

# Nearest Neighbour Algorithms for Forecasting Call Arrivals in Call Centers

Sandjai Bhulai, Wing Hong Kan, Elena Marchiori

August 12, 2005

## 1 Background and Motivation

Companies and governments often rely on call centers to provide service to their customers. The call center industry is of great economic interest; It is rapidly expanding, in terms of workforce and economic scope [1]. Because of its importance a lot of studies have been, and are still being, conducted on modelling call centers. These models are being developed to help understand the performance of call centers under different conditions. This understanding can then be used to cost effectively control the call center. Often call centers need to determine how to assign manpower (agents, in call center terminology) to reach predefined service level goals. These goals are often defined as, 80% of all calls answered within a waiting time of 20 seconds, or an average waiting time of  $x$  seconds. Call centers want to reach these goals with minimal costs (e.g., loan wages, overhead costs).

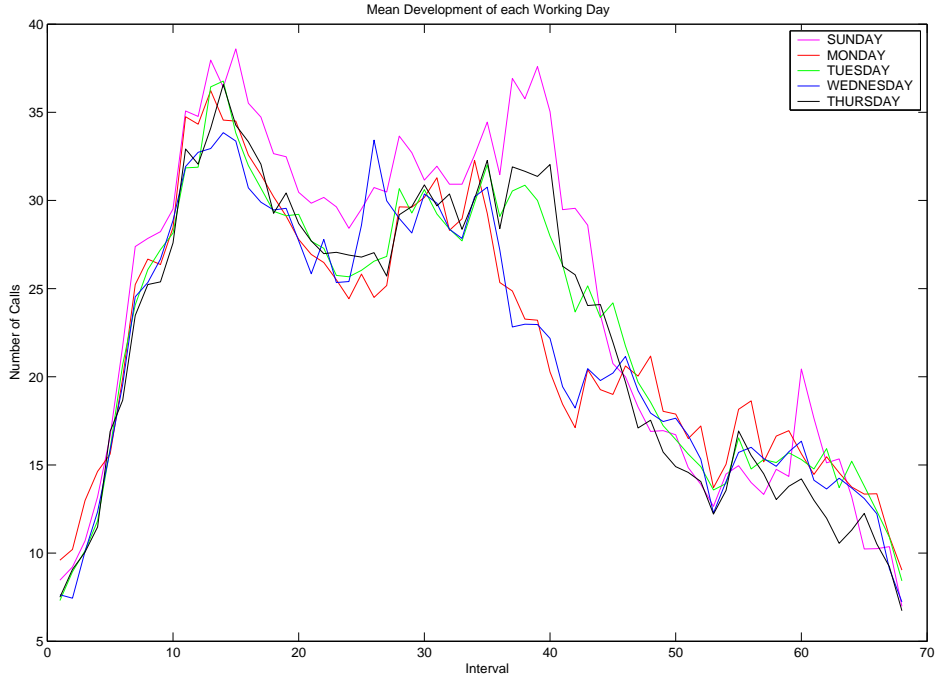
Stochastic queuing models are used to model call centers. Frequently used models, using Kendall's notation, are the M/M/c and the M/M/c/N. The first letter M denotes that arrivals to the call center are according to the (Markovian/memoryless) Poisson process. The second M denotes that service time distribution is exponential (Markovian/memoryless). The third term denotes the number of servers available to handle incoming calls. The fourth term denotes the total capacity of the system, that is the number of servers plus the number of places in the queue. When the system has an infinite queue the fourth term is often omitted. These are the basic properties of a queuing system used to model call centers. Calls arrive at the system with a certain rate  $\lambda$ . They then get routed to a server if there is an idle server, or they get routed to the queue until a server becomes idle. Servers handle their call at a certain rate  $\mu$  and become available for the next call when they have handled their previous call. From these models we can derive several key performance aspects such as, the average waiting time, the average server utilization, the percentage of callers that have to wait less than 20 seconds, etc. Various variations to these queuing frameworks are possible, including adaptations to the model that: handle servers with different skills, callers that abandon when they have waited for a certain amount of time, etc. However, this project is not about adapting or improving stochastic queuing models, so the interested reader is referred to [1].

The models described above assume a fixed arrival rate  $\lambda$ . When we use the model on a particular day as a whole the results will be unreliable; this is because the call arrival rate  $\lambda$  in call centers varies over the day. Green, Kolesar

and Soares introduced an algorithm, which is known as SIPP, to overcome this problem [2]. They split a day into several smaller, say 15 or 30 minutely, intervals and the model is applied to each of these intervals separately. This way they find the number of agents needed per time interval to reach service level goals (under certain assumptions). Forecasting the call arrivals per time interval, and thereby the arrival rate  $\lambda$ , accurately is important to get the most out of the models. Therefore a lot of research has been done in the field of forecasting, mostly from a mathematical perspective [3] [4], [5]. Typical mathematical approaches include statistical modelling (e.g., ARMA) and smoothing techniques (e.g., exponential smoothing, Holt Winters). Typically, forecasting becomes more difficult when the timescale on which we forecast gets smaller. Smoothing techniques work best when the time dependent variable changes gradually over time, in terms of call centers typically: calls per year/month. ARMA models can cope with trend and seasonal influences, in terms of call centers typically: calls per week/day. The models on call center performance assume a rather fixed arrival rate during the time-interval in which the model is applied, this implies hourly, 30 or even 15 minutely forecasts on call arrival rates. Forecasting of the last type is relatively unexplored, smoothing and statistical models are not completely satisfactory and there must be a better form of forecasting since we are not dealing with completely inapprehensible data.

The goal of estimating  $\lambda$  on small time scales during the day can be very interesting. We acknowledge that the number of agents for a particular day is staffed in advance and in general we cannot expect to call in additional agents or to send home surplus agents. So what is the use of estimating lambda during a day? The answer to this question is call blending. Many call centers do more than just handling incoming telephone calls; Other jobs include: handling faxes, replying to e-mails, making outbound calls, backoffice work, etc. Updating the forecast for  $\lambda$  during the day can help the call center plan their personnel over these different jobs flexibly, meeting service level goals for incoming calls while optimizing agent productivity. More on call blending can be found in this paper by Bhulai and Koole [6].

Relatively new methods for forecasting are the methods based on machine learning techniques. Rise of computational power and huge databanks enable us to look at forecasting from a new point of view most likely enabling us to increase the accuracy and precision of our forecasts. These techniques must be researched in order to get a better understanding of them and to develop better forecasting techniques. The particular machine learning technique we investigated in this paper is called the K-Nearest-Neighbour algorithm. This algorithm has applications in pattern recognition such as fingerprint recognition, face recognition and handwriting recognition. Shortly, the algorithm uses the property that patterns that look the same belong to the same class / behave the same. Incoming call patterns for different days usually have similar shapes, the K-nearest-neighbour algorithm might be able to exploit this to make better forecasts.



## 2 Problem Description

### 2.1 Data

Call centers have huge amounts of data about the calls they receive. For each call they log the starting time, the end time, and often a lot more statistics (waiting time, handling agent, etc.). The starting times of the calls can be aggregated into a more compact form of data, namely call arrivals per 15 or 30 minutely interval. For example, working days from 8:00 to 18:00 divided in 15 minutely intervals (45 call arrivals on 19-03-2005 from 8:30-8:45):

Day	8:00	8:15	8:30	...	17:30	17:45
05-03-2005	21	23	25	...	15	6
12-03-2005	16	12	18	...	14	5
19-03-2005	32	40	45	...	17	10
...	...	...	...	...	...	...

$n$  = number of time intervals per day

$t_i$  = time interval  $i$ ,  $i = 1, \dots, n$

$m$  = number of days in historical data

$d$  = day of historical data = 1, ...,  $m$

$h_{d,i}$  = call arrivals on day  $d$  in time interval  $t_i$

$$H = \begin{pmatrix} h_{1,1} & h_{1,2} & \dots & h_{1,n} \\ h_{2,1} & h_{2,2} & \dots & h_{2,n} \\ \dots & \dots & \dots & \dots \\ h_{m,1} & h_{m,2} & \dots & h_{m,n} \end{pmatrix}$$

Different business days can be in separate tables/matrices. All business days

can also be put in the same table/matrix. Without prior evidence which alternative is best, it may be safer to separate business days. In the following we will assume working on a particular (yet not a specific) matrix  $H$ . The effect of separating or combining business days will be investigated further along in this paper.

## 2.2 Conventional Forecasting

Assume that call arrivals are known for fixed small (15 or 30 minutely) time intervals for a large number of days in the past. Conventional forecasting methods just forecast the call arrivals for an interval by taking the mean number of call arrivals for that interval on past days. This method provides no further way to update the forecast when statistics on call arrivals for a new day become available.

$$F_i = \text{Forecast for call arrivals in interval } t_i = \frac{h_{1,i} + \dots + h_{m,i}}{m}, \quad i = 1, \dots, n$$

## 2.3 K-nearest-neighbour

Conventional forecasting provides no way to update the forecast during the day. K-nearest-neighbour algorithms on the other hand can be used to update the forecast. In this paper we study if the K-nearest-neighbour algorithm delivers better forecasts than the conventional forecasting method. Results will be compared using statistical tests. Also, we translate the problem back in terms of business operations. The different forecasting methods were run against real data and mathematical models to test if the K-nearest-neighbour algorithm performs better than conventional forecasting, in terms of reaching service level goals with minimal staffed personnel. In the following chapter we will go into the details of the K-nearest-neighbour algorithm.

# 3 K-Nearest-Neighbour

## 3.1 Basics

The K-nearest-neighbour algorithm is a machine learning algorithm that belongs to the class of instance based learners. The algorithm selects K instances (with known class or process) that resemble a certain reference instance (with unknown class or process) best. It then uses these K selected instances to classify the reference instance or predict the process of the reference instance. This algorithm can be used when instances that look similar also tend to belong to a similar class or tend to have a similar behavior.

## 3.2 K

K is a parameter for the algorithm and the optimal value for this parameter is generally not known in advance. Testing the algorithm for different values of K gives insight in the effects of K and helps us to choose a suitable good value for K.

### 3.3 Instances

Translating this into the context of our call center forecasting study. Suppose we have a (reference) day for which do not know the call arrivals for all intervals. We only know the realizations of call arrivals up to the first, say  $x$  (with  $x < n$ ), intervals. We shall call the vector of length  $x$  with these realized call arrivals our reference trace. This vector is filled with the call arrivals for the intervals  $t_1$  to  $t_x$  of the reference day:

$r_i = \text{call arrivals in interval } t_i \text{ of the reference day}$

$\underline{r}_x = \text{reference trace} = (r_1, \dots, r_x)$

The reference trace is compared to vectors of length  $x$  in the historical data to find suitable candidates (days) to base the forecast on:

$h_{d,x} = (h_{d,1}, \dots, h_{d,x}), \quad d = 1, \dots, m$

The reference trace and all the historical instances are fixed data, the calls in the intervals beyond the time scope of the reference trace is what we want to forecast.

### 3.4 Neighbour Selection / Distance Function

The K-nearest-neighbour algorithm selects the K-nearest candidates as basis for forecasting. We need to define a distance before we can define which candidates are nearest to the reference trace. For this we introduce the concept of a distance function. This can be any function that compares two elements [7], in this case the reference trace and a candidate trace, and returns a certain value between 0 and infinity (smaller distance results in a number closer to 0). Whilst it can be any function, it is advisable to choose a function with the properties of a mathematical norm. We choose to investigate two distance functions, one based on the Euclidian distance and the other based on the Pearson correlation coefficient.

Euclidian Distance (ED):

$$ED(\underline{r}_x, \underline{h}_{d,x}) = \sqrt{\sum_{i=1}^x (r_i - h_{d,i})^2}$$

The Euclidian Distance, or the Euclidian norm, is widely used and is one of the most obvious distance functions to use. It looks at the sum of the squared distances, thereby favoring samples that are close to the reference trace in a Euclidian-metric way.

Pearson Distance (PD):

$$PD(\underline{r}_x, \underline{h}_{d,x}) = 1 - |\text{Correlation}(\underline{r}_x, \underline{h}_{d,x})| = 1 - \left| \frac{(x-1) \sum_{i=1}^x (r_i - M_{\underline{r}_x})(h_{d,i} - M_{\underline{h}_{d,x}})}{\sum_{i=1}^x (r_i - M_{\underline{r}_x})^2 \sum_{i=1}^x (h_{d,i} - M_{\underline{h}_{d,x}})^2} \right|$$

where  $M_{\underline{r}_x} = \frac{\sum_{i=1}^x r_i}{x}$  and  $M_{\underline{h}_{d,x}} = \frac{\sum_{i=1}^x h_{d,i}}{x}$

The Pearson Distance is based on the Pearson correlation coefficient between two vectors. A value close to 0 denotes little similarity, whereas a value close to 1 denotes a lot of similarity. This distance function looks at similarities in the shape of two samples rather than at the Euclidian difference. We chose this distance to see if the similarities in shapes could be exploited.

### 3.5 Forecasting

Once the K-nearest-neighbours are found according to some distance function we can construct a forecast.

When the neighbours are found by means of the Euclidian distance we could forecast the call arrivals for every interval (beyond our reference trace), by taking the average of the call arrivals in the corresponding time intervals of the K-nearest-neighbours.

$K_j = \text{dayindex of } j^{\text{th}} \text{ nearest neighbour}$

$\kappa = \{K_1, \dots, K_K\}$

$$F_i = \frac{h_{K_1,i} + \dots + h_{K_K,i}}{K}, \quad i = x + 1, \dots, n$$

The rather simple approach described above can not be used when applying the Pearson distance. The Pearson distance compares similarity in shape rather than Euclidian proximity. When forecasting with the nearest-neighbours found with the Pearson distance we should apply some correction to account for Euclidian distance. We could do this by adding some terms to the historical call arrival values on which the forecast is based.

$$c_j = \text{term to be added to } h_{j,i} = \frac{\sum_{i=1}^x r_i - \sum_{i=1}^x h_{j,i}}{x},$$

$$j \in \{K_1, \dots, K_K\}, \quad i = x + 1, \dots, n$$

$$F_i = \frac{(h_{K_1,i} + c_{K_1}) + \dots + (h_{K_K,i} + c_{K_K})}{K},$$

$$i = x + 1, \dots, n$$

## 4 Implementation and Evaluation

### 4.1 The Dataset

To test the algorithms in this paper we needed data about call arrivals for a fairly large number of days, say a year. Avishai Mandelbaum [1] has provided such a dataset on his website; A pdf-file with explanations about the data can also be found at that page: <http://iew3.technion.ac.il/serveng/callcenterdata/>.

Main aspects of the data:

- Data is recorded over 12 months, from 01/01/1999 till 31/12/1999, at an anonymous call center in Israel.
- During weekdays (Sunday to Thursday), the call center is staffed from 7am to midnight. During weekends (Friday-Saturday), it closes at 2pm on Friday and reopens at around 8pm on Saturday. The automated service (VRU) operates 7 days a week, 24 hours a day.
- A lot of data is recorded for each call, including a timestamp denoting the arrival time of the call.

### 4.2 Aggregation, Converting the Raw Data

The algorithms in this paper use the number of calls per time-interval as input. However, the data is given in timestamp form. So the first step was to convert

the raw timestamp data into aggregated data. This was done using some scripting in Microsoft Excel. The data is aggregated per day and per 15 minutely interval, only for weekdays during times when the call center is staffed. These are the most interesting times concerning workforce management.

The call center supports six different types of services and priorities. From the explaining text accompanying the data, it is not precisely clear how the different calls are being handled by the agents (e.g., multi skill environment). Priorities are present, but the callers do not know about this fact, so this will have little impact on the call arrival rate. To simplify things, we do not distinguish the different types of calls.

After the whole aggregation process we got five 52x68 matrices, one matrix for each weekday (Sunday, Monday, ..., Thursday). Each matrix row denotes a different day, each matrix column denotes a different interval. These five matrices form the basic working data for this paper.

### 4.3 Implementation

Main implementation and testing of the algorithms were done in Matlab. Matlab was chosen because it provides easy ways for matrix and vector computations and scripting; It provides the nice compromise between spreadsheet functionalities and programming languages that we needed. Implementations mainly included: the algorithms described in the previous chapter, functions and scripts encapsulating these algorithms (for batch testing), functions and scripts to analyse test results, and some supporting functions and scripts to generate graphical output such as boxplots and graphs. In the previous chapter we already discussed the most important of these, the K-nearest-neighbour algorithms. The rest of the implementations are more of a programming matter and therefore we shall not discuss that here any further.

### 4.4 Testing

The algorithms that we investigate are all based on historical data. For our dataset, different days may have a different number of preceding days. This will lead to another variable in our analysis, namely, the number of historical days. This will lead to more difficult interpretation of the results, therefore we eliminated this variable by making the following assumptions: *Every year is fairly the same. When looking at a particular day being evaluated, we could bootstrap historical data by taking days beyond the evaluated day. Seasonal effects are quite the same each year, and trend effects are negligible* In other words, we treat future days as if it were past days to get the same amount of historical data for each evaluated day. Every evaluated day will have 51 historical days when not combining working days, and 259 historical days when working days are being combined. **It must be stressed out that all this is only necessary in our testing environment, in real life there usually is enough (many years) data to work with.**

We test the following forecasting methods:

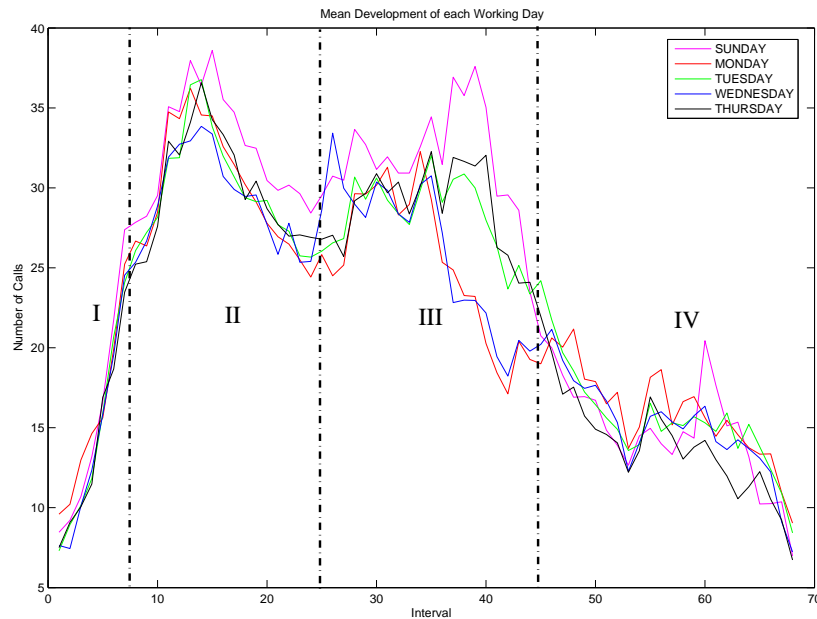
1. (CV) Conventional forecasting, simply taking averages over all historical days for each interval.

2. (ED) K-nearest-neighbour forecasting using Euclidian norm as a distance function.
3. (PD) K-nearest-neighbour forecasting using the Pearson correlation based distance function.

As mentioned earlier we could apply the methods above on separated working days (SWD) or on combined working days (CWD), so we actually have six different forecasting methods.

Each working day consists of 68 quarterly intervals. The update of the forecast can basically occur at any time between start and end of the day. It is impossible to analyse the update at all possible moments; The CPU time to calculate everything would take extremely long, and evaluating all data from the updates would be unmanageable. Therefore we chose to only update at specific moments. Most call centers have two peak moments during a typical day, with relatively a lot more arriving calls than the rest of the day; Often the first one occurs in the morning and the second one occurs somewhere in the afternoon. The call center in our dataset showed these two typical peaks as well. The moment in time during which these peaks occur are relatively stable. Each day is divided into four mutually exclusive periods and forecast updates occur at the beginning of each period (note: only CV is possible at start of period 1, since there are no traces and thus K-nearest-neighbour algorithms cannot be applied). **Updating the forecast, only happens on update moments (e.g. the starts of period 2, period 3 and period 4). When updating, we update the forecast for all the remaining intervals of the day all at once. We do no update the forecast on other moments.**

	Intervals	Description
I	01 to 08	From the beginning of the day till the beginning of the first peak.
II	09 to 24	From the beginning of the first peak till the end of the first peak.
III	25 to 44	From the beginning of the second peak till the end of the second peak.
IV	45 to 68	From the end of the second peak till the end of the day.



## 4.5 Evaluation from Statistical Perspective

We define the error of a forecast as the absolute difference between the forecasted value and the actual value. There are six forecasting methods and five working days. For every working day we did the following:

We applied different algorithms at the starts of period 2, 3 and 4 (K-nearest-neighbour algorithms were run with different values of K to determine the best K). When applying at the start of period 2, we looked at the error in forecast for period 2. When applying at the start of period 3, we looked at the error in forecast for period 3. When applying at the start of period 4, we looked at the error in forecast for period 4. It is not necessary to look at the forecast error in period  $x + i$  when updating the forecast at the start of period  $x$ , because the forecast for period  $x + i$  will be updated/overwritten at a later period.

We applied statistical tests in order to test for significant difference between different forecasting methods, at all the day/period combinations. The test used for this was the Wilcoxon rank test. This is a paired test that can determine whether a set A has a significantly lower median than a set B. Tests based on normality assumptions were inappropriate because QQ-plots and normality tests showed that the error distribution was not normal in general. Also the sample size of 52 is too small to assume applicability of normality based tests.

## 4.6 Evaluation from Business Perspective

Statistical evidence that a certain forecasting algorithm is better than another, may not necessarily mean that it is of direct practical importance from a business perspective (nor does lack of statistical evidence directly mean that there is no practical importance!). However, it is an indicator that there may be improvement possible in terms of business operations. How did we evaluate this?

As mentioned in the introduction, call centers use certain service level measures to determine their level of customer service quality. Known service levels are: the percentage of calls answered within 20 seconds, and the average time until a call is answered (Average Speed of Answer = ASA).

We chose an ASA of 30 seconds as service goal. We used the uniformization method as described in this book [8] and this paper [9], to find optimal staffing levels and tested these staffing levels under the true call arrivals. The ASA implementations are based on the web-implementation by Arnout Wattel (<http://www.math.vu.nl/~awattel/asa.php>). The web-implementation is a simpler version of the implementation we made, it has a limitation of ten intervals and it does not allow for any fixed staffing levels when optimizing, but gives a good idea how the staffing algorithm works.

We evaluated the following scenario's (remember: updates only occur at the starts of period 2, period 3 and period 4. All future intervals are then updated all at once. For past intervals, forecasts will be replaced by true call arrivals. Staffing levels for past intervals will not be altered, as we cannot go back in time and alter these staffing levels):

1. What is the minimum number of agents when the true call arrivals are used for planning?
2. What is the minimum number of agents when the conventional forecasting

is applied only at the beginning of the day? Found staffing level will be run against the true call arrivals to estimate the true service level.

3. What is the minimum number of agents when the conventional forecasting is applied at the beginning of the day, and future staffing levels are updated based on true past call arrivals/staffing levels (forecast is not updated)? Found staffing levels will be run against the true call arrivals to estimate the true service level.
4. What is the minimum number of agents when the conventional forecasting is applied at the beginning of the day, and staffing levels are updated with true past call arrivals/staffing levels (Also, the forecast is updated with the best algorithm at every day/period update combination.)? Found staffing levels will be run against the true call arrivals to estimate the true service level.

Scenario 1: gives us the true minimum number of agents. We can not expect to get better results than this true minimum.

Scenario 2: gives us the minimum number of agents as found when applying the conventional forecasting and staffing method used in call centers.

Scenario 3: shows the impact on the minimum staffing when staffing is being updated, taking known call arrivals/staffing levels into consideration, without updating future call arrival forecasts. Scenario 3 is meant to test whether It is possible that any positive results in scenario 4, which we ultimately want to test, are due to taking true past call arrivals/staffing levels into consideration, rather than due to better forecasting.

Scenario 4: shows the minimum staffing when staffing is being updated, taking known call arrival/staffing levels values into consideration, now with updating future call arrival forecasts.

Note: An agent is scheduled per hour during the day, and there is an additional half an hour at the end of the day for wrapping up calls. We chose to schedule agents per hour, because in real life it is impractical to schedule agents for say quarters, and applying uniformization and greedy search on quarters would take extremely long to run through a computer program.

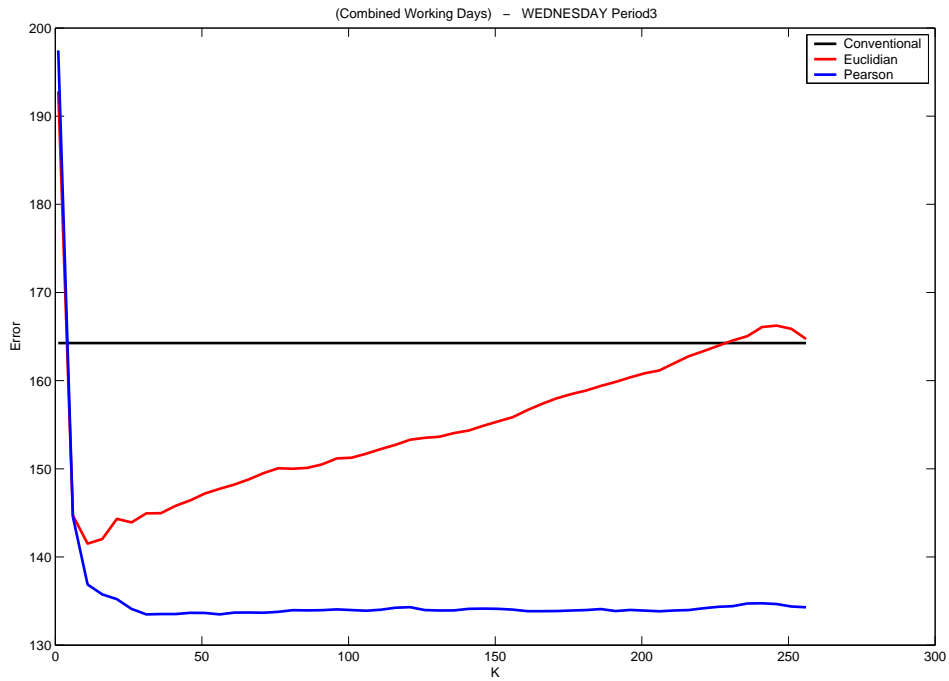
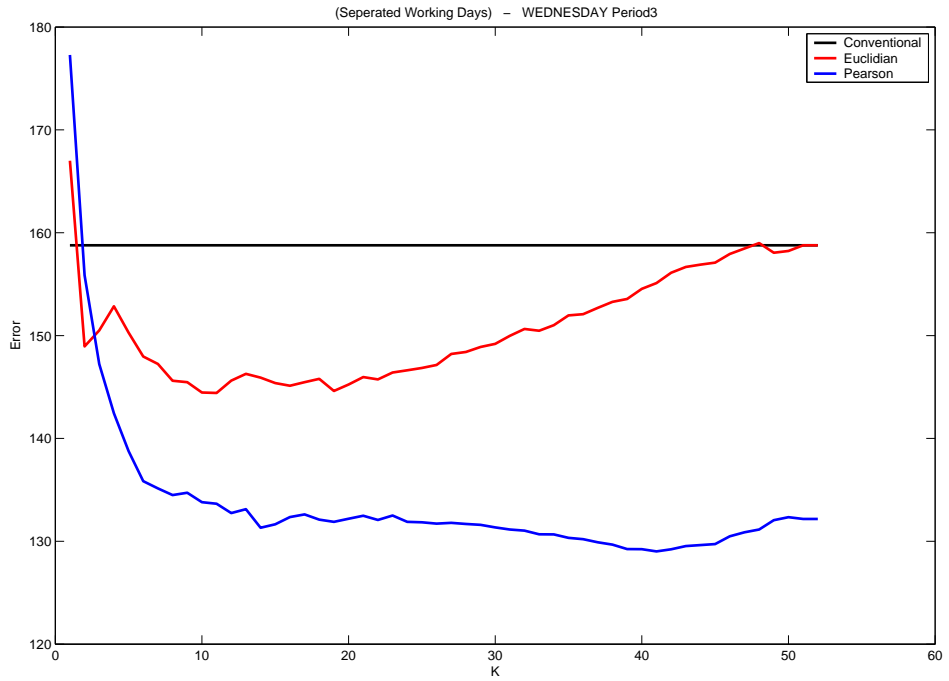
## 5 Evaluation Results

### 5.1 Evaluation from Statistical Perspective

It will take up too much space to present all the test results (dozens of pages with graphs and tables). Therefore we shall only show representable samples to illustrate the tests that were discussed in the previous chapter. After that we will sum up the conclusions that we have drawn from all the test results.

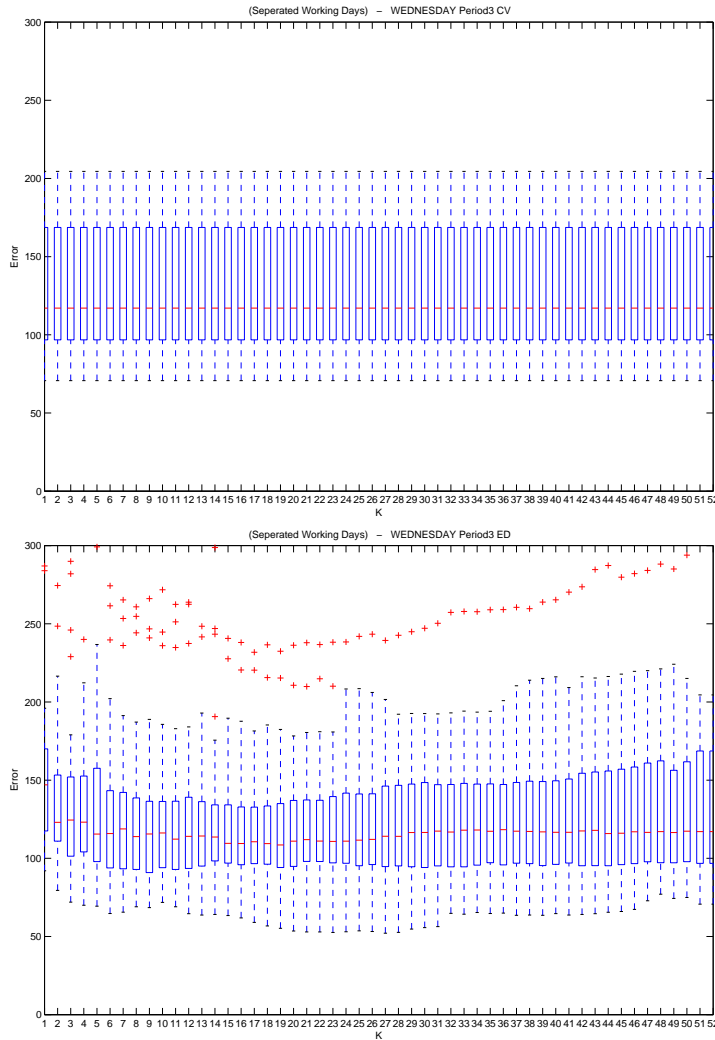
Recall the testing method from the previous chapter. Six forecasting methods, five working days and three update moments during a day. Five working days times three updates moments leads to fifteen day/period combinations. We shall now look at a particular day/period combination, Wednesday at the third period. That is, we look at the forecast error for the third period, when we update the forecast at the start of the third period. The following graphs show the errors for the different forecasting methods. The y-axis shows the error and the x-axis shows the value of  $K$ , the number of nearest neighbours. Note that there

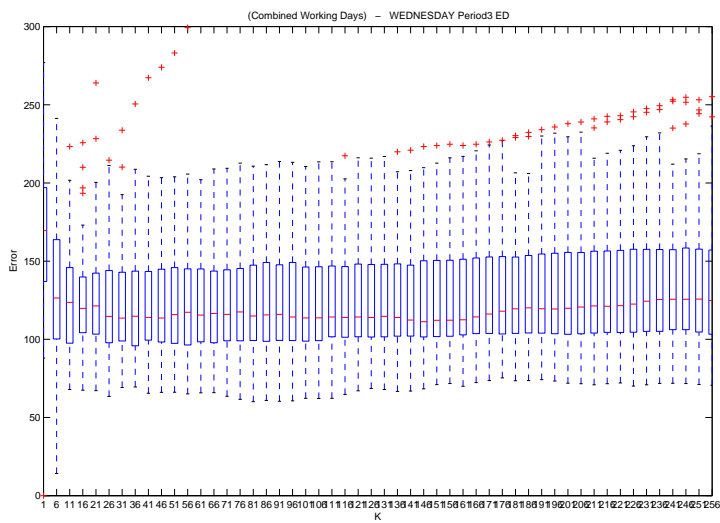
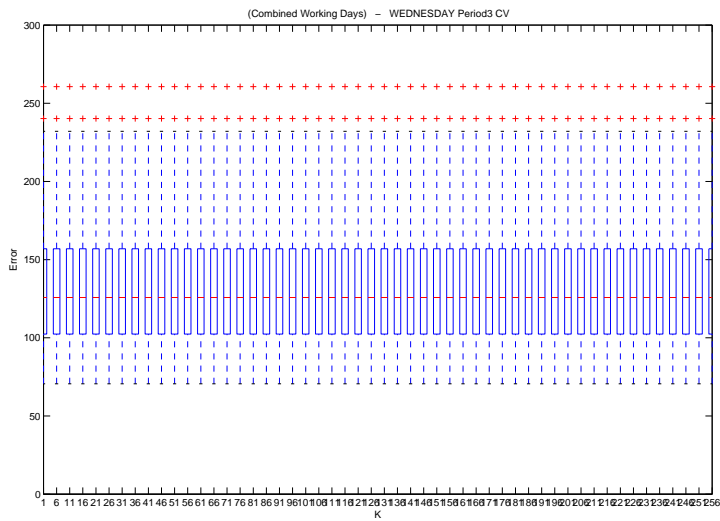
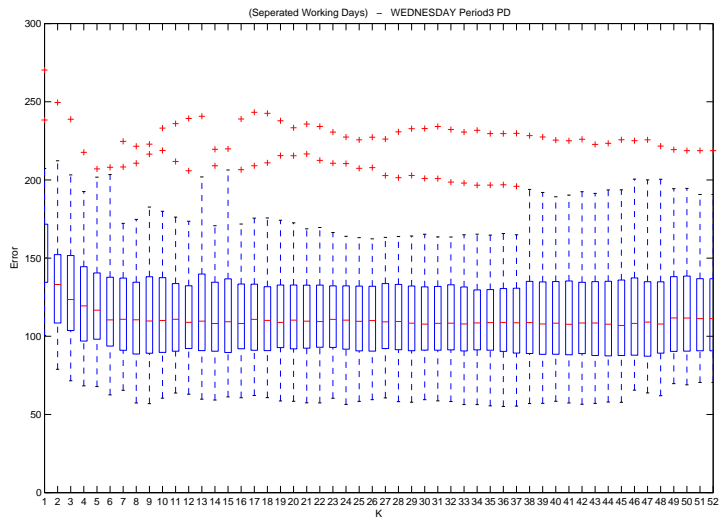
is a horizontal line for conventional forecasting methods because K is not used.

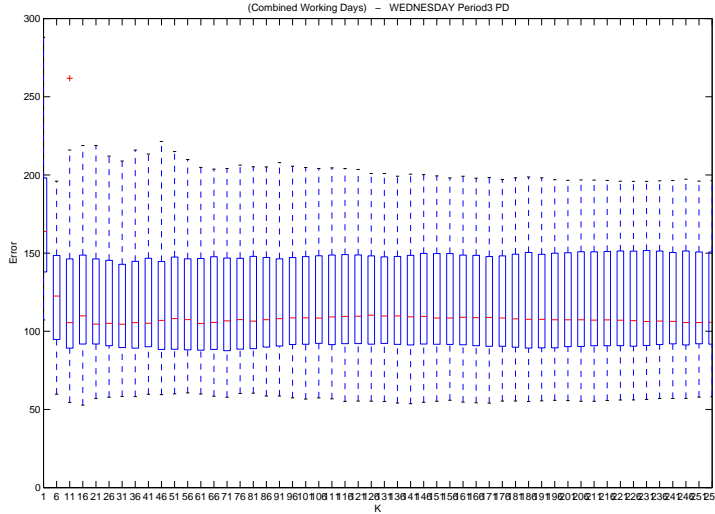


From these graphs one may easily be led to believe that the K-nearest-neighbour methods perform better than the conventional methods. The error however is an average over 52 tested Wednesdays, it is not unlikely that the lower errors using the K-nearest-neighbour methods are due to one, or a few,

extremely well forecasted samples. To gain more insight in the variability of these errors we could look at the standard deviation. However, the standard deviation is not an obvious measure when the distribution of the underlying variable is unknown. Therefore we used boxplots to study the spread of the error for the different methods.







These plots show that the errors for the K-nearest-neighbour based methods are denser around a lower error value. This approach is rather subjective, therefore we chose to verify these results with the earlier mentioned Wilcoxon rank test. We used this test for each pairing of methods (best K for each method, that is the K for which the method has the lowest mean error), to test whether the first method has a significantly lower median than the second method; SWD-CV vs. SWD-ED, SWD-ED vs SWD-PD, ..., in total 30 combination pairs. Consider doing this for a certain combination of methods at all possible day/period combinations, say, SWD-ED vs. SWD-CV. Then we could construct the following matrix with Wilcoxon rank test significance values. Matlab gave the following result:

$$\begin{aligned}
 &significance\_Swd\_ED\_Swd\_CV = \\
 &0.0052 \quad 0.0373 \quad 0.1931 \quad 0.0512 \quad 0.0009 \\
 &0.0215 \quad 0.0210 \quad 0.0008 \quad 0.0167 \quad 0.0000 \\
 &0.1988 \quad 0.0036 \quad 0.0640 \quad 0.0098 \quad 0.0934
 \end{aligned}$$

The row indicates the period, row 1 is period 2, row 2 is period 3, row 3 is period 4. The column indicates the day, column 1 is Sunday, ..., column 5 is Thursday. We can clearly see that the first method is significantly better than the second method for period 3 at all days. With exception of Tuesday and Wednesday (at a significance level of 0.05), it is also significantly better for the second period. For period 4 only Monday is better.

In general we saw that period 2 and period 3 were better forecasted with K-nearest-neighbour algorithms than with the conventional method. For period 4 there was not a clear indication that any method was better than another, for all working days. Often, the Pearson based method performed slightly better than the Euclidian method, we did not discover a particular pattern in when this happens however.

## 5.2 Evaluation from Business Perspective

Like the previous section, presenting all test results will consume too much space, we will again use representable samples to illustrate.

Recall the four scenarios mentioned in the previous chapter. The four scenarios have been tested on ten days of each working day. We did not do more than ten tests per working day because uniformization in combination with greedy search took very much CPU time as is. Below we show the results as obtained for the Wednesday, respectively for scenario 1, scenario 2, scenario 3 and scenario 4.

Scenario 1 investigates the true optimal number of agents. Every row denotes a different day (from the same working day, in this case Wednesday). The first column is the ASA and the second column is the number of working hours deployed for that day.

*ASA\_Test1\_WEDNESDAY* =

29.5783	89.0000
28.7840	105.0000
27.8924	84.0000
27.8958	70.0000
28.6960	111.5000
29.0204	102.0000
29.9510	92.0000
28.9923	82.0000
28.4895	108.0000
29.7751	117.0000

Scenario 2 looks at the conventional forecasting method without adjusting staffing levels. Again, the second column is the number of deployed working hours. The first column now shows the ASA as based on the forecasted number of call arrivals, and the third column shows the ASA based on the true number of call arrivals.

*ASA\_Test2\_WEDNESDAY* =

29.7509	89.0000	37.6896
29.3031	89.0000	194.9620
29.9878	89.0000	25.3804
28.1331	90.0000	8.6529
28.8497	89.0000	127.4359
29.1518	89.0000	117.5038
29.7500	89.0000	44.0074
27.7811	90.0000	25.0355
29.0360	89.0000	133.9537
28.5661	89.0000	239.8513

Scenario 3 looks at the conventional forecasting method (applied only at the beginning of the day), combined with adjustments of staffing levels based on true past call arrivals and staffing levels. The final adjustment before the end of the day is made at the start of period 4, therefore we focus on this period. The second column is the number of deployed working hours. The first column now

shows the ASA as based on the forecasted number of call arrivals for period 4 (and true past call arrivals for the periods before period 4), and the third column shows the ASA based on the true number of call arrivals.

*ASA\_Test3\_WEDNESDAY =*

29.3122	91.0000	30.9722
46.9935	154.0000	45.4014
28.8805	88.0000	27.6744
25.2684	86.0000	12.1418
76.9606	114.0000	75.4529
30.2945	139.0000	29.2838
29.6478	95.5000	30.0246
28.6448	88.0000	33.2402
44.0401	135.0000	43.1372
35.1920	148.0000	33.9807

Scenario 4 looks at the best forecasting method, combined with adjustments of staffing levels based on updated forecasts and true past call arrivals / staffing levels. Again we look at period 4. The second column is the number of deployed working hours. The first column now shows the ASA as based on the forecasted number of call arrivals for period 4 (and true past call arrivals for the periods before period 4), and the third column shows the ASA based on the true number of call arrivals.

*ASA\_Test4\_WEDNESDAY =*

29.1215	90.5000	34.7052
47.6546	157.0000	46.0928
31.4351	105.0000	32.7986
29.2502	74.0000	29.2221
71.8927	121.0000	72.9313
31.2303	134.0000	29.8620
29.4536	95.5000	30.2343
51.5946	101.0000	50.5396
31.3868	136.0000	31.4382
35.0088	142.0000	34.6535

In general we saw that scenario 2, conventional forecasting and no staffing level adjustments, performed poorly. In many cases the goals are far from met, the ASA is far above the desired 30 seconds, or the ASA is far below the 30 seconds which indicates a surplus of deployed working hours. The ASA that is achieved for the different days fluctuates a lot and exceeds 50 seconds, and even 100 seconds, quite a lot.

When looking at the results of scenarios 3 and 4, we see that these are far better than those of scenario 2. ASA results have a smaller variability about the desired 30 second service level goal. The ASA goal is met more often in scenarios 3 and 4 compared to scenario 2.

*Scenario 3 versus scenario 2:*

*40 cases better, 4 cases about the same, 6 cases worse, 50 cases total*

*Scenario 4 versus scenario 2:*

*32 cases better, 3 cases about the same, 15 cases worse, 50 cases total*

When comparing scenarios 3 and 4, we can not draw any clear conclusions. They perform about the same. Sometimes scenario 3 performs better and sometimes scenario 4 performs better. So we cannot conclude that the use of K-nearest-neighbour algorithms has a positive impact from a business perspective.

*Scenario 3 versus scenario 2:*

*16 cases better, 7 cases about the same, 26 cases worse, 50 cases total*

## 6 Conclusion

When purely looking at call volume prediction, from a statistical argument, applying nearest neighbour algorithms may improve quality of prediction significantly when applied at certain day-period combinations. When we analyse these better prediction prestations in practice, from a business perspective, no significant improvement is noticable. Applying forecast updates using K-nearest-neighbour algorithms and readjusting staffing levels accordingly may be better than the conventional planning method, in the sence that service goals are met in more cases or may be met closer to the goal (scenario 4 vs. scenario 2). However when we do not apply forecast updates but just readjust our staffing levels, according to true past call arrivals, no real improvement is noticable (scenario 4 vs. scenario 3). This leads to the suspicion that the real improvement in comparison to scenario 2 is due to the adjusting of the staffing levels with the use of true past call data, rather than better forecasting of future data.

It is interesting to see that the statistical argument does not automatically imply practical improvements. While forecasting has statistically significantly improved using K-nearest-neighbour algorithms, this improvement is still not big enough to cause any substantial improvements in business operations. Forecasts must be more accurate to achieve improvements, but how accurate can they get? It is impossible to reach results as in scenario 1 in real life. There will always be uncertainty in the number of call arrivals per interval and it is unknown how accurate forecasts can get. With all the knowledge we have about call centers today, it is still very difficult and challenging to manage them. A strong accent is put on deterministic-like optimization techniques (with underlying stochastic models), under the assumption that accurate forecasting of call arrivals make these optimization techniques good and useful. This project opened our eyes mattering this subject. Achieving better forecasts can be quite difficult, it may very well not be possible to achieve forecasts that are accurate enough to apply the usual mathematical models and optimization techniques succesfully. While research in stochastical models that handle skills and priorities may are very interesting and challenging from a mathematical perspective, its use in practice will be very limited if it does not cope with the stochastical nature of call arrival rates. More research should be done in the field of managing uncertainty in call arrival rates. That is the only way to get robust, and cost effective, schedules in practice. In combination with this research in better forecasting techniques will be more useful.

## References

- [1] Noah Gans, Ger Koole, Avishai Mandelbaum: Telephone Call Centers: Tutorial, Review, and Research Prospects, *Manufacturing & Service Operations Management*, 5:79-141, 2003.
- [2] L. Green, P. Kolesar, and J. Soares, An improved heuristic for staffing telephone call center centers with limited operating hours, *Production and Operations Management*, 12(1):4661, 2003
- [3] Samuel G. Steckley, Shane G. Henderson, Vijay Mehrotra: Service System Planning in the Presence of a Random Arrival Rate, November 1, 2004.
- [4] Athanassios N. Avramidis, Alexandre Deslauriers, Pierre L'Ecuyer: Modelling Daily Arrivals To A Telephone Call Center, *Management Science*, Volume: 50, Issue: 7, Cover date: July 2004
- [5] G. Jongbloed and G.M. Koole, Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17:307-318, 2001.
- [6] S. Bhulai and G.M. Koole. A queueing model for call blending in call centers. In *Proceedings of the 39th IEEE CDC*, pages 1421–1426. IEEE Control Society, 2000. 1.8, 3.5
- [7] Ian H. Witten, Eibe Frank: *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*, ISBN 1-55860-552-5
- [8] Sheldon M. Ross: *Introduction To Probability Models*, Sixth Edition, ISBN 0-12-598470-7.
- [9] Armann Ingolfsson, Elvira Akhmetshina, Susan Budge, Yongyue Li: A Survey and Experimental Comparison of Service Level Approximation Methods for Non-Stationary  $M(t)/M/s(t)$  Queueing Systems, [published in?]