ANALYSING STOCHASTIC CALL DEMAND WITH TIME VARYING PARAMETERS

A Thesis Submitted to the College of
Graduate Studies and Research
In Partial Fulfillment of the Requirements
For the Degree of Master of Science
In the Department of Mathematics and Statistics
University of Saskatchewan
Saskatoon

By

Song Li

ABSTRACT

In spite of increasingly sophisticated workforce management tools, a significant gap remains between the goal of effective staffing and the present difficulty in predicting the stochastic demand of inbound calls. We have investigated the hypothesized nonhomogeneous Poisson process model of modem pool callers of the University community. In our case, we tested if the arrivals could be approximated by a piecewise constant rate over short intervals. For each of 1 and 10-minute intervals, based on the close relationship between the Poisson process of arrivals and the exponential distribution of interarrival times, the test results did not show any sign of homogeneous Poisson process. We have examined the hypothesis of a nonhomogeneous Poisson process by a transformed statistic. Quantitative and graphical goodness-of-fit tests have confirmed nonhomogeneous Poisson process.

Further analysis on the intensity function revealed that linear rate intensity was woefully inadequate in predicting time varying arrivals. For sinusoidal rate model, difficulty arose in setting the period parameter. Spline models, as an alternative, had more control of balance between data fitting and smoothness, which was appealing to our analysis on call arrival process.

ACKNOWLEDGMENTS

This thesis grew out of a summer research project provided by my co-supervisor Dr. Raj Srinivasan. I'm indebted to Raj, who opened my eyes for queueing theory and grateful for his invaluable support. I'm also bequeathed this work to Dr. Mik Bickis, who agreed to guide me through and ensured it to a high standard, especially with his penetrating comments. I'd also like to extend my appreciation to the other committee members for their suggestions and comments.

Finally, special thanks go to Zhidong Zhang for his collaboration in data cleaning and Dr. Duncan Murdoch for pointing out the close relationship between spline regression and smoothing spline in R, a statistical computing package.

DEDICATION

To my parents

Jing and Kaiming

my wife

Ran

Table of Contents

List of Figures

List of Tables

Chapter 1
INTRODUCTION

1.1 Growing Call Centre Industry

As its name suggests, a call centre is a place where customers via phone lines seek help or services provided by a company or an organization. Traditionally, a call centre has a physical location and is composed of telephone equipment and representatives capable of delivering services by telephone. A modern definition extends a call centre to a centralized service entity, known as the contact centre, which can sell things or offer services through an automatic voice unit, fax, email or even an interactive online website [10]. A contact centre need not be a single work place but can be a network of multi-task units perhaps all over the world.

Twenty or thirty years ago, call centres were first introduced as tele-wired kiosks for financial institutions, airline and catalog ordering companies [10]. Nowadays, call centres have become one of the fastest growing industries in North America. Estimates of the number of call centres in North America range from 20,000 to as high as 200,000 depending on what definition of call centre is used [10]. Some experts only count centres that have above a certain number of representatives or agents. One typical example of a large scale call centre is for toll-free call service, which counts 104 million daily requests on the AT&T network [15]. Some experts also count the call centres that might only have four of five agents, because these small centres can probably handle all kinds of jobs with a much reduced request load when compared to a large scale toll-free call centre.

Regardless of the size, call centres face workforce and service management challenges, such as problems of recruiting, training and staffing of representatives and quality assurance of service. In 1999 in the United States there were 1.55 million representatives working in call centres with an unprecedented employment growth rate of more than 8% per year [7, 33]. The call centre has indeed evolved as one of the fastest growing industries in North America.

1.2 Modelling Stochastic Inbound Calls

There are call centres that handle *outbound, inbound* or both types of calls. Outbound calls are defined to be those calls that are initiated within a call centre. Therefore, the schedule for outbound calls is determined by the call centre. Examples of outbound services are telephone surveys and tele-marketing. In contrast, inbound calls are those initiated from outside a call centre, the call demand of which is unknown to the call centre. The managers at the inbound call centres have difficulty in scheduling the appropriate number of agents to handle incoming calls.

Both operational and quality efficiencies are essential to a call centre. In most call centres, the costs of hiring, training and retention of agents account for 60%-70% of operating expenses [15]. Given this kind of large costs, a good-practice call centre of high efficiency would like to achieve an average 90% to 95% agent utilization level so that in most of the time agents are busy answering the phone calls.

Meanwhile, on the quality side, a call centre should have an adequate number of agents so that, ideally speaking, almost no customers have to wait. In practice, the quality of a call centre can be measured by the average waiting time of customers. At an emergency 911 call centre, as an interesting example of service quality measured by the waiting time, one does not want to see any caller unanswered simply because there are not enough agents scheduled. To balance between utilization requirements and quality efficiency, one needs an optimal plan of scheduling for agents being able to handle phone calls during busy and non-busy times. For practical purposes, one might aim at accurate forecasts for call loads, at least down to the half hour as a good practice for hiring, so one would be able to schedule the agents accordingly.

1.3 Objectives

The primary purpose of this thesis is to provide input to the short term scheduling process of agent staffing, which has two aspects:

1. Difficulty in understanding and modelling the arrival process of a call centre is prevailing. We investigate the hypothesized nonhomogeneous Poisson process with respect to our data set.

2. Once the arrival process is deemed appropriate, we are mainly interested in modelling the time varying arrival rate. We examine the current modelling approaches.

1.4 Thesis Outline

A strong need for routine staff scheduling motivates our investigation of methods of forecasting for stochastic inbound call load. This chapter introduces the call centre as one of the most rapidly growing industries in North America. Research objectives are also presented in this chapter. As an overview, the thesis is composed of the following chapters,

In Chapter 2, we sketch a picture of the time varying arrival pattern of the University's dial-up service and present a viewpoint from a queueing perspective.

In Chapter 3, we survey methods of modelling time varying arrival rates with a review of nonhomogeneous Poisson processes. A number of estimation procedures for linear and sinusoidal rate models are discussed. A spline-based modelling scheme is also provided as an alternative to the linear and sinusoidal models.

In Chapter 4, we empirically test the homogeneous and nonhomogeneous Poisson models with respect to our dial-up call arrivals. A set of goodness-of-fit tests are performed. Graphical methods are provided to explore and examine the nature of data that purely quantitative tests might miss.

In Chapter 5, we compare the fit of the linear, sinusoidal and spline models to the intensity rate with a discussion about likelihood ratio test results. Estimates of parameters computed by *ordinary least squares* (OLS), *iterative weighted least squares* (IWLS) and *maximum likelihood* (ML) procedures are also presented for the parametric linear and sinusoidal rate models.

In Chapter 6, we highlight results regarding the underlying arrival process and the intensity rate. Limitations and future research directions are discussed in the end.

Chapter 2
MODEM POOL DATA: AN ANALOG OF A CALL CENTRE

2.1 Introduction

The University dial up remote access service (U-Connect) enables faculty, staff, and students to access the campus network for many network-based services, such as electronic mail, web servers, the library, shared servers, and the Internet, from their home and other off-campus locations as if they were on-campus. The service consists of a group of modems rented from the local phone company and a high speed connection from the modems to the University campus network, which represents an analog of an inbound call centre as shown in Figure 2.1.

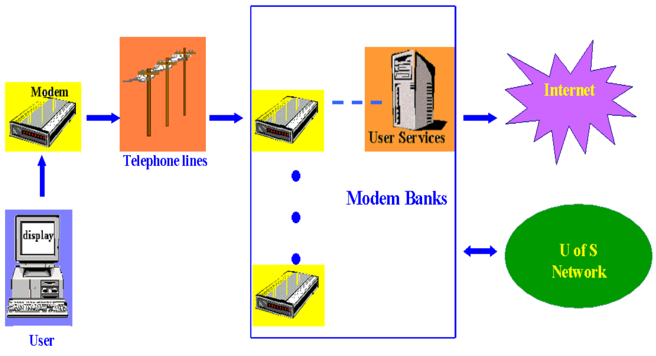


Figure 2.1 Network Layout

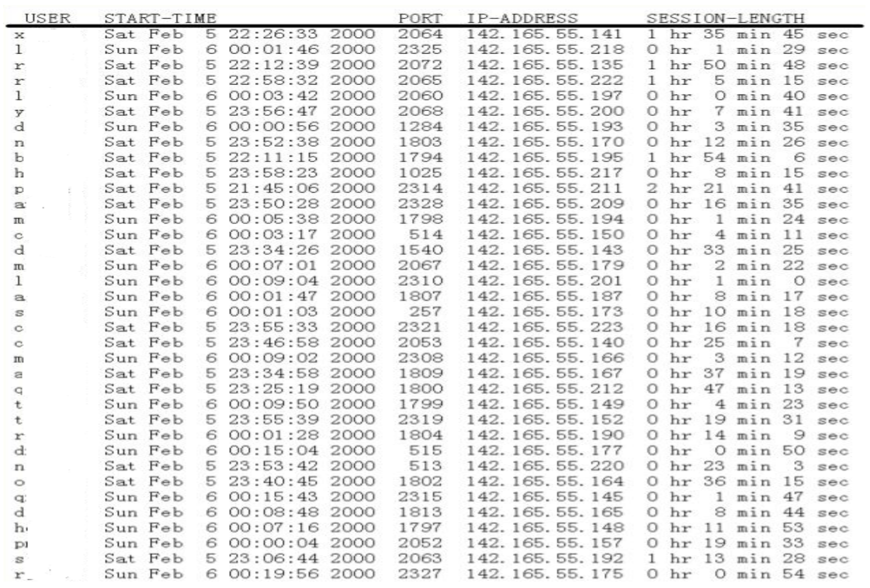
Unlike the traditional call centres, this service does not have to be staffed by any human representatives as a result of the modern technological network revolution. The service makes use of the fastest commonly available 56K modems and currently charges a reduced subscription cost of $11.95 per month for 60 hours to the University community users.

However, the dial-up service faces a harsh competitor, high speed internet. In 2002, commercial internet providers (ISP) like Shaw Cable introduced a $22.95 unlimited high-speed Internet service to the students, while the dial up lines caused a high degree of annoyance among the users, since the lines were almost always congested. In this situation, a cost and benefit analysis of resources allocated to the dial-up services such as modem renting, becomes more essential to the University.

2.2 The Raw Calling Record

The U-Connect service runs 24 hours a day, 7 days a week. The raw data consists of a five year record of dial-up user calling history since 1999. In terms of the number of transactions, the volume is huge. In 2000, for example, there were on average over half a million transactions, equivalent to 1,693 daily transactions. The physical volume of data files for all years can take up 1.4GB of a hard drive.

Figure 2.2 presents a snapshot of a raw calling record with minor modifications applied such as underlining the headings. Each row provides a complete connection record of a dial-up user, comprised of ID, start time, port number, assigned local IP and session length. In the first column, each **USER** is identified by a unique network service ID (NSID). Each NSID consists of 3 letters followed by 3 digits, which corresponds to an individual affiliated with the University, a student, a staff or a faculty member. For privacy reasons, we have removed 2 of 3 letters and all digits of the actual NSIDs from Figure 2.2. The second heading is **START-TIME** listed in columns as day of week, month of year, date, time and year. We usually call it *arrival time*. The subsequent columns are assigned **PORT** and local **IP**. The last heading is **SESSION-LENGTH** which is the total amount of time a user is connected.



```
USER     START-TIME                    PORT  IP-ADDRESS        SESSION-LENGTH
x        Sat Feb  5 22:26:33 2000      2064  142.165.55.141    1 hr 35 min 45 sec
l        Sun Feb  6 00:01:46 2000      2325  142.165.55.218    0 hr  1 min 29 sec
r        Sat Feb  5 22:12:39 2000      2072  142.165.55.135    1 hr 50 min 48 sec
r        Sat Feb  5 22:58:32 2000      2065  142.165.55.222    1 hr  5 min 15 sec
l        Sun Feb  6 00:03:42 2000      2060  142.165.55.197    0 hr  0 min 40 sec
y        Sat Feb  5 23:56:47 2000      2068  142.165.55.200    0 hr  7 min 41 sec
d        Sun Feb  6 00:00:56 2000      1284  142.165.55.193    0 hr  3 min 35 sec
n        Sat Feb  5 23:52:38 2000      1803  142.165.55.170    0 hr 12 min 26 sec
b        Sat Feb  5 22:11:15 2000      1794  142.165.55.195    1 hr 54 min  6 sec
h        Sat Feb  5 23:58:23 2000      1025  142.165.55.217    0 hr  8 min 15 sec
p        Sat Feb  5 21:45:06 2000      2314  142.165.55.211    2 hr 21 min 41 sec
a        Sat Feb  5 23:50:28 2000      2328  142.165.55.209    0 hr 16 min 35 sec
m        Sun Feb  6 00:05:38 2000      1798  142.165.55.194    0 hr  1 min 24 sec
c        Sun Feb  6 00:03:17 2000       514  142.165.55.150    0 hr  4 min 11 sec
d        Sat Feb  5 23:34:26 2000      1540  142.165.55.143    0 hr 33 min 25 sec
m        Sun Feb  6 00:07:01 2000      2067  142.165.55.179    0 hr  2 min 22 sec
l        Sun Feb  6 00:09:04 2000      2310  142.165.55.201    0 hr  1 min  0 sec
a        Sun Feb  6 00:01:47 2000      1807  142.165.55.187    0 hr  8 min 17 sec
s        Sun Feb  6 00:01:03 2000       257  142.165.55.173    0 hr 10 min 18 sec
c        Sat Feb  5 23:55:33 2000      2321  142.165.55.223    0 hr 16 min 18 sec
c        Sat Feb  5 23:46:58 2000      2053  142.165.55.140    0 hr 25 min  7 sec
m        Sun Feb  6 00:09:02 2000      2308  142.165.55.166    0 hr  3 min 12 sec
s        Sat Feb  5 23:34:58 2000      1809  142.165.55.167    0 hr 37 min 19 sec
q        Sat Feb  5 23:25:19 2000      1800  142.165.55.212    0 hr 47 min 13 sec
t        Sun Feb  6 00:09:50 2000      1799  142.165.55.149    0 hr  4 min 23 sec
t        Sat Feb  5 23:55:39 2000      2319  142.165.55.152    0 hr 19 min 31 sec
r        Sun Feb  6 00:01:28 2000      1804  142.165.55.190    0 hr 14 min  9 sec
d        Sun Feb  6 00:15:04 2000       515  142.165.55.177    0 hr  0 min 50 sec
n        Sat Feb  5 23:53:42 2000       513  142.165.55.220    0 hr 23 min  3 sec
o        Sat Feb  5 23:40:45 2000      1802  142.165.55.164    0 hr 36 min 15 sec
q        Sun Feb  6 00:15:43 2000      2315  142.165.55.145    0 hr  1 min 47 sec
d        Sun Feb  6 00:08:48 2000      1813  142.165.55.165    0 hr  8 min 44 sec
h        Sun Feb  6 00:07:16 2000      1797  142.165.55.148    0 hr 11 min 53 sec
p        Sun Feb  6 00:00:04 2000      2052  142.165.55.157    0 hr 19 min 33 sec
s        Sat Feb  5 23:06:44 2000      2063  142.165.55.192    1 hr 13 min 28 sec
r        Sun Feb  6 00:19:56 2000      2327  142.165.55.175    0 hr  0 min 54 sec
```

Figure 2.2 A Snapshot of a Raw Record

2.3 Data Cleaning

The accuracy of the recorded times of arrivals and session length, and the counts of users summarised by each time interval are essential to our statistical analysis of the process of calls. In practice, the data set is not always ready to use. We devoted a substantial amount of effort in data cleaning to reconcile any inconsistencies in the raw data set. Technical recording errors or exceptions were found quite often. Some examples of inconsistency that have been successfully dealt with (see Appendix A for scripts used for data cleaning) include:

- Incomplete call history: We found various instances of missing data. We removed the records which lack **START-TIME**, because our analysis of arrivals of dial-up user is concerned about times of arrivals. For the other missing records, for example, which do not contain IP addresses, we left them in the data set, since our analysis has nothing to do with IP address.

- Extreme values: We encountered a great number of extreme values in recording times. In one case, we found that a few records of **SESSION-LENGTH** in one day were extremely large. We implemented a script to detect and remove the extreme recording times (See Appendix A for details) with any session time beyond 72 hours being considered extreme.

- Repeated records: If there is indeed any repeated record in the same file, we only keep one record. We removed redundant records by matching call request attributes such as **USER**, **IP** and **START-TIME**.

- Corrupted data structure: We found some information was placed under a wrong heading. In one day's record, all columns of records were shifted. For example, user names were shifted one column to the right under the heading of **START-TIME**. To be safe, we wrote a UNIX script to read in all the information from all the raw data files and placed it under the correct headings into new files (See script **MyCleaner** in Appendix A for details).

- 24/7 call history: Call load was recorded on a continuous time scale. Each day was stored in one single file, containing 24 hours of records. However, it happened quite often that a call extends overnight. As one can see from the first few lines of Figure 2.2, February 6th contained calls initialized on the 5th and spanned over midnight. As our primary interest was the arrival process, we regrouped arrivals initiated on the same day into the same file, regardless of the session length.

- Embedded in an HTML environment: Since the records were embedded by HTML tags such as **Content-type**, **HTML**, **HEAD**, **PRE**, **BODY**, **TITLE** and **USER**, the script **MyCleaner** (in Appendix A) was used to remove these marks.

2.4 Elements of Time Varying Arrivals

The dial-up user call volume is neither constant nor easily predictable over time. The arrival pattern exhibits many elements of a time varying rate.

At the aggregate level, Figure 2.3 shows the number of arrivals in each month from January 2000 to June 2003. Call volume exhibited strong seasonal trends throughout the years. The kind of patterns clearly repeated from year to year. The call volume reached the global peak in 2000 and the overall trend was decreasing throughout the years. In the summer of 2002, it reached the global minimum, which is probably due to the introduction of the special student rate high speed offered by commercial ISPs. (Since no data is available after May 2003, the figure shows no arrivals at that point.)

Given the yearly patterns, one would like to go one step further. As one example, Figure 2.4 zooms in on calls in 2000. The figure presents a number of signs of seasonality. The seasonality, which is common amongst the other years, can be summarised as follows,

- In winter, the number of calls climbs up from January when people get back from Christmas holidays. It reaches the peak in March when it is close to the end of winter term. As we know the regular academic year usually ends at the start of April, March is the peak dial-up call time as the faculty and students might rush through unfinished materials and get prepared for final exams.
- In the spring and summer terms (from May through August) call volume is decreasing while there is a slight increase after July. In general, the spring and summer terms have lighter loads because of lower enrollment at the University.
- In September, the regular term resumes. It starts climbing up again until November when it reaches another peak.

This trend has a good match with the university academic calendar: three typical periods throughout a year. Namely, Period 1: from January to April (winter term); Period 2: from May to August (summer and spring term); Period 3: September to December (fall term).

Figure 2.5 shows a comparison amongst years. As we can see, these years share a common monthly pattern. The number of calls reaches a peak in March and November

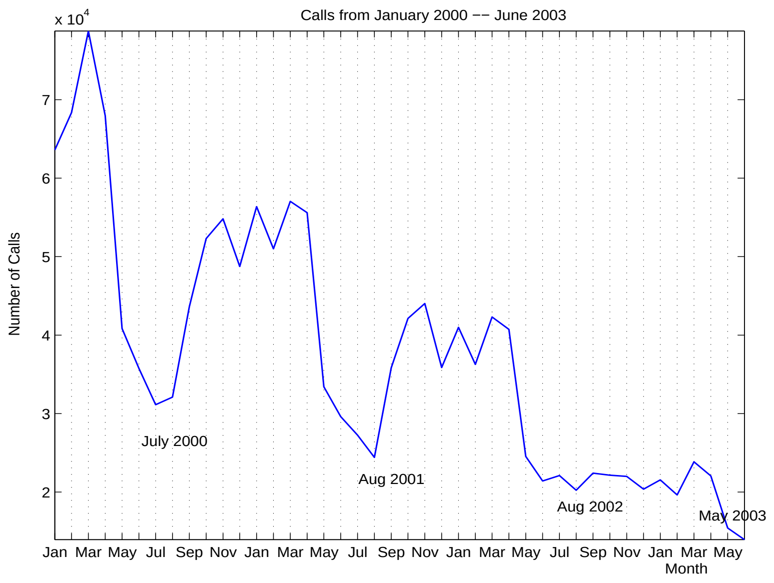


Figure 2.3 Aggregate Trend, January 2000 to June 2003

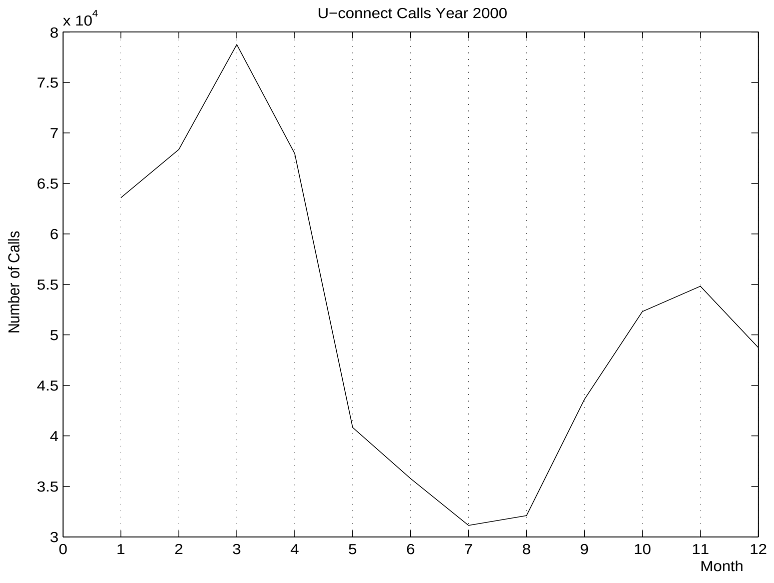


Figure 2.4 An Example of Annual Calls

except in 2002. Note that since the data is dated up to July 7th, 2003, the 2003 plot drops to zero after July 2003. This pattern also reinforces the conclusion we drew from Figure 2.4 that March and November are the busiest times of the year and during summer very few people use the dial-up service.

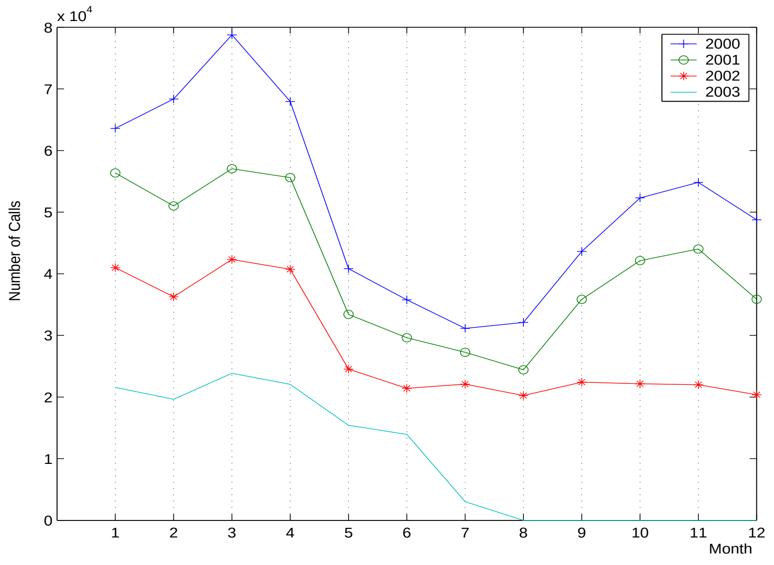


Figure 2.5 Annual Call Comparison

Based on these similar patterns amongst years, we can focus on one year's data. Interestingly, the number of arrivals in 2002 did not climb up after September as the previous years. One possibility for no climb-up is that people switched from dial-up to faster and not-so-expensive high speed internet as both Sasktel and Shaw (commercial ISPs) each offered a special deal for high speed internet to students/falculty members for $25/month at the time.

Based on the previous analysis of the call pattern, we pick three typical months to represent Period 1, Period 2 and Period 3, respectively. Each of Figures 2.6, 2.7 and 2.8 shows a comparison of the number of calls for three selected months in a year (i.e., 2000, 2001 and 2002). We do not consider 2003 due to its incomplete data. (For those months that do not have the 31st day, it shows zero on that day in the figures.)

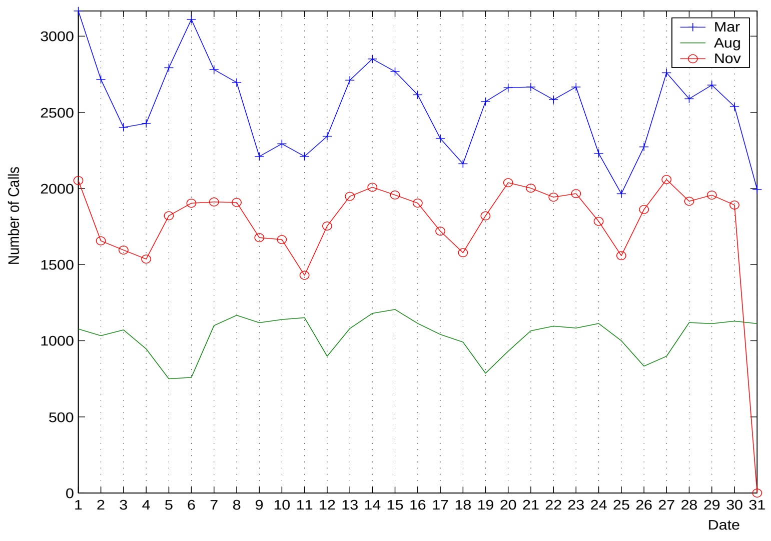


Figure 2.6 Monthly Comparison 2000

Based on these comparisons, we can see that the number of arrivals is directly related to day of week but not date of month. In Figure 2.7, for example, on the 10th of January, the number of arrivals reaches the monthly peak, while on the same date of November, the number of arrivals is close to the monthly minimum. One should note that the reason that in 2000 the arrival trend of these months looks almost identical in Figure 2.6 is not because of the same dates but the fact that the dates coincide with the same day of the week. For example, March 1st and November 1st

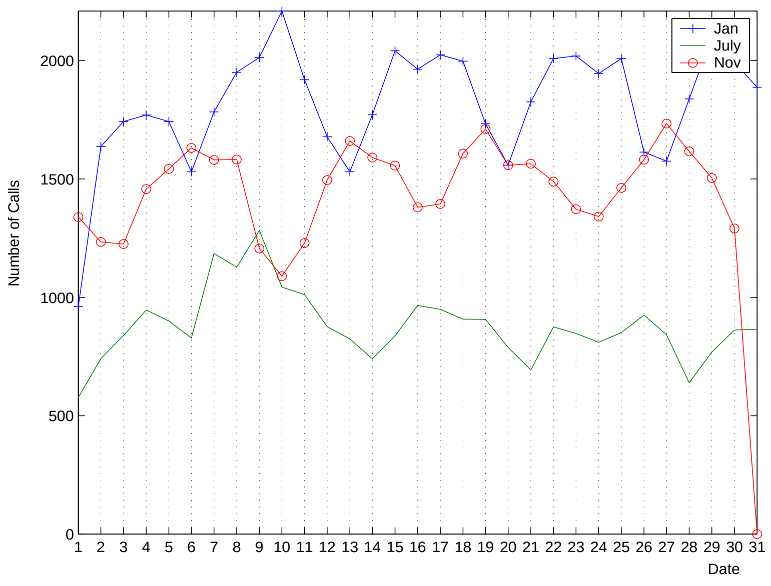


Figure 2.7 Monthly Comparison 2001

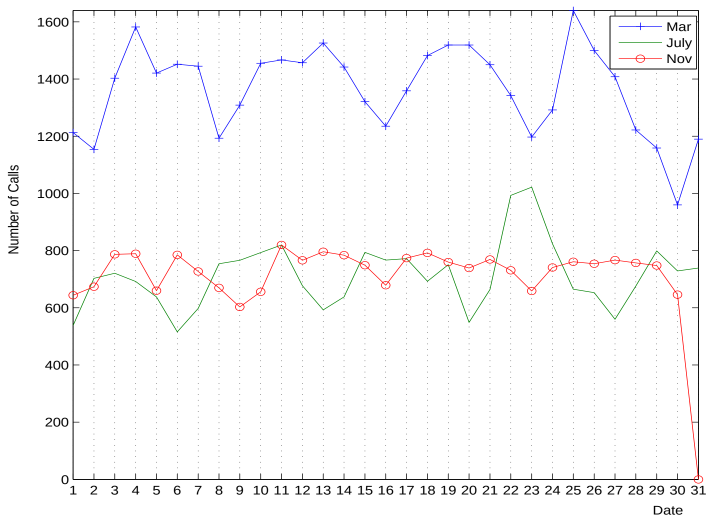


Figure 2.8 Monthly Comparison 2002

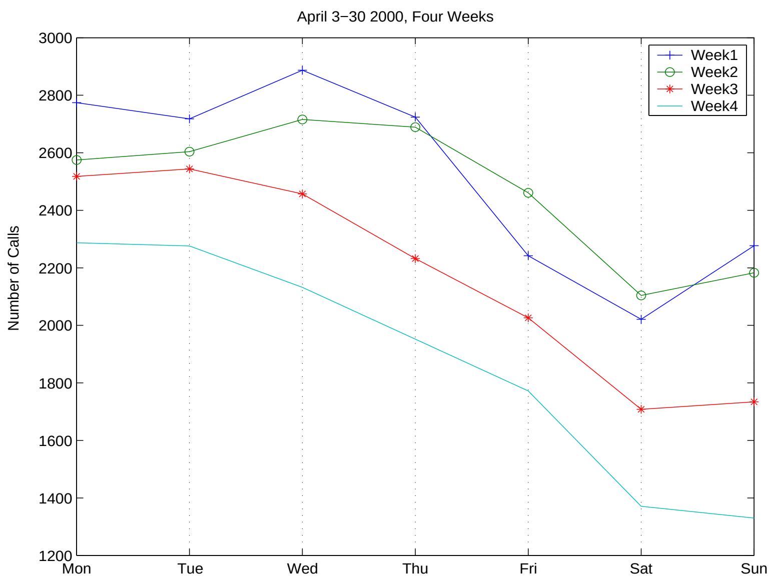


Figure 2.9 An Example of Weekly Comparison 2000

are both Tuesdays.

