The Doctor will See You Now: Personalised Waiting Times

Emma Tarmey, 2940 4045

Supervisors:
Florina Borca
Edilson Arruda

September 2022

Dedication

Dedicated to my close friends Charlotte Dean, Joshua Dickman and Tereza Rasova, whose warmth and support brought me through a very difficult year.

Abstract

Background: The COVID-19 pandemic has put incredible strains on all aspects of healthcare, both in the UK and globally. In particular, due to massively increased demand for emergency services across hospitals in the UK, large numbers of elective services and procedures have been cancelled or postponed. Consequently, the waiting list for backlogged elective procedures has become significantly longer and waiting times are more uncertain. In this paper, we seek to understand and estimate waiting list times for elective procedures carried out in the Trauma and Orthopaedics department of the University Hospital Southampton NHS Trust (UHS). We do so in order to provide personalised waiting time estimates to patients currently awaiting elective procedures within the Trauma and Orthopaedics department of UHS.

Methodology: We seek to do so by means of examining pseudonymised elective procedure data from UHS during a two-year window. We then perform exploratory analysis to identify the most key predictive variables for determining the real-world waiting time for a particular patient (service type, clinician assigned, age and so on). We then construct a model using these predictive variables to enable UHS to estimate personalised waiting times to patients using known results from queueing theory and stochastic simulation. Finally, using this repeated simulation of this model, we produce estimated waiting time distributions for various types of new patient joining the queue. We then conclude by identifying the most important correlation between patient factors and the patient waiting list.

Objectives: The three deliverables for this project are this dissertation, the predictive model itself and a presentation of key results and recommendations.

Contents

Chapter 1

Problem Introduction

1.1 Introduction

In this report, we outline the approach taken answering two main research questions:

1. Can we create a model enabling UHS to give a personalised waiting time estimate for patients on a waiting list?
2. Can we identify the most important correlation to the patient waiting list?

We also outline the results of our investigation into these questions, conclusions we draw from these results and finally room for further development. In particular, chapter one introduces the problem itself of patient waiting time estimation and the surrounding context of the UHS Trauma and Orthopaedics Department. Chapter two then summaries the literature review, looks at various proposed solutions and outlines the solution chosen as well as the design of our model. Chapter three outlines the design of the model in more detail, as well as going in-depth on the software-side of development, which forms our second deliverable. Chapter four then presents the results of repeated simulation using the model developed. Chapter five summarises the conclusions we draw from our results, as well as presenting the "most important correlation", our third deliverable. Chapter six then discusses further avenues for investigation, areas for improvement within our existing investigation and makes a case for repeating this experiment in two years time with updated data. Finally, the appendices act as a reference point for all programming code, raw command outline and additional plots used within the report.

1.2 UHS Trauma and Orthopaedics

In a hospital, a given ward will treat incoming patients. For various reasons, wards have a finite upper limit to the number of patients that are capable of treating in a given week. Despite attempts to maximise this capacity, referrals will still at times build up and form a waiting list. Moreover, different patients require different services from the ward and will have different clinical priority levels. Additionally, the rate of incoming new referrals to the ward is also itself random. As a result, we can see clearly that the referral waiting list for a given ward is a complicated system, with several independent sources of uncertainty. Moreover, the allocation of appointments to waiting patients is itself deterministically complicated, with capacity and service times being variable by service type.

In conclusion, we seek to determine how long will a new patient wait to be treated. Towards the goal of doing so, we determine the queue clear time for a given service area at a given priority level within the “Trauma and Orthopaedics” (T&O) unit at the University Hospital Southampton (UHS). Within T&O, and more broadly across the NHS, patients seek one of two types of service from a given healthcare provider:

- Elective
- Non-Elective (emergency)

Due to elective (non-emergency) procedures being non-life-threatening, non-elective procedures generally take priority when filling a given ward’s capacity. This creates three broad categories of patient priority to consider:

- 1 - Routine
- 2 - Urgent
- 3 - Two-week Wait

These priority levels are used internally within UHS to allocate available appointments to waiting patients. In particular, higher priority patients are always prioritised. This is mainly because “Urgent” referrals are life-threatening emergency cases, and as a result must be treated soon in order to save their lives. Whereas, “Routine” referrals, though also important, are not life-threatening cases. As such, they are not prioritised for available slots, instead being treated with capacity not already used in emergency cases.

1.3 Key Terminology

Across the investigation, we use a number of key terms and acronyms to describe T&O’s operations. For simplicity, we define them all upfront as follows:

- “Referrals” specify incoming requests to the T&O department for one given patient to receive one given treatment. These referrals are then processed ...

- "RTT" times specify "referral-to-treatment" times, measured either in days or weeks. This is the measure of time taken between a given referral being received by T&O and treatment being provided to the referred patient. Though, a patient referral may exit the referral system for reasons other than treatment, such as external care, additional diagnoses, patient cancellation or patient death.
- "TCI" dates specify "to-come-in" dates. These refer to the dates in the future at which treatment is expected to be provided.
- "Clock Starts" refer to the date at which referrals are received by T&O, and internal stop-clocks begin to measure the RTT time.
- "Clock Stops" refer to the date at which referrals exit the T&O system, for various reasons.
- "Incomplete referrals" specify referrals within the T&O system that have been received but have not yet exited, i.e referrals that have a given clock start but no clock stop. As a result, an RTT time is not yet measureable.
- "Complete referrals" specify referrals within the T&O system that have both a clock start ans clock stop recorded. As such, an RTT time is now measurable.
- "OPCS" refers to the "Office of Population Censuses and Surveys" Classification of Interventions and Procedures, a classification and description of all procedures carried out within the NHS. For the purposes of this report, we will refer to these simply as "OPCS codes" or "procedure codes".

1.4 Data-sets Provided

The UHS Trauma and Orthopaedics provided a number of data-sets to facilitate the investigation, these being:

- A copy of the "incompletes" data-set of currently backlogged referrals (snapshot taken 26-06-2022)
- A copy of the "clock-stops" data-set of historically completed referrals (snapshot from 22/04/2015 to 31/05/2022)
- A copy of the "emergency" data-set of historical treatments for emergency procedures (snapshot from 22/04/2015 to 31/05/2022)
- A copy of the "priorities" data-set detailing CWE prioritisation
- A procedure decode lookup table (OPCS codes into named procedures)
- A summary of referral numbers to T&O
- A breakdown of referrals by service area (also referred to as speciality)

- A summary of referrals by staff member (identified by staff code)
- A staff service area lookup table (staff codes into service areas)

These data were provided by the sponsors for this project, and form the basis of all our results. Though, on account of General Data Protection Regulation (GDPR), a non-disclosure agreement was signed at project start requiring all copies of the provided data to be deleted as of September 15^{th} 2022. Consequently, as the data used cannot then be provided alongside this report to reproduce the given results, further descriptions of the necessary input will be provided in section 4.2. For the purposes of conveying our results in absence of being able to provide our data, we will explain in detail the most important three data-sets provided in appendix D.

1.5 Variable Treatment Times, Referral Rates and Treatment Rates

Across T&O, we see great variability in the waiting times for patients to be treated. These times are measured by the "clock-stops" data-set, and measure the time between a given referral being received by T&O and treatment being provided. These referral-to-treatment times (RTT times) take the following distribution within the provided data:

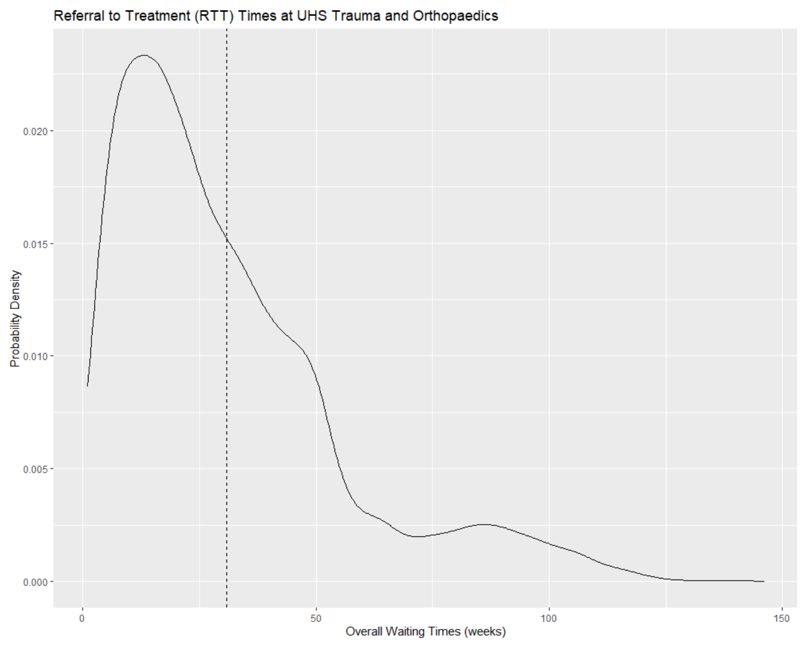
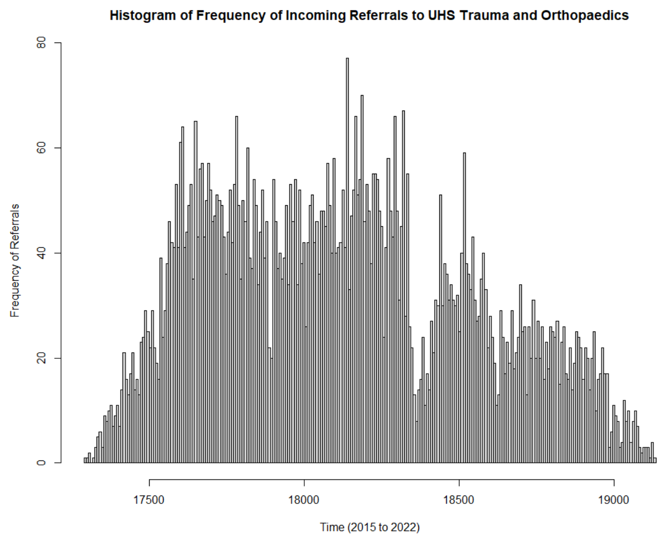
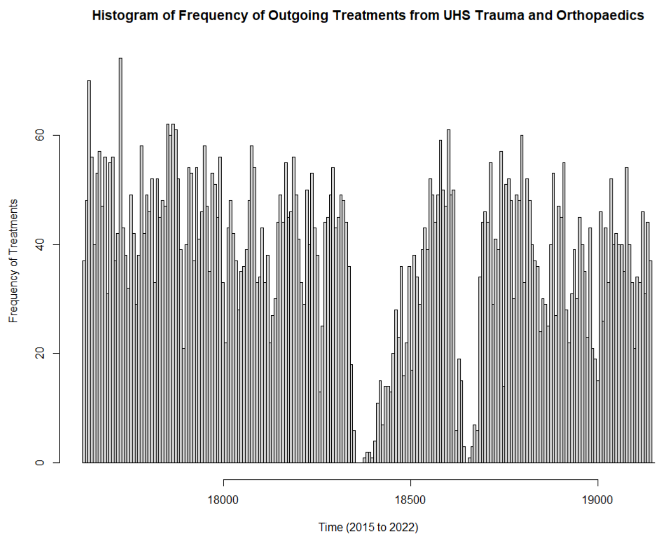


Figure 1.1: Incoming Referral Rates over Time to T&O

As we see in the above, patients across the clock stops data-set tend to almost always be seen in under 50 weeks. Though, the skew we see to the positive causes the mean value to be higher (further to the right than) the peak of the probability distribution.

Additionally, we see the following variability in both incoming referral rates and outgoing treatment rates for UHS Trauma and Orthopaedics:





As we see in the above, there exists great variability in both frequencies. Moreover, we see in the right-hand plot the effects of the COVID-19 pandemic on treatment rates within T&O, with large downward spikes near around beginning and end of 2020.

1.6 Project Objectives and Deliverables

From here, our key objectives are:

- To design and implement our model in software enabling UHS to provide personalised estimates of waiting times
- To identify from the data the most important correlation to real-world waiting times

We have the following project deliverables:

- This dissertation describing all method and results
- The software tool, as specified in appendix C
- The most important correlation, as specified in section 5.3

Chapter 2

Proposed Solutions

2.1 Exploratory Analysis of Provided Data

2.1.1 Basic Overview

To begin, we need to clean and re-format the provided data. Taking the input data-sets as CSV files, we prepare the data in the following way:

- Take as input the historical clock-stops and current incompletes, alongside relevant lookup tables as CSV files for ease of file reading and writing
- Filter all anomalies, clean the data (trimming white-space, extracting staff codes and encoding dates as numerical values)
- Filter data to only records concerning unit 110 Trauma and Orthopaedics (i.e: excluding unit 108 Spinal)
- Filter data to only records concerning inpatient referrals (i.e: no outpatient clock-stops or day-cases)
- Assign to each record in both data-sets a Service Area label, using lookup tables for staff service-areas as a go-between (see section 2.1.2)
- Separate out both data-sets into subsets by service-area and priority-level
- Generate time-series of incoming referrals and outgoing cases, separated as above
- Save all such time-series as CSV files

We began our investigation in RStudio, unpacking primarily the "clock-stops" and "incompletes" data-sets. The other data were used as lookup tables and contextual information towards the goal of identifying the variable or set of variables which best describe or predict the "rtt_weeks_wait" variable. Primarily, we explore the other columns of the clock-stops data-set to do so, as these columns specify other details for a given referral, including key information about the patient themselves. Though, we use the other data-sets to create the classification "Service Area" based on information directly from T&O. We will then attempt to use the predictive model produced to determine "rtt_weeks_wait" estimates for each record in the current incompletes backlog.

2.1.2 Identifying Key Variables

The clock-stops data as provided in the given time interval by T&O, after processing and filtering, provided 8143 patient records (observations) each of 55 variables. For simplicity, we categorise these 55 variables into several related clusters:

- Referral-to-treatment Data (rtt_id, rtt_start, rtt_days, rtt_weeks, rtt_end, rtt_end_reason)
- Pathway Data (path_act_str-id, path_act_int_id, patient_path_id)
- Last Attendance Data (last_att_id, last_att_date, last_att_out)
- Internal Organisational Data (cons_code, spec_code, awl_id, excl_reasons, wait_list_type)
- Personal Data (age_today, gender_code, ethnic_cat)
- Referral Source Data (ref_req_date, ref_id, dec_to_admit_date, adm_method_code)
- External Organisational Data (org_code_iss, source_ref, org_id_prov, org_site_id)
- Future Appointment Data (out_future_app_date, due_date, out_app_date)
- DNA / Cancellation Data (last_dna_date, cancel_date)
- Priority Data (procedure_priority, diag_priority, date_last_priority_review_review)

We then, in terms of identifying real-world predictors, seek to determine which cluster or clusters would work best to predict the variable "rtt_weeks_wait".
The appointment allocation algorithm will always prioritise higher priority patients. As a result, within a given service area, "Two-week wait" patients will always be seen first. Then emergency patients, and then finally routine patients. As a result, we know that this would in a predictive model always be relevant to consider.

Additionally, we have six key service areas to consider:

- Foot and Ankle
- Hand
- Hip

- Hip and Knee
- Knee
- Shoulder

We assign these service areas to individual clock-stop records based on the concultant assigned. Individual consultants at the ward specialise in one of the six above areas. As a result, it is worth considering these service areas as separate though dependent queues within a larger queuing system.

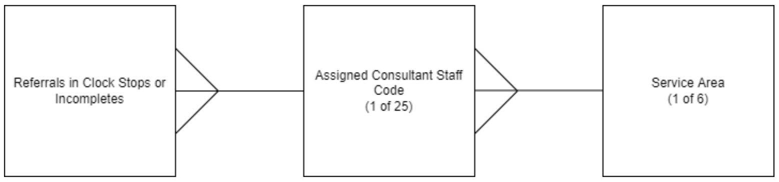

Referrals in Clock Stops or Incompletes
Assigned Consultant Staff Code (1 of 25)
Service Area (1 of 6)

Figure 2.1: Data Relationship

The drawback to this approach is that we create an additional service area categorisation "Hip & Knee", as some staff members specialise in both. Moreover, the OPCS lookup table does not necessarily disambiguate cases where staff specialising in these two areas take a referral, as procedure descriptions can be vague enough to be applied to multiple parts of the body. As a result, for simplicity, we allow for the number of stated service areas to increase from five to six. This does not affect our assumptions luckily, as staff allocations determine treatments and staff specialities determine service areas. Since staff always have one of these now six service areas listed, we may still categorise all referrals using consultant codes to assign service areas.

The task given by UHS was to personalise a waiting time estimate for a given patient beyond the population-wide estimate. As a result, the personalisation should be patient-dependent. Sub-setting the patient population and experimenting, using the data-sets provided, we decide to use the following factors to develop a personalised model:

- Patient Priority ("priority_type_code" in the "Clock Stops" Dataset)
- Service Area (as assigned to each referral using "consultant_code" in the "Clock Stops" Dataset and the staff service area looku[table)
- Date of referral to T&O (date that the "incompletes" snapshot was taken, specifying the existing backlog of referrals)

2.2 Methods Investigated

2.2.1 Purely Statistical Approach

We may approach by means of considering population-wise statistics about the nature of the historical waiting times. This would then enable a thorough description of how long T&O typically takes to treat a patient from referral. We do so by considering the referral-to-treatment waiting times (RTT times) from our existing data and summarising the distribution. We may then personalise these statistics for a given patient by isolating a given service area and given priority level. As a result, we will then be able to draw confidence intervals across this personalised RTT distribution in order to generate personalised waiting time estimates. Summaries could be prodiced in the following way:

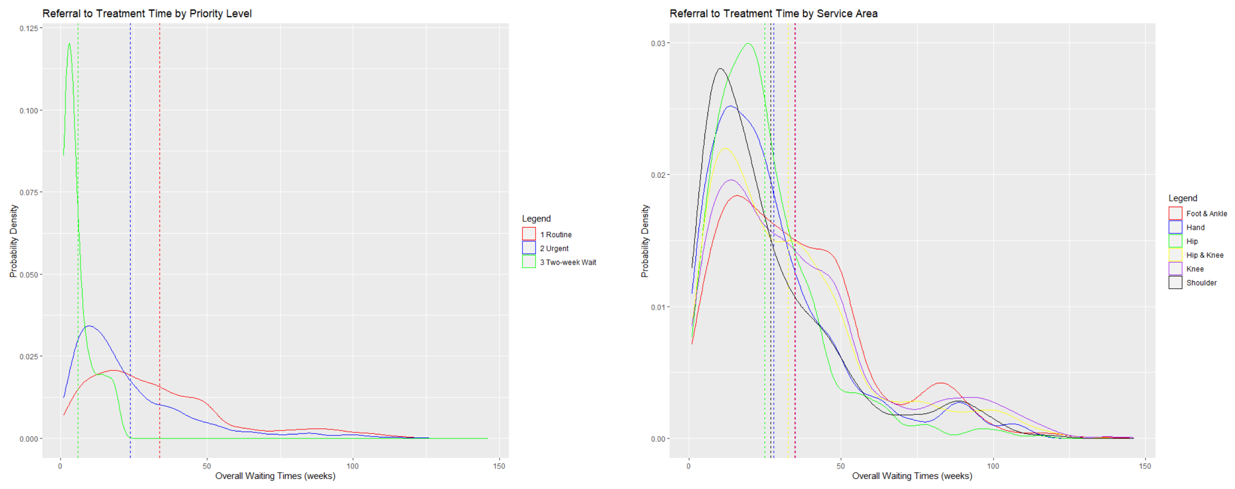


Figure 2.2: Referral to Treatment Times by Priority Level and Service Area

In the above, we see the RTT distribution curves vary in size and position based on the variable being examined. The categories are seperated by colour and specified by the corresponding legend key on the right-hand-side of each plot. We clearly see some variability in expected waiting times across each variable. However, this approach makes the error of considering completed referrals in a vacuum. Whereas, one may consider the present size and shape of the existing referral backlog to further refine these estimates. In practice, a new incoming patient who is referred "today" could experience a very different overall RTT waiting time from another patient referred several months ago, even for the same procedure and priority level. As a result, ignoring the sate of T&O's current backlogged waiting list seems a poor idea. Moreover, though this approach works well for overviews and summaries, it does not investigate deeper trends as well as with other predictive approaches.

2.2.2 Non-linear Regression Analysis (GLMs)

This approach would use GLM packages in R to predict the value of the variable "rtt_weeks_wait" using some combination of the other columns as explanatory variables. Building such models in R is fairly straigh-forward:

```
model.lm <- lm(rtt_weeks_wait ~ ServiceArea + priority_type_code,
               data = clock_stops_processed)
summary(model.lm)
anova(model.lm)
```

However, this approach suffers from some limitations. Firstly, it does not engage well with large amounts of the data provided, except for use as lookup tables. Secondly, other columns in the clock-stops data-set, in practice, lacked predictive power for "rtt_weeks_wait". Consequently, the model performed poorly in initial testing (high mean-squared error). Thirdly, this approach ignored the current state of T&O's backlog waiting list as specified by the incompletes data-set. Lastly, large numbers of the columns are categorical (label based) instead of continuous (numerical). As a consequence, regression modelling becomes much less suitable. Though not a fundamental limitation (it is not impossible), we are prevented from directly engagement for our prediction.

2.2.3 Nested Multivariate Queue System Design

This approach would use known results from queueing theory and probability theory to determine more directly "how long" until a given patient is seen. We do so by means of modelling the UHS T&O department using a nested queuing model (Angelo et al. [2017]). We then make changes to elementary queueing theory results (Shortle et al. [2004]) in order to provide a more nuanced simulation. The wider system considers the six service areas offered by T&O as six queues operating in parallel (Siqueiraa et al. [2022]):

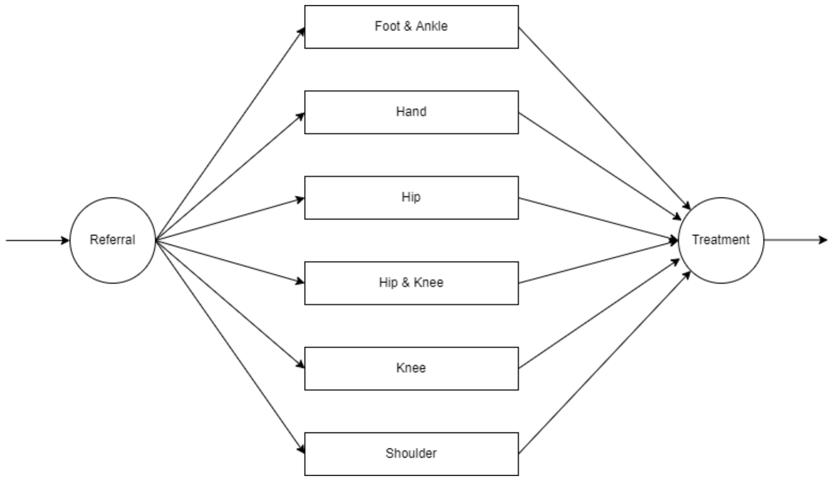


Figure 2.3: Parallel Queueing System Model

We consider this parallel system of independent queues an appropriate model due to the fact that each service area is serve only by consultants specialising in that area, using different resources. As a result, though there exists dependence between all aspects of hospital operation, different service areas can be modelled well as running independently of one another. Within each of these subqueues, we then track the current state with the following Markov Chain:

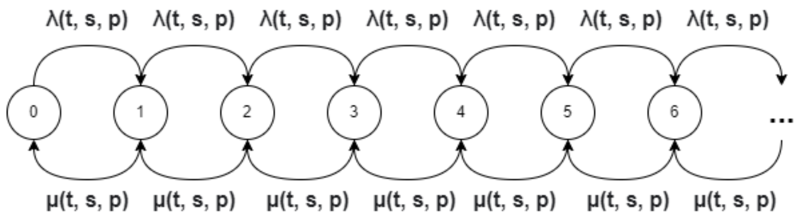


Figure 2.4: Sub-Queue Current State Markov Chain

Within this Markov Chain, we introduce the following notation:

- t is the variable for time within our simulation, specifying the point in the forecast we read values from
- s is the variable for Service Area within our simulation
- p is the variable for Priority Level within our simulation
- λ is the queue growth parameter, representing incoming referrals. We allow this to be variable, whose value depends on the value read at time t from the forecast corresponding to service area s and priority level p.
- μ is the queue reduction parameter, representing outgoing treatments. We allow this to be variable, whose value depends on the value read at time t from the forecast corresponding to service area s and priority level p.

Consequently, within our model, we extend more simplistic queuing theory models by changing the queue-growth and queue-shrink parameters into functions of three variables (time, speciality and priority), and allowing "queue-jumping" for higher priority patients to reflect the patient priority algorithm used by T&O and the NHS more broadly (see Rathnayake et al. [2021]). Lastly, we consider the following key pieces of information about how this queuing system behaves:

- Knowledge of how the existing queue is currently handled by T&O internally
- Forecasting the rate of incoming referrals for higher priority patients who may "jump the queue" ahead of a given patient
- Forecasting the rate of outgoing treatments for patients across priority levels but within a given service area.

Two variations on this approach were considered which we give the following names:

1. Many Worlds Variation
2. Forecast Value Variation

In the many worlds variation, we would account for the uncertainty in treatment rate by considering alternate scenarios in parallel, within which different numbers of people are treated in a given week. From here, one grows a multiverse "tree" of all possible scenarios in which the queue changes in different ways. Then, a clear time distribution is generated by considering the likelihood of each scenario and the state of each scenario. The personalisation by service-area is handled at set-up, and the priority personalisation is observed in real-time via priority allocation. This provides the personalised estimates.

In the forecast value variation, we take a more direct approach that would simplify the stochastic element by reading mean predicted values from the forecast and considering this one and only scenario. The stochastic uncertainty is then re-implemented at a higher level by modelling the error produced by the forecasts. This error is then modelled to fit a sampling distribution. For programming reasons, these distributions are converted to "Cumulative Distribution Functions" (CDF's) to enable pre-computing of these functions. These functions are then samples from using "Inverse Transformation Sampling". Finally, we run N total simulations ($N = 100$ and $N = 1000$ were tested), sampling randomly from the error distributions each time. Then, we produce a clear-time distribution from across these N simulations. This produces the same shape / form of result as approach the many-worlds approach, providing the personalised estimates.

Due to our limited project scope, and a desire to keep this design more tractable, we decide to employ the forecast-value variation of the Nested Multivariate Queue System Design.

With the above in mind, we may begin to construct a queuing system model of T&O, whose parameters are variable and informed based on historical data. We then consider estimating the queue clear time using known queuing theory survival analysis techniques. Next, using residual error distributions from the forecasts, we construct error distributions to be sampled from randomly when extracting forecast values. Lastly, with repeated simulation of our queuing system, we construct a queue clear time distribution for a given service area and given priority level. This would then inform a patient about the expected time required for all existing patients "ahead of them" in the queue to be treated (including higher priority patients who are always treated first). This queue clear time distribution will then act as an expected RTT wait time distribution for the patient. This in turn provides a waiting time estimate that is personalised across three factors: priority, service area and the present queue of backlogged referrals, while reflecting the uncertainty present in the prediction.

2.3 Visualising Existing Backlogged Referral Waiting List

The waiting list across the T&O ward can be split up across the 3 priority levels and 6 service areas into 18 sub-queues we may consider. We then arrange these 18 sub-queues in the following way to form our queuing model:

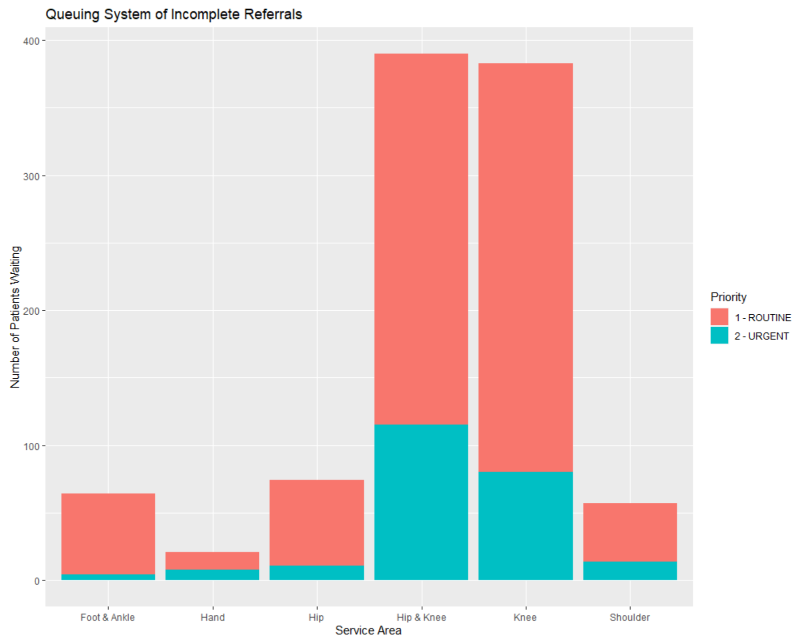


Figure 2.5: Incompletes Backlogged Referrals Bar-Chart

2.4 Forecasting Queue Arrivals and Exits

To accurately parameterise our above queuing system, we must first determine the values for λ and μ for each value of s and p. Since we have six total service areas and three priority levels, we would need thirty-six total forecasts. However, in practice, we do not necessarily need to fully parameterise our queuing system in order to produce waiting time estimates. This is for several reasons. Firstly, priority "1 - Routine" patients will never jump the queue ahead of anyone. As a result, to determine the waiting time for an existing patient of any priority level, we never need to consider any growth of the priority "1 - Routine" sub-queue. Consequently, forecasting incoming referrals for this priority level is entirely unnecessary. Secondly, we are able to aggregate all instances of μ across priority levels and forecast them as one. We then, within the simulation, distribute this forecasted availability according to priority-allocation.

