Analysis and modelling of train delay data

Mark Harris

Dissertation submitted for the MSc in Mathematics
with Modern Applications
Department of Mathematics, University of York, UK

Carried out at:
Complexity Research Group, BT, Martlesham
Supervisor: Keith Briggs
Academic supervisor: Jason Levesley

August 24, 2006

Contents

Abstract

The modelling of trains has many implications for all train users. Knowledge of train delay distributions and accurate models would make the choice between earlier or later trains much easier. A large amount of real-time data has been collected. I suggest functions for modelling arrival and departure times. I then fit the collected data with an appropriate function; parameters of these functions are optimised by the Levenberg-Marquardt Algorithm. These functions can be used to simulate future trains.

Acknowledgements

Many thanks must go to my supervisor Keith Briggs, who help and guidance has been invaluable over the past 3 months. I also offer my gratitude to BT, the company where my project has been based. All of the staff were friendly and it provided a stimulating and relaxed atmosphere.

Chapter 1

Introduction

1.1 Background

For many years people have studied delays and their effects around various types of networks; one particular network which has had much research done about it, is road traffic. Richard Gibbens of the University of Cambridge has produced a nice model of traffic speeds clockwise around the M25 [Gib], which show an interesting wave of delay propagating against the direction of the traffic. In comparison to other networks, little has been found out about the movement of rail traffic. The main problem one faces is getting enough data to accurately estimate the parameters for the models which they believe to be correct. The main aim of this project is to statistically analyze and look for emerging patterns in the dispersion of delays and create an effective stochastic model for distribution of the delays.

1.2 The railway system: a brief history

The British railway system; initially built in the 19th century, was originally owned and operated by four companies, Great Western Railways; the London, Midland and Scottish Railways; the London and North Eastern Railways and the Southern Railway, until the Second World War. The Transport Act in 1947 paved way for the nationalisation and British Rail came into existence shortly afterwards. In the following period a vast amount of money was spent modernising the network. In the late nineties the network was gradually privatised so that the tracks and stations (the infrastructure) are owned by Railtrack, and the trains owned by a variety of different companies. After privatisation confidence in the punctuality of the train networks has wained and now each train operator has strict quota they have to fulfil: 95% of trains have to be delay 5 minutes or less. This statistic is met by the vast majority of the train companies; however, this is not the end of the story. There is no information about the distribution of the delays bigger than 5 minutes. This statistic also doesn't contain much about smaller delays; if you had an important business meeting to attend to immediately after your journey you would want to know the chances of that train being on time, which from the statistic you can only say is less than 95% of the time.

1.3 The data set

The data set was collected by Keith Briggs and Richard Clegg, over the course of the last nine months. It consists of nearly all the train departures from 24 stations around the Railtrack network. A total of nearly one million delay times have been collected, a figure that is increasing constantly as the data is obtained from a *python* program written by Keith Briggs called `get_train_data.py` which gets the data from www.livedepartureboards.co.uk which is updated by Network Rail in real time.

The data collected is spread across a large number of `.dat` files, one per station for every day. In every file the scheduled departure time, destination station and delay time is stored for each departing train. The delay times are stored in a discrete number of minutes late; however, from the website it is unclear whether a delay of one minute is between a delay of 0 and 1 minutes or between 1 and 2 minutes.

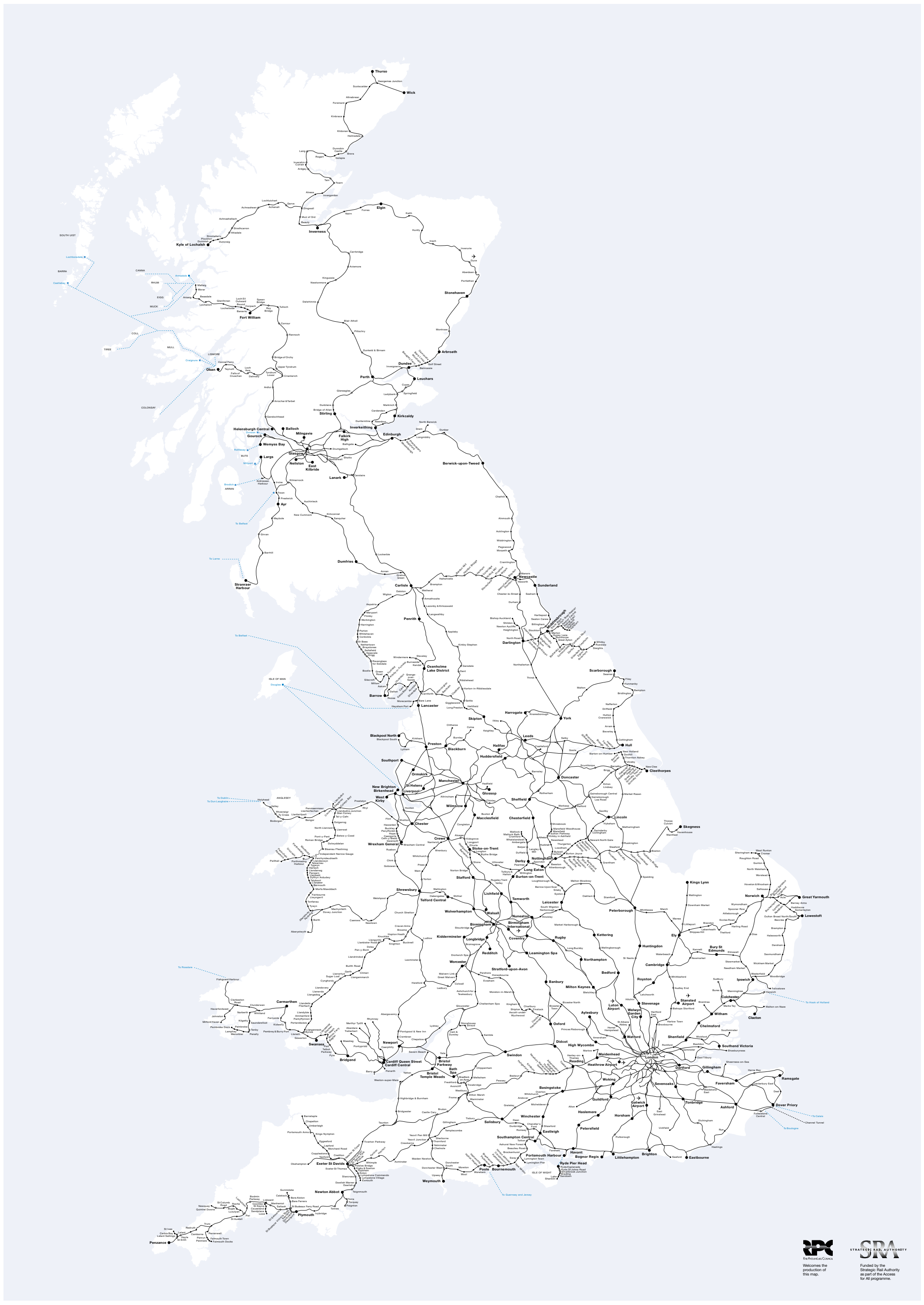


Figure 1.1: A diagram showing the whole UK rail network

Birmingham	27/09/05	Durham	03/07/06
Leicester	19/11/05	Manchester	18/10/05
Canterbury East	16/10/05	Doncaster	23/09/05
Cambridge	01/10/05	Edinburgh	16/11/05
Newcastle	25/10/05	Nottingham	19/11/05
Canterbury West	16/10/05	Ely	19/11/05
Colchester	02/12/05	Glasgow Central	03/07/06
Oxford	18/10/05	Peterborough	27/09/05
Coventry	02/12/05	Ipswich	27/09/05
Catterick	02/12/05	London Kings Cross	05/07/06
Sheffield	16/11/05	York	23/09/05
Darlington	03/07/06	Leeds	16/11/05

Table 1.1: A table of all stations to have data collected from and the first available date.

Chapter 2

Descriptive statistics

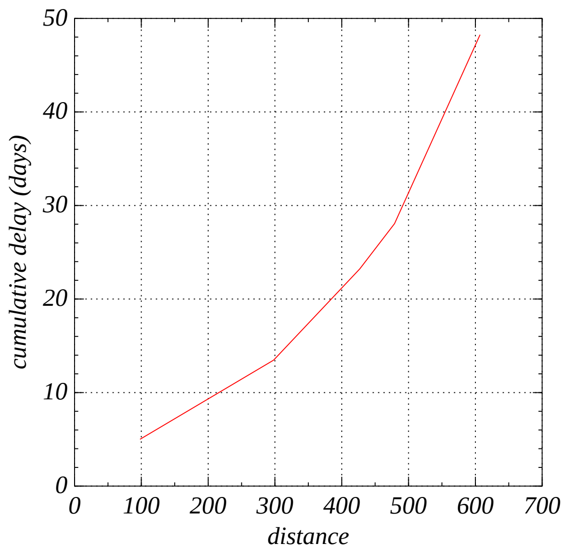
2.1 Cumulative delays

It is commonly thought that delays on a train line build up towards the end of a line nearing the end of the day. This is quite plausible since near the end of the day you have trains which have set off from far away locations, and covered a great distance. Logically these trains should be more likely to be delayed than a local train which has set of a short time ago from somewhere not to far away, since you can argue that the long-distance train has more time to experience a delay.

I concentrate on the Great North Eastern Railway (GNER) route from Glasgow Central to London Kings Cross, since that is one of the longest routes in the UK and there is more stations we have collected information from than other routes. We can see from the cumulative delays against distance (figure 2.1) that as we get further away from Glasgow Central the total of the delays increases at what appears at a glance to be some exponential rate. The average delays (also figure 2.1) causes a problem, because the expectation of a delay to any given

station	Edinburgh	Newcastle	York	Doncaster	Peterborough
distance (km)	98	298	427	479	607

Table 2.1: Shows distance in kilometers along the track from Glasgow Central for stations on the GNER route.



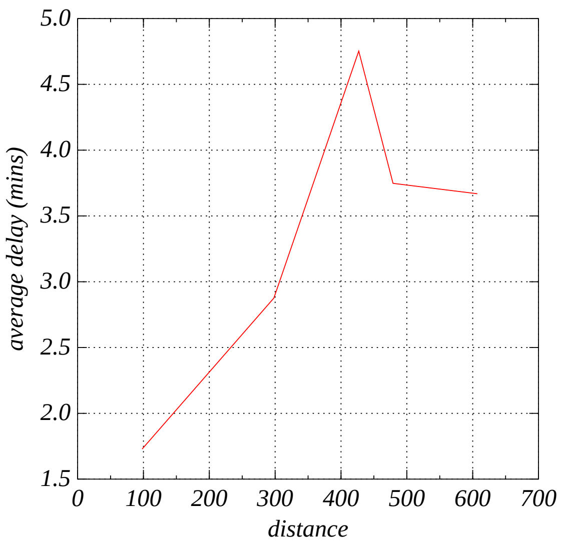


Figure 2.1: Cumulative delays (left) and average delays (right) of trains leaving stations along the southbound GNER route heading toward London Kings Cross relative to their distance from Glasgow Central.

train heading to London Kings Cross decreases. For all of the stations up to Doncaster all of the train have come from Glasgow, whereas when you reach Doncaster you get trains coming from Leeds heading towards and then at Peterborough even more trains from much closer than Glasgow Central which you would expect to have a much lower delay and drag down the overall mean of trains heading to London Kings Cross.

Removing these two stations where we cannot be sure of the origin of the train leaves us with only three valid stations, which is not enough to establish anything. What we need is some way of filtering out all of the train leaving Doncaster and Peterborough not originating from Glasgow Central. Removing these trains should make the cumulative delay graph much more accurate; however, how do you filter out trains based on where they have come from, when the data contains no information about a train's origin?

