Modular *shiny* apps for reproducible data communication

Bulukani Mlalazi[1], Anthony Ebert[1,2], Paul Wu[1,2], Kerrie Mengersen[1,2]

[1] School of Mathematical Sciences, Queensland University of Technology (QUT), Australia

[2] ARC Centre of Excellence for Mathematical and Statistical Frontiers, Queensland University of Technology (QUT), Australia

Introduction

Aim: Promote modular software design as a foundation for interactive data-driven apps

Case Study: inbound airport passenger management in the form of an interactive *shiny* web app.

Background: • Interactive visualisation via apps is an effective way to gain insights from complex data exploration and analyses and model interpretation. • However, app complexity typically increases with the number and nature of included features. • The *shiny* R package is a powerful tool for interactive data science communication. It connects the underlying analytical platform to the end users via a responsive web interface and features a built-in modularisation framework to mitigate app development complexity. This helps to manage app behaviour in the reactive code execution framework (Grolemund, 2015) that underpins *shiny*.

Results: *shiny* is an excellent deployment tool to present data-driven solutions implemented in R to broad audiences as it facilitates rapid prototyping and development, and modular app architecture without the need to re-implement core analytics components.

Methodology

• **Build an airport simulation model** to predict changes in queuing behaviour in a network of queues that represent the inbound passenger facilitation process in an international terminal.

• **Implement this in R** (R Core Team, 2017) based on the schema in Wu *et al.* (2014) (**Figure 1**).

• **Select metrics from model outputs** and design a simple graphical visualisation as the basis of our data visualisation framework. Key features: interpretability & consistency.

• **Construct a dashboard app in R** using *shiny* and *shinydashboard*. Leverage the modularisation framework in *shiny* (Cheng, 2017) to manage app complexity. Create a user input module to enable users to modify key input parameters and submit custom simulation forecast requests and understand the potential impact of operational changes.

