The Queensland government collects data on road accidents dating back to 1st January 2001 and details characteristics of the incident including,

* Location of the crash (lat / long coordinates)
* ABS statistical area codes (SA2-4, LGA, remoteness)
* Atmospheric and road conditions (weather, lighting, sealed / unsealed roads, speed limit zone, etc)
* Severity of the incident (minor injury to fatality)
* Types of vehicles involved (car, bus, truck, bike, etc) and
* Description of the incident

Mapping this data highlights hot spots where car accidents occur more often. In particular the dangerous areas in wet conditions, problematic intersections and the areas of Queensland which are more dangerous than others in terms of fatality rates.

I developed a Shiny App utilising leaflet to easily explore the data (and just for fun). It features,

* A collapsible panel for filtering the data by selecting statistical areas and other features.
* An insights panel showing the breakdown of severity, vehicles involved, accidents over time and a Bayesian estimate of the fatality rate for the selected area.
* Data explorer tab.

This data is of road accidents, so the estimate of fatality rate in this case is the fatality rate given the vehicle was involved in an accident, rather than the fatality rate by road accident in the population. It is a slightly different take on how this statistic is usually published, but a useful one.

The best way to view the app is to run the following code. Firstly, check to make sure you have the packages installed by running

check\_packages <- function(packages){

if(all(packages %in% rownames(installed.packages()))){

TRUE

}else{

cat("Install the following packages before proceding\n", packages[!(packages %in% rownames(installed.packages()))], "\n")

}

}

packages\_needed <- c("tidyverse", "shiny", "leaflet", "leaflet.extras", "magrittr", "htmltools", "htmlwidgets", "showtext", "data.table")

check\_packages(packages\_needed)

If all good, now run the line below and it will load the app.

runGitHub("doehm/road-accidents/", "doehm", launch.browser = TRUE)

This will launch it directly on your machine.

There are a lot of neat things we can do with this data and I’ll be adding to the app over time.

**Brisbane Inner**

A subset of the app focuses on the “Brisbane Inner” SA3 area to give a taste of what to expect. It shows car accidents in the city since 1st January 2013. When zooming in, hover over the marker to get a short description of the crash.

**Code bits**

Below is the underlying code of the example above leaflet map, but I strongly recommend running the code above to view the Shiny app. See [Github](https://github.com/doehm/road-accidents) for the full code.

# queensland road accident data

# libraries

library(tidyverse)

library(shiny)

library(leaflet)

library(leaflet.extras)

library(magrittr)

library(htmltools)

library(htmlwidgets)

library(showtext)

library(data.table)

# font

try({

font\_add\_google(name = "Montserrat", family = "mont")

showtext\_auto()

}, TRUE)

# load data

# or if it doesn't work grab the Rdata file from Github - see link above

load\_data <- function(){

if(!file.exists("locations.csv")){

cat('\n Download may take a few minutes...\n')

url <- "http://www.tmr.qld.gov.au/~/media/aboutus/corpinfo/Open%20data/crash/locations.csv"

download.file(url, destfile = "locations.csv", method="libcurl")

}

accidents\_raw <- read\_csv("locations.csv")

return(accidents\_raw)

}

accidents\_raw <- load\_data() %>% filter(Crash\_Severity != "Property damage only")

# sample of brisbane inner

accidents <- accidents\_raw %>%

filter(

Loc\_ABS\_Statistical\_Area\_3 == "Brisbane Inner",

Crash\_Year > 2013

) %>%

mutate(fatality = Count\_Casualty\_Fatality > 0)

# basic leaflet

m <- leaflet(accidents) %>%

addProviderTiles(providers$Stamen.Toner, group = "Black and white") %>%

addTiles(options = providerTileOptions(noWrap = TRUE), group="Colour") %>%

addMarkers(

lng = ~Crash\_Longitude\_GDA94,

lat = ~Crash\_Latitude\_GDA94,

clusterOptions = markerClusterOptions(),

label = ~htmlEscape(Crash\_DCA\_Description)

