Decision Support System (DSS) Span Analysis

https://github.com/jupyter/docker-stacks
https://hub.docker.com/r/jupyter/r-notebook/tags/

(optional) docker pull jupyter/r-notebook:latest

We want to make the DDS Prototype ~/analysis/ directory linked to the Juypter container. Use the following to mount the analysis directory (i.e. current working directory) as a volume in the Juypter container. Note that the directory needed to be added via the Docker Desktop Dashboard on Mac.

docker run -it –rm -d -p 10000:8888 -v ${PWD}:/home/jovyan/work –name notebook jupyter/r-notebook:latest

To find the token from the container:
docker exec -it notebook jupyter server list

Navigate to the container UI and enter the token: http://localhost:10000

DSS System Context

The following diagram depicts the context for the DSS. The DSS operator interacts with the DSS Prototype for decision assitance. The DSS relies on a aircraft database to gather real-time flight data to review in decision support algorithms.

DSS Container Architecture

Nine containers are instantiated as part of the DSS architecuture. Six provide the DSS implementation while the additional 3 support collection and calculation of metrics. Each application container was designed around the 12-Factor Application "Single Responsibility Principle"; e.g. each app has one purpose to enable rapid insertion of new capabilities with low cohesion to other functionality. At this time, all responses are canned without underlying calculation to focus on meeting the 500 ms hypothesis pryor to buddening the app with calculation latency.

DSS Applications

- opensky-int: Provides the OpenSky API for flight data. The app provides data about aircraft within 60 NM of Richmond (RIC) or Dulles (IAD) airports.
- tm-server: Provides sensor track data (e.g. OpenSky) and system tracks to support DSS services. System tracks represents the system-wide common understand of a track objects state for decisions.

- wa-app: The Weapon Assessment Application determines which weapons are capable to successfully engage a target. The wa-app use the tm-server api to get track data.
- te-app: The Trail Engage Application determines the success rate of an engagement with a specific weapon target pairing. The predicted track kinematic data at engagement time is provided; therefore, the current track kinematics from the tm-server are not queried prior to providing a response.
- test-app: Provides and ability to generate automated test. the test-app uses the dss-ui to call dss-ui endpoint to replicate operator interactions with the DSS Prototype.
- dss-ui: Provides a simple graphical interface to launch DSS services.

DSS Tools

- telem-jaeger: The open source Jaeger containter collects "span" data from the DSS applications. Spans collection duration data for service call to over container; e.g. latency. This the fundamental data that is being analysed here.
- grafana: The open source Grafana container connects to the telem-jaeger container to create visualization dashboard. Also, Grafana faciliates the export of data as a .csv file for analysis.
- notebook: The Jupyter Notebook container support analysis of the data exported by Grafana. It is the core datafile use by this tool.

Hypothesis

Hypothesis is "innocent until proven guilty." We'll assume that SpaceX and others have proven that DevSecOps tech can meet hard-real-time requirements but nothing available in the body of knowledge documents this.

Hypothesis: Modern DevSecOps architectures can be designed to meet hard-real-time latency (μ) requirements using modern computing environments and computing infrastructure.

$H_0 : \mu \leq 500ms$ with jitter within latency bounds
$H_a : \mu > 500ms$ with jitter exceeding latency bounds

Murphy, Alvin C. and Moreland Jr, James D. 'Integrating AI Microservices into Hard-Real-Time SoS to Ensure Trustworthiness of Digital Enterprise Using Mission Engineering'. 1 Jan. 2021 : 38 – 54.

```r
options(warn=-1)
```

```r
install.packages("stringr")                # Install packages and libraries in R
library("stringr", quietly = T)
```

```r
install.packages("dplyr")
library("dplyr", quietly = T)

install.packages("ggplot2")
install.packages("GGally")
library("ggplot2", quietly = T)
library("GGally", quietly = T)
```

```
Updating HTML index of packages in '.Library'

Making 'packages.html' ...
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' ...
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' ...
 done

Updating HTML index of packages in '.Library'

Making 'packages.html' ...
 done
```

```r
setwd('/home/jovyan/work/data')
```

```r
options(warn=-1)
spanData <- read.csv('DSS Span Data-data-2022-05-02 18_38_26.csv', header = TRUE)
attach(spanData)
```

Exploratory Data Analysis

```r
head(spanData)
```

A data.frame: 6 × 4

	Trace.ID <chr>	Trace.name <chr>	Start.time <chr>	Duration <chr>
1	9ee3577fb1b427bc4fd17fec5154be7d	dss-prototype: /TE	2022-05-02 10:25:01.366	36.0 ms
2	f05ddc4dc13aff5c30d8011b2a402401	dss-prototype: /tracks	2022-05-02 10:25:00.309	43.3 ms
3	2bd901fbbfc9ee8dfa7c9629d93a1567	dss-prototype: /IAD	2022-05-02 10:24:58.818	464 ms
4	69a48381a14e79da08aa2353f7db4b2	dss-prototype: /RIC	2022-05-02 10:24:57.307	494 ms
5	e83037dcb9438c04dd12fba373b5502f	dss-prototype: /WA	2022-05-02 10:24:56.128	139 ms
6	7e381cd880adb670bb9627ca47020938	dss-prototype: /TE	2022-05-02 10:24:55.081	30.3 ms

```r
summary(spanData)
```

```
   Trace.ID          Trace.name          Start.time          Duration
 Length:100         Length:100         Length:100         Length:100
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

Convert Data into Useable Metrics

```r
## Dictionary for converting data

DSSoperations <- c(
    "dss-prototype: /IAD" = "Get Dulles Airport Data (External)",
    "dss-prototype: /RIC" = "Get Richmond Airport Data (External)",
    "dss-prototype: /tracks" = "Get Stored Local DSS Tracks (Internal)",
    "dss-prototype: /TE" = "Trial Engage (Internal)",
    "dss-prototype: /WA" = "Assess Weapons (Internal)"
)
```

```r
DSSnumContainers <- c(
    "dss-prototype: /IAD" = 3,
    "dss-prototype: /RIC" = 3,
    "dss-prototype: /tracks" = 2,
    "dss-prototype: /TE" = 2,
    "dss-prototype: /WA" = 3
)

# Used docker run -it --rm gophernet/traceroute opensky-network.org
# to determine hops from Docker network to OpenSky -- number may change in different envir

DSStraceRoute <- c(
    "dss-prototype: /IAD" = 14,
    "dss-prototype: /RIC" = 14,
    "dss-prototype: /tracks" = 0,
    "dss-prototype: /TE" = 0,
    "dss-prototype: /WA" = 0
)
```

```r
# Convert character data into numeric metrics

spanMetrics <- spanData

# spanMetrics = cbind(spanMetrics, operation, containers)

for(i in 1:nrow(spanMetrics)) {       # for-loop over rows

    # Add operation and container data
    spanMetrics$useCase = DSSoperations[Trace.name]
    spanMetrics$numContainers = DSSnumContainers[Trace.name]
    spanMetrics$extNetworkHops = DSStraceRoute[Trace.name]

    # Convert span duration

    char = spanMetrics[i,4]
    len = str_length(char)
    duration = str_sub(char,1,(len-3))
    units = str_sub(char,(len-1),len)
    duration = as.numeric(duration)

    # print(duration)
```

```r
    # print(units)

    if(units == 'ms') {
        duration = duration / 1000                 # Convert to ms
    } else if (units == 'µs') {
        duration = duration / 1000000              # Convert to µs
    } else if (units == ' s') {
        duration = duration
    } else {
        print ('Unable to find specified units')
        print (units)
    }
    spanMetrics[i,4] = duration

    # Convert time

    time = spanMetrics[i,3]
    epoch <- as.POSIXct(time)
    epoch_int <- as.integer(epoch)
    spanMetrics[i,3] = epoch_int
}

# Convert columns for char to numeric

spanMetrics$Duration = as.numeric(spanMetrics$Duration)
spanMetrics$Start.time = as.numeric(spanMetrics$Start.time)