In section 2.2 is a detailed explanation of how my `traintracker.py` works. Figure 2.5 shows the average delay with increasing distance along the Edinburgh to London Kings Cross route. With the exception of Doncaster all of the stations lie on a near perfect straight line. Most trains going along this line pass

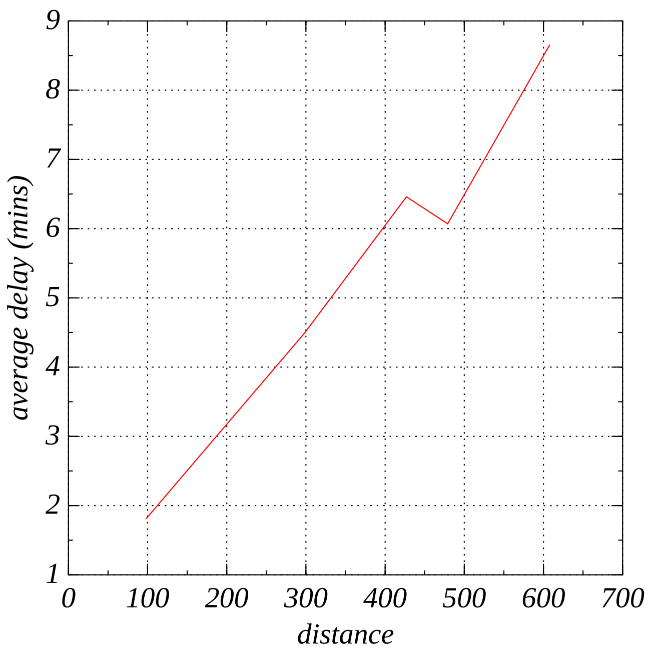


Figure 2.2: A new average delay plot against distance of the data output of the `traintracker.py` program.

straight through Doncaster as it the average is made up of only 700 trains, compared to just over 2,800 trains at each of the other stations. Doncaster and York are relatively closer together and the line between them has few other line crossing it; it is also approximately in the middle so the train operator could have put a lot of slack in the timetable, allowing the train to catch up lost time here. The reasons above I feel are enough to justify this result, and not make it a statistical anomaly.

Similar traits are seen in the limited other cases; however, we have data collected from few stations on the same route apart from the GNER route. From only a few points it is very difficult to draw any information out. Mixing the routes is not an option either, since, the plot produced showed little.

2.2 Train tracking

This section is to explain how the `traintracker.py` program works and what it hopes to achieve. `traintracker.py` is a program written by myself in *python*; it aims to track the progress of a train through the data set. Considering the train line as a series of nodes $n_1, n_2, ..., n_j$ representing stations, with a *station mapping* between each pair (n_i, n_{i+1}) called a_i, which represents the train line between stations. Firstly the program uses the `regular expression` module in *python* to establish what times trains leave from n_1 towards the destination, n_j. Each station mapping, a_j has an associated distance and *minimum time*. The *minimum time* of a station mapping is the smallest timetabled [Tho02] time any train takes to move between nodes. The algorithm processes each train individually applying the following steps:

1. Labels the last known station stop, y and time, t.
2. Moves on to the next station
3. Calculates the minimum time from current station, x, to station y by mintime $= \sum_{j=y}^{x-1} a_j$
4. Searches station x for trains departing for station n_j in time interval $[t + \text{mintime}, t + \alpha\text{mintime}]$ where α is a constant greater than one.
5. If no trains are found, go to 2
6. Checks all trains found in 4 and removes those which have caught up delay quickly based on the time taken.
7. Selects train with largest delay, go to 1, if no trains left, go to 2

This process is repeated until the last-but-one station, since we do not have a departure time from the destination station.

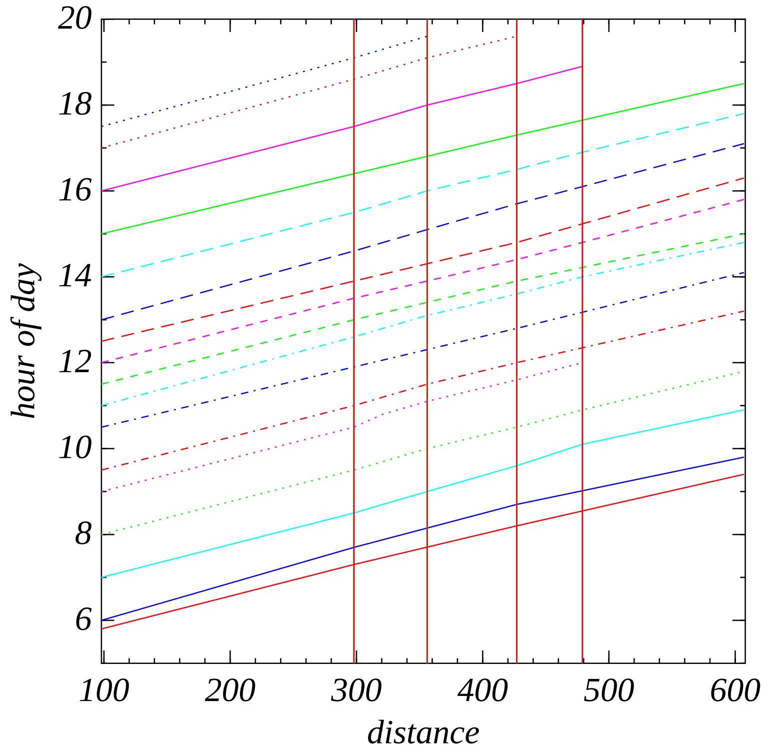
When the algorithm can not narrow the number of trains to one or fewer it chooses the train with has the largest delay. The reasoning behind this is due to the assumption that expectation of delay and distance is somehow related by a non-decreasing function. The program was created to track long distance trains long the GNER route, generally the algorithm is good at narrowing down the trains near the start of the sequence as over a longer distance the time table is fair widely spread (there is a long time between trains), however as you get closer to the destination, more local trains with the same destination are intermingled between the trains we are interested in. These trains are assumed to have originated closer to the station than the trains we are interested in, and hence are assumed to have a shorter delay. The program was tested over a series of days and compared to the timetabled time found to be about 90% accurate [GNE].

2.3 Distance-time plots

An advantage of being able to link elements of the data is it allows use to visualize what is happening to a number of trains on a distance-time plot. We can look for patterns on specific days to show how large delays effect trains running later on that day. The plots are only accurate when the train departs a station (denoted by a vertical line). For all the times between stations the distance is assumed to be linear, in other words, between stations the train is taken to travel at a constant speed.

We can see trains 'bunching' together (Figure 2.4); this appears to be caused by the delay of one train affecting the next. This could be because of a track fault, if this fault causes one train to be delayed it should really affect the next train, provided it has not been fixed in between the trains passing. In the main across the country train tracks travel in pairs, one track for each direction; this makes overtaking difficult. Assuming no overtaking takes place a delay bigger than the gap between trains has to affect the next train. In figure 2.4 there are a few trains which overtake; however, establishing where this can be difficult, because we can not track the train's movements between the stations.

The timetabled plots is displayed on the left of the observed plots as a reference to check `traintracker.py` has selected plausible times. The trains plotted run nearly the length of the country and are all so called high-speed trains; as a result they should not be timetabled to over take each other, if this occurs it is more than likely an error produced by `traintracker.py`.



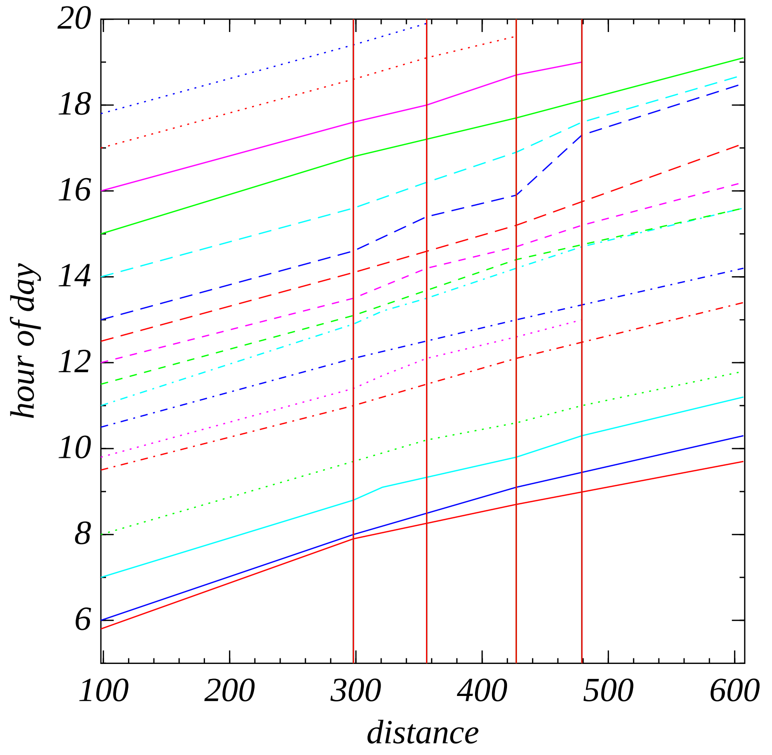


Figure 2.3: Distance-time graphs showing the timetabled (left) and observed (right) for Edinburgh to London Kings Cross on 17^{th} July 2006.

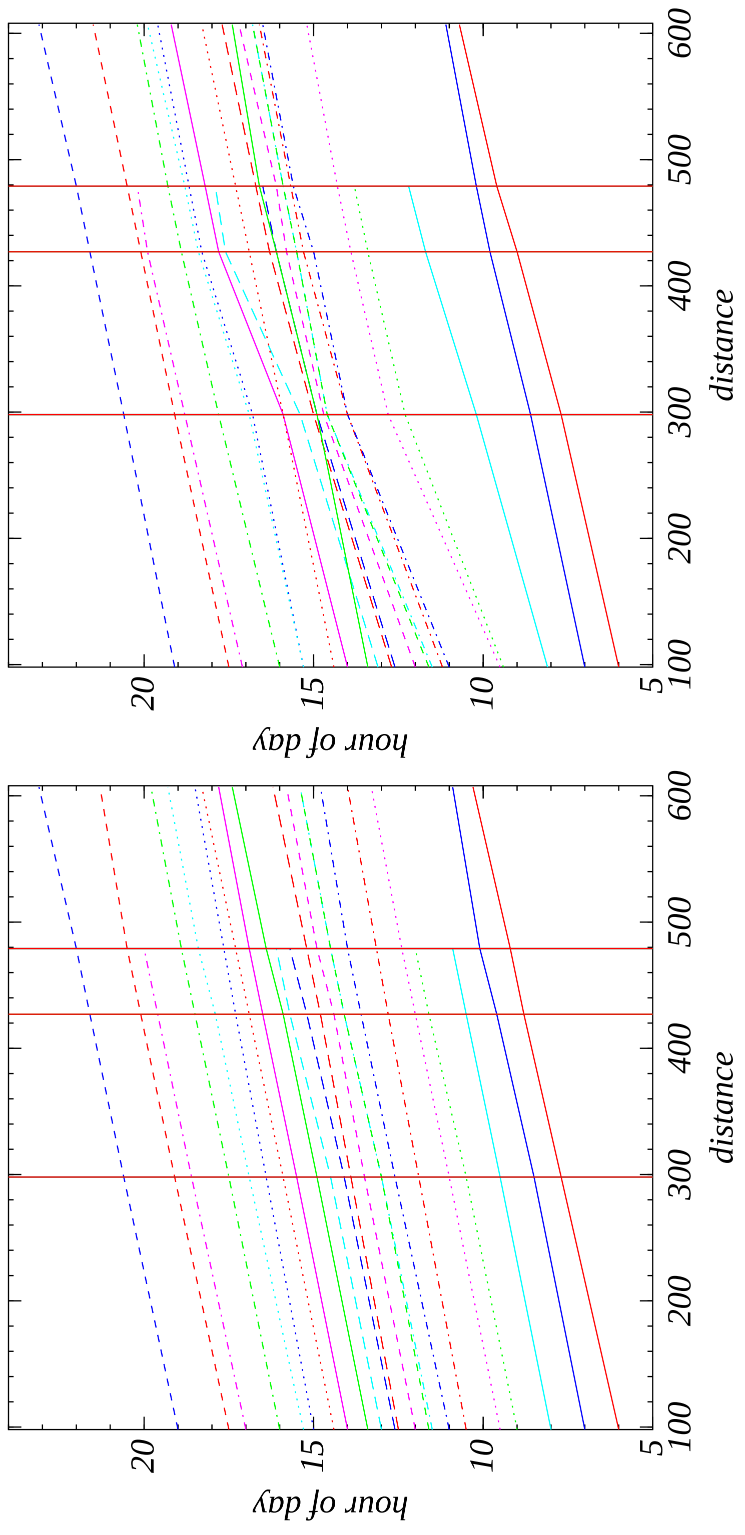


Figure 2.4: Distance-time graphs showing the timetabled (left) and observed (right) for Edinburgh to London Kings Cross on 29th December 2005.

