Module 9

Developing Data Products

Session II

Course Background

- This course covers the basics of creating data products using Shiny, R packages, and interactive graphics.
- The course will focus on the statistical fundamentals of creating a data product that can be used to tell a story about data to a mass audience.
- Students will learn a variety of core tools for creating data products in R and R Studio in specific.

Learning Objectives

Towards the end of this lesson, you should be able to:

- Plot interactive charts with plotly and GoogleVis
- Embed plots into R Presentation
- Create R Package

Assessment

- Quiz 1: 20% 11 January 2016 morning
- Quiz 2: 20% 18 January 2016 morning
- Quiz 3: 20% 25 January 2016 morning
- Assignment and Peer Assessment : 40% 25 January 2016 morning

- 70% is required for passing the class and 90% for distinction.
- Quiz scoring is the best score of your 3 attempts
- Hard deadlines of 5 day after Quiz is due, with each day incurring a 10% penalty (10% waived if you use a Late Day)

Assignment

Shiny App

- Write a shiny application with associated supporting documentation. The documentation should be thought of as whatever a user will need to get started using your application.
- Deploy the application on Rstudio's shiny server
- Share the application link by pasting it into the text box below
- Share your server.R and ui.R code on github

- The application must include the following:
  Some form of input (widget: textbox, radio button, checkbox, ...)
  - Some operation on the ui input in sever.R
  - Some reactive output displayed as a result of server calculations
  - You must also include enough documentation so that a novice user could use your application.
  - The documentation should be at the Shiny website itself. Do not post to an external link.

Assignment

Reproducible Pitch Presentation

- 5 slides to pitch our idea done in Slidify or Rstudio Presenter
- Your presentation pushed to github or Rpubs
- A link to your github or Rpubs presentation pasted into the text box below

Your presentation must satisfy the following

- It must be done in Slidify or Rstudio Presenter
- It must be 5 pages
- It must be hosted on github or Rpubs
- It must contained some embedded R code that gets run when slidifying the document

Exercise 1: Reactive Shiny app

- Download the Banting Air Pollutant Index file from here

(the original unprocessed data can be obtained from http://data.gov.my/view.php?view=280)

- Note that the Air Pollutant Index file capture data by the hour from 01-08-2013 till 05-02-2015
- Develop a shiny app to plot API reading by Year and Month
- Sample Output: https://kuanhoong.shinyapps.io/Banting_API/

R Packages

- R packages are a comfortable way to maintain collections of R functions and data sets
- Most users first see the packages of functions distributed with R or from CRAN
- Package system allows many more people to contribute to R while still enforcing some standards
- Packages are also a convenient way to maintain private functions and share them with your colleagues

Create R packages

- There are two popular ways of starting a new package:
  1. Load all functions and data sets you want in the package into a clean `R` session, and run `package.skeleton()`. The objects are sorted into data and functions, skeleton help files are created for them using `prompt()` and a `DESCRIPTION` file is created. The function then prints out a list of things for you to do next.
  2. Create it manually, which is usually faster for experienced developers

Structure of a package

- The extracted sources of an R package are simply a directory somewhere on your hard drive
- A file named DESCRIPTION with descriptions of the package, author, and license conditions
- in a structured text format that is readable by computers and by people.
    - A man/ subdirectory of documentation files.
    - An R/ subdirectory of R code.
    - A data/ subdirectory of datasets.
- Less commonly it contains
    - A src/ subdirectory of C, Fortran or C++ source.
    - tests/ for validation tests.
    - exec/ for other executables (eg Perl or Java).
    - inst/ for miscellaneous other stuff. The contents of this directory are completely copied to
- the installed version of a package.
    - A configure script to check for other required software or handle differences between systems