head(spanMetrics)
summary(spanMetrics)
```

A data.frame: 6×7

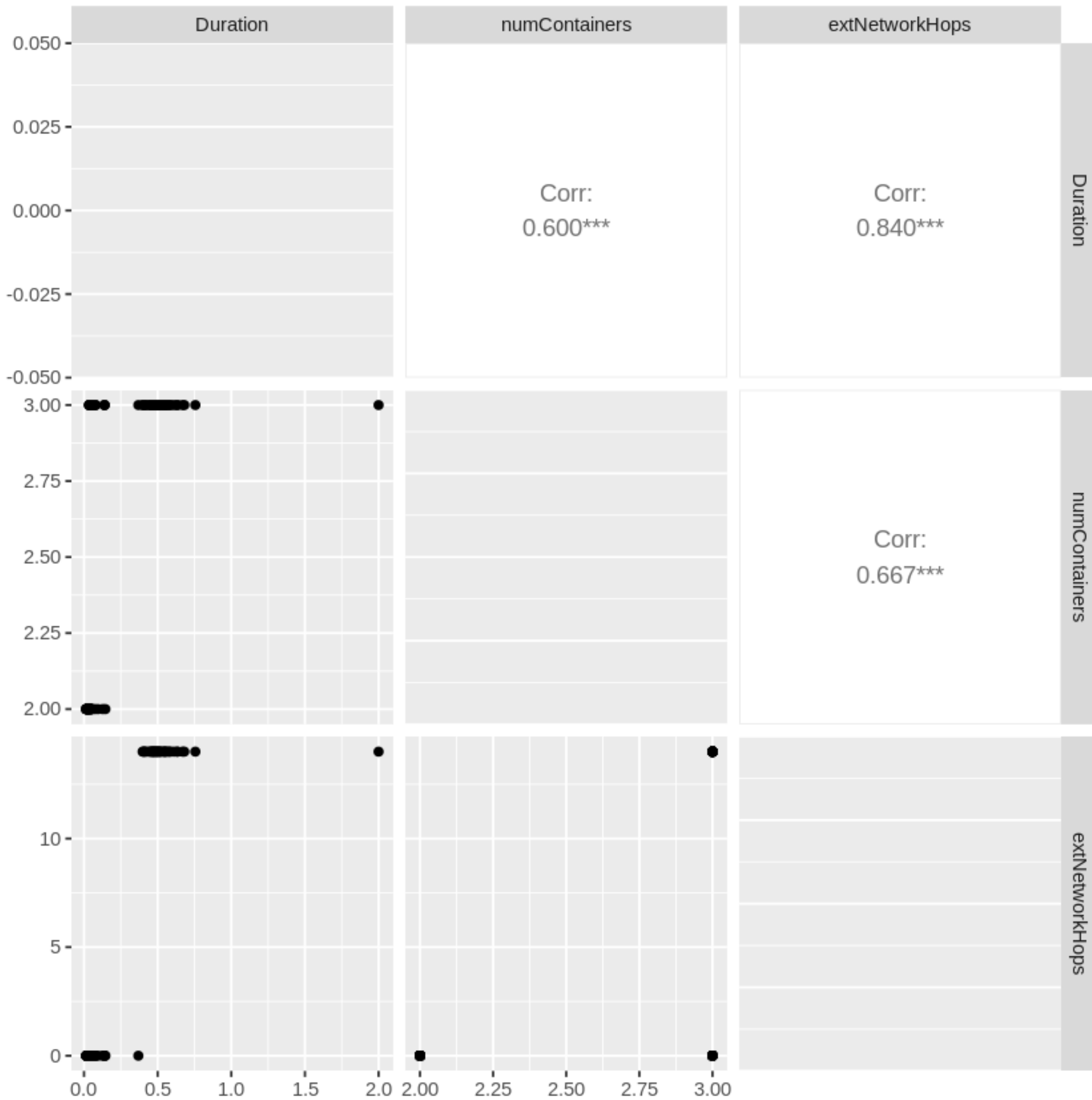
	Trace.ID	Trace.name	Start.time	Duration	useCase	numContainers	extNetworkHops
	<chr>	<chr>	<dbl>	<dbl>	<chr>	<dbl>	<dbl>
1	9ee3577fb1bd427bc4fc17fdc514847d1	ess-prototype: /TE	1651587701	0.0360	Trial Engage (Internal)	2	0

	Trace.ID <chr>	Trace.name <chr>	Start.time <dbl>	Duration <dbl>	useCase <chr>	numContainers <dbl>	extNetworkHops <dbl>
2	f05ddc4dc13[illegible]	dss-prototype: /tracks	[illegible]	0.0433	Get Stored Local DSS Tracks (Internal)	2	0
3	2bd901fbbfc[illegible]	dss-prototype: /IAD	[illegible]	0.4640	Get Dulles Airport Data (External)	3	14
4	69a48381a14[illegible]	dss-prototype: /RIC	[illegible]	0.4940	Get Richmond Airport Data (External)	3	14
5	e83037dcb94[illegible]	dss-prototype: /WA	[illegible]	0.1390	Assess Weapons (Internal)	3	0
6	7e381cd880a[illegible]	dss-prototype: /TE	[illegible]	0.0303	Trial Engage (Internal)	2	0

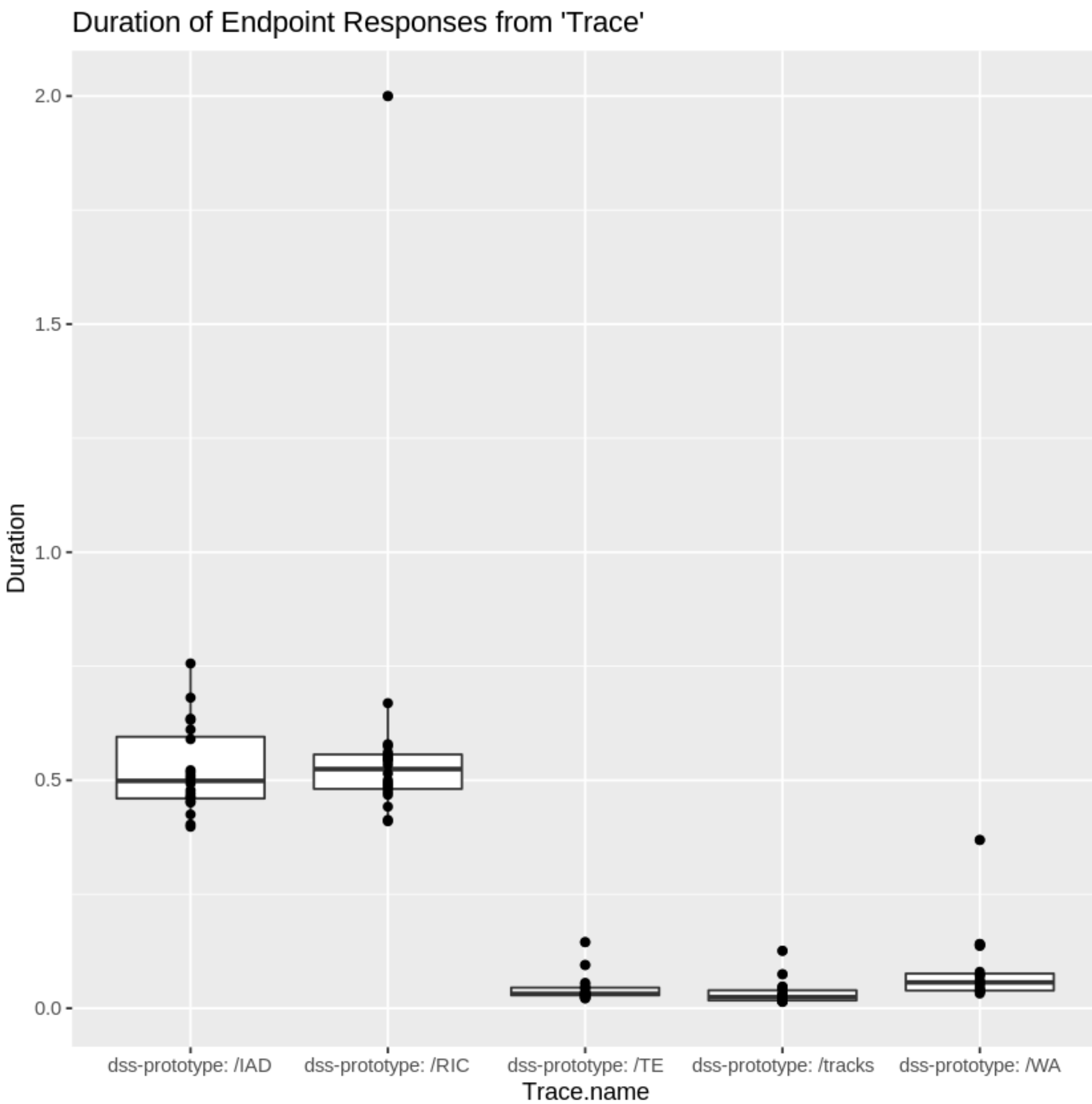
```
   Trace.ID           Trace.name          Start.time           Duration
 Length:100         Length:100         Min.   :1.651e+09   Min.   :0.01390
 Class :character   Class :character   1st Qu.:1.651e+09   1st Qu.:0.03275
 Mode  :character   Mode  :character   Median :1.651e+09   Median :0.07375
                                       Mean   :1.651e+09   Mean   :0.25404
                                       3rd Qu.:1.651e+09   3rd Qu.:0.48450
                                       Max.   :1.651e+09   Max.   :2.00000
   useCase          numContainers extNetworkHops
 Length:100         Min.   :2.0   Min.   : 0.0
 Class :character   1st Qu.:2.0   1st Qu.: 0.0
 Mode  :character   Median :3.0   Median : 0.0
                    Mean   :2.6   Mean   : 5.6
                    3rd Qu.:3.0   3rd Qu.:14.0
                    Max.   :3.0   Max.   :14.0
```

Exploratory Analysis Plots

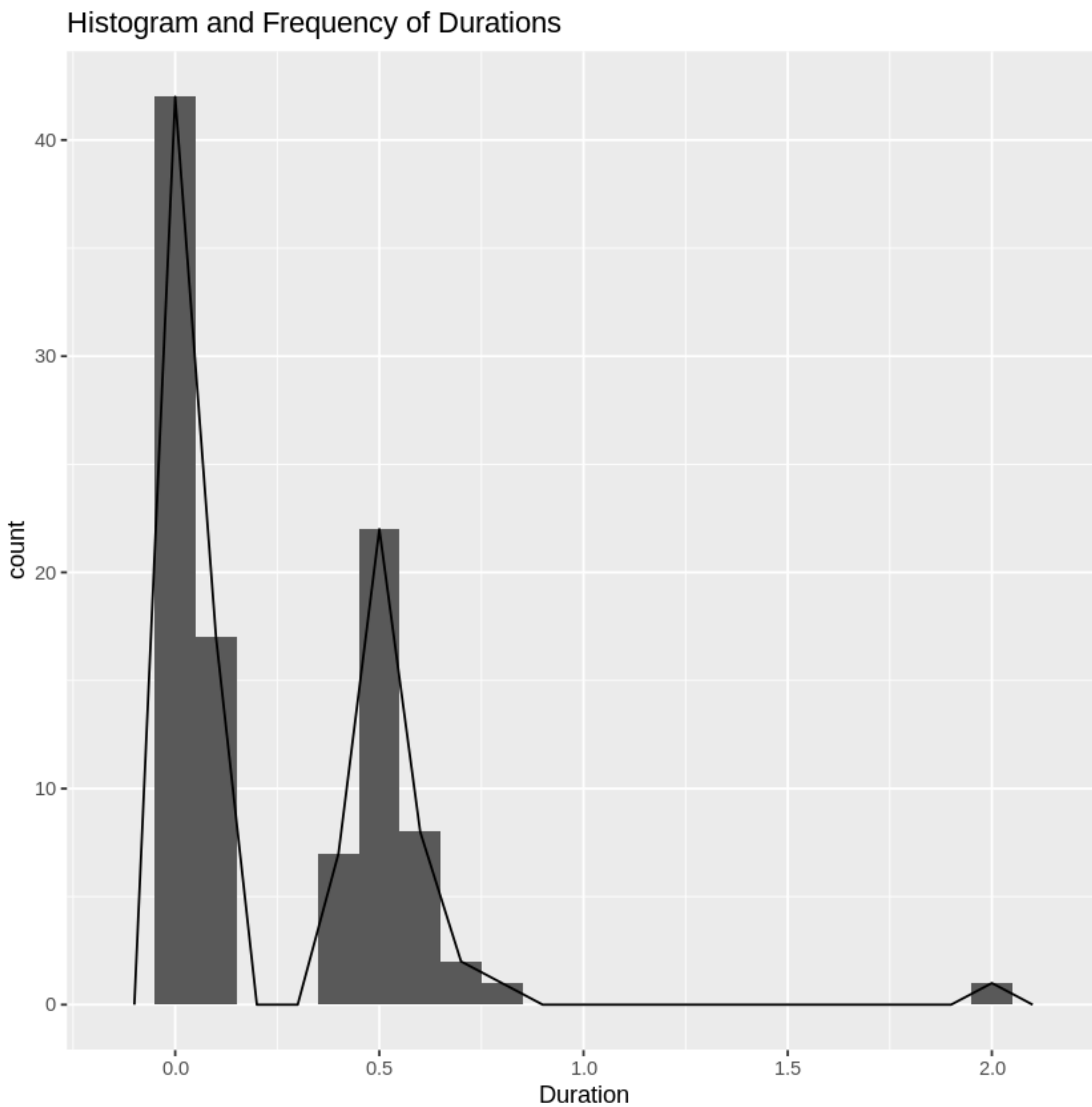
```
spanMetrics %>%
    select(Duration, numContainers, extNetworkHops) %>%
    ggpairs(spanMetrics)
```



```
spanMetrics %>%
    ggplot(aes(Trace.name, Duration)) +
    geom_boxplot() + geom_point() +
    ggtitle("Duration of Endpoint Responses from 'Trace'")
```

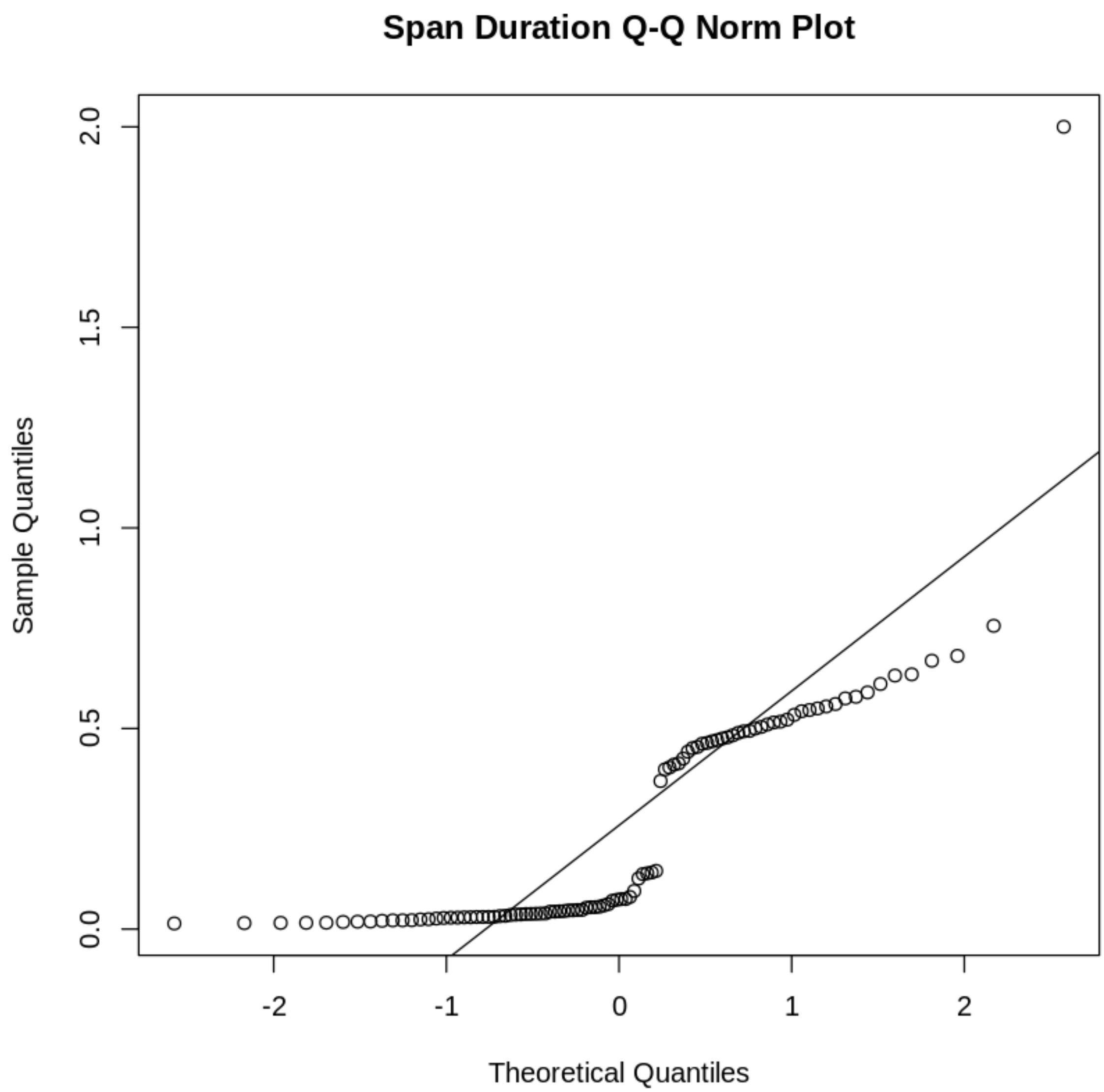


```r
spanMetrics %>%
    ggplot(aes(Duration)) +
    geom_histogram(binwidth = 0.1) +
    geom_freqpoly(binwidth = 0.1) +
    ggtitle("Histogram and Frequency of Durations")
```



Q-Q Normality Test

