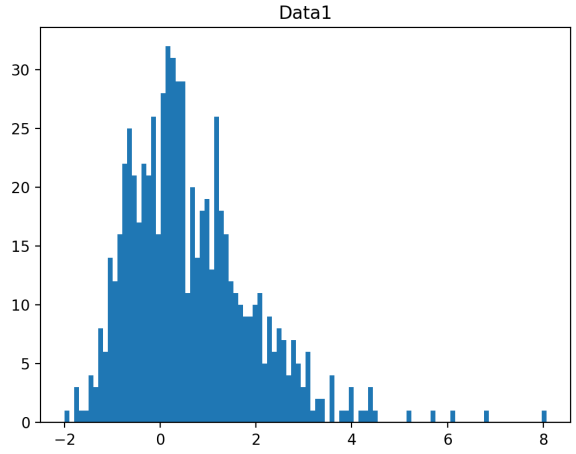
Statistical Analysis, Simulation, and Modeling.Part II
Semester-end project assignments

Lee Jui Chi
26001803777

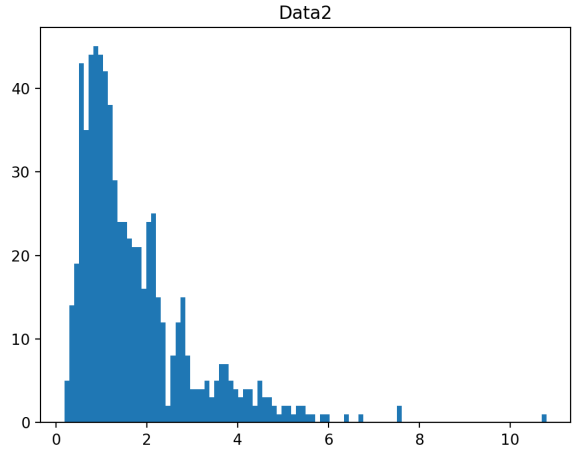
January 15, 2020

1 Conduct a preliminary analysis of the data. Based on the preliminary analysis, select one data sample would describe human reaction time to a stimulus

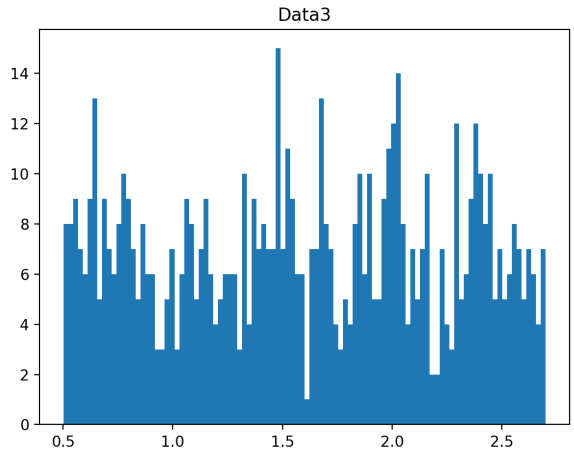
Get the samples from 4 .csv files and visualize them to conduct a preliminary analysis of the data. Figure 1 shows 4 histograms of samples, and Figure 2 shows the human reaction time histogram from humanbenchmark.com



