Homework 4

Alfred J. Williams

2020-11-21

Cleaning The Data:

1. Loaded the Mustang Database from URL into linux by clicking website link. Use gunzip command in linux to unzip.
2. Extractions:

- Extract completed jobs:grep COMPLETED mustang_release_v1.0beta.csv > CompletedMustang.csv.
- Extract Canceled jobs:grep CANCELLED mustang_release_v1.0beta.csv > CancelledMustang.csv
- Extract Timed Out jobs: grep TIMEOUT mustang_release_v1.0beta.csv > TimeOutMustang.csv

3. Use Notepad++ to remove all empty lines and lines containing blank entries (can also be done in R or Linux): Line editor, delete blank rows and find ‘,,’, then bookmark, then delete bookmarks.Save as:

- CompletedMustangScrub.csv
- CancelledMustangScrub.csv
- TimeOutMustangScrub.csv

4. since the original file is too large to fit into notepad ++, concatinate the 3 files together

- cat CompletedMustangScrub.csv CancelledMustangScrub.csv TimeOutMustangScrub.csv

5. Take columns 3,4,5 (the submit times, the start times, and the end times respectively) in linux using the command

- cut -d , -f 3,4,5,8 CompletedMustangScrub.csv > CompletedMustangTimes.csv
- cut -d , -f 3,4,5,8 CancelledMustangScrub.csv > CancelledMustangTimes.csv
- cut -d , -f 3,4,5,8 TimeOutMustangScrub.csv > TimeOutMustangTimes.csv

6. The -6:00 and -7:00 extenstions indicate timezone, convert all the data so that it is in the same timezone. -06:00 is EST and -07:00 is EDT
7. Load these datasets into the R workspace.

Extracting the Service times and Interarrival Times:

The service times were divided by the total number of “cores” used (number of nodes times 24). Some jobs required zero nodes, I have contacted the database managers to inquire about this, for now, I will assume that these jobs require a single set of cores, 24. Interarrival time was not altered. Both are Converted to a numeric. Note, all the elapsed times are recorded in seconds.

```
CompTimes = read.csv('CompletedMustangTimes.csv')

##Load csv, convert into timestamp data frame.
CompTimes = read.csv('CompletedMustangTimes.csv')

#Sort the times in chronological order of arrival.
CompTimes = CompTimes[order(as.POSIXct(CompTimes[,1])),,drop=FALSE]


#Extract arrival and start and end of service.
CompEnd = as.POSIXct(CompTimes[,3], tz = "Etc/GMT+5")
CompStart = as.POSIXct(CompTimes[,2], tz = "Etc/GMT+5")
CompSubmit = as.POSIXct(CompTimes[,1], tz = "Etc/GMT+5")


#Calculate Service Times
CompServiceTime = as.numeric(CompEnd-CompStart)

#Make the 0 nodes 1, divide by 24*number of nodes
CompCores = CompTimes[,4]
CompCores[CompCores == 0] = 1
AdjCompService = CompServiceTime / (24*CompCores)

#Calculate interarrival Times
CompInterArrival = as.numeric(diff(CompSubmit))

#Calculate Arrival Times
ComparrivalTimes = c(0, cumsum(CompInterArrival))

#Clear out nonpositive service times
ComparrivalTimes = ComparrivalTimes[- which(CompServiceTime <= 0)]
AdjCompService = AdjCompService[- which(CompServiceTime <= 0)]
```

My SSQ3 function

```
Myssq3 = function(ArrivalTimes, ServiceTimes){
ti = 0.0 #initial time
tf = ComparrivalTimes[length(ComparrivalTimes)]
Inft = (100*tf) #Some arbitrary large number

#Initialize job statistics

tinn = 0  #time integrated number in node
tinq = 0  #time integrated number in queue
tins = 0  #time integrated number in service

#Initialize Time Statistics

tarr = -1       #next arrival time
tcomp = -1    #next completion time
tcurr = -1    #current time
tnext = -1    #next event time
tlast = -1    #last arrival time

#initialize vector of stored departure times

DepTimes = numeric(0)


#Initialize Counters for departed jobs and number of jobs in node
i = 0
n = 0



#Functions to get next arrival and next service time
#getComparr <- function(n)
#{
#     return(ComparrivalTimes[n])
#}



# Extract the "next service time" function for input into ssq3.
#getAdjCompService <- function(n)
#{
#      return(AdjCompService[n])
#}

#Set the first clock
tcurr = ti                  #set the clock
tarr = ArrivalTimes[1]#Schedule the first arrival
tcomp = Inft                #The first event can't be a completion




while(tarr < tf || n > 0){

  tnext = min(tarr, tcomp)  #grab the next event, either arrival or completion
  if(n > 0){
    tinn = tinn + (tnext - tcurr)*n
    tinq = tinq + (tnext - tcurr)*(n-1)
    tins = tins + (tnext - tcurr)
  }

  tcurr = tnext #advance the clock

  if(tcurr == tarr){  #if we have an arrival,
    n = n + 1         #add one to the node
    tarr = ArrivalTimes[n+i+1] #get the next interarrivalarrival times

    if(is.na(tarr) || is.na(tf)){break}
    if(tarr > tf){    #do not process jobs after the door has closed
      tlast = tcurr   #end the clock as the last job arrives
      tarr = Inft     #the next event must be a completion (which can happen after doors
closed)
    }

    if(n == 1){
      tcomp = tcurr + ServiceTimes[i+1]  #get completion time for first person into syst
em
    }

  }else{
    i = i + 1
    if(tcomp < Inft){DepTimes[i] = tcomp} #store the dep
    n = n-1
    if(n > 0){
      tcomp = tcurr + ServiceTimes[i+1]
    }else{
      tcomp = Inft
    }
  }
}

# [[1]] Mean job delay time
# [[2]] System utilization
# [[3]] average waiting queue length
# [[4]] mean service time
# [[5]] Departure Times
return(list(tinq/i, tins/tcurr, tinq/tcurr, tins/i, DepTimes,i))
}
```