This better reflects the real-world operations of UHS itself, and still allows us to personalise outgoing treatment rates by priority level. For clear notation, we will refer to such forecasts as “Priority ALL”. Lastly, in practice, priority “3 - Urgent” patients are very uncommon. In particular, within the 8143 clock-stops records, we see the following:

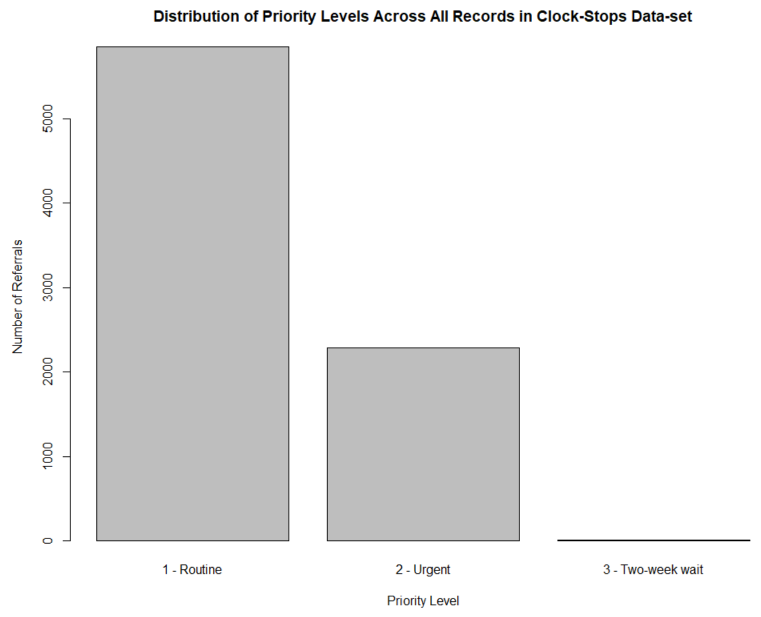


Figure 2.6: Ratio of Referrals between Different Priority Levels

As we can see in the above, we have 5846 priority 1 referrals, 2284 priority 2 referrals and 13 priority 3 referrals across all clock-stops records. Forecasting any kind of trend from so few data-points across our seven-year window is simply too random, as the time-series itself will be largely zero except for small occasional spikes. In turn, the forecasts produced will too be largely zero. Consequently, we treat the priority 3 referrals as statistical noise and ignore them. A prediction produced for a new priority 3 patient will need to instead take a summary from the purely statistical approach described previously.

In summary, we may parameterise our queuing model sufficiently with 2 forecasts for each service area (priority “2 - Urgent” incoming referrals and priority “ALL” outgoing treatments). By then considering priority “3 - Two-week wait” as statistical noise, we then generate queue clear times using the initial state of the queue and known behaviour of the system more broadly.

Chapter 3

Methodology

3.1 Building Software Tool

The software tool was built using the R programming language within the RStudio development environment. This was done for both ease of statistical computing and ease of extension by other developers post-handover to UHS. We separate the functionality of our model into four files:

- FINAL_full_data_pipeline.R, which takes the raw inputs, cleans the data and generates all our time-series
- generate_cdfs.R, which takes our time-series, forecasts based on these, and generates a number of CDF functions representing the distributions of uncertainty present in each forecast. We then sample from these distributions in the simulation to better reflect the uncertainty of the real-world system.
- predictive_tool.R, which takes all of the above, and runs N simulations of our queuing system model, generating our output files.
- presenting_results.R, which takes out output files, generates the plots and summaries used in the conclusion of this report.

3.1.1 Software Design

The software tool currently arranges all relevant files in the specified way:

```
Pack
├── AllRawData (truncated)
├── CDFLookups (truncated)
├── N100Results (truncated)
├── N1000Results (truncated)
└── ProcessedData (truncated)
    ├── TimeSeries (truncated)
    └── 20220626_Incompletes_anon.csv
```

```
|
|    |
|    |__ clock_stops_processed.csv
|    |__ Clock_stops_TandO_2018_Anon.csv
|    |__ Emergency_TandO_data_for_Cormsis.csv
|    |__ EPR_Priority_1.csv
|    |__ EPR_Priority_2.csv
|    |__ EPR_Priority_3.csv
|    |__ incompletes_processed.csv
|    |__ referrals_by_service.csv
|    |__ staff_names_and_codes.csv
|__ ResultsSummary (truncated)
|__ FINAL_full_data_pipeline.R
|__ generate_cdfs.R
|__ predictive_tool.R
|__ presenting_results.R
|__ README.txt
```

The above structure helps us to organise all our programming code, data-sets, time-series files, CDF files and results files. The pack as provided gives this structure, and the code itself assumes this structure to find the files it requires. To set to one's local file-system, all one must do it edit all calls to "setwd()" inside the four RStudio workbooks. The full specification of this file-system is available in appendix C.

3.1.2 Forecasting in R

The NHS commonly forecasts in order to predict incoming patient rates (see Shah et al. [2021] and Eyles et al. [2022]). We consider the techniques used in these sources and others available in R to forecast using the time-series we have generated.

The forecasting technique we decide to use is TBATS forecasting, which utilises the following features in generating predictions:

- Trigonometric seasonality
- Box-Cox transformation
- ARIMA errors
- Trend components
- Seasonal components

Other methods of forecasting considered are:

- Naive forecasting
- Exponential Smoothing
- Holt-Winters
- ARIMA

In testing within R, we see the following results for forecasting priority "1 - Routine" in service area "Foot & Ankle":

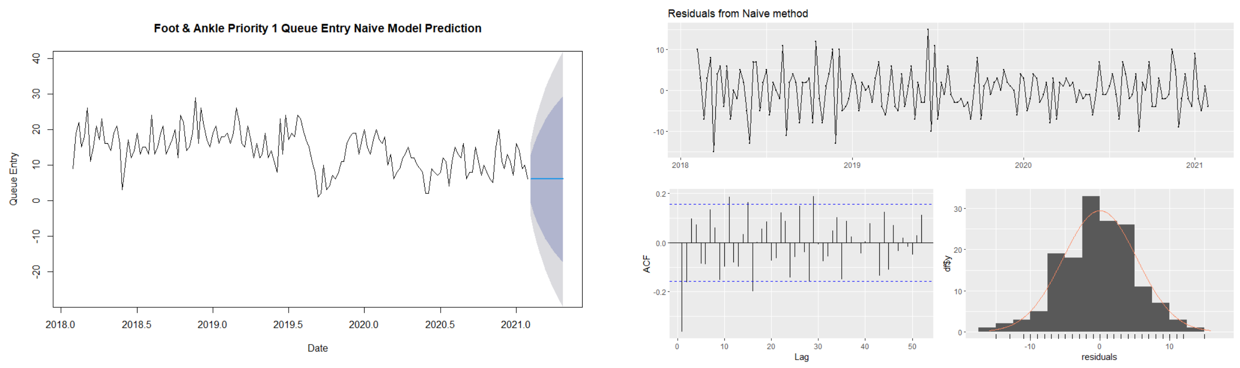


Figure 3.1: Forecasting with Naive Approach

In the above, we see the forecast predictions and residuals plot for the "Priority 1 Foot & Ankle" time-series, using the naive forecasting approach. In particular, we see very large amounts of uncertainty in predictions, which makes this approach seem a poor fit. Though, residuals are approximately normally distributed, which suggests it is an unbiased fit. Moreover, the residual time-series seems stable.

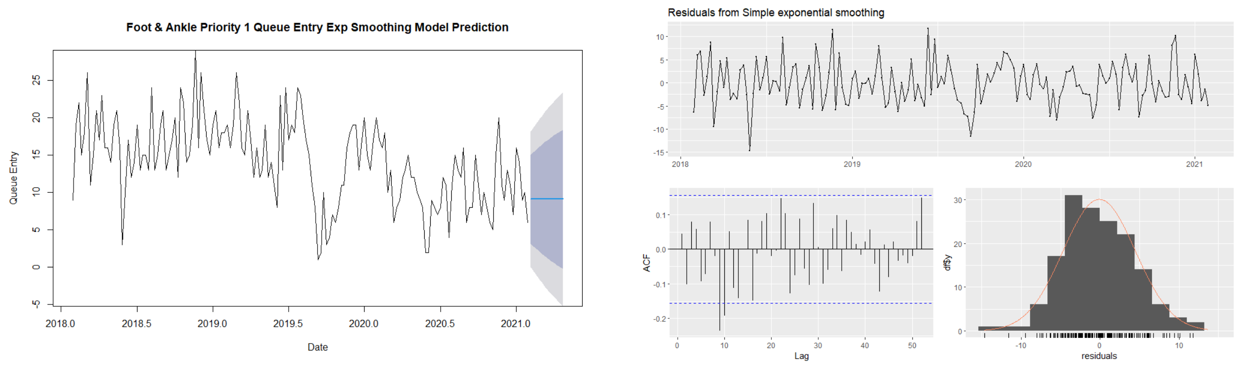
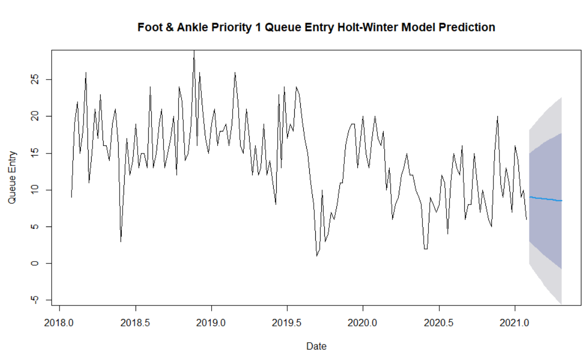


Figure 3.2: Forecasting with Exponential Smoothing Approach

In the above, we see the forecast predictions and residuals plot for the same "Priority 1 Foot & Ankle" time-series, using the exponential smoothing forecasting approach. In particular, we see less uncertainty in predictions (looking closely at the y-axis scale), though still rather poor fit for the data. Residuals are approximately normally distributed, but biased slightly towards the negative. The residual time-series seems stable.



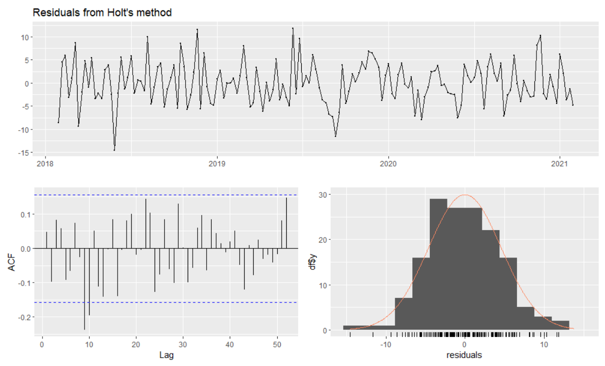
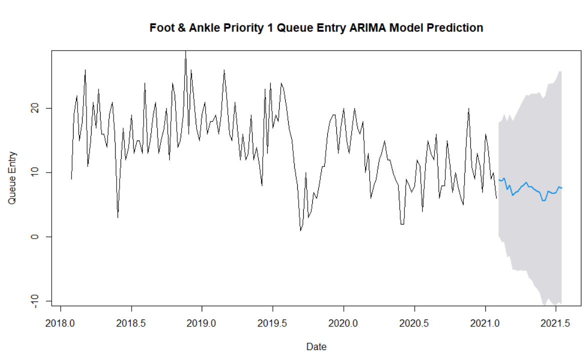


Figure 3.3: Forecasting with Holt-Winters Approach

In the above, we see the forecast predictions and residuals plot for the same "Priority 1 Foot & Ankle" time-series, using the Holt-Winters forecasting approach. In particular, we see less uncertainty in predictions than either previous (looking closely at the y-axis scale), though a slightly better fit than previous, having captured an overall trend of decrease in the time-series data. Residuals are approximately normally distributed and unbiased.



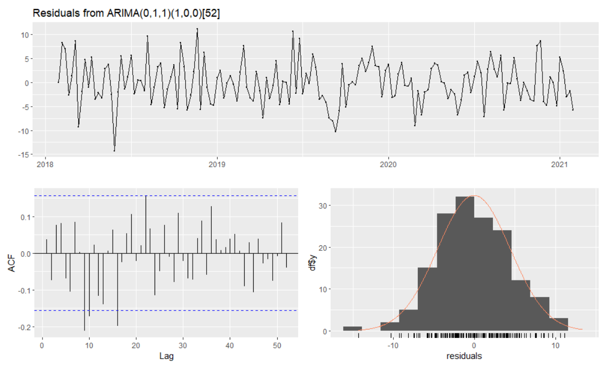


Figure 3.4: Forecasting with ARIMA Approach

In the above, we see the forecast predictions and residuals plot for the same "Priority 1 Foot & Ankle" time-series, using the ARIMA forecasting approach. In particular, we see similar uncertainty in predictions to previous (looking closely at the y-axis scale). Though, we also see a better fit as the forecast has captured some trend and variability. Residuals are approximately normally distributed and clearly unbiased.

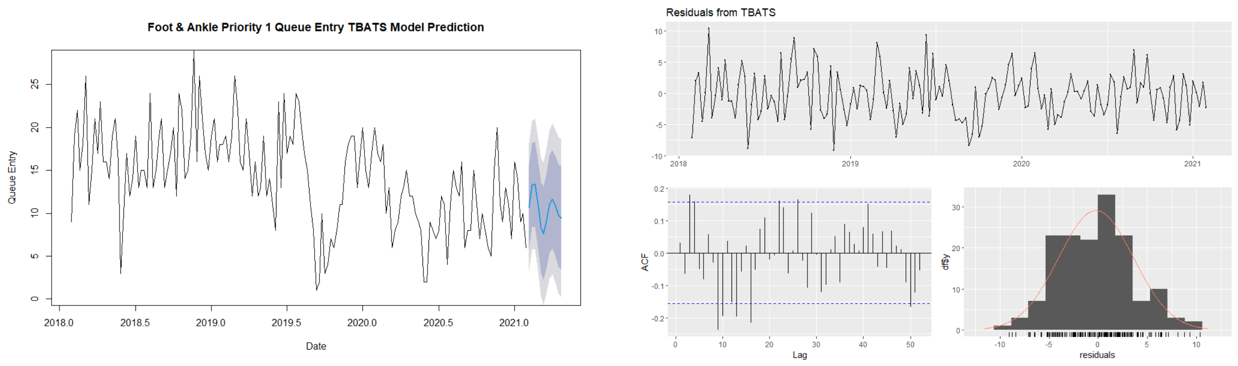


Figure 3.5: Forecasting with TBATS Approach

In the above, we see the forecast predictions and residuals plot for the same “Priority 1 Foot & Ankle” time-series, using the TBATS forecasting approach. In particular, we see less uncertainty in predictions than previous (looking closely at the y-axis scale). Additionally, we also see a better fit as the forecast has captured some trend, seasonality and variability. Residuals are approximately normally distributed, though slightly biased towards the positive. Forecasts for priority levels “1 - Routine” and “2 - Urgent” were similar for other Service Areas. As a result, we elected to use TBATS forecasting to produce the parameters of our queuing-system.

However, we ran into an issue attempting to forecast the priority level “3 - Two-week wait”. In particular, we see the following:

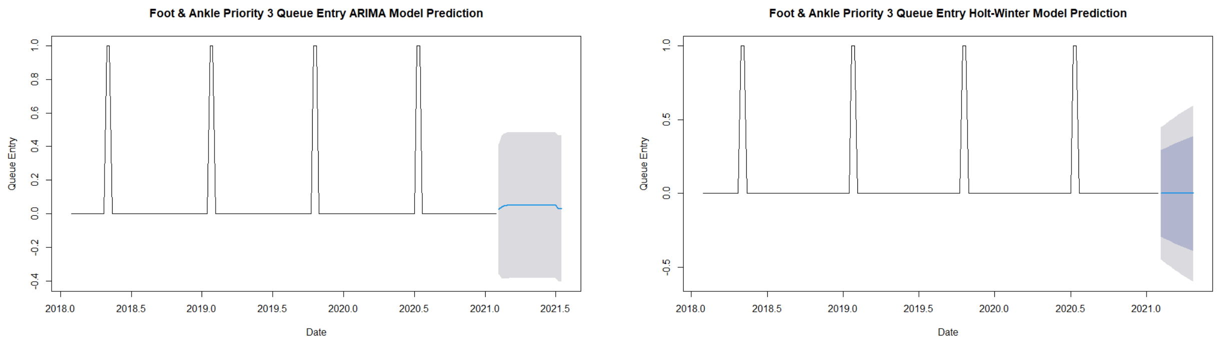
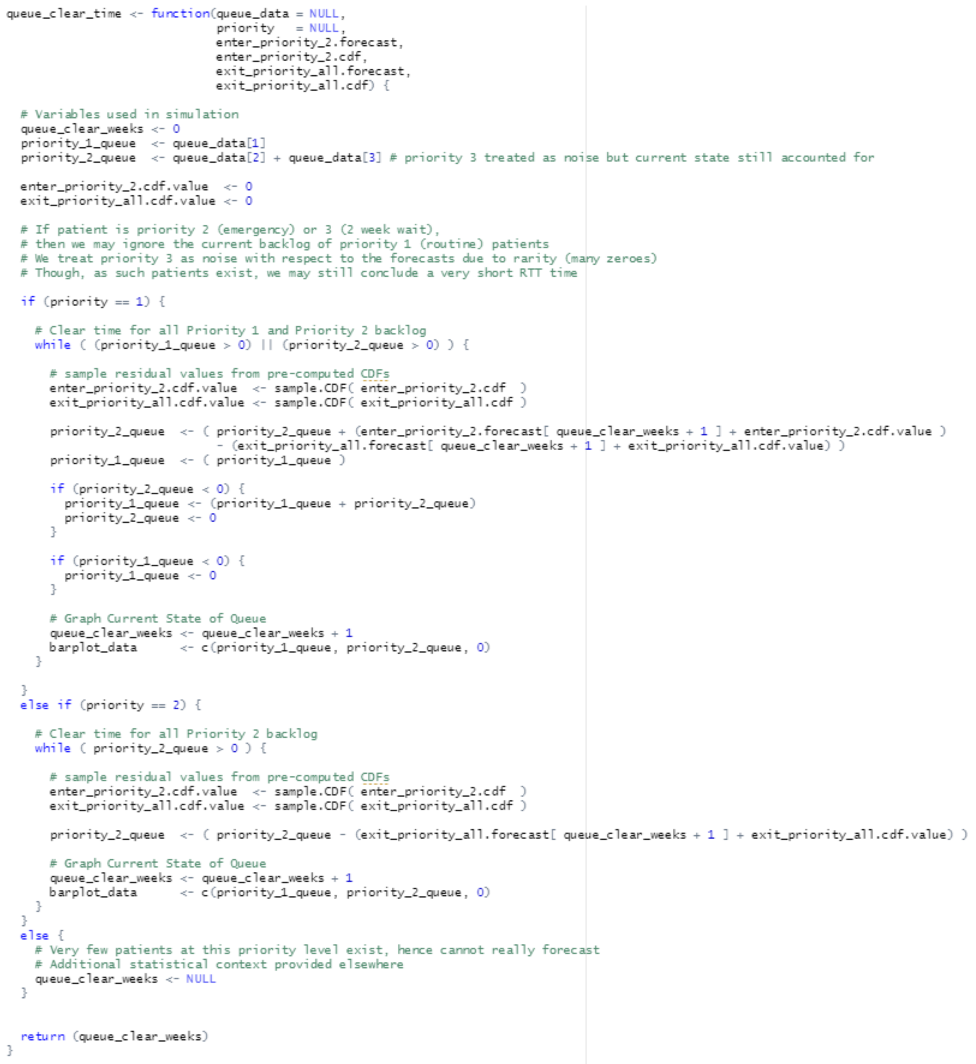


Figure 3.6: Attempted Forecasting of Priority 3 Patients

As we see in the above figure, the time-series is largely zero as mentioned. Forecasts for other service areas at this priority level were similar. Consequently, as feared earlier, we are unable to accurately forecast incoming referrals for priority 3 patients. As a result, for priority 3 patients, we default to the purely statistical method for a more general estimate that personalises by priority and service area, but not by current backlog queue state.

3.1.3 Queue Clear Time Estimation in R

We implement the queue clear-time estimation algorithm in R in the following way:



```r
queue_clear_time <- function(queue_data = NULL,
                             priority   = NULL,
                             enter_priority_2.forecast,
                             enter_priority_2.cdf,
                             exit_priority_all.forecast,
                             exit_priority_all.cdf) {

  # Variables used in simulation
  queue_clear_weeks <- 0
  priority_1_queue  <- queue_data[1]
  priority_2_queue  <- queue_data[2] + queue_data[3] # priority 3 treated as noise but current state still accounted for

  enter_priority_2.cdf.value  <- 0
  exit_priority_all.cdf.value <- 0

  # If patient is priority 2 (emergency) or 3 (2 week wait),
  # then we may ignore the current backlog of priority 1 (routine) patients
  # We treat priority 3 as noise with respect to the forecasts due to rarity (many zeroes)
  # Though, as such patients exist, we may still conclude a very short RTT time

  if (priority == 1) {

    # Clear time for all Priority 1 and Priority 2 backlog
    while ( (priority_1_queue > 0) || (priority_2_queue > 0) ) {

      # sample residual values from pre-computed CDFs
      enter_priority_2.cdf.value  <- sample.CDF( enter_priority_2.cdf  )
      exit_priority_all.cdf.value <- sample.CDF( exit_priority_all.cdf )

      priority_2_queue  <- ( priority_2_queue + (enter_priority_2.forecast[ queue_clear_weeks + 1 ] + enter_priority_2.cdf.value )
                           - (exit_priority_all.forecast[ queue_clear_weeks + 1 ] + exit_priority_all.cdf.value) )
      priority_1_queue  <- ( priority_1_queue )

      if (priority_2_queue < 0) {
        priority_1_queue <- (priority_1_queue + priority_2_queue)
        priority_2_queue <- 0
      }

      if (priority_1_queue < 0) {
        priority_1_queue <- 0
      }

      # Graph Current State of Queue
      queue_clear_weeks <- queue_clear_weeks + 1
      barplot_data      <- c(priority_1_queue, priority_2_queue, 0)
    }

  }
  else if (priority == 2) {

    # Clear time for all Priority 2 backlog
    while ( priority_2_queue > 0 ) {

      # sample residual values from pre-computed CDFs
      enter_priority_2.cdf.value  <- sample.CDF( enter_priority_2.cdf  )
      exit_priority_all.cdf.value <- sample.CDF( exit_priority_all.cdf )

      priority_2_queue  <- ( priority_2_queue - (exit_priority_all.forecast[ queue_clear_weeks + 1 ] + exit_priority_all.cdf.value) )

      # Graph Current State of Queue
      queue_clear_weeks <- queue_clear_weeks + 1
      barplot_data      <- c(priority_1_queue, priority_2_queue, 0)
    }
  }
  else {
    # Very few patients at this priority level exist, hence cannot really forecast
    # Additional statistical context provided elsewhere
    queue_clear_weeks <- NULL
  }


  return (queue_clear_weeks)
}
```

As we see in this algorithm, we treat each priority level as a separate case:

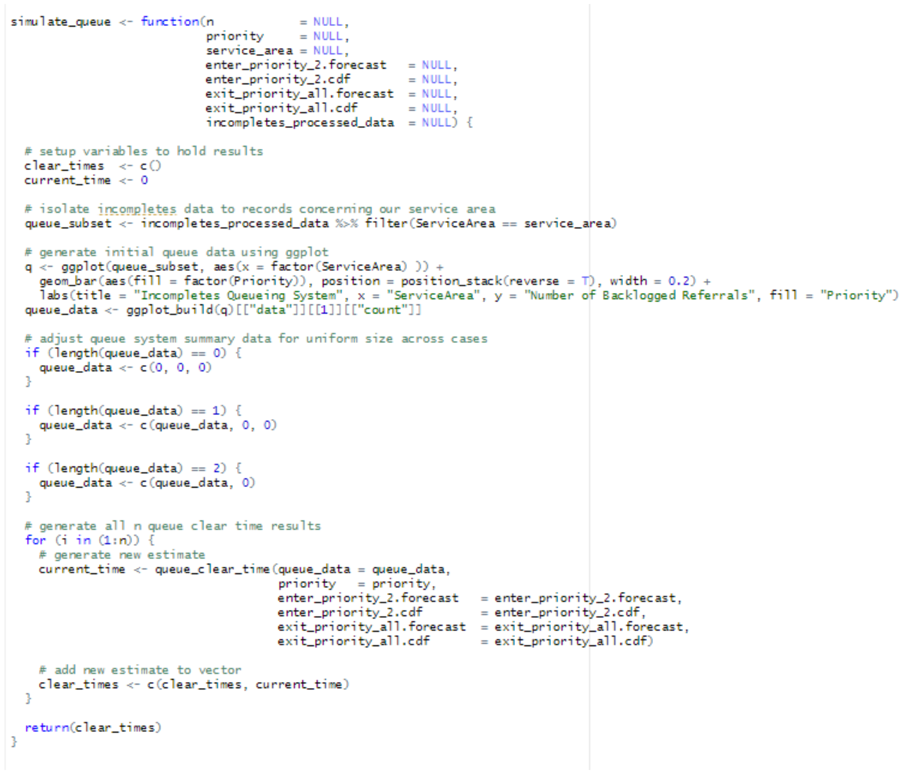
1. For priority "1 - Routine" patients, we calculate the clear time for all existing patients in the queuing system across all priority levels. Additionally, we allow for new referrals at priority level 2 to jump the queue, priority 3 arrivals are assumed zero due to the time-series. We then iterate week-by-week, tracking the state of the queue each week until the priority 1 queue has zero (or negative) length. At such a point, we know our new priority 1 patient will be seen. This queue clear time estimate forms our expected waiting time estimate.
2. For priority "2 - Urgent" patients, we calculate the clear time for all existing patients in the queuing system across priority levels 2 and 3. Additionally, we do not allow for

new referrals at priority level 2 to jump the queue, priority 3 arrivals are assumed zero due to the time-series. We then proceed week-by-week as previously to determine the queue-system clear time.

3. For priority “3 - Two-week wait” patients, we instead opt for other approaches to convey expected waiting time to such patients, as forecasting future emergency referrals at this priority level is infeasible.

The stochastic element comes from the repeated sampling from our forecast error distributions at each iteration. The idea being that we reflect the uncertainty present in the predictions by fitting the residuals of the forecast to a probability distribution. Then, when extracting any values from the forecast to determine how many patients enter or exit in a given week, we also sample randomly from the residual error distribution. This will cause the final queue clear-time estimate to vary with this uncertainty. We then return this resultant queue clear-time value for use elsewhere.