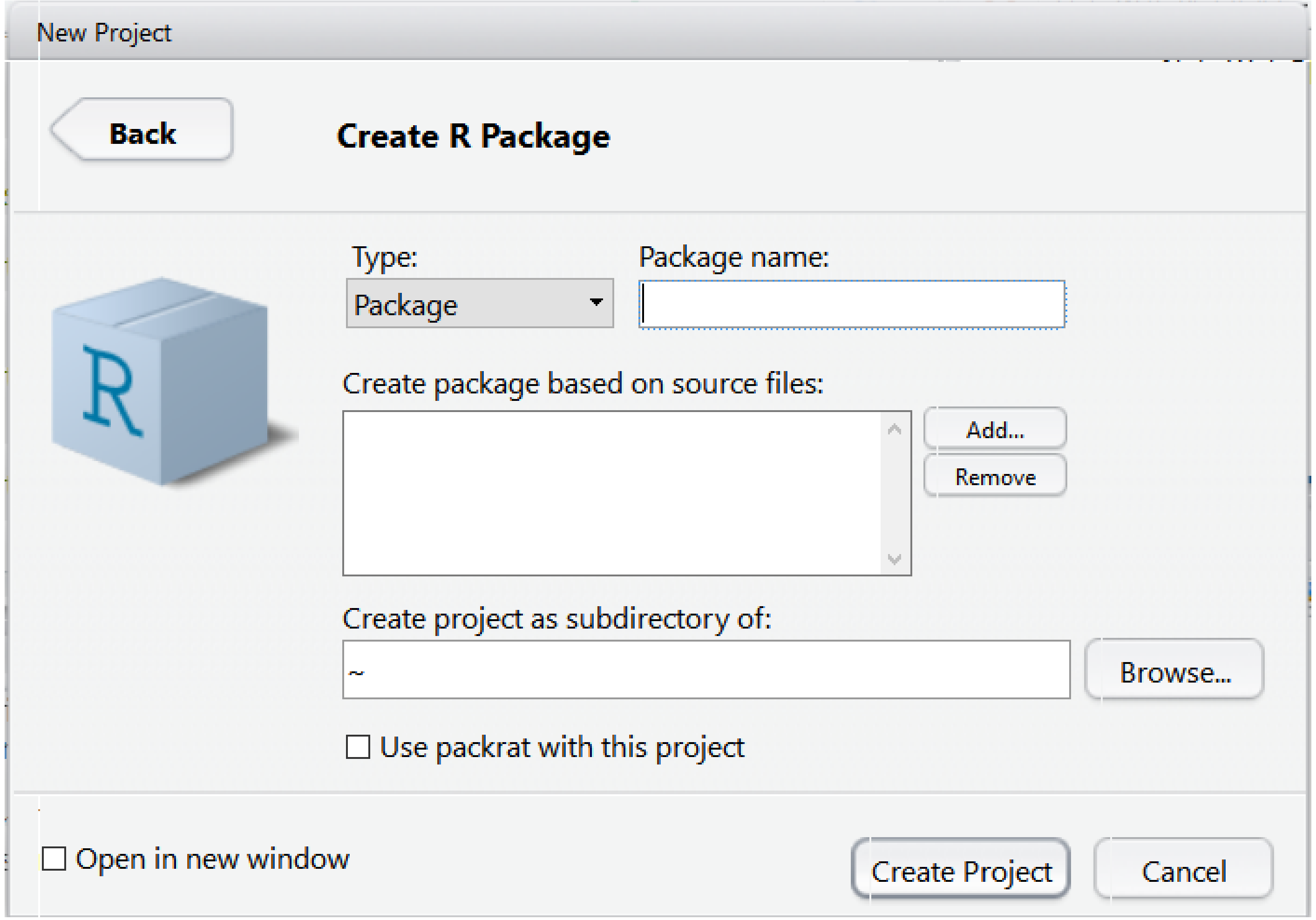
* All but the DESCRIPTION file are optional, though any useful package will have man/ and at least one of R/ and data/

Starting a package

- To start a package for our R code all we have to do is run function `package.skeleton()` and pass it the name of the package we want to create plus a list of all source code files.

Create a new package with RStudio

- To create a new package use the Create Project command (available on the Projects menu and on the global toolbar) and select the New Directory option. Then on the following screen specify the project type as R Package:



New Project

Create R Package

Type: Package

Package name:

Create package based on source files:

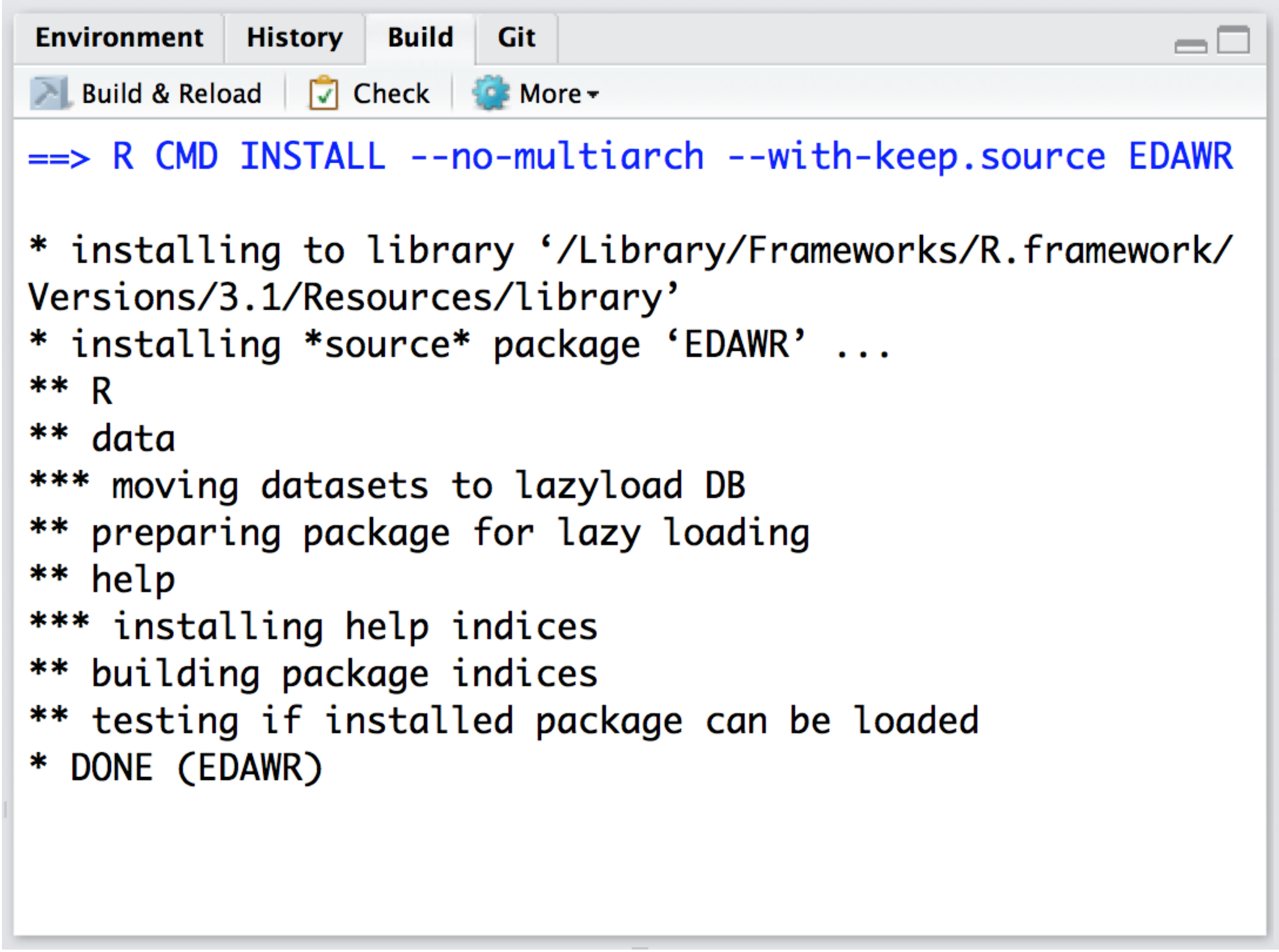
Create project as subdirectory of: ~

□ Use packrat with this project

□ Open in new window

Building a new package with RStudio

- To work with packages in RStudio you use the Build pane, which includes a variety of tools for building and testing packages. While iteratively developing a package in RStudio, you typically use the Build and Reload command to re-build the package and reload it in a fresh R session:



```
==> R CMD INSTALL --no-multiarch --with-keep.source EDAWR

* installing to library ‘/Library/Frameworks/R.framework/
Versions/3.1/Resources/library’
* installing *source* package ‘EDAWR’ ...
** R
** data
*** moving datasets to lazyload DB
** preparing package for lazy loading
** help
*** installing help indices
** building package indices
** testing if installed package can be loaded
* DONE (EDAWR)
```

Building a new package

- The Build and Reload command performs several steps in sequence to ensure a clean and correct result:
  - Unloads any existing version of the package (including shared libraries if necessary).
  - Builds and installs the package using R CMD INSTALL.
  - Restarts the underlying R session to ensure a clean environment for re-loading the package.
  - Reloads the package in the new R session by executing the library function.

Plotly

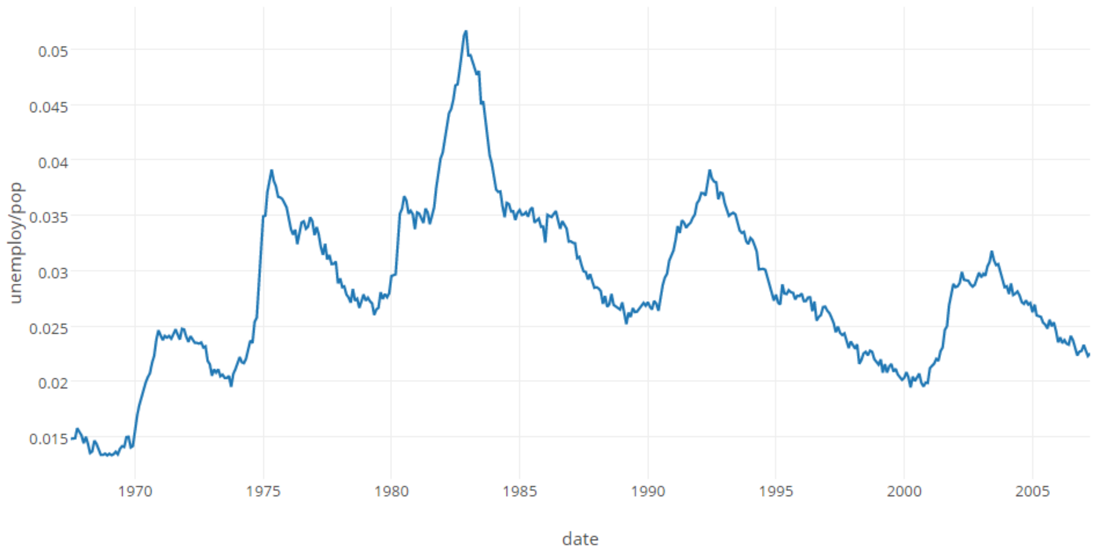
- Plotly for R is an interactive, browser-based charting library built on the open source JavaScript graphing library, plotly.js.
- It works entirely locally, through the HTML widgets framework.
- Plotly graphs are interactive.
- Plotly objects are data frames with a class of plotly and an environment that tracks the mapping from data to visual properties.
- By default, plotly for R runs locally in your web browser or R Studio's viewer. You can publish your graphs to the web by creating a plotly account.

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

Plotly

- To create a plotly visualization, start with `plot_ly()`. Your graph will appear in your web browser or in RStudio's viewer.

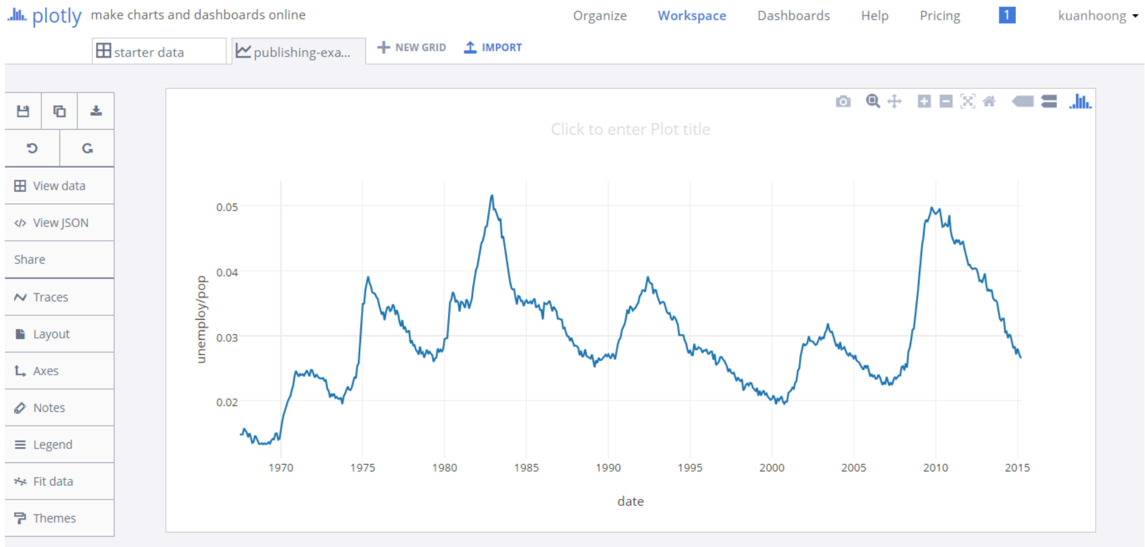
```
library(plotly)
p <- plot_ly(economics,
x = date, y = unemploy / pop)
p
```