Comparing Job average and time average statistics for Exponentially Driven and Trace Driven Simulations with the same mean service time.

```
TraceSim = Myssq3(ComparrivalTimes, AdjCompService)


ExpSim = Myssq3(ComparrivalTimes, rexp(length(AdjCompService), 1/TraceSim[[4]]))

sprintf("For the trace driven simulation, the mean job delay time is %f. The System Util
ization is %f. The average waiting queue length is %f. The mean service time is %f" ,Tra
ceSim[[1]], TraceSim[[2]], TraceSim[[3]],TraceSim[[4]])
```

```
## [1] "For the trace driven simulation, the mean job delay time is 574587.744040. The S
ystem Utilization is 0.396735. The average waiting queue length is 6567.724693. The mean
service time is 34.708986"
```

```
sprintf("For the exponentially driven simulation, the mean job delay time is %f. The Sys
tem Utilization is %f. The average waiting queue length is %f. The mean service time is
 %f" ,ExpSim[[1]], ExpSim[[2]], ExpSim[[3]],ExpSim[[4]])
```

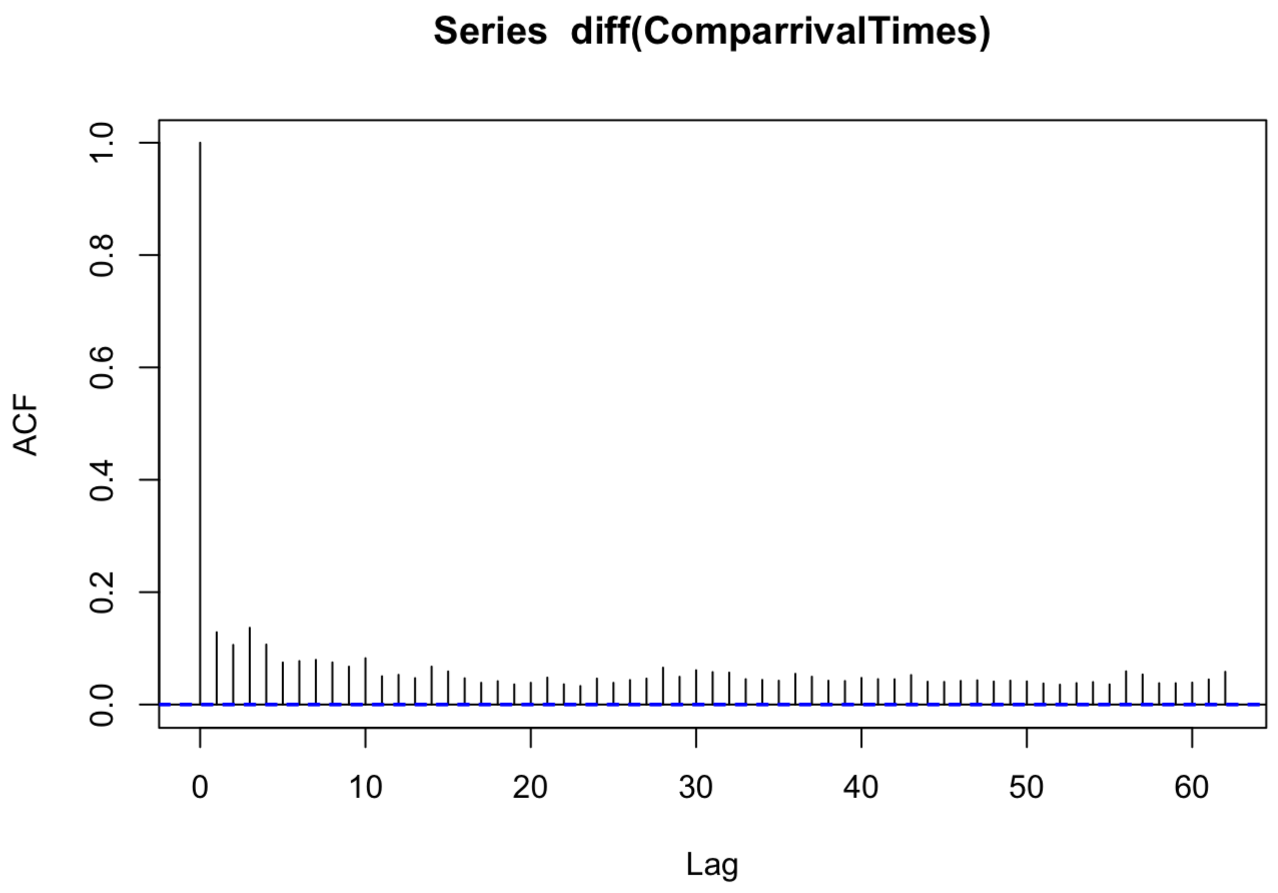
```
## [1] "For the exponentially driven simulation, the mean job delay time is 6428678.7794
87. The System Utilization is 0.396809. The average waiting queue length is 73481.83873
4. The mean service time is 34.715507"
```