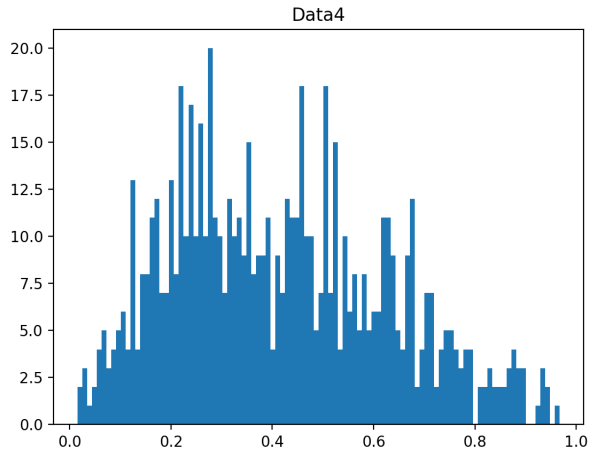
(a) Data sample 1



(b) Data sample 2



(c) Data sample 3



(d) Data sample 4

Figure 1: The histograms of data sample 1 to data sample 4

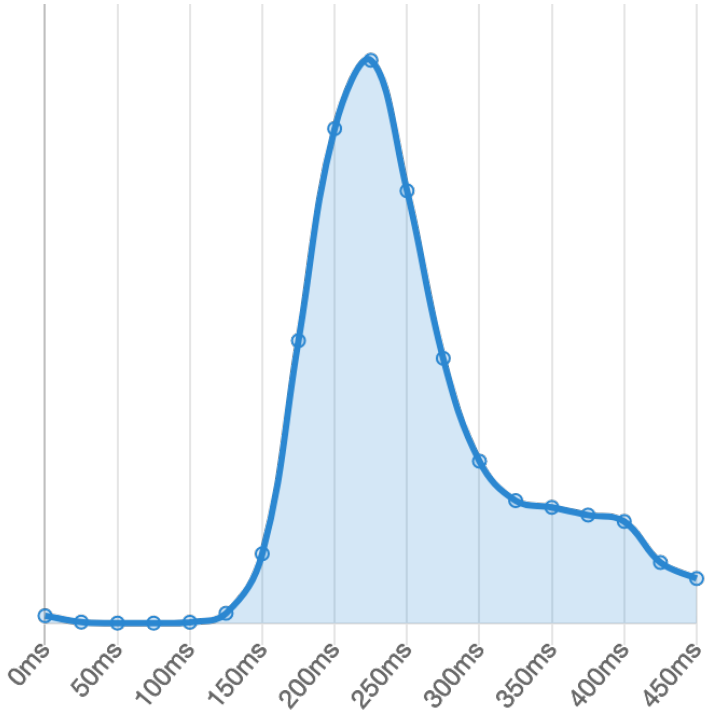


Figure 2: The statistics of human reaction time from humanbenchmark.com

Comparing 4 data samples histograms and the statistics I got from humanbenchmark.com, in my opinion, data sample 2 is the best one to describe human reaction time to a stimulus because the shape of them looks very similar. Although data sample 1 also has a similar shape, it contains the negative numbers which is impossible for human reaction time.
Candidate: data sample 2
The code down below is for plotting the histograms

Plotting the samples

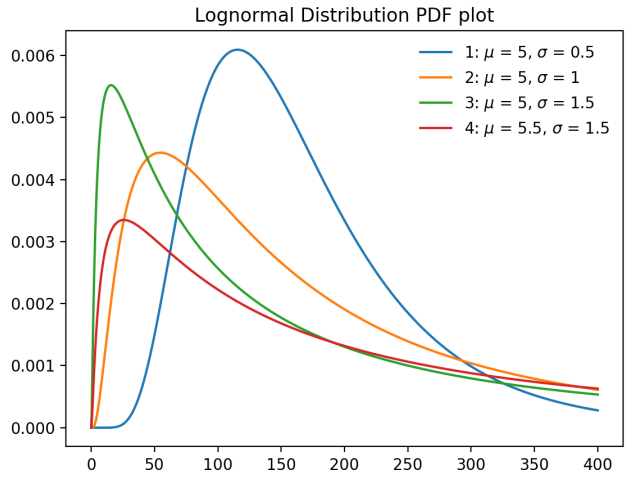
```
import csv
from matplotlib import pyplot as plt

data1 = []
data2 = []
data3 = []
data4 = []

# Read the number in every row and add it into the list
with open('datafile1.csv') as csvfile1:
    reader = csv.reader(csvfile1)
    for row in reader:
        data1.append(float(row[0]))

with open('datafile2.csv') as csvfile2:
    reader = csv.reader(csvfile2)
    for row in reader:
        data2.append(float(row[0]))

with open('datafile3.csv') as csvfile3:
    reader = csv.reader(csvfile3)
    for row in reader:
        data3.append(float(row[0]))

with open('datafile4.csv') as csvfile4:
    reader = csv.reader(csvfile4)
    for row in reader:
        data4.append(float(row[0]))

# building histogram with 100 bins
# hist1 = plt.hist(data1, bins=100)
```

```
# hist2 = plt.hist(data2, bins=100)
# hist3 = plt.hist(data3, bins=100)
hist4 = plt.hist(data4, bins=100)

# plt.title("Data1")
# plt.title("Data2")
# plt.title("Data3")
plt.title("Data4")
plt.show()
```

2 Plot PDFs of the three distributions, and explore the shapes of the PDFs by varying (assigning different values to) their parameters

2.1 Lognormal distributions (two parameters, μ and σ)

$$f(x) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}; \; x > 0; \; \sigma > 0.$$



Discussion:

- The lognormal distribution is a distribution skewed to the right
- The pdf starts at zero, increases to its mode, and decreases thereafter
- The degree of skewness increases as σ increases, for a given μ
- For the same σ, the "peak" of pdf decrease when μ increases.