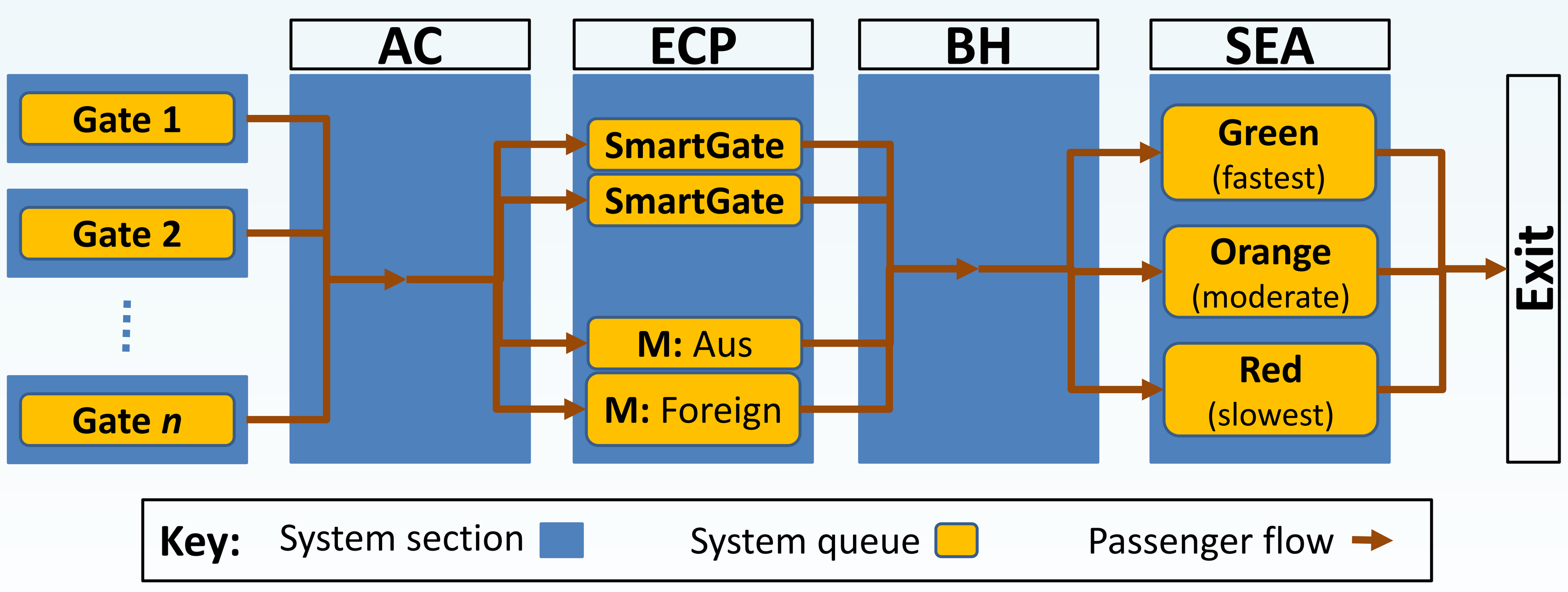

Figure 1: Schema of airport system used to construct simulation model. Simulated passengers enter the system on inbound flights at preassigned arrival gates represented as individual queues modelling passenger disembarkation into the Arrivals Concourse (AC). Transit through the AC to the immigration Entry Control Point (ECP) is defined by a combination of gate queue departure time and simulated mobility demographics randomly assigned to each passenger. Nationality and other routing demographics for automated self-service SmartGate systems or manual processing booths (M) for Australian (Aus) and foreign passport holders, are also computed to assign virtual passengers to one of the four immigration queues. Passengers are randomly assigned zero or more baggage items with associated arrival and screening demographics. The baggage collection process is captured in the Baggage Hall (BH), before passenger screening in the Secondary Examination Area (SEA) whose three queues represent levels of scrutiny and associated delay. For simplicity, SEA processing represents an arbitrary combination of examination by customs or quarantine personnel, after which passengers immediately exit the system.

Results

App development: • Modular design that enabled rapid development of an interactive visualisation app based on a complex data modelling engine; • Robust, reusable and updatable app components; • Individual components that could be seamlessly integrated into a single, navigable design to systematically communicate results to end users.

Data visualisation framework: • The app communicates a system state via two metrics, occupancy (passenger counts) and dwell time (area clearance time). • The single, reusable visualisation module enhances communication and interpretation by preserving important aesthetics app-wide. • Consequently, this produces an app that provides more effective decision support.

Case study context: The situation described in **Figure 2** provides a concrete demonstration of the effectiveness of our app design approach and its potential impact on operational decision making.

The scenario depicts the case of a manager in the ECP who needs to make decisions about delayed flights due to adverse weather at 8:30 am. Using the forecast section, the manager is able to visualise the potential operational impact. Comparing the manager's forecast (**Figure 2c**) to the baseline scenario (**Figure 2a**), delayed arrivals are projected to create substantial congestion between 10:00 am and 12:00 pm. The significant increase in passengers during a relatively slow period is projected to substantially increase passenger wait times. However, the manager is able to quantify the potential disruption and is empowered by this insight to more efficiently allocate the resources at their disposal to pre-empt the adverse outcome.

