Investigating the Variability of Railway Performance

Ethan Mattison (200562322)

May 2023

BSc. Computer Science

Project Supervisor – Pedro Pinto Da Silva

Word Count: 13,125

Abstract

In the realm of transportation, travel times must be as consistent as possible; finding and solving reasons for large or increasing travel times can be time and money-saving for companies as well as passengers. In addition, transport demand continues to increase congestion, making the flow of trains harder to manage.

This dissertation uses a vast dataset of train signals from NetworkRail to explore how different aspects of the data affect each other, primarily using data visualisation to find new information about travel times that could lead to more efficient travel. Moreover, this research emphasises the importance of presenting complex data findings in an easily understandable format, especially for people who may not be well-versed in how railway networks work.

The findings of this dissertation show that season and time of day, in general, have a minute impact on the time taken for a train to progress a set distance and rather it is the type of train that has a greater influence on how long it takes for the train to travel the length of a berth. Faults in signalling can also cause misinterpretations of travel times.

Acknowledgements

A great deal of gratitude has to be given to the project supervisor, Pedro Pinto Da Silva, for his guidance and support throughout my research. I would also like to thank NetworkRail, for providing me with the railway data to use for my project.

Declaration

“I declare that this dissertation is my own work, conducted under the guidance of Pedro Da Pinto Silva at Newcastle University, except where otherwise stated.”

-Ethan Mattison
-May 2023

Table of Contents

Table of Figures

1 Introduction

Motivation & Rationale

Railway transportation is an essential part of modern society, with trains being integral for not just companies' deliveries of goods but for the general public as well. Trains running consistently and on time can be crucial for the working populations income. According to the government Department of Transport, in 2019, around 57% of all trips by rail were for commuting or business purposes [1]. Being late for work or a meeting costs companies and workers a great deal of time and money every year.

Train signalling plays a crucial role in ensuring these railway networks run safely and efficiently. Evaluating, understanding, and analysing the patterns of existing data from trains that are running daily and visualising them can help future trains run more smoothly.

This dissertation uses the Python programming language, which gives many possibilities for data analysis and visualisation due to its many libraries: Matplotlib, Seaborn, Datetime, etc. Finding the best ways to find and present interesting new patterns within the data is a big part of the progression of this work and is made possible by these vast and sometimes complex Python libraries. Being able to plot various graphs and vigorously study them to understand why things look the way they do within the graphs is another important part of this dissertation.

Currently, few reports use data visualisation to explore large railway datasets, which presents a gap in the current understanding of how to keep the travel times of trains as consistent as possible.

This dissertation continues the initial work from Pedro Da Pinto Silva [10], especially his work presenting the metric of TSAR (Time Signal at Red), which is the time that a train spends with a "red" signal, meaning no other trains can enter its berth to prevent trains from interacting. TSAR can be an important value to understand when looking at railway performance. It refers to the signalling of railways and is discussed in depth later in this dissertation along with other metrics, including why each one is important and what they all show about performance. Therefore, the idea of TSAR is new to railway data and has not been widely explored, as in this project.

Current metrics for analysing railway performance, such as simply the delay times, are limited when trying to discover patterns of behaviour in trains, as they do not give a full picture of the cause of this delay. However, looking at TSAR can provide relationships between train signals and where the issue has occurred in the train's journey to its next signal. Furthermore, train delays are usually only for a train's final destination or when it has stopped at a station, but TSAR is accounted for at every signal of a train's journey.

1.1 Overall Project Aim

The overall aim of this dissertation is to investigate the variability of railway performance using Python data visualisations and find patterns of behaviour to try and reduce the travel time of trains.

1.2 Aims & Objectives

1. Use the given dataset of railway performance to visualise and evaluate whether any locations exhibit higher or lower travel time and TSAR values.

2. Use the given dataset of railway performance to visualise and evaluate how train types and sizes affect travel times and TSAR, decide which is more important.

3. Summarise 3 or 4 pieces of previous research on railway performances with data science.

4. Use a dataset of railway performance to visualise and evaluate how much the weather season will affect travel times and TSAR.

5. Evaluate how much time different train types spend with different aspect signals. Do specific locations give more Y signals?

6. Use the given dataset to check for inconsistencies or failings in the time taken to switch signals via visualisations.

7. Explore how the time of day affects railway performance. Do peak times give higher travel times?

1.3 Objective Descriptions

Objective 1: Use Python data libraries to plot visualisations of travel time and TSAR against every berth (location) to see which exudes the highest value. Also compare the berths with stations and which station has a higher TSAR.

Objective 2: Use Python data libraries to plot visualisations of train types (fast trains and slow trains) against travel times and TSAR. Different trains also have their own time profiles, as they are also powered differently with some trains being electric and some being diesel fuelled. Which of these train attributes affects travel times the most?

Objective 3: Although there is limited previous research on TSAR, there are still numerous studies of railway performance using data science. Summarising this previous research is important for making sure that the work in this dissertation is hitting on new ideas of railway performance.

Objective 4: Compare the data from the two seasons given in the dataset (spring and autumn) and use Python visualisation libraries to show which of these seasons exhibits higher travel times and TSAR times; are these findings due to weather?

Objective 5: Use Python data visualisation libraries to plot graphs showing how often trains spend with different warning signals. These signals will be the permissive signals that are seen by the driver and prevent trains from going too fast and keeps trains from being on the same berth, which includes G, Y, and YY signals. Find out which signal is most common and if different train types or locations show more or less of a specific signal.

Objective 6: Make use of Python visualisation libraries to make plots comparing offset times, which is the time taken to switch from a red (warning) signal back to a proceed aspect, to each of the signal IDs to check for inconsistencies in signalling. The same needs to be done for onset time, which is the time taken to switch from a proceed aspect to red. Can abnormal offset and onset times affect travel time?

Objective 7: Use the times for each signal in the dataset to split the data into times of the day and use visualisations to compare travel duration and TSAR to different intervals of a 24-hour day. Do travel times increase at peak periods of the day?

1.4 Dissertation Structure

Section 1: This section is the Introduction to the project, which includes the motivation and rationale, the overall aims and objectives of this dissertation, and their full descriptions of how they are to be completed.

Section 2: The background review, includes an evaluation of the sources used to do background research on railway signalling and exploratory data analysis. Furthermore, this section covers what each part of the dataset is, and what it means and describes its relationship with how railway networks are run. Understanding how the train signalling system works is important context to the data visualisation plotting.

Section 3: Programming and implementation, this section explains all of the programming and functions used within the programming as well as why these methods of plotting were used as opposed to other methods.

Section 4: Results and evaluation, this section shows figures of all of the visualisations and explains the meaning behind these graphs. Understanding what results are shown and suggesting reasons for them.

Section 5: Conclusions, this section explains the overall conclusions that have been obtained from the data visualisation and data exploring done within the project, it will also go through the original objectives and answer any posed questions as well as explain any failings.

1.5 Changes Made from the Proposal

The changes made from the project proposal to this dissertation relate to the motivation, aims, and objectives sections. Firstly, some small changes to the motivation for the project were made as the proposal had been written before more extensive research into the background of the field.

Approaching the railway performance area of data science in a way that had not been done before became the primary aim. Especially because TSAR is a new concept that can be thoroughly explored.

In the aims and objectives for the proposal, machine learning was included to try and predict TSAR and other data values based on various changing attributes. This part of the dissertation was scrapped due to a few reasons: firstly, it made much more sense to have the entire code base as a data visualisation project to keep consistency and focus.

Focusing on AI based prediction made the project more convoluted. Further exposure to the dataset made it apparent that there were more aspects to try and visualise that weren't being accounted for in the original proposal.

The objectives that were added to the dissertation include the visualisation of the aspect signals and of the offset and onset times, which became important during the research and data exploration phases.

2 Background Review

This dissertation uses exploratory data analysis to find out about railway performance. It is important to first establish why exploratory data analysis is useful.

Exploratory data analysis involves trying to understand and summarise the main attributes of a piece of data. However, it is practically impossible to make sense of data sets containing more than a handful of data points without the help of computer programmes [8]. Because of this, using programming languages like Python can be prolific for finding out new things about a piece of data. The following review section will discuss the libraries, functions, and plots used to make sense of the railway dataset.

To be able to maximise what is learned from a piece of data, adherence to two main principles is commonly required: scepticism and openness. Scepticism is when one should be sceptical of measures that summarise data since they can often be misleading, and openness is being ready for unanticipated results that lead to answers and findings [9]. This dissertation uses these principles within its conclusions and visualisations of the data.

2.1 Railway Signal Networks

Understanding how railway signal networks work is integral to being able to draw conclusions from the data visualisations of train data within this project.

The main terminology that is important to grasp to be able to understand the ongoings of the data visualisations within this project is: firstly, “berth,” which refers to a section of a track protected by signals so that only one train can be within a berth at one time. Figure 1 shows how these signals come before the entry to a berth to tell other trains whether that train has left the berth. Figure 1 also shows how some berths contain stations within them, which was important to distinguish when comparing travel times.

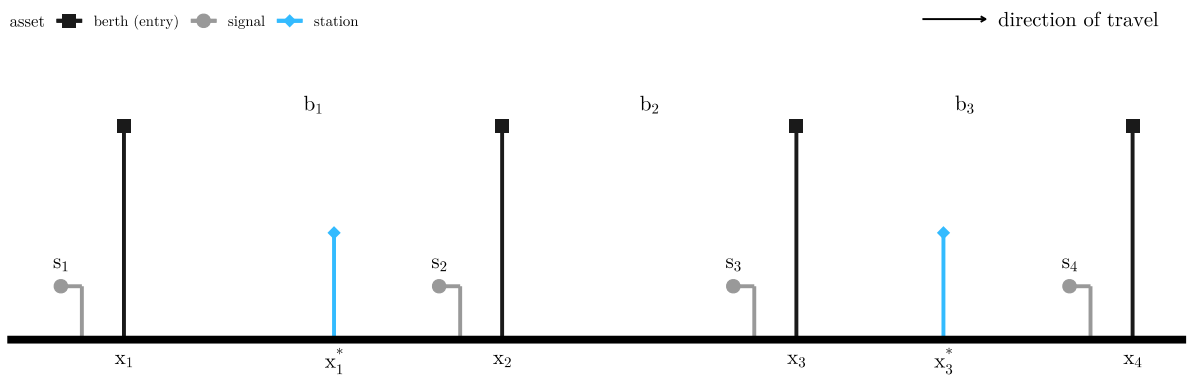


Figure 1, Shows the railway signal network and its properties. [10]

These signals are automatic; they gather information about the train at that point in time at which it was recorded. All of the information gathered by these signals is the data that is used throughout this dissertation to understand more about railway performances.

2.2 Delay Management

Understanding delay management and how it is done is an important concept for being able to make observations about the dataset, as knowing how delays are dealt with currently can give better ideas of where things are going wrong and how new ideas can be offered.

Delay management addresses the question of whether connecting trains should wait (or not) for delayed passengers [4]. This is an important piece of information to know when looking at railway data, as connecting trains waiting for passengers could be an answer to some data values, like travel time, being higher. In this scenario, although the travel times would be higher, which seems like something to avoid, it may need to be done to ensure certain passengers don't miss their connecting trains.

If the connecting train waits, delays would get transferred through the network [4]. This concept of a domino-like effect with train delays is another important concept when looking at railway data. One train being delayed can cause an entire day of following trains to have delays. Better communication between trains can help reduce this effect.

Railway timetables are another big part of understanding train data and the context of the data. Timetabling is among the most important tasks for optimisation in public transport. Besides technical restrictions and optimisation of costs, the main focus lies in finding optimal timetables from the passenger's point of view [5]. This information is important to the dataset that this dissertation is dealing with, as there are different train types, not just fast and stopping trains but also trains with normal and abnormal timetables. This gives context to these types, as abnormal travel times may need to be implemented at times due to passenger optimisation.

Simply changing train timetables to suit delays or travel times is not always a solution for lowering travel times.

2.3 Resilience in Railway Transport Systems

The resilience of railway transport systems is the ability of a railway system to provide effective services in normal conditions [6]. The smooth functioning of a railway system can be heavily affected by disruptions ranging from natural disasters to technical failures and attacks. The potential outside context for the railway dataset in this project could become important when looking at travel times.

Critical infrastructure networks for transport are essential for the functioning of society and its economy [6]. Having resilience against potential disruptions and anticipating things going wrong within a railway system is crucial to its ability to adapt.

Signalling between trains is important not only for safety but also for delay times. If train signalling is not instantaneous or as close to that as possible, there could be serious

repercussions, such as train collisions. Because of this fact, researching times to switch signals in the train data became an important aspect of this dissertation.

In the wider context of this project, the technical failures discussed in the paper on resilience in railway transport systems, relate to the offset and onset data in the railway dataset given, as well as the ideas of how railway timetables are designed to anticipate cascading train delays after an incident.

2.4 Technologies Used

Before any technical work is explained, this section will go through the technologies used and why. Explaining what these technologies are and their background is important for understanding how and why they're being applied to this project.

2.4.1 Seaborn and Matplotlib Libraries

Matplotlib is a library used for data visualisation in Python. Matplotlib is well integrated with the NumPy and Pandas libraries [16].

Seaborn extends the Matplotlib library for creating better-looking graphics using very little code with the use of its many functions [16]. This library is used throughout this project to better visualise the railway dataset that was given.

2.4.2 Heatmaps

There is a wide range of plot types within Seaborn, and there is a huge example gallery that can be used to find relevant plot types that this library offers [2].

The most relevant plots include, firstly, heatmaps, which can be used to make a correlation matrix that can help in figuring out which numeric values within the dataset correlate the most and would benefit the most from being compared.

Heatmap plots work by giving the relationship between every column in the dataset a score, with the value closest to one being a higher correlation, which is represented visually by a gradient, with the lighter colours being higher correlations.

2.4.3 Scatterplots

Scatterplots can be used to compare the relationship between two numeric values. Scatterplots have been recognised as one of the most versatile and useful techniques in data visualisation. However, scatterplots do not handle large amounts of data or a high number of dimensions of data very well [21].

Although most of the data for the given railway dataset is categorical, there are a few time values that can be compared against each other, such as travel time, time taken to travel the length of a coach, time that the signal is at red, etc. Understanding the relationships between these time variables is integral to finding out how to improve railway performance.

Scatterplots can make it easy to understand the characteristics of relationships, from positive relationships when values go from the lower left-hand corner to the upper right-hand corner to negative relationships when values go from the upper left-hand corner to the lower right-hand corner [9].

2.4.4 Boxplots and Bar Plots

Most of the given railway dataset for this project is suitable for boxplots and bar plots. These plot types within Seaborn allow creation with just a couple of lines of code, with the function for creating the two plots having many different parameters to be able to show.

Boxplots and bar plots help show a categorical variable against a numeric variable and are primarily used in this project to compare train or time attributes to travel times and TSAR times.

The distinction between boxplots and bar plots was important to understand when deciding which to use. The bar plot is typically used to compare data sets based on simple statistical measures, usually the mean but, may fail to give a full description of the underlying differences in the data [17].

Box plots, however, represent the summary of the data and its full distribution. Box plots show the minimum value, lower quartile, median, upper quartile, and maximum value for any piece of data [17]. This is much more useful when looking at travel times, as they can vary heavily, and just the average may not give a full description of what is happening in the data.

2.4.5 Subplots and Facet Grids

The previous plot types discussed on their own can only convey limited amounts of information and splitting a piece of data into multiple parts is also necessary in some cases.

This is where subplots and facet grids come in. Subplots are a part of Matplotlib, which allows the plotting of multiple graphs. Subplots can have their own titles, labels, and legends and occupy their own spot in the grid [20]. For example, different boxplots are split into train types and plotted within this subplot to be able to compare the train types.

Facet Grids are a Seaborn version of the Matplotlib subplot that allows the creation of a subplot with a lot less code, so they can be more time-efficient but are more limited in the information they show.

2.4.6 Boxen plots and Violin plots

Boxen plots and violin plots are variations of a boxplot that are offered by Seaborn as some of its functions.