Additionally, we implement the repeated simulation of our queuing system in the following way:



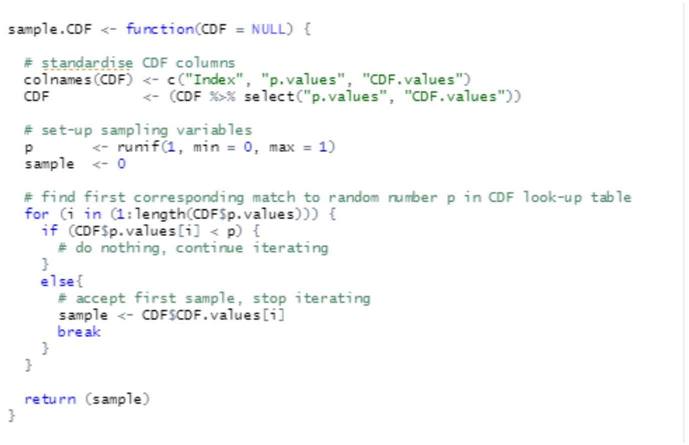
```
simulate_queue <- function(n            = NULL,
                           priority     = NULL,
                           service_area = NULL,
                           enter_priority_2.forecast   = NULL,
                           enter_priority_2.cdf        = NULL,
                           exit_priority_all.forecast  = NULL,
                           exit_priority_all.cdf       = NULL,
                           incompletes_processed_data  = NULL) {

  # setup variables to hold results
  clear_times  <- c()
  current_time <- 0

  # isolate incompletes data to records concerning our service area
  queue_subset <- incompletes_processed_data %>% filter(ServiceArea == service_area)

  # generate initial queue data using ggplot
  q <- ggplot(queue_subset, aes(x = factor(ServiceArea) )) +
    geom_bar(aes(fill = factor(Priority)), position = position_stack(reverse = T), width = 0.2) +
    labs(title = "Incompletes Queueing System", x = "ServiceArea", y = "Number of Backlogged Referrals", fill = "Priority")
  queue_data <- ggplot_build(q)[["data"]][[1]][["count"]]

  # adjust queue system summary data for uniform size across cases
  if (length(queue_data) == 0) {
    queue_data <- c(0, 0, 0)
  }

  if (length(queue_data) == 1) {
    queue_data <- c(queue_data, 0, 0)
  }

  if (length(queue_data) == 2) {
    queue_data <- c(queue_data, 0)
  }

  # generate all n queue clear time results
  for (i in (1:n)) {
    # generate new estimate
    current_time <- queue_clear_time(queue_data = queue_data,
                                     priority   = priority,
                                     enter_priority_2.forecast   = enter_priority_2.forecast,
                                     enter_priority_2.cdf        = enter_priority_2.cdf,
                                     exit_priority_all.forecast  = exit_priority_all.forecast,
                                     exit_priority_all.cdf       = exit_priority_all.cdf)

    # add new estimate to vector
    clear_times <- c(clear_times, current_time)
  }

  return(clear_times)
}
```

As we see in this algorithm, we first initialise the data for use in the “queue_clear_time” algorithm. In particular, we first subset the data to exclusively the service area specified in the parameters of the function. Next, we take advantage of the “ggplot2” package in order to transform (group by priority and count length) the incompletes data subset into a vector three numbers:

1. The number of priority “1 - Routine” patients currently queuing for the given service area.
2. The number of priority “2 - Urgent” patients currently queuing for the given service area.
3. The number of priority “3 - Two-week wait” patients currently queuing for the given service area.

We then, for computational reasons, ensure that the vector is always of the same length by appending zeroes to the end as necessary. Lastly, we run the “queue_clear_time” algorithm N times independently, storing the results of each iteration in a vector of size N. We then return this results vector for use elsewhere.

Lastly, within the "queue_clear_time" algorithm, we implement inverse transformation sampling to sample from the error distributions of our forecasts in the following way:



```r
sample.CDF <- function(CDF = NULL) {

  # standardise CDF columns
  colnames(CDF) <- c("Index", "p.values", "CDF.values")
  CDF           <- (CDF %>% select("p.values", "CDF.values"))

  # set-up sampling variables
  p       <- runif(1, min = 0, max = 1)
  sample  <- 0

  # find first corresponding match to random number p in CDF look-up table
  for (i in (1:length(CDF$p.values))) {
    if (CDF$p.values[i] < p) {
      # do nothing, continue iterating
    }
    else{
      # accept first sample, stop iterating
      sample <- CDF$CDF.values[i]
      break
    }
  }

  return (sample)
}
```

As we see in this algorithm, we have the given CDF pre-computed for use similarly to a lookup table. We begin, for computational reasons, by standardising the column names of this lookup table. Next, we generate a random number p between 0 and 1 (inclusive) from a uniform distribution. Then, we iterate through the "p.values" column of our CDF lookup to find the first instance of a column value being greater than or equal to our random value p. We then record this value as our sample and stop iterating, as we only wish to return the first match. Lastly, we return this random sample for use elsewhere.

3.2 Using Software Tool

Generating new estimates is fairly straightforward given the existing design. To do so, perform the following steps:

1. Ensure that the file-system is set up as specified as above, with new data-sheets replacing the existing data sheets inside of "AllRawData" and "ProcessedData" (the data does not need to be processed yet, this is simply the working directory we will use to do so). Please ensure that the file extension "CSV" is used for all files in "ProcessedData".
2. Run the R entire file "FINAL_full_data_pipeline.R" in RStudio using the command "Run All" of the keyboard shortcut "ctrl-shift-enter".
3. Run the R entire file "generate_cdfs.R" in RStudio using the command "Run All" of the keyboard shortcut "ctrl-shift-enter".
4. Run the R entire file "predictive_tool.R" in RStudio using the command "Run All" of the keyboard shortcut "ctrl-shift-enter".
5. Run the R entire file "presenting_results.R" in RStudio using the command "Run All" of the keyboard shortcut "ctrl-shift-enter".
6. The updated results will now be displayed in the command output window and the plot window of RStudio.

Chapter 4

Solution and Results

4.1 Waiting Time Estimates

Using the existing incompletes data snapshot provided by UHS, we may generate the following waiting time distribution estimates for patients in each of the following combinations of service area and priority level:

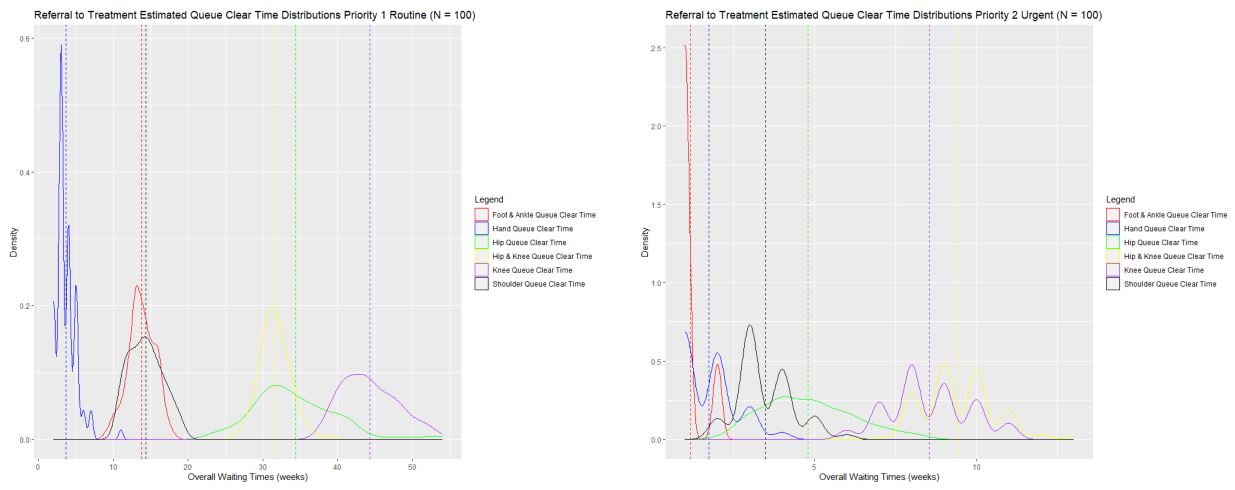


Figure 4.1: Final Results Queue Clear Time Distributions ($N = 100$)

In the above visualisations, we see the distributions of the predicted waiting time for a new incoming patient across N simulations of the queuing system. The left-side plot sets priority level “1 - Routine” and the right-side plot sets priority level “2 - Urgent”. Both plots use the same colours to represent each of the service areas: red for “Foot & Ankle”, blue for “Hand”, green for “Hip”, yellow for “Hip & Knee”, purple for “Knee” and black for “Shoulder”.

Looking at the left-side plot of the above figure, we see that the different service areas will see to patients in very different amounts of time. Service areas rank in the following order for average waiting times:

1. Priority 1 Patients for “Hand” are seen to in the shortest amount of time on average, with a mean wait of 3.69 weeks (median 3.0 weeks)
2. Priority 1 Patients for “Foot & Ankle” are seen to in the second shortest amount of time on average, with a mean wait of 13.77 weeks (median 14.0 weeks)
3. Priority 1 Patients for “Shoulder” are seen to in the third shortest amount of time on average, with a mean wait of 14.33 weeks (median 14.0 weeks)
4. Priority 1 Patients for “Hip & Knee” are seen to in the fourth shortest amount of time on average, with a mean wait of 31.64 weeks (median 31.5 weeks)
5. Priority 1 Patients for “Hip” are seen to in the fifth shortest amount of time on average, with a mean wait of 34.33 weeks (median 33.5 weeks)
6. Priority 1 Patients for “Knee” are seen to in the sixth shortest amount of time on average, with a mean wait of 44.34 weeks (median 44.0 weeks)

Additionally, looking at the left-side plot of the above figure, we see that the different service areas will also face very different variability in waiting times. Service areas rank in the following order for uncertainty in final waiting times:

1. Priority 1 Patients for “Hand” were seen with most consistent waiting times, with an inter-quartile range of 1 week
2. Priority 1 Patients for “Foot & Ankle” were seen with the second most consistent waiting times, with an inter-quartile range of 2 weeks
3. Priority 1 Patients for “Hip & Knee” were seen with the third most consistent waiting times, with an inter-quartile range of 3 weeks
4. Priority 1 Patients for “Shouder” were seen with the fourth most consistent waiting times, with an inter-quartile range of 4 weeks
5. Priority 1 Patients for “Knee” were seen with the fifth most consistent waiting times, with an inter-quartile range of 6 weeks
6. Priority 1 Patients for “Hip” were seen with the sixth most consistent waiting times, with an inter-quartile range of 6.25 weeks

Based on these findings, we see that service area directly affects both the average waiting time for a given patient and the uncertainty surrounding waiting time, with slightly different rankings for shortest waiting times and most consistent waiting times.

Looking at the right-hand plot in the above figure, looking closely at the x-axis values, we see that waiting times are very significantly shorter across the entire distribution for each service area. Additionally, we see that the rankings change for shortest average waiting times:

1. Priority 2 Patients for “Foot & Ankle” are seen to in the shortest amount of time on average, with a mean wait of 1.16 weeks (median 1.0 weeks)
2. Priority 2 Patients for “Hand” are seen to in the second shortest amount of time on average, with a mean wait of 1.74 weeks (median 2.0 weeks)
3. Priority 2 Patients for “Shoulder” are seen to in the third shortest amount of time on average, with a mean wait of 3.47 weeks (median 3.0 weeks)
4. Priority 2 Patients for “Hip” are seen to in the fourth shortest amount of time on average, with a mean wait of 4.8 weeks (median 5.0 weeks)
5. Priority 2 Patients for “Knee” are seen to in the fifth shortest amount of time on average, with a mean wait of 8.55 weeks (median 8.0 weeks)
6. Priority 2 Patients for “Hip & Knee” are seen to in the sixth shortest amount of time on average, with a mean wait of 9.44 weeks (median 9.0 weeks)

Moreover, we see the following change to the rankings of uncertainty in final waiting times:

1. Priority 2 Patients for “Foot & Ankle” were seen with the most consistent waiting times, with an inter-quartile range of 1 week

2. Priority 2 Patients for “Hand”, “Hip & Knee”, “Knee” and “Shoulder” were all seen with the second most consistent waining times, with an inter-quartile range of 2 weeks
3. Priority 2 Patients for “Hip” were seen with the third most consistent waiting times, with an inter-quartile range of 3 weeks

From the above, we see that not only were waiting times much shorter on average, but also much more consistent across every service area. We also see mild changes in the way that service area affects these two aspects of the waiting time distributions.

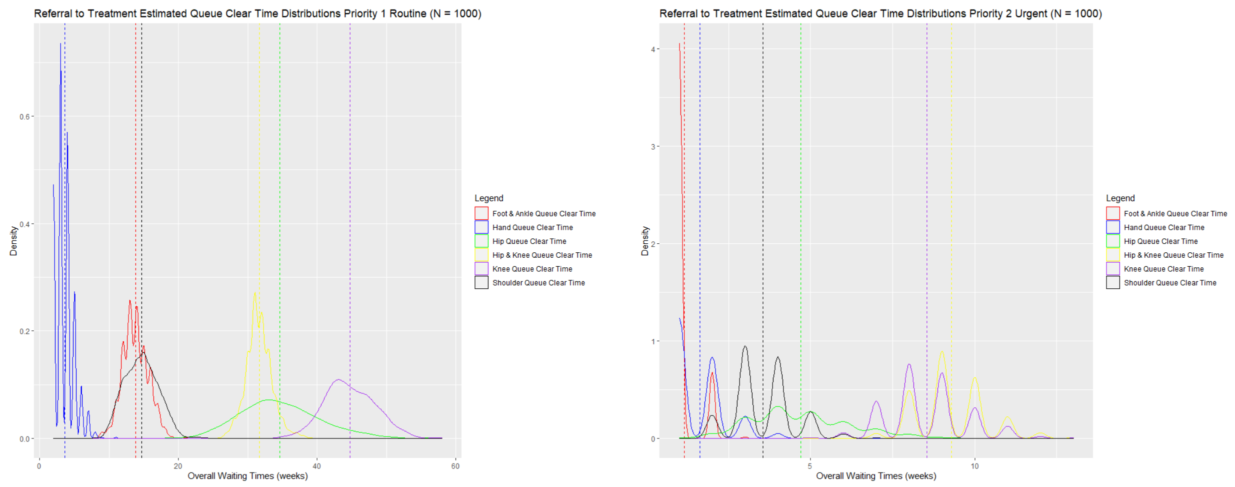


Figure 4.2: Final Results Queue Clear Time Distributions ($N = 1000$)

Comparing figures 4.1 and 4.2, we see the same pattern bare out. As a result, we conclude that our rankings hold consistently across large values of N. Based on this, we draw four central conclusions regarding research question one:

1. The Priority Level of a patient very directly affects the expected waiting time of a given patient
2. The Service Area of a patient also very directly affects the expected waiting time of a given patient
3. The Priority Level of a patient very directly affects the uncertainty around expected waiting time of a given patient
4. The Service Area of a patient also very directly affects the uncertainty around expected waiting time of a given patient

4.2 Note on Reproduction of Results

In accordance with ethics guidance and UHS policy, an NDA was signed by the author and supervisors prior to the project being started. In particular, this NDA strictly prohibits the keeping of any data related to the project beyond the project completion date (Thursday September 15^{th} 2022). Moreover, it also prohibited to provide the data used in generating our results with this paper. This makes reproducing the provided results difficult.

To do so, please perform the following steps:

1. Contact UHS Trauma and Orthopaedics to request access to three key data-sets:
   (i) Historical Clock Stops Data across a sufficiently large window
   (ii) Current Incompletes Data snapshot taken from the present moment
   (iii) Relevant service area look-up tables
2. Input the three data-sets into the RStudio workbook pack as per the file-system specified in appendix C
3. Proceed as per section 3.2 to generate new results

Having done so, the estimates may look different as the snapshots were taken at different times. In particular, the snapshot used by the author was generated by T&O on Sunday June 26^{th} 2022. Though, historical time series will look similar and estimates will likely continue following the same general patterns. This is not a bug, but a feature, as the snapshot date is itself a predictor we utilise implicitly in our queuing model simulation. We consider the RTT start date as itself a key predictor in final RTT waiting times. Intuitively, it makes sense that the current state of the backlogged existing referral queue would affect the expected waiting time of a new patient joining the queue - the date of referral directly affects the expected waiting time for a given patient as the initial state of their queue depends on when they join.

Chapter 5

Conclusions

5.1 Summary of Key Results

In response to research question one, the answer is yes - we can personalise a waiting time estimate beyond a population-wide estimate. In particular, we make five central findings from the repeated simulation of our model:

1. The waiting times are significantly lower for higher-priority patients.
2. The waiting times are significantly more consistent (less uncertainty present) for higher priority patients.
3. The waiting time length is directly impacted by service area required and patient priority level jointly, ranking overall in the following order:
    - (i) Priority 2 Foot & Ankle (mean 1.16 weeks, median 1.0 week)
    - (ii) Priority 2 Hand (mean 1.74 weeks, median 2.0 weeks)
    - (iii) Priority 2 Shoulder (mean 3.47 weeks, median 3.0 weeks)
    - (iv) Priority 1 Hand (mean 3.69 weeks, median 3.0 weeks)
    - (v) Priority 2 Hip (mean 4.8 weeks, median 5.0 weeks)
    - (vi) Priority 2 Knee (mean 8.55 weeks, median 8.0 weeks)
    - (vii) Priority 2 Hip & Knee (mean 9.44 weeks, median 9.0 weeks)
    - (viii) Priority 1 Foot & Ankle (mean 13.77 weeks, median 14.0 weeks)
    - (ix) Priority 1 Shoulder (mean 14.33 weeks, median 14.0 weeks)
    - (x) Priority 1 Hip & Knee (mean 31.64 weeks, median 31.5 weeks)
    - (xi) Priority 1 Hip (mean 34.33 weeks, median 33.5 weeks)
    - (xii) Priority 1 Knee (mean 44.34 weeks, median 44.0 weeks)

4. The waiting time consistency is directly affected by service area required and patient priority level jointly, ranking overall in the following order:
    (i) Priority 2 Foot & Ankle (IQR 0 weeks)
    (ii) Priority 2 Hand, Priority 2 Shoulder, Priority 1 Hand, Priority 2 Knee and Priority 2 Hip & Knee (IQR 1 week)
    (iii) Priority 1 Foot & Ankle and Priority 2 Hip (IQR 2 weeks)
    (iv) Priority 1 Hip & Knee (IQR 3 weeks)
    (v) Priority 1 Shoulder (IQR 4 weeks)
    (vi) Priority 1 Knee (IQR 6 weeks)
    (vii) Priority 1 Hip (IQR 6.25 weeks)
5. The waiting time length is directly impacted by the start date of patient referral

5.2 Model Specification

The model produced for UHS Trauma and Orthopaedics is a Nested Multivariate Parallel Queuing Model, whose parameters are determined by historical referral record data and whose initial state is determined by current backlogged referral data. Resulting queue clear-time distributions are then generated from this model with repeated simulation. These results then seek to inform a given clinician about the specifics of a given patient's expected waiting time by considering patient service area, patient priority level, current backlog of existing referrals and historical waiting times. It is hoped that the clinician would then be able to convey the changes in expected waiting time and uncertainty around waiting time to a patient in order to put their mind at ease and improve patient well-being.

5.3 Important Correlation to Waiting Times

In considering research question two, we consider again the clusters of referral-record variables considered during variable selection:

- Referral-to-treatment Data (rtt_id, rtt_start, rtt_days, rtt_weeks, rtt_end, rtt_end_reason)
- Pathway Data (path_act_str-id, path_act_int_id, patient_path_id)
- Last Attendance Data (last_att_id, last_att_date, last_att_out)
- Internal Organisational Data (cons_code, spec_code, awl_id, excl_reasons, wait_list_type)
- Personal Data (age_today, gender_code, ethnic_cat)
- Referral Source Data (ref_req_date, ref_id, dec_to_admit_date, adm_method_code)
- External Organisational Data (org_code_iss, source_ref, org_id_prov, org_site_id)
- Future Appointment Data (out_future_app_date, due_date, out_app_date)
- DNA / Cancellation Data (last_dna_date, cancel_date)
- Priority Data (procedure_priority, diag_priority, date_last_priority_review_review)

From our analysis and model development, we found that variables most important to predicting the patient waiting time were the referral start date, priority level and service area were key predictors. Whereas, by contrast, personal variables, DNA variables, External Organisation variables, and so on had little predictive power. Moreover, many of these variables are categorical (label-based) as opposed to continuous (number-based) making them difficult to use in numerical predictive models.

Zooming out a little on our findings, we see that variables relating to hospital operating policy and patient requirement (not patient personal or otherwise) were the best predictors. From a perspective of cause-and-effect, we see that these are centrally determined by other outside factors that do not appear in the referral records:

- Resource limitations in the real-world (theatre time, available workforce, available beds, stock of medical supplies)
- How UHS, and the NHS more broadly, organise themselves and allocate appointments
- Staff scheduling within UHS
- Weekly rotas for which operating activites are carried out at which times

None of these factors will be present in patient records or in referral records, but will always directly affect expected waiting times. Hence, we claim the most important correlation to RTT waiting times is algorithmic policy of UHS, not necessarily any of the fields within personal patient data.

5.4 Recommendation to UHS

Based on the above, we recommend to UHS that we shift the nature of the investigation itself: from personalising individual waiting times, to refining population waiting times. We face massive uncertainty and variability in real-world waiting times within the clock stops data that are not caused necessarily by the factors we use to predict them such as patient prioritisation, service area or clinician. As a result, we move beyond a purely data-science based approach, instead switching to an algorithmic analysis of UHS staff allocation and treatment allocation based on more management-style sciences. This will produce less personal results, but will take better account of real-world causality. Moreover, adverse conditions such as outbreaks of disease, staffing shortages, resource shortages and patient cancellation become much more tractable to account for.

5.5 Limitations of Results

5.5.1 Length of Research Project

It is of course worth noting that the scope of this project was limited by the twelve-week duration. As a result, the depth of research conducted into existing healthcare modelling was limited. Moreover, a more in-depth simulation of T&O was infeasible given the deadline required.

5.5.2 Volatility of Waiting Times to Outside Factors

Real-world patient waiting times will always be severely impacted by outside factors that predictive models will likely be unable to account for, such as natural disasters, sudden outbreaks of disease, available funding for NHS healthcare, individual staff members falling ill, popular events taking place and so on. As a result, accurate forecasting of incoming referrals and outgoing treatments becomes very difficult. Consequently, predictive models become very difficult to develop, even with access to very large quantities of data.

5.5.3 Inherent Intractability of the Research Question

Within this project, though forecasting-based techniques work well and are commonly present within the literature review, there will always exist large amounts of uncertainty in attempting to predict the future (expected waiting time from now to treatment) even with large quantities of data. We have produced useful statistical summaries and provided an solid context for understanding and conveying the nature of the waiting-list itself. This will, presented well, achieve the deliverable of managing patient expectation, how long they are likely to wait for their treatment, improve patient anxiety and quality of life.

Despite this, questions as broad as "how long should I expect to wait until I can see a doctor?" are never going to boil down well to simply one number for one patient. The only way to account for the uncertainty present is to simply state predictions alongside measures of uncertainty using population-wide data until a prospective date to come in is formally booked.

5.5.4 Queue-clear algorithm not guaranteed to terminate

As currently designed, the model assumes broadly that the queuing system will allow a new patient joining the back of the queue to eventually be seen. However, in practice, this is not necessarily guaranteed as a queue whose growth parameter (λ) is greater than it's shrink parameter (μ) will always grow over time. As a result, we do not have a stable system under these conditions. Consequently, the queue will never terminate without this condition changing. If the priority "2 - Urgent" queue were to do so within our simulation, no priority "1 - Routine" would ever be seen, as new higher priority patients will jump the queue indefinitely. For UHS Trauma and Orthopaedics, this was not an issue, as the queue system is shrinking overall with time and priority "1 - Routine" patients are seen with reasonable waiting times. Though, this assumption is not guaranteed to hold, and as a result the algorithm would not work for a system whose higher-priority queue grows too steeply over time.

A real-world example of this happening for a healthcare system in the UK would be the NHS Gender Identity Clinic System, whose waiting times for treatment have become largely impossible to calculate as a result of this happening (see Tavistock and Portman [2022]). Moreover, the impacts of these extensive and uncertain waiting times on individual patients and their outcomes is devastating (see Henderson et al. [2022]).

Chapter 6

Further Research Development

6.1 More in-depth Modelling

With a longer project timeline and more in-depth access to UHS internals, more options open up with the investigation to build a more in-depth model of UHS:

- Expanding the form of the model proposed in section 2.3.2. In particular, forecasting separately the many reasons a patient may exit the queue (treatment, death, cancellation, transfer to outgoing status and so on).
- Incorporating staff allocation algorithms, resource allocation and ward allocation to the simulation. This would massively increase the scope of the project but could provide more in-depth analysis and potential insight.
- More in-depth non-predictive modelling. We believe that modelling the change in the "historically backwards-constructed incompletes" as one moves through the clock-stops could gather insight into how the queue behaves over time. Moreover, foresight is certain and consequently we will not need to contend with as much uncertainty in this analysis.

6.2 Repeating Simulation with New Time-Window

The research and development for this project was carried out from early June 2022 until mid September 2022, using data gathered in late June 2022. This is relevant because UK health-care systems are largely still in recovery from the wider impacts of the COVID-19 pandemic which began affecting the NHS adversely in late February / early March 2020. Moreover, the historical data for RTT wait times was very strongly affected, with severe increases to RTT wait times during the pandemic time-window. Additionally, people were less likely to refer themselves to the GP in order to receive treatment during this time (see Duncan and Cheng [2021]). Further still, among people attempting to access treatment, the people of the UK are in some towns largely unable to access GP appointments (see Thorlby et al. [2019]). As a result, estimates generated using data from inside this pandemic time-window will look very different from estimates generated outside of it.

As a result, it would be worth repeating the simulations of T&O in several years time using updated data. In particular, a window of time broadly post-pandemic, which would likely yield very different results.

6.3 Re-considering Patient Priority Levels

During the pandemic, the NHS changed the manner in which it handles patient prioritisation. This was on account of the severe restrictions imposed by the pandemic meaning significantly lower capacity to see patients. As a result, further categorisation of referral urgency was required. In particular, where we currently look at patient priority on a three-point scale (Routine, Urgent and Two-week wait), two additional measures of prioritisation were introduced:

1. Clinical Prioritisation, a five-point scale:
   - (i) 1a - Emergency
   - (ii) 1b - Urgent - operation needed with 72 hours
   - (iii) 2 - Surgery that can be deferred for up to 4 weeks
   - (iv) 3 - Surgery that can be delayed for up to 3 months
   - (v) 4 - Surgery that can be delayed for more than 3 months
2. Patient Prioritisation (different from what we have previously referred to as "Patient Priority", which is referred to simply as "Priority" in the data)
   - (i) 6 - Delay due to other reason
   - (ii) 7 - Patient doesn't wish to delay
   - (iii) 8 - Patient asked but yet to respond
   - (iv) 9 - Long waiter re-added from monitoring / PIFU

Due to their recent addition, these priority categories are present in the incompletes data, but not in the historical clock-stops data. Consequently, we cannot at present use these to produce forecasts for a model. In the future, this will of course change if these categories continue to be used by UHS and more completed referral data accumulates. As a result, repeating the experiment a few years in the future with these two new priority categories incorporated into the model could produce more accurate predictions.

6.4 Additional Applications

Applications exist in other aspects of healthcare that respond more linearly with regards to their wait-lists and queueing systems. For example, one may consider UK Gender Identity Clinics (GICs) as a queuing system and do so. By considering services offered under a categorisation of "Service Area" in other healthcare systems, the Nested Multivariate Parallel Queueing Model could be easily re-applied to model other healthcare systems.

Bibliography

Simone A. Angelo, Edilson F. Arruda, Rosane Goldwasser, Maria S.C. Lobo, Andre Salles, and Jose Roberto Lapa e Silva. Demand forecast and optimal planning of intensive care unit (icu) capacity. *Pesquisa Operacional*, 37(2)(2):229–245, 2017. URL https://www.scielo.br/j/pope/a/qVMTvFv9mwfks44jvyc54jN/?lang=en. [Accessed June 24th 2022].

Lorna J. Duncan and Kelly F.D. Cheng. Public perception of nhs general practice during the first six months of the covid-19 pandemic in england. *National Library of Medicine*, 2021. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8749896/. [Accessed September 1st 2022].

Emily Eyles, Maria Theresa Redaniel, Tim Jones, Marion Prat, and Tim Keen. Can we accurately forecast nonelective bed occupancy and admissions in the nhs? a time-series msarima analysis of longitudinal data from an nhs trust. *BMJ Open*, 2022. URL https://bmjopen.bmj.com/content/12/4/e056523. [Accessed June 24th 2022].

N. Henderson, V. Selwyn, J. Beezhold, R. Howard, R. Gilmore, and I. Bartolome. The impact of gender identity clinic waiting times on the mental health of transitioning individuals. *European Psychiatry*, 2022. URL https://www.cambridge.org/core/journals/european-psychiatry/article/impact-of-gender-identity-clinic-waiting-times-on-the-mental-health-of-transitioning-individuals/9D78C1C223D689F3A6B97BD9EED0A825. [Accessed September 1st 2022].

Dimuthu Rathnayake, Mike Clarke, and Viraj Jayasinghe. Patient prioritisation methods to shorten waiting times for elective surgery: A systematic review of how to improve access to surgery. *PLoS ONE*, 16(8):e0256578, 2021. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0256578. [Accessed June 25th 2022].