2.4 Peak hours

As with road traffic, many people use the trains to commute to work in a morning and back home at the end of the working day. During these time train will have a shorter time interval between trains, have more carriages and trains and platforms will be more congested. These peak-hour effects should be detrimental to the service; you hear countless commuters complaining about how unreliable the train service is and yet the train companies fulfil their tight quotas. This section will try to establish whether the commuters are correct and the trains are performing badly at important times which is made up for at relatively unimportant off-peak hours.

In the train timetable there are four different time periods. The four periods are peak weekdays, off-peak weekdays, Saturday and Sunday. The timetable is different for each one of them. Peak times are defined to be 8:00 to 8:59 and 17:00 to 17:59, since the majority of commuters will work 9-5.

Any peak variations caused by large quantities of commuters will only be seen in big city stations, like for example Manchester and Birmingham. They are unlikely to be seen in smaller town stations like Peterborough because passenger will not commute to Peterborough en masse. Table 2.2 shows the means and variances for each time period. The statistics for Manchester and Birmingham are more reliable as they are big stations, where we have data for more than 100,000 trains. Peterborough and Ipswich are much smaller stations, where even though the data is collected over the same time-period there are much fewer trains involved, around 40,000.

With the exception of Peterborough all stations show signs of performing worse during the working week than they do during the weekend, and then even worse at peak hours on a weekday. During the weekend we get a very mixed bag with Saturday performing better than Sunday and some stations and vice-verse at others. Out of the selected stations Manchester performs the best (has the lower mean delay) and Peterborough consistently performs worst.

The variance seems pretty well correlated with the mean as through the table the variance increases with the mean delay.

Station	Time period of train			
	Peak	Off-peak	Saturday	Sunday
MAN	1.199 14.66	0.989 12.62	0.849 9.28	0.921 11.97
BHM	1.479 21.94	1.239 21.94	0.969 20.42	0.791 20.25
IPS	2.364 52.83	1.875 35.81	1.181 20.13	1.310 30.72
PBO	2.211 57.92	2.672 68.85	2.104 48.75	3.090 75.77

Table 2.2: A table showing the mean delay and variance of trains traveling at different times from Manchester, Birmingham, Ipswich and Peterborough.

2.5 Distribution of delays

In this section we shall look for appropriate functions with which to model the distribution of the data. Looking at the data cumulatively is preferred to probability density as it produces a smoother plot. Let's start by looking at some cumulative frequency plots from various stations. Figure 2.5 shows a few cumulative distributions with a log scale on the probability axis. The graphs show the how the probability that any given random delay is greater or equal to an arbitrary delay, x. The zero minute or more value is omitted as it is clearly 1.

The trend of the data is decreasing in a universal case, which is sensible. As the probability decays at what seems to be a straight line on a logarithmic scale, an exponential distribution could potentially model the data. The case for modelling the data by this distribution is backed up by lots of other stations which show similar features, albeit with differing gradients; figures for more routes can be found in Appendix A.

In the previous section 2.4, there were signs that station's performance varied dependent on the time of travel. Now let us analyze the actual distributions behind the means and variances of these stations at varying times.

Figure 2.6 shows cumulative plots of the probability of observing a delay greater than x for Manchester and Birmingham. The Birmingham plot show an interesting feature, because during peak hours the chance of suffering a delay less than 20 minutes is more. The chance of a large delay is smaller than at other time periods, however this data is less reliable. A 20 minute or more delay from

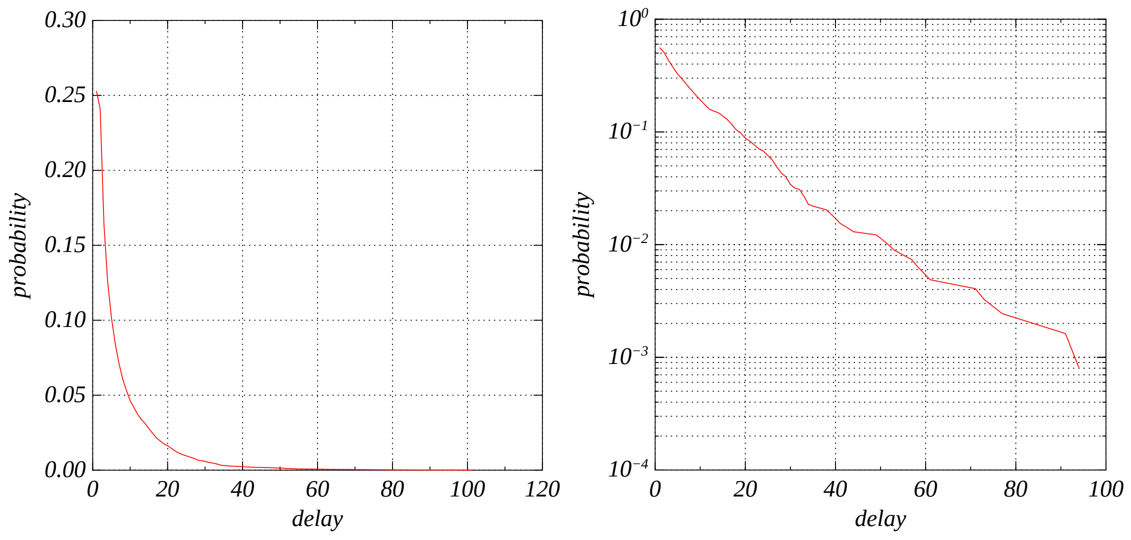
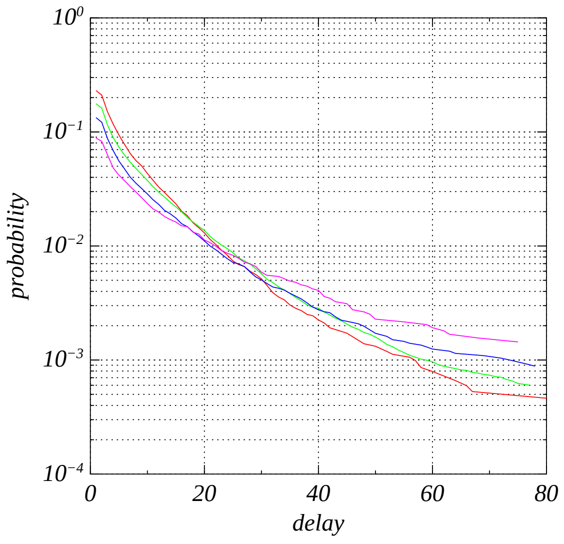


Figure 2.5: Cumulative plots showing the probability of a delay being greater than x minutes for Coventry to Birmingham (left) and Edinburgh (right, on a log scale)

Birmingham is just a likely during any time period.

The Manchester plot does not have the same properties of the Birmingham one, and is much simpler. The chance of any size of delay is greater during peak hours from Manchester.



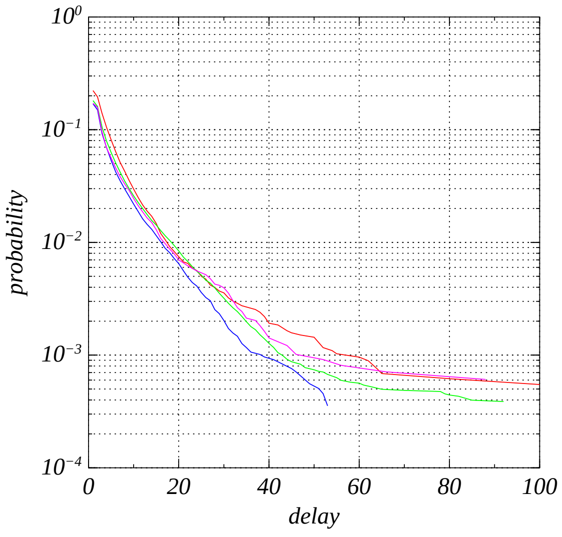


Figure 2.6: Cumulative plots for Birmingham (left) and Manchester (right) showing the difference in shape of the distribution during peak, off-peak, Saturdays and Sundays.

Chapter 3

Time series analysis

3.1 Time series

Trains are scheduled to leave the station in a particular order; this order is unique since each train is allocated its own distinct minute to depart (so that passengers can easily identify trains). The actual order that trains leave however is not unique; in peak hours trains depart very frequently and since the information is in discrete minutes, small delays can result in two trains departing during the same minute. These problems are in a big minority, later on in the section become almost entirely redundant. Now let's look at analyzing the data as a *discrete time series*. For more information on time series analysis, there nice book written by G.E.P Box and G.M. Jenkins [BJ94].

Definition 3.1 (Time Series)**.** *A time series is an ordered sequence of data, measured typically at successive times or distances.*

Generally for time series analysis the time difference between each element of the series should be constant; however, trains have a gap during the night where they do not run. In any given day there are not enough trains to give any meaningful information. We get around this problem by doing what is also done with analysis of the stock market (which also faces this problem as the stock exchange is not trading all of the time) and stringing the end of one day to the beginning of the next.

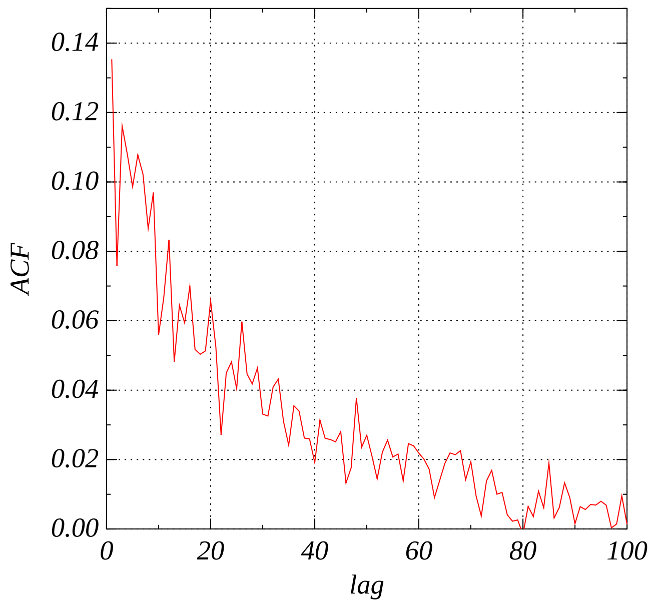


Figure 3.1: A graph showing the ACF with varying lag of all trains leaving Peterborough.

3.2 Autocorrelation

The distance-time plots in section 2.3 shows quite clearly that delays to trains earlier in the day effect the performance of later running trains. The *autocorrelation* function is more commonly used for looking for looking for periodicity in the data, however it is equally applicable to this case.

Definition 3.2 (Autocorrelation)**.** *For a discrete time series, $X = \{X_1, X_2, X_3, ...\}$, the autocorrelation function (ACF) is defined by:*

$$ACF(t, s) = \frac{E[(X_t - \mu)(X_s - \mu)]}{\sigma^2},$$

where $E[\,]$ is the expectation operator. This function is well-defined for the vast majority of functions however it is undefined for any time series with zero (a constant series) or infinite variance, σ^2.

The autocorrelation function is also a useful tool in determining whether the data is produced randomly or is affected by something that happened previously. Peterborough has a mean time between trains of 6.8 minutes, which indicates that it is possible that the delays could affect trains a small amount several hours later (Figure 3.1). These results show a distinct decreasing trend; however, the correlation is pretty weak and the trend is very noisy.