The boxen plot is similar to a standard boxplot but includes more quantiles for better visualisation [22]. This plot type has lots of boxes of different sizes and shapes, representing where the larger quantities of the data occur.

The violin plot combines the box plot and density trace (or smoothed histogram) into a single display that shows distributions in the data [23]. Wherever the violin plot is wider, it shows where the larger quantities of the data occur; this can be more visually pleasing than a box plot and can show more about exactly where the data distribution occurs.

2.4.7 Jupyter Notebook

All of the Python code in this project was programmed within Jupyter Notebook. Jupyter Notebook is an open-source, browser-based IDE that supports workflows, code, data, and visualisations detailing the research process [11]. Which is perfect for exploratory data analysis.

For this project specifically, Jupyter Notebook was favourable for creating a new notebook for each part of the data analysis, which made things easier in the coding phase. Jupyter Notebook also allows the user to keep the code in different sections to make it more readable, and each section can be run individually for its output, which can prevent an overcrowded output section.

2.4.8 Pandas Library

Pandas is a Python library with many data structures and tools for working with data and contains many functions that can be applied to the analysis of data sets [3]. Within this project, pandas are used to either explore the dataset or prepare the data for later plotting in Seaborn.

2.5 Visualisation Techniques

One of the main goals of this project is to make visualisations that are easy to understand and make sense to the average person.

Colours are carefully selected for certain visualisation types. When colour is used to represent data, it is important to choose appropriate colours that are easy on the eye and display only what is necessary. In some visualisations, colour is used with a key to represent a category of the data for comparison [18].

Using colours can help represent data in a more abstract way to make sure what the data is showing is easy to understand. For railway data analysis, these techniques can help expose the driving factors for higher travel times.

Another visualisation technique is using spatial variables such as position, size, and shape to represent elements of the data [18]. Boxen plots and violin plots are examples of how shapes can show what is happening within a piece of data.

3 Programming and Implementation

This section first describes the dataset and its origins for context. This section then explains programming and implementation of all the Python code for every visualisation and data value made, how they were achieved, and what technologies were employed to do so.

The order of the subsections is in the order in which the plots were programmed. The advantages and disadvantages of every function, method, or piece of software used will be discussed throughout this section, as well as how some plots were changed throughout development and why.

3.1 Understanding the Data

To understand the data visualisations within this project, it is important to know what the dataset is that was given to make these plots and what the different attributes refer to in real railway settings.

3.1.1 Explanation of Data Origins

Multiple datasets were provided throughout the development of this project. Initially, there was a set of four small datasets with around one thousand entries. These smaller datasets were used early on to create initial data plots. However, this section will go through the larger dataset's attributes and their meaning, as this dataset is more useful for getting a good idea of relationships due to how vast it is.

It is important to mention that these datasets are real-world railway data provided by NetworkRail, which owns, repairs, and develops the railway infrastructure in England, Scotland, and Wales [7].

The dataset can be split into different types of attributes to make it easier to understand, the first being time events that come from the real data collected by NetworkRail. The second type is duration values that are calculated using the original time event data from previous work done by Pedro Pinto da Silva [10]. The last type is categorical data originally collected by NetworkRail, such a train type, berth, etc.

3.1.2 Explanation of TSAR

The attribute "TSAR" is one of the most important parts of the dataset as it is unique to this project. It was a value created by the project supervisor that was given to this project as a starting point. The project supervisor provided slides and diagrams for the initial idea of TSAR. This idea of TSAR was to be expanded upon, which is what has been attempted in this project.

TSAR (Time Signal at Red) is calculated by the clearance time, which is the time taken for the train to clear the berth, taking away the onset time, which is the time taken to switch from a proceed aspect (green, yellow, or double-yellow) to red, added to the offset time, which is the time taken to switch from red back to a proceed aspect (Figure 2).

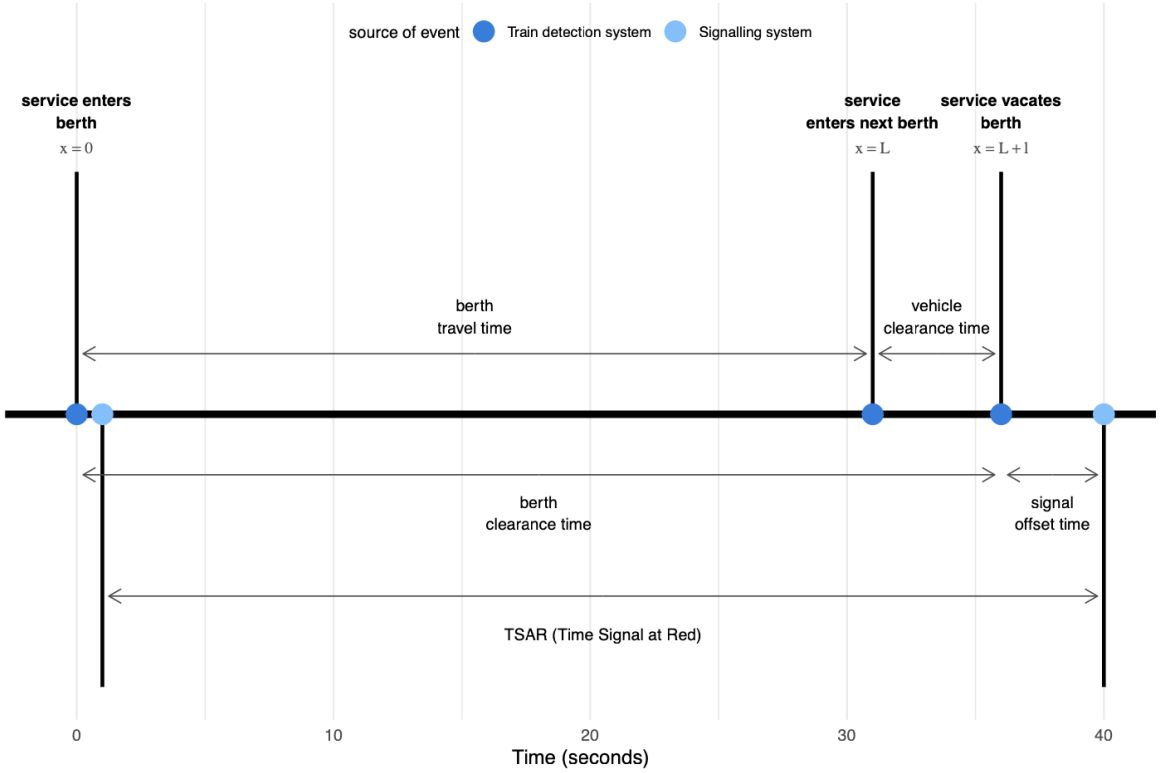
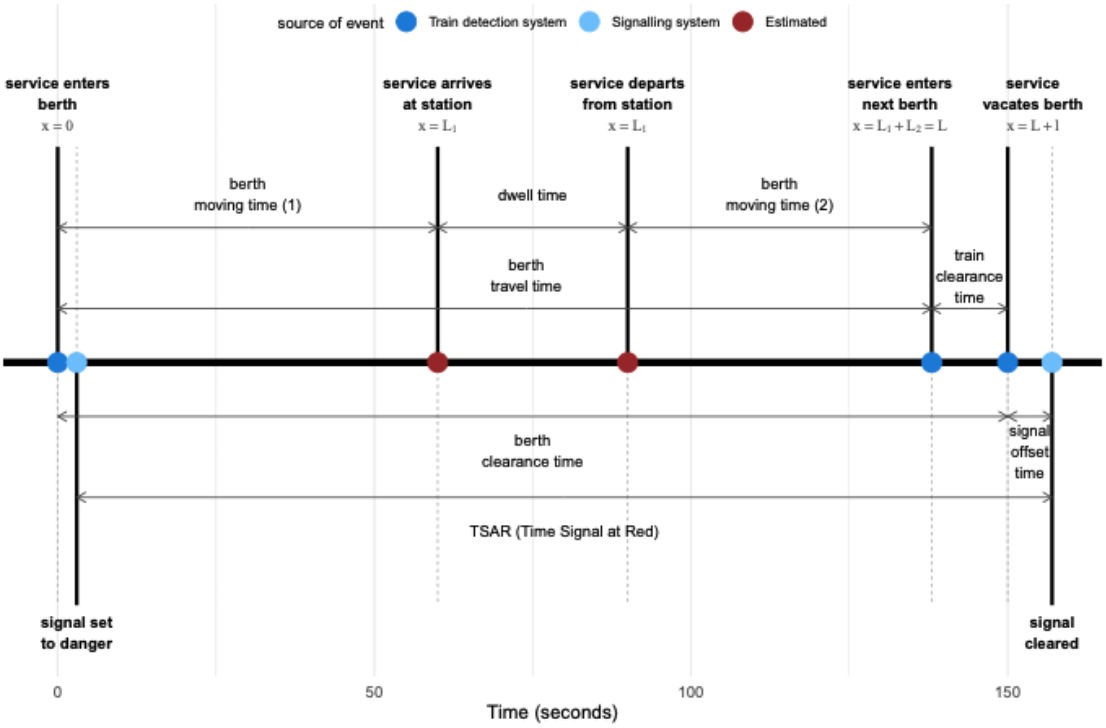


Figure 2, Timeline of events for time values (40-second time scale). [10]

Figure 2 shows a timeline of events for a non-stopping train taken from a berth on the UK Southwest mainline containing Walton-on-Thames Station. Figure 3 shows the same as Figure 2, but for stopping trains [10].



(b) Example timeline for a stopping train (150 seconds time scale).

Figure 3, Timeline of events for a stopping train (150 seconds time scale). [10]

Figure 3 shows how stopping trains include a "dwell" time, which is another attribute that has to be calculated for travel times and TSAR and increases these values significantly [10].

Time travelled has a very similar value to TSAR and can be easily confused. Travel time is simply the time taken from when the train enters a berth until it enters the next berth, although TSAR considers the vehicle clearance time and the signal offset time and can be more useful for identifying sensitive points in the network or sensitive periods.

In some cases, TSAR needs to be normalised, as longer berths naturally mean more time spent with a red signal, so dividing the TSAR value by the berth length per 100 metres can give more accurate readings of what or where is giving higher TSAR values. This will be discussed in more detail in this dissertation's programming and implementation section.

3.1.3 Explanation of Train Types

Another important part of the data given for this project is the train type. Understanding each train type is essential for getting a full idea of what is going on in some of the data visualisations that were created, as they are split up for each train type. In the dataset, there are six different train types; however, two of them, trains that only stopped at Esher station and trains that only stopped at Esher and Walton-on-Thames stations, came with a very small sample size and weren't big enough to draw accurate conclusions.

The four train types that were looked at included: fast trains that have a normal travel time profile; fast trains that have an abnormal travel time profile; stopping trains that call at all stations; and stopping trains that call at Walton-on-Thames station. Travel time profiles have

been classified by the supervisor of this project and refer to the travel time behaviour shown in Figure 4.

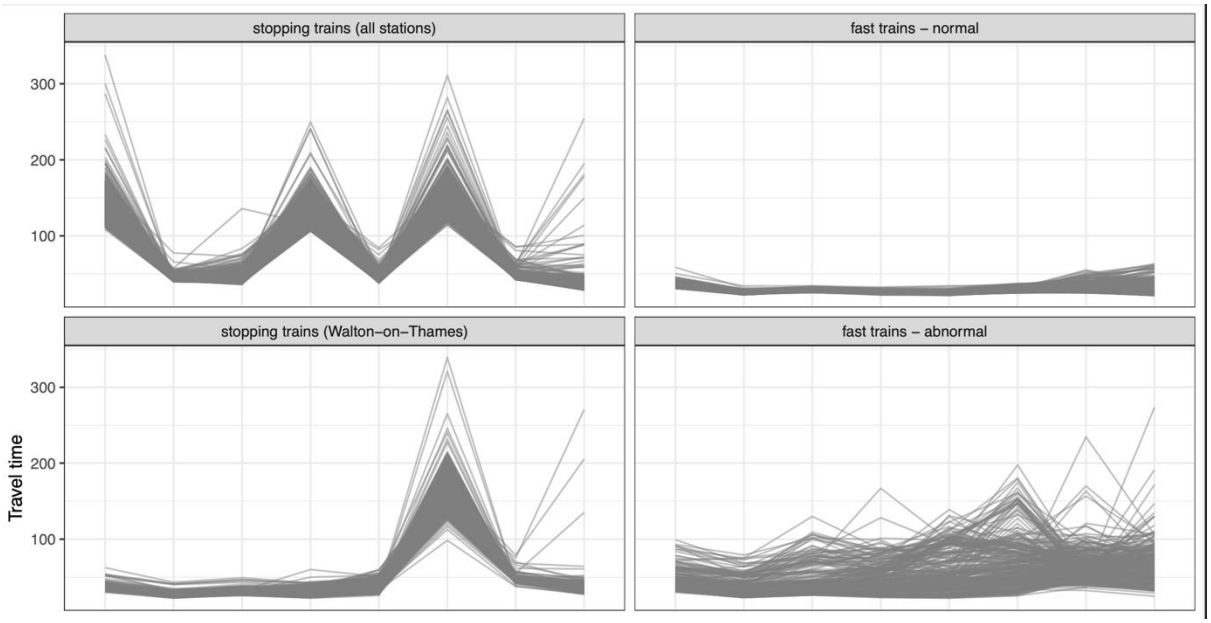


Figure 4, Graphs showing reasons for classing of the four train types in the dataset [10].

The classification of these train types was decided upon by using a clustering algorithm (k-means algorithms) before any work was done on this project. The clustering shown in Figure 4 is due to normal trains travelling through mostly G aspect signals, which means they are less delayed, whereas abnormal train types see a lot more Y and YY signals.

3.1.4 Normalising Data

Normalisation is a pre-processing technique where data is scaled or transformed to make an equal contribution. This approach makes sure that all features have an equal numeric contribution [19].

For railway data analysis, a feature that may need to be normalised is travel time. Travel times may need to be normalised, specifically when looking at location-based results. This is due to berths having a unique length, which may be the reason for their higher travel times. Making travel times per 100 meters would be a way of normalising this attribute.

3.2 Data Preparation

Before any programming of data plots, it was important to use Pandas functions to understand, explore, and import the data.

3.2.1 GitHub and Library Imports

To begin the project, a GitHub repository was created by the project supervisor, which contained all of the data previously explained in Section 2.4. This was pulled down from Git to a local file. Once a file structure had been created for the project with a “data” folder and a

"code" folder, the code was opened within Jupyter Notebook, and the data was pulled through using the "pd.read_csv" function from Pandas. This function allows CSV files to be set as variables within a code base.

The desktop version of GitHub was used throughout this project, primarily for pulling any new data information given by the project supervisor.

Another task that was completed before any data programming could begin was the import of all of the standard libraries that are useful for exploratory data analysis. These include Pandas, Matplotlib.pyplot, Seaborn, Datetime, Numpy, and Matplotlib Dates.

3.2.2 Investigation and Preparation of Data Using Pandas

The first Pandas function to use on the given railway dataset was the ".head()" function, which allows the viewing of the first five data entries. This was useful to have open when writing code to plot graphs, as it has all of the data columns and a view of some of the values included within the dataset.

When given the dataset, not all of the important data columns were combined into the same dataset. The "merge" function within Pandas allowed for the combination of datasets that have similar columns (Figure 5). This was due to the berth length information, which had information regarding all berth lengths and travel times, and the berth class information, which had train-type information, being given in different chunks of smaller datasets.

	berth	station	L1	L2	signal	train_id	aspect	t_enters	t_red_on	t_enters_next
0	AAV	Esher	1050	124	S135	1	G	2022-05-29T05:44:33Z	2022-05-29T05:44:34Z	2022-05-29T05:45:19Z
1	AAV	Esher	1050	124	S135	2	G	2022-05-29T06:39:48Z	2022-05-29T06:39:49Z	2022-05-29T06:41:59Z
2	AAV	Esher	1050	124	S135	3	G	2022-05-29T07:06:56Z	2022-05-29T07:06:57Z	2022-05-29T07:09:17Z

Figure 5 Result of merging the different datasets together.