) %>%

addCircleMarkers(

lng = ~Crash\_Longitude\_GDA94[accidents$fatality],

lat = ~Crash\_Latitude\_GDA94[accidents$fatality],

color = "#8B0000",

stroke = FALSE,

fillOpacity = 0.8,

group = "Fatalities"

) %>%

addHeatmap(

lng = ~Crash\_Longitude\_GDA94,

lat = ~Crash\_Latitude\_GDA94,

radius = 17,

blur = 25,

cellSize = 25

) %>%

addLayersControl(

overlayGroups = c("Fatalities"),

baseGroups = c("Black and white","Colour"),

options = layersControlOptions(collapsed = FALSE)

)

Shiny App Full Code

Server.R

|  |
| --- |
| # libraries |
|  | library(tidyverse) |
|  | library(shiny) |
|  | library(leaflet) |
|  | library(leaflet.extras) |
|  | library(magrittr) |
|  | library(htmltools) |
|  | library(htmlwidgets) |
|  | library(showtext) |
|  | library(data.table) |
|  | library(lazyeval) |
|  |  |
|  | # load data |
|  | load("./data/road-accident-data.Rdata") |
|  |  |
|  | # font |
|  | try({ |
|  | font\_add\_google(name = "Montserrat", family = "mont") |
|  | showtext\_auto() |
|  | }, TRUE) |
|  |  |
|  | # set my theme and colours |
|  | my\_theme <- function(y\_on = element\_blank()){ |
|  | theme\_minimal() + |
|  | theme( |
|  | legend.position = "none", |
|  | axis.text = element\_text(family = "mont", size = 12), |
|  | axis.text.x = element\_text(angle = 45, vjust = 0.5), |
|  | axis.text.y = y\_on, |
|  | axis.title = element\_blank(), |
|  | plot.title = element\_text(family = "mont", hjust = 0.5, face = "bold"), |
|  | plot.subtitle = element\_text(family = "mont", hjust = 0.5, face = "bold") |
|  | ) |
|  | } |
|  |  |
|  | my\_cols <- function(n = 16) colorRampPalette(c("darkmagenta", "turquoise"))(n) |
|  |  |
|  | theme\_forest <- function(scale = 1){ |
|  | theme\_minimal() + |
|  | theme( |
|  | legend.position = "none", |
|  | axis.text = element\_text(family = "mont", size = 16\*scale), |
|  | axis.text.x = element\_text(vjust = 0.5), |
|  | axis.title.y = element\_blank(), |
|  | axis.title.x = element\_text(family = "mont", size = 16\*scale), |
|  | plot.title = element\_text(family = "mont", hjust = 0.5, size = 26\*scale, face = "bold"), |
|  | plot.subtitle = element\_text(family = "mont", hjust = 0.5, size = 18\*scale), |
|  | plot.caption = element\_text(size = 12\*scale) |
|  | ) |
|  | } |
|  |  |
|  | posterior\_f <- function(df, y, n, a = 1.30, b = 77, inflator = 100) { |
|  | out <- data.frame() |
|  | qs <- c(0.025, 0.1, 0.5, 0.9, 0.975) |
|  | for(k in 1:nrow(df)){ |
|  | out <- rbind(out, inflator\*rbeta(1e4, shape1 = a+y[k], shape2 = b+n[k]-y[k]) %>% quantile(qs)) |
|  | } |
|  | colnames(out) <- paste0("q", 100\*qs) |
|  | return(out) |
|  | } |
|  |  |
|  |  |
|  |  |
|  |  |
|  | set.seed(190513) |
|  | # fatalities |
|  | areas <- grep("Area", colnames(accidents\_raw), value = TRUE)[3:4] |
|  | names(areas) <- c("sa3", "sa4") |
|  | fatality\_fn <- function(area){ |
|  | accidents\_raw %>% |
|  | group\_by\_(area) %>% |
|  | filter( |
|  | Crash\_Year == 2017, |
|  | ) %>% |
|  | summarise( |