```
qqnorm(spanMetrics$Duration,main="Span Duration Q-Q Norm Plot")
qqline(spanMetrics$Duration)
```



A transformation is needed to apply statistical analysis.

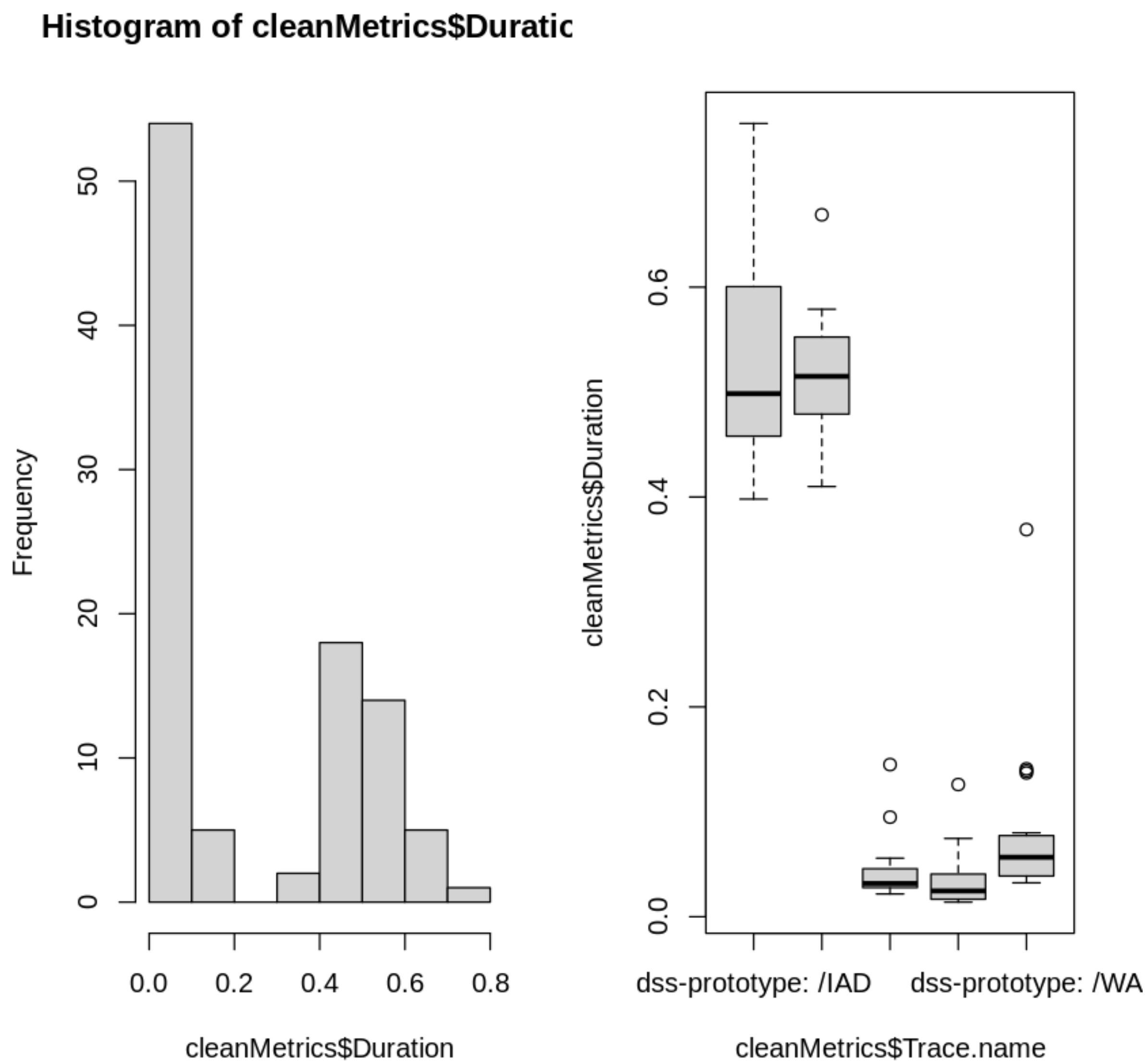
Clean the Data

Search for outliers

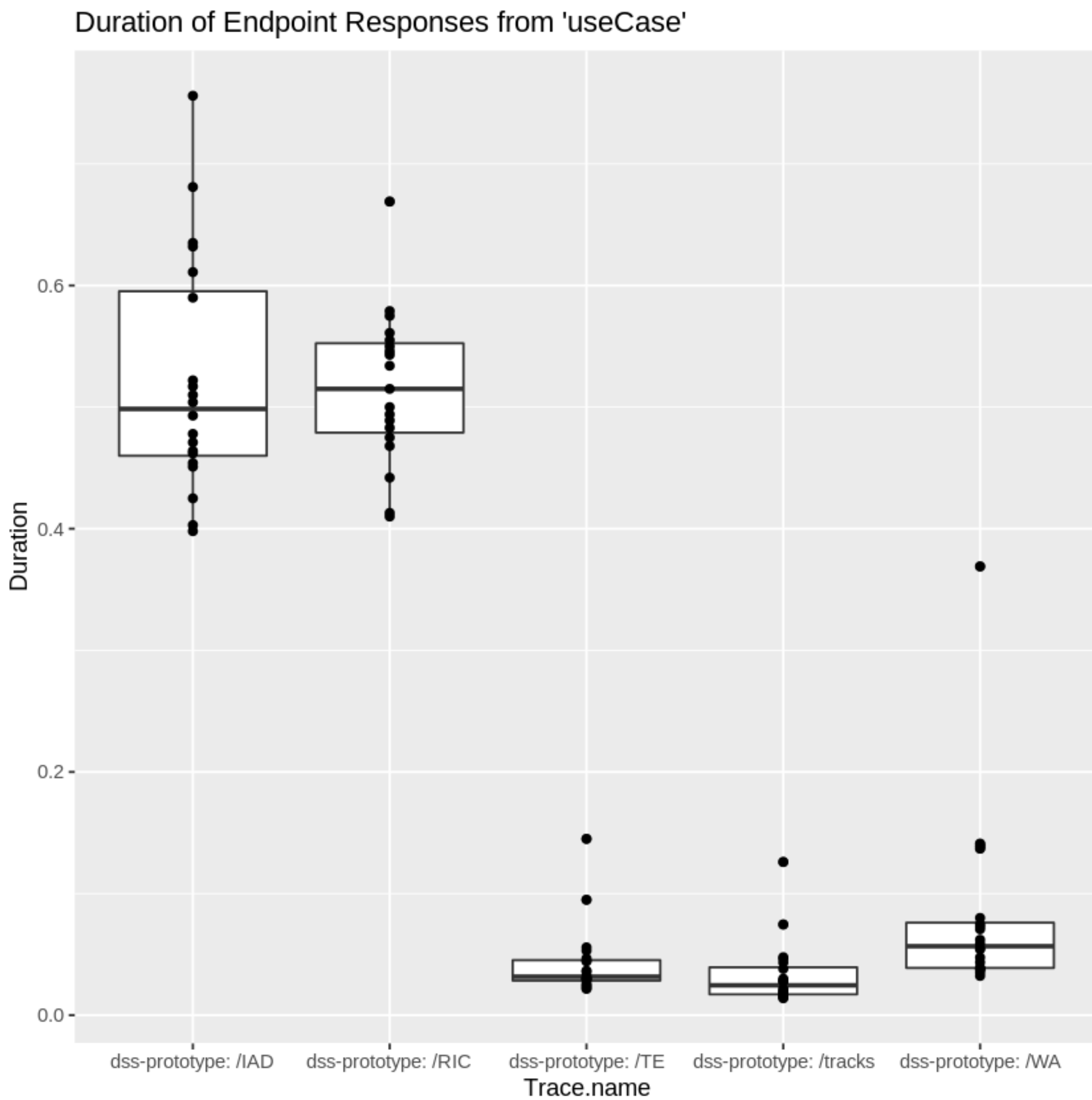
```
# Use this to get the values of the statistical outliers in trk_update_data from R
outliers <- boxplot(spanMetrics$Duration, plot = FALSE)$out
outliers

cleanMetrics <- spanMetrics
cleanMetrics <- cleanMetrics[-which(cleanMetrics$Duration %in% outliers),]
```

2

```
par(mfrow=c(1,2))
hist(cleanMetrics$Duration)
boxplot(cleanMetrics$Duration~cleanMetrics$Trace.name)
```



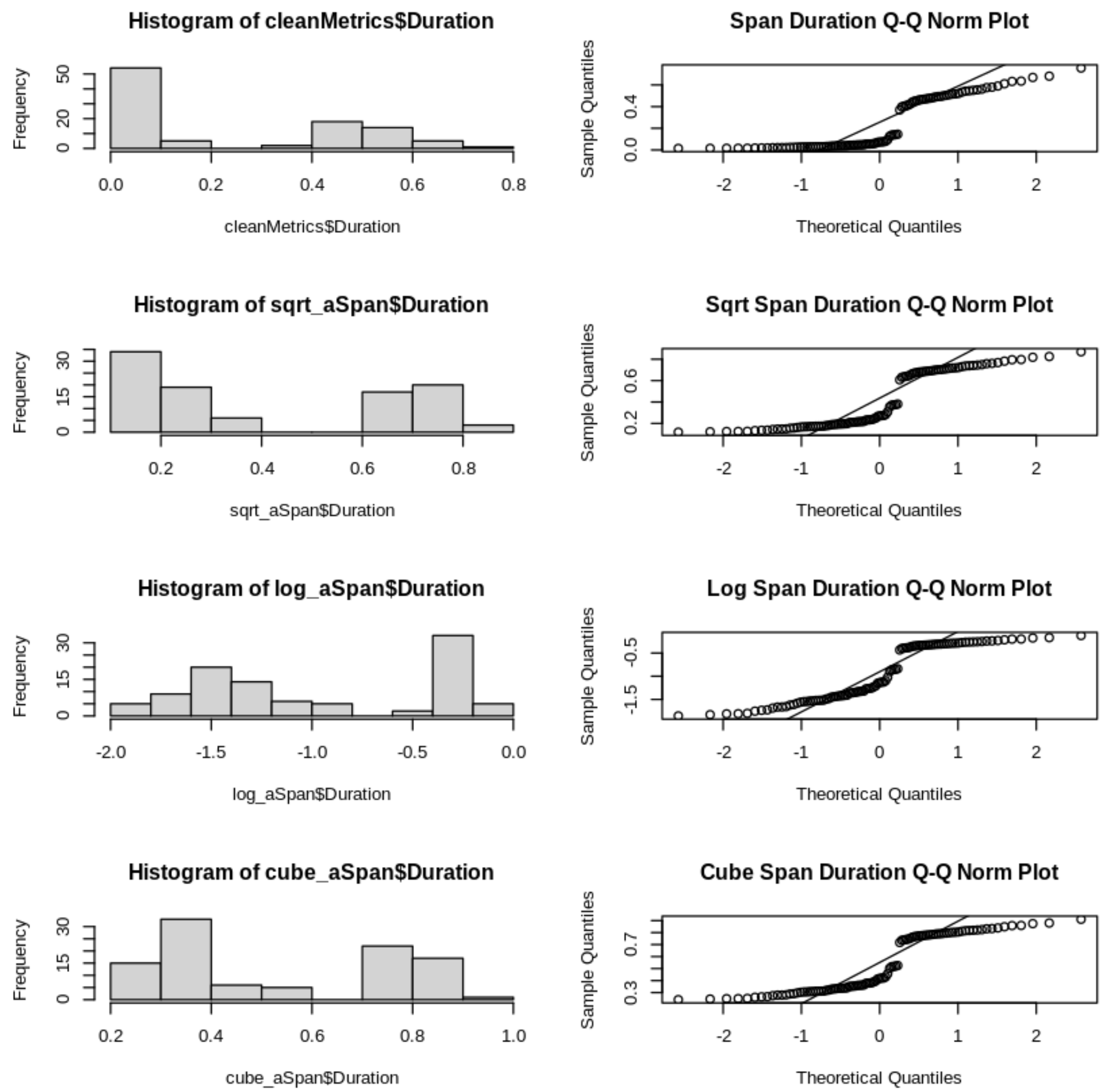

```
cleanMetrics %>%
    ggplot(aes(Trace.name, Duration)) +
    geom_boxplot() + geom_point() +
    ggtitle("Duration of Endpoint Responses from 'useCase'")
```


Duration of Endpoint Responses from 'useCase'
Duration
0.6
0.4
0.2
0.0
dss-prototype: /IAD
dss-prototype: /RIC
dss-prototype: /TE
dss-prototype: /tracks
dss-prototype: /WA
Trace.name

Transformation of Clean Metrics

Sqrt, Log, and Cube Transformations

```r
sqrt_aSpan <- cleanMetrics
sqrt_aSpan$Duration=sqrt(sqrt_aSpan$Duration)
log_aSpan <- cleanMetrics
log_aSpan$Duration=log10(log_aSpan$Duration)
cube_aSpan <- cleanMetrics
cube_aSpan$Duration=cube_aSpan$Duration^(1/3)