Given the day of week pattern, let's examine the weekly calls a bit closer. Figures 2.9, 2.10 and 2.11 sample three months, respectively, for years 2000, 2001 and 2002. These figures illustrate the dramatic differences in call volume between the peak volume time (on a week day) and the minimum volume time (at the weekend). They also indicate a similar weekly pattern.

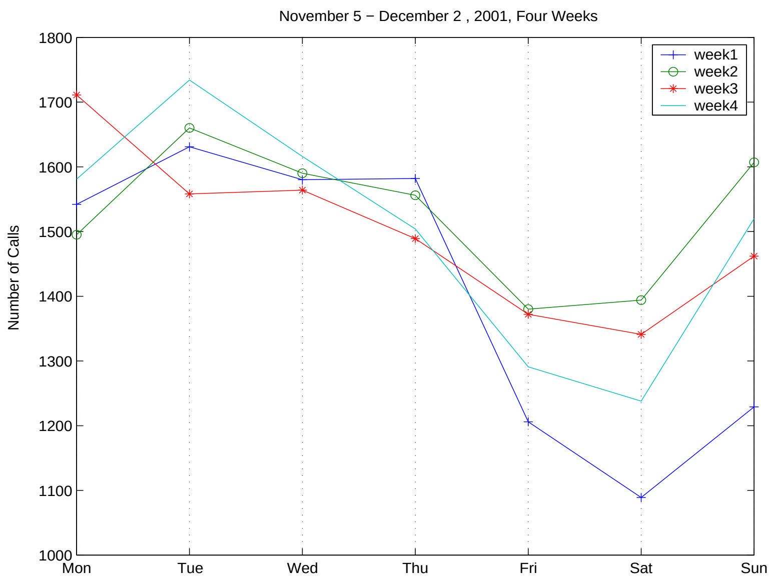


Figure 2.10 An Example of Weekly Comparison 2001

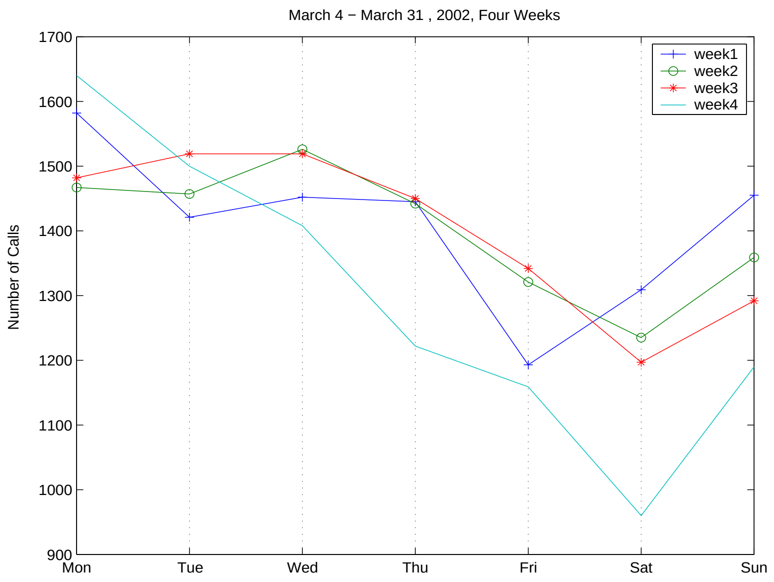


Figure 2.11 An Example of Weekly Comparison 2002

Figure 2.12 compares two work days with two holidays. According to this figure, a work day generally possesses a higher call volume than a holiday. Christmas and New Years days tend to have a smaller number of call requests. Nonetheless, they do share a common hourly pattern, in which the number of arrivals reaches the maximum after supper and the minimum before breakfast.

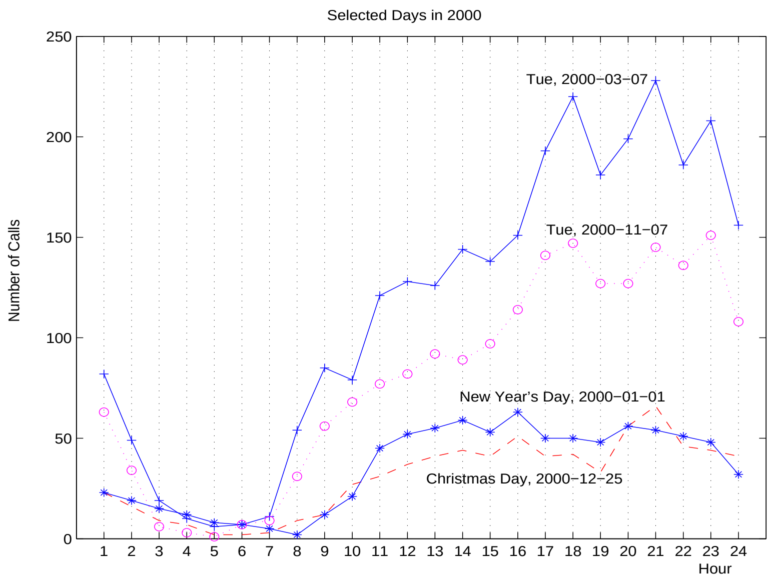


Figure 2.12 Holiday and Weekday Comparison

The influence of time of day on arrivals throughout a day is demonstrated more clearly on January 1st, 2000. Figure 2.13, 2.14 and 2.15 represent three versions of arrivals in the same day, namely by hour, half-hour and quarter-hour.

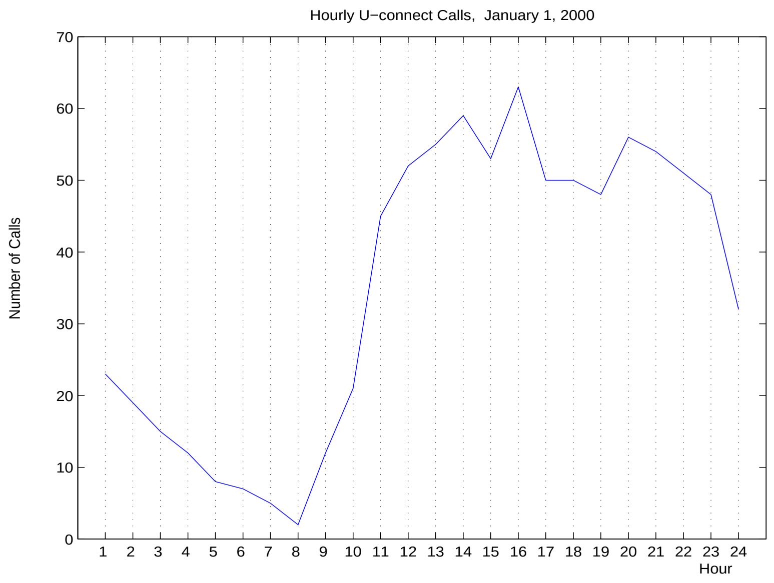


Figure 2.13 Hourly Calls

Once we zoom in, we see more fluctuations of arrivals in shorter intervals moving from hour to quarter-hour. Since our research models the unknown arrivals as controlled by factors such as time of day, day of week and academic season, in the next section, a review of the relevant literature on methods of modelling is conducted for this challenge.

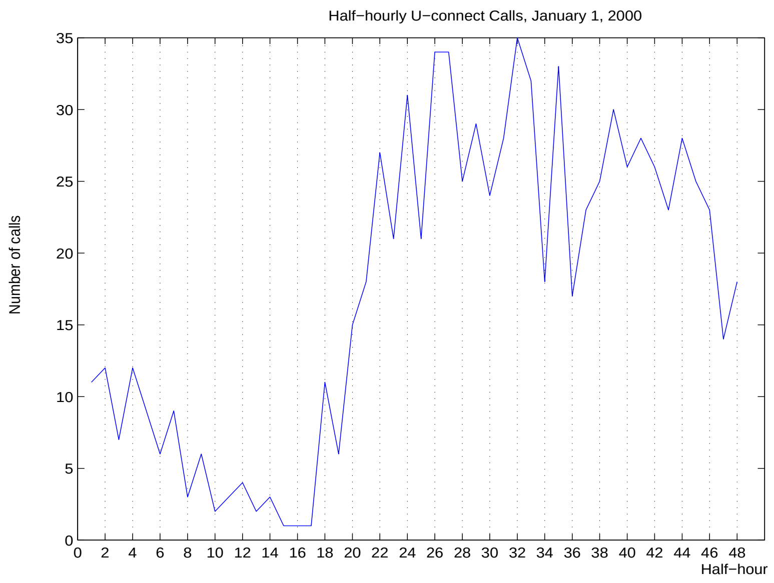


Figure 2.14 Half-hour Calls

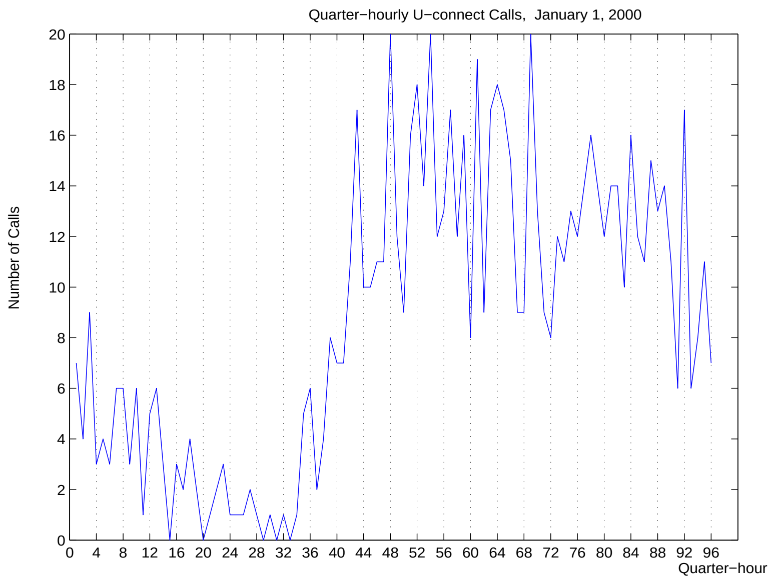


Figure 2.15 Quarter-hour Calls

Chapter 3

METHODS OF MODELLING STOCHASTIC ARRIVALS

This chapter provides a review of the development of modelling a stochastic arrival process at a call centre. As outlined by Whitt [34], it is useful to classify the modelling strategies based on sources of uncertainty: (1) model uncertainty, (2) parameter uncertainty or (3) process uncertainty. One of these three stochastic components can govern another with respect to the arrival process being studied. One would be interested in comparing these components by quantifying their randomness. Recently, Brown et al. [5] investigated the arrival process of an Israeli bank call centre and developed a statistical test to determine if the calls followed a nonhomogeneous Poisson process with a slowly varying arrival rate. In accommodating the unknown parameters of the arrival rate, Massey et al. [27] confined themselves to a linear arrival model. Another approach to tackling model and parameter uncertainty is by employing nonparametric methods. For example, Leemis [23] was interested in nonparametric techniques for estimating the arrival rate if the parent intensity function can be well defined such as, piecewise linear, sinusoidal, power and exponential-like. Similar to Whitt's outline, our review does not attempt to be comprehensive, but rather focuses on the arrival process and the common practice of modelling the arrival rate in the context of call centres.

3.1 A Systematic View From A Queueing Perspective

3.1.1 Characteristics of queueing systems

Danish engineer and mathematician A.K. Erlang, in 1909, applied queueing models to telephone switches [18]. Since then, queueing theory has been used very extensively in various applications. A queueing system can be described using the following characteristics [18]:

1. arrival pattern of customers,
2. service pattern of servers,
3. queue discipline,
4. system capacity,
5. number of service channels,
6. number of service stages.

A queueing system is represented by five symbols namely,

$$A/B/X/Y/Z, \tag{3.1}$$

where

A provides the description of the arrival process in terms of the interarrival times. Some of the commonly used arrival processes are M (*Markovian*), D (*Deterministic*), GI (*Renewal*) and G (*General*).

B describes the service times which can be exponential (M) or deterministic (D) or general (G).

X represents the number of *servers*.

Y is the system capacity restriction which is usually represented by the waiting space.

Z represents the service discipline used in the system. The service discipline could be FCFS (first come first served), LCFS (last come last served) or priority-based.

For example, in an M/M/N/N queue known as the Erlang B model, arrivals form a Poisson process with a constant arrival rate and the service times are assumed to be independently identically distributed exponential random variables. No waiting is possible and the service discipline is FCFS that is usually suppressed in the standard notation .

In practice, however, the queueing system is fairly complex and most of the characteristics are *time-dependent*, in which case we add a t subscript to the standard notations. For example, for the M_t/M/N/N queue [20] the arrival process is nonhomogeneous Poisson and the arrival rate is characterized by a time-dependent arrival rate function.

3.2 Poisson Arrival Model

3.2.1 Poisson process

Since Erlang used the Poisson process in applications of telecommunication systems, it has become prevalent in modelling a great number of arrival processes beyond its use in telecommunication [18]. The Poisson process is an example of a stochastic process known as a counting process. Call arrivals at a call centre can be modelled as a counting process $\{N(t), t \geq 0\}$ which counts the cumulative number of arrivals at time t. If the number of calls in any time interval of length t is governed by a Poisson process with rate λ, then the process satisfies the following:

(i) $N(0) = 0$.

(ii) The process has independent increments, which implies that $[N(t_4) - N(t_3)]$ is independent of $[N(t_2) - N(t_1)]$ for $t_4 > t_3 > t_2 > t_1$.

(iii) The number of arrivals in any interval of length t has a Poisson distribution, i.e.,

$$P\{N(t+s) - N(s) = n\} = \exp(-\lambda t)\frac{(\lambda t)^n}{n!}, \quad n = 0, 1, \ldots \tag{3.2}$$

with the mean and variance both equal to λt. One should also note that immediately following (iii), a Poisson process has stationary increments [31], which means the distribution of the number of arrivals over a given time interval only depends on the length of the interval.

3.2.2 Exponential interarrival time

The Poisson process and the exponential distribution are closely related, i.e., if events occur corresponding to a Poisson process, then the interarrival times between these events are independent and identical (i.i.d.) exponentially distributed random variables [22]. In order to test whether the dial-up session traffic can be modelled by a Poisson process, we investigate if the interarrival times between the sessions are i.i.d. exponentially distributed. Using $f(x)$ and $F(x)$ to denote the probability density and cumulative distribution functions respectively, for an exponential distribution with mean $1/\lambda$ ($\lambda > 0$), we have

$$\begin{aligned} f(x) &= \lambda \exp(-\lambda x) \quad \text{for} \quad x \geq 0 \quad \text{and} \quad 0 \quad \text{elsewhere, and} \\ F(x) &= 1 - \exp(-\lambda x) \quad \text{for} \quad x \geq 0 \quad \text{and} \quad 0 \quad \text{elsewhere.} \end{aligned} \tag{3.3}$$

The variance of an exponentially distributed random variable is $1/\lambda^2$.

3.2.3 Nonhomogeneous Poisson process

It is often assumed that call arrivals follow a homogeneous Poisson process with a constant arrival rate λ. With this assumption, the blocking probability known as Erlang B formula can be obtained using M/M/N/N. This blocking probability is used in practice as one measure of quality efficiency. However, the real situation quite often becomes complicated when the arrival rate is $\lambda(t)$, a function of time t. In the case of operating a call centre during peak hours, the stationary assumption

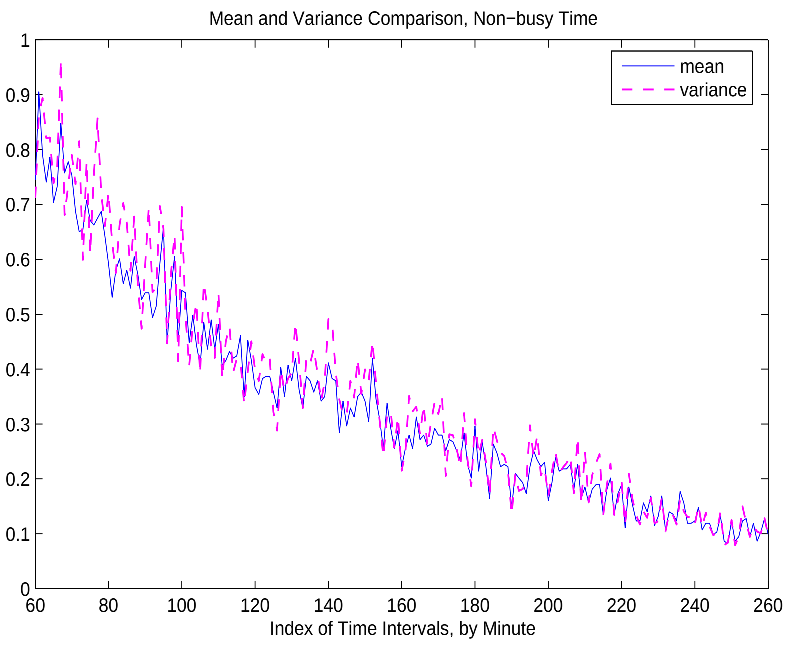


Figure 3.1 Undistinguishable Mean and Variance, Non-busy Time

of a Poisson process can be easily violated. It is likely to observe abrupt changes in arrivals where the constant arrival rate λ is barely in place. On January 1st, 2000, the mean and variance are apparently unequal during the peak hours shown in Figure 3.2 but hardly distinguishable during the low traffic hours shown in Figure 3.1. Given the large variation of call volume in incoming traffic across different time frames, it would be very problematic to schedule staffing in achieving a "no-wait" standard using the traditional M/M/N/N model.

The solution to this problem is to use the *nonhomogeneous Poisson model.* Numerous authors [31], [22] and [32] have advocated the use of the nonhomogeneous Poisson process. The nonhomogeneous Poisson process comes in handy, as an al-

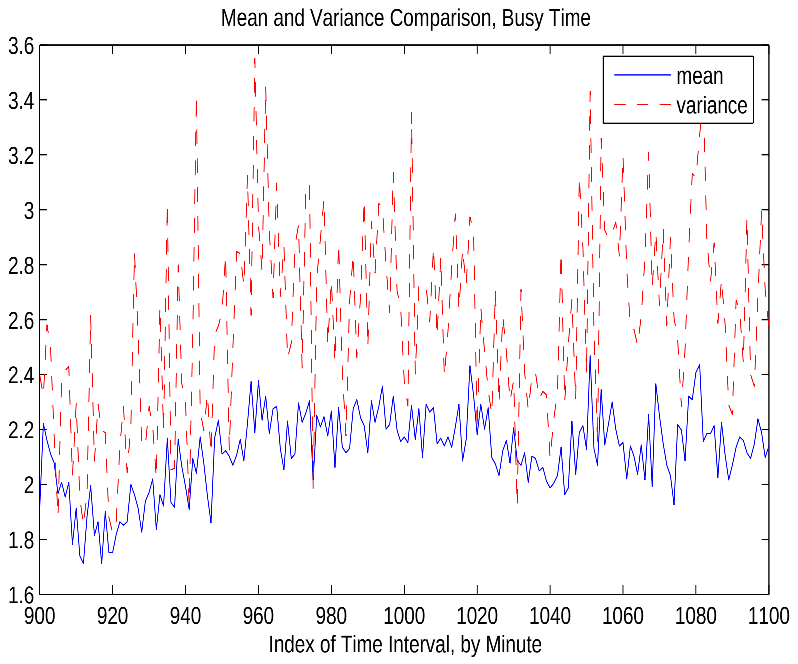


Figure 3.2 Unequal Mean and Variance, Busy Time

ternative to the homogeneous Poisson process, to tackle the time varying property of call arrivals. The nonhomogeneous Poisson process allows us to analyse arrivals by differentiating by time of the day and day of the week. Let $\{N(t), t \geq 0\}$ be a nonhomogeneous Poisson process which counts the number of arrivals up to time t with *intensity function* $\lambda(t)$. Thus it has the following properties:

(i) $N(0) = 0$.

(ii) The process has independent increments as in the homogeneous case.

(iii) The number of arrivals in any interval of length t has a Poisson distribution, i.e.,

$$P\{N(t+s)-N(s)=n\}=\exp\{-[\Lambda(t+s)-\Lambda(s)]\}\frac{[\Lambda(t+s)-\Lambda(s)]^n}{n!}, \quad n=0,1,\ldots \tag{3.4}$$

with mean $\Lambda(t+s)-\Lambda(s)$ where $\Lambda(t)=E[N(t)]=\int_0^t \lambda(\tau)d\tau$ [31]. $E[N(t)]$ is the expected number of arrivals up to t.

The form of the intensity function $\lambda(t)$ is considered one of the key differences between homogeneous and nonhomogeneous Poisson processes. In the case of the homogeneous Poisson process, $\lambda(t)$ is constant, say equal to λ. Then, the expected number arrivals up to t is $E[N(t)]=\int_0^t \lambda(\tau)d\tau=\int_0^t \lambda d\tau=\lambda t$. On the other hand, where arrivals follow a nonhomogeneous Poisson process, $\lambda(t)$, the form of which needs to be determined from the data.

3.3 Parametric Estimation of Intensity Rate

As pointed out by Whitt [34], if the nonhomogeneous model is deemed appropriate, then the concern is whether we can reduce the infinite-dimensional parameter space due to $\lambda(t)$. Many approaches exist in the literature for modelling the time varying arrival rate. While $\lambda(t)$ is nonconstant, the question of interest is if the form of $\lambda(t)$ can be determined by some simple model.

3.3.1 Linear rate model by Massey et al.

If the intensity rate varies proportional to time, it makes good sense to regard this rate as linear across time spans. As proposed by Massey et al. [27], a *piecewise linear* *model* fits well within single hours, especially with the arrivals summarised by counts over short subintervals, say 5 minutes, for the data from the AT & T long distance network. The motivation to approximate it linearly also lies at simplification of the model which reduces the infinite-dimensional parameter space. Of course, the arrival rate of our modem pool data can vary significantly from hour to hour, but the linear rate might have a good fit to the dial-up user arrivals within an hour.

Massey et al. [27] compared three different estimators in fitting the linear rate. They were OLS, IWLS and ML estimators.

Simple linear regression and the OLS estimator

To fit a piecewise linear model over a time span T, one assumes

$$\lambda(t) = a + bt, \quad 0 \le t \le T \tag{3.5}$$

To carry out the estimation procedure, we partition the given time span (say, T) into n equal subintervals with x_i being the midpoint of each subinterval i. Then, we count the number of arrivals y_i in each of these n subintervals where the expected number of arrivals for each interval is [27]

$$E[y_i] = \lambda_i = \int_0^{\frac{T}{n}} \lambda(\tau) d\tau = \frac{T}{n}(a + bx_i). \tag{3.6}$$

For the simple linear regression model $y_i = \alpha + \beta x_i + \epsilon_i$, with the random error $\varepsilon_i \sim N(0, \sigma^2)$, the OLS procedure is often used to estimate the regression coefficients. We would like to present it here in order to compare it with IWLS later. We can compute the estimates $\widehat{\alpha}$ and $\widehat{\beta}$ based on the paired data (x_i, y_i) in the following fashion,

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \text{ and } \overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{3.7}$$

The sums of squares are

$$S_{xx} = \sum_{i=1}^{n}(x_i - \overline{x})^2 \text{ and } S_{yy} = \sum_{i=1}^{n}(y_i - \overline{y})^2, \tag{3.8}$$

and the sum of cross-product is

$$S_{xy} = \sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}). \tag{3.9}$$

$\widehat{\alpha}$ and $\widehat{\beta}$ are given by

$$\widehat{\beta} = \frac{S_{xy}}{S_{xx}} \text{ and } \widehat{\alpha} = \overline{y} - \widehat{\beta}\overline{x}. \tag{3.10}$$

Then, using the means $\lambda_i = \frac{T}{n}(a + bx_i)$ of the Poisson random variables y_i, subject to $\lambda_i \geq 0$, we obtain

$$\widehat{b} = \frac{n}{T}\widehat{\beta} \text{ and } \widehat{a} = \frac{n}{T}\widehat{\alpha}. \tag{3.11}$$

Heteroscedasticity and the IWLS estimator

Heteroscedasticity refers to nonconstant error variance. Recall that for Poisson random variables y_i, mean and variance are equal. In the time nonhomogeneous model, we do not expect constant error variance throughout T when regressing on y_i.

Thus the Gauss-Markov theorem which asserts OLS estimators are the *best linear unbiased estimators* (known as BLUE) is no longer applicable, since the underlying assumption of constant variance is violated. Now the regression model becomes $y_i = \alpha' + \beta' x_i + \epsilon_i$ with ϵ_i distributed with different variance σ_i^2, which deviates slightly from OLS. If the unequal variance structure is known, we use positive weight w_i to minimize the weighted sum of squared residuals, $\sum_{i=1}^{n} w_i \epsilon_i^2$. Since the variance is not known in advance, Massey et al. [27] used the weights that produced the minimum variance estimator amongst linear functions of y_i. Namely, the weights were

$$w_i = \frac{n}{\lambda_i} / \sum_{i=1}^{n} (\frac{1}{\lambda_i}) \text{ with mean of } y_i,\ \lambda_i = \frac{T}{n}(a + bx_i). \tag{3.12}$$

Similarly, α' and β' can be estimated as follows,

$$\overline{x}' = \frac{1}{n} \sum_{i=1}^{n} w_i x_i \text{ and } \overline{y}' = \frac{1}{n} \sum_{i=1}^{n} w_i y_{i.} \tag{3.13}$$

The sums of squares become

$$S'_{xx} = \sum_{i=1}^{n} w_i (x_i - \overline{x}')^2 \text{ and } S'_{yy} = \sum_{i=1}^{n} w_i (y_i - \overline{y}')^2, \tag{3.14}$$

and the sum of cross-product becomes

$$S'_{xy} = \sum_{i=1}^{n} w_i (x_i - \overline{x}')(y_i - \overline{y}'). \tag{3.15}$$

$\widehat{\alpha}'$ and $\widehat{\beta}'$ are given by

$$\widehat{\beta}' = \frac{S'_{xy}}{S'_{xx}} \text{ and } \widehat{\alpha}' = \overline{y}' - \widehat{\beta}'\overline{x}'. \tag{3.16}$$

Then, obtain

$$\widehat{b} = \frac{n}{T}\widehat{\beta} \text{ and } \widehat{a} = \frac{n}{T}\widehat{\alpha}. \tag{3.17}$$

The error variance, a and b were unknown in advance. To estimate parameters a and b using w_i, Massey et al. employed the iterative method (IWLS) to approach w_i. The IWLS procedure is discussed in detail by Carroll and Ruppert [6], and is as follows: First, estimate a and b using OLS. Second, use the estimated a and b to compute estimates for λ_i and w_i from Equation 3.12. Third, use these estimates to initiate the weighted least square approach as described by Equations 3.13, 3.14, 3.15, 3.16 and 3.17. Fourth, iteratively update the estimates for a, b and w_i until a preset error tolerance (i.e., 10^{-6} was used in our codes) for stabilizing w_i has achieved. The iterative procedure usually took at most 5 iterations to converge as reported by Massey et al. [27].

The ML estimator

The third procedure they [27] presented was to estimate a and b so as to maximize the likelihood of the Poisson random variable y_i, $i = 1, 2, \ldots, n$. Let $L(\lambda|y)$ denote the likelihood function with parameter vector $\lambda = (\lambda_i, \ldots, \lambda_n)$. Thus,

$$L(\lambda|y) = \frac{\exp(-\sum\limits_{i=1}^{n} \lambda_i) \prod\limits_{i=1}^{n} \lambda_i^{y_i}}{\prod\limits_{i=1}^{n} (y_i!)}. \tag{3.18}$$

Then, the log likelihood is

$$\ln L(\lambda|y) = -\sum_{i=1}^{n} \lambda_i + \sum_{i=1}^{n} y_i \ln \lambda_i - \sum_{i=1}^{n} \ln(y_i!). \tag{3.19}$$

For the linear rate model, we still replace λ_i by $\frac{T}{n}(a+bx_i)$ and apply the midpoint rule that $x_i = (i-1/2)\frac{T}{n}$. Then, the log likelihood becomes [27]

$$\begin{aligned} \ln L(a,b|y) &= -\sum_{i=1}^{n} \frac{T}{n}(a+bx_i) + \sum_{i=1}^{n} y_i \ln[\frac{T}{n}(a+bx_i)] - \sum_{i=1}^{n} \ln(y_i!) \\ &= -aT - \frac{bT^2}{2} + \sum_{i=1}^{n} y_i \ln[\frac{T}{n}(a+bx_i)] - \sum_{i=1}^{n} \ln(y_i!). \end{aligned} \tag{3.20}$$

Taking advantage of the monotonic increasing property of the log function, Massey et al. [27] obtained the ML estimates $\widehat{a}^M$ as solution to Equation 3.21 and $\widehat{b}^M$ by the following scheme, with

$$g(a) \equiv \sum_{i=1}^{n} \frac{y_i}{a\overline{x} + x_i(\frac{S}{T} - a)} = 2 \text{ where } S \equiv \sum_{i=1}^{n} y_i \tag{3.21}$$

(i) find the root a of $g(a) = 2$ in $(0, S/T)$, $b = 2(S - aT)/T^2$;

(ii) if no root is found, then $a = a^* + bT, b = -b^*$ for the root a^* of $g^*(a) = 2$ in $(0, S/T)$ and $b^* = 2(S - a^*T)/T^2$ where g^* is g in Equation 3.21,

(iii) if no root can be found in (i) or (ii), evaluate Expression 3.22 with solutions (1) $a = S/T$ and $b = 0$, (2) $a = 0$ and $b = 2S/T^2$ and (3) $a = -bT$ and $b = -2S/T^2$

$$\sum_{i=1}^{n} y_i \ln(a + bx_i) \tag{3.22}$$

and choose the one that produces the maximum.

