Installing and Running the Transit Dashboard

The purpose of this document is to provide details as to the steps required necessary to install and update the bus transit dashboard. The document is divided into two parts. In the first, instructions are provided for the installation of the necessary software and interfaces, R and RStudio. In the second, details are provided on the steps required to update different aspects of the application and source data. For further details regarding the project you should visit https://www.cdrc.ac.uk/research/healthymobility/.

Part 1: Dashboard Installation and Setup

Download and setup

The dashboard application has been developed using R and the R Shiny framework. This approach enables the application to incorporate key data-processing capabilities of the R language whilst delivering the content in visually appealing and user-friendly dashboard-based environment. Applications developed using R Shiny may be run locally using the R Studio IDE or served on the web using R Studio Server. The explanations provided here assume that the service will be run locally.

1. Installation of R and R Studio

First, download and install the operating specific version of R for your machine from https://cloud.r-project.org/. Here, we can consider R as the analytical engine.

Second, download and install the operating system specific version of R Studio for your machine. You can use the community edition. RStudio is available from https://www.rstudio.com/products/rstudio/download/#download. Where R provides the analytical capability, R Studio is an integrated development environment favoured by many developers.

Once both are installed, you will now have a usable copy of R on your computer system and the majority of what is needed for the application to run locally.

2. Having installed R and R Studio, the next step is to download the dashboard application from either GitHub or to navigate to the file directory where the application directory on your local machine. The file is approximately 300mb and may take several minutes to download. The directory contains several files and folders. The important files/folders are:

File	Description
app.R	Contains all the R code required to create the user interface and perform the background computation.
data	Folder of subfolders containing all necessary data for building and running the dashboard.

The contents of the file should be identical to the following screenshot.

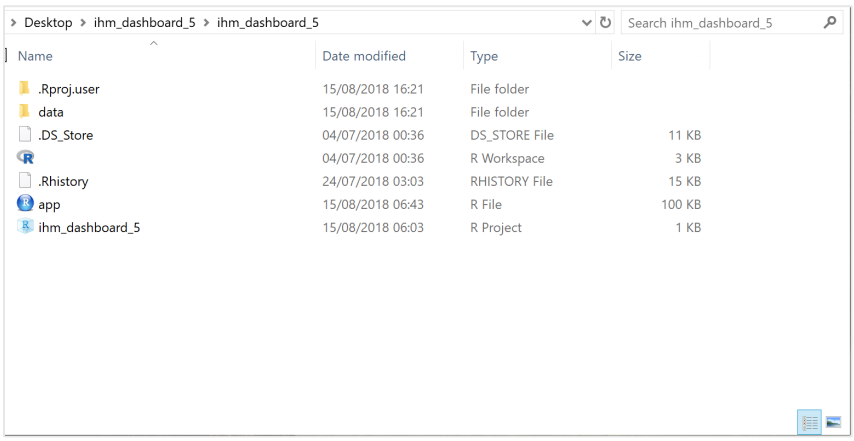


Figure 1: Screenshot of Windows explorer showing files in the application file directory.

Running the Application

Having installed R and R Studio and also downloaded the application, you are now in a position to run the application for the first time. During the first run it is likely that various packages/libraries will have to be updated or installed. Running the application is achieved in two steps. First, the application script is loaded into R. Second, the application is run from R Studio.

To open the application, first open the R Studio application. It is then necessary to open the application script. To do this, from the taskbar, select “File” and then “Open File…”. In the file selector dialogue, navigate to where the file is downloaded and then select and open app.R. The app.R script contains everything necessary to create the dashboard interface and perform the background computation. R Studio should appear identical to the image below.