Getting to know the data continues; now that the size and column names of the dataset have been seen and noted, the next step is to examine the data more systematically [12]. Using the ".info()" function allowed the ability to view all the columns in the dataset with their data types; this was essential during the programming stage to know what type of plot is best for a given attribute as well as knowing which columns can be compared (Figure 6).

```
berth_events.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 38376 entries, 0 to 38375
Data columns (total 25 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   signal             38376 non-null  object
 1   berth              38376 non-null  object
 2   train_id           38376 non-null  int64
 3   aspect             38369 non-null  object
 4   t_enters           38376 non-null  object
 5   t_red_on           38376 non-null  object
 6   t_enters_next      38376 non-null  object
 7   t_vacates          38376 non-null  object
 8   t_red_off          38376 non-null  object
 9   TSAR               38376 non-null  float64
 10  T_onset            38376 non-null  float64
 11  T_clear            38376 non-null  float64
 12  T_offset           38376 non-null  float64
 13  T_travel           38376 non-null  float64
 14  T_coach            38376 non-null  float64
 15  class              38376 non-null  object
 16  median_travel.sec  38376 non-null  float64
 17  v.mph              38376 non-null  int64
 18  L.miles            38376 non-null  float64
```

Figure 6, Shows the data type information for the train dataset.

Furthermore, an important aspect of exploratory data analysis is checking for missing values in the data. The Pandas function "isna()" allows to check for NaN values along with the "sum()" function to see which attributes have missing values. After running these functions, it showed that only the "aspect" column had seven missing values, which meant this wasn't a big concern. However, dealing with these missing values was still important, and just removing those columns and treating them as redundant seemed like the best course of action (Figure 7).

```
In [192]: berth_events.isna().sum()
```

```
Out[192]: signal             0
          berth              0
          train_id           0
          aspect             7
          t_enters           0
          t_red_on           0
          t_enters_next      0
          t_vacates          0
          t_red_off          0
          TSAR               0
          T_onset            0
          T_clear            0
          T_offset           0
          T_travel           0
          T_coach            0
          class              0
          median_travel.sec  0
          v.mph              0
          L.miles            0
```

Figure 7, Using Pandas functions to check for missing data values.

The Pandas function "dropna()" was then used to drop all missing values to get rid of any unusable data.

One last vital part of preparing the data is checking for outliers. Checking for outliers using Pandas was done by using the "describe()" function, which gives information about the values of the dataset such as the min, max, and median values.

When outlier detection was used on the railway dataset, it showed a disparity in the high and median values for the TSAR and travel time values. However, the context of this data means that there could have been an incident on the railway causing this long travel time and therefore may need to be looked at. Although an interesting outlier, the offset time data had very high values with no apparent context, which were removed using comparison Python code.

Python code was used to set all of the offset time data values to itself but only the values that are less than or equal to 150 to remove data that was deemed to be false.

3.2.3 Creation of New Data Attributes

Despite the dataset that was given to the project being quite vast and having many different attributes, to be able to make plots for some of the areas that were planned to be looked at, new columns had to be created based on the given data information to be able to make plots...

3.2.3.1 Time-of-day Categorical Attribute

For looking at the time-of-day for travel times and TSAR values to see how time-of-day affects these attributes, creating new categories for what part of the day a signal or a travel time occurred was the first step.

The time for every entry in the original dataset was a date-time type, which was harder to create intervals for. Therefore, these date times were converted to integers for the number of seconds in the day for easier comparisons between the values. The original date-time values before conversion are shown in Figure 8 and are the columns beginning with a lowercase "t.".

	signal	berth	train_id	aspect	t_enters	t_red_on	t_enters_next	t_vacates	t_red_off	TSAR	T_onset	T_clear	T_offset	T_travel	T_coach
0	S135	AAV	1	G	2022-05-29T05:44:33Z	2022-05-29T05:44:34Z	2022-05-29T05:45:19Z	2022-05-29T05:45:23Z	2022-05-29T05:45:29Z	54.929	0.829	49.490	6.268	45.327	4.163
1	S137	AAW	1	G	2022-05-29T05:45:19Z	2022-05-29T05:45:19Z	2022-05-29T05:45:53Z	2022-05-29T05:45:57Z	2022-05-29T05:45:59Z	39.972	0.821	38.417	2.376	34.247	4.170
2	S139	AAX	1	G	2022-05-29T05:45:53Z	2022-05-29T05:45:54Z	2022-05-29T05:46:32Z	2022-05-29T05:46:36Z	2022-05-29T05:46:39Z	45.000	0.820	43.434	2.386	39.268	4.166
3	S141	AAY	1	G	2022-05-29T05:46:32Z	2022-05-29T05:46:33Z	2022-05-29T05:47:07Z	2022-05-29T05:47:11Z	2022-05-29T05:47:14Z	40.843	0.821	39.174	2.490	35.115	4.059
4	S143	AAZ	1	G	2022-05-29T05:47:07Z	2022-05-29T05:47:08Z	2022-05-29T05:47:44Z	2022-05-29T05:47:47Z	2022-05-29T05:47:50Z	41.642	0.815	39.971	2.486	36.758	3.213

Figure 8, Results beginning with a lower-case letter "t" show original date-times from railway dataset.

To convert the "time enters" attribute, which is when the train enters the berth, to seconds, it started with converting the full date to just the 24-hour time. This was done by a Pandas function that converts the format of date-times, and this was done for every value and saved to a new column.

The piece of code developed gave the value a more stable datetime format using the Pandas "to_datetime" function. The second piece of code got just the time part of the whole value by removing the date part, leaving just the time of the day that the event occurred.

Now that the value had been converted to a 24-hour time, it then needed to be converted into seconds for comparison. This required the "lambda" function in Python that was used to split the hours, minutes, and seconds of the time value into three separate numbers. Then multiply the hours by 3600 (3600 seconds in one hour), multiply the minutes by 60 (60 seconds in a minute), leave the seconds as they are, and add all these values together to get the total number of seconds. The total number of seconds was then set to equal a new attribute within the dataset.

Every value was now an integer representing how many seconds into the day it had occurred, and then the "cut" function from the Pandas library was used to create a categorical value for each data entry based on its "second of the day" value. There was a different value for this categorical attribute before 9 a.m. and then for every three hours of the day, 9 a.m.-12 p.m., etc.

The interval integers on which the categories are based were calculated with the knowledge that every three hours is equal to 10,800 seconds. This was set to the new categorical attribute using the "cut" function from Pandas and inputting the different intervals in the "bins" parameter with their corresponding label in the "labels" parameter. This code is shown in Appendix A.

3.2.3.2 Seasonal Categorical Attribute

Another categorical attribute that needed to be created for looking at factors that affect railway variability was splitting the data into seasons. In the railway dataset given, only data for spring and autumn were included. Converting the dates to integers followed a similar approach to the time-of-day attribute.

Firstly, the date and time values were all converted to a total number of seconds by using a variable "type" function from Python, using floor division for a whole number, and dividing by 10 to the power of 9 to get a much more manageable number. Now all the data had an ascending integer for when it occurred.

Then the same "cut" function from the time-of-day attribute was used, and the threshold was found by viewing the data and seeing where the number split between the two different months occurred. The two labels "Autumn" and "Spring" were set in the labels parameter of the "cut" function. This code is shown in Appendix A.

Now that all of the relevant attributes had been created, the visualisation and data analysis were able to start.

3.3 Implementation of a Correlation Matrix

Before any interesting comparisons could be made, it was first important to use a correlation matrix to check for interesting relationships between the numeric values within the dataset.

As correlation matrices only deal with numeric values, the first step was to remove all categorical data from the dataset. This can be done using the drop function from Pandas and manually entering all columns from the dataset that weren't numeric; this was helped by referencing the data types using the previously mentioned "info" function.

Once every attribute in the data was numeric, the correlation function from Pandas could be used in tandem with Seaborn to create a visual representation of correlations between values. The Pandas function was first used on the new numeric dataset and set equal to a new variable name. This new variable was put into the Seaborn "heatmap" function with the annotations for each value set to true within its parameters to show all correlation values.

For more appealing results, Matplotlib allows for the rotation of the X-axis labels to make them easier to read. Techniques like these were used throughout the implementation to make the data plots easy on the eye and easy to understand.

The results of this code and its evaluation can be found in Section 4.1 of this dissertation.

3.4 Time of Day and Seasonal Visualisation Code

Now that all of the appropriate attributes were created and the correlation between the data was assessed, the code base was ready to start plotting the data. The type of plot (boxplot, bar plot, line plot, etc.) was decided while programming, as Seaborn allows for the easy change to a different plot type if necessary.

It is important to understand that the "travel time" that is plotted in the following graphs is plotted instead of TSAR due to TSAR including variables such as clearance time and offset time, which are to be excluded. However, keep in mind that TSAR and travel time are linear, and any findings about travel times also apply to TSAR.

3.4.1 Time-of-Day in Intervals Box Plot Code

Using the time category attribute, that was created earlier, boxplots were created to show the travel times for different times of the day. However, this gave very varied results, with a lot of outliers and wide ranges. This was discovered to be because the train type needed to be separated for a fair comparison of the time of day against travel times. This was done using subplots in Matplotlib.

The data was split up into four parts, one for each of the four train types. This was done before the plotting of the data using the "locate" function from Pandas and setting a new variable for all four types and setting them equal to the original dataset, but only where the "class" is equal to whichever train type is needed.

A boxplot plot type was used as opposed to a bar plot, for example, because of its ability to show quartiles and give a full range of values. Including those ranges and quartiles gave a much better coverage of information and comparison of the different train types.

The data could now be sub-plotted using Matplotlib. The season category was also added as a "hue" parameter, which is an option in Seaborn boxplots for giving a colour for the boxes for each category. This allowed for the analysis of more than one attribute per plot. The previously created data frames for each train type were used for their respective subplots in the "data" parameter of the Seaborn box plot function.

Each plot was set to one of the four squares of the subplot, and the data for each plot uses a different data frame. Seaborn boxplots allowed for the customisation of turning off outlier dots by setting "showfliers" to false; this was done to only focus on the more concrete data.

The plots were also given customised Y-axis numbers with the "yticks" parameter; this is due to the different plots requiring different ranges for their axes. This was customised by manually writing an array in Python for each of the ranges for all subplots.
Seaborn also allows for the customisation of X and Y labels as well as titles. Having appropriately named labels was important for the understanding of the visualisations. Each plot has a title depicting which train type it represents, and there is also a title given to the entire plot itself to describe what is being shown. A "legend" was also added, which shows a key for what the colour of each box means and is able to change its position using the location identifier given by Seaborn. This code is shown in Appendix B.

The results of this code and its evaluation can be found in Section 4.2.1 of this dissertation.

3.4.2 Time-of-Day by Hours Box Plot Code

For an alternative to the plot from Section 3.4.1 that used the time intervals created earlier on, a data attribute named "hour" was created with the specific hour of the day that each event occurred. This again used Pandas and the "to_datetime". Pandas has a specific "hour" function that can be used with "to_datetime" to give its specific hour of the day from 0 to 23. This code is shown in Appendix B.

Boxplots were used again for gathering the range and differences between the different periods; bar plots were attempted as a way of plotting this graph but gave little variation between the times, making it harder to gather conclusions.

The new "hour" attribute was then plotted in another Matplotlib subplot for each train type, using the Seaborn box plot function once again for each plot. The "palette" parameter from the box plot function was used and set to "blues" for a better aesthetic.

The results of this code and its evaluation can be found in Section 4.2.2 of this dissertation.

3.4.3 Time-of-Day Line Plot

Another plot type that was used to display how the time of day affects travel times, was line plots. Now that the hour of the day attribute had been created, this allowed for the creation of a line plot showing what the average travel times are for each hour of the day. Line plots were better for showing progression over time.

More implementation of some of the Seaborn parameters was used for better visualisation. The error bar parameter was set to "None" for this plot as this shows the range of uncertainty around the estimate, which wasn't needed for this plot and created too much to look at, especially when dealing with four different lines [13].

To differentiate between the different train types in the hour of the day line plot, the "hue" and the "style" were set to the class of the train to give a clearer distinction between the different types. The X and Y axis labels were customised again using the "xticks" and

"yticks" functions from Seaborn; changing these labels helped show every hour of the day along the bottom for a more precise evaluation of exactly when different events occur.

The results of this code and its evaluation can be found in Section 4.2.3 of this dissertation.

3.5 Location Visualisation Code

The next aspect of the railway data to look at was the locations. This section will go through the code developed to create visualisations that show how different locations affect travel times.

There were two different data attributes to consider when developing visualisations of locations against travel times. These included the "berths," which each have their own ID and can be individually looked at to check which give the highest travel times. The other attribute for location is the three different station names included in the data.

3.5.1 Each Berth and Their Travel Times Code

The first plot created for evaluating locations was comparing all of the berths by splitting them using subplots (similar to Section 3.4.1). A four-by-two subplot was created to include each of the eight berths, each plot having the time of the day against travel time to also include time-of-day comparisons.

The location function from Pandas was again used for splitting the data by berth ID, which was done before the creation of the subplot.

This subplot was created using Matplotlib and the Seaborn box plot functions and split into train types. Each plot gave the time of day along the X-axis and the travel time on the Y-axis.

Boxplots were used as the form of plot here to again give better comparisons. The bar plots were checked for potential use again but gave less comparable results to the boxplot.

Due to many plots being put together like this, some labels were bunched together. This was fixed using the "rotation" parameter, which was set to 45 to tilt the labels 45 degrees.

Once this data had been plotted, it became apparent that this subplot was used for comparing all the berths, but it included berths that have stations and berths that do not. The labels for the subplots for berths with stations included the fact they have a station; however, it seemed appropriate during development for a plot to be created that separated these two into their individual sections.

The results and evaluation of the code shown in this section can be found in Section 4.3.1 of this dissertation.

3.5.2 Berths with and Without Stations Code

To separate berths that contain a station from berths that do not, another subplot was created, but with a two-by-one plot.

For this plot, a similar style of subplot to the ones made previously was created, however, instead of a standard boxplot, this plot uses a "boxen plot" from Seaborn, which is a variation of a boxplot that shows the distribution in quantiles [14].

The "boxen plot" felt more appropriate to show more accurately what the highest travel times are for each berth and looked more aesthetically pleasing. These styles of plots tend to work better when there are fewer boxes to view and when each box needs to be assessed more thoroughly.

The results and evaluation of the code shown in this section can be found in Section 4.3.2 of this dissertation.

3.5.3 Stations and Their Travel Times Code

Another locational visualisation that was made using the railway dataset was for looking at travel times at the three different stations. This was another plot that was sub-plotted into different train types for a fairer assessment. This plot used another variation of a box plot called a "violin plot". Just like a "boxen plot", a violin plot shows density and is visualised as smoothed histograms along the data points [15].

This plot followed a similar development to the previous subplots where the train types were split into their variables. Seaborn makes it easy to change from a box plot to a violin plot, although they have slightly different parameters. The "inner" parameter of the violin plot function was set to "quart" to draw the quartiles of the distribution. The "split" parameter was set to true so that the violins would split into two for two different categories; these categories were set to the season in which the data occurred. This code is shown in Appendix B.

The results and evaluation of the code shown in this section can be found in Section 4.3.3 of this dissertation.

3.6 Signal Aspect Visualisation Code

The next aspect of the railway dataset to explore and visualise was the signal aspect. The "signal" attribute in the dataset includes three values: Y, YY, G. These three values refer to the speed the train is allowed to travel, with G being full speed, YY being slightly less than full speed, and Y being the slowest. Being able to assess how long trains are travelling in each aspect was important to check for understanding performances.

3.6.1 Aspects with Their Travel Times Code

Another subplot of four different Seaborn graphs of each train type was created with the "boxen plot" function again to better show distributions than a regular box plot.

For a plot visualisation with three categories that essentially represent a colour similar to a traffic light system, using Seaborn parameters to ensure the correct colours were given and that they were displayed in order of speed was essential. The parameters for "palette" and "order" were changed to give the appropriate colours for each category and the order in which they should appear.