Plotly

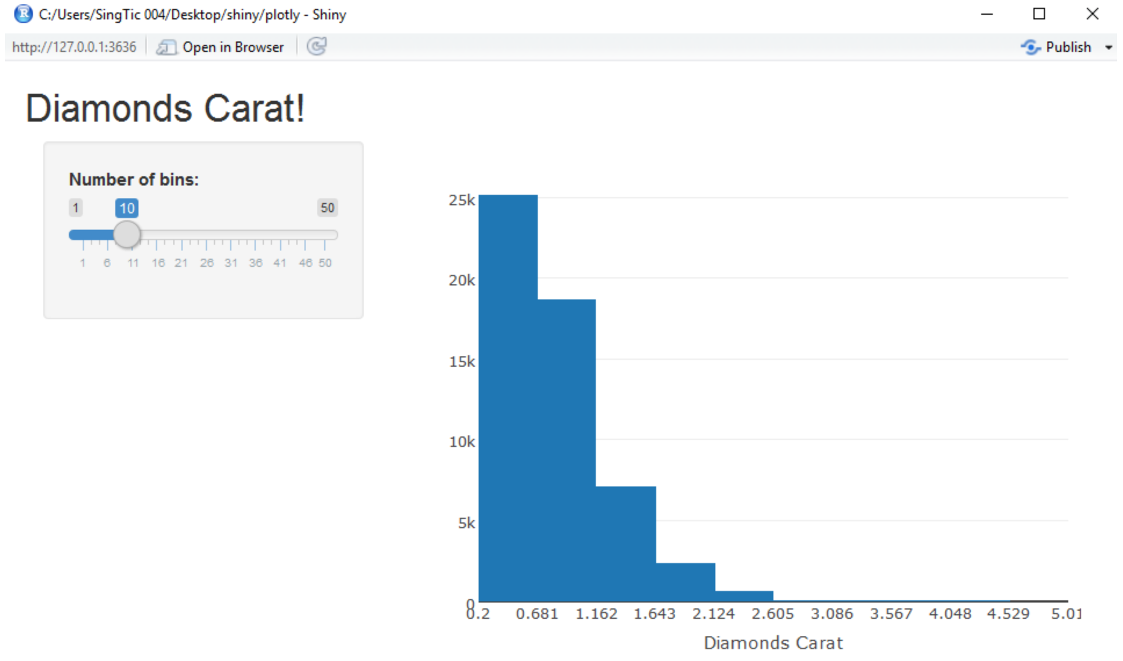
- If you want to publish your graphs to the web, you can host your graphs in an online plotly account with `plotly_POST`

```
library(plotly) p <- plot_ly(economics, x =
date, y = unemploy / pop) plotly_POST(p,
filename="r-user-guide/publishing-example")
```



Exercise 2: Plotly Graphs in Shiny

- Plotly works with Shiny entirely *client-side* either through plotly.js or with Plotly's postMessage API.
- This means that updating the Plotly graphs is *fast* because no external calls are made to a Plotly server.
- You can export Plotly graphs to your account by clicking the Edit chart link on the bottom right of the chart.