Asymptotic likelihood ratio test for linear rate

In addition to estimating the parameters, we can rely on the asymptotic *likelihood ratio test* (LRT) for testing the linear rate model. The rationale behind the LRT is that we would like to evaluate the deviance of the hypothesized model from a more general model by the ratio of their likelihood in order to test

$$\begin{aligned} H_0 &: \quad \theta \in \Theta_0 \\ H_a &: \quad \theta \in \Theta_0^C. \end{aligned}$$

where $\Theta_0 \subseteq \Theta$ and Θ is the general parameter space.

The numerator of the ratio is the maximum probability of the observed sample under the null hypothesis, whereas the denominator is the maximum probability of the observed sample over all possible parameters. If this ratio is small, then given the data it is less likely to accept the null hypothesis than the alternative. To use the asymptotic distribution, we let the sample size go to infinity, which allows us to simplify calculations or to acquire properties that arise from large samples. In our case, we do not know any particular test designed to assess the fitness of the linear rate model. The asymptotic distribution of the likelihood ratio of the model in question over all possible models gives us an explicit expression in doing so. The likelihood ratio is defined by

$$l(\mathbf{x}) = \frac{\sup_{\Theta_0} L(\theta|\mathbf{x})}{\sup_{\Theta} L(\theta|\mathbf{x})} \tag{3.23}$$

where $\sup_{\Theta_0} L(\theta|\mathbf{x})$ is the supremum of the likelihood over the hypothesized parameter space (i.e., Θ_0). Then, by the following theorem, the distribution of the test statistic $-2\ln l(\mathbf{x})$ converges to a χ^2 distribution as the sample size $n \to \infty$ [3].

Theorem 1 (Berger and Casella, 2002) *Let $X_1, \ldots, X_n$ be a random sample from a pdf or pmf $f(x|\theta)$. Under the regularity conditions (see Miscellanea 10.6.2 in [3]),*

$$\textit{Reject } H_0 : \theta \in \Theta_0 \textit{ if and only if } -2\ln l(\boldsymbol{X}) \geq \chi^2_{\nu,\alpha} \tag{3.24}$$

where the degrees of freedom ν is the difference between the number of free parameters specified by $\theta \in \Theta_0$ and the number of free parameters specified by Θ.

In the language of statistical testing, our hypothesis can be formulated by

$$H_0 \;:\; \lambda_i = \frac{T}{n}(a + bx_i) \tag{3.25}$$

$$H_a \;:\; \text{Otherwise.} \tag{3.26}$$

To find the numerator $\sup_{\Theta_0} L(\theta|\mathbf{x})$, we can use $\widehat{a}^M$ and $\widehat{b}^M$ for the linear rate, which can be calculated from the ML scheme given earlier. Similarly, to find the denominator $\sup_\Theta L(\theta|\mathbf{x})$, we need to find the ML estimator of λ_i under the full model. It means we differentiate the log likelihood $\ln L(\lambda|y_i's)$ in Equation 3.19 with respect to λ_i. Then,

$$\frac{\partial}{\partial\lambda_i}\ln L(\lambda|y) = \frac{\partial}{\partial\lambda_i}[-\sum_{i=1}^{n}\lambda_i + \sum_{i=1}^{n} y_i \ln\lambda_i - \sum_{i=1}^{n}\ln(y_i!)] = -1 + \frac{y_i}{\lambda_i}. \tag{3.27}$$

After setting the equation to zero, we find y_i is the ML estimator of λ_i. The resulted test statistic having χ^2 distribution with degrees of freedom $\nu = n-2$ becomes

$$-2\ln l(\mathbf{X}) = -2\ln\left(\frac{\exp[-\sum_{i=1}^{n}\frac{T}{n}(\widehat{a}^M + \widehat{b}^M x_i)]\prod_{i=1}^{n}\frac{T}{n}(\widehat{a}^M + \widehat{b}^M x_i)^{y_i}}{\exp(-\sum_{i=1}^{n} y_i)\prod_{i=1}^{n} y_i^{y_i}}\right). \tag{3.28}$$

For a large sample, the test statistic approximates a χ^2 distribution. What happens if our sample size is small? For testing one parameter, McCullagh and Nelder pointed out that this large sample approximation is usually quite accurate even for small samples [28]. We apply LRT to our data using this approximation and the results are presented in Section 5.1.2, Chapter 5.

3.3.2 Sinusoidal model

A natural question is what if the arrival rate is not linear. It is likely that arrivals oscillate very much even within an hour. In this case, the rate is too fluctuating to be captured by the linear model. As cited in Leemis [23], attempts such as using the so-called power law or Weibull process

$$E[N(t)] = \Lambda(t) = (\alpha t)^\beta \tag{3.29}$$

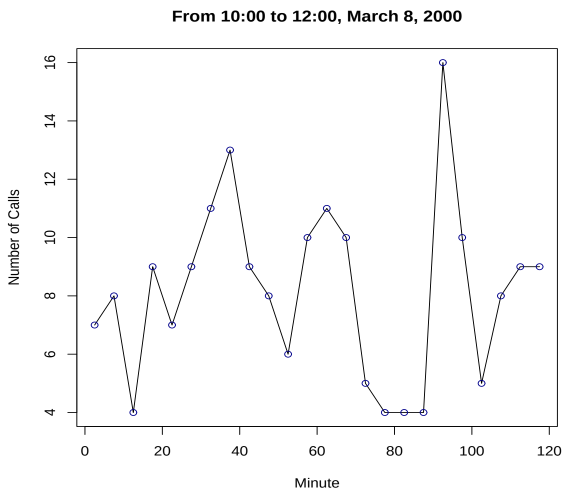


Figure 3.3 An Example Beyond Linear Rate Model, 10:00am-12:00am

have been made. A general model extends the power law to an exponential-polynomial-trigonometric function. The general model for intensity rate function which is suggested by Crawford et al. [8], has the form

$$\lambda(t) = \exp[\sum_{i=1}^{m} \alpha_i t^i + \gamma \sin(\varpi t + \phi)]. \tag{3.30}$$

Figure 3.3 presents us a situation beyond the capacity of a linear rate model. The figure shows a 2-hour call arrivals on Monday, March 8, 2000. Each data point represents the number of arrivals within 5 minutes, spanning from 10:00am to 12:00am. Roughly speaking, within each hour, there is a global minimum of 4 arrivals each indexed by 12 and 87, meaning 2 valleys occur around 10:12am and 11:17am. Similarly, a global maximum, at least each within 1 hour span, of 16 and 13 arrivals, indexed by 37 and 92, suggesting 2 peaks are at 10:37am and 11:32am. Both valleys and peaks are almost 1 hour apart. Regarding this hourly pattern which Figure 3.3 exhibits, a *sinusoidal* model can do well in capturing the fluctuations. We can set a cycle of 1 hour, to model the cyclic behaviour of the arrivals. The sinusoidal model, which we propose, has the following form

$$\lambda(t) = a + b_1 \sin(ct) + b_2 \cos(ct). \tag{3.31}$$

Although the general form as in Equation 3.30 seems more flexible, difficulties arise when one attempts to estimate the model parameters. The sinusoidal form has a couple of advantages over the general one:

- It preserves the properties that we have developed from the linear rate model: it is still linear in terms of the trigonometric dependent variables; since it is linear, we can apply the OLS, IWLS and MLE procedures to estimate the intercept and coefficients using the linear regression model

$$y_i = a + b_1 \sin(cx_i) + b_2 \cos(cx_i) + \epsilon_i. \tag{3.32}$$

- The cyclic behaviour can be easily modelled by adjusting parameter 'c'. The sinusoidal rate has the cycle $2\pi/c$. As shown previously in Figure 3.3, if we let $c = 2\pi/60$, then the sinusoidal model will approximate the arrivals with a cycle of 1 hour (i.e., 60 minutes).

It is common that one breaks the cycle into subintervals, say 5 minutes each, so one can estimate the parameters based on the sample within that cycle. Then, one can extend the procedure to the subsequential cycles. Green, Kolesar and Soares [16] applied a sinusoidal model of the form $\lambda(t) = \lambda + A\sin(2\pi t/24)$ with a 24-hour cycle, because of the many applications where a daily cyclic effect is evident. In contrast, we sample a 2-hour realization simply to demonstrate the cyclic effect of arrivals. We can still use the sinusoidal rate to model any other time span as long as the cyclicity can be well captured by setting the period parameter c in the model.

The sinusoidal ML estimator

As has been noted, the OLS and IWLS procedures for the sinusoidal model are exactly the same as for the linear rate model. However, we need to compute the ML estimates. As in Equation 3.19, we replace λ_i by $\frac{T}{n}(a + b_1 \sin cx_i + b_2 \cos cx_i)$, the log likelihood becomes

$$\begin{aligned}
\ln L(a, b|y) &= -\sum_{i=1}^{n} \frac{T}{n}(a + b_1 \sin cx_i + b_2 \cos cx_i) \\
&\quad + \sum_{i=1}^{n} y_i \ln\left[\frac{T}{n}(a + b_1 \sin cx_i + b_2 \cos cx_i)\right] - \sum_{i=1}^{n} \ln(y_i!) \\
&= -aT - \frac{T}{n}(b_1 \sum_{i=1}^{n} \sin cx_i + \sum_{i=1}^{n} \cos cx_i) \\
&\quad + \sum_{i=1}^{n} y_i \ln\left[\frac{T}{n}(a + b_1 \sin cx_i + b_2 \cos cx_i)\right] - \sum_{i=1}^{n} \ln(y_i!). \qquad (3.33)
\end{aligned}$$

After differentiating the log likelihood with respect to a, b_1 and b_2 and setting the results to zero, we obtain equations

$$\sum_{i=1}^{n} \frac{y_i}{a + b_1 \sin cx_i + b_2 \cos cx_i} = T \qquad (3.34)$$

$$\sum_{i=1}^{n} \frac{y_i \sin cx_i}{a + b_1 \sin cx_i + b_2 \cos cx_i} = \frac{T}{n} \sum_{i=1}^{n} \sin cx_i \qquad (3.35)$$

$$\sum_{i=1}^{n} \frac{y_i \cos cx_i}{a + b_1 \sin cx_i + b_2 \cos cx_i} = \frac{T}{n} \sum_{i=1}^{n} \cos cx_i. \tag{3.36}$$

In contrast to the linear rate case, analytically solving this system of equations raises a number of technical difficulties. However, the solutions can be computed numerically. We use **fsolve** in MATLAB to solve for the ML estimates $\widehat{a}^M$, $\widehat{b}_1^M$ and $\widehat{b}_2^M$ (see Appendix G for a review of the numerical algorithm used by **fsolve**).

Asymptotic likelihood ratio test for sinusoidal rate

Similar to OLS and IWLS procedures, the asymptotic LRT for sinusoidal rate preserves the structure of the linear rate case. In other words, the hypothesis becomes

$$H_0 \;:\; \lambda_i = \frac{T}{n}(a + b_1 \sin cx_i + b_2 \cos cx_i) \tag{3.37}$$

$$H_a \;:\; \text{Otherwise.} \tag{3.38}$$

The corresponding test statistic $-2\ln l(\mathbf{X})$ equals

$$-2\ln\left(\frac{\exp[-\sum_{i=1}^{n} \frac{T}{n}(\widehat{a}^M + \widehat{b}_1^M \sin cx_i + \widehat{b}_2^M \cos cx_i)] \prod_{i=1}^{n} \frac{T}{n}(\widehat{a}^M + \widehat{b}_1^M \sin cx_i + \widehat{b}_2^M \cos cx_i)^{y_i}}{\exp(-\sum_{i=1}^{n} y_i) \prod_{i=1}^{n} y_i^{y_i}}\right) \tag{3.39}$$

with the degrees of freedom $\nu = n - 3$, where $\widehat{a}^M$, $\widehat{b}_1^M$ and $\widehat{b}_2^M$ are the ML estimates and can be obtained numerically.

3.4 Spline Models

Spline regression is a method for fitting and smoothing the twists and turns (local characteristics) of a time varying process. We choose this approach because it is the simplest when the *knots* are few and known in advance. Note that the number of arrivals are summarised by every 5 minutes or so. If we treat each 5-minute point on the time span as a knot, and the subintervals between the knots are fixed (e.g., 5 minutes), then we can model the arrivals using splines.

3.4.1 An introduction to splines

The method of *splines* is traditionally more familiar to numerical analysis than to be used by statisticians. We can interpolate or extrapolate data using splines. We would like to review data interpolation using splines here in order to appreciate its nice properties. A spline is a segmented polynomial separated by knots, an ordered set of points $\{\xi_i\}$. The segmented nature allows splines more flexibility than a simple polynomial to adjust the local characteristics of data. Also, splines have a set of certain continuity properties at the local points. A more precise definition of a spline of order r with knots at $\xi_1, \ldots, \xi_k$ is given by Eubank [11],

$$s(t) = \sum_{i=0}^{r-1} \theta_i t^i + \sum_{i=1}^{k} \delta_i (t - \xi_i)_+^{r-1} \tag{3.40}$$

for some set of real coefficients θ_0 ,. . . , θ_{r-1}, δ_1 ,. . . , δ_k, where

$$(t-\xi_i)_+^{r-1} = \begin{cases} (t-\xi_i)^{r-1}, & \text{t} \geq \xi_i, \\ \\ 0, & \text{t} < \xi_i. \end{cases} \tag{3.41}$$

This definition is also equivalent to the following conditions,

1. s is a piecewise polynomial of order r on any subinterval $[\xi_i, \xi_k]$.

2. s has $r-2$ continuous derivatives.

3. s has an $(r-1)$st derivative that is a step function with jumps at ξ_1 ,...,ξ_k.

Restrained by the above conditions, a spline is indeed a piecewise polynomial whose different polynomial segments have been joined together at the knots $\xi_1, \ldots, \xi_k$.

3.4.2 Cubic spline interpolation

A *cubic spline*, for example, by the definition above can be expressed by

$$s(t) = \sum_{i=0}^{3} \alpha_i t^i + \sum_{i=1}^{N-1} \beta_i (t-\xi_i)_+^3 \tag{3.42}$$

for a partition on $[a, b]$, $a = t_0 < t_1 < \ldots < t_N = b$, where

$$(t-\xi_i)_+^{3} = \begin{cases} (t-\xi_i)^{3}, & \text{t} \geq \xi_i, \\ \\ 0, & \text{t} < \xi_i. \end{cases} \tag{3.43}$$

Given values $y(\xi_1), \ldots, y(\xi_N)$ at the knots, we choose the coefficients α_i ($i = 0, 1, 2$ and 3) and β_i ($i = 0, 1, \ldots, N-1$) to satisfy the continuity conditions such that,

$$s^-(\xi_i) \;=\; y(\xi_i) = s^+(\xi_i) \tag{3.44}$$

$$s^{-'}(\xi_i) = s^{+'}(\xi_i) \tag{3.45}$$

$$s^{-''}(\xi_i) = s^{+''}(\xi_i) \text{ for } i = 1, \ldots, N \tag{3.46}$$

where $s^-(\xi_i)$ and $s^+(\xi_i)$ are the splines to the left and right of the knot ξ_i, respectively. That is to say, cubic splines are smooth curves by forcing the first and second derivatives of the function to agree at the knots.

A *natural cubic spline* imposes zero first and second derivatives at the boundary points $x = a$ and b. This natural boundary condition ensures the fitted curve becomes linear beyond the boundary, which avoids wild behaviour of the curve. Green and Silverman [17] also provided proofs of its attractive properties regarding smoothness in detail. The properties can be summarised by the following two points:

1. Amongst all curves f that interpolates the data, a natural cubic spline minimizes $\int_a^b (f''(x))^2$, which quantifies the roughness of the curve. (*minimum property*)

2. Provided $N \geq 2$, there is exactly one such cubic spline. (*uniqueness property*)

As a result, we can always find a unique minimizer of $\int f''^2$ which interpolates any given set of data. Our data are arrival counts generated by a stochastic process, and this result provides us with the natural cubic spline regression to model the arrival rate which does not exhibit wild fluctuations.

3.4.3 Spline regression model

Replacing the linear function $a + bt$ in Equation 3.5 by a nonlinear function $f(t)$ becomes,

$$\lambda(t) = f(t). \tag{3.47}$$

Instead of estimating model parameters in a linear model, we are interested in modelling $f(t)$ itself. If we model $f(t)$ using a natural cubic spline, then the corresponding natural cubic spline regression model is

$$y_i = \sum_{i=0}^{3} \alpha_i t^i + \sum_{i=1}^{N-1} \beta_i (t - \xi_i)_+^3 + \varepsilon_i \tag{3.48}$$

with the random error $\varepsilon_i \sim N(0, \sigma^2)$.

Spline regression helps relax the assumption of linearity, but may raise difficulty in interpreting the model, whereas parametric models often allow us to explain the response variable by the rate of change of the independent variables. Due to its flexibility and continuous properties, the natural cubic spline can be a good choice. While the function $f(t)$ usually satisfies some conditions of differentiability, $f(t)$ cannot be any continuous function. For example, there are some drawbacks when fitting a polynomial function. Green and Silverman [17] listed two major drawbacks:

1. Some observed data can exert influential effects on the remote part of the polynomial in an unexpected way.

2. Adjusting the degree of polynomial in order to fit the data can only be controlled in discrete steps but not continuously.

In short, polynomial regression can be useful in many situations. However, the choice of terms is not always obvious, and small effects can be greatly magnified or lost completely by the wrong choice.

3.4.4 Smoothing splines

In contrast to natural cubic spline regression, *smoothing splines* arises from the roughness penalty approach. For n pairs (x_i, y_i), a smoothing spline balances the trade-off between the fit and degree of smoothness of the function for the *penalized residual sum of squares*

$$SS(\gamma) = \sum_{i=1}^{n} w_i[y_i - f(x_i)]^2 + \gamma \int_a^b (f''(x))^2 dx \tag{3.49}$$

where γ is the smoothing parameter. That is, we fit a curve $f(t)$ to the data subject to the constraint above.

The first term is the usual weighted sum of squares of errors. As alluded to previously, the second term is the roughness penalty. If $f(t)$ is globally linear, then it contributes nothing after twice differentiation to the roughness penalty term, so the roughness penalty approach is an extension of the least square estimation method used in the linear regression. The addition of the roughness penalty guards against any rapid change of the fitted curve. The smoothing parameter γ is the lever that balances between fitness and smoothness. The larger the smoothing parameter, the more weight on the smoothness. If $\gamma = 0$, then $f(t)$ simply interpolates the data with completely flexible slope that may produce extreme roughness. At the other extreme, if $\gamma \to \infty$, then $f(t)$ should be chosen so that $f^{''}(t) = 0$ everywhere, which is simply a least square linear regression line.

Choosing smoothing parameter by cross validation

In R (a statistical computing package) [29], we can specify smoothing parameter γ or equivalently the degrees of freedom by setting **df** in function **smooth.spline** or **sreg**. Alternatively, we can employ *cross validation* (CV) to choose γ. Cross validation applies a leave-one-out approach. Omitting the ith observation at x_i, we find the resulting curve $\widehat{y}_{-i}$. Then the cross validation is

$$CV(\gamma) = \sum_{i=1}^{n} w_i[y_i - \widehat{y}_{-i}(\gamma, x_i)]^2. \tag{3.50}$$

The key idea is we choose the smoothing parameter that can best predict the data. The best γ helps us minimize the mean square error. By default, R uses *generalized cross validation* (GCV) which is a modified version of CV (see [17] for a substantial discussion on the difference between the two criteria).

3.4.5 Relationship between spline models

The major difference between spline regression (e.g., implemented by **ns** in R) and smoothing spline lies at the eigenvalues of the *hat matrix*. The hat matrix is analogous to the one in the usual linear model. As in the linear model (Equation 3.5), the fitted value given by the regressors can be expressed by [14]

$$\widehat{\mathbf{y}} = \mathbf{X}\mathbf{b}$$

$$\begin{aligned} &= \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \\ &= \mathbf{H}\mathbf{y} \end{aligned} \tag{3.51}$$

where $H = X(X'X)^{-1}X'$ is called the hat matrix. The hat matrix projects y into the subspace spanned by the columns of the model matrix X. For regression spline, all eigenvalues are 0 and 1. It is a projection. In contrast, smoothing spline has a projection part as well as a shrinking part, which makes its eigenvalues vary between 0 and 1. Discussions on hat matrix for splines can be found in books by Eubank [11], Green and Silverman [17], and Hastie and Tibshirani [19]. A comparison of results between different spline models is presented in Chapter 5.

Chapter 4
GOODNESS-OF-FIT TESTS OF THE ARRIVAL PROCESS

In the statistical context of hypothesis testing, we usually employ the so-called *goodness-of-fit* tests to assess how well the data follows the proposed distribution. In this chapter, we recapitulate some goodness-of-fit methods related to the exponential distribution when applied to the nonhomogeneous Poisson process and a statistical testing scheme for independence. We focus on the *empirical distribution function* (EDF) based tests. The null hypothesis H_0 is proposed for a particular distribution, and then the test statistics are computed from the available data to measure the discrepancy of its EDF from the theoretical *cumulative distribution function* CDF of the proposed model. Results by graphical methods as a visual aid to the numerical tests are also provided.

4.1 Testing the Exponential Distribution

To test if the unknown population comes from some exponential distribution, we make the decision by employing a *test of significance.* The null hypothesis is that the distribution of our sample data, say Y_i, is exponential. Let F denote the underlying distribution of the sample data. Then, we test the null hypothesis over the alternative

$$H_0 \quad : \quad F \in \{F_0 : F_0(x) = 1 - \exp^{-\lambda x}; \lambda > 0\}$$

$$H_a \quad : \quad \text{Otherwise}$$

where F_0 in this case is the CDF that belongs to a family of one-parameter exponential distributions.

In this section, we consider various methods of goodness-of-fit when applied to exponential interarrival times, such as Chi-squared, EDF based *Kolmogorov-Smirnov* (K-S) and *Anderson-Darling* (A-D) tests. We do not thoroughly explore our data set by the Chi-squared test. One of the Chi-square's difficulties is matching the EDF: To calculate the Chi-square distributed test statistic, we need to bin the data into arbitrary subintervals. However, the length of bin is usually hard to set. Nonetheless, the algorithm for the Chi-squared test is written in MATLAB (See Appendix E for the implementation).

4.1.1 Anderson-Darling test

To test whether the sample data are exponentially distributed with mean λ, we use the (A-D) test, which checks if a given sample is drawn from a population with a specified distribution [1, 2]. While K-S test is commonly used, A-D test is more appropriate in testing the exponential distribution since A-D does not require the knowledge of the true population parameters, but uses those estimated from the data [1, 2]. A-D takes the advantage of the specified distribution as shown in Equation 4.1. A-D also picks up the difference at the tails better than K-S since A-D assigns more weight to larger values [9]. Another drawback of K-S is that it would not compute the correct test statistics if there are any ties. We only need to implement A-D as

	0.900	0.950	0.975	0.990
Test statistics	1.070	1.326	1.587	1.943

Table 4.1 A-D critical values

K-S is an existing function of Splus and MATLAB. Results tested by K-S are still presented for comparison purposes.

Let $Y_{(1)}, Y_{(2)}, ..., Y_{(n)}$ be the sample values sorted in ascending order. F_0 denotes the CDF of the exponential random variables where $\widehat{\lambda} = 1/\bar{Y}$, $\bar{Y} = \sum_{i=1}^{N} Y_i/N$ and N is the sample size. The test statistic A^2 is defined by

$$A^2 = -N - \sum_{i=1}^{N} \frac{2i-1}{N} \left\{ \ln[F(Y_i)] + \ln[1 - F(Y_{N+1-i})] \right\}. \tag{4.1}$$

As A-D test uses the estimated mean, A^2 has to be multiplied by a constant correction factor [24], so the corrected statistic is

$$A_*^2 = A^2(1 + 0.6/N). \tag{4.2}$$

Table 4.1 from Kelton and Law [22] provides a set of critical values for the adjusted A-D test statistic. For convenience in coding, we only implemented the test for the 5% significance level. That is, the null hypothesis that the sample is drawn from an exponential distribution is rejected if $A_*^2 \geq 1.326$.

4.1.2 Quantile-quantile plots

A useful graphical method, *quantile-quantile* (Q-Q) plot can help us carefully examine the decision when the numerical test shows a rejection of the null hypothesis. The Q-Q plot is the plot of the sample quantiles $Y_{(i)}$ as in Section 4.1.1 against theoretical quantiles of the specified distribution. Loosely speaking, if we see these points lie at a fairly straight line on the Q-Q plot, then we can conclude a good fit of the sample data to the hypothesized exponential distribution. In addition to the Q-Q plot, there is another plot, the percentage-percentage (P-P) plot, which is the plot of the CDF of one distribution against another. However, Wilk and Gnanadesikan had pointed out some limitations about P-P plot [9], and we do not use this plot in our analysis.

4.1.3 Transformed test statistic R_{ij}

Brown et al.[5] suggested that if the intensity rate function $\lambda(t)$ varies smoothly, then one could regard this rate as a piecewise constant step function over a number of short intervals. Therefore, we test the piecewise constant arrivals by hypothesizing

$$\begin{aligned} H_0 &: \lambda(t) \text{ is constant within a short length of time.} \\ H_a &: \text{Otherwise.} \end{aligned}$$

To test this hypothesis, we use the transformed test statistic R_{ij} proposed by Brown et al.[5]. We choose I blocks of equal time-length L. These I blocks can be the same time on various days or successive blocks on a given day. Let T_{ij} be the jth ordered arrival time in an ascending order in the ith block, $i = 1, ..., I$ and $j = 1, ..., J(i)$ where $J(i)$ is the total number of arrivals in the ith block. Let $T_{i0} = 0$ and

$$R_{ij} = [J(i) + 1 - j] \left[-\ln \left(\frac{L - T_{ij}}{L - T_{ij-1}} \right) \right]. \tag{4.3}$$

If the null hypothesis that we have a constant arrival rate within time interval of length L is correct, then we should have independent standard exponential variables R_{ij}. In testing this standard exponential distribution, we use both K-S and A-D tests. In addition, we show the corresponding Q-Q plots. The results are summarised in Section 4.2.

We perform our tests by varying L, progressively. We start testing the time intervals summarised by 1 minute, because in 1 minute the arrival rate of the dial-up users is very likely to be constant. Multiple 1-minute intervals for the same time are used. If we have a good fit, of course statistically, then we continue testing for the length of 1, 5 and 10 minutes until any lack of fit appears. We chose these intervals based on the test results shown in Section 4.2.1. One might question whether or not the length depends on the intensity of traffic. Nevertheless, our algorithm can be easily adapted to any length of time (i.e., as accurate as in seconds) as written in script **MyTestData** in Appendix B.

4.2 Results and Discussions

We first test the null hypothesis that the interarrival time conforms to an independent exponential distribution. Then, we test if the transformed test statistics R_{ij} are independent standard exponential random variables. To test each of these hypotheses, we execute our plan of testing at two levels based on the time varying properties that we have explored in Chapter 2.

Level I: we show the results tested on the aggregate data of year 2000.

Level II: we test the three typical months from the three academic terms in 2000, namely, March, July and November.

We also present the results based on the intensity of traffic as in Figure 3.1 and 3.2 in Chapter 2. In particular, we take two data sets, 1-hour from 1:00am and 4:00pm, respectively to represent our low and high traffic, respectively. Thus, we can easily extend this plan of testing to any other years.

4.2.1 Testing exponential interarrival times

Distribution of the aggregated year 2000 data

Figures 4.1, 4.2, 4.3 and 4.4 each show the density of interarrival times, for either 1 minute or 10 minutes from the target time. We stop our test at 10 minutes where a sign of lack of fit shows (e.g., See Table 4.2 for a summary of K-S and A-D tests). These figures are based on the aggregated 2000 data. It is difficult to interpret if any of these figures looks like the exponential distribution.