Peterborough as with most stations has several tracks along which trains come into the station. It seems far-fetched to believe that some delay on the line coming in from Cambridge could affect a train coming along the line to Peterborough from London Kings Cross. As we have trains from different lines all mingled together in our time series, this could possibly explain the sharp fluctuation with lag in the ACF. Peterborough is a station in a useful position for analysis. Any train leaving Peterborough to northern stations Edinburgh, Glasgow Central and Newcastle is always timetabled to arrive at Peterborough from London Kings Cross. In Figure 3.2 there is a much stronger correlation coefficient than before, which decays down with a much shorter lag than in Figure 3.1. Looking at the average time between trains leaving from Peterborough to the north (Glasgow Central, Edinburgh and Newcastle) explains this, since the average time between trains is much bigger, meaning that a delay can cause an effect in the network for a few hours after it occurs. This effect diminishes with each passing train till about 8 trains later, where virtually no effect is seen.

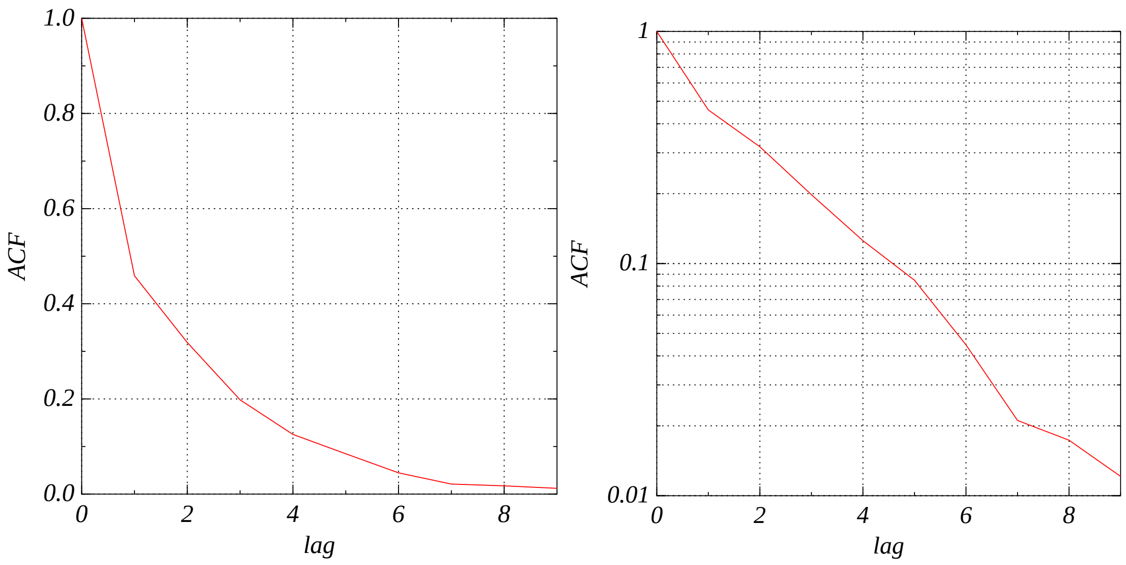


Figure 3.2: A graph showing ACF with lag for train leaving Peterborough for the north on a normal scale (left) and log scale (right)

Chapter 4

Modelling the data

To be able to model the data efficiently we need to establish what it is that we want the model to do for us. The aim of the model to create a framework to produce stochastic model of the www.thetrainline.co.uk, where a user inputs their chosen departure and destination station, for a train in the future. The user is returned a timetabled time which the train arrives at the destination station.

We have departure times for 20 stations in the UK, therefore with this limited number of stations it would be impractical make predictions based on stations we know nothing about, hence producing something as broad as `thetrainline.co.uk` would be silly.

We showed briefly in Chapter 3 that any dependencies on previous trains decrease with increased time, any correlation between trains days apart is negligible if anything. For the purpose of modelling we will assume that the user is looking up trains for a few days ahead of any information possessed, therefore the trains are not affected by any previous trains. This assumption is key as the model we shall look at is *memoryless*.

For the model to work we need any arbitrary function that we choose to accurately fit a variety of data. We need to be careful selecting which trains we include in the modelling process, to get an accurate output.

4.1 Two possible models

As discussed in section 2.5, we are looking for suitable functions to fit the delays cumulative distributions. A function which accurately fits the cumulative distribution should accurately model the data.

4.1.1 Exponential model

An exponential distribution is a continuous distribution. All time data is discrete to some degree so modelling with a continuous distribution is not a problem.

Definition 4.1. *The cumulative distribution function of an exponential distribution is defined by:*

$$F(x;\beta) = \begin{cases} 1 - e^{-\beta x}, & x \geq 0 \\ 0, & \textit{otherwise}, \end{cases}$$

for some rate parameter $\beta > 0$.

The standard cumulative density function represents $\mathbb{P}(x < y)$ for some $x, y \in \mathbb{R}$ where $\mathbb{P}()$ denotes the probability operator. The 'cumulative' graphs in this document show $\mathbb{P}(x > y)$. The relationship between the two is:

$$\mathbb{P}(x > y) = 1 - \mathbb{P}(x < y).$$

The exponential function, e^{-x} is smooth around $x = 1$; however, our data jumps between 0 and 1 minute delays. What could be causing this effect? Here is a possible theory to why this occurs. Trains often wait in stations; many trains going through a station have an arrival time and a departure time. A train will not leave the station before its departure time, so in fact the zero minute delay covers all trains which arrive on the departure time or before.

We don't have any information on arrival data so we can not shift the delay axis and include any 'negative' delays. As a result of this for the purpose of fitting the exponential the zero delay value is ignored and we introduce another parameter to the model, which is an intercept. With the intercept parameter and the change from the cumulative density function, the function we are trying fit the data with becomes:

$$F(x;\beta,\alpha) = \begin{cases} 1-e^{-\beta x-\alpha}, & x>0 \\ 0, & \text{otherwise,} \end{cases}$$

This produces a valid cumulative density function for $\alpha, \beta > 0$. $F(X)$ is non-decreasing since:

$$F'(X=x) = \beta e^{-\beta x-\alpha} > 0, \text{ since } \beta > 0.$$

$F(X) \in [0,1]$ for all $X \in \{\text{delays}\}$ and the limit as $x \to \infty$ of $F(X) = 1$ since:

$$\begin{aligned} \lim_{x\to\infty} F(X=x) &= \lim_{x\to\infty} 1 - e^{-\beta x-\alpha} \\ &= 1 - e^{-\alpha} \lim_{x\to\infty} e^{-\beta x} = 1, \text{ since } \alpha, \beta > 0. \end{aligned}$$

4.1.2 q-Exponential model

The q-exponential is not a very widely known function. It a part of a whole family of *q-analogs* of widely known functions like, for example, the factorial and the binomial coefficient. There are several papers on this subject, which give helpful insight to why many of the properties possessed by functions are retained by their q-analogs [FW] [BTMA04].

Definition 4.2 (q-analog)**.** *For a given function, $f(x)$, a q-analog, denoted $f(x;q)$ or $f_q(x)$ of the function is any parametrization, q, which returns the original function as q tends to one, more rigorously:*

$$\lim_{q\to 1^-} f_q(x) = f(x), \ \forall x.$$

We shall be looking at the properties possessed by a q-analog of the exponential function. The exponential function is defined by:

$$\exp(x) = \lim_{n\to+\infty} \left(1+\frac{x}{n}\right)^n, \text{ for } x \in \mathbb{R}.$$

Definition 4.3 (q-exponential)**.** *The q-exponential is defined as:*

$$e_q(x) = (1+(1-q)x)^{\frac{1}{1-q}}$$

Theorem 4.1. *The q-exponential is a q-analog of the exponential function. In other words:* $\lim_{q\to 1} e_q(x) = \exp(x)$.

Proof. Starting with the left-hand side:

$$\begin{aligned}\lim_{q\to 1} e_q(x) &= \lim_{q\to 1}(1+(1-q)x)^{\frac{1}{1-q}} \\ &= \lim_{n\to+\infty}\left(1+\frac{x}{n}\right)^n \\ &= \exp(x)\end{aligned}$$

$\square$

In the majority of books the q-exponential is defined by a different formula [Que] [Mor04]. This is also a perfectly valid q-analog obtained from parametrize the infinite power series expansion of the exponential. The two q-exponential definitions are distinct and seems to be no function mapping between them; they are only equal to each other in the limit as $q \to 1$.

Let us look at how the choice of parameter q, effects the q-exponential function relative to the exponential function. Plots of the q-exponential function are shown in figure 4.1 for q between 0.5 and 2. The q parameter in q-exponential functions allow the control of the curvature on a logarithmic plot much like that observed in the distribution delays of train departing some of the stations. We may also add α and β parameters in to control the y intercept and gradient respectively.

As in the exponential model, the q-exponential model will be fit with a intercept parameter, on the cumulative data without the zero or greater delay probability. The q-exponential function we try to fit the data with has three parameters is defined by:

$$F(x;\alpha,\beta,q) = \begin{cases} 1-(1+(q-1)(\beta x+\alpha))^{\frac{1}{1-q}}, & x>0 \\ 0, & \text{otherwise,} \end{cases}$$

This is not a valid cumulative density function for all values of the parameters. F(x) needs to be in [0,1] for all values x ∈{delays} = $[0,\infty)$. This is equivalent to $e_q(-\beta x-\alpha) \in [0,1]$. This is shown by splitting the problem in to two cases, one where $q>1$ and the other where $q>1$. Suppose $q>1,\ \alpha,\beta>0$, then:

$$e_q(-x) = [1+(q-1)(\beta x+\alpha)]^{\frac{1}{1-q}},$$

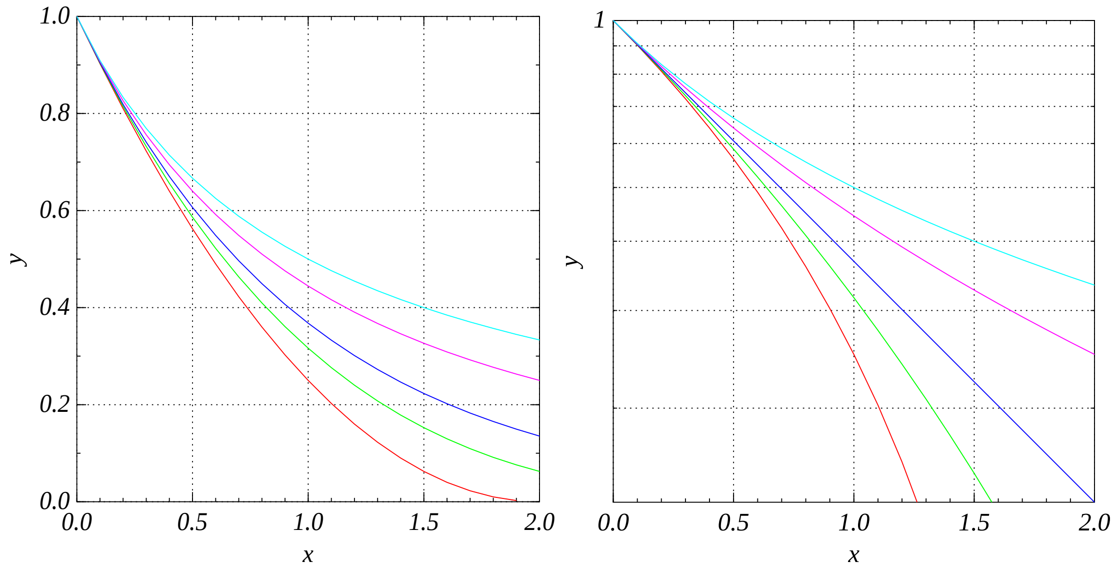


Figure 4.1: A plot of $e_q(-x)$ with parameter values, q = 0.5, 0.75, ≈1, 1.5 and 2

since everything is positive we have a number greater than 1 to a negative power. Taking the reciprocal gives a number belonging to $[0, 1]$ to a positive power, which has to belong to $[0, 1]$. In the case where $q < 1$ we get a number less than one to a fractional power. This is good, however there is no lower bound to this number as $x \to \infty$ as a result this can become negative. A negative number to a fractional power is a tricky concept and should be avoided. For $\alpha, \beta > 0$ and $q > 1$, $F(x) \in [0, 1]$ for all $x \in \{\text{delays}\}$. Let us now show that in the limit as x tend to infinity, F(x) for the specified parameter values, tends to 1. This is equivalent to saying $\lim_{x\to\infty} e_q(-\beta x - \alpha) \to 0$. For large x, the x dominates the inside of the power bracket. Defining $1/(1-q)$ as $-\gamma$ where $\gamma > 0$ since $q > 1$ gives:

$$\begin{aligned}\lim_{x\to\infty} e_q(-\beta x - \alpha) &= \lim_{x\to\infty} x^{-\gamma}\\ &= \lim_{x\to\infty} \left(\frac{1}{x}\right)^{\gamma}\\ &= 0.\end{aligned}$$

This fact is proof that the $\lim_{x\to\infty} F(x) = 1$ and therefore $F(x)$ is a valid cumulative density function.