C:/Users/SingTic 004/Desktop/shiny/plotly - Shiny

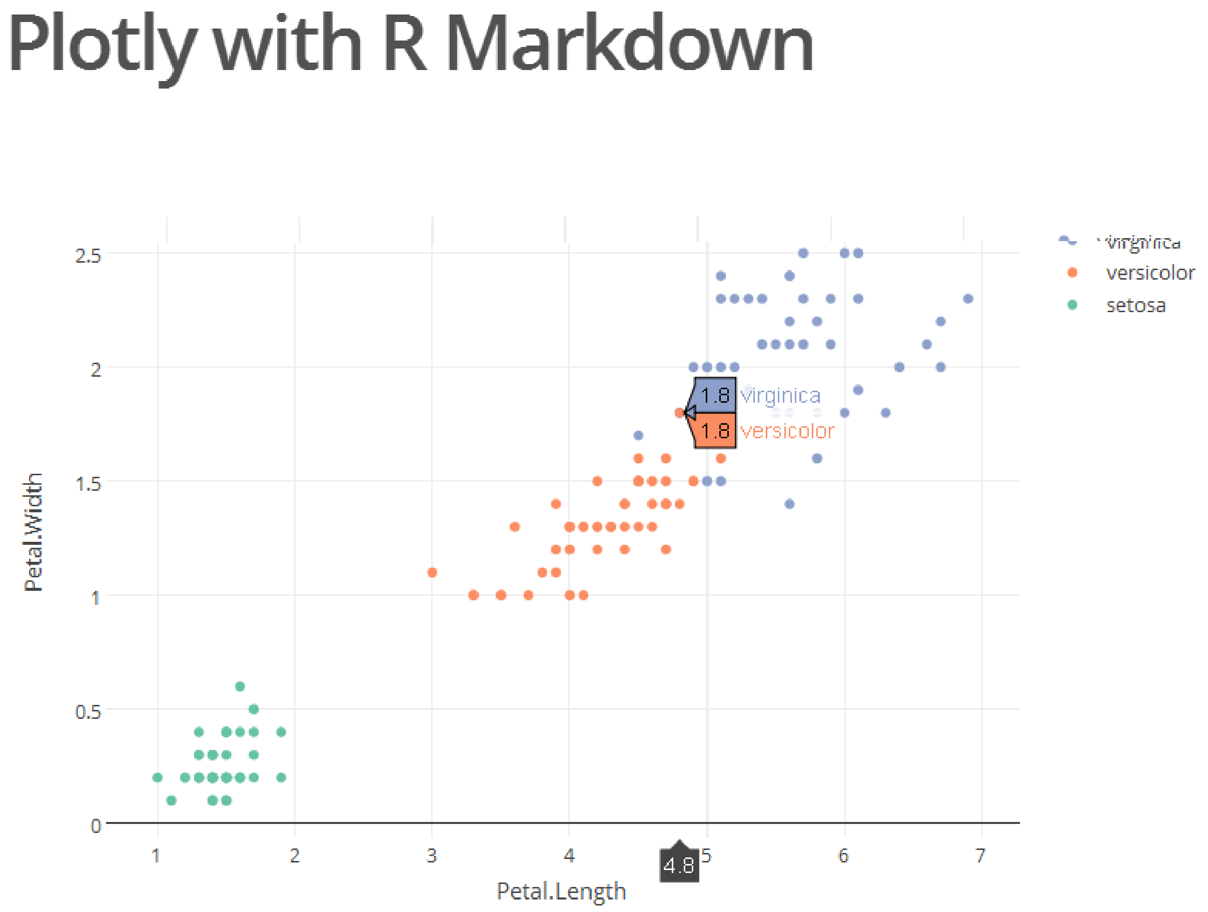
http://127.0.0.1:3636

Diamonds Carat!

Number of bins: 10

Exercise 3: Embed Plotly in R Markdown Presentation

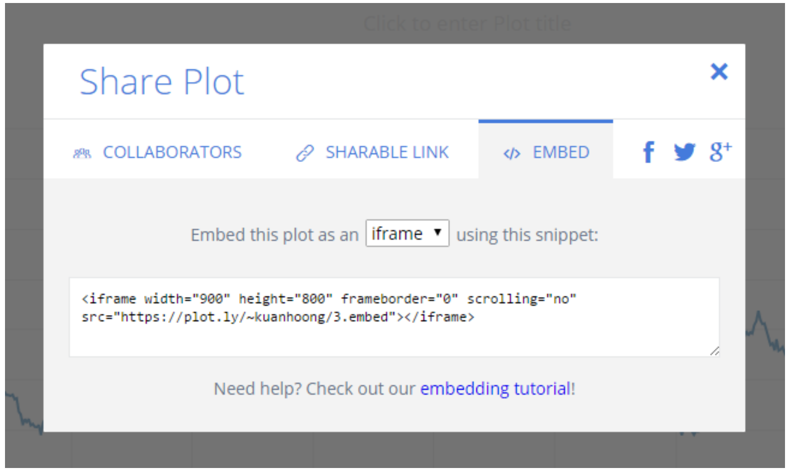
~~~
## Plotly with R Markdown

```{r, plotly=TRUE, echo=FALSE, message=FALSE}
library(ggplot2)
library(plotly)
p <- plot_ly(iris, x = Petal.Length, y = Petal.Width,
color = Species, mode = "markers")
p
```
~~~



Plotly with R Markdown

Exercise 4: Embed plot from Plotly website

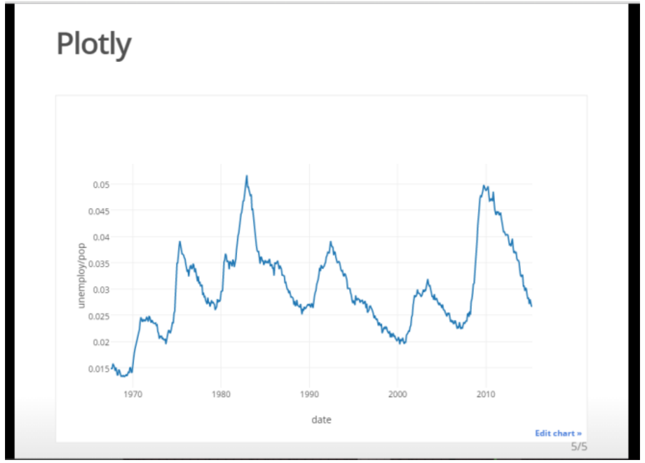
- Go to your plotly account and access your online plot files
- Select the plot that you want to embed and click the share
- Copy the iframe or html code
- **result may not show in preview, click open in browser*



Embed this plot as an iframe ▾ using this snippet:

```
<iframe width="900" height="800" frameborder="0" scrolling="no"
src="https://plot.ly/~kuanhoong/3.embed"></iframe>
```

Need help? Check out our embedding tutorial!



~~~
## Plotly

```{r results='asis', message=FALSE, echo=FALSE}

cat('<iframe width="900" height="800"
frameborder="0" scrolling="no"
src="https://plot.ly/~kuanhoong/3.embed"></ifr
ame>')

```
~~~