par(mfrow=c(4,2))
hist(cleanMetrics$Duration)
qqnorm(cleanMetrics$Duration,main="Span Duration Q-Q Norm Plot")
qqline(cleanMetrics$Duration)
hist(sqrt_aSpan$Duration)
qqnorm(sqrt_aSpan$Duration,main="Sqrt Span Duration Q-Q Norm Plot")
qqline(sqrt_aSpan$Duration)
hist(log_aSpan$Duration)
qqnorm(log_aSpan$Duration,main="Log Span Duration Q-Q Norm Plot")
qqline(log_aSpan$Duration)
hist(cube_aSpan$Duration)
qqnorm(cube_aSpan$Duration,main="Cube Span Duration Q-Q Norm Plot")
qqline(cube_aSpan$Duration)
```



None of these transformation yield distributions that would be considered normal. Most likely due to access to external and internal services with differing latency. Let try another transformation.

Box-Cox Transformation

Box and Cox (1964) developed a family of transformations designed to reduce nonnormality of the errors in a linear model. Applying this transform often reduces non-linearity as well, and heteroscedascity.

The idea is to transform the response variable Y to a replacement response variable $Y_i^{(\lambda)}$, leaving the right-hand side of the regression model unchanged, so that the regression residuals become normally-distributed. Note that the regression coefficients will also change, because the response variable has changed; therefore, the regression coefficients must be interpreted with respect to the transformed variable. Also, any predictions made with the model have to be back-transformed, to be interpreted in the original units.

The standard (simple) Box-Cox transform is:

$$Y_i^{(\lambda)} = \begin{cases} \frac{Y_i^{\lambda}-1}{\lambda}, & (\lambda \neq 0) \\ log(Y_i), & (\lambda = 0) \end{cases}$$

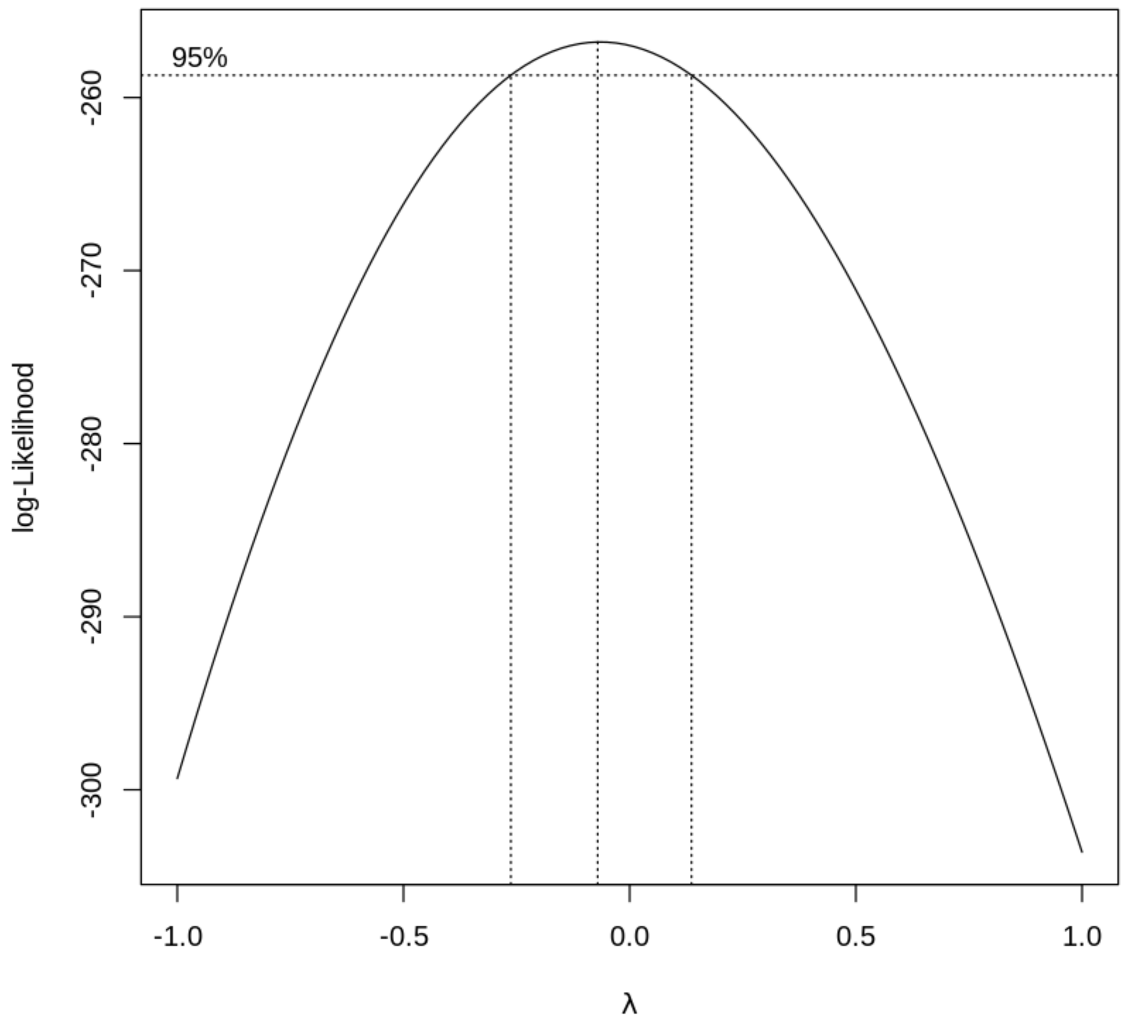
Box, G. E. P., & Cox, D. R. (1964). An Analysis of Transformations. Journal of the Royal Statistical Society, Series B (Metholodogical), 26(2), 211-252.

http://www.css.cornell.edu/faculty/dgr2/_static/files/R_html/Transformations.html

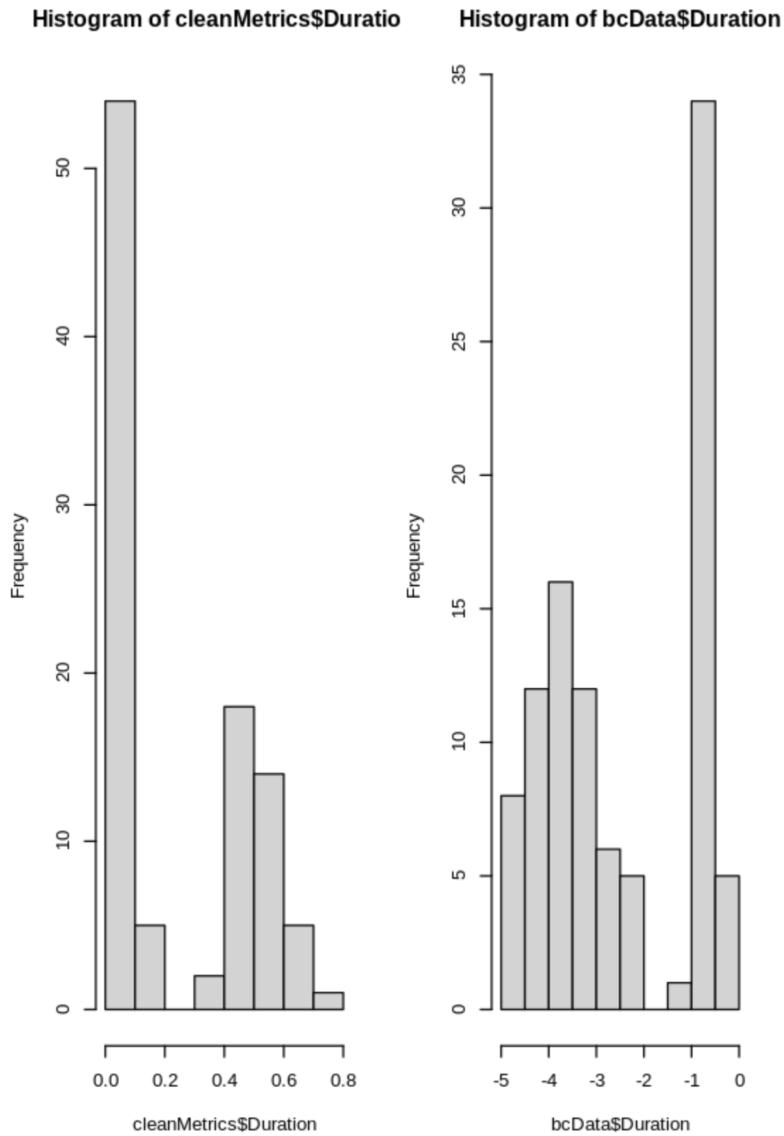
```
library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':

    select
```

```
bcData = cleanMetrics
x <- bcData$Duration
bc = boxcox(lm(x ~ 1), seq(-1,1,.1))
#bc = boxcox(lm(x ~ 1))
lambda <- bc$x[which.max(bc$y)]
new_x_exact <- (x ^ lambda - 1) / lambda
```




```
bcData$Duration = new_x_exact
par(mfrow=c(1,3))
hist(cleanMetrics$Duration)
hist(bcData$Duration)
```



Normality Testing of the Trasformation

Shapiro-Wilk

The null-hypothesis of this test is that the population is normally distributed. Thus, if the p value is less than the chosen alpha level, then the null hypothesis is rejected and there is evidence that the data tested are not normally distributed. On the other hand, if the p value is greater than the chosen alpha level, then the null hypothesis (that the data came from a normally distributed population) can not be rejected (e.g., for an alpha level of .05, a data set with a p value of less than .05 rejects the null hypothesis that the data are from a normally distributed population).

https://en.wikipedia.org/wiki/Shapiro–Wilk_test

```
shapiro.test(bcData$Duration)
```

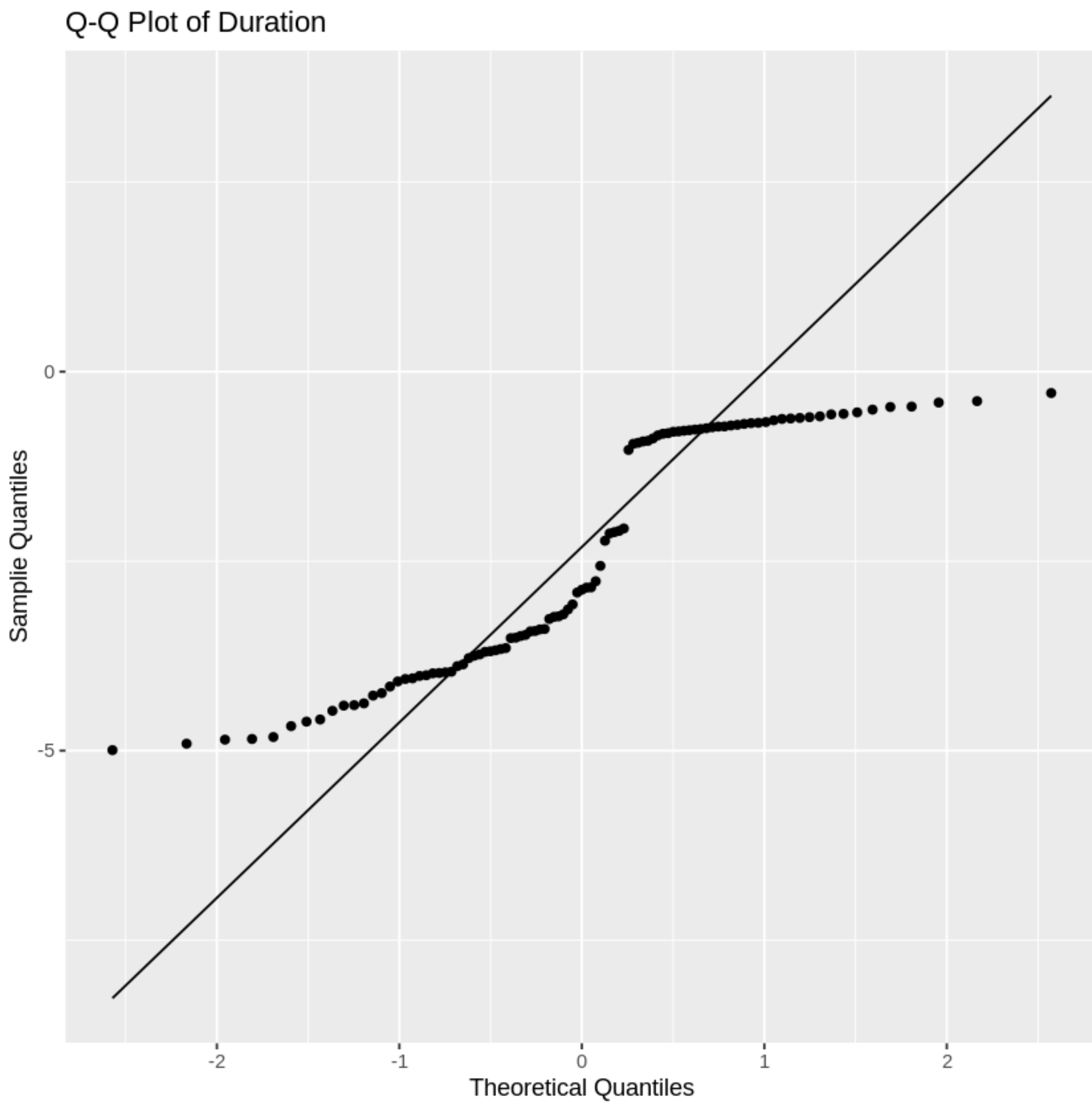
```

	Shapiro-Wilk normality test

data:  bcData$Duration
W = 0.85873, p-value = 2.852e-08
```

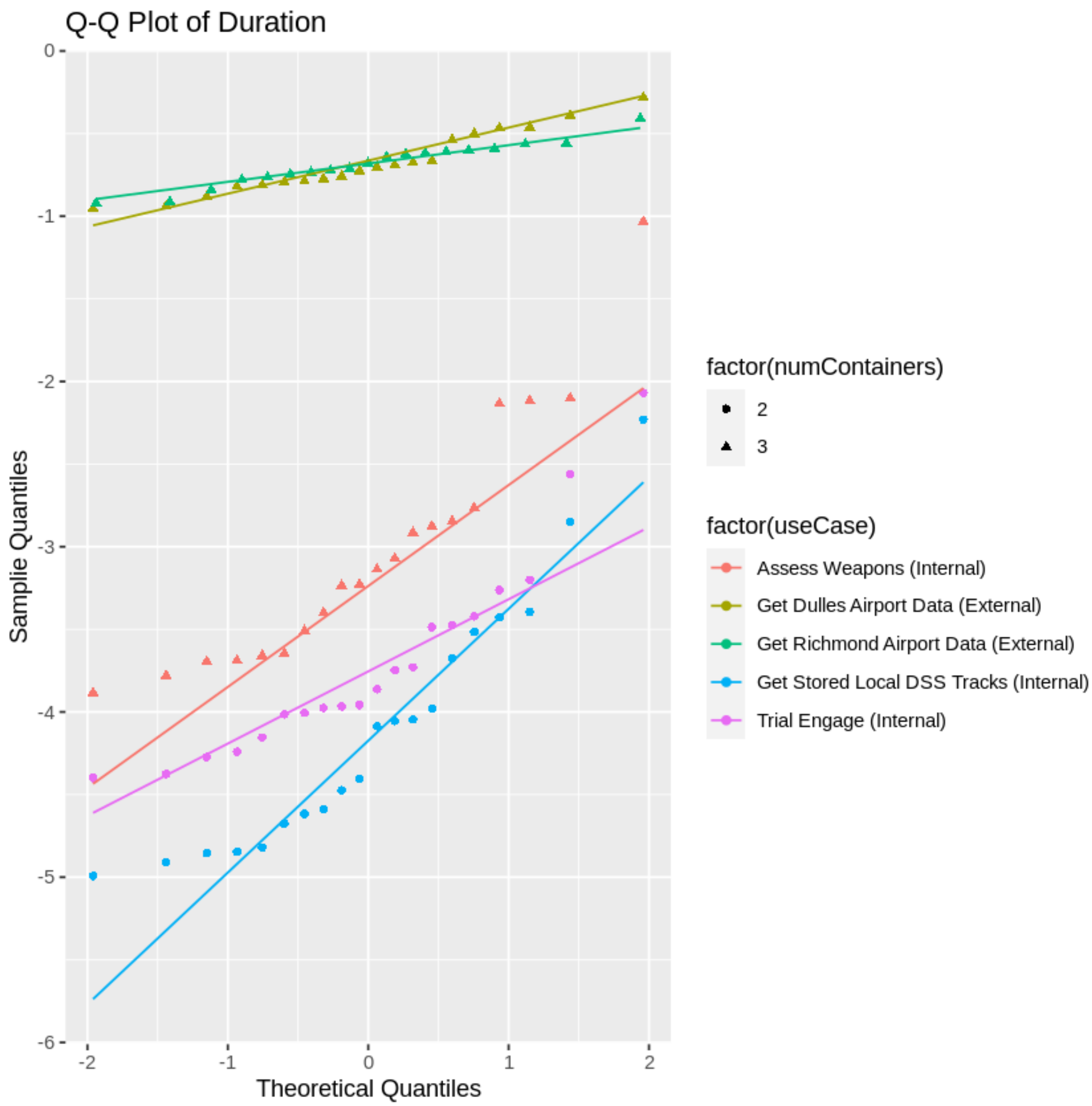
With p-value of 2.852e-08 < 0.05 we reject the null hypothesis that the data are from a normally distributed population. But we'll also do a Q-Q Norm plot to visually see the results.

"if the p value is greater than the chosen alpha level, then the null hypothesis (that the data came from a normally distributed population) can not be rejected"

Q-Q Norm