4.2 Fitting the models

The models are fitted using the program called `nlfit_q_exp` and `nl_fit_exp` for their respective models. Both programs use the same implementation of an Levenberg-Marquardt algorithm [A.L05]. Given an initial values for a set of parameters an iterative procedure is applied to estimate the parameters by minimising the mean square error (MSE). From arbitrary initial conditions the program can only find a local minimum of the MSE; this is due to the non-linear nature of this the function we a trying to fit. Computationally we can get around this problem by starting the program off from various initial parameters.

Not all of the data contains the same amount of information. The information of the trains with a small delay is greater than that for a big delay. As the actual probability of an event occurring decreases, the amount of observations needed to obtain a valid estimate increases quickly. For example if the actual chance of a train being delayed more than 120 minutes is 1 in 1000 we really need millions of observations to accurately predict such a probability. This fact also means that the variance for a set number of observations is greater for an event which has a smaller probability; therefore, the fitted curve should be weighted towards the lower delay values.

4.2.1 Weighting

Weighting allows a statistician to assign a weight, w_i, to each point. For n points, these weights the minimum MSE towards the points with greater weighting. Point with greater weight are more significant when fitting an optimal curve. One appropriate assignment of weights is make the weight of an element of the data inversely proportional to its variance. The variance of the data is unknown; however, it should be inversely proportional to n_i, where n_i is the number of trains which make the ith point. The probability is proportional to n_i. The weights for each point should be proportional to $\mathbb{P}(\text{delay} \geq x_i)$; these weights do not have to sum to one, so need to be normalised. The normalised weights are:

$$w_i = \frac{\mathbb{P}(\text{delay} \geq x_i)}{\sum_{i=1}^{n} w_i}$$

These weights are the standard way weighting such a problem, however the fitting program uses a simpler method. The fitting program uses a constant multiple of $1/x$ as the weight for each point. Figure 4.2 shows plots of the best fits for a few data sets; more of these can be found in Appendix A. In the majority of cases the q-exponential seems to be the better fit. The distributions vary fairly drastically between stations, some of which produce particularly straight plots on an logarithmic scale; for these data set the optimal q-exponential is very close to one. For computational purposes when q is close to 1 a series expansion is used to avoid raising to a power which is shooting off to infinity.

Data sets where the optimal q-exponential q value is very close to 1, then you would expect the simpler exponential model to be preferred. A common trait in the fitting of the exponential curves are that for large delays the curve is consistently under the observed data, this would result in any model using this distribution to under-predict large delays. The q-exponential curves trend to do the opposite; for the big delays the fitted curve trend to leave the observed data, and has a tendency to over-predict large delays. These are not major problems as these traits are at very small probabilities, and can only be observed on a logarithmic plot.

4.3 Which model is best?

The *test statistic* used is obtained from the `nlfit` program [GSL], it is defined by:

$$\chi^2 = \sum_{i=1}^{n} \left[\frac{o_i - e_i}{w_i} \right]^2 ,$$

where n is the number of points, o_i and e_i are obtained and estimated values for the ith point and w_i are the weights. This is then divided by the number of degrees of freedom $n - p$, where p is the number of parameters in the model. This is preferred to the chi-squared goodness-of-fittest because it includes information about weights. We are not trying to justify why the model works, just

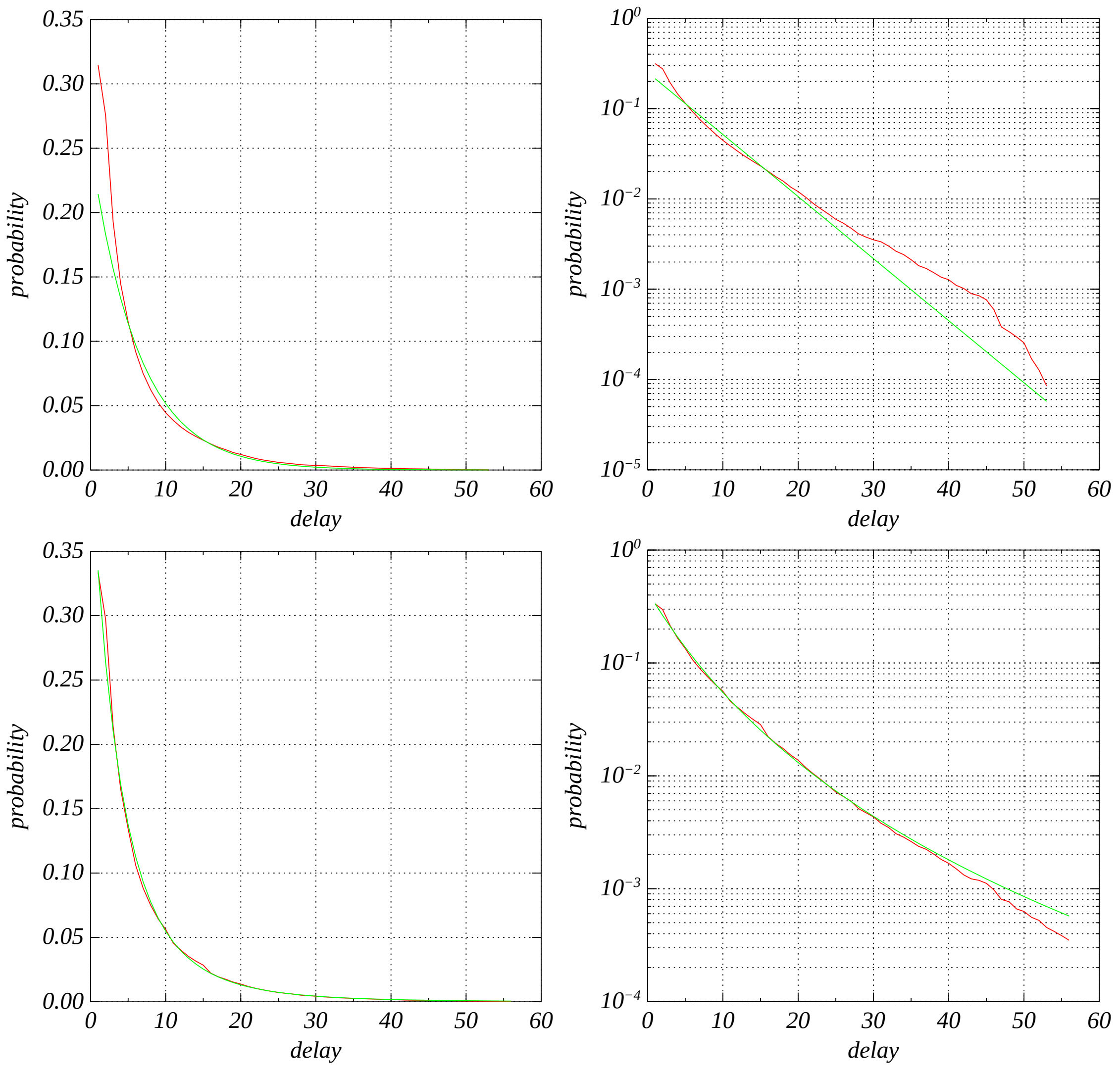


Figure 4.2: Exponential best fit (Top) and q-exponential best fit (Bottom) for trains leaving Manchester for Manchester Airport.

selecting it because it appear to fit the data well. This test allows us to choose relatively between the two proposed models.

The exponential model is *nested* within the q-exponential because it is the limiting case of the q-exponential where $q = 1$. To decide which is distribution is better, nesting of models needs to be considered. Assuming that we have the best q-exponential fit this has to have a significantly smaller χ^2 than the best exponential fit. What needs to be established, is whether the q-exponential is a significantly better fit; should this fail then the exponential distribution is the favored model.

Table 4.1 shows a list of routes with their χ^2/(degrees of freedom) values for both the exponential and q-exponential models of best fit. Plots of these fits can be found in Appendix A. In the vast majority of cases the q-exponential is clearly a much better model. There are a few values for the exponential fit which are very marginally smaller than the q-exponential, this is not because the model has a lower sum of square errors, which would mean there is something wrong with the fitting. In these cases, the q parameter in the q-exponential is very close to 1, and since the exponential model has one more degree of freedom, it is this that causes the smaller test statistic.

Cases where the exponential performs anywhere near the q-exponential is limited. There are quite a few routes where the test statistic for the exponential is massively above the relative q-exponential therefore, we will choose the q-exponential as the better of models for this data.

route	q-exponential	exponential
Manchester to Manchester Airport	2.3	23.1
Coventry to Birmingham	7.1	32.3
Coventry to Edinburgh	81.8	81.7
Coventry to Liverpool Lime Street	4.2	4.1
York to Edinburgh	14.9	126.4
York to Scarborough	5.2	7.5
York to London Kings Cross	13.2	241.4
Newcastle to London Kings Cross	5.5	94.4
Peterborough to Edinburgh	9.1	98.7
Doncaster to Edinburgh	11.2	248.6

Table 4.1: A table showing the χ^2/(degrees of freedom) values for the exponential and q-exponential best fits for given routes.

Chapter 5

Simulation

For simulation purposes, the cumulative density function has to be manipulated in to a probability density function. This process differs slightly between continuous time and discrete time. As the test data is in discrete minutes, the simulation should also be in discrete form. For discrete time, converting the cumulative density function to a probability density function is done using the following formula:

$$f(X = x) = F(X = x + 1) - F(X = x).$$

This also conveniently gets around the lack of continuity of the cumulative density function at $x = 0$, which would have been a problem in continuous time. For $f(X)$ to be a valid probability density function the probabilities should sum to 1 and be in the range [0,1] for all x:

$$\begin{aligned}\sum_{x=0}^{n} f(X = x) &= f(X = 0) + f(X = 1) + f(X = 2) + \cdots + f(X = n),\\ &= -F(X = 0) + \cdots + F(X = n - 1) - F(X = n - 1) + F(X = n),\\ &= -F(X = 0) + F(X = n),\end{aligned}$$

$$\begin{aligned}\sum_{x=0}^{\infty} f(X = x) &= \lim_{n\to\infty} f(X = n),\\ &= -F(X = 0) + \lim_{n\to\infty} F(X = n),\\ &= 0 + 1 = 1.\end{aligned}$$

Since $F(X = x)$ is in $[0, 1]$ for all x, and $F(X)$ is non-decreasing, $f(X = x) = F(X = x+1) - F(X = 0)$ is in $[0, 1]$. f(X) is a valid probability density function.