The results of this code and its evaluation can be found in Section 4.4.1 of this dissertation.

3.6.2 Aspect Travel Times at Different Berths Code

The previous visualisation code described is good for showing how fast different train types can travel, however, being able to look at the locations (berths) and how much different the speed is that the trains are travelling at was also an interesting part to visualise.

To visualise which berths, have the highest travel times and what part of those travel times are spent at the three signal aspects, another two-by-two subplot was created to split into the four train types yet again. The "palette" parameter was again changed to reflect the three aspects. This plot used a "bar plot" to give average travel times and removed the error bars through Seaborn parameters to give a better visual experience.

The "hue" parameter of the bar plot function was used to have a different bar for each aspect signal; this automatically split the data into the three signal aspects for each berth on the X axis. This code is shown in Appendix B.

The results of this code and its evaluation can be found in Section 4.4.2 of this dissertation.

3.7 Train Type Visualisation Code

For looking at and visualising train types, the original plan was to give train types and sizes individual plots. However, after more evaluation of the dataset, it became apparent that the train type is essential for comparing and understanding values, to the point where every other plot made to view other attributes against each other became subplots divided into the four different train types.

3.7.1 Time Taken to Travel Train Length Code

However, some other visualisations were able to be made to understand how train types and sizes affect travel times and TSAR. The first of which involved comparing the time it takes for a train to travel its length against travel times for each train type. This plot used the scatterplot function from Seaborn and the Matplotlib features for sub-plotting.

A scatterplot was used for this graph, as this is the comparison of two numeric values, and the correlation between these two attributes can be best visualised through a scatterplot.

Seaborn parameters were used to ensure each plot of the scatterplot stood out against its background. To do so, the edge colour of the dots was set to white, and the line width was set to 0.3 to make sure if the dots were bunched up together, they were still distinguishable from one another. This code is shown in Appendix B.

The results of this code and its evaluation can be found in Section 4.5.1 of this dissertation.

3.7.2 Plotting What Trains are Powered by Code

Another way to compare train types and how they perform is by looking at how trains that are powered differently perform. However, the only data that was able to be accessed for how the trains are powered is from the smaller dataset that was given for this project, and it should be said that these visualisations are potentially less reliable for drawing conclusions.

The Seaborn bar plot function was used to create a basic plot comparing electric and diesel trains and their TSAR values. The bar plot parameter "alpha" was changed to 0.8 to make the bars translucent for a better reading of the data. The parameter "errorbar" was also set to None to remove excess information about errors in the data.

The results of this code and its evaluation can be found in Section 4.5.2 of this dissertation.

3.8 Signal Visualisation Code

The final attribute of the analysed and visualised data was the signal times. The signal times looked at include the onset and offset times; this was done to check for inconsistencies in signals.

Both onset and offset times were plotted against each of the eight different signal IDs and graphed using box plots.

Box plots were used for this visualisation to give the distribution of the values of offset and onset times. Bar plots would not show the same range of information for this comparison.

This plot used a standard box plot implementation from Seaborn for the offset times; the same approach was taken for the onset times. Appropriate labels were created using the X label and Y label parameters from Seaborn, as well as appropriate titles with the "title" parameter.

The "palette" parameter of the Boxplot function was set to "blues" to give a more appealing visualisation. This code is shown in Appendix B.

The results of this code and its evaluation can be found in Sections 4.6.1 and 4.6.2 of this dissertation.

3.9 Creating Normalisation Attribute

Throughout the development of the visualisation talked about previously, it became apparent that some graphs would benefit from using a normalised version of TSAR and travel time.

To normalise these two attributes (TSAR and travel time), the first step was to convert the length of each berth from miles to meters. This was done by creating a new "L.meters" attribute and making it equal to the mile's variable multiplied by 1609, which is the conversion from miles to meters.

Once the length of the berth was in meters, a "length_per_100" attribute was created to give how many hundreds of meters each berth was. This was done by simply dividing the length in meters by 100.

To make all TSAR and travel time values equal, so they are all per 100 meters, both TSAR and travel time were divided by the "length_per_100" attribute to give their normalised values, which were made into new attributes in the dataset. This code is shown in Appendix A.

4 Results and Evaluation

This section discusses the results and findings of the code and technical explanation given in Section 3 in the form of figures and descriptions of all of the graphs made during this project. All the understanding gained from the visualisations created will also be explained in detail.

4.1 Correlation Matrix Results

The correlation matrix gave very little in the way of ideas of what may be some of the causes of higher TSAR and travel times due to the dataset having mostly categorical attributes. Most of the numeric values that were plotted and gave good correlation scores were already known to be related in some way (Figure 9).

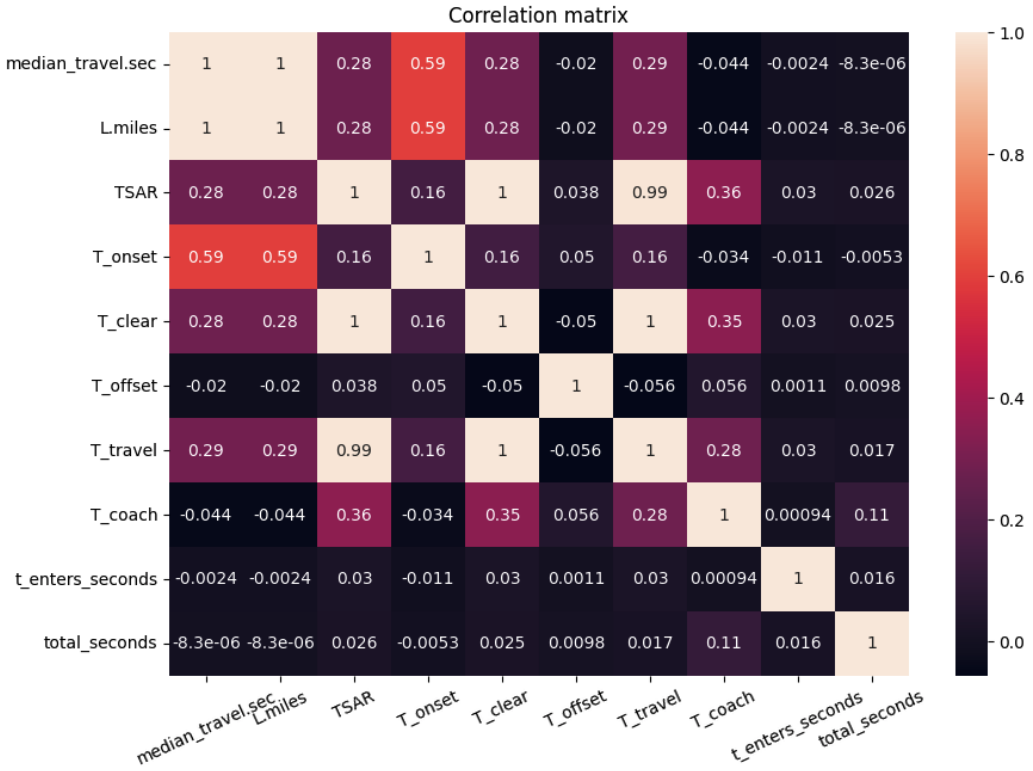


Figure 9, Correlation matrix for numeric values in railway dataset.

The correlation matrix in Figure 9 shows mostly darker colours, which show very little correlation for most relationship comparisons. Some of the close-to-perfect correlations were expected relationships, such as travel time and TSAR, the time taken to clear berth and TSAR, etc. These attributes were all based on one another and were already deemed to be correlated.

Some more interesting findings from this correlation matrix were that the relationship between the time taken for a coach to travel its length and TSAR got a 0.36 for correlation. This showed that potentially one of the biggest reasons for a higher TSAR could be bigger trains carrying more passengers and having to stop at the stations for longer.

Due to the results of this correlation matrix, more time was given to visualising how coach sizes affect travel times. As well as why there is a good correlation between onset times and berth lengths, which is shown in Figure 9 as 0.56.

The correlation matrix created gave some small indicators of attributes that should be looked into more and compared; however, not a lot of conclusions could be drawn from this plot.

4.2 Time-of-Day and Seasonal Visualisation Results

This section covers the results and evaluation of the Python visualisations representing time-of-day and seasonal variables.

4.2.1 Time-of-Day in Intervals Box Plot

For the first plot for looking at time-of-day and seasons, a set of sub-plotted box plot graphs was created (Figure 10).

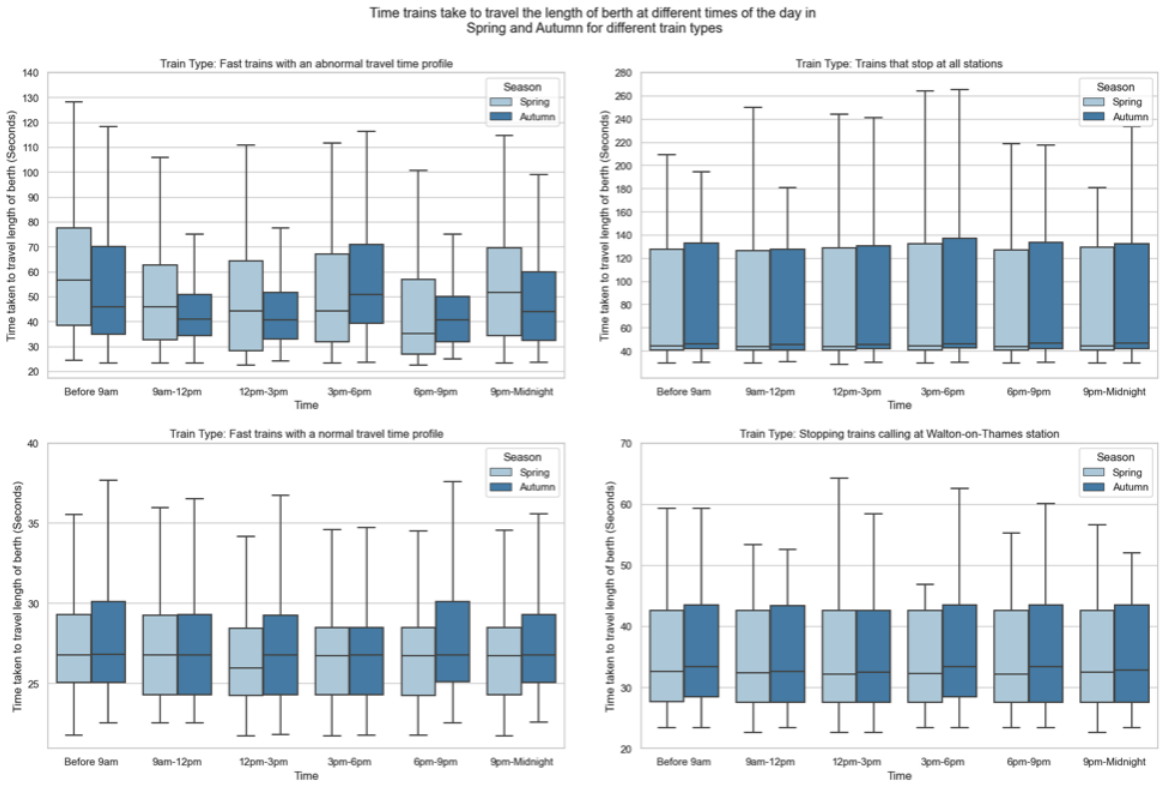


Figure 10, Results of a graph showing how the time of day and different seasons affect travel times.

The plot shown in Figure 10 shows fairly consistent travel times throughout the day. As for most of the four train types, there is very little change in where the boxes are from left to right.

Although, for fast trains with an abnormal travel time profile, there seems to be slightly more variation throughout the different periods, with the peak time of 3 p.m.-6 p.m. and the trains

before 9 a.m. Fast trains having higher travel times before 9 a.m. could be due to having more trains throughout the night, whereas stopping trains are generally running through the day for passengers.

Another takeaway from Figure 10 is that trains that stop at all stations have a wide range of travel times and have a huge upper quartile box, which suggests more about train types than the time of day as the boxes are still consistent throughout the day.

As for what is shown in terms of seasons and how it affects travel times, the autumn data and the spring data are very similar for all train types except for fast trains with an abnormal travel time profile where spring has higher results at some times, but autumn gives higher results at other times. Autumn is typically of higher value, but only by a small margin.

4.2.2 Time-of-Day by Hours Box Plot

This plot was made for a slightly more specific look at the time at which events were occurring using hours of the day as opposed to intervals (Figure 11).

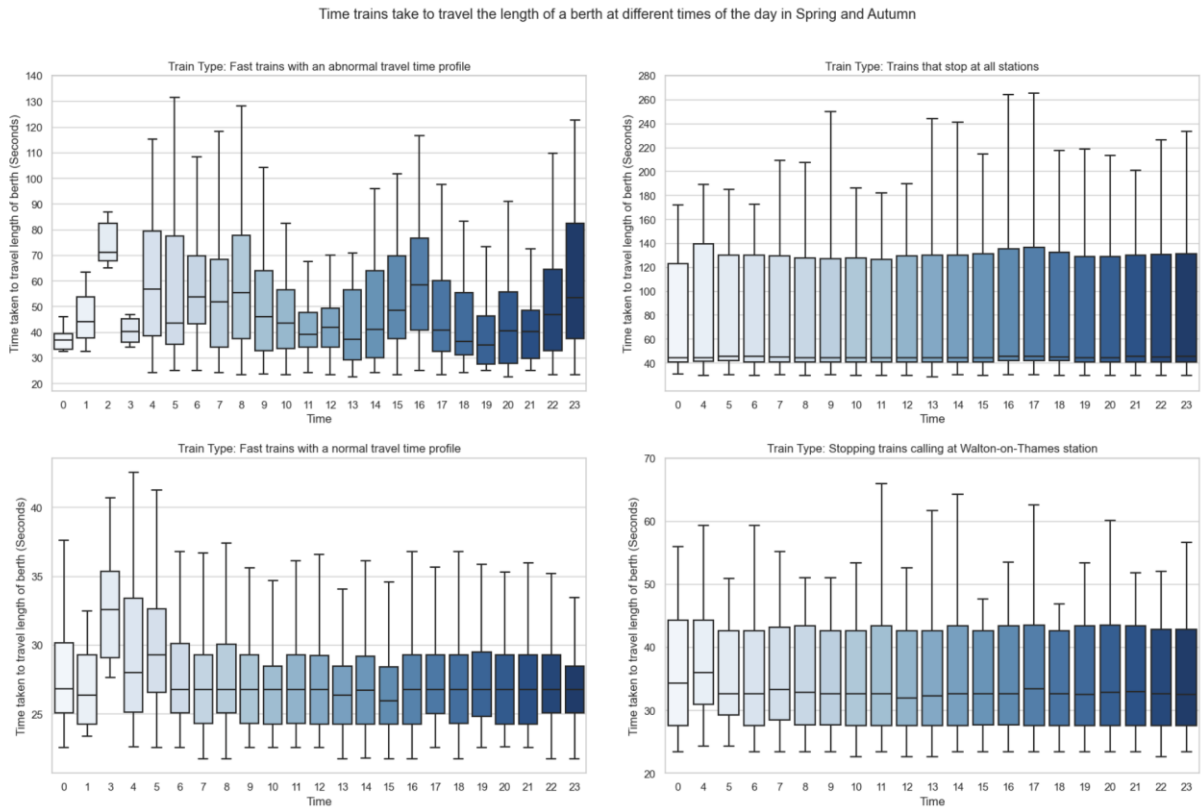


Figure 11, Results of a graph showing how the hours of the day affect travel times.

Figure 11 shows how both stopping train types are extremely consistent throughout the day in their travel times, however, both fast train types have very varied results for the early hours of the day. This could be due to the small sample size for trains running between the hours of 1 and 5 of the 24-hour day.

Fast trains with an abnormal travel time profile show an increase in travel times between 2 p.m. and 5.pm., which was expected to be the case for all train types as this is the peak time for travelling by train from work.

This visualisation (Figure 11) was able to give much more context and information about how time of day affects travel times than the previous interval-based system.

