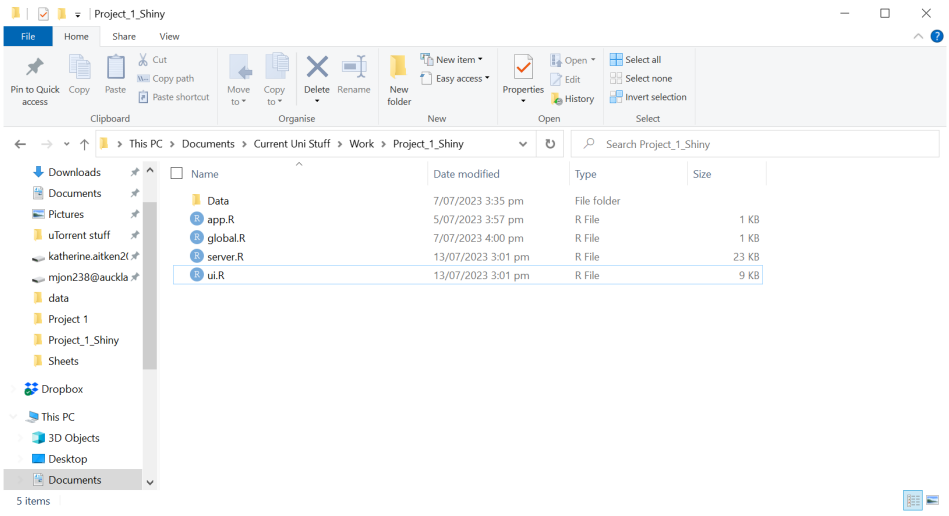
Summary Process on R-Shiny

Contents

1 Template for Shiny App

Create a project with the following empty R scripts, app.R, global.R, ui.R, server.R. Also create a Data folder.



1.1 Data

Add raw data files in this folder, we also collate them into a list and save them, but do that later.

1.2 app.R

Add this code to app.R file, this will allow the app to run.

```
source(global.R)

ui <- source("ui.R")
server <- source("server.R")

shinyApp(ui = ui, server = server)

runApp()
```

1.3 global.R

In global.R load our raw-data and packages:

```
#R Shiny Packages
library(shiny)
library(shinydashboard)
library(shinyWidgets)
library(shinyjs)
library(shinycssloaders)

#Data Cleaning packages
library(tidyverse)
library(janitor)
library(rrapply)
library(lubridate)
library(stringr)
library(plotly)
library(magrittr)
library(reshape2)

#Deploy Shiny App Packages
library(DT)
library(googlesheets4)

#Read all Data in
load("Data/shiny_app_data.Rmd")
```

1.4 ui.R

Create a shiny-dashboard template:

```
#Start Page ----
dashboardPage(
  title = "",
  dashboardHeader(
    title = "Template"
  ),

  #Add Sidebar Inputs
  dashboardSidebar(
    ),

  #Add Body (Outputs)
  dashboardBody(
  #Add tags (CSS Edits) Here

  #Add Actual Outputs Below

)
)
```

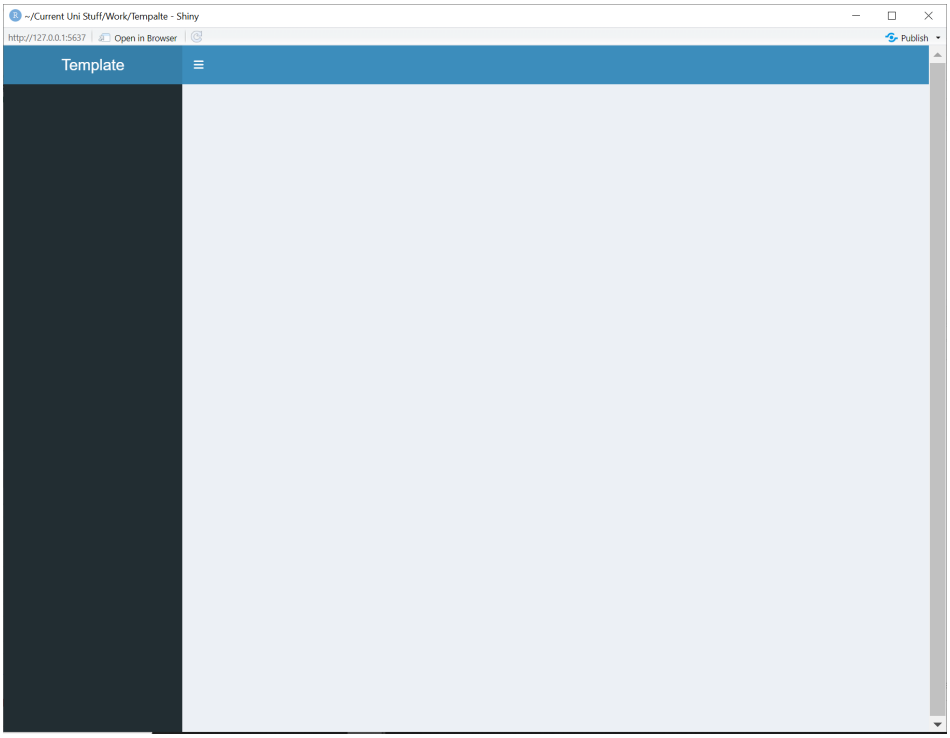
1.5 server.R

Create an empty server:

```
function(input, output) {
  #Start Server
}
```

1.6 Run the app

When we run the app, we will have the default template shiny app below, add the outputs in the center and inputs on the dark-blue section (left):