The expectation and variance of the model can be obtained from the probability density function via the following formulas:

$$\mathbb{E}(X^n) = \sum_{x=0}^{\infty} x^n \times f(X = x),$$
$$\text{var}(X) = \mathbb{E}(X^2) - (\mathbb{E}(X))^2.$$

These infinite sums need to converge. For the q-exponential this is not guaranteed for all the parameters that produce a valid probability density function. For computational purposes an infinite sum is not ideal, because the random number generator would need to produce a random number to a infinite number of decimal places. An arbitrarily big delay is not feasible, as the train company would cancel any train with a huge delay. As a result of these factors, the tail of the distribution should be cut off at some appropriate delay size. This should be determined by the largest delay seen in the data set. The `get_train_data.py` program is not perfect, as very occasionally it will produce a delay of greater than 1000, which is impossible since a train with such a delay would arrive after the data is written; as a result of this the distribution is cut at the greatest delay less than 1000.

For testing purposes the data should be split into training data and testing data. As the Manchester to Manchester Airport route has the most trains, this is a sensible example to choose. The training data consists of all of the trains departing Manchester for Manchester Airport from December 2005 to the start of July 2006 and will be used to select the optimal parameters. Figure 5.1 shows fitted curve. The fitted parameter values are passed to the `simulate.py` program, which calculates the probability density function and simulates a set number of iterations. Figure 5.2 is a comparison of the test data and a simulation of the same number of trains. The test data is actual data of trains leaving Manchester for Manchester Airport from July 2006 to August 2006. The two plots do not look too dissimilar; the real data has quite a number of big delays towards the end of the data, this indicate that the system went through a bad patch, which can not be predicted by our model since each element of the prediction is independent. The means and variances are pretty well estimated:

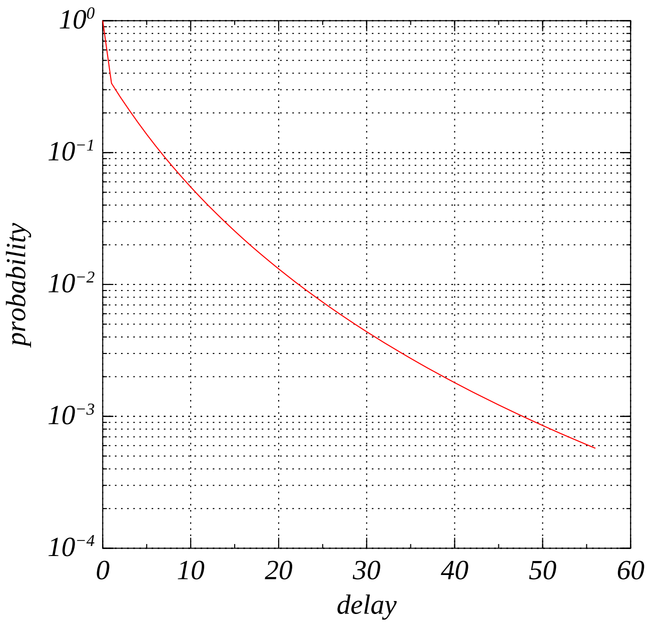


Figure 5.1: A graph showing the estimated cumulative distribution of trains leaving Manchester for Manchester Airport.

	mean	variance
real	1.804	19.503
simulated	1.712	20.415

The randomness of the random number generator used can be checked here since the distribution used by the simulator has an expectation. The maximum single delay was set at 180 minutes, therefore:

$$\mathbb{E}(\text{delay}) = \sum_{x=0}^{180} x f(x) = 1.675$$

$\mathbb{E}(x^2) = 22.521$, hence the calculated variance should be:

$$\text{Var}(x) = \mathbb{E}(x^2) - (\mathbb{E}(x))^2 = 19.715.$$

The expected mean and variance are very close to that of the simulation; this suggest that the random number generator used is unbiased and the test data set is large enough to give accurate results.

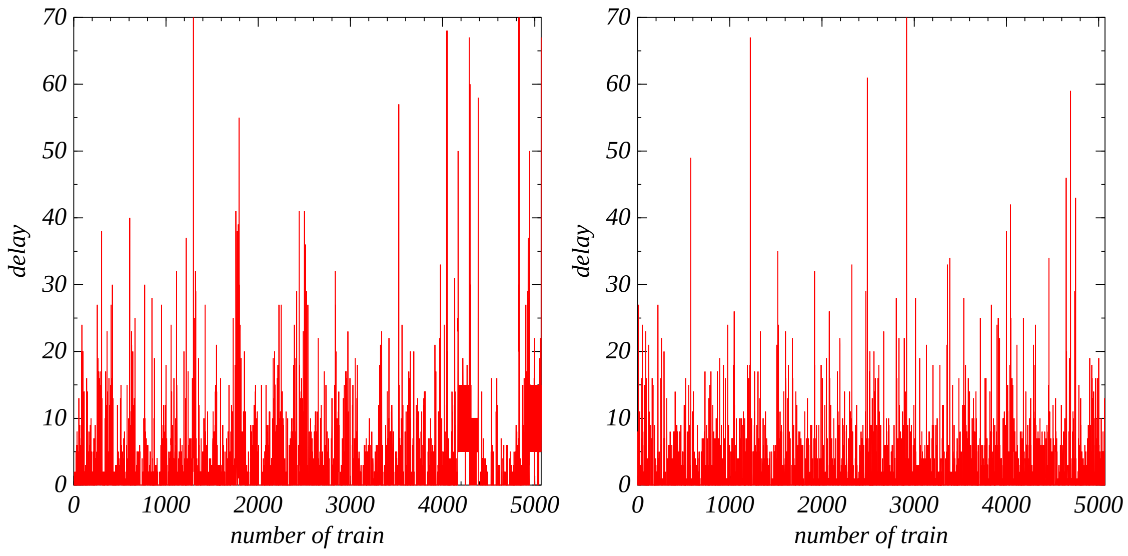


Figure 5.2: Fractal plots showing the delay of train with relation to the order in which they occurred for actual data (left) and simulated (right)

An interesting observation in the real data of figure 5.2 is two block of delays in the bottom right of the plot. Looking at the data series which produces the plot there is a sequence of 5,10 and 15 minute delays forming a regular pattern one after the other. This is very suspicious and probably not accurate. This shows that the data collection method does occasionally go wrong. For purposes of computing the averages, these noisy points were removed.

Chapter 6

Application

Now that we can accurately model various distributions of train data of trains departing stations around the country, we need some way of making this data useful. Suppose we wanted to leave London Kings Cross and travel north to Peterborough. The information we need is is the eventual destination of the train that is planned for travel, in this case Edinburgh. We then obtain the distribution of past trains leaving Peterborough for Edinburgh, with the assumption that the delay of the departure time is the same as the arrival time.

This works well for this case, however if you are traveling the other way and want to leave York for Peterborough. The destination of the train is London Kings Cross but lots of local trains travel from Peterboroughto London Kings Cross. As discussed in Chapter 2, these local trains interfere with the mean, and hence distribution of delays. We want to narrow down as much as possible trains similar to the one intended for travel.

Figure 6.1 shows the distributions of trains leaving their respective stations heading for London Kings Cross originating from Glasgow Central. This data can be used to give information on the chance of arriving at one of the stations from any of the proceeding stations. We are not dealing with conditional probabilities, say what the chance of delay going from York to Peterborough given our journey begins on time from York. The distance of our journey is not important, however the distance of trains journey to our destination station is.

The means of the distributions get steadily bigger, with the exception of Doncaster. This is in keeping with the mean delay result in chapter 2. Table 6.1

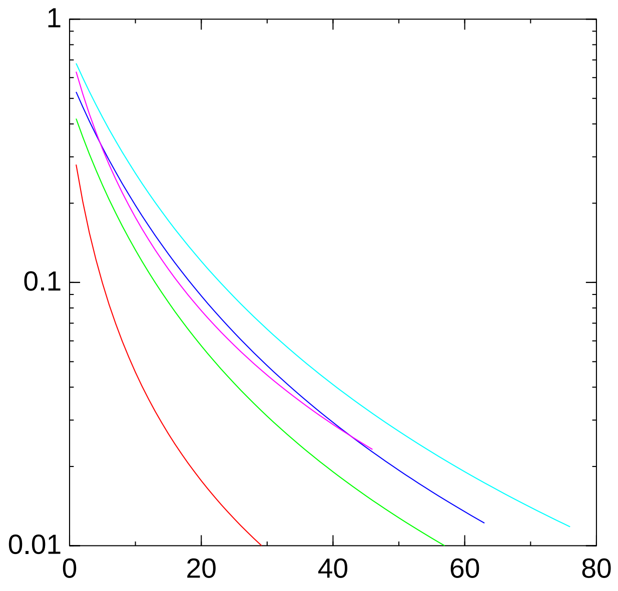


Figure 6.1: Cumulative plots of the estimated distribution for Edinburgh, Newcastle, York, Doncaster and Peterborough.

shows the parameter estimates with their standard errors. Edinburgh is by far the worst fit since the standard error values are significantly higher for all of the parameters than in the other stations.

This limits the choices of station considerably, since there is only the GNER line where we have collected data from enough stations to make train tracking possible. For travel to stations not on the GNER line, the non-filtered data is the best estimate we can do. There is no reason why `traintracker.py` should not work on other lines since the line is set by list of stations and minimum times. Extending the number of stations the data is collected from would allow expansion and more accurate results to other lines, where the destination is near the end of the line.

	q	α	β
Edinburgh	1.53±0.03	1.1 ±0.2	0.70±0.1
Newcastle	1.42±0.01	0.81±0.03	0.23±0.007
York	1.37±0.01	0.55±0.02	0.16±0.004
Doncaster	1.53±0.02	0.26±0.07	0.26±0.01
Peterborough	1.38±0.01	0.27±0.03	0.14±0.004

Table 6.1: A table showing the parameter values with their standard errors for the plots in figure 6.1

Chapter 7

Concluding remarks

We began with the intention of analysis and modelling the performance of trains on the Railtrack network. We showed that while the majority of the trains reach the performance target of 95% being less than 5 minutes late, the actual performance of different stations and route varies quite considerably.

The performance of long distance trains is also masked by the regular short distance trains, Peterborough meets the 95% statistic. However, my program `traintracker.py` indicated the *average* of train arriving from Edinburgh to be substantially over 5 minutes. We have shown that the effects of one delayed train can last up to a period of several hours, but these effects weaken over the course of a few days. The simulation of the future trains showed gave further weight to the q-exponential accurately modelling the data and produced a very similar plot to the actual data, albeit with the autocorrelation property of the data lost.

This project leaves various paths for future research. There were a few limitations already discussed, many the lack of stations with a substantial amount of available data. This is an obvious area of improvement, which would possibly lead onto the analysis of routes involving multiple trains and the change between them.

Appendix A

More figures

This appendix is purely plots. There is another distance-time plot for 13th February 2006, and plots of the observed cumulative density, exponential fit and q-exponential fit of routes used in the comparison of the exponential model and q-exponential model. The routes are labeled in the figures and are the same for each figure.

