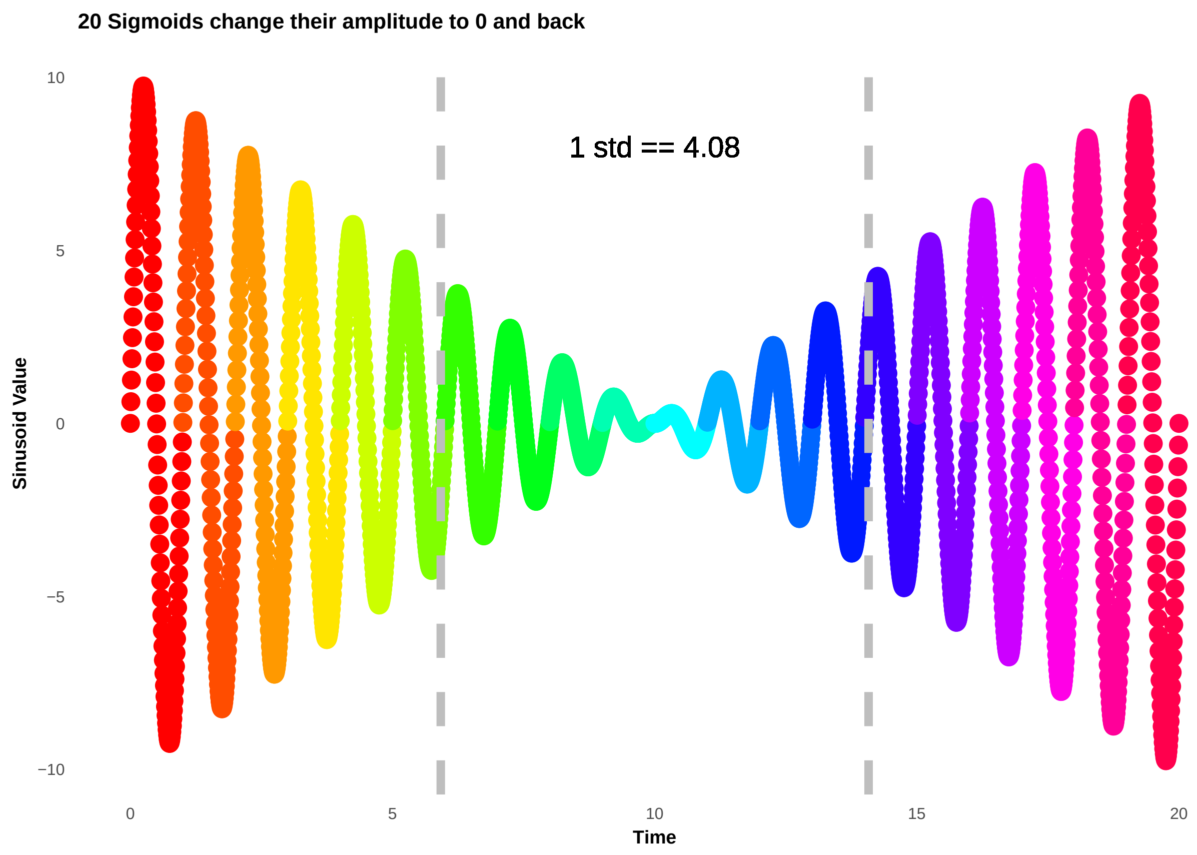


Problem 6

Our sigmoid starts from (0,0), it has the shape of sin so it is sinusoid Sinusoidal function: $y = A\sin(Bx)$ where A is amplitude (max height of sinusoid) $B = \frac{2\pi}{\text{period}}$ period $= 20/20 = 1$ (we have 20 sigmoids from range 0-20) Final formula : $y = A_{\text{for each sin}} \cdot \sin(2\pi x)$

```
# evenly spaced 2000 pnts
x <- seq(0, 20, length.out = 2000)

# amplitudes that decrease on 10-0 and increase on 0-10,
# combinind in one vector and calculating values of y
amplitude_values <- c(seq(10, 0, length.out = 1000), seq(0, 10, length.out = 1000))
y <-  amplitude_values * sin(2*pi*x)

# creating df of my pnts
sin_df <- data.frame(x,y)
```

```r
# dividing my x values into 20 parts, where the color changes will occur
color_variable <- cut(sin_df$x, breaks = 20)
custom_colors <- rainbow(20) # taking 20 colors from rainbow palette

std <- 4.08

ggplot(sin_df, aes(x,y, color = factor(color_variable))) +
  geom_point(size = 3, shape = 16) +
  # mapping x, y, and color as a factor of color_variable

  # giving bold title to the graph, x, y axis
  labs(title = expression(bold('20 Sigmoids change their amplitude to 0 and back')),
       x = expression(bold('Time')),
       y = expression(bold('Sinusoid Value')))+

  # since std = 10 and our mean value is 10, the girds will be x = 10 +- std
  geom_vline(xintercept = 10 + std, color = "grey",