Kanan Shah, Akarsh Sharma, Chris Moulton, Simon Swift, Clifford Mann, and Simon Jones. Forecasting the requirementfor nonelectivehospital beds in the national health service of the united kingdom: Model development study. *JMIR Medical Informatics*, 2021. [Accessed June 24th 2022].

John F. Shortle, Percy H. Brill, Martin J. Fischer, Donald Gross, and Denise M. B. Masi. An algorithm to compute the waiting time distribution for the m/g/1 queue. *INFORMS Journal on Computing*, 16(2):152–161, 2004. URL https://doi.org/10.1287/ijoc.1030.0045. [Accessed June 24th 2022].

Cecília L. Siqueiraa, Edilson F. Arruda, Laura Bahiensea, Germana L. Bahr, and Geraldo R. Motta. Long-term integrated surgery room optimization and recovery ward planning, with a case study in the brazilian national institute of traumatology and orthopedics (into). *European Journal of Operational Research*, 264(2):870–883, 2022. URL https://www.journals.elsevier.com/european-journal-of-operational-research. [Accessed June 24th 2022].

Tavistock and Portman. Gender identity clinic current waiting times. *The Tavistock and Portman NHS Foundation Trust*, 2022. URL `https://gic.nhs.uk/appointments/waiting-times/`. [Accessed September 1st 2022].

Ruth Thorlby, Tim Gardner, and Catherine Turton. Nhs performance and waiting times: Priorities for the next government. *The Health Foundation*, 2019. URL `https://www.health.org.uk/sites/default/files/2019-11/nhs-performance-and-waiting-times-priorities-for-the-next-government-ge02-.pdf`. [Accessed September 1st 2022].

Appendix A

Acknowledgements

We acknowledge the following people and groups whose ongoing support was invaluable in completing this project:

- Edilson Arruda: Lecturer in Business Analytics / Management Science and one of my supervisors, the academic supervisor. His support in brainstorming ideas, supporting managing time across the project, supporting in liaising with UHS and friendliness was immensely beneficial.
- Florina Borca: Senior Information Analyst for Research and Development at UHS and one of my supervisors, the sponsor supervisor. She provided the project itself, the data, insight into UHS internals, regular meetings with other UHS staff and general support. Without her the project would never have gotten off of the ground.
- David Higgs: Consultant Orthopaedic Surgeon with UHS Trauma and Orthopaedics. He provided valuable insight and answered all my questions.
- Jason Teoh: Director of Data and Analytics with UHS. He provided valuable insight, answered all my questions and providing valuable feedback on versions of the model as they were being developed.
- Jonathan Watson: Associate Care Group Manager with UHS. He provided valuable insight into UHS internals and the nature of the queues themselves.
- The UHS Trauma and Orthopaedics Department itself, which provided the opportunity for the project itself, and co-operated helpfully with project development and completion.

Appendix B

Programming Code

B.1 Data Processing and Preparation

This RStudio workbook collates all steps in data processing used in this project. In particular, we transform the "clock-stops" data-set of historically completed referrals into an appropriate set of frequency time-series.

```
# ****************************************
# The doctor will see you now - Personalised Waiting Times
#
# Final re-factor of all existing data manipulation work: Putting all data preparation into one file
#
# This file will perform the following actions in this order:
#     (1) Take as input the clock stops data, incompletes queue data and appropriate lookup tables
#     (2) Filter all anomalies from / generally clean the clock stops data and incompletes queue data
#     (3) Filter to exclusively 110 Trauma and Orthopaedics cases (i.e: excluding 108 Spinal)
#     (4) Filter to exclusively Inpatient cases (i.e: no outpatient clockstops or day case incompletes)
#     (5) Assign to every case in both key datasets a Speciality / Service Area based on staff and OPCS codes
#     (6) Generate time series of incoming / outgoing cases, separated by Speciality and Priority
#     (7) Save all such time series to CSV files, forecasting is then conducted elsewhere
#
# Emma Tarmey
# 18/08/2022
# ****************************************


# ----- Preamble -----
library(dplyr)
library(stringr)

rm(list = ls())
setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Data Analysis/ProcessedData")


# ----- Load data from Excel Sheets -----
clock_stops_data      <- read.csv("Clock_stops_TandO_2018_Anon.csv")
incompletes_data      <- read.csv("20220626_Incompletes_anon.csv")
#procedure_lookup_data <- read.csv("Procedure_decode.csv")
staff_data            <- read.csv("staff_names_and_codes.csv")

# trim white-space from data
clock_stops_data      <- (clock_stops_data %>% mutate(across(where(is.character), str_trim)))
incompletes_data      <- (incompletes_data %>% mutate(across(where(is.character), str_trim)))
#procedure_lookup_data <- (procedure_lookup_data %>% mutate(across(where(is.character), str_trim)))
staff_data            <- (staff_data %>% mutate(across(where(is.character), str_trim)))

# clean-up staff lookup data
colnames(staff_data) <- c("Code", "Title", "Firstname", "Surname", "ServiceArea")
staff_data$Code      <- gsub('[^[:alnum:] ]', '', staff_data$Code)
staff_data$Code      <- gsub(' ', '', staff_data$Code)
staff_data_lookup    <- staff_data %>% select("Code", "ServiceArea")


# ----- Filter all Anomalies -----


# ----- Filter to exclusively Trauma and Orthopaedics Cases -----
clock_stops_TO_data <- clock_stops_data %>% filter(activity_treatment_function_code == 110)
incompletes_TO_data <- incompletes_data %>% filter(Specialty.code == 110)
```

```r
# ----- Filter to exclusively Inpatient Cases -----
clock_stops_TO_inpatient_data <- clock_stops_TO_data %>% filter(waiting_list_type == "IRTT")
incompletes_TO_inpatient_data <- incompletes_TO_data %>% filter(AWL.intended.mgmt == "1 - INPATIENT")


# ----- Assign each case a Service Area -----
clock_stops_TO_inpatient_with_service_data <- merge(clock_stops_TO_inpatient_data,
                                                    staff_data_lookup,
                                                    by.x  = "consultant_code",
                                                    by.y  = "Code",
                                                    all.x = TRUE)
incompletes_TO_inpatient_with_service_data <- merge(incompletes_TO_inpatient_data,
                                                    staff_data_lookup,
                                                    by.x  = "Consultant.code",
                                                    by.y  = "Code",
                                                    all.x = TRUE)


# ----- Generate Historical Time Series for each Priority and Service Area -----

# convert dates to numeric values
clock_stops_TO_inpatient_with_service_data$rtt_startdate <- as.numeric(
                                                            as.Date(clock_stops_TO_inpatient_with_service_data$rtt_startdate,
                                                                       format = "%Y-%m-%d",
                                                                       origin = "1900-01-01"))
clock_stops_TO_inpatient_with_service_data$rtt_enddate   <- as.numeric(
                                                            as.Date(clock_stops_TO_inpatient_with_service_data$rtt_enddate,
                                                                       format = "%Y-%m-%d",
                                                                       origin = "1900-01-01"))

# subset data by service area
foot_ankle_data <- (clock_stops_TO_inpatient_with_service_data %>% filter(ServiceArea == "Foot & Ankle")) %>% select(rtt_startdate, rtt_days_wait, rtt_weeks_wait, rtt_enddate, prior
hand_data       <- (clock_stops_TO_inpatient_with_service_data %>% filter(ServiceArea == "Hand"))         %>% select(rtt_startdate, rtt_days_wait, rtt_weeks_wait, rtt_enddate, prior
hip_data        <- (clock_stops_TO_inpatient_with_service_data %>% filter(ServiceArea == "Hip"))          %>% select(rtt_startdate, rtt_days_wait, rtt_weeks_wait, rtt_enddate, prior
hip_knee_data   <- (clock_stops_TO_inpatient_with_service_data %>% filter(ServiceArea == "Hip & Knee"))   %>% select(rtt_startdate, rtt_days_wait, rtt_weeks_wait, rtt_enddate, prior
knee_data       <- (clock_stops_TO_inpatient_with_service_data %>% filter(ServiceArea == "Knee"))         %>% select(rtt_startdate, rtt_days_wait, rtt_weeks_wait, rtt_enddate, prior
shoulder_data   <- (clock_stops_TO_inpatient_with_service_data %>% filter(ServiceArea == "Shoulder"))     %>% select(rtt_startdate, rtt_days_wait, rtt_weeks_wait, rtt_enddate, prior


# subset data by priority level
foot_ankle_priority_1_data <- foot_ankle_data %>% filter(priority_type_code == 1)
foot_ankle_priority_2_data <- foot_ankle_data %>% filter(priority_type_code == 2)
foot_ankle_priority_3_data <- foot_ankle_data %>% filter(priority_type_code == 3)

hand_priority_1_data       <- hand_data %>% filter(priority_type_code == 1)
hand_priority_2_data       <- hand_data %>% filter(priority_type_code == 2)
hand_priority_3_data       <- hand_data %>% filter(priority_type_code == 3)

hip_priority_1_data        <- hip_data %>% filter(priority_type_code == 1)
hip_priority_2_data        <- hip_data %>% filter(priority_type_code == 2)
hip_priority_3_data        <- hip_data %>% filter(priority_type_code == 3)

hip_knee_priority_1_data   <- hip_knee_data %>% filter(priority_type_code == 1)
hip_knee_priority_2_data   <- hip_knee_data %>% filter(priority_type_code == 2)
hip_knee_priority_3_data   <- hip_knee_data %>% filter(priority_type_code == 3)

knee_priority_1_data       <- knee_data %>% filter(priority_type_code == 1)
knee_priority_2_data       <- knee_data %>% filter(priority_type_code == 2)
knee_priority_3_data       <- knee_data %>% filter(priority_type_code == 3)

shoulder_priority_1_data   <- shoulder_data %>% filter(priority_type_code == 1)
shoulder_priority_2_data   <- shoulder_data %>% filter(priority_type_code == 2)
shoulder_priority_3_data   <- shoulder_data %>% filter(priority_type_code == 3)


# isolate frequencies of each occurence to form time-series

# foot ankle priority 1 enter queue
bins   <- seq(min(foot_ankle_priority_1_data$rtt_startdate), max(foot_ankle_priority_1_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(foot_ankle_priority_1_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
foot_ankle_enter_priority_1_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# foot ankle priority 2 enter queue
bins   <- seq(min(foot_ankle_priority_2_data$rtt_startdate), max(foot_ankle_priority_2_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(foot_ankle_priority_2_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
foot_ankle_enter_priority_2_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# foot ankle priority 3 enter queue
bins   <- seq(min(foot_ankle_priority_3_data$rtt_startdate), max(foot_ankle_priority_3_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(foot_ankle_priority_3_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
foot_ankle_enter_priority_3_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# hand priority 1 enter queue
bins   <- seq(min(hand_priority_1_data$rtt_startdate), max(hand_priority_1_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hand_priority_1_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
```

```
hand_enter_priority_1_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# hand priority 2 enter queue
bins   <- seq(min(hand_priority_2_data$rtt_startdate), max(hand_priority_2_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hand_priority_2_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hand_enter_priority_2_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# hand priority 3 enter queue
bins   <- seq(min(hand_priority_3_data$rtt_startdate), max(hand_priority_3_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hand_priority_3_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hand_enter_priority_3_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# hip priority 1 enter queue
bins   <- seq(min(hip_priority_1_data$rtt_startdate), max(hip_priority_1_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hip_priority_1_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_enter_priority_1_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# hip priority 2 enter queue
bins   <- seq(min(hip_priority_2_data$rtt_startdate), max(hip_priority_2_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hip_priority_2_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_enter_priority_2_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# hip priority 3 enter queue
# NOTE: In this particular data-set, there exist no priority 3 hip referrals
#       Consequently, a time-series cannot be generated.
#bins   <- seq(min(hip_priority_3_data£rtt_startdate), max(hip_priority_3_data£rtt_startdate)+7, by=7)
#ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
#freq   <- hist(hip_priority_3_data£rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_enter_priority_3_timeseries <- data.frame(range = c(), frequency = c())



# hip knee priority 1 enter queue
bins   <- seq(min(hip_knee_priority_1_data$rtt_startdate), max(hip_knee_priority_1_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hip_knee_priority_1_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_knee_enter_priority_1_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# hip knee priority 2 enter queue
bins   <- seq(min(hip_knee_priority_2_data$rtt_startdate), max(hip_knee_priority_2_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hip_knee_priority_2_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_knee_enter_priority_2_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# hip knee priority 3 enter queue
bins   <- seq(min(hip_knee_priority_3_data$rtt_startdate), max(hip_knee_priority_3_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hip_knee_priority_3_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_knee_enter_priority_3_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# knee priority 1 enter queue
bins   <- seq(min(knee_priority_1_data$rtt_startdate), max(knee_priority_1_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(knee_priority_1_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
knee_enter_priority_1_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# knee priority 2 enter queue
bins   <- seq(min(knee_priority_2_data$rtt_startdate), max(knee_priority_2_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(knee_priority_2_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
knee_enter_priority_2_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# knee priority 3 enter queue
bins   <- seq(min(knee_priority_3_data$rtt_startdate), max(knee_priority_3_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(knee_priority_3_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
knee_enter_priority_3_timeseries <- data.frame(range = ranges, frequency = freq$counts)



# shoulder priority 1 enter queue
bins   <- seq(min(shoulder_priority_1_data$rtt_startdate), max(shoulder_priority_1_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(shoulder_priority_1_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
shoulder_enter_priority_1_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# shoulder priority 2 enter queue
bins   <- seq(min(shoulder_priority_2_data$rtt_startdate), max(shoulder_priority_2_data$rtt_startdate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(shoulder_priority_2_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
shoulder_enter_priority_2_timeseries <- data.frame(range = ranges, frequency = freq$counts)

# shoulder priority 3 enter queue
# NOTE: In this particular data-set, there exist no priority 3 shoulder referrals
#       Consequently, a time-series cannot be generated.
```

```
#bins   <- seq(min(shoulder_priority_3_data$rtt_startdate), max(shoulder_priority_3_data$rtt_startdate)+7, by=7)
#ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
#freq   <- hist(shoulder_priority_3_data$rtt_startdate, breaks=bins, include.lowest=TRUE, plot=TRUE)
shoulder_enter_priority_3_timeseries <- data.frame(range = c(), frequency = c())



# foot ankle exit queue
bins   <- seq(min(foot_ankle_data$rtt_enddate), max(foot_ankle_data$rtt_enddate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(foot_ankle_data$rtt_enddate, breaks=bins, include.lowest=TRUE, plot=TRUE)
foot_ankle_exit_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# hand exit queue
bins   <- seq(min(hand_data$rtt_enddate), max(hand_data$rtt_enddate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hand_data$rtt_enddate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hand_exit_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# hip exit queue
bins   <- seq(min(hip_data$rtt_enddate), max(hip_data$rtt_enddate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hip_data$rtt_enddate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_exit_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# hip knee exit queue
bins   <- seq(min(hip_knee_data$rtt_enddate), max(hip_knee_data$rtt_enddate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(hip_knee_data$rtt_enddate, breaks=bins, include.lowest=TRUE, plot=TRUE)
hip_knee_exit_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# knee exit queue
bins   <- seq(min(knee_data$rtt_enddate), max(knee_data$rtt_enddate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(knee_data$rtt_enddate, breaks=bins, include.lowest=TRUE, plot=TRUE)
knee_exit_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# shoulder exit queue
bins   <- seq(min(shoulder_data$rtt_enddate), max(shoulder_data$rtt_enddate)+7, by=7)
ranges <- paste(head(bins,-1), bins[-1], sep=" - ")
freq   <- hist(shoulder_data$rtt_enddate, breaks=bins, include.lowest=TRUE, plot=TRUE)
shoulder_exit_timeseries <- data.frame(range = ranges, frequency = freq$counts)


# ----- Write Resultant Time-Series to CSV ----

write.csv(clock_stops_TO_inpatient_with_service_data, file = "clock_stops_processed.csv", row.names = TRUE)
write.csv(incompletes_TO_inpatient_with_service_data, file = "incompletes_processed.csv", row.names = TRUE)

setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Data Analysis/ProcessedData/TimeSeries")

write.csv(foot_ankle_enter_priority_1_timeseries, file = "foot_ankle_priority_1_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(foot_ankle_enter_priority_2_timeseries, file = "foot_ankle_priority_2_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(foot_ankle_enter_priority_3_timeseries, file = "foot_ankle_priority_3_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(foot_ankle_exit_timeseries,             file = "foot_ankle_priority_all_exit_by_week_timeseries.csv", row.names = TRUE)

write.csv(hand_enter_priority_1_timeseries, file = "hand_priority_1_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hand_enter_priority_2_timeseries, file = "hand_priority_2_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hand_enter_priority_3_timeseries, file = "hand_priority_3_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hand_exit_timeseries,             file = "hand_priority_all_exit_by_week_timeseries.csv", row.names = TRUE)

write.csv(hip_enter_priority_1_timeseries, file = "hip_priority_1_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hip_enter_priority_2_timeseries, file = "hip_priority_2_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hip_enter_priority_3_timeseries, file = "hip_priority_3_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hip_exit_timeseries,             file = "hip_priority_all_exit_by_week_timeseries.csv", row.names = TRUE)

write.csv(hip_knee_enter_priority_1_timeseries, file = "hip_knee_priority_1_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hip_knee_enter_priority_2_timeseries, file = "hip_knee_priority_2_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hip_knee_enter_priority_3_timeseries, file = "hip_knee_priority_3_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(hip_knee_exit_timeseries,             file = "hip_knee_priority_all_exit_by_week_timeseries.csv", row.names = TRUE)

write.csv(knee_enter_priority_1_timeseries, file = "knee_priority_1_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(knee_enter_priority_2_timeseries, file = "knee_priority_2_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(knee_enter_priority_3_timeseries, file = "knee_priority_3_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(knee_exit_timeseries,             file = "knee_priority_all_exit_by_week_timeseries.csv", row.names = TRUE)

write.csv(shoulder_enter_priority_1_timeseries, file = "shoulder_priority_1_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(shoulder_enter_priority_2_timeseries, file = "shoulder_priority_2_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(shoulder_enter_priority_3_timeseries, file = "shoulder_priority_3_enter_by_week_timeseries.csv",  row.names = TRUE)
write.csv(shoulder_exit_timeseries,             file = "shoulder_priority_all_exit_by_week_timeseries.csv", row.names = TRUE)
```