	Trace Driven Simulation	**Exp. Driven Simulation**
System Utilization	0.397	0.397
Average Waiting Queue Length	6584 Jobs	73410 Jobs
Mean Job Delay	575477 Seconds	6416063 Seconds
Coefficient of Variation of Service Times	4.496382	1

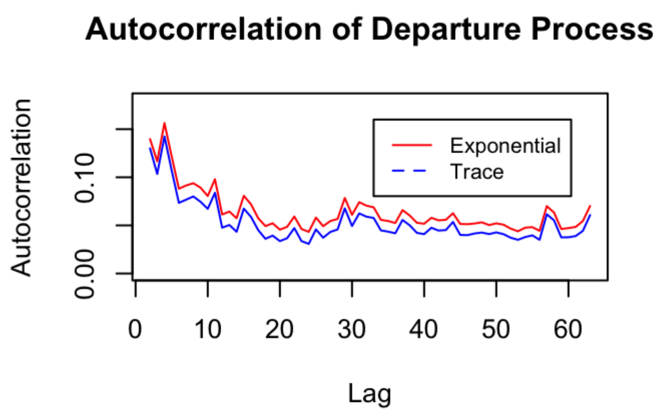
Summary of Values

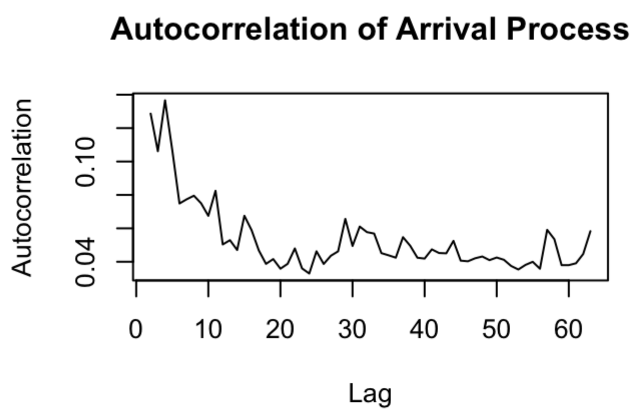
Autocorrelations

```
#Auto correlation of the departure process as a function of lag.
ACFE = acf(diff(TraceSim[[5]]), plot = FALSE)
ACFT = acf(diff(ExpSim[[5]]), plot = FALSE)

ACFS = acf(AdjCompService, plot =FALSE)
ACFSex = acf(rexp(length(AdjCompService), 1/TraceSim[[4]]), plot = FALSE, lag.max = 200)

ACFArr = acf(diff(ComparrivalTimes))
```

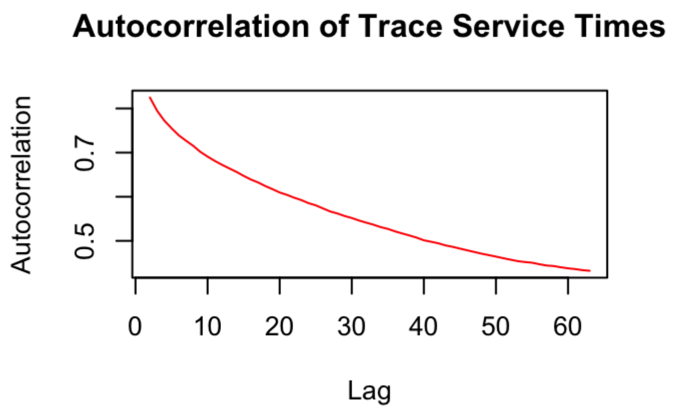


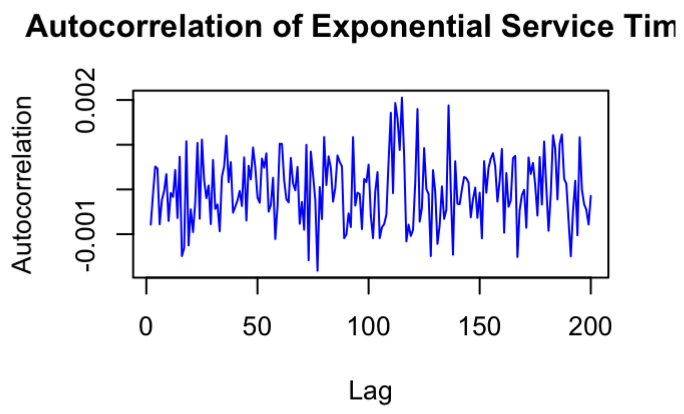
```
par(mfrow = c(2,2))

plot(2:63, ACFE$acf[-1], type = "l", col = 'red', main = 'Autocorrelation of Departure P
rocess', ylab = 'Autocorrelation', xlab = 'Lag', ylim = c(0,.18))
lines(2:63, ACFT$acf[-1], type = "l", col = 'blue')

legend(33, y=0.16, legend = c("Exponential", "Trace"), col = c('red', 'blue'), lty=1:2,
 cex=0.8)

plot(2:63, ACFArr$acf[-1], type = "l", main = 'Autocorrelation of Arrival Process',ylab
 = 'Autocorrelation', xlab = 'Lag')

plot(2:63, ACFS$acf[-1], type = "l", col = 'red',main = 'Autocorrelation of Trace Servic
e Times', ylab = 'Autocorrelation', xlab = 'Lag')

plot(2:200, ACFSex$acf[c(-1,-201)], type = "l", col = 'blue', main = 'Autocorrelation of
Exponential Service Times', ylab = 'Autocorrelation', xlab = 'Lag' )
```