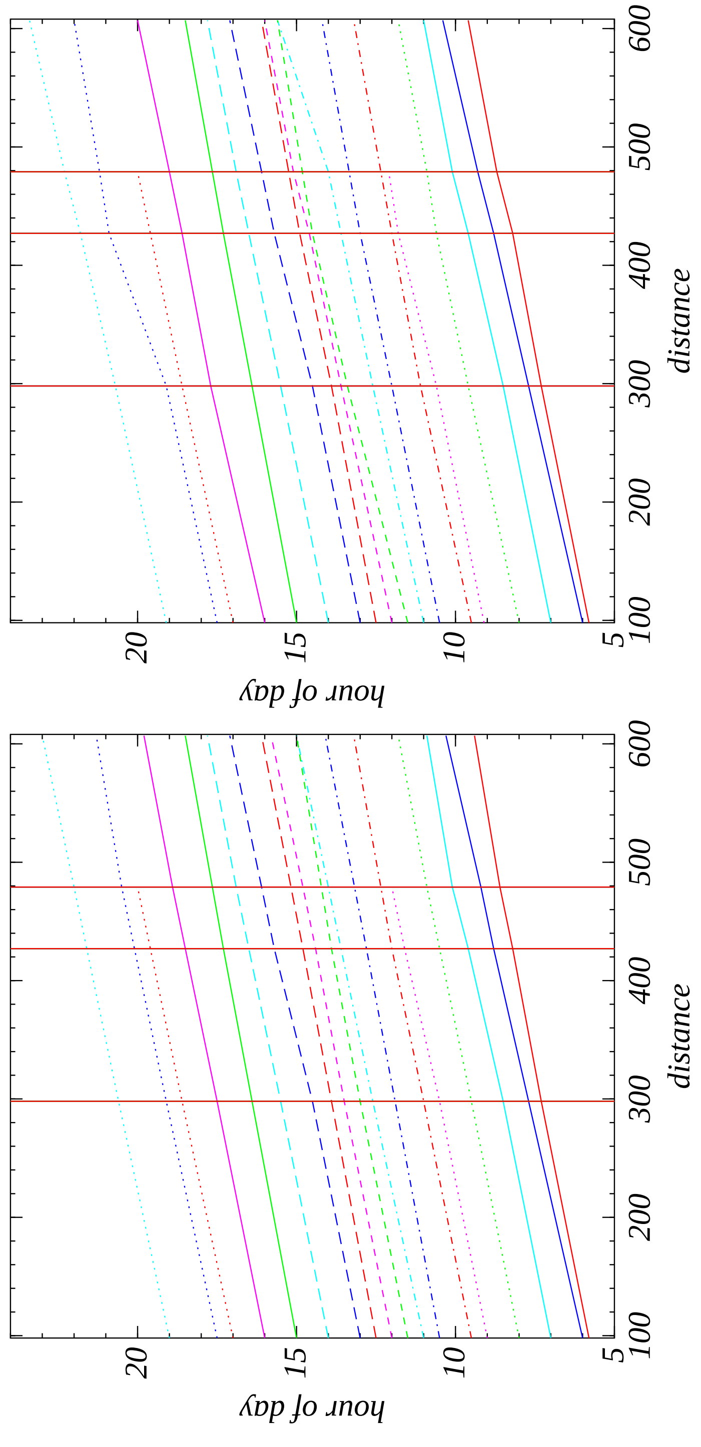


Figure A.1: A distance-time plot of timetabled (left) and observed (right) trains from Glasgow Central to London Kings Cross on the 13th February 2006.

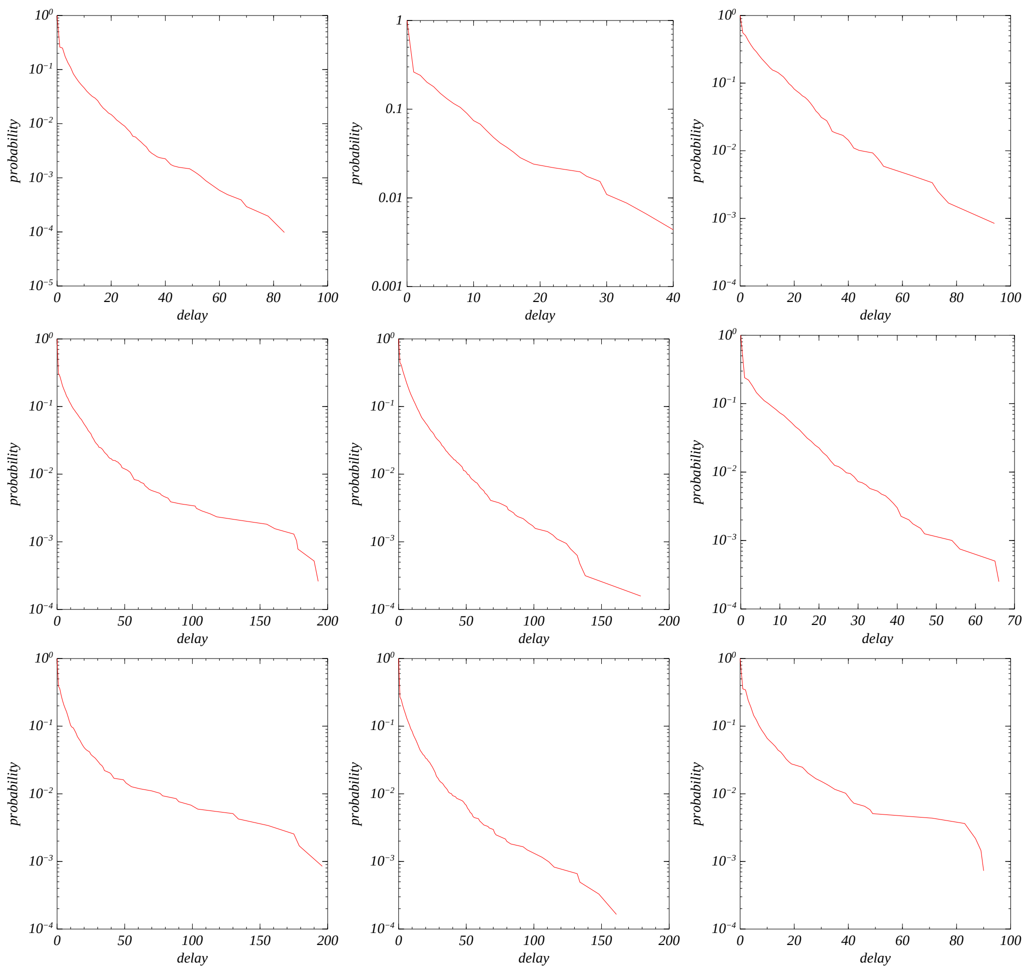


Figure A.2: Cumulative distribution plots for trains leaving (**Top**) Coventry to Birmingham, Liverpool Lime Street and Edinburgh(left to right). (**Middle** York to Edinburgh, London Kings Cross and Scarborough (left to right). (**Bottom**) Doncaster to Edinburgh, Newcastle to London Kings Cross and Peterborough to Edinburgh(left to right).

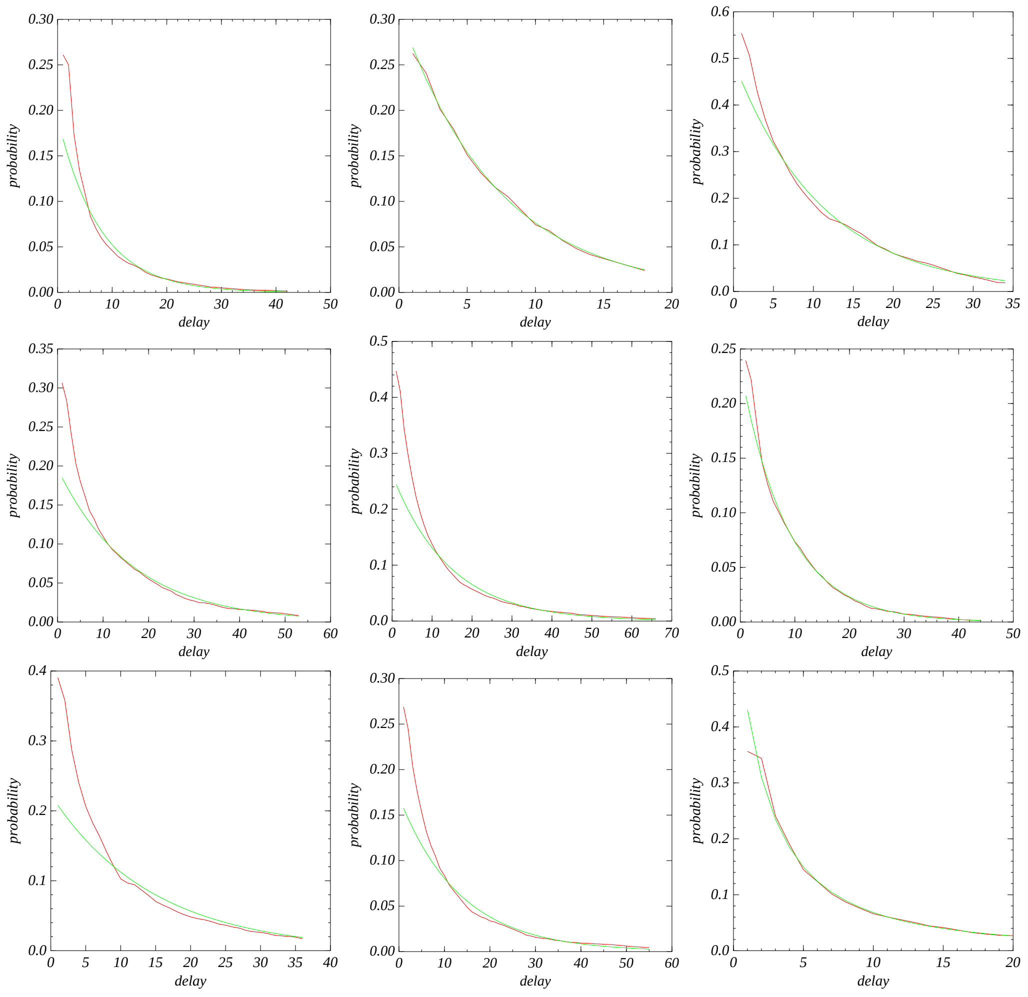


Figure A.3: Plots showing the fitting of the exponential function to the cumulative distribution of different routes (orientation as in figure A.2).

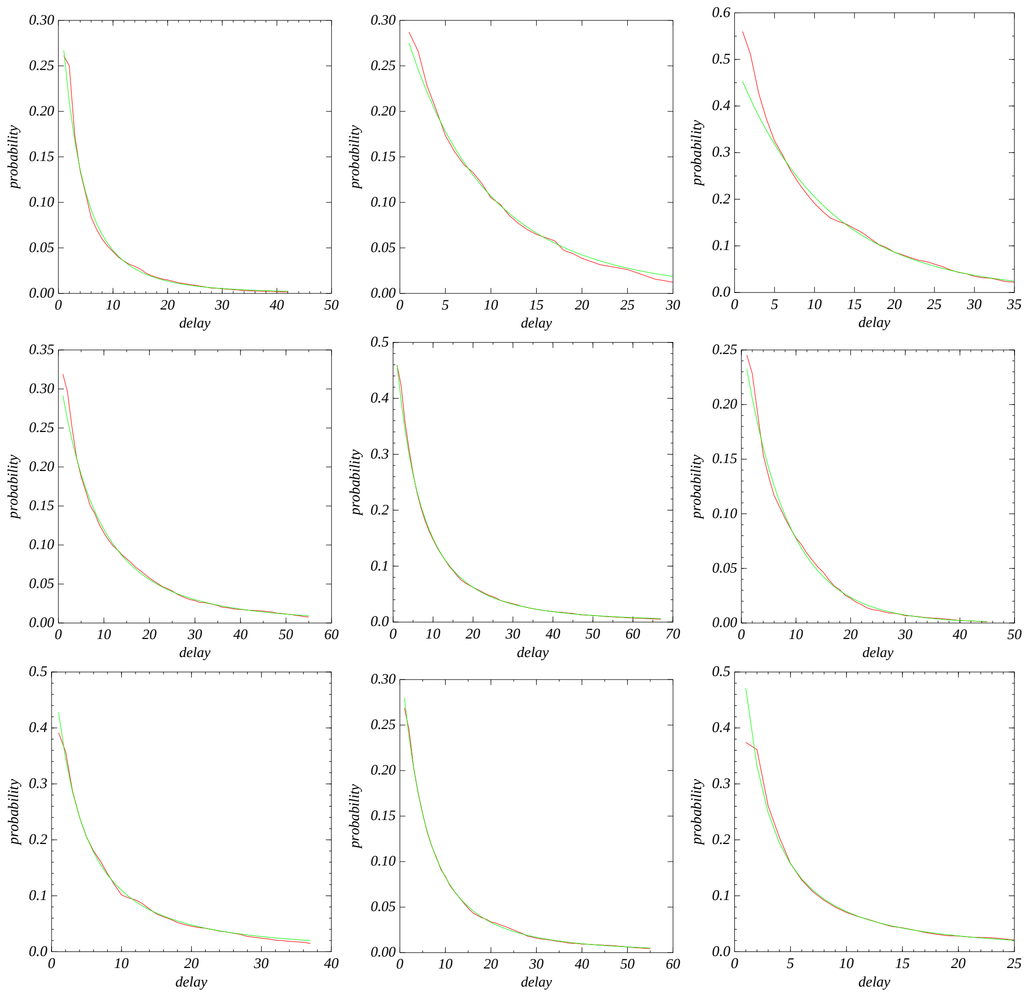


Figure A.4: Plots showing the fitting of the q-exponential function to the cumulative distribution of different routes (orientation as in figure A.2).

