Anhang 2

Graphische Darstellung der VB (und Pi) Messungen

```
setwd("/home/lisa/Darmstadt/05_Speicher und Datennetze IoT/Praktikum/Git/mqtt-qos-rountrip/R_Analysis/0
options(digits.secs=3) # needs to be set from time to time - otherwise R doesn't allow for ms
library("data.table", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.4")
library("h2o", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.4")
library("tidyr", lib.loc="~/R/x86_64-pc-linux-gnu-library/3.4")
library("plyr")
library(knitr)
library(kableExtra)

load("./latenzVB.Rda")
load("./logs_pi.Rda")
```

Übersicht und notwendige Anpassung der Messungen die mit dem Pi erzeugt wurden.

```
logsPi[,4]<-NA # Die Spalten Speed und Min sind bei diesem Datensatz nicht mit sinnvollen Werten belegt
logsPi[,5]<-NA # Um das gleiche Format wie die TC Datensätze zu behalten, werden die Werte nur durch NA
logsPi$Byte<-logsPi$Size
logsPi$Byte[logsPi$Byte == "1Byte"] <- 1
logsPi$Byte[logsPi$Byte == "10Byte"] <- 10
logsPi$Byte[logsPi$Byte == "100Byte"] <- 100
logsPi$Byte[logsPi$Byte == "1KByte"] <- 1000
logsPi$Byte[logsPi$Byte == "1500Byte"] <- 1500
logsPi$Byte[logsPi$Byte == "10KByte"] <- 10000
logsPi$Byte[logsPi$Byte == "100KByte"] <- 100000
logsPi$Byte[logsPi$Byte == "500KByte"] <- 500000
logsPi$Byte[logsPi$Byte == "1MByte"] <- 1000000
logsPiSum <- summary(logsPi)
logsPiSum
#>       sent                          QoS                Size
#>  Min.   :2018-05-18 13:42:40.51   Length:16994       Length:16994
#>  1st Qu.:2018-05-18 13:47:51.32   Class :character   Class :character
#>  Median :2018-05-18 13:52:01.95   Mode  :character   Mode  :character
#>  Mean   :2018-05-18 13:52:50.32
#>  3rd Qu.:2018-05-18 13:55:47.97
#>  Max.   :2018-05-18 16:40:37.01
#>    Min            Speed              rec
#>  Mode:logical   Mode:logical   Min.   :2018-05-18 13:42:40.51
#>  NA's:16994     NA's:16994     1st Qu.:2018-05-18 13:48:08.60
#>                                Median :2018-05-18 13:52:02.83
#>                                Mean   :2018-05-18 13:52:52.33
#>                                3rd Qu.:2018-05-18 13:55:48.18
#>                                Max.   :2018-05-18 16:40:42.42
#>    r_newid               rtt                 id             Byte
#>  Length:16994       Min.   : 0.0030   Min.   :  1.00   Length:16994
#>  Class :character   1st Qu.: 0.1640   1st Qu.: 90.25   Class :character
#>  Mode  :character   Median : 0.2310   Median :332.00   Mode  :character
#>                     Mean   : 1.7474   Mean   :389.04
#>                     3rd Qu.: 0.6997   3rd Qu.:660.00
#>                     Max.   :59.2090   Max.   :999.00
```

Übersicht und notwendige Anpassung der Messungen die mit dem Laptop erzeugt wurden.

```
latenzVB[,5]<-NA
rtt_zero<-latenzVB[latenzVB$rtt == 0,]

latenzVB<-latenzVB[latenzVB$rtt > 0,]

latenzVB$Byte<-latenzVB$Size
latenzVB$Byte[latenzVB$Byte == "1Byte"] <- 1
latenzVB$Byte[latenzVB$Byte == "10Byte"] <- 10
latenzVB$Byte[latenzVB$Byte == "100Byte"] <- 100
latenzVB$Byte[latenzVB$Byte == "1KByte"] <- 1000
latenzVB$Byte[latenzVB$Byte == "1500Byte"] <- 1500
latenzVB$Byte[latenzVB$Byte == "10KByte"] <- 10000
latenzVB$Byte[latenzVB$Byte == "100KByte"] <- 100000
latenzVB$Byte[latenzVB$Byte == "500KByte"] <- 500000
latenzVB$Byte[latenzVB$Byte == "1MByte"] <- 1000000
latenzVB$Byte[latenzVB$Byte == "10MByte"] <- 10000000
latenzVBSum <- summary(latenzVB)
latenzVBSum
#>       sent                          QoS                Size
#>  Min.   :2018-05-23 22:09:04.22   Length:262759      Length:262759
#>  1st Qu.:2018-05-23 22:18:38.74   Class :character   Class :character
#>  Median :2018-05-23 22:28:09.52   Mode  :character   Mode  :character
#>  Mean   :2018-05-23 22:31:06.81
#>  3rd Qu.:2018-05-23 22:42:37.84
#>  Max.   :2018-05-23 23:25:12.82
#>      Min              Speed               rec
#>  Length:262759      Mode:logical   Min.   :2018-05-23 22:09:04.22
#>  Class :character   NA's:262759    1st Qu.:2018-05-23 22:18:38.75
#>  Mode  :character                  Median :2018-05-23 22:28:09.53
#>                                    Mean   :2018-05-23 22:31:08.27
#>                                    3rd Qu.:2018-05-23 22:42:37.85
#>                                    Max.   :2018-05-23 23:25:13.25
#>    r_newid               rtt                id            Byte
#>  Length:262759      Min.   : 0.001   Min.   :    1   Length:262759
#>  Class :character   1st Qu.: 0.002   1st Qu.: 2639   Class :character
#>  Mode  :character   Median : 0.007   Median : 5495   Mode  :character
#>                     Mean   : 1.461   Mean   : 5515
#>                     3rd Qu.: 0.030   3rd Qu.: 8353
#>                     Max.   :39.289   Max.   :11424
```

Aggregation der Daten zur Beantwortung der Fragestellung bzgl. Latenzzeiten in Abhängigkeit zu QoS Level und Paketgröße. Pi

```
logsPiAgg <- aggregate(logsPi$rtt ~ logsPi$QoS+logsPi$Size+logsPi$Byte, logsPi, mean)
logsPiAgg$`logsPi$Byte`<-as.numeric(logsPiAgg$`logsPi$Byte`)
logsPiAgg<-logsPiAgg[order(logsPiAgg$`logsPi$Byte`),]

logsPiAgg %>%
  kable() %>%
  kable_styling()
```

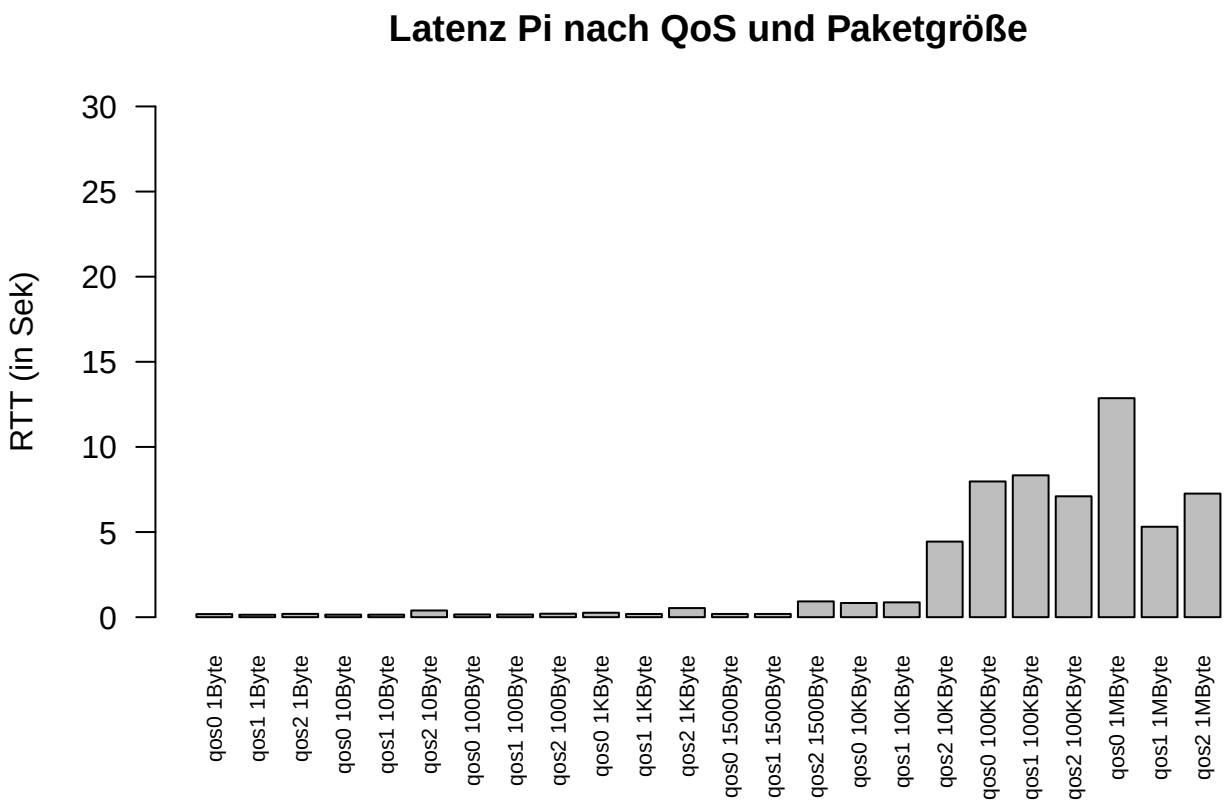
	logsPi$QoS	logsPi$Size	logsPi$Byte	logsPi$rtt
1	qos0	1Byte	1	0.1813469
2	qos1	1Byte	1	0.1419741
3	qos2	1Byte	1	0.1934519
4	qos0	10Byte	10	0.1480741
5	qos1	10Byte	10	0.1468307
6	qos2	10Byte	10	0.3937651
7	qos0	100Byte	100	0.1596423
8	qos1	100Byte	100	0.1565406
9	qos2	100Byte	100	0.2034270
10	qos0	1KByte	1000	0.2593080
11	qos1	1KByte	1000	0.1876840
12	qos2	1KByte	1000	0.5348480
16	qos0	1500Byte	1500	0.1902710
17	qos1	1500Byte	1500	0.1899368
18	qos2	1500Byte	1500	0.9265137
13	qos0	10KByte	10000	0.8326793
14	qos1	10KByte	10000	0.8707715
15	qos2	10KByte	10000	4.4394398
19	qos0	100KByte	100000	7.9708026
20	qos1	100KByte	100000	8.3323956
21	qos2	100KByte	100000	7.1009707
22	qos0	1MByte	1000000	12.8628915
23	qos1	1MByte	1000000	5.3124085
24	qos2	1MByte	1000000	7.2580572

Laptop (volle bandbreite - VB)

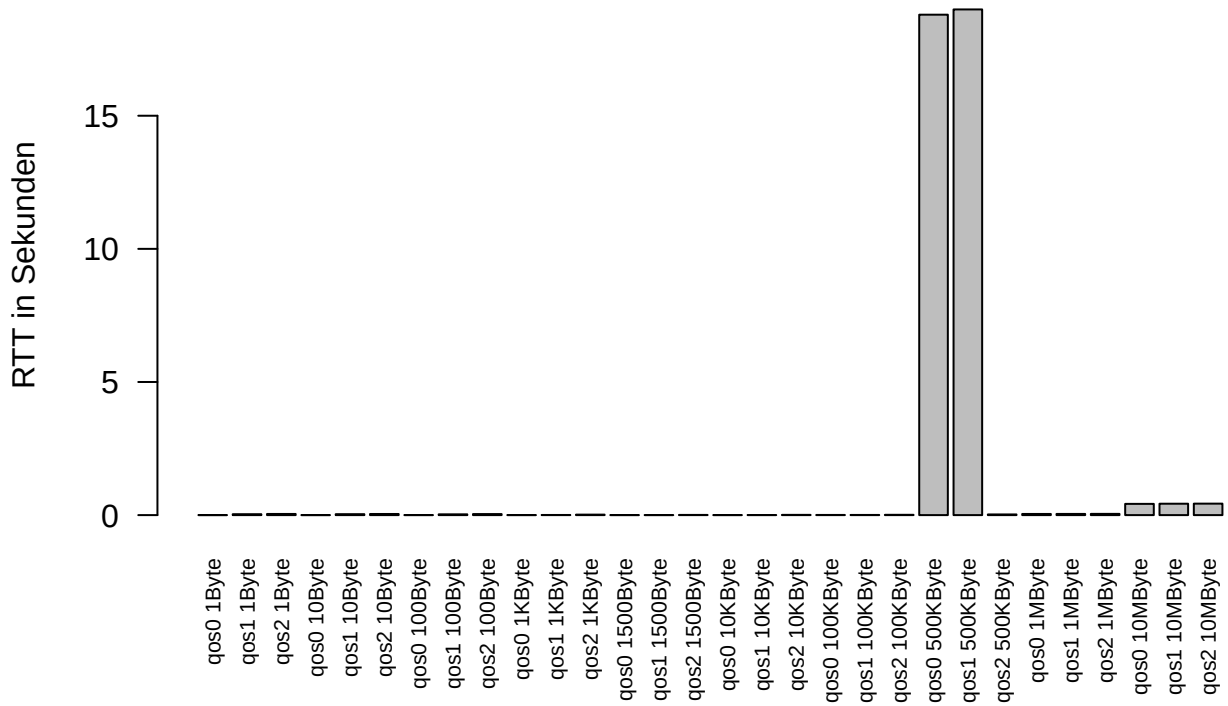
```
logsVBAgg <- aggregate(latenzVB$rtt ~ latenzVB$QoS+latenzVB$Size+latenzVB$Byte, latenzVB, mean)
logsVBAgg$`latenzVB$Byte`<-as.numeric(logsVBAgg$`latenzVB$Byte`)
logsVBAgg<-logsVBAgg[order(logsVBAgg$`latenzVB$Byte`),]


logsVBAgg %>%
  kable() %>%
  kable_styling()
```

	latenzVB$QoS	latenzVB$Size	latenzVB$Byte	latenzVB$rtt
1	qos0	1Byte	1.0e+00	0.0012375
2	qos1	1Byte	1.0e+00	0.0339261
3	qos2	1Byte	1.0e+00	0.0435640
4	qos0	10Byte	1.0e+01	0.0014071
5	qos1	10Byte	1.0e+01	0.0349903
6	qos2	10Byte	1.0e+01	0.0424829
7	qos0	100Byte	1.0e+02	0.0015745
8	qos1	100Byte	1.0e+02	0.0288117
9	qos2	100Byte	1.0e+02	0.0386520
10	qos0	1KByte	1.0e+03	0.0016336
11	qos1	1KByte	1.0e+03	0.0054663
12	qos2	1KByte	1.0e+03	0.0178899
16	qos0	1500Byte	1.5e+03	0.0017385
17	qos1	1500Byte	1.5e+03	0.0022554
18	qos2	1500Byte	1.5e+03	0.0068914
13	qos0	10KByte	1.0e+04	0.0020183
14	qos1	10KByte	1.0e+04	0.0027985
15	qos2	10KByte	1.0e+04	0.0085598
19	qos0	100KByte	1.0e+05	0.0055814
20	qos1	100KByte	1.0e+05	0.0056585
21	qos2	100KByte	1.0e+05	0.0109123
28	qos0	500KByte	5.0e+05	18.7941932
29	qos1	500KByte	5.0e+05	18.9917111
30	qos2	500KByte	5.0e+05	0.0247059
22	qos0	1MByte	1.0e+06	0.0434897
23	qos1	1MByte	1.0e+06	0.0440231
24	qos2	1MByte	1.0e+06	0.0465334
25	qos0	10MByte	1.0e+07	0.4217931
26	qos1	10MByte	1.0e+07	0.4274530
27	qos2	10MByte	1.0e+07	0.4308966

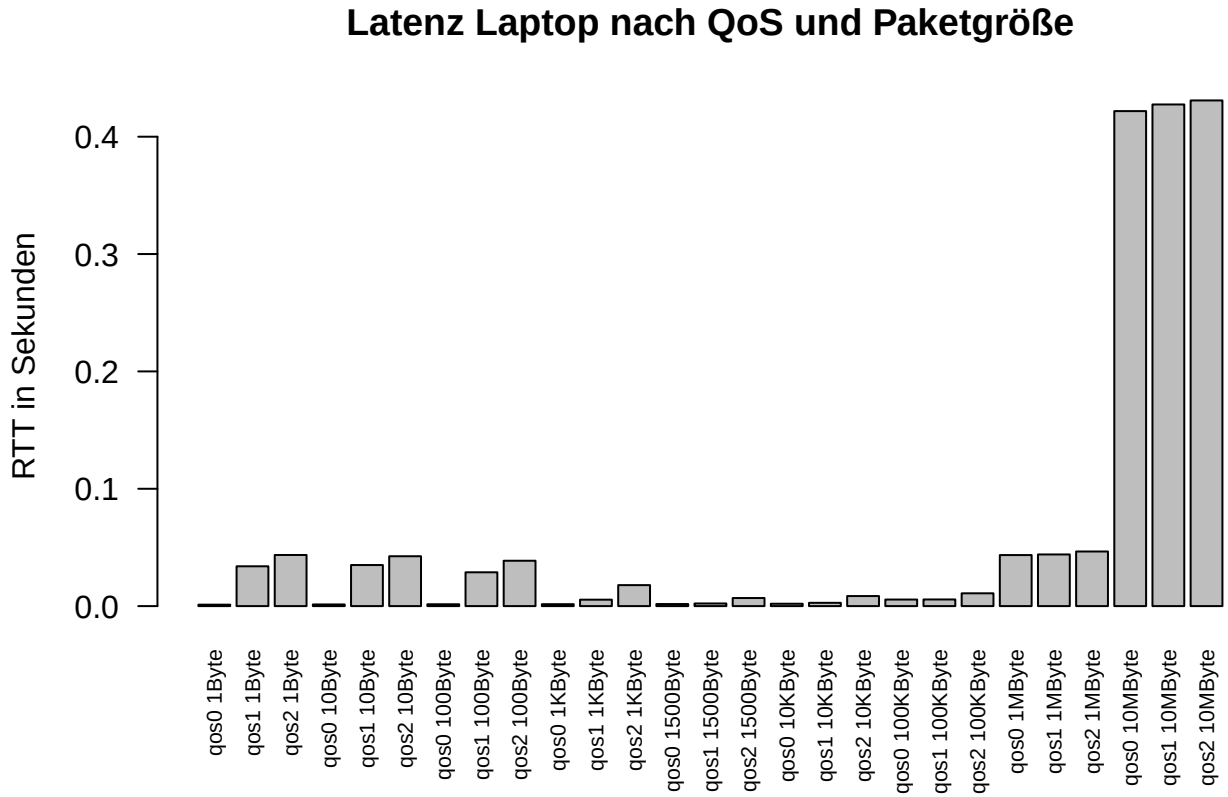
```
logsPiAgg$Names <- paste(logsPiAgg$`logsPi$QoS`, logsPiAgg$`logsPi$Size`)
logsPiAgg<-logsPiAgg[order(logsPiAgg$`logsPi$Byte`),]
barplot(logsPiAgg$`logsPi$rtt`, ylim = c(0, 30), ylab = "RTT (in Sek)", main = "Latenz Pi nach QoS und
```