4.2.3 Time-of-Day Line Plot

The last plot that explored how the time of day affects travel times was a Seaborn line plot, which shows how the average travel times from the railway dataset change through a 24-hour day. This plot considers all the data from both seasons and is split into four lines for each train type (Figure 12).

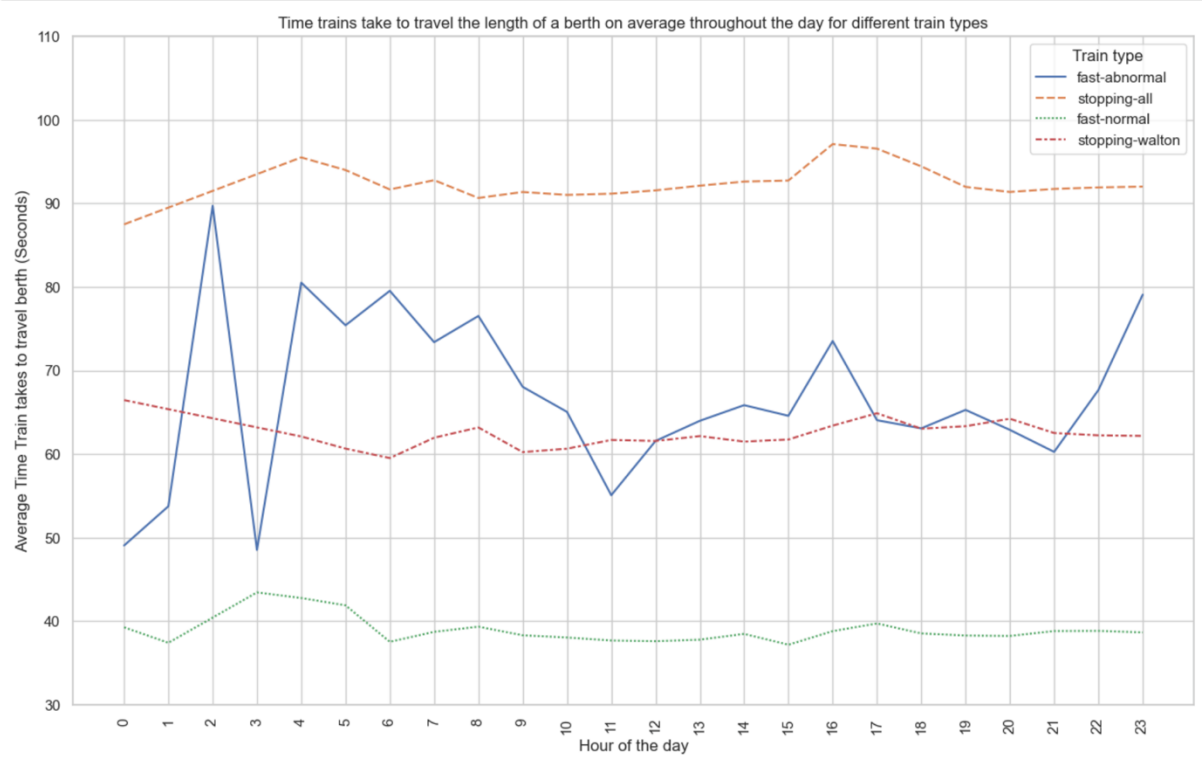


Figure 12, Line graph for all four train types and how their average travel times change throughout the day.

The results shown in Figure 12 show each train type has a unique relationship with travel times, which will be discussed later. However, the lines for both stopping trains and fast trains with a normal travel time profile show that there are very few peaks or troughs, and the average travel times stay relatively similar throughout a 24-hour day.

Based on the visualisations shown, the time of day has very little effect on travel times aside from a slight increase at peak time around 17:00.

4.3 Location Visualisation Results

This section covers the results and evaluation of the three different types of visualisations that looked at how location affects travel times and TSAR.

4.3.1 Each Berth and Their Travel Time Results

The first plot created for looking at location and travel times gave a good look at every individual berth and how their travel times were performing (Figure 13).

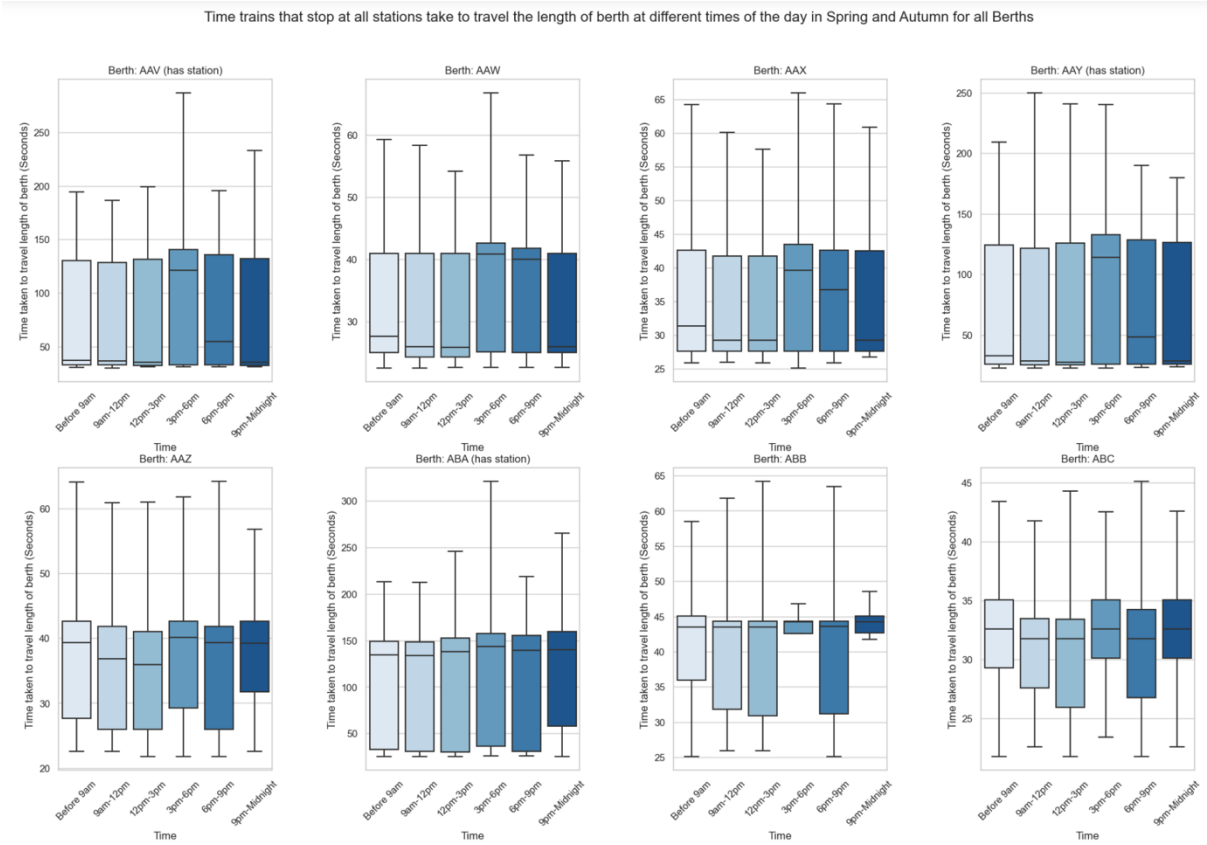


Figure 13, Results of subplot showing the travel time throughout the day for each location.

Just from looking at the Y-axis numbers for each subplot in Figure 13, it becomes apparent that berths with stations have a considerably higher travel time range. All the berths without a station have travel times around the 25- to 65-second range, with berth ABC giving slightly less range of results.

There also appears to be a big increase in median travel times during the 3 p.m.-6 p.m. period for berths that contain a station. This would make sense due to rush hours for passengers that have just finished work; however, the ABA berth does not follow this trend, as this berth instead seems to maintain its high travel time throughout the day. This could simply be due to berth ABA running more trains in the first half of the day.

In most of the berths without stations, the median line tends to be much lower on the first three boxes, giving the impression that travel times increase later in the day for these locations.

4.3.2 Berths with and Without Stations Results

The next visualisation created gave more of a distinction between berths that contain a station and berths that do not (Figure 14).

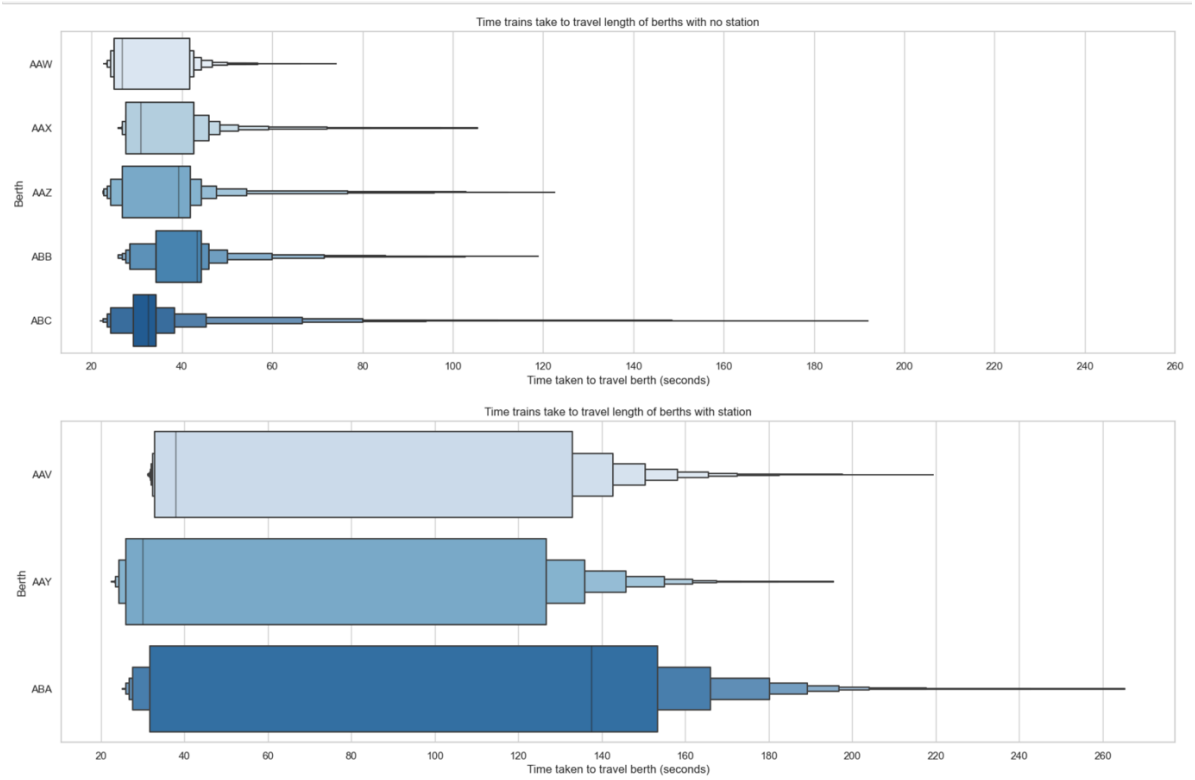


Figure 14, Using two Boxenplots two show travel times for berths with and without stations.

Figure 14 shows the huge disparity in travel times between berths with stations (on the bottom) and berths without stations (on the top). This plot also made it easier to compare similar berths to one another. This shows that berths without stations all have similar median travel times, with the range of their travel times varying, with berth ABC having the largest range and AAW having only a third of that range, which shows a clear effect of location on travel times.

4.3.3 Stations and Their Travel Times Results

For the final visualisation for looking at locations and travel times, the three stations that are included in the dataset were looked at and split into the four train types (Figure 15).

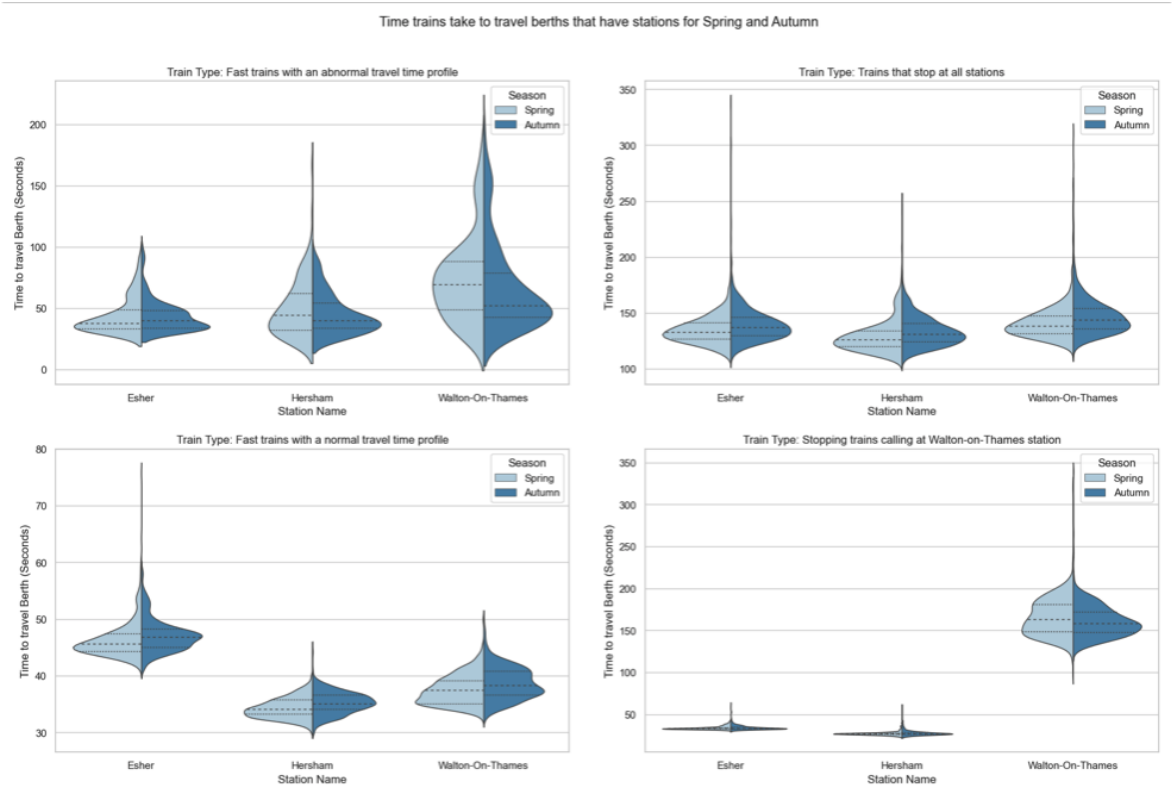


Figure 15, Violin subplot for travel times at different stations.

Figure 15 shows an expected result for trains stopping at Walton-on-Thames to have a much higher travel time at that station. An interesting result, however, for trains that stop at all stations is that the three violin plots for the three stations appear similar for both seasons, suggesting there is very little fluctuation between travel times at different stations.

The symmetry of most of the violin plots gives more credence to the idea from earlier in the project that the season in which a train is travelling has very little effect on travel times.

For fast trains with a normal travel time profile, the travel times were considerably higher when passing through the berth that contains Esher station. This was found to be due to Esher having a considerably longer berth of over 1000 meters compared to the others having around 600. This issue of some values needing to be normalised will be later explored in Section 4.7.

4.4 Signal Aspect Visualisation Results

This section shows the results of plotting visualisations for looking at signal aspects and how they affect a railway system, as well as describing what these results mean.

4.4.1 Aspects of Their Travel Times Results

Using “boxen plots” again gave a good visual description of the distribution of how much time is spent at a berth at each of the three signal aspects (Figure 16).