2 Creating the inputs

The next step is to add the inputs into the ui.R file, most inputs go into dashboardSideBar(), but tabPanels go into dashboardBody().

Find more inputs at: https://shiny.posit.co/r/gallery/widgets/widget-gallery/

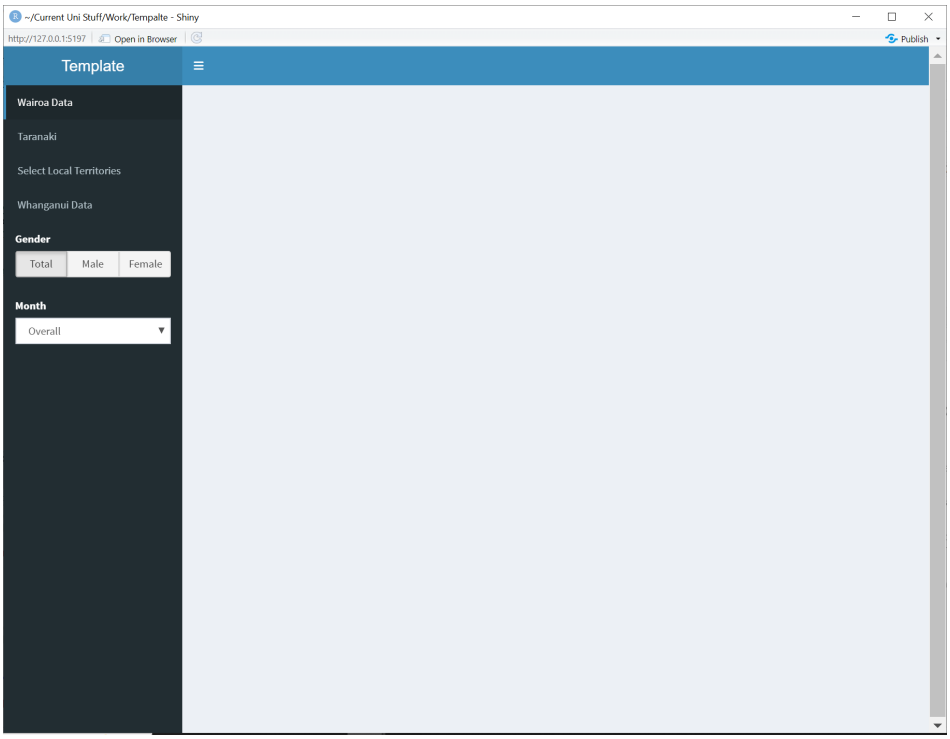
2.1 dashboardSideBar Inputs - Menu Items, Radio Group Buttons, Select Inputs

For the example I've added three different types of inputs, menu items, radio group buttons and select inputs, in that order. Menu items filters by our raw data folders, radio group buttons filters by gender, and select inputs filter by Months including an all months option called Overall. The code is included below.

```
dashboardSidebar(
#SideBar Inputs
sidebarMenu(
  id = "folder",
  menuItem("Wairoa Data", tabName = "WairoaData"),
  menuItem("Taranaki", tabName = "Taranaki22"),
  menuItem("Select Local Territories", tabName = "SelectLocalities"),
  menuItem("Whanganui Data", tabName = "WhanganuiData")
),

#Radio Group Buttons
radioGroupButtons(inputId = "gender",
                  label = "Gender",
                  choices = c("Total", "Male", "Female"),
                  selected = "Total",
                  justified = TRUE),

#Select Inputs
selectInput(inputId = "Month",
            label = "Month",
```

```
                choices = c("Overall",
                            "January 2021", "February 2021", "March 2021",
                            "April 2021", "May 2021", "June 2021", "July 2021"
                                ,
                            "August 2021", "September 2021", "October 2021",
                            "November 2021", "December 2021", "January 2022",
                            "February 2022", "March 2022", "April 2022",
                            "May 2022", "June 2022", "July 2022", "August 2022
                                ",
                            "September 2022", "October 2022", "November 2022",
                            "December 2022", "January 2023", "February 2023",
                               "March 2023"),
                selected = "Overall",
                selectize = F)
    ),
```

Running the app, we get the following result.

