Part I - Using R To Analyze Data

Download the following file (https://www.dropbox.com/s/zjf0lm830wzmkwz/snmp-delta.csv?dl=0 (https://www.dropbox.com/s/zjf0lm830wzmkwz/snmp-delta.csv?dl=0)), which contains the following as table as comma-separated values.

here are a total of 8 columns of data. You should be able to read the data into R using the provided commands below.

These are measurements in packets per interval and bytes per interval for specific interfaces on the router from the prior assignment, ideally structured as a notebook. For this assignment, you're going to hand in an "R Program" that shows how you analyzed the data.

You should hand in a single PDF document with your answers to the questions, including any graphics needed. R can produce plots of graphs (described below) -- if you don't or can't use the notebook form (which will include the plots directly), you should be able to suck the plots into OpenOffice or other word processing tools.

In [1]:
```
snmpData = read.csv('snmp-delta.csv')
snmpData
```

Out[1]:

	sample	if.	outUCastPkts	inOctets	outOctets	inUcastPkts	outNUCastPkt
1	0	1	0	0	0	0	0
2	0	2	672	184683	90820	2226	0
3	0	3	96	116273	14370	866	0
4	0	6	119	97039	16290	751	0
5	0	7	678	149297	92040	2201	0
6	0	9	0	0	0	0	0
7	0	10	1621	0	142038	0	0
8	0	11	1621	0	142038	0	0
9	1	1	0	0	0	0	0
10	1	2	376	143765	52072	1619	0
11	1	3	105	138865	19345	1023	0
12	1	6	122	116917	21265	906	0
13	1	7	350	107185	49532	1486	0
14	1	9	0	0	0	0	0
15	1	10	1209	0	109328	0	0
16	1	11	1209	0	109328	0	0
17	2	1	0	0	0	0	0
18	2	2	353	148319	48741	1805	0
19	2	3	111	138016	16479	1019	0
20	2	6	131	116547	18399	903	0
21	2	7	343	116354	48081	1743	0
22	2	9	0	0	0	0	0
23	2	10	1468	0	130136	0	0
24	2	11	1468	0	130136	0	0
25	3	1	0	0	0	0	0
26	3	2	391	151097	49396	1673	0
27	3	3	107	116672	22340	871	0
28	3	6	127	97440	24260	757	0
29	3	7	381	116048	48736	1575	0
30	3	9	0	0	0	0	0

31	3	10	1266	0	114390	0	0
32	3	11	1266	0	114390	0	0
33	4	1	0	0	0	0	0
34	4	2	425	141174	56971	1506	0
35	4	3	132	107612	26716	767	0
36	4	6	152	89764	28636	652	0
37	4	7	415	112020	56311	1433	0
38	4	9	0	0	0	0	0
39	4	10	1095	0	100278	0	0
40	4	11	1097	0	100446	0	0
41	5	1	0	0	0	0	0
42	5	2	298	96632	38035	1011	0
43	5	3	72	93532	10593	664	0
44	5	6	88	78364	12129	568	0
45	5	7	290	72746	37507	931	0
46	5	9	0	0	0	0	0
47	5	10	687	0	68431	0	0
48	5	11	685	0	68263	0	0
49	6	1	0	0	0	0	0
50	6	2	443	132585	56252	1243	0
51	6	3	113	98585	26859	719	0
52	6	6	133	81409	28779	604	0
53	6	7	433	103649	55592	1146	0
54	6	9	0	0	0	0	0
55	6	10	787	0	77028	0	0
56	6	11	787	0	77028	0	0
57	7	1	0	0	0	0	0
58	7	2	1167	175776	119737	1907	0
59	7	3	92	97305	13360	705	0
60	7	6	112	79809	15280	588	0
61	7	7	1157	142476	119077	1846	0