             linetype = "dashed", linewidth = 1.5) +
  geom_vline(xintercept = 10 - std, color = "grey",
             linetype = "dashed", linewidth = 1.5) +

  # adding another geom object which is text
  geom_text(x = 10, y = 8, fontface = "plain",
            label = '1 std == 4.08',
            size = 4, color = 'black') +

  # this part says use the breaks of color_variable
  # and the colors of custom_colors variable
  scale_color_manual(values = setNames(custom_colors,
                                       levels(color_variable)),
                     breaks = levels(color_variable)) +

  # again removing bckg, grids, ticks, legend, adjusting text size
  theme(panel.background = element_blank(),
        panel.grid = element_blank(),
        axis.ticks = element_blank(),
        legend.position = 'none',
        text = element_text(size = 7))
```



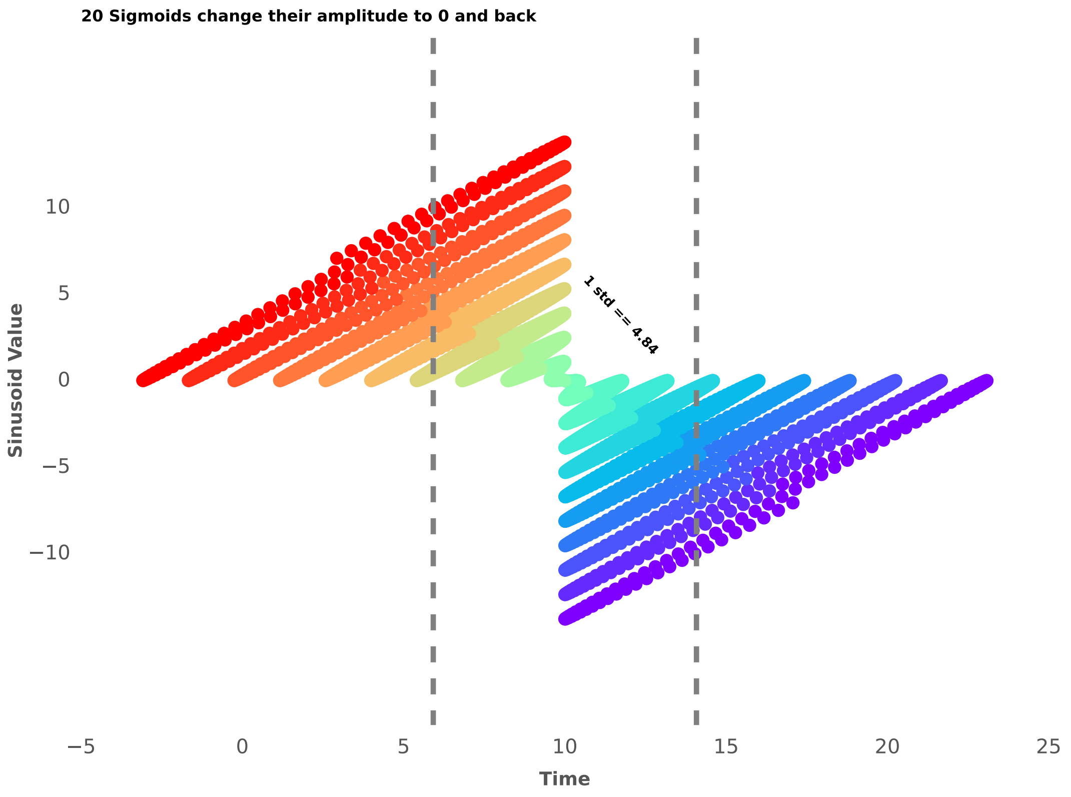
Problem 7

```
plt.style.use('ggplot')
x = np.linspace(0, 20, 2000)
amplitude_values = np.array([np.linspace(10, 0, 1000), np.linspace(0, 10, 1000)]).flatten()
y = amplitude_values * np.sin(2*np.pi*x)
std = 4.08


# From here, the problem is rotating our [x,y] vector by -45-180 radians
# negative, since in the picture it is moved anti clockwise
# used https://www.youtube.com/watch?v=EZufiIwwqFA for this part


beta = np.radians(-45-180)
```