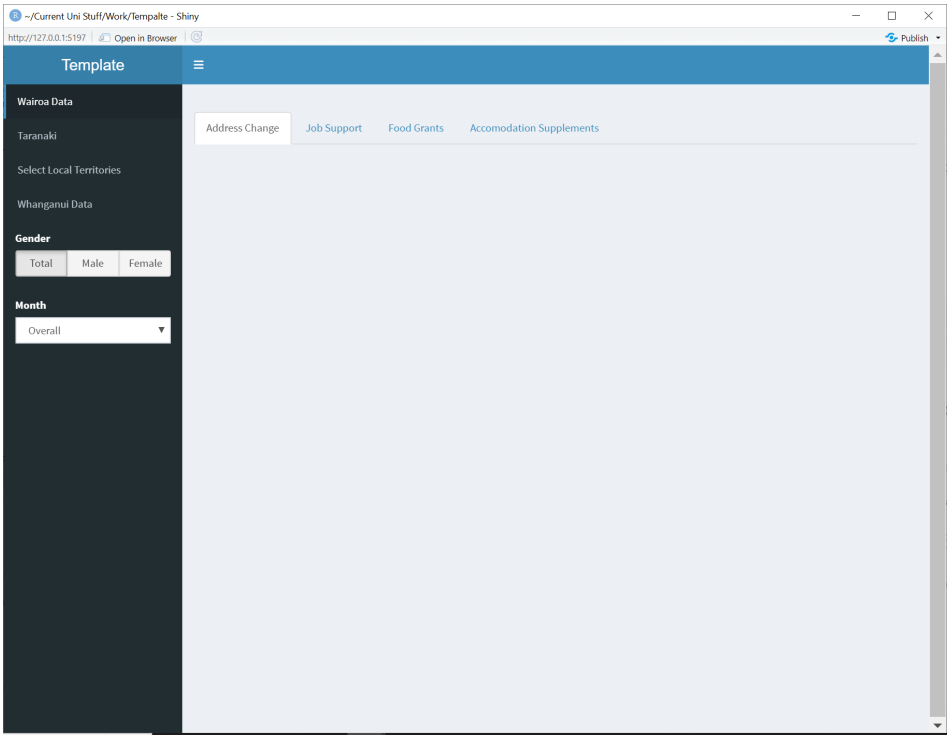


2.2 dashboardBody Inputs - Tab Panels

We can also include tab panel inputs, these are located in the body of the app rather than the sidebar. I have designed the tab panels to filter by the raw data files inside each raw data folder.

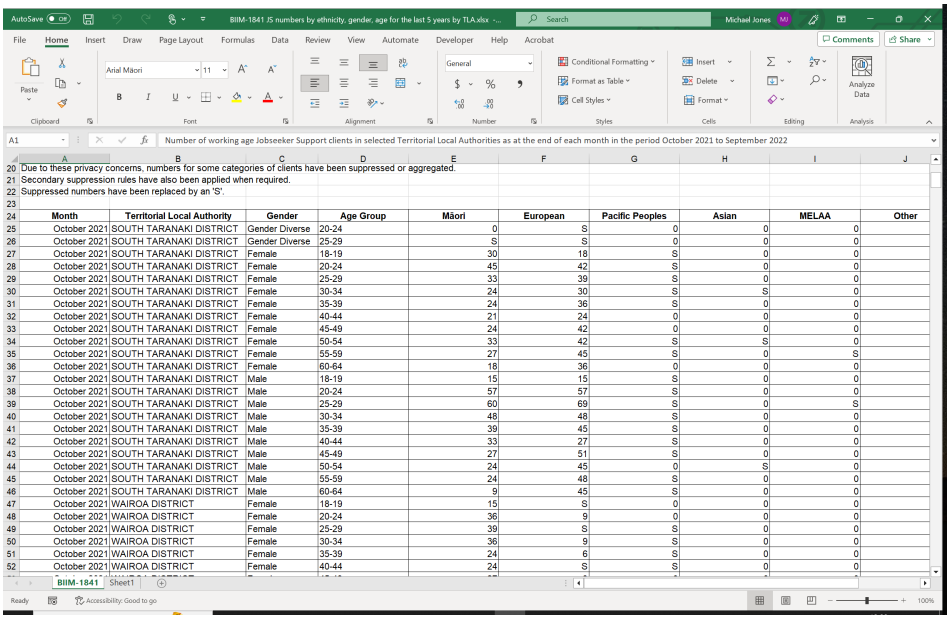
```
  #Add Body (Outputs)
dashboardBody(
#Add tags (CSS Edits) Here

#Add Actual Outputs Below
tabsetPanel(id = "sheets",
            tabPanel("Address Change", value = "addressChange"),
            tabPanel("Job Support", value = "jobSupport"),
            tabPanel("Food Grants", value = "foodGrants"),
            tabPanel("Accmmodation Supplements", value = "accomSup")
            )
)
```

Running the app:



2.3 Example Data-Format

The current data I've used is divided into four folders with 3-4 files each. Each folder contains raw data sheets for address change, job support, accommodation supplement and food grants (hence my tab names). The data is in the format below:



Number of working age Jobseeker Support clients in selected Territorial Local Authorities as at the end of each month in the period October 2021 to September 2022

Due to these privacy concerns, numbers for some categories of clients have been suppressed or aggregated.
Secondary suppression rules have also been applied when required.
Suppressed numbers have been replaced by an 'S'.