62	7	9	0	0	0	0	0
63	7	10	769	0	75590	0	0
64	7	11	769	0	75590	0	0
65	8	1	0	0	0	0	0
66	8	2	352	118004	46771	1197	0
67	8	3	123	105176	21262	748	0
68	8	6	143	87501	23182	632	0
69	8	7	342	89619	46111	1099	0
70	8	9	0	0	0	0	0
71	8	10	794	0	76886	0	0
72	8	11	794	0	76886	0	0
73	9	1	0	0	0	0	0
74	9	2	409	189142	49274	1448	0
75	9	3	170	100705	82537	761	0
76	9	6	190	82941	84457	646	0
77	9	7	399	162175	48614	1386	0
78	9	9	0	0	0	0	0
79	9	10	1043	0	96241	0	0
80	9	11	1043	0	96241	0	0
81	10	1	0	0	0	0	0
82	10	2	375	136024	51156	1409	0
83	10	3	115	101904	24067	719	0
84	10	6	131	85966	25603	623	0
85	10	7	367	105098	50628	1317	0
86	10	9	0	0	0	0	0
87	10	10	1015	0	94793	0	0
88	10	11	1015	0	94793	0	0
89	11	1	0	0	0	0	0
90	11	2	353	164905	45573	2007	0
91	11	3	93	97730	17421	701	0
92	11	6	113	80806	19341	586	0

93	11	7	343	125181	44913	1908	0
94	11	9	0	0	0	0	0
95	11	10	1632	0	142141	0	0
96	11	11	1632	0	142141	0	0
97	12	1	0	0	0	0	0
98	12	2	302	138347	37912	1738	0
99	12	3	96	95037	13133	695	0
100	12	6	116	78197	15053	580	0
101	12	7	292	107413	37252	1677	0
102	12	9	0	0	0	0	0
103	12	10	1450	0	128684	0	0
104	12	11	1450	0	128684	0	0
105	13	1	0	0	0	0	0
106	13	2	1011	200268	108698	2054	0
107	13	3	121	103523	29601	768	0
108	13	6	141	85099	31521	650	0
109	13	7	1001	159978	108038	1957	0
110	13	9	0	0	0	0	0
111	13	10	1018	0	94315	0	0
112	13	11	1018	0	94315	0	0
113	14	1	0	0	0	0	0
114	14	2	416	118987	56668	1118	0
115	14	3	142	109140	28796	784	0
116	14	6	162	90961	30716	668	0
117	14	7	406	96640	56008	1056	0
118	14	9	0	0	0	0	0
119	14	10	736	0	73171	0	0
120	14	11	736	0	73171	0	0
121	15	1	0	0	0	0	0
122	15	2	308	105711	41327	1103	0
123	15	3	91	107006	13166	779	0

124	15	6	107	90228	14702	683	0
125	15	7	300	78976	40799	1010	0
126	15	9	0	0	0	0	0
127	15	10	757	0	74865	0	0
128	15	11	771	0	75033	0	0
129	16	1	0	0	0	0	0
130	16	2	365	109018	47345	1114	0
131	16	3	91	115535	12932	854	0
132	16	6	111	96469	14852	739	0
133	16	7	355	81888	46685	1017	0
134	16	9	0	0	0	0	0
135	16	10	721	0	70871	0	0
136	16	11	707	0	70703	0	0
137	17	1	0	0	0	0	0
138	17	2	1203	198918	128047	2113	0
139	17	3	111	110780	24193	789	0
140	17	6	131	92624	26113	674	0
141	17	7	1193	161505	127387	2036	0
142	17	9	0	0	0	0	0
143	17	10	914	0	86079	0	0
144	17	11	914	0	86079	0	0
145	18	1	0	0	0	0	0
146	18	2	355	123138	45032	1189	0
147	18	3	103	97773	25374	711	0
148	18	6	123	80827	27294	598	0
149	18	7	345	94958	44372	1092	0
150	18	9	0	0	0	0	0
151	18	10	811	0	78500	0	0
152	18	11	811	0	78500	0	0

Problem #1

Data Manipulation: Show how to compute a table that contain the sum of all packets (input and outputt, unicast and non-unicast) for each interface and the sum of bytes (input and outputt, unicast and non-unicast) for each interface for each output link. The table should have a total of 3 columns (interface, packets, bytes) and a row for each interface.