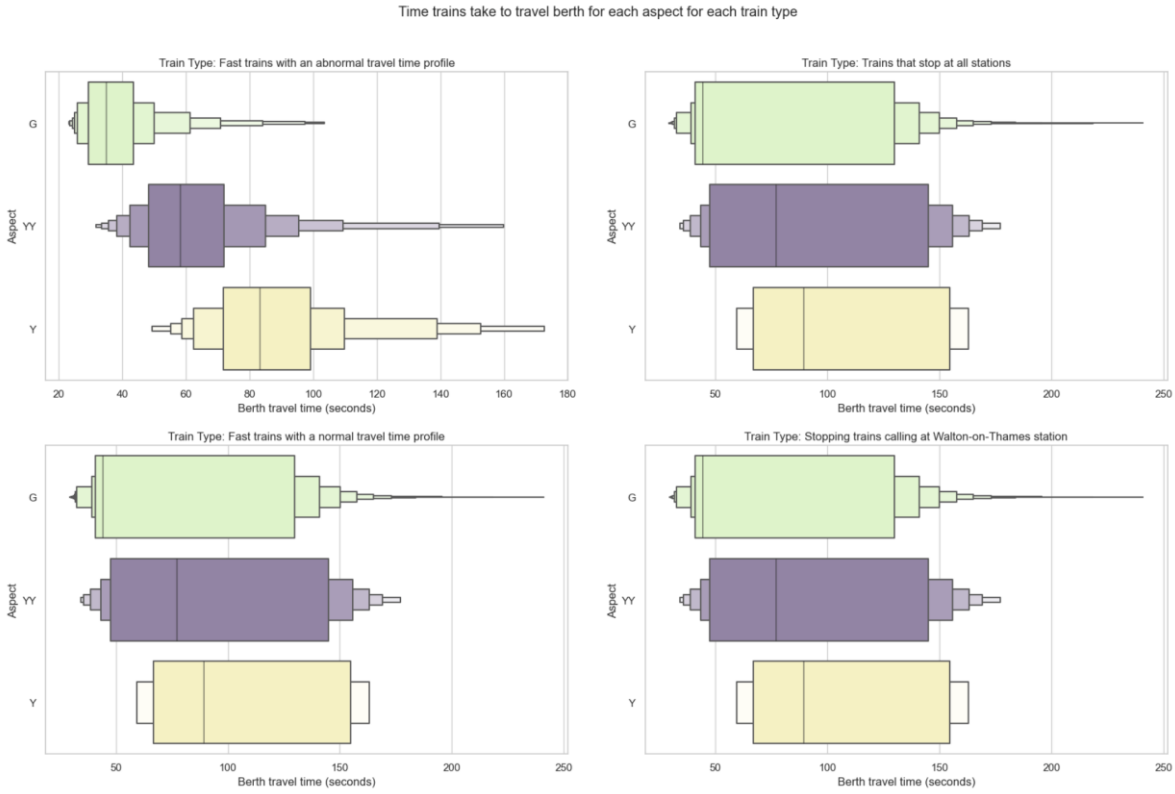


Figure 16, Subplot of Boxen plots showing travel times at each aspect signal for all train types.

Figure 16 shows that for all train types, the Y signal is most commonly travelled at, followed by YY and then G. This means that all trains are typically travelling at their slowest speed to account for trains or stoppages ahead of them. The shape of the plots for stopping trains is very similar to fast trains with a normal travel time profile. However, fast trains with an abnormal travel time profile have a more spread-out shape, meaning their travel times are less predictable.

The information given in this plot tells us that a train’s type has little impact on how long it spends at each aspect signal. This was interesting, as a more expected result would be that fast trains with a normal travel time profile would predominantly be at a G signal.

4.4.2 Aspect Travel Times at Different Berths Results

A slightly different plot for analysing aspect signals was to include locational aspects (Figure 17).

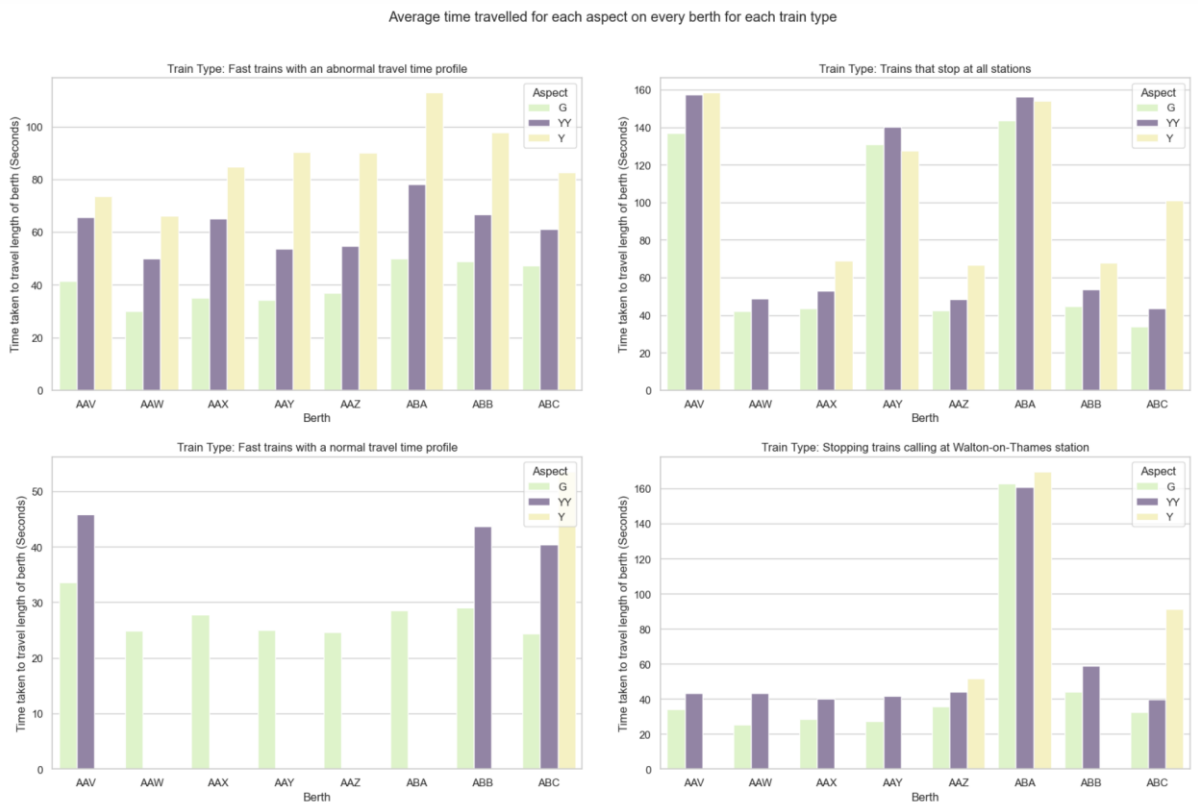


Figure 17, Subplot for all train types and which aspect signals they travel at the most at each berth.

Figure 17 shows that bringing location into the context of aspect signals tells us much more than the previous plot. In the previous plot, it looked like fast trains with a normal travel time profile travelled for longer with a Y signal than a G, which is true, although that is because there is a small sample size of times that these train types travelled with a Y signal, and they regularly travel at G.

In the normal travel time profile, fast trains have hardly any time spent with a Y signal, and the graph on stopping trains that stop at Walton-on-Thames only has Y signals before and during their stopping berth. This shows that stations have the biggest impact on how fast following trains can go.

4.5 Train Type Visualisation Results

This section will show the results of the coding for creating visualisations that show how train types and sizes affect travel times and TSAR.

4.5.1 Time Taken to Travel Train Length Results

The first plot for comparing train types was a scatterplot, which compared the time it takes for a train to travel the length of its coaches to its berth travel time (Figure 18).

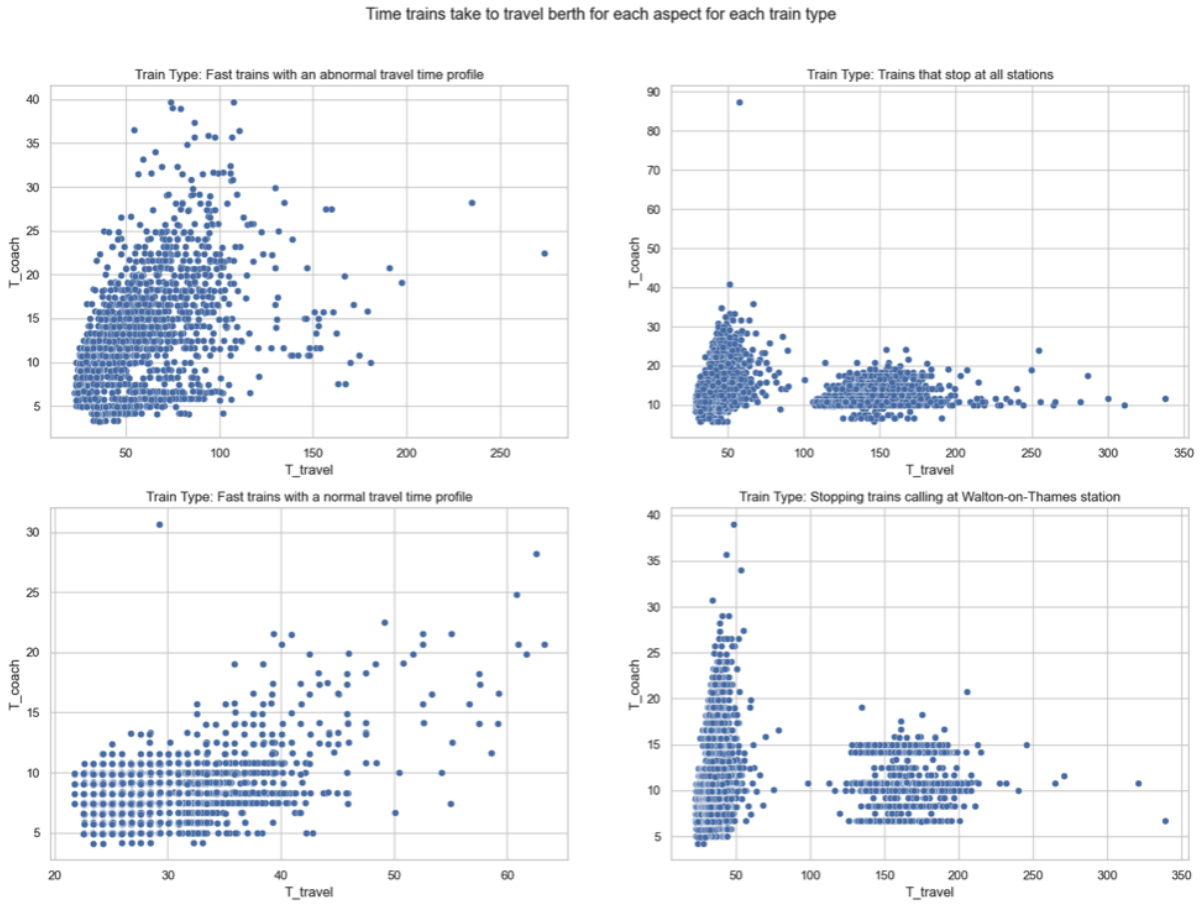


Figure 18, Four-part subplot for each train type and how the time taken to travel their coach length affects travel times.

For this plot, the expectation was that the longer it would take to travel, the longer the coach would be. This is the case for fast trains, which both have a positive correlation; however, stopping trains have more of a cluster that isn't always positive. This could be due to the fact there is a wider range of travel times due to the stations, so the positive correlation is harder to see.

Figure 18 gives evidence to the claim that the size of a train's coaches has a big impact on travel times.

4.5.2 Plotting What Trains Are Powered by Results

When comparing how trains are powered, plotting bar graphs for comparing electric-powered trains and diesel-powered trains helped understand train types. Although this data was from a smaller dataset, it gave insight into how the two types operate (Figure 19).

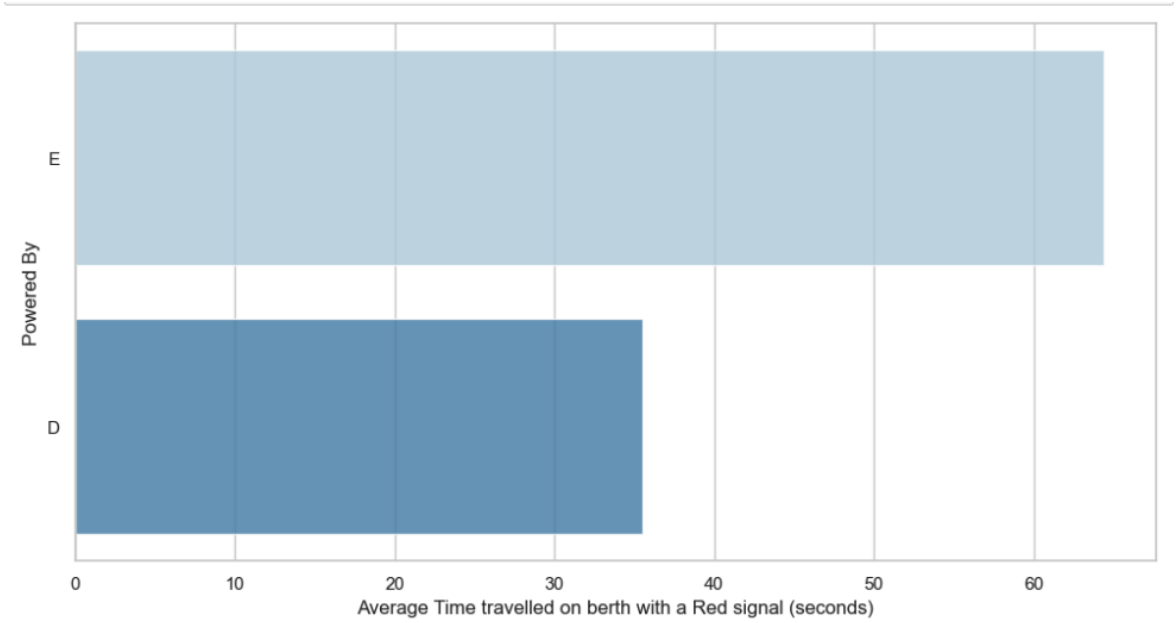


Figure 19, Bar plot showing average travel times for trains powered by diesel and trains powered by electric.

Figure 19 shows that trains powered electrically have longer times spent with a red signal on average than diesel-powered trains. However, after further research into the smaller dataset, it was clear that this fact is due to all diesel trains in the dataset being "fast" trains (Figure 20).

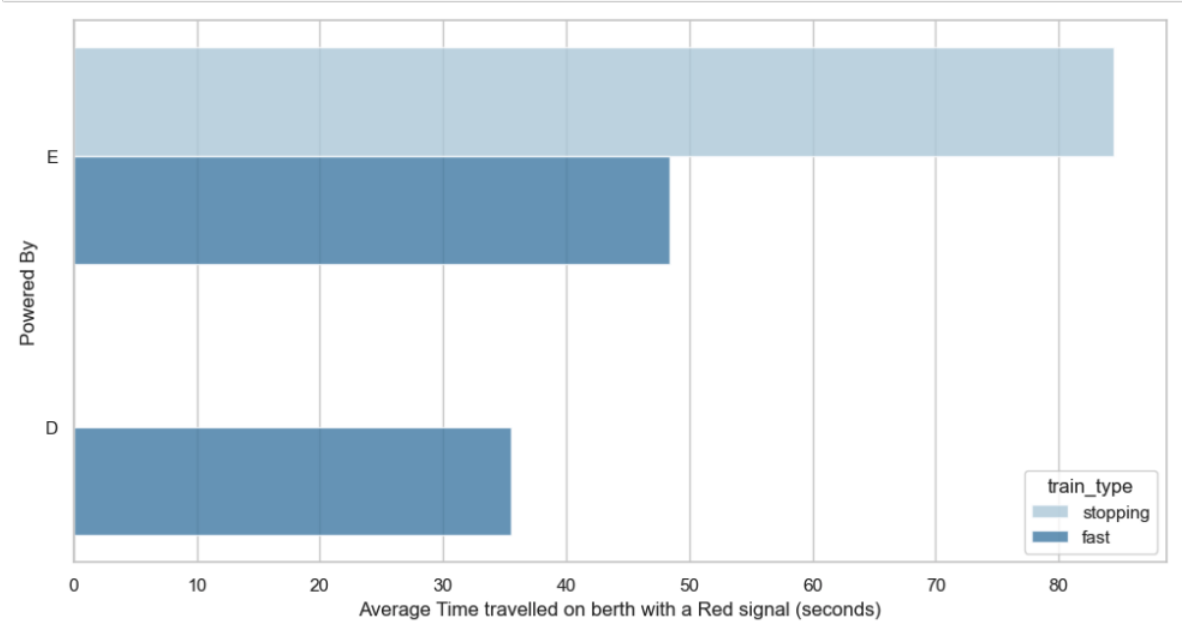


Figure 20, Bar plot showing average travel times for trains powered by diesel and trains powered by electric with their train types included.