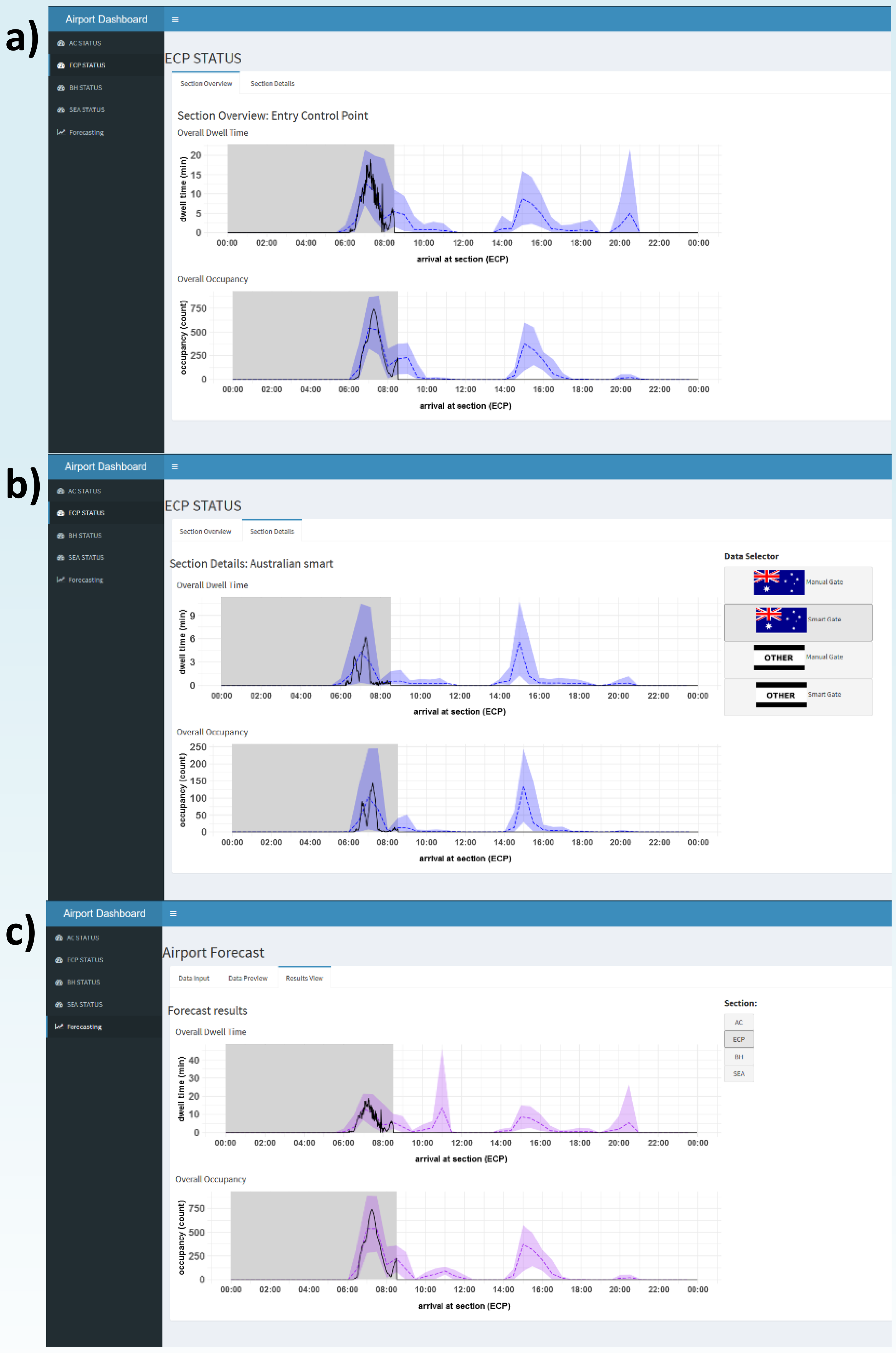

Figure 2: Airport app user interface snapshots illustrating visualisation model Implementation. Data outputs pertaining to the ECP section of the airport simulation model (Figure 1) demonstrate the versatility and consistency of the app's core visualisation module. Metrics of interest are area dwell time (minutes) and occupancy (passenger counts) and the app communicates system-generated forecasts presented as **a)** an overview summary of the entire section and **b)** finer detail for each individual queue route. Specific information is accessible via a linked data selector menu. **c)** A dedicated forecast submission section enables users to submit and view the results of custom forecast scenarios as overview summaries comparable to those generated by the system **a)** to facilitate decision making by scenario evaluation. The depicted user scenario evaluates deviation from the system forecast where 6 flights (1250 passengers approx) scheduled to arrive between 08:30 and 09:30 were each delayed by exactly 60 min. Each time series trace uses common visual elements to facilitate rapid comparability. A central dotted line denotes the expected metric value surrounded by a coloured area representing the prediction interval, set as 99% for development to provide the most representative simulation ensemble. Colour provides distinction between forecasts generated by the system (blue) and user (purple). In addition, the grey area indicates time elapsed and represents 08:30 in all snapshots. The black line indicates the values of live estimates where available with simulated data in use for development purposes.

Discussion

Benefits of app development through modularisation techniques:

- **Improved overall app quality** though focused development of robust app subunits that are integrated into a cohesive product. Compartmentalised app logic is easier to develop and debug, which enhances code reliability and focuses development efforts.
- **Reduction in overall app development time** with faster feature development cycles as modular components are readily reconfigurable for reuse. Further, therefore new features can be developed in isolation with minimal disruption of the existing codebase.
- **Improved data communication outcomes** as careful design of modular visualisation components results in the seamless establishment of consistent visual semantics throughout the app, improving data comparability.
- Methodology is **scalable** and **generalizable**.

Acknowledgements

This work was funded through the Australian Research Council (ARC) linkage grant "Improving the Productivity and Efficiency of Australian Airports" (LP140100282).

References

1. Cheng, J., 2017. Shiny - Modularizing Shiny app code [WWW Document]. URL https://shiny.rstudio.com/articles/modules.html (accessed 10.25.17).
2. Grolemund, G., 2015. Shiny - How to understand reactivity in R [WWW Document]. URL https://shiny.rstudio.com/articles/understanding-reactivity.html (accessed 11.6.17).
3. R Core Team, 2017. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.
4. Wu, P.P.Y., Pitchforth, J., Mengersen, K., 2014. A Hybrid Queue-based Bayesian Network framework for passenger facilitation modelling. Transp. Res. Part C Emerg. Technol. 46, 247–260. doi:10.1016/j.trc.2014.05.005