GoogleVis

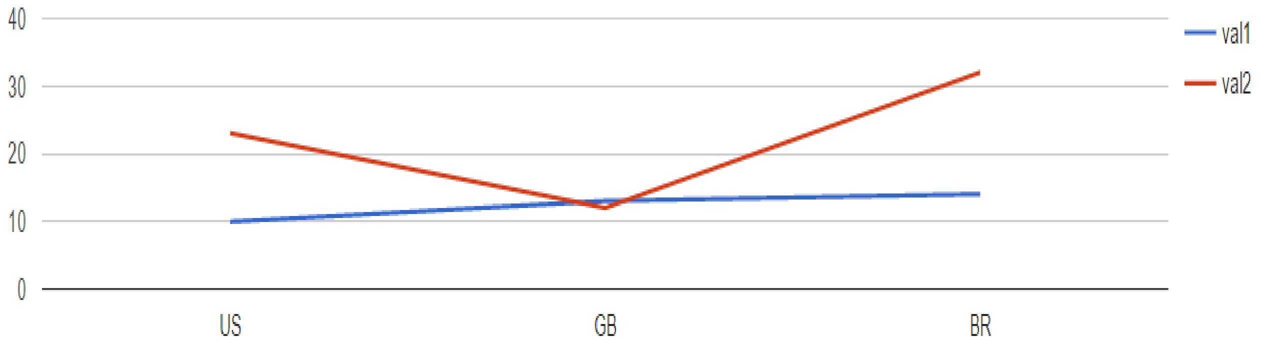
- R Interface to Google Charts
- Allows users to create interactive charts based on data frames.
- Charts are displayed locally via the R HTTP help server
- All plot commands begin with “gvis”
- For more info, refer to Google Charts

https://developers.google.com/chart/interactive/docs/reference

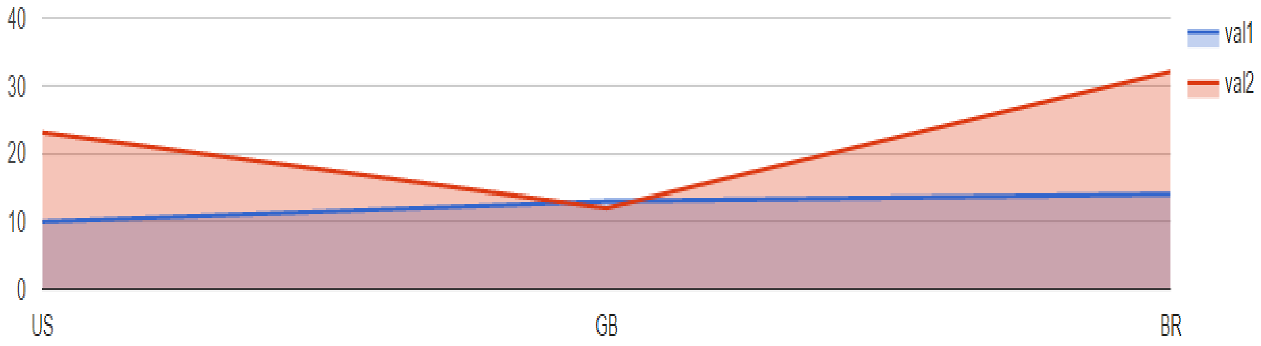
```
install.packages("googleVis")
library(googleVis)
demo(googleVis)
```

GoogleVis

```
# Create a dataframe df
df <- data.frame (
country=
c("US", "GB", "BR"),
val1=c(10,13,14),
val2=c(23,12,32))

# Line Chart
Line <- gvisLineChart(df)
plot(Line)

# Area Chart
Area <- gvisAreaChart(df)
plot(Area)
```



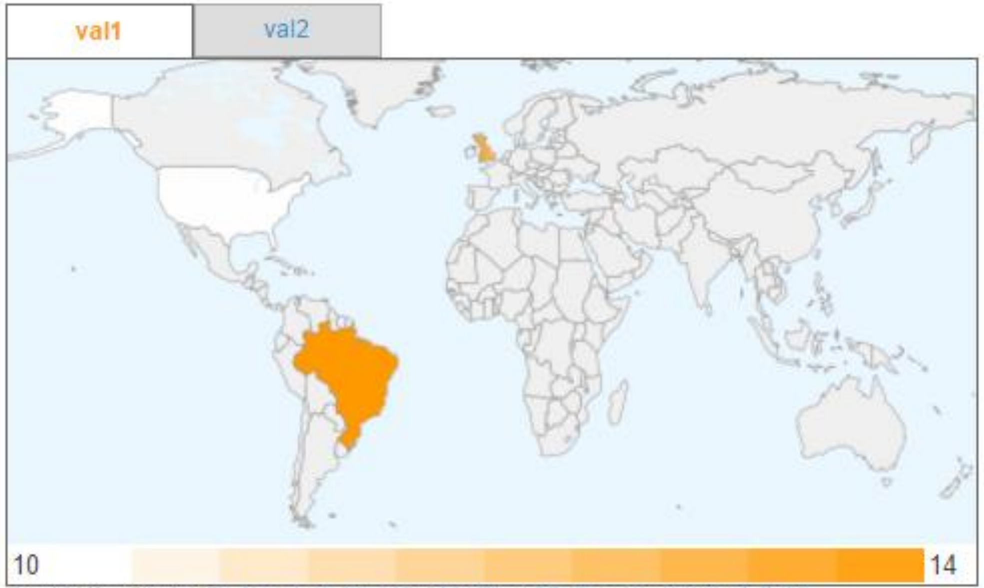
Data: data • Chart ID: LineChartID116c21fb52a4 • googleVis-0.5.10
R version 3.2.1 (2015-06-18) • Google Terms of Use • Documentation and Data Policy