Month	Territorial Local Authority	Gender	Age Group	Māori	European	Pacific Peoples	Asian	MELAA	Other
October 2021	SOUTH TARANAKI DISTRICT	Gender Diverse	20-24	0	S	0	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Gender Diverse	25-29	S	S	0	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	18-19	30	18	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	20-24	45	42	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	25-29	33	39	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	30-34	24	30	S	S	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	35-39	24	36	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	40-44	21	24	0	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	45-49	24	42	0	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	50-54	33	42	S	S	0	
October 2021	SOUTH TARANAKI DISTRICT	Female	55-59	27	45	S	0	S	
October 2021	SOUTH TARANAKI DISTRICT	Female	60-64	18	36	0	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	18-19	15	15	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	20-24	57	57	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	25-29	60	69	S	0	S	
October 2021	SOUTH TARANAKI DISTRICT	Male	30-34	48	48	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	35-39	39	45	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	40-44	33	27	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	45-49	27	51	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	50-54	24	45	0	S	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	55-59	24	48	S	0	0	
October 2021	SOUTH TARANAKI DISTRICT	Male	60-64	9	45	S	0	0	
October 2021	WAIROA DISTRICT	Female	18-19	15	S	0	0	0	
October 2021	WAIROA DISTRICT	Female	20-24	36	9	0	0	0	
October 2021	WAIROA DISTRICT	Female	25-29	39	S	S	0	0	
October 2021	WAIROA DISTRICT	Female	30-34	36	9	S	0	0	
October 2021	WAIROA DISTRICT	Female	35-39	24	6	S	0	0	
October 2021	WAIROA DISTRICT	Female	40-44	24	S	S	0	0	

It is a count by territory, ethnicity, gender and month for each category.

I aim to filter by folder, sheet, month and gender, while also outputting several tables for each unique territory.

2.4 Loading the Data

Load the data into a list, and save them into a list based on your inputs, we want our input options to match each name in the list. For example the list below matches first my menu item names then my tab panel names. This makes filtering easier later on. In this part I also fix up an discrepancies in the raw data, such as mismatching column names or mistake data entries. I also match unspecified observations across the whole list using the rrapply() function.

In this example I only load in the Wairoa Data, for simplicity.

```
    df <- list(WairoaData = list(addressChange = NULL,
                                 jobSupport = NULL,
                                 foodGrants = NULL,
                                 foodGrantsCost = NULL)
)


#Read in the data

#Category 1
df$WairoaData$addressChange <- readxl::read_xlsx("Wairoa_Data_Jan18_23/
    INTERNAL_BIIM-2705_Change_of_Address_by_ethnicity.xlsx", range = "A17:L177
    8")

#Category 2
df$WairoaData$jobSupport <- readxl::read_xlsx("Wairoa_Data_Jan18_23/INTERNAL_
    BIIM-2705_JS_Clients.xlsx", range = "A17:L2627")

#Match Column Names
names(df$WairoaData$jobSupport)[names(df$WairoaData$jobSupport) == 'Total␣
    clients'] <- 'Total␣Clients'

#Category 3
df$WairoaData$foodGrants <- readxl::read_xlsx("Wairoa_Data_Jan18_23/INTERNAL_
    BIIM-2705_SNG_food_5_years.xlsx", range = "A19:N1772")

#Match Column Names
names(df$WairoaData$foodGrants)[names(df$WairoaData$foodGrants) == 'Total␣
    Distinct␣Clients'] <- 'Total␣Clients'

#Remove Incomplete Observations
nas <- which(df$WairoaData$foodGrants$`Total Clients` == ".")
df$WairoaData$foodGrants <- df$WairoaData$foodGrants[-nas,]

#Category 4
df$WairoaData$accomSup <- readxl::read_xlsx("Wairoa_Data_Jan18_23/INTERNAL_
    BIIM-2705_current_AS_5_years.xlsx", range = "A17:M2461")

#Match Column Names
names(df$WairoaData$accomSup)[names(df$WairoaData$accomSup) == 'Total␣clients'
    ] <- 'Total␣Clients'

#Edit Specific Data Entries
df <- rrapply(df, condition = function(x, .xname) is.data.frame(x),f =
    function(x) {
  x$Gender[x$Gender == "Gender␣Diverse"] <- "Unspecified"
  x$`Age Group`[x$Gender == "Gender␣Diverse"] <- "Unspecified"
  names(x)[names(x) == "Month␣End"] <- "Month"
  x
}, classes = "data.frame")

save(df, file = "Data/shiny_app_data.Rmd")
```

3 Creating the Shiny App

We create the function of the app inside server.R, this include our summaries. ui.R is used to visualise everything we do inside server.R.

