Group Assessment

James Farrow

10/10/2018

HDAT 9800 Group Assessment

Your task is to build a shiny app showcasing some of the things we have been learning about this semester.

You have been provided with some data files from AIHW and the ABS.

You will build an application with three main parts (tasks). The user must be able to switch between them using the UI. The UI will change depending on what the user has selected.

Each group member should do a separate task and each task is worth 20 marks. Each group member should be responsible for the overall design of each part but this is a group assignment so collaboration is encouraged. There must be evidence (via git) that the workload has been shared but it does not necessarily have to be entirely split by task. The group mark will be the sum of the member marks (normalised to be out of 30).

The UI should contain three tabs, one for each task. Each task should be contained entirely within its respective tab. Each task has been configured with a sidebar and a main panel which should be used appropriately.

All maps should have a base (non-data) layer with appropriate tiles. You can use the default `addTiles()` or custom terrain tiles, whichever you feel is better and results in a clearer map. Not all tasks need to use the same tiles.

All maps/plots should have a legend (and title, axes, *&c.*) as appropriate.

All UI elements should be appropriately labelled in the UI.

There will be data that does not change for the lifetime of the application. Give careful thought to where you load that data. Also give careful thought to where you do calculations as you will be marked down if your application repeatedly calculates things unnecessarily.

Different tasks may have the similar inputs, *e.g.* `State` in Task 1 and Task 2. You might want to name these inputs `Task1.State` and `Task2.State` to avoid confusion.

Your plots and maps should be appropriately titled and labelled and have any other supporting documentation necessary to make sense of them (such as legends and/or supporting text).

Task 1: Hospitals

The file `hospitals.csv` contains information about hospitals in Australia from AIHW taken from https://www.myhospitals.gov.au/about-the-data/download-data.

Some of the names of columns in this file contain spaces which makes processing in R awkward.

If you have read the CSV data into the variable `hospitals`, rename the columns using the `make.names()` function as follows.

```
names(hospitals) <- make.names(names(hospitals))
```

Display a map with one data layer: markers for each hospital in the file.

There should be a 'State' filter, a 'Sector' filter and a 'Beds' filter which allows the user to restrict the displayed hospitals appropriately.

Note that in each case below you may need to pre-process/clean the data to ensure correct matching.

The values selectable on the 'State' filter should be:

- `All`, `New South Wales`, `Victoria`, `Queensland`, `South Australia`, `Tasmania`, `Western Australia`, `Northern Territory`, and `Australian Capital Territory`.

and should select on the `State` field of the data.

The values selectable on the 'Sector' filter should be:

- `All`, `Public`, and `Private`

and should select on the `Sector` field of the data.

The values selectable on the 'Beds' filter should be:

- `Any`, `>500`, `200-500`, `100-199`, `50-99`, `<50` and `Other`

and should select on the `Beds` field of the data.

Markers should use the Font Awesome 'hospital-o' icon in white on a coloured background. Public hospitals should be `"green"`. Private hospitals should use the colour `"orange"`.

The correct way to do this is to create a vector of icons and select from this vector when mapping:

```
HospitalIcons <- awesomeIconList(
  Public = makeAwesomeIcon(icon = 'hospital-o', markerColor = 'green', iconColor = 'white', library = "
  Private = makeAwesomeIcon(icon = 'hospital-o', markerColor = 'orange', iconColor = 'white', library =
)

# and then when using addAwesomeMarkers() use icon = ~HospitalIcons[Sector] as an argument
```

The user should be able to change the filters and have the map display the selected hospitals.

