Shiny Instructions Manual

Table of Contents

Introduction **3**
- **About The Dashboard** **3**

Environment Set Up **5**
- **Installing R** **5**
- **Installing R Studio** **7**
- **Installing R Packages** **9**

General Setting **10**
- **Setting The Path** **10**
  - The path where the “results.csv” is located 10
  - The path where the dictionary file is located 10
  - The path where the “WorldBank-IndustryProject” folder is located 11
- **Run The Shiny Application** **11**
  - Run in the R console 11
  - Run in the RStudio 12

Edit Welcome Page **13**
- **Edit The Text** **13**
- **Edit The Style Of The Introduction Button** **14**

Edit The Introduction Tutorial **16**
- **Edit The Steps** **16**
  - Reordering the steps 16
  - Deleting a step 18
  - Adding a step 21
- **Edit The Content For The Introduction Steps** **24**
- **Edit The Text On The Buttons** **25**

Edit The Help Dropdown Menu **26**
- **Edit The Question Icon** **26**
- **Edit The Icon Inside** **27**
- **Edit The Text** **28**
- **Add/Delete Help Items** **29**

Edit The Logo And The Link in the Sidebar **32**
- **Change The Logo Image And The Link** **32**

Edit ABOUT US **34**
- **Change The Title And Link** **34**

Graph Options **35**

Adding More Graphs **35**
Adding the new graphic option into the sidebar 35
Adding new graph options 36
Transformation Options **38**
Adding More Formulas **38**
Adding a new formula option in the sidebar 38
Adding the new formulas 39
Exporting to the Web **40**
Appendix **46**

Introduction

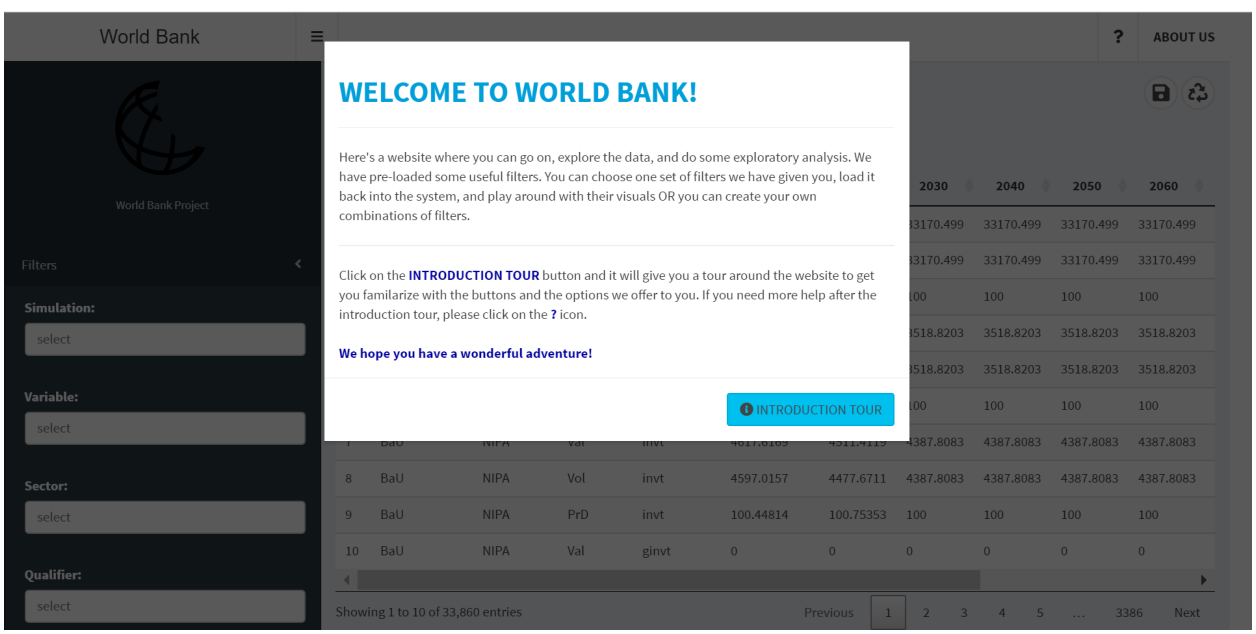
This document will show how to use the World Bank Dashboard powered by R Shiny in users' client servers. Shiny is an interactive tool that can easily change variables then produce direct plots. Also, Shiny can let the user save and load the Dashboard data, and users can delete or edit the stored plots. In the following document, we will show how this works and how to make any changes in R.

You will need these following files or folders to execute the instructions:

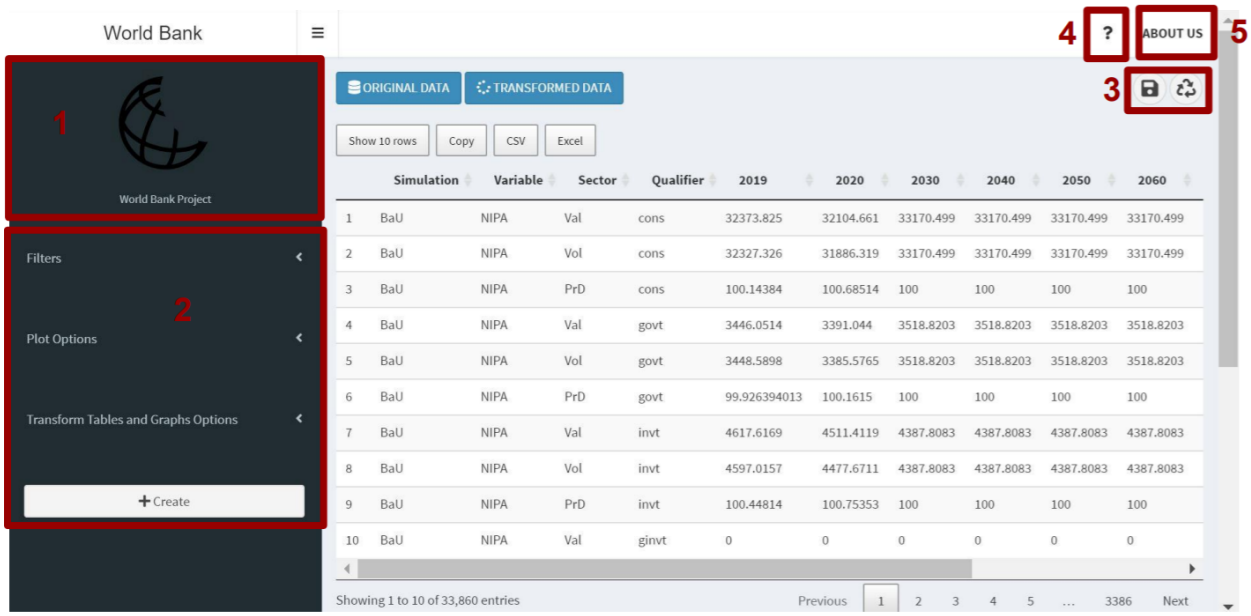
(1) main.R
(2) server.R
(3) Intro_Help_Pages.R
(4) graphs.R
(5) formulas.R
(6) ui.R
(7) www folder

About The Dashboard

Before you can make any selection on the sidebar, a pop up dialog will welcome us and give an option to tour around the Shiny dashboard. If you want to skip the introduction tutorial, simply click outside of the modal dialog.



Here's what the dashboard looks like when the introduction tutorial is finished.



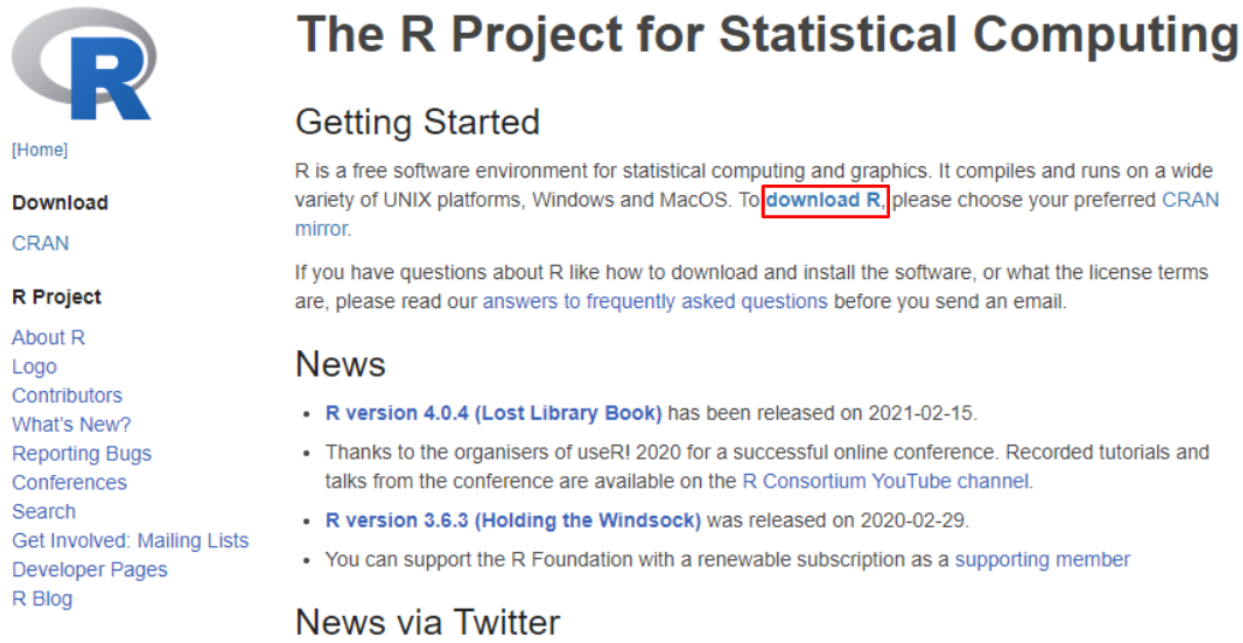
If you miss the tutorial, here's a brief recap of each components on the dashboard:

- **Box 1** is where the logo image is located.
- **Box 2** is where we select the inputs to include in the main body of the dashboard.
- **Box 3** is where the "Save" and "Load" drop-down buttons are located. The one with the "Save" icon allows us to save the selection we made on the sidebar. While, the "Recycle" icon allows us to load the saved inputs back into the sidebar.
- **Box 4** is the drop-down help menu. If users have trouble navigating the dashboard, they can refer to this menu for help.
- **Box 5** directs us to the World Bank's website when we click on "ABOUT US".