3.1 Titles

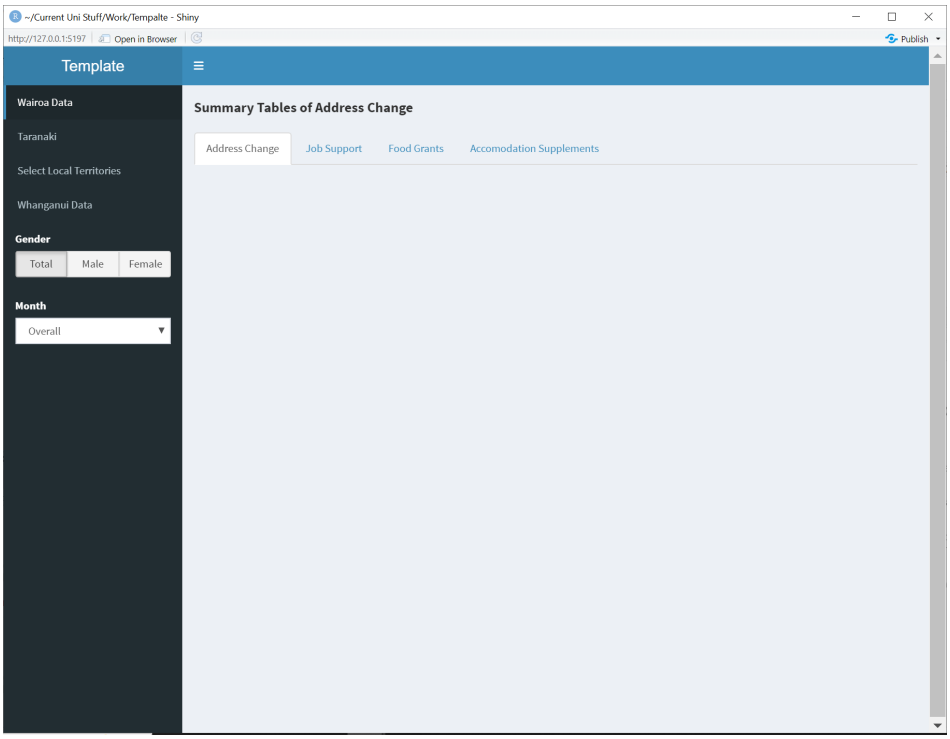
3.1.1 Main Title

We need to make clear titles before going to the summaries. In this example we're generating a summary for each sheet, so I make that clear in the title.

```
    output$title <- renderUI({

    #Choose title based on what tab (which sheet) is selected.
    label <- switch(input$sheets,
                    "addressChange" = "Address␣Change",
                    "jobSupport" = "Job␣Support",
                    "foodGrants" = "Food␣Grants",
                    "accomSup" = "Accommodation␣Supplements"
    )

    #Make HTML edits to make the title look nice
  title <- HTML("<div␣style='font-size:␣18px;'>␣<b>Summary␣Tables␣of␣", label)

   title
  })
```

I then place this output inside the ui.R File inside the dashboard body:

```
    dashboardBody(
    uiOutput("title")
    ...
    )
```



3.1.2 Table Titles

I intend to create a summary table for each unique territory in my selected raw data sheet. I will make the title for each table here.

I first create a reactive value called places, which concatenates all the unique territories in the raw-data folder I've selected (remember I select the raw data folder in menu items).

Creating reactive values is useful if you use them multiple times and need them to change depending on inputs.

```
    places <- reactive({


   places <- c("Total", unique(df[[input$folder]][[input$sheets]]$`Territorial
       Local Authority`))

   places
  })
```

Given this variable is reactive, the next time I use it I will need to put brackets afterwards, because R Shiny says so.

Next I need to output the territory titles (called places()). The following code is repetitive because each tab needs to have its own title and so does each place. In R shiny you can't repeat the same output twice inside the same ui.R file, the best solution is to give one output multiple names, as I have done below.