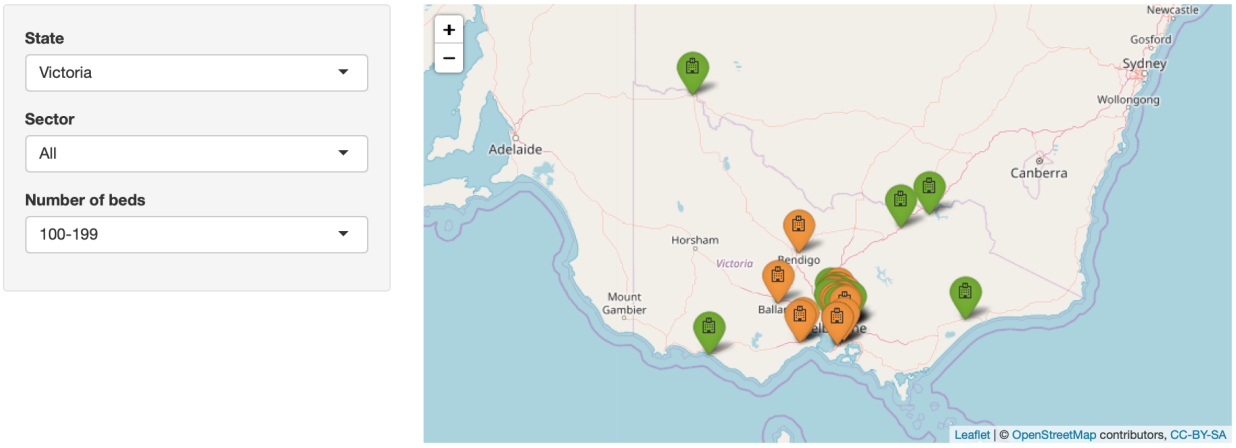
The map should redraw automatically when any dropdown is changed.

Make sure your programme sensibly handles the situation where the filters result in no hospitals to display. (A sensible outcome might be to display an empty map of Australia with a descriptive message.)

The tooltip for each marker should be the name of the hospital.

The popup text when a marker is clicked should be the name and contact details for the hospital along with the website and description if present in the data file.

Your map should look something like this.



Task 2: Distance from emergency department

Using the data in `hospitals.csv` determine hospitals which have an emergency department. This information is in the `Description` field.

Draw a map with two data layers.

There should be a drop down menu (a `selectInput`) labelled 'State' (`input$Task2.state`) which allows the user to filter the displayed hospitals appropriately. Only hospitals from the selected state should be used.

The values selectable on the 'State' filter should be:

- `New South Wales`, `Victoria`, `Queensland`, `South Australia`, `Tasmania`, `Western Australia`, `Northern Territory`, and `Australian Capital Territory`

and should select on the `State` field of the hospitals data.

There should be an entry box labelled 'Maximum distance (in km)' (`input$Task2.maxdistance`) to be used in the map drawing step.

There should be an entry box labelled 'Number of grid points' (`input$Task2.npoints`) to be used in the map drawing step. You will not want to put anything larger than 10000 in this box as it will slow down computation enormously. If you find 10000 takes too long to plot, use smaller values.

There should be a checkbox labelled 'Show markers' (`input$Task2.show.markers`) to turn on and off the display of markers for the hospitals.

There should be an `Update` button.

The map should only be redrawn when the `Update` button is pressed. Changes in the `State` selector, the `maxdistance` entry box, the `npoints` entry box or the `show.markers` checkbox should not result in an immediate redraw.

(A way to do this is to wrap each of these four inputs in an `eventReactive` wrapper triggered by `input$Task2.update` and to make sure your `renderLeaflet` functions does not use any `input$` values directly but only uses the `eventReactive` wrapper functions.)

The map should have a blue boundary showing the selected state polygon.

(Note: the state polygons are in the `STE_2016_AUST.shp` file and can be selected by state using `STE_NAME16`.)

You might want to simply the state geometries to speed things up. You can do this if desired using:

```
polys <- rgeos::gSimplify(polys, tol = 0.01)
```

One layer should have icons like the AwesomeIcons in Task 1 for hospitals but coloured `"red"`. (See the Task 1 section for code. You will only have one type of icon so there is no need for an array.) The same tooltip/popup details as in Task 1 should be displayed.

These markers should be able to be shown/hidden using the `input$Task2.show.markers` checkbox.

A second layer (below the marker layer and above the terrain layer) should have a heat map coloured by distance from the nearest emergency department calculated using the value given for the grid resolution.