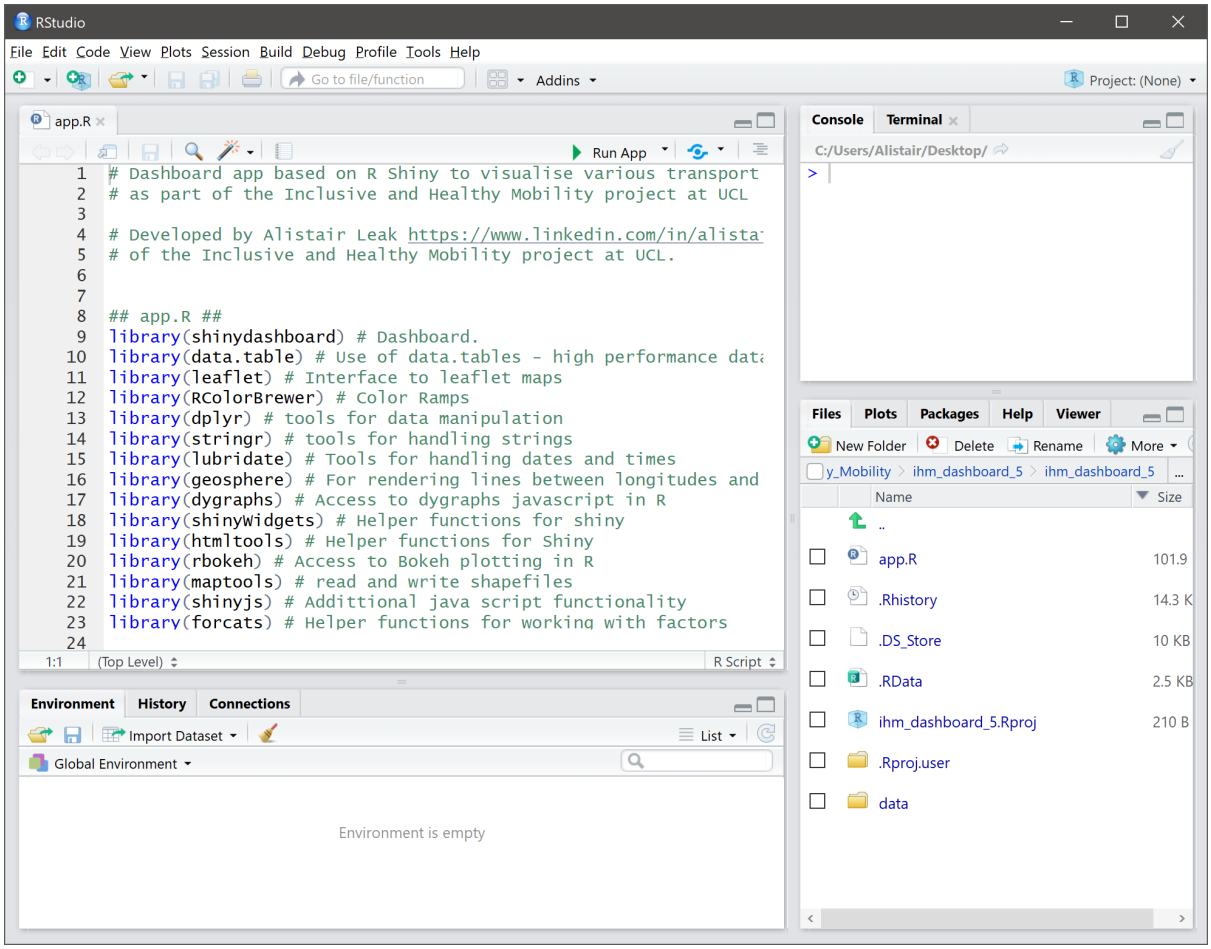


Figure 2: Screenshot of RStudio showing the app.R script open in the script window (top left).

For the application to run, it may be necessary to install or update each of the required libraries (packages). The simplest way to do this is using the R Studio Package installer. To open the package installer from the top taskbar select “Tools” and then “Install Packages…”. You can insert multiple packages to be installed. To speed things up, copy the list of packages below into the install window – but ensure that “Install dependencies” is selected and press “Install”.

shinydashboard, data.table, leaflet, RColorBrewer, dplyr, stringr, lubridate, geosphere, dygraphs, shinyWidgets, htmltools, rbokeh, maptools, shinyjs, forcats, rintrojs

The installation may take several minutes. Note, you will require access to the internet to perform the following installations.

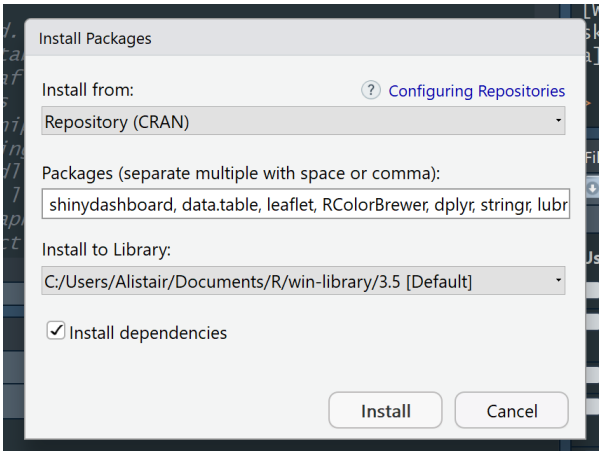


Figure 3: RStudio Package Installer Window. Including comma separated list of required packages.

Assuming the libraries/packages are installed successfully, you are now able to run the dashboard. In the scripts panel, a green play button with the label “Run App” should be visible. Before pressing this, use the small dropdown arrow to specify external. This will mean the Dashboard launches using your default web browser and is running locally.

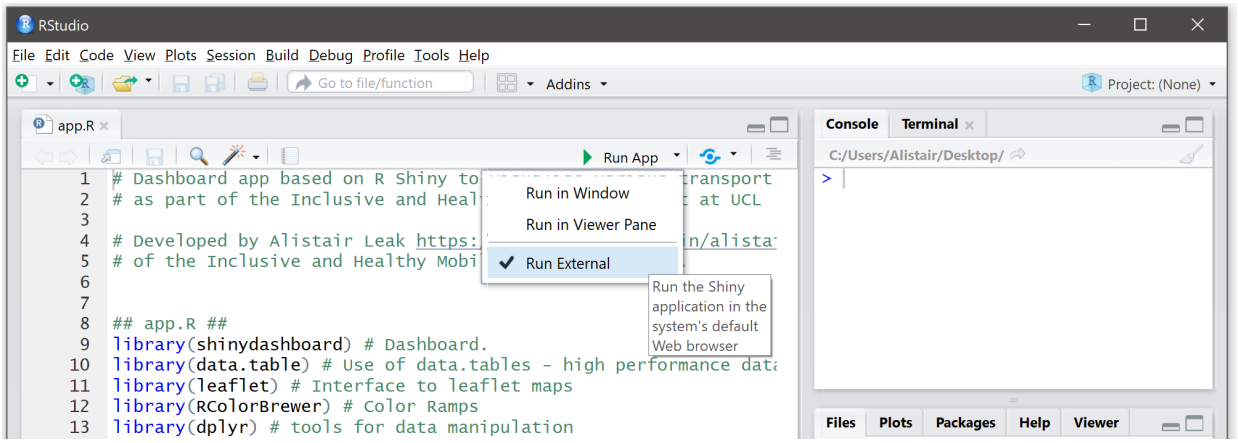


Figure 4: RStudio Interface highlight open script panel and the dropdown to select where the application is to run.

Once you press “Run App”, your web browser will automatically open the dashboard page. A progress bar will appear promptly and indicate the progress of importing data. If the loading fails for any reason, indicated by a greyed our screen, refresh the page. You will notice that the URL of the dashboard indicates that the app is running on the local host (127.0.0.1).

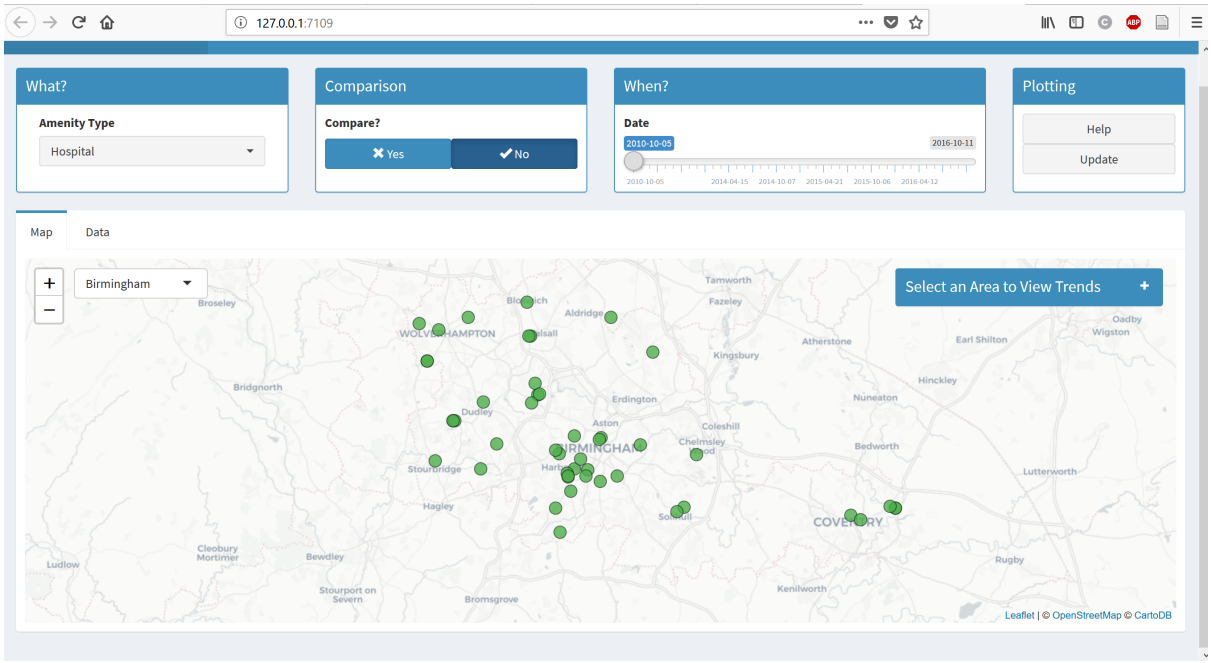


Figure 1: Example page from the application

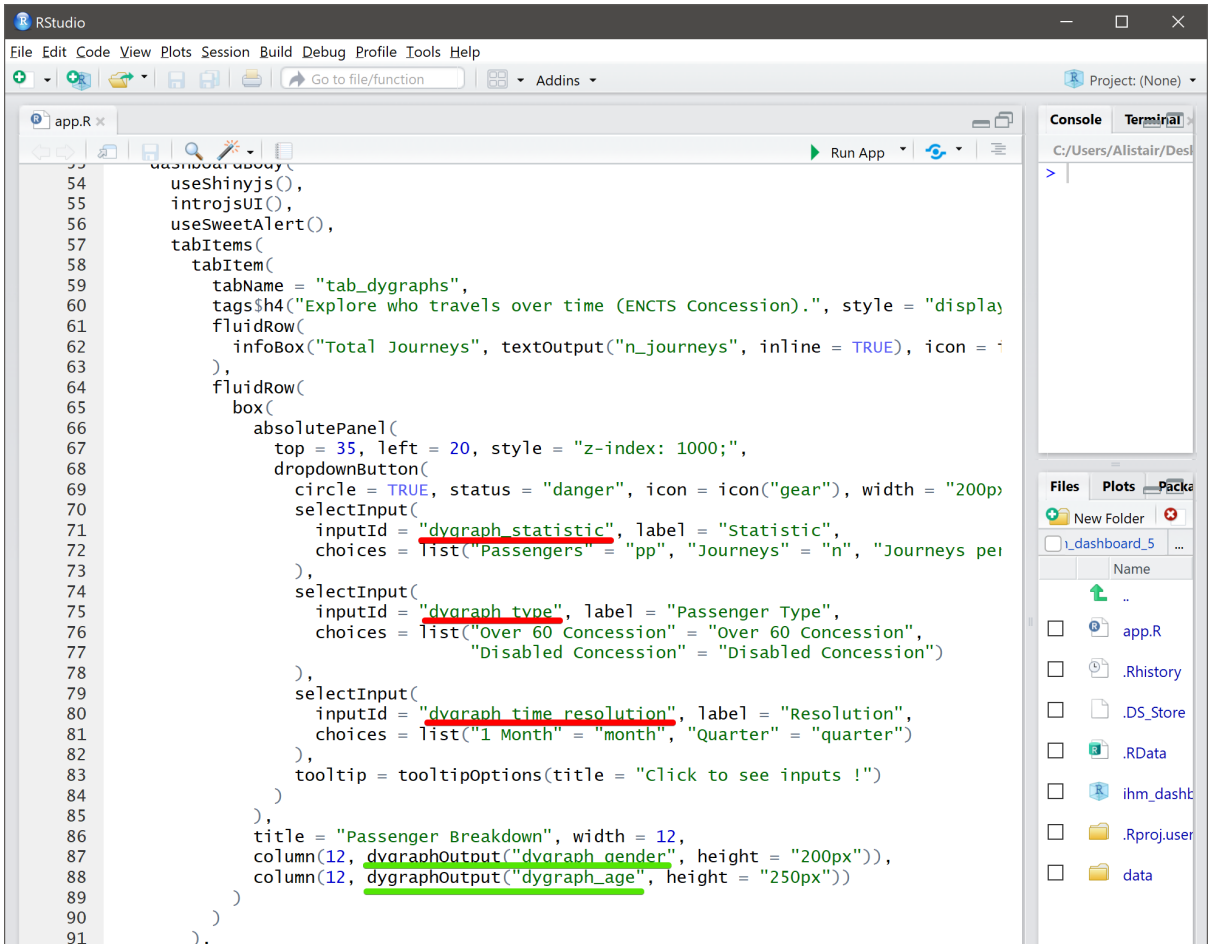
Part 2: Updating the data and user interface

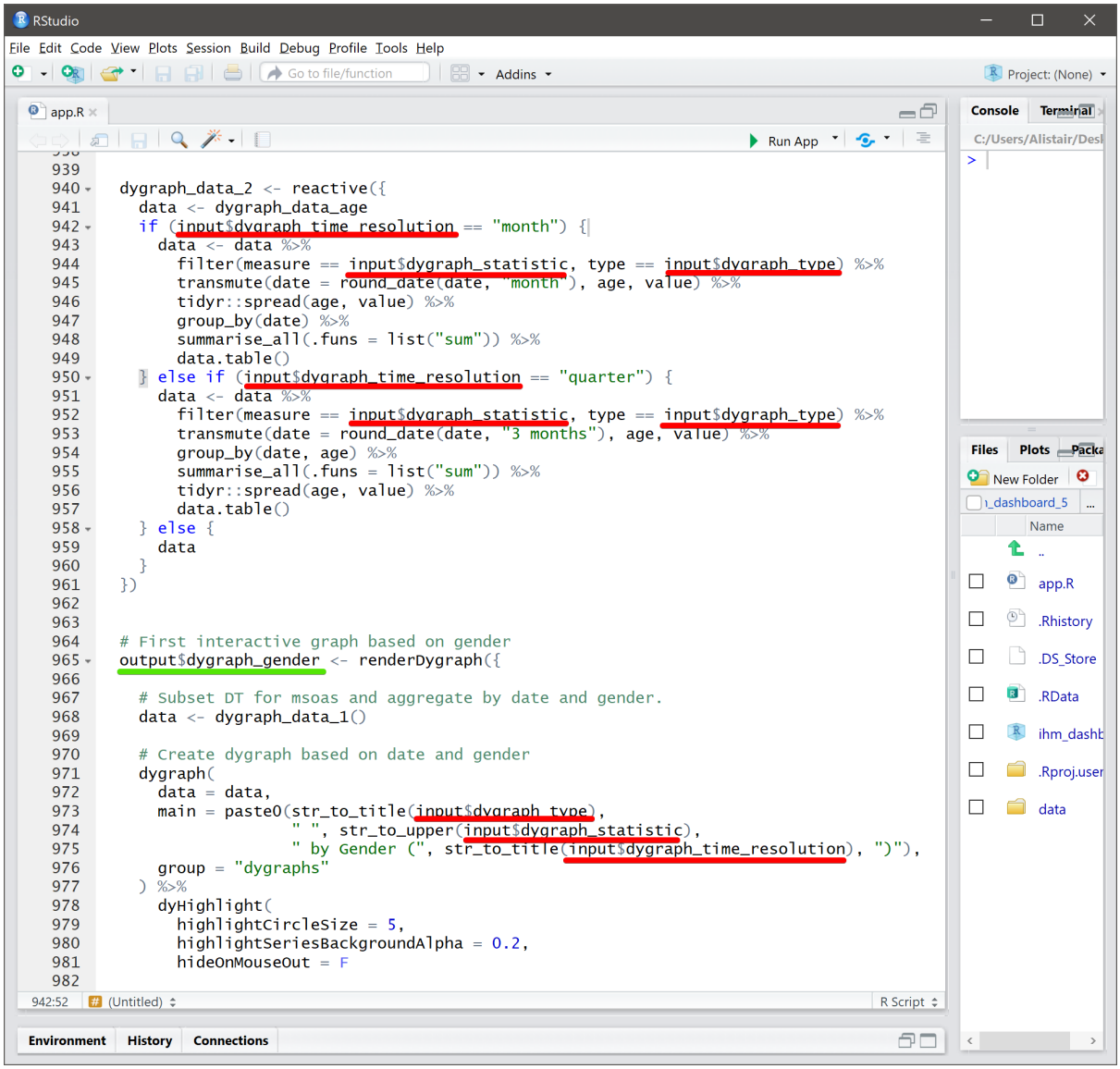
Application Structure

In the previous section, the focus was on setting up and running the application in its original form. Here, the we consider the steps necessary to update the application for the purpose of incorporating updated or geographically distinct data.

Before detailing the steps necessary, a brief overview of how Shiny Applications function is provided. The shiny application may be considered as two parts. The user interface (UI) and the server.

The UI provides all details necessary to construct the aesthetic aspects of the application in terms of structure and navigation. The computation is completed by the server portion of the application. From a user's perspective, the dashboard allows the user to use various R-based analytical techniques through a simple and streamlined user interface. Key to the functionality of the application is the interface between the UI and the Server. Within the app.R file, you will notice two key variables which reoccur regularly, "input" and "output". In the following screenshot, a few examples of these variables are highlighted. Input variables are highlighted in Red and Output Variable are highlighted in Green. Note that in the UI (The first screenshot), the input/output notation is not explicit whereas, for the server (second screenshot), the variable names are explicit.





It can be helpful to think of the input and output variables as containers used to store all of the input elements and output elements respectively. For example, if it is necessary to edit the variable "od_c_real_date_start_1", in the UI, you would find an input with the name "od_c_real_date_start_1" whereas, in the server you would find input$od_c_real_date_start_1.

Each dashboard widget on the dashboard is recorded as an input which can be used by the server. Outputs are generated within the server and can then be visualised as part of the user interface. In both cases, "inputs" and "outputs" are assigned an identifier such that they can be referenced. Where any changes are required to either "input" or "output", we refer to these in the form input$<identifier>.

Aside from the UI and server, the other major component of the application is the data. Here, the application utilises multiple data resources of both spatial and flat-file types. All of the data required to run the application are stored in the data directory within the application folder. Further sub-directories are employed to split the data into manageable parts. The following table provides a brief introduction to the data in each sub-directory. It is worth familiarising yourself with these data.

Table 1: Table containing a summary of the source data structure used for the app

Sub-Directory	Content
accessibility	A single file containing the mean minimum journey time to reach specific POIs across a 24-hour period at 30 minute intervals for each of the timetable periods.
eligibility	Two files. The first using mid-year population estimates between 2010 and 2016 and a second using population forecast data between 2016 and 2041.
od_data_real	One file which contains real origin-destination data aggregated by month.
od_data_synthetic	Single files for each date containing **synthetic** individual origin-destination data appended with demographic characteristics of the traveller.
POIs	Further sub-directories containing shapefiles for each of the POI datasets.
spatial_data	Shapefiles of Output Areas, LSOAs and Local Authorities in the West Midlands.
trends	Table containing the Frequency of journeys per month split first by Age Cohort and second by Gender

Updating the application

Having discussed the underlying functionality and structure of the application, the following instructions cover the steps necessary to update the underlying data within the application. These instructions are provided for each of the main dashboard pages.

Updating Trends

The two dygraphs are each created using a distinct file. These files are trends_age.csv and trends_gender.csv. Both files can be found in the "trends" sub-directory in the "data" folder. The File structure is date, (age5/gender), type, n, pp and npp. N indicated number of journeys, pp indicates the number of passengers and npp is the number of journeys per person. These files can simply be updated with new data on an ad-hoc basis and the graphs will update to reflect the new data.

Updating Origin-Destination (Public Monthly resolution)

The monthly origin-destination source data can be found in the "od_data_monthly" sub-directory in the "data" directory. The file structure is year, month, orig_lsoa, dest_lsoa and frequency. There are two aspects to updating this data. First, the origin-destination data may be updated to extend or alter the temporal coverage. Second, the UI will require updating to account for the new start and end data of the data.

For the data:

1) The easiest approach is to open and add to the existing csv file such that the data structure is maintained. The file is (file_name.csv)

For the UI edit the app.R file as indicated:

1) Update "od_c_real_date_start_1" with the correct start date, end date and starting value.
2) Update "od_c_real_date_start_2" with the same start and end dates to match the new data.

Updating Origin-Destination (Private High resolution)

The synthetic dataset used to demonstrate the potential of the dashboard takes the form that would be necessary for the input data. Unlike the "Public Monthly Resolution" data, this data allows the

user to drill down into the data picking custom time segments and specify demographics. Consequently, several steps are required to incorporate new data.

1) Here, a file exists for each day of journeys. Each file should be names based on the Date for which it is valid. The columns which are required are (date, type, age, gender, n, hour, orig_oa11cd, dest_oa11cd, orig_lsoa11cd, dest_lsoa11cd, orig_msoa11cd and dest_msoa11cd). The remaining columns were created as part of the synthetic data creation process.
2) Within the UI, the following input should be updated.
    a. “od_c_date_start_1” to set the default start date.
    b. “od_c_date_start_custom” with the same start date.
    c. “od_c_demographics” with any addition demographic characteristics. Specifically, any age categories used in the data or gender.
    d. “od_c_type_1” to match the card_types in the input data.
    e. “od_c_date_start_2” to set the default start date for the second time period where a comparison is being performed.

Updating Accessibility

As before, it may be necessary to alter both the UI and data in order to incorporate new or updated data into the dashboard.

The data:

1) The accessibility data can be found in the accessibility sub-directory within the data directory. Assuming no new categories are added, two steps are required to update the accessibility data.
2) The raw accessibility data should be updated maintaining the same file structure.

For the UI edit the app.R file as indicated:

1) Within the application UI section, the dates should be update to reflect those in the accessibility file. The ids of the two sections where new dates are required are “ac_date” and “ac_date_comp”.
2) If you need to add a new amenity type
    a. Update “ac_type” in the UI to include an appropriate option in the layer selector.
    b. Maintain the same file structure, upload a points shapefile into the POIs subdirectory of data. The points should be in WGS84 (EPSG4326).
    c. Update app.R to import new points from the POI directory. Section heading in the script is
    “# Import Accessibility POI Shapefiles”
    d. Update script to add new circle markers for these amenity. Section heading in the script is

"# Accessibility Amenity Circle Markers". You should use an identical identifier to that used in the UI "ac_type" selector.

e. In the same block of code, add the new layer to the hideGroup function.

f. Go to the "# Accessibility Map update" section and see the map update observer function. In the leafletProxy function include the new layer in the hideGroup parameter.

Updating Eligibility

The eligibility tab uses two datasets in its computation. ONS Mid-year population estimates at Output Area Level between 2010 and 2016 and ONS Population Projections at Local Authority level between 2016 and 2041. Updates to both the UI and underlying data are required.

The data:

1) The files "eligible_pop_estimates.csv" and "eligbility_pop_forecasts.csv" should be updated to reflect the new data. The file structure is area_code, year, count. In the case of the estimates, the area code relates to LSOA while for forecasts the scale is Local Authority.

2) In the event that the location is changed, it will be necessary to update both the Local Authority and Output Areas shapefiles found in the spatial_data directory.

For the UI edit the app.R file as indicated:

1) The UI should be updated to reflect the year range.

2) Update "elig_year" to reflect the start and end year for the mid-year population data.

3) Update "elig_year_comp" to reflect the start and end year for the mid-year population data.

4) Update "elig_forecast_year" to reflect the start and end year for the population projection data.

5) Update "elig_forecast_year_comp" to reflect the start and end year for the population projection data.

Updating Spatial Data

In the case of all of the spatial boundary data and points of interest, the data are limited to coverage of the West Midlands Combined Authority. Consequently, if recreating or updating the map it will be necessary that all of these data are updated to reflect the new locations. The main requirement of any new data are that they are in WGS84 (EPSG:4326). Spatial data are stored in two places within the applications, the POIs sub-directory and the spatial_data folder.

Three key steps are:

1) The spatial_data folder should be updated with the appropriate Local Authority, LSOA and OA shapefiles which correspond with the data used within the application.

2) Based on the data in the spatial_data folder, a csv area lookup file should be created which contains area_code, scale, area_name, longitude and latitude. This file is important for plotting markers and the origin-destination matrices.

3) As discussed previously, the POIs should be updated to reflect the correct extent of the data being mapped.

Alistair Leak IHM

Dashboard: Useful Information/Websites

Intro.JS # Used to create the guided tours

https://github.com/carlganz/rintrojs

Shiny Dashboard # Framework for creating R Shiny Dashboard

https://rstudio.github.io/shinydashboard/

Shiny Widgets # Better input controls than standard shiny

https://github.com/dreamRs/shinyWidgets

rleaflet # Used for all mapping

https://rstudio.github.io/leaflet/

rbokeh # used to create interactive plots

https://hafen.github.io/rbokeh/

rdygraphs # used to create dygraphs on app landing page.

https://rstudio.github.io/dygraphs/