###





The Curious Case of Shifted Departure Process:

Although it appears that the exponential departure process is just a “shifted up” version of the trace departure process. This is not the case as shown below. This vector shows the difference between the two for every 5th lag value. My hypothesis is: this similarity in the shape of the curve may be caused by the fact that both departures are influenced by the same arrival times, and service times of the same mean. It seems that this indicates that arrival times have a much greater influence on the interdeparture autocorrelation than the service times.

```
#every 5th difference be
(ACFE$acf[-1]- ACFT$acf[-1])[seq(from = 1, to = 61, by =5)]
```

```
##  [1] 0.009621986 0.014747529 0.013678400 0.012281189 0.012026678 0.011318664
##  [7] 0.011441479 0.010224699 0.010636244 0.010342360 0.009448311 0.008442750
## [13] 0.010235988
```

Smoothness of Autocorrelation Curves

Since the interarrival times have a much larger standard deviation that the service times, the autocorrelation curve of the arrival process is much more ‘jagged’. For completely random data, like that of the exponential service time, the autocorrelation plot just looks like noise. By observing the formula for autocorrelation below, it is clear that a larger standard deviation will cause the numerator of the equation to be more sensitive to changes in lag, as the accumulation of $y - \bar{y}$ will be larger for more varied data. This might explain why the that arrival times have a much greater influence on the interdeparture autocorrelation than the service times, leading to the seemingly “shifted” departure process above.

$$r_k = \frac{\sum_{i=1}^{N-k}(Y_i - \bar{Y})(Y_{i+k} - \bar{Y})}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}$$

Comparison:

The trace driven and exponentially driven data yielded the same system utilization. This is expected, as system utilization is entirely dependent on: mean service time, the number of jobs, and the total time of simulation– all of these are the same for both simulations. I was very surprised that both resulted in such a small system utilization. This may be caused by the “division by total number of cores”, as this decreased the mean service time but not the total time of simulation, and the long duration that the simulation takes place. I thought at first, that this also might be caused by the large gaps in machine idle time. After analysis of the Interarrival times, the largest gap where no job arrives only takes up about 0.32% of the total time of simulation, and it is possible (and probable) that the server is processing jobs during this time. The average waiting queue length and job delay time for the trace driven simulation was much smaller than that of the exponentially driven simulation. This was strange to me, as a larger coefficient of variation should indicate longer queues and job delay time. The autocorrelation of the departure process as a function of Lag of the departure times looks very similar for the trace driven and exponentially driven simulation, except the exponential is a little bit more self correlated. This similarity in the departure process autocorrelation shape is most likely caused by the shared arrival time data and mean service time. Another curious aspect of this data is the fact that the autocorrelation stays relatively constant as lag increases, I would expect a steeper decline in autocorrelation.

The Effect of system Utilization on Average Queue Length and Wait Time: Original Trace

In order to find interarrival times that gave the desired System utilization, I used a simple guess and test method. Also, I used that fact that when the mean interarrival time increases, system utilization will decrease. I kept the interarrival times constant throughout the simulation for more control.