You will want to create a grid of points with the given number for each state and clip it to the state. For each point of the grid, you need to calculatte the distance to the nearest emergency department. This is the minimum of the distances to all emergency departments.

A reactive grid function to calculate the spanning grid (assuming the `npoints()` `eventRactive` wrapper as discussed above and other reactive function `state_poly` which returns the state polygon based on the `Task2.state` input) would look as follows:

```
grid <- reactive({
  # make a grid spanning the state polygon
  makegrid(state_poly(), n = npoints())
})
```

To calculate the grid resolution use the following reactive function

```
gridresolution <- reactive({
  # take the grid and return the distance between the first two points
  g <- grid()
  sp::spDists(data.matrix(g[1,]), data.matrix(g[2,]))
})
```

We can create grid points from the grid an clip to the state polygon as follows

```
grid.points <- sp::SpatialPointsDataFrame(g, data.frame(n=1:nrow(g)), proj4string = CRS(proj4string(sta
grid.points <- grid.points[state_poly(), ]
```

To turn the `Longitude` and `Latitude` columns in the `hospitals` data frame into a useable set of spatial points we can use the `coordinates()` function.

Assuming `state_hospitals` contains a subset of `hospitals` for a given state:

```
coordinates(state_hospitals) <- c("Longitude", "Latitude")
proj4string(state_hospitals) <- CRS(proj4string(state_poly()))
```

will add the points and an appropriate projection.

Given a set of points, we can find the distance to the closest of a second set of points by finding the distance to every one of a second set of points and taking the minimum.

That is, if we have a vector of grid points and a vector of hospitals points we can create a matrix of distances with one row for each point and one column for each hospital. The distance to the nearest hospital for each point is thus the minimum value for each row.

```
distance.matrix <- sp::spDists(grid.points, state_hospitals)
grid.points$ed.distance = apply(distance.matrix, 1, min)
```

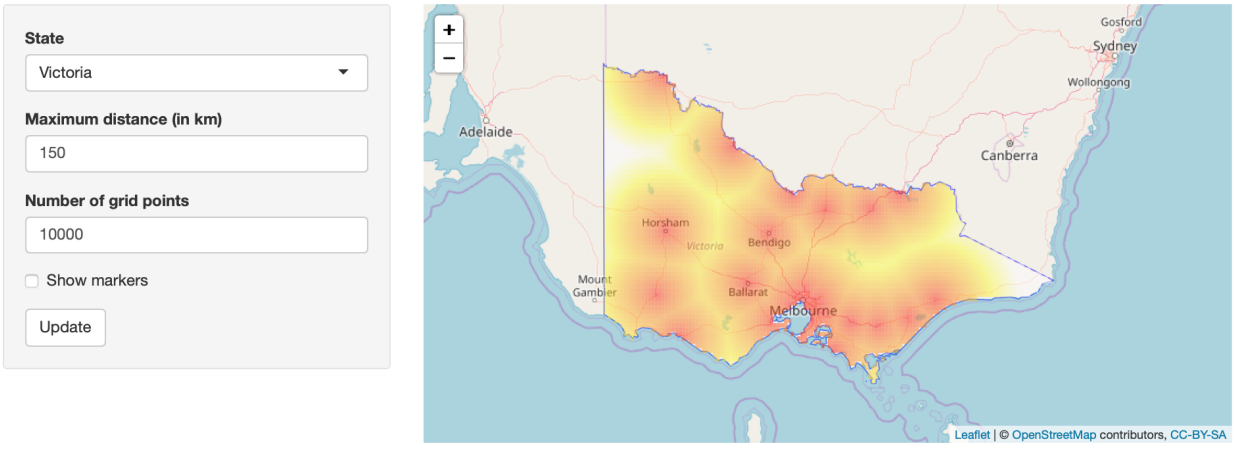
Don't forget, you'll need to also restrain `ed.distance` to `maxdistance` before plotting otherwise you'll end up with strangely coloured areas of your map.