Environment Set Up

Installing R

A. Go to https://www.r-project.org/ and then click "Download R".

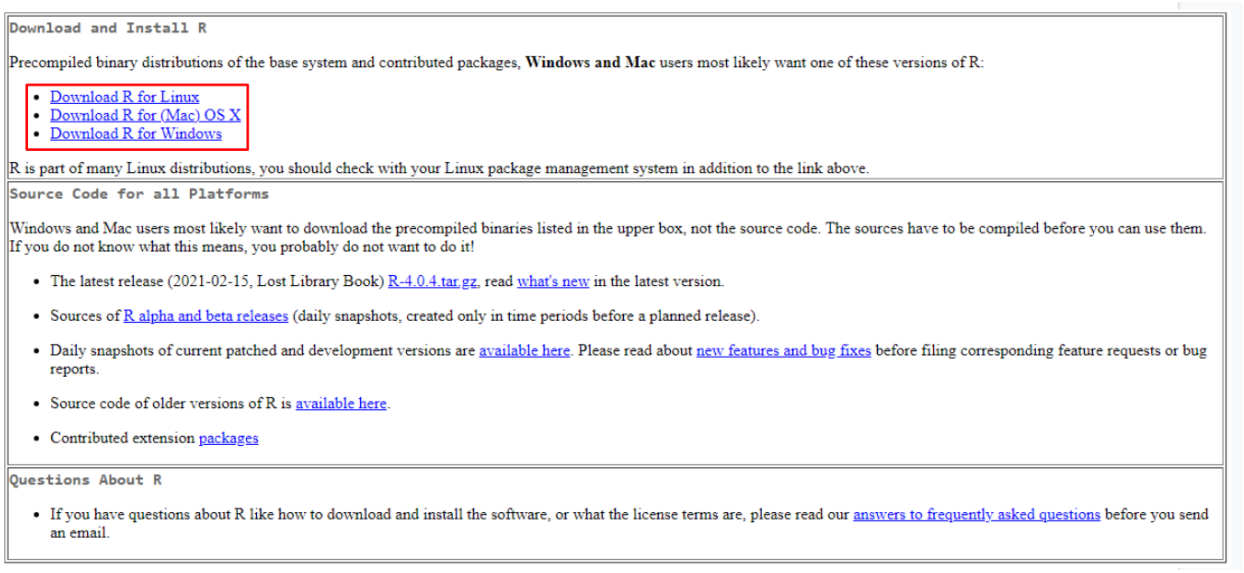


B. Search for the USA and choose the one that is closest to your location.

USA

https://mirror.las.iastate.edu/CRAN/	Iowa State University, Ames, IA
http://ftp.ussg.iu.edu/CRAN/	Indiana University
https://rweb.crmda.ku.edu/cran/	University of Kansas, Lawrence, KS
https://repo.miserver.it.umich.edu/cran/	MBNI, University of Michigan, Ann Arbor, MI
http://cran.wustl.edu/	Washington University, St. Louis, MO
http://archive.linux.duke.edu/cran/	Duke University, Durham, NC
https://cran.case.edu/	Case Western Reserve University, Cleveland, OH
https://ftp.osuosl.org/pub/cran/	Oregon State University
http://lib.stat.cmu.edu/R/CRAN/	Statlib, Carnegie Mellon University, Pittsburgh, PA
http://cran.mirrors.hoobly.com/	Hoobly Classifieds, Pittsburgh, PA
https://mirrors.nics.utk.edu/cran/	National Institute for Computational Sciences, Oak Ridge, TN
https://cran.microsoft.com/	Revolution Analytics, Dallas, TX

C. Installing R to your local computer.



Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

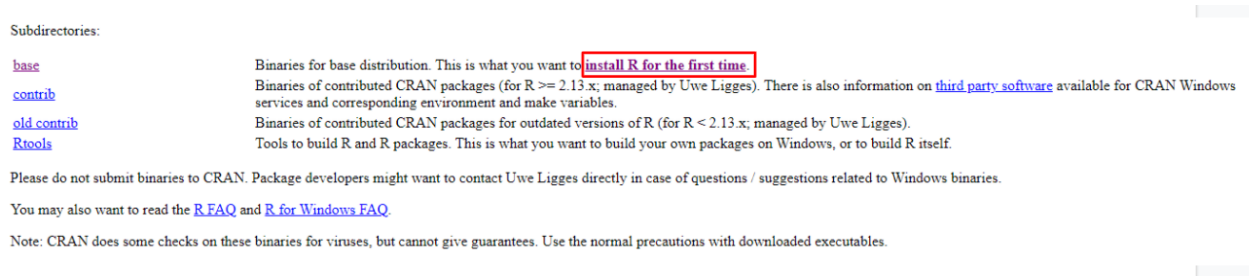
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2021-02-15, Lost Library Book) R-4.0.4.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

D. Windows - Click " install R for the first time."



Subdirectories:

base	Binaries for base distribution. This is what you want to install R for the first time.
contrib	Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

E. Windows - Click "Download R 4.0.4 for Windows "

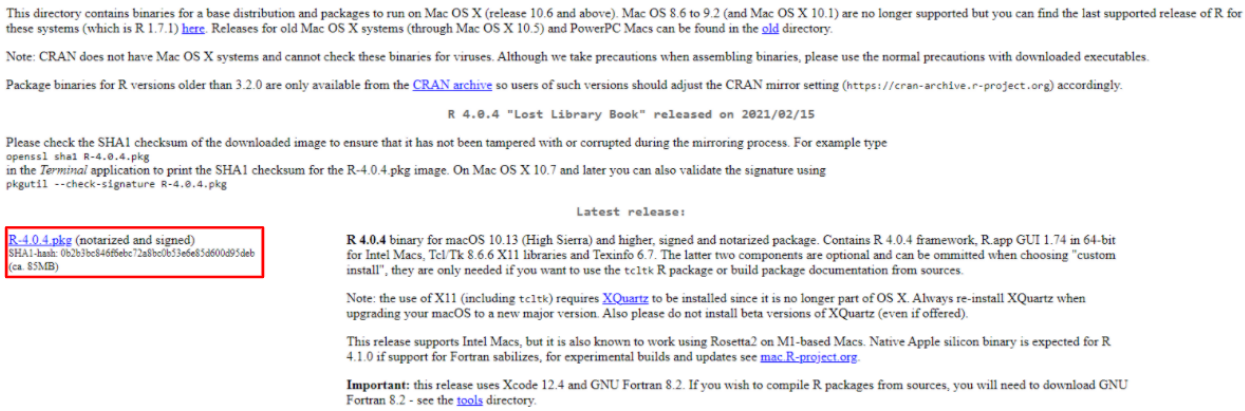
Download R 4.0.4 for Windows (85 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

F. Mac - Choose the latest release, which is usually the first option.



This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) here. Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the old directory.

Note: CRAN does not have Mac OS X systems and cannot check these binaries for viruses. Although we take precautions when assembling binaries, please use the normal precautions with downloaded executables.

Package binaries for R versions older than 3.2.0 are only available from the CRAN archive so users of such versions should adjust the CRAN mirror setting (https://cran-archive.r-project.org) accordingly.

R 4.0.4 "Lost Library Book" released on 2021/02/15

Please check the SHA1 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
openssl sha1 R-4.0.4.pkg
in the *Terminal* application to print the SHA1 checksum for the R-4.0.4.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
pkgutil --check-signature R-4.0.4.pkg

Latest release:

R-4.0.4.pkg (notarized and signed)
SHA1-hash: 0b2b3bc846f6ebc72a8bc0b53e6e85d600d95deb
(ca. 85MB)

R 4.0.4 binary for macOS 10.13 (High Sierra) and higher, signed and notarized package. Contains R 4.0.4 framework, R.app GUI 1.74 in 64-bit for Intel Macs, Tcl/Tk 8.6.6 X11 libraries and Texinfo 6.7. The latter two components are optional and can be ommitted when choosing "custom install", they are only needed if you want to use the tcltk R package or build package documentation from sources.

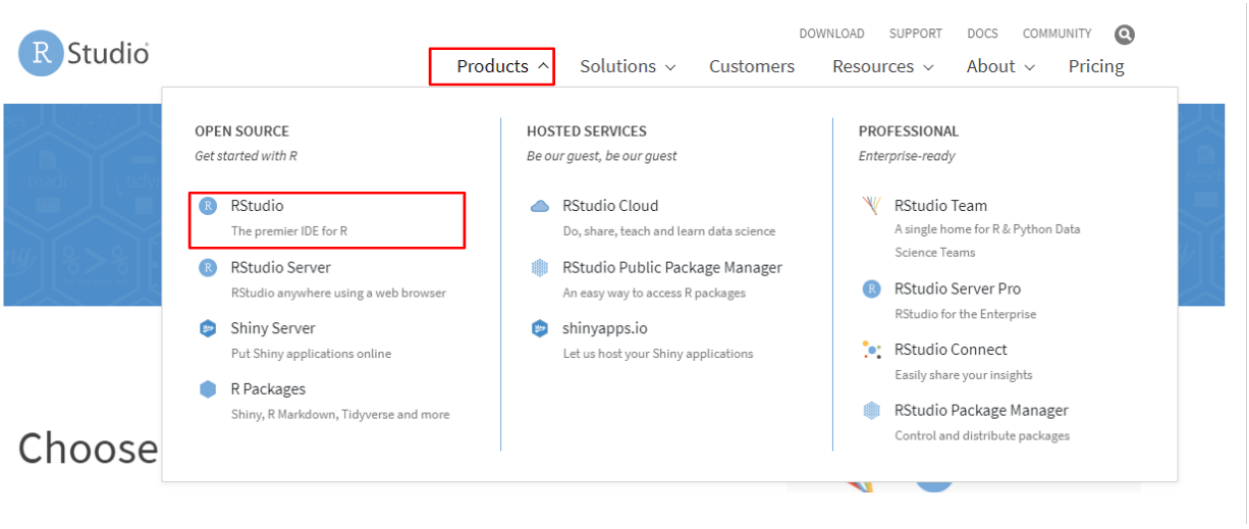
Note: the use of X11 (including tcltk) requires XQuartz to be installed since it is no longer part of OS X. Always re-install XQuartz when upgrading your macOS to a new major version. Also please do not install beta versions of XQuartz (even if offered).

This release supports Intel Macs, but it is also known to work using Rosetta2 on M1-based Macs. Native Apple silicon binary is expected for R 4.1.0 if support for Fortran sabilizes, for experimental builds and updates see mac.R-project.org.

Important: this release uses Xcode 12.4 and GNU Fortran 8.2. If you wish to compile R packages from sources, you will need to download GNU Fortran 8.2 - see the tools directory.

Installing R Studio

A. Go to https://rstudio.com/products/rstudio/download/. Click on "Products" and then "RStudio".



B. Scroll down the webpage to find RStudio Desktop, and then click Free Download.

R Studio Desktop

	Open Source Edition	RStudio Desktop Pro
Overview	• Access RStudio locally • Syntax highlighting, code completion, and smart indentation • Execute R code directly from the source editor • Quickly jump to function definitions • View content changes in real-time with the Visual Markdown Editor • Easily manage multiple working directories using projects • Integrated R help and documentation • Interactive debugger to diagnose and fix errors • Extensive package development tools	All of the features of open source; plus: • A commercial license for organizations not able to use AGPL software • Access to priority support • RStudio Professional Drivers • Connect directly to your RStudio Server Pro instance remotely
Support	Community forums only	• Priority Email Support • 8 hour response during business hours (ET)
License	AGPL v3	RStudio License Agreement
Pricing	Free	$995/year
	DOWNLOAD RSTUDIO DESKTOP	DOWNLOAD FREE RSTUDIO DESKTOP PRO TRIAL Purchase | Contact Sales

C. Scroll down the webpage and click Download. Then choose the package that is suitable for your operating system.

RStudio Desktop 1.4.1106 - Release Notes

1. Install R. RStudio requires R 3.0.1+.

2. Download RStudio Desktop. Recommended for your system:

DOWNLOAD RSTUDIO FOR WINDOWS
1.4.1106 | 155.97MB

Requires Windows 10/8/7 (64-bit)

All Installers

Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

OS	Download	Size	SHA-256
Windows 10/8/7	RStudio-1.4.1106.exe	155.97 MB	d2ff8453
macOS 10.13+	RStudio-1.4.1106.dmg	153.35 MB	c64d2cda

Installing R Packages

Here is the list of R packages we need to install for our Shiny application. Open "main.R" and **uncomment** all the codes that start with "install.packages(...)". Highlight these lines of code and press "Ctrl-Enter" or "Cmd+Return" on your keyboard. This will run the code individually from the app and install all the necessary packages we needed to run the Shiny application. After we finish installing, highlight the same lines of code, and press "Ctrl+Shift+C" or "Cmd+Shift+C" to comment the line selection.

```
# install.packages("shiny")
# install.packages("shinyWidgets")
# install.packages("shinydashboard”)
# install.packages("shinyalert")
# install.packages("shinyBS")
# install.packages("shinyjs”)
# install.packages("rintrojs")
# install.packages("readr")
# install.packages("readxl")
# install.packages("plyr")
# install.packages("tidyverse")
# install.packages("DT")
# install.packages("rlang")
# install.packages("plotly")
# install.packages("ggplot2")
```

General Setting

Setting The Path

Before we start running the Shiny application, we need to make some changes. We need to correct the path to where the data set, the dictionary, and the codes are located.

A. The path where the "results.csv" is located

In the main.R file, we can find this code in Line 47 . The code highlighted in yellow is the location of the "results.csv". Also, if users want to upload a new dataset, they can replace "results.csv" with some other csv file.

```
results <- read_csv("C:/Users/Owner/Downloads/results.csv")
```

B. The path where the dictionary file is located

In the main.R file, we can find this code in Line 56 . The code highlighted in yellow is the location of the dictionary file. As for the one highlighted in orange, it is the sheet where we will find the lookup table. The lookup table is where we will get the real names for the abbreviated characters (i.e. NIPA = National Income and Product Accounts).

```
dict <- read_excel("C:/Users/Owner/Downloads/results_real_names.xlsm",
                   sheet = "Lookup Table")
```

Also make sure that the lookup table is following this structure:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Simulation			Variable			Sector			Qualifier						
2	BaU	Business As Usual		NIPA	National Accounts		Val	Value		cons	Consumption					
3	ctax_debt	Carbon Price - Debt		sav	Savings		Vol	Quantity		govt	Government					
4	ctax_ftax	Carbon Price - Labor Tax		savf	Foreign investment in foreign currency terms		PrD	Price Deflator		invt	Investment					
5	RC85	RC85		rsg	Real government savings		Pct	Percentage Growth		ginvt	Government Investment					
6	RC85_ad	RC85 with Adaptation		inv	Investment		g-govt	Government		tdelst	Total Stock Change					
7				rinv	Real Investment		hhtot	Household Total		rtdelst	Aggregate real change in stocks					
8				rshrGDP	Real share of GDP		h-urQ1	Household Urban Quntile 1		pdelst	Aggregate change in stocks price deflator					
9				nshrGDP	Nominal share of GDP		h-urQ2	Household Urban Quntile 2		texp	Total Exports					
10				er	Exchange rate		h-urQ3	Household Urban Quntile 3		rtexp	Real Total Exports					
11				trb	Trade Balance		h-urQ4	Household Urban Quntile 4		ptexp	Unit Price of Exports					
12				fexp	Export Value		h-urQ5	Household Urban Quntile 5		timp	Import tax					
13				fimp	Import Value		h-ruQ1	Household Rural Quntile 1		rtimp	Real Total Imports					
14				inc	Income		h-ruQ2	Household Rural Quntile 2		ptimp	Unit Price of Imports					
15				ygov	Total government revenues		h-ruQ3	Household Rural Quntile 3		gdpmp	Nominal GDP at market price					
16				yg	Vector of government revenues		h-ruQ4	Household Rural Quntile 4		rgdpmp	Real GDP at market price					
17				htax	Household Income Tax		h-ruQ5	Household Rural Quntile 5		pgdpmp	GDP at market price deflator					
18				gexp	Government Expenditure		w-rowd	Rest of the World		rgdppc	Real GDP per capita					
19				yh	Total household income		blank	(blank)		ggdppc	Per capita GDP growth					
20				laby	Labor Income		i-insv	Investment		gl	Uniform labor productivity shifter					
21				uh	Household utility function		i-dstk	Stock Change		gdpfc	GDP at factor cost excl indirect taxes					
22				kapy	Gross profits		c-whet	Wheat		rgdpfc	Real GDP at factor cost excl indirect taxes					
23				transf	Transfers		c-maiz	Maize (Corn)		pgdpfc	GDP at factor cost excl indirect taxes deflator					
24				xp	Gross output		c-rice	Rice (Paddy)		trent	Economy wide rate of return to new capital					
25				toTR	Total Transfers		c-vege	Vegetables		blank	(blank)					
26				Pop	Population		c-ocer	Other Cereals		g-govt	Government					
27				exRates	Exchange Rates		c-frut	Fruits and Nuts		i-insv	Investment					
28				lst	Aggregate labor supply		c-oisd	Oilseeds and Oleaginous Fruits		i-adpPri	Private Adaptation Investment					
29				wage	Aggregate wage		c-pota	Potatoes and Sweet Potatoes		i-ginv	Government investment					
30				pf	Aggregate expenditure price index		c-spic	Stimulant, Spice, and Aromatic Crops		i-adpPub	Public Adaptation Investment					
31				kstock	Aggregate capital stock in level		c-puls	Pulses (Dried Leguminous Vegetables)		h-urQ1	Household Urban Quntile 1					
32				pnum	Choice of domestic deflator		c-sugr	Sugar Cane and Sugar Beet		h-urQ2	Household Urban Quntile 2					
33				ptland	Average price of land		c-fora	Forage Products, Fibers, Living Plants, Cut Flo		h-urQ3	Household Urban Quntile 3					
34				tland	Total land supply		c-cott	Cotton		h-urQ4	Household Urban Quntile 4					
35				kd	Demand for capital by vintage		c-ofbr	Other Fiber Crops		h-urQ5	Household Urban Quntile 5					
36				xpv	Output by vintage		c-lvst	Live Animals		h-ruQ1	Household Rural Quntile 1					
37				pk	Sectoral price of capital by vintage		c-milk	Raw Milk		h-ruQ2	Household Rural Quntile 2					
38				xf	Demand for factor f		c-anpr	Other Animal Products		h-ruQ3	Household Rural Quntile 3					
39				swage	Sectoral wage by skill		c-fore	Forestry and Logging Products		h-ruQ4	Household Rural Quntile 4					
40				rrat	Rate of return of old capital relative to new capital		c-fish	Fish and Other Fishing Products		h-ruQ5	Household Rural Quntile 5					
41				pp	Producer price		c-coal	Coal and Lignite; and Peat (minerals)		w-rowd	Rest of the World					
42				land	Land		c-petC	Crude Petroleum and Natural Gas (Distributior		PLT15	Population age 15 and below					
43				pland	Price of land by sector		c-mine	Other Minerals		P1564	Population age 15-64					

C. The path where the "WorldBank-IndustryProject" folder is located

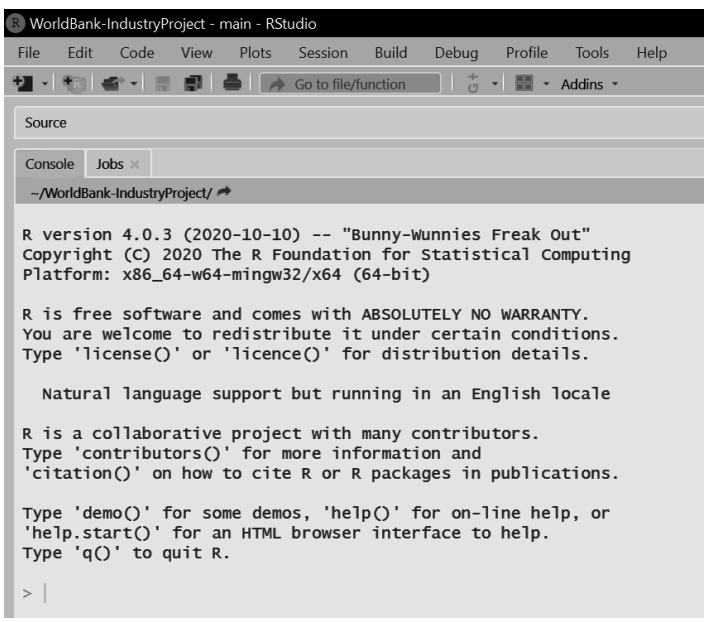
In the main.R file, we can find this code in Line 82 . Replace the code highlighted in yellow to the location where you store the "WorldBank-IndustryProject" folder. This folder stores all the codes we needed to run the Shiny application. For example, main.R, server.R, and etcetera.

```
setwd("C:/Users/Owner/Documents/WorldBank-IndustryProject")
```

Run The Shiny Application

There are two ways to run the application in RStudio and get the dashboard.

Run in the R console



Open RStudio. We first have to download and load the "Shiny" package in the console to get the basic function that we need for the next step. Here are the command lines:

```
install.packages("shiny")
library(shiny)
```

Then, we can run this "runApp" function to run the dashboard. One thing to make sure is the path is where the dashboard is located. After running this code in the console the dashboard will come out.

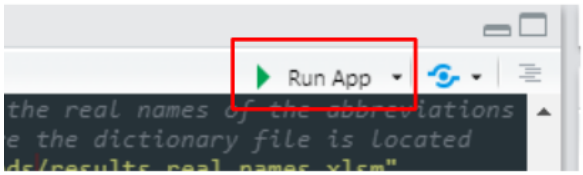
These are two choices we can run. Both will give us the same dashboard.

```
runApp('C:/Users/Owner/Downloads/WorldBank-IndustryProject')
```

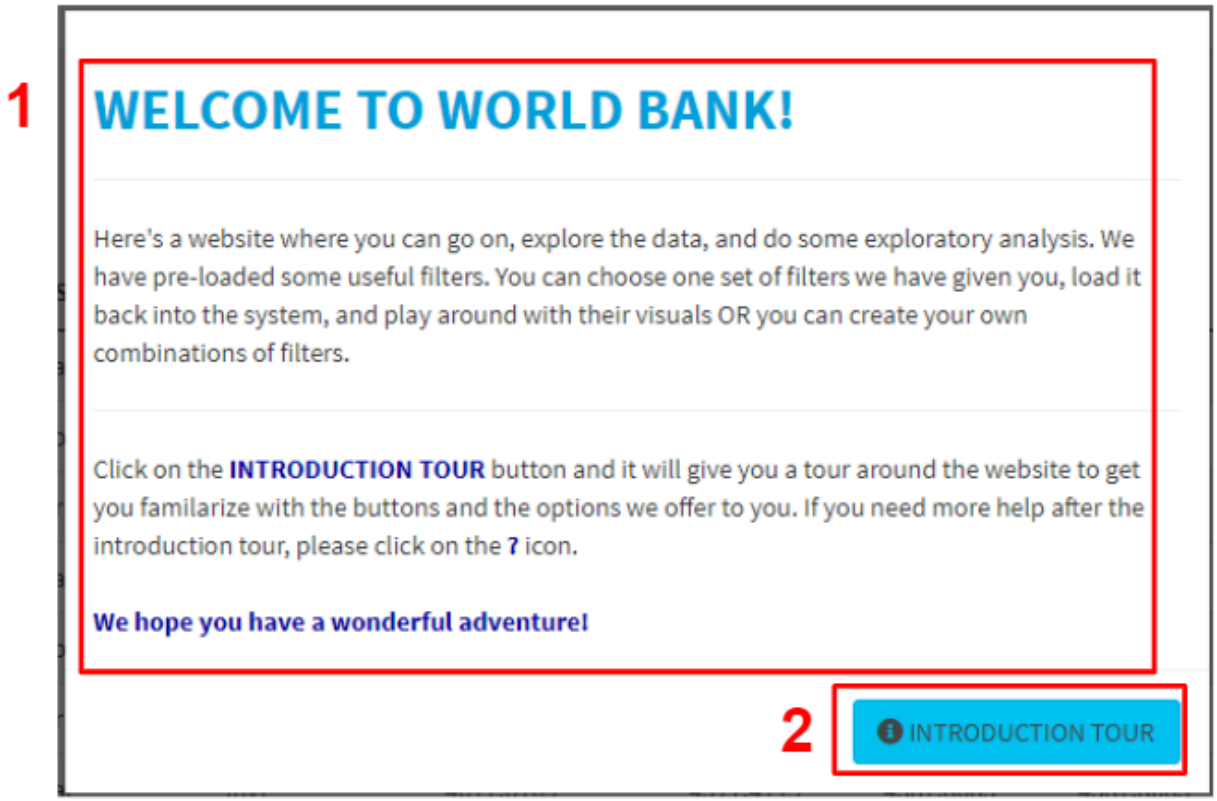
```
runApp('C:/Users/Owner/Downloads/WorldBank-IndustryProject/main.R')
```

Run in the RStudio

Open the “main.R” in the file, and then click the “Run App” button on the top right to pop out the dashboard.



Edit Welcome Page



1. Edit The Text

Open the R file called "intro_Help_Pages.R". On your keyboard, press "Ctrl+F" or "Cmd+F" and search for "welcome_page" (Line 17). If you want to change the text, format style, or add a new line in the welcome page, you can make the changes in this block of code.

In this code, we use basic HTML to build the text information. For example, we can set the color of the text with code " ", or add a line with <br>.

In the following code, we use yellow to highlight the welcome page variable, and orange to highlight the text that can be changed easily. The rest of the code is in the HTML format that helps customize the words style.

```
welcome_page <- HTML(
```

```
            "<h2><span style='color: #009FDA;'><b>WELCOME TO WORLD
            BANK!</b></span> <br> <hr></h2>

            Here's a website where you can go on, explore the data,
            and do some exploratory analysis. We have pre-loaded some
            useful filters. You can choose one set of filters we have
            given you, load it back into the system, and play around
            with their visuals OR you can create your own
            combinations of filters. <br> <hr>

            Click on the <span style='color:
            #0000A0;'><b>INTRODUCTION TOUR</b></span> button and it
            will give you a tour around the website to get you
            familiarized with the buttons and the options we offer to
            you. If you need more help after the introduction tour,
            please click on the <span style='color:
            #0000A0;'><b>?</b></span> icon. <br> <br>

            <span style='color: #0000A0;'><b>We hope you have a
            wonderful adventure!</b></span>"

        )
```

2. Edit The Style Of The Introduction Button

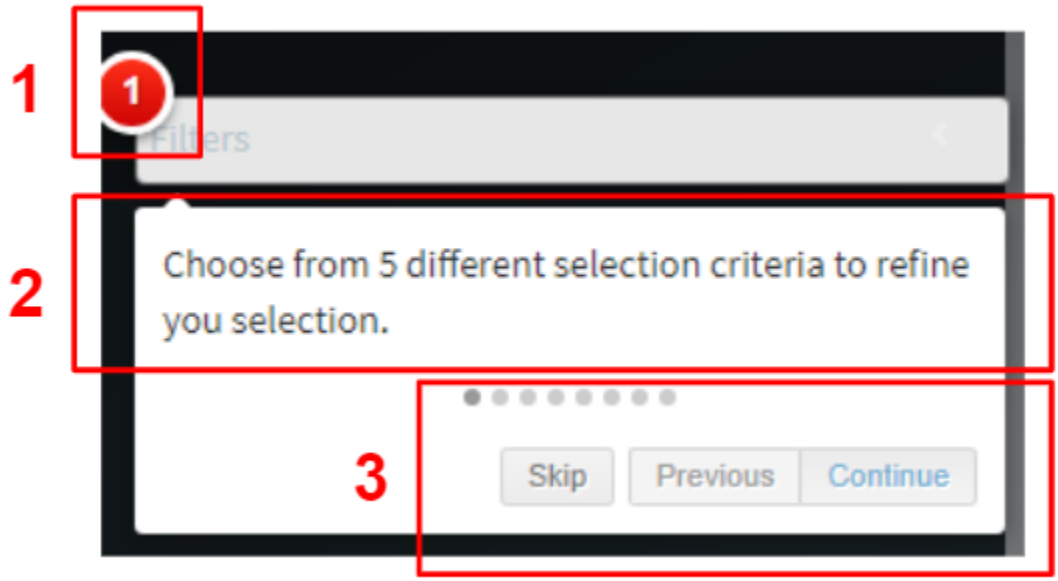
In box 2, the button style can also be customized. Open the R file called "server.R". On your keyboard, press "Ctrl+F" or "Cmd+F" and search for "welcome_page" in Line 19. The code inside the observeEvent function will trigger when the Shiny application starts running. That's why the first thing you see on the dashboard is a welcome page.

In this part we will focus on customizing the button style and its icon. We use yellow to highlight the content of the welcome page and orange to highlight the UI function for the action button. We can set up the "label" to change the button text, and the "icon" to change different icons.

Here is the list of icons we can use:
https://fontawesome.com/icons?d=gallery&p=2&m=free

```
observeEvent("", {
  showModal(modalDialog(
    welcome_page,
    easyClose = TRUE,
    footer = tagList(

      # Click on button to have a introduction tour
      actionButton(inputId = "intro", label = "INTRODUCTION TOUR",
                   icon = icon("info-circle"), class = "btn-info")

    )
  ))
})
```

Edit The Introduction Tutorial



1. Edit The Steps

A. Reordering the steps

Open the R file called "ui.R". On your keyboard, press "Ctrl+F" or "Cmd+F" and search for "introBox" (Line 228). Any elements that are wrapped around by the introBox() function will be included in the introduction tutorial. If you want to change the order the elements are introduced, change the number "data.step" is set to.

In the example code below, the buttons that direct you to the different panels on the dashboard are introduced in step 4.

```
column(
          width = 5,
          introBox(
            bsButton("original",
                     label = "ORIGINAL DATA",
                     icon = icon("database"),
                     style = "primary"),
            bsButton("transformed",
```

```
                      label = "TRANSFORMED DATA",
                      icon = icon("spinner", class =
"spinner-box"),
                      style = "primary"),
            data.step = 4, data.intro = intro$text[4]
          ) # End of the introBox() function
), # End of the column() function
```

If you change the order of the steps, don't forget to update the "data.step" argument for all other steps wrapped by the introBox() function. It's because the number we used for "data.step" must be unique for each introduction box.

It is also suggested to change the order of the text content for the introduction tutorial. That way the content we use, defined in the "data.intro" argument, is saved in the same order each step is introduced.

Open the file, "Intro_Help_Pages.R". Search for the code that creates a data frame object called "intro" in Line 3. In the code below, the yellow-highlighted region is the content for step 4 in the introduction tutorial. So if we change the "data.step" above to step 5, we need to swap the content for step 4 (highlighted in yellow) with the content with step 5 (highlighted in grey) to change the order.

```
intro <- data.frame(
  step = c(1:8),
  text = c("Choose from 5 different selection criteria
           to refine you selection.",

           "Here are some graph options you can choose to
           generate in the <b>ORIGINAL DATA</b> panel. Don't
           forget to click on <b>CREATE</b> if you want the
           visuals to show up.",

           "Here are the options to transform the values to
           percentage or other formats. A table  or graphs will
           appear in the <b>TRANSFORMED DATA</b> panel once a
           selection is made",

           "You start automatically in the <b>ORIGINAL DATA</b>
           panel. The first object you will see in this panel
```

```
            is the full dataset. Check the other panel for
            transformed data value",

            "Here's the original data set. Once, you select some
            filters, the table will automatically update. On the
            top, there are options to download and copy the data
            that is on this visible page.",

            "You can save the filters for later use. Don't
            forget to name the set of filters",

            "You can reuse the filters you saved previously and
            load it back in. There's also an option to delete
            whatever you saved",

            "Here are some helpful instructions to navigate the
            app if you get lost."
           )
)
```

B. Deleting a step

Similarly, if you want to take out element(s) from the introduction tutorial, we need to find the element in the code and take out the introBox() function.

Let's say we want to take out the data table from the introduction tutorial. Open the file, "ui.R", and press on "Ctrl+F" or "Cmd+F". Search for "introBox" and press next until we find the line of code where an introBox() function is wrapped around the data table element.

```
fluidRow(
        div(
          id = "original_panel",
          column(width = 12,
                 conditionalPanel(
                   condition = "input.all_graphs > 0",
                   plotlyOutput("plot")
                 ),
                 br(),
```

```
                  introBox(data.step = 5,
                           data.intro = intro$text[5],
                           dataTableOutput("table")),
          ) # End of column()
        ) # End of div()
), # End of fluidRow() - Original panel
```

After finding the code that specifies the data table as one of the elements to be introduced, take out all codes relating to the introBox() function. But keep the element, which is the data table for this example. See below for an example. We will be taking out everything that is striked out and is highlighted yellow.

```
fluidRow(
        div(
          id = "original_panel",
          column(width = 12,
                 conditionalPanel(
                   condition = "input.all_graphs > 0",
                   plotlyOutput("plot")
                 ),
                 br(),
                 introBox(data.step = 5,
                          data.intro = intro$text[5],
                          dataTableOutput("table")),
          ) # End of column()
        ) # End of div()
), # End of fluidRow() - Original panel
```

The code should now look like this. Only the data table elements remain in the code.

```
fluidRow(
        div(
          id = "original_panel",
          column(width = 12,
                 conditionalPanel(
                   condition = "input.all_graphs > 0",
                   plotlyOutput("plot")
```

```
                    ),
                    br(),
                    dataTableOutput("table"),
            ) # End of column()
          ) # End of div()
), # End of fluidRow() - Original panel
```

Given that we have taken out an element from the introduction tutorial, we need to update the "data.step" argument in all remaining introduction steps.

For instance, we took out step 5, the data table, from the introduction tutorial. Now what was originally designated as step 6 will become step 5.

Additionally, since we took out an introduction step, it is suggested that we also update the content for the introduction steps. Open the file, "Intro_Help_Pages.R". Search for the "intro" data frame in Line 3. Take out the content for step 5 in the intro data frame and update the argument, "data.intro", for all remaining steps. Now there are a total of 7 steps remaining, instead of 8.

```
intro <- data.frame(
  step = c(1:7),
  text = c("Choose from 5 different selection criteria
           to refine you selection.",

           "Here are some graph options you can choose to
           generate in the <b>ORIGINAL DATA</b> panel. Don't
           forget to click on <b>CREATE</b> if you want the
           visuals to show up.",

           "Here are the options to transform the values to
           percentage or other formats. A table  or graphs will
           appear in the <b>TRANSFORMED DATA</b> panel once a
           selection is made",

           "You start automatically in the <b>ORIGINAL DATA</b>
           panel. The first object you will see in this panel
           is the full dataset. Check the other panel for
           transformed data value",
```

```
            ~~"Here's the original data set. Once, you select some filters, the table will automatically update. On the top, there are options to download and copy the data that is on this visible page.",~~

            "You can save the filters for later use. Don't forget to name the set of filters",

            "You can reuse the filters you saved previously and load it back in. There's also an option to delete whatever you saved",

            "Here are some helpful instructions to navigate the app if you get lost."
           )
)
```

C. Adding a step

To add steps to the introduction tutorial, we will wrap the introBox() function around the element. Open the file, "ui.R". If you are adding a new step in the middle of the introduction tour you will need to rearrange the order of the steps. Please refer to Reordering the steps above. But if it is added at the end of the tour, then we don't need to rearrange the order.

If the element is a menu item on the sidebar, like "Plot Options", wrap an introBox() function around the text.

```
menuItem(
          text = introBox("Plot Options",
                          data.step = 2,
                          data.intro = intro$text[2]),
          tabName = "plot_options",

          # Create the choices for the type of visuals you
          want to create

           selectizeInput(inputId = "plots", label = "Plot
```

```
          Options", choices = c("bar", "stack", "histogram",
          "line", "area", "bubble", "scatter", "boxplot")),

          selectizeInput(inputId = "x_val", label = "Please
          choose a variable as X", choices = colnames(results),
          selected = "Year"),

          selectizeInput(inputId = "y_val", label = "Please
          choose a variable as Y", choices = colnames(results),
          selected = "Value"),

          selectizeInput(inputId = "facet", label = "Please
          choose a variable to facet by", choices =
          colnames(results)
)
```

If the element is a drop-down button/menu like the “Save” options in Line 253, wrap the introBox() around the icon.

```
dropdownButton(
            inputId = "reuse_saved_filters",
            inline = T,
            size = "sm",
            right = T,
            icon = introBox(icon("recycle"),
                            data.step = 7,
                            data.intro = intro$text[7]),
            ........................
)
```

If the element is other than what we specified above, wrap the introBox() function around the whole UI element like in Line 228.

```
column(
          width = 5,
          introBox(
            bsButton("original",
                     label = "ORIGINAL DATA",
                     icon = icon("database"),
```

```
                        style = "primary"),
              bsButton("transformed",
                        label = "TRANSFORMED DATA",
                        icon = icon("spinner", class =
"spinner-box"),
                        style = "primary"),
              data.step = 4, data.intro = intro$text[4]
            ) # End of the introBox() function
), # End of the column() function
```

Once we are done wrapping the introBox() around the element(s), we need to specify the content for the new step. Open the “Intro_Help_Pages.R" file. Search for the “intro” data frame and add in a new row and specify the content. If we added in one new step, then we now have 9 introduction steps below.

```
intro <- data.frame(
  step = c(1:9),
  text = c("Choose from 5 different selection criteria
           to refine you selection.",

           "Here are some graph options you can choose to
           generate in the <b>ORIGINAL DATA</b> panel. Don't
           forget to click on <b>CREATE</b> if you want the
           visuals to show up.",

           "Here are the options to transform the values to
           percentage or other formats. A table  or graphs will
           appear in the <b>TRANSFORMED DATA</b> panel once a
           selection is made",

           "You start automatically in the <b>ORIGINAL DATA</b>
           panel. The first object you will see in this panel
           is the full dataset. Check the other panel for
           transformed data value",

           “The New Content”,
```

```
            "Here's the original data set. Once, you select some
            filters, the table will automatically update. On the
            top, there are options to download and copy the data
            that is on this visible page.",

            "You can save the filters for later use. Don't
            forget to name the set of filters",

            "You can reuse the filters you saved previously and
            load it back in. There's also an option to delete
            whatever you saved",

            "Here are some helpful instructions to navigate the
            app if you get lost."
           )
)
```

2. Edit The Content For The Introduction Steps

Open the "Intro_Help_Pages.R" file. Search for the "intro" data frame object. Right now, there are a total of 8 steps. If you want to change the content for any of the existing steps, edit the text highlighted in yellow. But if you want to add, delete, or reorder the steps, please refer to this section, Editing The Steps.

```
intro <- data.frame(
  step = c(1:8),
  text = c("Choose from 5 different selection criteria
           to refine you selection.",

           "Here are some graph options you can choose to generate in
           the <b>ORIGINAL DATA</b> panel. Don't forget to click on
           <b>CREATE</b> if you want the visuals to show up.",

           "Here are the options to transform the values to
           percentage or other formats. A table  or graphs will
           appear in the <b>TRANSFORMED DATA</b> panel once a
           selection is made",
```

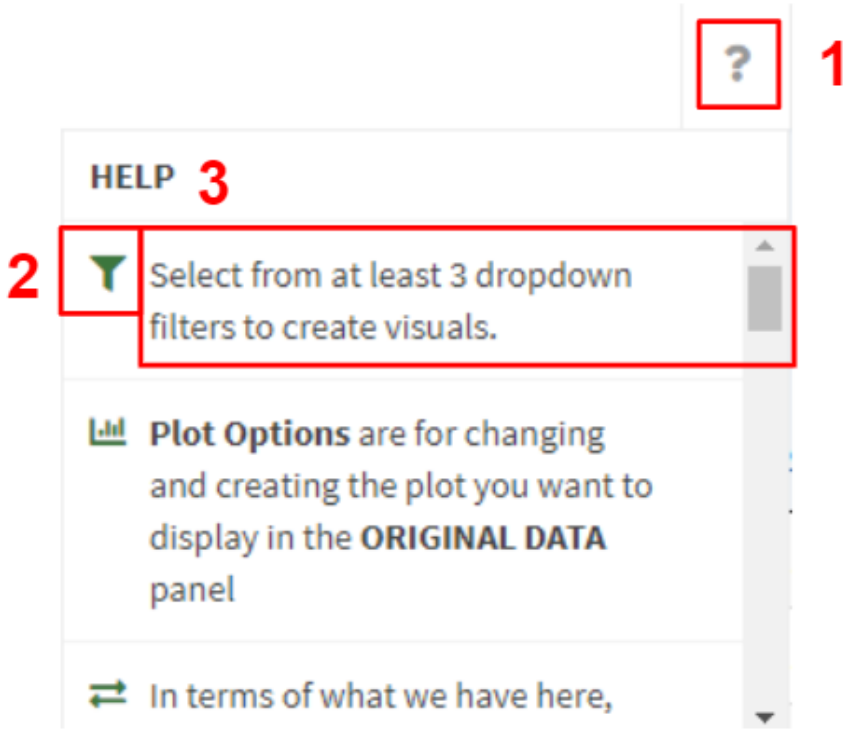
```
            "You start automatically in the <b>ORIGINAL DATA</b>
            panel. The first object you will see in this panel is the
            full dataset. Check the other panel for transformed data
            value",

            "Here's the original data set. Once, you select some
            filters, the table will automatically update. On the top,
            there are options to download and copy the data that is on
            this visible page.",

            "You can save the filters for later use. Don't forget to
            name the set of filters",

            "You can reuse the filters you saved previously and load
            it back in. There's also an option to delete whatever you
            saved",

            "Here are some helpful instructions to navigate the app if
            you get lost."
          )
)
```

3. Edit The Text On The Buttons

Open the "server.R" file. On your keyboard, press "Ctrl+F" or "Cmd+F" and search for "input$intro". If we want to change the labels on the buttons for the introduction box, then we will edit the strings that are highlighted in yellow.

```
observeEvent(input$intro, {
  introjs(session, options = list("nextLabel" = "Continue",
                                  "prevLabel" = "Previous",
                                  "doneLabel" = "Alright. Let's go"))
})
```

Edit The Help Dropdown Menu



1. Edit The Question Icon

Open the "ui.R" file. On our keyboard, press "Ctrl+F" or "Cmd+F" and search for "help". You will find a "HELP" within the drop-down menu object in Line 19. If you want to change the icon to something besides a question mark, then change the code highlighted in orange under the icon argument.

Here is the list of icons we can use:
https://fontawesome.com/icons?d=gallery&p=2&m=free

```
type = "notifications",
       headerText = strong("HELP"),
       icon = introBox(data.step = 8,
                       data.intro = intro$text[8],
                       icon("question")),
       badgeStatus = NULL,
```

2. Edit The Icon Inside

Open the “ui.R” file. On our keyboard, press “Ctrl+F” or “Cmd+F” and search for “notificationItem”. If you want to change the icon to something besides a plus sign for the 4th help item, then change the code highlighted in orange under the icon argument. The steps will be the same if you want to change the icons for the other help items.

Here is the list of icons we can use:
https://fontawesome.com/icons?d=gallery&p=2&m=free

```
notificationItem(
    text = wrap_text(help$text[1]),
    icon = icon("filter"),
),
notificationItem(
    text = wrap_text(help$text[2]),
    icon = icon("chart-bar")
),
notificationItem(
    text = wrap_text(help$text[3]),
    icon = icon("exchange-alt")
),
notificationItem(
    text = wrap_text(help$text[4]),
    icon = icon("plus")
),
notificationItem(
    text = wrap_text(help$text[5]),
    icon = icon("download")
),
notificationItem(
    text = wrap_text(help$text[6]),
    icon = icon("camera")
),
notificationItem(
    text = wrap_text(help$text[7]),
    icon = icon("save")
),
notificationItem(
    text = wrap_text(help$text[8]),
```

```
    icon = icon("recycle")
)
```

3. Edit The Text

Open the "Intro_Help_Pages.R" file. Search for the "help" data frame object. Right now, there are a total of 8 items. If you want to change the content for any of the existing items, edit the text highlighted in yellow. But if you want to add or delete the items, please refer to this section, Add/Delete Help Items .

```
help <- data.frame(
  step <- c(1:8),
  text <- c("Select from at least 3 dropdown filters to create visuals.",

            "<b>Plot Options</b> are for changing and creating the plot you
            want to display in the <b>ORIGINAL DATA</b> panel",

            "In terms of what we have here, <b>TRANSFORM</b> means that the
            values in the data are converted into another format or plugged
            into a formula and created a new value. So the <b>Transform
            Tables and Graph Options</b> are for changing and creating the
            table and plot you want to display in the <b>TRANSFORMED
            DATA</b> panel.",

            "Click on the <b>Create</b> button to generate all visuals and
            transformed table in the <b>ORIGINAL</b> and <b>TRANSFORMED</b>
            data panel.",

            "<b>You can only download the data that is on the visible
            page.</b> For example, if the table only shows 10 rows, the csv
            or excel files will only saved those 10 rows. If you want to
            save and display more than 10 rows in the data table, click on
            the <b>Show 10 rows</b> button and click the other length
            options.",

            "Hover your mouse on the top-right corner of the graphs and
            click on the <b>Camera</b> icon. This allows you to download
            the static form of the plot in a png file. Feel free to explore
            the other icons next to the <b>Camera</b> icon.",

            "If you want to save the inputs and filters you have selected,
```

```
            click on the <b>Save</b> icon in the top-right corner and type
            in a name and click <b>Save</b>. If the name you use is already
            in the system, you have the option to click <b>Cancel</b> and
            choose another name or <b>Update</b> the one you already
            have.",

            "If you want to load back in the filters you have previously
            saved, click on the <b>Recycle</b> icon in the top-right corner
            and choose the one you want and click <b>Load</b>. Once the
            filters load back in successfully, click on <b>CREATE</b> again
            to see all visuals. But if you want to delete what you have
            previously saved, click on the <b>Delete</b> button instead."

          )
)
```

4. Add/Delete Help Items

Open the "ui.R" file. On our keyboard, press "Ctrl+F" or "Cmd+F" and search for "help". You will find a "HELP" within the drop-down menu object in Line 18.

In order to add in a new help item, we need to create a new notifactionItem() in the code (like the one highlighted in orange). The notificationItem() function has two arguments -- text and icon. Text is for the content and icon is for the image printed next to the text.

But if we want to delete a help item, we will instead delete the notificationItem() in orange.

```
dropdownMenu(
        type = "notifications",
        headerText = strong("HELP"),
        icon = introBox(data.step = 8,
                        data.intro = intro$text[8],
                        icon("question")),
        badgeStatus = NULL,
        notificationItem(
          text = wrap_text(help$text[1]),
          icon = icon("filter"),
        ),
        notificationItem(
```

```
          text = wrap_text(help$text[2]),
          icon = icon("chart-bar")
        ),
```

To complete the addition or deletion, we also need to make an edit in the "Intro_Help_Pages.R" file. On our keyboard, press "Ctrl+F" or "Cmd+F" and search for the "help" data frame object.

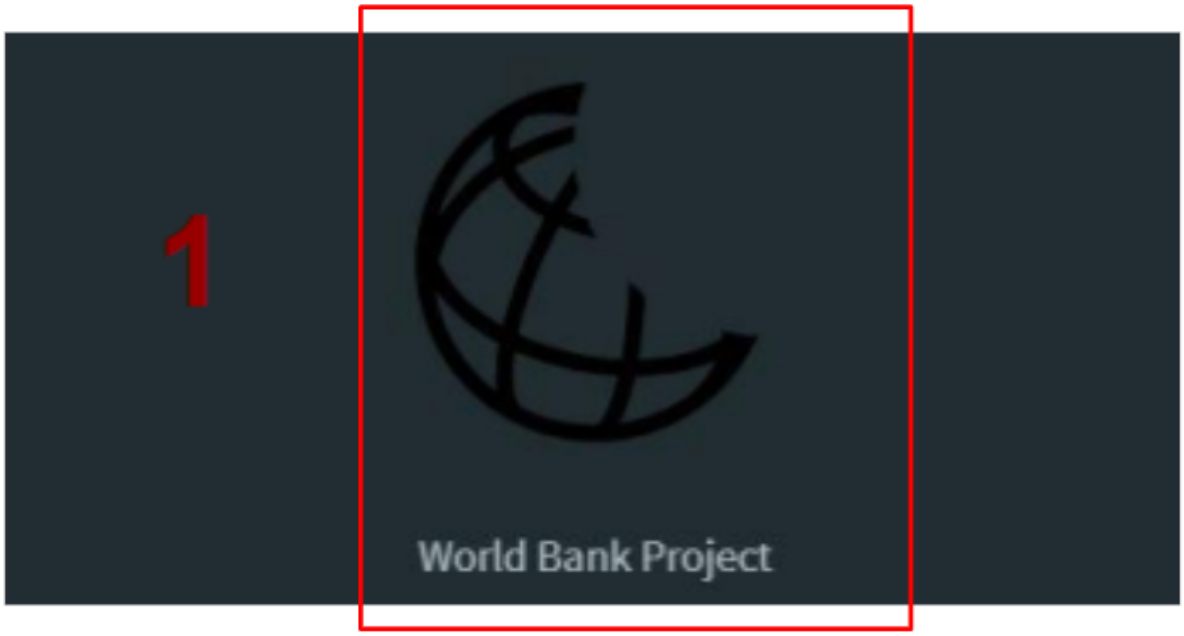
Since we deleted a help item, we will delete its content highlighted in orange from the "help" data frame. But if we added a new notification item, we will instead add the content for this new item into this data frame.

Once we are done, we will also update the number of steps, the one highlighted in yellow. If we deleted an item, the number 8 would change to 7. Else it will change to 9.

```
help <- data.frame(
  step <- c(1:8),
  text <- c(
            "NEW CONTENT -- Select from at least 3 dropdown filters to
            create visuals.",

            "<b>Plot Options</b> are for changing and creating the plot you
            want to display in the <b>ORIGINAL DATA</b> panel",

            "In terms of what we have here, <b>TRANSFORM</b> means that the
            values in the data are converted into another format or plugged
            into a formula and created a new value. So the <b>Transform
            Tables and Graph Options</b> are for changing and creating the
            table and plot you want to display in the <b>TRANSFORMED
            DATA</b> panel.",

            "Click on the <b>Create</b> button to generate all visuals and
            transformed table in the <b>ORIGINAL</b> and <b>TRANSFORMED</b>
            data panel.",

            "<b>You can only download the data that is on the visible
            page.</b> For example, if the table only shows 10 rows, the csv
            or excel files will only saved those 10 rows. If you want to
            save and display more than 10 rows in the data table, click on
            the <b>Show 10 rows</b> button and click the other length
            options.",
```

```
            "Hover your mouse on the top-right corner of the graphs and
            click on the <b>Camera</b> icon. This allows you to download
            the static form of the plot in a png file. Feel free to explore
            the other icons next to the <b>Camera</b> icon.",

            "If you want to save the inputs and filters you have selected,
            click on the <b>Save</b> icon in the top-right corner and type
            in a name and click <b>Save</b>. If the name you use is already
            in the system, you have the option to click <b>Cancel</b> and
            choose another name or <b>Update</b> the one you already
            have.",

            "If you want to load back in the filters you have previously
            saved, click on the <b>Recycle</b> icon in the top-right corner
            and choose the one you want and click <b>Load</b>. Once the
            filters load back in successfully, click on <b>CREATE</b> again
            to see all visuals. But if you want to delete what you have
            previously saved, click on the <b>Delete</b> button instead."

          )
)
```

Edit The Logo And The Link in the Sidebar



If we click on the image, we'll be directed to the specific content we specified. By default, we'll be directed to the World Bank's home page. To change the logo and the link, please follow the steps in this section.

1. Change The Logo Image And The Link

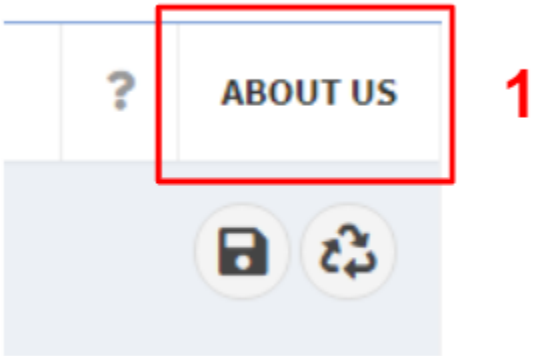
Open the R file called "ui.R". On your keyboard, press "Ctrl+F" or "Cmd+F" and search for "sidebarMenu" in Line 76. In the beginning of the function, we can see there is a HTML function that would include HTML code to set up the logo and the link in the sidebar.

First of all, we need to check whether there is a "www" folder somewhere in the client's computer. If it exists and the folder is not saved under the same folder as main.R, server.R, and the other code files, we need to move the "www" folder to the same directory. After that, we can place the new image file into the "www" folder.

Lastly, we need to set the "src" argument (which is the first orange highlight) to the new image file name. Other than the logo, we can see other orange highlights that focus on a link and the caption. These can also be changed. All we need to do is put a new link and the new text in the same place.

```
sidebarMenu(
        HTML(paste0(
        "<br>",
        "<a href='https://www.worldbank.org/en/home' target='_blank'><img
style = 'display: block; margin-left: auto; margin-right: auto;'
src='world.svg' width = '186'></a>",
        "<br>",
        "<p style = 'text-align: center;'><small><a
href='https://www.worldbank.org/en/where-we-work' target='_blank'> World
Bank Project</a></small></p>",
        "<br>"
      )),
```

Edit ABOUT US



1. Change The Title And Link

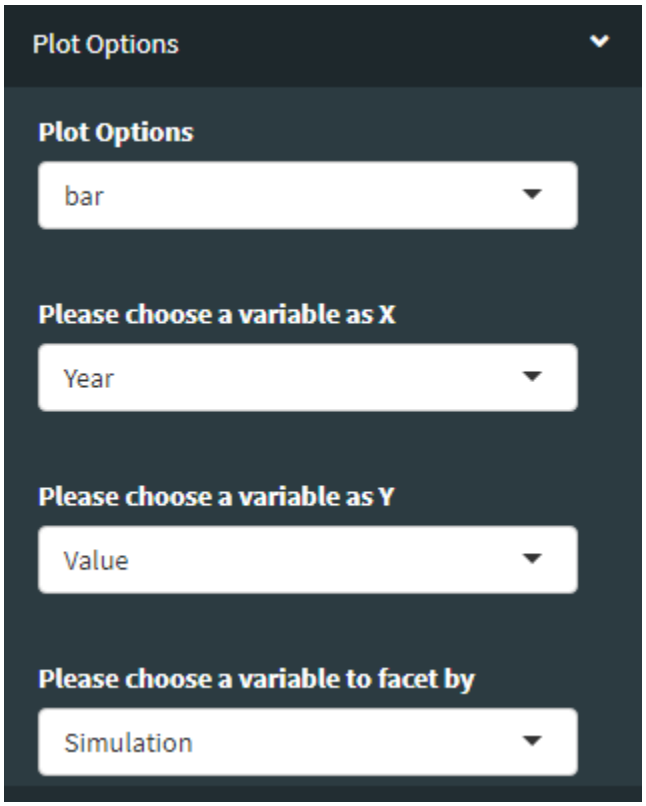
Open the R file called "ui.R". On your keyboard, press "Ctrl+F" or "Cmd+F" and search for "tags$li" in Line 60. Here are the steps to change the title and the link that are highlighted in orange. For the title, if you want to keep the text unbolded, then take out the strong() function and leave the text. If you want to change the text to something besides "ABOUT US", then replace the lettering in between the quotation marks. As for changing the link, replace the https address in the "href" argument.

```
tags$li(
  a(
    strong("ABOUT US"),
    height = 40,
    href = "https://www.worldbank.org/en/home",
    title = "",
    target = "_blank"
),
```

Graph Options

Adding More Graphs

In order to add new plot types to the dashboard, we need to complete the 2 steps below.



A. Adding the new graphic option into the sidebar

Open the R file called "ui.R". On your keyboard, press "Ctrl+F" or "Cmd+F" and search for "Plot Options". For example, if we want to create a new plot type called "boxplot", we first need to add "boxplot" as a new choice in **two** drop-down lists. In these selectizeInput functions, add the "boxplot" options into their "choices" lists. Now we can see "boxplot" as one of the new plot options in the sidebar.

```
menuItem(
            introBox("Plot Options",
                     data.step = 2,
                     data.intro = intro$text[2]),
```

```
            tabName = "plot_options",

            # select the type of visuals you want to create
            selectizeInput(inputId = "plots", label = "Plot Options", choices
            = c("bar", "stack", "histogram", "line", "area", "pie", "bubble",
            "scatter", "boxplot")),

            selectizeInput(inputId = "x_val", label = "Please choose a
            variable as X", choices = colnames(results), selected = "Year"),

            selectizeInput(inputId = "y_val", label = "Please choose a
            variable as Y", choices = colnames(results), selected = "Value"),

            selectizeInput(inputId = "facet", label = "Please choose a
            variable to facet by", choices = colnames(results))
),
```

```
conditionalPanel(
     condition = "input.displayTransf == 'Graph'",
     conditionalPanel(
     condition = "input.typeConv != 'Proportion of Sector Per Year'",
     selectizeInput(inputId = "plotType", label = "Plot Options",
                    choices = c("bar", "stack", "histogram", "line",
                                "area", "bubble", "scatter","boxplot"))
     ),
```

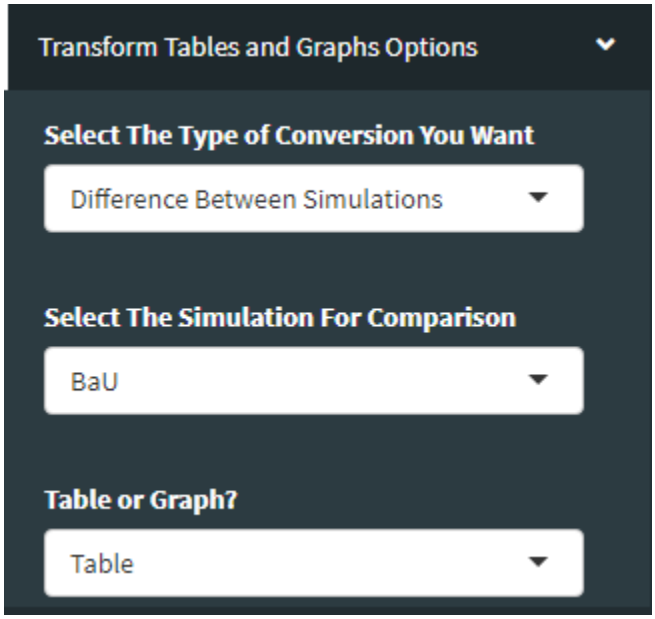
B. Adding new graph options

Open the R file called "graphs.R". For example, if we want to create a new plot type called "boxplot", we will write some code to produce the plot. We will first search for the "graphs" function, and then create a new "if statement". Inside the if statement, we will construct a plot object using the ggplot function from the ggplot2 package. The content inside the if statement will only execute when the condition, "plot_type == "boxplot"", evaluates to true. This means the user wants to create a boxplot.

```
 graphs <- function(dat, plot_type, x, y, facet){
    if(plot_type == "bar"){
      g <- ggplot(dat, aes(fill = as.factor(UQ(as.name(facet))),
                           text = paste0(facet, ": ",
as.factor(UQ(as.name(facet)))),
```

```
                            y = UQ(as.name(y)),
                            x = UQ(as.name(x)))) +
        geom_bar(position ="dodge", stat = "identity") +
        scale_fill_brewer(palette = "Blues") +
        guides(fill = guide_legend(nrow = 1, byrow = TRUE, title = facet)) +
        theme_minimal() +
        theme(legend.position = "bottom", legend.justification = "left")
      p <- ggplotly(g, tooltip = c("x", "y", "text"))
    }
    if(plot_type == "boxplot"){
      g <- ggplot(dat, aes(fill = as.factor(UQ(as.name(facet))),
                            text = paste0(facet, ": ",
as.factor(UQ(as.name(facet)))),
                            y = UQ(as.name(y)),
                            x = UQ(as.name(x)))) +
        geom_bar(position ="stack", stat = "identity") +
        scale_fill_brewer(palette = "Blues") +
        guides(fill = guide_legend(nrow = 1, byrow = TRUE, title = facet)) +
        theme_minimal() +
        theme(legend.position = "bottom", legend.justification = "left")
      p <- ggplotly(g, tooltip = c("x", "y", "text"))
    }
```

Transformation Options



Adding More Formulas

In order to add in new formulas to the dashboard, we need to complete the 2 steps below.

A. Adding a new formula option in the sidebar

Open the R file called “ui.R”. On your keyboard, press “Ctrl+F” or “Cmd+F” and search for “typeConv” in Line 153. For example, if we want to create a new formula called "Percentage”, we first need to add “Percentage” as a new choice in the drop-down list. In this selectizeInput function, add in “Percentage” as one of the new choices in the “choices” argument. Now percentage will show up as an option in the sidebar.

```
selectizeInput(inputId = "typeConv",
               label = "Select The Type of Conversion You Want",
               choices = c("Difference Between Simulations",
                           "Percent Difference Between Simulations",
                           "LCU to USD Per Billion Ton",
                           "Proportion of Sector Per Year",
                           "Percentage")),
```

B. Adding the new formulas

Open the R file called “formulas.R”. You will see a function called “transf”. This is the function that performs the conversion on the value.

For example, if we want to create a new formula called "Percentage”, then we need to write some code. We will find the “transf” function and then add in a new “if statement”. The condition for the if statement will be: “formula == “Percentage””. In between the brackets we would write the code that transforms the value to the percentage format.

```
 transf <- function(sim1, sim2, var, sec, qual, yr, formula){
    val_sim1 <- res_mod() %>%
      filter(Simulation == sim1, Variable == var, Sector == sec,
Qualifier == qual, Year == yr) %>% select(Value)

    if(formula == "Difference Between Simulations"){
      val_sim2 <- results %>%
        filter(Simulation == sim2, Variable == var, Sector == sec,
Qualifier == qual, Year == yr) %>% select(Value)
      return(round(val_sim1-val_sim2, 2))
    }
    if(formula == "Percentage"){
      return(round(val_sim1*100, 3))
    }
 }
```

Exporting to the Web

In this section, we have used this link as a reference. We will follow the steps in this link with some tweaks to host our dashboard on the AWS cloud using services like EC2 and Route 53.

EC2 is a virtual computing environment that allows you to run applications in the AWS cloud. You can launch instances of these computing environments with different operating systems, programs and packages quickly and easily.

As for Route 53, it is a highly available and scalable cloud Domain Name System (DNS) web service. It allows developers and businesses to effectively route end users to Internet applications by translating names like www.example.com into the numeric IP addresses like 192.0.

In the following, we would describe how to connect to the EC2 instance, install R in the instance, and host the Shiny app on the web.

A. Create an EC2 instance

Log in to AWS account and follow Step 1:Create an EC2 instance from this link.

The steps below require us to have access to a Unix environment.

- For **Window** users, you can skip step B and instead follow "Step 2: Access the EC2 instance via SSH from Putty (Windows based)" from this link.
- While for **Mac** users, Mac already runs a variant of UNIX. To get into the Unix environment, launch the Terminal application (that's Finder → Applications → Utilities → Terminal). Then follow all the steps below.

B. Connect the EC2 instance to our terminal

To connect to your instance, launch your Unix terminal and use the following commands to connect. You will need the **.pem file** you downloaded in the step above.

The first command makes your public key not publicly viewable (only needs to be run the first time you use your SSH key).

The second connects to your EC2 instance. Replace the name and path highlighted in yellow with your key's name and location, replace publicDNS with your public DNS. You may be prompted to proceed.

```
chmod 400 ~/.ssh/mykey.pem
ssh -i ~/.ssh/mykey.pem ubunut@publicDNS
```

If you want to disconnect from your EC2 instance, type exit into your terminal and you'll be logged out of your EC2 instance.

C. Install R in the EC2 instance

The first step is to go to the root. Then we install R. Type these following command:

```
sudo -i
sudo apt-get install r-base
```

D. Install the required function to set up the environment that would make us install the R packages successfully.

When we are installing this, make sure we are in our root directory.

```
sudo apt-get install libssl-dev
sudo apt-get install libxml2-dev
sudo apt-get install libcurl4-openssl-dev
sudo apt-get install gdebi-core
```

E. Install RStudio into EC2 instance

When we are installing RStudio, make sure we are in our root directory.

```
wget
https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-1.5.16.958-a
md64.deb
```

F. Install Shiny Server

When we are installing Shiny Server, make sure we are in our root directory.

```
sudo gdebi shiny-server-1.5.16.958-amd64.deb
```

G. Transfer the R shiny app components via Github

After installing everything above, a directory called 'shiny-server' would have been created in the path /srv/shiny-server/. Check with this command: ls /srv

The next step is to create a folder inside the directory shiny-server. This is the folder where we place our R shiny app components (ui.R, server. R, data files or R programs).

At first we may not be able to create a folder inside the shiny-server folder, to do this execute the below commands first.

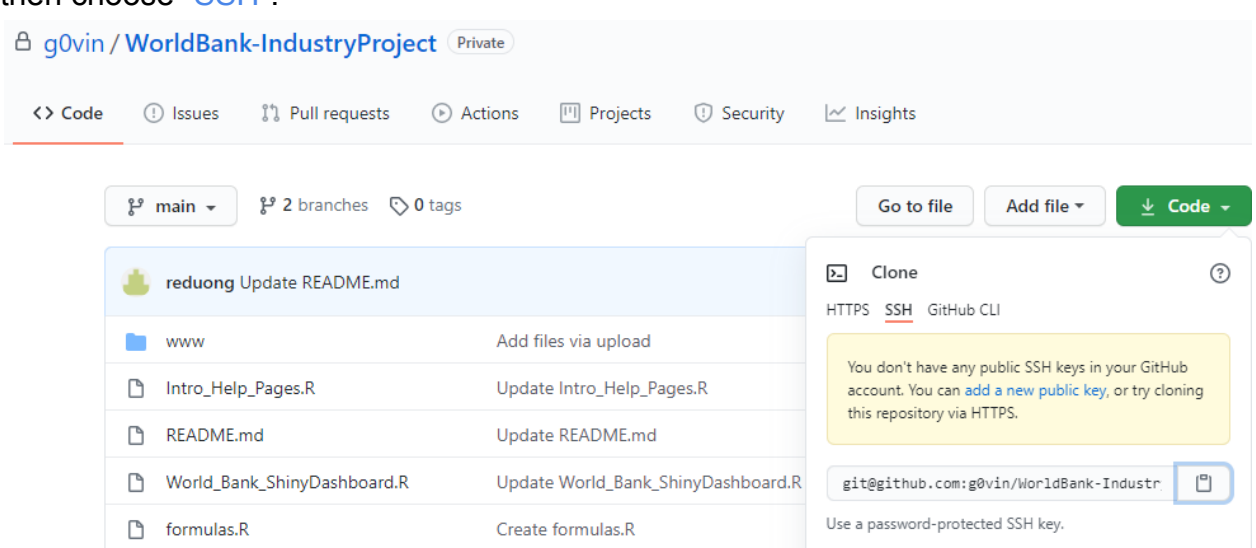
```
sudo chmod 777 /srv/shiny-server
sudo mkdir /srv/shiny-server/worldbank
```

In the above command I have created a folder 'worldbank' to place all the R shiny app components.

Since all the R shiny app components are saved on GitHub. We first need to install git and then pull the components in this following way:

```
sudo apt-get update
sudo apt-get install git
cd /srv/shiny-server/worldbank
git clone git@github.com:g0vin/WorldBank-IndustryProject.git
```

The line highlighted in yellow is the SSH key for cloning the WorldBank-IndustryProject repository, which can be obtained from Github. Click on the green button, "**Code**", and then choose "SSH".



H. Install the packages in R Server

Once we are connected to the EC2 instance and have installed everything above, type "R" in the command line to start running R. When R is running, install these following packages with these commands below.

```
install.packages("shiny")
install.packages("shinyWidgets")
install.packages("shinydashboard")
install.packages("shinyalert")
install.packages("shinyBS")
install.packages("shinyjs")
install.packages("rintrojs")
install.packages("readr")
install.packages("readxl")
install.packages("plyr")
install.packages("tidyverse")
install.packages("DT")
install.packages("rlang")
install.packages("plotly")
install.packages("ggplot2")
```

For steps F to H, we have to do it while R is still running.

I. Set the TCP port and IPV4 address

This is the TCP port and the IPV4 address the Shiny application should listen to. To allow other clients to connect, we need to use the value "0.0.0.0" for the host.

```
options(shiny.port = 3838, shiny.host = "0.0.0.0")
```

J. Load the Shiny package

```
library(shiny)
```

K. Run the Shiny application

```
runApp("main.R")
```

L. Hosting the app

In the amazon console, go to your running EC2 instance. Copy the Public DNS (IPV4) e.g. : ec2-54-214-93-126.us-west-2.compute.amazonaws.com.
Copy this in the browser and suffix: 3838 and press enter.

You will get an http address similar to this:
http://ec2-54-214-93-126.us-west-2.compute.amazonaws.com:3838/

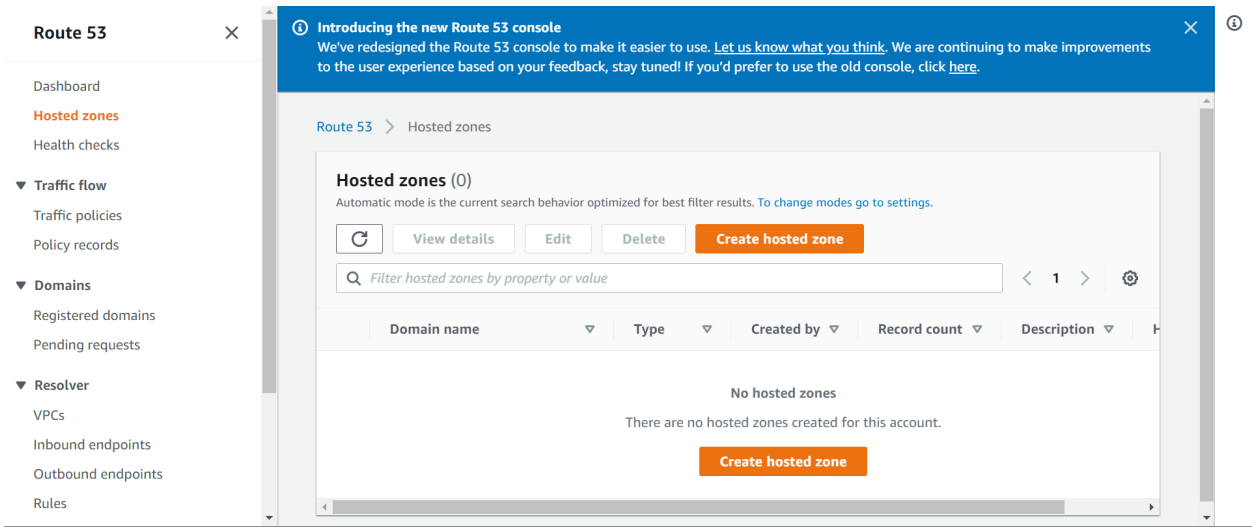
M. Creating an alternate name for the http address

In step L, we manage to host the Shiny application on the web. This step is **optional**.

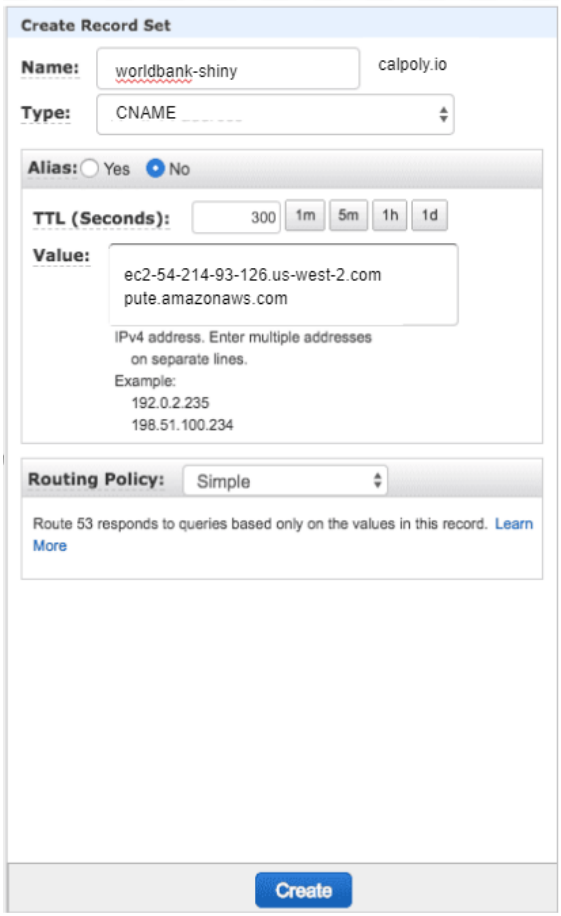
If you want to use a personalized URL, instead of the one we were assigned to in step L, we will need Route 53 to create an alias for the http address (e.g. http://ec2-54-214-93-126.us-west-2.compute.amazonaws.com:3838/).

First, sign in to the AWS Management Console and click on EC2 under the 'Compute' header or click on EC2 under 'Recently visited services'. Go to your running EC2 instance and copy the Public DNS (IPV4) (e.g. ec2-54-214-93-126.us-west-2.compute.amazonaws.com.

After we copied the Public DNS, open the AWS Route 53 console. In the **navigation pane**, choose Hosted zones. If you already have a hosted zone for your domain, skip this step. Otherwise, click on the "Create Hosted Zone" button and perform the applicable procedure to create a hosted zone.



On the Hosted zones page, choose the name of the hosted zone that you want to create records in. For example, we have created a hosted zone named “calpoly.io”. Click on this hosted zone and then click on “Create Record Set”. Type in a **Name**. For instance, “worldbank-shiny”. As for **Type**, choose “CNAME”. Then paste the Public DNS we copied into the **Value** box. When we are done, click on the “CREATE” button at the bottom.



Now we will get a personalized http address similar to this: http://worldbank-shiny.calpoly.io:3838/, which is hosted under the calpoly.io zone. Don’t forget to add in the suffix, 3838, after calpoly.io.

Appendix

Notice how there's a file called "World_Bank_ShinyDashboard.R". That is a compilation of all the separate files we've seen so far -- main.R, ui.R, server,R, graphs.R, formulas.R, and Intro_Help_Pages.R . Basically, instead of saving the codes into separate files, the codes are all saved under one file.

The purpose of separating up the code in "World_Bank_ShinyDashboard.R" was for the ease of access to make changes. Small files are easier to stomach. If every file serves only one topic, we will know quickly where to look and immediately know what does not belong to the topic, without having to read through the commentary.

In short, the "World_Bank_ShinyDashboard.R" file is here for your record. In case you want to revert the changes you made on the separate files, you can reference this file for the codes.