Refer to "slices" on page 5 and 6 of the SimpleR manual and the **data.frame** function to construct a data frame from vectors. In my solution, I defined functions that sliced the data for a specific interface and then used **sapply** to compute the data for all interfaces.

In [2]:
```
getPackets <- function(iff){
    (inter <- snmpData[snmpData$if. == iff, ])
    invisible(inter$packets <- data.frame(inter$outUCastPkts,int
er$inUcastPkts,inter$outNUCastPkts,inter$inNUcastPkts))
    invisible(packets_rows <- data.frame(inter$if., rowSums(inte
r$packets)))
    (packets_total <- sum(packets_rows[,-1]))
    return (packets_total)
}

getBytes <- function(iff){
    invisible(inter <- snmpData[snmpData$if. == iff, ])
    invisible(inter$bytes <- data.frame(inter$inOctets,inter$out
Octets))
    invisible(bytes_rows <- data.frame(inter$if., rowSums(inter
$bytes)))
    (bytes_total <- sum(bytes_rows[,-1]))
    return (bytes_total)
}

ifs <- c(1,2,3,6,7,9,10,11)
packets <- sapply(ifs, getPackets)
bytes <- sapply(ifs, getBytes)

data.frame(interface=ifs, packets=packets, bytes=bytes)
```

Out[2]:

	interface	packets	bytes
1	1	0	0
2	2	39054	3946320
3	3	17027	2493713
4	6	15260	2186780
5	7	37306	3340889
6	9	0	0
7	10	19793	1833765
8	11	19793	1833765

Problem #2

Summary Statistics: Using the data for the inUcastPkts packets on interface #2, show how you would use R to compute the mean, variance, standard deviation and COV "by hand". For example, if you had read the data into data frame x, you could compute the sum of the first column using a **sum** over the slice of the inUcastPkts for interface 2, and then use this to compute the mean. You should compute these terms using e.g. sum(), length() and other functions following the computations in the textbook. Your solution should show both the formulas and their output.

It will be easiest to first extract the data you are working on to a vector that you use in the remainder of your calculations.

In [3]:
```
inter <- snmpData[snmpData$if. == 2, ]
ucast <- inter$inUcastPkts

#mean = sum/length
m = sum(ucast)/length(ucast)
sprintf('Mean = %f', m)

#variance = sum((X_i-mean)^2)/(length-1)
var = sum((ucast-m)^2)/(length(ucast)-1)
sprintf("Variance = %f", var)

#std = sqrt(variance)
std = (var)^(1/2)
sprintf("Standard Deviation = %f", std)

#cov = std/mean
cov = std/m
sprintf("COV = %f", cov)
```

Out[3]: 'Mean = 1551.578947'

Out[3]: 'Variance = 151828.923977'

Out[3]: 'Standard Deviation = 389.652312'

Out[3]: 'COV = 0.251133'

Problem #3

R Bulit-in: Use the built-in R commands (see the SimpleR manual) to compute the mean, variance, sd & COV for the other columns (I don't think there is a function for COV in R, so you will need to compute that by hand, or better yet, define an R function).

In [4]:
```
COV <- function(vect){
    ccv = sd(vect)/mean(vect)
    return(ccv)
}
#mean(ucast)
#var(ucast)
#sd(ucast)
#COV(ucast)

stats <- function(vect){
    mm <- mean(vect)
    vv <- var(vect)
    ss <- sd(vect)
    cc <- COV(vect)
    return(list(mm,vv,ss,cc))
}

#stats(ucast)

all_cols = function(col_name){
    asdf <- eval(parse(text = paste('inter$',col_name, sep="")))
    return(stats(asdf))
}

#inter$outUCastPkts,inter$inUcastPkts,inter$outNUCastPkts,inter
$inNUcastPkts, inter$inOctets,inter$outOctets
cols = c('outUCastPkts','inUcastPkts','outNUCastPkts','inNUcastP
kts', 'inOctets', 'outOctets')

#(eval(parse(text = 'inter$outUCastPkts')))

inter_2 <- snmpData[snmpData$if. == 2, ]

sdf = sapply(cols, all_cols)
rownames(sdf) <-c('Mean', 'Variance', 'Stdv.', 'COV')

(sdf)
```