Finally, we'll need to create a small square polygon to colour in around each grid point.

```
grid.squares <- rgeos::gBuffer(grid.points, width = gridresolution() / 2, quadsegs = 1, capStyle = "SQU
```

Colour these `grid.squares` polygons in the same way we did for population density using a `colorNumeric` colour pallett function which goes from `0` to `maxdistance` and spans the colours `"red"`, `"orange"`, `"yellow"` and `"white"`.

Your map should look something like this.



Task 3: Length of hospital stay

The file `myhospitals-average-length-of-stay-data.csv` contains information about the length of stay in hospitals for various reporting years.

You will need to do appropriate type-castings and cleaning and pre-processing to use this data file. Everything loaded

This task is to generate a plot of this data controlled by various UI elements similar to https://www.myhospitals.gov.au/compare-hospitals/average-length-of-stay/interactive-charts.

Make sure you follow the specifications of this assignment rather than blindly copy the above web page.

Your UI should have:

A selector for 'Peer group' on the `Peer group` column of the data.

A selector for 'Category' on the `Category` column of the data.

The horizontal axis of your graph should the `Time period` (ordered chronologically).

The vertical axis of your graph should be `Days` and have limits of 1 and 6 days.

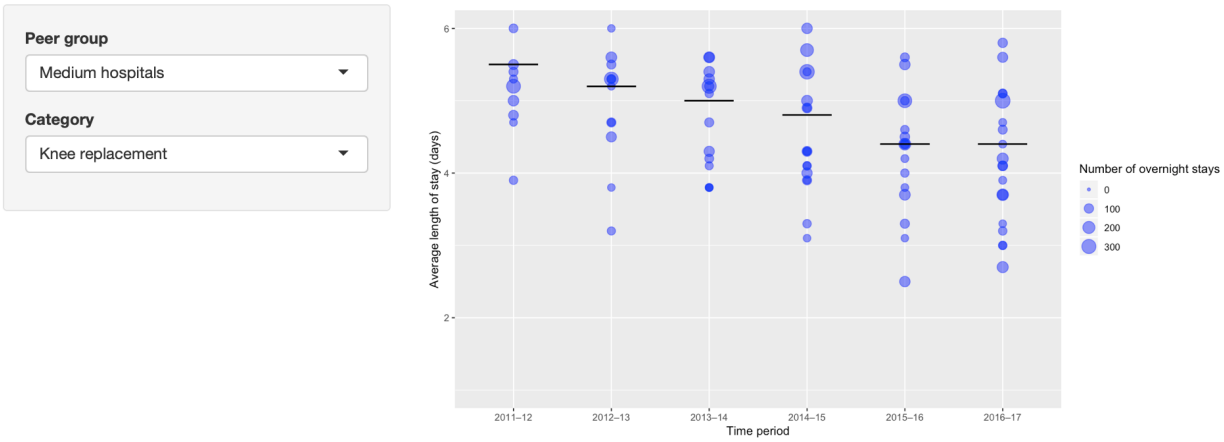
Plot a circle for each hospital using the `Average length of stay` value.

Size each circle using the `Number of overnight stays` value.

Add an indicator for each time period showing the average for the peer group as a whole.

This can be done using `geom_segment`.

Your plot should look something like this.



Presentation

Part of the mark for this assignment will be a short presentation on how you approached your task and what it showed.

You can do this in class on the 22nd, during a special arranged session during the week 19-23 November. or as a short 1-9 minute recorded video presentation submitted separately. The group as a whole can submit 1 video or each person can submit separately a video talking about their task. Each person should speak from 2-3 minutes on their task. Include a demonstration of your task.

Marking

This assignment is due Monday, November 19, 2018 at 5 p.m.

Each task shall be marked out of 20 as follows:

Task	Mark
Use of git to collaborate (good commit frequency, conflict resolution, shared workload)	4
Correct shiny UI elements	4
Correct reactivity for shiny	2
Correct output elements for task (including labelling/documentation)	8
Video/oral presentation	2

The group mark will be the sum of the member marks (normalised to be out of 30).

Everyone in the group will receive the same final mark.