Figure 20 shows that there are no stopping trains that are also diesel-powered within the dataset, making the comparison of how the trains are powered unfair. Although Figure 20 does show us a fairer comparison of fast trains for both power types, this also gives us the result that electric-powered trains have higher travel times on average.

4.6 Signal Visualisation Results

This section shows the results of plotting signal times for each signal ID and explores the possible reasons for the results.

4.6.1 Signal Offset Time Visualisation Results

The first signal plot gave the offset times for each signal ID; the original idea of plotting this data was to check if all the signals were working properly and if they were consistent with one another (Figure 21).

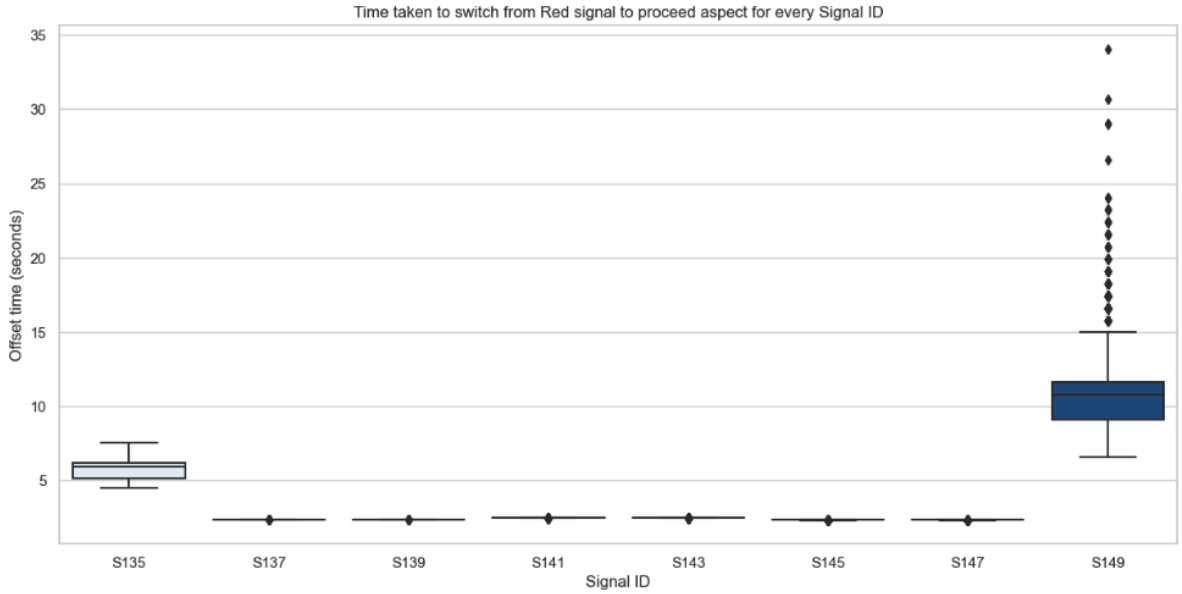


Figure 21, Box plot to show offset times for each signal.

Figure 21 shows a huge difference in offset times between S135 and S149 compared to the rest of the signals. The thin line for the other six signals shows that they have a very small range, and all take around three seconds to change, whereas S135 takes twice as long even at its best change times, and S149 has a range more than triple that of the other six signals. The black dots above the box for S149 show the outliers, meaning that there were many outliers, which were very high for signal times.

4.6.2 Signal Onset Time Visualisation Results

Onset times were also checked for their time in seconds against each signal ID. Checking the time taken to switch from a proceed aspect to a red signal gave more context to the offset time results (Figure 22).

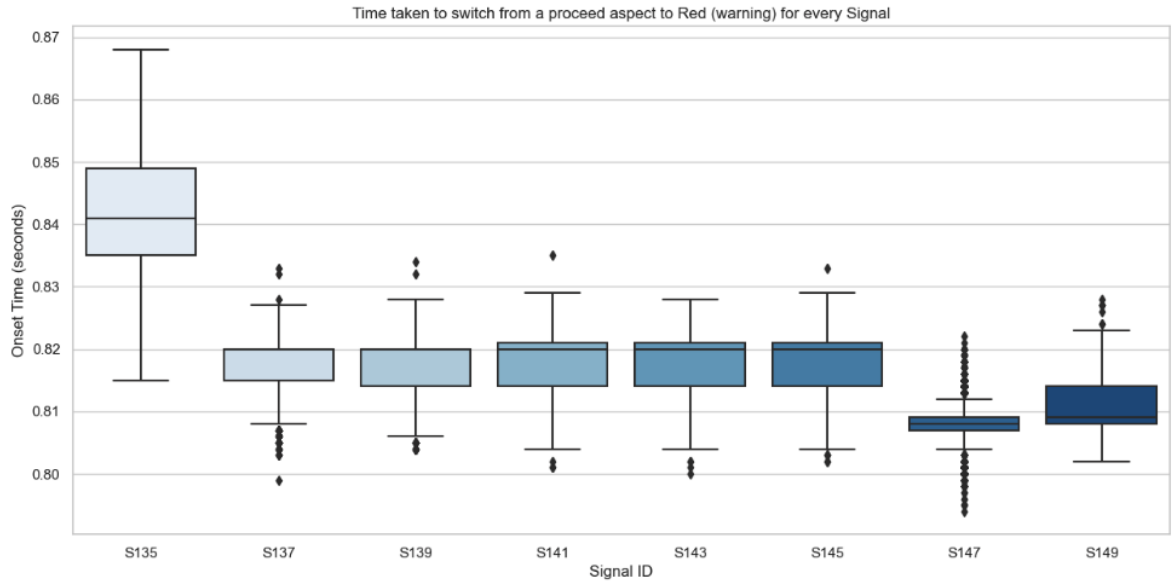


Figure 22, Box plot to show onset times for each signal.

Figure 22 shows that all of the signals have very short onset times compared to their corresponding offset times. Signal S135 gives a higher result, but only by 0.02 seconds on average.

None of these times from Figure 22 give unexpected results, and they all remain consistent, meaning that the signals only have an issue when switching from a red signal to a proceed signal for S135 and S149. This was an important discovery as it gives context to the potentially higher TSAR for the berths that contain the signals S135 and S149 and explains the locational disparity of travel times.

4.7 Normalised Visualisation Results

During the development of the various visualisations presented in this dissertation, it became apparent that for some plots, normalisation of travel time and TSAR was essential for giving a fair comparison. This was due to some berths being longer than others. The types of plots that needed to be normalised were generally location-based.

This section will discuss any amendments to some of the previous visualisations after normalisation and explain what they now show.

4.7.1 Normalised Violin Plot Results

The violin plot used to show the three different stations and their travel times from Section 4.3.3 was normalised to give travel time per 100 meters (Figure 23).

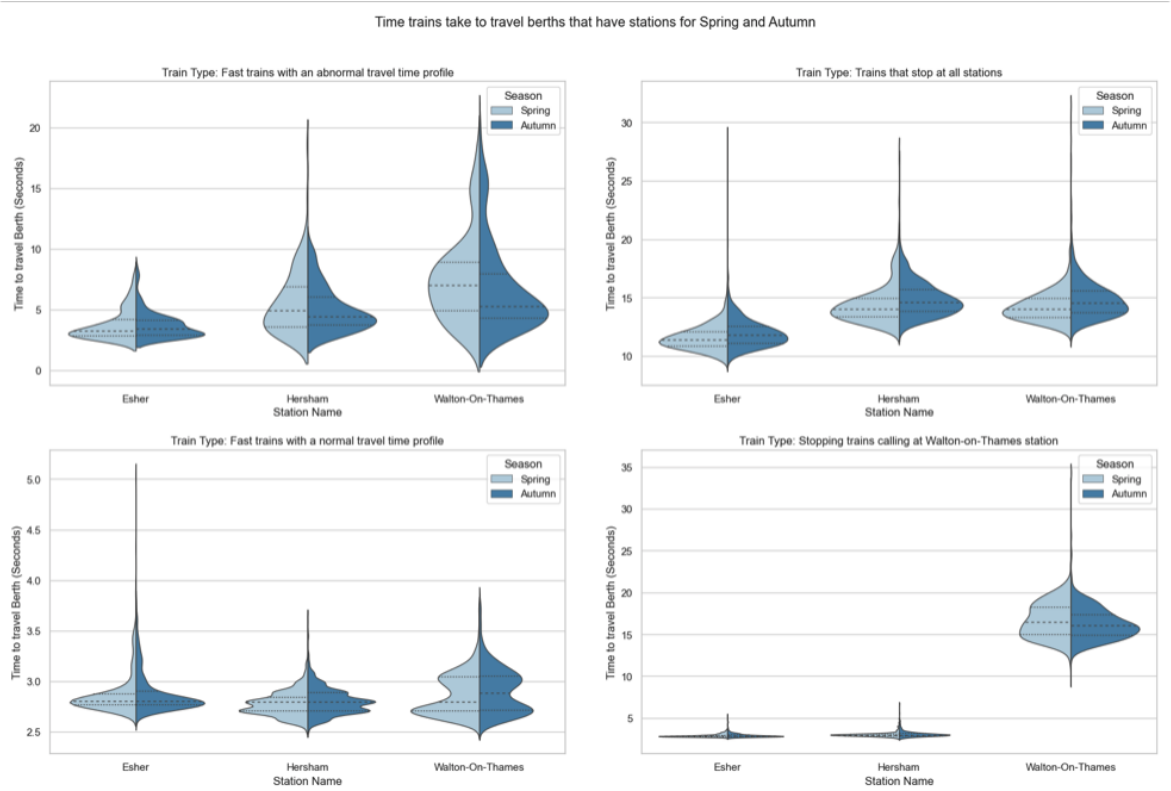


Figure 23, Normalised travel times for each station at different seasons.

Figure 23 now displays a different story for how each station performs with their travel times. Before normalisation, it would have seemed that Esher had much higher travel times for fast trains with a normal travel time profile; however, it can now be seen that this fact was only due to Esher having a much longer berth.

Figure 23 also shows that for trains stopping at all stations, Esher has the lowest travel times by an easily visible distance. This is in contrast to the original plot (Figure 15), which displays Esher as having the highest travel times.

4.7.2 Normalised Aspect for Berth Plot Results

The other visualisation that benefited from normalisation was the plot for viewing how much travel time is spent at each berth for each aspect signal; this can be found in Section 4.4.2.

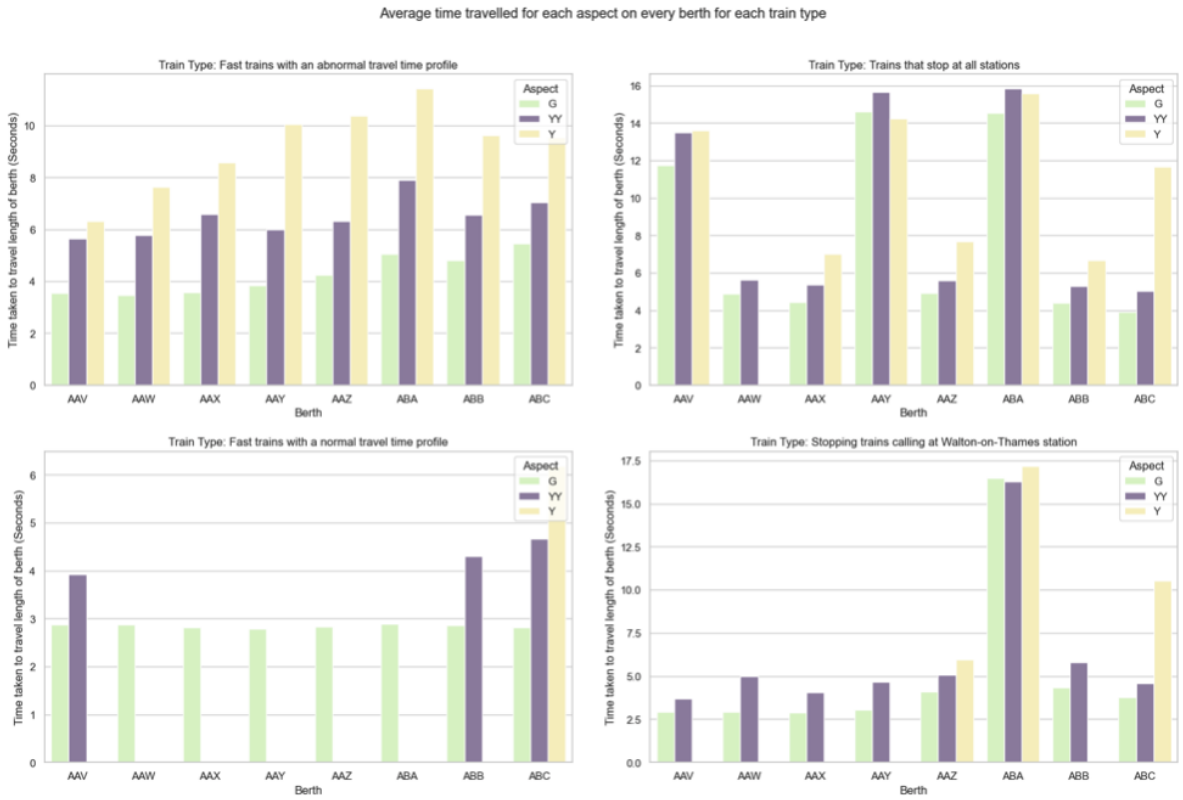


Figure 24, Subplot for all train types and which aspect signals they travel at the most at each berth (normalised).

Figure 24 shows the normalised version of this plot. This version clearly shows how each berth performs with travel times. For fast trains with an abnormal travel time, the graph of ascending travel times up until berth ABA gives an interesting idea that travel times increase along a railway. This could be due to the fact that the further a train gets down a railway, the more trains have been in front, meaning all of the delays from the previous trains trickle down the railway system in a domino-like effect.

5 Conclusion

This section will go through everything learned from the results of this project, what could have been done differently, and any future work that could be expanded upon. It will also check through the original aims and objectives to see which were fulfilled.

5.1 What was Learned

Through the six different aspects of railway data explored throughout this project, a lot was learned about what goes into longer travel times and what could be done to prevent such high travel times.

The visualisations created for analysing how the time of day affects travel times evidenced that the time of day has little to no impact on travel times. This is followed by the conclusion that the season in which a train is travelling has little impact on travel time. There was an increase in autumn, but a larger increase was expected as autumn typically has more varied weather.

Location has a big effect on travel times; this is mainly due to some locations containing stations and others not. However, once stations were removed, it was shown that berth location ABC gave higher travel times than expected, and this could be due to it being the last berth in the dataset. The stations themselves didn't give very varied results in terms of travel times between each other.

When different signal aspects appear in the dataset, they are shown to vary for different locations. Fast trains with a normal travel time profile have little to no aspect signals other than G. Trains start to receive slower aspect signals before and after they stop at a station. The longer times spent with slower signals on berth ABC could be due to that berth containing the signal that was shown to be faulty.

Train types were split up for almost every visualisation within this project and were found to be a big component of what affects travel times. Fast trains and stopping trains acted as expected, with fast trains producing much lower travel times. Fast trains with abnormal travel time profiles should give low travel times but are affected by seeing much more Y and YY signals throughout their journey.

While plotting graphs to look at offset times, it was made clear that the signals S149 and S135 were giving a clear error in their timings. This was found to be a potential cause of higher travel times in the data for specific berths that can contain these faulty signals.