Mean Waiting Queue Length and Delay Comparison By System Utilization

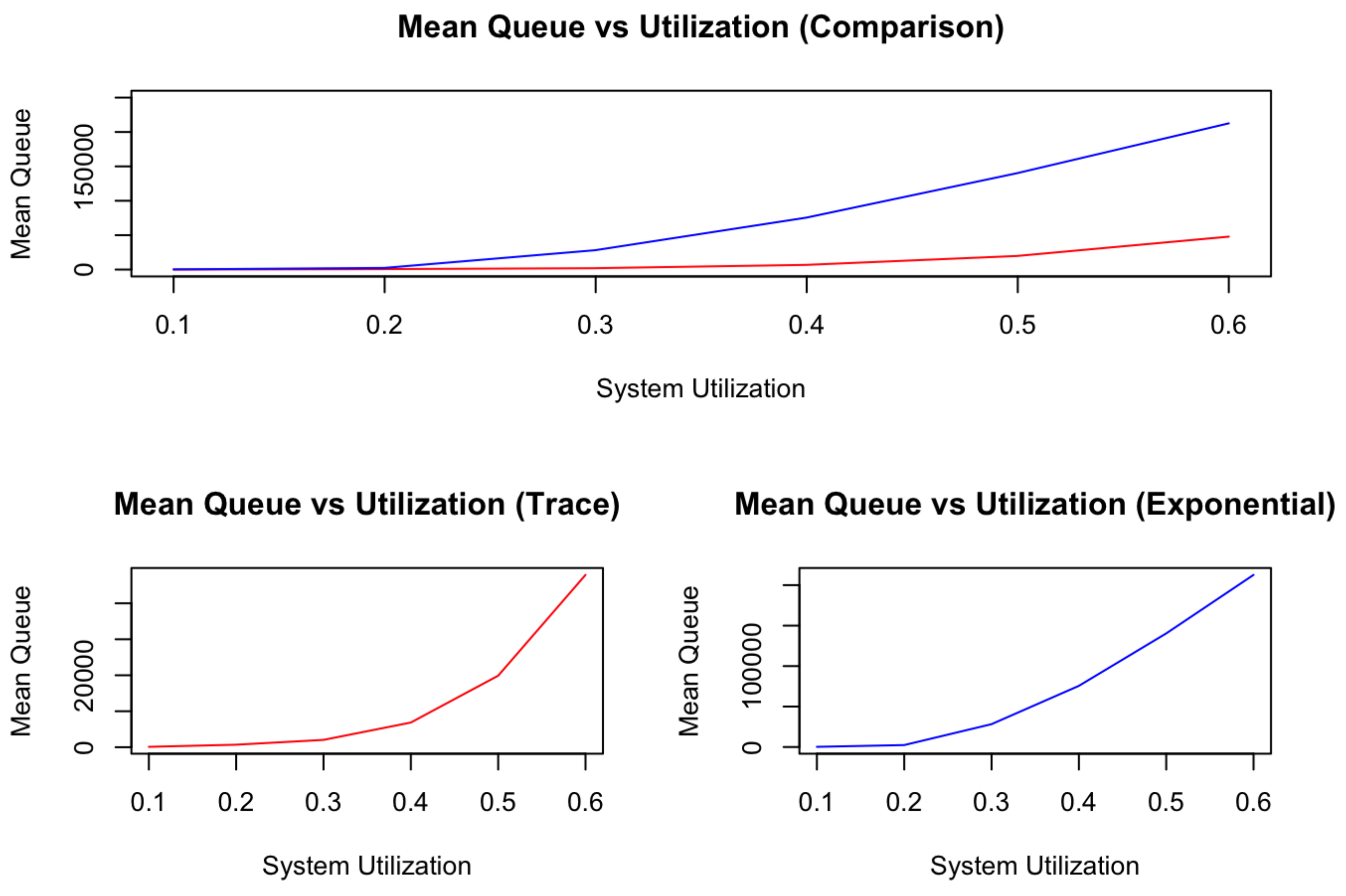
```
#Scalar multiply each that result in system utilizations from 0.1 to 0.9
x = c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6)  #trace

#Average Queue Length for system utilization from 0.1 to 0.9

MeanWaitingQTrace = numeric(0)
MeanWaitingQExp = numeric(0)

mean = TraceSim[[4]]
OGUtil = TraceSim[[2]]
L = length(AdjCompService)

for(i in 1:length(x)){
  MeanWaitingQTrace[i] = Myssq3(ComparrivalTimes, (x[i]/OGUtil)*AdjCompService)[[3]]

  MeanWaitingQExp[i] = Myssq3(ComparrivalTimes, rexp(L, 1/((x[i]/OGUtil)*mean)))[[3]]
}

#Average delay for system utilization from 0.1 to 0.9

MeanDelayTrace = numeric(0)
MeanDelayExp = numeric(0)


for(i in 1:length(x)){
  MeanDelayTrace[i] = Myssq3(ComparrivalTimes, (x[i]/OGUtil)*AdjCompService)[[1]]
  MeanDelayExp[i] = Myssq3(ComparrivalTimes, rexp(L, 1/((x[i]/OGUtil)*mean)))[[1]]
}
```

```
#Testing the scalar multiples
mean = TraceSim[[4]]
x = c(0.1, 0.2, 0.3, 0.4, 0.5, 0.6)
for(i in 1:length(x)){
print(Myssq3(ComparrivalTimes, rexp(length(AdjCompService),1/(mean*(x[i]/0.39673))))[[2
]])
}
```

```
## [1] 0.09995356
## [1] 0.1998644
## [1] 0.29991
## [1] 0.4000473
## [1] 0.5005738
## [1] 0.5995883
```

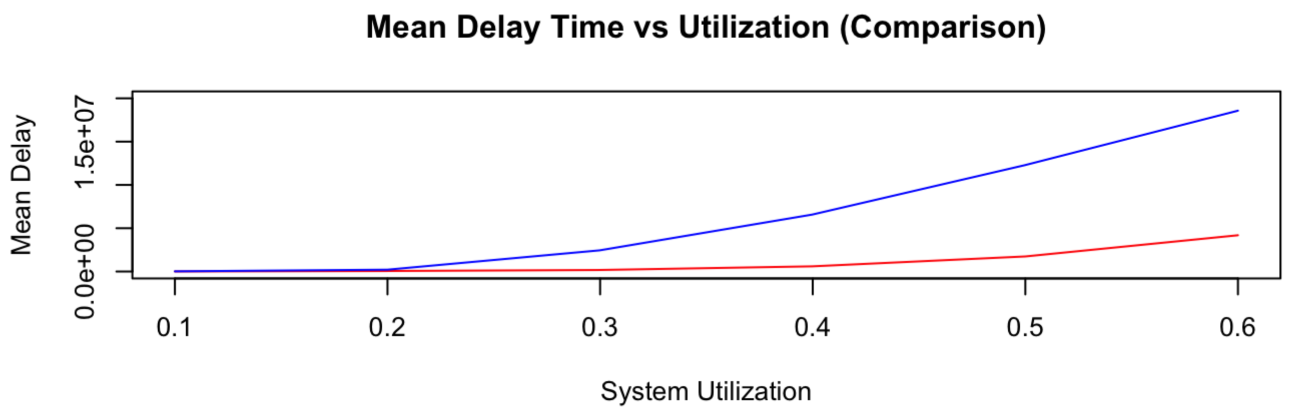
Mean Queue Length Comparison

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))

plot(x, MeanWaitingQTrace, type = "l", col = 'red', main = 'Mean Queue vs Utilization (C
omparison)', ylab = 'Mean Queue', xlab = 'System Utilization', ylim = c(0,250000))

lines(x, MeanWaitingQExp, type = "l", col = 'blue')

legend(0.2, y=600000, legend = c( "Trace", "Exponential"), col = c('red', 'blue'), lty=1
:2, cex=0.8)

plot(x, MeanWaitingQTrace, type = "l", col = 'red', main = 'Mean Queue vs Utilization (T
race)', ylab = 'Mean Queue', xlab = 'System Utilization')

plot(x, MeanWaitingQExp, type = "l", col = 'blue', main = 'Mean Queue vs Utilization (Ex
ponential)', ylab = 'Mean Queue', xlab = 'System Utilization')
```

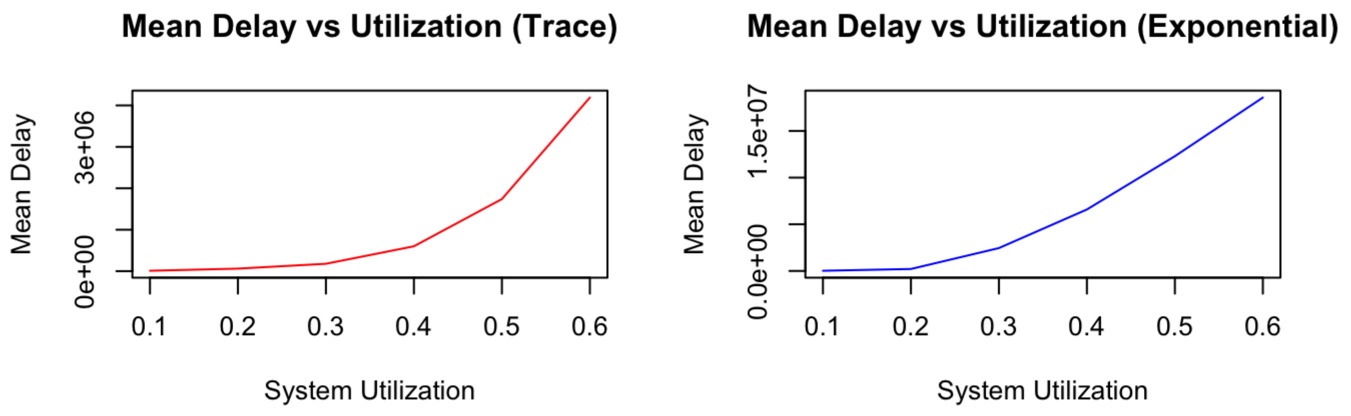


Mean Job Delay Comparison

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE))

plot(x, MeanDelayTrace, type = "l", col = 'red', main = 'Mean Delay Time vs Utilization
 (Comparison)', ylab = 'Mean Delay', xlab = 'System Utilization', ylim = c(0,2*10^7))

lines(x, MeanDelayExp, type = "l", col = 'blue')

legend(0.2, y = 30000000, legend = c( "Trace", "Exponential"), col = c('red', 'blue'), l
ty=1:2, cex=0.8)

plot(x,  MeanDelayTrace, type = "l", col = 'red', main = 'Mean Delay vs Utilization (Tra
ce)', ylab = 'Mean Delay', xlab = 'System Utilization')

plot(x,  MeanDelayExp, type = "l", col = 'blue', main = 'Mean Delay vs Utilization (Expo
nential)', ylab = 'Mean Delay', xlab = 'System Utilization')
```



###



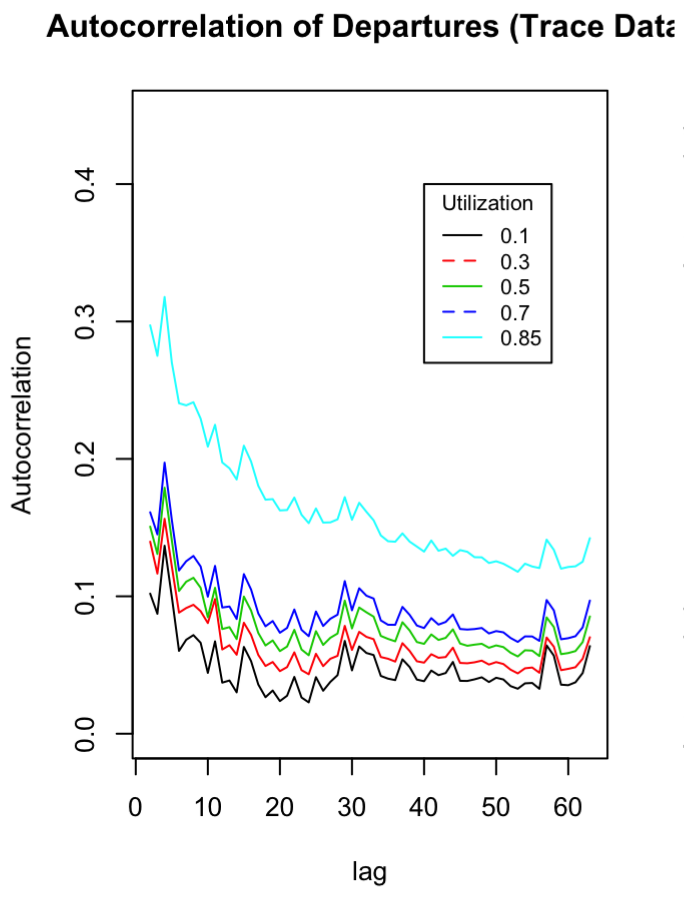
Comparison of trace driven and exponential Delay and Trace by system utilization.

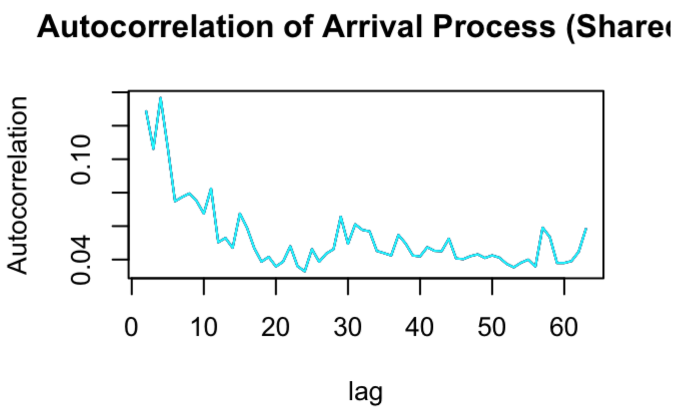
The queue length and delay follow the same pattern: an exponential increase for the trace data and a seemingly exponential increase and leveling off for the Exponential data.

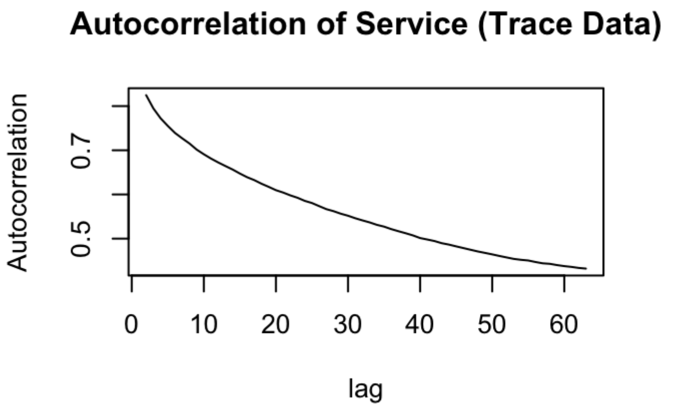
Autocorrolation of the Departure Process for different values of System Utilization

```
#Scalar multiply each that result in system utilizations from 0.1 to 0.9
x = c(1.77, 1, 0.66, 0.55, 0.15)  #trace
y = c(2.50, 1, 0.66, 0.3, 0.1)    #Exp

Util = seq(from = 0.1, to = 0.9, by = 0.2)
#Arrival times for exponential data that result in system utilization from 0.1 to 0.9

DepTimesTrace = vector(mode = "list", length = 5)
DepTimesExp = vector(mode = "list", length = 5)

for(i in 1:length(x)){
  DepTimesTrace[[i]] = Myssq3(x[i]*ComparrivalTimes, AdjCompService)[[5]]

  DepTimesExp[[i]] = Myssq3(y[i]*ComparrivalTimes, rexp(length(AdjCompService), 1/TraceS
im[[4]]))[[5]]
}



#Autocorrelation as a function of lag for Trace data Departures
Autotrace = vector(mode = "list", length = 5)
for(i in 1:length(x)){
Autotrace[[i]] = acf(diff(DepTimesTrace[[i]]), plot = FALSE, lag.max = 62)
}

#Autocorrelation as a function of lag for Trace data interarrival

Intertrace = vector(mode = "list", length = 5)
for(i in 1:length(x)){
Intertrace[[i]] = acf(diff(x[i]*ComparrivalTimes), plot = FALSE, lag.max = 62)
}

ACFService = acf(AdjCompService, plot = FALSE, lag.max = 62)

#Plot of departure process autocorrelation
layout(matrix(c(1,2,1,3), 2, 2, byrow = TRUE))
plot(2:63, Autotrace[[1]]$acf[-1], type = "l", col = 1,main = 'Autocorrelation of Depart
ures (Trace Data)', ylab = 'Autocorrelation', xlab = "lag", ylim = c(0,0.45))
lines(2:63, Autotrace[[2]]$acf[-1], type = "l", col = 2)
lines(2:63, Autotrace[[3]]$acf[-1], type = "l", col = 3)
lines(2:63, Autotrace[[4]]$acf[-1], type = "l", col = 4)
lines(2:63, Autotrace[[5]]$acf[-1], type = "l", col = 5)
legend(40, y=0.40, title = 'Utilization', legend = c(0.1,0.3,0.5,0.7,0.85), col = 1:5, l
ty=1:2, cex=0.8)

#Plot of arrival process autocorrelation
plot(2:63, Intertrace[[1]]$acf[-1], type = "l", col = 1,main = 'Autocorrelation of Arriv
al Process (Shared)', ylab = 'Autocorrelation', xlab = "lag")
lines(2:63, Intertrace[[2]]$acf[-1], type = "l", col = 2)
lines(2:63, Intertrace[[3]]$acf[-1], type = "l", col = 3)
lines(2:63, Intertrace[[4]]$acf[-1], type = "l", col = 4)
lines(2:63, Intertrace[[5]]$acf[-1], type = "l", col = 5)
legend(50, y=0.30, title = 'Utilization', legend = c(0.1,0.3,0.5,0.7,0.85), col = 1:5, l
ty=1:2, cex=0.8)
```

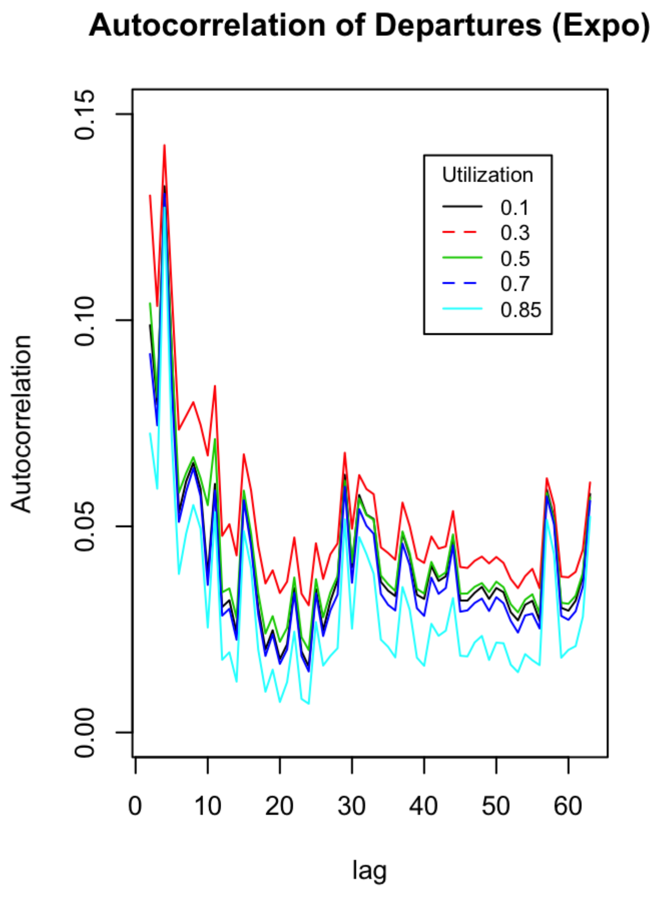
```
#Service Times autocorrelation
plot(2:63, ACFService$acf[-1], type = "l", col = 1,main = 'Autocorrelation of Service (T
race Data)', ylab = 'Autocorrelation', xlab = "lag")
```

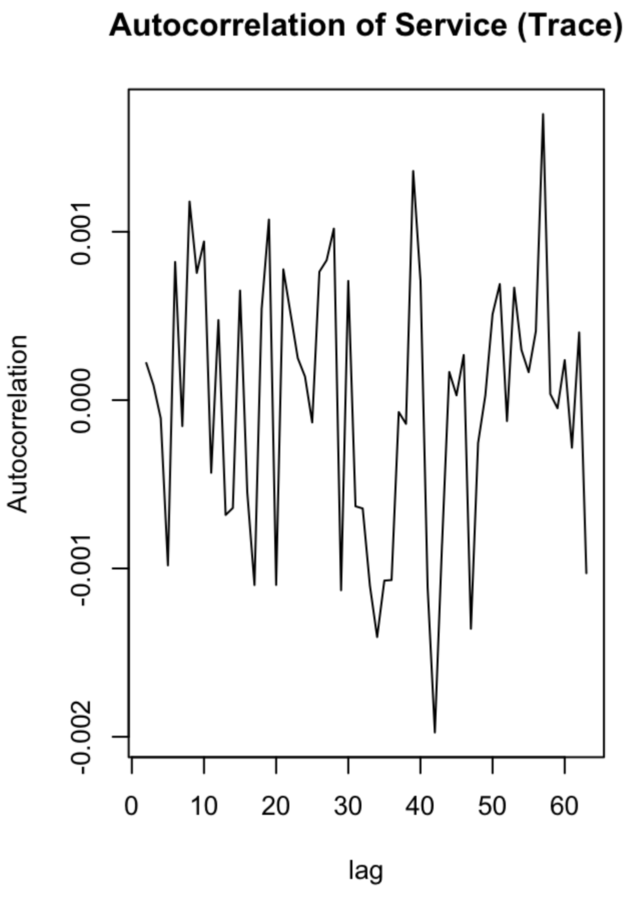






```
#Autocorrelation as a function of lag for Exp

Autoexp = vector(mode = "list", length = 5)
for(i in 1:length(x)){
Autoexp[[i]] = acf(diff(DepTimesExp[[i]]), plot = FALSE, lag.max = 62)
}


ServEx = acf(rexp(length(AdjCompService), 1/TraceSim[[4]]), plot = FALSE, lag.max = 62)

#Plot
layout(matrix(c(1,2,1,2), 2, 2, byrow = TRUE))

plot(2:63, Autoexp[[1]]$acf[-1], type = "l", col = 1,main = 'Autocorrelation of Departur
es (Expo)', ylab = 'Autocorrelation', xlab = "lag", ylim = c(0,0.15))
lines(2:63, Autoexp[[2]]$acf[-1], type = "l", col = 2)
lines(2:63, Autoexp[[3]]$acf[-1], type = "l", col = 3)
lines(2:63, Autoexp[[4]]$acf[-1], type = "l", col = 4)
lines(2:63, Autoexp[[5]]$acf[-1], type = "l", col = 5)
legend(40, y=0.14, title = 'Utilization', legend = c(0.1,0.3,0.5,0.7,0.85), col = 1:9, l
ty=1:2, cex=0.8)


plot(2:63, ServEx$acf[-1], type = "l", col = 1,main = 'Autocorrelation of Service (Trac
e)', ylab = 'Autocorrelation', xlab = "lag")
```





Analysis of the Effects of arrival times with increasing System Utilization on mean queue length, job delay, and autocorrelation of departure process.

The autocorrelation of the trace and exponential departure data behaved as expected; larger system utilization yielded a much stronger autocorrelation. For the exponential data, the autocorrelation is lesser overall and much closer for different values of utilization.