$$\begin{bmatrix} x_{\text{rotated}} \\ y_{\text{rotated}} \end{bmatrix} = \begin{bmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

where $(x_{\text{rotated}}, y_{\text{rotated}})$ are the coordinates after the rotation, (x, y) are the original coordinates, and β is the angle of rotat

```python
# Rotation matrix
rotation_matrix = np.array([[np.cos(beta), -np.sin(beta)],
                            [np.sin(beta), np.cos(beta)]])

# since we rotate about (0,10), y will stay the same
# and x values will be shifted to left
x_origin = x - 10
y_origin = y

# getting the dot product of two vectors as shown in the formula
x_rotated, y_rotated = np.dot(rotation_matrix, np.vstack((x_origin, y_origin)))

# recentering since we shifted to the left
x_rotated += 10


# open a figure and make scatterplot of y_rotated vs x_rotated
_= plt.figure()


# Divide x into 20 equal intervals
breakpoints = np.linspace(0, len(x), 21).astype(int)

# assign color to each sigmoid's interval
color_map = mpl.cm.get_cmap('rainbow', len(breakpoints) - 1)

# Plot the data points with colors based on the intervals
for i in range(len(breakpoints) - 1):
    start_index = breakpoints[i]
    end_index = breakpoints[i + 1]
    #take corresponding x,y coords and set color to ith
    _= plt.scatter(x_rotated[start_index:end_index], y_rotated[start_index:end_index],
                color=color_map(i))

# give names to the graph and axis
_= plt.xlabel('Time', size=7, fontweight='bold')
_= plt.ylabel('Sinusoid Value', size=7, labelpad=1, fontweight='bold')
_= plt.title('20 Sigmoids change their amplitude to 0 and back',
          loc='left', size=6, fontweight='bold')

# customizing the VALUES of ticks and their sizes
_= plt.yticks(np.arange(-10,15,5), size=7.5); _= plt.xticks(size=7.5)


_= plt.tick_params(
    axis='x',           # Apply to x-axis
    which = 'both',
    length = 0          # Remove ticks
)

_= plt.tick_params(
    axis='y',           # Apply to y-axis
    which = 'both',
    length = 0          # Remove ticks
)

_= plt.grid(False) # remove grids
_= plt.gca().set_facecolor('none')  # remove bckg

# verticle lines at x = 10+- std which should be dashed
_= plt.axvline(x=10 - std, color='gray', linestyle='--', linewidth = 2)
_= plt.axvline(x=10 + std, color='gray', linestyle='--', linewidth = 2)

# putting the text bolded and rotating it clockwise by 47 radians
_= plt.text(10.5, 1.5, '1 std == 4.84', fontsize=5, rotation=-47, fontweight='bold')

_= color_map = plt.cm.get_cmap('rainbow', 20)


_= plt.show() # displaying the result
```