Out[4]:

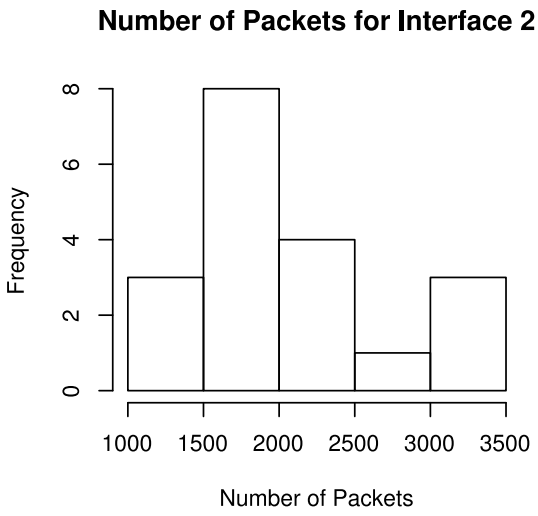
	outUCastPkts	inUcastPkts	outNUCastPkts	inNUcastPkts	inOctets
Mean	503.8947	1551.579	0	0	146131.2
Variance	84325.1	151828.9	0	0	101158520
Stdv.	290.3878	389.6523	0	0	31805.43
COV	0.5762867	0.2511328	NaN	NaN	0.2176498

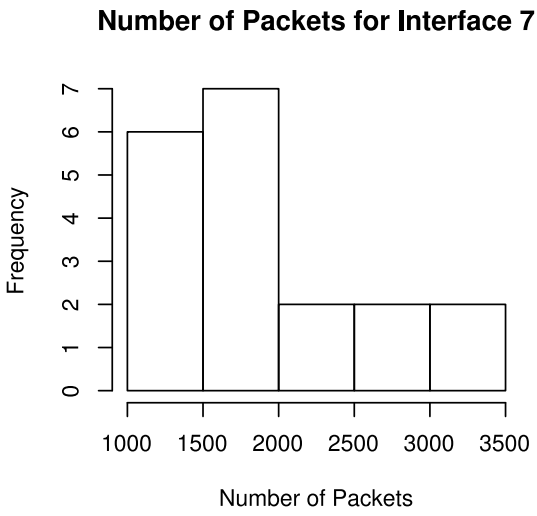
Problem #4

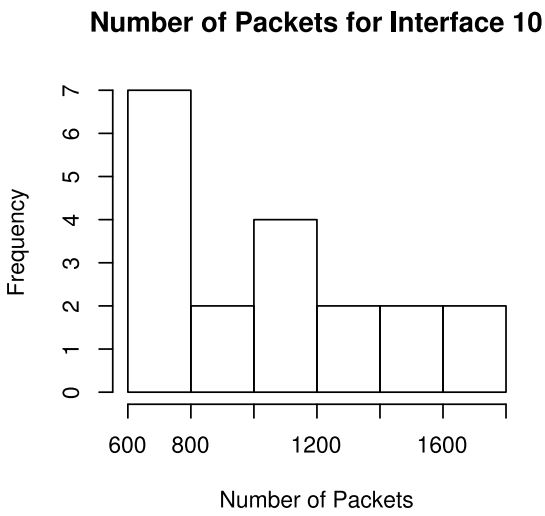
Plot histograms for the number of packets (input and output of all types) per interval on interfaces 2, 7, 10 and 11. You can arrange plots into a grid using the following commands:

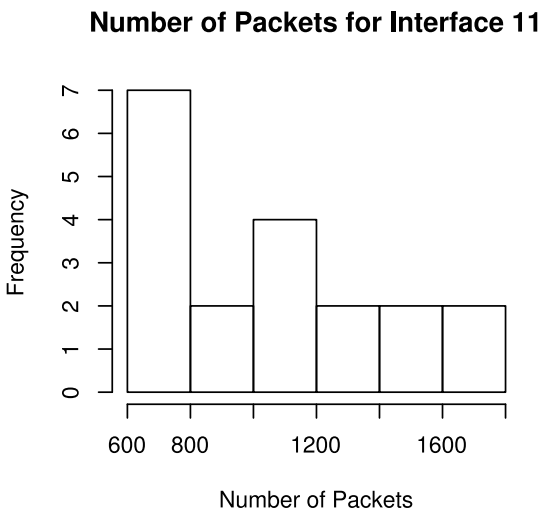
```
par(mfrow=c(2,2))
plot(....)
etc
plot(...)
```

In [9]:
```
inter_2 <- snmpData[snmpData$if. == 2, ]
inter_7 <- snmpData[snmpData$if. == 7, ]
inter_10 <- snmpData[snmpData$if. == 10, ]
inter_11 <- snmpData[snmpData$if. == 11, ]

all_cols2 = function(col_name, interf){
    asdf <- eval(parse(text = paste('inter',"_",interf,"$",col_n
ame, sep="")))
    #print(paste('inter',"_",interf,"$",col_name, sep=""))
    #print(eval(parse(text = paste('inter',"_",interf,"$",col_na
me, sep=""))))
    return (asdf)
}

#inter$outUCastPkts,inter$inUcastPkts,inter$outNUCastPkts,inter
$inNUcastPkts, inter$inOctets,inter$outOctets
cols = c('outUCastPkts','inUcastPkts','outNUCastPkts','inNUcastP
kts')

par(mfrow=c(2,2))

Result <- list()


AddItemNaive2 <- function(item)
{
    .GlobalEnv$Result <- c(.GlobalEnv$Result, item)
}

for (i in c('2','7','10','11')){
    result <- sapply(cols, all_cols2, interf=i)
    #print(result)
    summ = data.frame(rowSums(result))
    AddItemNaive2(summ)

    hist(rowSums(result),main=paste('Number of Packets for Inter
face ', i, sep=""),xlab="Number of Packets")
    #print(summ)
}

#print(Result[1])
```









Problem #5

Is the behavior across any of interfaces 2, 7, 10 and 11 correlated? In other words, are the number of packets sent on one link similar to the number of packets sent on another link? A similar property might be seen for the number of bytes. Use the "correlation" function (corr) across interfaces 2, 7, 10 and 11 for packets (as above) to argue what links have the most correlated behavior.

In [10]:
```
cor(as.numeric(unlist(Result[1])), as.numeric(unlist(Result
[4])))
cor(as.numeric(unlist(Result[2])), as.numeric(unlist(Result
[4])))
cor(as.numeric(unlist(Result[3])), as.numeric(unlist(Result
[4])))
cor(as.numeric(unlist(Result[1])), as.numeric(unlist(Result
[3])))
cor(as.numeric(unlist(Result[2])), as.numeric(unlist(Result
[3])))
cor(as.numeric(unlist(Result[1])), as.numeric(unlist(Result
[2])))

print('According to the cor function, interfaces 10 and 11 are t
he highest correlated because the correlation is closest to 1.')
print('This is closely followed by interfaces 2 and 7')
```

Out[10]: 0.390500964864765

Out[10]: 0.396253619377111

Out[10]: 0.999888610646897

Out[10]: 0.390749159036285

Out[10]: 0.396481388012108

Out[10]: 0.99910454763274

```
[1] "According to the cor function, interfaces 10 and 11 are the
highest correlated because the correlation is closest to 1."
[1] "This is closely followed by interfaces 2 and 7"
```

Problem #6

Using the equations from Baudec (section 2.2.3), compute the confidence interval for the mean for the number of inUcastPkts packets per interval on interface #2. You can use the built in commands in R to compute the mean and variance (or sd). Compute the 90 and 95% confidence intervals "by hand" (i.e. follow the equations in the text and don't use the built-in tests).

In [11]:
```
inter <- snmpData[snmpData$if. == 2, ]
ucast <- inter$inUcastPkts



conf_interval = function(vect, conf){
    mm = mean(vect)
    vv = var(vect)
    ss = sd(vect)
    len = length(vect)

    zz = (1+conf)/2
    qq = qnorm(zz)

    interval = qq*ss/(len^(1/2))

    return(interval)
}

ci90 <- conf_interval(ucast, 0.9)
sprintf("90 confidence interval => %f to %f", (mean(ucast)-ci9
0), (mean(ucast)+ci90))


ci95 <- conf_interval(ucast, 0.95)
sprintf("95 confidence interval => %f to %f", (mean(ucast)-ci9
5), (mean(ucast)+ci95))
```

Out[11]: '90 confidence interval => 1404.541582 to 1698.616313'

Out[11]: '95 confidence interval => 1376.373120 to 1726.784775'

Problem #7

Using paired confidence intervals (you can use the t.test function -- see SimpleR, page 69) to argue whether links 3 and 6 have a statistically identical number of total packets (input + output, all types) per sampling period at the 90, 95 and 99% confidence level. Show the calculations and justify your answer.

In [26]:
```
inter_3 <- snmpData[snmpData$if. == 3, ]
inter_6 <- snmpData[snmpData$if. == 6, ]

cols = c('outUCastPkts','inUcastPkts','outNUCastPkts','inNUcastP
kts')

all_cols2 = function(col_name, interf){
    asdf <- eval(parse(text = paste('inter',"_",interf,"$",col_n
ame, sep="")))
    #print(paste('inter',"_",interf,"$",col_name, sep=""))
    #print(eval(parse(text = paste('inter',"_",interf,"$",col_na
me, sep=""))))
    return (asdf)
}

# t.test(x, y = NULL, alternative = c("two.sided", "less", "grea
ter"), mu = 0, paired =
# FALSE, var.equal = FALSE, conf.level = 0.95)

for (i in c(.90,.95,.99)){

    result3 <- sapply(cols, all_cols2, interf='3')
    result6 <- sapply(cols, all_cols2, interf='6')
    #print(result)
    summ3 = data.frame(rowSums(result3))
    summ6 = data.frame(rowSums(result6))
    #print(summ6)
    #print(colSums(summ))
    print(t.test(summ3$rowSums.result3.,summ6$rowSums.result6.,p
aired=TRUE,conf.level=i))
}

print('They are not statistically identical because the t-test c
onfidence interval is not zero, or even near zero.')
print('In order for links 3 and 6 to be statistically identical,
the t-test confidence interval and')
print('difference between the means should be near zero.')
```

```
        Paired t-test

data:  summ3$rowSums.result3. and summ6$rowSums.result6.
t = 67.252, df = 18, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to
0
90 percent confidence interval:
 90.60205 95.39795
sample estimates:
mean of the differences
                     93


        Paired t-test

data:  summ3$rowSums.result3. and summ6$rowSums.result6.
t = 67.252, df = 18, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to
0
95 percent confidence interval:
 90.09473 95.90527
sample estimates:
mean of the differences
                     93


        Paired t-test

data:  summ3$rowSums.result3. and summ6$rowSums.result6.
t = 67.252, df = 18, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to
0
99 percent confidence interval:
 89.01954 96.98046
sample estimates:
mean of the differences
                     93

[1] "They are not statistically identical because the t-test con
fidence interval is not zero, or even near zero."
[1] "In order for links 3 and 6 to be statistically identical, t
he t-test confidence interval and"
[1] "difference between the means should be near zero."
```

Problem #8

Use the R "bootstrap" method (see Boudec Algorithm 2.1 and http://www.statmethods.net/advstats/bootstrapping.html (http://www.statmethods.net/advstats/bootstrapping.html)) to compute the confidence interval for the MEDIAN number of total packets (input + output, all types) per sampling period for link 3 at the 90 and 95 level.

In [50]:
```
library(boot)

rsq <- function(data, indices) {
  d <- data[indices] # allows boot to select sample
  return(median(d))
}

#R=ceiling(50/(1-conf)-1)
#print(summ3$rowSums.result3.)

# get 90% confidence interval
gamma=ceiling(50/(1-.90)-1)
bobj <- boot(summ3$rowSums.result3.,rsq,R=gamma)
boot.ci(bobj, type="perc", conf=.9)


# get 95% confidence interval
gamma=ceiling(50/(1-.95)-1)
bobj <- boot(summ3$rowSums.result3.,rsq,R=gamma)
boot.ci(bobj, type="perc")
```

```
 [1]  962 1128 1130  978  899  736  832  797  871  931  834  794
791  889  926
[16]  870  945  900  814
```

Out[50]:
```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 500 bootstrap replicates

CALL :
boot.ci(boot.out = bobj, conf = 0.9, type = "perc")

Intervals :
Level     Percentile
90%   (832, 926 )
Calculations and Intervals on Original Scale
```

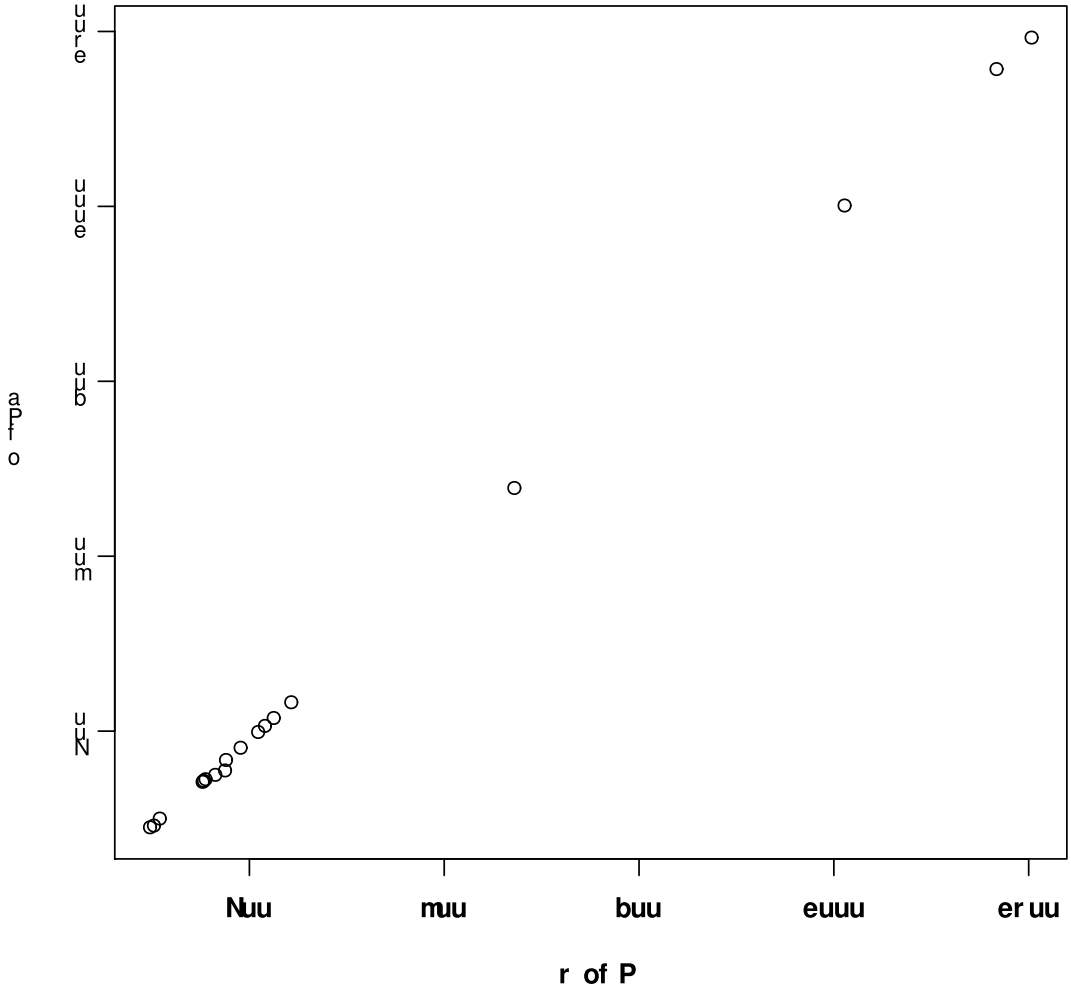
Out[50]:
```
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 999 bootstrap replicates

CALL :
boot.ci(boot.out = bobj, type = "perc")

Intervals :
Level     Percentile
95%   (832, 931 )
Calculations and Intervals on Original Scale
```

Problem #9

Do the output packets of links 2 and 7 have a similar distribution of packets per samping period? One way to determine this is to use the correlation function. Another is to produce a qqplot showing the relationship of one link to another. Produce such a plot and include it in your report.

In [59]:
```
inter_2 <- snmpData[snmpData$if. == 2, ]
inter_7 <- snmpData[snmpData$if. == 7, ]

cols = c('outUCastPkts','outNUCastPkts')

all_cols3 = function(col_name, interf){
    asdf <- eval(parse(text = paste('inter',"_",interf,"$",col_n
ame, sep="")))
    #print(paste('inter',"_",interf,"$",col_name, sep=""))
    #print(eval(parse(text = paste('inter',"_",interf,"$",col_na
me, sep=""))))
    return (asdf)
}

result2 <- sapply(cols, all_cols2, interf='2')
result7 <- sapply(cols, all_cols2, interf='7')
r2rows <- rowSums(result2)
r7rows <- rowSums(result7)

(cor(r2rows,r7rows))
qqplot(r2rows,r7rows)
print('Based on the plot and correlation of .999, they are very
similarly distributed.')
```

```
Out[59]: 0.999831800299426

         [1] "Based on the plot and correlation of .999, they are very si
         milarly distributed."
```



Problem #10

This question does not use the data, but you need to produce an R program. Rather than record the full vector of data when computing a mean, you can use a pair of variables -- you simply record the sum of the terms in the vector (Sx) and the number of items (n) and then report Sx/n. Rearrange the equations equation for variance ($\sum_i^n (x_i - \bar{x})^2$) to demonstrate how you can compute the variance of an arbitrary number of data samples using a small, fixed number of variables, much as I demonstrated for computing the mean. Your solution will need to record three values from the data. You should show your derivation step-by-step. You can either do this using code (i.e. define a series of MyVar() functions that simplify expression into terms that don't involve the mean) or using math (notebooks support MathJax and LaTeX).

In [ ]:
```
#1 var = sum_i_to_n((x_i-x_bar)^2)
# expanding eq1 yields:
#2 var = sum_i_to_n(x_i^2-2*x_i*x_bar+x_bar^2)
#3 x_bar = sum_i_to_n(x_i/n)
# substituting eq3 in for eq 2 yields:
#4 var = sum_i_to_n(x_i^2-2*x_i*sum_i_to_n(x_i/n)-(sum_i_to_n(x_
i/n))^2)
#5 var = sum_i_to_n(x_i^2-2/n*x_i*sum_i_to_n(x_i)-(1/n^2)*(sum_i
_to_n(x_i))^2)

myVar <- function(data){
    n <- length(data)
    result <- sum(data^2-(2/n)*data*sum(data)-(1/n^2)*(sum(data)
^2))
}
```