|  | count = length(Count\_Casualty\_Total), |
|  | n\_fatalities = sum(Count\_Casualty\_Fatality) |
|  | ) %>% |
|  | bind\_cols(posterior\_f(df = ., y = .$n\_fatalities, n = .$count)) %>% |
|  | arrange(q50) %>% |
|  | mutate\_(area = interp(~factor(v, level = v), v = as.name(area))) %>% |
|  | ggplot() + |
|  | geom\_segment(mapping = aes(x = q2.5, xend = q97.5, y = area, yend = area)) + |
|  | geom\_segment(mapping = aes(x = q10, xend = q90, y = area, yend = area, col = q50), size = 2) + |
|  | geom\_point(mapping = aes(x = q50, y = area), pch = 3) + |
|  | theme\_forest() + |
|  | scale\_colour\_gradientn(colors = my\_cols()) + |
|  | labs( |
|  | title = "Fatality rate given observed road accidents", |
|  | subtitle = paste("Bayesian estimate of the fatality rate for", toupper(names(area)), "areas in 2017"), |
|  | x = "Fatality rate (%)") |
|  | } |
|  | fatality\_plots <- list(sa3 = fatality\_fn(areas[1]), sa4 = fatality\_fn(areas[2])) |
|  |  |
|  |  |
|  |  |
|  | # cyclists |
|  | cyclist\_fn <- function(area){ |
|  | accidents\_raw %>% |
|  | group\_by\_(area) %>% |
|  | filter(Crash\_Year == 2017) %>% |
|  | summarise( |
|  | count = n(), |
|  | n\_bicycles = sum(Count\_Unit\_Bicycle > 0) |
|  | ) %>% |
|  | bind\_cols(posterior\_f(df = ., y = .$n\_bicycles, n = .$count, a = 1.55, b = 25)) %>% |
|  | arrange(q50) %>% |
|  | mutate\_(area = interp(~factor(v, level = v), v = as.name(area))) %>% |
|  | ggplot() + |
|  | geom\_segment(mapping = aes(x = q2.5, xend = q97.5, y = area, yend = area)) + |
|  | geom\_segment(mapping = aes(x = q10, xend = q90, y = area, yend = area, col = q50), size = 2) + |
|  | geom\_point(mapping = aes(x = q50, y = area), pch = 3) + |
|  | theme\_forest() + |
|  | scale\_colour\_gradientn(colors = my\_cols()) + |
|  | labs( |
|  | title = "Rate of cyclists involved in road accidents", |
|  | subtitle = paste("Bayesian estimate of the rate of accidents involving cyclists for", toupper(names(area)), "areas in 2017"), |
|  | x = "Accidents involving cyclists (%)" |
|  | ) |
|  | } |
|  | cyclist\_plots <- list(sa3 = cyclist\_fn(areas[1]), sa4 = cyclist\_fn(areas[2])) |
|  |  |
|  |  |
|  | # pedestrians |
|  | pedestrian\_fn <- function(area){ |
|  | accidents\_raw %>% |
|  | group\_by\_(area) %>% |
|  | filter(Crash\_Year == 2017) %>% |
|  | summarise( |
|  | count = n(), |
|  | n\_bicycles = sum(Count\_Unit\_Pedestrian > 0) |
|  | ) %>% |
|  | bind\_cols(posterior\_f(df = ., y = .$n\_bicycles, n = .$count, a = 3, b = 60)) %>% |
|  | arrange(q50) %>% |
|  | mutate\_(area = interp(~factor(v, level = v), v = as.name(area))) %>% |
|  | ggplot() + |
|  | geom\_segment(mapping = aes(x = q2.5, xend = q97.5, y = area, yend = area)) + |
|  | geom\_segment(mapping = aes(x = q10, xend = q90, y = area, yend = area, col = q50), size = 2) + |
|  | geom\_point(mapping = aes(x = q50, y = area), pch = 3) + |
|  | theme\_forest() + |
|  | scale\_colour\_gradientn(colors = my\_cols()) + |
|  | labs( |
|  | title = "Rate of pedestrians involved in road accidents", |
|  | subtitle = paste("Bayesian estimate of the rate of accidents involving pedestrians for", toupper(names(area)), "areas in 2017"), |
|  | x = "Pedestrians injured from the accident (%)" |
|  | ) |
|  | } |
|  | pedestrian\_plots <- list(sa3 = pedestrian\_fn(areas[1]), sa4 = pedestrian\_fn(areas[2])) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | function(input, output, session) { |
|  |  |
|  | mainFilter <- reactive({ |
|  | accidents\_raw %>% |
|  | {if(input$loc == "Loc\_ABS\_Statistical\_Area\_2") filter(., Loc\_ABS\_Statistical\_Area\_2 == input$sa2) else .} %>% |
|  | {if(input$loc == "Loc\_ABS\_Statistical\_Area\_3") filter(., Loc\_ABS\_Statistical\_Area\_3 == input$sa3) else .} %>% |
|  | {if(input$loc == "Loc\_ABS\_Statistical\_Area\_4") filter(., Loc\_ABS\_Statistical\_Area\_4 == input$sa4) else .} %>% |
|  | {if(input$loc == "Loc\_Local\_Government\_Area") filter(., Loc\_Local\_Government\_Area == input$lga) else .} %>% |
|  | {if(input$loc == "Loc\_ABS\_Remoteness") filter(., Loc\_ABS\_Remoteness == input$remote) else .} %>% |
|  | {if(input$unit == "Bicycle") filter(., Count\_Unit\_Bicycle > 0) else .} %>% |
|  | {if(input$unit == "Car") filter(., Count\_Unit\_Car > 0) else .} %>% |
|  | {if(input$unit == "Motocycle") filter(., Count\_Unit\_Motorcycle\_Moped > 0) else .} %>% |
|  | {if(input$unit == "Truck") filter(., Count\_Unit\_Truck > 0) else .} %>% |
|  | {if(input$unit == "Pedestrian") filter(., Count\_Unit\_Pedestrian > 0) else .} %>% |
|  | {if(input$unit == "Bus") filter(., Count\_Unit\_Bus > 0) else .} %>% |
|  | {if(input$day\_of\_week != "..." & !is.null(input$day\_of\_the\_week)) filter(., Crash\_Day\_Of\_Week == input$day\_of\_week) else .} %>% |
|  | {if(input$month != "..." & !is.null(input$month)) filter(., Crash\_Month == input$month) else .} %>% |
|  | {if(input$weather != "..." & !is.null(input$weather)) filter(., Crash\_Atmospheric\_Condition == input$weather) else .} %>% |
|  | {if(input$driving\_conditions != "..." & !is.null(input$driving\_conditions)) filter(., Crash\_Lighting\_Condition == input$driving\_conditions) else .} %>% |
|  | {if(input$road\_condition != "..." & !is.null(input$road\_condition)) filter(., Crash\_Road\_Surface\_Condition == input$road\_condition) else .} %>% |
|  | {if(input$speed\_limit != "..." & !is.null(input$speed\_limit)) filter(., Crash\_Speed\_Limit == input$speed\_limit) else .} %>% |
|  | {if(input$road\_feature != "..." & !is.null(input$road\_feature)) filter(., Crash\_Roadway\_Feature == input$road\_feature) else .} %>% |
|  | {if(input$crash\_type != "..." & !is.null(input$crash\_type)) filter(., Crash\_Nature == input$crash\_type) else .} %>% |
|  | {if(input$crash\_severity != "..." & !is.null(input$crash\_severity)) filter(., Crash\_Severity == input$crash\_severity) else .} %>% |
|  | mutate(fatality = Count\_Casualty\_Fatality > 0) |
|  | }) |
|  |  |
|  | selectData <- reactive({ |
|  | mainFilter() %>% |
|  | filter(Crash\_Year >= input$year[1] & Crash\_Year <= input$year[2]) |
|  | }) |
|  |  |
|  | # data frame for fatality rate simulation |
|  | fatalityRate <- reactive({ |
|  |  |
|  | accidents\_raw %>% |
|  | {if(input$loc == "Loc\_ABS\_Statistical\_Area\_2") filter(., Loc\_ABS\_Statistical\_Area\_2 == input$sa2) else .} %>% |
|  | {if(input$loc == "Loc\_ABS\_Statistical\_Area\_3") filter(., Loc\_ABS\_Statistical\_Area\_3 == input$sa3) else .} %>% |
|  | {if(input$loc == "Loc\_ABS\_Statistical\_Area\_4") filter(., Loc\_ABS\_Statistical\_Area\_4 == input$sa4) else .} %>% |
|  | {if(input$loc == "Loc\_Local\_Government\_Area") filter(., Loc\_Local\_Government\_Area == input$lga) else .} %>% |
|  | {if(input$loc == "Loc\_ABS\_Remoteness") filter(., Loc\_ABS\_Remoteness == input$remote) else .} %>% |
|  | filter(Crash\_Year == 2017) %>% |
|  | summarise( |
|  | count = length(Crash\_Ref\_Number), |
|  | n\_fatalities = sum(Count\_Casualty\_Fatality) |
|  | ) |
|  |  |
|  | }) |
|  |  |
|  | # draws from the posterior |
|  | fatalityRateSim <- reactive({ |
|  | rbeta(1e4, shape1 = 1.3 + fatalityRate()$n\_fatalities, shape2 = 77 + fatalityRate()$count - fatalityRate()$n\_fatalities) |
|  | }) |
|  |  |
|  | # count of casualty type |
|  | output$casualty <- renderPlot({ |
|  | selectData() %>% |
|  | summarise( |
|  | Fatality = sum(Count\_Casualty\_Fatality), |
|  | Hospitalised = sum(Count\_Casualty\_Hospitalised), |
|  | `Medically treated` = sum(Count\_Casualty\_MedicallyTreated), |
|  | `Minor Injury` = sum(Count\_Casualty\_MinorInjury) |
|  | ) %>% |
|  | gather(casualty\_type, count) %>% |
|  | ggplot(aes(x = casualty\_type, y = count, fill = count)) + |
|  | geom\_bar(stat = "identity", alpha = 0.75) + |
|  | labs(title = "Severity of crash") + |
|  | scale\_fill\_gradientn(colors = my\_cols()) + |
|  | my\_theme() + |
|  | geom\_text(aes(x = casualty\_type, y = max(count)/4, label = count), family = "mont") |
|  | }) |
|  |  |
|  | # number of units involved with the crash |
|  | output$unit <-renderPlot({ |
|  | selectData() %>% |
|  | summarise( |
|  | Car = sum(Count\_Unit\_Car), |
|  | Motorcycle = sum(Count\_Unit\_Motorcycle\_Moped), |
|  | Truck = sum(Count\_Unit\_Truck), |
|  | Bus = sum(Count\_Unit\_Bus), |
|  | Bike = sum(Count\_Unit\_Bicycle), |
|  | Pedestrian = sum(Count\_Unit\_Pedestrian), |
|  | Other = sum(Count\_Unit\_Other) |
|  | ) %>% |
|  | gather(unit\_type, count) %>% |
|  | ggplot(aes(x = unit\_type, y = count, fill = count)) + |
|  | geom\_bar(stat = "identity", alpha = 0.75) + |
|  | labs(title = "Types of vehicles involved in accident") + |
|  | scale\_fill\_gradientn(colors = my\_cols()) + |
|  | my\_theme() + |
|  | geom\_text(aes(x = unit\_type, y = max(count)/4, label = count), family = "mont") |
|  | }) |
|  |  |
|  | # crashes over time |
|  | output$time\_series <- renderPlot({ |
|  | df <- mainFilter() %>% |
|  | filter(Crash\_Year != 2018) %>% |
|  | group\_by(Crash\_Year) %>% |
|  | summarise(count = length(Crash\_Ref\_Number)) |
|  |  |
|  | ggplot(df, aes(x = Crash\_Year, y = count)) + |
|  | geom\_line(col = "darkmagenta") + |
|  | geom\_point(col = "darkmagenta") + |
|  | coord\_cartesian(ylim = c(0, max(df$count))) + |
|  | my\_theme(y\_on = element\_text(family = "mont")) + |
|  | labs(title = "Number of accidents from 2001-2017") |
|  | }) |
|  |  |
|  | # rate dist |
|  | output$fatality\_rate <- renderPlot({ |
|  | data.frame(x = 100\*fatalityRateSim()) %>% |
|  | ggplot(aes(x = x)) + |
|  | geom\_histogram(fill = "turquoise") + |
|  | my\_theme() + |
|  | labs(title = paste0("Rate of road accident fatalities in 2017 (", round(100\*median(fatalityRateSim()), 1), ")")) |
|  | }) |
|  |  |
|  | # leaflet map |
|  | output$accident\_map <- renderLeaflet({ |
|  | leaflet(selectData()) %>% |
|  | addProviderTiles(providers$Stamen.Toner, group = "Black and white") %>% |
|  | addTiles(options = providerTileOptions(noWrap = TRUE), group="Colour") %>% |
|  | addMarkers( |
|  | lng = ~Crash\_Longitude\_GDA94, |
|  | lat = ~Crash\_Latitude\_GDA94, |
|  | clusterOptions = markerClusterOptions(), |
|  | label = ~htmlEscape(paste0(Crash\_Ref\_Number, ": ", Crash\_DCA\_Description)) |
|  | ) %>% |
|  | addCircleMarkers( |
|  | lng = ~Crash\_Longitude\_GDA94[selectData()$fatality], |
|  | lat = ~Crash\_Latitude\_GDA94[selectData()$fatality], |
|  | color = "#8B0000", |
|  | stroke = FALSE, |
|  | fillOpacity = 0.8, |
|  | group = "Fatalities" |
|  | ) %>% |
|  | addHeatmap( |
|  | lng = ~Crash\_Longitude\_GDA94, |
|  | lat = ~Crash\_Latitude\_GDA94, |
|  | radius = 17, |
|  | blur = 25, |
|  | cellSize = 25 |
|  | ) %>% |
|  |  |
|  | # adding in some controls directly into leaflet rather than through shiny |
|  | # this way we won't have to draw a new map every time |
|  | addLayersControl( |
|  | overlayGroups = c("Fatalities"), |
|  | baseGroups = c("Black and white","Colour"), |
|  | options = layersControlOptions(collapsed = FALSE) |
|  | ) |
|  | }) |
|  |  |
|  | output$leafl <- renderUI({ |
|  | if(!is.null(input$GetScreenHeight)){ |
|  | width <- session$clientData$output\_image1\_width |
|  | height <- session$clientData$output\_image1\_height |
|  | leafletOutput("accident\_map", width = "100%", height = input$GetScreenHeight) |
|  | } |
|  | }) |
|  |  |
|  | output$crash\_data <- renderDataTable({ |
|  | selectData() |
|  | }, options = list(pageLength = 15)) |
|  |  |
|  | # TODO: fix the height of this output - make it dynamic |
|  | # also just make this better - for now, path of least resistance |
|  | selectForestPlot <- reactive({ |
|  | if(input$floc == "Loc\_ABS\_Statistical\_Area\_3"){ |
|  | k <- 1 |
|  | }else if(input$floc == "Loc\_ABS\_Statistical\_Area\_4"){ |
|  | k <- 2 |
|  | } |
|  | if(input$measure == "Fatality"){ |
|  | fatality\_plots[[k]] |
|  | }else if(input$measure == "Cyclists"){ |
|  | cyclist\_plots[[k]] |
|  | }else if(input$measure == "Pedestrians"){ |
|  | pedestrian\_plots[[k]] |
|  | } |
|  | }) |
|  |  |
|  | output$forest\_plot <- renderPlot({ |
|  | selectForestPlot() |
|  | }, width = 900, height = 1200) |
|  |  |
|  | } |

