Advanced Quantitative methods

prof. Gerald Q. Maguire Jr.
School of Information and Communication Technology (ICT)
KTH Royal Institute of Technology
http://web.ict.kth.se/~maguire
&
Prof. em. Marilyn E. Noz, Ph. D.
School of Medicine
New York University

II2202 Fall 2012
2012.09.24

Selected Topics in Quantitative analysis

Power

Parametric and non-Parametric Data

If your data is “normally distributed”, then it can be examined using “parametric” statistics such as **mean**, **variance**, **standard deviation** etc.

In R:

```
meandata<- mean(data_set)

variance.data<-var(data_set)

std.data<-(sqrt(var(data_set)))
```

Normal or Parametric Data

```
data.new ← Read.table(“tmp.data.head”, header=TRUE)
t.test(data.new[,2])
        One Sample t-test
data:  data.new[, 2]
t = 2876.076, df = 41, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 11.05714 11.07268
sample estimates:
mean of x
 11.06491
```

What does this mean? The data is highly significantly different from zero.

Normal or Parametric Data - 2

Try using a mu (mean) different from zero:

```
t.test(data.new[,2], mu= 11.066)
```

```
One Sample t-test

data:  data.new[, 2]
t = 1.2767, df = 41, p-value = 0.069
alternative hypothesis: true mean is not equal to 11.06
95 percent confidence interval:
 11.05714 11.07268
sample estimates:
mean of x
 11.06491
```

This is much better. Now p > 0.05 meaning the data is not significantly different from 11.06.

Two Sample Student’s t-test

```
data.new2 ← read.table(“tmp.data.head2”,
  +header = TRUE)
```

```
t.test(data.new[,2], data.new2[,2],
  +paired=TRUE)
```

Get values.

Non-Parametric Data

Necessary to use non-parametric statistics when the data is **not normal**.

Usually the Mann-Whitney U test is recommended.
In R can use the Wilcoxon Signed Rank Test which is very similar.

```
wilcox.test(data.new[,2], mu=11.066)
     Wilcoxon signed rank test with continuity correction

data:  data.new[, 2]
V = 518, p-value = 0.4092
alternative hypothesis: true location is not equal to 11.066

Warning message:
In wilcox.test.default(dov.head[, 2], mu = 11.066) :
  cannot compute exact p-value with ties
```

Statistical Power

The power level of one or more data sets describes how good the inferences drawn from your data are likely to be.

For example, you make 119 measurements of some quantity in millimeters Then you determine your data is best analyzed using a one sample Student's t-test, and you find that the mean of your sample measurements is 11.06 mm.

Now the question to ask is:

How likely is it that **someone else** who performs these measurements will also get a mean of 11.06 mm?

This is the "power" of your inference.

Statistical Power

In order to calculate power, it is necessary to calculate the effective sample size or "effect size" which is generally categorized as **small**, **medium**, or **large**.

In other words you need know how big the effect is expected to be. This you have to do by examining your data.

If the effect is small, more samples are needed to achieve the usually acceptable power level of at least 80%.

Statistical Power - Effect Size

The effect size associated with your data, is dependent on the type of test that is involved in analyzing the sample - ANOVA, Student's t-test, proportions, chisq (and also, when two tests are being compared, if the samples being analyzed are even or uneven). Once this is known, then the effect size can be calculated.

When the effect size has been determined, then the numbers of samples you have can be used to see what the power level of your data is. It is also possible to state the power level that you would like to have and compute the number of samples that you would need to achieve this power level.

In R, the package to obtain effect size and power is “pwr”.

Statistical Power – Calculate Effect Size

```
# enter data, examine it to determine which test is best for
# analysis (Student's t-test, ANOVA, proportions ...), and
# determine whether the effect is small, medium, or large.

# Read in the data:

data.new <- read.table("tmp.head_all", header=TRUE)

# examine data and decide on Student’s t-test
# invoke the library to be used

library(pwr)

cohen.ES(test="t", size="medium")
```

```
     Conventional effect size from Cohen (1982)
                              test = t
                              size = medium
                              effect.size = 0.5
```

Statistical Power–Calculate Power Given Sample Size

```
pwr.t.test(n = 119, d = 0.2, sig.level = 0.05, power = NULL,
  +type = "one.sample", alternative = "two.sided")
```

```
One-sample t test power calculation
          N = 119
          d = 0.2        ←  effect size
  sig.level = 0.05
      Power = 0.5808414
alternative = two.sided
```

58% power level with 119 samples; this is **unacceptable**

We would like to have 80% or 90% power level, thus we must obtain more samples.

But if the effect size is **medium**, i.e., d = 0.5, then the power = 0.9997192, which is acceptable

Statistical Power–Calculate Sample Size Given Power

```
pwr.t.test(n = NULL, d = 0.2, sig.level = 0.05,
  +power = 0.90, type = "one.sample",
  +alternative = "two sided”)
```

```
One-sample Student's t-test power calculation
                N = 264.6137
                d = 0.2
                sig.level = 0.05
                power = 0.9
                alternative = two.sided
```

effect size

For 90% power level, 265 samples are required if the effect size is small. If the power level is reduced to 80% power level, then fewer samples would be needed.

If effect size is **medium**, d = 0.5, the number of samples needed would only be only 44, which would clearly be a easier number of samples to obtain.

Writing your own functions in R

Simple Functions in R

One of the nice features in R is the ability to:

- Write new functions in R
- Write functions in R that perform many R functions in a sequence

  It is convenient to work in an editor like emacs, try things out, find all the components needed to do the job and then save the set as an R function (e.g., cup.measures).

- Call R functions from a C program
- Call C programs from an R function

Function to determine kidney function using a radionuclidic method:

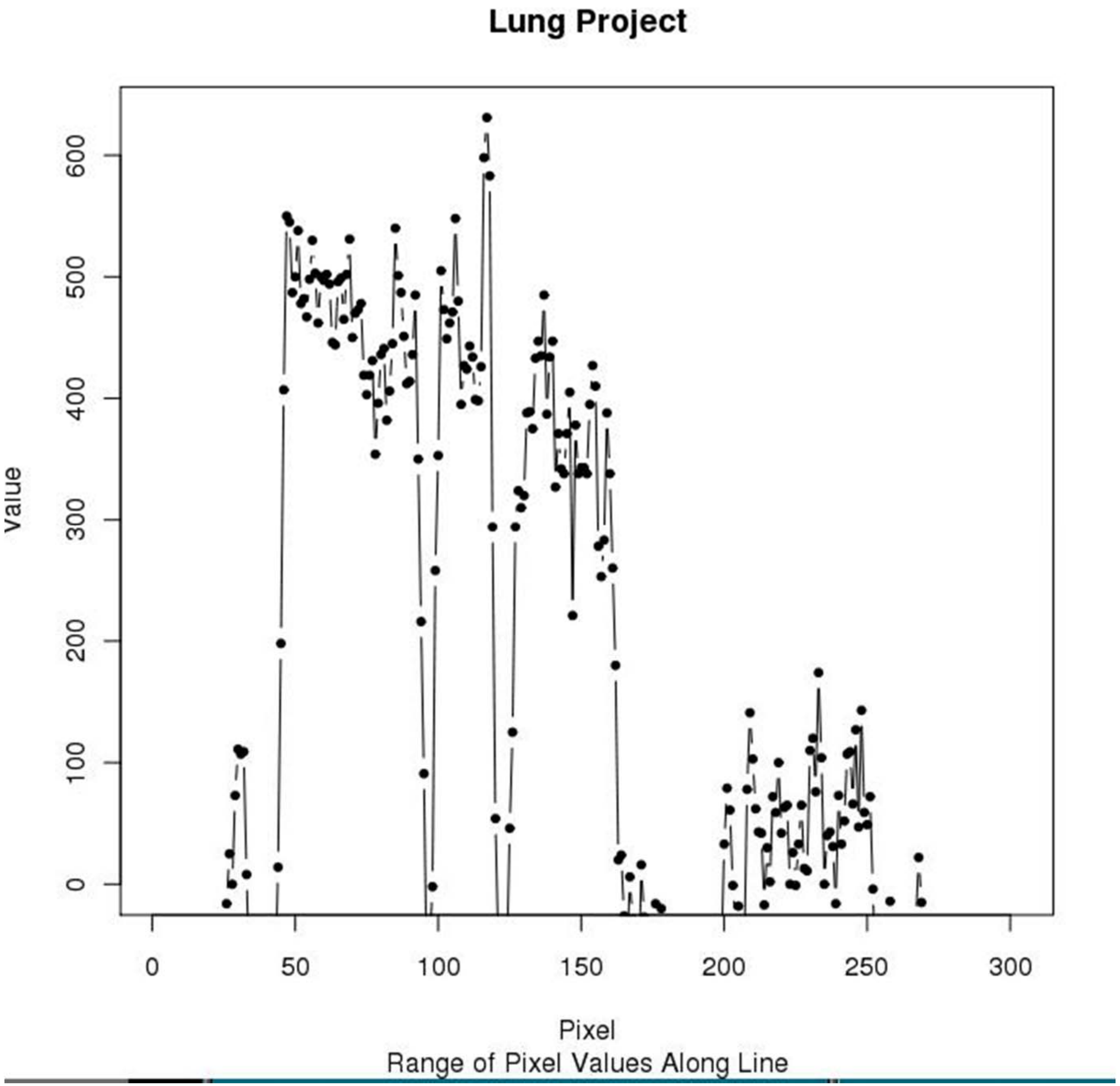
```
GFR1 <- function(dose, standard, filtert, time)
{
print("Dose (counts/min-ml) is:", quote = FALSE); print(dose)
print("standard volume (ml) is:", quote = FALSE); print(standard)
print("Filter (counts/min-ml) at T is:", quote = FALSE); print(filtert)
print("Time (minutes) is:", quote = FALSE); print(time)
filt <- filtert * 0.94
dose1 <- dose * standard
first <- (-0.278 * time) + 119.1 + (2405./time)
print("A is:", quote = FALSE); print(first)
third <- first * logb(dose1/filt)
print("first term is:", quote = FALSE); print(third)
fourth <- (2.866 * time) - 1222.9 - (16820./time)
print("B is:", quote = FALSE); print(fourth)
five <- (third + fourth);
print("GFR1 is :", quote = FALSE); print(five)
}
```

Note that the first four lines suggest the units that must be used for a successful result!

R Plot function

```
rplot<-function(filename) {
print(filename)
x11(xpos=-50)
ybuff <- (scan(filename))
ymaxvalue <- max(ybuff)
yminvalue <- min(ybuff)
plot(ybuff, ylab="Value", xlab="Pixel", type="b",
+ylim=range(yminvalue,ymaxvalue), main="Lung
+Project", sub="Range of Pixel Values Along
+Line",pch=20)
system("sleep 5")
}
```

Details of calling this function and generating the output shown on the following slide will be covered in a later lecture.


Lung Project
Value
0
100
200
300
400
500
600
0
50
100
150
200
250
300
Pixel
Range of Pixel Values Along Line

Call This Function from a C Program

In the C program:

```
/* open file for R instructions */
    sprintf(str,"%s/rfile.input", Dxmenu_Dir);
    if ( (fpR = creat(str, MODE)) == -1) {
      fprintf(stderr, "Cannot open file for R instructions.\n");
      return;
    }
 /* load the  file to be plotted */
    sprintf(datafilename, "%s_%d_%d_profile", name, slice,where);
/* Put this in the file opened above  and close the file*/
    sprintf(str, "rplot(\"%s\")\n", datafilename);
    write(fpR,str,sizeof(char)*strlen(str));
    close(fpR);
 /* run the R program */
   sprintf(str, "/bin/csh %s/rfile.sh %s", Path1, Path2);
   system(str);
  }
```

Form of rfile.input and rfile.sh

rfile.input: rplot("lung_case_three_256_profile")

rfile.sh:

```
# shell script for making a plot
echo $0 $1
# remove the output file before you try to write it
#
/bin/rm $1/rfile.output
#
# Run R from C program getting input from a command file
# and write the output to another file
#
/usr/bin/R -q --no-save <$1/rfile.input >$1/rfile.output
```

NOTE: the input/output directories could be different.

Normality Tests

The qqnorm plot produces a quantile-quantile plot which has the data being tested on one axis and the corresponding quantiles of a standard normal distribution on the other.

The density plot is a smooth version of the histogram; i.e., smooth estimates of the population frequency or probability density. Using “width=2iqd” in density, sets the degrees of smoothness of the density plot a good way.

Histogram and density = best picture of the population shape

The qqnorm and boxplot = best picture of outliers

Inference – Check data normality

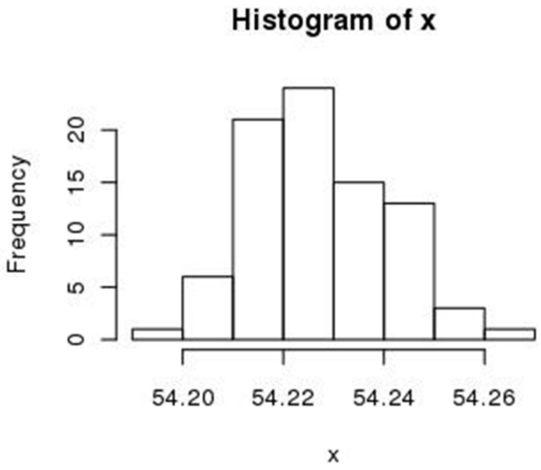
```
eda.shape <- function(x)
{
        par(mfrow = c(2, 2))
        hist(x)
        boxplot(x)
        iqd <- summary(x)[5] - summary(x)[2]
        plot(density(x, width = 2 * iqd), xlab = "x",
+               ylab = "", type = "l")
        qqnorm(x, pch = 1)
        qqline(x)
        invisible()
}
```

Arrange layout of multigraph

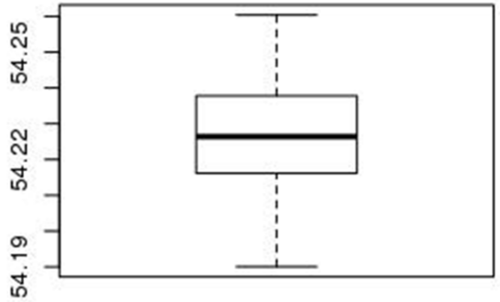
Line continuation with “+”

Result of simple check for Normality

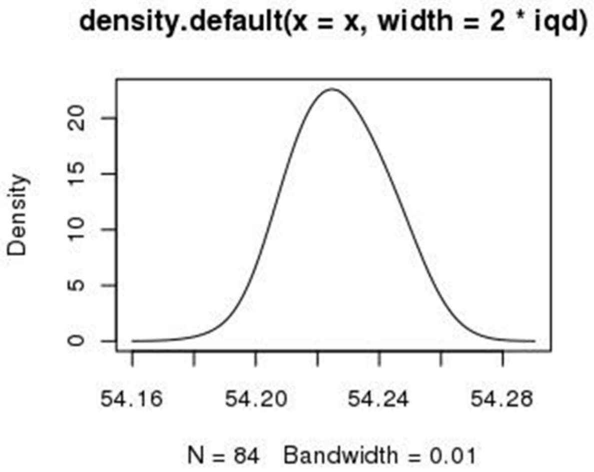
Histogram



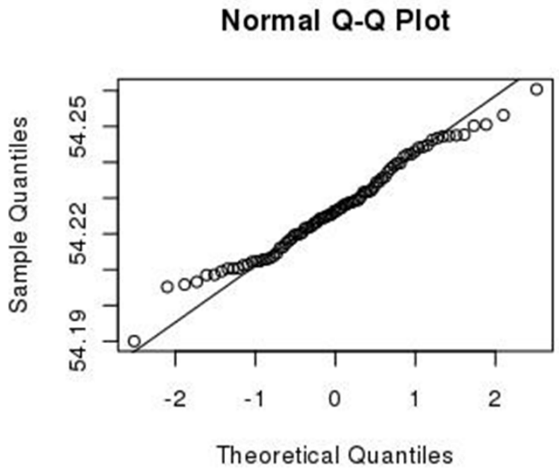
Boxplot



Plot of density



Quantiles-Quantiles plot



Inference #2 – put a title above a multiple graph

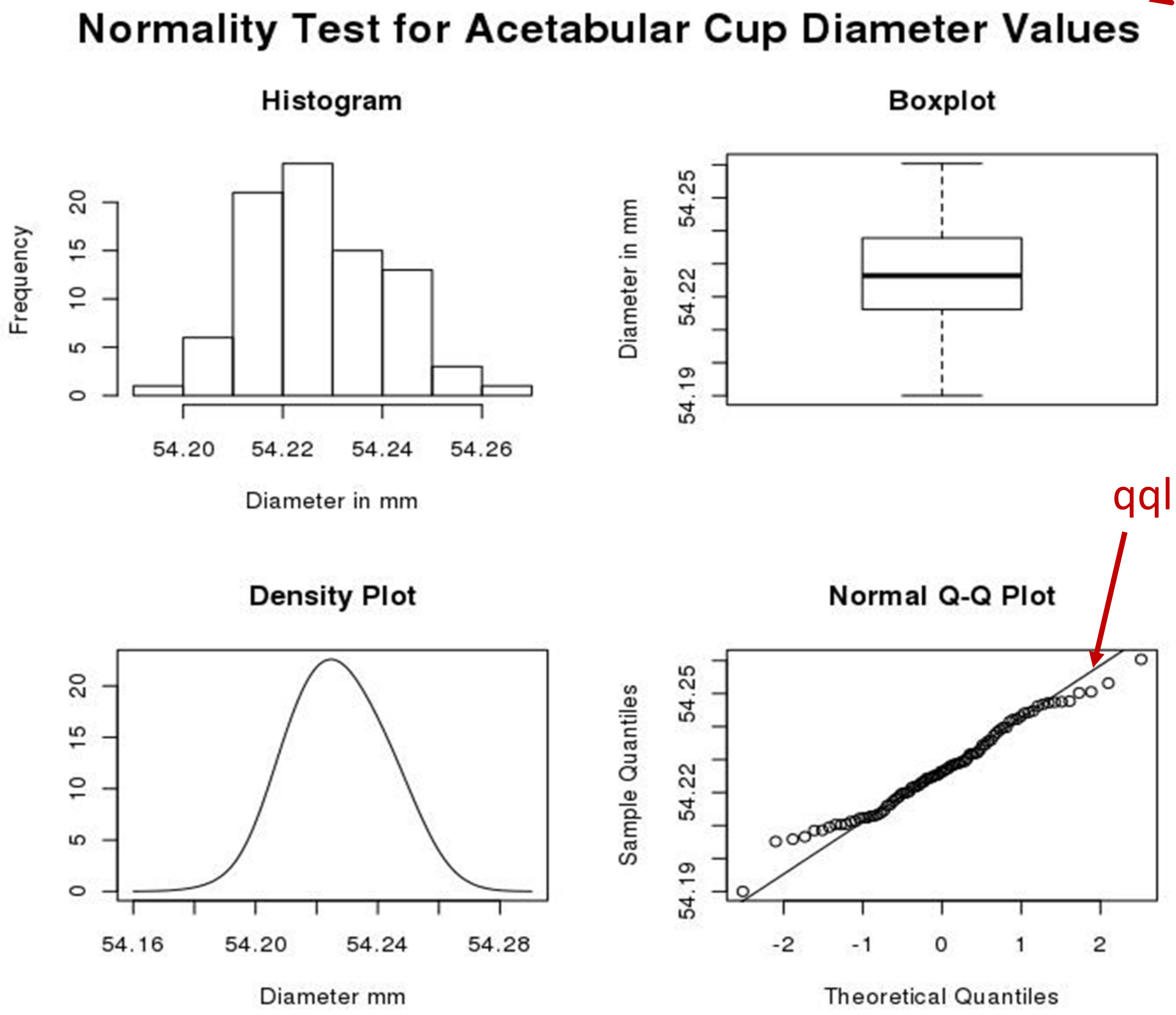
```
eda.shape <- function(x, y, z)
{

        par(oma=c(0,0,5,0))
        par(mfrow = c(2, 2))
         hist(x,xlab=z, main="Histogram")
         boxplot(x, ylab=z, main="Boxplot")
         iqd <- summary(x)[5] - summary(x)[2]
         plot(density(x, width = 2 * iqd), xlab = z, ylab = "", type = "l",
+                main = "Density Plot")
         qqnorm(x, pch = 1)
         qqline(x)
         invisible()
         mtext(y, side=3, outer=T, cex= 1.2)
}
```

x is the data
y is the main title
z is the Y label for the box plot
and the X label for the histogram & density plot

Add space for main title

Add main title

Result

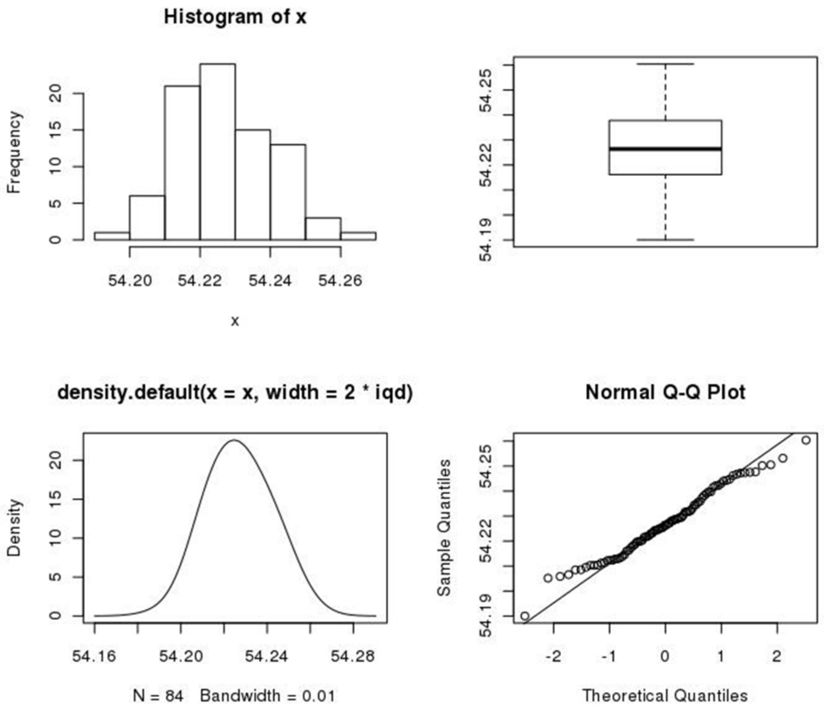


main title

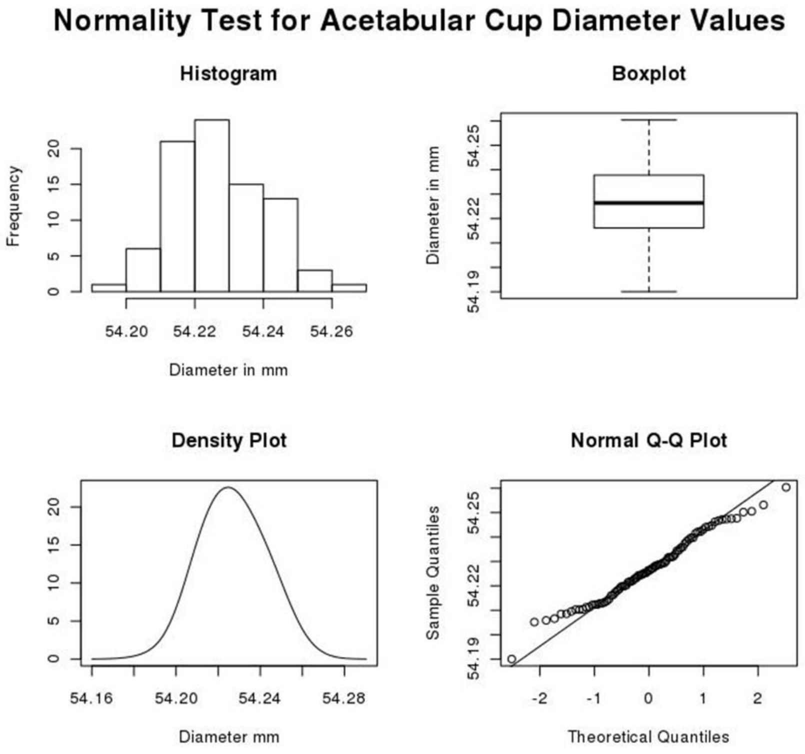
qqline

Compare the two versions

First version



Second version



Call a unix program from an R function

```
invitro.cals ← function(string, flag, string2, string3)
{
# string is the directory location of all the files to be used
# string2 is the directory location of all the files if a radionuclide is being modeled
# string3 is the spine data to be used for the bone marrow sample calculation (see 22)
print("flag:", quote = FALSE)
print(flag)
thalf <- scan(paste(string, "std.decay.time", sep = ""))
filename <- paste(string, "decay.time", sep = "")
if(unix(paste("test_file", filename), output = F)) {
                thalf.model <- scan(paste(string, "decay.time", sep = ""))
}
else {
                thalf.model <- scan(paste(string, "std.decay.time", sep = ""))
}
if((flag == 0) || (flag == 5) || (flag == 10) || (flag == 15)) {
  filename <- paste(string, "counts.blood", sep = "")
     if(unix(paste("test_file", filename), output = F)) {
                          starter.file <- dget(paste(string, "counts.blood", sep = ""))
                          flag.invitro <- 1
     }
    else { ….
                     }
               }   …..
```

Here is the call to the program

“if” Statements and “for” Loops

```
….

  if(flag.invitro == 0) {
     for(a in 1:count.len)
        counts[a, 1] <-
(as.numeric(starter.file[a]/(RF*(as.numeric(std))))*100000
     dput(counts, paste(string, "counts.blood", sep = ""))
   }
  else {
   for(a in 1:count.len)
     counts[a, 1] <- (as.numeric(starter.file[a]))
  }
  ……
}
```

Data analysis

Data Analysis – Linear Regression

Read in data and set up factors:

```
Measure <- factor(LETTERS[1:6])
Measure
# [1] A B C D E F
# Levels: A B C D E F
pat_dat <-scan("tmp.patient_t1")
# Read 42 items
# pat_dat
# [1]  2.97 -6.54  1.17  0.20  0.66 -0.59  1.62 -8.20 -1.11  0.14  1.98  2.14 1.41 -7.68  0.79
      -0.16 -0.70 -1.24
# [19] 1.11 -3.52  3.21 -0.02 -0.28  1.04 1.67 -6.24  1.36  0.35  0.74  1.09  0.07 -0.73
      1.32 -0.41 -0.57  0.62
# [37]  1.96 -1.07  2.78  0.97  0.57  0.05
Patient<-factor(c(rep(1,6), rep(2,6), rep(3,6), rep(4,6), rep(5,6), rep(6,6), rep(7,6)))
# Patient
# [1] 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 5 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7
# Levels: 1 2 3 4 5 6 7
```

Data Analysis – Linear Regression

Make a data frame:

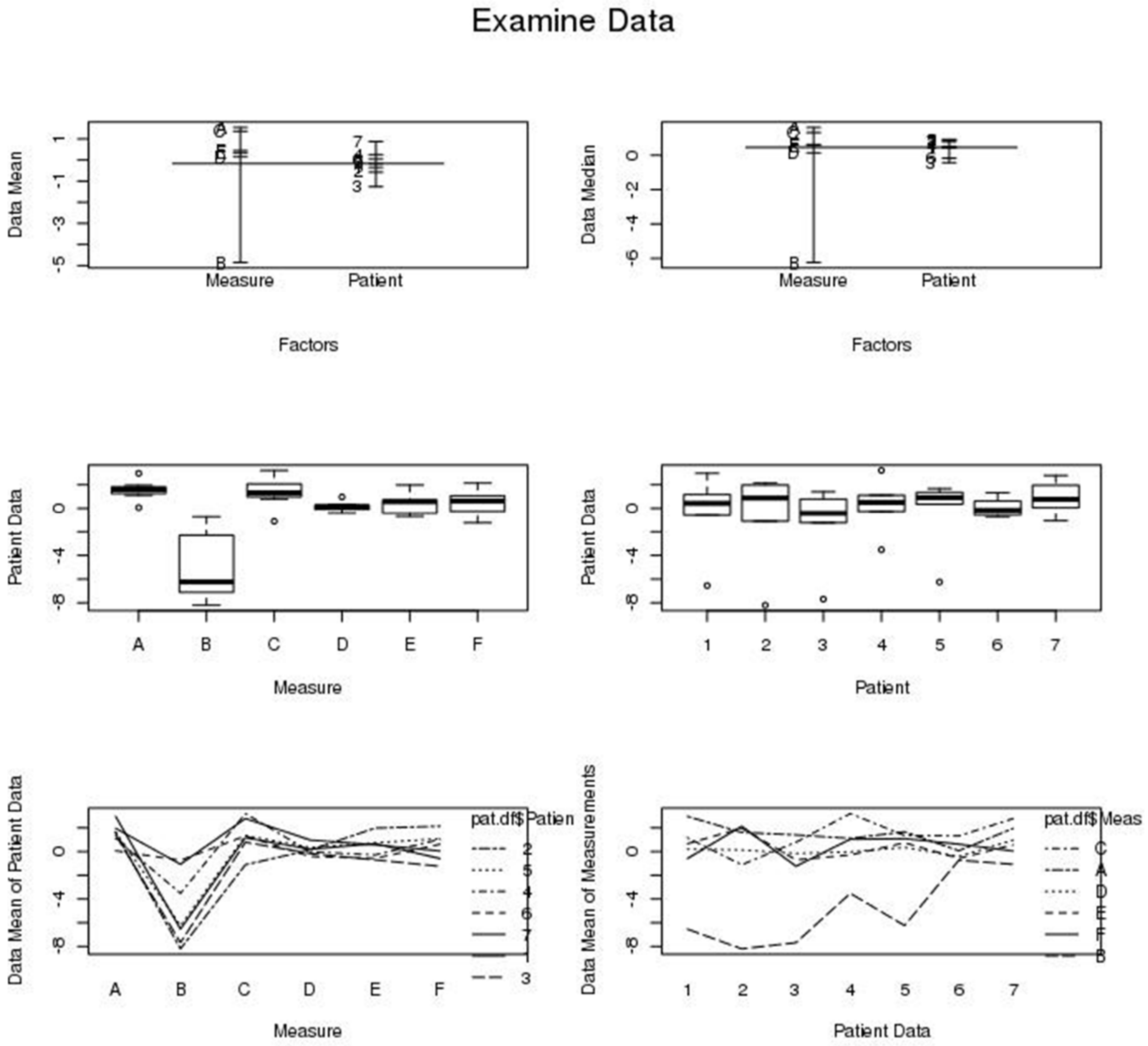
```
pat.df <-data.frame(Measure,Patient,pat_dat)

pat.df

   Measure Patient pat_dat
1       A       1    2.97
2       B       1   -6.54
3       C       1    1.17
4       D       1    0.20
5       E       1    0.66
6       F       1   -0.59
7       A       2    1.62
8       B       2   -8.20
9       C       2   -1.11
10      D       2    0.14
11      E       2    1.98
12      F       2    2.14
13      A       3    1.41
14      B       3   -7.68
15      C       3    0.79
16      D       3   -0.16
….
```

Data Analysis – Linear Regression

Graphically examine the data:

```
par(mfrow=c(3,2))
plot.design(pat.df)
plot.design(pat.df,fun=median)
plot(pat_dat~Measure+Patient,
  data=pat.df)
interaction.plot(pat.df$Measure,
  pat.df$Patient, pat.df$pat_dat)
interaction.plot(pat.df$Patient,
  pat.df$Measure, pat.df$pat_dat)
```

Data Analysis – Linear Regression



Data Analysis – Linear Regression

From the curves: there is no interaction between the patients and the measurements, but there is difference in the measurements for each patient

```
aov.pat1 <- aov(pat_dat~Measure*Patient, pat.df)
summary(aov.pat1)
                Df   Sum Sq Mean Sq
Measure          5   195.768  39.154
Patient          6    16.301    2.717
Measure:Patient 30   71.452    2.382
```

*   means interaction between these factors, i.e. Measure depends upon Patient

+ means difference between these factors i.e. Measure is independent of Patient

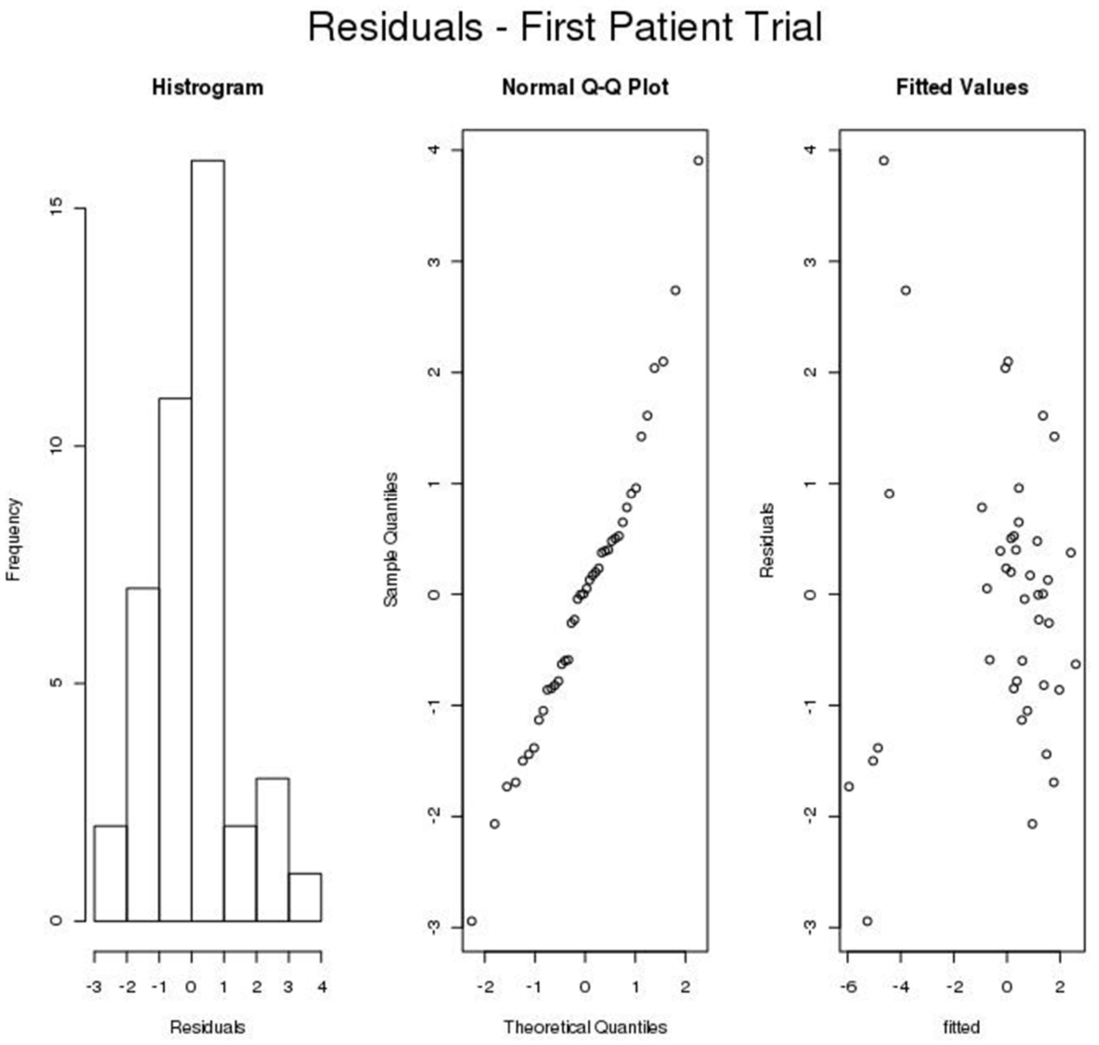
```
aov2.pat1 <- aov(pat_dat~Measure+Patient, pat.df)
summary(aov2.pat1)
           Df  Sum Sq Mean Sq   F value    Pr(>F)
Measure     5  195.768   39.154  16.4391   8.162e-08 ***
Patient     6   16.301    2.717   1.1407   0.3634
Residuals  30  71.452   2.382
```

Data Analysis – Linear Regression

Examine the residuals graphically:

```
par(oma=c(0,0,5,0))
par(mfrow=c(1,3))
hist(resid(aov2.pat1), main = "Histrogram",
   xlab +="Residuals", ylab = "Frequency")
qqnorm(resid(aov2.pat1))
plot(fitted(aov2.pat1), resid(aov2.pat1),
   xlab = "fitted", ylab += "Residuals",
   main = "Fitted Values")
mtext("Residuals - First Patient Trial", side=3,
   +outer=TRUE, cex=1.5)
```

Plots of the residuals



Multiple Plots 1

```
eda10.shape <- function(x,y,t,s)
{
        par(oma = c(10,0,5,0))
        par(fig = c(0.1, 0.4, 0.25, 0.75), mar = c(2,0.5,0.5,0.5), mgp = c(0,0.5,0))
#       par(fig = c(0.1, 0.4, 0.25, 0.75))
        iqd <- summary(x)[5] - summary(x)[2]
        plot(density(x, width = 2 * iqd), xlab = "", type = "l", ylim=c(0, 1.4), pch=15,
+     lty = 1, ylab = "", main = "")
        iqd <- summary(y)[5] - summary(y)[2]
        points(density(y, width = 2 * iqd), type = "l", xlab = "", ylab = "", pch=17,  lty = 2)
        iqd <- summary(t)[5] - summary(t)[2]
        points(density(t, width = 2 * iqd), type = "l", pch=18, xlab = "", ylab = "", lty = 3)
        iqd <- summary(s)[5] - summary(s)[2]
        points(density(s, width = 2 * iqd), type = "l", pch=16, xlab = "", ylab = "", lty = 4)
     legend(locator(1), legend=list("Model Studies", "Patients:", "  Examiner 1",
+    "  Examiner 2", "Inter-examiner"), lty=c(1,-1,2,3,4), pch=c(16,-1,17,18,15), cex=0.70)
     mtext("Density Plot of Angular Differences - Two Trials", side = 1, outer = T,
+    cex = 0.85, at = 0.2)
     mtext("Precision Experiments for 10 Patients and 24 Model Studies",
+    side = 3, outer = T, cex = 1.5)
#
```

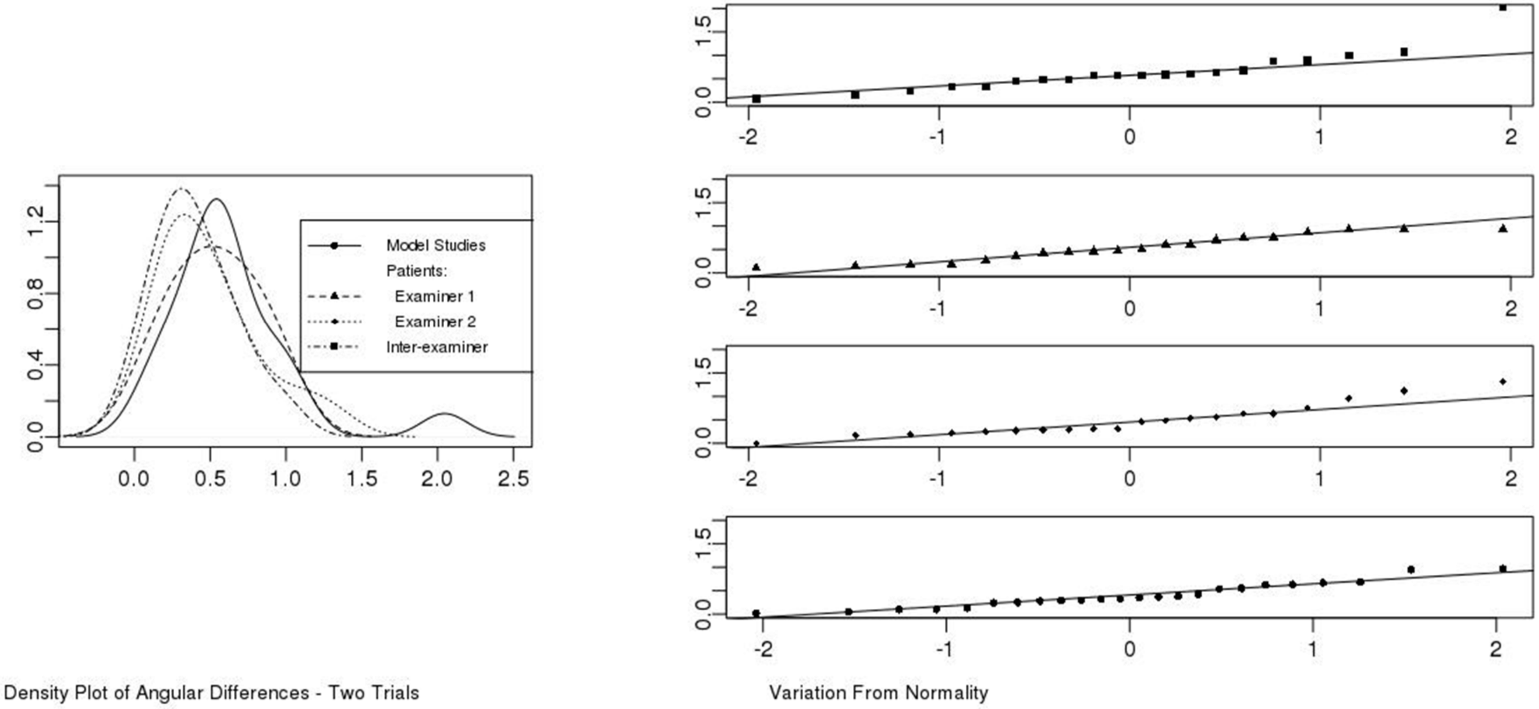
Multiple Plots 2

```
        par(fig=c(0.5,1, 0.75, 1),mar = c(2,0.5,0.5,0.5),mgp=c(0,0.5,0), new=TRUE)
        qqnorm(x, pch = 15, xlab ="", ylab="", cex = 0.75, main="",ylim=c(0, 2.0))
        qqline(x)
        invisible()
        par(fig=c(0.5,1, 0.5, 0.75),new=TRUE)
        qqnorm(y, pch = 17, xlab ="", ylab="", cex = 0.75,main="",ylim=c(0, 2.0))
        qqline(y)
        invisible()
        par(fig=c(0.5,1, 0.25, 0.5),new=TRUE)
        qqnorm(t, pch = 18, xlab ="", ylab="", cex = 0.75, main="",ylim=c(0, 2.0))
        qqline(t)
        invisible()
        par(fig=c(0.5,1, 0, 0.25),new=TRUE)
        qqnorm(s, pch = 16, xlab ="", ylab="", cex = 0.75, main="",ylim=c(0, 2.0))
        qqline(s)
        invisible()
    mtext("Variation From Normality", side = 1, outer = T, cex = 0.85, at = 0.6)
}
```

Use as: eda10.shape(file1,file2,file3, file4)

EXAMPLE:
eda10.shape(scan("lot_hen_pat"), scan("lotta_pat1-2"), scan("hen_pat1-2"), scan("tmp.lotta1-2"))

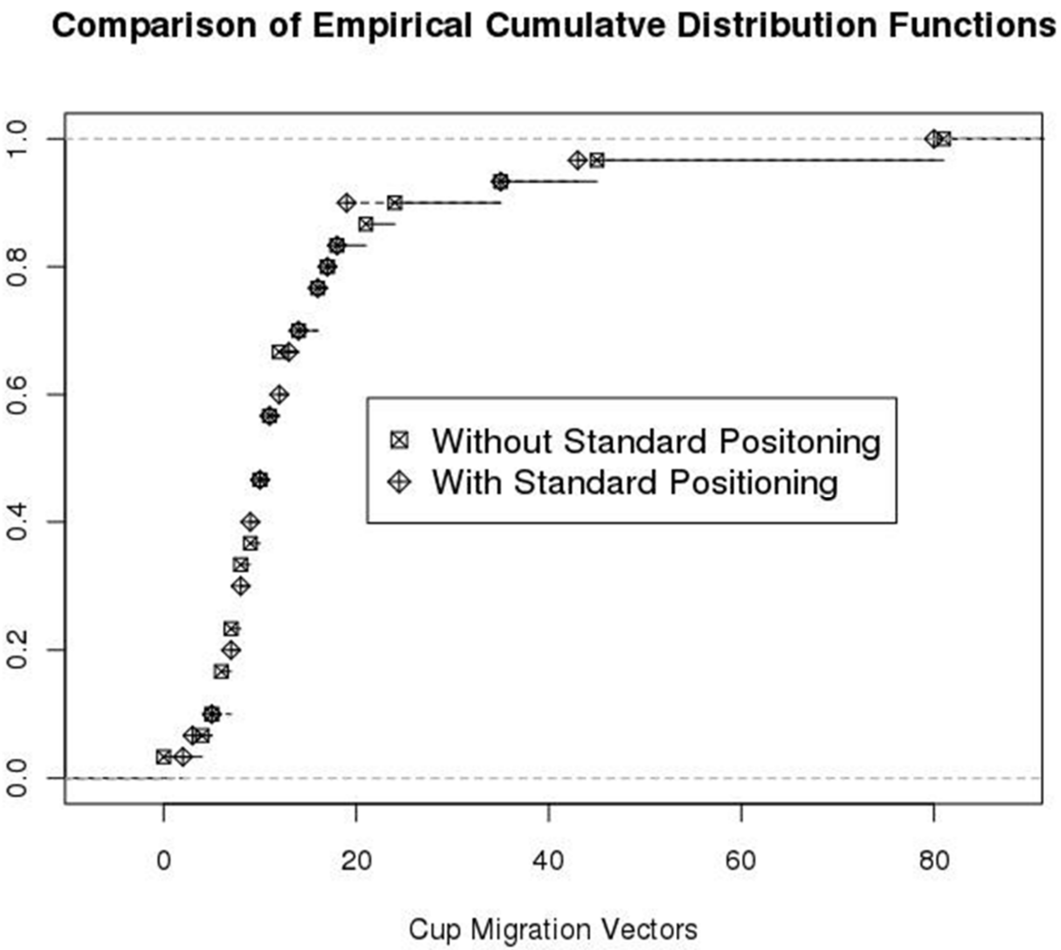
Precision Experiments for 10 Patients and 24 Model Studies



Multiple ECDF functions

```
 plot(ecdf(scan("tmp.after_std")),main="Comparison of
+  Empirical Cumulatve Distribution Functions", xlab = "Cup
+  Migration Vectors", ylab= "", sub = "", pch = 7)

plot(ecdf(scan("tmp.before_std")), add=TRUE, pch=9, lty = 2)


legend(locator(1), c("Without Standard Positoning","With
+  Standard Positioning"), pch=c(7,9), cex = 1.2)
```

Result with multiple ECDFs



Using Excel for statistics and plotting

Experiment 1 (again)

Captured packets using Wireshark during a long (2150.12 second) VoIP call

⇒ at least: 107,505 RTP packets in each direction

⇒429 RTCP packets in one direction

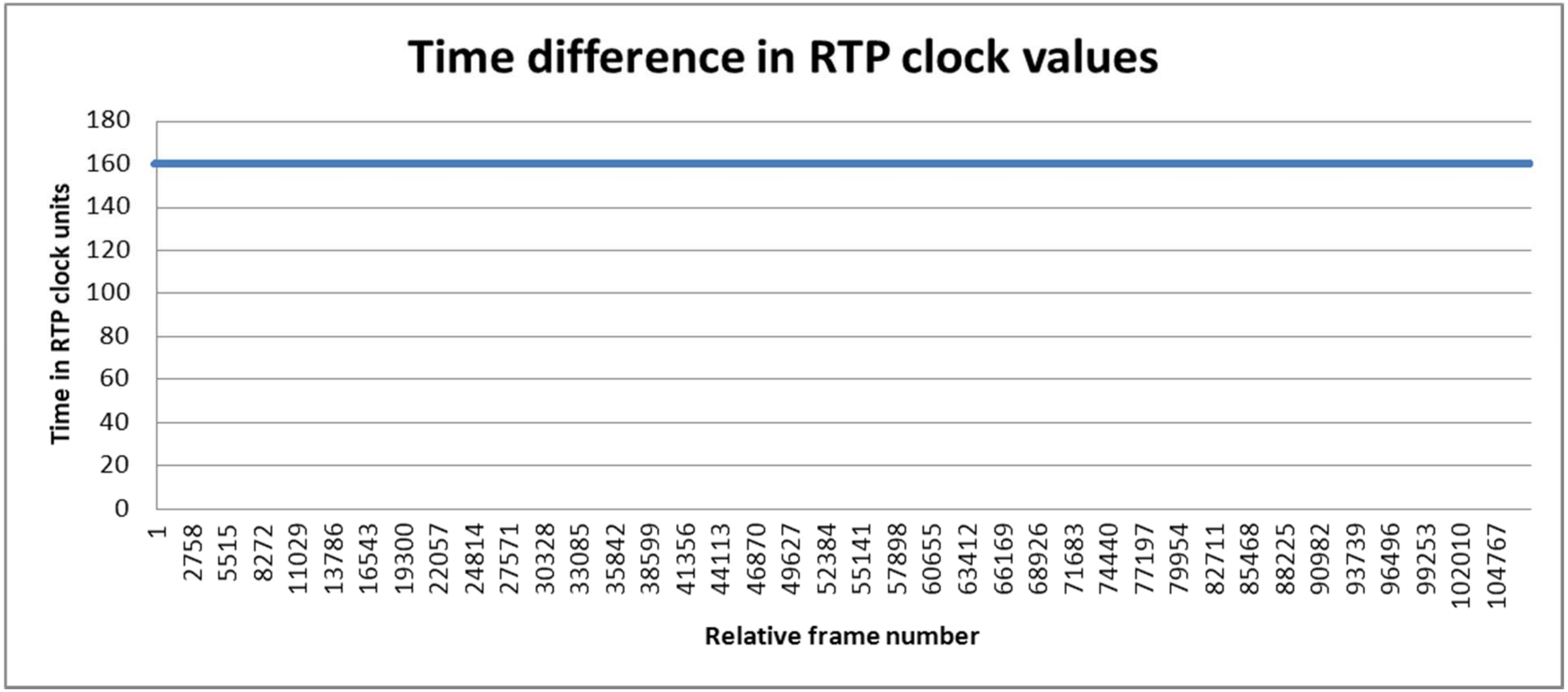
From network to local user agent

Raw output fom Microsoft Excel 2010 (Beta)

Difference in RTP clock from previous sample	
Mean	160
Standard Error	0
Median	160
Mode	160
Standard Deviation	0
Sample Variance	0
Kurtosis	#DIV/0!
Skewness	#DIV/0!
Range	0
Minimum	160
Maximum	160
Sum	17200960
Count	107506
Confidence Level(95.0%)	0

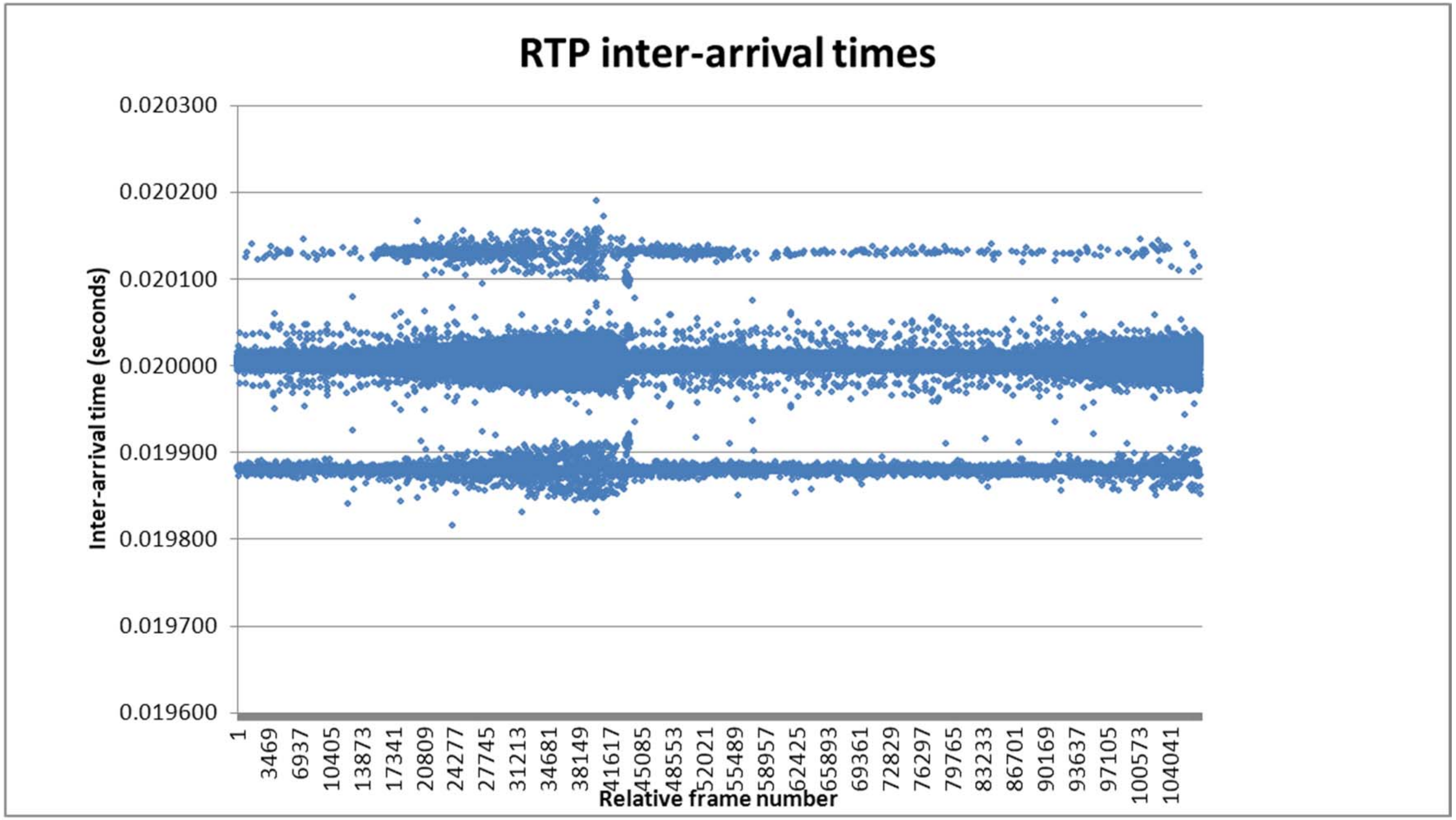
Inter-arrival times (in seconds) of RTP packets	
Mean	0.019999999
Standard Error	9.28526E-08
Median	0.020004
Mode	0.020005
Standard Deviation	3.04446E-05
Sample Variance	9.26874E-10
Kurtosis	12.36652501
Skewness	-2.054662184
Range	0.000374
Minimum	0.019815
Maximum	0.020189
Sum	2150.11991
Count	107506
Confidence Level(95.0%)	1.8199E-07

First look at the RTP clock differences

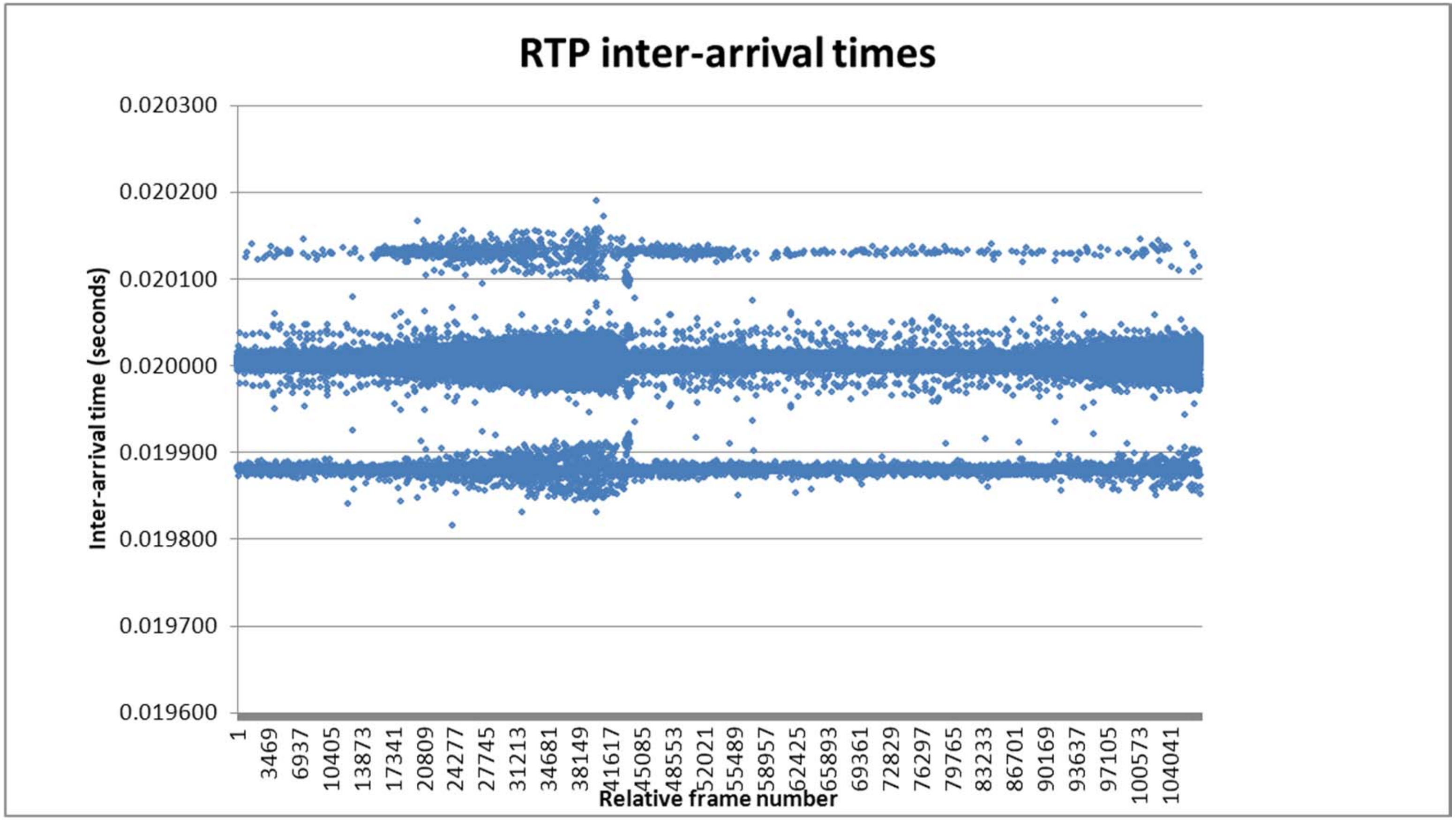


Conclusion: 160 audio samples per frame, with a frame time of 0.20 ms
⇒ 8 K sample/second sampling rate – consistent with ITU-T G.711 PCMA encoding

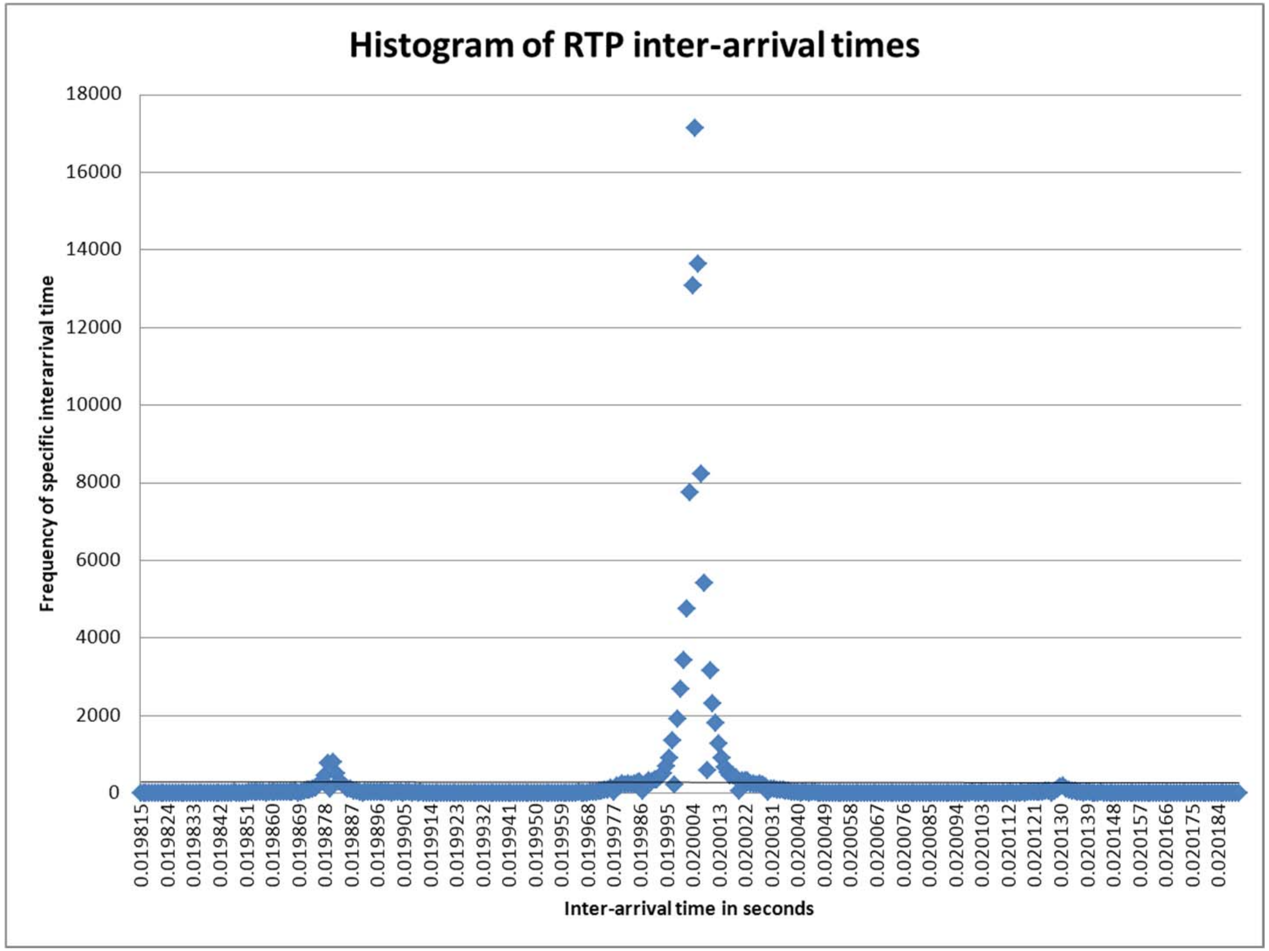
RTP inter-arrival times as measured by Wireshark



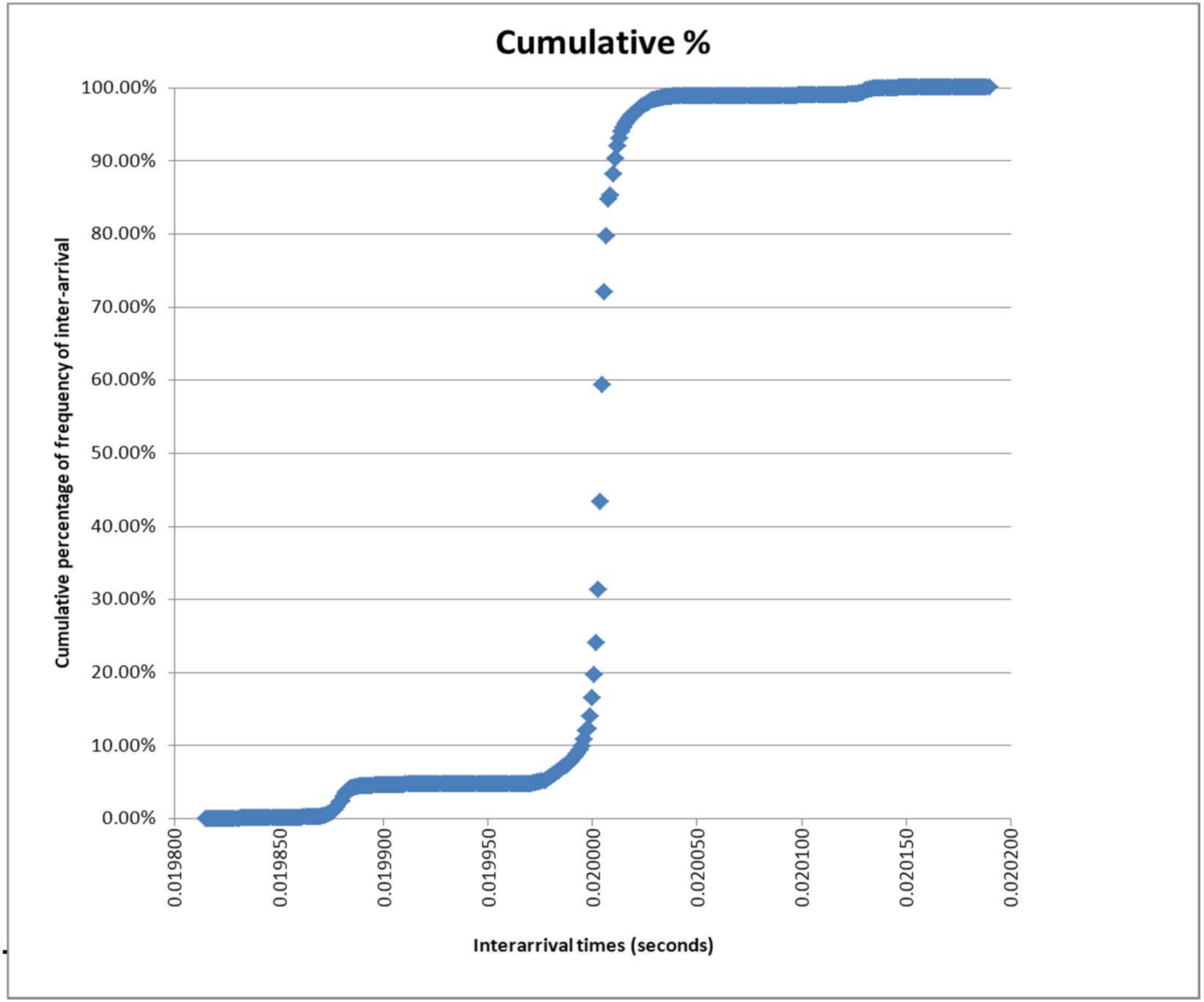
Re-scale vertical axis



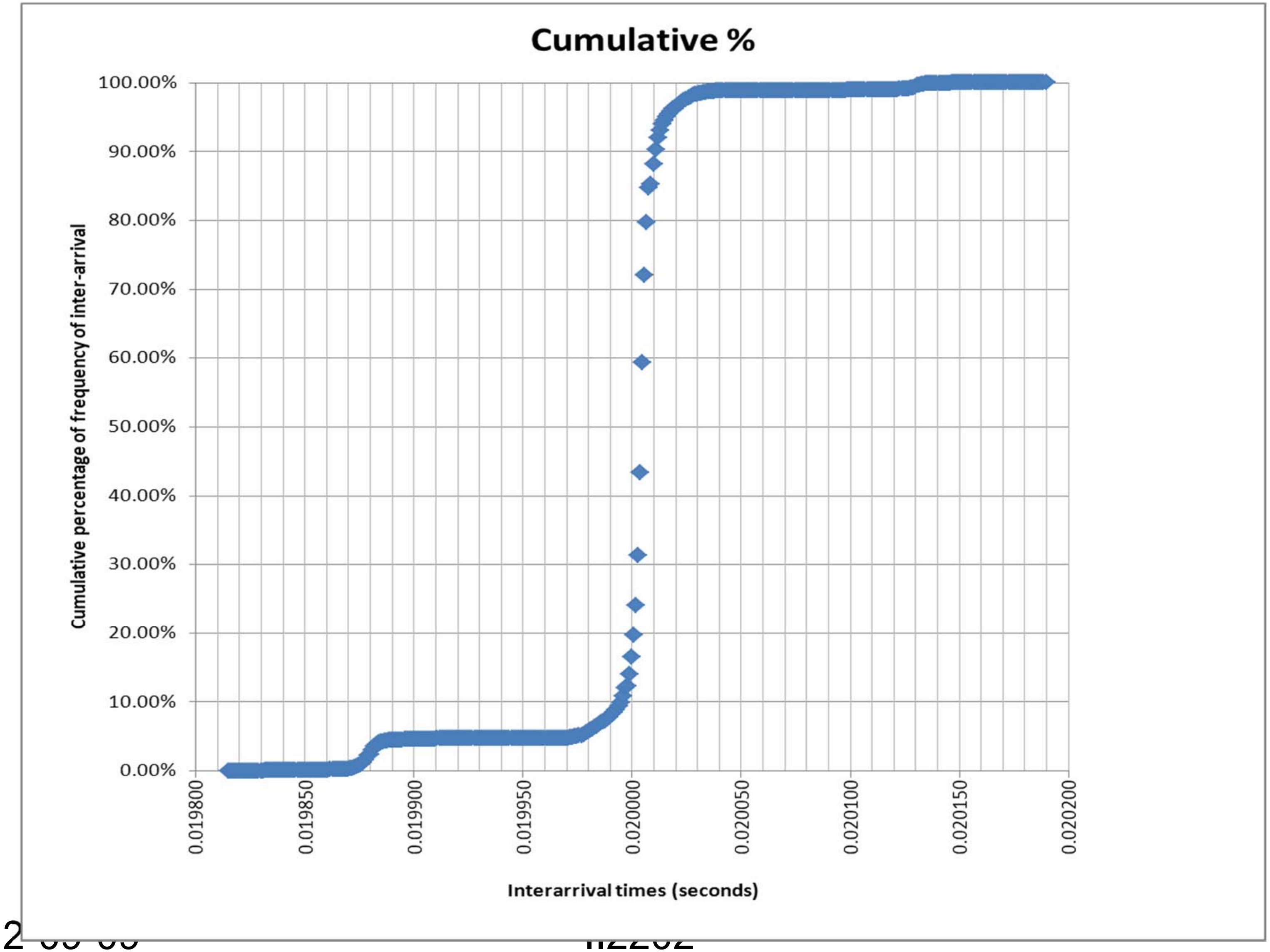
Compute histogram



Plot as a Cumulative Distibution



Add grid lines



As numbers - near median

	seconds	frequency	Cumulative %
	0.019995	687	9.92%
	0.019996	895	10.75%
	0.019997	1334	11.99%
	0.019998	209	12.18%
	0.019999	1898	13.95%
Mean	0.020000	2671	16.44%
	0.020001	3403	19.60%
	0.020002	4747	24.02%
	0.020003	7742	31.22%
Median	0.020004	13059	43.37%
Mode	0.020005	17121	59.30%
	0.020006	13630	71.98%
	0.020007	8211	79.62%
	0.020008	5404	84.64%
	0.020009	570	85.18%
	0.020010	3158	88.11%
	0.020011	2305	90.26%
	0.020012	1787	91.92%
	0.020013	1262	93.09%
	0.020014	886	93.92%

With varying numbers of samples

Descriptive Statistics	First 100	First 1K	First 10K	First 100K
Mean	0.02000071	0.020000066	0.020000004	0.02
Standard Error	2.12714E-06	7.53406E-07	2.51164E-07	9.69855E-08
Median	0.020005	0.020004	0.020004	0.020004
Mode	0.020005	0.020005	0.020005	0.020005
Standard Deviation	2.12714E-05	2.38248E-05	2.51164E-05	3.06695E-05
Sample Variance	4.52471E-10	5.67621E-10	6.30831E-10	9.40618E-10
Kurtosis	28.87137928	21.46428225	19.07376827	12.23083198
Skewness	-5.453831468	-4.509853108	-3.831289593	-2.003065575
Range	0.000135	0.000252	0.000277	0.000374
Minimum	0.01988	0.019872	0.019868	0.019815
Maximum	0.020015	0.020124	0.020145	0.020189
Sum	2.000071	20.000066	200.000044	1999.999951
Count	100	1000	10000	100000
Confidence Level(95.0%)	4.2207E-06	1.47844E-06	4.92331E-07	1.9009E-07

With varying numbers of samples

Descriptive Statistics	First 100	First 1K	First 10K	First 100K
Mean				2
Standard Error				8
Median				4
Mode				5
Standard Deviation				5
Sample Variance				.0
Kurtosis				8
Skewness				5
Range				4
Minimum				.5
Maximum				39
Sum				1
Count				0
Confidence Level(95.0%)				7

```
foo<-function(n){
v <-1:12
v[1]=mean(To_Chip_RTP_delay[1:n])
v[2]=std.error(To_Chip_RTP_delay[1:n])
v[3]=names(sort(-table(To_Chip_RTP_delay[1:n])))[1]
v[4]=sd(To_Chip_RTP_delay[1:n])
v[5]=var(To_Chip_RTP_delay[1:n])
v[6]=kurtosis(To_Chip_RTP_delay[1:n])
v[7]=skewness(To_Chip_RTP_delay[1:n])
v[8]=min(To_Chip_RTP_delay[1:n])
v[9]=max(To_Chip_RTP_delay[1:n])
v[10]=sum(To_Chip_RTP_delay[1:n])
v[11]=length(To_Chip_RTP_delay[1:n])
v[12]=qnorm(0.965)*std.error(To_Chip_RTP_delay[1:n])
return(v)}
seq1<-c(foo(100),foo(1000),foo(10000),foo(100000))
mat1<-matrix(seq1,  ncol=4)
```

Applying a function to a list of arguments

Descriptive Statistics	First 100	First 1K	First 10K	First 100K
Mean	[illegible]	[illegible]	[illegible]	[illegible]
Standard Error				
Median				
Mode				
Standard Deviation				
Sample Variance				
Kurtosis				
Skewness				
Range				
Minimum				
Maximum				
Sum				
Count				
Confidence Level(95.0%)				

```
foo<-function(m,n){v <- 1:12
v[1]=mean(m[1:n])
v[2]=std.error(m[1:n])
v[3]=names(sort(-table(m[1:n])))[1]
v[4]=sd(m[1:n])
v[5]=var(m[1:n])
v[6]=kurtosis(m[1:n])
v[7]=skewness(m[1:n])
v[8]=min(m[1:n])
v[9]=max(m[1:n])
v[10]=sum(m[1:n])
v[11]=length(m[1:n])
v[12]=qnorm(0.965)*std.error(m[1:n])
return(v)}

fee<-function(n) {foo(To_Chip_RTP_delay, 10^n)}

lapply(c(2:5), fee)
[[1]] [1] "0.0200006800000119"   "2.12697347407497e-06" "0.0200049999984913“
      [4] "2.12697347407497e-05" "4.52401615941855e-10" "30.3672958382318“ …
```

Uplink inter-arrival times stats

```
library(plotrix);library(moments)foo
<-function(m,n){v <- 1:12
v[1]=mean(m[1:n])
v[2]=std.error(m[1:n])
v[3]=names(sort(-table(m[1:n])))[1]
v[4]=sd(m[1:n])
v[5]=var(m[1:n])
v[6]=kurtosis(m[1:n])
v[7]=skewness(m[1:n])
v[8]=min(m[1:n])
v[9]=max(m[1:n])
v[10]=sum(m[1:n])
v[11]=length(m[1:n])
v[12]=qnorm(0.965)*std.error(m[1:n])
return(v)}
```

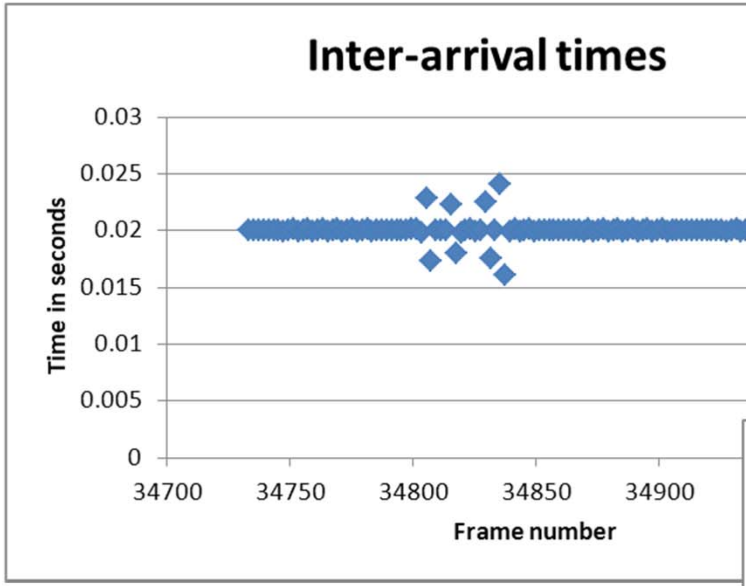
```
foo(From_Chip_RTP_delay, 10^5)
"0.02000027577"
"3.63331229733734e-07"
"0.0200049999984913"
"0.000114895423102849"
"1.32009582499827e-08"
"742.581556664333"
"0.633658007213615"
"0.0136249999995925"
"0.0263840000006894"
"2000.027577"
"100000"
"6.58323732971544e-07"
```

What to put into a report:

```
       0.020000 s
       3.63e-07 s
       0.020005 s
       0.000115 s
       1.32e-08 s
     742.58
       0.634
       0.013625 s
       0.026384 s
    2000.027577 s
100000
       6.58e-07 s
```

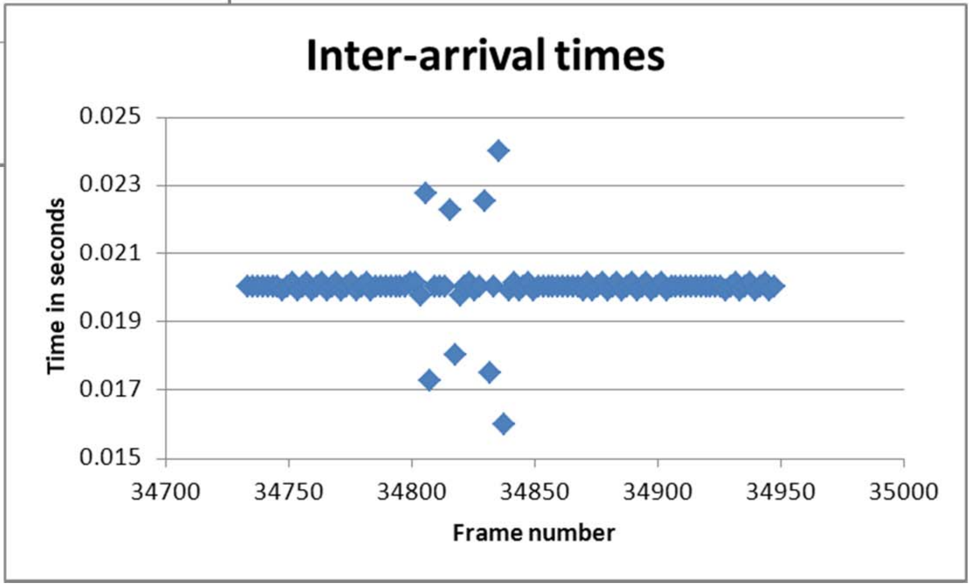
Truncated to meaningful number of digits, added units, decimal align the numbers, set in fixed width font (Courier)

Zooming in on behavior

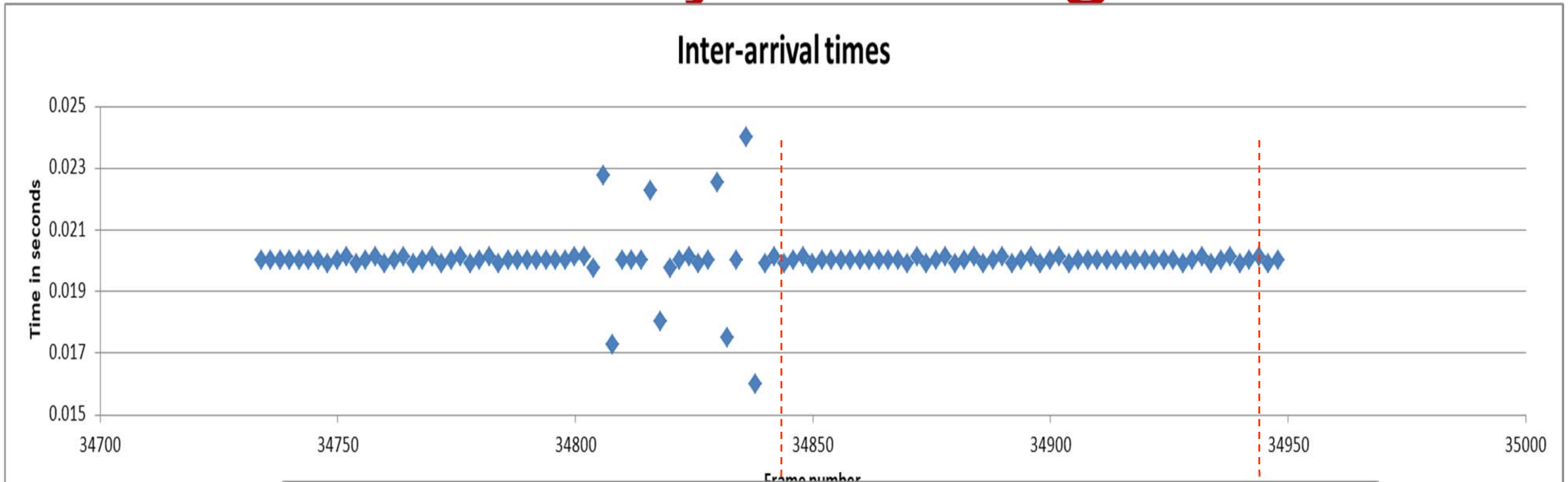


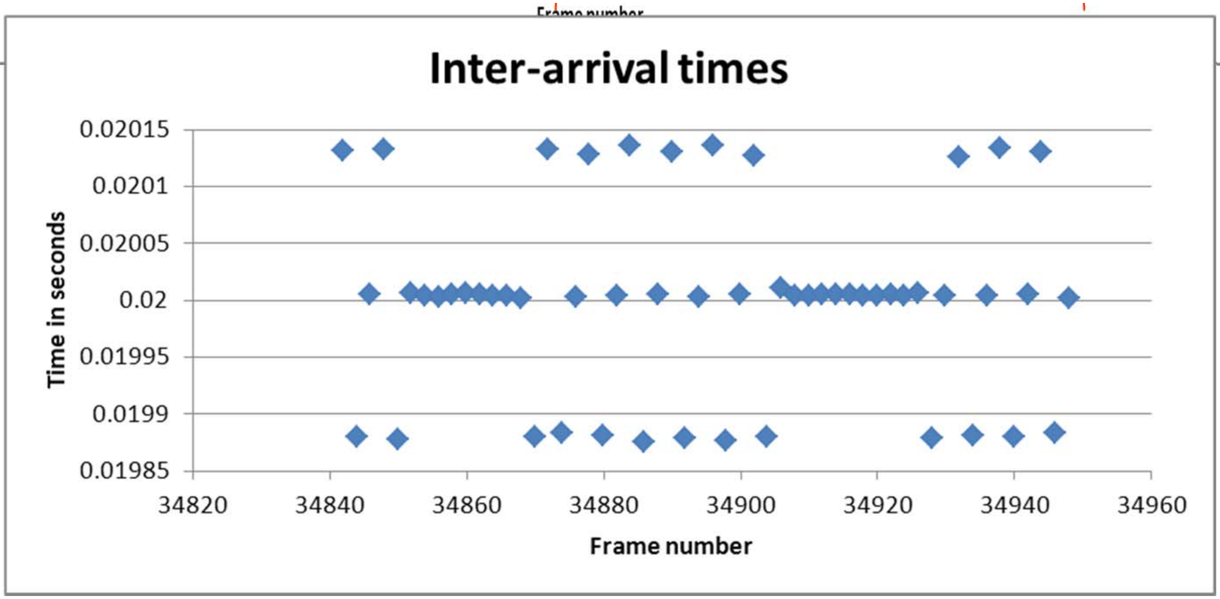
Note that the plot is now a **scatter plot**.

Re-scale

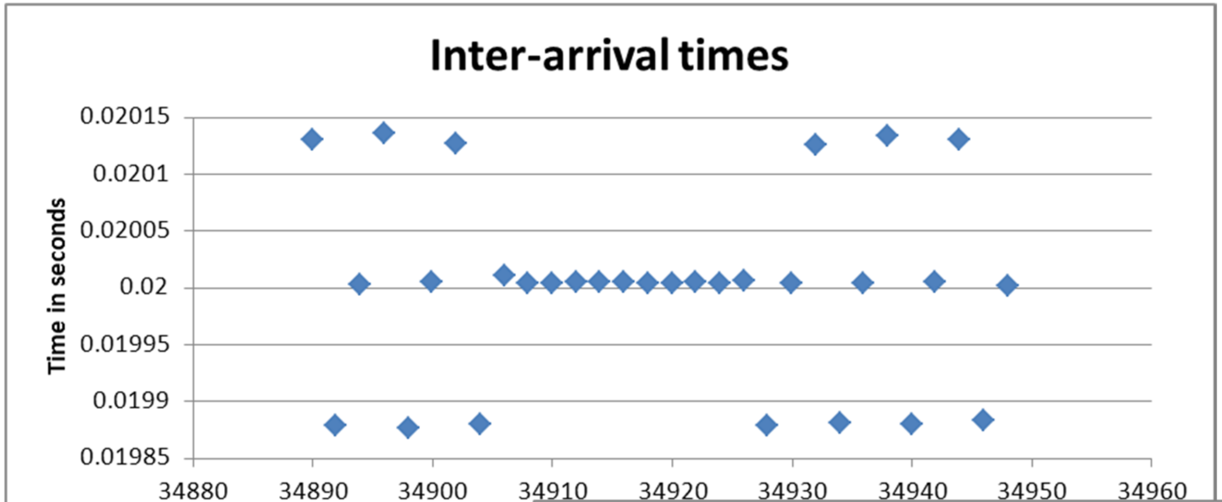


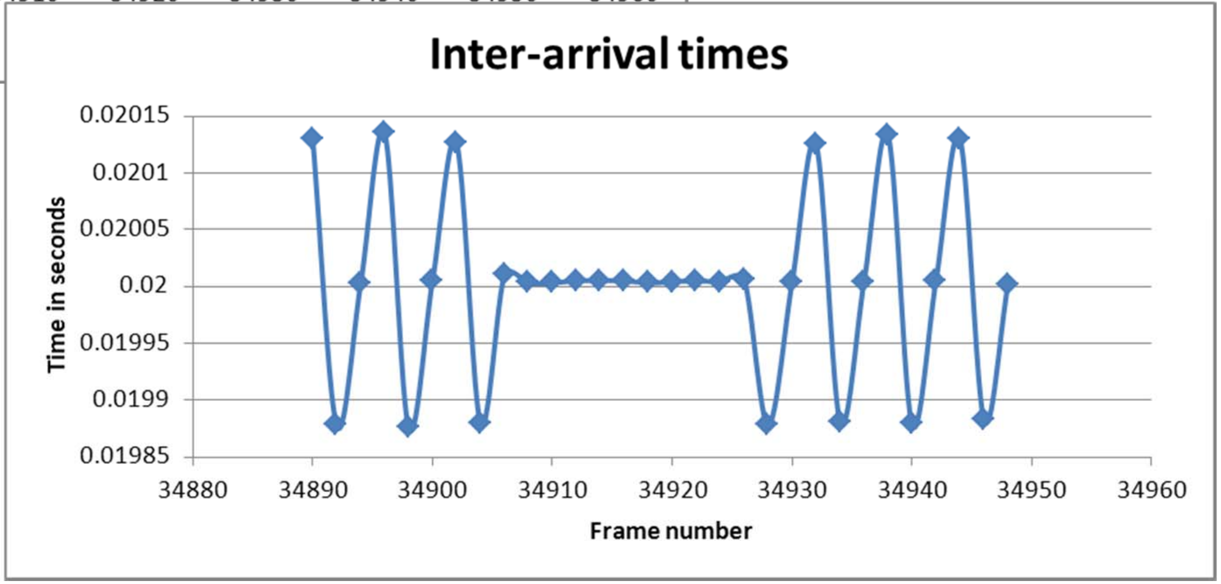
Looking in more detail at a relatively “flat” region





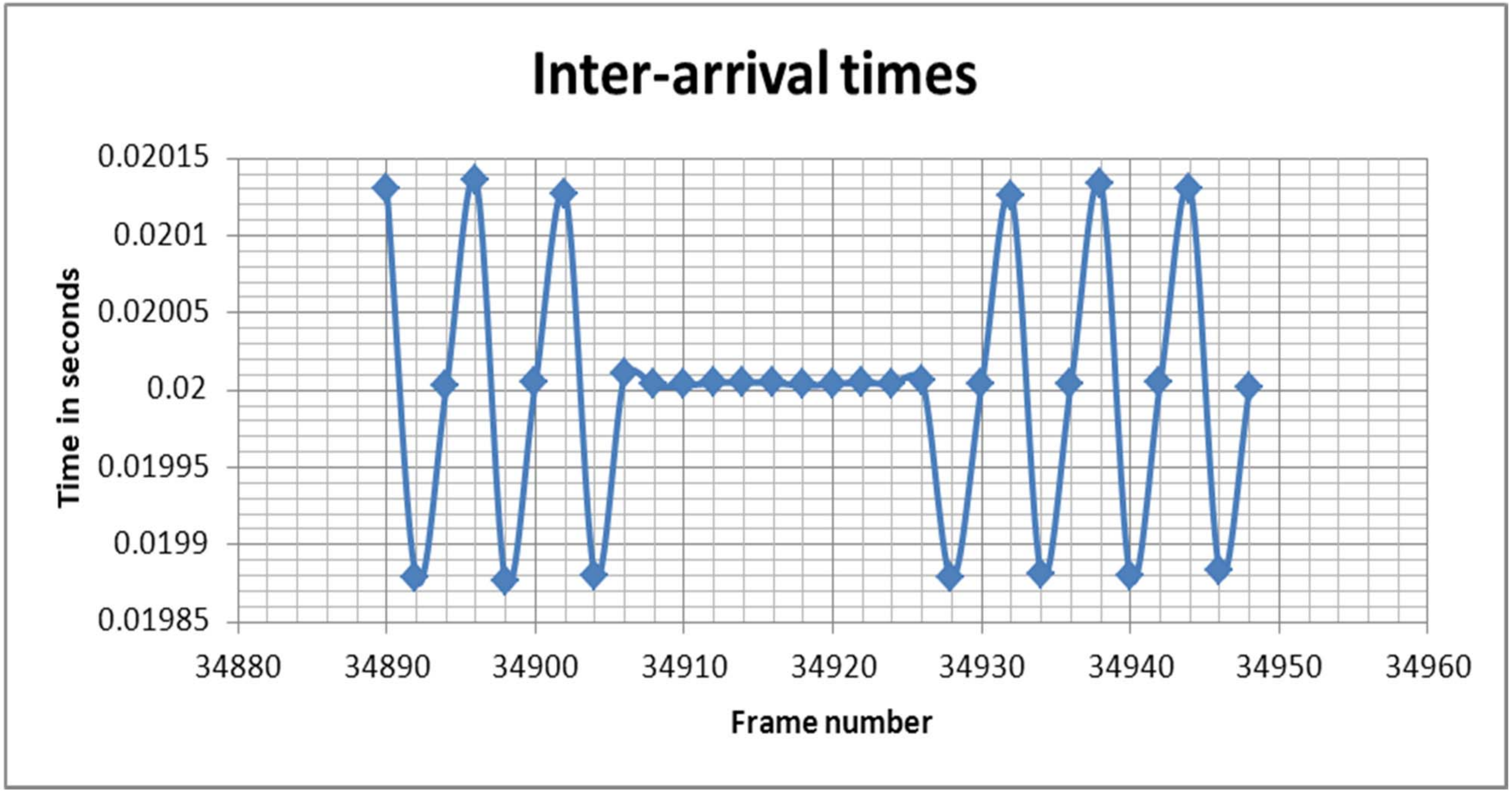
Is there some pattern?



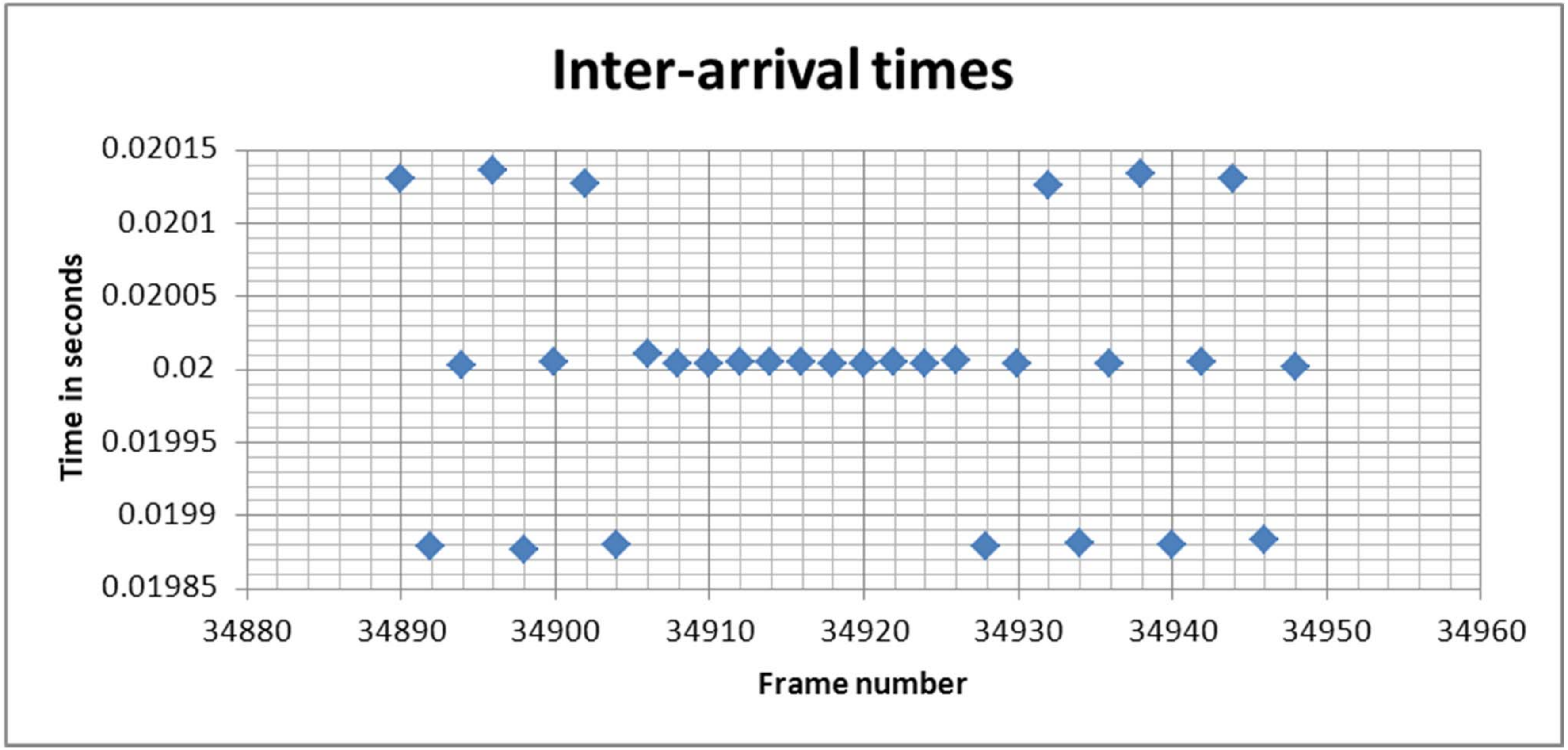


Not continuous data, but connecting with lines shows the values oscillate

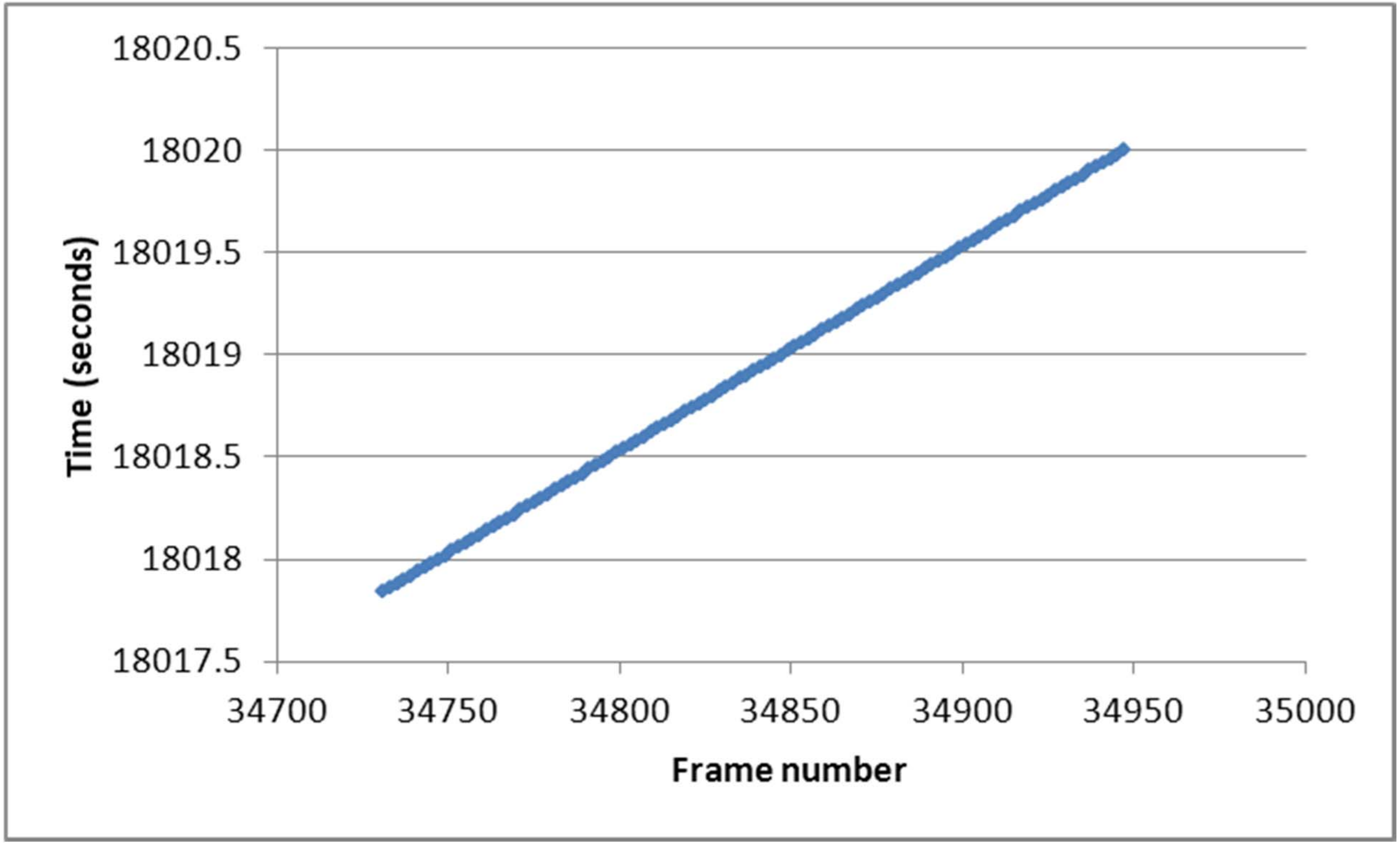
Adding grid lines



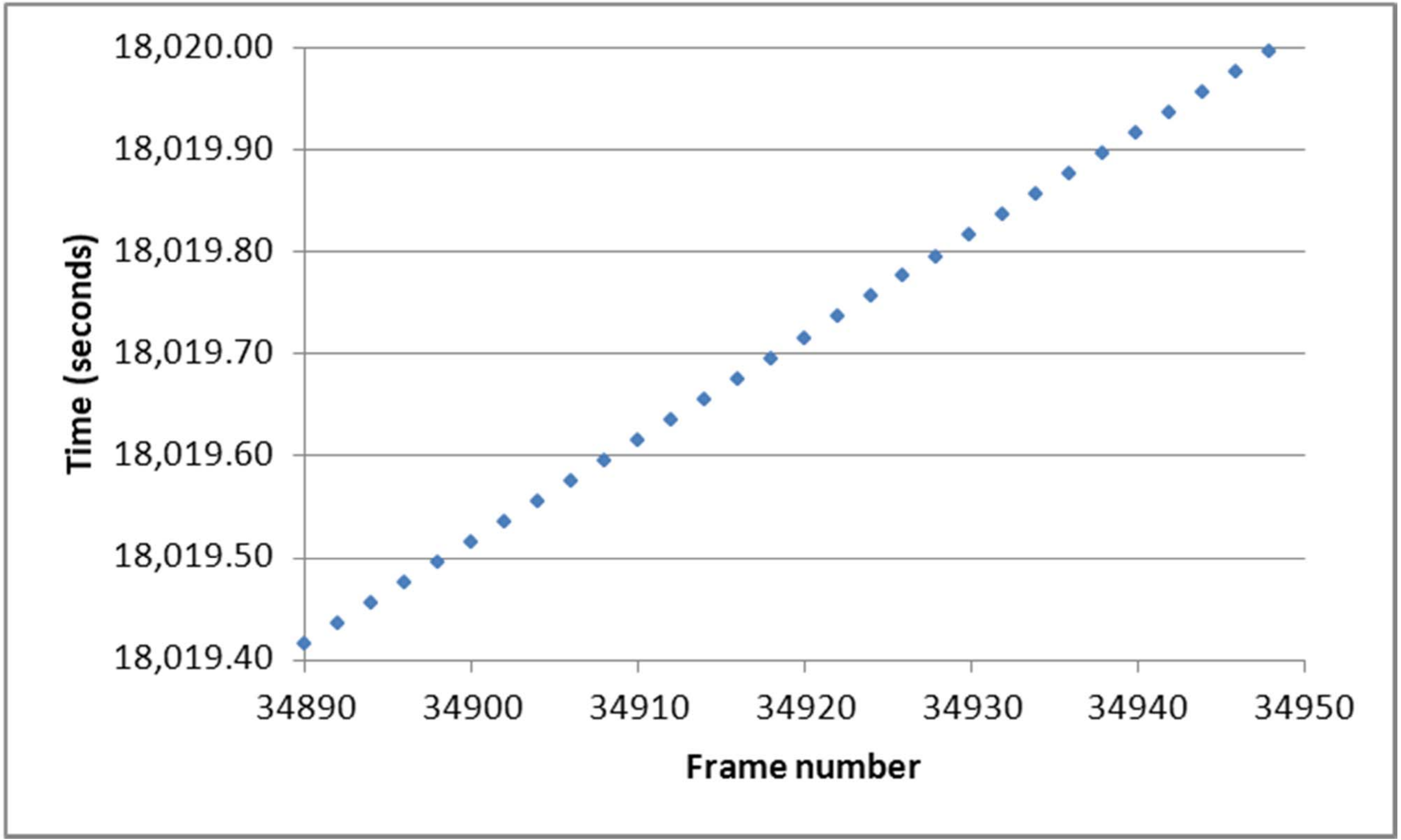
Are grid lines alone sufficient?



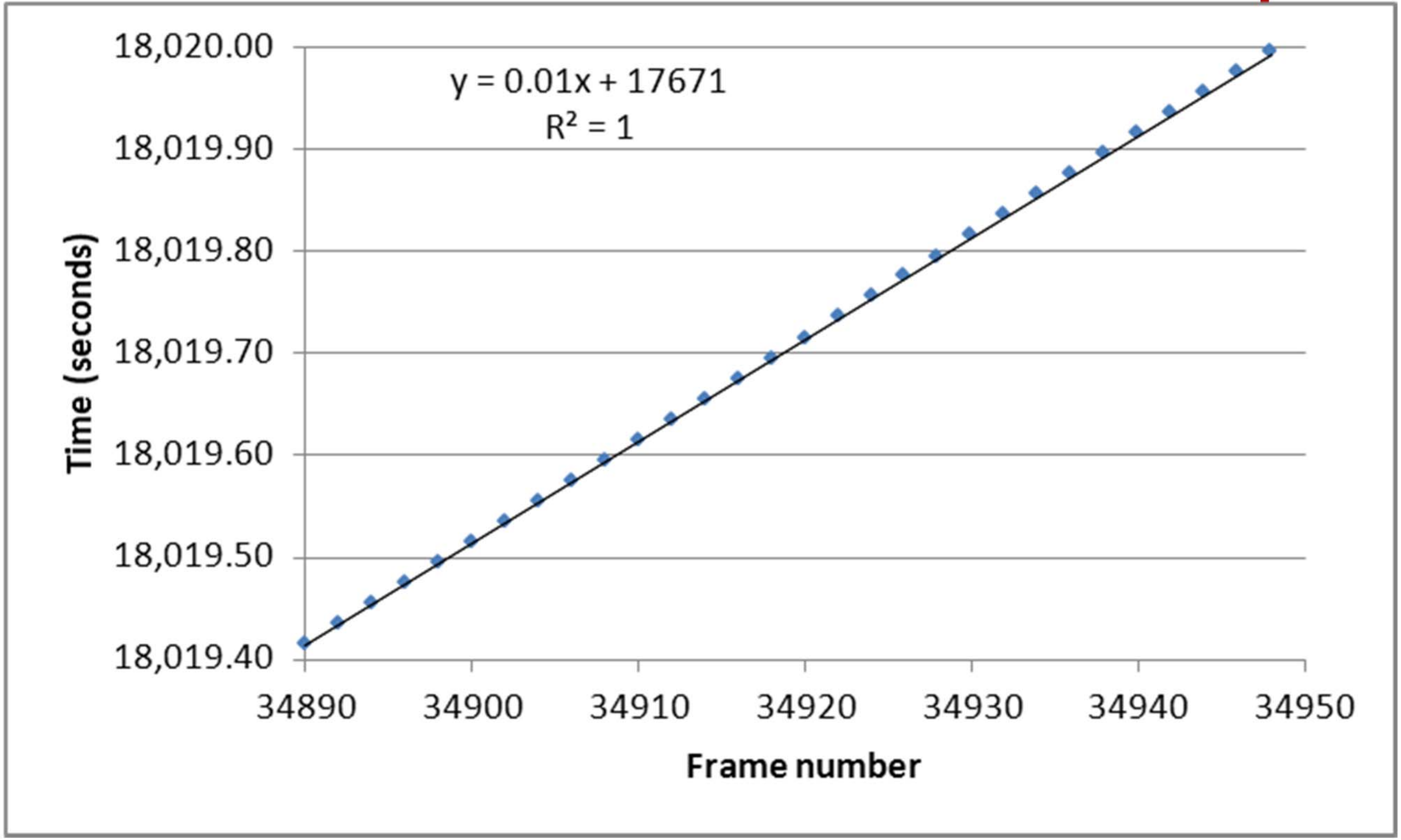
Scatter plots of frame # versus time



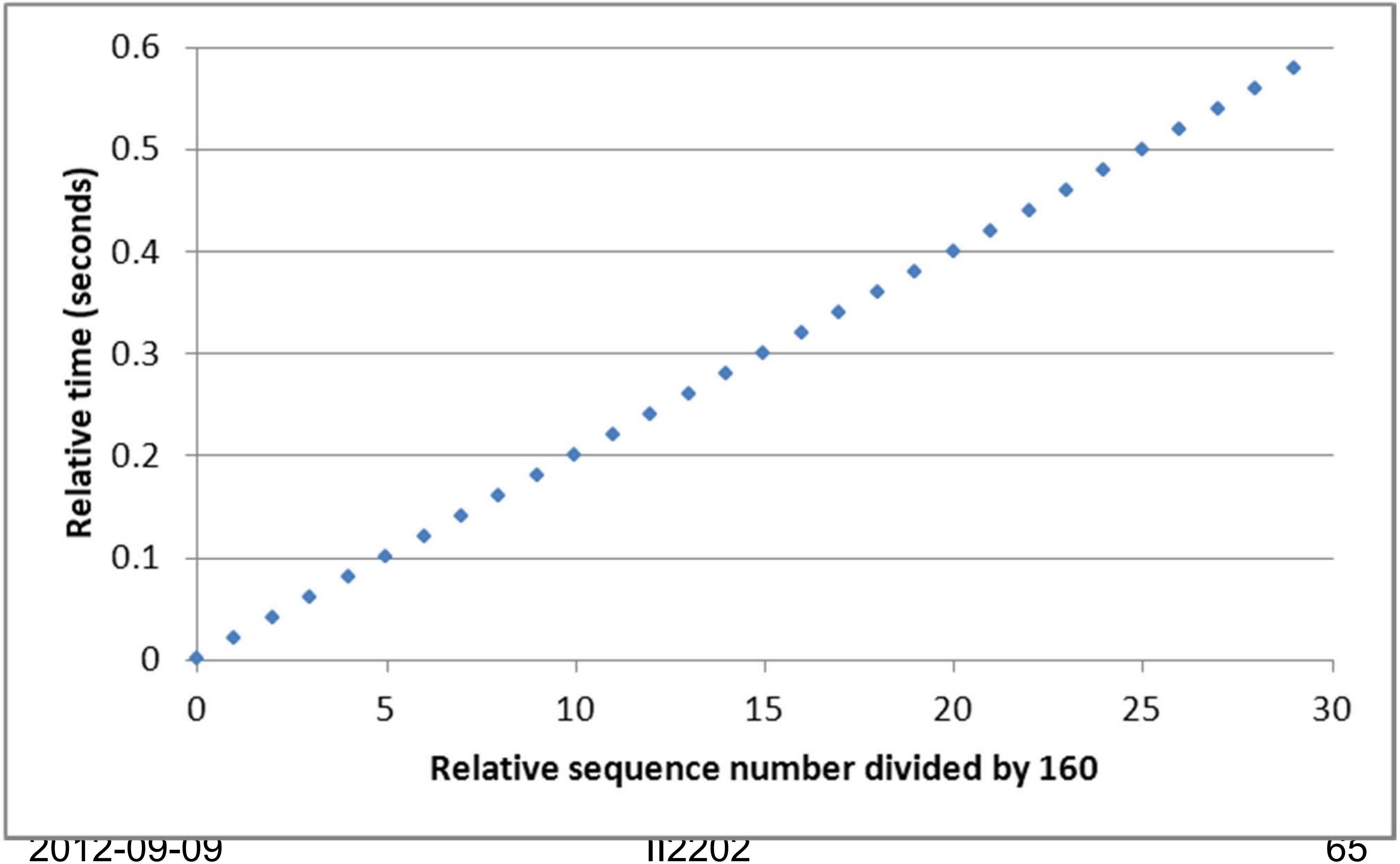
Zoom in on last few samples



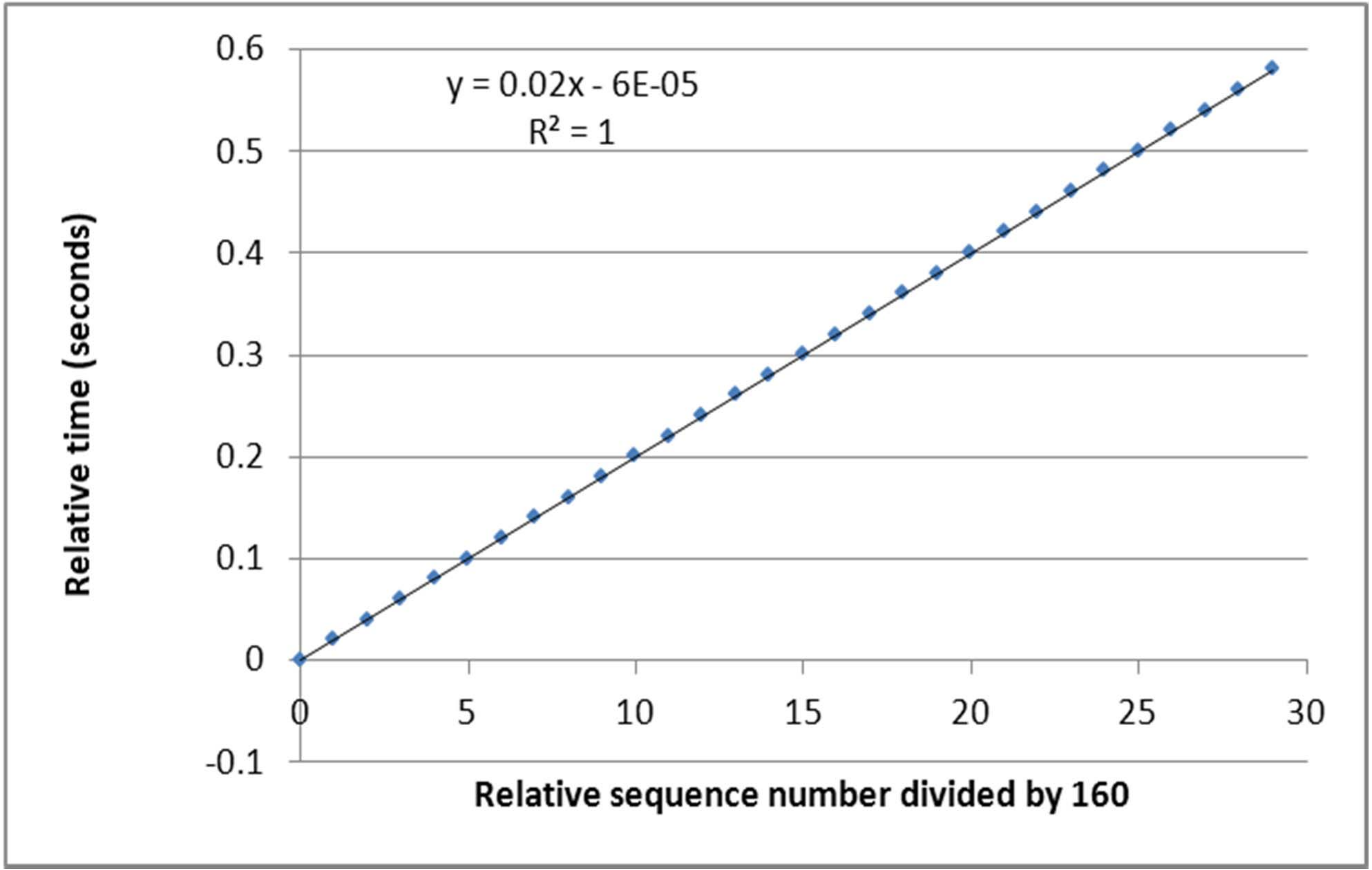
Add a trendline and show eqn.

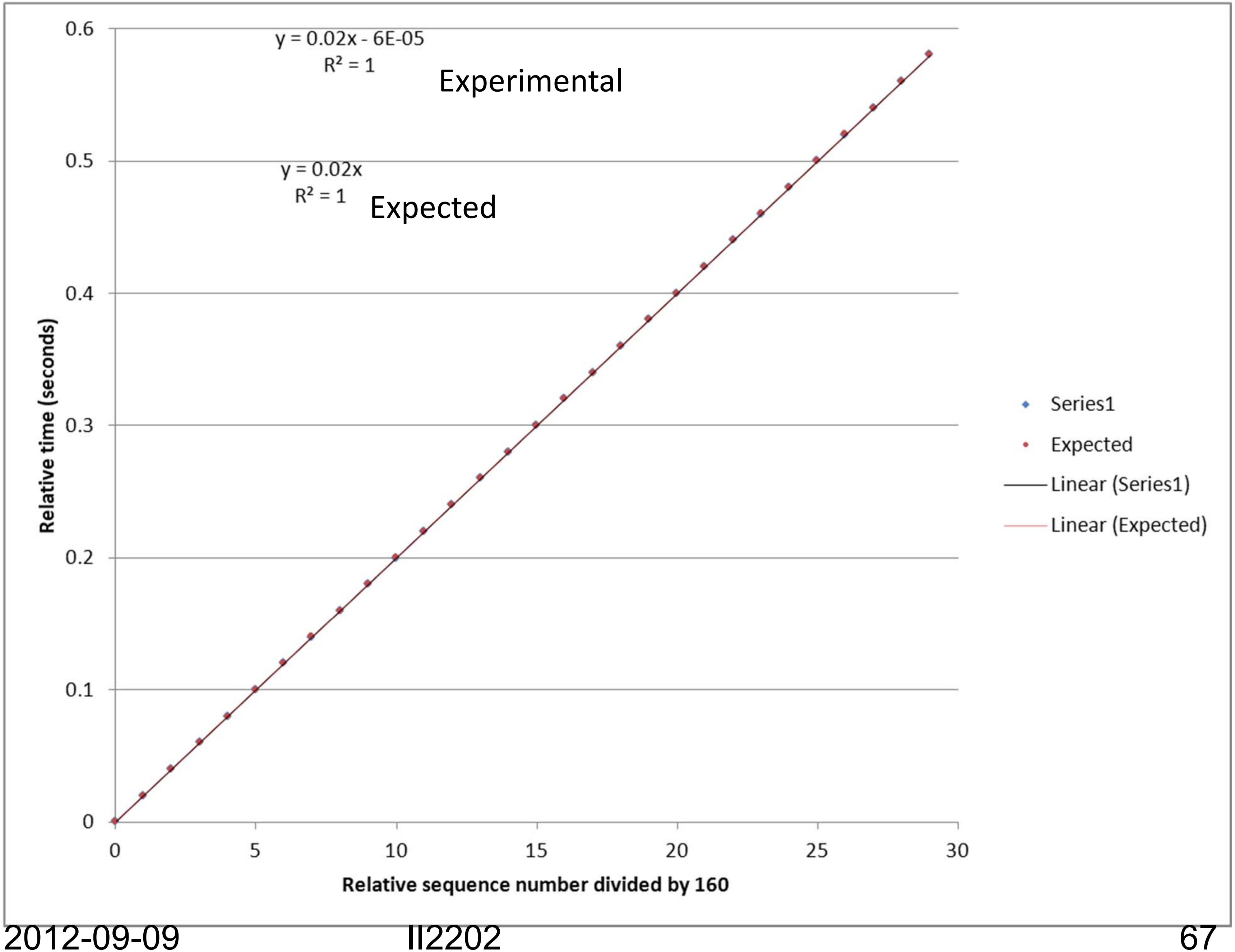


Computing new axis

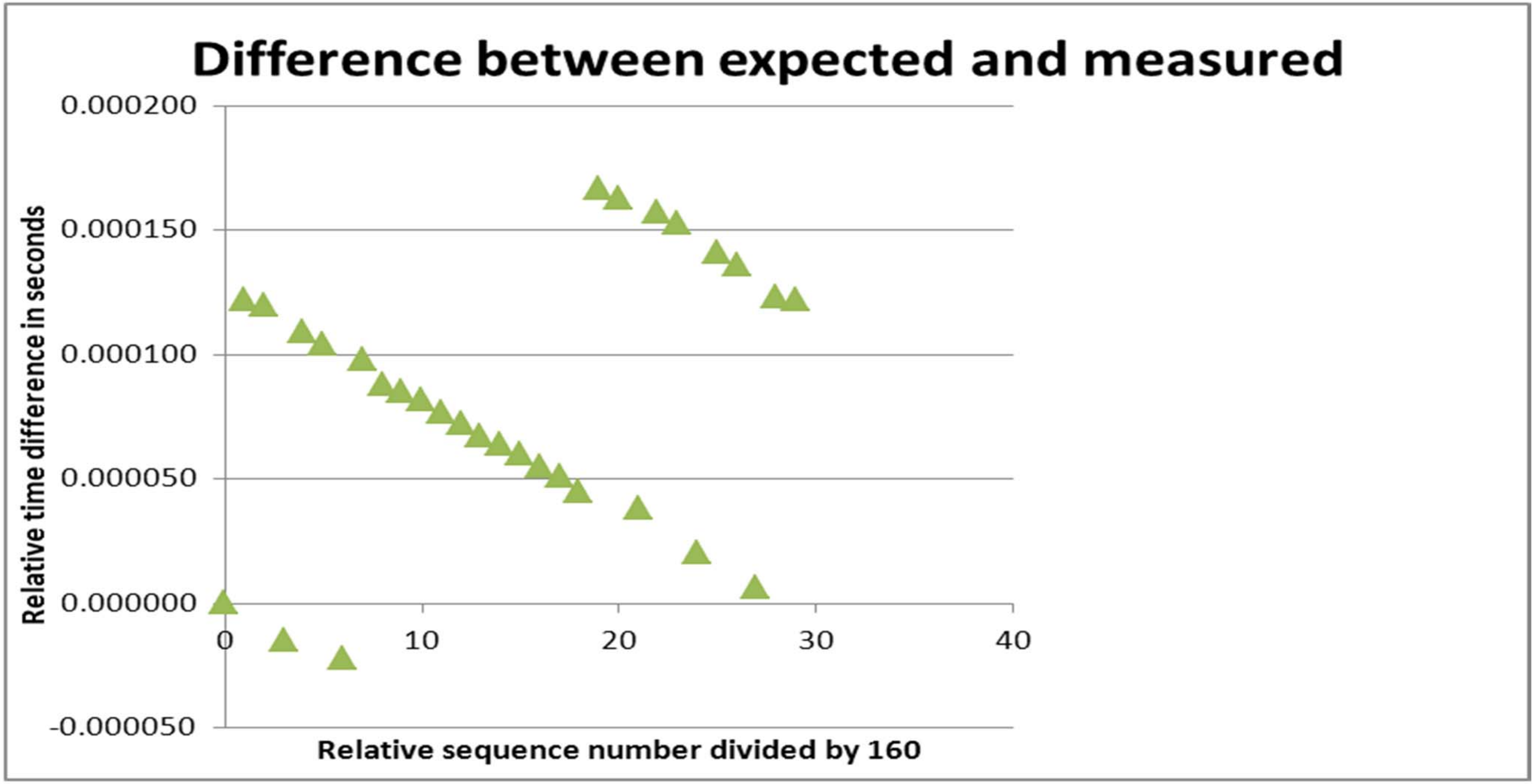


Now add the trendline

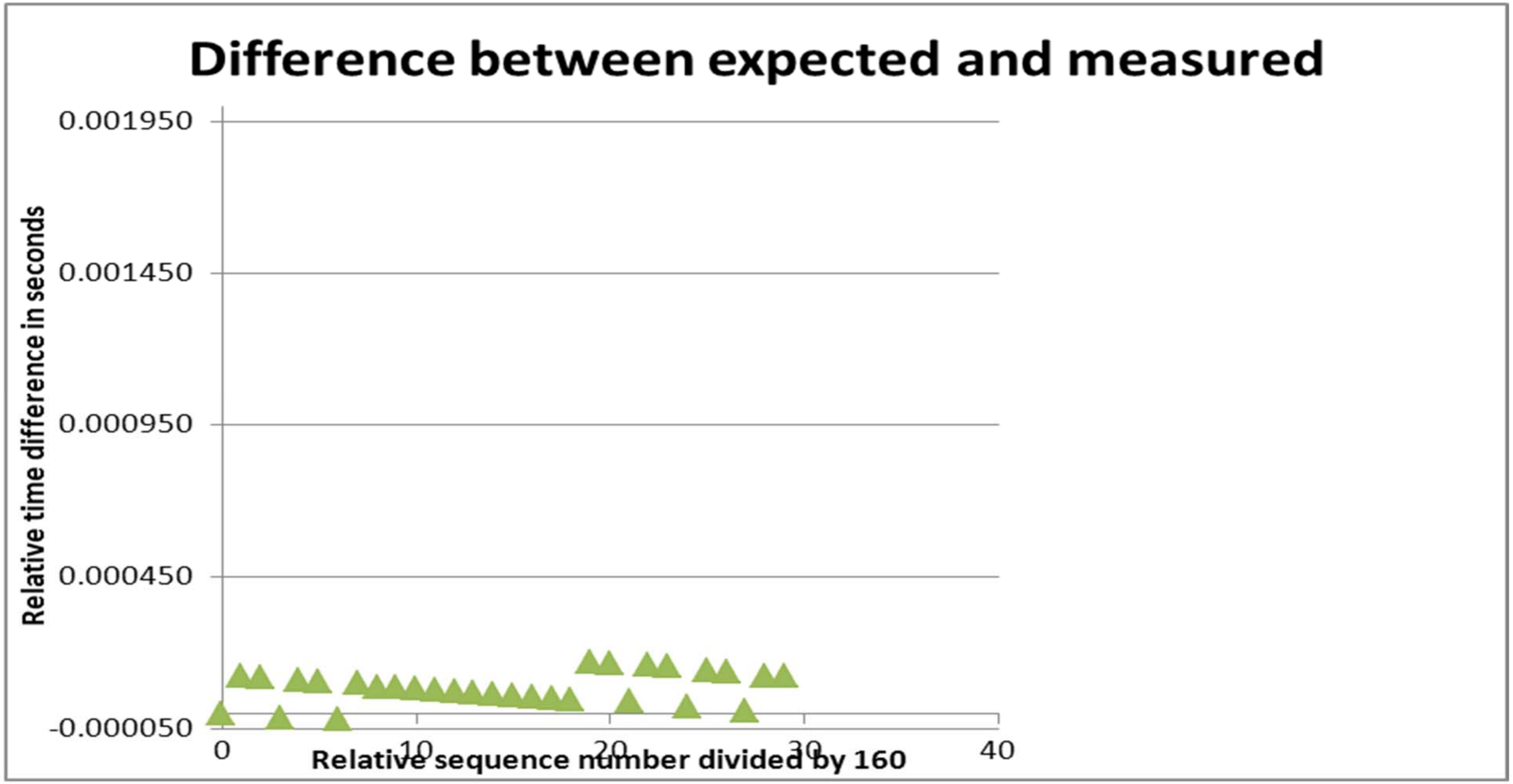



y = 0.02x - 6E-05
R² = 1
Experimental
y = 0.02x
R² = 1
Expected
0.6
0.5
0.4
0.3
0.2
0.1
0
Relative time (seconds)
0
5
10
15
20
25
30
Relative sequence number divided by 160
Series1
Expected
Linear (Series1)
Linear (Expected)

How does the measured data differ from the expected data?



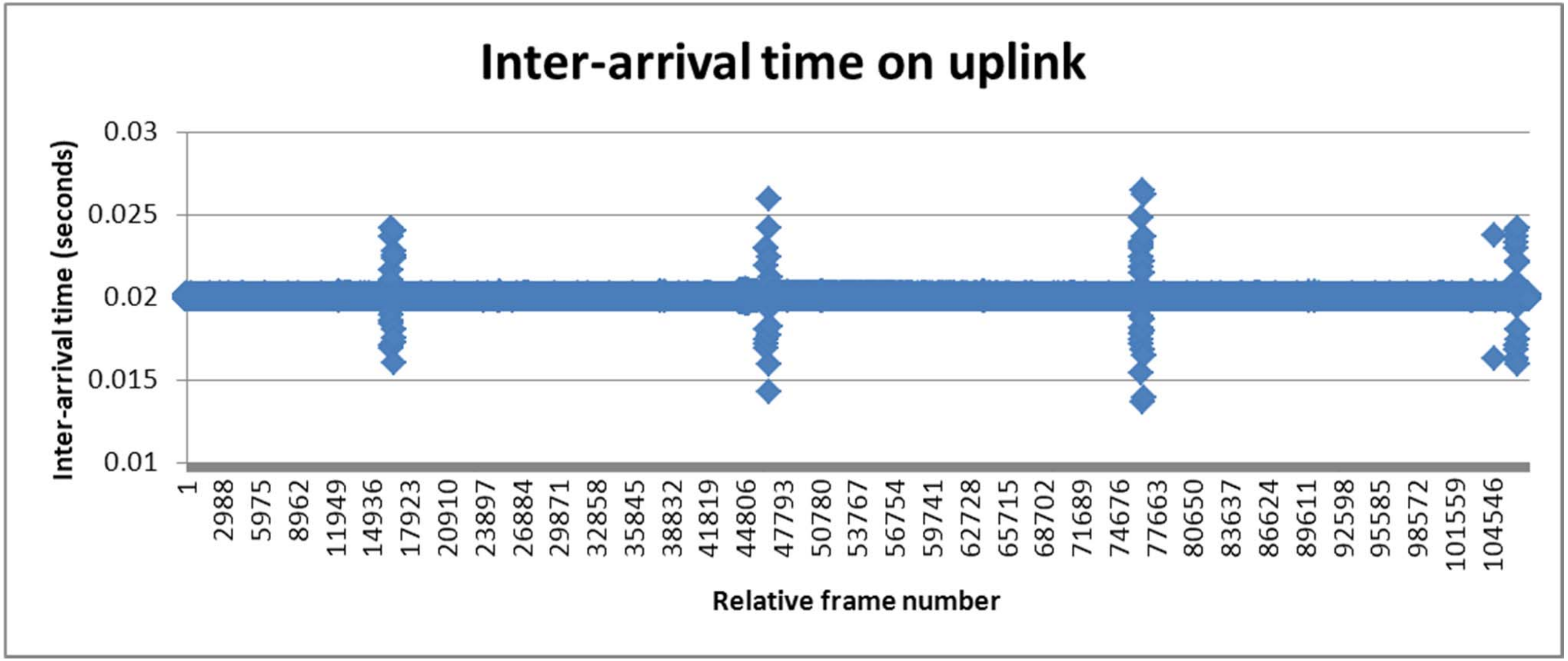
Does the difference matter? Plot scaled to 1/10 of the inter-arrival time period



For traffic in the opposite direction

Mean	0.020000275
Standard Error	3.6743E-07
Median	0.020004
Mode	0.020005
Standard Deviation	0.000120472
Sample Variance	1.45135E-08
Kurtosis	670.0855429
Skewness	0.482218958
Range	0.012759
Minimum	0.013625
Maximum	0.026384
Sum	2150.109545
Count	107504
Confidence Level(95.0%)	7.20157E-07

Uplink inter-arrival times



What is going on?

Note the spikes near:		time in seconds	difference in time in seconds
16453		329.06	
46682		933.64	604.58
76657		1533.14	599.5
106512		2130.24	597.1

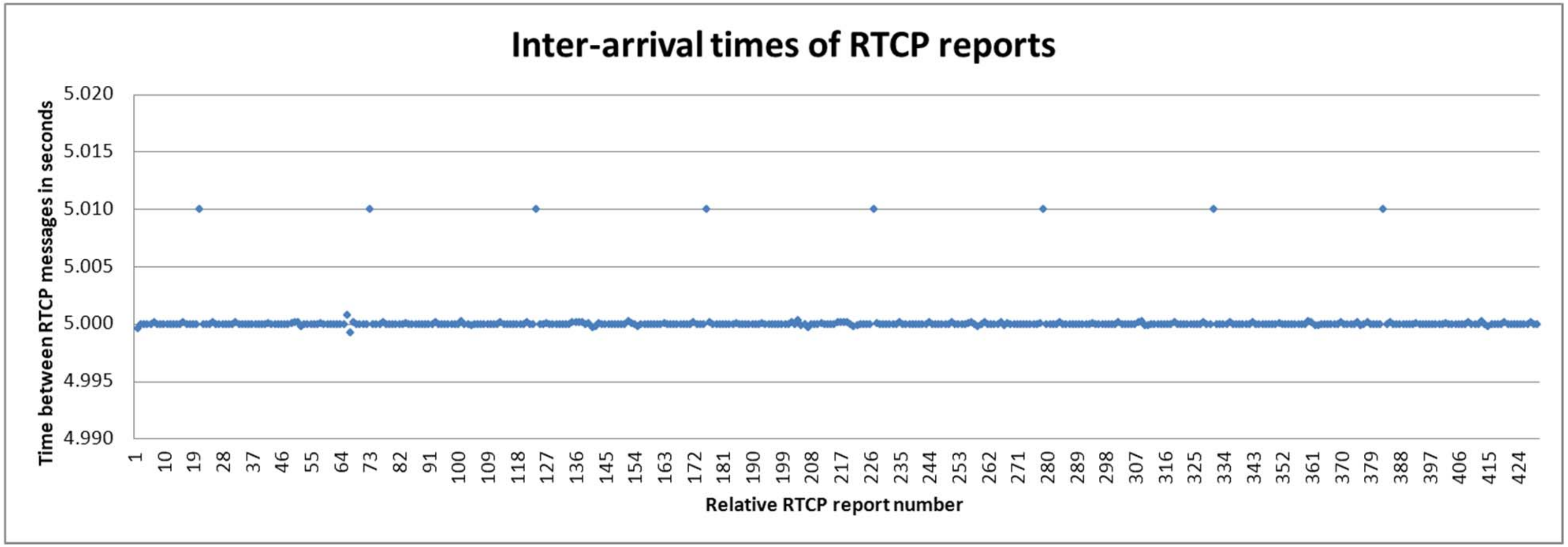
Q: What happens roughly every 600 seconds?

A: DHCP requests

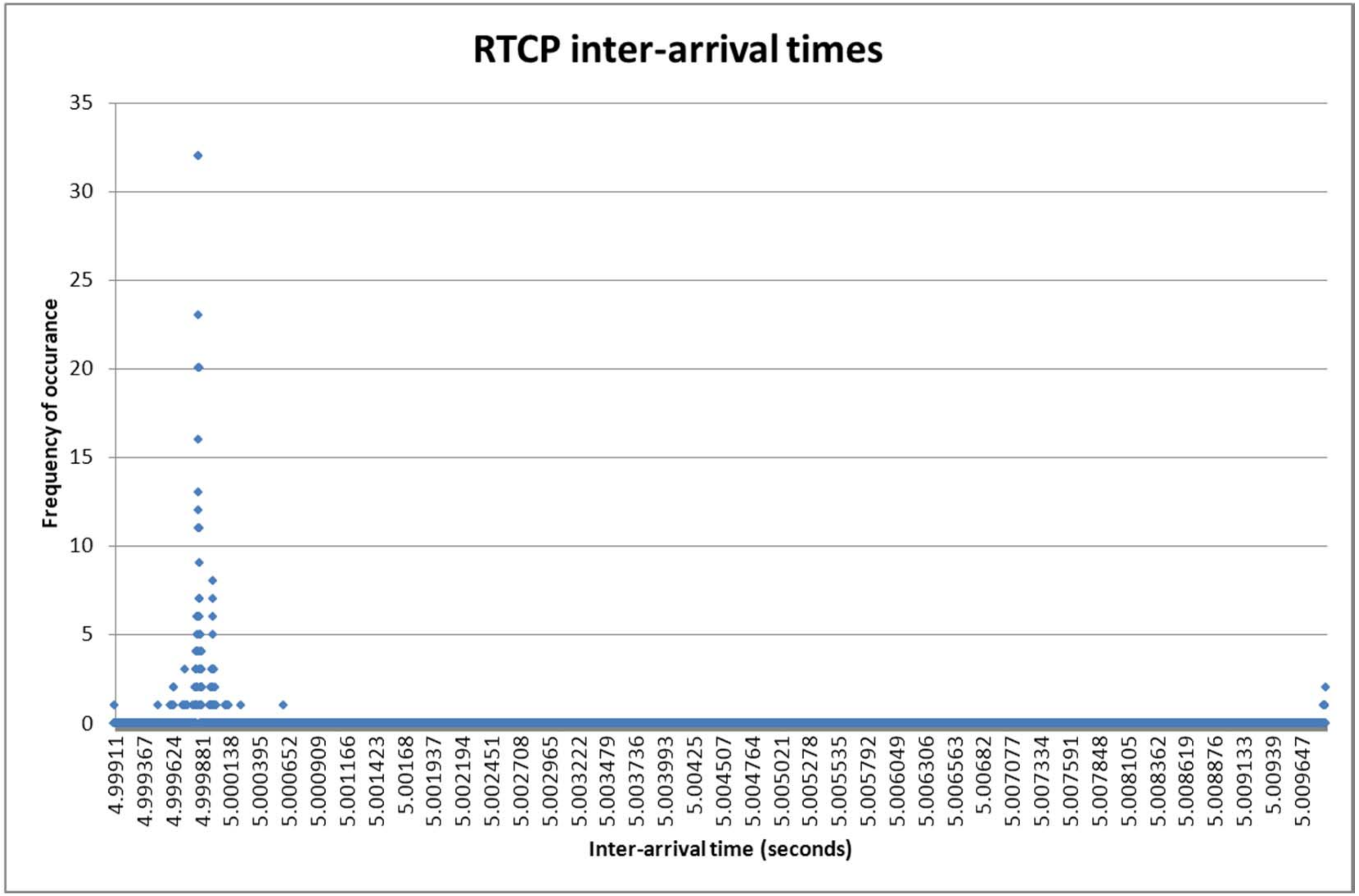
RTCP descriptive statistics

Mean	5.00006104
Standard Error	6.54393E-05
Median	4.999861
Mode	4.99986
Standard Deviation	0.001355399
Sample Variance	1.83711E-06
Kurtosis	48.80806181
Skewness	7.096344028
Range	0.010758
Minimum	4.99911
Maximum	5.009868
Sum	2145.026186
Count	429
Confidence Level(95.0%)	0.000128622

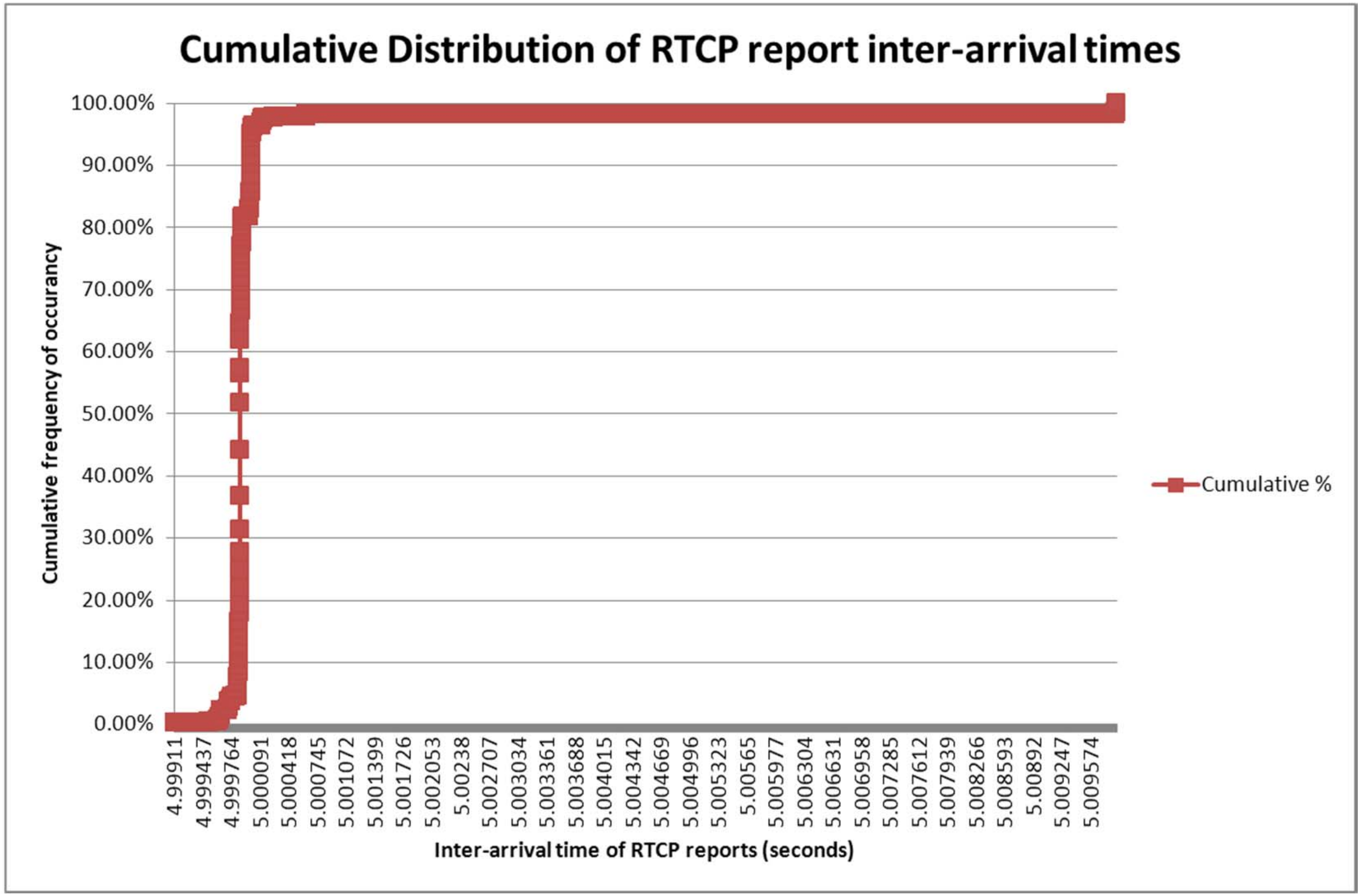
Plot of inter-arrival times of RTCP reports



Histogram of RTCP inter-arrivals



RTCP CDF


Cumulative Distribution of RTCP report inter-arrival times
100.00%
90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
Cumulative frequency of occurancy
4.99911
4.999437
4.999764
5.000091
5.000418
5.000745
5.001072
5.001399
5.001726
5.002053
5.00238
5.002707
5.003034
5.003361
5.003688
5.004015
5.004342
5.004669
5.004996
5.005323
5.00565
5.005977
5.006304
5.006631
5.006958
5.007285
5.007612
5.007939
5.008266
5.008593
5.00892
5.009247
5.009574
Inter-arrival time of RTCP reports (seconds)
Cumulative %

References

[1] Tom Tullis and Bill Albert, “Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics”, Morgan-Kaufmann, 2008, ISBN 978-0-12-373558-4

[2] R Graphics Gallery, http://addictedtor.free.fr/graphiques/

[3] Hadley Wickham, ggplot2: Elegant Graphics for Data Analysis (Use R),Springer; 2nd Printing. August 7, 2009, 216 pages, ISBN-10: 0387981403 and ISBN-13: 978-0387981406, website for the book: http://had.co.nz/ggplot2/book/

[4] Hadley Wickham, website of Hadley Wickham, Rice University, Houston TX, USA, 2010,last accessed Wed 15 Sep 2010 04:51:27 PM CEST, http://had.co.nz/

[5] Dong-Yun Kim, "MAT 356 R Tutorial, Spring 2004", web page, Department of Mathematics, Illinois State University, Normal, IL, USA, last modified: 14 January 2004 07:51:38 AM CET, http://math.illinoisstate.edu/dhkim/rstuff/rtutor.html

[6] Frank McCown, Producing Simple Graphs with R, web page, Computer Science Department, Harding University, Searcy, AR, USA, last modified: 06/08/2008 01:06:21, http://www.harding.edu/fmccown/r/

[7] Michael Wexler, R GUIs, web page, last modified Wed 08 Sep 2010 05:02:06 PM CEST, http://www.nettakeaway.com/tp/?s=R (VP of Web Analytics at Barnes and Noble.com)

[8] Dennis R. Mortensen, Yahoo! Web Analytics 9.5 Launched. Visual.revenue blog,New York City, Tuesday, April 28, 2009, http://visualrevenue.com/blog/2009/04/yahoo-web-analytics-95-launched.html

[9] Julian J. Faraway, “Linear Models with R”, Chapman & Hall/CRC Texts in Statistical Science, 2005, 242 pages, ISBN 0-203-50727-4

Additional R References

http://svn.r-project.org/R/trunk/src/library/stats/R/ecdf.R

General plotting of different distributions in R:

Vito Ricci, Fitting Distributions with R, http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

Yet more references

See the course web page:

http://www.ict.kth.se/courses/II2202/