```
bcData %>%
    ggplot(aes(sample = Duration)) +
    stat_qq() +
    stat_qq_line() +
    labs(title="Q-Q Plot of Duration",
       x = "Theoretical Quantiles", y = "Samplie Quantiles")

bcData %>%
    ggplot(aes(sample = Duration, colour = factor(useCase), shape = factor(numContainers))
    stat_qq() +
    stat_qq_line() +
    labs(title="Q-Q Plot of Duration",
       x = "Theoretical Quantiles", y = "Samplie Quantiles")
```


Q-Q Plot of Duration
Sample Quantiles
0
-5
-2
-1
0
1
2
Theoretical Quantiles

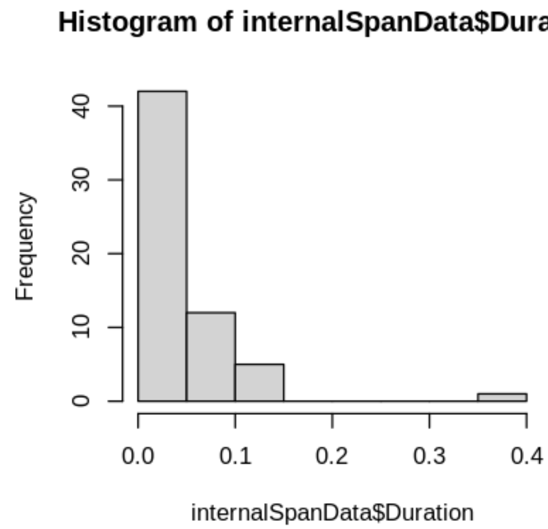


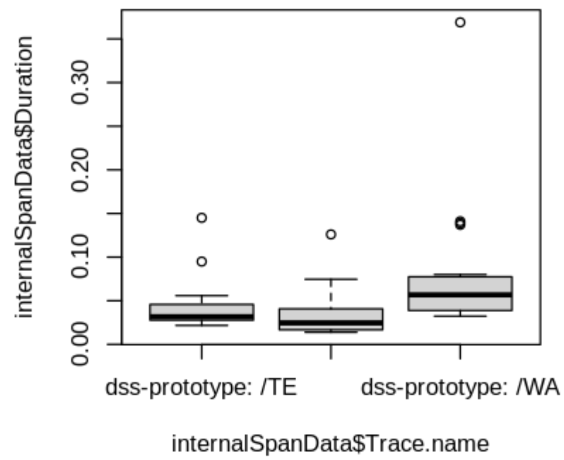
Our assumption here is that the separation of **Sample Quantiles** is from the difference between internal and external span durations (e.g. latency). Let's see what happens when we split the samples.

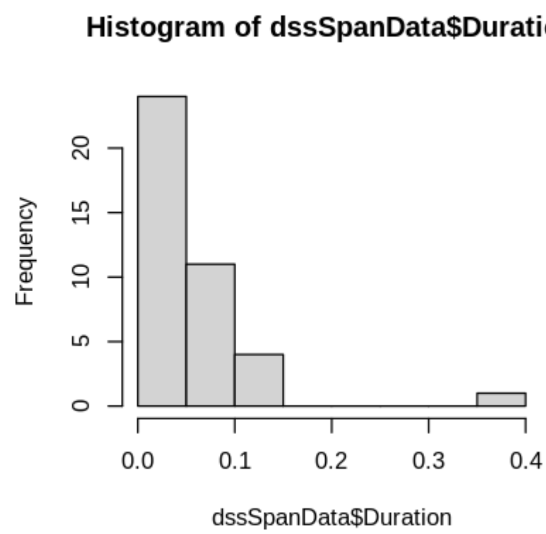
Separating "Original" Internal from External Data

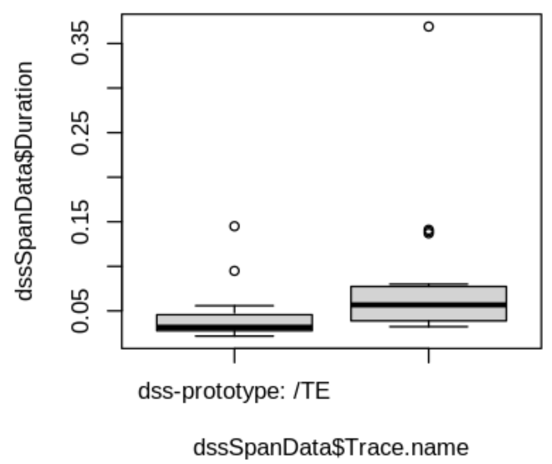
Internal Data

```r
tracksSpanData = subset(spanMetrics, Trace.name == "dss-prototype: /tracks")
TE_SpanData = subset(spanMetrics, Trace.name == "dss-prototype: /TE")
WA_SpanData = subset(spanMetrics, Trace.name == "dss-prototype: /WA")

internalSpanData <- rbind(tracksSpanData, TE_SpanData, WA_SpanData)
dssSpanData <- rbind(TE_SpanData, WA_SpanData)

# head(tracksSpanData)
# head(TE_SpanData)
# head(WA_SpanData)
# head(internalSpanData)

par(mfrow=c(2,2))
hist(internalSpanData$Duration)
boxplot(internalSpanData$Duration~internalSpanData$Trace.name)
hist(dssSpanData$Duration)
boxplot(dssSpanData$Duration~dssSpanData$Trace.name)
```







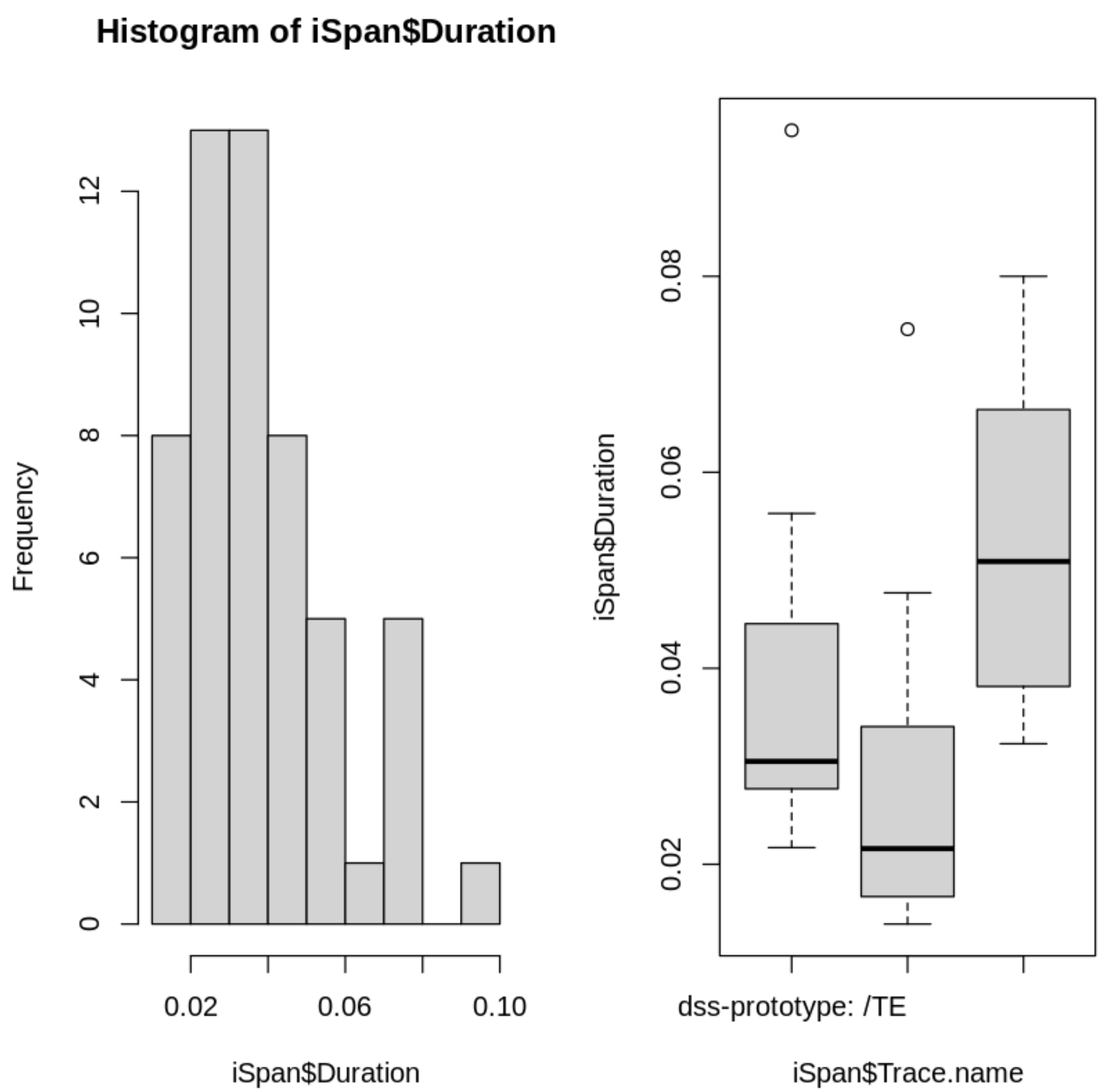


This result looks much better. However, we'll remove internal span outliers.

```
outliers <- boxplot(internalSpanData$Duration, plot = FALSE)$out
outliers

iSpan <- internalSpanData
iSpan <- iSpan[-which(iSpan$Duration %in% outliers),]
```

```r
par(mfrow=c(1,2))
hist(iSpan$Duration)
boxplot(iSpan$Duration~iSpan$Trace.name)
```

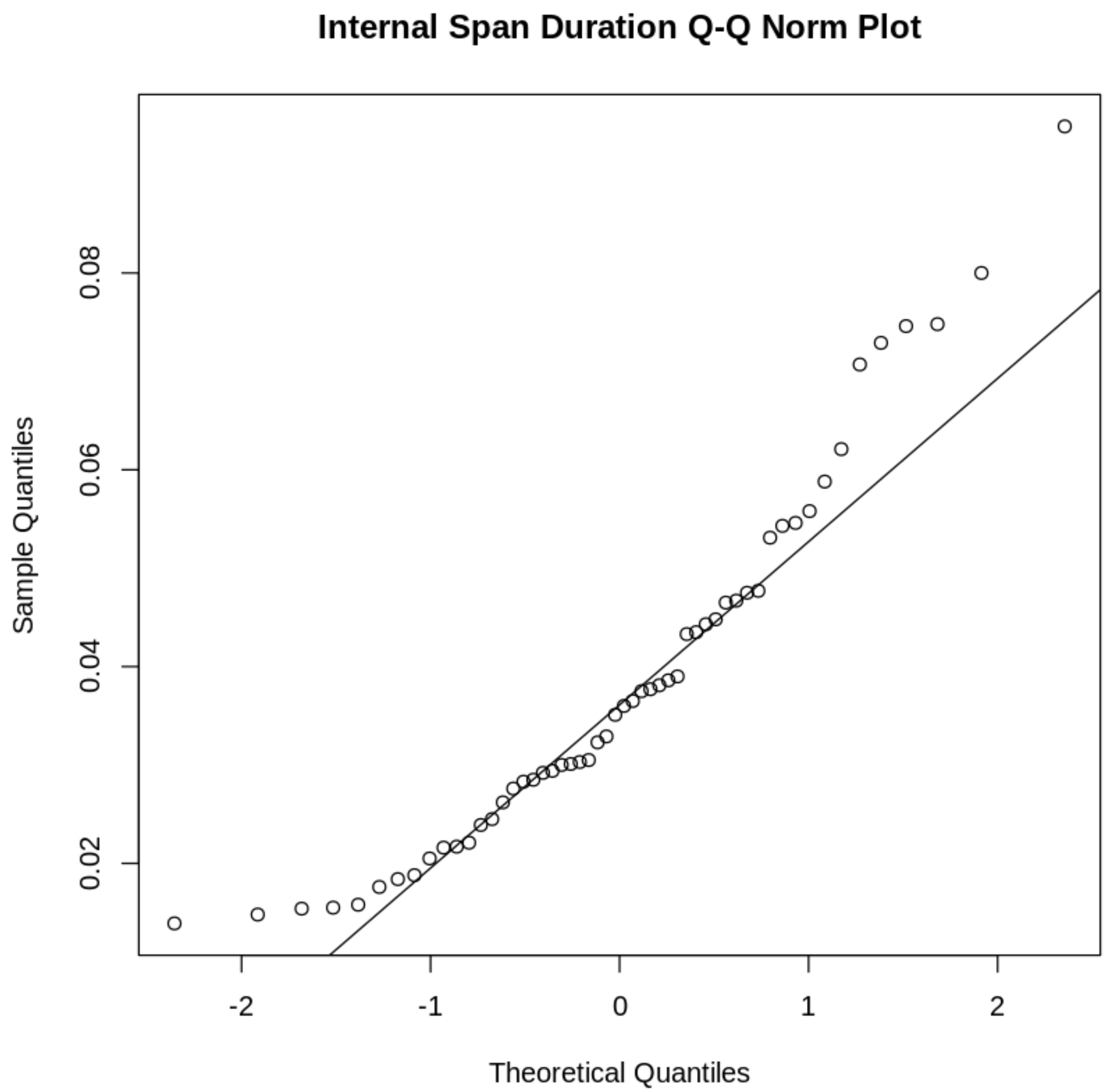
1. 0.126
2. 0.145
3. 0.139
4. 0.369
5. 0.137
6. 0.141



Q-Q Norm Plot of "Clean" Internal Span Data

We'll look a the Q-Q Norm Plot and Shapiro-Wilk Test

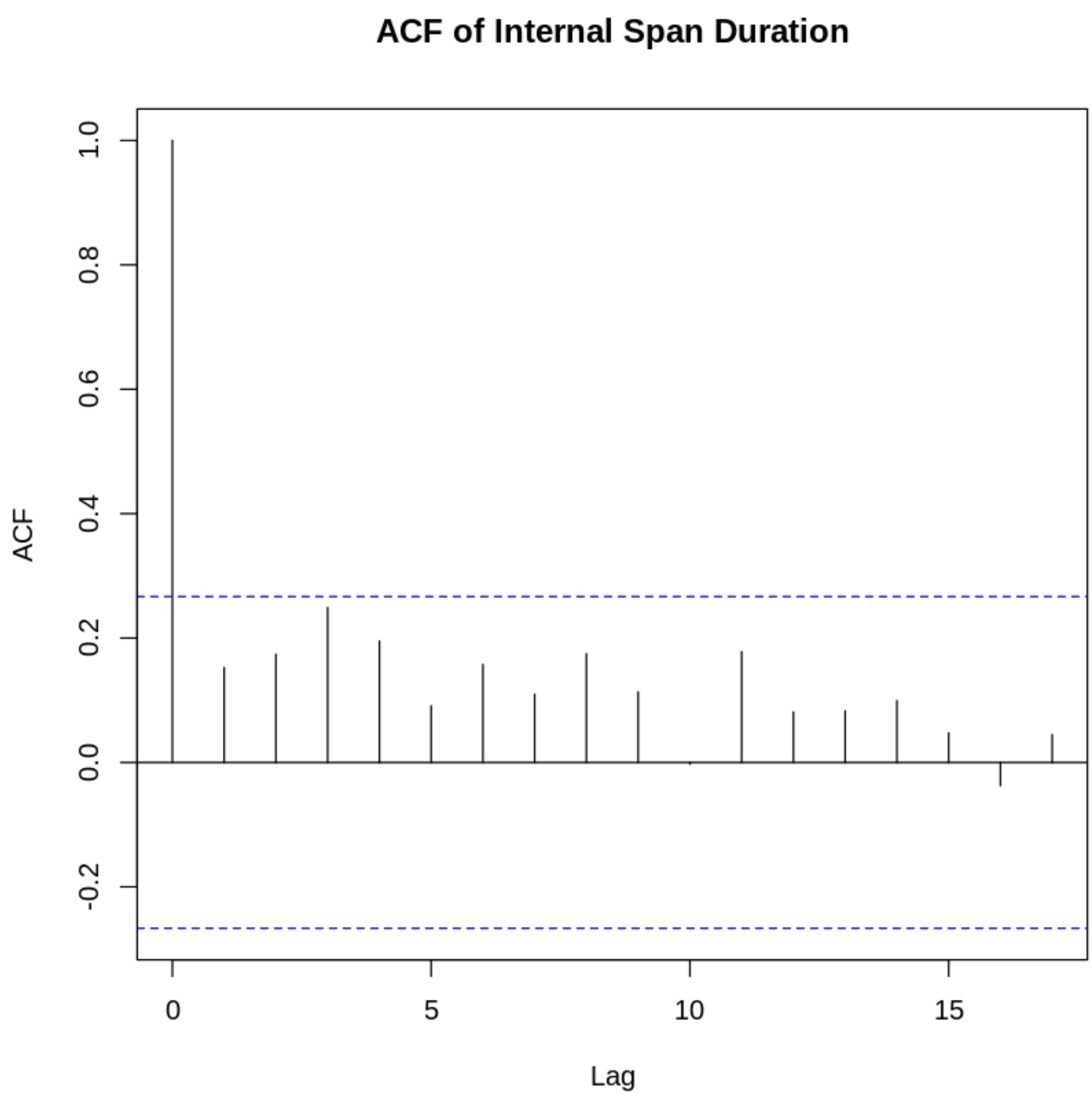
```
qqnorm(iSpan$Duration, main="Internal Span Duration Q-Q Norm Plot")
qqline(iSpan$Duration)
```



Autocorrelation

Autocorrelation plots are a commonly-used tool for checking randomness in a data set. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero.