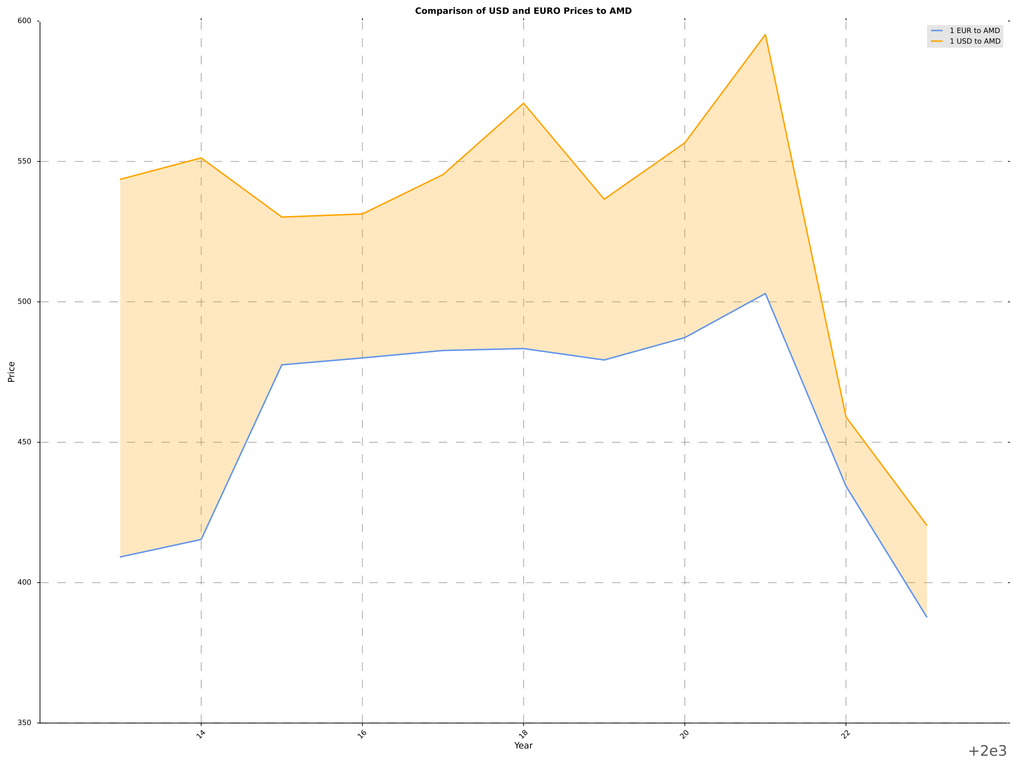


Problem 8

```
# loading datas
e_a = pd.read_csv('EUR_AMD Historical Data.csv')
```

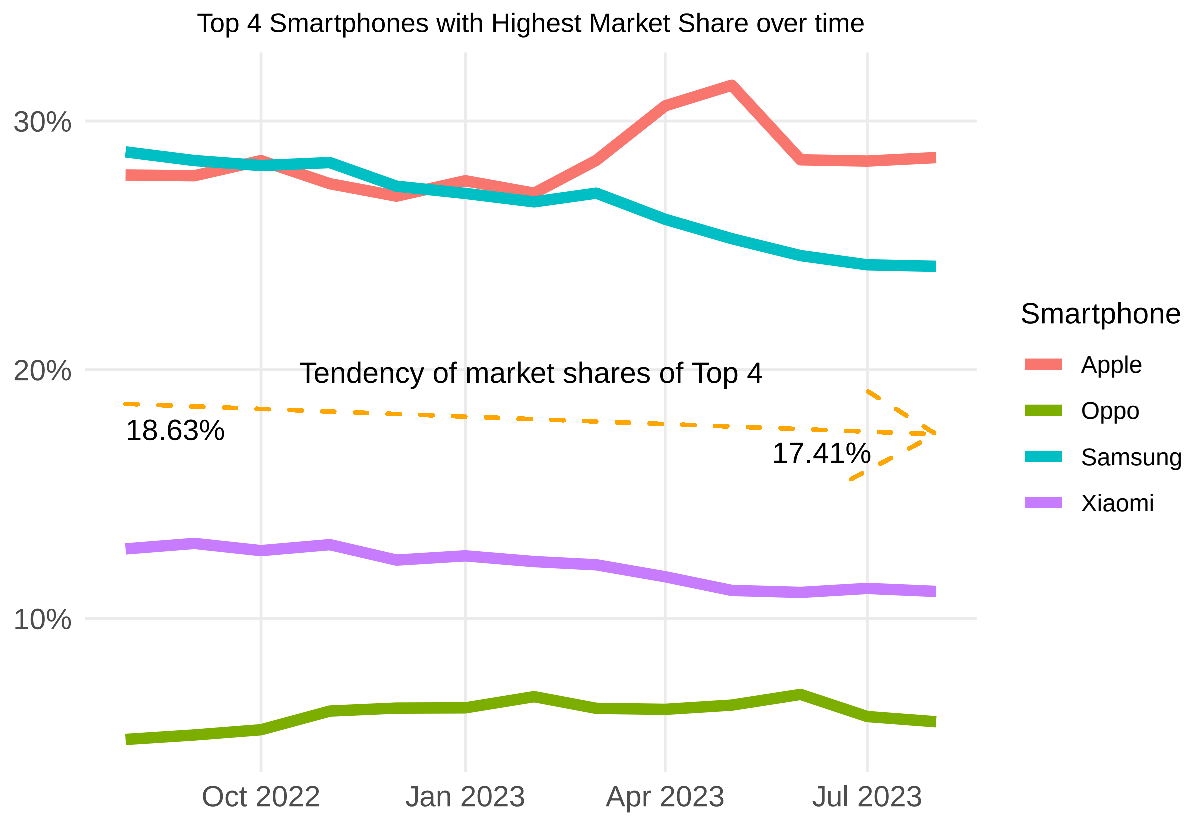
```python
a_a = pd.read_csv('USD_AMD Historical Data.csv')


def get_points(df):
    # Extracting the year from the 'Date' column
    df['Year'] = pd.to_datetime(df['Date']).dt.year

    # Grouping by 'Year' and calculating the mean of 'Price' for each year
    grouped_data = df.groupby('Year')['Price'].mean().reset_index()

    return grouped_data



data1 = get_points(e_a)
data2 = get_points(a_a)
# initializing figuresize
_= plt.figure(figsize=(12,9))

# x axis is the years y axis is the means of each group
_= plt.plot(data2['Year'], data2['Price'], color='#6495ED', label='1 USD to AMD')
_= plt.plot(data1['Year'], data1['Price'], color='orange', label='1 EUR to AMD')

# filling between two plots
_= plt.fill_between(data1['Year'], data1['Price'], data2['Price'], color='orange', alpha=0.25)

# grids should be dashed and gray
_= plt.grid(color='gray', linestyle='--', linewidth=0.3)

#making place for xlab
_= plt.subplots_adjust(bottom=0.15)

# labeling
_= plt.xlabel('Year', size=6, labelpad=1,  color='black')
_= plt.ylabel('Price', size=6, labelpad=1, color='black')
_= plt.title('Comparison of USD and EURO Prices to AMD', loc='center', size=6, fontweight='bold')

# adjust y ticks, set the values for x axis, rotate by 45 degree , adjust their position
_= plt.yticks(size=5, color='black')
_= plt.xticks(list(range(2014,2023,2)), size=5,  color='black', rotation = 45, va='center_baseline')

# setting legend size and position
legend = plt.legend(labels=['1 EUR to AMD', '1 USD to AMD'], prop={'size': 5})

_= plt.gca().set_facecolor('none')  # make the background transparent

_= plt.gca().spines['bottom'].set_color('black') # show bottom spine
_= plt.gca().spines['left'].set_color('black')   # Show left spine

_= plt.gca().spines['bottom'].set_linewidth(0.5) # set their widths
_= plt.gca().spines['left'].set_linewidth(0.5)

_= plt.gca().tick_params(axis='both', color='black', width=0.5, size=2)
```

```
_= plt.show()
```