Figures 4.1 and 4.2 are the densities for low traffic (i.e., 1:00) whereas Figure 4.3

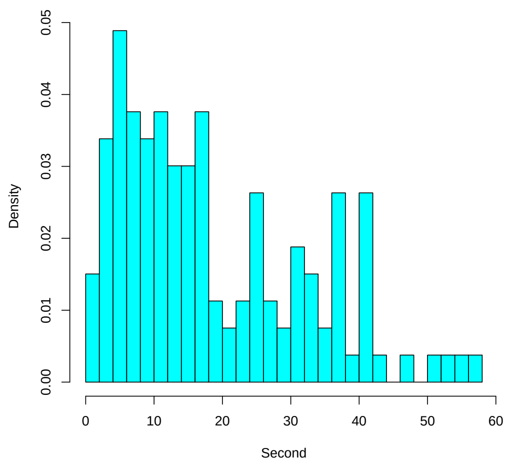


Figure 4.1 Density for Low Traffic, 1 minute

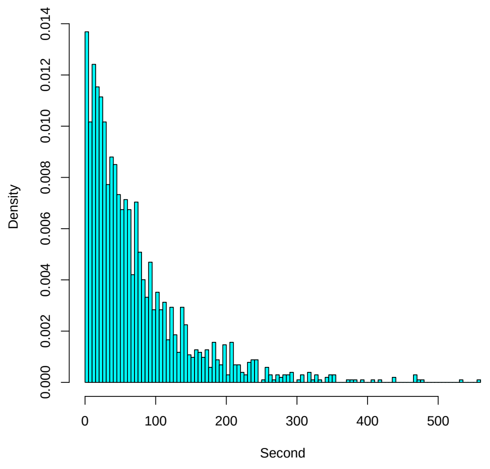


Figure 4.2 Density for Low Traffic, 10 minutes

and 4.4 show the densities for high traffic (i.e., 4:00). One interesting finding is the exponential-like figure seems to be related to the sample size, regardless of traffic intensity. The larger the sample size, the more exponential the density plot looks like.

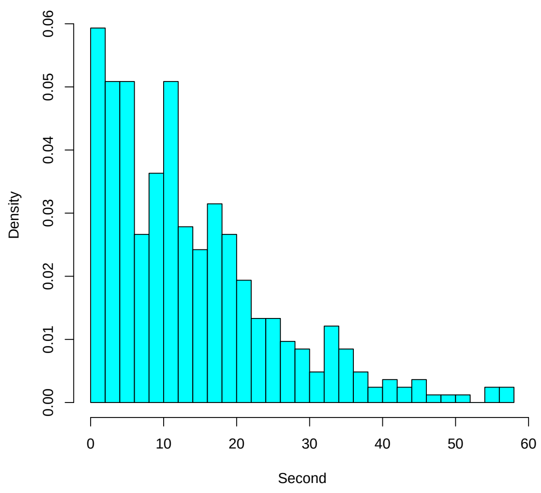


Figure 4.3 Density for High Traffic, 1 minute

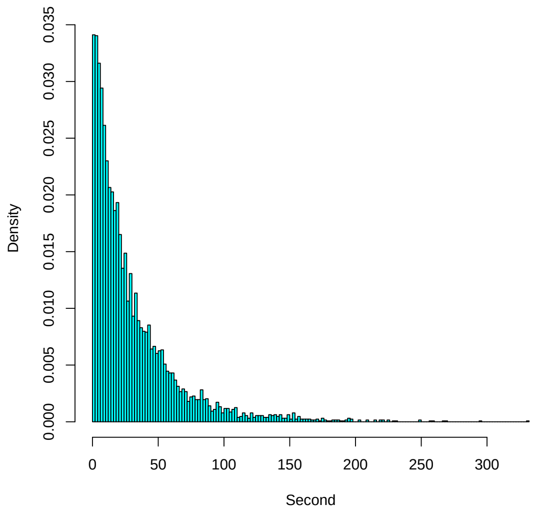


Figure 4.4 Density for High Traffic, 10 minutes

Test summary, low traffic: As proposed by Floyd and Paxson [13], by replacing the unknown population mean by the sample mean we apply the A-D and K-S tests. If the interarrival times are exponentially distributed with the same mean, we should see that the A-D and K-S tests accept the null hypothesis. Table 4.2 summarises the A-D and K-S test results for low traffic. K-S accepts H_0 but the K-S test statistic may not be accurate as there are always ties in all our tests for any large samples (i.e., for $N > 30$). Neither 1-minute nor 10-minute interval accepts the null hypothesis by A-D.

Q-Q plots, low traffic: The Q-Q plots confirm the rejection of the exponential distribution as either the very large and very small quantiles shows significant deviations from the straight line in Figures 4.5 and 4.6. Note that 95% confidence intervals using bootstrapping (see Fox [14] for details) marked by dotted lines in the Q-Q plots

	1 min	10 min
N	133	2046
K-S (p-value)	0.1222	0.4211
A-D	Reject H_0	Reject H_0

Table 4.2 Aggregate Low Traffic

are provided too.

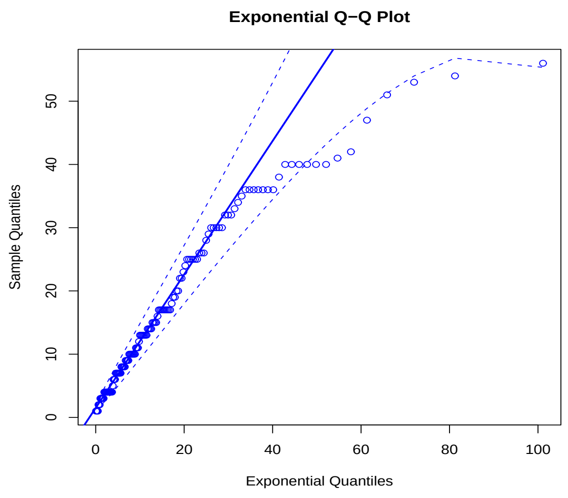


Figure 4.5 Q-Q Plot for Low Traffic, 1 Minute

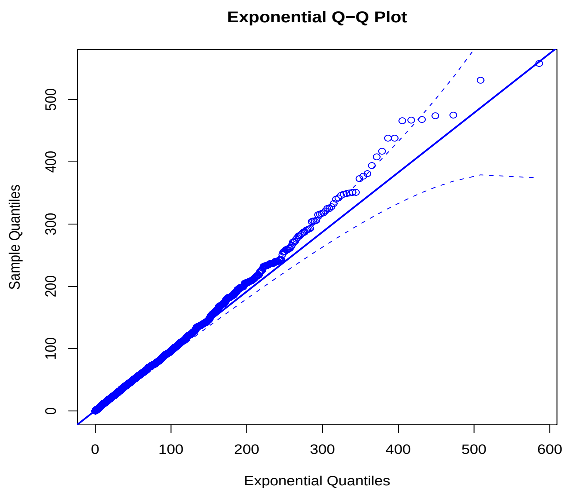


Figure 4.6 Q-Q Plot for Low Traffic, 10 Minutes

Test summary, high traffic: When we look at the A-D and K-S test results tabulated in Table 4.3, we find none of these sample conforms to the exponential distribution. We report the p-value with 2 significant digits only. When the p-value is too small, less than 10^{-3}, we denote this value by '$0 * **$'.

	1 min	10 min
N	413	6390
K-S (p-value)	0.0050	0***
A-D	Reject H_0	Reject H_0

Table 4.3 Aggregate High Traffic

Q-Q plots, high traffic: The Q-Q plots reject the null hypothesis. A great number of points are off the lines shown in Figures 4.7 and 4.8. However, the deviations are very difficult to quantify.

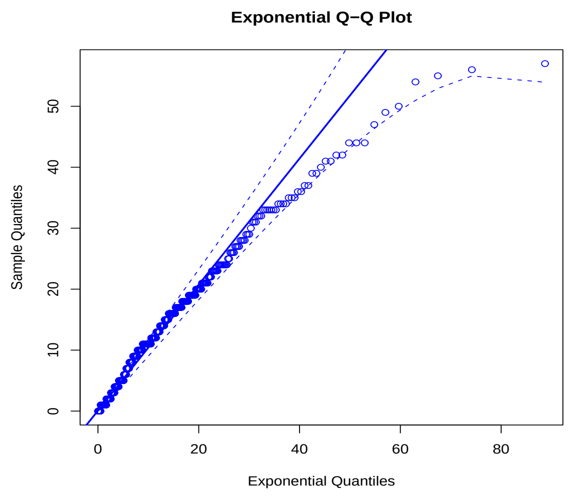


Figure 4.7 Q-Q Plot for High Traffic, 1 Minute

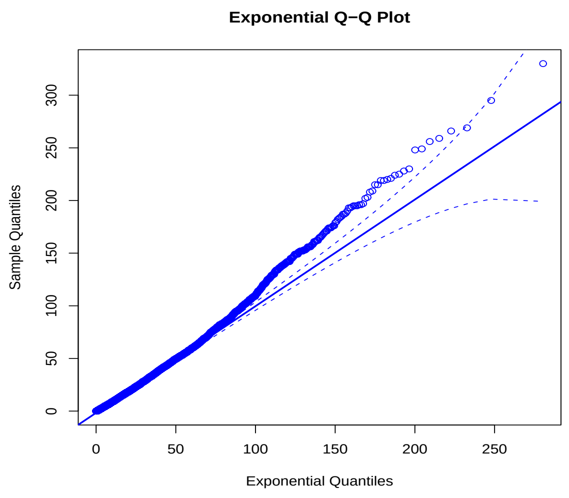


Figure 4.8 Q-Q Plot for High Traffic, 10 Minutes

Distribution of selected months

Due to strong seasonality and monthly pattern, it makes good sense that the tests at Level I (based on the aggregated data) reject the exponential distribution for all intervals of time. To accommodate the monthly arrival pattern, at Level II we choose March, July and November to represent three academic terms (i.e., fall, winter and spring & summer). We repeat the similar procedures as at Level I by distinguishing traffic intensity.

EDF against CDF, low traffic: The heuristic histograms, like Figures 4.1, 4.2, 4.3 and 4.4 were ambiguous in helping us decide the fit. We use a more helpful graphical method in this section instead, the EDF against CDF. Figures 4.9 and 4.10 each show a comparison of the EDF of the sample to the CDF of the generated exponential distribution. The mean of the sample as $1/\lambda$ is used to generate the exponential distribution. In order to achieve a smooth CDF curve, we generate a great number of exponentially distributed random variables, at least 100 points within each intervals. From this point on, the light smooth curve appearred in EDF vs CDF figures is CDF while EDF is depicted by dark steps. Since data are too few for low traffic to smooth an EDF curve, we can hardly tell if EDF matches CDF shown in Figure 4.9.

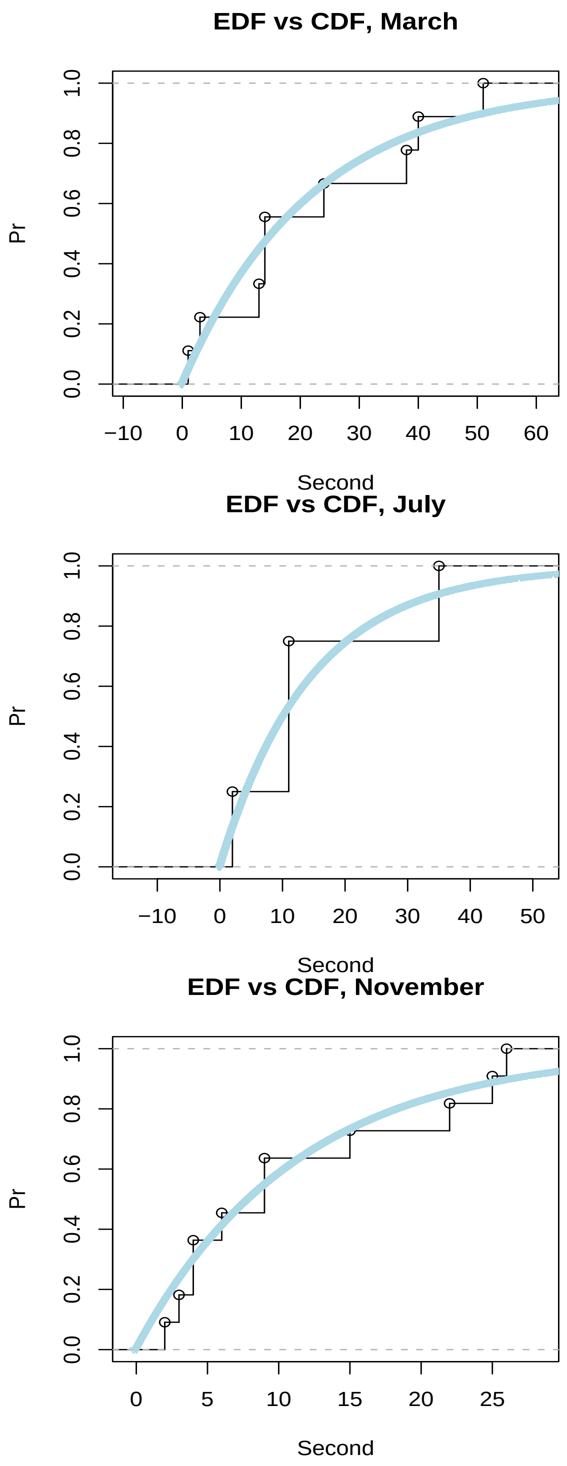


Figure 4.9 EDF vs CDF for Low Traffic, 1 Minute by Month

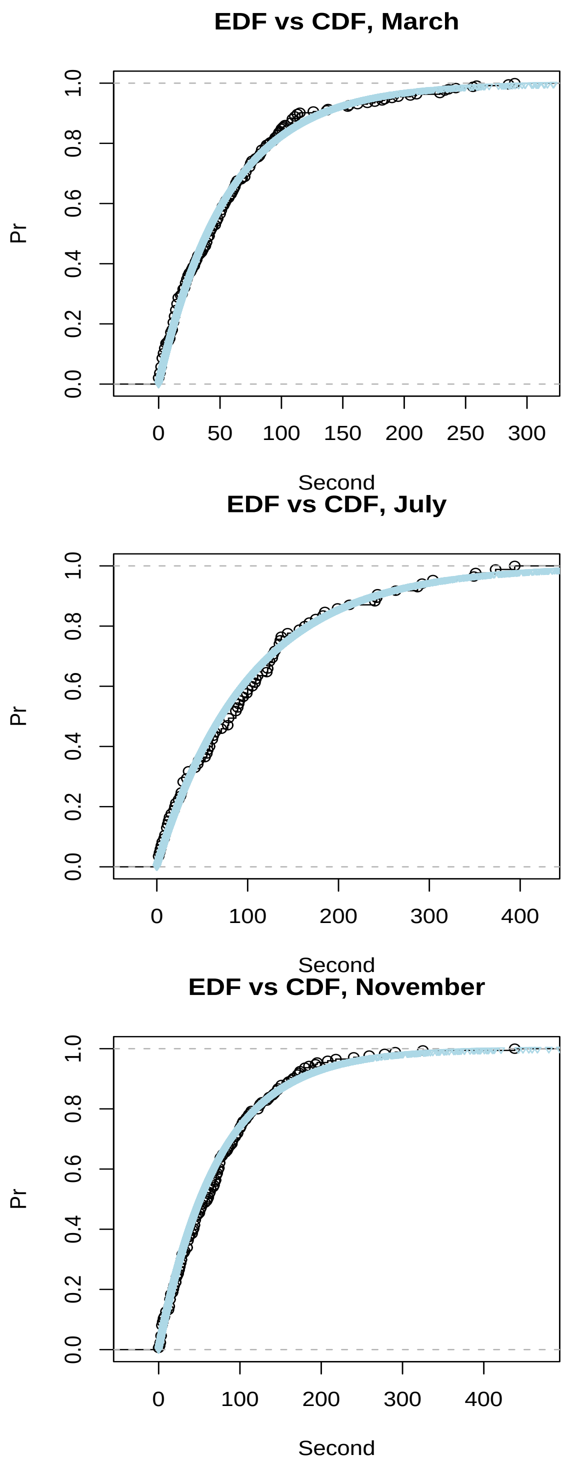


Figure 4.10 EDF vs CDF for Low Traffic, 10 Minutes by Month

	1 min	10 min
N	9	244
K-S (p-value)	0.76	0.70
A-D	Accept H_0	Reject H_0

Table 4.4 Low Traffic, March 2000

	1 min	10 min
N	4	85
K-S (p-value)	0.92	0.78
A-D	Accept H_0	Accept H_0

Table 4.5 Low Traffic, July 2000

	1 min	10 min
N	11	174
K-S (p-value)	0.94	0.63
A-D	Accept H_0	Reject H_0

Table 4.6 Low Traffic, November 2000

Test summary, low traffic: Note that the sample sizes are small for all 1-minute intervals (see Tables 4.4, 4.5 and 4.6). That A-D accepts H_0 for all 1-minute intervals is probably because there are not enough data, suggesting that the power of the test is too small.

Q-Q plot, low traffic: For the same reason of small samples, for 1-minute intervals we cannot conclude a match/mismatch with the exponential distribution. For 10-minute intervals, in Figure 4.12 each plot shows quite a few points off the straight lines.

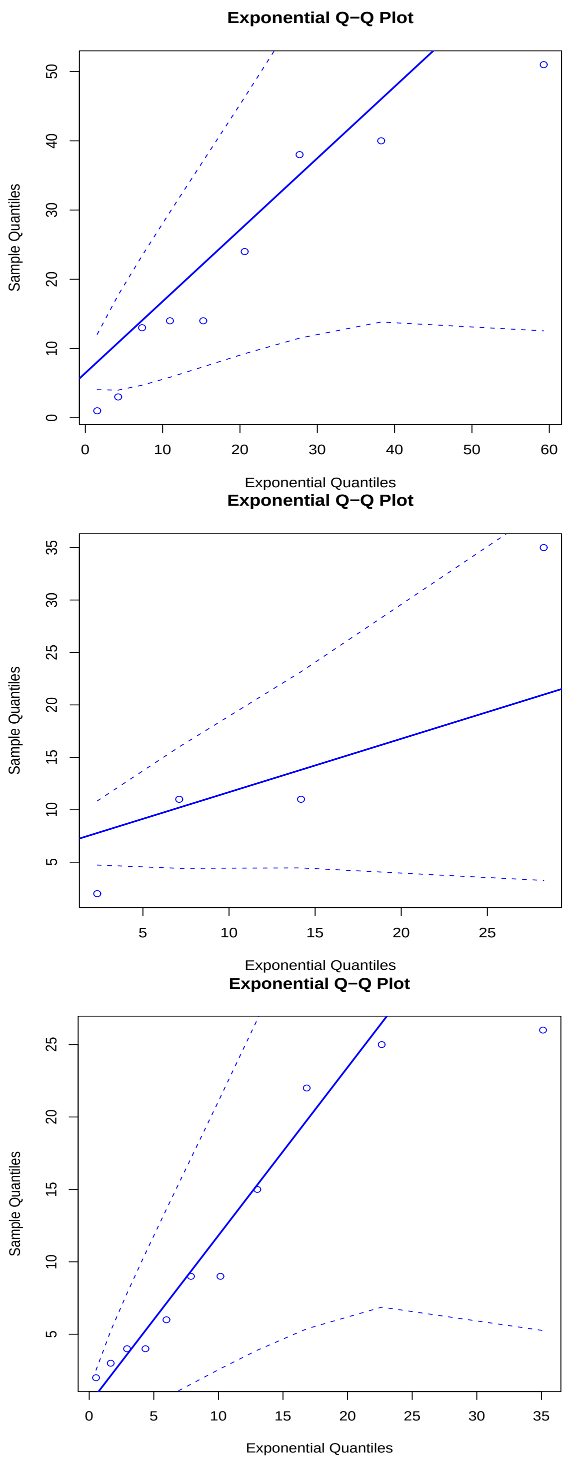


Figure 4.11 Q-Q plot for Low Traffic, 1 Minute by Month

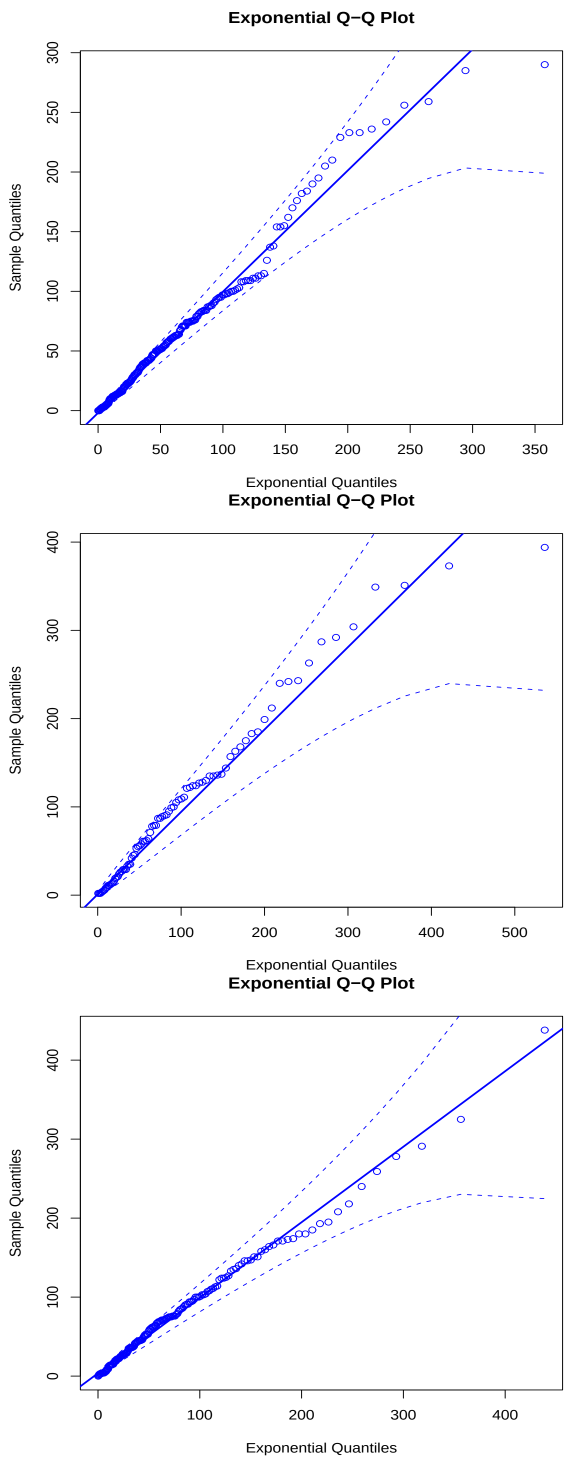


Figure 4.12 Q-Q plot for Low Traffic, 10 Minutes by Month

EDF vs CDF, high traffic: In Figures 4.13 and 4.14, from the top to the bottom, each plot represents March, July and November, respectively. For 1-minute intervals, samples are small, which makes the comparison hard. In contrast, for 10 minutes, Figure 4.14 shows that EDF matches CDF in each plot.

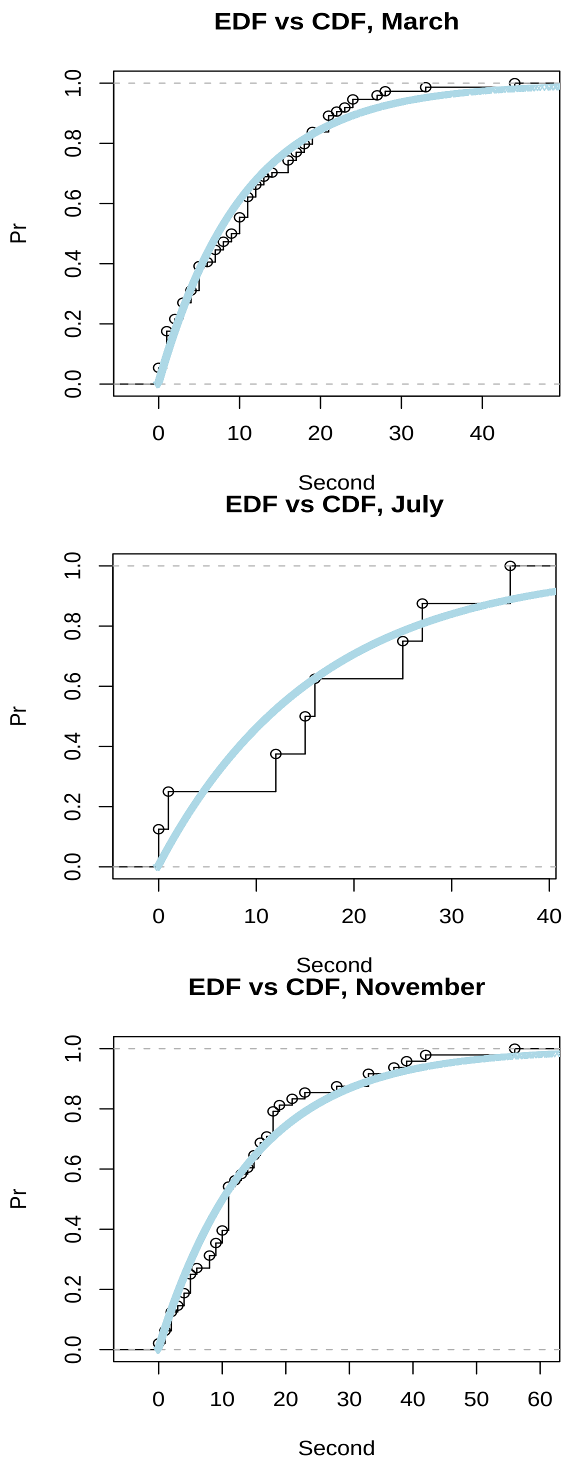


Figure 4.13 EDF vs CDF for High Traffic, 1 Minute by Month

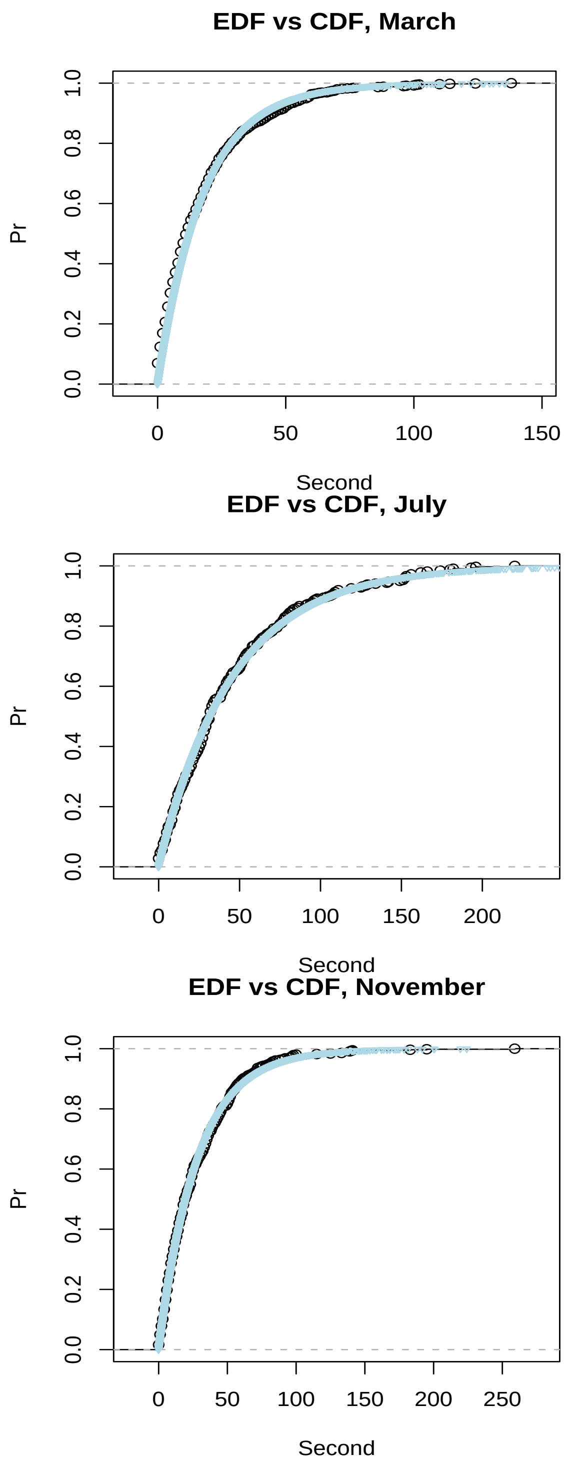


Figure 4.14 EDF vs CDF for High Traffic, 10 Minute by Month

Test summary, high traffic: A-D rejects the null hypothesis for all intervals. K-S accepts it for all intervals too except for 10-minute interval, March 2000 shown in Table 4.7.

	1 min	10 min
N	74	921
K-S (p-value)	0.33	0**
A-D	Reject H_0	Reject H_0

Table 4.7 High Traffic, March 2000

	1 min	10 min
N	8	322
K-S (p-value)	0.62	0.69
A-D	Reject H_0	Reject H_0

Table 4.8 High Traffic, July 2000:

	1 min	10 min
N	48	561
K-S (p-value)	0.20	0.74
A-D	Reject H_0	Reject H_0

Table 4.9 High Traffic, November 2000

Q-Q plot, high traffic: For 1 minute, it is still hard to make a conclusion due to limited number of points. For 10 minutes, we see points at the tails are more likely fall off the straight lines than those at the heads shown in Figure 4.16.