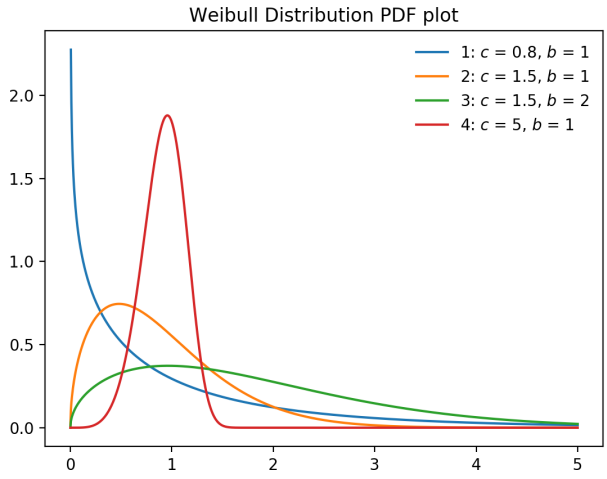
Plotting the lognormal distribution pdf with different parameters

```
import numpy as np
from matplotlib import pyplot as plt
import scipy.stats as ss
from scipy.stats import lognorm

# Lognormal Distribution from scipy: https://docs.scipy.org/doc/scipy/reference/generated/
    scipy.stats.lognorm.html
# Wikipedia: https://en.wikipedia.org/wiki/Log-normal_distribution

mu1 = 5
mu2 = 5.5
sigma1 = 0.5
sigma2 = 1
sigma3 = 1.5

# show from 0 to 400
```

```
x = np.linspace(0,400, 1000)

# lognormal pdf function plot from scipy, assigning sigma and mu

# A common parametrization for a lognormal random variable Y is
# in terms of the mean, mu, and standard deviation, sigma, of the
# unique normally distributed random variable X such that exp(X) = Y.
# This parametrization corresponds to setting s = sigma and scale = exp(mu).
y1 = lognorm.pdf(x=x, s=sigma1, scale=np.exp(mu1))
y2 = lognorm.pdf(x=x, s=sigma2, scale=np.exp(mu1))
y3 = lognorm.pdf(x=x, s=sigma3, scale=np.exp(mu1))
y4 = lognorm.pdf(x=x, s=sigma3, scale=np.exp(mu2))

fig, ax = plt.subplots(1, 1)
ax.plot(x, y1, label = '1: $\mu$ = ' + str(mu1) + ", $\sigma$ = " + str(sigma1))
ax.plot(x, y2, label = '2: $\mu$ = ' + str(mu1) + ", $\sigma$ = " + str(sigma2))
ax.plot(x, y3, label = '3: $\mu$ = ' + str(mu1) + ", $\sigma$ = " + str(sigma3))
ax.plot(x, y4, label = '4: $\mu$ = ' + str(mu2) + ", $\sigma$ = " + str(sigma1))

ax.legend(loc='best', frameon=False)
plt.title("Lognormal Distribution PDF plot")
plt.show()
```

2.2 Weibull distribution(two parameters, c and b)

$$f(x) = \frac{cx^{c-1}}{b^c e^{(x^c/b^c)}}; x > 0;\ c > 0,\ b > 0.$$



Discussion:

- The Weibull distribution is a distribution skewed to the right
- For $0 < c < 1$, the pdf tends to ∞ as x approaches 0 from above and is strictly decreasing
- For $c = 1$, the pdf tends to $1/\lambda$ as x approaches zero from above and is strictly decreasing
- For $c > 1$, the pdf tends to 0 as x approaches zero from above, increases until its mode and decreases after it
- For the same c, if b increases, it will stretching out the pdf and the "peak" of pdf will also decrease