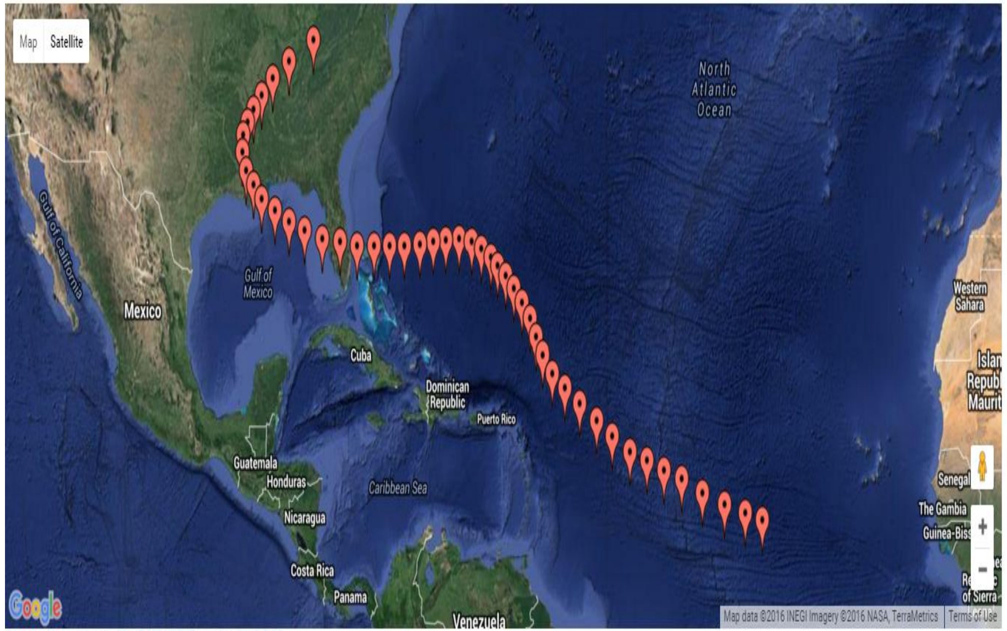
Data: data • Chart ID: AreaChartID116c17bf63c1 • googleVis-0.5.10
R version 3.2.1 (2015-06-18) • Google Terms of Use • Documentation and Data Policy

GoogleVis

```
# Intensity Map
Intensity <- gvisIntensityMap(df)
plot(Intensity)

## Plot Hurricane Andrew (1992) storm path

data(Andrew)
M1 <- gvisMap(Andrew, "LatLong", "Tip",
options=list(
showTip=TRUE,
showLine=TRUE,
enableScrollWheel=TRUE,
mapType='hybrid',
useMapTypeControl=TRUE,
width=800,height=400))

plot(M1)
```



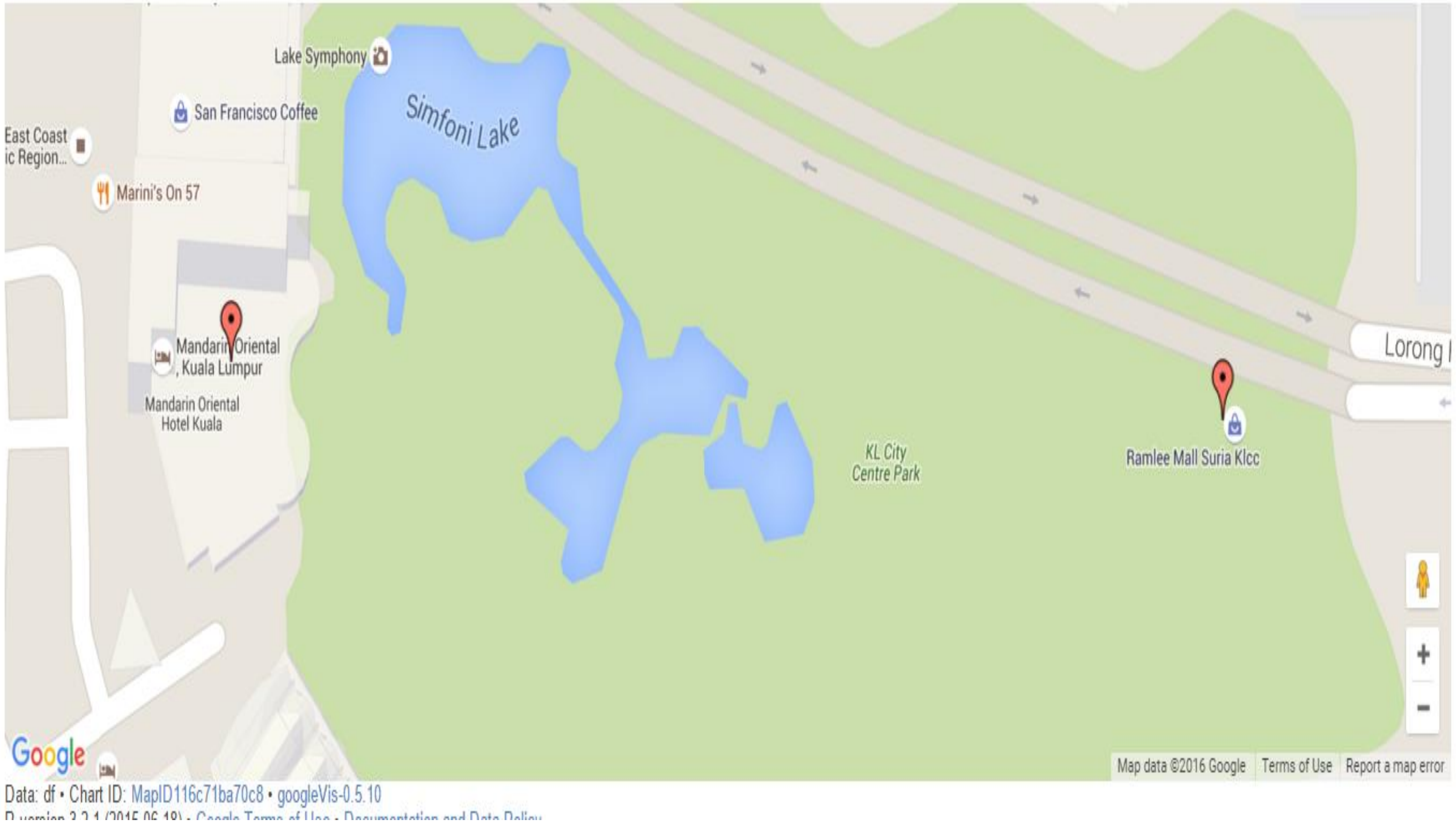
Data: df • Chart ID: IntensityMapID116c7bee35c3 • googleVis-0.5.10
R version 3.2.1 (2015-06-18) • Google Terms of Use • Documentation and Data Policy