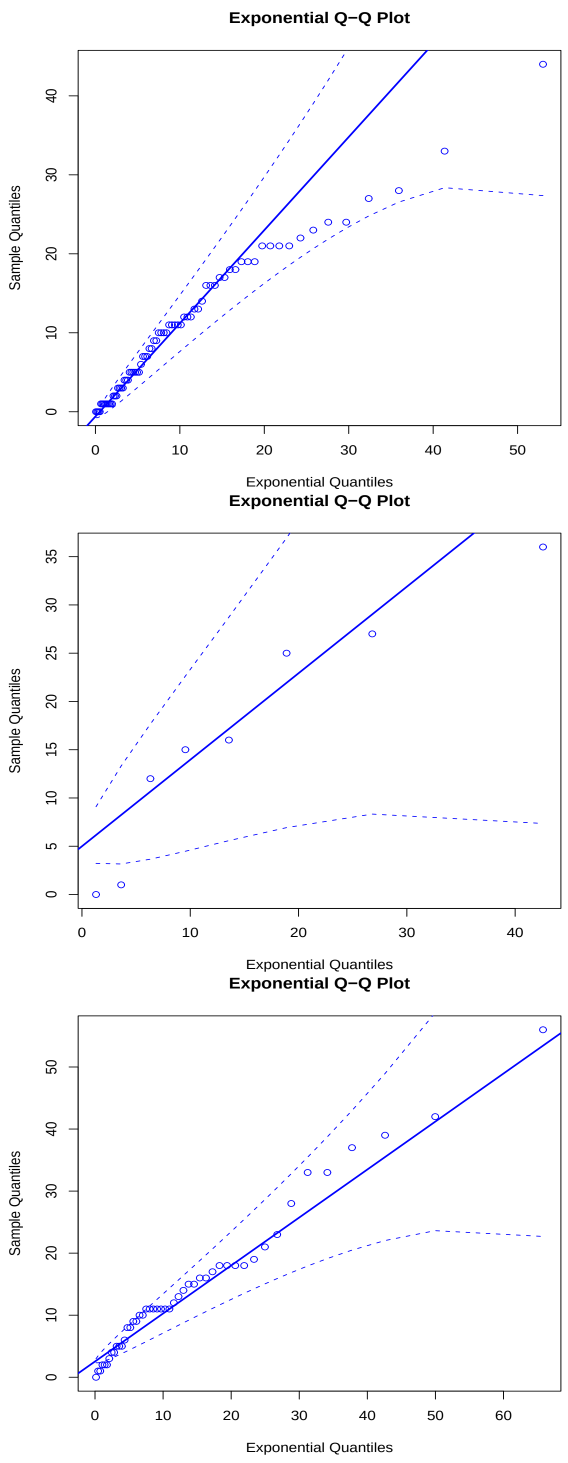


Figure 4.15 Q-Q plot for High Traffic, 1 Minute by Month

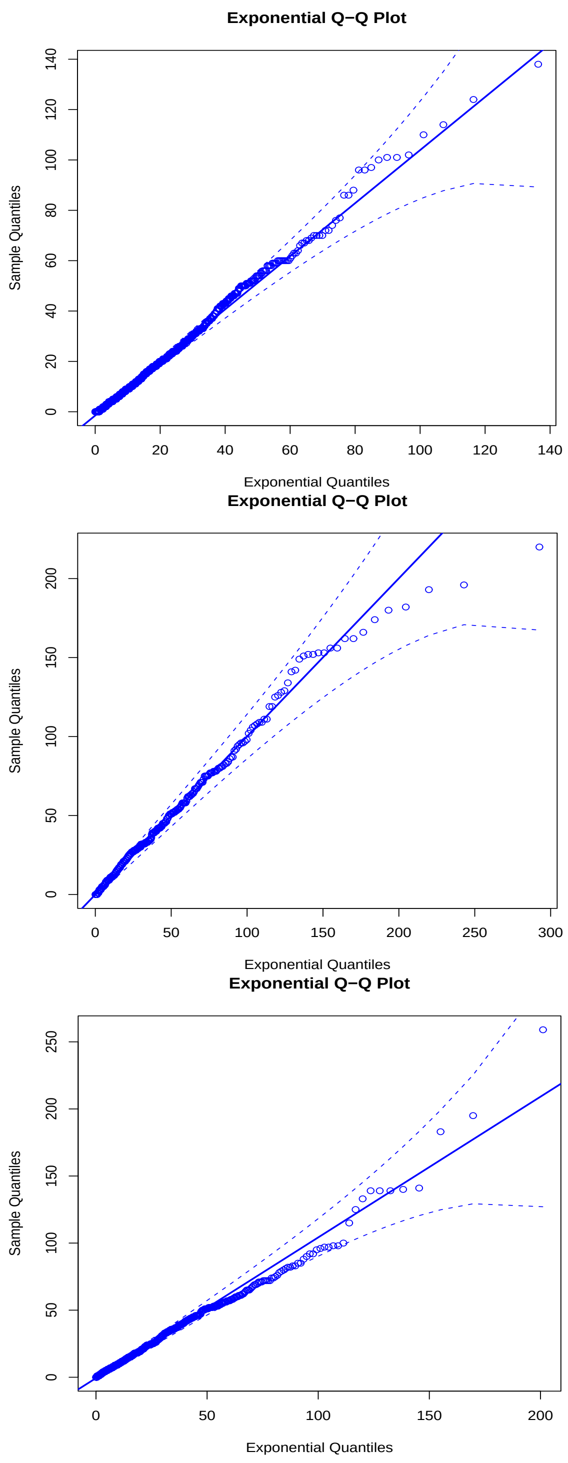


Figure 4.16 Q-Q plot for High Traffic, 10 Minute by Month

According to the results of interarrival time tests, we should not ignore the seasonal and monthly effects. Even though a much larger data sample could be achieved, these effects are reinforced when we aggregate data. Regardless of traffic intensity, it suggests that we should not choose any interval either exceeding 10 minutes or as small as 1 minute. In the first case, the null hypothesis is rejected. In the second case, the sample are too small. 5 minutes seems a good choice in between.

4.2.2 Testing exponential R_{ij}

We test independent standard exponential distribution of R_{ij} in order to validate the nonhomogeneous Poisson model. We apply a similar plan of testing as previously. That is, we test 1-, 5- and 10-minute intervals, respectively. As discussed in Section 4.1.3, once we calculated transformed R_{ij} from the data, R_{ij} is independent of time of day, day of month or month of year, but is a standard exponential random variable. Thus, we test the null hypothesis by aggregating R_{ij} (which is similar to the first step in testing the interarrival times).

EDF vs CDF: When the sample size is not large enough, we can see the steps of EDF as EDF is a step function. In Figures 4.17 and 4.18, EDF coincides CDF for both types of traffic.

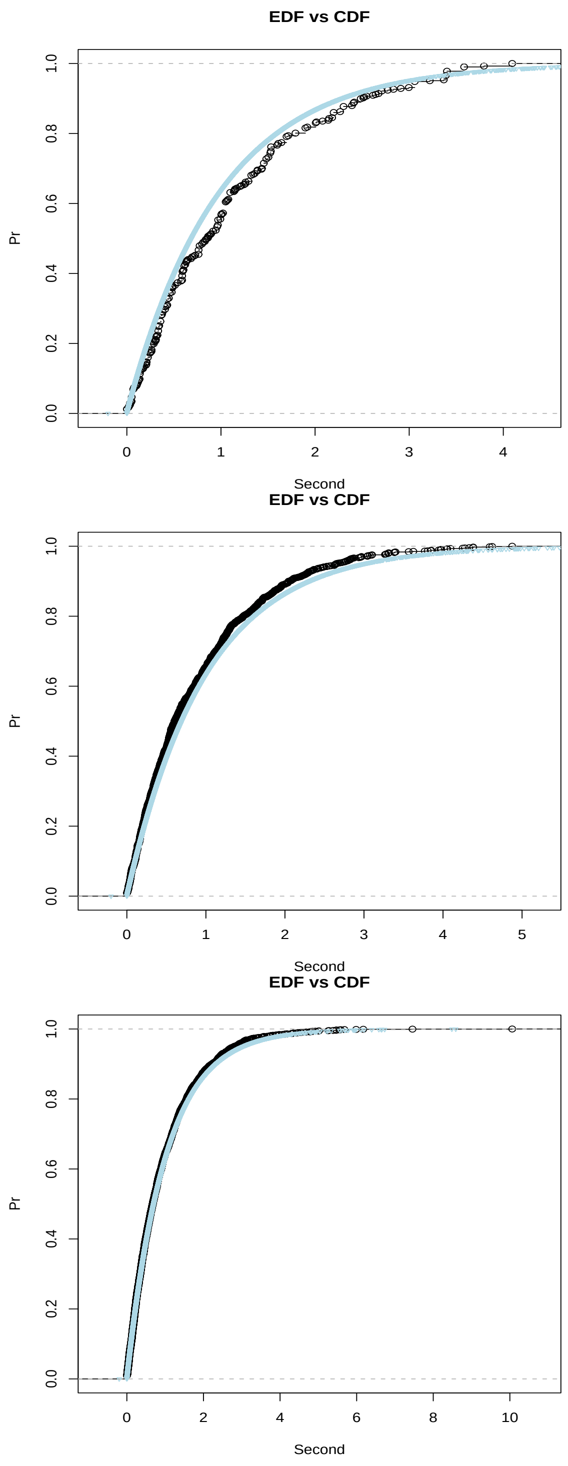


Figure 4.17 EDF vs CDF for R_{ij}, Low Traffic

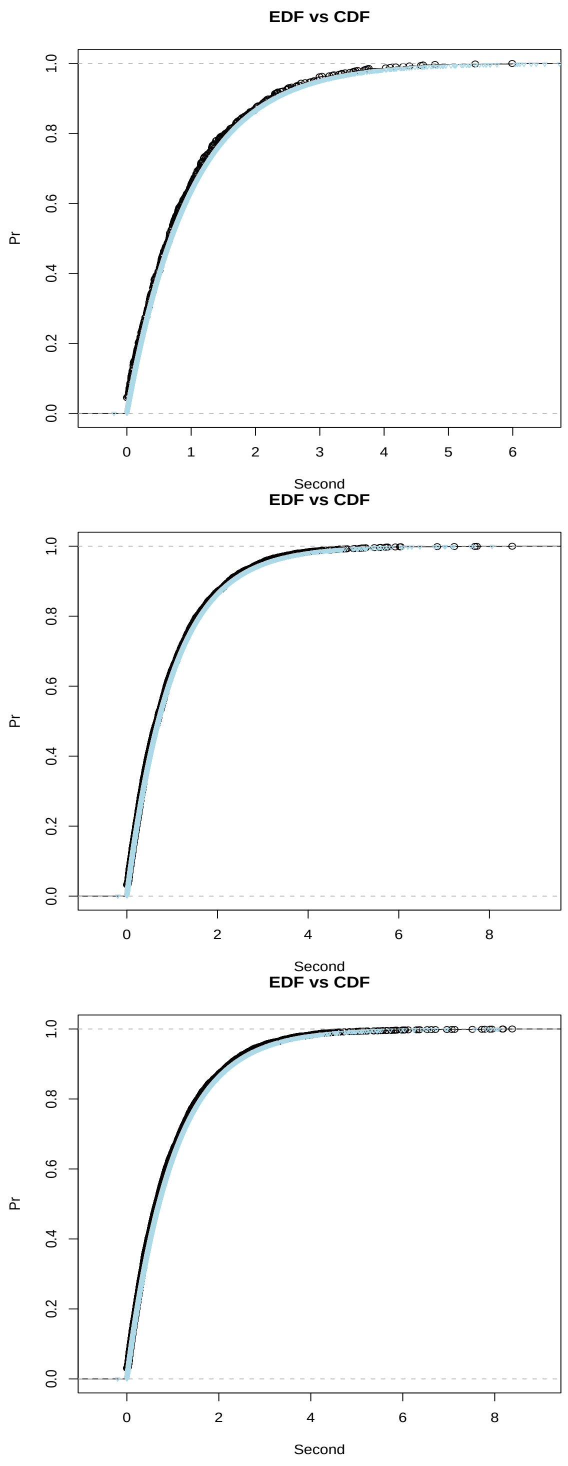


Figure 4.18 EDF vs CDF for R_{ij}, High Traffic

	1 min	5 min	10 min
N	407	1221	2407
Mean	1.098	0.89	0.97
SD	0.95	0.85	0.96
CV	0.87	0.95	0.99
K-S (p-value)	0.12	0.50	0.95
A-D	Accept H_0	Accept H_0	Accept H_0

Table 4.10 Test Summary for R_{ij}, Aggregate Low Traffic

Test summary for R_{ij}: If R_{ij}s are standard exponentially distributed random variables, the mean and standard deviation (SD) are both equal to 1, meaning the coefficient of variation (CV) defined by the ratio of the SD over the mean is equal to 1, too. Both Tables 4.10 and 4.11 show that means and SDs are all close to 1. CVs are around 1 except 0.87 at 1 minute for low traffic, which is off by 13%.

Results tested by K-S and A-D are fairly consistent with each other for all intervals except for the 5-minute intervals for the high traffic. The disagreement between the two tests is shown in Table 4.11. Based on the results by K-S and A-D, we use intervals of 5 minutes to model the intensity rate.

	1 min	5 min	10 min
N	711	3470	6753
Mean	0.94	0.97	0.96
SD	0.93	0.98	0.99
CV	1.00	1.02	1.03
K-S (p-value)	0.11	0.0013	0***
A-D	Accept H_0	Accept H_0	Reject H_0

Table 4.11 Test Summary for R_{ij}, Aggregate High Traffic

Q-Q plots: In Figure 4.20, some of the points fall off the 95% confidence interval. Q-Q plots confirm our decision that within 5-minute intervals the arrival rate is constant.

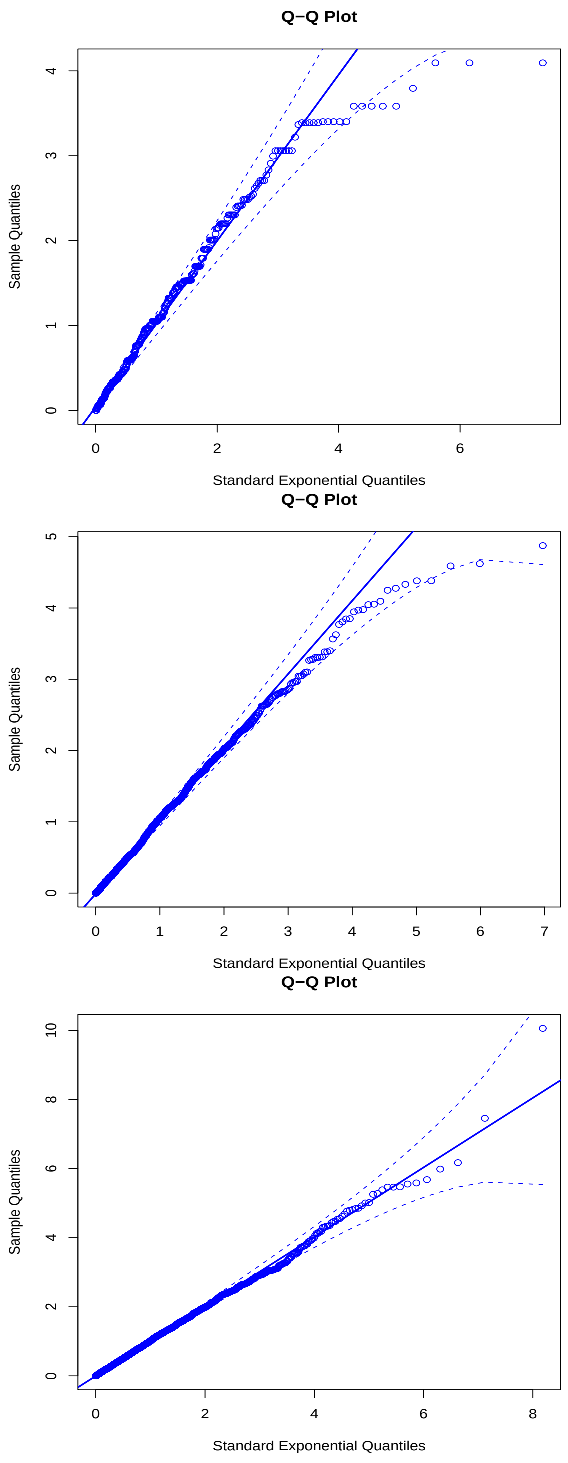


Figure 4.19 Q-Q Plot for R_{ij}, Low Traffic

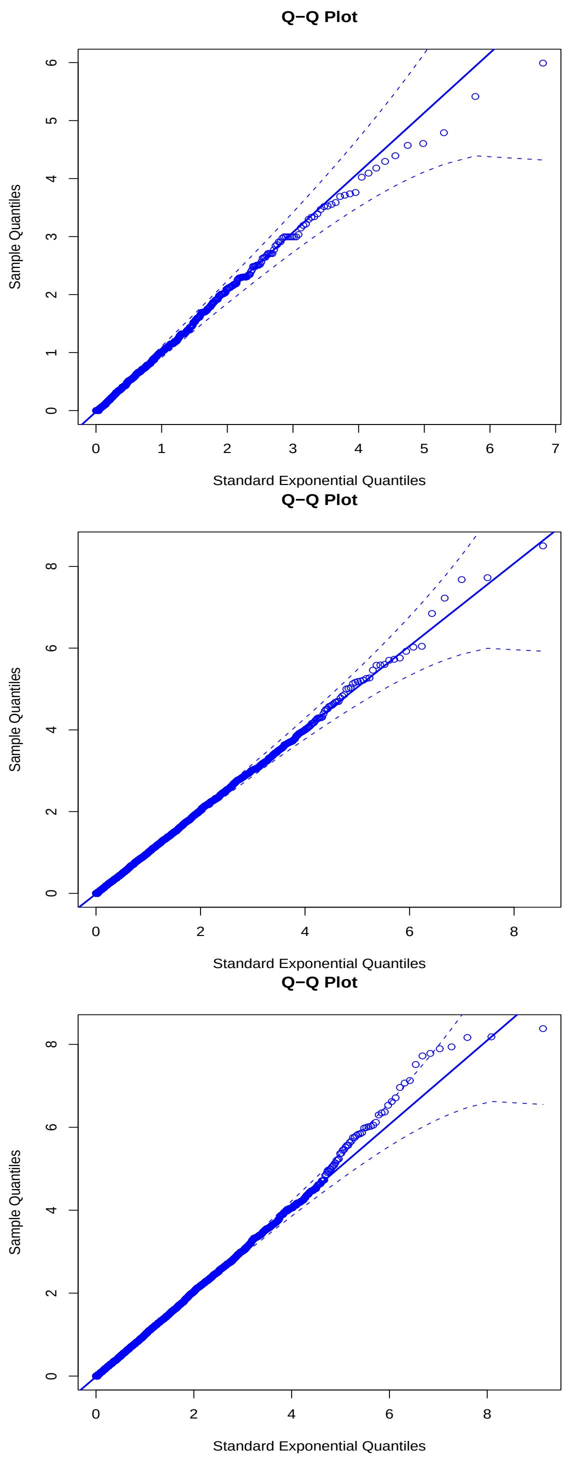


Figure 4.20 Q-Q Plot for R_{ij}, High Traffic

4.2.3 Independence of R_{ij}

Since many of our estimation or testing procedures such as MLE and Chi-squared tests rely on independence, we test whether the sample data are independent within each time interval, as well as between the first lag of the 1-minute or 5-minute intervals. Note that we choose this length of interval based on the test result in Section 4.2. In addition, since the previous tests show that statistics R_{ij}s are exponentially distributed for 5-minute interval, we then proceed to test the independence of R_{ij}s.

Testing autocorrelation (ACF) by t test

To test this hypothesis, we test if the autocorrelation at lag 1 is significantly different from zero. We partition R_{ij}s into two subsets denoted by $\{X_i\}$ and $\{Y_i\}$. Namely, $\{X_i\}$ is the sample of N R_{ij}s discounting the last element by the order of their occurrences, whereas $\{Y_i\}$ discounts the first element. Thus, the sample correlation is,

$$\hat{r}_{xy} = \frac{\sum_{i=1}^{N-1}(X_i - \bar{X})(Y_i - \bar{Y})}{(N-1)\hat{\sigma}_x\hat{\sigma}_y}, \tag{4.4}$$

where $\bar{X}, \bar{Y}, \hat{\sigma}_x$ and $\hat{\sigma}_y$ are the sample means and sample standard deviations for the subsets. Then the corresponding t-statistic is

$$t = \frac{\hat{r}_{xy}\sqrt{(N-1)-2}}{\sqrt{1-\hat{r}_{xy}^2}}, \tag{4.5}$$

with the degrees of freedom $\nu = N - 3$ if these two subsets are normally distributed. However, the normality requirement is not stringent as the test statistic can

	1 min	5 min	10 min
$\hat{r}_{xy}$	0.071	-0.0064	-0.0028
t-value	1.429	-0.225	-0.139
p-value	0.15	0.82	0.89

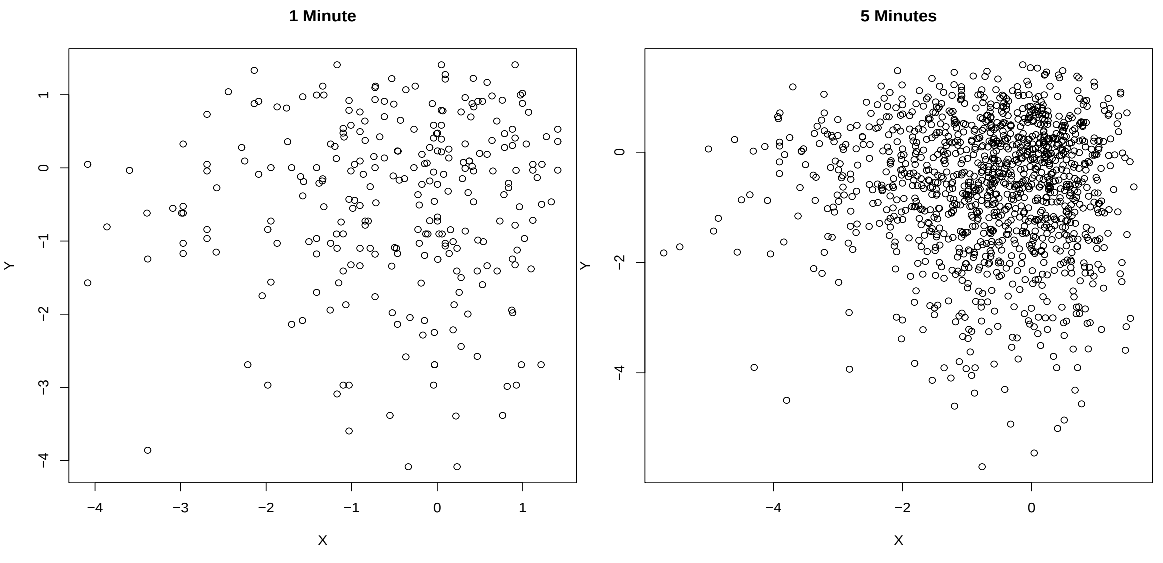
Table 4.12 Independence Test Summary for R_{ij}, Aggregate Low Traffic

	1 min	5 min	10 min
$\hat{r}_{xy}$	-0.075	-0.012	-0.010
t-value	-2.007	-0.716	-0.827
p-value	0.045	0.47	0.41

Table 4.13 Independence Test Summary for R_{ij}, Aggregate High Traffic

approximate t distribution [26]. We present the test results in Tables 4.12 and 4.13.

Except for 1-minute high traffic, all p-values are greater than 0.05. Since the p-value is 0.045 for 1-minute high traffic which is very close to 0.05, there is no strong evidence to reject the independence hypothesis at that level as the first plot in Figure 4.22 does not show any pattern. In addition to the quantitative test, we can examine if there is any pattern or relation between $\{X_i\}$ and $\{Y_i\}$ for $i = 1, \ldots, N-1$. Figures 4.21 and 4.22 are scatter plots for paired $\{X_i, Y_i\}$ and due to the large number of pairs, all the figures are plotted on a log scale in order to disperse the data. These figures help us make the conclusion that all R_{ij}s are independent for the tested times.



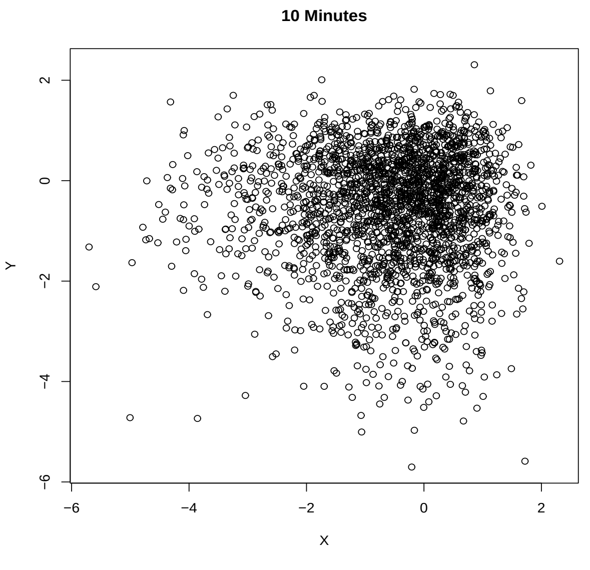
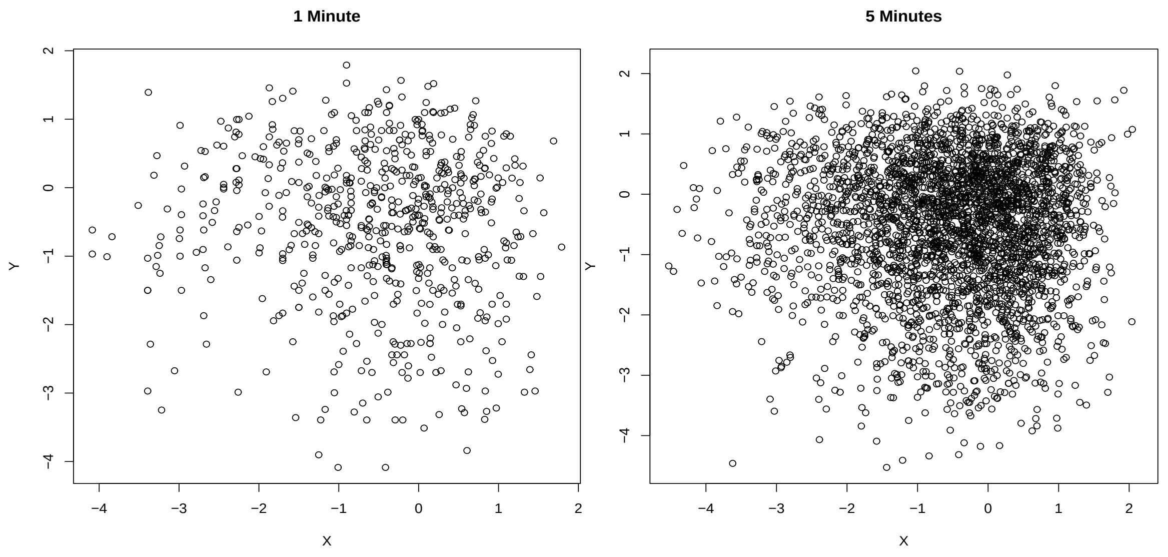


Figure 4.21 Scatter Plots for $\ln R_{ij}$ (lag 1), Low Traffic



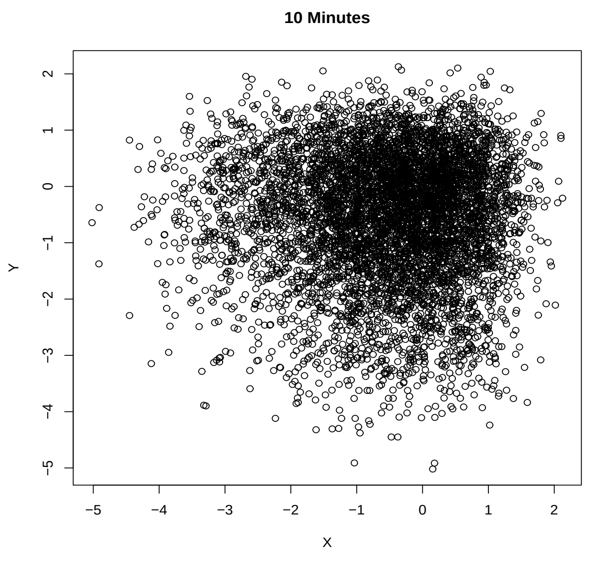


Figure 4.22 Scatter Plots for $\ln R_{ij}$ (lag 1), Low Traffic

Low traffic	1 min	5 min	10 min
p-value	0.15	0.82	0.89
High traffic	1 min	5 min	10 min
p-value	0.05	0.47	0.41

Table 4.14 Ljung-Box Test Summary for R_{ij}

Ljung-Box test

Alternatively, we can test the null hypothesis that the arrivals are independent in a given time series using the Box-Pierce or Ljung-Box tests [25]. The advantage of this test lies in its robustness, without the assumption of the normality of the arrivals distribution. Nevertheless, the normality can be achieved by either a large sample through the central limit theorem or a large sample mean through the normal approximation to the exponential distribution [3].

Results in Table 4.14 agree with results by t-test. The ACD plots do not exceed 0.2 at all lags, which indicate large degree of independence amongst R_{ij}.

In summary, we would like to use the data summarised by every 5 minutes. Also, transformed R_{ij} shows the standard exponential distribution hypothesis is accepted even for 10-minute interval. It seems that R_{ij}s are better at accommodating seasonality and monthly factor than directly using interarrival time when testing the null hypothesis.