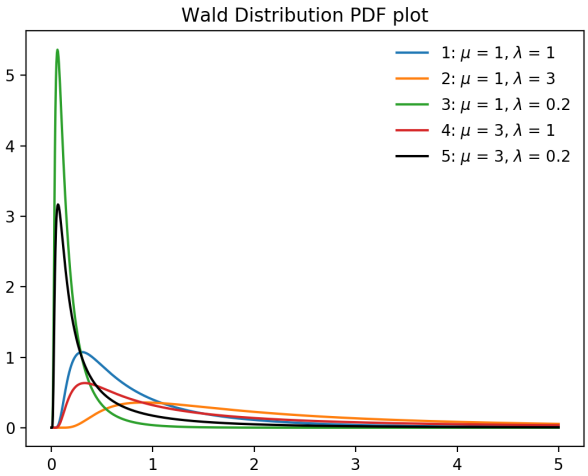
Plotting the Weibull distribution PDF with different parameters

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import weibull_min

# Weibull Distribution from scipy: https://docs.scipy.org/doc/scipy/reference/generated/
    scipy.stats.weibull_min.html
# Wikipedia: https://en.wikipedia.org/wiki/Weibull_distribution
```

```
# show from 0 to 5
x = np.linspace(0, 5, 1000)

c1 = 0.8
c2 = 1.5
c3 = 5
b1 = 1
b2 = 2


y1 = weibull_min.pdf(x, c1, scale=b1)
y2 = weibull_min.pdf(x, c2, scale=b1)
y3 = weibull_min.pdf(x, c2, scale=b2)
y4 = weibull_min.pdf(x, c3, scale=b1)

fig, ax = plt.subplots(1, 1)

ax.plot(x, y1, label='1: $c$ = ' + str(c1) + ", $b$ = " + str(b1))
ax.plot(x, y2, label='2: $c$ = ' + str(c2) + ", $b$ = " + str(b1))
ax.plot(x, y3, label='3: $c$ = ' + str(c2) + ", $b$ = " + str(b2))
ax.plot(x, y4, label='4: $c$ = ' + str(c3) + ", $b$ = " + str(b1))


ax.legend(loc='best', frameon=False)
plt.title("Weibull Distribution PDF plot")

plt.show()
```

2.3 Wald distribution(two parameters, μ and λ)

$$f(x) = \sqrt{\frac{\lambda}{2\pi x^3}} e^{-\lambda\frac{(x-\mu)^2}{2\mu^2 x}};\ x > 0;\ \mu > 0,\ \lambda > 0.$$



Discussion:

- The Wald distribution is a distribution skewed to the right
- The pdf starts at zero, increases to its mode, and decreases thereafter
- For the same μ, if λ increases, it will stretching out the pdf and the "peak" of pdf will also decrease
- For the same λ, if μ increases, the "peak" of pdf will decrease

Plotting the Wald distribution PDF with different parameters

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import invgauss
```

```
# Wald Distribution from scipy: https://docs.scipy.org/doc/scipy/reference/generated/scipy
    .stats.invgauss.html
# Wikipedia: https://en.wikipedia.org/wiki/Inverse_Gaussian_distribution
mu1 = 1
mu2 = 3
l1 = 1
l2 = 3
l3 = 0.2

x = np.linspace(0, 5, 1000)

# scale = lambda
y1 = invgauss.pdf(x, mu1, scale=l1)
y2 = invgauss.pdf(x, mu1, scale=l2)
y3 = invgauss.pdf(x, mu1, scale=l3)
y4 = invgauss.pdf(x, mu2, scale=l1)
y5 = invgauss.pdf(x, mu2, scale=l3)

fig, ax = plt.subplots(1, 1)

ax.plot(x, y1, label = '1: $\mu$ = ' + str(mu1) + ", $\lambda$ = " + str(l1))
ax.plot(x, y2, label = '2: $\mu$ = ' + str(mu1) + ", $\lambda$ = " + str(l2))
ax.plot(x, y3, label = '3: $\mu$ = ' + str(mu1) + ", $\lambda$ = " + str(l3))
ax.plot(x, y4, label = '4: $\mu$ = ' + str(mu2) + ", $\lambda$ = " + str(l1))
ax.plot(x, y5, 'k', label = '5: $\mu$ = ' + str(mu2) + ", $\lambda$ = " + str(l3))

ax.legend(loc='best', frameon=False)
plt.title("Wald Distribution PDF plot")
plt.show()
```

3 Use the data sample selected in (I) and obtain parameter estimates for the three models of (II). Assess and write down the accuracy of the parameter estimates obtained by constructing the corresponding CIs

For estimating the parameters for the three models, I used a fitting function from Scipy, in the form of distribution.fit(data). It will conduct the parameter estimates for generic data, return shape, loc, and scale. By researching Scipy's documentation and Wikipedia, I found that they are representing different parameters for those three distributions.