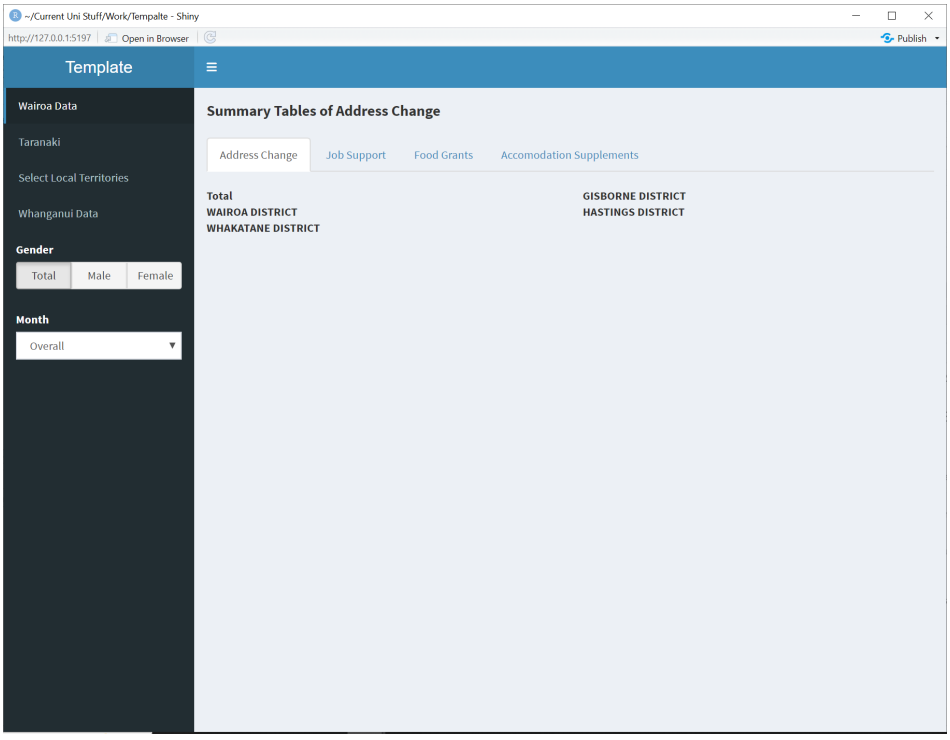
```
    output$tableTitleAC1 <- output$tableTitleJS1 <-
    output$tableTitleFG1 <- output$tableTitleAS1 <- renderUI({
    HTML("<div style='font-size: 14px;'> <b>", places()[[1]])
  })
  output$tableTitleAC2 <- output$tableTitleJS2 <-
    output$tableTitleFG2 <- output$tableTitleAS2 <- renderUI({
    HTML("<div style='font-size: 14px;'> <b>", places()[[2]])
  })
  output$tableTitleAC3 <- output$tableTitleJS3 <-
    output$tableTitleFG3 <- output$tableTitleAS3 <- renderUI({
    HTML("<div style='font-size: 14px;'> <b>", places()[[3]])
  })
  output$tableTitleAC4 <- output$tableTitleJS4 <-
    output$tableTitleFG4 <- output$tableTitleAS4 <- renderUI({
    HTML("<div style='font-size: 14px;'> <b>", places()[[4]])
  })
   output$tableTitleAC5 <- output$tableTitleJS5 <-
    output$tableTitleFG5 <- output$tableTitleAS5 <- renderUI({
    HTML("<div style='font-size: 14px;'> <b>", places()[[5]])
  })
```

I then put these outputs inside my dashboardBody in ui.R so they are visible in the app. Specifically, I put them inside the each respective tabPanel.

```
    dashboardBody(
    uiOutput("title"),
  br(),
tabsetPanel(id = "sheets", selected = "foodGrants",
            tabPanel(title = "Address Change", value = "addressChange", br(),
                     fluidRow(column(6, uiOutput("tableTitleAC1"),
                                        uiOutput("tableTitleAC2"),
                                        uiOutput("tableTitleAC5")),
                                column(6, uiOutput("tableTitleAC3"),
                                        uiOutput("tableTitleAC4")))
                     ),
            tabPanel(title = "Job Support", value = "jobSupport", br(),
```

```
                    fluidRow(column(6, uiOutput("tableTitleJS1"),
                                       uiOutput("tableTitleJS2"),
                                       uiOutput("tableTitleJS5")),
                             column(6, uiOutput("tableTitleJS3"),
                                       uiOutput("tableTitleJS4")))
                    ),
           tabPanel(title = "Food Grants", value = "foodGrants", br(),
                    fluidRow(column(6, uiOutput("tableTitleFG1"),
                                       uiOutput("tableTitleFG2"),
                                       uiOutput("tableTitleFG5")),
                             column(6, uiOutput("tableTitleFG3"),
                                       uiOutput("tableTitleFG4")))
                                ),
           tabPanel(title = "Accomodation Supplements", value = "accomSup",
               br(),
                    fluidRow(column(6, uiOutput("tableTitleAS1"),
                                       uiOutput("tableTitleAS2"),
                                       uiOutput("tableTitleAS5")),
                             column(6, uiOutput("tableTitleAS3"),
                                       uiOutput("tableTitleAS4")))
                                )
           )
)
)
```

The result is below:



3.2 Clean the Data

I first clean the data inside a reactive environment in my server.R. I do this in a reactive environment, because the clean data will react to the users chosen inputs.

I'll create a cleaner raw data set by filtering through my original raw data list and selecting only the relevant folders, and sheets. I've also mutated ages into several age-groups using the dplyr package.