Appendix B

Program listings

I wrote several programs for the is project. Here are streamlined versions of a few of the more important ones.

```
#! /usr/bin/python
#to make distance-time plots like in Chapter 2, use the linux command
#./traintracker.py –station=EDB –destination=KGX | graph -Tps -C
#the path variable needs to be changed for the program to locate
#the data/ directory.

import re
from time import *
from commands import getoutput
from sys import stderr,exit,argv
import getopt

from CRS_dict import station2CRS,CRS2station # from CRS_codes-1.0/CRS_dict.py
from line_dict import line2stations

def findstarts(station,destination):
  lfn=path+'/data/'+station+'_'+date+'.dat'
  try:
    f=open(lfn,'r')
  except:
    print>>stderr, 'No dat file'
    return None
  info=re.compile('(?P<time>\d\d:\d\d)\t'+CRS2station[destination])
  page=f.read()
```

```
    return info.finditer(page)

def traintracker(start_time,start_station,destination):
    time=str2sec(start_time)
    times=[]
    for i,station in enumerate(stations):
        if i==0:
            r=searchstation(station,sec2str(time),0,destination)
            if r: times.append(r[0])
            time+=60*mintimes[i]
        else:
            trains=searchstation(station,sec2str(time),variation,destination)
            trains=choosetrain(trains,times[−1])
            if trains:
                times.append(trains)
                time=60*mintimes[i]+str2sec(times[−1][1])
            else:
                time+=60*mintimes[i]−300
    return times

def searchstation(station,time,span,destination):#span in minute
    first=1
    start_time=str2sec(time)
    end_time=start_time+60*(span+1)
    time=str2sec(time)
    info=re.compile('(?P<time>\d\d:\d\d)\t'+CRS2station[destination])
    list=[]
    try:
        f=open(path+'/data/'+station+'_'+date+'.dat')
    except:
        if first:
            print>>stderr, "Couldn't open file for ", CRS2station[station]
            first=0
        return list
    for line in f:
        x=info.match(line)
        if x:
            if str2sec(x.group('time')) in range(int(start_time),int(end_time),60):
                fields=line[:−1].split('\t')
                list.append((station,fields[0],fields[−1]))
                time=str2sec(x.group('time'))
```

```
      elif str2sec(x.group('time'))>end_time:
        break
  return list

def choosetrain(trains,last_stop):
  if len(trains)==0: return None
  if last_stop==None:
     print>>stderr, 'No last stop'
     exit(3)
  last=str2sec(last_stop[1])
  for i in range(len(trains)−1,−1,−1):
    timetaken=(str2sec(trains[i][1])−last)/60
    time,stps=find_time(last_stop[0],trains[i][0])
    test=(timetaken−time)/time
    testdelay=timetaken+int(trains[i][2])−int(last_stop[2])
    if timetaken/time>1.35 or testdelay<(time*0.9−5*stps):
      del trains[i]
  j,delay=0,0
  if int(last_stop[−1])==−1:
    return None
  for i,train in enumerate(trains): #biggest delay is always selected
    if train[−1]>delay:
      delay=train[−1]
      j=i
  try:
    return trains[j]
  except:
    i=1

def printstd(times):
  dist=0
  for stop in times:
    if type==0: hour=(str2sec(stop[−2])+60*float(stop[−1])−str2sec('00:00'))/3600
    else: hour=(str2sec(stop[−2])−str2sec('00:00'))/3600
    distance=find_dist(stop[0])
    print '%i\t%.1f'%(distance,hour)
  print

def printcontour(times,number):
  dist=0
  for stop in times:
        time=(str2sec(stop[1])−str2sec('00:00'))/3600
        delay=int(stop[−1])
        distance=find_dist(stop[0])
        print '%i\t%.2f\t%i'%(distance,time,delay)

def fileprint(times,start_time):
    savepath=path+'/testdata/'+date+'.dat'
    f=open(savepath,'a')
    for t in times:
        f.write('%s\t%s\t%s\n'%(t[0],t[1],t[2]))
    f.write('\n\n')

def find_dist(stop):
    j=line2stations[line][0].index(stop)
    return sum(line2stations[line][1][0:j])

def find_time(start,stop):
    i=stations.index(start)
    j=stations.index(stop)
    return sum(mintimes[i:j]),j−i

def str2sec(time): #Needs global variable date
    return mktime(strptime(date+'_'+time,'%Y_%b_%d_%H:%M'))

def sec2str(secs):
    return strftime('%H:%M',localtime(secs))

def addday(date,fmt='%Y_%b_%d'):
    t=strptime(date,fmt)
    t=localtime(mktime(t)+(60*60*24))
    return strftime(fmt,t)

def stations_select(line,start,stop):
    x=line2stations[line]
    stations=x[0]
    distances=x[1]
    mintimes=x[2]
    try:
        i,j=stations.index(start),stations.index(stop)
        stations=stations[i:j+1]
        distances=distances[i:j]
        distances.append(0)
        mintimes=mintimes[i:j]
        mintimes.append(0)
    except:
        print>>stderr, 'Could not find the stations on the line'
        exit(3)
    return stations,distances,mintimes

def run(station_code,dest_code):
    times=findstarts(station_code,dest_code)
    try:
        for i,y in enumerate(times):
            time=y.group('time')
            if type==0 or type==3: printstd(traintracker(time,station_code,dest_code))
            if type==1: printcontour(traintracker(time,station_code,dest_code),i);
            if type==2: fileprint(traintracker(time,station_code,dest_code),time)
            print
    except:
        print>>stderr,'No trains from ',CRS2station[station_code],' to ',
                    CRS2station[dest_code],' on ',date

#Global Info and defautls
path='/home/mark/train_data-1.0'
date='2005_Dec_29'
line='gner'
type=0
variation=25

try:
    opts,args=getopt.getopt(argv[1:],'hS:l:D:d:t:',
    ['help','station=','line=','destination=','date=','type='])
except getopt.GetoptError:
    print>>stderr, "Need opts."
    exit(1)

for o,a in opts:
    if o in ('-S','--station'):
        station=a[:3]
        try: test=CRS2station[a]
        except:
            print>>stderr, 'Invalid station code, should be like KGX'
```

```
      exit(1)
  if o in ('-l','--line'):
    line=a
  if o in ('-D','--destination'):
    destination=a
    try: test=CRS2station[a]
    except:
      print>>stderr, 'Invalid destination code, should be like KGX'
      exit(1)
  if o in ('-d','--date'):
    date=a
    try:
      test=addday(a)
    except:
      print>>stderr, 'Date invalid date, using default: 2005_Dec_29'
      date='2005_Dec_29'

  if o in ('-t','--type'):
    if a=='c': type=1 #contour
    elif a=='f': type=2 #file
    elif a=='t': type=3 #timetable
    else: type=0 #stdoutput

stations,distances,mintimes=stations_select(line,station,destination)
run(station,destination)
```

```
#! /usr/bin/env python

#./get_distribution.py –station=MAN –destination=MIA to produce
#distributions, use -e to change from q-exp to exp.

from commands import getstatusoutput,getoutput
from sys import exit,argv,stderr
import getopt
import re

path=getoutput('pwd')
exp=False
```

```
try:
  opts,args=getopt.getopt(argv[1:],'hs:d:e',['help','station=','destination='])
except getopt.GetoptError:
  print>>stderr, "Need opts - use '-h' or '--help' for more information."
  exit(1)

for o,a in opts:
  if o in ['-h','--help']:
    print>>stderr, "'-s' station code: '-d' destination(s) code"
  if o in ['-s','--station']:
    station=a
  if o in ['-d','--destination']:
    dests=a
  if o in ['-e']:
    exp=1
if not station: print>>stderr,'needs station code ',
if not dests: print>>stderr,'needs destination code'

s,o=getstatusoutput('./autocorre.py -s'+station+' -d'+dests+'
                    | colex 2 | ./cumulative.py >| temp.dat')
s>>8
if s:
  print>>stderr,'...failed to create temp.dat'
  exit(1)

f=open(path+'/temp.dat','r')
g=open(path+'/temp1.dat','w')
g=open(path+'/temp1.dat','a')

last_min=0
num=False

x=f.read().split('\n')
for i,a in enumerate(x[:-1]):
  min=int(a.split('\t')[0])
  if last_min+2<min:
    num=i
    break
  last_min=min
if not num: num=len(x)
for i in x[1:num]:
  g.write(i+'\n')
g.close()

s,o=getstatusoutput('rm temp.dat')
s>>8
if s:
  print>>stderr,'Unable to remove temp.dat, remove manually'

if exp==1:
  s,out=getstatusoutput('cat temp1.dat | nlfit_exp | gr.py -x1 -y2,3')
  print out
else:
  s,out=getstatusoutput('cat temp1.dat | nlfit_q_exp | gr.py -x1 -y2,3')
  print out
```

```
#! /usr/bin/env python

# simulate.py generates a series of random numbers and applies
# them to a model set by beta, q and c paramters.

from sys import stdin,stderr
from random import random

def eq_x(x,beta,p,c):
  a=1+(beta*x-c)*(1-p)
  b=1/(1-p)
  if not a<0: eq_x=pow(a,b)
  return eq_x

beta=-0.34599
q=1.22727
c=0.99605
iters=5069
maxdelay=600
i=1
prob=[1-eq_x(1,beta,q,c)]
last=1-eq_x(1,beta,q,c)
e_x=0.0
e_x2=0.0
```

```
# produces the model in the variable prob from the given parameters
for i in range(1,maxdelay):
  x=eq_x(i,beta,q,c)-eq_x(i+1,beta,q,c)
  e_x+=i*x
  e_x2+=i*i*x
  prob.append(x+last)
  last=x
  i+=1
print>>stderr,'%.2f\t%.2f'%(e_x,e_x2)

# generates random numbers on the unit interval and applies them to prob
average=0.0
for i in range(0,iters):
  rand=random()
  for j,p in enumerate(prob):
    if rand<p:
      print '%i %i'%(i+1,j)
      average+=(j-average)/(i+1)
      break
```

Bibliography

[AL83] B. Abraham and J. Ledolter. *Statistical Methods for Forecasting*. 1983.

[A.L05] Manolis I. A.Lourakis. A Brief Description of the Levenberg-Marquardt Algorithm Implemented by levmar. Febuary 2005.

[BJ94] G. E. P. Box and G. M. Jenkins. *Time Series Analysis, Forecasting and Control, 3rd ed.* 1994.

[BTMA04] Ernesto P. Borges, Constantino Tsallis, Jose G. V. Miranda, and Roberto F. S. Andrade. Mother wavelet functions generalized through q-exponentials, 2004.

[FW] David Fairlie and Ming-Yuan Wu. The Reversed q-Exponential Functional Relation.

[Gib] Richard Gibbens. www.cl.cam.ac.uk/users/rg31/m25.

[GNE] GNER timetable valid from 12 June to 10 December 2005.

[GSL] http://www.gnu.org/software/gsl/.

[Mor04] Kent E. Morrison. q-Exponential families, 2004.

[Pol05] Katherine S. Pollard. Test statistics null distributions in multiple testing: Simulation studies and applications to genomics. July 2005.

[Que] C. Quesne. Jackson's q-exponential as the exponential of a series. ULB/229/CQ/03/2.

[Sne89] George W. Snedecor. *Statistical Methods, Eighth Edition*. 1989.

[SW03] M. Schader and W.Gaul. *Between Data Science of Applied Data Analysis*. 2003.

[Tho02] Thomas Cook European Timetable, 2002.