- For Lognormal model, μ = log(scale) = 0.3252398215490648 , and σ = shape = 0.6625941633383446
- For Weibull model, c = shape = 1.5264770156305112, and b = scale = 1.9331832297950118
- For Wald model, μ = shape = 0.5433642842402152, and λ = scale = 3.177165829971904

For checking the accuracy of the parameter estimates, I obtained the 95 percent confidence interval of three models by using a Scipy function in the form of distribution.interval(). It will return the endpoints of the range that contains a specific percent of the distribution. Then I visualize the confidence interval with the data2 in Figure 3 to Figure 5.

- Lognormal's interval: (0.37778845799120303, 5.072838475583722)
- Weibull's interval: (0.173921120962139, 4.546188137441968)
- Wald's interval: (0.3982995166152734, 5.139352801182957)

From the numbers and the plots, I found that Lognormal model and Wald model has similar endpoints. However, Weibull model's endpoints are slightly smaller than the others, and the left endpoint looks like almost equal to or smaller than the data's smallest value, which is not good for accuracy. The other two model looks more nature to me.

Parameter estimate and confidence interval plot for lognormal distribution model

```
# https://medium.com/@rrfd/what-is-maximum-likelihood-estimation-examples-in-python
    -791153818030
# import libraries
import numpy as np
from matplotlib import pyplot as plt
```

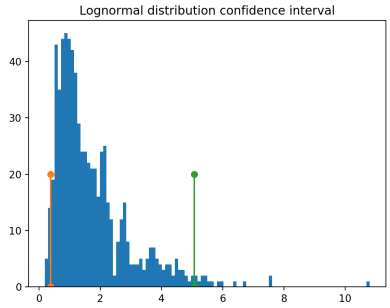


Figure 3: Lognormal distribution confidence interval plot

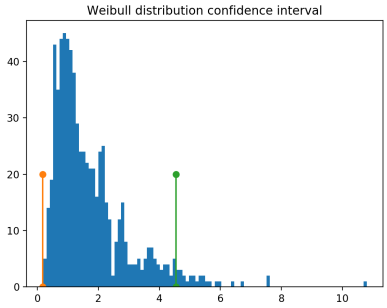


Figure 4: Weibull distribution confidence interval plot

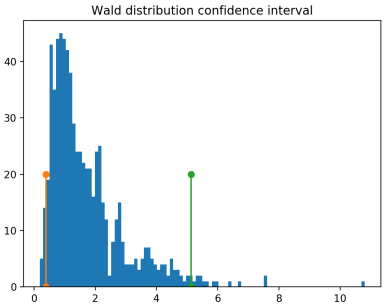


Figure 5: Wald distribution confidence interval plot

```
import csv
from scipy.stats import lognorm

data2 = []

with open('datafile2.csv') as csvfile2:
    reader = csv.reader(csvfile2)
    for row in reader:
        data2.append(float(row[0]))

plt.figure()
_ = plt.hist(data2,  bins=100)

# Parameter estimates for generic data
# the argument floc=0 to ensure that it does not treat the location as a free parameter
shape1, loc1, scale1 = lognorm.fit(data2, floc=0)
mu1 = np.log(scale1)
sigma1 = shape1
print("Estimated mu = " + str(mu1))
print("Estimated sigma = " + str(sigma1))
# 0.95 is the alpha value, which specifies a 95 percentile point, as the corresponding
    1.96 standard deviations of the mean is given in the formula.
ci1 = lognorm.interval(0.95, s=sigma1, loc=loc1, scale=scale1)
print("Lognorm function CI = " + str(ci1))
# confidence interval left line
one_x12, one_y12 = [ci1[0],ci1[0]], [0, 20]
# confidence interval right line
two_x12, two_y12 = [ci1[1],ci1[1]], [0, 20]

plt.plot(one_x12, one_y12, two_x12, two_y12, marker = 'o')
plt.title("Lognormal distribution confidence interval")
plt.show()
```