I use the complete function to add in zeros in the data set for all combinations of columns. This is

because in the summary, we intend to have zero rows, rather than empty rows.

```
dfClean <- reactive({

      #Generalise Data
        ageSum <- data.frame(Month = df[[input$folder]][[input$sheets]]$
            Month,
                             `Age Group` = df[[input$folder]][[input$sheets
                                 ]]$`Age Group`,
                             "Territorial␣Local␣Authority" =
                               df[[input$folder]][[input$sheets]]$`
                                   Territorial Local Authority`,
                             "Gender" = df[[input$folder]][[input$sheets]]$
                                 Gender,
                             "Maori" = df[[input$folder]][[input$sheets]]$
                                 Maori * costMeasure[[input$folder]][[input$
                                 sheets]],
                             "Non-Maori" = (df[[input$folder]][[input$sheets
                                 ]]$European +
                                              df[[input$folder]][[input$
                                                  sheets]]$`Pacific Peoples`
                                                   +
                                              df[[input$folder]][[input$
                                                  sheets]]$Asian + df[[input
                                                  $folder]][[input$sheets]]$
                                                  MELAA +
                                              df[[input$folder]][[input$
                                                  sheets]]$Other +
                                              df[[input$folder]][[input$
                                                  sheets]]$Unknown) *
                                                  costMeasure[[input$folder
                                                  ]][[input$sheets]],
                             "European" = df[[input$folder]][[input$sheets
                                 ]]$European * costMeasure[[input$folder]][[
                                 input$sheets]],
                             "Other" = (df[[input$folder]][[input$sheets]]$`
                                 Pacific Peoples` +
                                          df[[input$folder]][[input$sheets
                                              ]]$Asian + df[[input$folder
                                              ]][[input$sheets]]$MELAA +
                                          df[[input$folder]][[input$sheets
                                              ]]$Other +
                                          df[[input$folder]][[input$sheets
                                              ]]$Unknown) * costMeasure[[
                                              input$folder]][[input$sheets
                                              ]],
                             "Total.Clients" = as.numeric(df[[input$folder
                                 ]][[input$sheets]]$`Total Clients`)) %>%
          mutate(`Age.Group` = replace(`Age.Group`,
                                       `Age.Group` %in% c("<16", "16", "17",
                                                          "18-19", "18", "19
                                                              "),
                                       "Under␣20"))%>%
          mutate(`Age.Group` = replace(`Age.Group`,
                                       `Age.Group` %in% c("20-24", "25-29"),
                                       "20-29"))%>%
          mutate(`Age.Group` = replace(`Age.Group`,
                                       `Age.Group` %in% c("30-34", "35-39"),
                                       "30-39"))%>%
          mutate(`Age.Group` = replace(`Age.Group`,
                                       `Age.Group` %in% c("40-44", "45-49"),
                                       "40-49"))%>%
```

```
    mutate(`Age.Group` = replace(`Age.Group`,
                                 `Age.Group` %in% c("50-54", "55-59",
                                    "60-64"),
                                 "50-64"))%>%
    complete(Month, Age.Group, Gender, `Territorial.Local.Authority`,
             fill = list(Maori = 0, Non.Maori = 0,
                         European = 0, Other = 0,
                         Total.Clients = 0))

  ageSum


})
```

3.3 Summarise Clean Data

We will create another reactive value that will generate the summary tables. We do this all inside the R shiny server.R file we created.

To make this file:

I first lapply over all the possible territories (places()) to create a reactive list in the Form Data – places – data frame.

Next, I create a totals data frame, which is a summary for both genders Male and Female combined into one. To create this data frame, I remove unspecified entries, then filter by month and territory, then I summarise all numeric columns (grouping by age group).

Finally, I create a data frame for male and female summaries by repeating the same process, filtering out unspecified entries, filter by month and territory, then summarise across all numeric columns (grouping by age group and gender). After this I combine this data frame with totals, so I have a summary with 3 gender categories, finally I filter by the gender input.

```
Data <- reactive({

  setNames(lapply(places(), function (i){

      ageSum <- dfClean()

      #Filter, then summarise
      totals <- ageSum%>%
        filter(Age.Group != "Unspecified",
               if(input$month != "Overall")Month == input$month else TRUE,
               if(i != "Total") Territorial.Local.Authority == i else TRUE)
                  %>%
        group_by(Age.Group)%>%
        summarise(across(c(Maori, European, Other, `Total.Clients`), sum))
            %>%
        mutate(Gender = "Total")

      #filter, then summarise, then bind rows, then sort, then filter again
          by gender.
      ageSum <- ageSum%>%
        filter(Age.Group != "Unspecified",
               if(input$month != "Overall")Month == input$month else TRUE,
               if(i != "Total") Territorial.Local.Authority == i else TRUE)
                  %>%
        group_by(`Age.Group`, `Gender`)%>%
        summarise(across(c(Maori, European, Other, `Total.Clients`), sum))
            %>%
        bind_rows(totals)%>%
```

```
        arrange(factor(`Gender`,
                       levels = c("Male", "Female", "Total")))%>%
        arrange(factor(`Age.Group`, levels = c("Under 20", "20-29",
                                               "30-39", "40-49", "50-64",
                                               "65+", "Unspecified")))%>%
        filter(Gender == input$gender)



    }), places())

})
```

3.4 Finishing the App

Next, I create many data-tables for each territory/place, I use the DT package.