UI.R

|  |
| --- |
| # queensland road accident data |
|  |  |
|  | library(shiny) |
|  | library(tidyverse) |
|  | library(magrittr) |
|  | library(data.table) |
|  |  |
|  | # load data |
|  | load("./data/road-accident-data.Rdata") |
|  |  |
|  | # set filter vars |
|  | check\_box <- c("Day of the week", "Month", "Weather conditions", "Unit type involved", "Driving conditions", "Speed limit", |
|  | "Road feature", "Crash type", "Crash severity") |
|  | year <- accidents\_raw$Crash\_Year %>% unique %>% sort |
|  | day\_of\_week <- c("...", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday") |
|  | month <- c("...", "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December") |
|  | weather\_conditions <- c("...", accidents\_raw$Crash\_Atmospheric\_Condition %>% unique %>% sort) |
|  | units <- c("...", "Bicycle", "Car", "Truck", "Motorcycle", "Pedestrian", "Bus") |
|  | driving\_conditions <- c("...", accidents\_raw$Crash\_Lighting\_Condition %>% unique %>% sort) |
|  | road\_condition <- c("...", accidents\_raw$Crash\_Road\_Surface\_Condition %>% unique %>% sort) |
|  | speed\_limit <- c("...", accidents\_raw$Crash\_Speed\_Limit %>% unique %>% sort) |
|  | road\_feature <- c("...", accidents\_raw$Crash\_Roadway\_Feature %>% unique %>% sort) |
|  | crash\_type <- c("...", accidents\_raw$Crash\_Nature %>% unique %>% sort) |
|  | crash\_severity <- c("...", accidents\_raw$Crash\_Severity %>% unique %>% sort) |
|  | loc\_type <- c("State level", colnames(accidents\_raw)[str\_detect(colnames(accidents\_raw), "Loc\_ABS|Loc\_Local")]) |
|  | loc\_list <- sapply(loc\_type, function(x) accidents\_raw[[x]] %>% unique %>% sort) |
|  |  |
|  | # js code for making the map fit the screen height |
|  | jscode <- ' |
|  | $(document).on("shiny:connected", function(e) { |
|  | var jsHeight = window.innerHeight; |
|  | Shiny.onInputChange("GetScreenHeight",jsHeight); |
|  | }); |
|  | ' |
|  |  |
|  | # ui function |
|  | navbarPage("Road accidents in Queensland", id="road", |
|  | tabPanel("Heat map", |
|  | div(class="outer", |
|  | p(), |
|  | tags$script(jscode), |
|  | uiOutput("leafl"), |
|  | absolutePanel( |
|  | id = "controls", class = "panel panel-default", fixed = TRUE, |
|  | draggable = TRUE, top = 90, left = "auto", right = 180, bottom = "auto", |
|  | width = 250, height = "auto", |
|  | HTML('<button data-toggle="collapse" data-target="#main">Show filters</button>'), |
|  | tags$div(id = 'main', class="collapse", |
|  |  |
|  | # this isn't exactly elegant but it's the path of least resistance |
|  |  |
|  | # select year |
|  | sliderInput("year", label = "Year", min = min(year), max = max(year), value = c(2013, 2018), round = TRUE, step = 1), |
|  |  |
|  | # select statistical area |
|  | # TODO: put better labels on these |
|  | selectInput("loc", "Location category", loc\_type, selected = "Loc\_ABS\_Statistical\_Area\_3"), |
|  |  |
|  | # if sa2 select area with sa2 |
|  | conditionalPanel( |
|  | condition = "input.loc == 'Loc\_ABS\_Statistical\_Area\_2'", |
|  | selectInput("sa2", "Location", loc\_list[["Loc\_ABS\_Statistical\_Area\_2"]], selected = "Brisbane City") |
|  | ), |
|  |  |
|  | # if sa3 select area with sa3 |
|  | conditionalPanel( |
|  | condition = "input.loc == 'Loc\_ABS\_Statistical\_Area\_3'", |
|  | selectInput("sa3", "Location", loc\_list[["Loc\_ABS\_Statistical\_Area\_3"]], selected = "Brisbane Inner") |
|  | ), |
|  |  |
|  | # if sa4 select area with sa4 |
|  | conditionalPanel( |
|  | condition = "input.loc == 'Loc\_ABS\_Statistical\_Area\_4'", |
|  | selectInput("sa4", "Location", loc\_list[["Loc\_ABS\_Statistical\_Area\_4"]], selected = "Brisbane Inner City") |
|  | ), |
|  |  |
|  | # if lga select area with lga |
|  | conditionalPanel( |
|  | condition = "input.loc == 'Loc\_Local\_Government\_Area'", |
|  | selectInput("lga", "Location", loc\_list[["Loc\_Local\_Government\_Area"]], selected = "Brisbane City") |
|  | ), |
|  |  |
|  | # if remoteness select area with remoteness |
|  | conditionalPanel( |
|  | condition = "input.loc == 'Loc\_ABS\_Remoteness'", |
|  | selectInput("remote", "Location", loc\_list[["Loc\_ABS\_Remoteness"]], selected = "Major Cities") |
|  | ), |
|  |  |
|  | # checkboxGroupInput("check\_box", "Activate filters", check\_box), |
|  | checkboxInput('day\_of\_week\_t', 'Day of the week', value = FALSE), |
|  | checkboxInput('month\_t', 'Month', value = FALSE), |
|  | checkboxInput('Weather\_t', 'Weather condition', value = FALSE), |
|  | checkboxInput('unit\_t', 'Unit type involved', value = FALSE), |
|  | checkboxInput('driving\_conditions\_t', 'Driving conditions', value = FALSE), |
|  | checkboxInput('road\_condition\_t', 'Road conditions', value = FALSE), |
|  | checkboxInput('speed\_limit\_t', 'Speed limit', value = FALSE), |
|  | checkboxInput('road\_feature\_t', 'Road feature', value = FALSE), |
|  | checkboxInput('crash\_type\_t', 'Crash type', value = FALSE), |
|  | checkboxInput('crash\_severity\_t', 'Crash severity', value = FALSE), |
|  |  |
|  | # as conditional panels |
|  | conditionalPanel( |
|  | condition = "input.day\_of\_week\_t", |
|  | selectInput("day\_of\_week", "Day of the week", day\_of\_week, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.month\_t", |
|  | selectInput("month", "Month", month, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.weather\_t", |
|  | selectInput("weather", "Weather condition", weather\_conditions, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.unit\_t", |
|  | selectInput("unit", "Involving unit type", units, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.driving\_conditions\_t", |
|  | selectInput("driving\_conditions", "Driving condition", driving\_conditions, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.road\_condition\_t", |
|  | selectInput("road\_condition", "Road condition", road\_condition, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.speed\_limit\_t", |
|  | selectInput("speed\_limit", "Speed limit", speed\_limit, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.road\_feature\_t", |
|  | selectInput("road\_feature", "Road feature", road\_feature, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.crash\_type\_t", |
|  | selectInput("crash\_type", "Crash type", crash\_type, selected = "...") |
|  | ), |
|  |  |
|  | conditionalPanel( |
|  | condition = "input.crash\_severity\_t", |
|  | selectInput("crash\_severity", "Crash severity", crash\_severity, selected = "...") |
|  | ) |
|  | ) |
|  | ), |
|  |  |
|  | absolutePanel( |
|  | id = "controls", class = "panel panel-default", fixed = TRUE, |
|  | draggable = TRUE, top = 180, left = 20, right = "auto", bottom = "auto", |
|  | width = 400, height = "auto", |
|  | plotOutput("casualty", height = 200), |
|  | plotOutput("unit", height = 200), |
|  | plotOutput("time\_series", height = 200), |
|  | plotOutput("fatality\_rate", height = 200), |
|  | p(HTML('Created by <a target="\_blank" href="https://twitter.com/danoehm">@danoehm</a> / <a target="\_blank" href="http://gradientdescending.com/">gradientdescending.com</a>')), |
|  | p(HTML('Source: Dept. of Transport and Main Roads Qld')) |
|  | ) |
|  | ) |
|  | ), |
|  |  |
|  | tabPanel( |
|  | "Estimation", |
|  | sidebarPanel( |
|  | p(), |
|  | selectInput("floc", "Location", c("SA3" = "Loc\_ABS\_Statistical\_Area\_3", "SA4" = "Loc\_ABS\_Statistical\_Area\_4")), |
|  | selectInput("measure", "Measure", c("Fatality", "Cyclists", "Pedestrians")), |
|  | p(), |
|  | width = 2 |
|  | ), |
|  | mainPanel( |
|  | plotOutput('forest\_plot') |
|  | ) |
|  | ), |
|  |  |
|  | tabPanel( |
|  | "Data table", |
|  | div( |
|  | class = "outer", |
|  | p(), |
|  | tags$script(jscode), |
|  | dataTableOutput("crash\_data") |
|  | ) |
|  | ) |
|  | ) |