Parameter estimate and confidence interval plot for Weibull distribution model

```
# import libraries
import numpy as np, pandas as pd
from matplotlib import pyplot as plt
from scipy.stats import weibull_min
import csv

data2 = []

with open('datafile2.csv') as csvfile2:
    reader = csv.reader(csvfile2)
    for row in reader:
        data2.append(float(row[0]))
plt.figure()
_ = plt.hist(data2,  bins=100)
```

```
# Parameter estimates for generic data
shape2, loc2, scale2= weibull_min.fit(data2, floc=0)
c = shape2
b = scale2
print("Estimated c = " + str(c))
print("Estimated b = " + str(b))
ci2 = weibull_min.interval(0.95, c, loc=loc2, scale=scale2)
print("Weibull CI = " + str(ci2))
print("————————————————————")
# cnfidence interval left line
one_x12_, one_y12_ = [ci2[0],ci2[0]], [0, 20]
# cnfidence interval right line
two_x12_, two_y12_ = [ci2[1],ci2[1]], [0, 20]
plt.plot(one_x12_, one_y12_, two_x12_, two_y12_, marker = 'o')
plt.title("Weibull distribution confidence interval")
plt.show()
```

Parameter estimate and confidence interval plot for Wald distribution model

```
import numpy as np, pandas as pd
from matplotlib import pyplot as plt
import csv
from scipy.stats import invgauss

data2 = []
with open('datafile2.csv') as csvfile2:
    reader = csv.reader(csvfile2)
    for row in reader:
        data2.append(float(row[0]))
plt.figure()
_ = plt.hist(data2,  bins=100)

# Parameter estimates for generic data
shape3, loc3, scale3 = invgauss.fit(data2, floc=0)
mu3 = shape3
print(mu3,loc3, scale3)
print("Estimated mu = " + str(mu3))
print("Estimated lambda = " + str(scale3))

ci3 = invgauss.interval(0.95, mu3, loc=loc3, scale=scale3)
print("Wald distribution CI = " + str(ci3))

# cnfidence interval left line
one_x12__, one_y12__ = [ci3[0],ci3[0]], [0, 20]
# cnfidence interval right line
two_x12__, two_y12__ = [ci3[1],ci3[1]], [0, 20]
plt.plot(one_x12__, one_y12__, two_x12__, two_y12__, marker = 'o')
plt.title("Wald distribution confidence interval")
plt.show()
```

4 Consider the three models from (III) and conduct a model selection analysis. Select "the best model"for the data(i.e. the "best" distribution from (II) to describe the human reaction time –the data selected in (I)). Justify your choice of "the best model"(e.g. use Q-Q plots and/or quantitative model selection criteria)

For selecting models, I decided plot the Q-Q plot first. Parameters are fit in models by using distribution.fit() from Scipy. Figure 6 - 8 shows three Q-Q plots of models.

Again, I found that Lognormal model and Wald model have similar result, but Weibull model is more different. In my opinion, both Lognormal model and Wald model look good from the plots for describing the human reaction time. However,

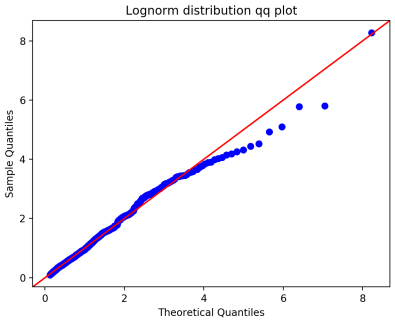


Figure 6: Lognormal distribution qq plot

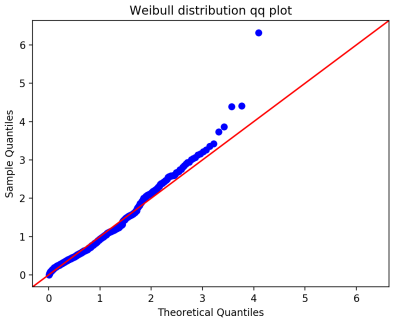


Figure 7: Weibull distribution qq plot

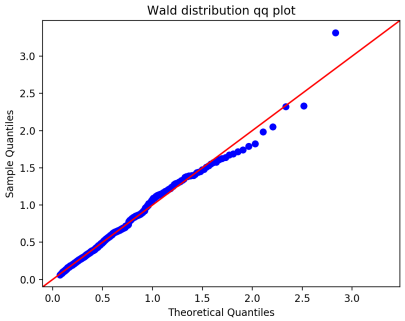


Figure 8: Wald distribution qq plot

Q-Q plot is not enough for choosing the best model from them because I can't really tell Lognormal model or Wald model is a better one.