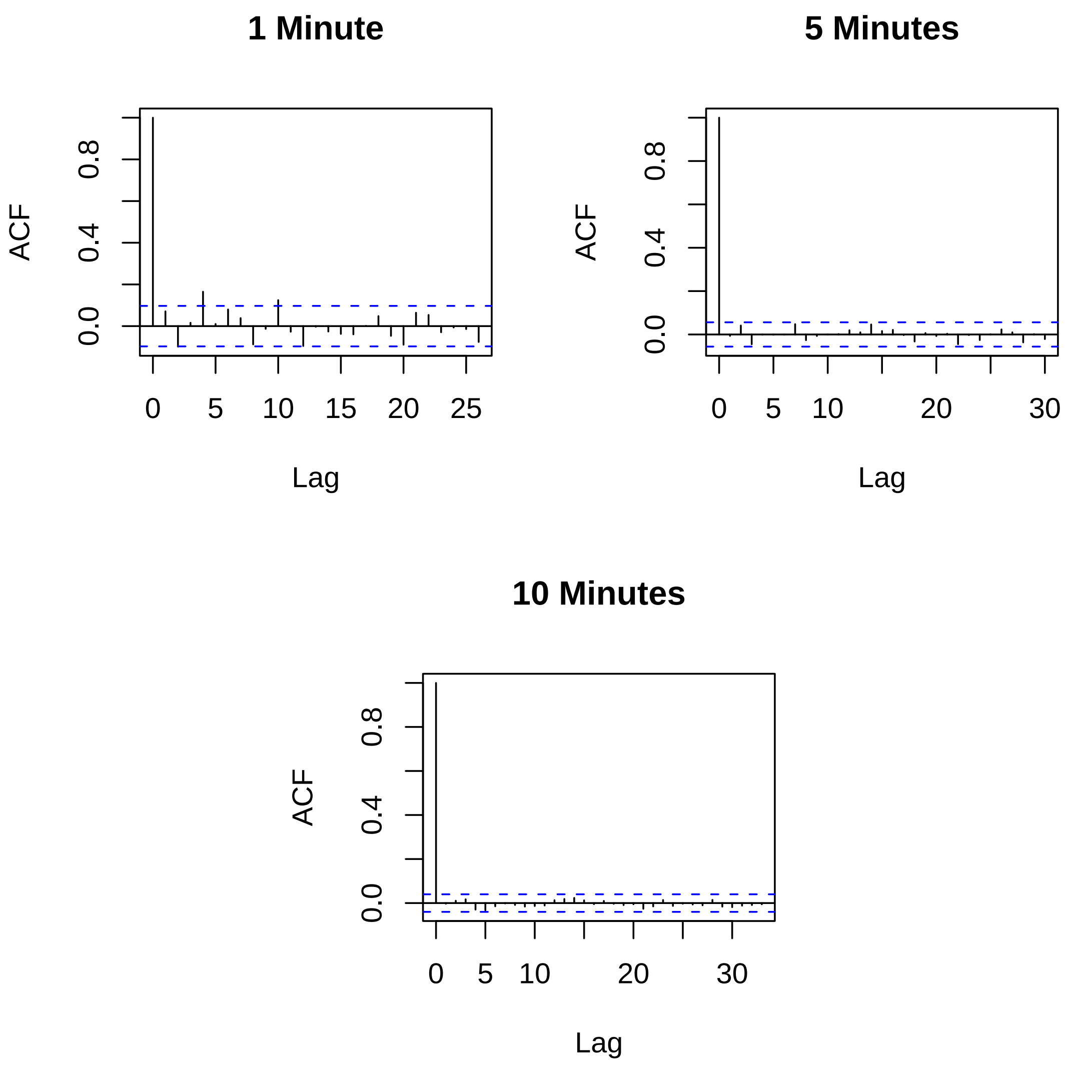


Figure 4.23 ACF Plots for R_{ij}, Low Traffic

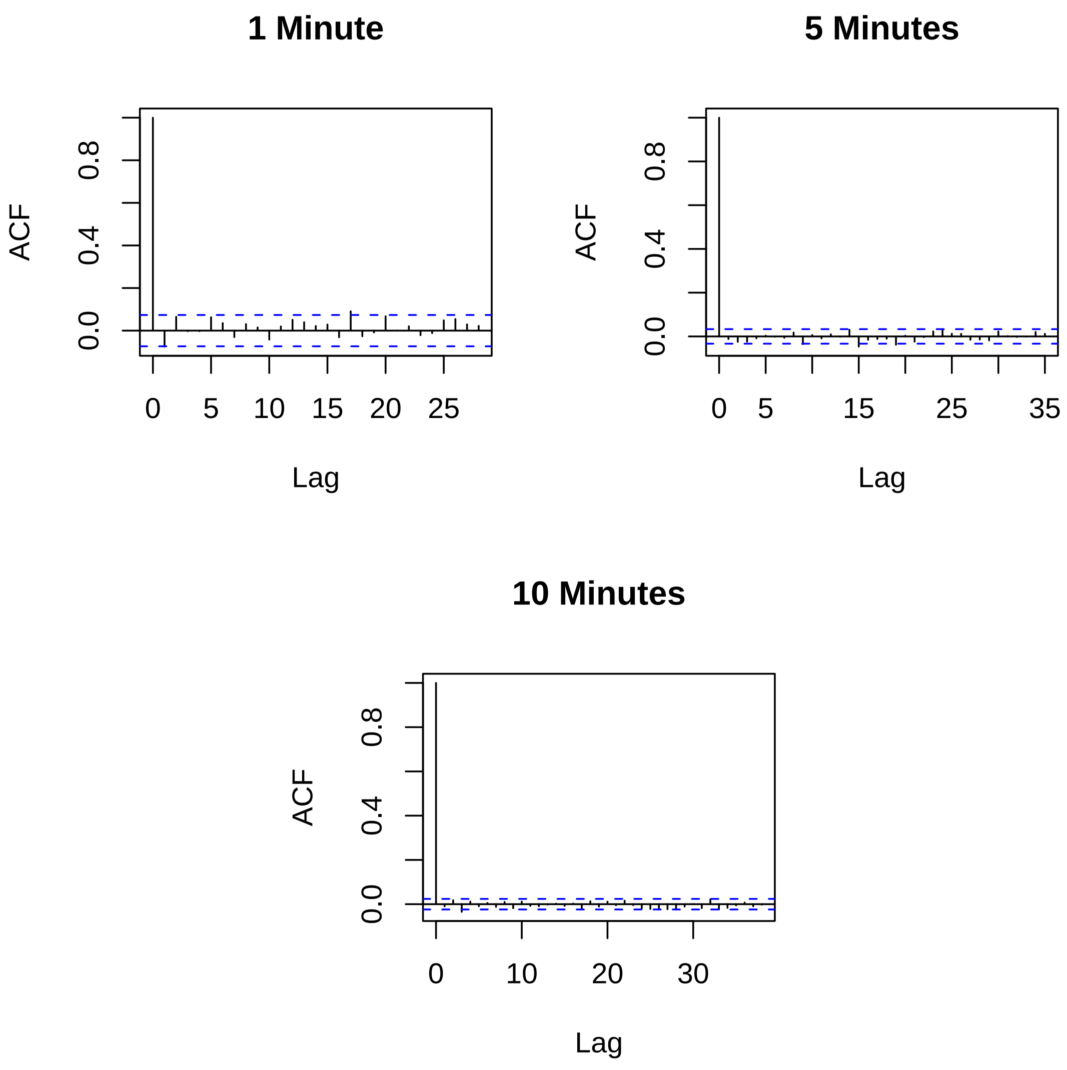


Figure 4.24 ACF Plots for R_{ij}, High Traffic

Chapter 5
MODELLING THE INTENSITY RATE FUNCTION

Earlier test results based on the interarrival times in Section 4.2 suggest that the homogeneous Poisson process is not appropriate for arrivals, especially when the tested interval exceeds 10 minutes. Then, our null hypothesis becomes that the arrivals follow a nonhomogeneous Poisson process. We have accepted the null hypothesis of standard independently exponentially distributed R_{ij} for 5-minute interval. That is to say, our arrivals process can be modelled by a nonhomogeneous Poisson with a piecewise constant (moderately changed time-varying) arrival rate. The next question is whether we can model this rate. If the form of model is nice, say it is a linear function, then the future call demand can be predicted by using the intercept and slope estimated by the data set.

5.1 Testing Linear Intensity Rate

Given the null hypothesis that the arrivals conform to a nonhomogeneous Poisson process at the levels of 1 minute and 5 minute, we choose to apply the linear rate model $\lambda(t) = a + bt$ to the data summarised by 5 minutes. We start testing the fit of a linear rate model by sampling 1 hour arrival from different times of a day. We still separate the types of traffic as we would like our plans of test to be consistent.

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	7.1	1.3	5.28	0***
Slope	0.044	0.039	1.13	0.28
	R-square	Adjusted R-square	F statistic	p-value
	0.11	0.025	1.285	0.28

Table 5.1 Test Summary for Linear Rate by OLS, Low Traffic

5.1.1 Results by OLS, IWLS and MLE

Low traffic

Since the traffic around 1:00 is too low, we pick some time instead of midnight in order to visualize the fit result. We choose interval from 10:00 to 11:00, Wednesday, March 8, 2000 to present low traffic, averaging 8.6 arrivals per hour. We present a summary of test results by type of estimators along with graphs of fitted line as follows,

OLS: The dependent variable y represents the number of calls regressed by independent variable x, time intervals. In Table 5.1, the intercept valued at 7.1 is significant (indicated by zero p-value) whereas the slope valued at 0.044 is insignificant. We could say the rate of increase is constant. However, the summary also shows a very low R^2 (0.1139) which tells us how much the variation of the independent variable (number of calls) can be explained by the linear model. We even have a much lower adjusted R^2 which takes into account the degrees of freedom.

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	6.99	1.3	5.48	0***
Slope	0.048	0.03 8	1.24	0.24

	R-square	Adjusted R-square	F statistic	p-value
	0.13	0.046	1.536	0.24

Table 5.2 Test Summary for Linear Rate by IWLS, Low Traffic

IWLS: The test takes 5 iterations to converge with a preset tolerance level at 10^{-6}. We have a significant 6.99 intercept but an insignificant 0.048 slope. The result by IWLS is similar to OLS in comparing Table 5.2 to Table 5.1. We also have very low R^2 and adjusted R^2.

MLE: Using MLE, we have a 6.99 intercept and a 0.048 slope by following discussion in Section 3.3.1, Chapter 4. The results of IWLS and MLE coincide. Figure 5.1 superimposes fitted lines by these procedures. As proved by Massey et al. [27], these three estimation procedures do not differ much if the number of arrivals in the first and the last intervals are not zero. ML estimators also coincide with the solution of IWLS.

High traffic

We choose the interval from 16:00 to 17:00, averaging 17.2 calls per 5-minute interval which double the low traffic. The tests in a sense repeat what we have seen

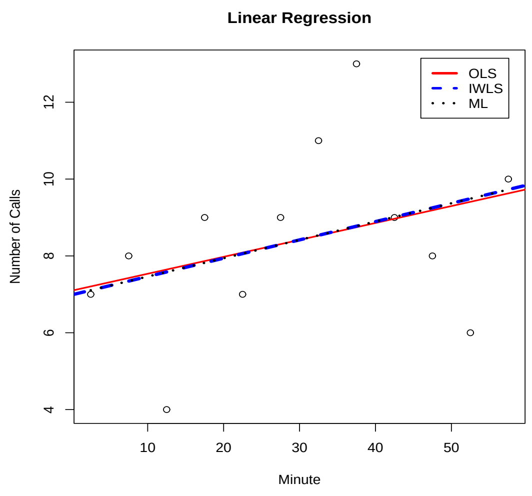


Figure 5.1 Comparison of OLS, IWLS and MLE for Linear Rate, Low Traffic

for low traffic. Similarly, we put OLS, IWLS and MLE together in the analysis.

OLS: In Table 5.3 shows a similar result as in the low traffic case with a significant intercept and insignificant slope. The R-squares are still low in this case.

IWLS: IWLS takes 3 iterations to converge. It also presents similar results as OLS.

MLE: MLE produces a 17 intercept and a 0.0015 slope. IWLS and MLE have identical intercepts and slopes. Fitted lines in Figure 5.2 confirms this identical observation.

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	17	2.6	6.48	0***
Slope	0.0014	0.074	0.019	0.99
	R-square	Adjusted R-square	F statistic	p-value
	0	-0.10	0	0.96

Table 5.3 Test Summary for Linear Rate by OLS, High Traffic

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	17	2.6	6.48	0***
Slope	0.0015	0.074	0.020	0.98
	R-square	Adjusted R-square	F statistic	p-value
	0	-0.10	0	0.98

Table 5.4 Test Summary for Linear Rate by IWLS, High Traffic

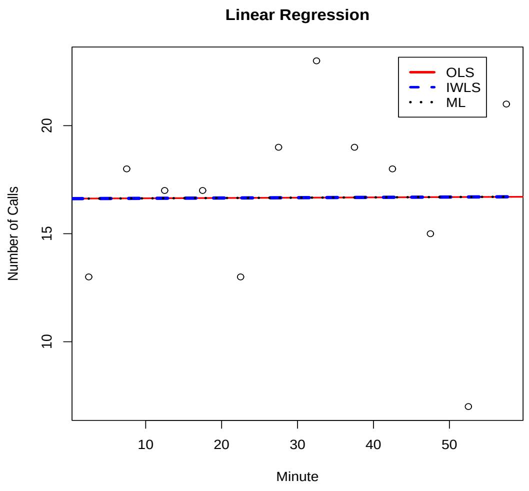


Figure 5.2 Comparison of OLS, IWLS and MLE for Linear Rate, High Traffic

5.1.2 LRT for the linear rate

For low traffic a 0.77 p-value with the degrees of freedom 10 accepts the null hypothesis that the linear rate fits the data. For high traffic, the Chi-squared test statistic has p-value 0.22 with the same degrees of freedom also accept the linear rate. Recall in Section 3.3.1, Chapter 3, the asymptotic LRT usually works fine for small samples too, although our sample may be too small with only 12 data points.

In summary, a large proportion of variation cannot be explained by the model, regardless of estimator used. The low R squares also indicate that the linear intensity model may not be a good fit to the data. The results suggest we should look at the sinusoidal rate as a alternative.

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	8.4	0.59	14.25	0
$sin(cx)$	-1.3	0.84	-1.59	0.15
$cos(cx)$	-1.4	0.84	-1.734	0.12
	R-square	Adjusted R-square	F statistic	p-value
	0.38	0.24	2.773	0.12

Table 5.5 Test Summary for Sinusoidal Rate by OLS, Low Traffic

5.2 Testing the Sinusoidal Rate

We apply the sinusoidal model with the same estimation procedures and LRT method as in the linear case. As a comparison to the linear rate, we also use 10:00 to 11:00 and 16:00 to 17:00 to represent low and high traffic.

Similar to the linear rate model, we restrict our attention to 1 hour. If we assume there is only one cycle, then recall the model is

$$\lambda(t) = a + b_1 \sin(ct) + b_2 \cos(ct) \text{ for } c = 2\pi/60.$$

5.2.1 Results by OLS, IWLS and MLE

Low traffic

OLS: The intercept 8.4167 has little difference from the one in the linear model. Neither of the slopes for $\sin(cx)$ and $\cos(cx)$ is significant. While we see large improvements in both of R^2 and adjusted R^2, they are lower than 0.40, meaning at least 60% variation cannot be explained by the sinusoidal model.

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	8.4	0.59	14.25	0
$sin(cx)$	-1.3	0.84	-1.59	0.15
$cos(cx)$	-1.4	0.84	-1.734	0.12
	R-square	Adjusted R-square	F statistic	p-value
	0.38	0.24	2.773	0.12

Table 5.6 Test Summary for Sinusoidal Rate by IWLS, Low Traffic

IWLS: The IWLS converges immediately at 10^{-6} level of tolerance. As noted earlier in the previous section about the linear rate model, the result is identical to OLS, which is shown in Figure 5.3.

MLE: The results of IWLS and MLE are almost the same. By MLE, the estimates are 8.4, -1.4 and -1.3. Figure 5.3 superimposes fitted lines by these procedures. As shown in the linear rate case, the three estimation procedures do not different much, and moreover, ML estimators coincide with the solution of IWLS.

High traffic

We still choose interval from 16:00 to 17:00 for high traffic. In a similar fashion, we combine OLS, IWLS and MLE together.

OLS: In contrast to low traffic, the R squares are much reduced. It suggests that it is not appropriate to use the same period parameter c as in low traffic. Fixing c does

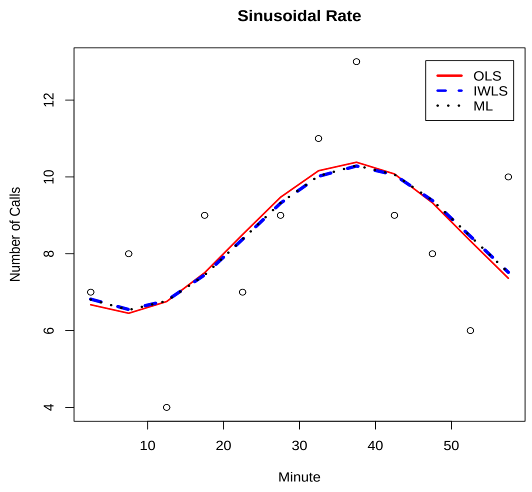


Figure 5.3 Comparison of OLS, IWLS and MLE for Sinusoidal Rate, Low Traffic

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	17	1.2	13.44	0
$sin(cx)$	0.23	1.8	0.13	0.90
$cos(cx)$	-2.24	1.8	-1.28	0.23

	R-square	Adjusted R-square	F statistic	p-value
	0.16	-0.033	0.826	0.47

Table 5.7 Test Summary for Sinusoidal Rate by OLS, High Traffic

	Estimate of Coefficient	Standard Error	t value	p-value
Intercept	17	1.3	13.05	0
$sin(cx)$	0.42	1.8	0.13	0.82
$cos(cx)$	-2.17	1.8	-1.21	0.26
	R-square	Adjusted R-square	F statistic	p-value
	0.14	-0.047	0.753	0.50

Table 5.8 Test Summary for Sinusoidal Rate by IWLS, High Traffic

not make sense as the arrival pattern of high traffic is different from the low traffic.

IWLS: It only takes 1 iteration to converge. The estimates are close to those produced by OLS. The same model for high traffic also shows lower R squares compared to low traffic.

MLE: MLE produces an intercept valued at 17 and slopes at 0.43 and -2.19. We once again see MLE is closer to IWLS than OLS as we have discussed before. In addition, we superimpose lines estimated by these three procedures in Figure 5.4.

5.2.2 LRT for the sinusoidal rate

For low traffic with a p-value 0.90 and the degrees of freedom 10, we accept the null hypothesis that the sinusoidal rate fits the data. However, for high traffic, we reject the sinusoidal rate with a zero p-value. Rejection at the high traffic level results from difficulty in setting the period parameter. This difficulty leads to limited use of

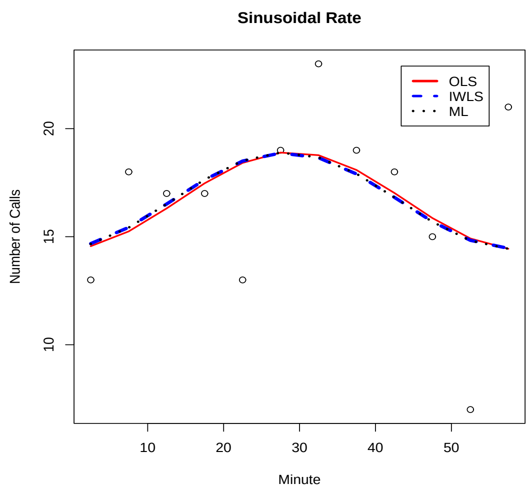


Figure 5.4 Comparison of OLS, IWLS and MLE for Sinusoidal Rate, High Traffic

sinusoidal rate.

5.3 Testing Spline models

5.3.1 Results by spline regression

We apply spline based models to low and high traffic data, again. Since spline regression is still a linear model, we can compare its R squares with those of the linear and sinusoidal rate models. Table 5.9 summarises the test results for low and high traffic.

For both low and high traffic, we use the degrees of freedom 7 and have much larger R squares than the previous models. The more knots used, the larger unadjusted R square can be obtained. We can specify the number of knots by way of setting the degrees of freedom. However, if one tries to merely obtain a large R square by interpolating data at each knot, this would result in an R square value close to 1.

Low traffic	R-square	Adjusted R-square	F statistic	p-value
	0.67	0.40	2.448	0.15
High traffic	**R-square**	**Adjusted R-square**	**F statistic**	**p-value**
	0.71	0.19	1.375	0.40

Table 5.9 Test Summary by Spline Regression

But, we would lose the smoothness. In general, the difficulty in choosing the number and location of the knots is a drawback of spline smoothing. Note that Figures 5.5, 5.6, 5.7 are the linear interpolation of the fitted data.

5.3.2 Results by smoothing spline

We use R function **sreg** to implement the natural cubic smoothing spline with roughness penalty approach. Since smoothing spline is not merely regression, we have no more R squares for comparison with other models. Nevertheless, Figures 5.7 and 5.8 provide us with a good visual demonstration of fitted curves. The smoothing parameters γ for low and high traffic are 10 and 0.52 chosen by GCV, respectively. With such small γ, Figure 5.8 actually interpolates data.

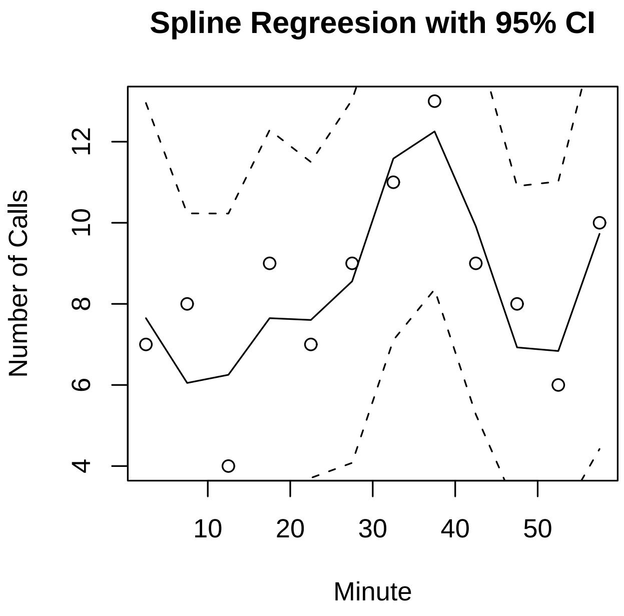


Figure 5.5 Spline Regression, Low Traffic

5.4 Model Selection

In spite of different eigenvalues of the hat matrix as discussed in Section 3.4.5, spline regression and smoothing spline make no difference in practice. Spline regression provide more control if one wants the fitted curve smoother in some locations and rougher in others. However, spline regression gives coarser control of the amount of smoothing to apply. We choose smoothing spline to compare with the parametric models.

As an overview, we superimpose lines fitted by linear, sinusoidal rate, and smoothing spline in Figures 5.9 and 5.10 for low and high traffic, respectively (Legend **ss** is a shorthand for smoothing spline). We plot fitted curves estimated by OLS for para-

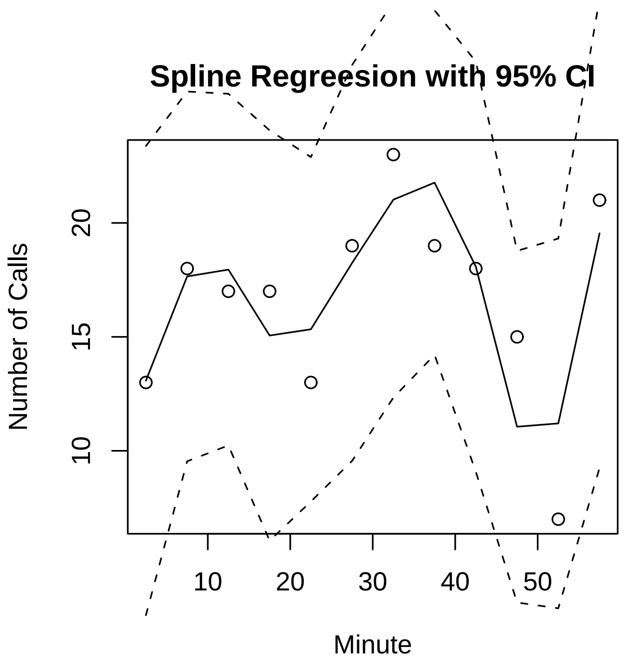


Figure 5.6 Spline Regression, High Traffic

metric models since OLS, IWLS and ML do not differ much as concluded previously. In both figures, splines allow more control of smoothness than parametric linear and sinusoidal rate models. Linear rate model, at one extreme, is globally smooth. It fails to capture the humps and valleys of the figures. Smoothing spline for high traffic, at another extreme, interpolates data and we lose the smoothness. We can specify a larger smoothing parameter to avoid interpolating too much, but the difficulty is in the choice of the degree of smoothness. A guideline provided by Hastie and Tibshirani [19] on choosing the degrees of freedom for spline regression should also shed light on this issue.

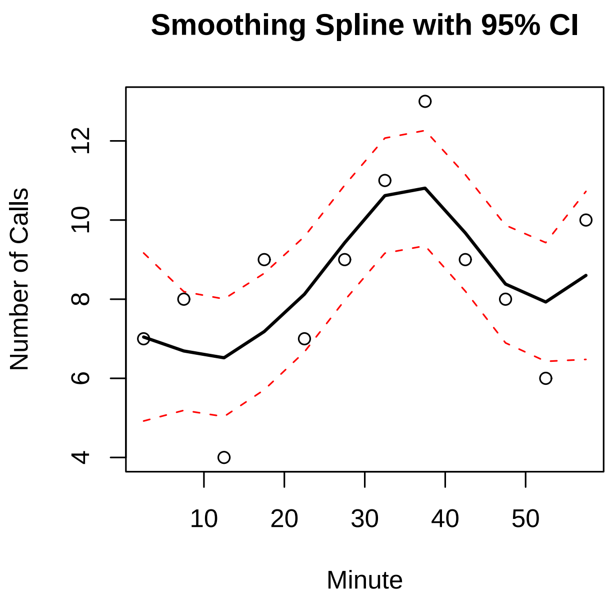


Figure 5.7 Smoothing Spline, Low Traffic

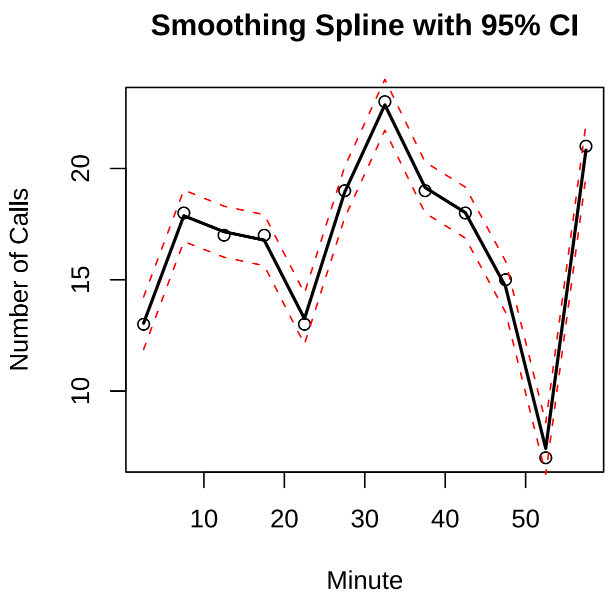


Figure 5.8 Smoothing Spline, High Traffic

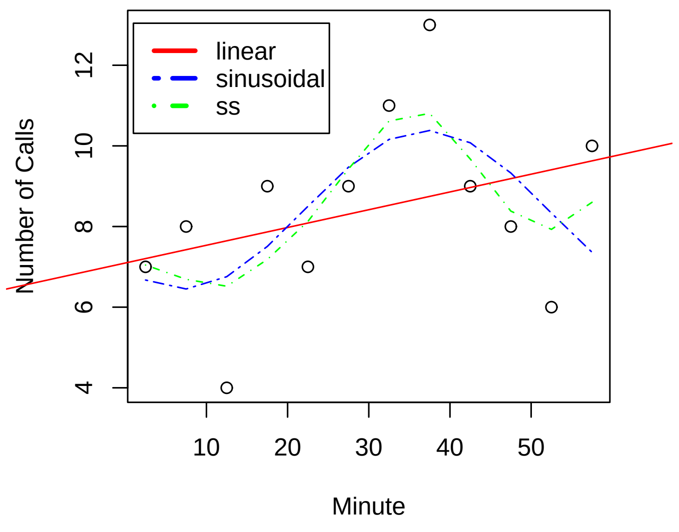


Figure 5.9 Fitted Lines, Low Traffic

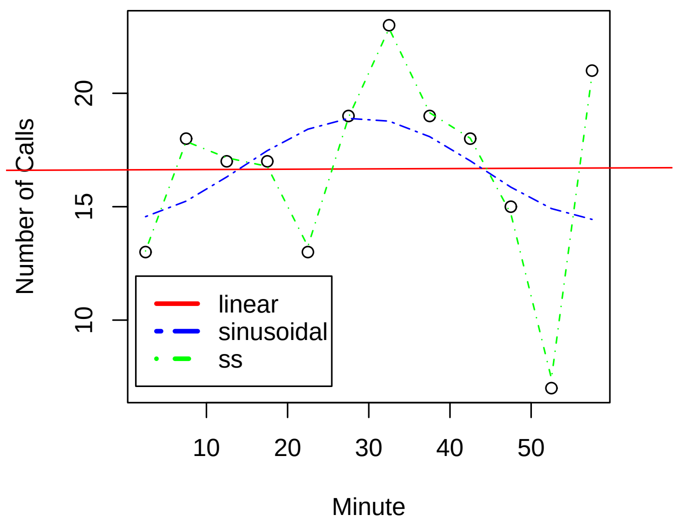


Figure 5.10 Fitted Lines, High Traffic

Chapter 6
CONCLUSION AND FUTURE RESEARCH

Despite the recent technological development of call centres and their successor contact centres, little advance has been made on the problem of staffing call centre agents. Even with increasingly sophisticated workforce management tools, a significant gap remains between the goal of effective staffing and the present difficulty to predict stochastic demand of inbound calls. To tackle this difficulty, we have investigated a time varying arrival process of modem pool users at the University. We have also estimated the intensity rate using linear, sinusoidal models and splines.