Problem 9

```
# loading
df <-  read.csv("MobileSmartphones.csv")
# head(df)

# changing the type of Date column to date format by concatinating
# the day first, then converting into date with below format
df$Date <- as.Date(paste0(df$Date, "-01"), format = "%Y-%m-%d")


# dropping the columns except for the ones mentioned in select
# gather is equivalent to melt
# -Date indicates that Date will be the column remaining identical (id.vars = Date)
# key param. is equivalent to measure.var in melt, this column is named Smartphone
# value specifies the name of the new column that will contain the values corresponding to the measure.
df <- df %>%
  select(Date, Apple, Oppo, Samsung, Xiaomi) %>%
  gather(key = "Smartphone", value = "MarketShare", -Date) %>%
  filter(Smartphone %in% c('Apple', 'Oppo', 'Samsung', 'Xiaomi'))

# use df, x as Date, y as MarketShare, color by each unique entry from Smartphone
ggplot(df, aes(x = Date, y = MarketShare, color = Smartphone)) +
  geom_line(lwd = 2) + theme_minimal() +
  scale_y_continuous(labels = scales::percent_format(scale = 1)) + # make y axis scaled with %
  labs(x = '', y = '', color = 'Smartphone',
       title = 'Top 4 Smartphones with Highest Market Share over time') + # labeling, also for color pa

  theme(plot.title = element_text(hjust = 0.5, size = 10), # bring title to center
        panel.grid.minor = element_blank(), # remove minor grids
        axis.text = element_text(size = 11)) + # make ticks labels of size 11

  geom_segment(data = df, # create arrow with given starting and ending coords
               mapping = aes(x = min(Date), y = 18.63,
               xend = max(Date), yend = 17.41),
               color = 'orange', linewidth = 0.72, linetype = 'dashed',
               linejoin = 'round', lineend = 'round', # make it rounded and dashed
               arrow = arrow(length = unit(0.5, "inches"))) + # arrow sizes

  annotate("text", x = mean(df$Date), y = 18.63,
           label = "Tendency of market shares of Top 4",
           hjust = 0.5, vjust = -1) +

  annotate("text", x = min(df$Date), y = 18.63 - 2.3,
           label = "18.63%", hjust = 0, vjust = -1) +

  annotate("text", x = max(df$Date), y = 17.41 - 2,
           label = "17.41%", hjust = 1.65, vjust = -1)
```



This shows that Apple had the highest market share out of 4 types of smartphones. The overall tendency decreases but not considerably. There was something happening from (approximately) Feb 2023 to May 2023 for Apple, which can be our further focus in our analysis based on the context. For all other categories, market share either stayed almost the same, or decreased but not considerably.