B.2 Generating Forecast Error CDFs

This RStudio workbook generates all necessary CDF functions, allowing pre-computation by saving these functions as CSV files. These functions are then utilised in the final simulation to reflect the uncertainty present in the forecasts.

```r
# ***************************************
# The doctor will see you now - Personalised Waiting Times
#
# Forecasting Waitlist - Fitting Error Distribution
# This will nuance the predictions produced
#
# Emma Tarmey
# 14/08/2022
# ***************************************


# ----- Preamble -----

library(readr)
library(ggplot2)
library(forecast)
library(fpp2)
library(TTR)
library(dplyr)
library(stringr)
library(reshape2)
library(MLmetrics)
library(actuar)
library(GoFKernel)

rm(list = ls())
setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Forecasting/ProcessedData/TimeSeries")

# suppress warnings
oldw <- getOption("warn")
options(warn = -1)



# ----- Load data from Excel Sheets -----

ts_foot_ankle_priority_1_enter_by_week  <- read.csv("foot_ankle_priority_1_enter_by_week_timeseries.csv")
ts_foot_ankle_priority_2_enter_by_week  <- read.csv("foot_ankle_priority_2_enter_by_week_timeseries.csv")
ts_foot_ankle_priority_all_exit_by_week <- read.csv("foot_ankle_priority_all_exit_by_week_timeseries.csv")

ts_hand_priority_1_enter_by_week  <- read.csv("hand_priority_1_enter_by_week_timeseries.csv")
ts_hand_priority_2_enter_by_week  <- read.csv("hand_priority_2_enter_by_week_timeseries.csv")
ts_hand_priority_all_exit_by_week <- read.csv("hand_priority_all_exit_by_week_timeseries.csv")

ts_hip_priority_1_enter_by_week  <- read.csv("hip_priority_1_enter_by_week_timeseries.csv")
ts_hip_priority_2_enter_by_week  <- read.csv("hip_priority_2_enter_by_week_timeseries.csv")
ts_hip_priority_all_exit_by_week <- read.csv("hip_priority_all_exit_by_week_timeseries.csv")

ts_hip_knee_priority_1_enter_by_week  <- read.csv("hip_knee_priority_1_enter_by_week_timeseries.csv")
ts_hip_knee_priority_2_enter_by_week  <- read.csv("hip_knee_priority_2_enter_by_week_timeseries.csv")
ts_hip_knee_priority_all_exit_by_week <- read.csv("hip_knee_priority_all_exit_by_week_timeseries.csv")

ts_knee_priority_1_enter_by_week  <- read.csv("knee_priority_1_enter_by_week_timeseries.csv")
ts_knee_priority_2_enter_by_week  <- read.csv("knee_priority_2_enter_by_week_timeseries.csv")
ts_knee_priority_all_exit_by_week <- read.csv("knee_priority_all_exit_by_week_timeseries.csv")

ts_shoulder_priority_1_enter_by_week  <- read.csv("shoulder_priority_1_enter_by_week_timeseries.csv")
ts_shoulder_priority_2_enter_by_week  <- read.csv("shoulder_priority_2_enter_by_week_timeseries.csv")
ts_shoulder_priority_all_exit_by_week <- read.csv("shoulder_priority_all_exit_by_week_timeseries.csv")



# ----- Make Time-series Object -----

# set window across all data-sets here
# note: incompletes begin at 01/06/2020
window_start = c(2018, 5)
window_end   = c(2021, 5)

test_start = c(2021, 6)
test_end   = c(2022, 6)



# FOOT & ANKLE

# convert time/date codes
ts_foot_ankle_priority_1_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_foot_ankle_priority_1_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_foot_ankle_priority_1_enter_by_week.timeseries    <- ts(ts_foot_ankle_priority_1_enter_by_week$frequency,
                                                           start     = c(2016, 9),
                                                           end       = c(2022, 5),
                                                           frequency = 52)
```

```r
# window to subset timeseries
ts_foot_ankle_priority_1_enter_by_week.testseries <- window(ts_foot_ankle_priority_1_enter_by_week.timeseries,
                                                            start = test_start,
                                                            end   = test_end)
ts_foot_ankle_priority_1_enter_by_week.timeseries <- window(ts_foot_ankle_priority_1_enter_by_week.timeseries,
                                                            start = window_start,
                                                            end   = window_end)
plot(ts_foot_ankle_priority_1_enter_by_week.timeseries, main = "Foot & Ankle Priority 1 Queue Enter Data")



# convert time/date codes
ts_foot_ankle_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_foot_ankle_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_foot_ankle_priority_2_enter_by_week.timeseries    <- ts(ts_foot_ankle_priority_2_enter_by_week$frequency,
                                                           start     = c(2016, 9),
                                                           end       = c(2022, 5),
                                                           frequency = 52)

# window to subset timeseries
ts_foot_ankle_priority_2_enter_by_week.timeseries <- window(ts_foot_ankle_priority_2_enter_by_week.timeseries,
                                                            start = window_start,
                                                            end   = window_end)
plot(ts_foot_ankle_priority_2_enter_by_week.timeseries, main = "Foot & Ankle Priority 2 Queue Enter Data")




# convert time/date codes
ts_foot_ankle_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_foot_ankle_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_foot_ankle_priority_all_exit_by_week.timeseries    <- ts(ts_foot_ankle_priority_all_exit_by_week$frequency,
                                                            start     = c(2016, 9),
                                                            end       = c(2022, 5),
                                                            frequency = 52)

# window to subset timeseries
ts_foot_ankle_priority_all_exit_by_week.timeseries <- window(ts_foot_ankle_priority_all_exit_by_week.timeseries,
                                                             start = window_start,
                                                             end   = window_end)
plot(ts_foot_ankle_priority_all_exit_by_week.timeseries, main = "Foot & Ankle Priority All Queue Exit Data")



# HAND

# convert time/date codes
ts_hand_priority_1_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hand_priority_1_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hand_priority_1_enter_by_week.timeseries    <- ts(ts_hand_priority_1_enter_by_week$frequency,
                                                     start     = c(2016, 9),
                                                     end       = c(2022, 5),
                                                     frequency = 52)

# window to subset timeseries
ts_hand_priority_1_enter_by_week.testseries <- window(ts_hand_priority_1_enter_by_week.timeseries,
                                                      start = test_start,
                                                      end   = test_end)
ts_hand_priority_1_enter_by_week.timeseries <- window(ts_hand_priority_1_enter_by_week.timeseries,
                                                      start = window_start,
                                                      end   = window_end)
plot(ts_hand_priority_1_enter_by_week.timeseries, main = "Hand Priority 1 Queue Enter Data")



# convert time/date codes
ts_hand_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hand_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hand_priority_2_enter_by_week.timeseries    <- ts(ts_hand_priority_2_enter_by_week$frequency,
                                                     start     = c(2016, 9),
                                                     end       = c(2022, 5),
                                                     frequency = 52)

# window to subset timeseries
ts_hand_priority_2_enter_by_week.timeseries <- window(ts_hand_priority_2_enter_by_week.timeseries,
                                                      start = window_start,
                                                      end   = window_end)
plot(ts_hand_priority_2_enter_by_week.timeseries, main = "Hand Priority 2 Queue Enter Data")




# convert time/date codes
ts_hand_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_hand_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_hand_priority_all_exit_by_week.timeseries    <- ts(ts_hand_priority_all_exit_by_week$frequency,
                                                      start     = c(2016, 9),
                                                      end       = c(2022, 5),
                                                      frequency = 52)

# window to subset timeseries
ts_hand_priority_all_exit_by_week.timeseries <- window(ts_hand_priority_all_exit_by_week.timeseries,
                                                       start = window_start,
                                                       end   = window_end)
plot(ts_hand_priority_all_exit_by_week.timeseries, main = "Hand Priority All Queue Exit Data")
```

```r
# HIP

# convert time/date codes
ts_hip_priority_1_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_priority_1_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_priority_1_enter_by_week.timeseries    <- ts(ts_hip_priority_1_enter_by_week$frequency,
                                                    start     = c(2016, 9),
                                                    end       = c(2022, 5),
                                                    frequency = 52)

# window to subset timeseries
ts_hip_priority_1_enter_by_week.testseries <- window(ts_hip_priority_1_enter_by_week.timeseries,
                                                     start = test_start,
                                                     end   = test_end)
ts_hip_priority_1_enter_by_week.timeseries <- window(ts_hip_priority_1_enter_by_week.timeseries,
                                                     start = window_start,
                                                     end   = window_end)
plot(ts_hip_priority_1_enter_by_week.timeseries, main = "Hip Priority 1 Queue Enter Data")



# convert time/date codes
ts_hip_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_priority_2_enter_by_week.timeseries    <- ts(ts_hip_priority_2_enter_by_week$frequency,
                                                    start     = c(2016, 9),
                                                    end       = c(2022, 5),
                                                    frequency = 52)

# window to subset timeseries
ts_hip_priority_2_enter_by_week.timeseries <- window(ts_hip_priority_2_enter_by_week.timeseries,
                                                     start = window_start,
                                                     end   = window_end)
plot(ts_hip_priority_2_enter_by_week.timeseries, main = "Hip Priority 2 Queue Enter Data")




# convert time/date codes
ts_hip_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_priority_all_exit_by_week.timeseries    <- ts(ts_hip_priority_all_exit_by_week$frequency,
                                                     start     = c(2016, 9),
                                                     end       = c(2022, 5),
                                                     frequency = 52)

# window to subset timeseries
ts_hip_priority_all_exit_by_week.timeseries <- window(ts_hip_priority_all_exit_by_week.timeseries,
                                                      start = window_start,
                                                      end   = window_end)
plot(ts_hip_priority_all_exit_by_week.timeseries, main = "Hip Priority All Queue Exit Data")



# HIP & KNEE

# convert time/date codes
ts_hip_knee_priority_1_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_knee_priority_1_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_knee_priority_1_enter_by_week.timeseries    <- ts(ts_hip_knee_priority_1_enter_by_week$frequency,
                                                         start     = c(2016, 9),
                                                         end       = c(2022, 5),
                                                         frequency = 52)

# window to subset timeseries
ts_hip_knee_priority_1_enter_by_week.testseries <- window(ts_hip_knee_priority_1_enter_by_week.timeseries,
                                                          start = test_start,
                                                          end   = test_end)
ts_hip_knee_priority_1_enter_by_week.timeseries <- window(ts_hip_knee_priority_1_enter_by_week.timeseries,
                                                          start = window_start,
                                                          end   = window_end)
plot(ts_hip_knee_priority_1_enter_by_week.timeseries, main = "Hip & Knee Priority 1 Queue Enter Data")



# convert time/date codes
ts_hip_knee_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_knee_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_knee_priority_2_enter_by_week.timeseries    <- ts(ts_hip_knee_priority_2_enter_by_week$frequency,
                                                         start     = c(2016, 9),
                                                         end       = c(2022, 5),
                                                         frequency = 52)

# window to subset timeseries
ts_hip_knee_priority_2_enter_by_week.timeseries <- window(ts_hip_knee_priority_2_enter_by_week.timeseries,
                                                          start = window_start,
                                                          end   = window_end)
plot(ts_hip_knee_priority_2_enter_by_week.timeseries, main = "Hip & Knee Priority 2 Queue Enter Data")




# convert time/date codes
ts_hip_knee_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_knee_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_knee_priority_all_exit_by_week.timeseries    <- ts(ts_hip_knee_priority_all_exit_by_week$frequency,
                                                           start     = c(2016, 9),
                                                           end       = c(2022, 5),
                                                           frequency = 52)

# window to subset timeseries
ts_hip_knee_priority_all_exit_by_week.timeseries <- window(ts_hip_knee_priority_all_exit_by_week.timeseries,
                                                            start = window_start,
                                                            end   = window_end)
plot(ts_hip_knee_priority_all_exit_by_week.timeseries, main = "Hip & Knee Priority All Queue Exit Data")



# KNEE

# convert time/date codes
ts_knee_priority_1_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_knee_priority_1_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_knee_priority_1_enter_by_week.timeseries    <- ts(ts_knee_priority_1_enter_by_week$frequency,
                                                      start     = c(2016, 9),
                                                      end       = c(2022, 5),
                                                      frequency = 52)

# window to subset timeseries
ts_knee_priority_1_enter_by_week.testseries <- window(ts_knee_priority_1_enter_by_week.timeseries,
                                                       start = test_start,
                                                       end   = test_end)
ts_knee_priority_1_enter_by_week.timeseries <- window(ts_knee_priority_1_enter_by_week.timeseries,
                                                       start = window_start,
                                                       end   = window_end)
plot(ts_knee_priority_1_enter_by_week.timeseries, main = "Knee Priority 1 Queue Enter Data")



# convert time/date codes
ts_knee_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_knee_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_knee_priority_2_enter_by_week.timeseries    <- ts(ts_knee_priority_2_enter_by_week$frequency,
                                                      start     = c(2016, 9),
                                                      end       = c(2022, 5),
                                                      frequency = 52)

# window to subset timeseries
ts_knee_priority_2_enter_by_week.timeseries <- window(ts_knee_priority_2_enter_by_week.timeseries,
                                                       start = window_start,
                                                       end   = window_end)
plot(ts_knee_priority_2_enter_by_week.timeseries, main = "Knee Priority 2 Queue Enter Data")




# convert time/date codes
ts_knee_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_knee_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_knee_priority_all_exit_by_week.timeseries    <- ts(ts_knee_priority_all_exit_by_week$frequency,
                                                       start     = c(2016, 9),
                                                       end       = c(2022, 5),
                                                       frequency = 52)

# window to subset timeseries
ts_knee_priority_all_exit_by_week.timeseries <- window(ts_knee_priority_all_exit_by_week.timeseries,
                                                        start = window_start,
                                                        end   = window_end)
plot(ts_knee_priority_all_exit_by_week.timeseries, main = "Knee Priority All Queue Exit Data")



# SHOULDER

# convert time/date codes
ts_shoulder_priority_1_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_shoulder_priority_1_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_shoulder_priority_1_enter_by_week.timeseries    <- ts(ts_shoulder_priority_1_enter_by_week$frequency,
                                                          start     = c(2016, 9),
                                                          end       = c(2022, 5),
                                                          frequency = 52)

# window to subset timeseries
ts_shoulder_priority_1_enter_by_week.testseries <- window(ts_shoulder_priority_1_enter_by_week.timeseries,
                                                           start = test_start,
                                                           end   = test_end)
ts_shoulder_priority_1_enter_by_week.timeseries <- window(ts_shoulder_priority_1_enter_by_week.timeseries,
                                                           start = window_start,
                                                           end   = window_end)
plot(ts_shoulder_priority_1_enter_by_week.timeseries, main = "Shoulder Priority 1 Queue Enter Data")



# convert time/date codes
ts_shoulder_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_shoulder_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_shoulder_priority_2_enter_by_week.timeseries    <- ts(ts_shoulder_priority_2_enter_by_week$frequency,
                                                          start     = c(2016, 9),
                                                          end       = c(2022, 5),
                                                          frequency = 52)

# window to subset timeseries
```

```
ts_shoulder_priority_2_enter_by_week.timeseries <- window(ts_shoulder_priority_2_enter_by_week.timeseries,
                                                          start = window_start,
                                                          end   = window_end)
plot(ts_shoulder_priority_2_enter_by_week.timeseries, main = "Shoulder Priority 2 Queue Enter Data")




# convert time/date codes
ts_shoulder_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_shoulder_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_shoulder_priority_all_exit_by_week.timeseries    <- ts(ts_shoulder_priority_all_exit_by_week$frequency,
                                                          start     = c(2016, 9),
                                                          end       = c(2022, 5),
                                                          frequency = 52)

# window to subset timeseries
ts_shoulder_priority_all_exit_by_week.timeseries <- window(ts_shoulder_priority_all_exit_by_week.timeseries,
                                                           start = window_start,
                                                           end   = window_end)
plot(ts_shoulder_priority_all_exit_by_week.timeseries, main = "Shoulder Priority All Queue Exit Data")






# ----- Forecasting -----


# FOOT & ANKLE

#  PRIORITY 1 ENTRY

# Build TBATS Model
ts_foot_ankle_priority_1_enter_by_week.tbats <- tbats(ts_foot_ankle_priority_1_enter_by_week.timeseries)
summary(ts_foot_ankle_priority_1_enter_by_week.tbats)

# Predict TBATS Values
ts_foot_ankle_priority_1_enter_by_week.tbats.forecast <- forecast(ts_foot_ankle_priority_1_enter_by_week.tbats, h = 12)
plot(ts_foot_ankle_priority_1_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Foot & Ankle Priority 1 Queue Entry TBATS Model Prediction")


# PRIORITY 2 ENTRY

# Build TBATS Model
ts_foot_ankle_priority_2_enter_by_week.tbats <- tbats(ts_foot_ankle_priority_2_enter_by_week.timeseries)
summary(ts_foot_ankle_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_foot_ankle_priority_2_enter_by_week.tbats.forecast <- forecast(ts_foot_ankle_priority_2_enter_by_week.tbats, h = 12)
plot(ts_foot_ankle_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Foot & Ankle Priority 2 Queue Entry TBATS Model Prediction")


# PRIORITY ALL EXIT

# Build TBATS Model
ts_foot_ankle_priority_all_exit_by_week.tbats <- tbats(ts_foot_ankle_priority_all_exit_by_week.timeseries)
summary(ts_foot_ankle_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_foot_ankle_priority_all_exit_by_week.tbats.forecast <- forecast(ts_foot_ankle_priority_all_exit_by_week.tbats, h = 12)
plot(ts_foot_ankle_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Foot & Ankle Priority All Queue Exit TBATS Model Prediction")



# HAND

#  PRIORITY 1 ENTRY

# Build TBATS Model
ts_hand_priority_1_enter_by_week.tbats <- tbats(ts_hand_priority_1_enter_by_week.timeseries)
summary(ts_hand_priority_1_enter_by_week.tbats)

# Predict TBATS Values
ts_hand_priority_1_enter_by_week.tbats.forecast <- forecast(ts_hand_priority_1_enter_by_week.tbats, h = 12)
plot(ts_hand_priority_1_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hand Priority 1 Queue Entry TBATS Model Prediction")


# PRIORITY 2 ENTRY

# Build TBATS Model
ts_hand_priority_2_enter_by_week.tbats <- tbats(ts_hand_priority_2_enter_by_week.timeseries)
summary(ts_hand_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_hand_priority_2_enter_by_week.tbats.forecast <- forecast(ts_hand_priority_2_enter_by_week.tbats, h = 12)
plot(ts_hand_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hand Priority 2 Queue Entry TBATS Model Prediction")


# PRIORITY ALL EXIT
```

```r
# Build TBATS Model
ts_hand_priority_all_exit_by_week.tbats <- tbats(ts_hand_priority_all_exit_by_week.timeseries)
summary(ts_hand_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_hand_priority_all_exit_by_week.tbats.forecast <- forecast(ts_hand_priority_all_exit_by_week.tbats, h = 12)
plot(ts_hand_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hand Priority All Queue Exit TBATS Model Prediction")



# HIP

#  PRIORITY 1 ENTRY

# Build TBATS Model
ts_hip_priority_1_enter_by_week.tbats <- tbats(ts_hip_priority_1_enter_by_week.timeseries)
summary(ts_hip_priority_1_enter_by_week.tbats)

# Predict TBATS Values
ts_hip_priority_1_enter_by_week.tbats.forecast <- forecast(ts_hip_priority_1_enter_by_week.tbats, h = 12)
plot(ts_hip_priority_1_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hip Priority 1 Queue Entry TBATS Model Prediction")


# PRIORITY 2 ENTRY

# Build TBATS Model
ts_hip_priority_2_enter_by_week.tbats <- tbats(ts_hip_priority_2_enter_by_week.timeseries)
summary(ts_hip_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_hip_priority_2_enter_by_week.tbats.forecast <- forecast(ts_hip_priority_2_enter_by_week.tbats, h = 12)
plot(ts_hip_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hip Priority 2 Queue Entry TBATS Model Prediction")


# PRIORITY ALL EXIT

# Build TBATS Model
ts_hip_priority_all_exit_by_week.tbats <- tbats(ts_hip_priority_all_exit_by_week.timeseries)
summary(ts_hip_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_hip_priority_all_exit_by_week.tbats.forecast <- forecast(ts_hip_priority_all_exit_by_week.tbats, h = 12)
plot(ts_hip_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hip Priority All Queue Exit TBATS Model Prediction")



# HIP & KNEE

#  PRIORITY 1 ENTRY

# Build TBATS Model
ts_hip_knee_priority_1_enter_by_week.tbats <- tbats(ts_hip_knee_priority_1_enter_by_week.timeseries)
summary(ts_hip_knee_priority_1_enter_by_week.tbats)

# Predict TBATS Values
ts_hip_knee_priority_1_enter_by_week.tbats.forecast <- forecast(ts_hip_knee_priority_1_enter_by_week.tbats, h = 12)
plot(ts_hip_knee_priority_1_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hip & Knee Priority 1 Queue Entry TBATS Model Prediction")


# PRIORITY 2 ENTRY

# Build TBATS Model
ts_hip_knee_priority_2_enter_by_week.tbats <- tbats(ts_hip_knee_priority_2_enter_by_week.timeseries)
summary(ts_hip_knee_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_hip_knee_priority_2_enter_by_week.tbats.forecast <- forecast(ts_hip_knee_priority_2_enter_by_week.tbats, h = 12)
plot(ts_hip_knee_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hip & Knee Priority 2 Queue Entry TBATS Model Prediction")


# PRIORITY ALL EXIT

# Build TBATS Model
ts_hip_knee_priority_all_exit_by_week.tbats <- tbats(ts_hip_knee_priority_all_exit_by_week.timeseries)
summary(ts_hip_knee_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_hip_knee_priority_all_exit_by_week.tbats.forecast <- forecast(ts_hip_knee_priority_all_exit_by_week.tbats, h = 12)
plot(ts_hip_knee_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hip & Knee Priority All Queue Exit TBATS Model Prediction")



# KNEE

#  PRIORITY 1 ENTRY

# Build TBATS Model
ts_knee_priority_1_enter_by_week.tbats <- tbats(ts_knee_priority_1_enter_by_week.timeseries)
summary(ts_knee_priority_1_enter_by_week.tbats)

# Predict TBATS Values
```

```r
ts_knee_priority_1_enter_by_week.tbats.forecast <- forecast(ts_knee_priority_1_enter_by_week.tbats, h = 12)
plot(ts_knee_priority_1_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Knee Priority 1 Queue Entry TBATS Model Prediction")


# PRIORITY 2 ENTRY

# Build TBATS Model
ts_knee_priority_2_enter_by_week.tbats <- tbats(ts_knee_priority_2_enter_by_week.timeseries)
summary(ts_knee_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_knee_priority_2_enter_by_week.tbats.forecast <- forecast(ts_knee_priority_2_enter_by_week.tbats, h = 12)
plot(ts_knee_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Knee Priority 2 Queue Entry TBATS Model Prediction")


# PRIORITY ALL EXIT

# Build TBATS Model
ts_knee_priority_all_exit_by_week.tbats <- tbats(ts_knee_priority_all_exit_by_week.timeseries)
summary(ts_knee_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_knee_priority_all_exit_by_week.tbats.forecast <- forecast(ts_knee_priority_all_exit_by_week.tbats, h = 12)
plot(ts_knee_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Knee Priority All Queue Exit TBATS Model Prediction")



# SHOULDER

#  PRIORITY 1 ENTRY

# Build TBATS Model
ts_shoulder_priority_1_enter_by_week.tbats <- tbats(ts_shoulder_priority_1_enter_by_week.timeseries)
summary(ts_shoulder_priority_1_enter_by_week.tbats)

# Predict TBATS Values
ts_shoulder_priority_1_enter_by_week.tbats.forecast <- forecast(ts_shoulder_priority_1_enter_by_week.tbats, h = 12)
plot(ts_shoulder_priority_1_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Shoulder Priority 1 Queue Entry TBATS Model Prediction")


# PRIORITY 2 ENTRY

# Build TBATS Model
ts_shoulder_priority_2_enter_by_week.tbats <- tbats(ts_shoulder_priority_2_enter_by_week.timeseries)
summary(ts_shoulder_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_shoulder_priority_2_enter_by_week.tbats.forecast <- forecast(ts_shoulder_priority_2_enter_by_week.tbats, h = 12)
plot(ts_shoulder_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Shoulder Priority 2 Queue Entry TBATS Model Prediction")


# PRIORITY ALL EXIT

# Build TBATS Model
ts_shoulder_priority_all_exit_by_week.tbats <- tbats(ts_shoulder_priority_all_exit_by_week.timeseries)
summary(ts_shoulder_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_shoulder_priority_all_exit_by_week.tbats.forecast <- forecast(ts_shoulder_priority_all_exit_by_week.tbats, h = 12)
plot(ts_shoulder_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Shoulder Priority All Queue Exit TBATS Model Prediction")




# ----- Extracting Appropriate Error Values from Forecasts -----


# FOOT & ANKLE

# PRIORITY 1 ENTER

# syntax sugar for readability
fc <- ts_foot_ankle_priority_1_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.footankle.p1.enter.mean <- fc$mean
fc.footankle.p1.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.footankle.p1.enter.res  <- residuals(fc)
fc.footankle.p1.enter.mean
fc.footankle.p1.enter.sd
fc.footankle.p1.enter.res


# PRIORITY 2 ENTER

# syntax sugar for readability
fc <- ts_foot_ankle_priority_2_enter_by_week.tbats.forecast
```

```r
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.footankle.p2.enter.mean <- fc$mean
fc.footankle.p2.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.footankle.p2.enter.res  <- residuals(fc)
fc.footankle.p2.enter.mean
fc.footankle.p2.enter.sd
fc.footankle.p2.enter.res


# PRIORITY ALL EXIT

# syntax sugar for readability
fc <- ts_foot_ankle_priority_all_exit_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.footankle.pALL.exit.mean <- fc$mean
fc.footankle.pALL.exit.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.footankle.pALL.exit.res  <- residuals(fc)
fc.footankle.pALL.exit.mean
fc.footankle.pALL.exit.sd
fc.footankle.pALL.exit.res


# HAND

# PRIORITY 1 ENTER

# syntax sugar for readability
fc <- ts_hand_priority_1_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hand.p1.enter.mean <- fc$mean
fc.hand.p1.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hand.p1.enter.res  <- residuals(fc)
fc.hand.p1.enter.mean
fc.hand.p1.enter.sd
fc.hand.p1.enter.res


# PRIORITY 2 ENTER

# syntax sugar for readability
fc <- ts_hand_priority_2_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hand.p2.enter.mean <- fc$mean
fc.hand.p2.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hand.p2.enter.res  <- residuals(fc)
fc.hand.p2.enter.mean
fc.hand.p2.enter.sd
fc.hand.p2.enter.res


# PRIORITY ALL EXIT

# syntax sugar for readability
fc <- ts_hand_priority_all_exit_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hand.pALL.exit.mean <- fc$mean
fc.hand.pALL.exit.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hand.pALL.exit.res  <- residuals(fc)
fc.hand.pALL.exit.mean
fc.hand.pALL.exit.sd
fc.hand.pALL.exit.res


# HIP

# PRIORITY 1 ENTER
# syntax sugar for readability
fc <- ts_hip_priority_1_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hip.p1.enter.mean <- fc$mean
fc.hip.p1.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hip.p1.enter.res  <- residuals(fc)
fc.hip.p1.enter.mean
fc.hip.p1.enter.sd
fc.hip.p1.enter.res


# PRIORITY 2 ENTER

# syntax sugar for readability
fc <- ts_hip_priority_2_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hip.p2.enter.mean <- fc$mean
fc.hip.p2.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hip.p2.enter.res  <- residuals(fc)
fc.hip.p2.enter.mean
fc.hip.p2.enter.sd
fc.hip.p2.enter.res


# PRIORITY ALL EXIT

# syntax sugar for readability
fc <- ts_hip_priority_all_exit_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hip.pALL.exit.mean <- fc$mean
fc.hip.pALL.exit.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hip.pALL.exit.res  <- residuals(fc)
fc.hip.pALL.exit.mean
fc.hip.pALL.exit.sd
fc.hip.pALL.exit.res


# HIP & KNEE

# PRIORITY 1 ENTER

# syntax sugar for readability
fc <- ts_hip_knee_priority_1_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hipknee.p1.enter.mean <- fc$mean
fc.hipknee.p1.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hipknee.p1.enter.res  <- residuals(fc)
fc.hipknee.p1.enter.mean
fc.hipknee.p1.enter.sd
fc.hipknee.p1.enter.res


# PRIORITY 2 ENTER

# syntax sugar for readability
fc <- ts_hip_knee_priority_2_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hipknee.p2.enter.mean <- fc$mean
fc.hipknee.p2.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hipknee.p2.enter.res  <- residuals(fc)