```r
acf(iSpan$Duration, main="ACF of Internal Span Duration")
```



Shapiro-Wilk Normality Test

```
shapiro.test(iSpan$Duration)
```

```

    Shapiro-Wilk normality test

data:  iSpan$Duration
W = 0.92499, p-value = 0.002321
```

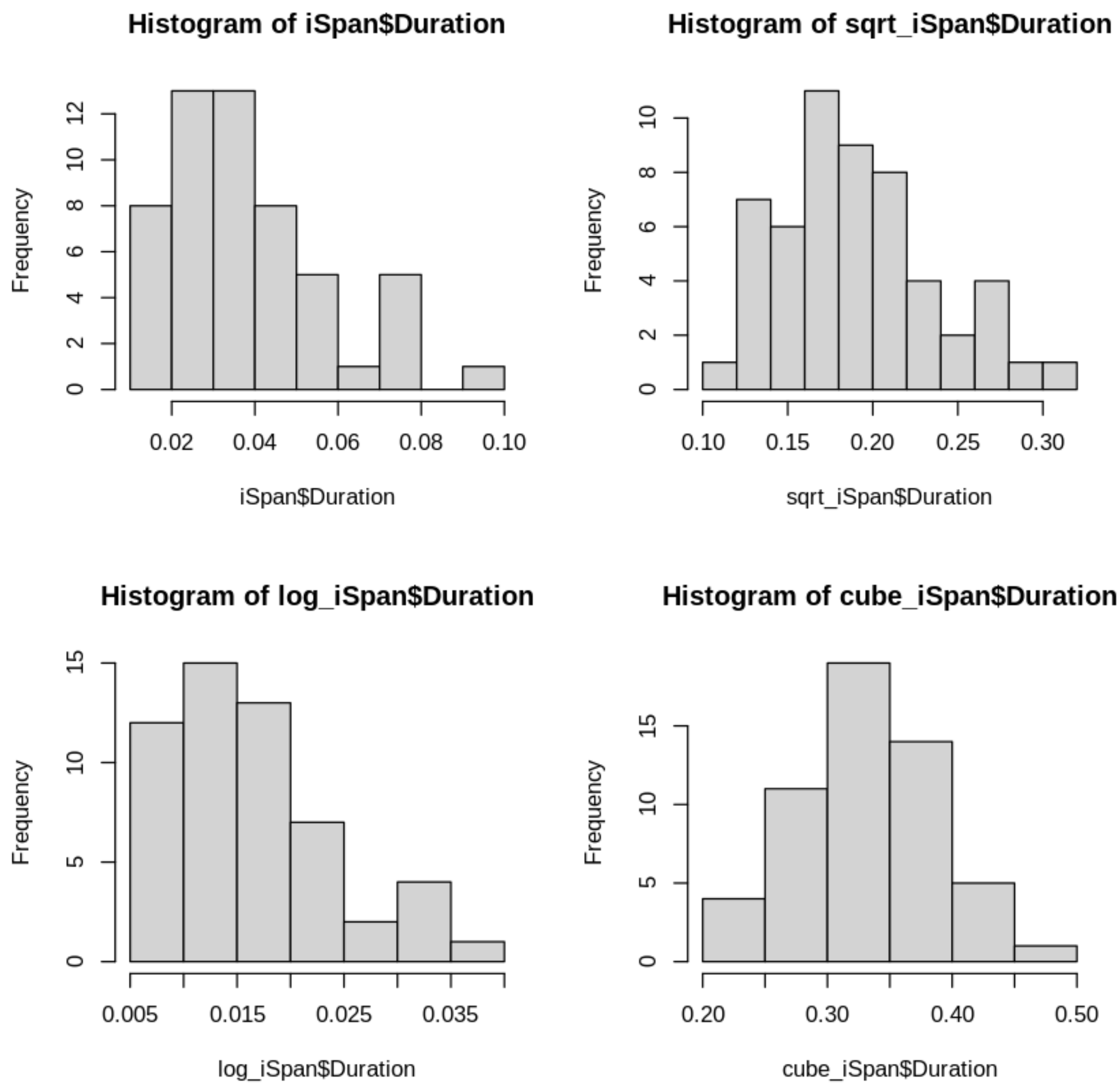
With p-value of $0.002321 < 0.05$ we reject the null hypothesis that the data are from a normally distributed population.

"if the p value is greater than the chosen alpha level, then the null hypothesis (that the data came from a normally distributed population) can not be rejected"

Data Transformations

Sqrt-Log-Cube Transformations

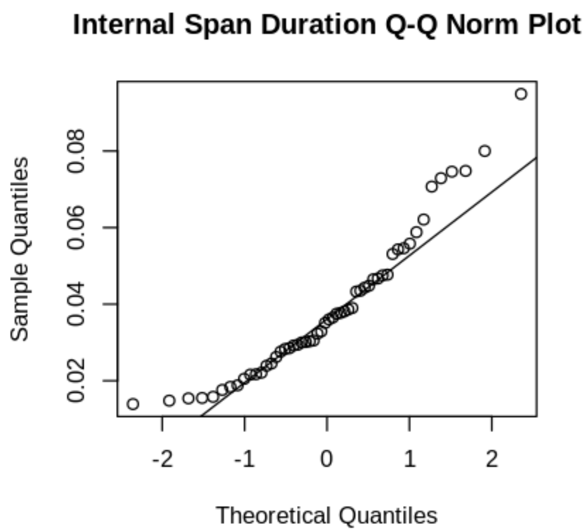
```
sqrt_iSpan <- iSpan
sqrt_iSpan$Duration=sqrt(sqrt_iSpan$Duration)
log_iSpan <- iSpan
log_iSpan$Duration=log10(log_iSpan$Duration + 1)
cube_iSpan <- iSpan
cube_iSpan$Duration=cube_iSpan$Duration^(1/3)

par(mfrow=c(2,2))
hist(iSpan$Duration)
hist(sqrt_iSpan$Duration)
hist(log_iSpan$Duration)
hist(cube_iSpan$Duration)
```

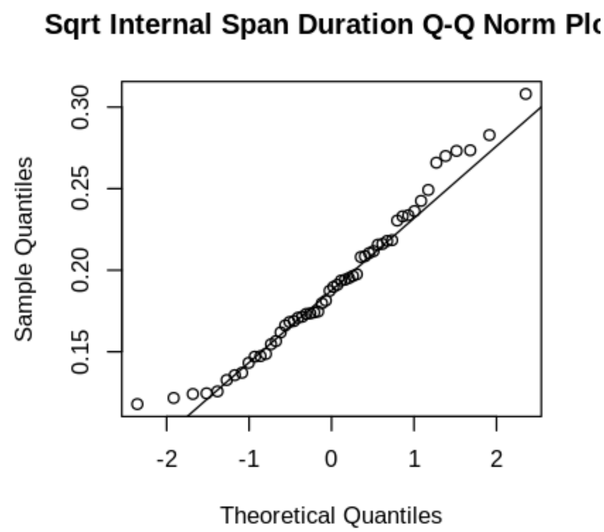


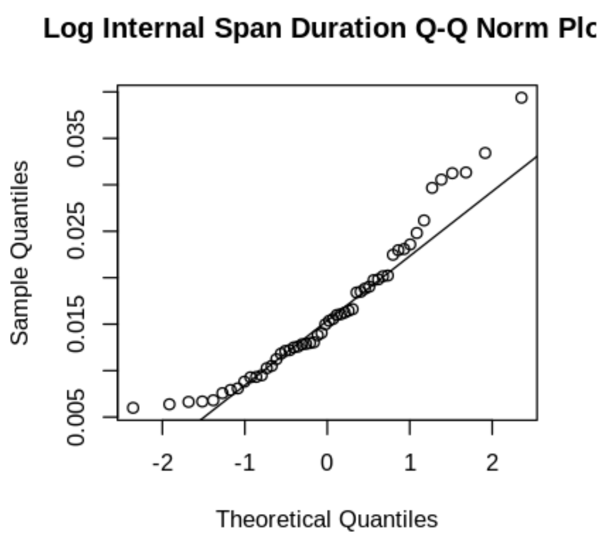
Q-Q Norm Sqrt-Log-Cube

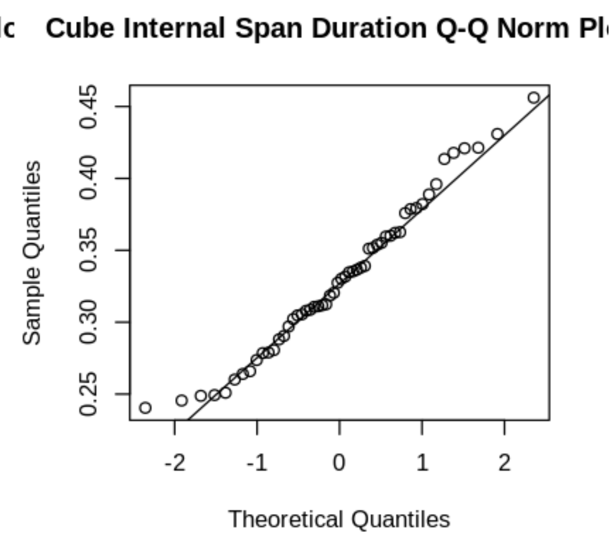
```
par(mfrow=c(2,2))
qqnorm(iSpan$Duration,main="Internal Span Duration Q-Q Norm Plot")
qqline(iSpan$Duration)
qqnorm(sqrt_iSpan$Duration,main="Sqrt Internal Span Duration Q-Q Norm Plot")
qqline(sqrt_iSpan$Duration)
```

```r
qqnorm(log_iSpan$Duration,main="Log Internal Span Duration Q-Q Norm Plot")
qqline(log_iSpan$Duration)
qqnorm(cube_iSpan$Duration,main="Cube Internal Span Duration Q-Q Norm Plot")
qqline(cube_iSpan$Duration)
```









Shapiro-Wilk Testing Sqrt-Log-Cube

```r
shapiro.test(sqrt_iSpan$Duration)
shapiro.test(log_iSpan$Duration)
shapiro.test(cube_iSpan$Duration)
```

```
    Shapiro-Wilk normality test

data:  sqrt_iSpan$Duration
W = 0.9683, p-value = 0.1621


    Shapiro-Wilk normality test

data:  log_iSpan$Duration
W = 0.92922, p-value = 0.003398


    Shapiro-Wilk normality test

data:  cube_iSpan$Duration
W = 0.97633, p-value = 0.3593
```

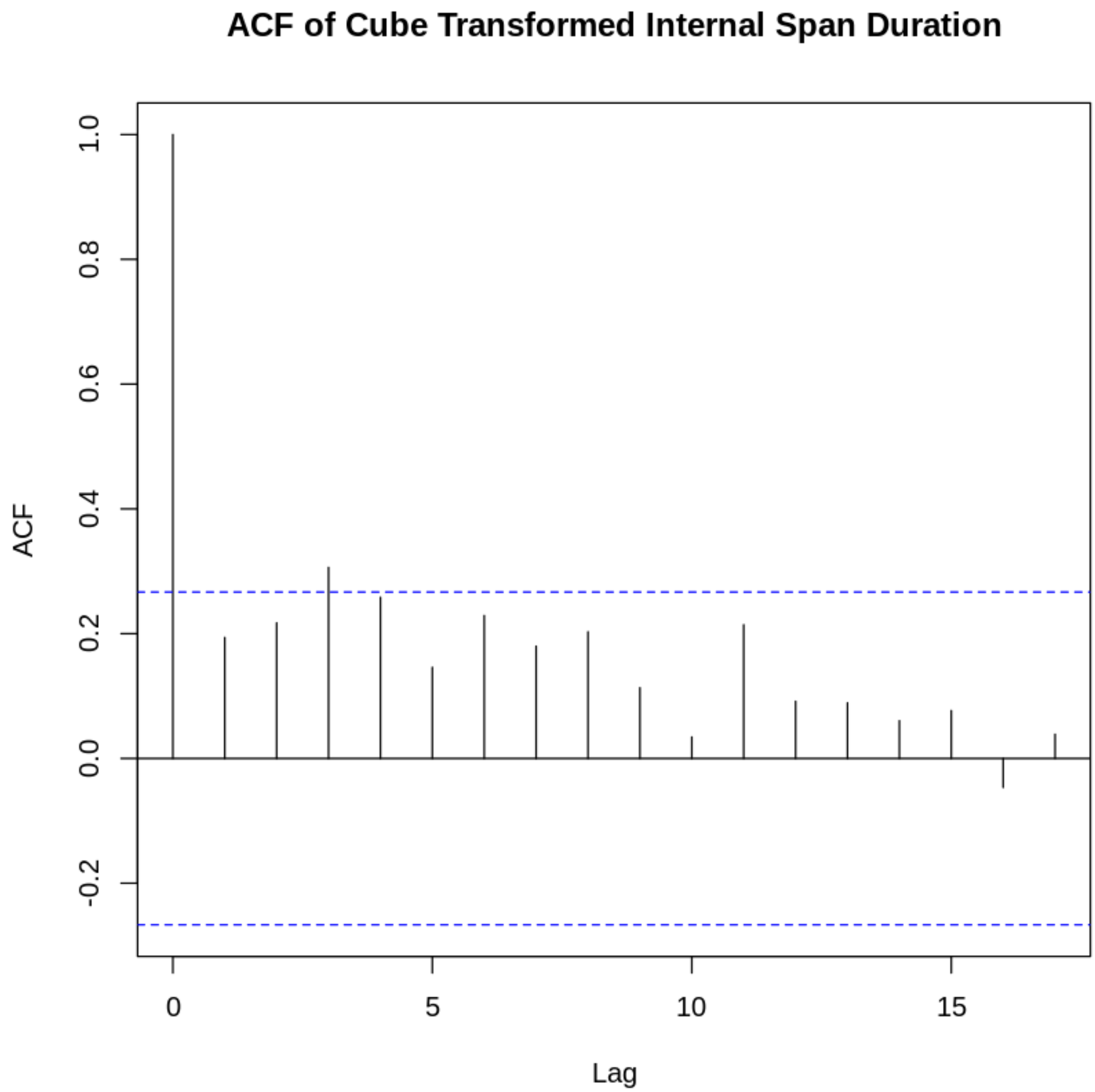
The **cube transformation** seems to provide the best q-q plot fit. With a p-value of 0.3593 > 0.05 we fail to reject the null hypothesis and assume we now have a normal distribution.

"if the p value is greater than the chosen alpha level, then the null hypothesis (that the data came from a normally distributed population) can not be rejected"

Autocorrelation

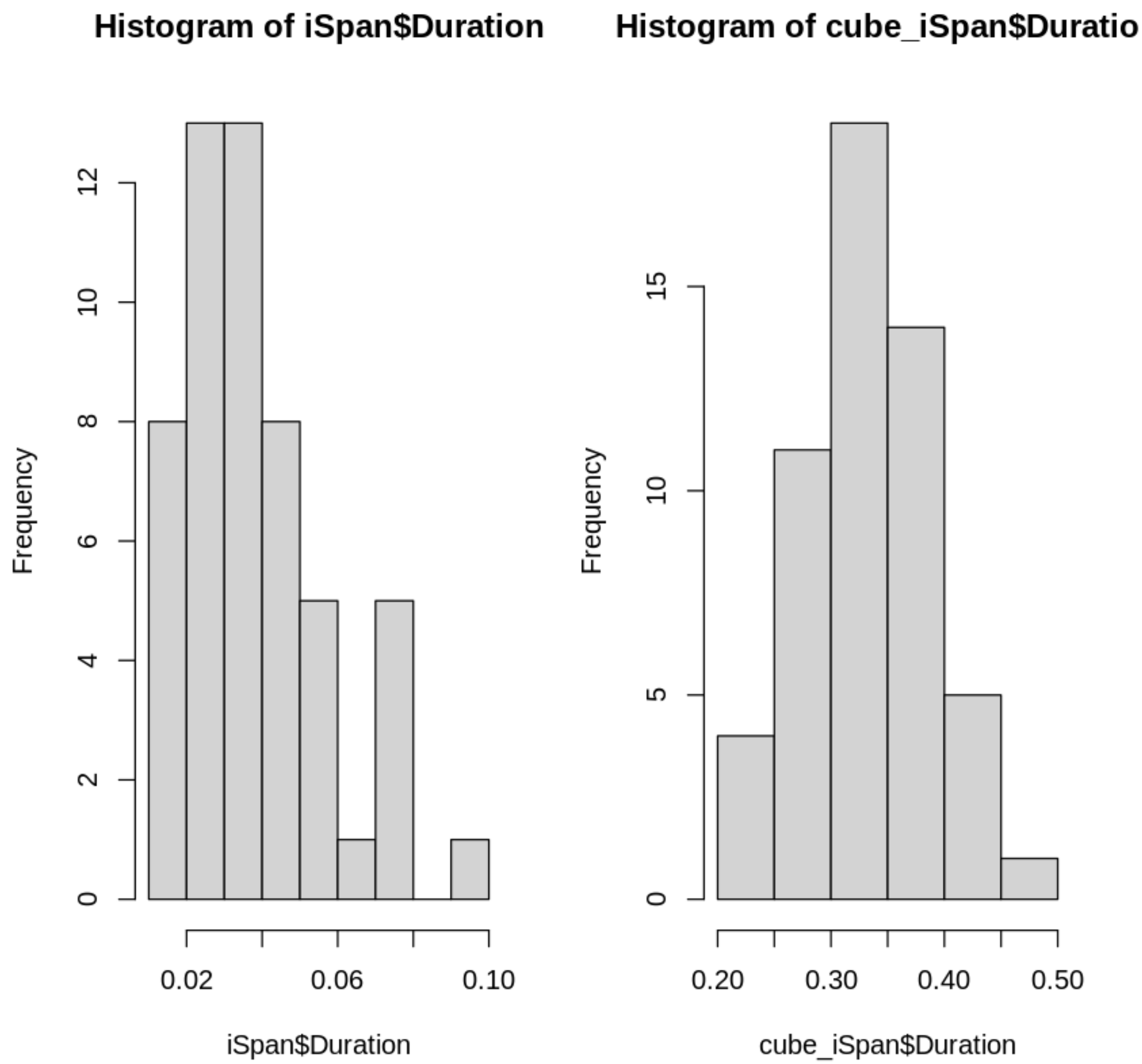
Autocorrelation plots are a commonly-used tool for checking randomness in a data set. This randomness is ascertained by computing autocorrelations for data values at varying time lags. If random, such autocorrelations should be near zero for any and all time-lag separations. If non-random, then one or more of the autocorrelations will be significantly non-zero.

```r
acf(cube_iSpan$Duration, main="ACF of Cube Transformed Internal Span Duration")
```



The ACF indicates that the data is random since the results are near zero.

```
par(mfrow=c(1,2))
hist(iSpan$Duration)
hist(cube_iSpan$Duration)
```



Hypothesis Testing of Transformed Internal Data

We will use a Student's t-Test to test the hypothesis on **normal** internal span data. Our mean is 500 ms (e.g. $\mu = 0.5$ seconds) and our null hypthesis is less than 500 ms.

```
mu = 0.5
x = cube_iSpan$Duration
```

```r
cube_mu = mu^(1/3)
t.test(x=x, mu=cube_mu, alternative = 'greater')
```

```
    One Sample t-test

data:  x
t = -64.323, df = 53, p-value = 1
alternative hypothesis: true mean is greater than 0.7937005
95 percent confidence interval:
 0.3178723       Inf
sample estimates:
mean of x
0.3299424
```

```r
mu = 0.5
x = iSpan$Duration
t.test(x=x, mu=mu, alternative = 'greater')
```

```
    One Sample t-test

data:  x
t = -180.44, df = 53, p-value = 1
alternative hypothesis: true mean is greater than 0.5
95 percent confidence interval:
 0.03440894        Inf
sample estimates:
 mean of x
0.03868889
```

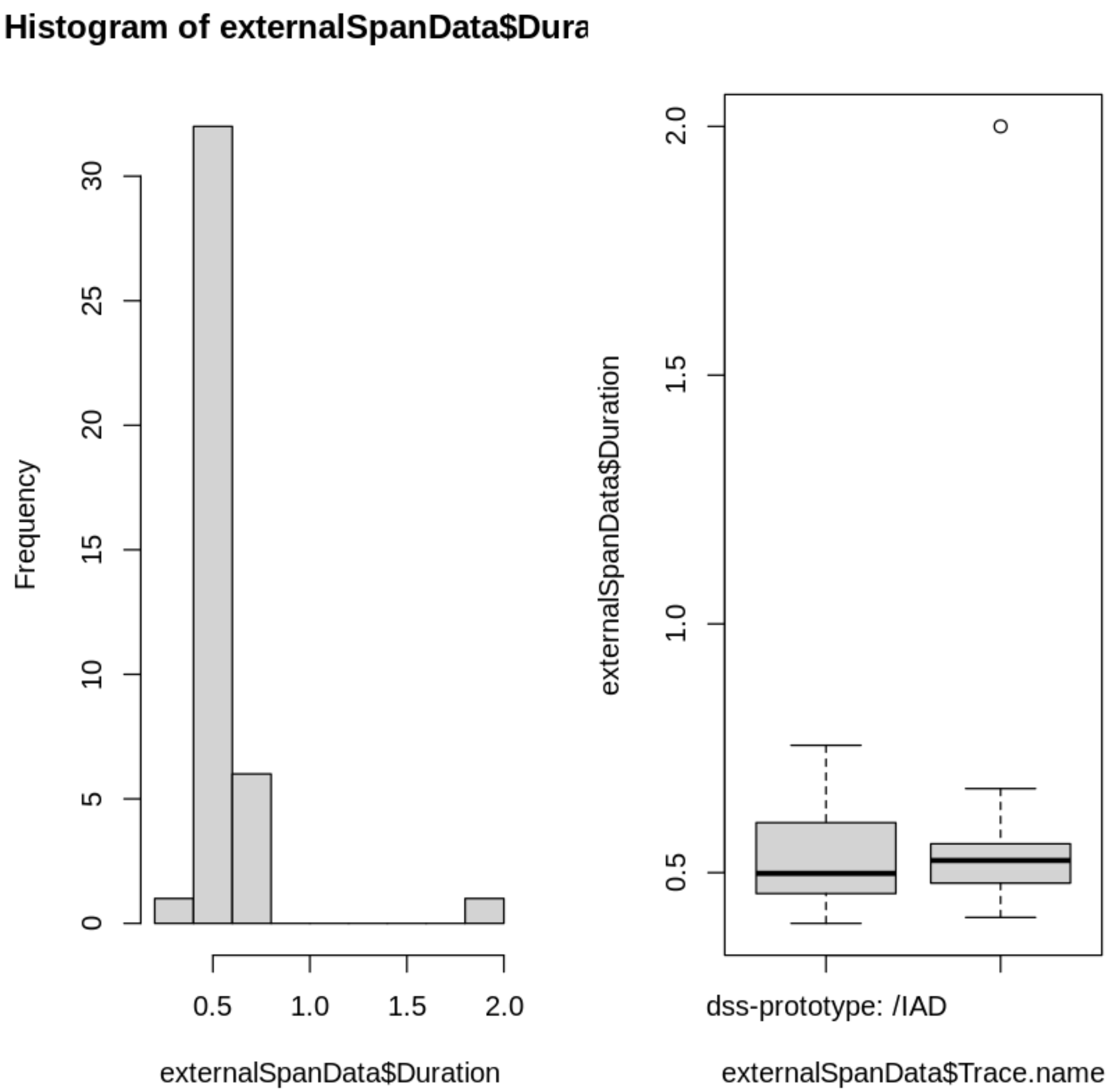
With a original and transformation with a p-value of $1 > 0.05$ we fail to reject the null hypothesis, i.e. we assume that latency will be less than 500 ms.

"If the p value is greater than the chosen alpha level, then the null hypothesis (that latency is < 500 ms) can not be rejected"

External Data

```
RIC_SpanData = subset(spanMetrics, Trace.name == "dss-prototype: /RIC")
IAD_SpanData = subset(spanMetrics, Trace.name == "dss-prototype: /IAD")

externalSpanData <- rbind(RIC_SpanData, IAD_SpanData)

# head(RIC_SpanData)
# head(IAD_SpanData)

summary(externalSpanData)

par(mfrow=c(1,2))
hist(externalSpanData$Duration)
boxplot(externalSpanData$Duration~externalSpanData$Trace.name)
```

```
   Trace.ID           Trace.name           Start.time           Duration
 Length:40          Length:40          Min.   :1.651e+09   Min.   :0.3980
 Class :character   Class :character   1st Qu.:1.651e+09   1st Qu.:0.4670
 Mode  :character   Mode  :character   Median :1.651e+09   Median :0.5070
                                       Mean   :1.651e+09   Mean   :0.5565
                                       3rd Qu.:1.651e+09   3rd Qu.:0.5645
                                       Max.   :1.651e+09   Max.   :2.0000
   useCase          numContainers extNetworkHops
 Length:40          Min.   :3     Min.   :14
 Class :character   1st Qu.:3     1st Qu.:14
 Mode  :character   Median :3     Median :14
                    Mean   :3     Mean   :14
                    3rd Qu.:3     3rd Qu.:14
                    Max.   :3     Max.   :14
```

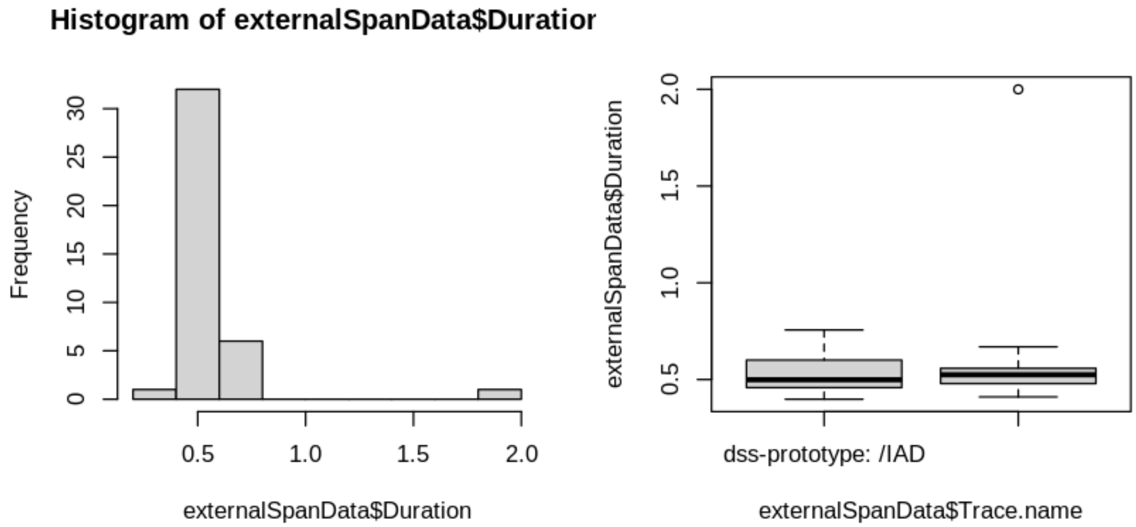


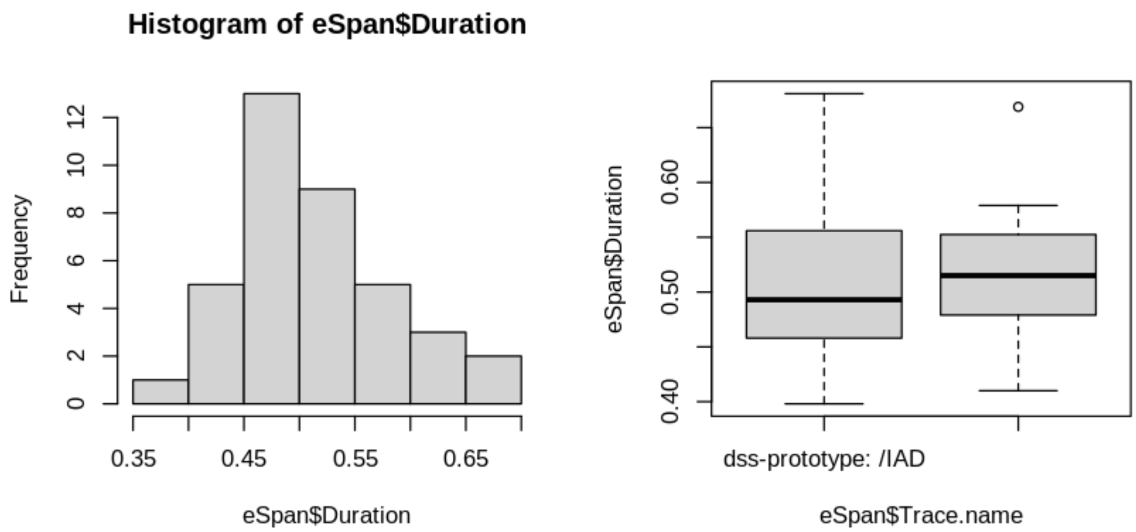

```
outliers <- boxplot(externalSpanData$Duration, plot = FALSE)$out
outliers

eSpan <- externalSpanData
eSpan <- eSpan[-which(eSpan$Duration %in% outliers),]

summary(eSpan)
```

```r
par(mfrow=c(2,2))
hist(externalSpanData$Duration)
boxplot(externalSpanData$Duration~externalSpanData$Trace.name)
hist(eSpan$Duration)
boxplot(eSpan$Duration~eSpan$Trace.name)
```

1. 2
2. 0.756

```
  Trace.ID           Trace.name          Start.time           Duration     
 Length:38          Length:38          Min.   :1.651e+09   Min.   :0.3980  
 Class :character   Class :character   1st Qu.:1.651e+09   1st Qu.:0.4650  
 Mode  :character   Mode  :character   Median :1.651e+09   Median :0.5020  
                                       Mean   :1.651e+09   Mean   :0.5132  
                                       3rd Qu.:1.651e+09   3rd Qu.:0.5537  
                                       Max.   :1.651e+09   Max.   :0.6810  
  useCase          numContainers extNetworkHops
 Length:38          Min.   :3     Min.   :14    
 Class :character   1st Qu.:3     1st Qu.:14    
 Mode  :character   Median :3     Median :14    
                    Mean   :3     Mean   :14    
                    3rd Qu.:3     3rd Qu.:14    
                    Max.   :3     Max.   :14    
```

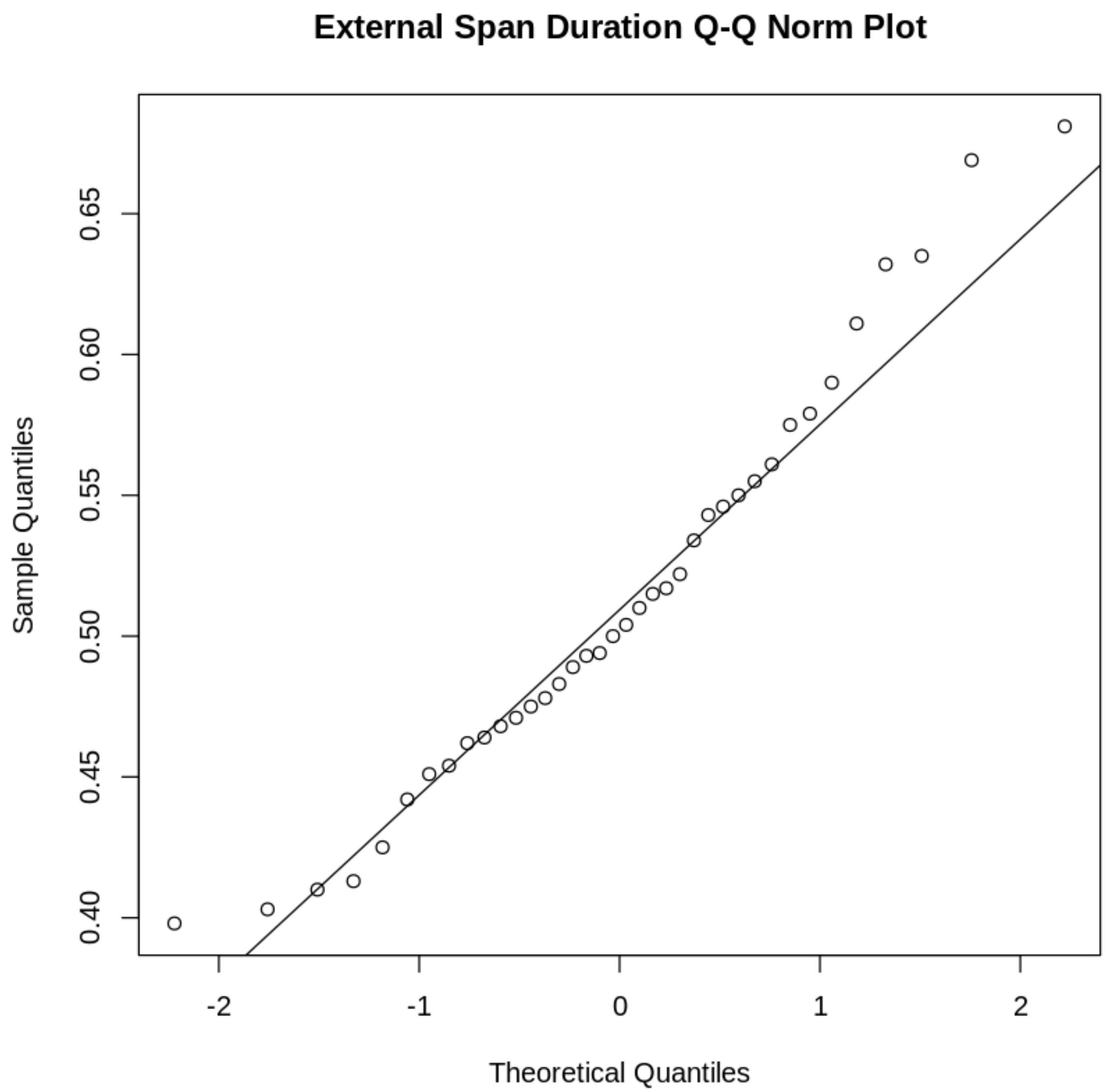




Q-Q Norm Plot of "Clean" External Span Data

We'll look a the Q-Q Norm Plot and Shapiro-Wilk Test

```
qqnorm(eSpan$Duration, main="External Span Duration Q-Q Norm Plot")
qqline(eSpan$Duration)
```



Shapiro-Wilk Normality Test

```
shapiro.test(eSpan$Duration)
```

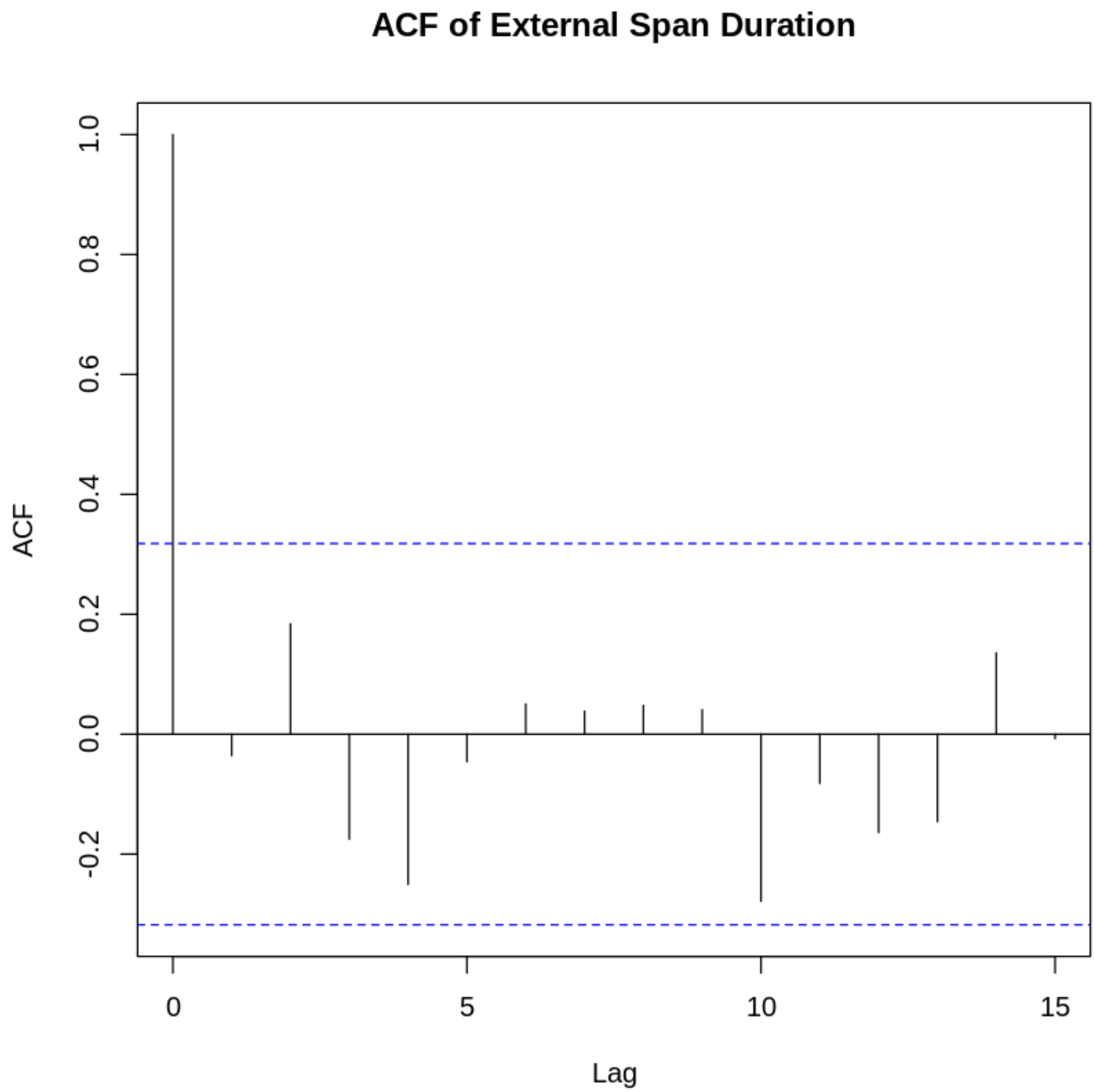
```
	Shapiro-Wilk normality test
```

```
data:  eSpan$Duration
W = 0.96564, p-value = 0.2878
```

With a p-value of $0.2878 > 0.05$ we fail to reject the null hypothesis, i.e. we assume that we have a normal distribution.

Autocorrelation

```r
acf(eSpan$Duration, main="ACF of External Span Duration")
```



The ACF indicates that the data is random since the results are near zero.

Hypothesis Testing

We will use a Student's t-Test to test the hypothesis on external span data. Our mean is 500 ms (e.g. $\mu = 0.5$ seconds) and our null hypthesis is less than 500 ms.

```r
mu = 0.5
```

```r
x = eSpan$Duration
t.test(x=x, mu=mu, alternative = 'greater')
```

```
	One Sample t-test

data:  x
t = 1.1267, df = 37, p-value = 0.1336
alternative hypothesis: true mean is greater than 0.5
95 percent confidence interval:
 0.4934287       Inf
sample estimates:
mean of x 
0.5132105 
```

With a p-value of $0.1336 > 0.05$ we fail to reject the null hypothesis, i.e. we assume that 500 ms can be maintained for external service requests.

"If the p value is greater than the chosen alpha level, then the null hypothesis (that latency is < 500 ms) can not be rejected"

Observations

General Discussion of Normality

It was required to separate external data from internal to establish normality of the data samples. The internal data set required transformation to establish normality, while the external data did not require a transformation.

Hypothesis Results

Hypothesis testing using the Student's t-Test indicates that latency constraints of 500 ms can be maintained internally and external. However, serveral external samples were greater than 500 ms. This is most likely due to the non-deterministic nature of internet (e.g. http) requests. Within the internal environment, data is directly routed between microservices within the Docker environment within a private network. The data shows that a container based microservice architecture can meet the requirement; however, care must be taken to manage processing per container that may increase container response times.