To achieve it, I decided to conduct Information Criteria analysis(Week 11 slide 18-19). I chose Akaike Information Criterion (AIC) as my model selection method.

AIC = -2lnL + 2m

L means the log-likelihood, which is calculated in the form of np.sum(np.log(model))
m is the number of model parameters, in this case they all have 2

- Lognormal's log-likelihood = -932.8100298110222, AIC = 1869.6200596220444
- Weibull's log-likelihood = -988.3942459920452, AIC = 1980.7884919840903
- Wald's log-likelihood = -930.1608286412311 , AIC = 1864.3216572824622

The criteria are used to rank candidate models, smallest is the best.
Wald AIC<Lognormal AIC<Weibull AIC

Plotting qq plots and calculate AIC for three models

```
import numpy as np
from matplotlib import pyplot as plt
import csv
from scipy.stats import weibull_min, invgauss, lognorm
import statsmodels.api as sm

data2 = []
# reading the data and append it into the list
with open('datafile2.csv') as csvfile2:
    reader = csv.reader(csvfile2)
    for row in reader:
        data2.append(float(row[0]))

# sort the data
data2 = np.sort(data2)

# number for parameters in the distribution for k value for AIC, each one has two
    parameters need to be estimated
num_params = 2

# Parameter estimates for generic data
shape1, loc1, scale1 = lognorm.fit(data2, floc=0)
mu1 = np.log(scale1)
sigma1 = shape1
y1 = lognorm.pdf(data2, s=sigma1, scale=np.exp(mu1))
log_likelihood1 = np.sum(np.log(y1))
print("Lognorm loglikelihood = " + str(log_likelihood1))
aic1= -2 * log_likelihood1 + 2 * num_params
print("Lognorm AIC = " + str(aic1))

# https://stackoverflow.com/questions/33070724/determine-weibull-parameters-from-data
```

```
# Parameter estimates for generic data
shape2, loc2, scale2 = weibull_min.fit(data2, floc=0)
c = shape2
b = scale2
y2 = weibull_min.pdf(data2, c, scale=b)
log_likelihood2 = np.sum(np.log(y2))
print("Weibull loglikelihood = " + str(log_likelihood2))
aic2= -2 * log_likelihood2 + 2 * num_params
print("Weibull AIC = " + str(aic2))

# https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.invgauss.html
# the argument floc=0 to ensure that it does not treat the location as a free parameter
# Parameter estimates for generic data
shape3, loc3, scale3 = invgauss.fit(data2, floc=0)
mu3 = shape3
lambda3 = scale3
# fitting the data with the estimated parameters
y3 = invgauss.pdf(data2, mu3, scale=lambda3)
# calculate the log_likelihood
log_likelihood3 = np.sum(np.log(y3))
print("Wald loglikelihood = " + str(log_likelihood3))
# calculate AIC
aic3= -2 * log_likelihood3 + 2 * num_params
print("Wald AIC = " + str(aic3))

# ploting qq plot with 3 distributions, fit the parameters by calling distribution.fit()
sm.qqplot(data2, lognorm, fit=True, loc=0, line='45')
plt.title('Lognorm distribution qq plot')
sm.qqplot(data2, weibull_min, fit=True, loc=0, line='45')
plt.title('Weibull distribution qq plot')
sm.qqplot(data2, invgauss, fit=True, loc=0, line='45')
plt.title('Wald distribution qq plot')
plt.show()
```

5 Conclusion

From the Q-Q plots and Information Criteria analysis, I concluded that Wald distribution is the best model for describing the human reaction time selected in data 2. Its Q-Q plot looks good and it also has a smallest Akaike Information Criterion.

Best model: Wald distribution model