fc.hipknee.p2.enter.mean
fc.hipknee.p2.enter.sd
fc.hipknee.p2.enter.res
```

```
# PRIORITY ALL EXIT

# syntax sugar for readability
fc <- ts_hip_knee_priority_all_exit_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.hipknee.pALL.exit.mean <- fc$mean
fc.hipknee.pALL.exit.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.hipknee.pALL.exit.res  <- residuals(fc)
fc.hipknee.pALL.exit.mean
fc.hipknee.pALL.exit.sd
fc.hipknee.pALL.exit.res



# KNEE

# PRIORITY 1 ENTER

# syntax sugar for readability
fc <- ts_knee_priority_1_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.knee.p1.enter.mean <- fc$mean
fc.knee.p1.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.knee.p1.enter.res  <- residuals(fc)
fc.knee.p1.enter.mean
fc.knee.p1.enter.sd
fc.knee.p1.enter.res


# PRIORITY 2 ENTER

# syntax sugar for readability
fc <- ts_knee_priority_2_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.knee.p2.enter.mean <- fc$mean
fc.knee.p2.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.knee.p2.enter.res  <- residuals(fc)
fc.knee.p2.enter.mean
fc.knee.p2.enter.sd
fc.knee.p2.enter.res


# PRIORITY ALL EXIT

# syntax sugar for readability
fc <- ts_knee_priority_all_exit_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.knee.pALL.exit.mean <- fc$mean
fc.knee.pALL.exit.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.knee.pALL.exit.res  <- residuals(fc)
fc.knee.pALL.exit.mean
fc.knee.pALL.exit.sd
fc.knee.pALL.exit.res



# SHOULDER

# PRIORITY 1 ENTER

# syntax sugar for readability
fc <- ts_shoulder_priority_1_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.shoulder.p1.enter.mean <- fc$mean
fc.shoulder.p1.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.shoulder.p1.enter.res  <- residuals(fc)
```

```r
fc.shoulder.p1.enter.mean
fc.shoulder.p1.enter.sd
fc.shoulder.p1.enter.res


# PRIORITY 2 ENTER

# syntax sugar for readability
fc <- ts_shoulder_priority_2_enter_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.shoulder.p2.enter.mean <- fc$mean
fc.shoulder.p2.enter.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.shoulder.p2.enter.res  <- residuals(fc)
fc.shoulder.p2.enter.mean
fc.shoulder.p2.enter.sd
fc.shoulder.p2.enter.res


# PRIORITY ALL EXIT

# syntax sugar for readability
fc <- ts_shoulder_priority_all_exit_by_week.tbats.forecast
fc$mean
fc$upper[,2] # 95% confidence interval
fc$lower[,2] # 95% confidence interval
residuals(fc)

# extract distribution information
fc.shoulder.pALL.exit.mean <- fc$mean
fc.shoulder.pALL.exit.sd   <- (fc$upper[,2] - fc$lower[,2]) / (2 * qnorm(.5 + fc$level[1] / 200))
fc.shoulder.pALL.exit.res  <- residuals(fc)
fc.shoulder.pALL.exit.mean
fc.shoulder.pALL.exit.sd
fc.shoulder.pALL.exit.res




# ----- Fitting Discretised, Truncated, Re-Normalised Predictive Distribution -----

# FOOT & ANKLE

# PRIORITY 1 ENTER

mean(fc.footankle.p1.enter.res)
sd(fc.footankle.p1.enter.res)
max(fc.footankle.p1.enter.res)
min(fc.footankle.p1.enter.res)

density( fc.footankle.p1.enter.res )
ecdf( fc.footankle.p1.enter.res )

fc.footankle.p1.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.footankle.p1.enter.res )
  return( cdf(x) )
}

fc.footankle.p1.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.footankle.p1.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}


plot(fc.footankle.p1.enter.res,
     main = "Foot & Ankle Priority 1 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.footankle.p1.enter.res ),
     main = "Foot & Ankle Priority 1 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.footankle.p1.enter.res ),
     main = "Foot & Ankle Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")
```

```
plot(fc.footankle.p1.enter.res.cdf,
     -10,
     10,
     main = "Foot & Ankle Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.footankle.p1.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "Foot & Ankle Priority 1 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")




# PRIORITY 2 ENTER

mean(fc.footankle.p2.enter.res)
sd(fc.footankle.p2.enter.res)
max(fc.footankle.p2.enter.res)
min(fc.footankle.p2.enter.res)

density( fc.footankle.p2.enter.res )
ecdf( fc.footankle.p2.enter.res )

fc.footankle.p2.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.footankle.p2.enter.res )
  return( cdf(x) )
}

fc.footankle.p2.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.footankle.p2.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.footankle.p2.enter.res,
     main = "Foot & Ankle Priority 2 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.footankle.p2.enter.res ),
     main = "Foot & Ankle Priority 2 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.footankle.p2.enter.res ),
     main = "Foot & Ankle Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.footankle.p2.enter.res.cdf,
     -10,
     10,
     main = "Foot & Ankle Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.footankle.p2.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "Foot & Ankle Priority 2 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")




# PRIORITY ALL EXIT

mean(fc.footankle.pALL.exit.res)
sd(fc.footankle.pALL.exit.res)
max(fc.footankle.pALL.exit.res)
min(fc.footankle.pALL.exit.res)

density( fc.footankle.pALL.exit.res )
ecdf( fc.footankle.pALL.exit.res )

fc.footankle.pALL.exit.res.cdf <- function(x) {
  cdf <- ecdf( fc.footankle.pALL.exit.res )
  return( cdf(x) )
}

fc.footankle.pALL.exit.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.footankle.pALL.exit.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.footankle.pALL.exit.res,
     main = "Foot & Ankle Priority ALL Exit Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.footankle.pALL.exit.res ),
     main = "Foot & Ankle Priority ALL Exit Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.footankle.pALL.exit.res ),
     main = "Foot & Ankle Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.footankle.pALL.exit.res.cdf,
     -10,
     10,
     main = "Foot & Ankle Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.footankle.pALL.exit.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "Foot & Ankle Priority ALL Exit Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")


# Hand

# PRIORITY 1 ENTER

mean(fc.hand.p1.enter.res)
sd(fc.hand.p1.enter.res)
max(fc.hand.p1.enter.res)
min(fc.hand.p1.enter.res)

density( fc.hand.p1.enter.res )
ecdf( fc.hand.p1.enter.res )

fc.hand.p1.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.hand.p1.enter.res )
  return( cdf(x) )
}

fc.hand.p1.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hand.p1.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}


plot(fc.hand.p1.enter.res,
     main = "Hand Priority 1 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hand.p1.enter.res ),
     main = "Hand Priority 1 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hand.p1.enter.res ),
     main = "Hand Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hand.p1.enter.res.cdf,
     -10,
     10,
     main = "Hand Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hand.p1.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "Hand Priority 1 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")




# PRIORITY 2 ENTER

mean(fc.hand.p2.enter.res)
sd(fc.hand.p2.enter.res)
max(fc.hand.p2.enter.res)
min(fc.hand.p2.enter.res)

density( fc.hand.p2.enter.res )
ecdf( fc.hand.p2.enter.res )

fc.hand.p2.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.hand.p2.enter.res )
  return( cdf(x) )
}

fc.hand.p2.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hand.p2.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.hand.p2.enter.res,
     main = "Hand Priority 2 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hand.p2.enter.res ),
     main = "Hand Priority 2 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hand.p2.enter.res ),
     main = "Hand Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hand.p2.enter.res.cdf,
     -10,
     10,
     main = "Hand Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hand.p2.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "Hand Priority 2 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")




# PRIORITY ALL EXIT

mean(fc.hand.pALL.exit.res)
sd(fc.hand.pALL.exit.res)
max(fc.hand.pALL.exit.res)
min(fc.hand.pALL.exit.res)

density( fc.hand.pALL.exit.res )
ecdf( fc.hand.pALL.exit.res )

fc.hand.pALL.exit.res.cdf <- function(x) {
  cdf <- ecdf( fc.hand.pALL.exit.res )
  return( cdf(x) )
}

fc.hand.pALL.exit.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hand.pALL.exit.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.hand.pALL.exit.res,
     main = "Hand Priority ALL Exit Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hand.pALL.exit.res ),
     main = "Hand Priority ALL Exit Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hand.pALL.exit.res ),
     main = "Hand Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hand.pALL.exit.res.cdf,
     -10,
     10,
     main = "Hand Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hand.pALL.exit.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "Hand Priority ALL Exit Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")


# hip

# PRIORITY 1 ENTER

mean(fc.hip.p1.enter.res)
sd(fc.hip.p1.enter.res)
max(fc.hip.p1.enter.res)
min(fc.hip.p1.enter.res)

density( fc.hip.p1.enter.res )
ecdf( fc.hip.p1.enter.res )

fc.hip.p1.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.hip.p1.enter.res )
  return( cdf(x) )
}

fc.hip.p1.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hip.p1.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}


plot(fc.hip.p1.enter.res,
     main = "hip Priority 1 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hip.p1.enter.res ),
     main = "hip Priority 1 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hip.p1.enter.res ),
     main = "hip Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hip.p1.enter.res.cdf,
     -10,
     10,
     main = "hip Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hip.p1.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "hip Priority 1 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")




# PRIORITY 2 ENTER

mean(fc.hip.p2.enter.res)
sd(fc.hip.p2.enter.res)
max(fc.hip.p2.enter.res)
min(fc.hip.p2.enter.res)

density( fc.hip.p2.enter.res )
ecdf( fc.hip.p2.enter.res )

fc.hip.p2.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.hip.p2.enter.res )
  return( cdf(x) )
}

fc.hip.p2.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hip.p2.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.hip.p2.enter.res,
     main = "hip Priority 2 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hip.p2.enter.res ),
     main = "hip Priority 2 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hip.p2.enter.res ),
     main = "hip Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hip.p2.enter.res.cdf,
     -10,
     10,
     main = "hip Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hip.p2.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "hip Priority 2 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")




# PRIORITY ALL EXIT

mean(fc.hip.pALL.exit.res)
sd(fc.hip.pALL.exit.res)
max(fc.hip.pALL.exit.res)
min(fc.hip.pALL.exit.res)

density( fc.hip.pALL.exit.res )
ecdf( fc.hip.pALL.exit.res )

fc.hip.pALL.exit.res.cdf <- function(x) {
  cdf <- ecdf( fc.hip.pALL.exit.res )
  return( cdf(x) )
}

fc.hip.pALL.exit.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hip.pALL.exit.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }
  return (max(try.seq))
}

plot(fc.hip.pALL.exit.res,
     main = "hip Priority ALL Exit Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hip.pALL.exit.res ),
     main = "hip Priority ALL Exit Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hip.pALL.exit.res ),
     main = "hip Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hip.pALL.exit.res.cdf,
     -10,
     10,
     main = "hip Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hip.pALL.exit.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "hip Priority ALL Exit Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")


# hipknee

# PRIORITY 1 ENTER

mean(fc.hipknee.p1.enter.res)
sd(fc.hipknee.p1.enter.res)
max(fc.hipknee.p1.enter.res)
min(fc.hipknee.p1.enter.res)

density( fc.hipknee.p1.enter.res )
ecdf( fc.hipknee.p1.enter.res )

fc.hipknee.p1.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.hipknee.p1.enter.res )
  return( cdf(x) )
}

fc.hipknee.p1.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hipknee.p1.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}


plot(fc.hipknee.p1.enter.res,
     main = "hipknee Priority 1 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hipknee.p1.enter.res ),
     main = "hipknee Priority 1 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hipknee.p1.enter.res ),
     main = "hipknee Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hipknee.p1.enter.res.cdf,
     -10,
     10,
     main = "hipknee Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hipknee.p1.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "hipknee Priority 1 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")
```

```
# PRIORITY 2 ENTER

mean(fc.hipknee.p2.enter.res)
sd(fc.hipknee.p2.enter.res)
max(fc.hipknee.p2.enter.res)
min(fc.hipknee.p2.enter.res)

density( fc.hipknee.p2.enter.res )
ecdf( fc.hipknee.p2.enter.res )

fc.hipknee.p2.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.hipknee.p2.enter.res )
  return( cdf(x) )
}

fc.hipknee.p2.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hipknee.p2.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.hipknee.p2.enter.res,
     main = "hipknee Priority 2 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hipknee.p2.enter.res ),
     main = "hipknee Priority 2 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hipknee.p2.enter.res ),
     main = "hipknee Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hipknee.p2.enter.res.cdf,
     -10,
     10,
     main = "hipknee Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hipknee.p2.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "hipknee Priority 2 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")




# PRIORITY ALL EXIT

mean(fc.hipknee.pALL.exit.res)
sd(fc.hipknee.pALL.exit.res)
max(fc.hipknee.pALL.exit.res)
min(fc.hipknee.pALL.exit.res)

density( fc.hipknee.pALL.exit.res )
ecdf( fc.hipknee.pALL.exit.res )

fc.hipknee.pALL.exit.res.cdf <- function(x) {
  cdf <- ecdf( fc.hipknee.pALL.exit.res )
  return( cdf(x) )
}

fc.hipknee.pALL.exit.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.hipknee.pALL.exit.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}
```

```
plot(fc.hipknee.pALL.exit.res,
     main = "hipknee Priority ALL Exit Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.hipknee.pALL.exit.res ),
     main = "hipknee Priority ALL Exit Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.hipknee.pALL.exit.res ),
     main = "hipknee Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.hipknee.pALL.exit.res.cdf,
     -10,
     10,
     main = "hipknee Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.hipknee.pALL.exit.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "hipknee Priority ALL Exit Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")


# knee

# PRIORITY 1 ENTER

mean(fc.knee.p1.enter.res)
sd(fc.knee.p1.enter.res)
max(fc.knee.p1.enter.res)
min(fc.knee.p1.enter.res)

density( fc.knee.p1.enter.res )
ecdf( fc.knee.p1.enter.res )

fc.knee.p1.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.knee.p1.enter.res )
  return( cdf(x) )
}

fc.knee.p1.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.knee.p1.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}


plot(fc.knee.p1.enter.res,
     main = "knee Priority 1 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.knee.p1.enter.res ),
     main = "knee Priority 1 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.knee.p1.enter.res ),
     main = "knee Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.knee.p1.enter.res.cdf,
     -10,
     10,
     main = "knee Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.knee.p1.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "knee Priority 1 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")
```

```r
# PRIORITY 2 ENTER

mean(fc.knee.p2.enter.res)
sd(fc.knee.p2.enter.res)
max(fc.knee.p2.enter.res)
min(fc.knee.p2.enter.res)

density( fc.knee.p2.enter.res )
ecdf( fc.knee.p2.enter.res )

fc.knee.p2.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.knee.p2.enter.res )
  return( cdf(x) )
}

fc.knee.p2.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.knee.p2.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.knee.p2.enter.res,
     main = "knee Priority 2 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.knee.p2.enter.res ),
     main = "knee Priority 2 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.knee.p2.enter.res ),
     main = "knee Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.knee.p2.enter.res.cdf,
     -10,
     10,
     main = "knee Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.knee.p2.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "knee Priority 2 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")



# PRIORITY ALL EXIT

mean(fc.knee.pALL.exit.res)
sd(fc.knee.pALL.exit.res)
max(fc.knee.pALL.exit.res)
min(fc.knee.pALL.exit.res)

density( fc.knee.pALL.exit.res )
ecdf( fc.knee.pALL.exit.res )

fc.knee.pALL.exit.res.cdf <- function(x) {
  cdf <- ecdf( fc.knee.pALL.exit.res )
  return( cdf(x) )
}

fc.knee.pALL.exit.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.knee.pALL.exit.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.knee.pALL.exit.res,
     main = "knee Priority ALL Exit Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")
```

```r
plot(density( fc.knee.pALL.exit.res ),
     main = "knee Priority ALL Exit Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.knee.pALL.exit.res ),
     main = "knee Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.knee.pALL.exit.res.cdf,
     -10,
     10,
     main = "knee Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.knee.pALL.exit.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "knee Priority ALL Exit Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")


# shoulder

# PRIORITY 1 ENTER

mean(fc.shoulder.p1.enter.res)
sd(fc.shoulder.p1.enter.res)
max(fc.shoulder.p1.enter.res)
min(fc.shoulder.p1.enter.res)

density( fc.shoulder.p1.enter.res )
ecdf( fc.shoulder.p1.enter.res )

fc.shoulder.p1.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.shoulder.p1.enter.res )
  return( cdf(x) )
}

fc.shoulder.p1.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.shoulder.p1.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}


plot(fc.shoulder.p1.enter.res,
     main = "shoulder Priority 1 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.shoulder.p1.enter.res ),
     main = "shoulder Priority 1 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.shoulder.p1.enter.res ),
     main = "shoulder Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.shoulder.p1.enter.res.cdf,
     -10,
     10,
     main = "shoulder Priority 1 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.shoulder.p1.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "shoulder Priority 1 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")



# PRIORITY 2 ENTER

mean(fc.shoulder.p2.enter.res)
sd(fc.shoulder.p2.enter.res)
```

```
max(fc.shoulder.p2.enter.res)
min(fc.shoulder.p2.enter.res)

density( fc.shoulder.p2.enter.res )
ecdf( fc.shoulder.p2.enter.res )

fc.shoulder.p2.enter.res.cdf <- function(x) {
  cdf <- ecdf( fc.shoulder.p2.enter.res )
  return( cdf(x) )
}

fc.shoulder.p2.enter.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.shoulder.p2.enter.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.shoulder.p2.enter.res,
     main = "shoulder Priority 2 Entry Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.shoulder.p2.enter.res ),
     main = "shoulder Priority 2 Entry Forecast Residual Distribution",
     xlab = "Residual",
     ylab = "Probability")

plot(ecdf( fc.shoulder.p2.enter.res ),
     main = "shoulder Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.shoulder.p2.enter.res.cdf,
     -10,
     10,
     main = "shoulder Priority 2 Entry Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.shoulder.p2.enter.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "shoulder Priority 2 Entry Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")



# PRIORITY ALL EXIT

mean(fc.shoulder.pALL.exit.res)
sd(fc.shoulder.pALL.exit.res)
max(fc.shoulder.pALL.exit.res)
min(fc.shoulder.pALL.exit.res)

density( fc.shoulder.pALL.exit.res )
ecdf( fc.shoulder.pALL.exit.res )

fc.shoulder.pALL.exit.res.cdf <- function(x) {
  cdf <- ecdf( fc.shoulder.pALL.exit.res )
  return( cdf(x) )
}

fc.shoulder.pALL.exit.res.inversecdf <- function(x) {
  try.seq <- seq(from = -10, to = 10, by = 1)

  for (t in try.seq) {
    cdf.value <- fc.shoulder.pALL.exit.res.cdf(t)
    if (cdf.value >= x) {
      return (t)
    }
  }

  return (max(try.seq))
}

plot(fc.shoulder.pALL.exit.res,
     main = "shoulder Priority ALL Exit Forecast Residual Series",
     xlab = "Time",
     ylab = "Residual")

plot(density( fc.shoulder.pALL.exit.res ),
     main = "shoulder Priority ALL Exit Forecast Residual Distribution",
     xlab = "Residual",
```

```r
     ylab = "Probability")

plot(ecdf( fc.shoulder.pALL.exit.res ),
     main = "shoulder Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

plot(fc.shoulder.pALL.exit.res.cdf,
     -10,
     10,
     main = "shoulder Priority ALL Exit Forecast Residual CDF",
     xlab = "Residual",
     ylab = "CDF")

barplot(fc.shoulder.pALL.exit.res.cdf( seq(from = -10, to = 10, by = 1) ),
        main = "shoulder Priority ALL Exit Forecast Residual Discretised CDF",
        names.arg = seq(from = -10, to = 10, by = 1),
        xlab = "Residual",
        ylab = "CDF")





# ----- Sampling from DTRN Predictive Distribution -----

# generate.p1.enter.CDF.sample <- function() {
#   p      <- runif(1, min = 0, max = 1)
#   sample <- fc.footankle.p1.enter.res.inversecdf(p)
#   return (sample)
# }
#
# generate.p2.enter.CDF.sample <- function() {
#   p      <- runif(1, min = 0, max = 1)
#   sample <- fc.footankle.p2.enter.res.inversecdf(p)
#   return (sample)
# }
#
# generate.pALL.exit.CDF.sample <- function() {
#   p      <- runif(1, min = 0, max = 1)
#   sample <- fc.footankle.pALL.exit.res.inversecdf(p)
#   return (sample)
# }
#
#
# generate.CDF.sample <- function(priority = NULL) {
#   if (priority == 1) {
#     return ( generate.p1.enter.CDF.sample() )
#   }
#
#   else if (priority == 2) {
#     return ( generate.p2.enter.CDF.sample() )
#   }
#
#   else {
#     return ( generate.pALL.exit.CDF.sample() )
#   }
# }



# ----- Translate CDF Functions into Lookup Tables -----


p.values <- seq(from = 0, to = 1, by = 0.001)


# Foot & Ankle Priority 1 Entry
fc.footankle.p1.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.footankle.p1.enter.res.inversecdf.value  <- fc.footankle.p1.enter.res.inversecdf(p)
  fc.footankle.p1.enter.res.inversecdf.values <- c(fc.footankle.p1.enter.res.inversecdf.values, fc.footankle.p1.enter.res.inversecdf.value)
}
fc.footankle.p1.enter.res.inversecdf.lookup <- data.frame(p.values, fc.footankle.p1.enter.res.inversecdf.values)

# Foot & Ankle Priority 2 Entry
fc.footankle.p2.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.footankle.p2.enter.res.inversecdf.value  <- fc.footankle.p2.enter.res.inversecdf(p)
  fc.footankle.p2.enter.res.inversecdf.values <- c(fc.footankle.p2.enter.res.inversecdf.values, fc.footankle.p2.enter.res.inversecdf.value)
}
fc.footankle.p2.enter.res.inversecdf.lookup <- data.frame(p.values, fc.footankle.p2.enter.res.inversecdf.values)

# Foot & Ankle Priority ALL Exit
fc.footankle.pALL.exit.res.inversecdf.values <- c()
for (p in p.values) {
  fc.footankle.pALL.exit.res.inversecdf.value  <- fc.footankle.pALL.exit.res.inversecdf(p)
  fc.footankle.pALL.exit.res.inversecdf.values <- c(fc.footankle.pALL.exit.res.inversecdf.values, fc.footankle.pALL.exit.res.inversecdf.value)
}
fc.footankle.pALL.exit.res.inversecdf.lookup <- data.frame(p.values, fc.footankle.pALL.exit.res.inversecdf.values)
```

```
# Hand Priority 1 Entry
fc.hand.p1.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hand.p1.enter.res.inversecdf.value  <- fc.hand.p1.enter.res.inversecdf(p)
  fc.hand.p1.enter.res.inversecdf.values <- c(fc.hand.p1.enter.res.inversecdf.values, fc.hand.p1.enter.res.inversecdf.value)
}
fc.hand.p1.enter.res.inversecdf.lookup <- data.frame(p.values, fc.hand.p1.enter.res.inversecdf.values)

# Hand Priority 2 Entry
fc.hand.p2.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hand.p2.enter.res.inversecdf.value  <- fc.hand.p2.enter.res.inversecdf(p)
  fc.hand.p2.enter.res.inversecdf.values <- c(fc.hand.p2.enter.res.inversecdf.values, fc.hand.p2.enter.res.inversecdf.value)
}
fc.hand.p2.enter.res.inversecdf.lookup <- data.frame(p.values, fc.hand.p2.enter.res.inversecdf.values)

# Hand Priority ALL Exit
fc.hand.pALL.exit.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hand.pALL.exit.res.inversecdf.value  <- fc.hand.pALL.exit.res.inversecdf(p)
  fc.hand.pALL.exit.res.inversecdf.values <- c(fc.hand.pALL.exit.res.inversecdf.values, fc.hand.pALL.exit.res.inversecdf.value)
}
fc.hand.pALL.exit.res.inversecdf.lookup <- data.frame(p.values, fc.hand.pALL.exit.res.inversecdf.values)



# hip Priority 1 Entry
fc.hip.p1.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hip.p1.enter.res.inversecdf.value  <- fc.hip.p1.enter.res.inversecdf(p)
  fc.hip.p1.enter.res.inversecdf.values <- c(fc.hip.p1.enter.res.inversecdf.values, fc.hip.p1.enter.res.inversecdf.value)
}
fc.hip.p1.enter.res.inversecdf.lookup <- data.frame(p.values, fc.hip.p1.enter.res.inversecdf.values)

# hip Priority 2 Entry
fc.hip.p2.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hip.p2.enter.res.inversecdf.value  <- fc.hip.p2.enter.res.inversecdf(p)
  fc.hip.p2.enter.res.inversecdf.values <- c(fc.hip.p2.enter.res.inversecdf.values, fc.hip.p2.enter.res.inversecdf.value)
}
fc.hip.p2.enter.res.inversecdf.lookup <- data.frame(p.values, fc.hip.p2.enter.res.inversecdf.values)

# hip Priority ALL Exit
fc.hip.pALL.exit.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hip.pALL.exit.res.inversecdf.value  <- fc.hip.pALL.exit.res.inversecdf(p)
  fc.hip.pALL.exit.res.inversecdf.values <- c(fc.hip.pALL.exit.res.inversecdf.values, fc.hip.pALL.exit.res.inversecdf.value)
}
fc.hip.pALL.exit.res.inversecdf.lookup <- data.frame(p.values, fc.hip.pALL.exit.res.inversecdf.values)



# hipknee Priority 1 Entry
fc.hipknee.p1.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hipknee.p1.enter.res.inversecdf.value  <- fc.hipknee.p1.enter.res.inversecdf(p)
  fc.hipknee.p1.enter.res.inversecdf.values <- c(fc.hipknee.p1.enter.res.inversecdf.values, fc.hipknee.p1.enter.res.inversecdf.value)
}
fc.hipknee.p1.enter.res.inversecdf.lookup <- data.frame(p.values, fc.hipknee.p1.enter.res.inversecdf.values)

# hipknee Priority 2 Entry
fc.hipknee.p2.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hipknee.p2.enter.res.inversecdf.value  <- fc.hipknee.p2.enter.res.inversecdf(p)
  fc.hipknee.p2.enter.res.inversecdf.values <- c(fc.hipknee.p2.enter.res.inversecdf.values, fc.hipknee.p2.enter.res.inversecdf.value)
}
fc.hipknee.p2.enter.res.inversecdf.lookup <- data.frame(p.values, fc.hipknee.p2.enter.res.inversecdf.values)

# hipknee Priority ALL Exit
fc.hipknee.pALL.exit.res.inversecdf.values <- c()
for (p in p.values) {
  fc.hipknee.pALL.exit.res.inversecdf.value  <- fc.hipknee.pALL.exit.res.inversecdf(p)
  fc.hipknee.pALL.exit.res.inversecdf.values <- c(fc.hipknee.pALL.exit.res.inversecdf.values, fc.hipknee.pALL.exit.res.inversecdf.value)
}
fc.hipknee.pALL.exit.res.inversecdf.lookup <- data.frame(p.values, fc.hipknee.pALL.exit.res.inversecdf.values)



# knee Priority 1 Entry
fc.knee.p1.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.knee.p1.enter.res.inversecdf.value  <- fc.knee.p1.enter.res.inversecdf(p)
  fc.knee.p1.enter.res.inversecdf.values <- c(fc.knee.p1.enter.res.inversecdf.values, fc.knee.p1.enter.res.inversecdf.value)
}
fc.knee.p1.enter.res.inversecdf.lookup <- data.frame(p.values, fc.knee.p1.enter.res.inversecdf.values)

# knee Priority 2 Entry
fc.knee.p2.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.knee.p2.enter.res.inversecdf.value  <- fc.knee.p2.enter.res.inversecdf(p)
  fc.knee.p2.enter.res.inversecdf.values <- c(fc.knee.p2.enter.res.inversecdf.values, fc.knee.p2.enter.res.inversecdf.value)
}
fc.knee.p2.enter.res.inversecdf.lookup <- data.frame(p.values, fc.knee.p2.enter.res.inversecdf.values)

# knee Priority ALL Exit
fc.knee.pALL.exit.res.inversecdf.values <- c()
for (p in p.values) {
  fc.knee.pALL.exit.res.inversecdf.value  <- fc.knee.pALL.exit.res.inversecdf(p)
  fc.knee.pALL.exit.res.inversecdf.values <- c(fc.knee.pALL.exit.res.inversecdf.values, fc.knee.pALL.exit.res.inversecdf.value)
}
fc.knee.pALL.exit.res.inversecdf.lookup <- data.frame(p.values, fc.knee.pALL.exit.res.inversecdf.values)



# shoulder Priority 1 Entry
fc.shoulder.p1.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.shoulder.p1.enter.res.inversecdf.value  <- fc.shoulder.p1.enter.res.inversecdf(p)
  fc.shoulder.p1.enter.res.inversecdf.values <- c(fc.shoulder.p1.enter.res.inversecdf.values, fc.shoulder.p1.enter.res.inversecdf.value)
}
fc.shoulder.p1.enter.res.inversecdf.lookup <- data.frame(p.values, fc.shoulder.p1.enter.res.inversecdf.values)

# shoulder Priority 2 Entry
fc.shoulder.p2.enter.res.inversecdf.values <- c()
for (p in p.values) {
  fc.shoulder.p2.enter.res.inversecdf.value  <- fc.shoulder.p2.enter.res.inversecdf(p)
  fc.shoulder.p2.enter.res.inversecdf.values <- c(fc.shoulder.p2.enter.res.inversecdf.values, fc.shoulder.p2.enter.res.inversecdf.value)
}
fc.shoulder.p2.enter.res.inversecdf.lookup <- data.frame(p.values, fc.shoulder.p2.enter.res.inversecdf.values)

# shoulder Priority ALL Exit
fc.shoulder.pALL.exit.res.inversecdf.values <- c()
for (p in p.values) {
  fc.shoulder.pALL.exit.res.inversecdf.value  <- fc.shoulder.pALL.exit.res.inversecdf(p)
  fc.shoulder.pALL.exit.res.inversecdf.values <- c(fc.shoulder.pALL.exit.res.inversecdf.values, fc.shoulder.pALL.exit.res.inversecdf.value)
}
fc.shoulder.pALL.exit.res.inversecdf.lookup <- data.frame(p.values, fc.shoulder.pALL.exit.res.inversecdf.values)




# ----- Write CDF functions to CSV files -----

setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Forecasting/CDFLookups")

write.csv(fc.footankle.p1.enter.res.inversecdf.lookup,  file = "foot_ankle_priority_1_residual_cdf.csv")
write.csv(fc.footankle.p2.enter.res.inversecdf.lookup,  file = "foot_ankle_priority_2_residual_cdf.csv")
write.csv(fc.footankle.pALL.exit.res.inversecdf.lookup, file = "foot_ankle_priority_ALL_residual_cdf.csv")

write.csv(fc.hand.p1.enter.res.inversecdf.lookup,  file = "hand_priority_1_residual_cdf.csv")
write.csv(fc.hand.p2.enter.res.inversecdf.lookup,  file = "hand_priority_2_residual_cdf.csv")
write.csv(fc.hand.pALL.exit.res.inversecdf.lookup, file = "hand_priority_ALL_residual_cdf.csv")

write.csv(fc.hip.p1.enter.res.inversecdf.lookup,  file = "hip_priority_1_residual_cdf.csv")
write.csv(fc.hip.p2.enter.res.inversecdf.lookup,  file = "hip_priority_2_residual_cdf.csv")
write.csv(fc.hip.pALL.exit.res.inversecdf.lookup, file = "hip_priority_ALL_residual_cdf.csv")

write.csv(fc.hipknee.p1.enter.res.inversecdf.lookup,  file = "hip_knee_priority_1_residual_cdf.csv")
write.csv(fc.hipknee.p2.enter.res.inversecdf.lookup,  file = "hip_knee_priority_2_residual_cdf.csv")
write.csv(fc.hipknee.pALL.exit.res.inversecdf.lookup, file = "hip_knee_priority_ALL_residual_cdf.csv")

write.csv(fc.knee.p1.enter.res.inversecdf.lookup,  file = "knee_priority_1_residual_cdf.csv")
write.csv(fc.knee.p2.enter.res.inversecdf.lookup,  file = "knee_priority_2_residual_cdf.csv")
write.csv(fc.knee.pALL.exit.res.inversecdf.lookup, file = "knee_priority_ALL_residual_cdf.csv")

write.csv(fc.shoulder.p1.enter.res.inversecdf.lookup,  file = "shoulder_priority_1_residual_cdf.csv")
write.csv(fc.shoulder.p2.enter.res.inversecdf.lookup,  file = "shoulder_priority_2_residual_cdf.csv")
write.csv(fc.shoulder.pALL.exit.res.inversecdf.lookup, file = "shoulder_priority_ALL_residual_cdf.csv")




# ----- Post-amble -----

# reset warning level
options(warn = oldw)

message("complete <3")
```