Here's an example data table output.

```
  output$summaryFG1 <- renderDataTable({

    datatable(Data()[[places()[1]]],
              options = list(scrollX = TRUE,
                             pageLength = 25,
                             dom = 't'))
  })
```

Some info about data table options is available at https://rstudio.github.io/DT/options.html

I have to do this many items, like we did with the place titles, we repeat over each tab and for every territory/place.

See below:

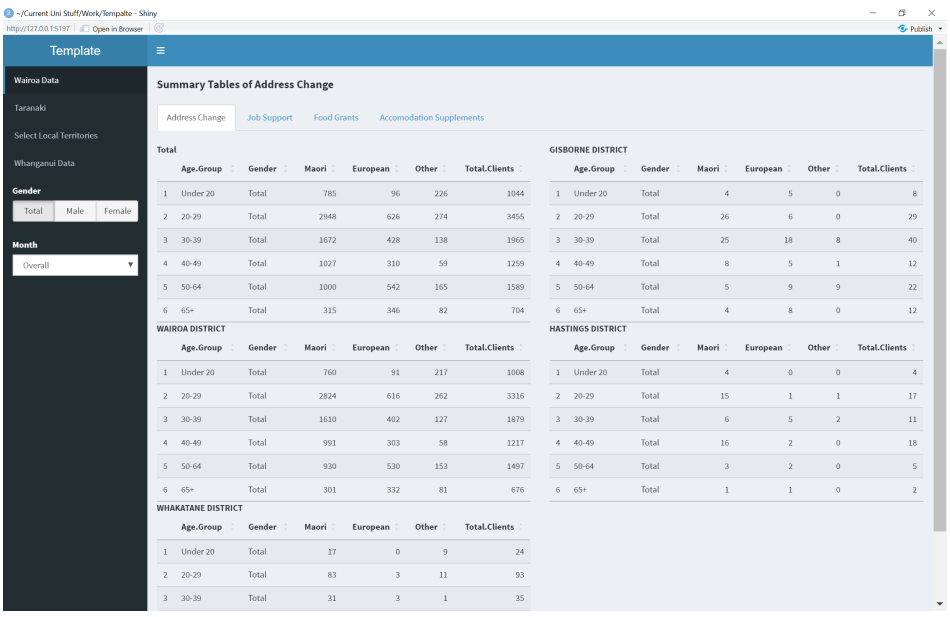
```
  output$summaryAC1 <- output$summaryJS1 <-
  output$summaryFG1 <- output$summaryAS1 <- renderDataTable({

    datatable(Data()[[places()[1]]],
              options = list(scrollX = TRUE,
                             pageLength = 25,
                             dom = 't'))
  })

output$summaryAC2 <- output$summaryJS2 <-
  output$summaryFG2 <- output$summaryAS2 <- renderDataTable({

    datatable(Data()[[places()[2]]],
              options = list(scrollX = TRUE,
                             pageLength = 25,
                             dom = 't'))
  })

output$summaryAC3 <- output$summaryJS3 <-
  output$summaryFG3 <- output$summaryAS3 <- renderDataTable({

    datatable(Data()[[places()[3]]],
              options = list(scrollX = TRUE,
                             pageLength = 25,
                             dom = 't'))
  })

output$summaryAC4 <- output$summaryJS4 <-
  output$summaryFG4 <- output$summaryAS4 <- renderDataTable({
```

```
    datatable(Data()[[places()[4]]],
              options = list(scrollX = TRUE,
                             pageLength = 25,
                             dom = 't'))
  })

output$summaryAC5 <- output$summaryJS5 <-
  output$summaryFG5 <- output$summaryAS5 <- renderDataTable({

    datatable(Data()[[places()[5]]],
              options = list(scrollX = TRUE,
                             pageLength = 25,
                             dom = 't'))
  })
```

Finally, I include this inside the UI, I'll just do one example for one tab, to avoid including unnecessary code.

```
tabsetPanel(id = "sheets", selected = "addressChange",
        tabPanel(title = "Address Change", value = "addressChange", br(),
                 fluidRow(column(6, uiOutput("tableTitleAC1"),
                                    DTOutput("summaryAC1"),
                                 uiOutput("tableTitleAC2"),
                                    DTOutput("summaryAC2"),
                                 uiOutput("tableTitleAC3"),
                                    DTOutput("summaryAC3")),
                          column(6, uiOutput("tableTitleAC4"),
                                    DTOutput("summaryAC4"),
                                 uiOutput("tableTitleAC5"),
                                  DTOutput("summaryAC5")))
                 ) .......
```

Finally we have our completed app, see below



4 Conclusion

I've made this app simple to avoid complicating things. You can add lots more details inside the app and CSS edits to make the app more appealing. The best tools for issues you face using R Shiny are

Google and ChatGPT, what ever problem you have someone has already asked on the internet.

Access my simple app at https://github.com/mjon238/ShinyTemplate

I made a small addition that makes the Month input reactive, you check it out in the repository link above.

Below is a screenshot of my app so far: