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")
```

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

```
latenzVB[,5]<-NA
latenzVB<-latenzVB[latenzVB$Size != "500KByte",] # ausfiltern der 500KByte Daten
rtt_zero<-latenzVB[latenzVB$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:241462      Length:241462
#>  1st Qu.:2018-05-23 22:18:10.25   Class :character   Class :character
#>  Median :2018-05-23 22:26:28.65   Mode  :character   Mode  :character
#>  Mean   :2018-05-23 22:29:05.01
#>  3rd Qu.:2018-05-23 22:40:09.93
#>  Max.   :2018-05-23 23:25:12.82
#>      Min             Speed              rec
#>  Length:241462      Mode:logical   Min.   :2018-05-23 22:09:04.22
#>  Class :character   NA's:241462    1st Qu.:2018-05-23 22:18:10.25
#>  Mode  :character                  Median :2018-05-23 22:26:28.65
#>                                    Mean   :2018-05-23 22:29:05.03
#>                                    3rd Qu.:2018-05-23 22:40:09.94
#>                                    Max.   :2018-05-23 23:25:13.25
#>    r_newid               rtt                   id            Byte
#>  Length:241462      Min.   :0.0009999   Min.   :    1   Length:241462
#>  Class :character   1st Qu.:0.0020001   1st Qu.: 2693   Class :character
#>  Mode  :character   Median :0.0060000   Median : 5568   Mode  :character
#>                     Mean   :0.0151744   Mean   : 5579
#>                     3rd Qu.:0.0200000   3rd Qu.: 8444
```

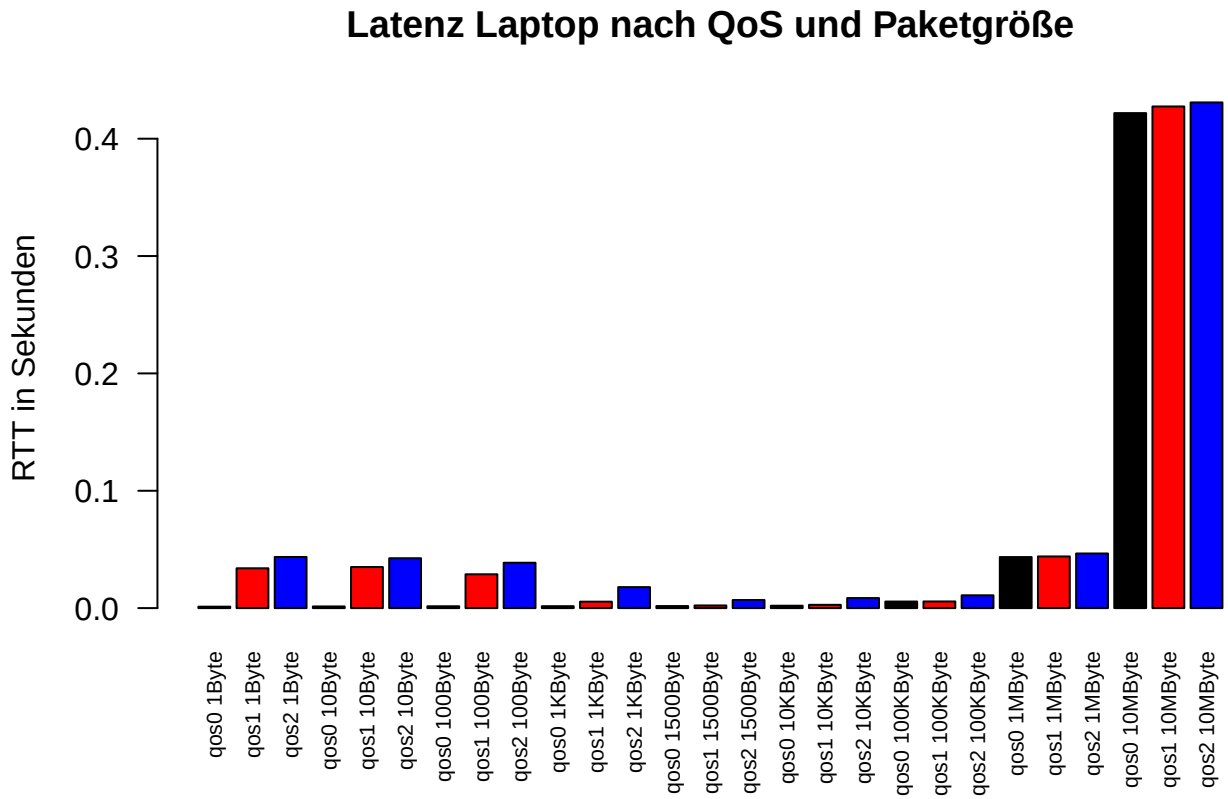
```
#>                     Max.   :0.6880002   Max.   :11424
```

Laptop (volle bandbreite - VB)

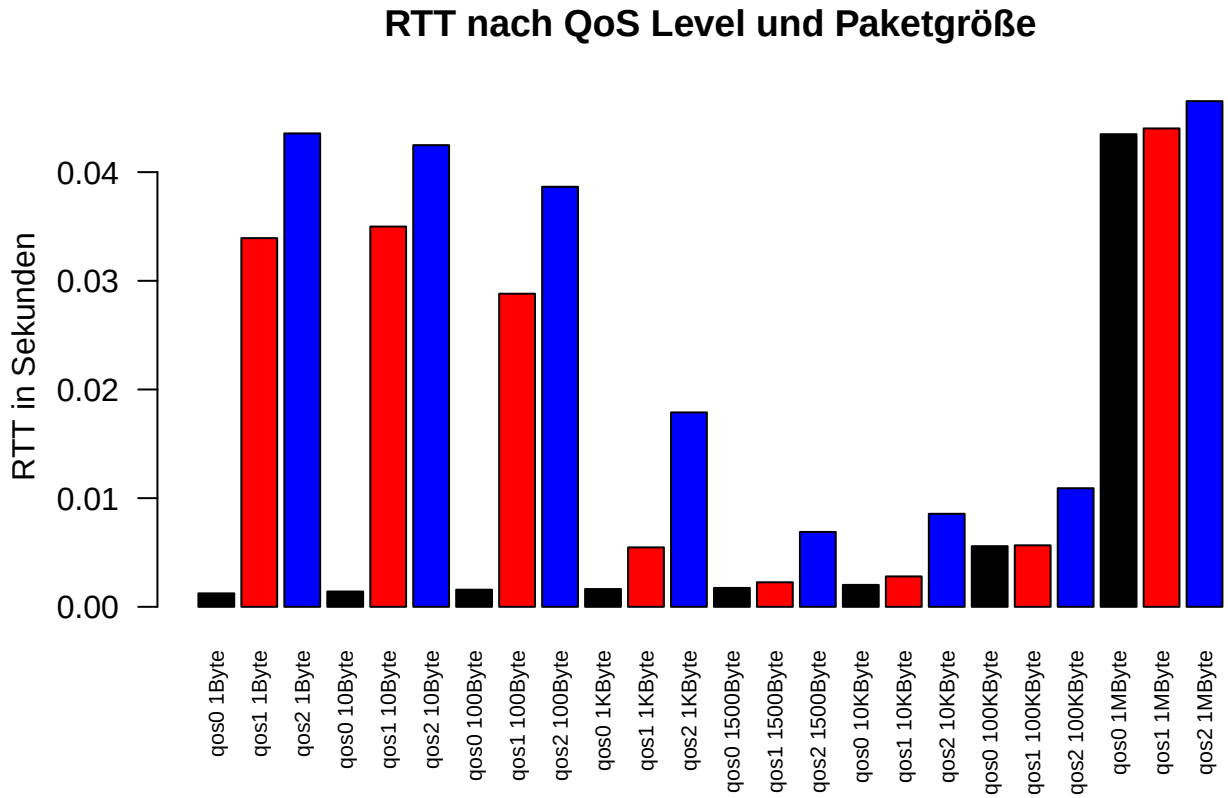
```
logsVBAgg <- aggregate(latenzVB$rtt ~ latenzVB$QoS+latenzVB$Size+latenzVB$Byte, latenzVB, mean)
logsVBAgg$`latenzVB$Byte`<-as.numeric(logsVBAgg$`latenzVB$Byte`)
logsVBAgg<-logsVBAgg[order(logsVBAgg$`latenzVB$Byte`),]


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
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

```
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", col = c("black", "red
```

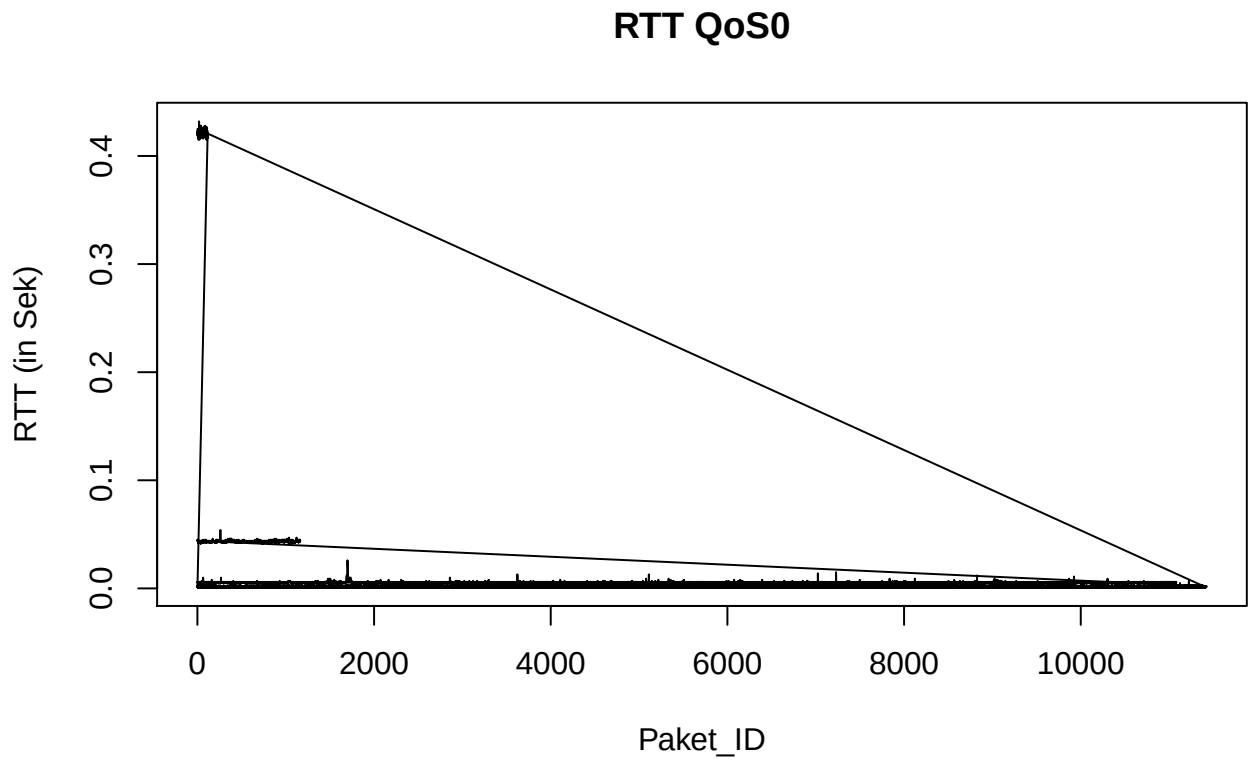



```
logsVBAgg_no500_no100M<-logsVBAgg[logsVBAgg$`latenzVB$Byte`!=500000 & logsVBAgg$`latenzVB$Byte`!= 10000
logsVBAgg_no500_no100M$Names <- paste(logsVBAgg_no500_no100M$`latenzVB$QoS`, logsVBAgg_no500_no100M$`la
barplot(logsVBAgg_no500_no100M$`latenzVB$rtt`, main = "RTT nach QoS Level und Paketgröße", col = c("bla
```

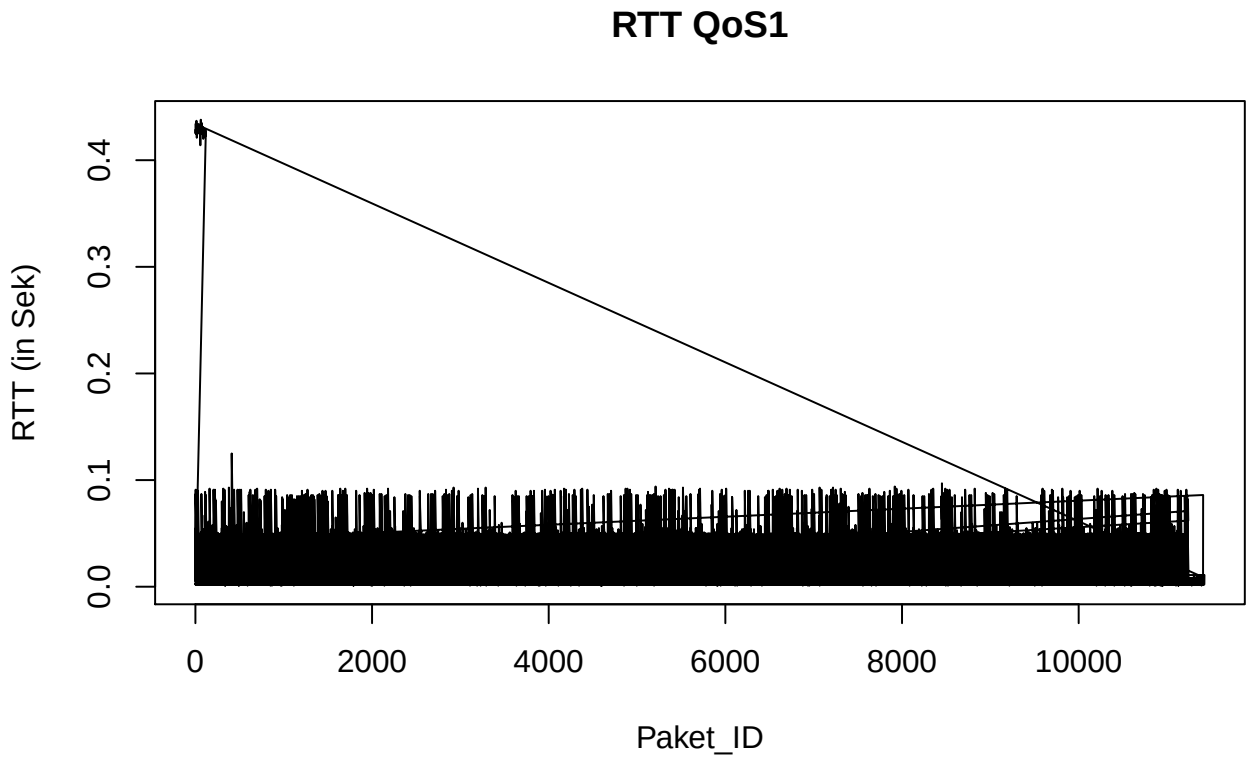


Im nächsten Schritt wird die statistische Abhngigkeit der rtt von QoS und Größe (Byte) untersucht. Im Falle einer einfachen linearen Regression sind nur qos2 und hohe Byte Zahlen signifikant.

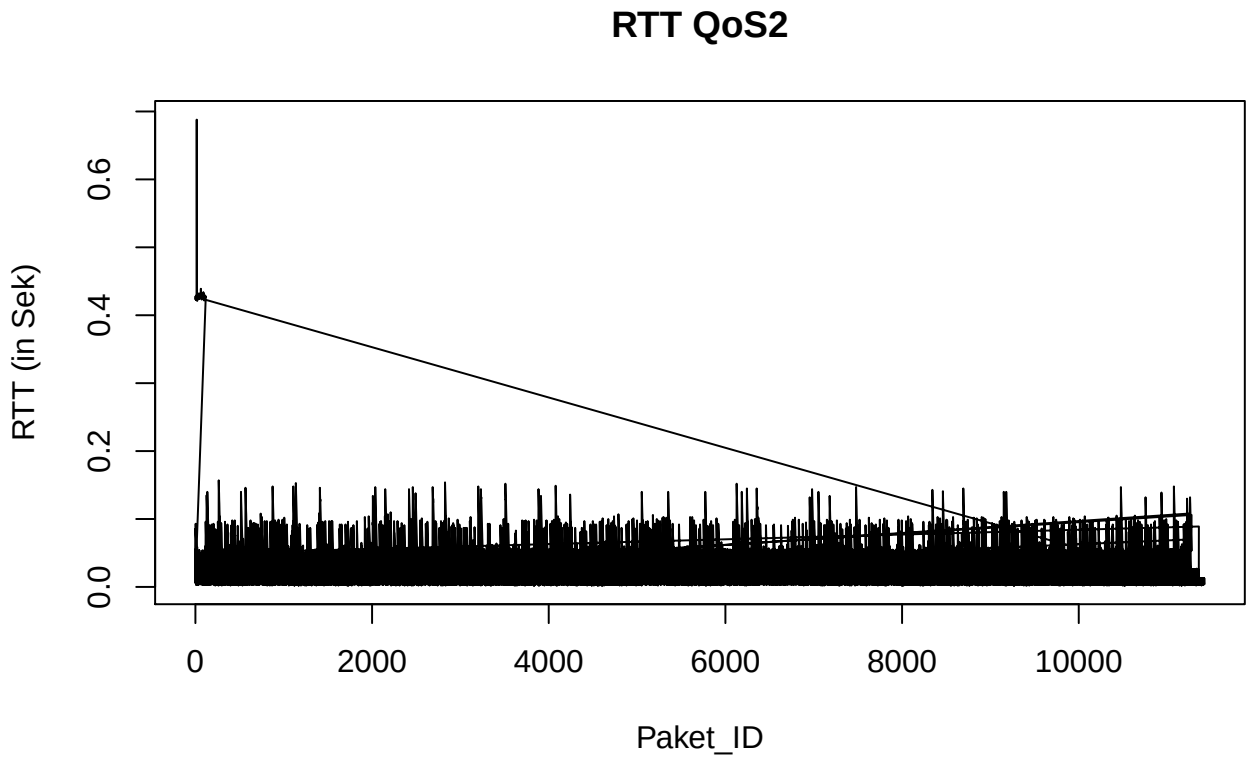
```
######################
# Aufteilen nach QoS #
######################

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")
```

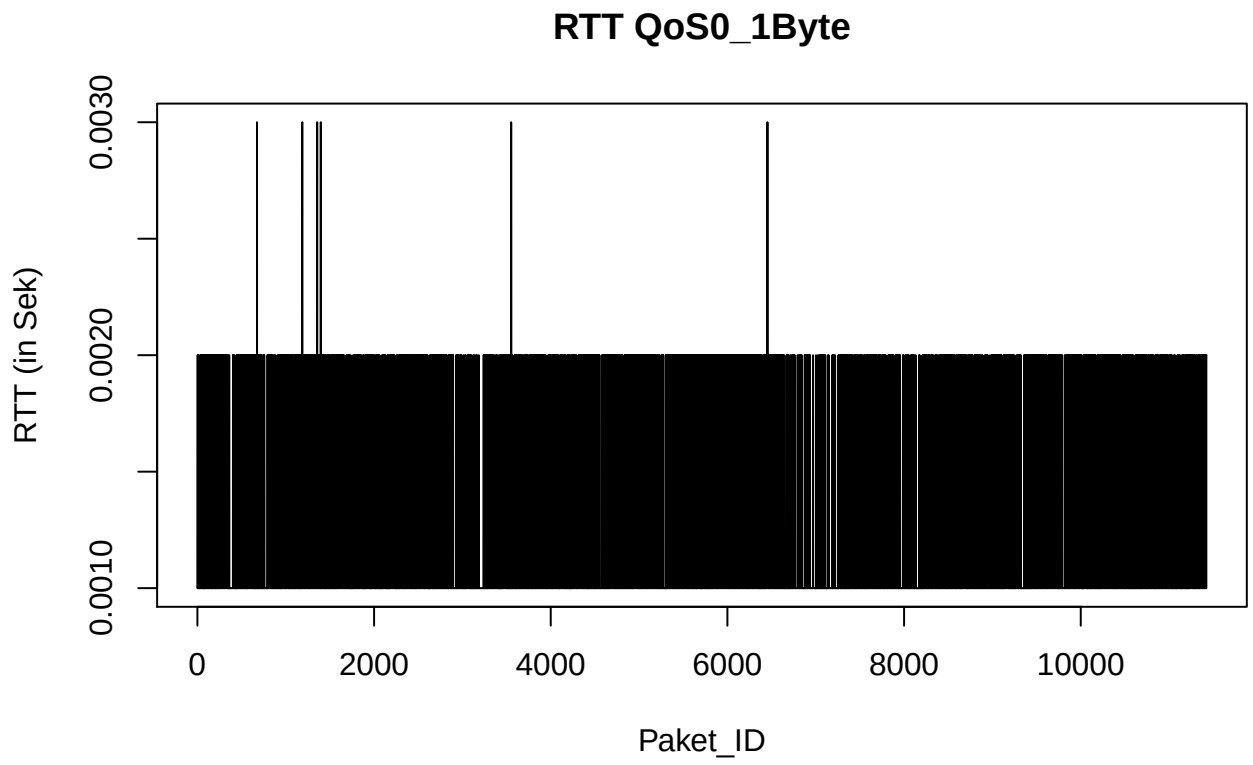



```
plot(latenzVBQoS2$id, latenzVBQoS2$rtt, type = "l", main = "RTT QoS2",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```

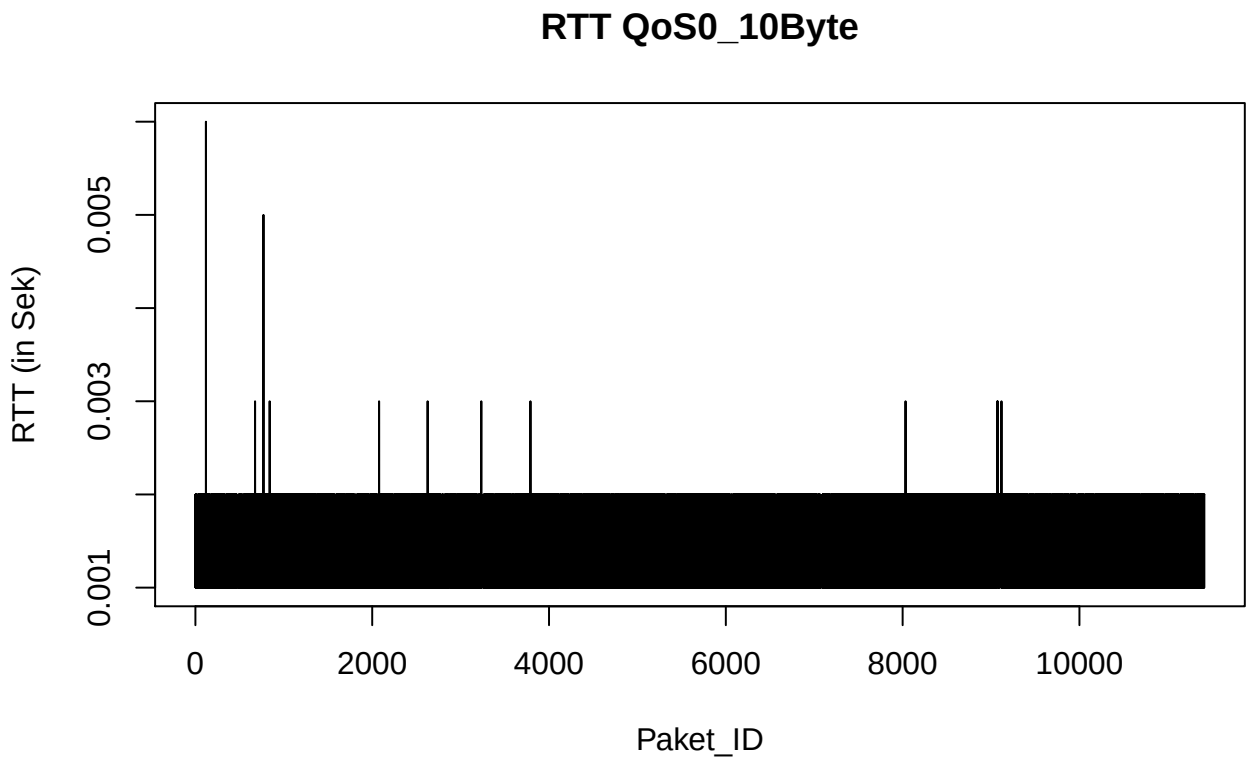



```
########################################
# QoS_0 Aufteilen nach Payload/ Größe #
########################################
```

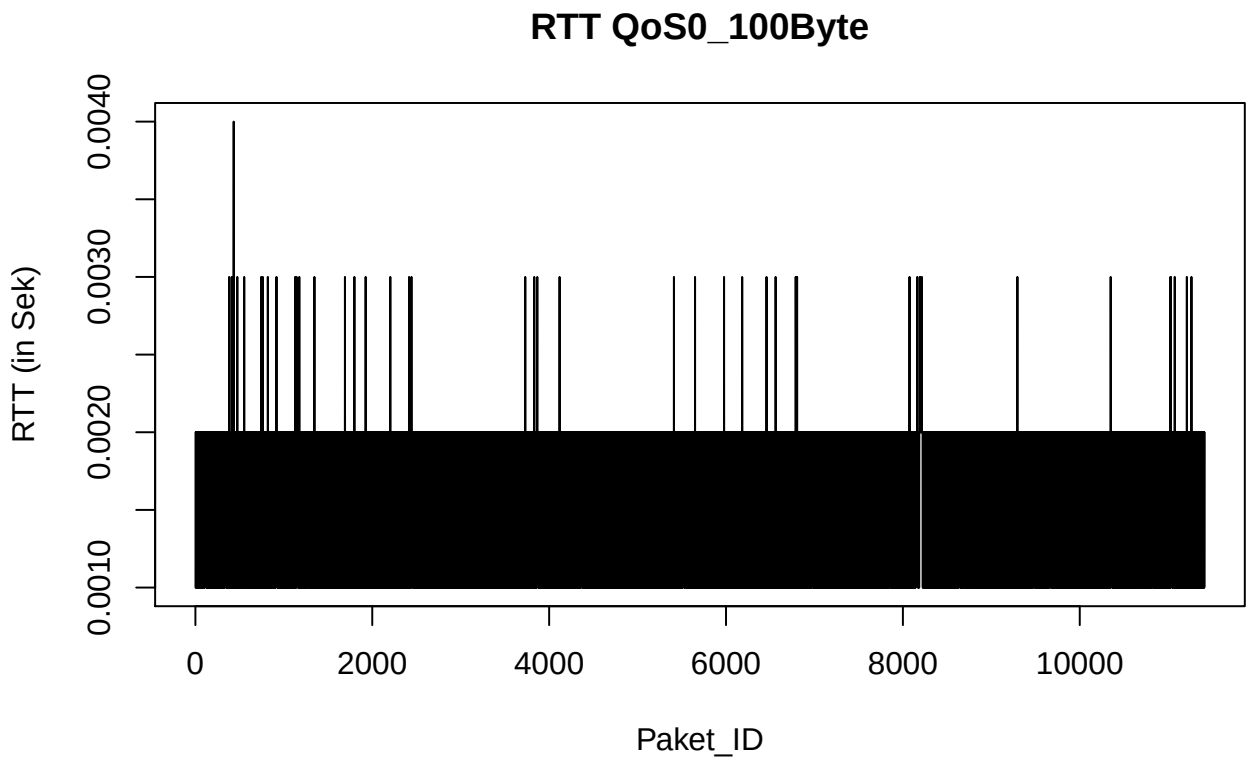
```
latenzVBQoS01Byte<-latenzVBQoS0[latenzVBQoS0$Size == "1Byte",]
latenzVBQoS010Byte<-latenzVBQoS0[latenzVBQoS0$Size == "10Byte",]
latenzVBQoS0100Byte<-latenzVBQoS0[latenzVBQoS0$Size == "100Byte",]
latenzVBQoS01KByte<-latenzVBQoS0[latenzVBQoS0$Size == "1KByte",]
latenzVBQoS01500Byte<-latenzVBQoS0[latenzVBQoS0$Size == "1500Byte",]
latenzVBQoS010KByte<-latenzVBQoS0[latenzVBQoS0$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")
```



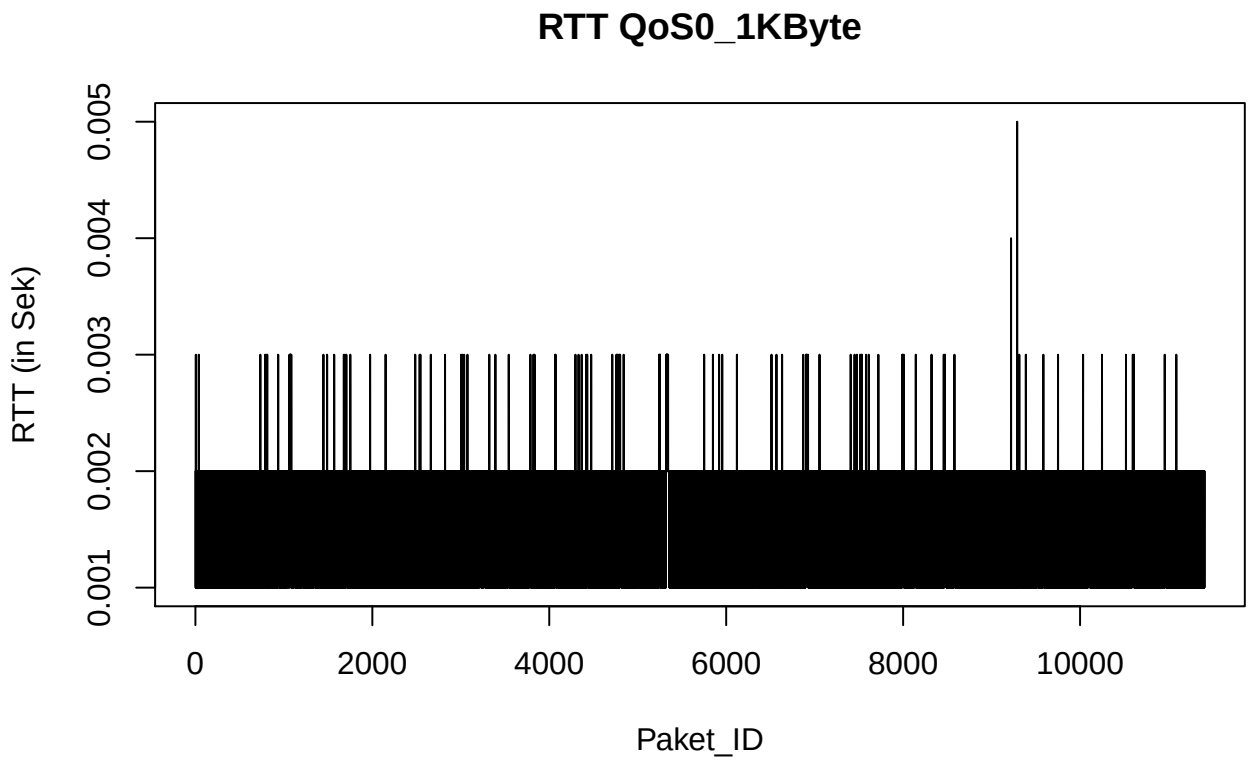

```
plot(latenzVBQoS010Byte$id, latenzVBQoS010Byte$rtt, type = "l", main = "RTT QoS0_10Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



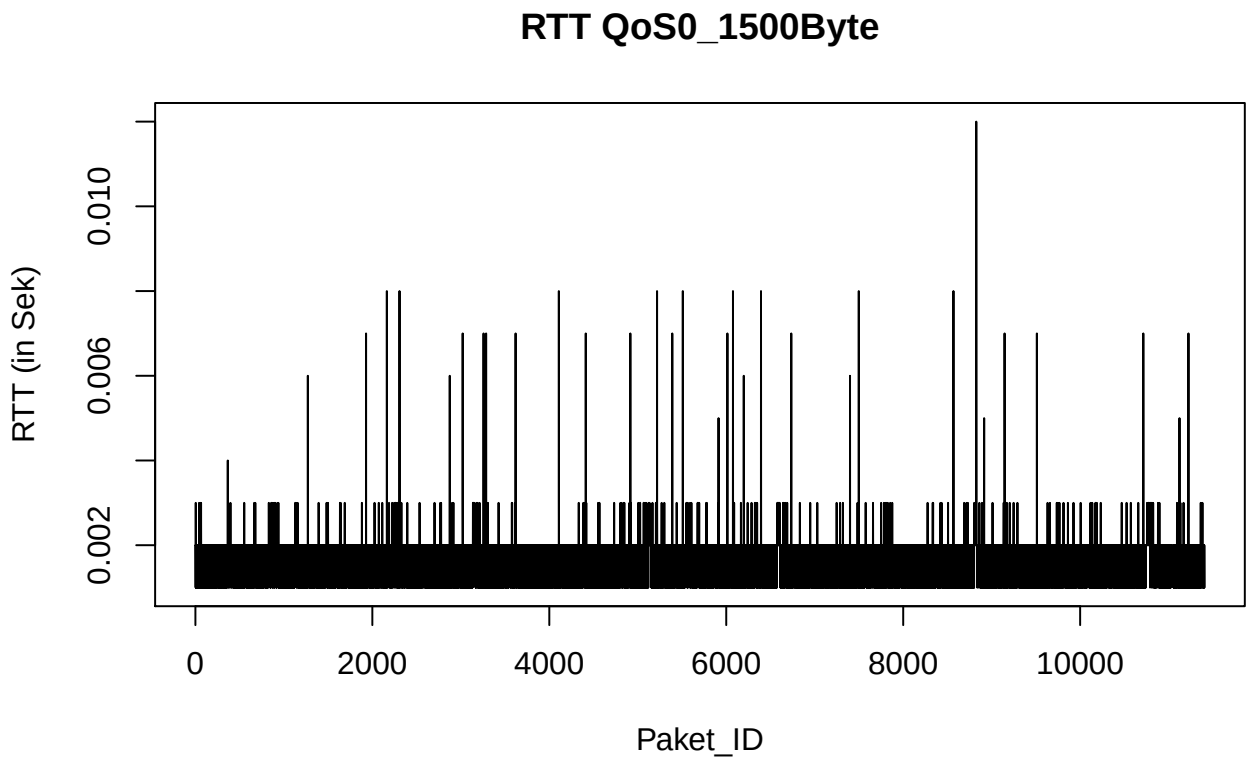

```
plot(latenzVBQoS0100Byte$id, latenzVBQoS0100Byte$rtt, type = "l", main = "RTT QoS0_100Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



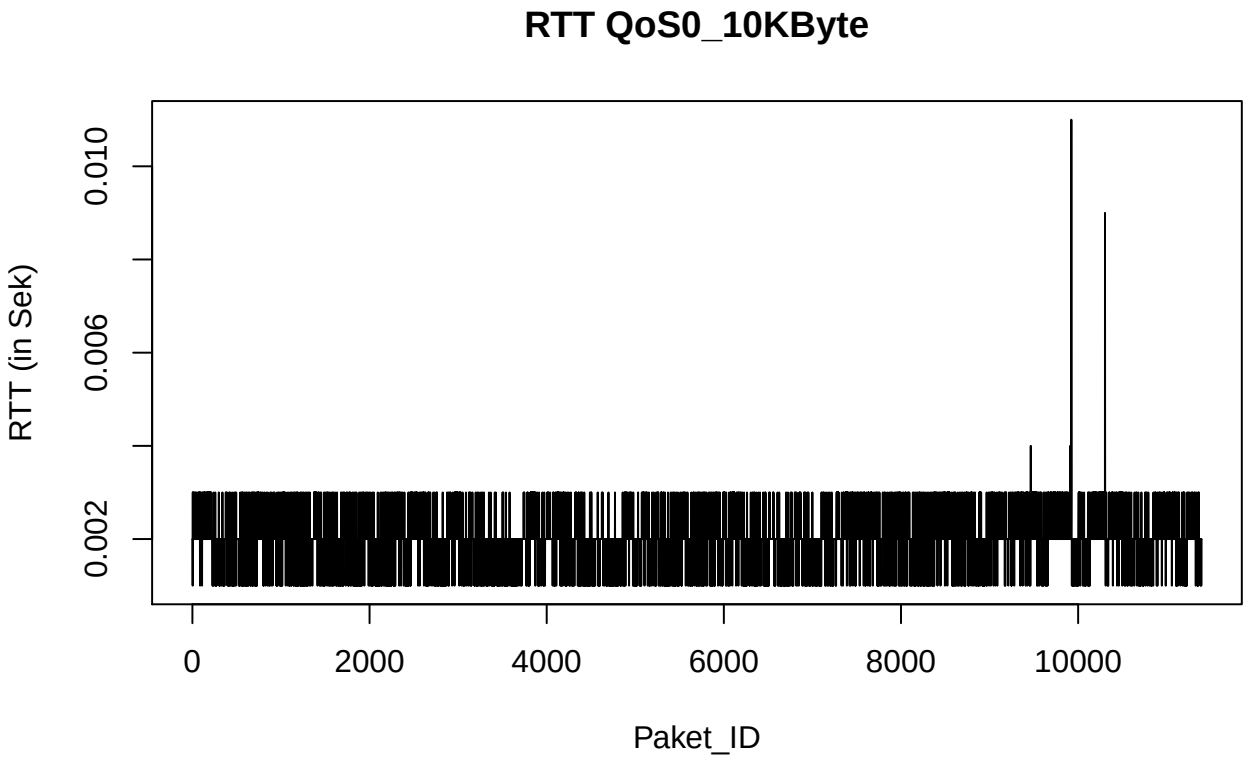

```
plot(latenzVBQoS01KByte$id, latenzVBQoS01KByte$rtt, type = "l", main = "RTT QoS0_1KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



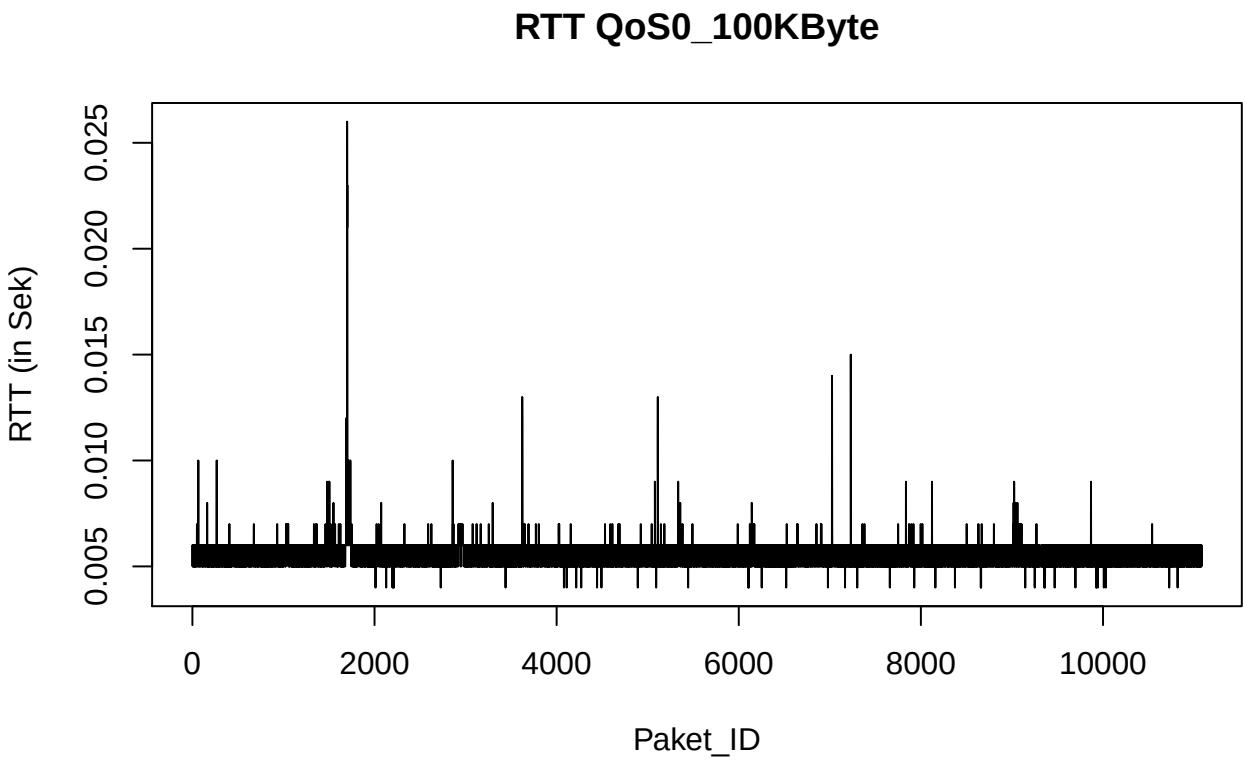

```
plot(latenzVBQoS01500Byte$id, latenzVBQoS01500Byte$rtt, type = "l", main = "RTT QoS0_1500Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



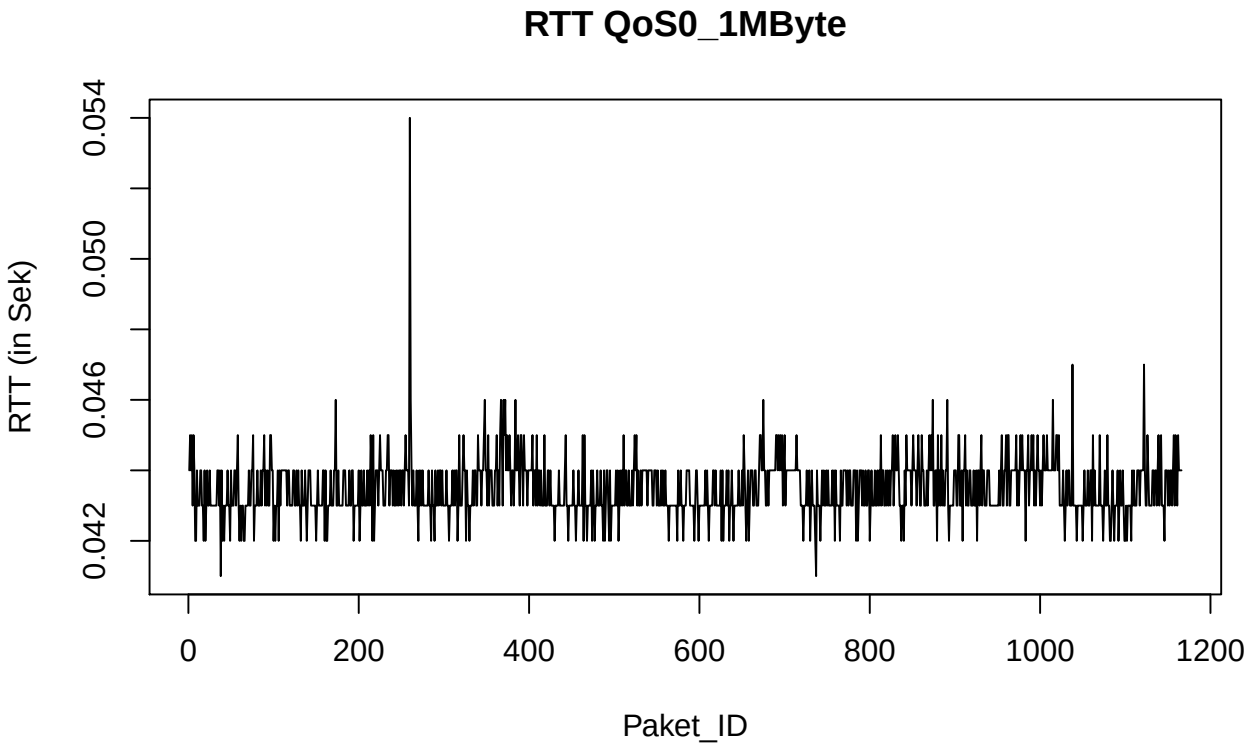

```
plot(latenzVBQoS010KByte$id, latenzVBQoS010KByte$rtt, type = "l", main = "RTT QoS0_10KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



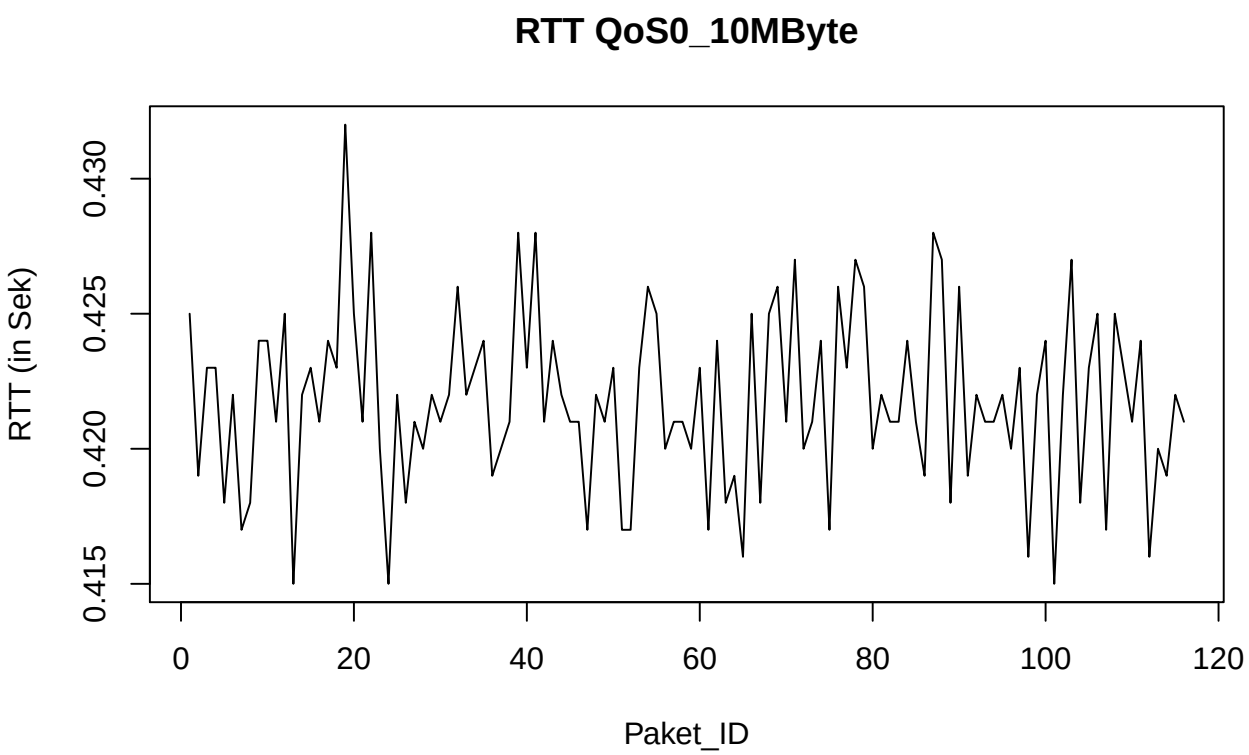

```
plot(latenzVBQoS0100KByte$id, latenzVBQoS0100KByte$rtt, type = "l", main = "RTT QoS0_100KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```




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

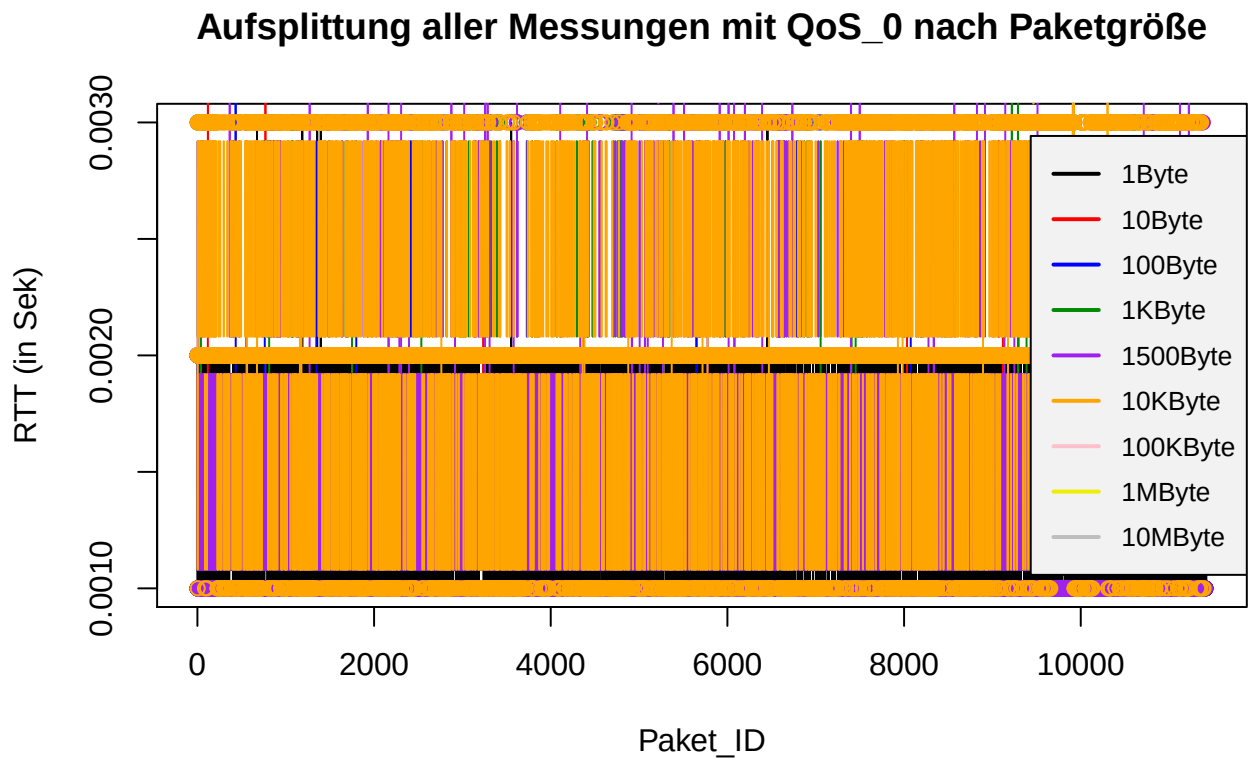



```
plot(latenzVBQoS010MByte$id, latenzVBQoS010MByte$rtt, type = "l", main = "RTT QoS0_10MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



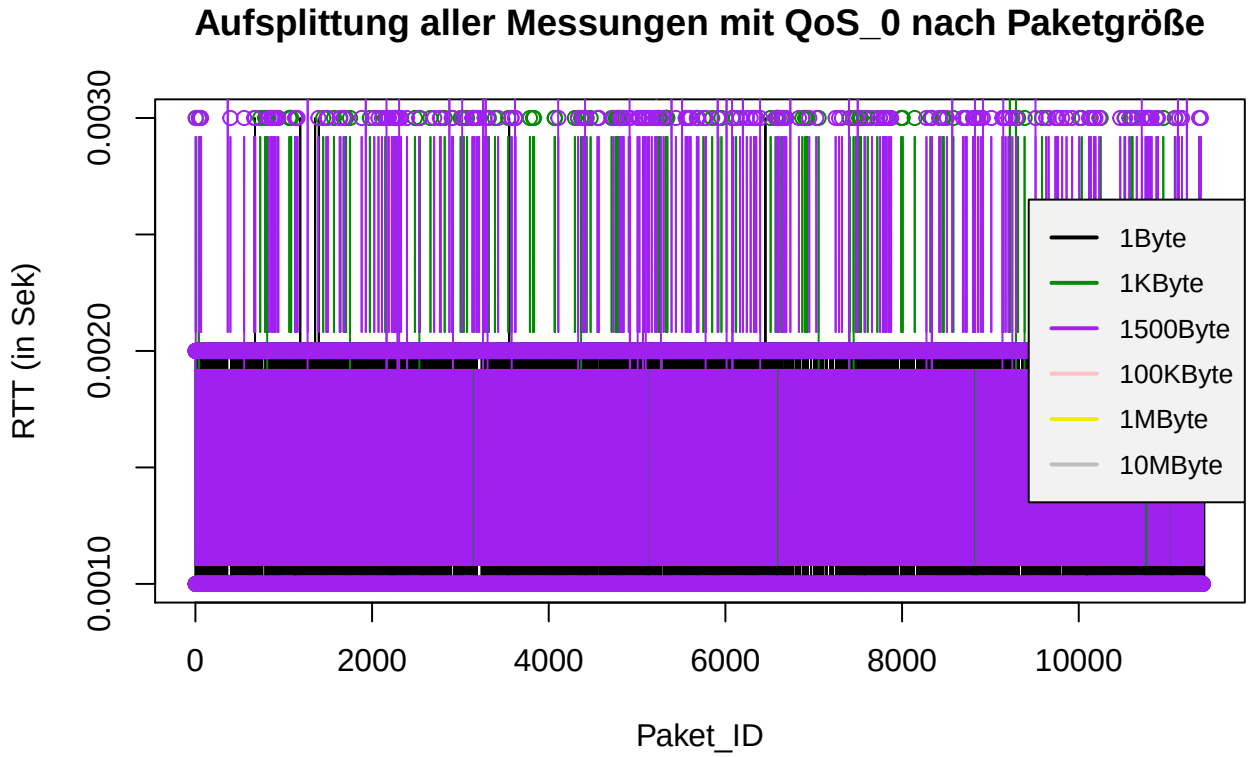

```
## QoS0 - Aufsplittung - eine Grafik!
plot(latenzVBQoS01Byte$id, latenzVBQoS01Byte$rtt, type = "l", ylab = "RTT (in Sek)", xlab = "Paket_ID",
```

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


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

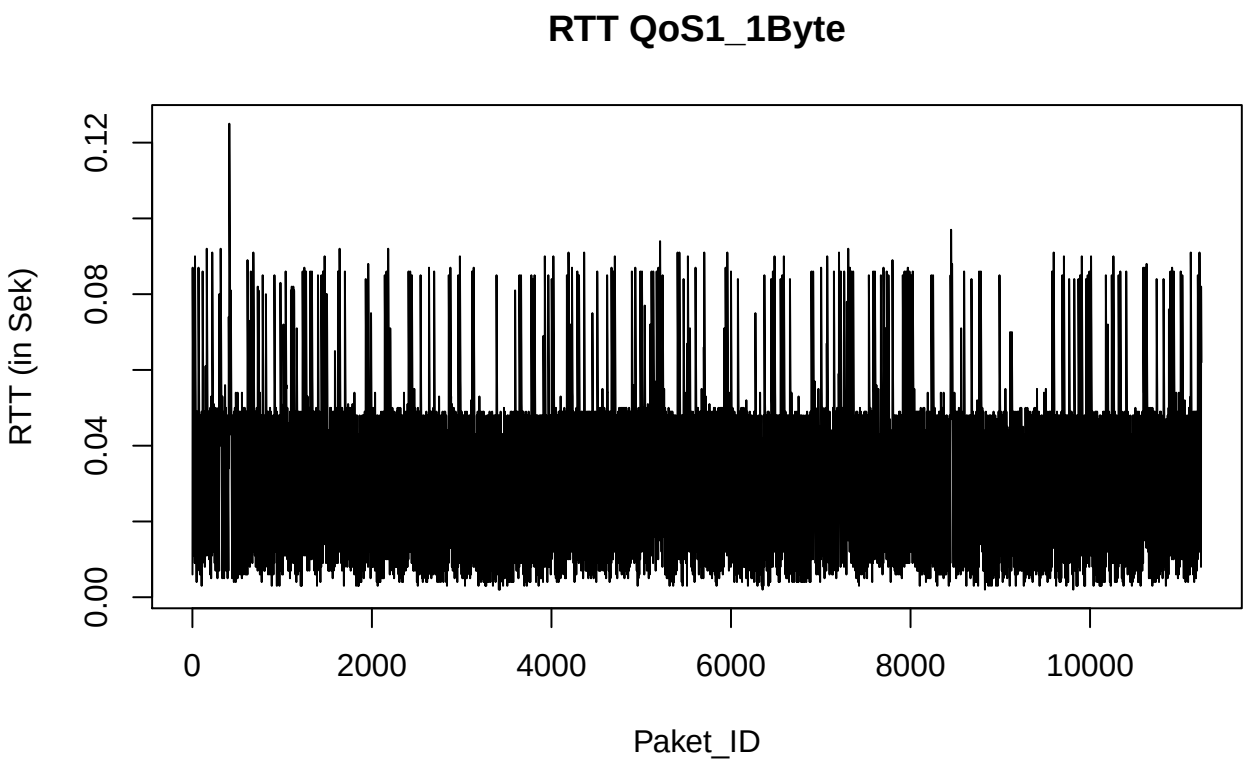



```
## QoS0 - Aufsplittung - eine Grafik!
plot(latenzVBQoS01Byte$id, latenzVBQoS01Byte$rtt, type = "l", ylab = "RTT (in Sek)", xlab = "Paket_ID",
     main = "Aufsplittung aller Messungen mit QoS_0 nach Paketgröße")
points(latenzVBQoS01KByte$id, latenzVBQoS01KByte$rtt, col = "green4", type = "b")
points(latenzVBQoS01500Byte$id, latenzVBQoS01500Byte$rtt, col = "purple", type = "b")
points(latenzVBQoS0100KByte$id, latenzVBQoS0100KByte$rtt, col = "pink", type = "b")
points(latenzVBQoS01MByte$id, latenzVBQoS01MByte$rtt, col = "yellow2", type = "b")
points(latenzVBQoS010MByte$id, latenzVBQoS010MByte$rtt, col = "gray", type = "b")


legend("right", c("1Byte", "1KByte", "1500Byte", "100KByte", "1MByte", "10MByte"),
       cex = 0.8,
```

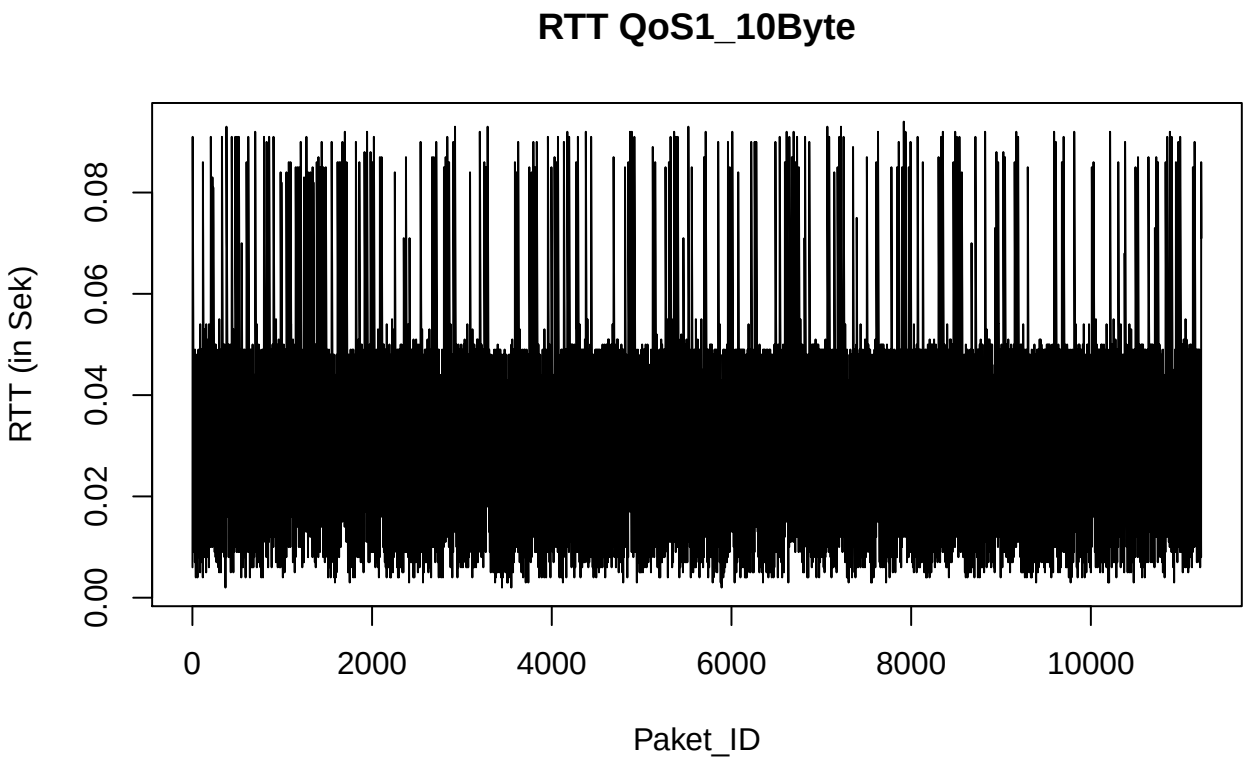
```
       col = c("black", "green4", "purple", "pink", "yellow2", "gray"),
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```



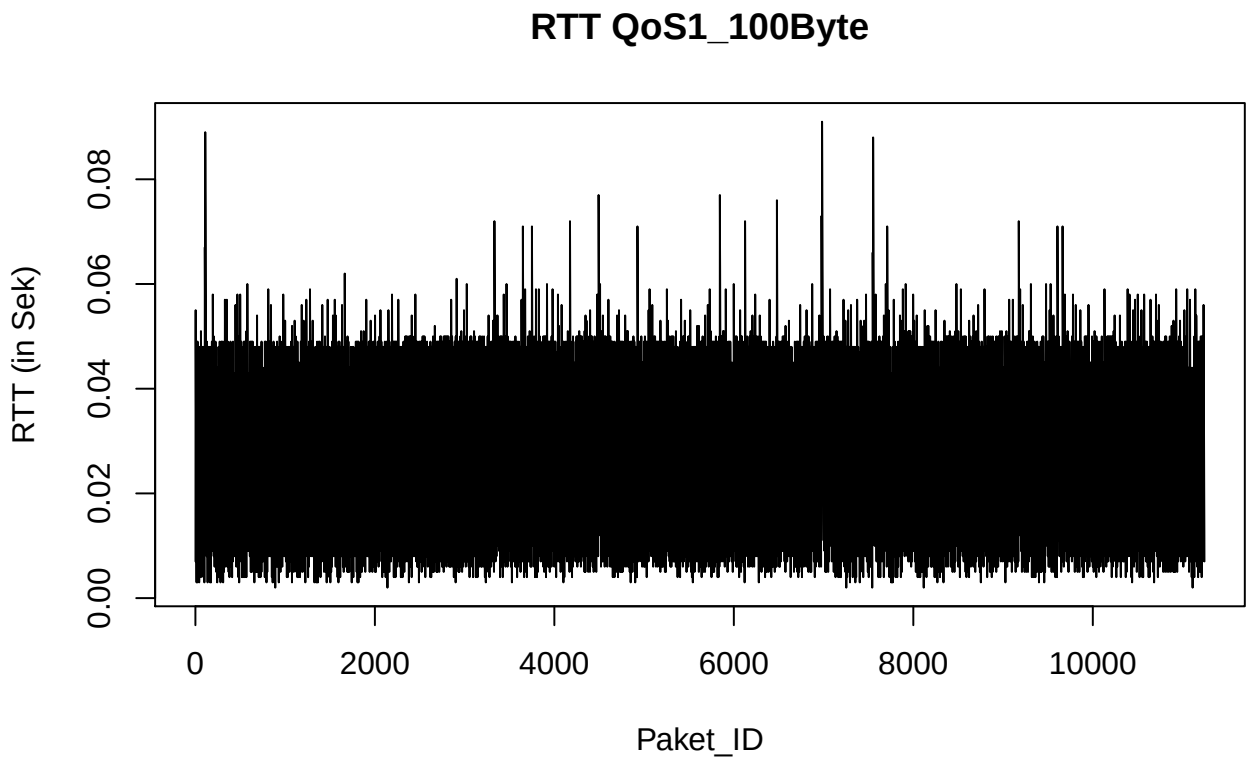

```
########################################
# QoS_1 Aufteilen nach Payload/ Größe #
########################################
latenzVBQoS11Byte<-latenzVBQoS1[latenzVBQoS1$Size == "1Byte",]
latenzVBQoS110Byte<-latenzVBQoS1[latenzVBQoS1$Size == "10Byte",]
latenzVBQoS1100Byte<-latenzVBQoS1[latenzVBQoS1$Size == "100Byte",]
latenzVBQoS11KByte<-latenzVBQoS1[latenzVBQoS1$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")
```



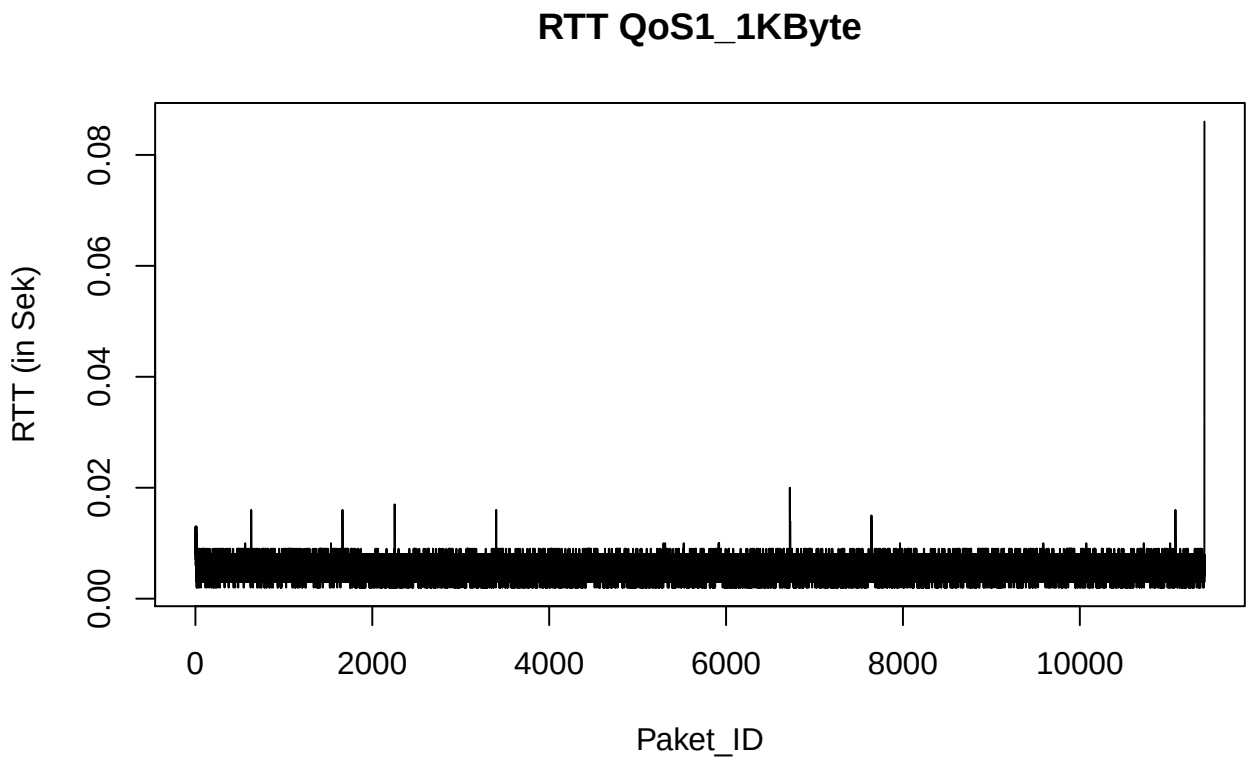

```
plot(latenzVBQoS110Byte$id, latenzVBQoS110Byte$rtt, type = "l", main = "RTT QoS1_10Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



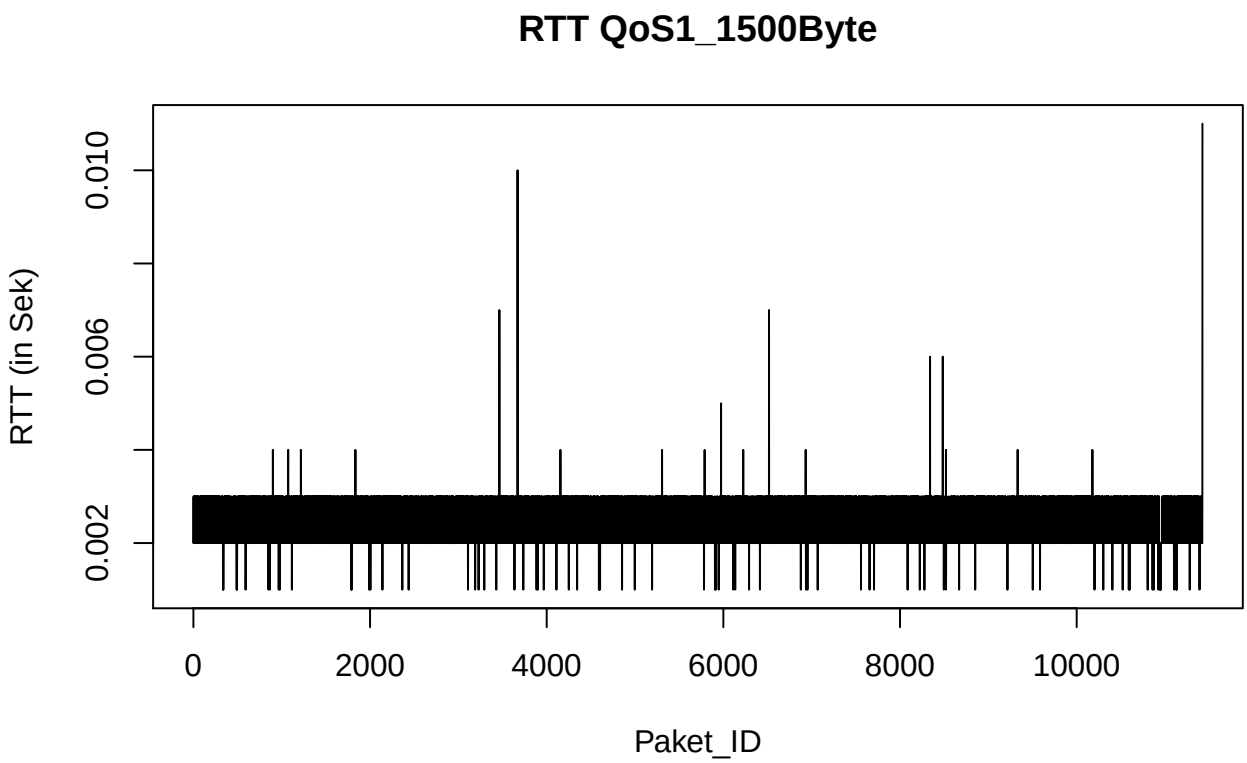

```
plot(latenzVBQoS1100Byte$id, latenzVBQoS1100Byte$rtt, type = "l", main = "RTT QoS1_100Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



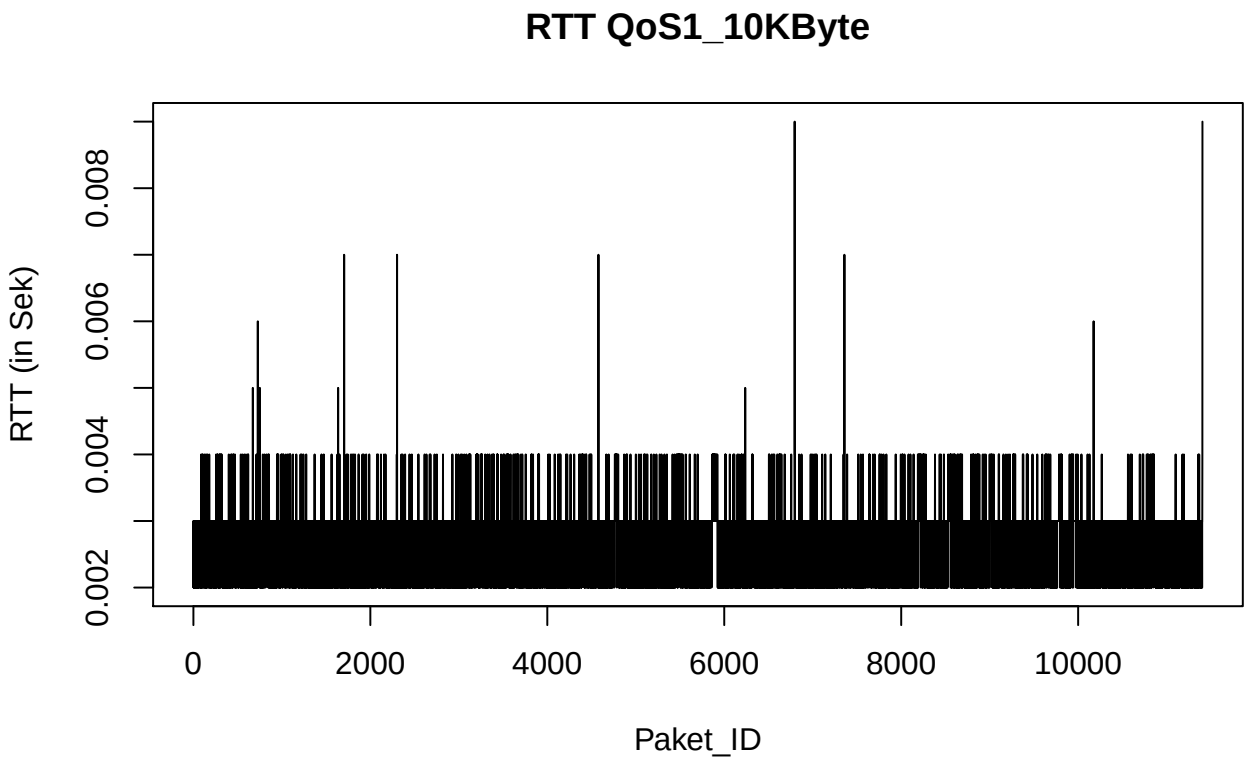

```
plot(latenzVBQoS11KByte$id, latenzVBQoS11KByte$rtt, type = "l", main = "RTT QoS1_1KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



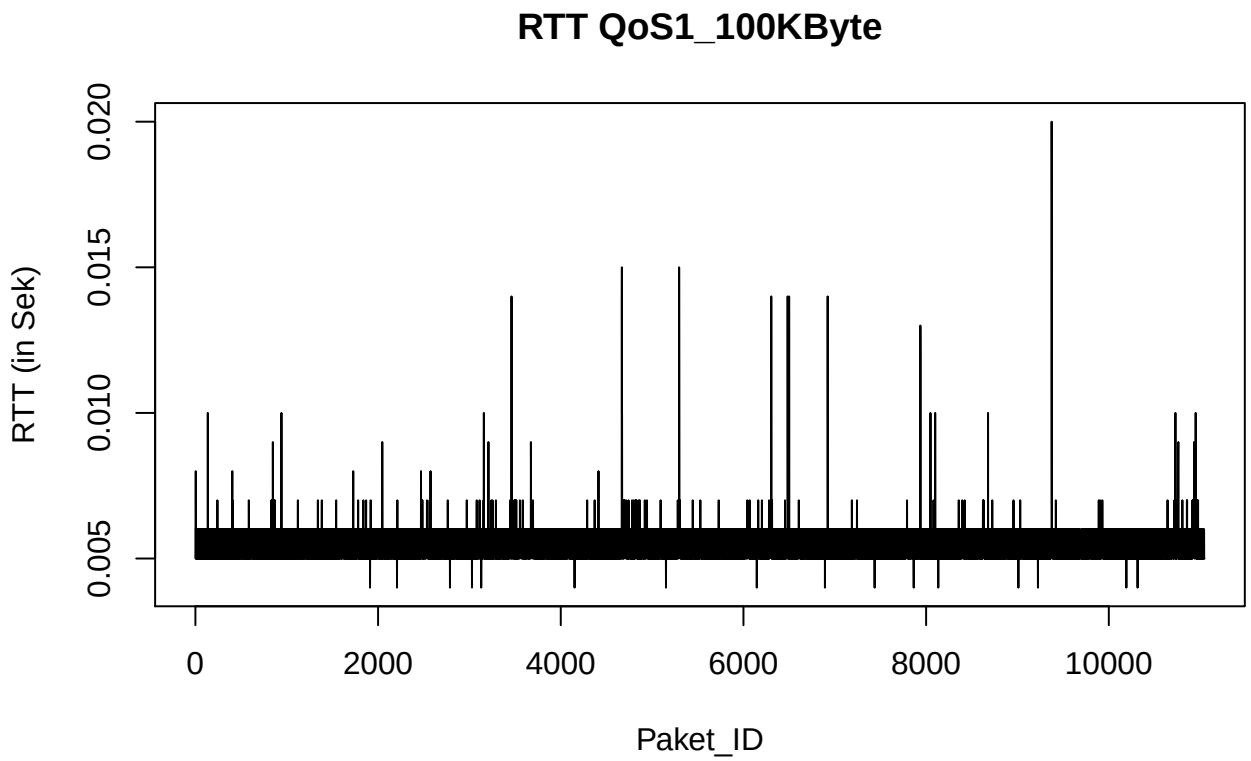

```
plot(latenzVBQoS11500Byte$id, latenzVBQoS11500Byte$rtt, type = "l", main = "RTT QoS1_1500Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



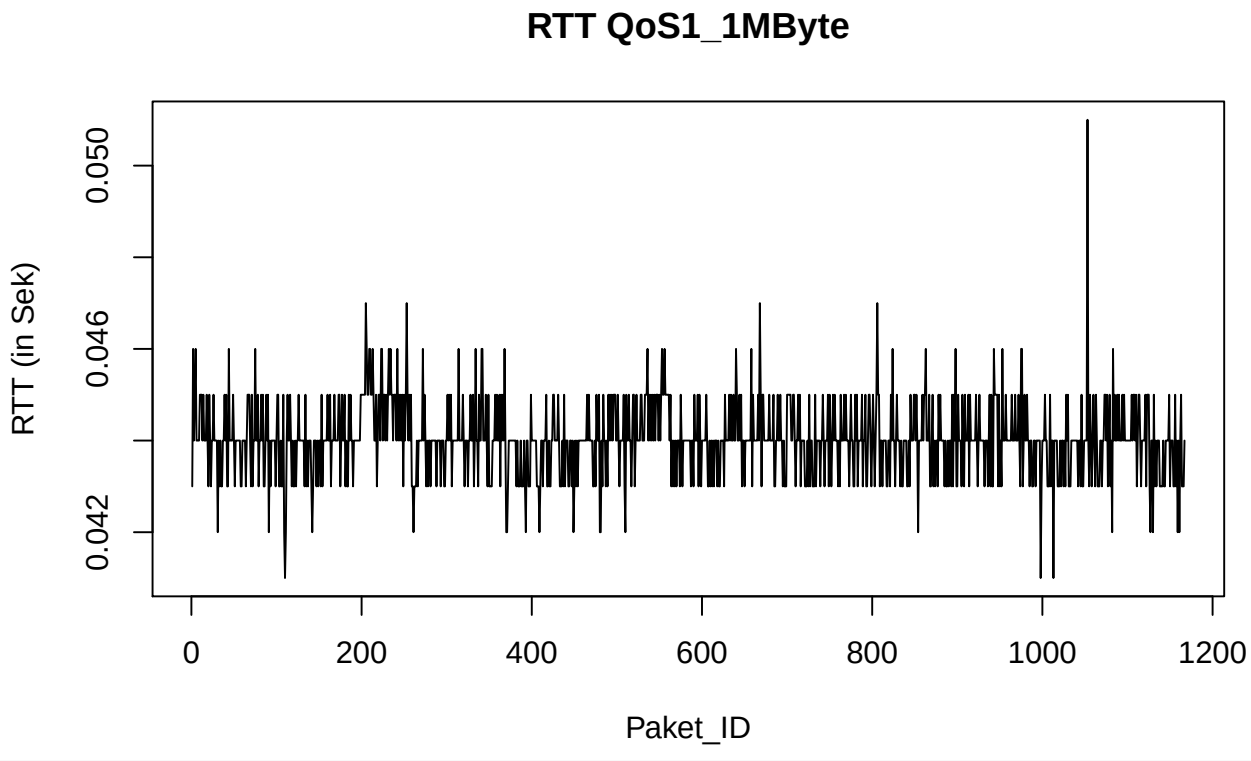

```
plot(latenzVBQoS110KByte$id, latenzVBQoS110KByte$rtt, type = "l", main = "RTT QoS1_10KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



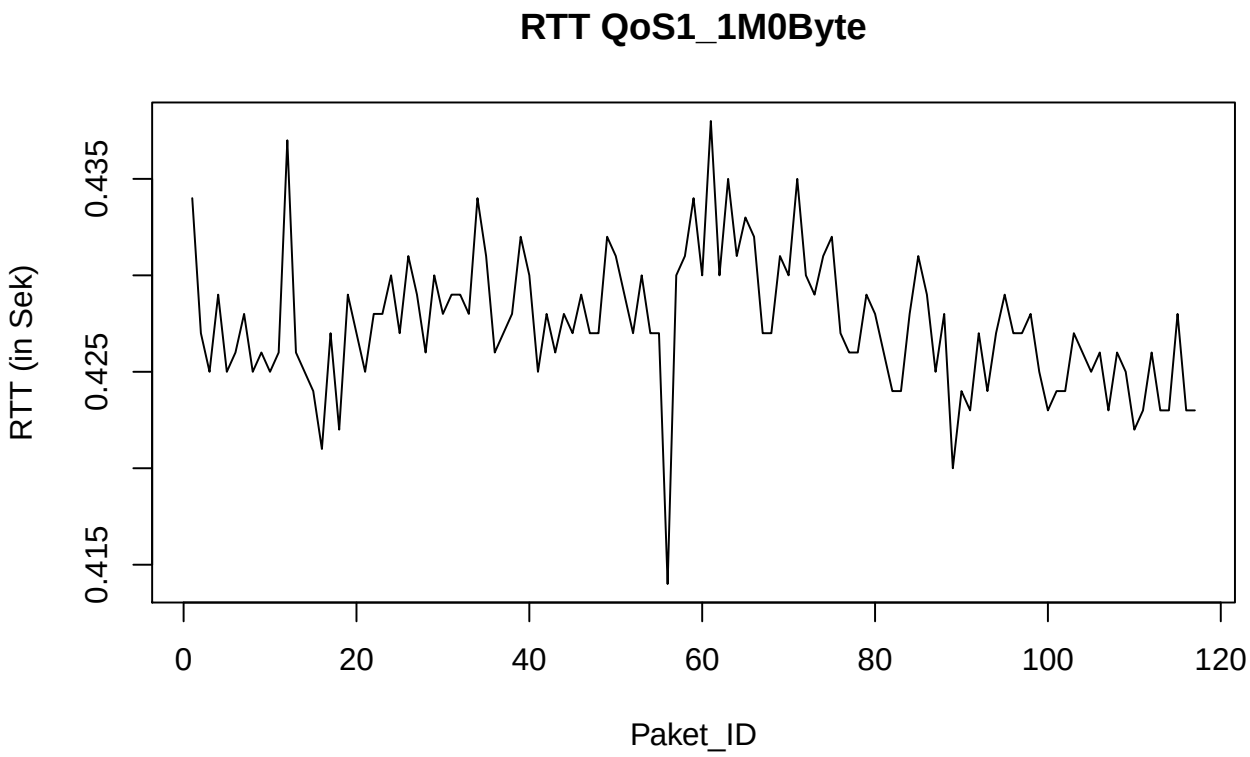

```
plot(latenzVBQoS1100KByte$id, latenzVBQoS1100KByte$rtt, type = "l", main = "RTT QoS1_100KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



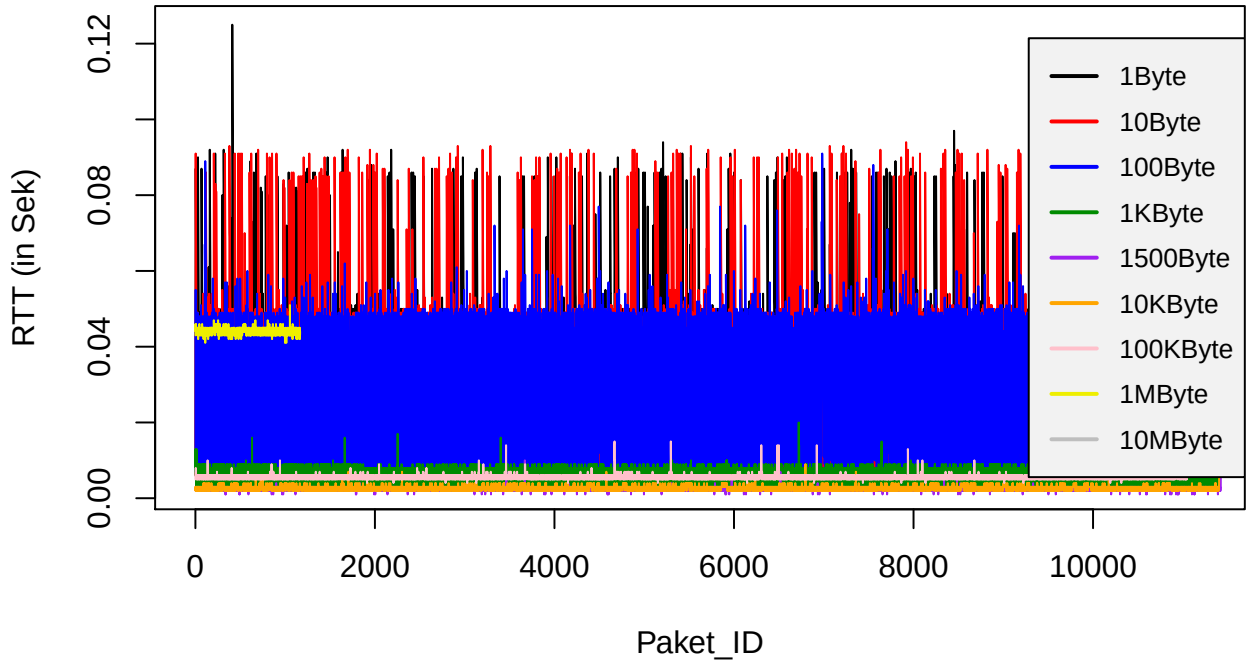

```
plot(latenzVBQoS11MByte$id, latenzVBQoS11MByte$rtt, type = "l", main = "RTT QoS1_1MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



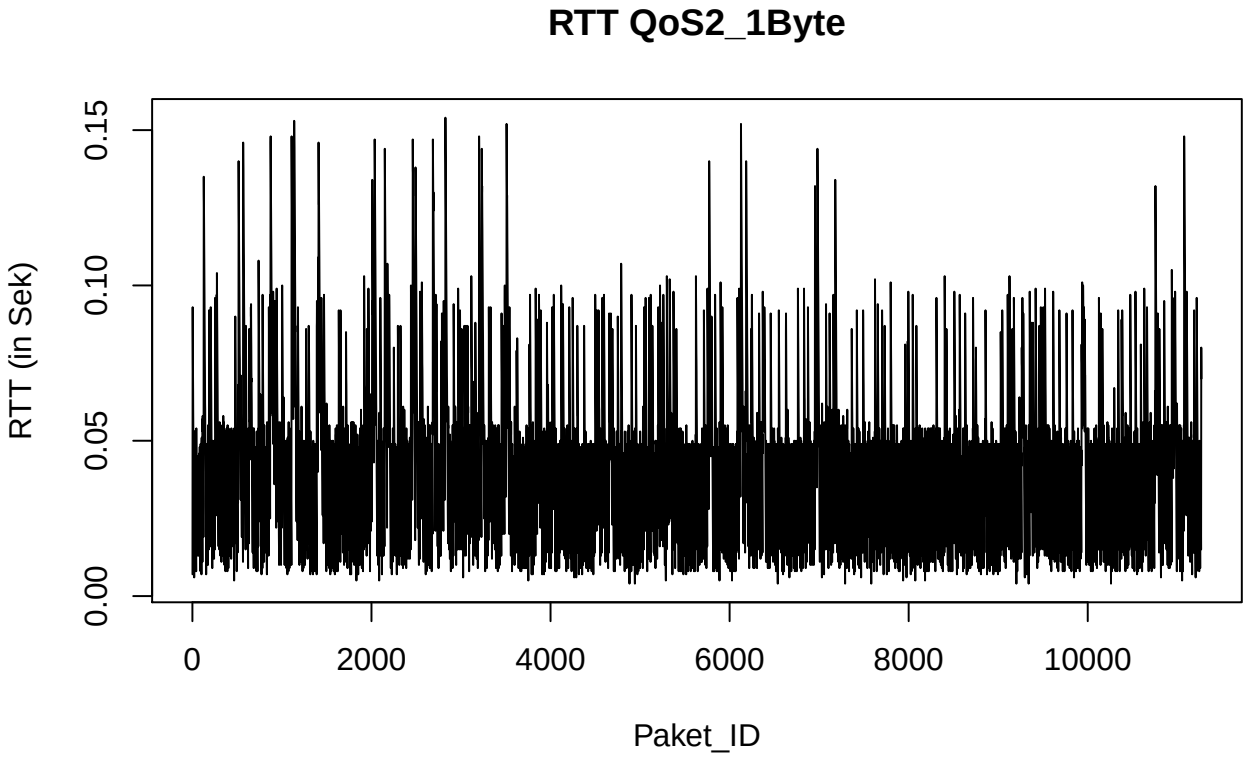

```
plot(latenzVBQoS110MByte$id, latenzVBQoS110MByte$rtt, type = "l", main = "RTT QoS1_10MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



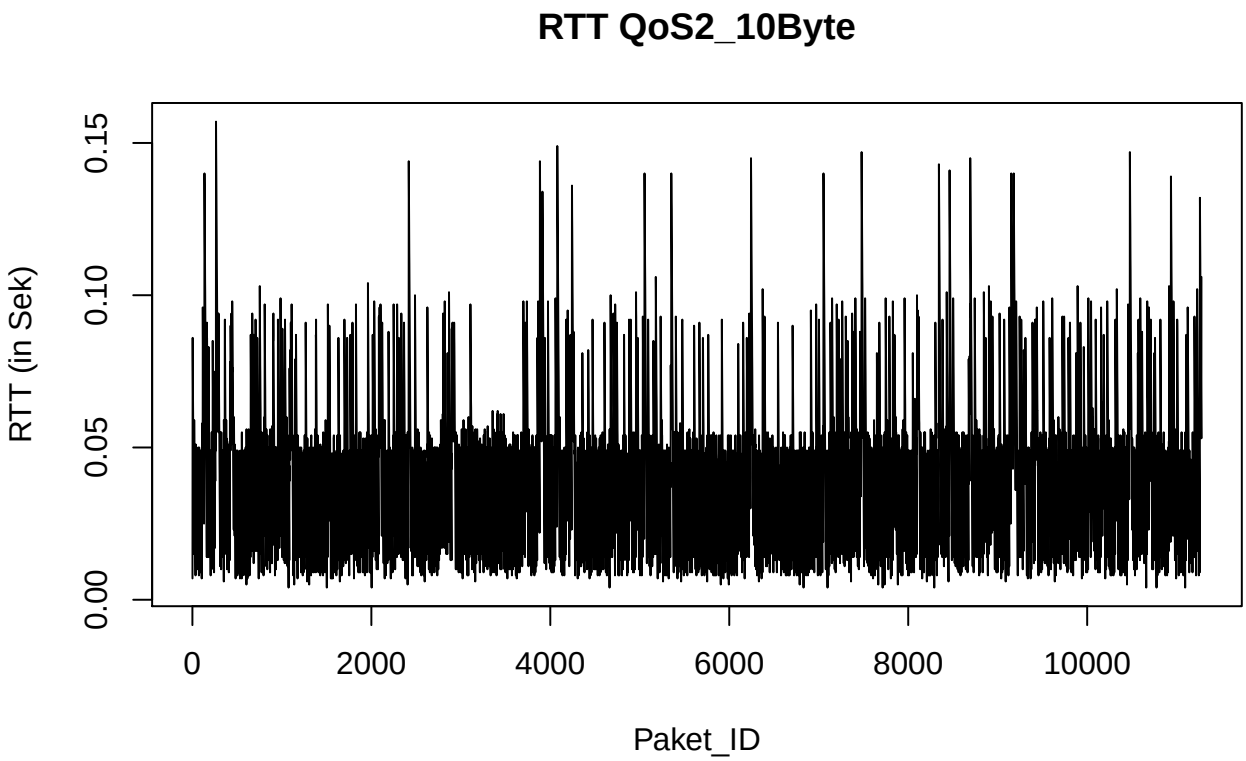

```
## QoS1 - Aufsplittung - eine Grafik!
plot(latenzVBQoS11Byte$id, latenzVBQoS11Byte$rtt, type = "l", ylab = "RTT (in Sek)", xlab = "Paket_ID",
     main = "Aufsplittung aller Messungen mit QoS_0 nach Paketgröße")
points(latenzVBQoS110Byte$id, latenzVBQoS110Byte$rtt, col = "red", type = "l")
points(latenzVBQoS1100Byte$id, latenzVBQoS1100Byte$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(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", "1MByte", "
       cex = 0.8,
       col = c("black", "red", "blue", "green4", "purple", "orange", "pink", "yellow2", "gray"),
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```



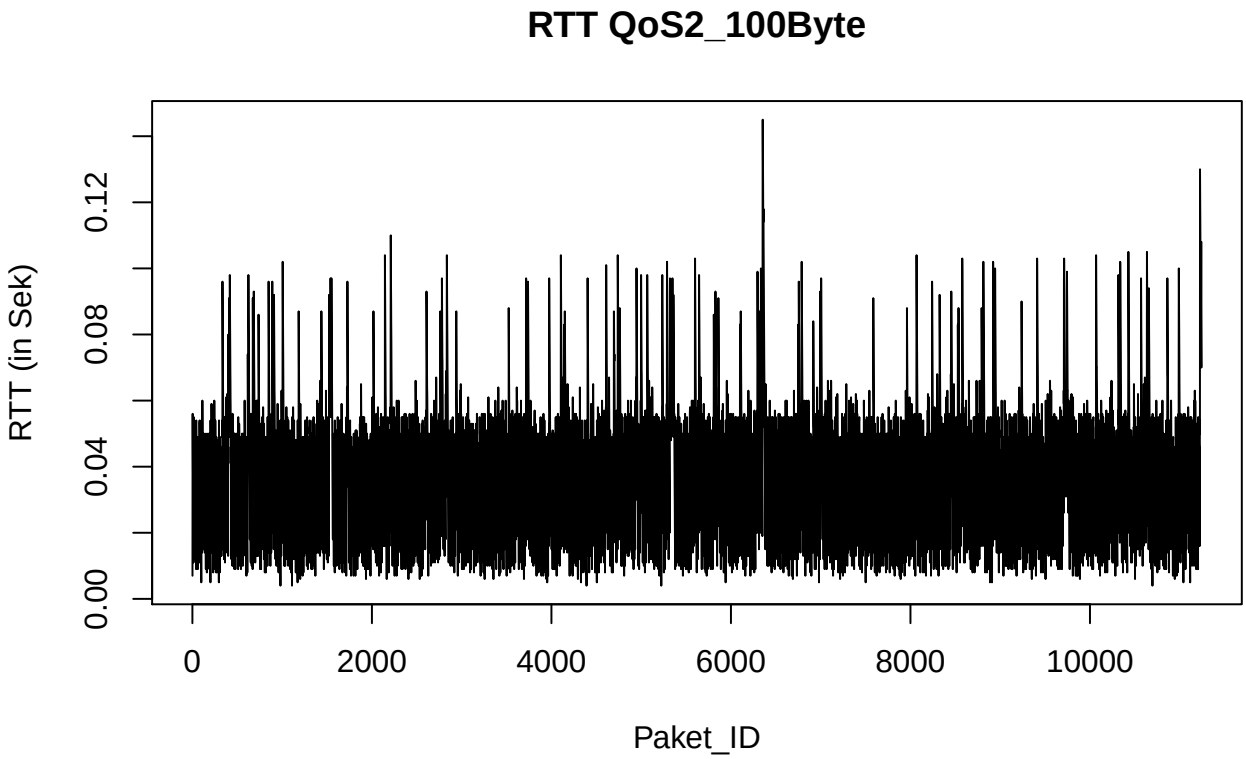

```
########################################
# QoS_2 Aufteilen nach Payload/ Größe #
########################################
latenzVBQoS21Byte<-latenzVBQoS2[latenzVBQoS2$Size == "1Byte",]
latenzVBQoS210Byte<-latenzVBQoS2[latenzVBQoS2$Size == "10Byte",]
latenzVBQoS2100Byte<-latenzVBQoS2[latenzVBQoS2$Size == "100Byte",]
latenzVBQoS21KByte<-latenzVBQoS2[latenzVBQoS2$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")
```



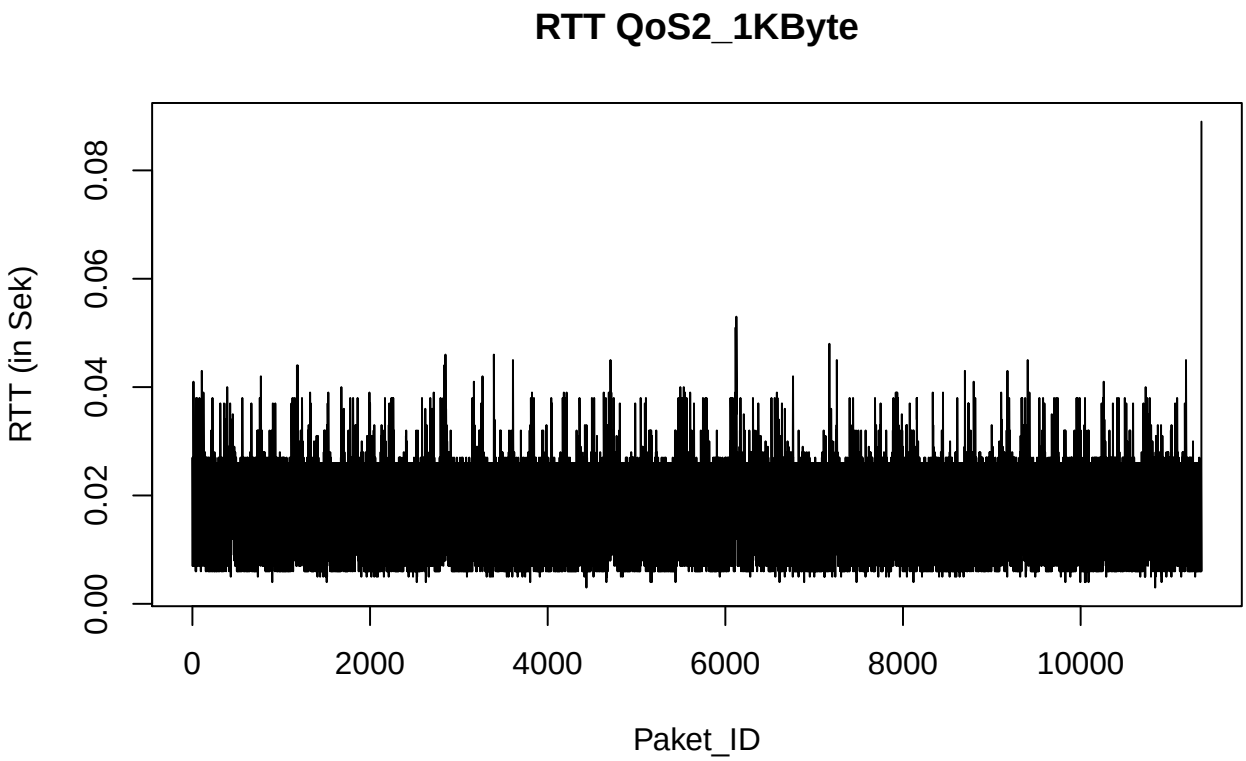

```
plot(latenzVBQoS210Byte$id, latenzVBQoS210Byte$rtt, type = "l", main = "RTT QoS2_10Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



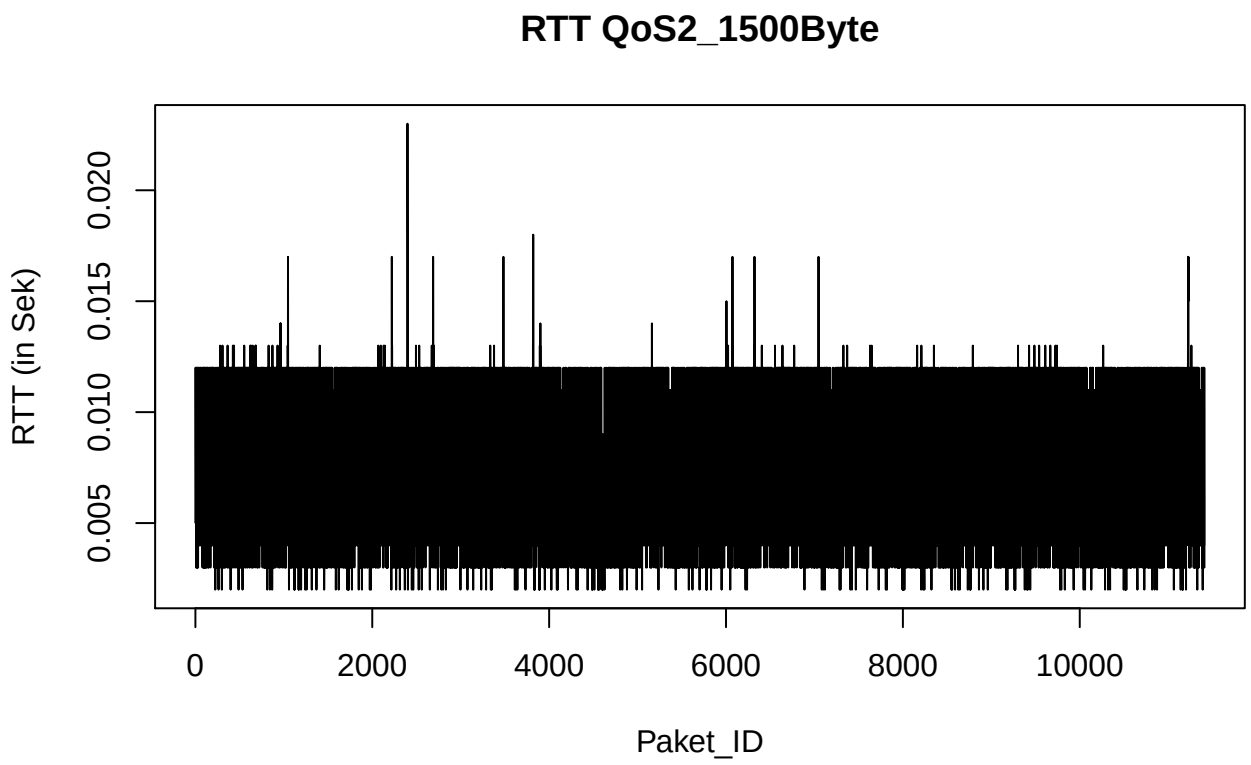

```
plot(latenzVBQoS2100Byte$id, latenzVBQoS2100Byte$rtt, type = "l", main = "RTT QoS2_100Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



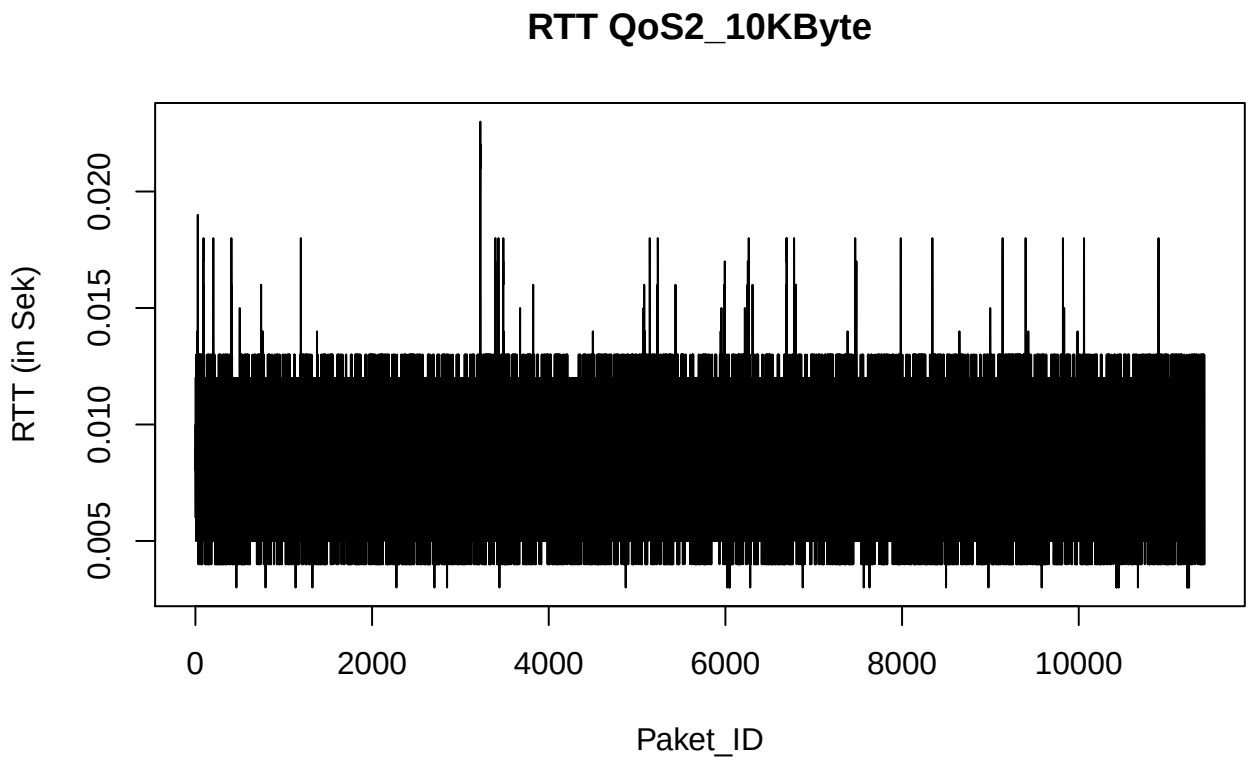

```
plot(latenzVBQoS21KByte$id, latenzVBQoS21KByte$rtt, type = "l", main = "RTT QoS2_1KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



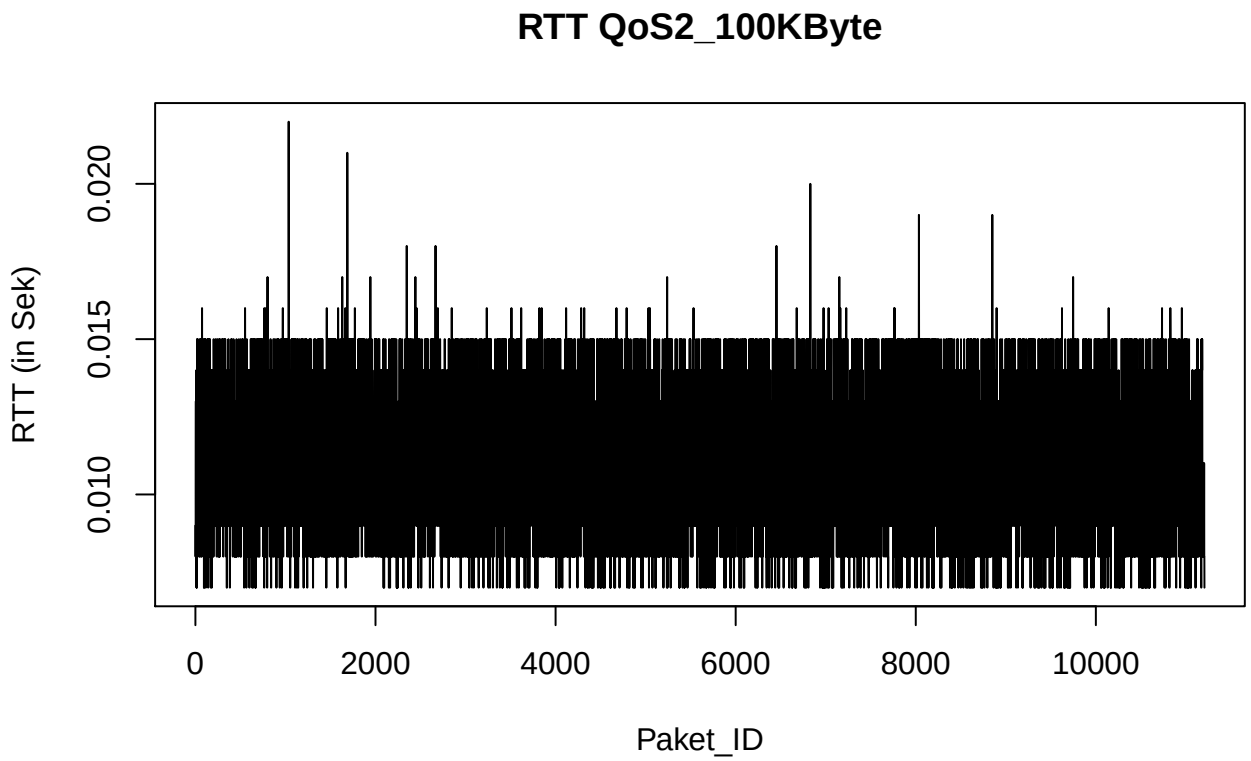

```
plot(latenzVBQoS21500Byte$id, latenzVBQoS21500Byte$rtt, type = "l", main = "RTT QoS2_1500Byte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



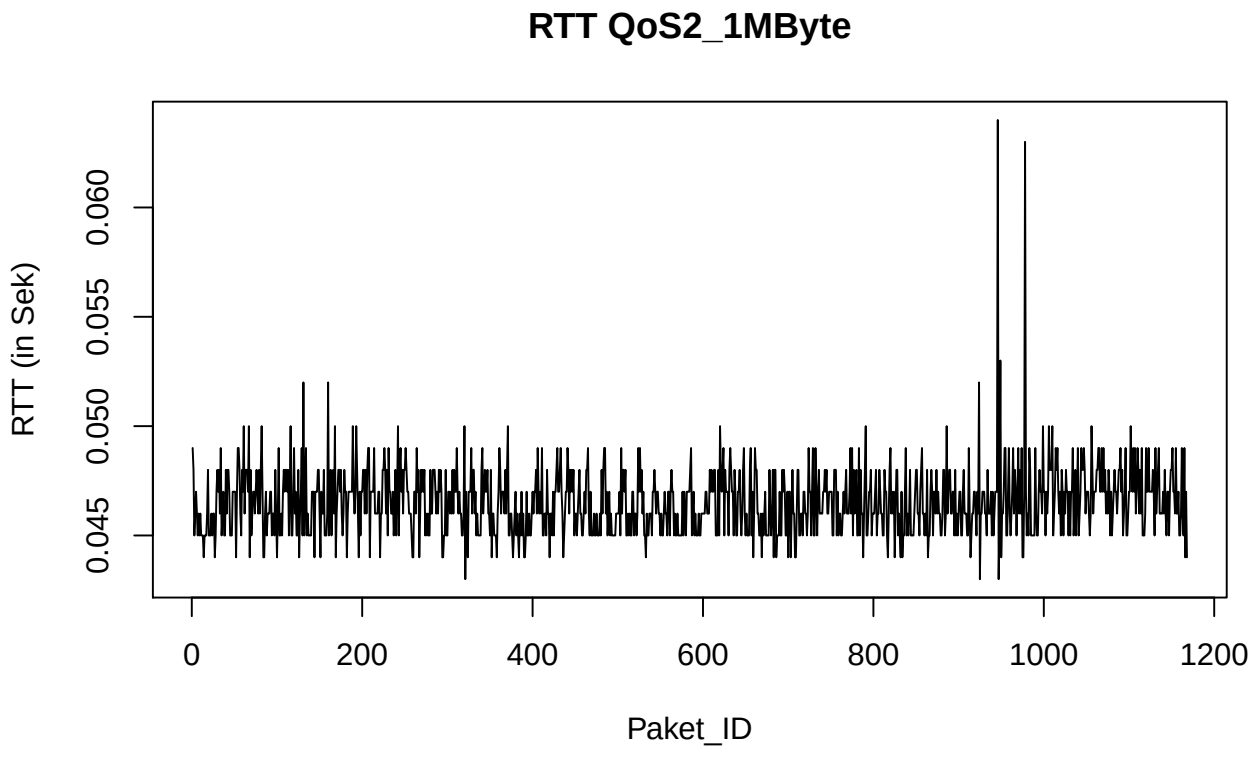

```
plot(latenzVBQoS210KByte$id, latenzVBQoS210KByte$rtt, type = "l", main = "RTT QoS2_10KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



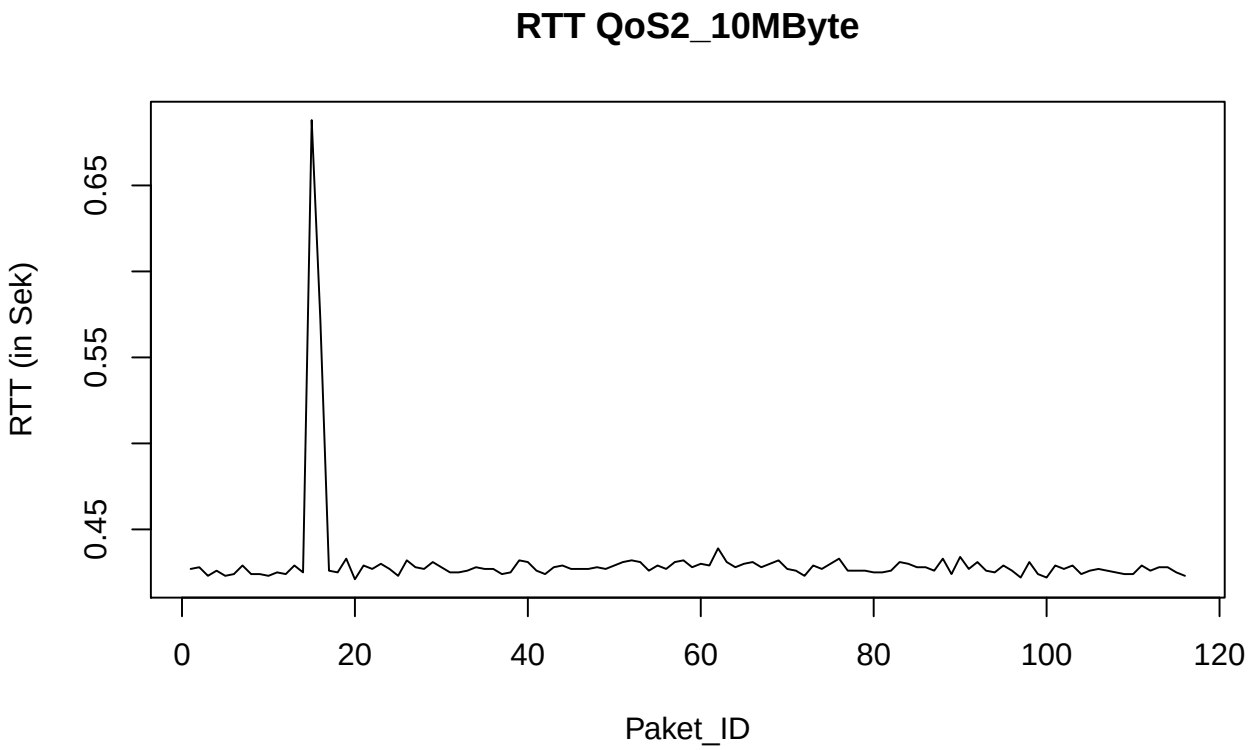

```
plot(latenzVBQoS2100KByte$id, latenzVBQoS2100KByte$rtt, type = "l", main = "RTT QoS2_100KByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



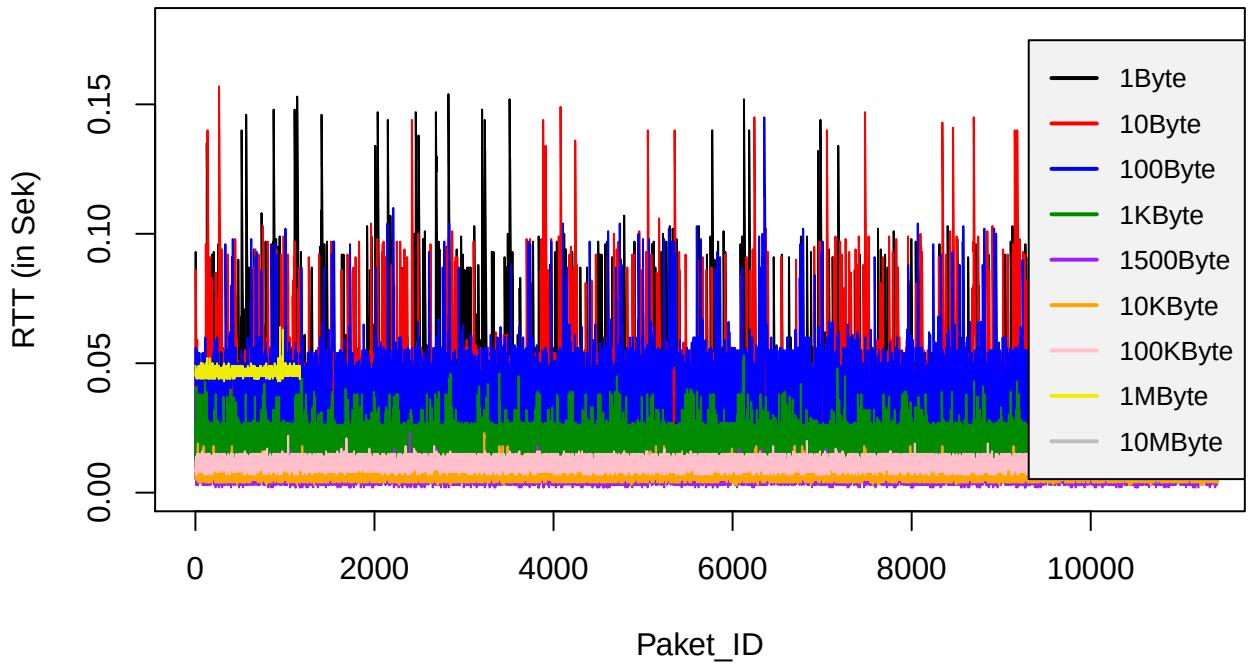

```
plot(latenzVBQoS21MByte$id, latenzVBQoS21MByte$rtt, type = "l", main = "RTT QoS2_1MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



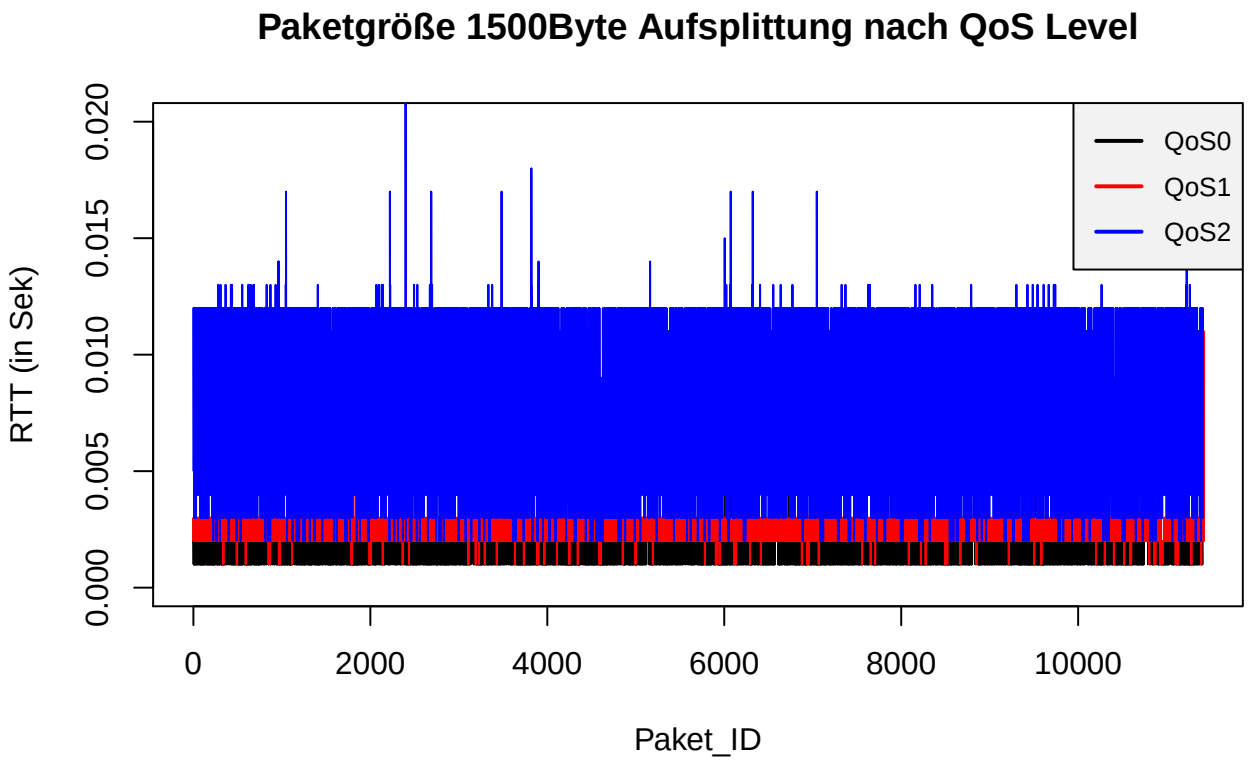

```
plot(latenzVBQoS210MByte$id, latenzVBQoS210MByte$rtt, type = "l", main = "RTT QoS2_10MByte",
     ylab = "RTT (in Sek)", xlab = "Paket_ID")
```



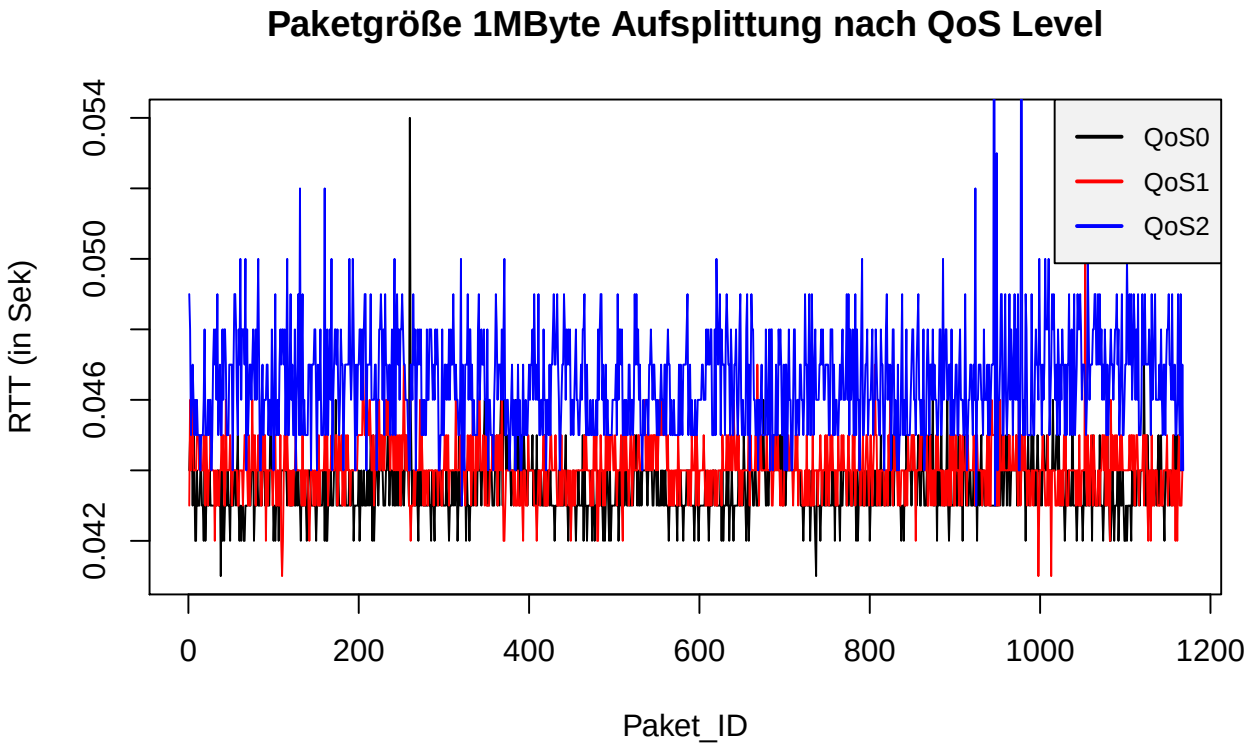

```
## QoS2 - Aufsplittung - eine Grafik!
plot(latenzVBQoS21Byte$id, latenzVBQoS21Byte$rtt, type = "l", ylim = c(0, 0.18), ylab = "RTT (in Sek)",
     main = "Aufsplittung aller Messungen mit QoS_0 nach Paketgröße")
points(latenzVBQoS210Byte$id, latenzVBQoS210Byte$rtt, col = "red", type = "l")
points(latenzVBQoS2100Byte$id, latenzVBQoS2100Byte$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(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", "1MByte",
       cex = 0.8,
       col = c("black", "red", "blue", "green4", "purple", "orange", "pink", "yellow2", "gray"),
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```




```
################################################
# Paketgröße 1500 Aufsplittung nach QoS Level #
################################################

## Paketgröße 1500Byte
plot(latenzVBQoS01500Byte$id, latenzVBQoS01500Byte$rtt, type = "l", ylim = c(0, 0.02), ylab = "RTT (in S
points(latenzVBQoS11500Byte$id, latenzVBQoS11500Byte$rtt, col = "red", type = "l")
points(latenzVBQoS21500Byte$id, latenzVBQoS21500Byte$rtt, col = "blue", type = "l")

legend("topright", c("QoS0", "QoS1", "QoS2"),
       cex = 0.8,
       col = c("black", "red", "blue"),
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```



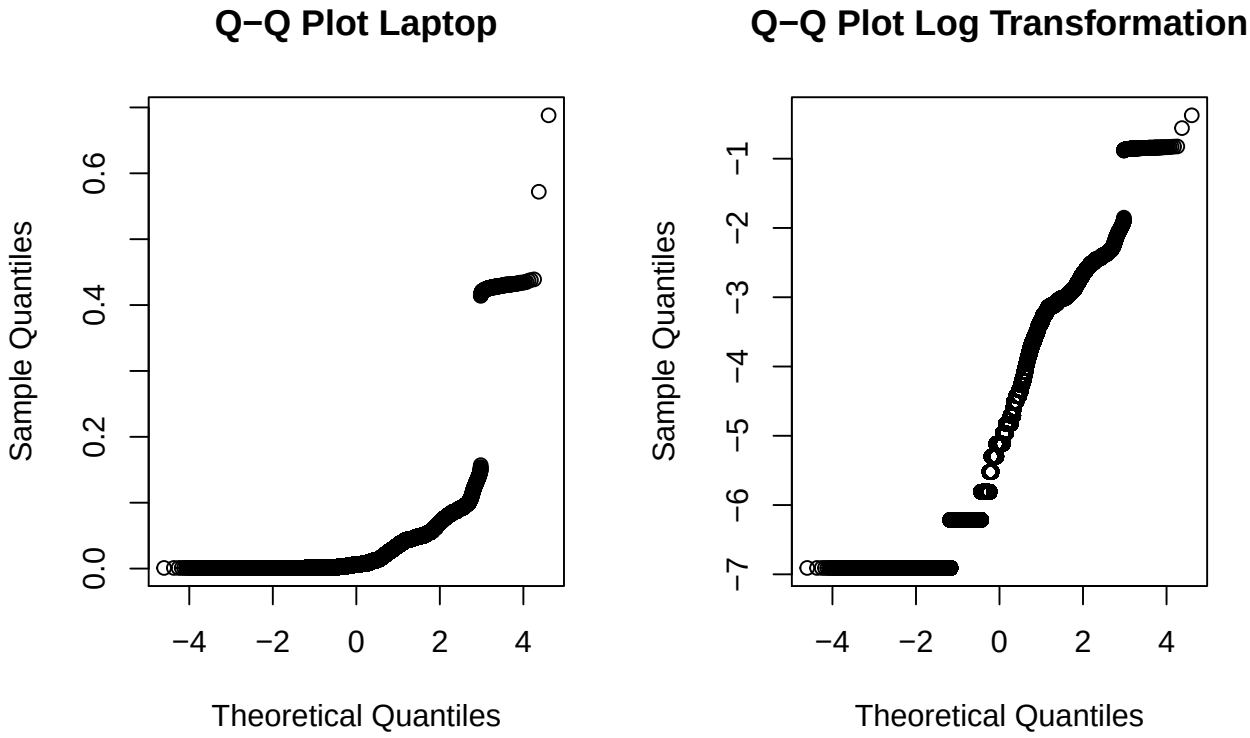

```
## Paketgröße 1MByte
plot(latenzVBQoS01MByte$id, latenzVBQoS01MByte$rtt, type = "l", ylab = "RTT (in Sek)", xlab = "Paket_ID
points(latenzVBQoS11MByte$id, latenzVBQoS11MByte$rtt, col = "red", type = "l")
points(latenzVBQoS21MByte$id, latenzVBQoS21MByte$rtt, col = "blue", type = "l")

legend("topright", c("QoS0", "QoS1", "QoS2"),
       cex = 0.8,
       col = c("black", "red", "blue"),
       text.col = "black" ,lwd = c(2, 2, 2),
       y.intersp = 1.5, merge = FALSE, bg = "gray95")
```




```
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
#> -0.032053 -0.009102 -0.003407  0.009238  0.251469
#>
#> Coefficients:
#>                       Estimate Std. Error  t value Pr(>|t|)
#> (Intercept)          1.436e-02  8.086e-05  177.545   <2e-16 ***
#> latenzVB$QoSqos1     1.391e-02  6.555e-05  212.137   <2e-16 ***
#> latenzVB$QoSqos2     2.168e-02  6.552e-05  330.973   <2e-16 ***
#> latenzVB$Byte10      1.240e-05  1.011e-04    0.123    0.902
#> latenzVB$Byte100    -3.254e-03  1.011e-04  -32.198   <2e-16 ***
#> latenzVB$Byte1000   -1.789e-02  1.009e-04 -177.366   <2e-16 ***
#> latenzVB$Byte10000  -2.177e-02  1.008e-04 -215.880   <2e-16 ***
#> latenzVB$Byte1500   -2.259e-02  1.008e-04 -224.109   <2e-16 ***
#> latenzVB$Byte1e+05  -1.886e-02  1.015e-04 -185.776   <2e-16 ***
#> latenzVB$Byte1e+06   1.846e-02  2.335e-04   79.051   <2e-16 ***
#> latenzVB$Byte1e+07   4.005e-01  7.076e-04  565.994   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.01315 on 241451 degrees of freedom
#> Multiple R-squared:  0.7135, Adjusted R-squared:  0.7134
#> F-statistic: 6.012e+04 on 10 and 241451 DF,  p-value: < 2.2e-16
```

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 /.

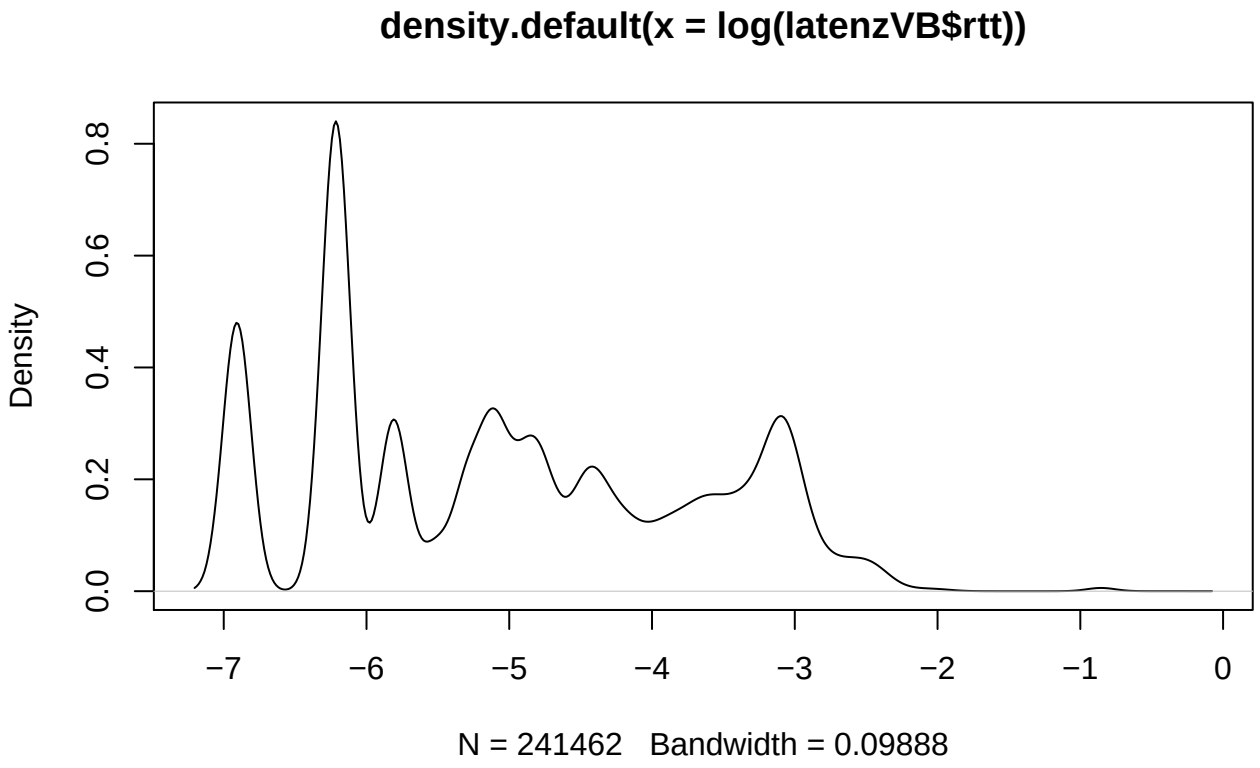
```
par(mfrow=c(1,2))
qqnorm(latenzVB$rtt, main = "Q-Q Plot Laptop")
qqnorm(log(latenzVB$rtt), main = "Q-Q Plot Log Transformation")
```



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)

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

```
plot(density(log(latenzVB$rtt)))
```


density.default(x = log(latenzVB$rtt))
Density
0.0
0.2
0.4
0.6
0.8
−7
−6
−5
−4
−3
−2
−1
0
N = 241462 Bandwidth = 0.09888