5.2 What Could Have Been Better

An error early on in the development of this project was the underestimation of how much of an impact normalising travel times would have on some plots, and understanding which plots needed this treatment earlier could have more impactful findings.

Outliers for specific values could have been looked into further to try and find a reason for them; for example, the offset times had some very high outliers that were removed from the data but could have been looked into more.

5.3 Further Work

If more time was given for this project, more analysis of train types would be done. More analysis would be done if there was more data available for train types, such as how a train is powered or the fleet a train comes from.

More investigation would go into how the further a berth is into a railway system, the higher the travel times. This would be looked into further with better visual representations, and more reasons would be given for why this is.

5.4 Which Aims & Objectives Were Achieved

This section will go through each objective that was outlined in Section 1.2 and decide if it was achieved and how.

Objective 1:

> ‘Use the given dataset of railway performance to visualise and evaluate whether any locations exhibit higher or lower travel time and TSAR values.’
>
> Objective verdict: Achieved.
>
> Explanation: Locations were evaluated for their travel times using Seaborn bar plots and box plots and evaluated to give concrete conclusions.

Objective 2:

> ‘Use the given dataset of railway performance to visualise and evaluate how train types and sizes affect travel times and TSAR, decide which is more important.’
>
> Objective verdict: Partly achieved.
>
> Explanation: The given railway dataset was used to visualise train types by using Matplotlib subplots to split the data into the four train classes. How trains are powered was also visualised and evaluated using Seaborn. The train type is more important than its size for travel times; although size does have a big impact, it became clear that the train type is the most important component in how it performs.

Objective 3:

> ‘Summarise 3 or 4 pieces of previous research on railway performances with data science.’
>
> Objective verdict: achieved.

Explanation: Three different pieces of previous research were summarised in the background review section of this dissertation and used to give extra information about the work done in this project.

Objective 4:

'Use a dataset of railway performance to visualise and evaluate how much the weather season will affect travel times and TSAR.'

Objective verdict: achieved.

Explanation: The dataset was used with Seaborn and data analysis libraries to plot box plots and violin plots displaying the data for spring and autumn respectively.

Objective 5:

'Evaluate how much time different train types spend with different aspect signals. Do specific locations give more Y signals?'

Objective verdict: achieved.

Explanation: Boxen plots were used to visualise how long different train types spend with different aspect signals; locations that contain a station gave higher time spent with a Y signal, and the ABC berth gave Y signals for every train type, even with fast trains.

Objective 6:

'Use the given dataset to check for inconsistencies or failings in the time taken to switch signals via visualisations.'

Objective verdict: achieved.

Explanation: Offset and onset times were visualised using box plots for each signal ID. The times were analysed for inconsistencies.

Objective 7:

'Explore how the time of day affects railway performance. Do peak times give higher travel times?'

Objective verdict: achieved.

Explanation: The railway data was used to determine when different events occurred in a 24-hour day and plotted using box plots. Peak times only cause a slight increase in travel times on average.

6 References

[1] *Transport statistics Great Britain 2020 - gov.uk* (2020). Available at: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/945829/tsgb-2020.pdf (Accessed: April 22, 2023).

[2] *Example gallery* (2012-2022) *Example gallery - seaborn 0.12.2 documentation*. Available at: https://seaborn.pydata.org/examples/index.html (Accessed: April 22, 2023).

[3] McKinney, W. (2011) *Pandas: A Foundational Python Library for Data Analysis and Statistics*. Available at: https://www.dlr.de/sc/Portaldata/15/Resources/dokumente/pyhpc2011/submissions/pyhpc2011_submission_9.pdf (Accessed: 08 May 2023).

[4] König, E. (2020) *A review on railway delay management - public transport*, *SpringerLink*. Springer Berlin Heidelberg. Available at: https://link.springer.com/article/10.1007/s12469-020-00233-1#Abs1 (Accessed: April 22, 2023).

[5] Christian Liebchen a *et al.* (2009) *Computing delay resistant railway timetables*, *Computers & Operations Research*. Pergamon. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0305054809000926 (Accessed: April 22, 2023).

[6] A. and Bešinović, N. (September 2019) *Resilience in Railway Transport Systems: A literature review and research agenda*, *Taylor & Francis*. Available at: https://www.tandfonline.com/doi/full/10.1080/01441647.2020.1728419 (Accessed: April 22, 2023).

[7] *Who we are* (2022) *Network Rail*. Available at: https://www.networkrail.co.uk/who-we-are/ (Accessed: April 22, 2023).

[8] Myatt, G.J. (2007) *Making sense of data*, *Google Books*. Google. Available at: https://books.google.co.uk/books?hl=en&lr&id=Z5gpG0u3jecC&oi=fnd&pg=PR5&dq=exploratory%2Bdata%2Banalysis&ots=TaQ5ncg1DS&sig=yXzJuDgoV3rJtv3uV9pi9qNfqRU&redir_esc=y#v=onepage&q=exploratory%20data%20analysis&f=false (Accessed: April 26, 2023).

[9] Hartwig, F. and Dearing, B.E. (1979) *Exploratory Data Analysis*, *Google Books*. Google. Available at: https://books.google.co.uk/books?hl=en&lr=&id=jF8QC-BkhvQC&oi=fnd&pg=PA5&dq=exploratory%2Bdata%2Banalysis&ots=KJ7qBKEAlM&sig=L3TmBytpKpXbiVdvyP_OoJjpg10&redir_esc=y#v=onepage&q=exploratory%20data%20analysis&f=false (Accessed: April 26, 2023).

[10] Da Silva, P.P. (February 2023) "A bundle of starting information given by the project supervisor for TSAR data and railway systems."

[11] Randles, B.M. *et al.* (2017) *Using the Jupyter Notebook as a Tool for Open Science: An Empirical Study*. Available at: https://ieeexplore.ieee.org/abstract/document/7991618 (Accessed: 08 May 2023).

[12] V, L.G. (2021) *Using pandas and Python to explore your dataset, Real Python*. Real Python. Available at: https://realpython.com/pandas-python-explore-dataset/ (Accessed: April 27, 2023).

[13] Waskom, M. (2022) *Statistical estimation and error bars, Statistical estimation, and error bars - seaborn 0.12.2 documentation*. Available at: https://seaborn.pydata.org/tutorial/error_bars.html#:~:text=The%20error%20bars%20around%20an,data%20has%20a%20broader%20spread. (Accessed: May 1, 2023).

[14] Sládek, M. *et al.* (2020) *Chronotype assessment via a large-scale socio-demographic survey favours yearlong standard time over daylight saving time in Central Europe, Nature News*. Available at: https://www.nature.com/articles/s41598-020-58413-9 (Accessed: 02 May 2023).

[15] Hu, K. (2020) *Become competent within one day in generating Boxplots and violin plots for a novice without prior R experience, MDPI*. Available at: https://www.mdpi.com/2409-9279/3/4/64 (Accessed: 02 May 2023).

[16] Bisong, E. (1970) *Matplotlib and Seaborn, SpringerLink*. Available at: https://link.springer.com/chapter/10.1007/978-1-4842-4470-8_12 (Accessed: 08 May 2023).

[17] Spitzer, M. *et al.* (2014) *BoxPlotR: A web tool for generation of box plots, Nature News*. Available at: https://www.nature.com/articles/nmeth.2811 (Accessed: 08 May 2023).

[18] Sadiku, M.N.O. *et al.* (2016) *Data Visualization - Researchgate*. Available at: https://www.researchgate.net/profile/Adebowale-Shadare/publication/311597028_DATA_VISUALIZATION/links/5851945608aef7d0309f20a7/DATA-VISUALIZATION.pdf (Accessed: 08 May 2023).

[19] Singh, D. and Singh, B. (2019) *Investigating the impact of data normalization on classification performance, Applied Soft Computing*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S1568494619302947 (Accessed: 08 May 2023).

[20] Pajankar, A. (2021) *Revisiting matplotlib visualizations, SpringerLink*. Available at: https://link.springer.com/chapter/10.1007/978-1-4842-7410-1_4 (Accessed: 08 May 2023).

[21] Nguyen, Q.V. *et al.* (2020) *Evaluation on interactive visualization data with scatterplots, Visual Informatics*. Available at: https://www.sciencedirect.com/science/article/pii/S2468502X20300358 (Accessed: 08 May 2023).

[22] Moon, H.S. *et al.* (2022) *Automated multimodal segmentation of acute ischemic stroke lesions on clinical MR images, Magnetic Resonance Imaging*. Available at: https://www.sciencedirect.com/science/article/pii/S0730725X22000959 (Accessed: 09 May 2023).

Appendix A

Python code for creating various data attributes, Pandas.

Time of day period categorical attribute:

```
# Convert full date to just time
berth_events['t_enters_seconds'] = pd.to_datetime(berth_events.t_enters, format='%Y/%m/%d %H:%M:%S.%f')
berth_events['t_enters_seconds'] = [time.time() for time in berth_events['t_enters_seconds']]

# Convert time into just seconds
berth_events['t_enters_seconds'] = berth_events['t_enters_seconds'].apply(lambda x: x.hour * 3600 +
                                                                          x.minute * 60 + x.second)

# Time intervals for times of the day
berth_events['time_cat'] = pd.cut(x=berth_events['t_enters_seconds'],
                                  bins=[0, 32400, 43200, 54000, 64800, 75600, 86400],
                                  labels=('Before 9am', '9am-12pm', '12pm-3pm', '3pm-6pm', '6pm-9pm',
                                          '9pm-Midnight'))
```

Season categorical attribute:

```
# Converting date to total seconds
berth_events['total_seconds'] = pd.to_datetime(berth_events['t_enters']).astype(int) // 10**9

# Creating intervals for seasons
berth_events['season_cat'] = pd.cut(x=berth_events['total_seconds'], bins=[0, 1665000000, 1765000000],
                                    labels=('Spring', 'Autumn'))
```

Hour of the day categorical attribute:

```
berth_events['hour'] = pd.to_datetime(berth_events['t_enters'],
                                      format='%Y/%m/%d %H:%M:%S.%f').dt.hour
```

Normalisation attributes:

```
berth_events['L.meters'] = berth_events['L.miles'] * 1609
berth_events['length_per_100'] = berth_events['L.meters'] / 100
berth_events['TSAR-NORM'] = berth_events['TSAR'] / berth_events['length_per_100']
berth_events['TRAVEL-NORM'] = berth_events['T_travel'] / berth_events['length_per_100']
```

Appendix B

Python code for creating various data visualisations using Seaborn.

Time-of-day subplot intervals:

```
sns.set_theme(style="whitegrid")


ticks00 = [20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140]
ticks01 = [40, 60, 80, 100, 120, 140, 160, 180, 200, 220, 240, 260, 280]
ticks10 = [25, 30, 35, 40]
ticks11 = [20, 30, 40, 50, 60, 70]

ax = sns.boxplot(ax = axes[0, 0], x="time_cat", y="T_travel", hue="season_cat", data=berth_events_fa,
                 showfliers = False).set(title="Train Type: Fast trains with an abnormal travel time profile",
                                         xlabel='Time', ylabel='Time taken to travel length of berth (Seconds)',)
axes[0, 0].legend(title='Season', loc='upper right')


ax = sns.boxplot(ax = axes[0, 1], x="time_cat", y="T_travel", hue="season_cat", data=berth_events_sa,
                 showfliers = False).set(title="Train Type: Trains that stop at all stations",
                                         xlabel='Time', ylabel='Time taken to travel length of berth (Seconds)', )
axes[0, 1].legend(title='Season', loc='upper right')


ax = sns.boxplot(ax = axes[1, 0], x="time_cat", y="T_travel", hue="season_cat", data=berth_events_fn,
                 showfliers = False).set(title="Train Type: Fast trains with a normal travel time profile",
                                         xlabel='Time', ylabel='Time taken to travel length of berth (Seconds)', )
axes[1, 0].legend(title='Season', loc='upper right')


ax = sns.boxplot(ax = axes[1, 1], x="time_cat", y="T_travel", hue="season_cat", data=berth_events_sw,
                 showfliers = False).set(title="Train Type: Stopping trains calling at Walton-on-Thames station",
                                         xlabel='Time', ylabel='Time taken to travel length of berth (Seconds)', )
axes[1, 1].legend(title='Season', loc='upper right')

fig.suptitle("""Time trains take to travel the length of berth at different times of the day in
Spring and Autumn for different train types""")
fig.subplots_adjust(top=0.9)
plt.show()
```

Signal offset time plot:

```
fig, ax = plt.subplots(figsize = (15, 7))

g = sns.boxplot(x="signal", y="T_offset", data=berth_events)

g.set(title="Time taken to switch from Red signal to proceed aspect for every Signal ID",
      xlabel='Signal ID', ylabel='Offset time (seconds)')

plt.show()
```

Correlation matrix:

```
berth_events_corr = berth_events_corr.corr()

plt.title("Correlation matrix")
sns.heatmap(berth_events_corr, annot=True)
plt.xticks(rotation=25)
plt.show()
```

Aspect signal subplots:

```
ax = sns.barplot(ax = axes[0, 0], x="berth", y="T_travel", hue="aspect",  errorbar=None,
                 palette={"G": "#D8FCBE", "Y": "#FFF9B5", "YY": "#957DAD"},
                 data=berth_events_fa, ).set(title="Train Type: Fast trains with an abnormal travel time profile",
                                             xlabel='Berth',
                                             ylabel='Time taken to travel length of berth (Seconds)')
axes[0, 0].legend(title='Aspect', loc='upper right')

ax = sns.barplot(ax = axes[0, 1], x="berth", y="T_travel", hue="aspect", errorbar=None,
                 palette={"G": "#D8FCBE", "Y": "#FFF9B5", "YY": "#957DAD"},
                 data=berth_events_sa, ).set(title="Train Type: Trains that stop at all stations",
                                             xlabel='Berth',
                                             ylabel='Time taken to travel length of berth (Seconds)')
axes[0, 1].legend(title='Aspect', loc='upper right')

ax = sns.barplot(ax = axes[1, 0], x="berth", y="T_travel", hue="aspect", errorbar=None,
                 palette={"G": "#D8FCBE", "Y": "#FFF9B5", "YY": "#957DAD"},
                 data=berth_events_fn, ).set(title="Train Type: Fast trains with a normal travel time profile",
                                             xlabel='Berth',
                                             ylabel='Time taken to travel length of berth (Seconds)')
axes[1, 0].legend(title='Aspect', loc='upper right')
```

Travel times against time to travel coach scatterplots:

```
ax = sns.scatterplot(ax = axes[0, 0], x="T_travel", y="T_coach", linewidth=0.3, edgecolor="w",
                 data=berth_events_fa, sizes=(1, 8))
axes[0, 0].set(title="Train Type: Fast trains with an abnormal travel time profile")

ax = sns.scatterplot(ax = axes[0, 1], x="T_travel", y="T_coach", linewidth=0.3, edgecolor="w",
                 data=berth_events_sa, sizes=(1, 8))
axes[0, 1].set(title="Train Type: Trains that stop at all stations")

ax = sns.scatterplot(ax = axes[1, 0], x="T_travel", y="T_coach", linewidth=0.3, edgecolor="w",
                 data=berth_events_fn, sizes=(1, 8))
axes[1, 0].set(title="Train Type: Fast trains with a normal travel time profile")

ax = sns.scatterplot(ax = axes[1, 1], x="T_travel", y="T_coach", linewidth=0.4, edgecolor="w",
                 data=berth_events_sw, sizes=(1, 8))
axes[1, 1].set(title="Train Type: Stopping trains calling at Walton-on-Thames station")
```

Travel time for stations, violin subplots:

```
ax = sns.violinplot(ax = axes[0, 1], x="station", y="T_travel", split=True,
                    inner="quart", linewidth=1, hue="season_cat", data=berth_events_stations_sa,
                    showfliers = False)
axes[0, 1].set(title="Train Type: Trains that stop at all stations", xlabel='Station Name',
               ylabel='Time to travel Berth (Seconds)')
axes[0, 1].legend(title='Season', loc='upper right')
```