6.1 Summary of Findings

The major findings of the study are:

- We have tested the homogeneous Poisson process hypothesis by testing if the interarrival times were exponentially distributed. We have tested interarrival times for each of 1- and 10-minute intervals in Section 4.2.1, Chapter 4. We also distinguished traffic types by intensity when we performed the tests. In most respects, the quantitative goodness-of-fit A-D and K-S tests have shown lack of fit to the exponential distribution, especially for the intervals exceeding (and including) 10 minutes, which suggested no sign of a homogeneous Poisson process. Graphical methods such as histograms, EDF and CDF comparison, and Q-Q plots as visual aids have also been employed to demonstrate the lack of fit.

- We then proceeded to examine the nonhomogeneous Poisson process hypothesis using the transformed statistic R_{ij} proposed by Brown et al. [5]. Using R_{ij} helps remove the time varying property if the arrival rate varies slowly. Results tested by A-D and K-S for 1- and 5-minute have shown a match of R_{ij} with a standard exponential distribution. In addition, we have confirmed the independence hypothesis between the R_{ij}s by t tests and Ljung-Box tests on the ACFs of the R_{ij}s.

- In estimating intensity rate, parameters such as intercept and slope have been estimated by OLS, IWLS and ML procedures for parametric linear and sinusoidal rate models. Generally, IWLS and ML coincide. In evaluating the likelihood of the linear and the sinusoidal models, LRT have performed well. In the sinusoidal model, LRT for high traffic suggested difficulty in setting a period parameter. As an alternative, spline-based models offered more control of smoothing than the linear and sinusoidal models. A comparison of all models has strengthened this conclusion.

6.2 Limitations and Future Research Directions

- When we tried to decide what the best choice of interval is (i.e., to choose amongst 1, 5 and 10 minutes) in Section 4.2.2, Chapter 4, we chose 5 minutes because for low traffic, the standard exponentially distributed R_{ij}s hypothesis was accepted, but for high traffic, the hypothesis was rejected with a K-S p-value 0.0013 (shown in Table 4.11). In this case, the rejection may not be caused by the long interval (5 as opposed to 1 minute) but by the possibility that the power of test was so large due to the large sample size that even a slightest deviation from the null hypothesis would lead to the rejection. Thus, we decomposed the 5-minute interval into five 1-minute intervals and calculated new R_{ij}s based on the decomposed intervals. We could compare the new test results to those based on the 5 minute. In this way, the two had the same sample size (i.e., 3470), which helped remove the possibility that a large sample led to the rejection.

However, when we computed the test statistic, we found that there were a large number of zero R_{ij}s, and these zeros raised difficulty for the numerical tests to compute the differences between EDF and CDF correctly. The differences were plotted in Figure 6.1. These zeros had the largest (about 0.05) and the smallest deviations (0), which caused K-S the difficulty.

While we could not compare the numerical test results between the 5-minute interval and the decomposed intervals, the Q-Q plots could shed some light on which fitted the standard exponential distribution better. By comparing Figure 6.2 to Figure 6.3, we concluded that the decomposed Q-Q plot had a better fit.

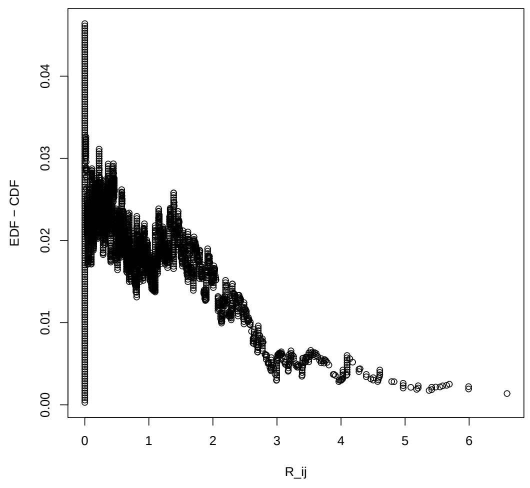


Figure 6.1 Deviations between EDF and CDF

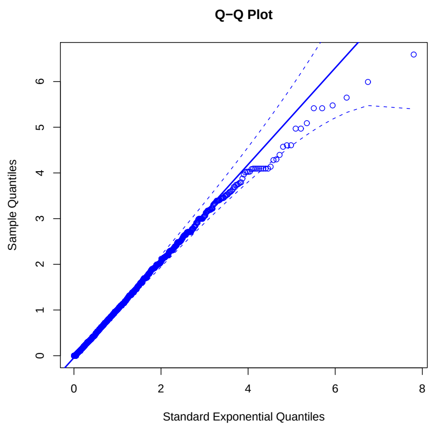


Figure 6.2 Q-Q Plot for 5 minute

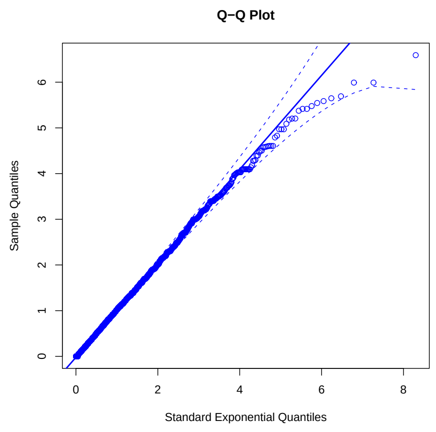


Figure 6.3 Q-Q Plot for Decomposed 5 1-minute

- The study does not review all the modelling methods that are available. For example, it is popular to use time series to analyse arrivals in the literature [15]. We only touched the surface of it as we employed ACF analysis for testing independence. Also, we feel splines are more flexible in modelling time varying arrivals with respect to call centres. Several books [11, 17, 19] have advocated a Bayesian approach to looking at spline models. Specifically, as in Hastie and Tibshirani [19], one can place a prior on $f(t)$ as in,

$$y = f(t) + \epsilon \quad . \tag{6.1}$$

When combined with a Gaussian model for the data, the conditional expectation of f or the posterior mean of f, can be shown to be a fitted smoothing spline with an appropriate smoothing parameter γ. This approach is an interesting direction for us to pursue if the intensity itself can be modelled by a Poisson mixture process as pointed out by Jongbloed and Koole [21].

- One last direction to pursue is to combine arrival with service time. Feldman et al. [12] have developed a simulation-based iterative staffing algorithm for which they assumed an $M_t/G/s_t + G$ model. (i.e., the model has a nonhomogeneous Poisson arrival process with a time-dependent arrival rate, generally distributed service times, a time-dependent number of servers and a non-exponential time-to-abandon distribution.) As our end goal is to provide input for staffing servers, the unified approach can achieve time-stable performance when we face general time varying arrivals.

Appendix A
Shell Scripts for Data Cleaning

A.1 Introduction

We list here the **sedscrpting environment** shell script files used for data cleaning. We format new data files after we clean the raw data set. By clean, we mean the data structure of the new data is consistent to the raw data set and is ready to feed in for most programming packages such as Splus and MATLAB. To this end, we remove the embedded HTML tags; we map the daily record in a separate file as the raw data set; 24/7 time continuous data is separated by the date the call is initialized as opposed to the call ended; we replace the information under the correct heading.

A.2 List of Script Files

sedscript Contains the HTML marks serving as a feed for **MyCleaner** to remove the HTML environment.

MyCleaner Read in the raw data records such as username and session start time and replace this kind of information under the correct heading into new data file.

MyBatchS Sort the data record and group them by the date the call is initialized. Apply this script after using **MyCleaner**.

chex Apply this command to all the scripts with their names starting with 'My' to make all the scripts executable, including those scripts in Appendix B.

sedscript

```
/Content-type:/ d
/HTML/ d
/HEAD/ d
/TITLE/ d
/BODY/ d
/PRE/ d
/USER/ d
/^$/ d
/!root/ d
```

MyCleaner

```
year=$1
### need an argument such as 2000
###input:sedscript;*.html for a year
###and the first day's html of the next year
###output: such as 2000Formated,2000Sorted,2000_01_01*s, 2000_01_01*

cat *.html >Merge$year  #include the first day's html file of the
next year
sed -f sedscript Merge$year > Filtered$year  #delete !root
rm Merge$year

awk '$6==yr {printf "%6s %3s %3s %-2s %8s %4s %-4s %-16s %6s %2s
%2s %-11s \n", $1, $2, $3, $4, $5, $6, $7, $8, $9, $11, $13,
$9*3600+$11*60+$13}' yr=$year Filtered$year > Formated1$year
rm Filtered$year

#Calculate the service time awk -F: '{print $1,  $2,  $3}'
Formated1$year |awk '{print $5*3600+$6*60+$7}'  > Formated2$year
paste Formated1$year Formated2$year >Formated$year
rm Formated1$year
rm Formated2$year

count=31
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Jan $count " Formated$year > $year"_01_0"$count
     sort +4  $year"_01_0"$count >$year"_01_0"$count"s"
   else
     grep "Jan $count " Formated$year > $year"_01_"$count
     sort +4  $year"_01_"$count  >$year"_01_"$count"s"
   fi

   count=‘expr $count - 1‘
done

count=29  # # of days in Feb while test $count -gt 0 do
   echo $count

   if test $count -lt 10
   then
     grep "Feb $count " Formated$year > $year"_02_0"$count
     sort +4  $year"_02_0"$count >$year"_02_0"$count"s"
   else
     grep "Feb $count " Formated$year > $year"_02_"$count
     sort +4  $year"_02_"$count  >$year"_02_"$count"s"
   fi

   count=‘expr $count - 1‘
done

count=31
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Mar $count " Formated$year > $year"_03_0"$count
     sort +4  $year"_03_0"$count >$year"_03_0"$count"s"
   else
     grep "Mar $count " Formated$year > $year"_03_"$count
     sort +4  $year"_03_"$count  >$year"_03_"$count"s"
   fi

   count=‘expr $count - 1‘
done
```

```
count=30
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Apr $count " Formated$year > $year"_04_0"$count
     sort +4  $year"_04_0"$count >$year"_04_0"$count"s"
   else
     grep "Apr $count " Formated$year > $year"_04_"$count
     sort +4  $year"_04_"$count  >$year"_04_"$count"s"
   fi


   count=`expr $count - 1`
done

count=31
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "May $count " Formated$year > $year"_05_0"$count
     sort +4  $year"_05_0"$count >$year"_05_0"$count"s"
   else
     grep "May $count " Formated$year > $year"_05_"$count
     sort +4  $year"_05_"$count  >$year"_05_"$count"s"
   fi

   count=`expr $count - 1`
done

count=30
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Jun $count " Formated$year > $year"_06_0"$count
     sort +4  $year"_06_0"$count >$year"_06_0"$count"s"
   else
     grep "Jun $count " Formated$year > $year"_06_"$count
     sort +4  $year"_06_"$count  >$year"_06_"$count"s"
   fi


   count=`expr $count - 1`
done

count=31
while test $count -gt 0
 do
   echo $count

   if test $count -lt 10
   then
     grep "Jul $count " Formated$year > $year"_07_0"$count
     sort +4  $year"_07_0"$count >$year"_07_0"$count"s"
   else
     grep "Jul $count " Formated$year > $year"_07_"$count
     sort +4  $year"_07_"$count  >$year"_07_"$count"s"
   fi


   count=`expr $count - 1`
done

count=31
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Aug $count " Formated$year > $year"_08_0"$count
     sort +4  $year"_08_0"$count >$year"_08_0"$count"s"
   else
     grep "Aug $count " Formated$year > $year"_08_"$count
     sort +4  $year"_08_"$count  >$year"_08_"$count"s"
   fi


   count=`expr $count - 1`
done

count=30
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Sep $count " Formated$year > $year"_09_0"$count
     sort +4  $year"_09_0"$count >$year"_09_0"$count"s"
   else
     grep "Sep $count " Formated$year > $year"_09_"$count
     sort +4  $year"_09_"$count  >$year"_09_"$count"s"
   fi


   count=`expr $count - 1`
done

count=31
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Oct $count " Formated$year > $year"_10_0"$count
     sort +4  $year"_10_0"$count >$year"_10_0"$count"s"
   else
     grep "Oct $count " Formated$year > $year"_10_"$count
     sort +4  $year"_10_"$count  >$year"_10_"$count"s"
   fi

   count=`expr $count - 1`
done

count=30
```

```
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Nov $count " Formated$year > $year"_11_0"$count
     sort +4  $year"_11_0"$count >$year"_11_0"$count"s"
   else
     grep "Nov $count " Formated$year > $year"_11_"$count
     sort +4  $year"_11_"$count  >$year"_11_"$count"s"
   fi

   count=`expr $count - 1`
done

count=31
while test $count -gt 0
do
   echo $count

   if test $count -lt 10
   then
     grep "Dec $count " Formated$year > $year"_12_0"$count
     sort +4  $year"_12_0"$count >$year"_12_0"$count"s"
   else
     grep "Dec $count " Formated$year > $year"_12_"$count
     sort +4  $year"_12_"$count  >$year"_12_"$count"s"
   fi

   count=`expr $count - 1`
done

mv Formated$year $year"Formated"
cat $year*s >$year"Sorted"

#Add week to filename
grep "Dec $count " Formated$year |awk '{a=$2}END {print a$year}'

# no non-s files :
grep "Dec $count " Formated$year |sort +4 >$year"_12_0"$count"s"
```

MyBatchS

```
year=2000

#cat $year* >Merge$year
#sed -f sedscript Merge$year>Filtered$year
#rm Merge$year
#awk '{printf "%6s %3s %3s %-2s %8s %4s %-4s %-16s %6s %2s %2s \n",
$1, $2, $3, $4, $5, $6, $7, $8, $9, $11, $13}' Filtered$year >
Formated$year
#rm Filtered$year

count=31
while test $count -gt 0
do
   echo $count
   grep "Jan $count " Formated$year > $year"_1_"$count
   count=`expr $count - 1`
done

count=29
while test $count -gt 0
do
   echo $count
   grep "Feb $count " Formated$year > $year"_2_"$count
   count=`expr $count - 1`
done

count=31
while test $count -gt 0
do
   echo $count
   grep "Mar $count " Formated$year > $year"_3_"$count
   count=`expr $count - 1`
done

count=30
while test $count -gt 0
do
   echo $count
   grep "Apr $count " Formated$year > $year"_4_"$count
   count=`expr $count - 1`
done
```

```
count=31
while test $count -gt 0
do
   echo $count
   grep "May $count " Formated$year > $year"_5_"$count
   count=‘expr $count - 1‘
done

count=30
while test $count -gt 0
do
   echo $count
   grep "Jun $count " Formated$year > $year"_6_"$count
   count=‘expr $count - 1‘
done

count=31
while test $count -gt 0
do
   echo $count
   grep "Jul $count " Formated$year > $year"_7_"$count
   count=‘expr $count - 1‘
done

count=31
while test $count -gt 0
do
   echo $count
   grep "Aug $count " Formated$year > $year"_8_"$count
   count=‘expr $count - 1‘
done

count=30
while test $count -gt 0
do
   echo $count
   grep "Sep $count " Formated$year > $year"_9_"$count
   count=‘expr $count - 1‘
done

count=31
```

```
while test $count -gt 0
do
   echo $count
   grep "Oct $count " Formated$year > $year"_10_"$count
   count=‘expr $count - 1‘
done

count=30
while test $count -gt 0
do
   echo $count
   grep "Nov $count " Formated$year > $year"_11_"$count
   count=‘expr $count - 1‘
done

count=31
while test $count -gt 0
do
   echo $count
   grep "Dec $count " Formated$year > $year"_12_"$count
   count=‘expr $count - 1‘
done
```

chex

```
#!/bin/sh
#make a file executable
chmod u+x $1
echo $1 is now exectuable;
ls -l $1
```

Appendix B
Shell Scripts for Preliminary Data Analysis

B.1 Introduction

We list seven shell script files used for preliminary data analysis. We name this part as preliminary because it does not involve any statistical analysis. These scripts serve for three purpose: first, to help us customize counting the number of calls requested summarised by various criteria such as year, month, week, hour, half hour, quarter hour and even down to a second (as shown in Figure 2.4 in Chapter 2); second, to compute interarrival time and service time, and produce test statistic R_{ij} in Chapter 4.

B.2 List of Script Files

MyMonthDay Compute total daily calls throughout the specified month.

MyWeekDay Compute total daily calls throughout each of the four weeks for the specified month.

MyDayHour Compute total hourly calls throughout the specified day.

MyDayHalf Compute total half-hour calls throughout the specified day.

MyDayQuarter Compute total quarter-hour calls throughout the specified day.

MyPaste Paste the data of each file as a column of the aggregate summary. The summary is of matrix form and ready to feed in MATLAB for further analysis.

MyCount Produce the customized counts summarised by various time intervals, for example, 300 seconds (i.e., 5 minutes). We separate each daily counts into a column and save these columns into one monthly count file.

MyTestData Compute interarrival time and service time .

MyRTest produce test statistic R_{ij} based on discussion in Section 4.1.3, ChaptertestNHPP.

MyMonthDay

```
year=$1
#python ,bash
#awk  '$4 == "2000_11_1" ,$4 =="2000_11_12"' NOofDayhelp

#### Create YearDay2000,YearMonth2000,MonthDay* MonthDayPasted2000
###from 2000_*s
#### Include argument such as 2000

wc -l *s > YearDay$year

month=1 while test $month -lt 13 do
   echo $month

   if test $month -lt 10
   then
     grep "$year"_0"$month" YearDay$year
     |awk '{n += $1} END{print n}' >>YearMonth$year
     grep "$year"_0"$month" YearDay$year
     |awk '{print $1}' >MonthDay$year"_0"$month
   else
     grep "$year"_"$month" YearDay$year
     |awk '{n += $1} END{print n}'  >>YearMonth$year
     grep "$year"_"$month" YearDay$year
     |awk '{print $1}' >MonthDay$year"_"$month
   fi
   month=‘expr $month + 1‘

done
paste MonthDay* > MonthDayPasted$year
```

MyWeekDay

```
year=$1

####input: 2000Sorted
####output: WkDay2000


sed '/Jan 1 / d' $year"Sorted" |sed '/Jan 2 / d' >WeekSorted$year
#sed '/Jan 2 / ' WeekSorted$year >WeekSorted$year

fgrep Mon WeekSorted$year > TWeMon2000
fgrep Tue WeekSorted$year > TWeTue2000
fgrep Wed WeekSorted$year > TWeWed2000
fgrep Thu WeekSorted$year > TWeThu2000
fgrep Fri WeekSorted$year > TWeFri2000
fgrep Sat WeekSorted$year > TWeSat2000
fgrep Sun WeekSorted$year > TWeSun2000

awk '{print $3,$4}' TWeMon2000 | uniq -c
|awk '{printf "%3s %2s %6s\n", $2,$3,$1}'> TWe1Mon2000
awk '{print $3,$4}' TWeTue2000 | uniq -c
|awk '{printf "%6s\n",$1}'> TWe2Tue2000
awk '{print $3,$4}' TWeWed2000 | uniq -c
|awk '{printf "%6s\n",$1}'> TWe3Wed2000
awk '{print $3,$4}' TWeThu2000 | uniq -c
|awk '{printf "%6s\n",$1}'> TWe4Thu2000
awk '{print $3,$4}' TWeFri2000 | uniq -c
|awk '{printf "%6s\n",$1}'> TWe5Fri2000
awk '{print $3,$4}' TWeSat2000 | uniq -c
|awk '{printf "%6s\n",$1}'> TWe6Sat2000
awk '{print $3,$4}' TWeSun2000 | uniq -c
|awk '{printf "%6s\n",$1}'> TWe7Sun2000

paste TWe[1-7]* > WeekDay$year
rm TW*
```

MyDayHour

```
day=$1
#### must add one argument-datefile
####output:DayHour$day

rm DayHour$day

hour=0
while test $hour -lt 24
do
   echo $hour
   if test $hour -lt 10
   then
     grep " 0$hour"":" $day |awk 'BEGIN { n=0 }{ n++ } END{print n}'
>> DayHour$day
   else
     grep " $hour"":" $day |awk 'BEGIN { n=0 }{ n++ } END{print n}'
>> DayHour$day
   fi
   hour=`expr $hour + 1`
done
more DayHour$day
```

MyDayHalf

```
####must: add one argument -datefile ####output: DayHalfHour$day
rm Temp rm DayHalfHour* num=0 day=$1 hour=0 while test $hour -lt
24 do
   echo $hour
   if test $hour -lt 10
   then
      min=0
      while test $min -lt 60
      do
       if test $min -lt 10
       then
          grep " 0$hour"":0$min"":" $day |wc -l >>Temp
       else
          grep " 0$hour"":$min"":" $day |wc -l >>Temp
          if test $min -eq 29
          then
             awk '{n+=$1} END{print n}' Temp >>DayHalfHour$day
             rm Temp
          fi

          if test $min -eq 59
          then
             awk '{n+=$1} END{print n}' Temp >>DayHalfHour$day
             rm Temp
          fi
        fi
        min=`expr $min + 1`
      done
   else
      min=0
      while test $min -lt 60
      do
       if test $min -lt 10
       then
          grep " $hour"":0$min"":" $day |wc -l >>Temp
       else
          grep " $hour"":$min"":" $day |wc -l >>Temp

          if test $min -eq 29
          then
             awk '{n+=$1} END{print n}' Temp >>DayHalfHour$day
             rm Temp
          fi

          if test $min -eq 59
          then
             awk '{n+=$1} END{print n}' Temp >>DayHalfHour$day
             rm Temp
          fi
       fi
       min=`expr $min + 1`
      done
   fi
   hour=`expr $hour + 1`
done
```

MyDayQuarter

```
rm Temp
####must: add one argument -datefile
####output:DayQuHour$day

#rm DayHalfHour*
num=0
day=$1
hour=0
while test $hour -lt 24
do
   echo $hour
   if test $hour -lt 10
   then
      min=0
      while test $min -lt 60
      do
       if test $min -lt 10
       then
          grep " 0$hour"":0$min"":" $day |wc -l >>Temp
       else
          grep " 0$hour"":$min"":" $day |wc -l >>Temp

          if test $min -eq 14
          then
             awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
             rm Temp
          fi

          if test $min -eq 29
          then
             awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
             rm Temp
          fi

          if test $min -eq 44
          then
             awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
             rm Temp
          fi
```

```
        if test $min -eq 59
        then
           awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
           rm Temp
        fi


      fi
      min=`expr $min + 1`
    done
else
   min=0
   while test $min -lt 60
   do
    if test $min -lt 10
    then
       grep " $hour"":0$min"":" $day |wc -l >>Temp
    else
       grep " $hour"":$min"":" $day |wc -l >>Temp

       if test $min -eq 14
       then
          awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
          rm Temp
       fi

       if test $min -eq 29
       then
          awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
          rm Temp
       fi

       if test $min -eq 44
       then
          awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
          rm Temp
       fi

       if test $min -eq 59
       then
          awk '{n+=$1} END{print n}' Temp >>DayQuHour$day
          rm Temp
          fi
       fi
       min=‘expr $min + 1‘
      done
   fi
   hour=‘expr $hour + 1‘
done
```

MyPaste

```
yr=$1
Month=$2
if test $Month -lt 10
then

paste "DayQuHour"$yr"_0"$Month"_0"[1-9]"s" >paste09
paste "DayQuHour"$yr"_0"$Month"_1"[0-9]"s" >paste19
paste "DayQuHour"$yr"_0"$Month"_2"[0-9]"s" >paste29
paste "DayQuHour"$yr"_0"$Month"_"3*"s" >paste30
paste paste09 paste19 paste29 paste30 >DayQuHour$yr"_0"$Month

else

paste "DayQuHour"$yr"_"$Month"_0"[1-9]"s" >paste09
paste "DayQuHour"$yr"_"$Month"_1"[0-9]"s" >paste19
paste "DayQuHour"$yr"_"$Month"_2"[0-9]"s" >paste29
paste "DayQuHour"$yr"_"$Month"_"3*"s" >paste30
paste paste09 paste19 paste29 paste30 >DayQuHour$yr"_"$Month

fi

rm paste*
```

MyCount

```
yr=$1
TimeLen=$2 # in terms of seconds
##input: *s files
##output:$yrcount_$2
rm count$yr"_"*$TimeLen
```

```
Mon=1
while test $Mon -lt 13
do

 if test $Mon -lt 10
 then
  day=1
  while test $day -lt 32
  do
    echo $day
    if test $day -lt 10
    then
      #sed /root/d $yr"_0"$Mon"_0"$day"s" > $yr"_0"$Mon"_0"$day"sn"
      leftSec=0
      rightSec=`expr $TimeLen - 1`
      while test $rightSec -lt 86400
      do
        awk '$13-lf>=0 && $13-rf<=0 '
        lf=$leftSec rf=$rightSec $yr"_0"$Mon"_0"$day"s"
        |awk 'BEGIN { n=0 }{ n++ } END{print n}' >> Ctemp0$day
        leftSec=`expr $leftSec + $TimeLen`
        rightSec=`expr $rightSec + $TimeLen`
      done
    else
      #sed /root/d $yr"_0"$Mon"_"$day"s" > $yr"_0"$Mon"_"$day"sn"
      leftSec=0
      rightSec=`expr $TimeLen - 1`
      while test $rightSec -lt 86400
      do
        awk '$13-lf>=0 && $13-rf<=0 '
        lf=$leftSec rf=$rightSec $yr"_0"$Mon"_"$day"s"
        |awk 'BEGIN { n=0 }{ n++ } END{print n}' >> Ctemp$day
        leftSec=`expr $leftSec + $TimeLen`
        rightSec=`expr $rightSec + $TimeLen`
      done
    fi
    day=`expr $day + 1`
  done
  paste "Ctemp0"[1-9] >paste09
  paste "Ctemp1"[0-9] >paste19
  paste "Ctemp2"[0-9] >paste29
  paste Ctemp3* >paste30
  paste paste09 paste19 paste29 paste30 >count$yr"_0"$Mon"_"$TimeLen
  rm Ctemp*
  rm paste*
 else
  day=1
  while test $day -lt 32
  do
    echo $day
    if test $day -lt 10
    then
      #sed /root/d $yr"_"$Mon"_0"$day"s" > $yr"_"$Mon"_0"$day"sn"
      leftSec=0
      rightSec=‘expr $TimeLen - 1‘
      while test $rightSec -lt 86400
      do
        awk '$13-lf>=0 && $13-rf<=0 '
        lf=$leftSec rf=$rightSec $yr"_"$Mon"_0"$day"s"
        |awk 'BEGIN { n=0 }{ n++ } END{print n}' >> Ctemp0$day
        leftSec=‘expr $leftSec + $TimeLen‘
        rightSec=‘expr $rightSec + $TimeLen‘
      done
    else
      leftSec=0
      rightSec=‘expr $TimeLen - 1‘
      while test $rightSec -lt 86400
      do
        awk '$13-lf>=0 && $13-rf<=0 '
        lf=$leftSec rf=$rightSec $yr"_"$Mon"_"$day"s"
        |awk 'BEGIN { n=0 }{ n++ } END{print n}' >> Ctemp$day
        leftSec=‘expr $leftSec + $TimeLen‘
        rightSec=‘expr $rightSec + $TimeLen‘
      done
    fi
    day=‘expr $day + 1‘
  done
  paste "Ctemp0"[1-9] >paste09
  paste "Ctemp1"[0-9] >paste19
  paste "Ctemp2"[0-9] >paste29
  paste Ctemp3* >paste30
  paste paste09 paste19 paste29 paste30 >count$yr"_"$Mon"_"$TimeLen
  rm Ctemp*
  rm paste*
 fi
 Mon=`expr $Mon + 1`
 echo $Mon
done
```

MyTestData

```
yr=$1
Mon=$2
FromHour=$3
FromMin=$4
EndHour=$5
EndMin=$6
##input: *s files
##output: *R, *InterArr, *Service
##Note: Before running,Delete relevant *R, *InterArr, *Service files

leftSec=`expr $FromHour \* 3600 + $FromMin \* 60`
rightSec=`expr $EndHour \* 3600 + $EndMin \* 60 + 59`

Len=`expr $rightSec - $leftSec`
Len=`expr $Len + 1`

echo $leftSec
echo $rightSec
echo $Len

day=1 while test $day -lt 32
do
  if test $Mon -lt 10
  then
    if test $day -lt 10
    then
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf, $13, $12}'
       lf=$leftSec rf=$rightSec $yr"_0"$Mon"_0"$day"s" >Tep

      awk '{print $3}' Tep >>
$yr"_0"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"Service"
      awk '{if (NR == 1){prev=$2} else
            {tem=$2-prev;print tem; prev=$2;}}' Tep >>
$yr"_0"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"InterArr"
```

```
      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1 } else
      {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>$yr"_0"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"R"
      #$yr_0$Mon"_0"$day"s_"$FromHour"_"$FromMin"_"$EndMin

    else
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf, $13, $12}'
      lf=$leftSec rf=$rightSec $yr"_0"$Mon"_"$day"s" >Tep

      awk '{print $3}' Tep >>
$yr"_0"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"Service"
      awk '{if (NR == 1){prev=$2} else
      {tem=$2-prev;print tem; prev=$2;}}'
Tep >> $yr"_0"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"InterArr"

      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1} else
       {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>$yr"_0"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"R"
      #$yr_0$Mon"_"$day"s_"$FromHour"_"$FromMin"_"$EndMin
    fi


  else
    if test $day -lt 10
    then
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf, $13, $12}'
      lf=$leftSec rf=$rightSec $yr"_"$Mon"_0"$day"s" >Tep

      awk '{print $3}' Tep >>
       $yr"_"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"Service"
      awk '{if (NR == 1){prev=$2} else
      {tem=$2-prev;print tem; prev=$2;}}' Tep >>
 $yr"_"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"InterArr"

      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1} else
```

```
       {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>$yr"_"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"R"
      #$yr_$Mon"_0"$day"s_"$FromHour"_"$FromMin"_"$EndMin

    else
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf, $13, $12}'
       lf=$leftSec rf=$rightSec $yr"_"$Mon"_"$day"s" >Tep

      awk '{print $3}' Tep >>
       $yr"_"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"Service"
      awk '{if (NR == 1){prev=$2} else
       {tem=$2-prev;print tem; prev=$2;}}' Tep >>
        $yr"_"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"InterArr"

      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1} else
       {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>$yr"_"$Mon"_"$FromHour"_"$FromMin"_"$EndHour"_"$EndMin"R"
      #$yr_$Mon"_"$day"s_"$FromHour"_"$FromMin"_"$EndMin
    fi
  fi

  day=`expr $day + 1`
done
rm Tep
rm Temp
```

MyRTest

```
Mon=$1
Hour=$2
FromMin=$3
EndMin=$4
leftSec=`expr $Hour \* 3600 + $FromMin \* 60`
rightSec=`expr $Hour \* 3600 + $EndMin \* 60 + 59`

Len=`expr $rightSec - $leftSec`
Len=`expr $Len + 1`

echo $leftSec
echo $rightSec
echo $Len

day=1 while test $day -lt 32 do
  if test $Mon -lt 10
  then
    if test $day -lt 10
    then
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf}'
      lf=$leftSec rf=$rightSec 2000_0$Mon"_0"$day"s" >Tep
      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1 } else
       {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>2000_0$Mon"_"$Hour"_"$FromMin"_"$EndMin
      #2000_0$Mon"_0"$day"s_"$Hour"_"$FromMin"_"$EndMin

    else
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf}'
      lf=$leftSec rf=$rightSec 2000_0$Mon"_"$day"s" >Tep
      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1} else
     {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>2000_0$Mon"_"$Hour"_"$FromMin"_"$EndMin
      #2000_0$Mon"_"$day"s_"$Hour"_"$FromMin"_"$EndMin
    fi


  else
    if test $day -lt 10
    then
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf}'
      lf=$leftSec rf=$rightSec 2000_$Mon"_0"$day"s" >Tep
      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1} else
       {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>2000_$Mon"_"$Hour"_"$FromMin"_"$EndMin
      #2000_$Mon"_0"$day"s_"$Hour"_"$FromMin"_"$EndMin

    else
```

```
      awk '$13-lf>=0 && $13-rf<=0 {print $13 - lf}'
      lf=$leftSec rf=$rightSec 2000_$Mon"_"$day"s" >Tep
      awk 'BEGIN { n=0 }{ n++ } END{printf "%s \n",n}' Tep > Temp
      awk 'BEGIN { n=0 }{ n++;printf "%s \n",$1}' Tep >>Temp
      awk 'BEGIN {j=1;prev=0}{if (NR == 1){ n=$1} else
       {tem=log(L-prev)-log(L-$1);print tem*(n+1-j); prev=$1;j++;}}'
L=$Len Temp >>2000_$Mon"_"$Hour"_"$FromMin"_"$EndMin
      #2000_$Mon"_"$day"s_"$Hour"_"$FromMin"_"$EndMin
    fi
  fi

  day=`expr $day + 1`
done
```

Appendix C
AD: An R function for A-D Exponential Test

C.1 Description

The A-D test for assessing the goodness-of-fit of the sample to the exponential distribution is written in R [29] as **AD**. Once it is called, the method shall return the message depending upon the given sample data accept or reject the exponential distribution with preset 5% level of significance. This level of significance can be easily customized if needs to.

Code for AD

```
#Implement A-D test for exponential distribution
AD=function(x){

y1=sort(x)
y2=sort(x,decreasing=T)
rate=1/mean(x)
N=length(x)
i=1:N

A_square=-N-sum((2*i-1)/N*(log(pexp(y1, rate))+log(1-pexp(y2,rate))))
A_square_star=A_square*(1+0.6/N)
if (A_square_star<=1.341)
print("A-D test passes, at the 5% level of significance")
    else print("A-D test fails, at the 5% level of significance")

}
```

Appendix D
expTest: An R Function for Goodness-of-fit Tests of Exponential Distribution

D.1 Description

expTest provides a set of tools to evaluate the goodness-of-fit to the exponential distribution is written with methods implemented in R libraries such as **truehist** from library MASS and **qq.plot** from library cat.

D.2 List of Methods

truehist Plot the density of the data.

ks.test Perform the K-S test for the exponential distribution.

AD Call the implemented A-D test in Appendix C.

qq.plot Compare the sample to the exponential distribution using Q-Q plot.

Code for expTest

```
expTest=function(feed){

library(MASS)
library(car)
#library car needs to be installed

data=feed
N=length(data)
###Density plot
truehist(data,nbins=2*sqrt(length(data)),xlab="Second", ylab="Density")
browser()

### See alternative MATLAB function 'mypdfcdf.m'
###for comparing EDF to exponential CDF
###EDF vs exponential CDF
x=seq(0,max(data),length=100*max(data))
#use 60*100=6000 points to smooth the CDF curve
Pr= pexp(x, 1/mean(x))
data.frame(x = x, Pr = Pr)
plot(c(-0.2, x), c(0, Pr), type = "s", xlab = "Second", ylab = "P",
        #main = "Cumulative distribution function",
        las = 1)
#-0.2 is a safe start to draw the zero probability.
#It can be any arbitrary negative value
#browser()
data.ecdf <- ecdf(data)
#summary(data.ecdf)
plot(data.ecdf, verticals= TRUE, do.p = FALSE)
#browser()
###K-S test
p_value=ks.test(data,"pexp",1/mean(data))$p.value

###A-D test at the 5% level of significance
source("AD")
AD(data)

###Q-Q plot

qq.plot(data,distribution='exp',rate=1/mean(data),#envelope=FALSE,
                main = "Exponential Q-Q Plot", col='blue',
                ylab = "Sample Quantiles",
                xlab = "Exponential Quantiles")


list(KS.p.value=p_value, sample.size=N )
#CV=sd(data)/mean(data)

}#end
```

Appendix E
mypdfcdf: An MATLAB Function for Goodness-of-fit Tests of Exponential Distribution

E.1 Description

As alternative to **expTest** in R, we have also written a MATLAB based test of goodness-of-fit, **mypdfcdf** to a class of distributions: normal, exponential and lognormal.

Code for mypdfcdf

```
function mypdfcdf(x,dist)
%dist==0 exponential
%dist==1 lognormal
%dist==2 normal
%Task 1: Compare the data pdf with the dist pdf
%Task 2: Compare the data cdf with the dist cdf

x=sort(x);
n=length(x);
ave=mean(x)
stdv=std(x)

[f,xf,u] = ksdensity(x);
plot(xf,f)
hold on
title('Density estimate ')
if dist == 0
    y=exppdf(x,ave);
    plot(x,y,'k')
elseif dist == 1
    if min(x)>0
       xx=log(x);%for lognormal
    else
       s='Note,x should be greater than 0'
       %return
       xx=x(x>0);
```

```
        xx=log(xx);
    end
    lave=mean(xx);
    lstdv=std(xx);
    y=lognpdf(x,lave,lstdv);
    plot(x,y,'k')
else
    y=normpdf(x,ave,stdv);
    plot(x,y,'k')
end

%[N,X]=hist(x,100);
%w=(x(length(x))-x(1))/100
%N=N/length(x);
%N=N/w;
%plot(X,N,'g')

figure
hist(x,100)
title('Histgraph')


figure
[f,xf] = ecdf(x);
stairs(xf,f,'g')
title('cdf graph')
hold on
%cdfplot(x)
if dist == 0
    y=expcdf(x,ave);
    plot(x,y,'k')
elseif dist == 1
    y=logncdf(x,lave,lstdv);
    plot(x,y,'k')
else
    y=normcdf(x,ave,stdv);
    plot(x,y,'k')
end
```

Appendix F
LIN.RATE: An R Function for Linear Rate Estimation and LRT

F.1 Description

Estimation algorithm based on linear intensity rate and LRT are implemented in R. The R function returns a set of estimates and the result of the asymptotic LRT.

F.2 List of Outputs

summary_OLS Summary of linear regression using OLS.

a Intercept of the regression line (OLS).

b Slope of the regression line (OLS).

summary_IWLS Summary of linear regression using IWLS.

a_IWLS Intercept of the regression line (IWLS).

b_IWLS Slope of the regression line (IWLS).

Iteration_IWL Number of iterations for the WLS to converge.

a_ML Intercept estimated by MLE

b_ML Slope estimated by MLE

LRT_Message LRT test result

Code for LIN.RATE

```
LIN.RATE=function(){

rm(list = ls())

### read data
####WILD CARD FOR A BATCH OF FILES#####
prefix="http://math.usask.ca/~sol573/UconnectData/count"
year=2000
m=readline("Enter the month, e.g. 3: ")
if (as.numeric(m)>12|as.numeric(m)<1)
    stop ("month has to be from 1 to 12")
if (as.numeric(m)<10) month=paste(0,m,sep="") else month=m
i=readline("Enter the interval in which arrival is summarised
(all possible values are '60', '120','300','600'): ")
interval=as.numeric(i)
if (interval==60|interval==120|interval==300|interval==600)
interval=interval else
 stop ("All possible values are '60', '120','300','600'")
#weekday=date
date=5 #Wednesdays
hour=10

address=paste(paste(prefix,year,sep=""),month,interval,sep="_")
TS=read.table(address)
tol=1.e-6 #global tolerance
#5% as p value cutoff

########## OLS ###########

###Regression with single realization

T=60 #mins
N=T/(interval/T)
x=1:N
start=hour*(3600/interval)+1
end=start+(N-1)
wd=TS[start:end,c(date)]
x=(x-1/2)*T/N#the rescaling has no statistical difference
y=wd
z=lm(y~x)
```

```
summary(z)
plot(x,y)
abline(z)
###attributes(z)
alpha=coef(z)[1]
beta=coef(z)[2]
a=N/T*alpha
b=N/T*beta

########## IWLS ###########

a_IWLS=a
b_IWLS=b
counter=0
repeat
{
    counter=counter+1
    oldA=a_IWLS
    oldB=b_IWLS
    lamda=(a_IWLS+b_IWLS*x)*T/N
    w=N/lamda/sum(1/lamda)
    z_IWLS=lm(y~x,weights=w)
    alpha=coef(z_IWLS)[1]
    beta=coef(z_IWLS)[2]
    a_IWLS=N/T*alpha
    b_IWLS=N/T*beta
    #tolorence=1.e-6
    if( abs(oldA-a_IWLS)/abs(oldA)<tol
     |abs(oldB-b_IWLS)/abs(oldB)<tol) break
}
summary(z_IWLS)
plot(x,y)
abline(z_IWLS)
counter
#Up to the 6th decimal (i.e. tolerance) usually after 6 iterations

########## ML ###########
S=sum(wd)
f <- function(a_ML) sum(wd/(a_ML*T/2+x*(S/T-a_ML)))-2
#as a=S/T is always a root, we use the tolerance 1.e-10
lEnd=f(0)
rEnd=f(S/T*(1-tol))
```

```
if (rEnd*lEnd<0)
{
    sol=uniroot(f, c(0,S/T*(1-tol)))
    a_ML=sol$root
    b_ML=2*(S-a_ML*T)/T^2
    }   else


########## Likelihood Ratio Test ###########

lin_rate=T/N*(a_ML+b_ML*x)
chi_square=-2*(-sum(lin_rate)+
sum(wd*log(lin_rate))+sum(wd)-sum(wd*log(wd)))
p_value=1-pchisq(chi_square,df=N-2)
if (p_value>=0.05) msg="LRT passes" else {msg="LRT fails"}


###Regression with 4 (MULTIPLE) realizations
wds=TS[start:end,c(date,date+7,date+7+7,date+7+7+7)]
wds=c(wds[,c(1)],wds[,c(2)],wds[,c(3)],wds[,c(4)])
x=1:N
x=c(rep(x,4)) #4 weekdays or x=c(rep(1:12,4))
x=(x-1/2)*T/N

y=wds
z=lm(y~x)
summary(z)
#plot(x,y)
abline(z)


#print(c("a_ML is",a_ML))
#cat("a_ML is",a_ML,"\n")

list(summary_OLS=summary(z),a=a,b=b,summary_IWLS=summary(z_IWLS),
 a_IWLS=a_IWLS,b_IWLS=b_IWLS,Iteration_IWLS=counter,
    a_ML=a_ML,b_ML=b_ML, LRT_Message=msg)
}
```

Appendix G
mysin: An MATLAB Function for Solving MLE for Sinusoidal Rate

G.1 Description

mysin solves Equations 3.36 in Chapter 3 for ML estimates. We choose to feed the estimates produced by OLS as initial value.

G.2 Numerical Algorithm Review for fsolve

According to the optimazation toolbox user's guide for MATLAB [4], by default fsolve uses the medium-scale algorithm and the trust-region dogleg method which is a variant of the Fortran Powell dogleg method discussed in [30].

Code for expTest

```
function F = mysin(V)

%Initial value by a scaled root if use OLS estimators such that,
%V=coefficients(z_IWLS)*N/T
c=2*pi/60; %same as in OLS and IWLS
x=[ 2.5 7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5 57.5];
y=[  7  8  4  9  7  9 11 13  9  8  6 10];
T=60;%set 1 hour
N=length(x);
F(1)=sum(y./(V(1)+V(2)*sin(c*x)+V(3)*cos(c*x)))-T;
F(2)=sum(y.*sin(c*x)./(V(1)+V(2)*sin(c*x)+V(3)*cos(c*x)))-
    T/N*sum(sin(c*x));
F(3)=sum(y.*cos(c*x)./(V(1)+V(2)*sin(c*x)+V(3)*cos(c*x)))-
    T/N*sum(cos(c*x));
```

Appendix H
SIN.RATE: An R Function for Sinusoidal Rate Estimation and LRT

H.1 Description

Estimation algorithm based on sinusoidal rate and LRT are implemented in R. **SIN.RATE** returns OLS, IWLS and ML estimates and superimposes fitted lines based on these estimates. In addition, it also performs the asymptotic LRT.

H.2 List of Outputs

summary_OLS Summary of regression using OLS.

a_OLS Intercept of the regression line (OLS).

b1_OLS, b2_OLS Slopes of the regression line (OLS).

summary_IWLS Summary of regression using IWLS.

a_IWLS Intercept of the regression line (IWLS).

b1_IWLS, b2_IWLS Slope of the regression line (IWLS).

Iteration_IWLS Number of iterations for the IWLS to converge.

a_ML Intercept estimated by MLE

b1_ML, b2_ML Slope estimated by MLE

LRT_Message LRT test result

Code for SIN.RATE

```
SIN.RATE=function(){

rm(list = ls())

### read data

TS=read.table("count2000_03_300")
interval=300
date=8
time=16
tol=1.e-6 #global tolerance
#5% as p value cutoff

########## OLS ###########

###Regression with single realization
duration=1
T=60*duration #mins
c=2*pi/60
#assuming hourly pattern, cycle=31.5 mins. i.e., if c=1, 1*2pi*5min=31.5
N=60*T/interval
#N=T/(interval/T)
x=1:N
start=time*(3600/interval)+1
end=start+(N-1)
wd=TS[start:end,c(date)]
x=(x-1/2)*T/N
y=wd
z=lm(y~sin(c*x)+cos(c*x))
summary(z)
plot(x,y, xlab="Minute", ylab="Number of Calls",
                main="Selected Hour" )
#abline(z)
###attributes(z)
alpha=coef(z)[1]
beta1=coef(z)[2]
beta2=coef(z)[3]
alpha_OLS=alpha
beta1_OLS=beta1
beta2_OLS=beta2
```

```
points(x,alpha+beta1*sin(c*x)+beta2*cos(c*x),
                type="l",col='dark blue',cex=0.5, pch=25)
a=N/T*alpha
b1=N/T*beta1
b2=N/T*beta2

########## IWLS ###########

a_IWLS=a
b1_IWLS=b1
b2_IWLS=b2
counter=0
repeat
{
    counter=counter+1
    oldA=a_IWLS
    oldB1=b1_IWLS
    oldB2=b2_IWLS
    lamda=(a_IWLS+b1_IWLS*sin(c*x)+b2_IWLS*cos(c*x))*T/N
    w=N/lamda/sum(1/lamda)
    z_IWLS=lm(y~sin(c*x)+cos(c*x),weights=w)
    alpha=coef(z_IWLS)[1]
    beta1=coef(z_IWLS)[2]
    beta2=coef(z_IWLS)[3]

    a_IWLS=N/T*alpha
    b1_IWLS=N/T*beta1
    b2_IWLS=N/T*beta2
    #tolorence=1.e-6
    if( abs(oldA-a_IWLS)/abs(oldA)<tol
     |abs(oldB1-b1_IWLS)/abs(oldB1)<tol|
abs(oldB2-b2_IWLS)/abs(oldB2)<tol) break
}
summary(z_IWLS)
alpha_IWLS=coef(z_IWLS)[1]
beta1_IWLS=coef(z_IWLS)[2]
beta2_IWLS=coef(z_IWLS)[3]
plot(x,y, xlab="Minute", ylab="Number of Calls",
                main="Selected Hour" )
points(x,alpha_IWLS+beta1_IWLS*sin(c*x)+beta2_IWLS*cos(c*x),
                type="l",col='dark blue',cex=0.5, pch=25)
```

```
counter
#Up to the 6th decimal (i.e. tolerance) usually after 6 iterations

########## ML ###########
#to be solved using fsolve in MATLAB
#Use estimates by OLS as initial values
a_OLS=N/T*alpha_OLS
b1_OLS=N/T*beta1_OLS
b2_OLS=N/T*beta2_OLS
##Once we have the estimate by MATLAB,
alpha_ML=T/N*a_ML
beta1_ML=T/N*b1_ML
beta2_ML=T/N*b2_ML


plot(x,y,xlab="Minute", ylab="Number of Calls",
            main="Sinusoidal Rate")
lines(x,alpha_OLS+beta1_OLS*sin(c*x)+beta2_OLS*cos(c*x),
                lty=1,lwd=2, col="red")
lines(x,alpha_IWLS+beta1_IWLS*sin(c*x)+beta2_IWLS*cos(c*x),
lty=2, lwd=4,col="blue")
lines(x, alpha_ML+beta1_ML*sin(c*x)+beta2_ML*cos(c*x),
lty=3,lwd=3,col="black")
legend(locator(1), lty=c(1,2,3),lwd=3, col=c("red","blue","black"),
legend=c('OLS', 'IWLS','ML'))


########## Likelihood Ratio Test ###########

sin_rate=T/N*(a_ML+b1_ML*sin(c*x)+b2_ML*cos(c*x))
chi_square=-2*(-sum(sin_rate)+
sum(wd*log(sin_rate))+sum(wd)-sum(wd*log(wd)))
p_value=1-pchisq(chi_square,df=N-2)
if (p_value>=0.05) msg="LRT passes" else {msg="LRT fails"}


###Regression with 4 (MULTIPLE) realizations
wds=TS[start:end,c(date,date+7,date+7+7,date+7+7+7)]
wds=c(wds[,c(1)],wds[,c(2)],wds[,c(3)],wds[,c(4)])
x=1:N
x=c(rep(x,4)) #4 weekdays or x=c(rep(1:12,4))
x=(x-1/2)*T/N
```

```
y=wds
z=lm(y~x)
summary(z)
#plot(x,y)
abline(z)


list(summary_OLS=summary(z),a_OLS=a_OLS,b1_OLS=b1_OLS,b2_OLS=b2_OLS,
summary_IWLS=summary(z_IWLS), a_IWLS=a_IWLS,b1_IWLS=b1_IWLS,
b2_IWLS=b2_IWLS,Iteration_IWLS=counter,
    a_ML=a_ML,b1_ML=b1_ML,b2_ML=b2_ML, LRT_Message=msg)
}
```

Appendix I
SPLINE.FIT: An R Function for Spline Regression and Smoothing Spline

I.1 Description

This function fits the data using spline regression and smoothing spline. Summary information can be provided by calling **summary** and **plot**. Fitted lines by spline as well as parametric models are superimposed for model comparison.

Code for SIN.RATE

```
SPLINE.FIT=function(){

rm(list = ls())
library(splines)
library(fields)

### read data
TS=read.table("count2000_03_300")
interval=300
date=8
time=16
tol=1.e-6 #global tolerance
#5% as p value cutoff

########## OLS ###########
###Linear rate model
duration=1
T=60*duration #mins
c=2*pi/60
#assuming hourly pattern, cycle=31.5 mins. i.e., if c=1, 1*2pi*5min=31.5
N=60*T/interval
#N=T/(interval/T)
x=1:N
start=time*(3600/interval)+1
end=start+(N-1)
wd=TS[start:end,c(date)]
x=(x-1/2)*T/N
```

```
y=wd
z=lm(y~x)
summary(z)
plot(x,y,xlab="Minute", ylab="Number of Calls",
            main="Linear Regression by OLS"
)
abline(z, lwd=2, col="red")
###attributes(z)
alpha=coef(z)[1]
beta=coef(z)[2]
a=N/T*alpha
b=N/T*beta

###This is sinusoidal.
duration=1
T=60*duration #mins
c=2*pi/60
#assuming hourly pattern, cycle=31.5 mins. i.e., if c=1, 1*2pi*5min=31.5
N=60*T/interval
#N=T/(interval/T)
x=1:N
start=time*(3600/interval)+1
end=start+(N-1)
wd=TS[start:end,c(date)]
x=(x-1/2)*T/N
y=wd

z_sin=lm(y~sin(c*x)+cos(c*x))
summary(z_sin)
plot(x,y, xlab="Minute", ylab="Number of Calls",
                main="Selected Hour" )


alpha=coef(z_sin)[1]
beta1=coef(z_sin)[2]
beta2=coef(z_sin)[3]
points(x,alpha+beta1*sin(c*x)+beta2*cos(c*x),
                type="l",col='dark blue',cex=0.5, pch=25)
a=N/T*alpha
b1=N/T*beta1
b2=N/T*beta2
```

```
####regression spline####
mod.ns=lm(y~ns(x,df=7))
lines(x,predict(mod.ns))

fit.ns=predict(mod.ns, data.frame(x=minute), interval="confidence",se.fit=T)
plot(x,y,ylab="Number of Calls",xlab="Minute",
        main="Spline Regreesion with 95% CI")
        fit.mod.ns=predict(mod.ns, data.frame(x=x),
        interval="confidence",se.fit=T)
        #Draw piecewise linear fits of the valued modelled by
        #regression spline at the knots
        lines(x, fit.mod.ns$fit[,"fit"])
        lines(x,fit.mod.ns$fit[,"lwr"],lty=2)
        lines(x,fit.mod.ns$fit[,"upr"],lty=2)
minute=seq(min(x),max(x),len=200)#use 200 points to predict/fit
#i.e.,spline fit at 200 equally spaced points
lines(minute,fit.ns$fit[,"fit"])
lines(minute,fit.ns$fit[,"lwr"],lty=2)
lines(minute,fit.ns$fit[,"upr"],lty=2)

####sreg####

     fit<- sreg(x,y)
     summary( fit)
         plot(fit)                       # diagnostic plots of  fit
     predict( fit) # fit$fitted.values, predicted values at data points


       #fit.sreg=predict(fit,data.frame(x=minute))
       # finding approximate standard errors at observations
     SE<- fit$shat.GCV*sqrt(fit$diagA)
     # 95% CI
     Zvalue<-  qnorm(.0975)
     upper<- fit$fitted.values + Zvalue* SE
     lower<- fit$fitted.values - Zvalue* SE
     plot(x,y,ylab="Number of Calls",xlab="Minute",
            main="Smoothing Spline with 95% CI")
     lines( fit$predicted, lwd=2)
     matlines( fit$x,
     cbind( lower, upper), type="l", col=c( 2,2), lty=2)
```

```
##superimpose fits by models
plot(x,y,ylab="Number of Calls",xlab="Minute")
abline(z,col='red',lty=1)
lines(x,alpha+beta1*sin(c*x)+beta2*cos(c*x),
                type="l",col='blue',lty=6)
#lines(minute,predict(mod.ns, data.frame(x=minute)),col='black',lty=3)
#lines(x,predict(mod.ns),col='black',lty=3)
lines(fit$predicted,col='green',lty=4)
legend(locator(1), lty=c(1,6,4),lwd=3, col=c("red","blue","green"),
            legend=c('linear', 'sinusoidal','ss'))

}
```

Bibliography

1. T.W. Anderson and D.A. Darling, Asymptotic theory of certain goodness-of-fit criteria based on stochastic processes, *Annals of Mathematical Statistics*, 23:193-212, 1952.

2. T.W. Anderson and D.A. Darling, A test of goodness-of-fit, *Journal of the American Statistical Association*, 49:765-769, 1954.

3. R.L. Berger and G. Casella. *Statistical inference*. Duxbury, 2002.

4. M.A Branch, T. Coleman and A Grace. *Optimization toolbox for use with MATLAB*. The MathWorks, Inc., 1999.

5. L. Brown, N. Gans, A. Mandelbaum, A. Sakov, H.Shen, S. Zeltyn, and L. Zhao. Statistical analysis of a telephone call center: A queueing-science perspective. *Journal of the American Statistical Association*, 100:36-50, 2005.

6. R. Carroll and D. Ruppert. *Transformation and weighting in regression*. Chapman Hall, 1988.

7. E. Chlebus. Empirical validation of call holding time distribution in cellular communications systems. In *Proceedings of the 15th International Teletraffice Conference*, 1997.

8. M.M. Crawford, J.R. Wilson and S. Lee.. Modeling and simulation of a nonhomogeneous Poisson process with cyclic features. *Communications in Statistics-Simulation and Computation*, 20:777-809, 1991.

9. R.B. D'Agostino and M. A. Stephens. *Goodness-of-fit techniques*. Marcel Dekker, 1986.

10. K. Dawson. *The call center handbook*. Gilroy, 2001.

11. R.L. Eubank. *Spline smoothing and nonparametric regression*. Marcel Dekker, 1988.

12. Z. Feldman, A. Mandelbaum, W.A Massey and W. Whitt. Staffing of time-varying queues to achieve time-stable performance. Working paper, 2005.

13. S. Floyd and V. Paxson. Wide-area traffic: the failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 1994.

14. J. Fox. *An R and S-plus companion to applied regression.* SAGE Publications, 2002.

15. N. Gans, G. Koole and A. Mandelbaum. Telephone call centers: Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, 5:79-141, 2003.

16. L.V. Green, P.J Kolesar and J. Soares. Improving the SIPP approach for staffing service systems that have cyclic demands. *Operations Research*, 49:549-564, 2001.

17. P.J. Green and B.W. Silverman. *Nonparametric regression and generalized linear model.* Chapman and Hall, 1994.

18. D. Gross and C.M. Harris. *Fundamentals of queueing theory.* Wiley, 1998.

19. T.J. Hastie and R.J. Tibshirani. *Generalized additive models.* Chapman and Hall, 1990.

20. O.B. Jennings, A. Mandelbaum, W.A Massey and W. Whitt. Server staffing to meet time-varying demand. Management Science, 42:1383-1394, 1996.

21. G. Jongbloed and G. Koole. Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17:307-318, 2001.

22. W.D Kelton and A.M. Law *Simulation modeling and analysis.* McGraw-Hill, 1991.

23. L.M. Leemis. Nonparametric estimation of the cummulative intensity function for a nonhomogeneous Poisson Process. *Management Science*, 37: 886-900, 1991.

24. S. Li, R. Pyke and P. Wang. Detecting hackers using Poisson model measure. In *Proceedings of the 8th annual PIMS-MITACS industrial problem solving workshop*, forthcoming.

25. Ljung, G. M. and Box, On a measure of lack of fit in time series models, *Biometrika*, 65: 553-564, 1978.

26. A. Mandelbaum. Statistical analysis of an arrival process. Teaching note, Technion.
URL http://ie.technion.ac.il/serveng/Arr_theory.pdf, 2001.

27. W.A. Massey, G.A. Parker, and W. Whitt. Esimating the parameters of a nonhomogeneous Poisson process with a linear rate. *Telecommunications Systems*, 5:361-388, 1996.

28. P. McCullagh and J.A. Nelder FRS. *Generalized linear models.* Chapman and Hall, 1989.

29. R Development Core Team . R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org, 2005.

30. P. Rabinowitz. *Numerical methods for nonlinear algebraic equations.* Gordon and Breach Science Publishers, 1970.

31. S. Ross. *Stochastic processes.* Wiley, 1996.

32. D.L. Snyder. *Random point processes in time and space.* Springer-Verlag, 1991.

33. R. Srinivasan and J. Talim. Performance analysis of a call center with interacting voice response units. Technical Report, University of Saskatchewan, 2001.

34. W. Whitt. Stochastic models for the design and management of customer contact centers: Some research directions. Working paper, Columbia University, 2002.