B.3 Generating Queue Clear Time Estimates

This RStudio workbook performs all necessary forecasting, simulation and result generation for determining queue clear-times for the “incompletes” data-set of currently backlogged referrals.

```r
# ***************************************
# The doctor will see you now - Personalised Waiting Times
#
# Predictive Tool for Estimating Queue Clear Times
#
# Emma Tarmey
# 09/08/2022
# ***************************************


# ----- Preamble -----
library(dplyr)
library(stringr)
library(ggplot2)
library(tidyverse)
library(forecast)

rm(list = ls())
setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/ProcessedData")


# ----- Load data from Excel Sheets -----


# backlog queue data

setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/ProcessedData")
incompletes_processed_data <- read.csv("incompletes_processed.csv")
clock_stops_processed_data <- read.csv("clock_stops_processed.csv")


# error cdf data

setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/CDFLookups")

cdf_foot_ankle_priority_2_lookup   <- read.csv("foot_ankle_priority_2_residual_cdf.csv")
cdf_foot_ankle_priority_ALL_lookup <- read.csv("foot_ankle_priority_ALL_residual_cdf.csv")

cdf_hand_priority_2_lookup   <- read.csv("hand_priority_2_residual_cdf.csv")
cdf_hand_priority_ALL_lookup <- read.csv("hand_priority_ALL_residual_cdf.csv")

cdf_hip_priority_2_lookup   <- read.csv("hip_priority_2_residual_cdf.csv")
cdf_hip_priority_ALL_lookup <- read.csv("hip_priority_ALL_residual_cdf.csv")

cdf_hip_knee_priority_2_lookup   <- read.csv("hip_knee_priority_2_residual_cdf.csv")
cdf_hip_knee_priority_ALL_lookup <- read.csv("hip_knee_priority_ALL_residual_cdf.csv")

cdf_knee_priority_2_lookup   <- read.csv("knee_priority_2_residual_cdf.csv")
cdf_knee_priority_ALL_lookup <- read.csv("knee_priority_ALL_residual_cdf.csv")

cdf_shoulder_priority_2_lookup   <- read.csv("shoulder_priority_2_residual_cdf.csv")
cdf_shoulder_priority_ALL_lookup <- read.csv("shoulder_priority_ALL_residual_cdf.csv")



# timeseries data

setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/ProcessedData/TimeSeries")

ts_foot_ankle_priority_2_enter_by_week   <- read.csv("foot_ankle_priority_2_enter_by_week_timeseries.csv")
ts_foot_ankle_priority_all_exit_by_week  <- read.csv("foot_ankle_priority_all_exit_by_week_timeseries.csv")

ts_hand_priority_2_enter_by_week   <- read.csv("hand_priority_2_enter_by_week_timeseries.csv")
ts_hand_priority_all_exit_by_week  <- read.csv("hand_priority_all_exit_by_week_timeseries.csv")

ts_hip_priority_2_enter_by_week   <- read.csv("hip_priority_2_enter_by_week_timeseries.csv")
ts_hip_priority_all_exit_by_week  <- read.csv("hip_priority_all_exit_by_week_timeseries.csv")

ts_hip_knee_priority_2_enter_by_week   <- read.csv("hip_knee_priority_2_enter_by_week_timeseries.csv")
ts_hip_knee_priority_all_exit_by_week  <- read.csv("hip_knee_priority_all_exit_by_week_timeseries.csv")

ts_knee_priority_2_enter_by_week   <- read.csv("knee_priority_2_enter_by_week_timeseries.csv")
ts_knee_priority_all_exit_by_week  <- read.csv("knee_priority_all_exit_by_week_timeseries.csv")

ts_shoulder_priority_2_enter_by_week   <- read.csv("shoulder_priority_2_enter_by_week_timeseries.csv")
ts_shoulder_priority_all_exit_by_week  <- read.csv("shoulder_priority_all_exit_by_week_timeseries.csv")



# ----- Find correctly corresponding forecasts -----


# set window across all data-sets here
# note: incompletes snapshot taken at 25/06/2022
window_start        <- c(2018, 5)
```

```
window_end          <- c(2021, 10)
weeks_to_skip       <- (4 * 8) # skip to present day
predictions_to_make <- (52 * 2)


# foot_ankle

# convert time/date codes
ts_foot_ankle_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_foot_ankle_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_foot_ankle_priority_2_enter_by_week.timeseries    <- ts(ts_foot_ankle_priority_2_enter_by_week$frequency,
                                                           start     = c(2016, 9),
                                                           end       = c(2022, 5),
                                                           frequency = 52)
# window to subset timeseries
ts_foot_ankle_priority_2_enter_by_week.timeseries <- window(ts_foot_ankle_priority_2_enter_by_week.timeseries,
                                                            start = window_start,
                                                            end   = window_end)
# convert time/date codes
ts_foot_ankle_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_foot_ankle_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_foot_ankle_priority_all_exit_by_week.timeseries    <- ts(ts_foot_ankle_priority_all_exit_by_week$frequency,
                                                            start     = c(2016, 9),
                                                            end       = c(2022, 5),
                                                            frequency = 52)
# window to subset timeseries
ts_foot_ankle_priority_all_exit_by_week.timeseries <- window(ts_foot_ankle_priority_all_exit_by_week.timeseries,
                                                             start = window_start,
                                                             end   = window_end)

# hand

# convert time/date codes
ts_hand_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hand_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hand_priority_2_enter_by_week.timeseries    <- ts(ts_hand_priority_2_enter_by_week$frequency,
                                                     start     = c(2016, 9),
                                                     end       = c(2022, 5),
                                                     frequency = 52)
# window to subset timeseries
ts_hand_priority_2_enter_by_week.timeseries <- window(ts_hand_priority_2_enter_by_week.timeseries,
                                                      start = window_start,
                                                      end   = window_end)
# convert time/date codes
ts_hand_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_hand_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_hand_priority_all_exit_by_week.timeseries    <- ts(ts_hand_priority_all_exit_by_week$frequency,
                                                      start     = c(2016, 9),
                                                      end       = c(2022, 5),
                                                      frequency = 52)
# window to subset timeseries
ts_hand_priority_all_exit_by_week.timeseries <- window(ts_hand_priority_all_exit_by_week.timeseries,
                                                       start = window_start,
                                                       end   = window_end)

# hip

# convert time/date codes
ts_hip_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_priority_2_enter_by_week.timeseries    <- ts(ts_hip_priority_2_enter_by_week$frequency,
                                                    start     = c(2016, 9),
                                                    end       = c(2022, 5),
                                                    frequency = 52)
# window to subset timeseries
ts_hip_priority_2_enter_by_week.timeseries <- window(ts_hip_priority_2_enter_by_week.timeseries,
                                                     start = window_start,
                                                     end   = window_end)
# convert time/date codes
ts_hip_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_priority_all_exit_by_week.timeseries    <- ts(ts_hip_priority_all_exit_by_week$frequency,
                                                     start     = c(2016, 9),
                                                     end       = c(2022, 5),
                                                     frequency = 52)
# window to subset timeseries
ts_hip_priority_all_exit_by_week.timeseries <- window(ts_hip_priority_all_exit_by_week.timeseries,
                                                      start = window_start,
                                                      end   = window_end)

# hip_knee

# convert time/date codes
ts_hip_knee_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_knee_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_knee_priority_2_enter_by_week.timeseries    <- ts(ts_hip_knee_priority_2_enter_by_week$frequency,
                                                         start     = c(2016, 9),
                                                         end       = c(2022, 5),
                                                         frequency = 52)
# window to subset timeseries
ts_hip_knee_priority_2_enter_by_week.timeseries <- window(ts_hip_knee_priority_2_enter_by_week.timeseries,
                                                          start = window_start,
                                                          end   = window_end)
# convert time/date codes
ts_hip_knee_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_hip_knee_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_hip_knee_priority_all_exit_by_week.timeseries    <- ts(ts_hip_knee_priority_all_exit_by_week$frequency,
                                                          start     = c(2016, 9),
                                                          end       = c(2022, 5),
                                                          frequency = 52)
# window to subset timeseries
ts_hip_knee_priority_all_exit_by_week.timeseries <- window(ts_hip_knee_priority_all_exit_by_week.timeseries,
                                                           start = window_start,
                                                           end   = window_end)

# knee

# convert time/date codes
ts_knee_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_knee_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_knee_priority_2_enter_by_week.timeseries    <- ts(ts_knee_priority_2_enter_by_week$frequency,
                                                     start     = c(2016, 9),
                                                     end       = c(2022, 5),
                                                     frequency = 52)
# window to subset timeseries
ts_knee_priority_2_enter_by_week.timeseries <- window(ts_knee_priority_2_enter_by_week.timeseries,
                                                      start = window_start,
                                                      end   = window_end)
# convert time/date codes
ts_knee_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_knee_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_knee_priority_all_exit_by_week.timeseries    <- ts(ts_knee_priority_all_exit_by_week$frequency,
                                                      start     = c(2016, 9),
                                                      end       = c(2022, 5),
                                                      frequency = 52)
# window to subset timeseries
ts_knee_priority_all_exit_by_week.timeseries <- window(ts_knee_priority_all_exit_by_week.timeseries,
                                                       start = window_start,
                                                       end   = window_end)

# shoulder

# convert time/date codes
ts_shoulder_priority_2_enter_by_week$week_starting <- as.Date( as.numeric(word(ts_shoulder_priority_2_enter_by_week$range, 1)), origin = "1900-01-01" )
ts_shoulder_priority_2_enter_by_week.timeseries    <- ts(ts_shoulder_priority_2_enter_by_week$frequency,
                                                         start     = c(2016, 9),
                                                         end       = c(2022, 5),
                                                         frequency = 52)
# window to subset timeseries
ts_shoulder_priority_2_enter_by_week.timeseries <- window(ts_shoulder_priority_2_enter_by_week.timeseries,
                                                          start = window_start,
                                                          end   = window_end)
# convert time/date codes
ts_shoulder_priority_all_exit_by_week$week_starting <- as.Date( as.numeric(word(ts_shoulder_priority_all_exit_by_week$range, 1)), origin = "1900-01-01" )
ts_shoulder_priority_all_exit_by_week.timeseries    <- ts(ts_shoulder_priority_all_exit_by_week$frequency,
                                                          start     = c(2016, 9),
                                                          end       = c(2022, 5),
                                                          frequency = 52)
# window to subset timeseries
ts_shoulder_priority_all_exit_by_week.timeseries <- window(ts_shoulder_priority_all_exit_by_week.timeseries,
                                                           start = window_start,
                                                           end   = window_end)




# ALL FORECAST MODELS

# Build TBATS Model
ts_foot_ankle_priority_2_enter_by_week.tbats <- tbats(ts_foot_ankle_priority_2_enter_by_week.timeseries)
summary(ts_foot_ankle_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_foot_ankle_priority_2_enter_by_week.tbats.forecast <- forecast(ts_foot_ankle_priority_2_enter_by_week.tbats, h = predictions_to_make)
plot(ts_foot_ankle_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Foot & Ankle Priority 2 Queue Entry TBATS Model Prediction")

# Build TBATS Model
ts_foot_ankle_priority_all_exit_by_week.tbats <- tbats(ts_foot_ankle_priority_all_exit_by_week.timeseries)
summary(ts_foot_ankle_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_foot_ankle_priority_all_exit_by_week.tbats.forecast <- forecast(ts_foot_ankle_priority_all_exit_by_week.tbats, h = predictions_to_make)
plot(ts_foot_ankle_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Foot & Ankle Priority All Queue Exit TBATS Model Prediction")


# Build TBATS Model
ts_hand_priority_2_enter_by_week.tbats <- tbats(ts_hand_priority_2_enter_by_week.timeseries)
summary(ts_hand_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_hand_priority_2_enter_by_week.tbats.forecast <- forecast(ts_hand_priority_2_enter_by_week.tbats, h = predictions_to_make)
plot(ts_hand_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hand Priority 2 Queue Entry TBATS Model Prediction")

# Build TBATS Model
ts_hand_priority_all_exit_by_week.tbats <- tbats(ts_hand_priority_all_exit_by_week.timeseries)
summary(ts_hand_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_hand_priority_all_exit_by_week.tbats.forecast <- forecast(ts_hand_priority_all_exit_by_week.tbats, h = predictions_to_make)
plot(ts_hand_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "Hand Priority All Queue Exit TBATS Model Prediction")
```

```r
# Build TBATS Model
ts_hip_priority_2_enter_by_week.tbats <- tbats(ts_hip_priority_2_enter_by_week.timeseries)
summary(ts_hip_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_hip_priority_2_enter_by_week.tbats.forecast <- forecast(ts_hip_priority_2_enter_by_week.tbats, h = predictions_to_make)
plot(ts_hip_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "hip Priority 2 Queue Entry TBATS Model Prediction")

# Build TBATS Model
ts_hip_priority_all_exit_by_week.tbats <- tbats(ts_hip_priority_all_exit_by_week.timeseries)
summary(ts_hip_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_hip_priority_all_exit_by_week.tbats.forecast <- forecast(ts_hip_priority_all_exit_by_week.tbats, h = predictions_to_make)
plot(ts_hip_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "hip Priority All Queue Exit TBATS Model Prediction")


# Build TBATS Model
ts_hip_knee_priority_2_enter_by_week.tbats <- tbats(ts_hip_knee_priority_2_enter_by_week.timeseries)
summary(ts_hip_knee_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_hip_knee_priority_2_enter_by_week.tbats.forecast <- forecast(ts_hip_knee_priority_2_enter_by_week.tbats, h = predictions_to_make)
plot(ts_hip_knee_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "hip_knee Priority 2 Queue Entry TBATS Model Prediction")

# Build TBATS Model
ts_hip_knee_priority_all_exit_by_week.tbats <- tbats(ts_hip_knee_priority_all_exit_by_week.timeseries)
summary(ts_hip_knee_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_hip_knee_priority_all_exit_by_week.tbats.forecast <- forecast(ts_hip_knee_priority_all_exit_by_week.tbats, h = predictions_to_make)
plot(ts_hip_knee_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "hip_knee Priority All Queue Exit TBATS Model Prediction")


# Build TBATS Model
ts_knee_priority_2_enter_by_week.tbats <- tbats(ts_knee_priority_2_enter_by_week.timeseries)
summary(ts_knee_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_knee_priority_2_enter_by_week.tbats.forecast <- forecast(ts_knee_priority_2_enter_by_week.tbats, h = predictions_to_make)
plot(ts_knee_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "knee Priority 2 Queue Entry TBATS Model Prediction")

# Build TBATS Model
ts_knee_priority_all_exit_by_week.tbats <- tbats(ts_knee_priority_all_exit_by_week.timeseries)
summary(ts_knee_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_knee_priority_all_exit_by_week.tbats.forecast <- forecast(ts_knee_priority_all_exit_by_week.tbats, h = predictions_to_make)
plot(ts_knee_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "knee Priority All Queue Exit TBATS Model Prediction")


# Build TBATS Model
ts_shoulder_priority_2_enter_by_week.tbats <- tbats(ts_shoulder_priority_2_enter_by_week.timeseries)
summary(ts_shoulder_priority_2_enter_by_week.tbats)

# Predict TBATS Values
ts_shoulder_priority_2_enter_by_week.tbats.forecast <- forecast(ts_shoulder_priority_2_enter_by_week.tbats, h = predictions_to_make)
plot(ts_shoulder_priority_2_enter_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "shoulder Priority 2 Queue Entry TBATS Model Prediction")

# Build TBATS Model
ts_shoulder_priority_all_exit_by_week.tbats <- tbats(ts_shoulder_priority_all_exit_by_week.timeseries)
summary(ts_shoulder_priority_all_exit_by_week.tbats)

# Predict TBATS Values
ts_shoulder_priority_all_exit_by_week.tbats.forecast <- forecast(ts_shoulder_priority_all_exit_by_week.tbats, h = predictions_to_make)
plot(ts_shoulder_priority_all_exit_by_week.tbats.forecast, xlab = "Date", ylab = "Queue Entry", main = "shoulder Priority All Queue Exit TBATS Model Prediction")




# interpret priority data
incompletes_processed_data$Priority <- as.numeric( word(incompletes_processed_data$Priority, 1) )
incompletes_processed_data <- ( incompletes_processed_data %>% drop_na(c("Priority")) )
incompletes_processed_data <- ( incompletes_processed_data %>% drop_na(c("ServiceArea")) )

# isolate speciality
speciality <- "Foot & Ankle"

queue_subset <- (incompletes_processed_data %>% filter(ServiceArea == speciality))

p <- ggplot(incompletes_processed_data, aes(x = factor(ServiceArea) )) +
     geom_bar(aes(fill = factor(Priority)), position = position_stack(reverse = T), width = 0.5) +
     labs(title = "Incompletes Queueing System", x = "ServiceArea", y = "Number of Backlogged Referrals", fill = "Priority")
p

q <- ggplot(queue_subset, aes(x = factor(ServiceArea) )) +
     geom_bar(aes(fill = factor(Priority)), position = position_stack(reverse = T), width = 0.2) +
     labs(title = "Incompletes Queueing System", x = "ServiceArea", y = "Number of Backlogged Referrals", fill = "Priority")
q

queue_data <- ggplot_build(q)[["data"]][[1]][["count"]]
```

```r
queue_data

if (length(queue_data) == 0) {
  queue_data <- c(0, 0, 0)
}

if (length(queue_data) == 1) {
  queue_data <- c(queue_data, 0, 0)
}

if (length(queue_data) == 2) {
    queue_data <- c(queue_data, 0)
}


# ----- Produce Queue System Summary -----


sample.CDF <- function(CDF = NULL) {

  # standardise CDF columns
  colnames(CDF) <- c("Index", "p.values", "CDF.values")
  CDF           <- (CDF %>% select("p.values", "CDF.values"))

  # set-up sampling variables
  p       <- runif(1, min = 0, max = 1)
  sample  <- 0

  # find first corresponding match to random number p in CDF look-up table
  for (i in (1:length(CDF$p.values))) {
    if (CDF$p.values[i] < p) {
      # do nothing, continue iterating
    }
    else{
      # accept first sample, stop iterating
      sample <- CDF$CDF.values[i]
      break
    }
  }

  return (sample)
}


queue_clear_time <- function(queue_data = NULL,
                             priority   = NULL,
                             enter_priority_2.forecast,
                             enter_priority_2.cdf,
                             exit_priority_all.forecast,
                             exit_priority_all.cdf) {

  # Variables used in simulation
  queue_clear_weeks <- 0
  priority_1_queue  <- queue_data[1]
  priority_2_queue  <- queue_data[2] + queue_data[3] # priority 3 treated as noise but current state still accounted for

  enter_priority_2.cdf.value  <- 0
  exit_priority_all.cdf.value <- 0

  # If patient is priority 2 (emergency) or 3 (2 week wait),
  # then we may ignore the current backlog of priority 1 (routine) patients
  # We treat priority 3 as noise with respect to the forecasts due to rarity (many zeroes)
  # Though, as such patients exist, we may still conclude a very short RTT time

  if (priority == 1) {

    # Clear time for all Priority 1 and Priority 2 backlog
    while ( (priority_1_queue > 0) || (priority_2_queue > 0) ) {

      # sample residual values from pre-computed CDFs
      enter_priority_2.cdf.value  <- sample.CDF( enter_priority_2.cdf  )
      exit_priority_all.cdf.value <- sample.CDF( exit_priority_all.cdf )

      priority_2_queue  <- ( priority_2_queue + (enter_priority_2.forecast[ queue_clear_weeks + 1 ] + enter_priority_2.cdf.value ) - (exit_priority_all.forecast[ queue_clear_weeks +
      priority_1_queue  <- ( priority_1_queue )

      if (priority_2_queue < 0) {
        priority_1_queue <- (priority_1_queue + priority_2_queue)
        priority_2_queue <- 0
      }

      if (priority_1_queue < 0) {
        priority_1_queue <- 0
      }

      # Graph Current State of Queue
      queue_clear_weeks <- queue_clear_weeks + 1
      barplot_data      <- c(priority_1_queue, priority_2_queue, 0)
    }

  }
  else if (priority == 2) {

    # Clear time for all Priority 2 backlog
    while ( priority_2_queue > 0 ) {

      # sample residual values from pre-computed CDFs
      enter_priority_2.cdf.value  <- sample.CDF( enter_priority_2.cdf  )
      exit_priority_all.cdf.value <- sample.CDF( exit_priority_all.cdf )

      priority_2_queue  <- ( priority_2_queue - (exit_priority_all.forecast[ queue_clear_weeks + 1 ] + exit_priority_all.cdf.value) )

      # Graph Current State of Queue
      queue_clear_weeks <- queue_clear_weeks + 1
      barplot_data      <- c(priority_1_queue, priority_2_queue, 0)
    }
  }
  else {
    # Very few patients at this priority level exist, hence cannot really forecast
    # Additional statistical context provided elsewhere
    queue_clear_weeks <- NULL
  }


  return (queue_clear_weeks)
}


# Graph Initial State of Queue
barplot_data      <- queue_data
barplot(barplot_data,
        main      = paste("Current State of Queueing System for Service Area", speciality, "\nWeek = ", 0),
        names.arg = c("Priority 1: \nRoutine", "Priority 2: \nUrgent", "Priority 3: \nTwo-Week Wait"),
        xlab      = "Patient Priority",
        ylab      = "Number of Backlogged Referrals in Queue")

queue_clear_time(queue_data = queue_data,
                 priority   = 1,
                 enter_priority_2.forecast   = as.integer( ts_foot_ankle_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                 enter_priority_2.cdf        = cdf_foot_ankle_priority_2_lookup,
                 exit_priority_all.forecast  = as.integer( ts_foot_ankle_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                 exit_priority_all.cdf       = cdf_foot_ankle_priority_ALL_lookup)

queue_clear_time(queue_data = queue_data,
                 priority   = 2,
                 enter_priority_2.forecast   = as.integer( ts_foot_ankle_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                 enter_priority_2.cdf        = cdf_foot_ankle_priority_2_lookup,
                 exit_priority_all.forecast  = as.integer( ts_foot_ankle_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                 exit_priority_all.cdf       = cdf_foot_ankle_priority_ALL_lookup)



# ----- Simulate Queue Clear n times to produce clear-time distribution -----

simulate_queue <- function(n            = NULL,
                           priority     = NULL,
                           service_area = NULL,
                           enter_priority_2.forecast   = NULL,
                           enter_priority_2.cdf        = NULL,
                           exit_priority_all.forecast  = NULL,
                           exit_priority_all.cdf       = NULL,
                           incompletes_processed_data  = NULL) {

  # setup variables to hold results
  clear_times  <- c()
  current_time <- 0

  # isolate incompletes data to records concerning our service area
  queue_subset <- incompletes_processed_data %>% filter(ServiceArea == service_area)

  # generate initial queue data using ggplot
  q <- ggplot(queue_subset, aes(x = factor(ServiceArea) )) +
    geom_bar(aes(fill = factor(Priority)), position = position_stack(reverse = T), width = 0.2) +
    labs(title = "Incompletes Queueing System", x = "ServiceArea", y = "Number of Backlogged Referrals", fill = "Priority")
  queue_data <- ggplot_build(q)[["data"]][[1]][["count"]]

  # adjust queue system summary data for uniform size across cases
  if (length(queue_data) == 0) {
    queue_data <- c(0, 0, 0)
  }

  if (length(queue_data) == 1) {
    queue_data <- c(queue_data, 0, 0)
  }

  if (length(queue_data) == 2) {
    queue_data <- c(queue_data, 0)
  }

  # generate all n queue clear time results
  for (i in (1:n)) {
    # generate new estimate
    current_time <- queue_clear_time(queue_data = queue_data,
                                     priority   = priority,
                                     enter_priority_2.forecast   = enter_priority_2.forecast,
                                     enter_priority_2.cdf        = enter_priority_2.cdf,
                                     exit_priority_all.forecast  = exit_priority_all.forecast,
                                     exit_priority_all.cdf       = exit_priority_all.cdf)

    # add new estimate to vector
    clear_times <- c(clear_times, current_time)
  }

  return(clear_times)
}



# ----- Generate Results from Repeat Simulation (N=100) -----

foot_ankle_p1_N100_final_results <- simulate_queue(n            = 100,
                                           priority     = 1,
                                           service_area = "Foot & Ankle",
                                           enter_priority_2.forecast   = as.integer( ts_foot_ankle_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                           enter_priority_2.cdf        = cdf_foot_ankle_priority_2_lookup,
                                           exit_priority_all.forecast  = as.integer( ts_foot_ankle_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                           exit_priority_all.cdf       = cdf_foot_ankle_priority_ALL_lookup,
                                           incompletes_processed_data  = incompletes_processed_data)

hand_p1_N100_final_results <- simulate_queue(      n            = 100,
                                           priority     = 1,
                                           service_area = "Hand",
                                           enter_priority_2.forecast   = as.integer( ts_hand_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                           enter_priority_2.cdf        = cdf_hand_priority_2_lookup,
                                           exit_priority_all.forecast  = as.integer( ts_hand_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                           exit_priority_all.cdf       = cdf_hand_priority_ALL_lookup,
                                           incompletes_processed_data  = incompletes_processed_data)

hip_p1_N100_final_results <- simulate_queue(       n            = 100,
                                           priority     = 1,
                                           service_area = "Hip",
                                           enter_priority_2.forecast   = as.integer( ts_hip_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                           enter_priority_2.cdf        = cdf_hip_priority_2_lookup,
                                           exit_priority_all.forecast  = as.integer( ts_hip_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                           exit_priority_all.cdf       = cdf_hip_priority_ALL_lookup,
                                           incompletes_processed_data  = incompletes_processed_data)

hip_knee_p1_N100_final_results <- simulate_queue(  n            = 100,
                                           priority     = 1,
                                           service_area = "Hip & Knee",
                                           enter_priority_2.forecast   = as.integer( ts_hip_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                           enter_priority_2.cdf        = cdf_hip_knee_priority_2_lookup,
                                           exit_priority_all.forecast  = as.integer( ts_hip_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                           exit_priority_all.cdf       = cdf_hip_knee_priority_ALL_lookup,
                                           incompletes_processed_data  = incompletes_processed_data)

knee_p1_N100_final_results <- simulate_queue(       n            = 100,
                                            priority     = 1,
                                            service_area = "Knee",
                                            enter_priority_2.forecast   = as.integer( ts_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                            enter_priority_2.cdf        = cdf_knee_priority_2_lookup,
                                            exit_priority_all.forecast  = as.integer( ts_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                            exit_priority_all.cdf       = cdf_knee_priority_ALL_lookup,
                                            incompletes_processed_data  = incompletes_processed_data)

shoulder_p1_N100_final_results <- simulate_queue(   n            = 100,
                                            priority     = 1,
                                            service_area = "Shoulder",
                                            enter_priority_2.forecast   = as.integer( ts_shoulder_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                            enter_priority_2.cdf        = cdf_shoulder_priority_2_lookup,
                                            exit_priority_all.forecast  = as.integer( ts_shoulder_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                            exit_priority_all.cdf       = cdf_shoulder_priority_ALL_lookup,
                                            incompletes_processed_data  = incompletes_processed_data)

foot_ankle_p2_N100_final_results <- simulate_queue(n            = 100,
                                                   priority     = 2,
                                                   service_area = "Foot & Ankle",
                                                   enter_priority_2.forecast   = as.integer( ts_foot_ankle_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                   enter_priority_2.cdf        = cdf_foot_ankle_priority_2_lookup,
                                                   exit_priority_all.forecast  = as.integer( ts_foot_ankle_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                   exit_priority_all.cdf       = cdf_foot_ankle_priority_ALL_lookup,
                                                   incompletes_processed_data  = incompletes_processed_data)

hand_p2_N100_final_results <- simulate_queue(      n            = 100,
                                                   priority     = 2,
                                                   service_area = "Hand",
                                                   enter_priority_2.forecast   = as.integer( ts_hand_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                   enter_priority_2.cdf        = cdf_hand_priority_2_lookup,
                                                   exit_priority_all.forecast  = as.integer( ts_hand_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                   exit_priority_all.cdf       = cdf_hand_priority_ALL_lookup,
                                                   incompletes_processed_data  = incompletes_processed_data)

hip_p2_N100_final_results <- simulate_queue(       n            = 100,
                                                   priority     = 2,
                                                   service_area = "Hip",
                                                   enter_priority_2.forecast   = as.integer( ts_hip_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                   enter_priority_2.cdf        = cdf_hip_priority_2_lookup,
                                                   exit_priority_all.forecast  = as.integer( ts_hip_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                   exit_priority_all.cdf       = cdf_hip_priority_ALL_lookup,
                                                   incompletes_processed_data  = incompletes_processed_data)

hip_knee_p2_N100_final_results <- simulate_queue(  n            = 100,
                                                   priority     = 2,
                                                   service_area = "Hip & Knee",
                                                   enter_priority_2.forecast   = as.integer( ts_hip_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                   enter_priority_2.cdf        = cdf_hip_knee_priority_2_lookup,
                                                   exit_priority_all.forecast  = as.integer( ts_hip_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                   exit_priority_all.cdf       = cdf_hip_knee_priority_ALL_lookup,
                                                   incompletes_processed_data  = incompletes_processed_data)

knee_p2_N100_final_results <- simulate_queue(       n            = 100,
                                                    priority     = 2,
                                                    service_area = "Knee",
                                                    enter_priority_2.forecast   = as.integer( ts_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_knee_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_knee_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

shoulder_p2_N100_final_results <- simulate_queue(   n            = 100,
                                                    priority     = 2,
                                                    service_area = "Shoulder",
                                                    enter_priority_2.forecast   = as.integer( ts_shoulder_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_shoulder_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_shoulder_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_shoulder_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)



# ----- Save Results to CSV (N=100) -----

setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/N100Results")

write.csv(foot_ankle_p1_N100_final_results,
          file = "foot_ankle_p1_N100_final_results.csv",
          row.names = TRUE)

write.csv(hand_p1_N100_final_results,
          file = "hand_p1_N100_final_results.csv",
          row.names = TRUE)

write.csv(hip_p1_N100_final_results,
          file = "hip_p1_N100_final_results.csv",
          row.names = TRUE)

write.csv(hip_knee_p1_N100_final_results,
          file = "hip_knee_p1_N100_final_results.csv",
          row.names = TRUE)

write.csv(knee_p1_N100_final_results,
          file = "knee_p1_N100_final_results.csv",
          row.names = TRUE)

write.csv(shoulder_p1_N100_final_results,
          file = "shoulder_p1_N100_final_results.csv",
          row.names = TRUE)

write.csv(foot_ankle_p2_N100_final_results,
          file = "foot_ankle_p2_N100_final_results.csv",
          row.names = TRUE)

write.csv(hand_p2_N100_final_results,
          file = "hand_p2_N100_final_results.csv",
          row.names = TRUE)

write.csv(hip_p2_N100_final_results,
          file = "hip_p2_N100_final_results.csv",
          row.names = TRUE)

write.csv(hip_knee_p2_N100_final_results,
          file = "hip_knee_p2_N100_final_results.csv",
          row.names = TRUE)

write.csv(knee_p2_N100_final_results,
          file = "knee_p2_N100_final_results.csv",
          row.names = TRUE)

write.csv(shoulder_p2_N100_final_results,
          file = "shoulder_p2_N100_final_results.csv",
          row.names = TRUE)


# ----- Generate Results from Repeat Simulation (N=1000) -----
```

```r
foot_ankle_p1_N1000_final_results <- simulate_queue(n            = 1000,
                                                    priority     = 1,
                                                    service_area = "Foot & Ankle",
                                                    enter_priority_2.forecast   = as.integer( ts_foot_ankle_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_foot_ankle_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_foot_ankle_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_foot_ankle_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

hand_p1_N1000_final_results <- simulate_queue(      n            = 1000,
                                                    priority     = 1,
                                                    service_area = "Hand",
                                                    enter_priority_2.forecast   = as.integer( ts_hand_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_hand_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_hand_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_hand_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

hip_p1_N1000_final_results <- simulate_queue(       n            = 1000,
                                                    priority     = 1,
                                                    service_area = "Hip",
                                                    enter_priority_2.forecast   = as.integer( ts_hip_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_hip_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_hip_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_hip_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

hip_knee_p1_N1000_final_results <- simulate_queue(  n            = 1000,
                                                    priority     = 1,
                                                    service_area = "Hip & Knee",
                                                    enter_priority_2.forecast   = as.integer( ts_hip_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_hip_knee_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_hip_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_hip_knee_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

knee_p1_N1000_final_results <- simulate_queue(       n            = 1000,
                                                     priority     = 1,
                                                     service_area = "Knee",
                                                     enter_priority_2.forecast   = as.integer( ts_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                     enter_priority_2.cdf        = cdf_knee_priority_2_lookup,
                                                     exit_priority_all.forecast  = as.integer( ts_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                     exit_priority_all.cdf       = cdf_knee_priority_ALL_lookup,
                                                     incompletes_processed_data  = incompletes_processed_data)

shoulder_p1_N1000_final_results <- simulate_queue(   n            = 1000,
                                                     priority     = 1,
                                                     service_area = "Shoulder",
                                                     enter_priority_2.forecast   = as.integer( ts_shoulder_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                     enter_priority_2.cdf        = cdf_shoulder_priority_2_lookup,
                                                     exit_priority_all.forecast  = as.integer( ts_shoulder_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                     exit_priority_all.cdf       = cdf_shoulder_priority_ALL_lookup,
                                                     incompletes_processed_data  = incompletes_processed_data)

foot_ankle_p2_N1000_final_results <- simulate_queue(n            = 1000,
                                                    priority     = 2,
                                                    service_area = "Foot & Ankle",
                                                    enter_priority_2.forecast   = as.integer( ts_foot_ankle_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_foot_ankle_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_foot_ankle_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_foot_ankle_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

hand_p2_N1000_final_results <- simulate_queue(      n            = 1000,
                                                    priority     = 2,
                                                    service_area = "Hand",
                                                    enter_priority_2.forecast   = as.integer( ts_hand_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_hand_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_hand_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_hand_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

hip_p2_N1000_final_results <- simulate_queue(       n            = 1000,
                                                    priority     = 2,
                                                    service_area = "Hip",
                                                    enter_priority_2.forecast   = as.integer( ts_hip_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_hip_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_hip_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_hip_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)

hip_knee_p2_N1000_final_results <- simulate_queue(  n            = 1000,
                                                    priority     = 2,
                                                    service_area = "Hip & Knee",
                                                    enter_priority_2.forecast   = as.integer( ts_hip_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                    enter_priority_2.cdf        = cdf_hip_knee_priority_2_lookup,
                                                    exit_priority_all.forecast  = as.integer( ts_hip_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                    exit_priority_all.cdf       = cdf_hip_knee_priority_ALL_lookup,
                                                    incompletes_processed_data  = incompletes_processed_data)
```

```r
knee_p2_N1000_final_results <- simulate_queue(       n            = 1000,
                                                      priority     = 2,
                                                      service_area = "Knee",
                                                      enter_priority_2.forecast   = as.integer( ts_knee_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                      enter_priority_2.cdf        = cdf_knee_priority_2_lookup,
                                                      exit_priority_all.forecast  = as.integer( ts_knee_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                      exit_priority_all.cdf       = cdf_knee_priority_ALL_lookup,
                                                      incompletes_processed_data  = incompletes_processed_data)

shoulder_p2_N1000_final_results <- simulate_queue(   n            = 1000,
                                                      priority     = 2,
                                                      service_area = "Shoulder",
                                                      enter_priority_2.forecast   = as.integer( ts_shoulder_priority_2_enter_by_week.tbats.forecast$mean   )[-c(1:weeks_to_skip)],
                                                      enter_priority_2.cdf        = cdf_shoulder_priority_2_lookup,
                                                      exit_priority_all.forecast  = as.integer( ts_shoulder_priority_all_exit_by_week.tbats.forecast$mean  )[-c(1:weeks_to_skip)],
                                                      exit_priority_all.cdf       = cdf_shoulder_priority_ALL_lookup,
                                                      incompletes_processed_data  = incompletes_processed_data)



# ----- Save Results to CSV (N=1000) -----

setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/N1000Results")

write.csv(foot_ankle_p1_N1000_final_results,
          file = "foot_ankle_p1_N1000_final_results.csv",
          row.names = TRUE)

write.csv(hand_p1_N1000_final_results,
          file = "hand_p1_N1000_final_results.csv",
          row.names = TRUE)

write.csv(hip_p1_N1000_final_results,
          file = "hip_p1_N1000_final_results.csv",
          row.names = TRUE)

write.csv(hip_knee_p1_N1000_final_results,
          file = "hip_knee_p1_N1000_final_results.csv",
          row.names = TRUE)

write.csv(knee_p1_N1000_final_results,
          file = "knee_p1_N1000_final_results.csv",
          row.names = TRUE)

write.csv(shoulder_p1_N1000_final_results,
          file = "shoulder_p1_N1000_final_results.csv",
          row.names = TRUE)

write.csv(foot_ankle_p2_N1000_final_results,
          file = "foot_ankle_p2_N1000_final_results.csv",
          row.names = TRUE)

write.csv(hand_p2_N1000_final_results,
          file = "hand_p2_N1000_final_results.csv",
          row.names = TRUE)

write.csv(hip_p2_N1000_final_results,
          file = "hip_p2_N1000_final_results.csv",
          row.names = TRUE)

write.csv(hip_knee_p2_N1000_final_results,
          file = "hip_knee_p2_N1000_final_results.csv",
          row.names = TRUE)

write.csv(knee_p2_N1000_final_results,
          file = "knee_p2_N1000_final_results.csv",
          row.names = TRUE)

write.csv(shoulder_p2_N1000_final_results,
          file = "shoulder_p2_N1000_final_results.csv",
          row.names = TRUE)
```