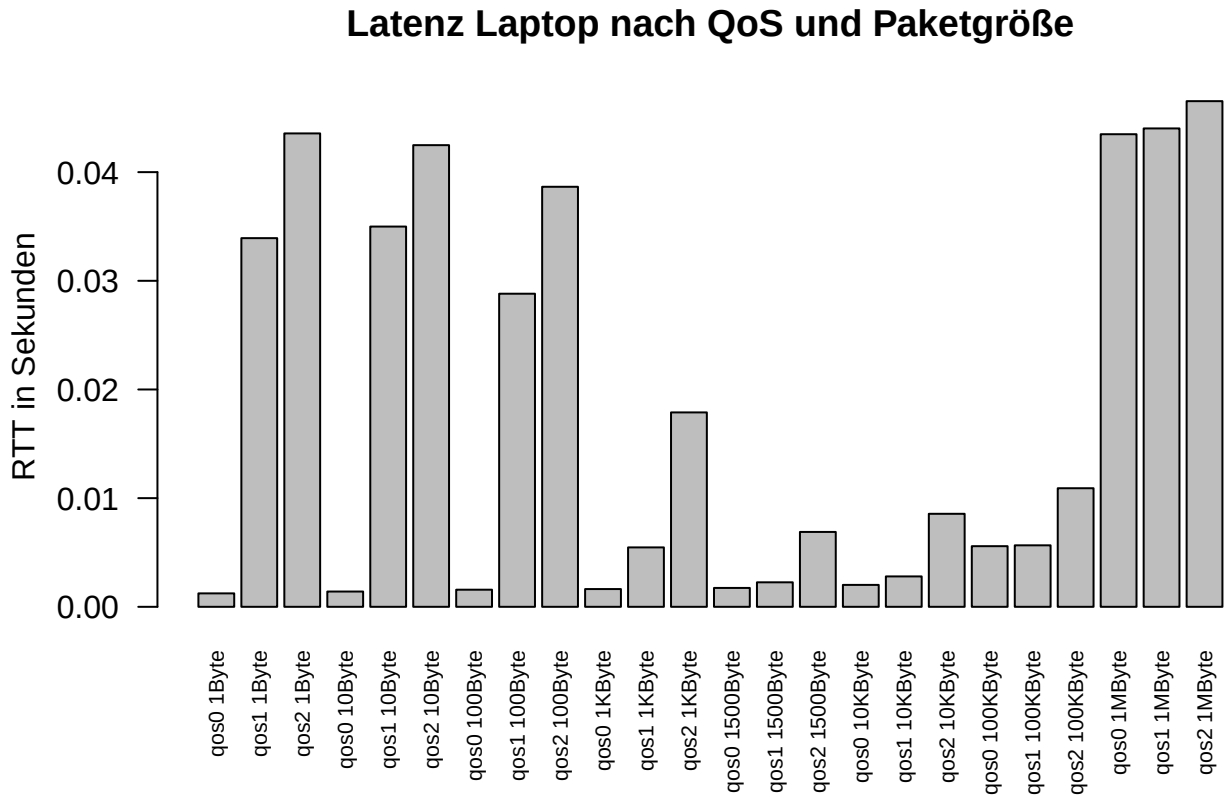

```
logsVBAgg$Names <- paste(logsVBAgg$`latenzVB$QoS`, logsVBAgg$`latenzVB$Size`)
logsVBAgg<-logsVBAgg[order(logsVBAgg$`latenzVB$Byte`),]

barplot(logsVBAgg$`latenzVB$rtt`, main = "Latenz Laptop nach QoS und Paketgröße", ylab = "RTT in Sekund
```




```
logsVBAgg_no500<-logsVBAgg[logsVBAgg$`latenzVB$Byte`!=500000,]
logsVBAgg_no500$Names <- paste(logsVBAgg_no500$`latenzVB$QoS`, logsVBAgg_no500$`latenzVB$Size`)
barplot(logsVBAgg_no500$`latenzVB$rtt`, main = "Latenz Laptop nach QoS und Paketgröße", ylab = "RTT in S
```




```
logsVBAgg_no500_no100M<-logsVBAgg[logsVBAgg$`latenzVB$Byte`!=500000 & logsVBAgg$`latenzVB$Byte`!= 10000
logsVBAgg_no500_no100M$Names <- paste(logsVBAgg_no500_no100M$`latenzVB$QoS`, logsVBAgg_no500_no100M$`la
barplot(logsVBAgg_no500_no100M$`latenzVB$rtt`, main = "Latenz Laptop nach QoS und Paketgröße", ylab = "
```



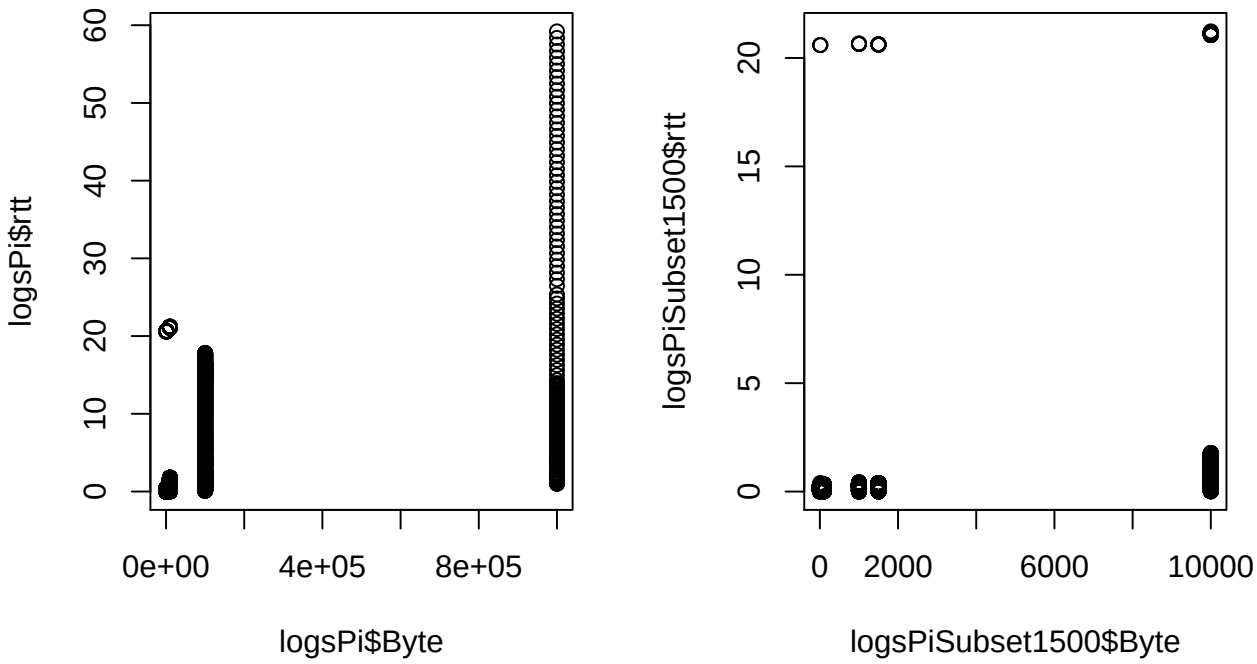
Im nächsten Schritt wird die statistische Abhngigkeit der rtt von QoS und Größe (Byte) untersucht. Im Falle einer einfachen linearen Regression sind nur qos2 und hohe Byte Zahlen signifikant.

```
reg_LogsPi <- lm(logsPi$rtt~logsPi$QoS+logsPi$Byte, data = logsPi)
summary(reg_LogsPi)
#>
#> Call:
#> lm(formula = logsPi$rtt ~ logsPi$QoS + logsPi$Byte, data = logsPi)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -9.569 -0.567  0.024  0.143 48.640
#>
#> Coefficients:
#>                   Estimate Std. Error t value Pr(>|t|)
#> (Intercept)       0.073717   0.074721   0.987   0.3239
#> logsPi$QoSqos1   -0.109345   0.057691  -1.895   0.0581 .
#> logsPi$QoSqos2    0.662022   0.069045   9.588   <2e-16 ***
#> logsPi$Byte10     0.026639   0.100035   0.266   0.7900
#> logsPi$Byte100   -0.004811   0.098328  -0.049   0.9610
#> logsPi$Byte1000   0.123495   0.097414   1.268   0.2049
#> logsPi$Byte10000  1.501956   0.093533  16.058   <2e-16 ***
#> logsPi$Byte1500   0.166982   0.098241   1.700   0.0892 .
#> logsPi$Byte1e+05  7.861722   0.098293  79.983   <2e-16 ***
#> logsPi$Byte1e+06 10.495737   0.196943  53.293   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 3.319 on 16984 degrees of freedom
#> Multiple R-squared:  0.4361, Adjusted R-squared:  0.4358
#> F-statistic:  1459 on 9 and 16984 DF,  p-value: < 2.2e-16
```

```
reg_LogsVB <- lm(latenzVB$rtt~latenzVB$QoS+latenzVB$Byte, data = latenzVB)
summary(reg_LogsVB)
#>
#> Call:
#> lm(formula = latenzVB$rtt ~ latenzVB$QoS + latenzVB$Byte, data = latenzVB)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -17.8248  -0.1420  -0.1103   0.2488  21.4012
#>
#> Coefficients:
#>                     Estimate Std. Error t value Pr(>|t|)
#> (Intercept)         0.1305348  0.0201175   6.489 8.68e-11 ***
#> latenzVB$QoSqos1    0.0377104  0.0155270   2.429   0.0152 *
#> latenzVB$QoSqos2   -0.3507694  0.0161027 -21.783  < 2e-16 ***
#> latenzVB$Byte10    -0.0002257  0.0253863  -0.009   0.9929
#> latenzVB$Byte100   -0.0038006  0.0253910  -0.150   0.8810
#> latenzVB$Byte1000  -0.0182091  0.0253392  -0.719   0.4724
#> latenzVB$Byte10000 -0.0215506  0.0253309  -0.851   0.3949
#> latenzVB$Byte1500  -0.0225965  0.0253274  -0.892   0.3723
#> latenzVB$Byte1e+05 -0.0177138  0.0254977  -0.695   0.4872
#> latenzVB$Byte1e+06  0.0186013  0.0586553   0.317   0.7511
#> latenzVB$Byte1e+07  0.4001275  0.1777599   2.251   0.0244 *
```

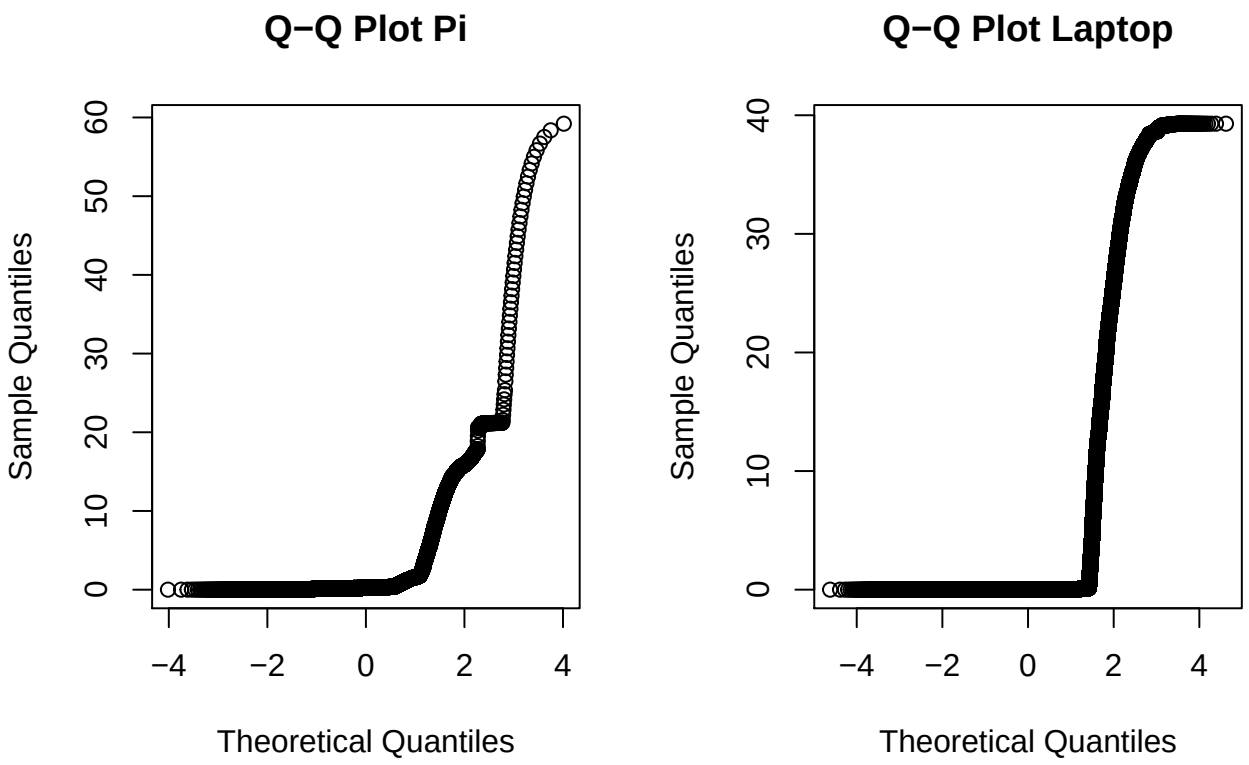
```
#> latenzVB$Byte5e+05 17.7195457  0.0291650 607.563  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 3.304 on 262747 degrees of freedom
#> Multiple R-squared:  0.6848, Adjusted R-squared:  0.6848
#> F-statistic: 5.19e+04 on 11 and 262747 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(1,2))
plot(logsPi$Byte, logsPi$rtt)
logsPiSubset1500 <- logsPi[logsPi$Byte <= 1500, ]
plot(logsPiSubset1500$Byte, logsPiSubset1500$rtt)
```



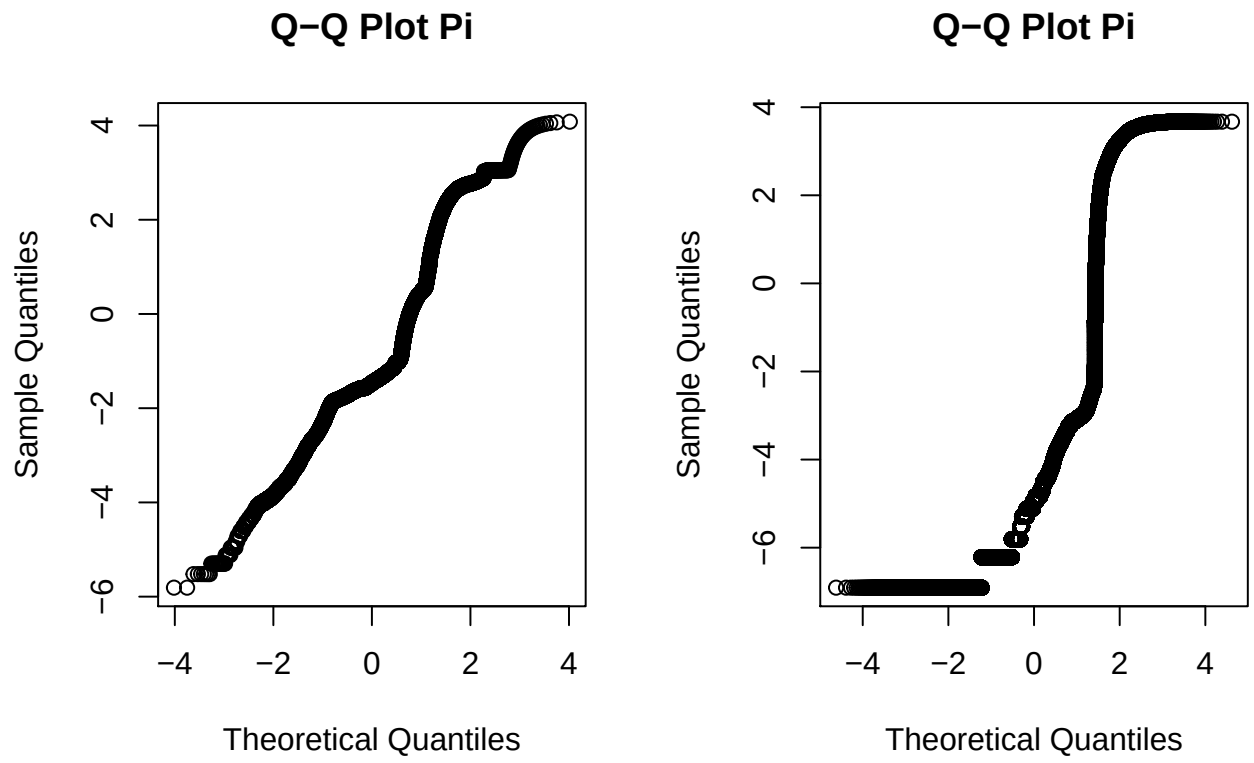
Scatterplots bringen bei der Größe wenig Übersicht aus zwei Gründen: 1.) Logische Datenypen, d.h. alle Beobachtungen sind gehäuft in den geweiligen Klassen 2.) Ohne Standardisierung/ Transformation der Daten haben die extremen Werte (MByte) einen überproportionalen Anteil -> Im folgenden wird die Verteilung von rtt in QQ Plots betrachtet /.

```
par(mfrow=c(1,2))
qqnorm(logsPi$rtt, main = "Q-Q Plot Pi")
qqnorm(latenzVB$rtt, main = "Q-Q Plot Laptop")
```



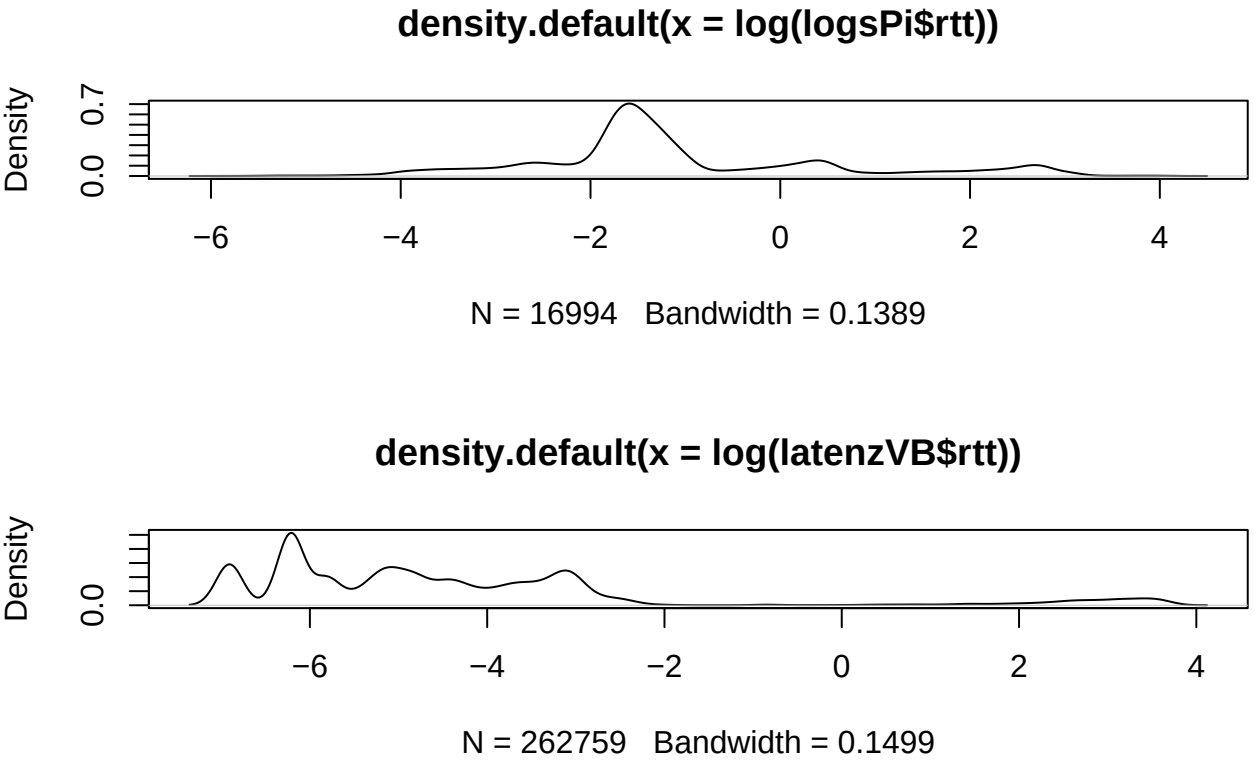
Da rtt nicht normal verteilt ist, liefert die Lineare Regression keine zuverlässigen Ergebnisse. Nach der Transformation (logarithmierung) nähert sich die Verteilung der Variable rtt der Normalverteilung. (Normalverteilung ist erreicht, wenn die Sample Quantile den Theoretischen entsprechen - die Beobachtugen also auf einer Geraden liegen)

```
par(mfrow=c(1,2))
qqnorm(log(logsPi$rtt), main = "Q-Q Plot Pi")
qqnorm(log(latenzVB$rtt), main = "Q-Q Plot Pi")
```



Trotz der Logarithmierung sind die Daten nicht perfekt Normalverteilt, jedoch annähernd.

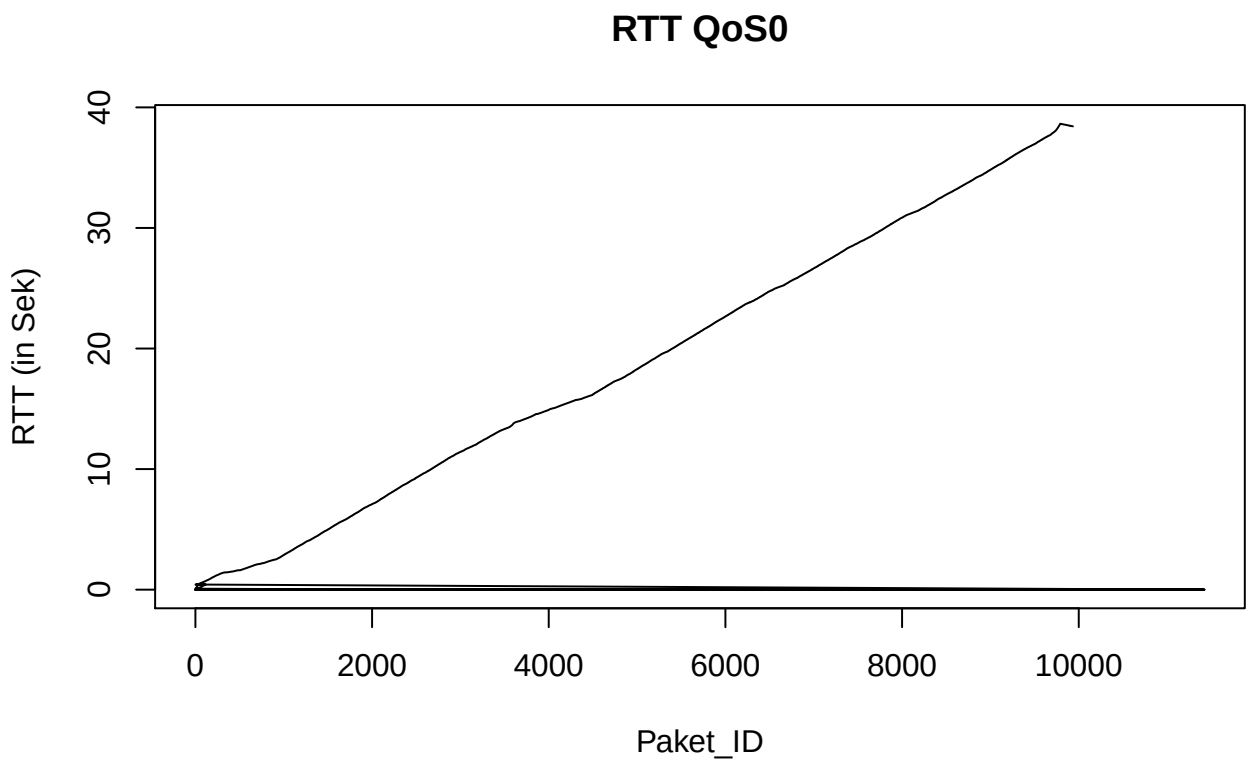
```
par(mfrow=c(2,1))
plot(density(log(logsPi$rtt)))
plot(density(log(latenzVB$rtt)))
```



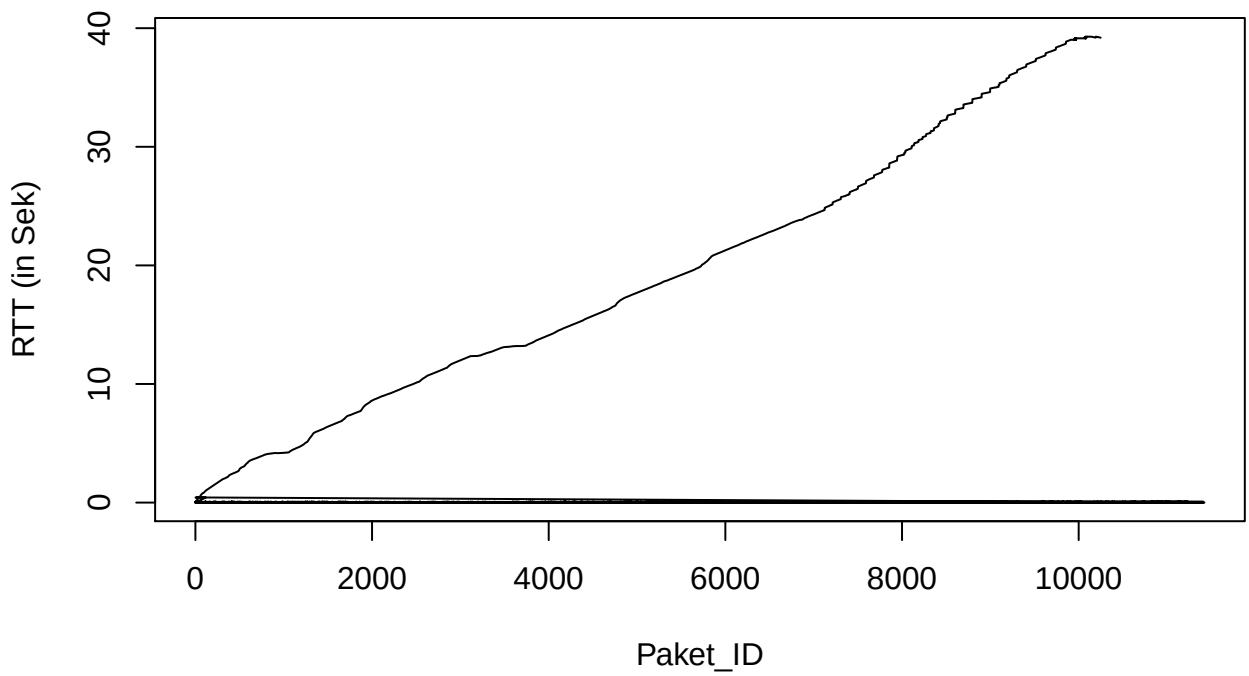
```
# shapiro.test(logsPi$rtt)
```

```
reg_LogsPi <- lm(log(logsPi$rtt)~logsPi$QoS+logsPi$Byte, data = logsPi)
summary(reg_LogsPi)
#>
#> Call:
#> lm(formula = log(logsPi$rtt) ~ logsPi$QoS + logsPi$Byte, data = logsPi)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -4.1942 -0.4105  0.2807  0.5366  4.8238
#>
#> Coefficients:
#>                  Estimate Std. Error t value Pr(>|t|)
#> (Intercept)      -1.95561    0.02077 -94.139  < 2e-16 ***
#> logsPi$QoSqos1   -0.10038    0.01604  -6.258 3.99e-10 ***
#> logsPi$QoSqos2    0.21459    0.01920  11.179  < 2e-16 ***
#> logsPi$Byte10    -0.05772    0.02781  -2.075    0.038 *
#> logsPi$Byte100   -0.04126    0.02734  -1.509    0.131
#> logsPi$Byte1000   0.23350    0.02708   8.622  < 2e-16 ***
#> logsPi$Byte10000  1.58765    0.02600  61.055  < 2e-16 ***
#> logsPi$Byte1500   0.14632    0.02731   5.357 8.57e-08 ***
#> logsPi$Byte1e+05  3.63646    0.02733 133.072  < 2e-16 ***
#> logsPi$Byte1e+06  3.70071    0.05475  67.588  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.9228 on 16984 degrees of freedom
#> Multiple R-squared:  0.6719, Adjusted R-squared:  0.6717
#> F-statistic:  3865 on 9 and 16984 DF,  p-value: < 2.2e-16
```

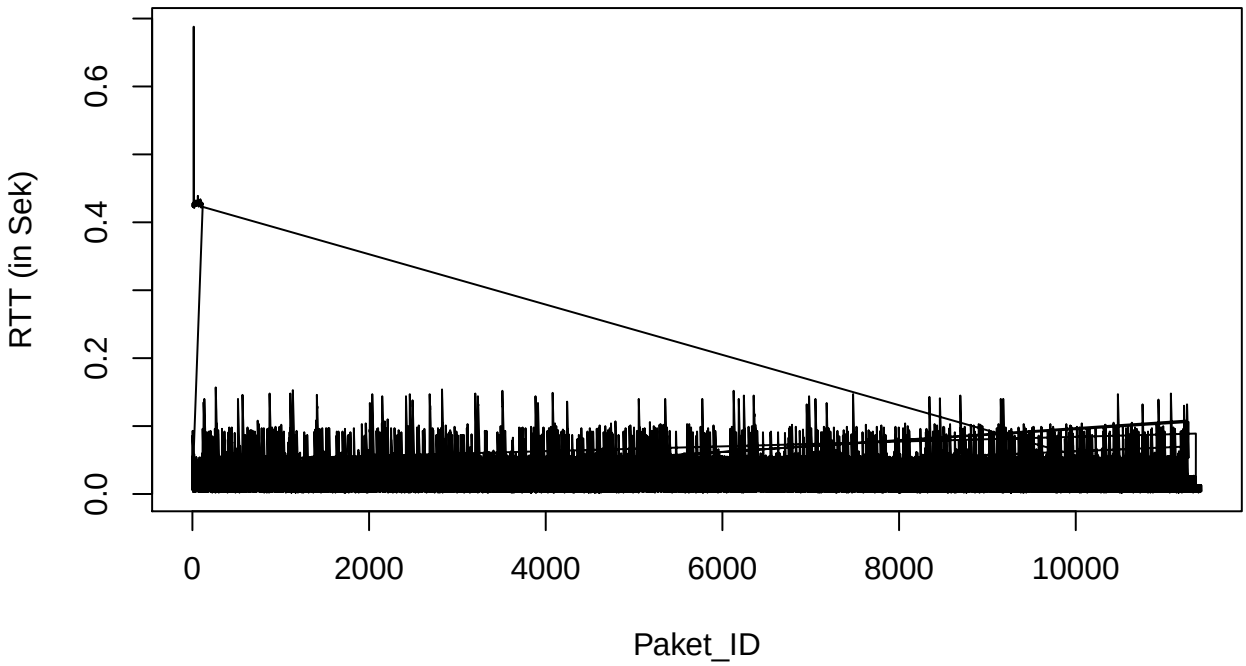
```
######################
# Aufteilen nach QoS #
######################

latenzVBQoS0<-latenzVB[latenzVB$QoS == "qos0",]
latenzVBQoS1<-latenzVB[latenzVB$QoS == "qos1",]
latenzVBQoS2<-latenzVB[latenzVB$QoS == "qos2",]

plot(latenzVBQoS0$id, latenzVBQoS0$rtt, type = "l", main = "RTT QoS0",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



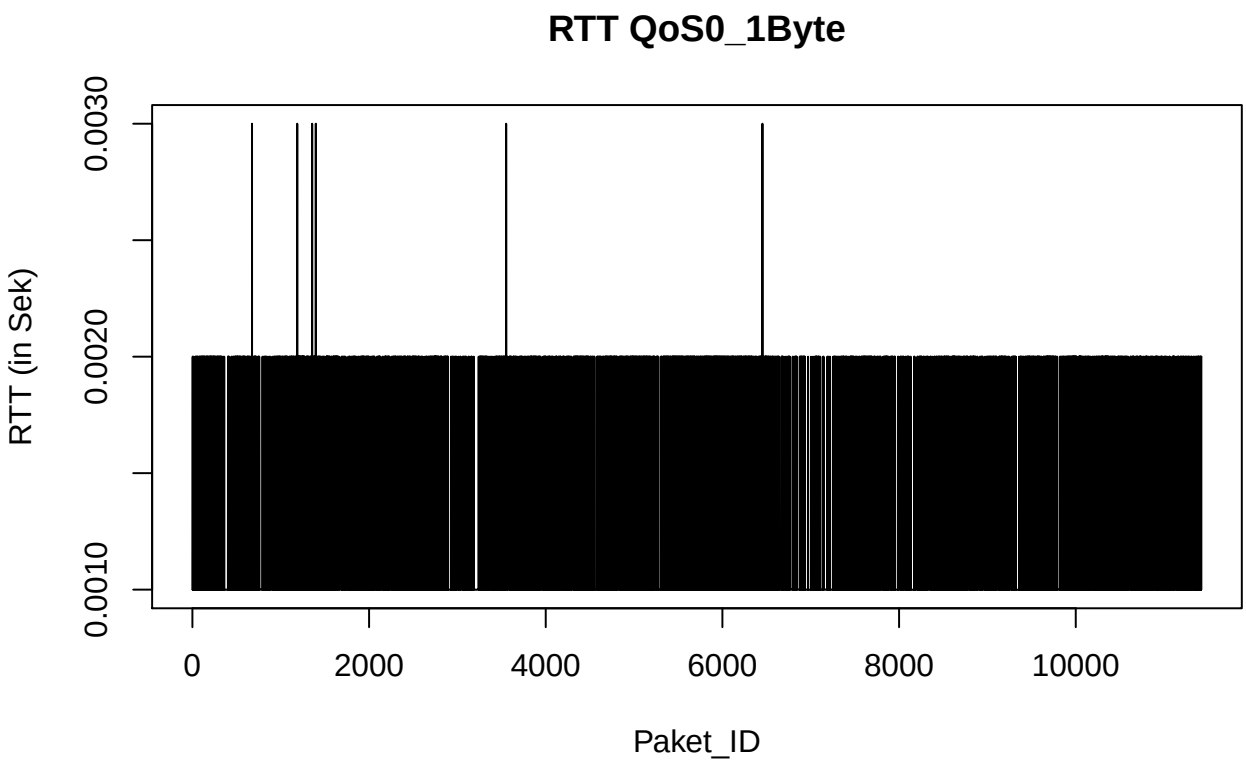

```
plot(latenzVBQoS1$id, latenzVBQoS1$rtt, type = "l", main = "RTT QoS1",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



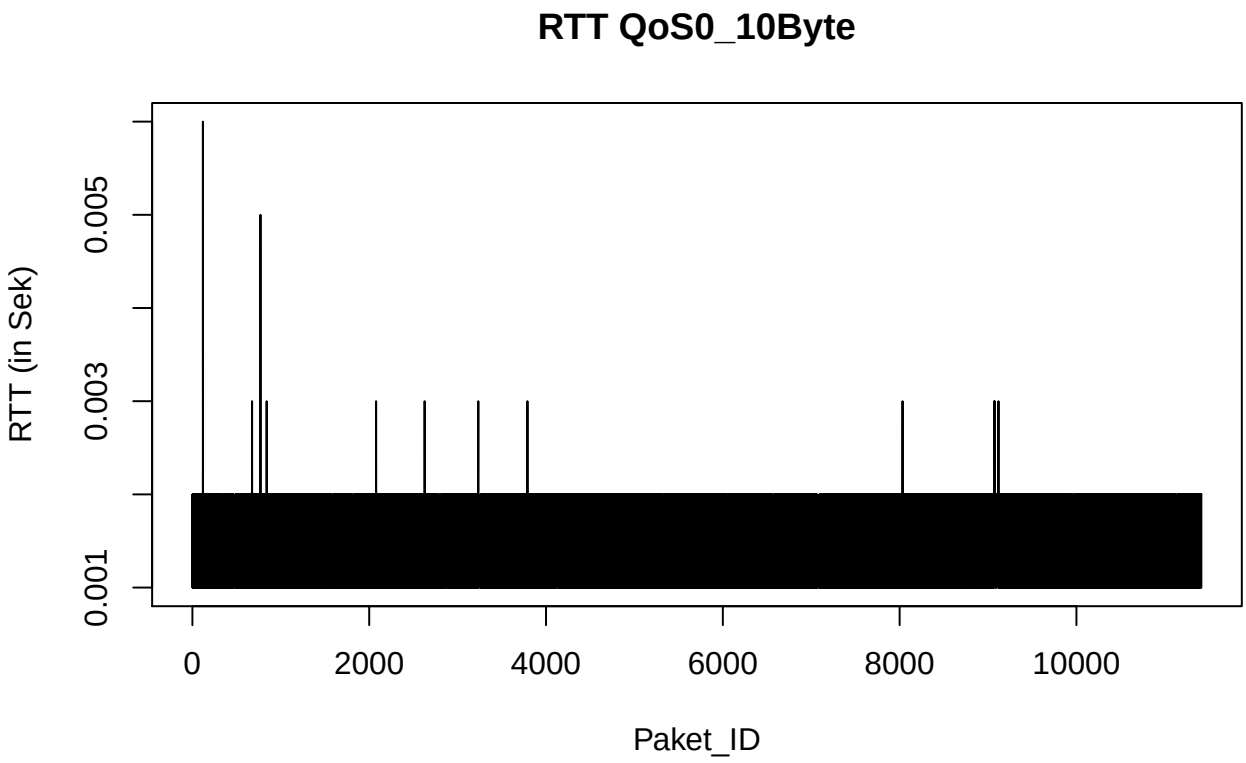

```
plot(latenzVBQoS2$id, latenzVBQoS2$rtt, type = "l", main = "RTT QoS2",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



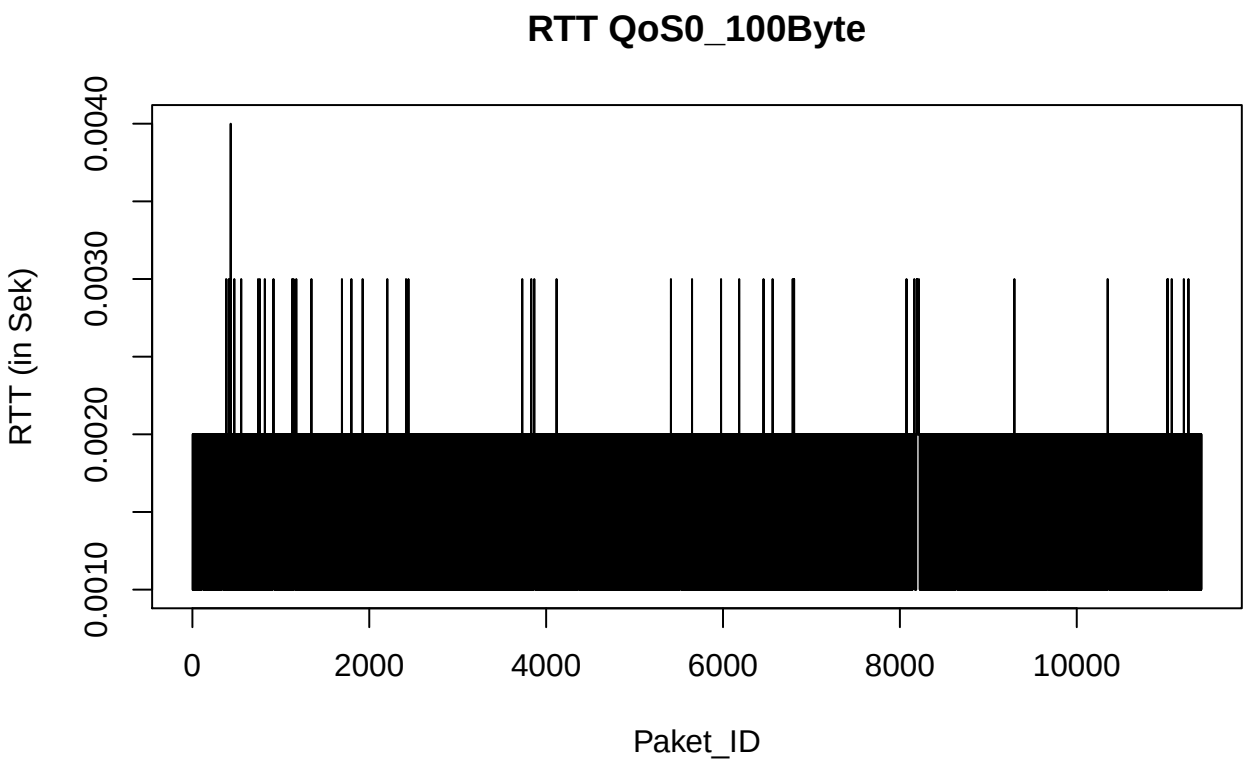

```
########################################
# QoS_0 Aufteilen nach Payload/ Größe #
########################################

latenzVBQoS01Byte<-latenzVBQoS0[latenzVBQoS0$Size == "1Byte",]
latenzVBQoS010Byte<-latenzVBQoS0[latenzVBQoS0$Size == "10Byte",]
latenzVBQoS0100Byte<-latenzVBQoS0[latenzVBQoS0$Size == "100Byte",]
latenzVBQoS01KByte<-latenzVBQoS0[latenzVBQoS0$Size == "1KByte",]
latenzVBQoS01500Byte<-latenzVBQoS0[latenzVBQoS0$Size == "1500Byte",]
latenzVBQoS010KByte<-latenzVBQoS0[latenzVBQoS0$Size == "10KByte",]
latenzVBQoS0100KByte<-latenzVBQoS0[latenzVBQoS0$Size == "100KByte",]
latenzVBQoS0500KByte<-latenzVBQoS0[latenzVBQoS0$Size == "500KByte",]
latenzVBQoS01MByte<-latenzVBQoS0[latenzVBQoS0$Size == "1MByte",]
latenzVBQoS010MByte<-latenzVBQoS0[latenzVBQoS0$Size == "10MByte",]

plot(latenzVBQoS01Byte$id, latenzVBQoS01Byte$rtt, type = "l", main = "RTT QoS0_1Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



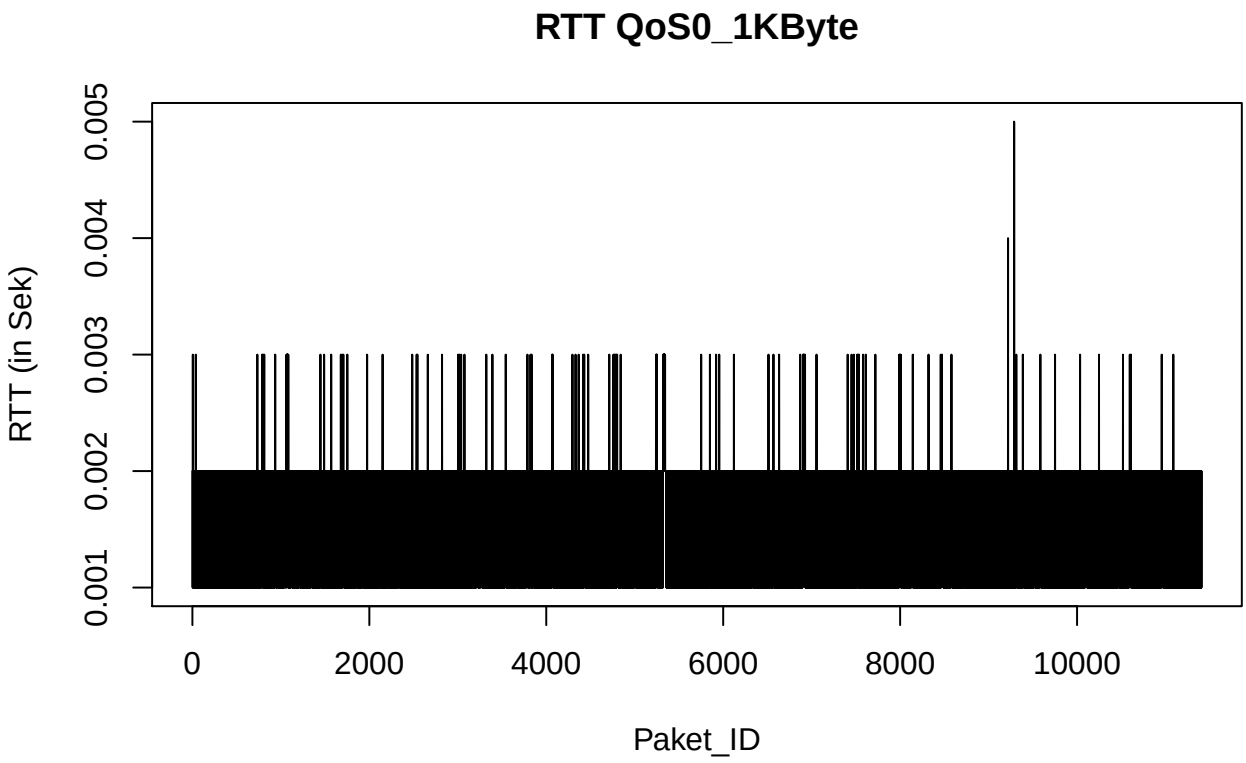

```
plot(latenzVBQoS010Byte$id, latenzVBQoS010Byte$rtt, type = "l", main = "RTT QoS0_10Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



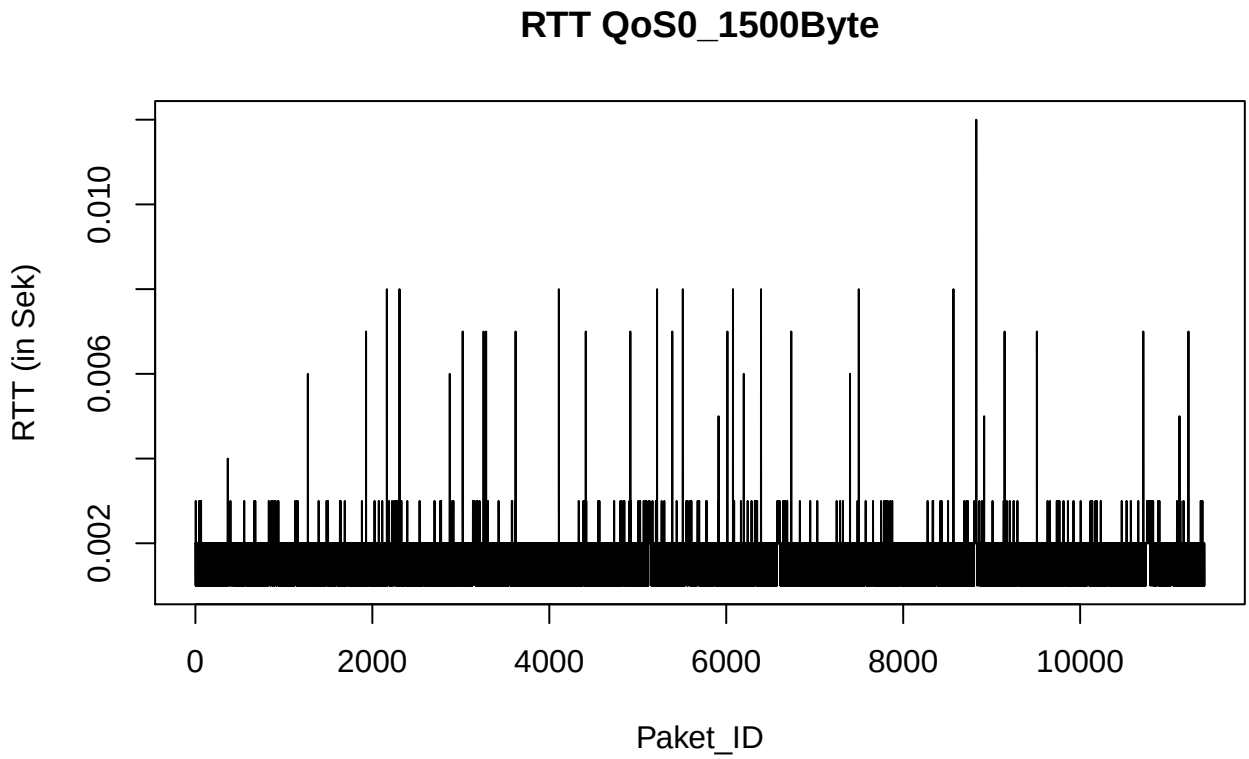

```
plot(latenzVBQoS0100Byte$id, latenzVBQoS0100Byte$rtt, type = "l", main = "RTT QoS0_100Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



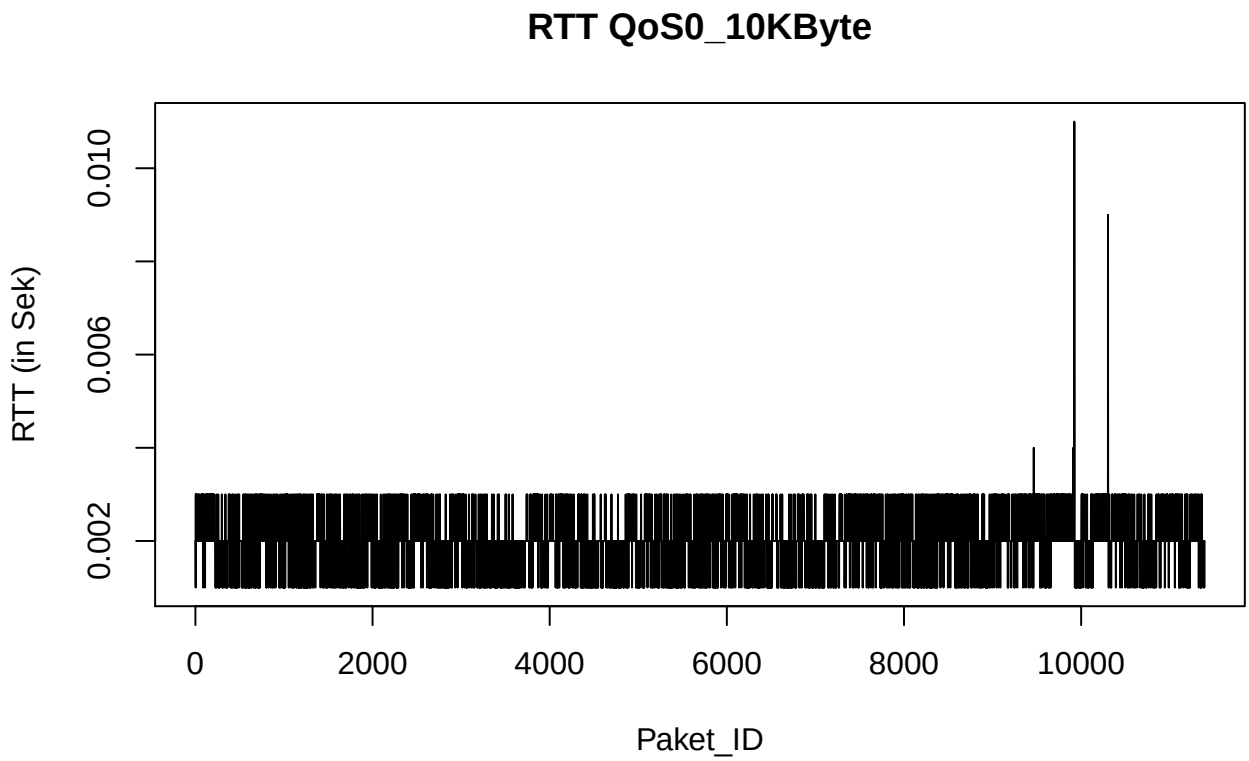

```
plot(latenzVBQoS01KByte$id, latenzVBQoS01KByte$rtt, type = "l", main = "RTT QoS0_1KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



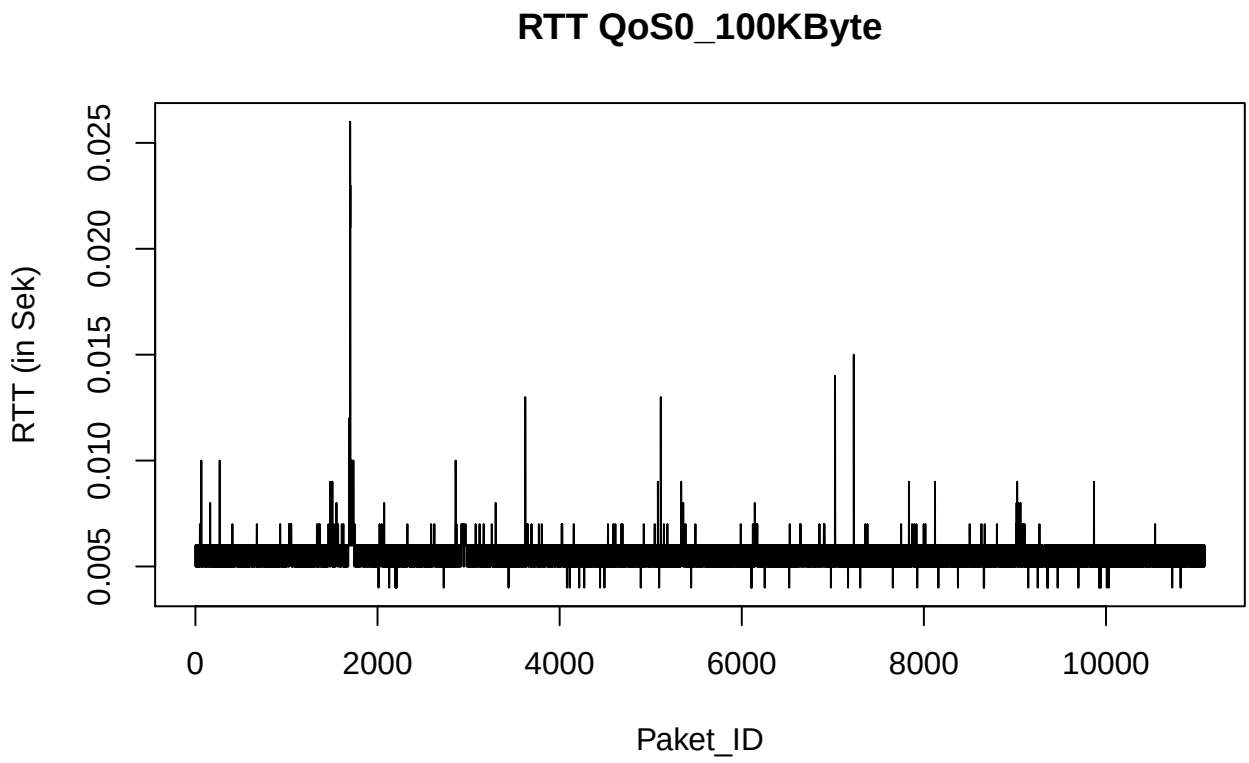

```
plot(latenzVBQoS01500Byte$id, latenzVBQoS01500Byte$rtt, type = "l", main = "RTT QoS0_1500Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



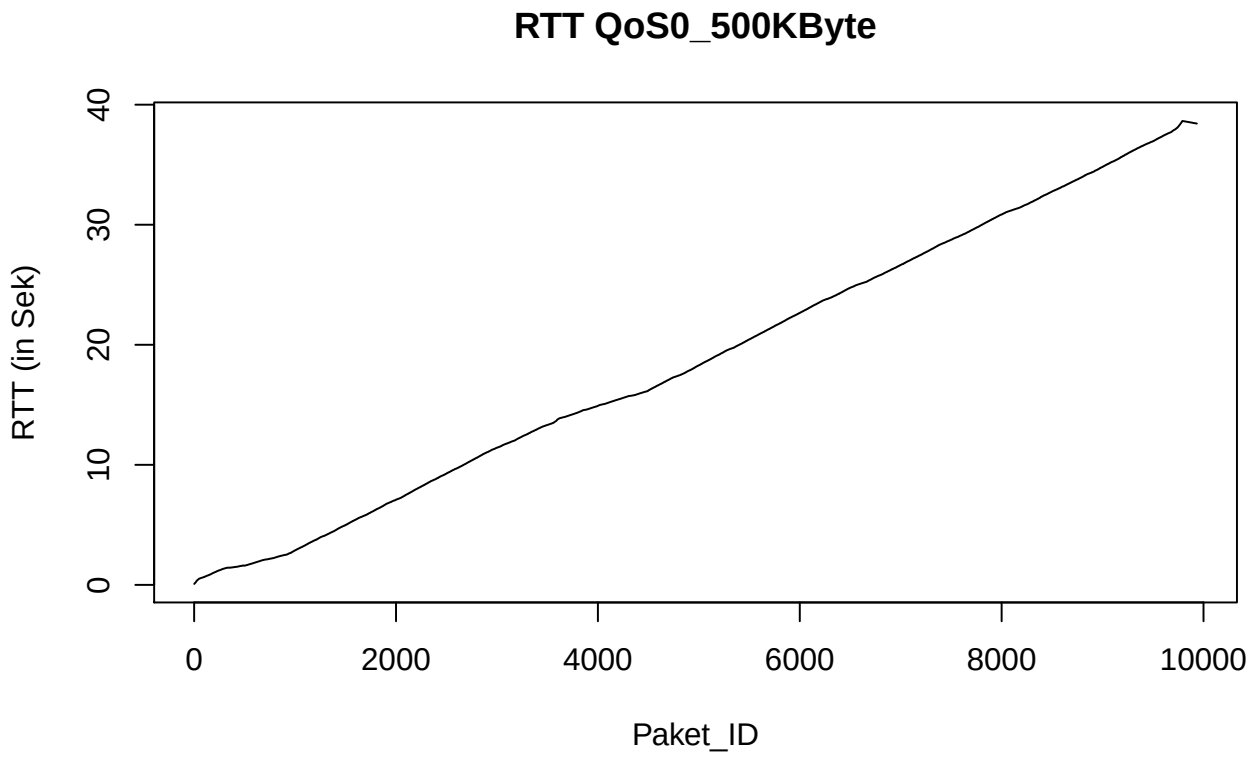

```
plot(latenzVBQoS010KByte$id, latenzVBQoS010KByte$rtt, type = "l", main = "RTT QoS0_10KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



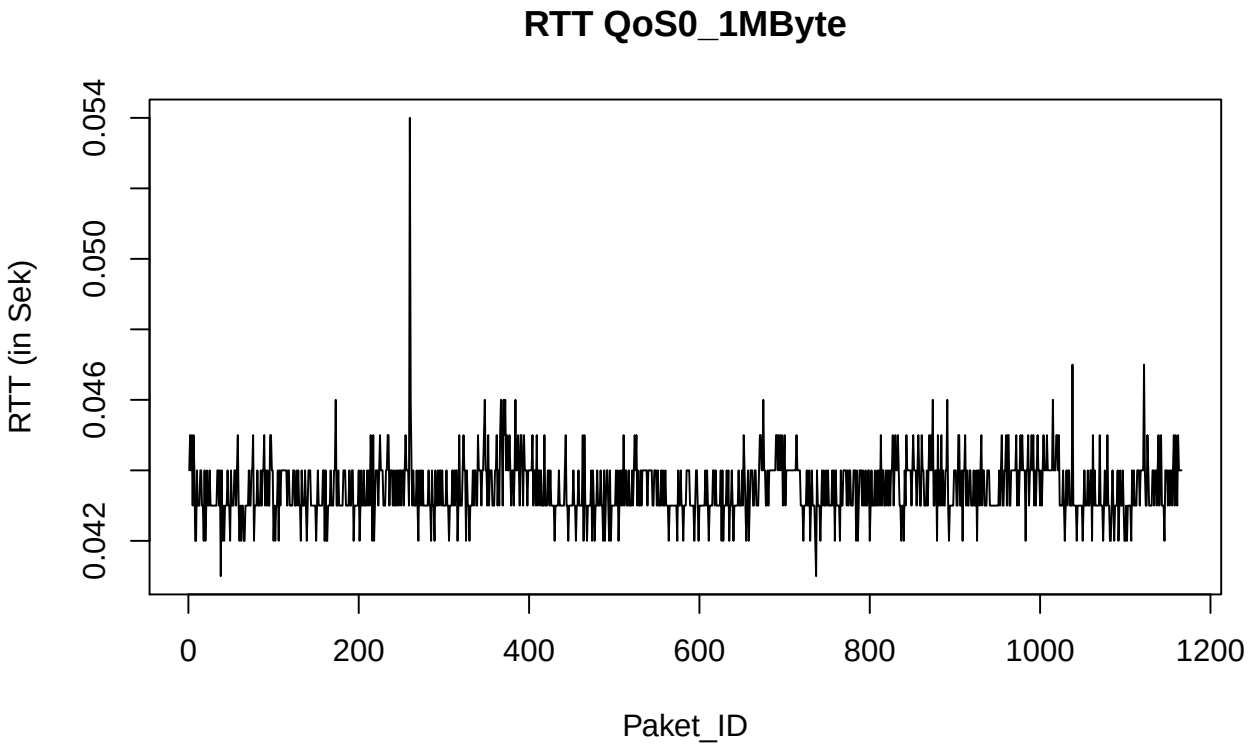

```
plot(latenzVBQoS0100KByte$id, latenzVBQoS0100KByte$rtt, type = "l", main = "RTT QoS0_100KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



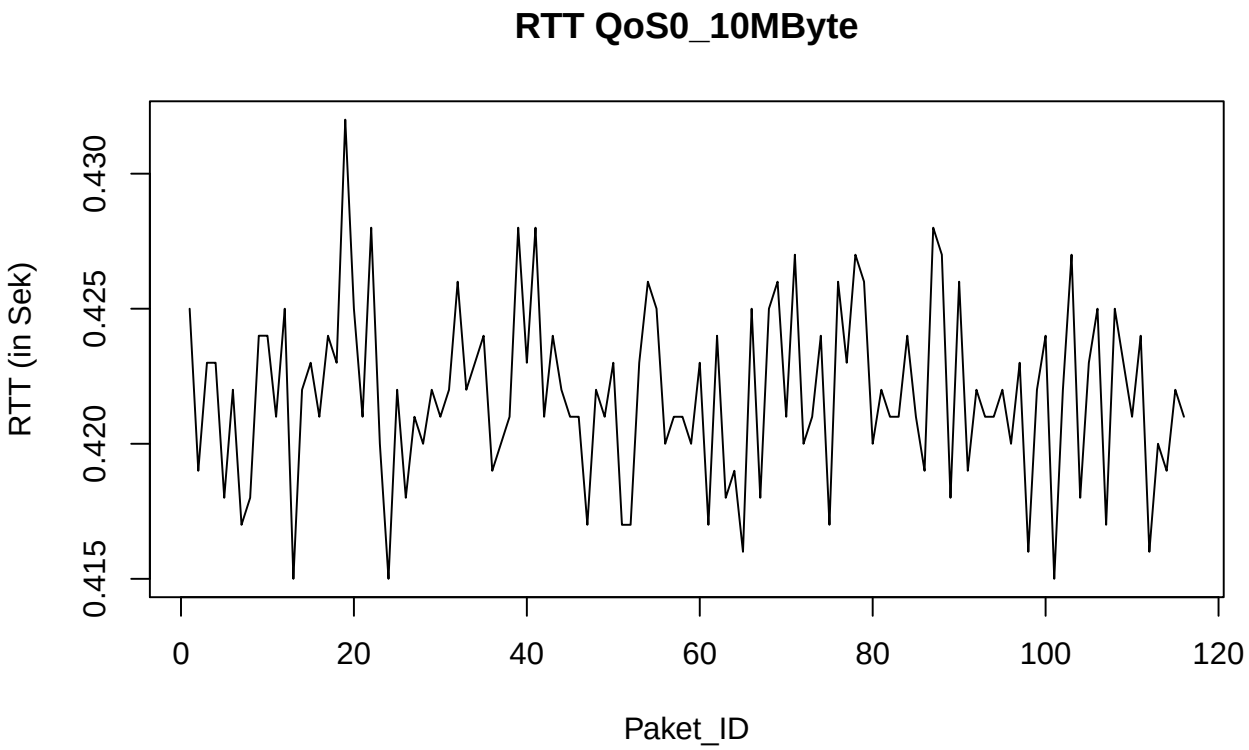

```
plot(latenzVBQoS0500KByte$id, latenzVBQoS0500KByte$rtt, type = "l", main = "RTT QoS0_500KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```




```
plot(latenzVBQoS01MByte$id, latenzVBQoS01MByte$rtt, type = "l", main = "RTT QoS0_1MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```

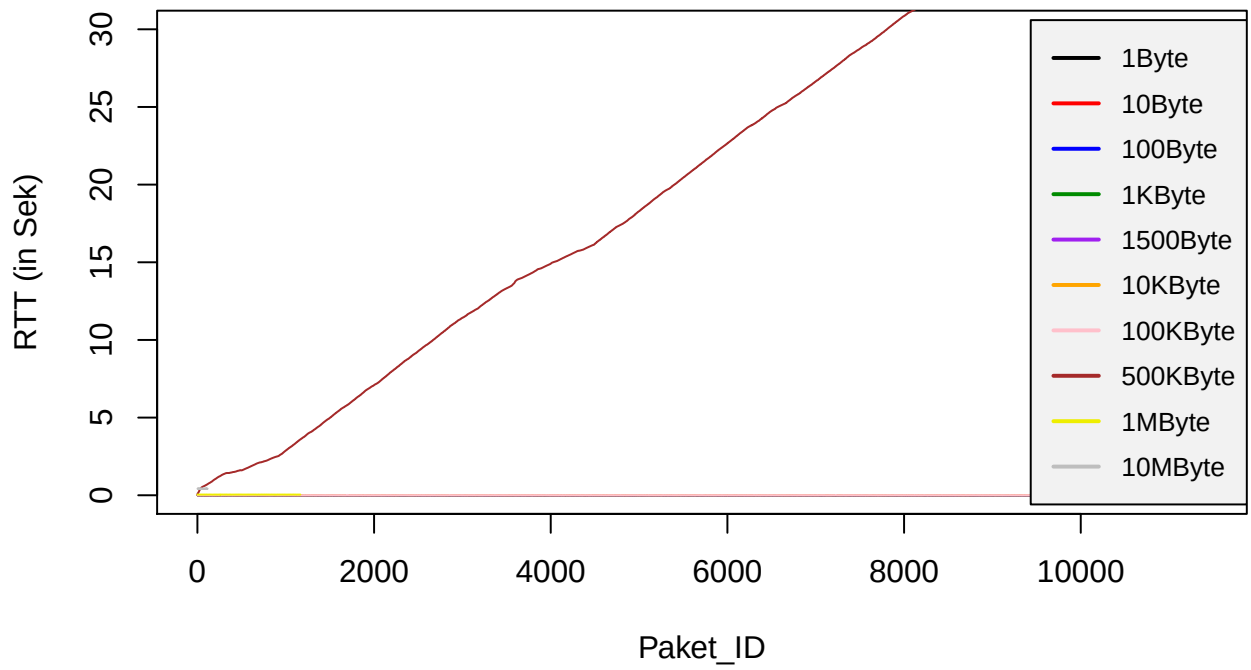



```
plot(latenzVBQoS010MByte$id, latenzVBQoS010MByte$rtt, type = "l", main = "RTT QoS0_10MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



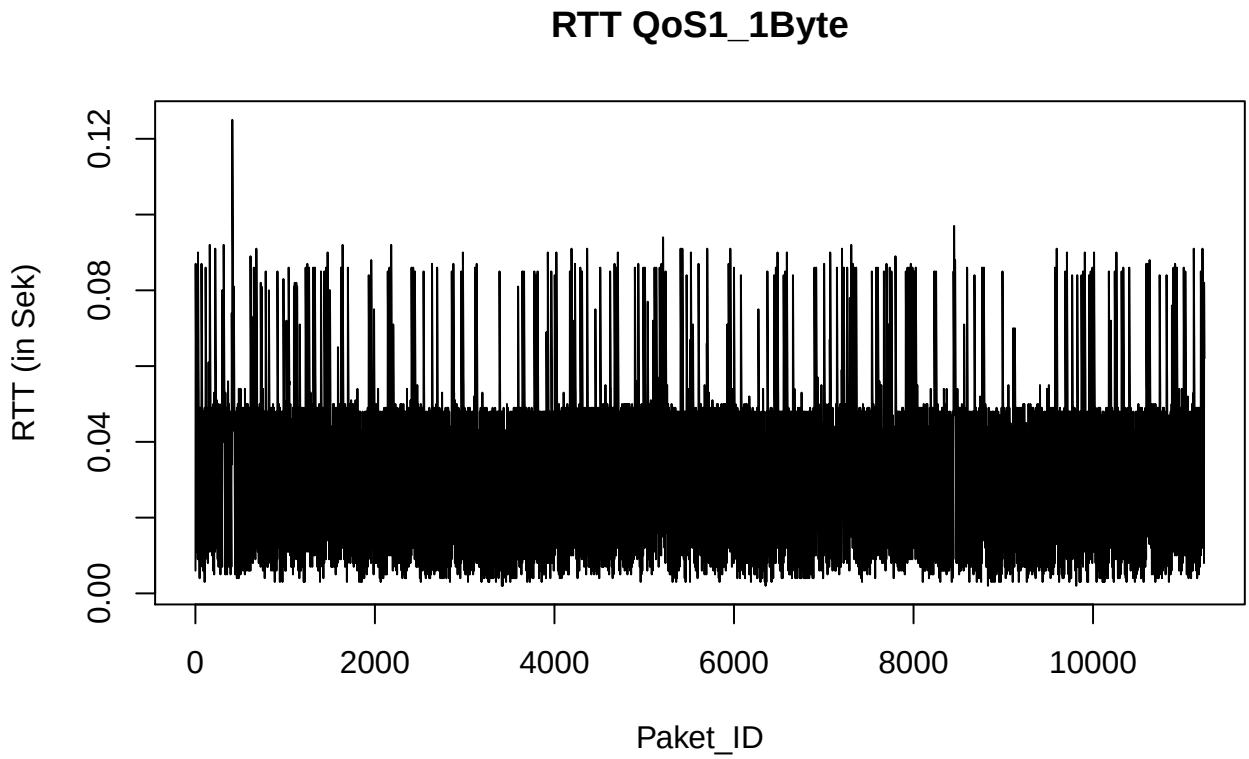

```
## QoS0 - Aufsplittung - eine Grafik!
plot(latenzVBQoS01Byte$id, latenzVBQoS01Byte$rtt, type = "l", ylim = c(0, 30), ylab = "RTT (in Sek)", xl
```

```
     main = "Aufsplittung aller Messungen mit QoS_0 nach Paketgröße")
points(latenzVBQoS010Byte$id, latenzVBQoS010Byte$rtt, col = "red", type = "l")
points(latenzVBQoS0100Byte$id, latenzVBQoS0100Byte$rtt, col = "blue", type = "l")
points(latenzVBQoS01KByte$id, latenzVBQoS01KByte$rtt, col = "green4", type = "l")
points(latenzVBQoS01500Byte$id, latenzVBQoS01500Byte$rtt, col = "purple", type = "l")
points(latenzVBQoS010KByte$id, latenzVBQoS010KByte$rtt, col = "orange",  type = "l")
points(latenzVBQoS0100KByte$id, latenzVBQoS0100KByte$rtt, col = "pink", type = "l")
points(latenzVBQoS0500KByte$id, latenzVBQoS0500KByte$rtt, col = "brown", type = "l")
points(latenzVBQoS01MByte$id, latenzVBQoS01MByte$rtt, col = "yellow2", type = "l")
points(latenzVBQoS010MByte$id, latenzVBQoS010MByte$rtt, col = "gray", type = "l")


legend("right", c("1Byte", "10Byte", "100Byte", "1KByte", "1500Byte", "10KByte", "100KByte", "500KByte"
       cex = 0.8,
       col = c("black", "red", "blue", "green4", "purple", "orange", "pink", "brown", "yellow2", "gray"
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```

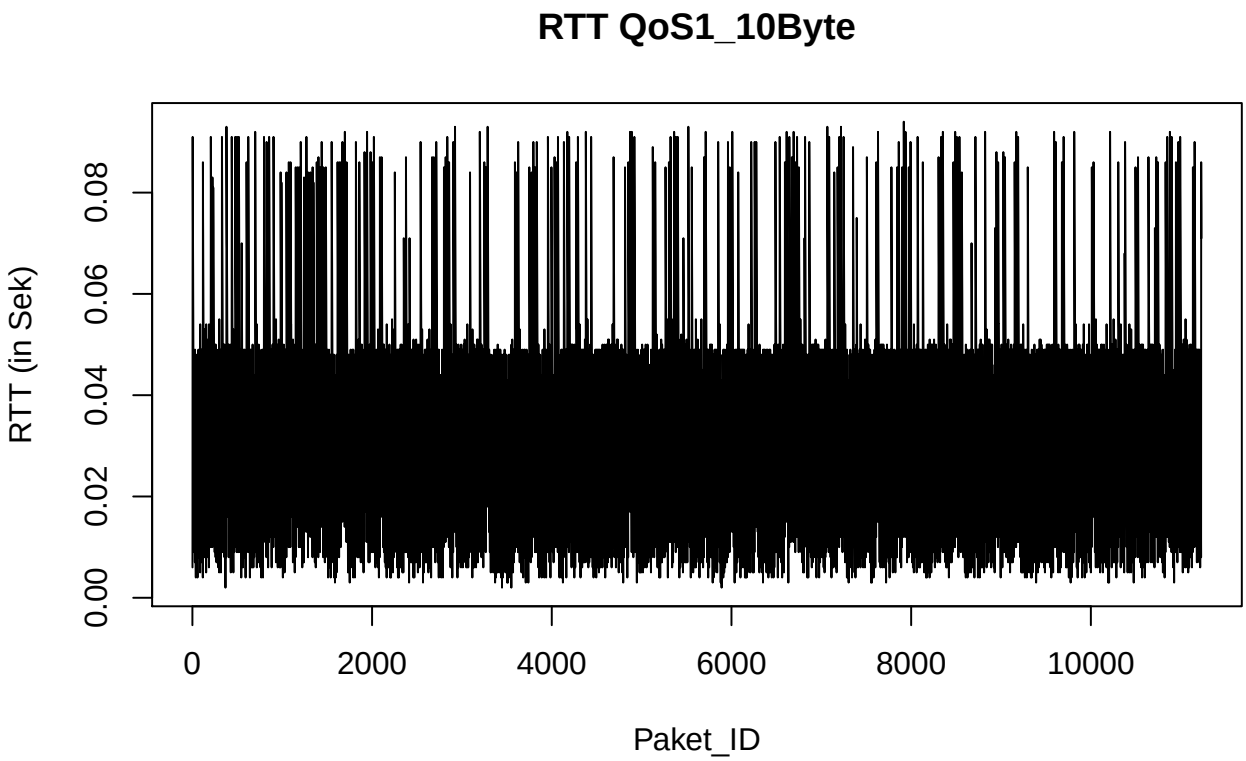



```
########################################
# QoS_1 Aufteilen nach Payload/ Größe #
########################################
latenzVBQoS11Byte<-latenzVBQoS1[latenzVBQoS1$Size == "1Byte",]
latenzVBQoS110Byte<-latenzVBQoS1[latenzVBQoS1$Size == "10Byte",]
latenzVBQoS1100Byte<-latenzVBQoS1[latenzVBQoS1$Size == "100Byte",]
latenzVBQoS11KByte<-latenzVBQoS1[latenzVBQoS1$Size == "1KByte",]
latenzVBQoS11500Byte<-latenzVBQoS1[latenzVBQoS1$Size == "1500Byte",]
latenzVBQoS110KByte<-latenzVBQoS1[latenzVBQoS1$Size == "10KByte",]
latenzVBQoS1100KByte<-latenzVBQoS1[latenzVBQoS1$Size == "100KByte",]
latenzVBQoS1500KByte<-latenzVBQoS1[latenzVBQoS1$Size == "500KByte",]
latenzVBQoS11MByte<-latenzVBQoS1[latenzVBQoS1$Size == "1MByte",]
```

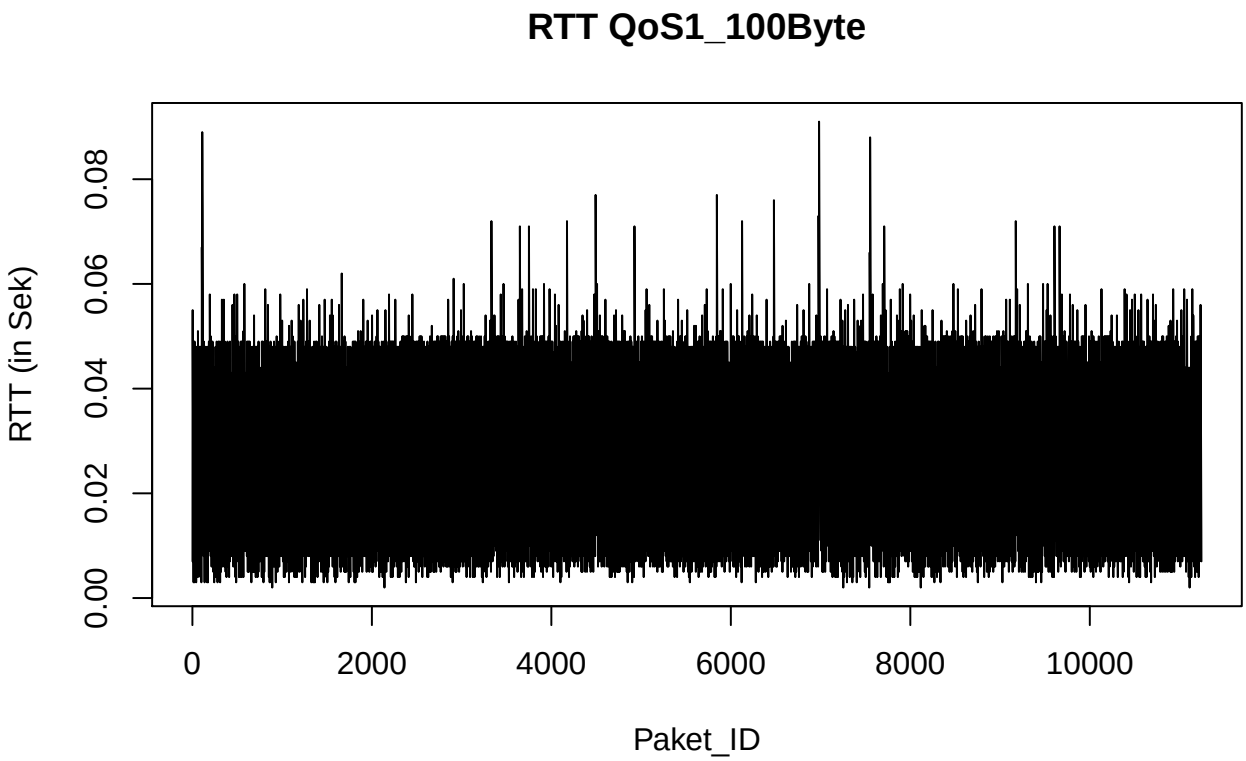
```
latenzVBQoS110MByte<-latenzVBQoS1[latenzVBQoS1$Size == "10MByte",]

plot(latenzVBQoS11Byte$id, latenzVBQoS11Byte$rtt, type = "l", main = "RTT QoS1_1Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



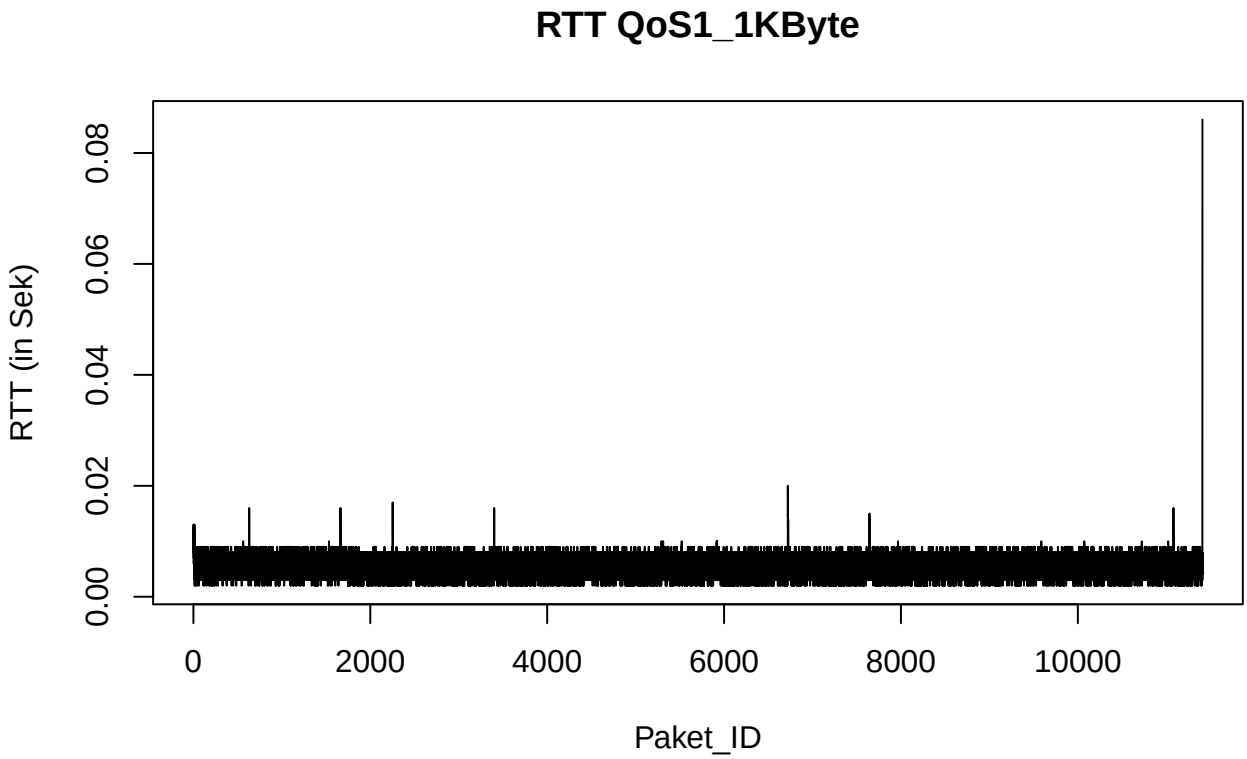

```
plot(latenzVBQoS110Byte$id, latenzVBQoS110Byte$rtt, type = "l", main = "RTT QoS1_10Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



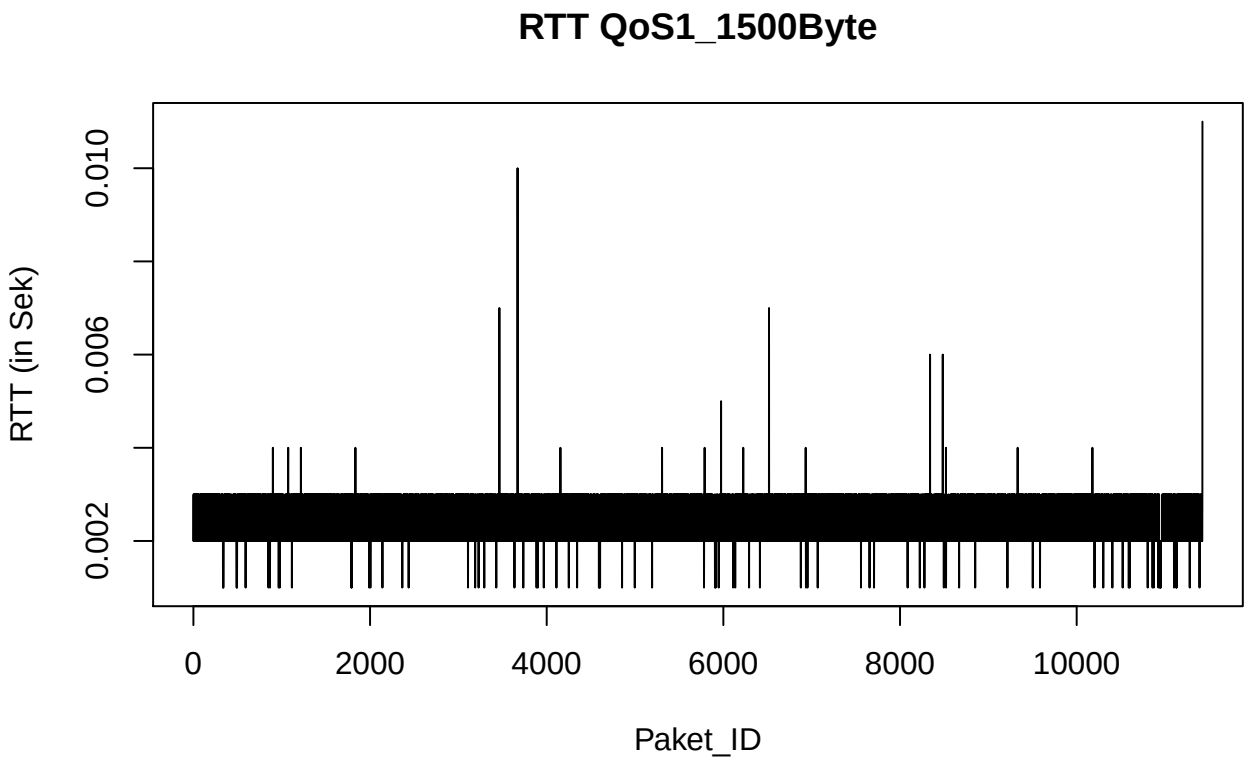

```
plot(latenzVBQoS1100Byte$id, latenzVBQoS1100Byte$rtt, type = "l", main = "RTT QoS1_100Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



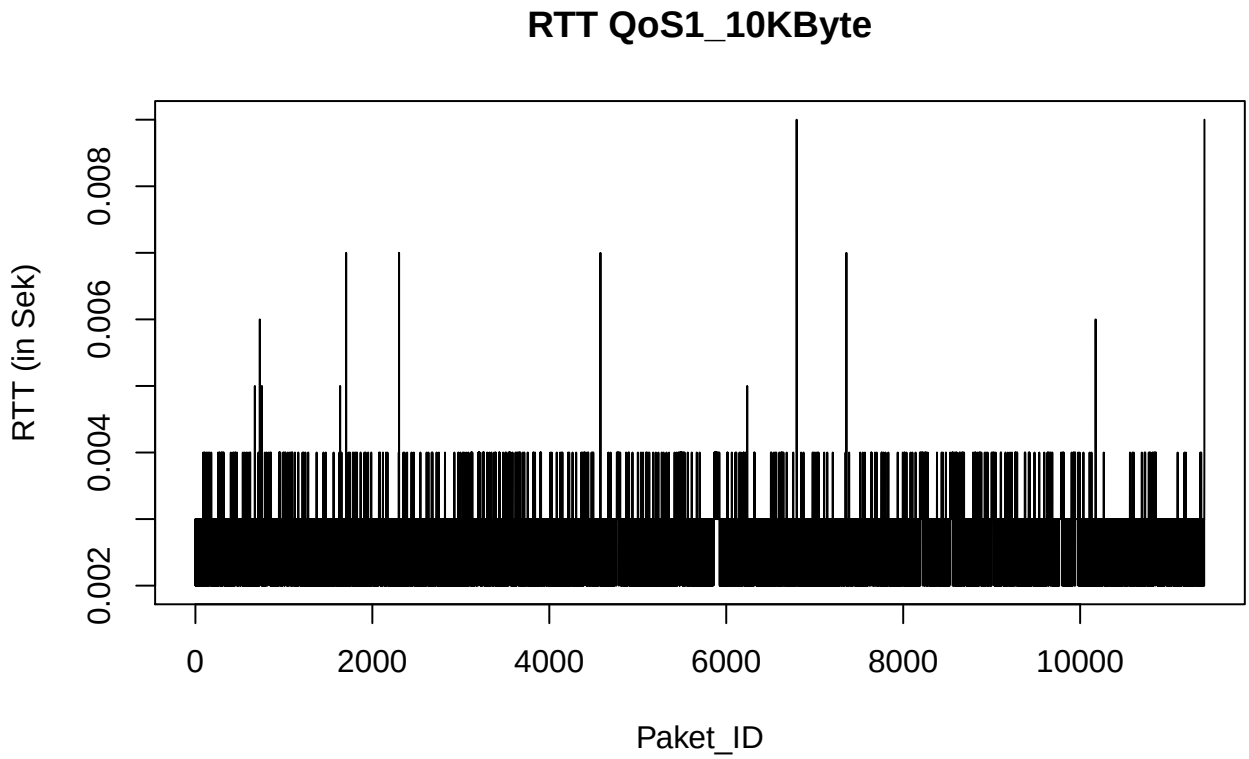

```
plot(latenzVBQoS11KByte$id, latenzVBQoS11KByte$rtt, type = "l", main = "RTT QoS1_1KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



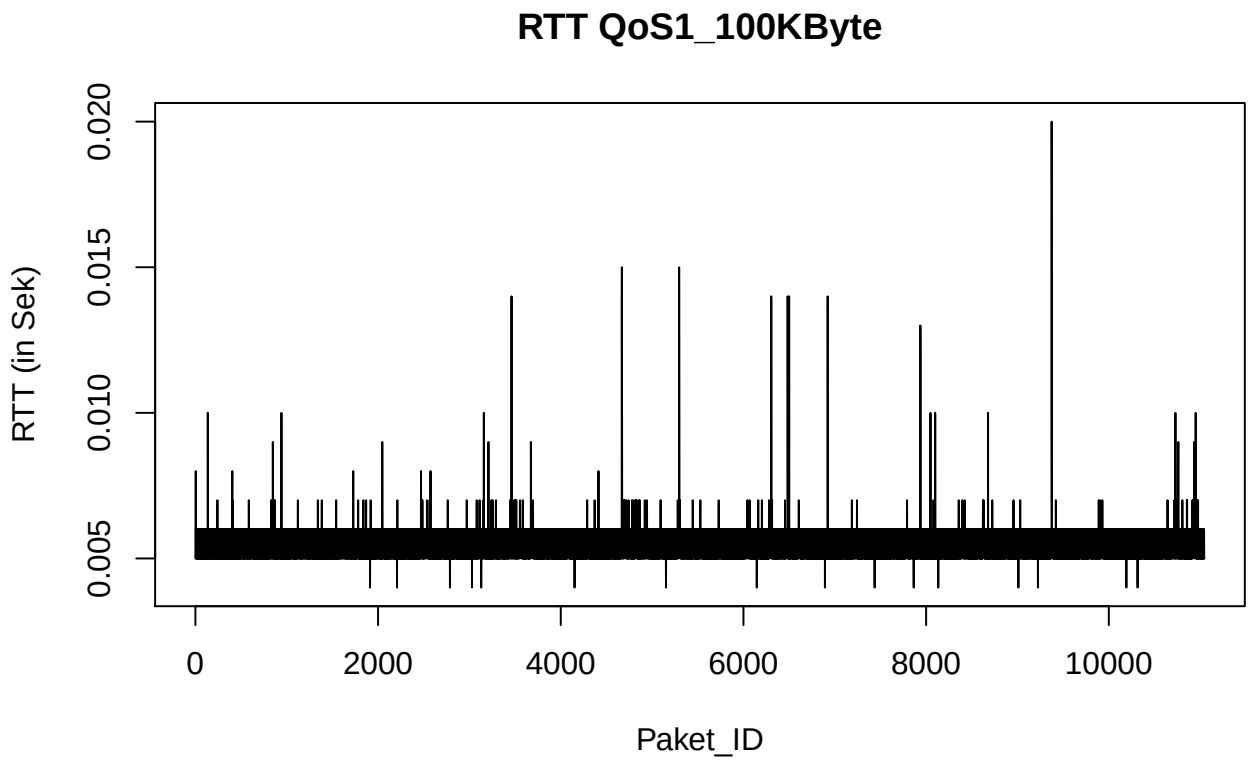

```
plot(latenzVBQoS11500Byte$id, latenzVBQoS11500Byte$rtt, type = "l", main = "RTT QoS1_1500Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



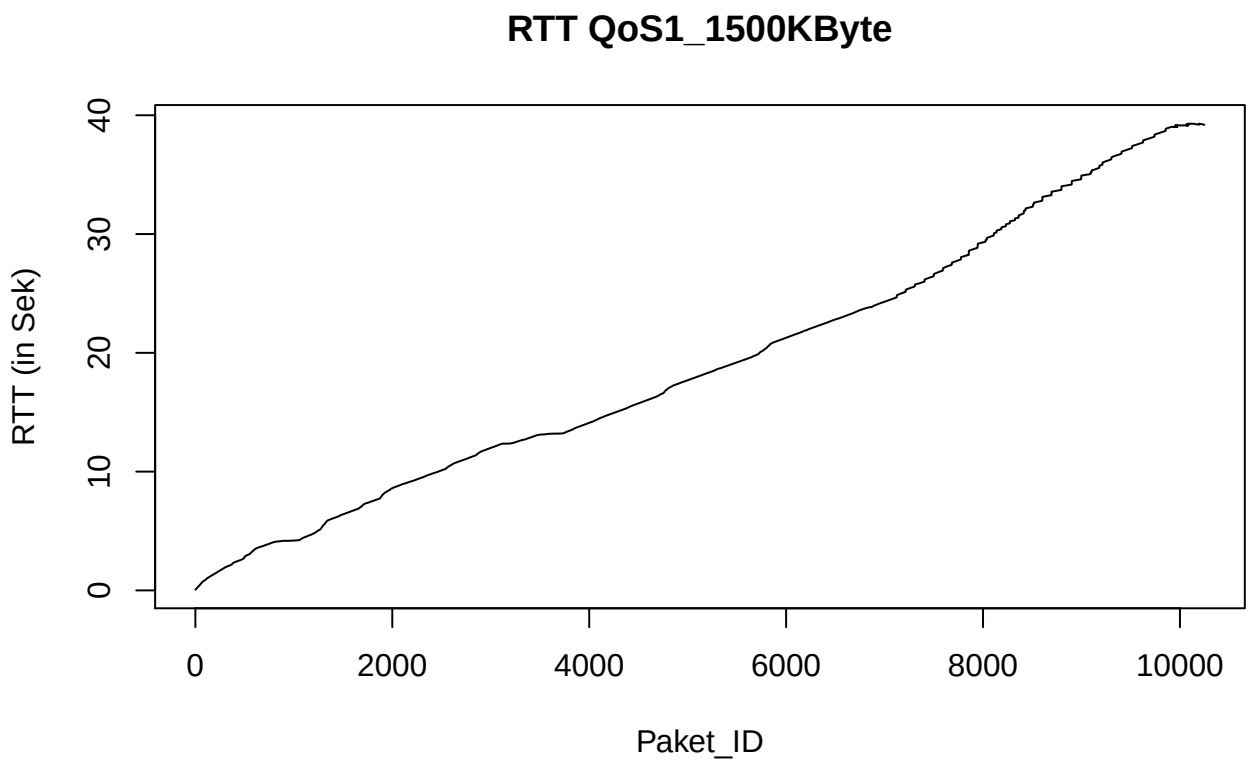

```
plot(latenzVBQoS110KByte$id, latenzVBQoS110KByte$rtt, type = "l", main = "RTT QoS1_10KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



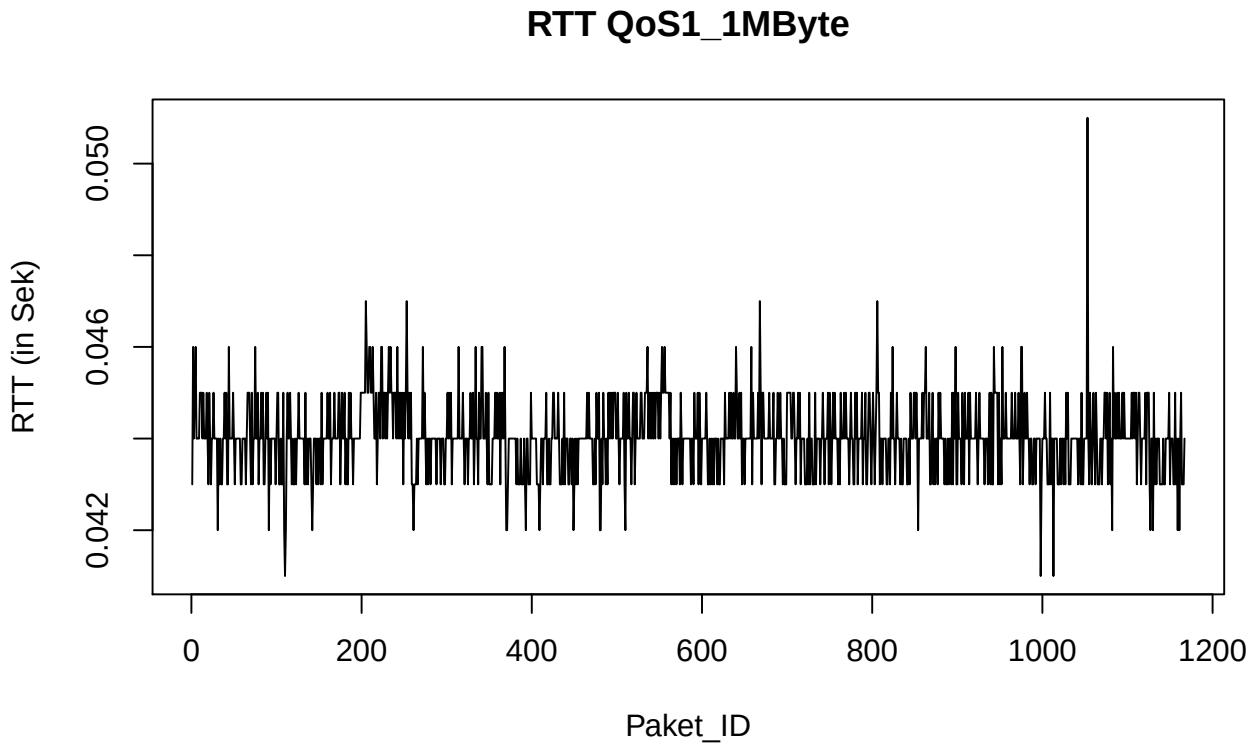

```
plot(latenzVBQoS1100KByte$id, latenzVBQoS1100KByte$rtt, type = "l", main = "RTT QoS1_100KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



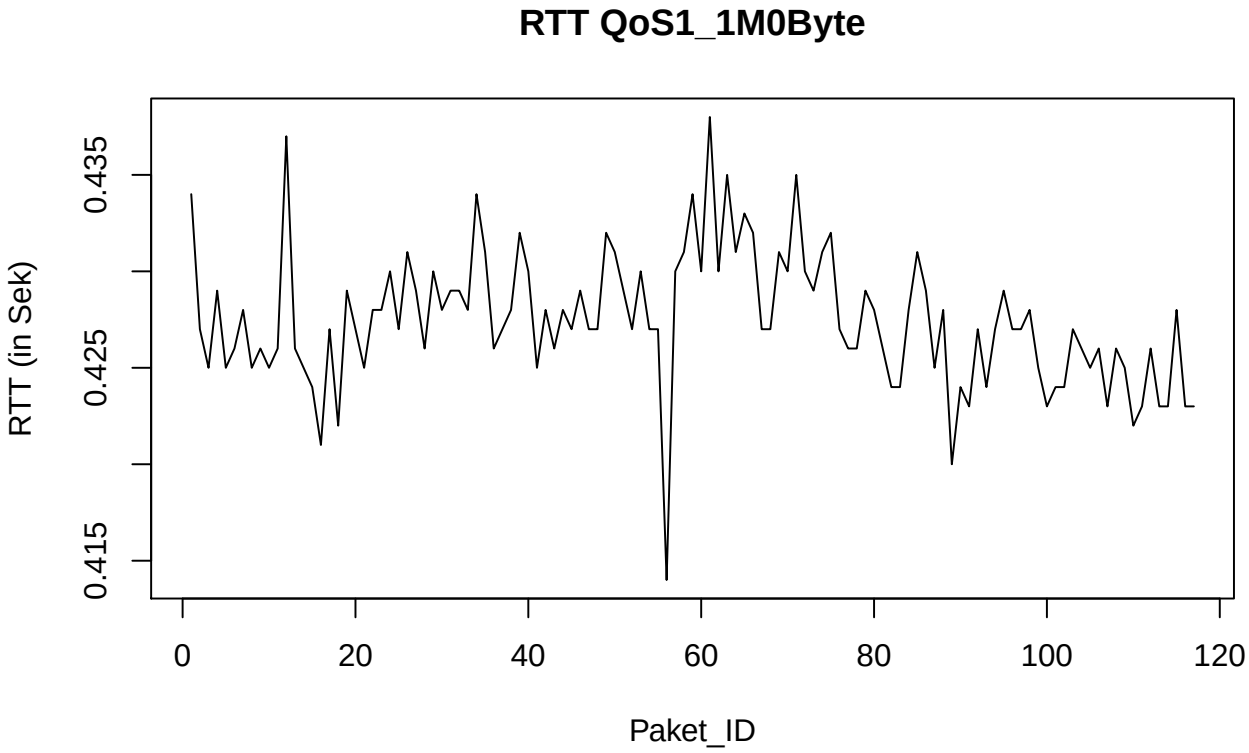

```
plot(latenzVBQoS1500KByte$id, latenzVBQoS1500KByte$rtt, type = "l", main = "RTT QoS1_1500KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



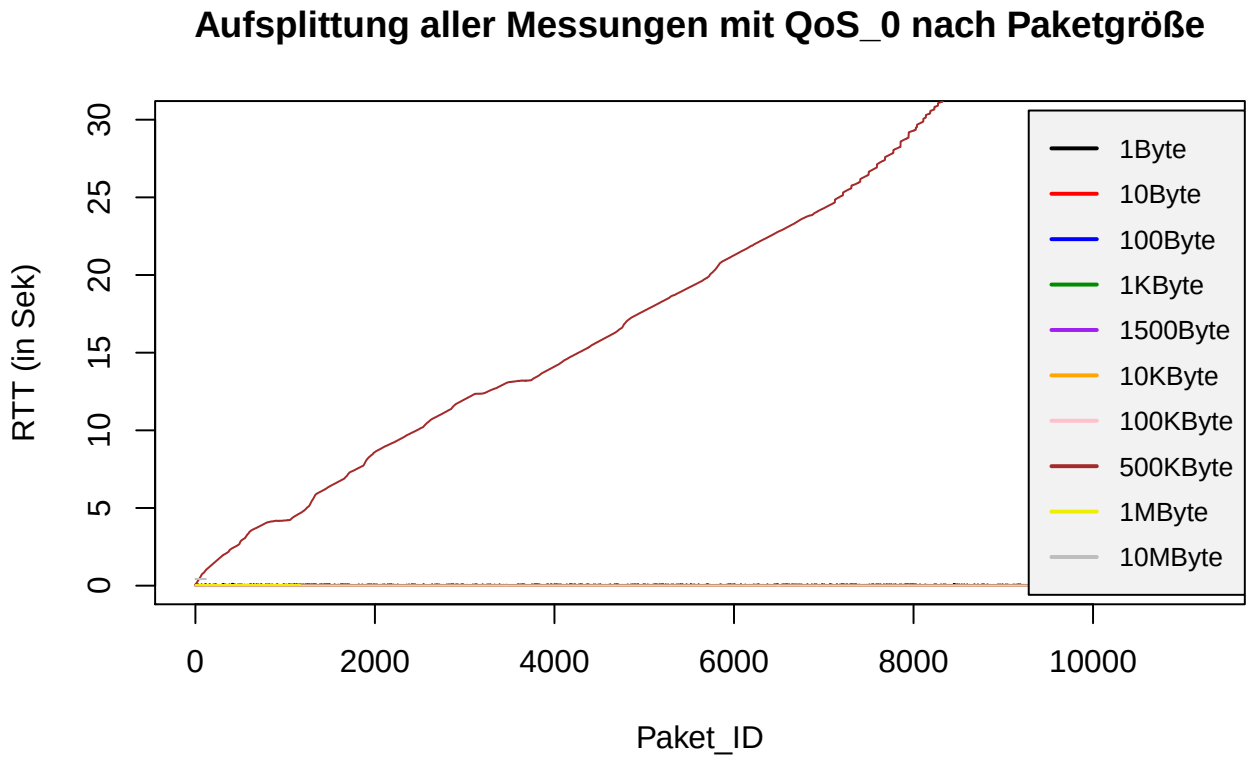

```
plot(latenzVBQoS11MByte$id, latenzVBQoS11MByte$rtt, type = "l", main = "RTT QoS1_1MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



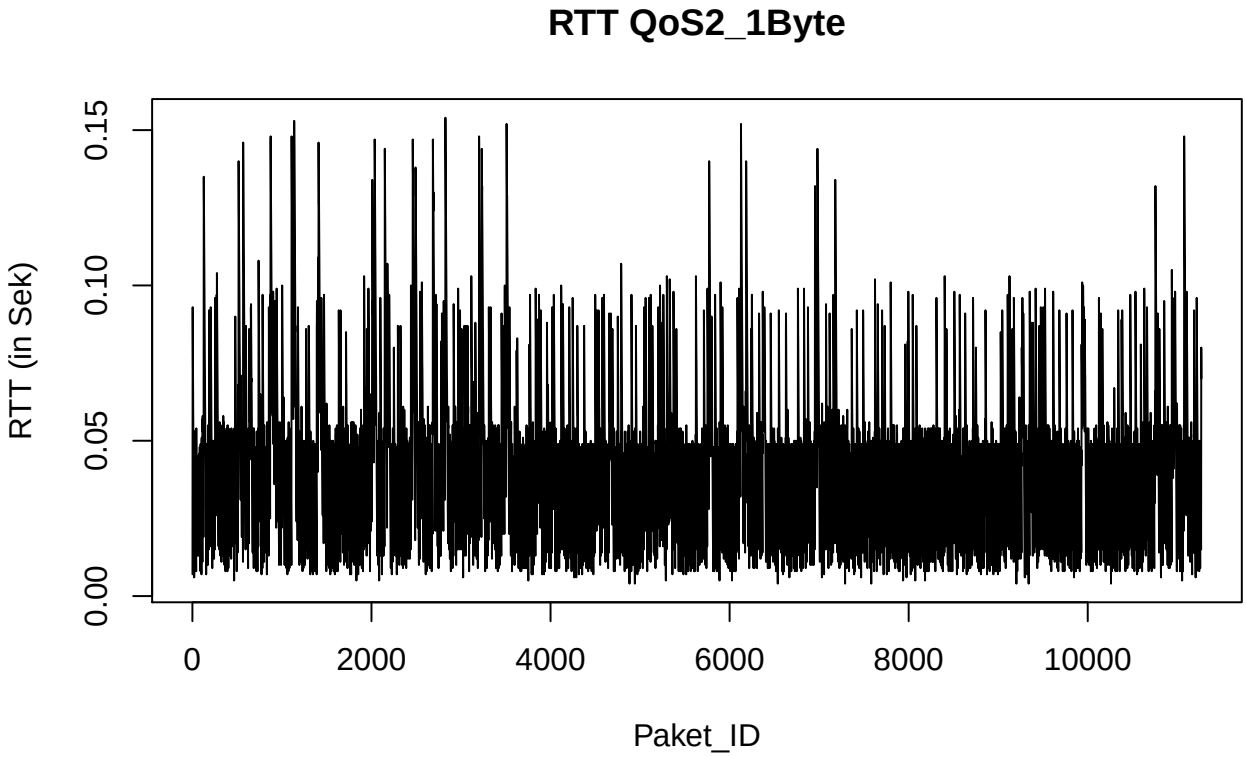

```
plot(latenzVBQoS110MByte$id, latenzVBQoS110MByte$rtt, type = "l", main = "RTT QoS1_1M0Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



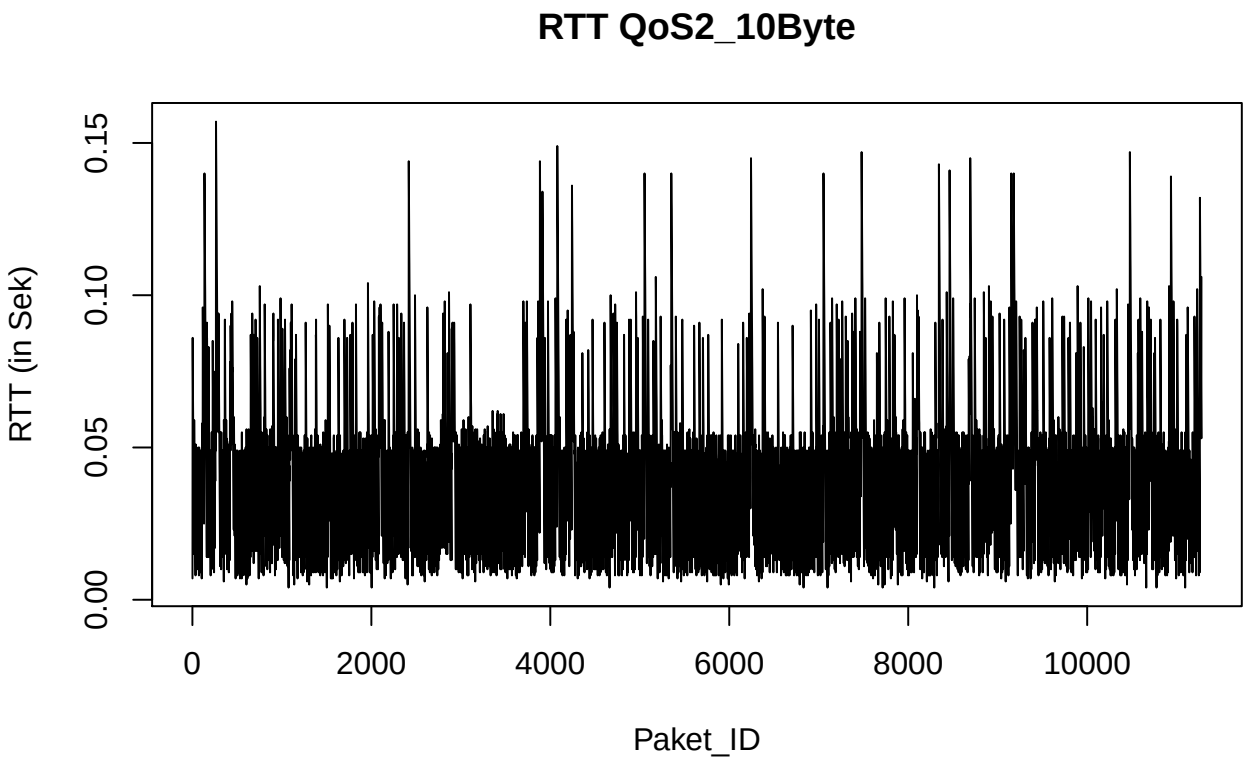

```
## QoS1 - Aufsplittung - eine Grafik!
plot(latenzVBQoS11Byte$id, latenzVBQoS11Byte$rtt, type = "l", ylim = c(0, 30), ylab = "RTT (in Sek)", xl
     main = "Aufsplittung aller Messungen mit QoS_0 nach Paketgröße")
points(latenzVBQoS110Byte$id, latenzVBQoS110Byte$rtt, col = "red", type = "l")
points(latenzVBQoS1100Byte$id, latenzVBQoS1100Byte$rtt, col = "blue", type = "l")
points(latenzVBQoS11KByte$id, latenzVBQoS11KByte$rtt, col = "green4", type = "l")
points(latenzVBQoS11500Byte$id, latenzVBQoS11500Byte$rtt, col = "purple", type = "l")
points(latenzVBQoS110KByte$id, latenzVBQoS110KByte$rtt, col = "orange",  type = "l")
points(latenzVBQoS1100KByte$id, latenzVBQoS1100KByte$rtt, col = "pink", type = "l")
points(latenzVBQoS1500KByte$id, latenzVBQoS1500KByte$rtt, col = "brown", type = "l")
points(latenzVBQoS11MByte$id, latenzVBQoS11MByte$rtt, col = "yellow2", type = "l")
points(latenzVBQoS110MByte$id, latenzVBQoS110MByte$rtt, col = "gray", type = "l")


legend("right", c("1Byte", "10Byte", "100Byte", "1KByte", "1500Byte", "10KByte", "100KByte", "500KByte"
       cex = 0.8,
       col = c("black", "red", "blue", "green4", "purple", "orange", "pink", "brown", "yellow2", "gray"]
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```



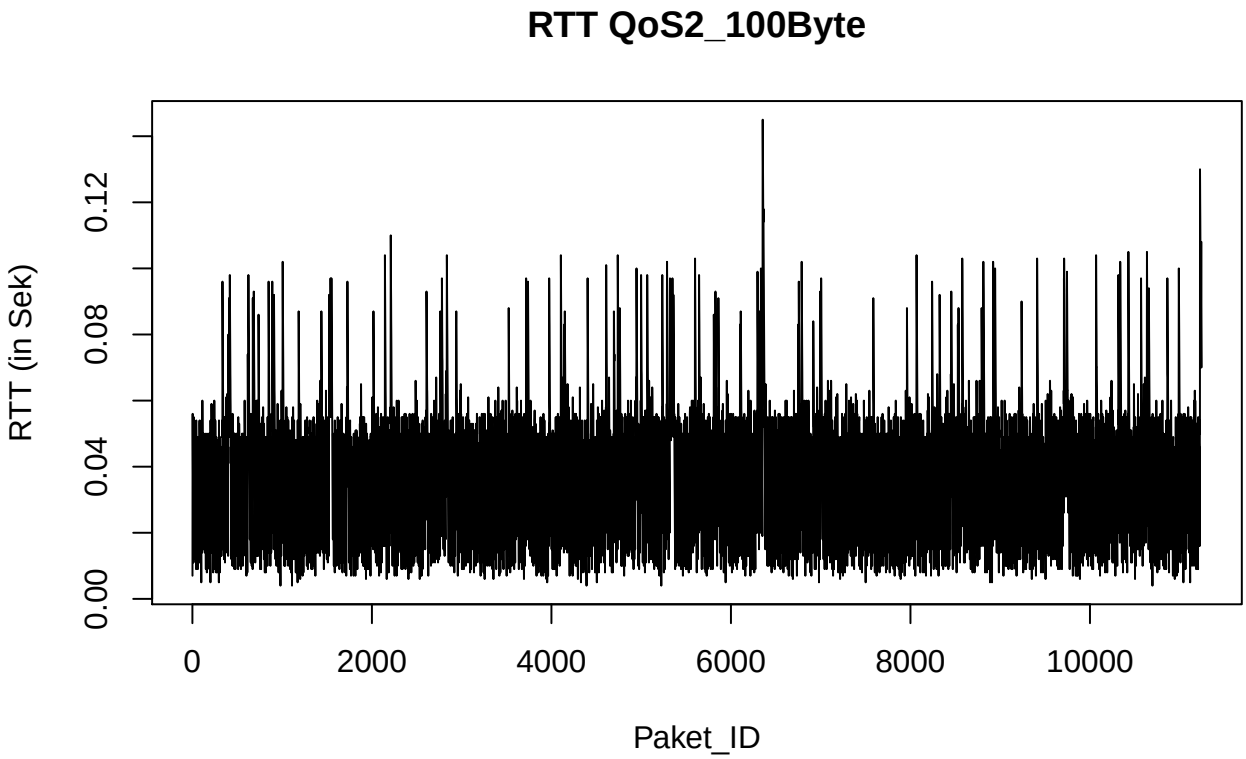

```
latenzVBQoS21Byte<-latenzVBQoS2[latenzVBQoS2$Size == "1Byte",]
latenzVBQoS210Byte<-latenzVBQoS2[latenzVBQoS2$Size == "10Byte",]
latenzVBQoS2100Byte<-latenzVBQoS2[latenzVBQoS2$Size == "100Byte",]
latenzVBQoS21KByte<-latenzVBQoS2[latenzVBQoS2$Size == "1KByte",]
latenzVBQoS21500Byte<-latenzVBQoS2[latenzVBQoS2$Size == "1500Byte",]
latenzVBQoS210KByte<-latenzVBQoS2[latenzVBQoS2$Size == "10KByte",]
latenzVBQoS2100KByte<-latenzVBQoS2[latenzVBQoS2$Size == "100KByte",]
latenzVBQoS2500KByte<-latenzVBQoS2[latenzVBQoS2$Size == "500KByte",]
latenzVBQoS21MByte<-latenzVBQoS2[latenzVBQoS2$Size == "1MByte",]
latenzVBQoS210MByte<-latenzVBQoS2[latenzVBQoS2$Size == "10MByte",]


plot(latenzVBQoS21Byte$id, latenzVBQoS21Byte$rtt, type = "l", main = "RTT QoS2_1Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



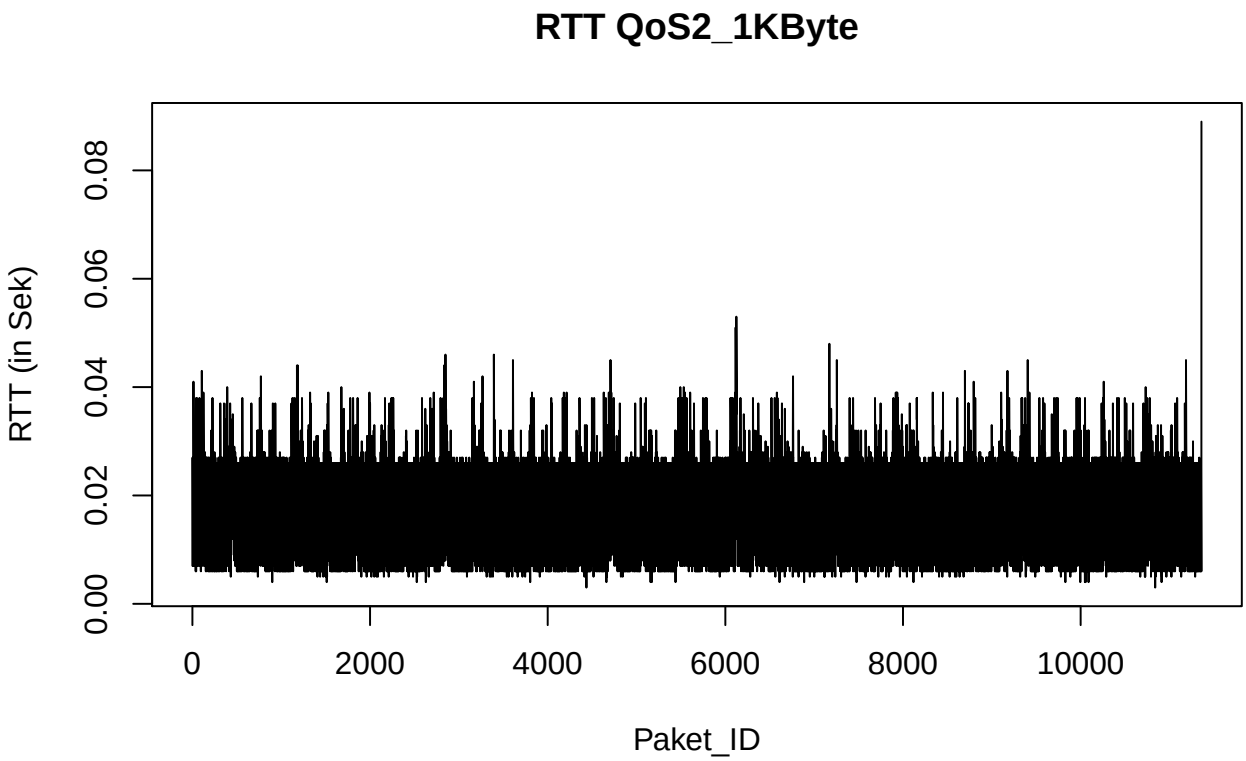

```
plot(latenzVBQoS210Byte$id, latenzVBQoS210Byte$rtt, type = "l", main = "RTT QoS2_10Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



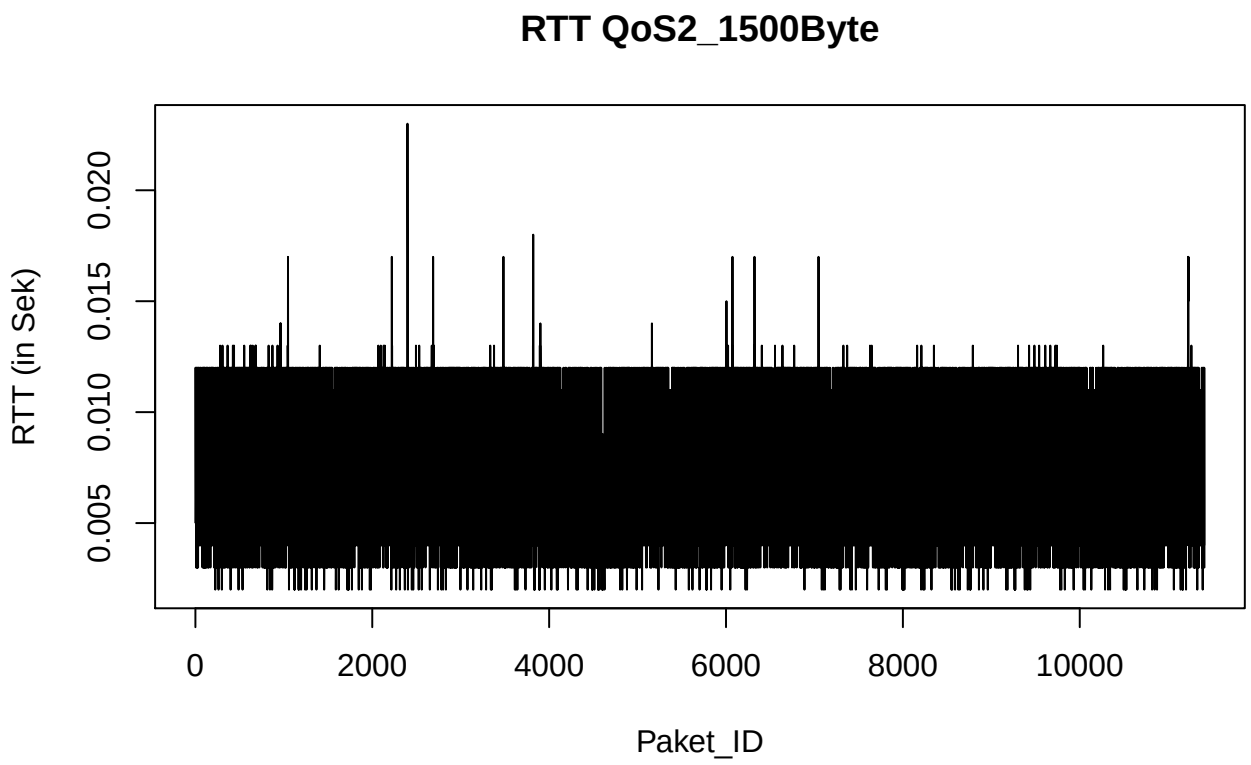

```
plot(latenzVBQoS2100Byte$id, latenzVBQoS2100Byte$rtt, type = "l", main = "RTT QoS2_100Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



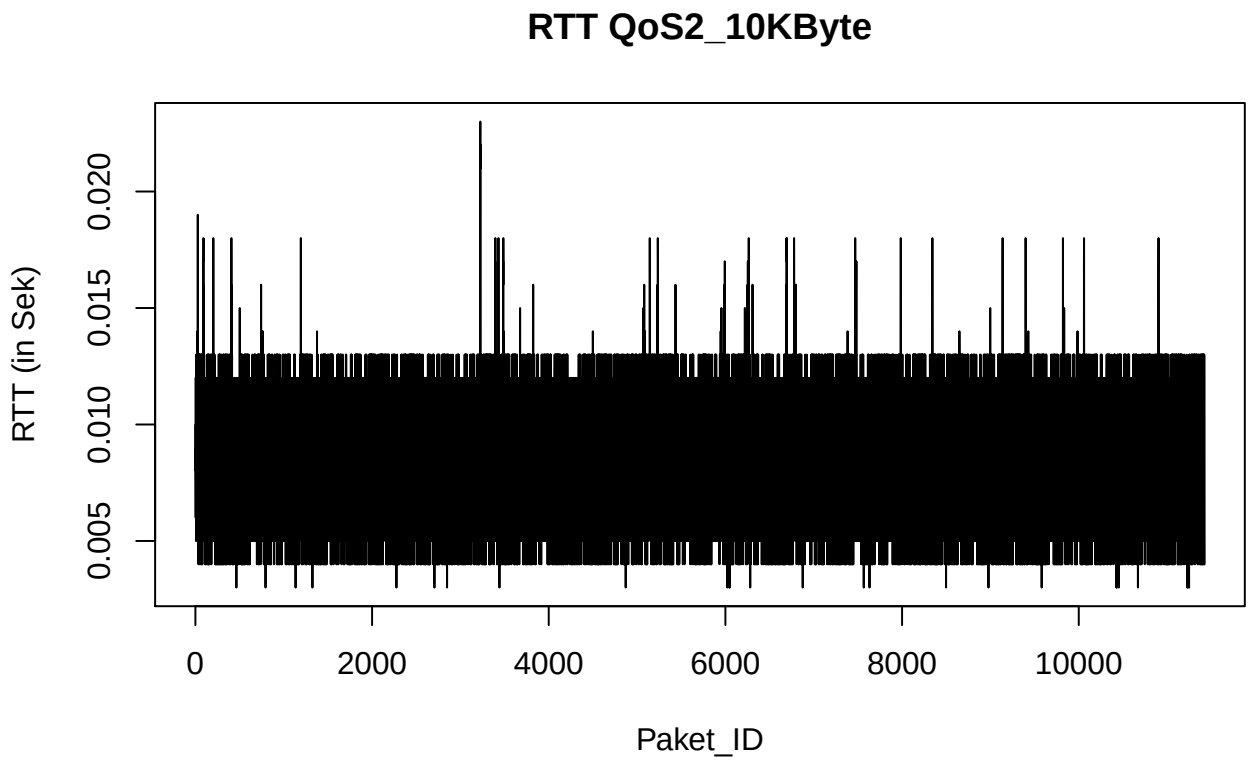

```
plot(latenzVBQoS21KByte$id, latenzVBQoS21KByte$rtt, type = "l", main = "RTT QoS2_1KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



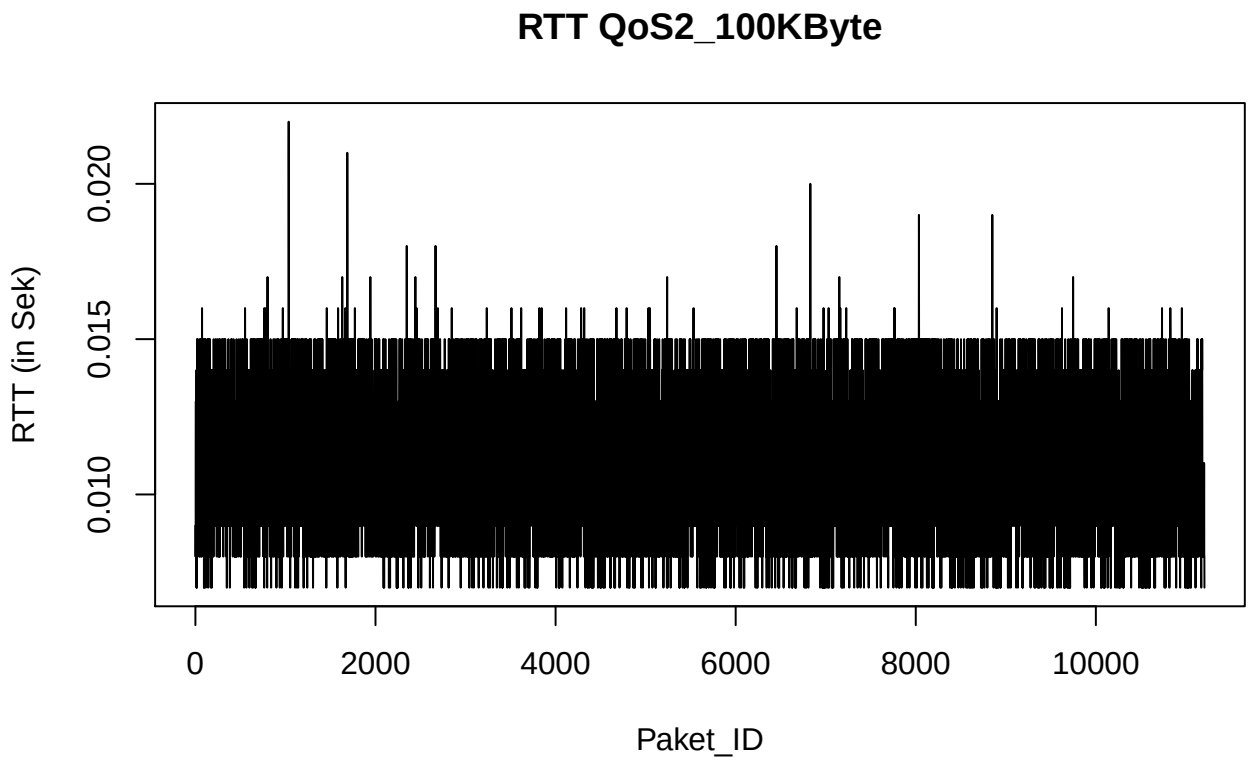

```
plot(latenzVBQoS21500Byte$id, latenzVBQoS21500Byte$rtt, type = "l", main = "RTT QoS2_1500Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



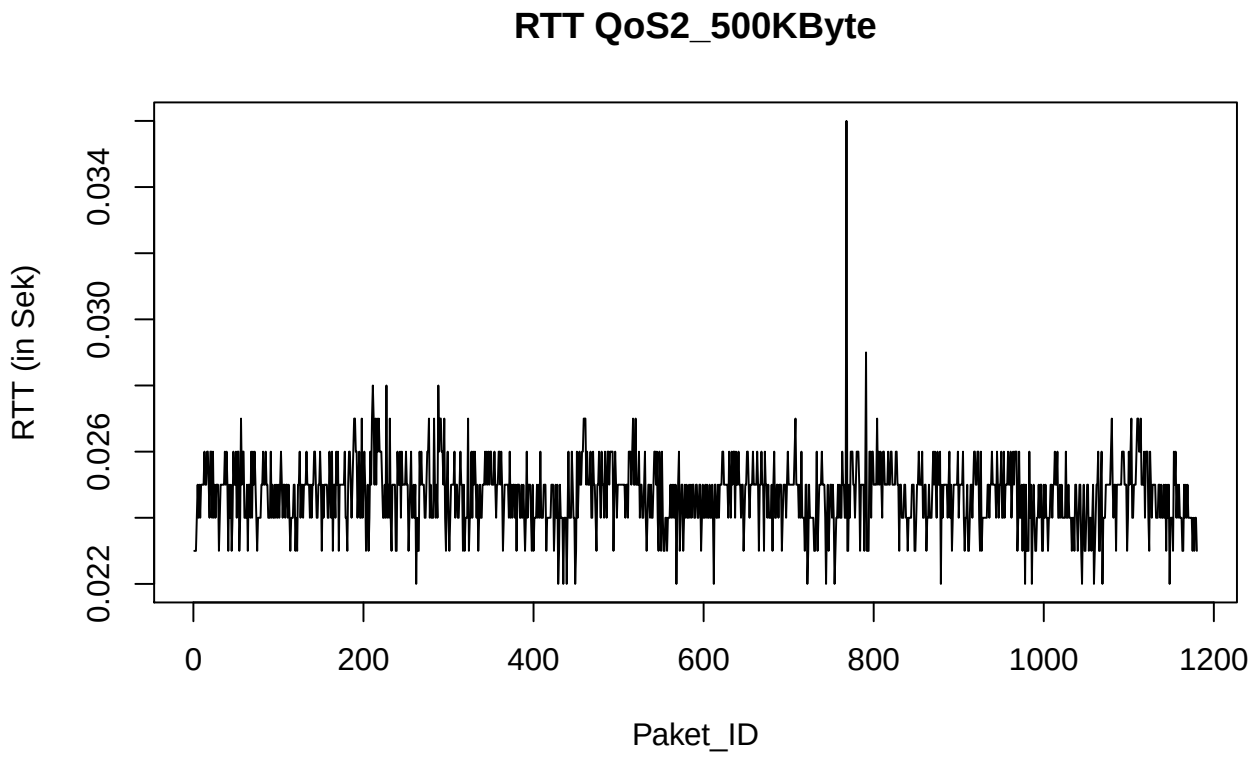

```
plot(latenzVBQoS210KByte$id, latenzVBQoS210KByte$rtt, type = "l", main = "RTT QoS2_10KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



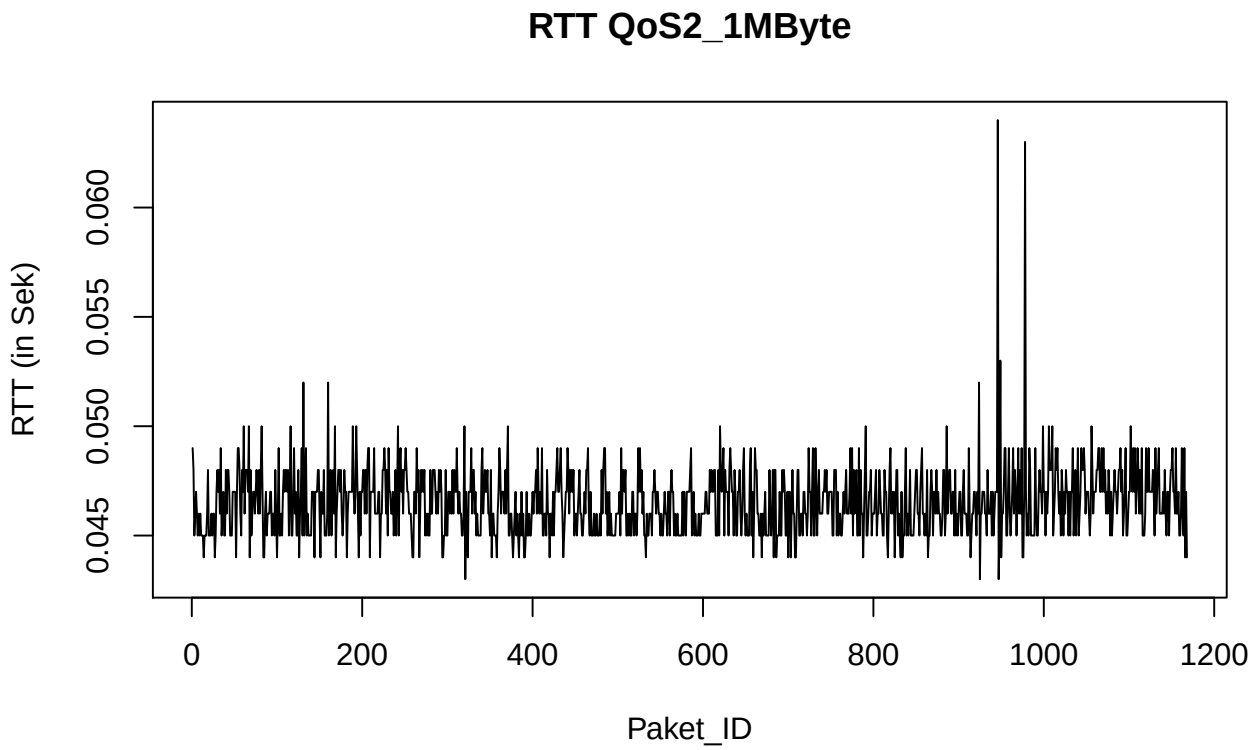

```
plot(latenzVBQoS2100KByte$id, latenzVBQoS2100KByte$rtt, type = "l", main = "RTT QoS2_100KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



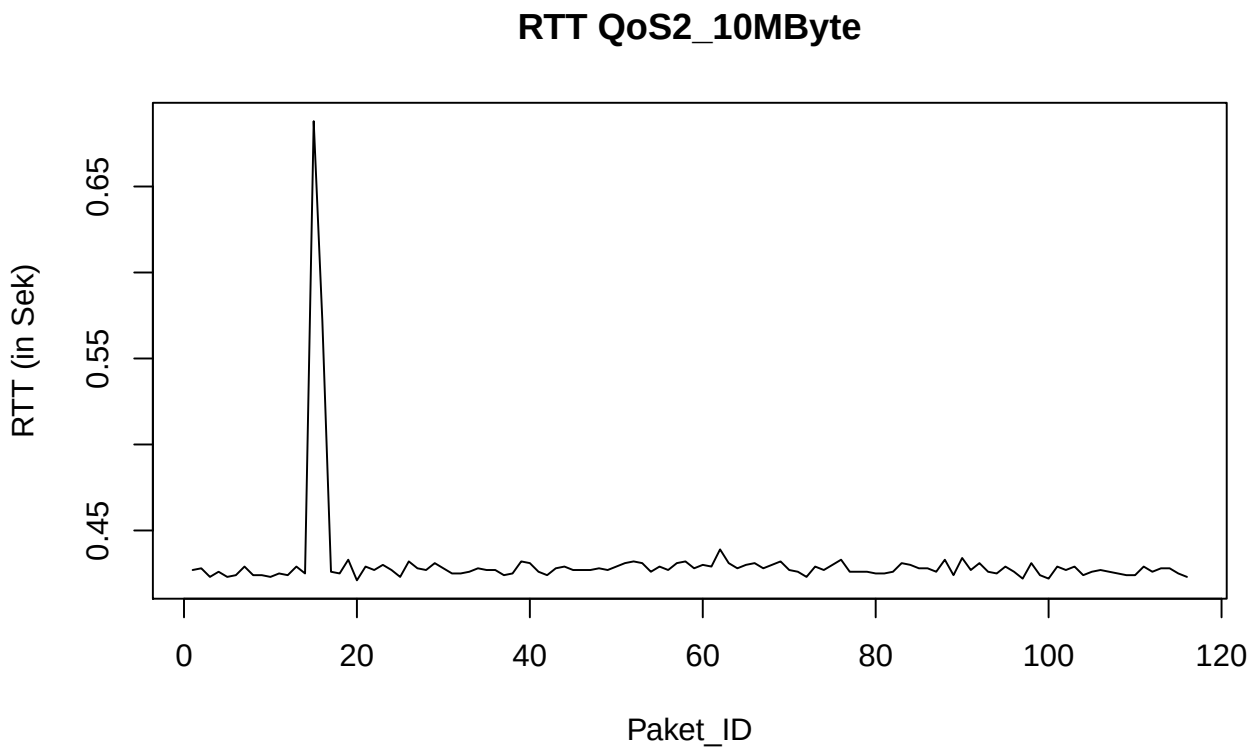

```
plot(latenzVBQoS2500KByte$id, latenzVBQoS2500KByte$rtt, type = "l", main = "RTT QoS2_500KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```




```
plot(latenzVBQoS21MByte$id, latenzVBQoS21MByte$rtt, type = "l", main = "RTT QoS2_1MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```




```
plot(latenzVBQoS210MByte$id, latenzVBQoS210MByte$rtt, type = "l", main = "RTT QoS2_10MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```




```
## QoS2 - Aufsplittung - eine Grafik!
plot(latenzVBQoS21Byte$id, latenzVBQoS21Byte$rtt, type = "l", ylim = c(0, 0.18), ylab = "RTT (in Sek)",
     main = "Aufsplittung aller Messungen mit QoS_0 nach Paketgröße")
```

```
points(latenzVBQoS210Byte$id, latenzVBQoS210Byte$rtt, col = "red", type = "l")
points(latenzVBQoS2100Byte$id, latenzVBQoS2100Byte$rtt, col = "blue", type = "l")
points(latenzVBQoS21KByte$id, latenzVBQoS21KByte$rtt, col = "green4", type = "l")
points(latenzVBQoS21500Byte$id, latenzVBQoS21500Byte$rtt, col = "purple", type = "l")
points(latenzVBQoS210KByte$id, latenzVBQoS210KByte$rtt, col = "orange",  type = "l")
points(latenzVBQoS2100KByte$id, latenzVBQoS2100KByte$rtt, col = "pink", type = "l")
points(latenzVBQoS2500KByte$id, latenzVBQoS2500KByte$rtt, col = "brown", type = "l")
points(latenzVBQoS21MByte$id, latenzVBQoS21MByte$rtt, col = "yellow2", type = "l")
points(latenzVBQoS210MByte$id, latenzVBQoS210MByte$rtt, col = "gray", type = "l")


legend("right", c("1Byte", "10Byte", "100Byte", "1KByte", "1500Byte", "10KByte", "100KByte", "500KByte"
       cex = 0.8,
       col = c("black", "red", "blue", "green4", "purple", "orange", "pink", "brown", "yellow2", "gray"
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```