B.4 Presenting Results

This RStudio workbook requires the queue clear time estimates generated by the predictive tool itself. It then generates all requires statistical summaries and graphical plots used in the narrative of this report's conclusions.

```
# ***************************************
# The doctor will see you now - Personalised Waiting Times
#
# Presenting Results from Predictive Tool Simulation
#
# Emma Tarmey
# 05/09/2022
# ***************************************


# ----- Preamble -----
library(dplyr)
library(stringr)
library(ggplot2)
library(tidyverse)
library(forecast)

rm(list = ls())
setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/ProcessedData")


# ----- Load data from Excel Sheets -----


# backlog queue data
setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/ProcessedData")
incompletes_processed_data <- read.csv("incompletes_processed.csv")
clock_stops_processed_data <- read.csv("clock_stops_processed.csv")


# results of simulation (N = 100)
setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/N100Results")
foot_ankle_p1_N100_final_results <- read.csv("foot_ankle_p1_N100_final_results.csv")$x
hand_p1_N100_final_results       <- read.csv("hand_p1_N100_final_results.csv")$x
hip_p1_N100_final_results        <- read.csv("hip_p1_N100_final_results.csv")$x
hip_knee_p1_N100_final_results   <- read.csv("hip_knee_p1_N100_final_results.csv")$x
knee_p1_N100_final_results       <- read.csv("knee_p1_N100_final_results.csv")$x
shoulder_p1_N100_final_results   <- read.csv("shoulder_p1_N100_final_results.csv")$x

p1_N100_final_results <- data.frame(foot_ankle_p1_N100_final_results,
                                    hand_p1_N100_final_results,
                                    hip_knee_p1_N100_final_results,
                                    hip_knee_p1_N100_final_results,
                                    knee_p1_N100_final_results,
                                    shoulder_p1_N100_final_results)

foot_ankle_p2_N100_final_results <- read.csv("foot_ankle_p2_N100_final_results.csv")$x
hand_p2_N100_final_results       <- read.csv("hand_p2_N100_final_results.csv")$x
hip_p2_N100_final_results        <- read.csv("hip_p2_N100_final_results.csv")$x
hip_knee_p2_N100_final_results   <- read.csv("hip_knee_p2_N100_final_results.csv")$x
knee_p2_N100_final_results       <- read.csv("knee_p2_N100_final_results.csv")$x
shoulder_p2_N100_final_results   <- read.csv("shoulder_p2_N100_final_results.csv")$x

p2_N100_final_results <- data.frame(foot_ankle_p2_N100_final_results,
                                    hand_p2_N100_final_results,
                                    hip_knee_p2_N100_final_results,
                                    hip_knee_p2_N100_final_results,
                                    knee_p2_N100_final_results,
                                    shoulder_p2_N100_final_results)


# results of simulation (N = 1000)
setwd("C:/Users/Tarmi/Desktop/University Notes/Semester 3/CORMSIS Project/Predictive Tool/N1000Results")
foot_ankle_p1_N1000_final_results <- read.csv("foot_ankle_p1_N1000_final_results.csv")$x
hand_p1_N1000_final_results       <- read.csv("hand_p1_N1000_final_results.csv")$x
hip_p1_N1000_final_results        <- read.csv("hip_p1_N1000_final_results.csv")$x
hip_knee_p1_N1000_final_results   <- read.csv("hip_knee_p1_N1000_final_results.csv")$x
knee_p1_N1000_final_results       <- read.csv("knee_p1_N1000_final_results.csv")$x
shoulder_p1_N1000_final_results   <- read.csv("shoulder_p1_N1000_final_results.csv")$x

p1_N1000_final_results <- data.frame(foot_ankle_p1_N1000_final_results,
                                     hand_p1_N1000_final_results,
                                     hip_knee_p1_N1000_final_results,
                                     hip_knee_p1_N1000_final_results,
                                     knee_p1_N1000_final_results,
                                     shoulder_p1_N1000_final_results)

foot_ankle_p2_N1000_final_results <- read.csv("foot_ankle_p2_N1000_final_results.csv")$x
hand_p2_N1000_final_results       <- read.csv("hand_p2_N1000_final_results.csv")$x
hip_p2_N1000_final_results        <- read.csv("hip_p2_N1000_final_results.csv")$x
hip_knee_p2_N1000_final_results   <- read.csv("hip_knee_p2_N1000_final_results.csv")$x
knee_p2_N1000_final_results       <- read.csv("knee_p2_N1000_final_results.csv")$x
shoulder_p2_N1000_final_results   <- read.csv("shoulder_p2_N1000_final_results.csv")$x
```

```r
p2_N1000_final_results <- data.frame(foot_ankle_p2_N1000_final_results,
                                     hand_p2_N1000_final_results,
                                     hip_knee_p2_N1000_final_results,
                                     hip_knee_p2_N1000_final_results,
                                     knee_p2_N1000_final_results,
                                     shoulder_p2_N1000_final_results)


# ----- Summaries of Results -----

summary( foot_ankle_p1_N100_final_results)
summary( hand_p1_N100_final_results)
summary( hip_p1_N100_final_results)
summary( hip_knee_p1_N100_final_results)
summary( knee_p1_N100_final_results)
summary( shoulder_p1_N100_final_results)

summary( foot_ankle_p2_N100_final_results)
summary( hand_p2_N100_final_results)
summary( hip_p2_N100_final_results)
summary( hip_knee_p2_N100_final_results)
summary( knee_p2_N100_final_results)
summary( shoulder_p2_N100_final_results)

summary( foot_ankle_p1_N1000_final_results)
summary( hand_p1_N1000_final_results)
summary( hip_p1_N1000_final_results)
summary( hip_knee_p1_N1000_final_results)
summary( knee_p1_N1000_final_results)
summary( shoulder_p1_N1000_final_results)

summary( foot_ankle_p2_N1000_final_results)
summary( hand_p2_N1000_final_results)
summary( hip_p2_N1000_final_results)
summary( hip_knee_p2_N1000_final_results)
summary( knee_p2_N1000_final_results)
summary( shoulder_p2_N1000_final_results)




# ----- Graphs of Results -----

colors <- c("Foot & Ankle Queue Clear Time" = "red",
            "Hand Queue Clear Time"         = "blue",
            "Hip Queue Clear Time"          = "green",
            "Hip & Knee Queue Clear Time"   = "yellow",
            "Knee Queue Clear Time"         = "purple",
            "Shoulder Queue Clear Time"     = "black")
ggplot() +
    geom_density(data = p1_N100_final_results,
                 aes(x = foot_ankle_p1_N100_final_results, color="Foot & Ankle Queue Clear Time")) +
    geom_vline(data = p1_N100_final_results,
               aes(xintercept = mean(foot_ankle_p1_N100_final_results)),
               color    = "red",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p1_N100_final_results,
                 aes(x = hand_p1_N100_final_results,       color="Hand Queue Clear Time")) +
    geom_vline(data = p1_N100_final_results,
               aes(xintercept = mean(hand_p1_N100_final_results)),
               color    = "blue",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p1_N100_final_results,
                 aes(x = hip_p1_N100_final_results,        color="Hip Queue Clear Time")) +
    geom_vline(data = p1_N100_final_results,
               aes(xintercept = mean(hip_p1_N100_final_results)),
               color    = "green",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p1_N100_final_results,
                 aes(x = hip_knee_p1_N100_final_results,   color="Hip & Knee Queue Clear Time")) +
    geom_vline(data = p1_N100_final_results,
               aes(xintercept = mean(hip_knee_p1_N100_final_results)),
               color    = "yellow",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p1_N100_final_results,
                 aes(x = knee_p1_N100_final_results,       color="Knee Queue Clear Time")) +
    geom_vline(data = p1_N100_final_results,
               aes(xintercept = mean(knee_p1_N100_final_results)),
               color    = "purple",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p1_N100_final_results,
                 aes(x = shoulder_p1_N100_final_results,   color="Shoulder Queue Clear Time")) +
    geom_vline(data = p1_N100_final_results,
               aes(xintercept = mean(shoulder_p1_N100_final_results)),
               color    = "black",
           linetype = "dashed",
           size     = 0.5) +
  ggtitle( paste( "Referral to Treatment Estimated Queue Clear Time Distributions", "Priority 1 Routine", "(N = 100)" ) ) +
  labs(x = "Overall Waiting Times (weeks)", y = "Density", color = "Legend") +
  scale_color_manual(values = colors)

ggplot() +
  geom_density(data = p2_N100_final_results,
               aes(x = foot_ankle_p2_N100_final_results, color="Foot & Ankle Queue Clear Time")) +
  geom_vline(data = p2_N100_final_results,
           aes(xintercept = mean(foot_ankle_p2_N100_final_results)),
           color    = "red",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p2_N100_final_results,
               aes(x = hand_p2_N100_final_results,       color="Hand Queue Clear Time")) +
  geom_vline(data = p2_N100_final_results,
           aes(xintercept = mean(hand_p2_N100_final_results)),
           color    = "blue",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p2_N100_final_results,
               aes(x = hip_p2_N100_final_results,        color="Hip Queue Clear Time")) +
  geom_vline(data = p2_N100_final_results,
           aes(xintercept = mean(hip_p2_N100_final_results)),
           color    = "green",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p2_N100_final_results,
               aes(x = hip_knee_p2_N100_final_results,   color="Hip & Knee Queue Clear Time")) +
  geom_vline(data = p2_N100_final_results,
           aes(xintercept = mean(hip_knee_p2_N100_final_results)),
           color    = "yellow",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p2_N100_final_results,
               aes(x = knee_p2_N100_final_results,       color="Knee Queue Clear Time")) +
  geom_vline(data = p2_N100_final_results,
           aes(xintercept = mean(knee_p2_N100_final_results)),
           color    = "purple",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p2_N100_final_results,
               aes(x = shoulder_p2_N100_final_results,   color="Shoulder Queue Clear Time")) +
  geom_vline(data = p2_N100_final_results,
           aes(xintercept = mean(shoulder_p2_N100_final_results)),
           color    = "black",
           linetype = "dashed",
           size     = 0.5) +
  ggtitle( paste( "Referral to Treatment Estimated Queue Clear Time Distributions", "Priority 2 Urgent", "(N = 100)" ) ) +
  labs(x = "Overall Waiting Times (weeks)", y = "Density", color = "Legend") +
  scale_color_manual(values = colors)

ggplot() +
  geom_density(data = p1_N1000_final_results,
               aes(x = foot_ankle_p1_N1000_final_results, color="Foot & Ankle Queue Clear Time")) +
  geom_vline(data = p1_N1000_final_results,
           aes(xintercept = mean(foot_ankle_p1_N1000_final_results)),
           color    = "red",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p1_N1000_final_results,
               aes(x = hand_p1_N1000_final_results,       color="Hand Queue Clear Time")) +
  geom_vline(data = p1_N1000_final_results,
           aes(xintercept = mean(hand_p1_N1000_final_results)),
           color    = "blue",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p1_N1000_final_results,
               aes(x = hip_p1_N1000_final_results,        color="Hip Queue Clear Time")) +
  geom_vline(data = p1_N1000_final_results,
           aes(xintercept = mean(hip_p1_N1000_final_results)),
           color    = "green",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p1_N1000_final_results,
               aes(x = hip_knee_p1_N1000_final_results,   color="Hip & Knee Queue Clear Time")) +
  geom_vline(data = p1_N1000_final_results,
           aes(xintercept = mean(hip_knee_p1_N1000_final_results)),
           color    = "yellow",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p1_N1000_final_results,
               aes(x = knee_p1_N1000_final_results,       color="Knee Queue Clear Time")) +
  geom_vline(data = p1_N1000_final_results,
           aes(xintercept = mean(knee_p1_N1000_final_results)),
           color    = "purple",
           linetype = "dashed",
           size     = 0.5) +
  geom_density(data = p1_N1000_final_results,
               aes(x = shoulder_p1_N1000_final_results,   color="Shoulder Queue Clear Time")) +
    geom_vline(data = p1_N1000_final_results,
               aes(xintercept = mean(shoulder_p1_N1000_final_results)),
               color    = "black",
               linetype = "dashed",
               size     = 0.5) +
    ggtitle( paste( "Referral to Treatment Estimated Queue Clear Time Distributions", "Priority 1 Routine", "(N = 1000)" ) ) +
    labs(x = "Overall Waiting Times (weeks)", y = "Density", color = "Legend") +
    scale_color_manual(values = colors)

ggplot() +
    geom_density(data = p2_N1000_final_results,
                 aes(x = foot_ankle_p2_N1000_final_results, color="Foot & Ankle Queue Clear Time")) +
    geom_vline(data = p2_N1000_final_results,
               aes(xintercept = mean(foot_ankle_p2_N1000_final_results)),
               color    = "red",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p2_N1000_final_results,
                 aes(x = hand_p2_N1000_final_results,       color="Hand Queue Clear Time")) +
    geom_vline(data = p2_N1000_final_results,
               aes(xintercept = mean(hand_p2_N1000_final_results)),
               color    = "blue",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p2_N1000_final_results,
                 aes(x = hip_p2_N1000_final_results,        color="Hip Queue Clear Time")) +
    geom_vline(data = p2_N1000_final_results,
               aes(xintercept = mean(hip_p2_N1000_final_results)),
               color    = "green",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p2_N1000_final_results,
                 aes(x = hip_knee_p2_N1000_final_results,   color="Hip & Knee Queue Clear Time")) +
    geom_vline(data = p2_N1000_final_results,
               aes(xintercept = mean(hip_knee_p2_N1000_final_results)),
               color    = "yellow",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p2_N1000_final_results,
                 aes(x = knee_p2_N1000_final_results,       color="Knee Queue Clear Time")) +
    geom_vline(data = p2_N1000_final_results,
               aes(xintercept = mean(knee_p2_N1000_final_results)),
               color    = "purple",
               linetype = "dashed",
               size     = 0.5) +
    geom_density(data = p2_N1000_final_results,
                 aes(x = shoulder_p2_N1000_final_results,   color="Shoulder Queue Clear Time")) +
    geom_vline(data = p2_N1000_final_results,
               aes(xintercept = mean(shoulder_p2_N1000_final_results)),
               color    = "black",
               linetype = "dashed",
               size     = 0.5) +
    ggtitle( paste( "Referral to Treatment Estimated Queue Clear Time Distributions", "Priority 2 Urgent", "(N = 1000)" ) ) +
    labs(x = "Overall Waiting Times (weeks)", y = "Density", color = "Legend") +
    scale_color_manual(values = colors)
```

Appendix C

Software Tool File System

The full specification of the software tool file system is as follows:

```
Pack
├── AllRawData
│   ├── 20220626_Incompletes_anon.xlsx
│   ├── Added to WL.xlsx
│   ├── Clock_stops_TandO_2018_Anon.xlsx
│   ├── Emergency TandO data for Cormsis.xlsx
│   ├── Priorities.xlsx
│   ├── Procedure_decode.xlsx
│   ├── referrals number to TO.xlsx
│   ├── Referrals.xlsx
│   └── staff_names_and_codes.csv
└── CDFLookups
    ├── foot_ankle_priority_1_residual_cdf.csv
    ├── foot_ankle_priority_2_residual_cdf.csv
    ├── foot_ankle_priority_ALL_residual_cdf.csv
    ├── hand_priority_1_residual_cdf.csv
    ├── hand_priority_2_residual_cdf.csv
    ├── hand_priority_ALL_residual_cdf.csv
    ├── hip_priority_1_residual_cdf.csv
    ├── hip_priority_2_residual_cdf.csv
    ├── hip_priority_ALL_residual_cdf.csv
    ├── hip_knee_priority_1_residual_cdf.csv
    ├── hip_knee_priority_2_residual_cdf.csv
    ├── hip_knee_priority_ALL_residual_cdf.csv
    ├── knee_priority_1_residual_cdf.csv
    ├── knee_priority_2_residual_cdf.csv
    ├── knee_priority_ALL_residual_cdf.csv
    ├── shoulder_priority_1_residual_cdf.csv
    ├── shoulder_priority_2_residual_cdf.csv
    └── shoulder_priority_ALL_residual_cdf.csv
```

```
│
├── N100Results
│   ├── foot_ankle_p1_N100_final_results.csv
│   ├── foot_ankle_p2_N100_final_results.csv
│   ├── hand_p1_N100_final_results.csv
│   ├── hand_p2_N100_final_results.csv
│   ├── hip_p1_N100_final_results.csv
│   ├── hip_p2_N100_final_results.csv
│   ├── hip_knee_p1_N100_final_results.csv
│   ├── hip_knee_p2_N100_final_results.csv
│   ├── knee_p1_N100_final_results.csv
│   ├── knee_p2_N100_final_results.csv
│   ├── shoulder_p1_N100_final_results.csv
│   └── shoulder_p2_N100_final_results.csv
├── N1000Results
│   ├── foot_ankle_p1_N1000_final_results.csv
│   ├── foot_ankle_p2_N1000_final_results.csv
│   ├── hand_p1_N1000_final_results.csv
│   ├── hand_p2_N1000_final_results.csv
│   ├── hip_p1_N1000_final_results.csv
│   ├── hip_p2_N1000_final_results.csv
│   ├── hip_knee_p1_N1000_final_results.csv
│   ├── hip_knee_p2_N1000_final_results.csv
│   ├── knee_p1_N1000_final_results.csv
│   ├── knee_p2_N1000_final_results.csv
│   ├── shoulder_p1_N1000_final_results.csv
│   └── shoulder_p2_N1000_final_results.csv
└── ProcessedData
    └── TimeSeries
        ├── foot_ankle_priority_1_enter_by_week_timeseries.csv
        ├── foot_ankle_priority_2_enter_by_week_timeseries.csv
        ├── foot_ankle_priority_3_enter_by_week_timeseries.csv
        ├── foot_ankle_priority_all_exit_by_week_timeseries.csv
        ├── hand_priority_1_enter_by_week_timeseries.csv
        ├── hand_priority_2_enter_by_week_timeseries.csv
        ├── hand_priority_3_enter_by_week_timeseries.csv
        ├── hand_priority_all_exit_by_week_timeseries.csv
        ├── hip_priority_1_enter_by_week_timeseries.csv
        ├── hip_priority_2_enter_by_week_timeseries.csv
        ├── hip_priority_3_enter_by_week_timeseries.csv
        ├── hip_priority_all_exit_by_week_timeseries.csv
        ├── hip_knee_priority_1_enter_by_week_timeseries.csv
        ├── hip_knee_priority_2_enter_by_week_timeseries.csv
        ├── hip_knee_priority_3_enter_by_week_timeseries.csv
        ├── hip_knee_priority_all_exit_by_week_timeseries.csv
        ├── knee_priority_1_enter_by_week_timeseries.csv
        └── knee_priority_2_enter_by_week_timeseries.csv
        knee_priority_3_enter_by_week_timeseries.csv
        knee_priority_all_exit_by_week_timeseries.csv
        shoulder_priority_1_enter_by_week_timeseries.csv
        shoulder_priority_2_enter_by_week_timeseries.csv
        shoulder_priority_3_enter_by_week_timeseries.csv
        shoulder_priority_all_exit_by_week_timeseries.csv
    20220626_Incompletes_anon.csv
    clock_stops_processed.csv
    Clock_stops_TandO_2018_Anon.csv
    Emergency_TandO_data_for_Cormsis.csv
    EPR_Priority_1.csv
    EPR_Priority_2.csv
    EPR_Priority_3.csv
    incompletes_processed.csv
    referrals_by_service.csv
    staff_names_and_codes.csv
ResultsSummary
    p1_N100_final_results_plot.png
    p1_N100_final_results_summary.txt
    p1_N1000_final_results_plot.png
    p1_N1000_final_results_summary.txt
    p2_N100_final_results_plot.png
    p2_N100_final_results_summary.txt
    p2_N1000_final_results_plot.png
    p2_N1000_final_results_summary.txt
FINAL_full_data_pipeline.R
generate_cdfs.R
predictive_tool.R
presenting_results.R
README.txt
```

Appendix D

Details concerning Provided Data-sets

As mentioned, the author is under NDA to not distribute the data provided. This makes conveying their precise nature difficult. As such, a verbose explanation of all columns in the key data-sets is provided here:

- The "clock-stops" data-set, as provided, has fifty-four fields (columns) for each entry:
    1. PseudoID (numerical, treated as label as order not meaningful)
    2. uni_epinumber (numerical, treated as label as order not meaningful)
    3. rtt_id (numerical, treated as label as order not meaningful)
    4. rtt_startdate (date)
    5. rtt_days_wait (numerical)
    6. rtt_weeks_wait (numerical)
    7. rtt_enddate (date)
    8. rtt_end_reason (categorical)
    9. stop_type (categorical)
    10. activity_type (categorical)
    11. event_date (date)
    12. pathway_activity_string_id (categorical)
    13. pathway_activity_integer_id (mainly NULLs)
    14. last_attendance_id (categorical)
    15. last_attendance_date (date)
    16. last_attendance_outcome (categorical)
    17. consultant_code (categorical)
    18. speciality_code (categorical)
    19. awl_id (categorical)
    20. exclusion_reasons (categorical)
    21. waiting_list_type (categorical)
    22. Age_today (numerical)

23. person_stated_gender_code (categorical)
24. ethnic_category (categorical)
25. referral_request_received (date)
26. referral_identifier (categorical)
27. patient_pathway_identifier (categorical)
28. organisation_code_issuer (categorical)
29. source_of_referral (categorical)
30. organisation_identifier_provider (categorical)
31. organisation_site_identifier (categorical)
32. activity_treatment_function_code (categorical)
33. rtt_period_start_date (date0
34. outpatient_future_appointment_date (date)
35. due_date (date)
36. outpatient_appointment_date (date)
37. date_last_attended (date)
38. last_dna_date (date)
39. cancellation_date (date)
40. outcome_of_attendance_code (categorical)
41. tci_date (date)
42. ref_to_treat_period_status (categorical)
43. decision_to_admit_date (date)
44. opcs_code (categorical)
45. admission_method_code (categorical)
46. priority_type_code (categorical)
47. rtt_period_end_date (categorical)
48. procedure_priority_code (categorical)
49. diagnostic_priority_code (categorical)
50. date_of_last_priority_review (categorical)
51. last_pas_validation_performed_by (categorical)
52. date_of_last_priority_review (categorical)
53. last_pas_validation_performed_by (categorical)
54. last_pas_validation_date (date)
55. last_pas_validation_comments (categorical)
56. inclusion_on_cancer_ptl (categorical)

- The "incompletes" data-set, as provided, is largely similar. Except that final RTT times are missing, final RTT dates are missing, outcome fields are missing and new fields concerning more in-depth prioritisation exist.
- The staff service area lookup table relates the staff assigned to a record (consultant_code) to the service area they specialise in.

Appendix E

Command Line Output

E.1 Waiting Time Estimate Distributions

E.1.1 Priority 1 Simulation Results (N = 100)

These distributions specify the distributions found from the corresponding 100 simulations of determining RTT wait time estimates for the given Service Area at Priority level “1 - Routine”.

```
> summary( foot_ankle_p1_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   9.00   13.00   14.00   13.77   15.00   18.00

> summary( hand_p1_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.00    3.00    3.00    3.69    4.00   11.00

> summary( hip_p1_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  23.00   31.00   33.50   34.33   37.25   54.00

> summary( hip_knee_p1_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  27.00   30.00   31.50   31.64   33.00   39.00

> summary( knee_p1_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  38.00   41.00   44.00   44.34   47.00   54.00

> summary( shoulder_p1_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  11.00   12.00   14.00   14.33   16.00   19.00
```

E.1.2 Priority 2 Simulation Results (N = 100)

These distributions specify the distributions found from the corresponding 100 simulations of determining RTT wait time estimates for the given Service Area at Priority level “2 - Urgent”.

```
> summary( foot_ankle_p2_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.00    1.00    1.16    1.00    2.00

> summary( hand_p2_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    1.00    2.00    1.74    2.00    4.00

> summary( hip_p2_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    2.0     4.0     5.0     4.8     6.0     8.0

> summary( hip_knee_p2_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   7.00    9.00    9.00    9.44   10.00   13.00

> summary( knee_p2_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   6.00    8.00    8.00    8.55    9.00   11.00

> summary( shoulder_p2_N100_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2.00    3.00    3.00    3.47    4.00    6.00
```

E.1.3 Priority 1 Simulation Results (N = 1000)

These distributions specify the distributions found from the corresponding 1000 simulations of determining RTT wait time estimates for the given Service Area at Priority level "1 - Routine".

```
> summary( foot_ankle_p1_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    9.0    13.0    14.0    13.8    15.0    21.0 

> summary( hand_p1_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   3.000   3.000   3.601   4.000  11.000 

> summary( hip_p1_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  19.00   31.00   34.00   34.64   38.00   57.00 

> summary( hip_knee_p1_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  26.00   30.75   31.00   31.62   33.00   39.00 

> summary( knee_p1_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  35.00   42.00   44.00   44.72   47.00   58.00 

> summary( shoulder_p1_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   8.00   13.00   15.00   14.65   16.00   24.00 
```

E.1.4 Priority 2 Simulation Results (N = 1000)

These distributions specify the distributions found from the corresponding 1000 simulations of determining RTT wait time estimates for the given Service Area at Priority level “2 - Urgent”.

```
> summary( foot_ankle_p2_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   1.000   1.000   1.149   1.000   3.000 

> summary( hand_p2_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   1.000   1.000   1.625   2.000   6.000 

> summary( hip_p2_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   4.000   4.500   4.695   6.000  12.000 

> summary( hip_knee_p2_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  7.000   9.000   9.000   9.287  10.000  13.000 

> summary( knee_p2_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  5.000   8.000   8.000   8.533   9.000  13.000 

> summary( shoulder_p2_N1000_final_results)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  2.000   3.000   3.000   3.554   4.000   7.000 
```