Data: Andrew • Chart ID: MapID116c62f067c • googleVis-0.5.10
R version 3.2.1 (2015-06-18) • Google Terms of Use • Documentation and Data Policy

GoogleVis

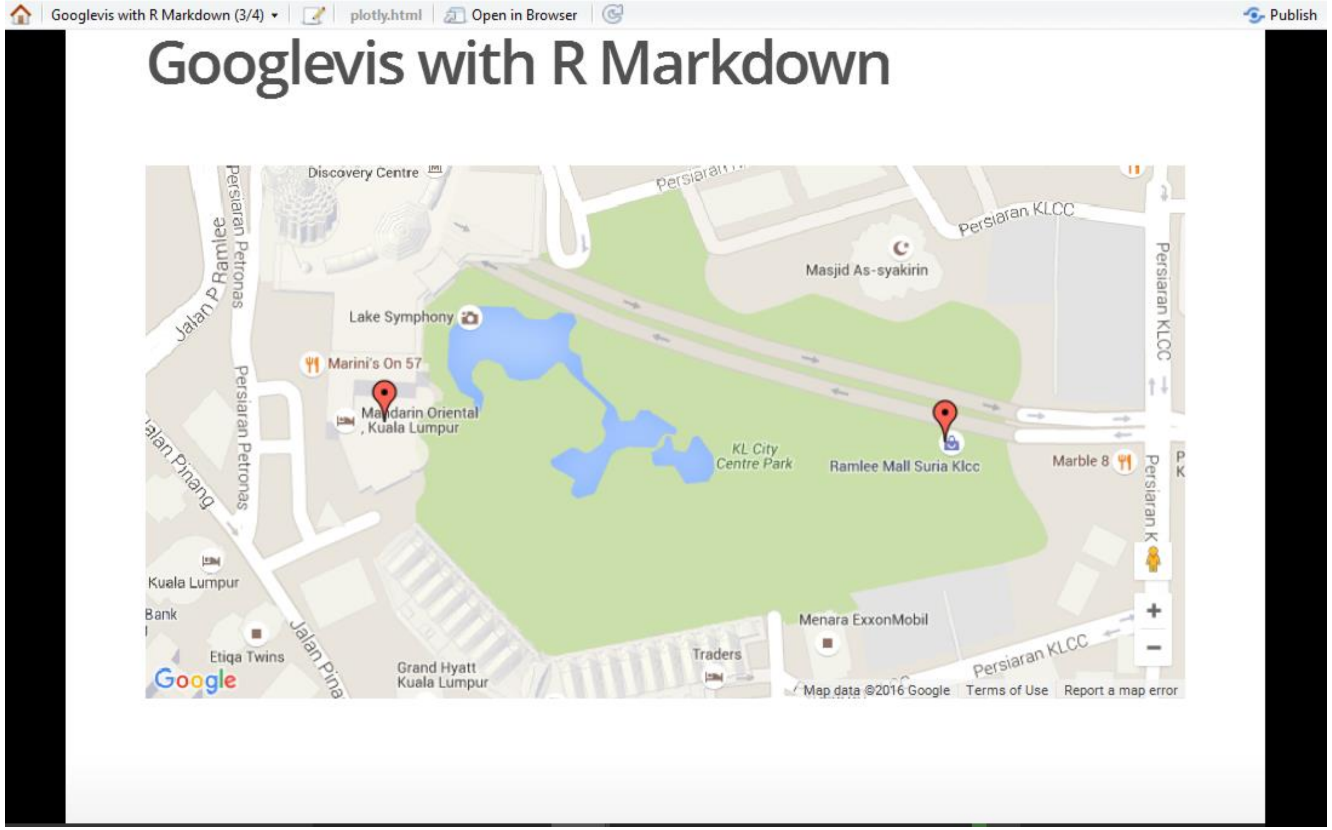
```
# Use GPS latitude:Longtitude
df <- data.frame(
loc=c("3.155762:101.716529",
"3.155923:101.712023"),
tip=c("Suria KLCC",
"Mandarin Oriental"))

M2<-gvisMap(df, "loc", "tip",
options=list(
showTip=TRUE, mapType="normal",
enableScrollWheel=TRUE))
plot(M2)
```



Data: df • Chart ID: MapID116c71ba70c8 • googleVis-0.5.10
R version 3.2.1 (2015-06-18) • Google Terms of Use • Documentation and Data Policy

Exercise 5: Googlevis and R Markdown

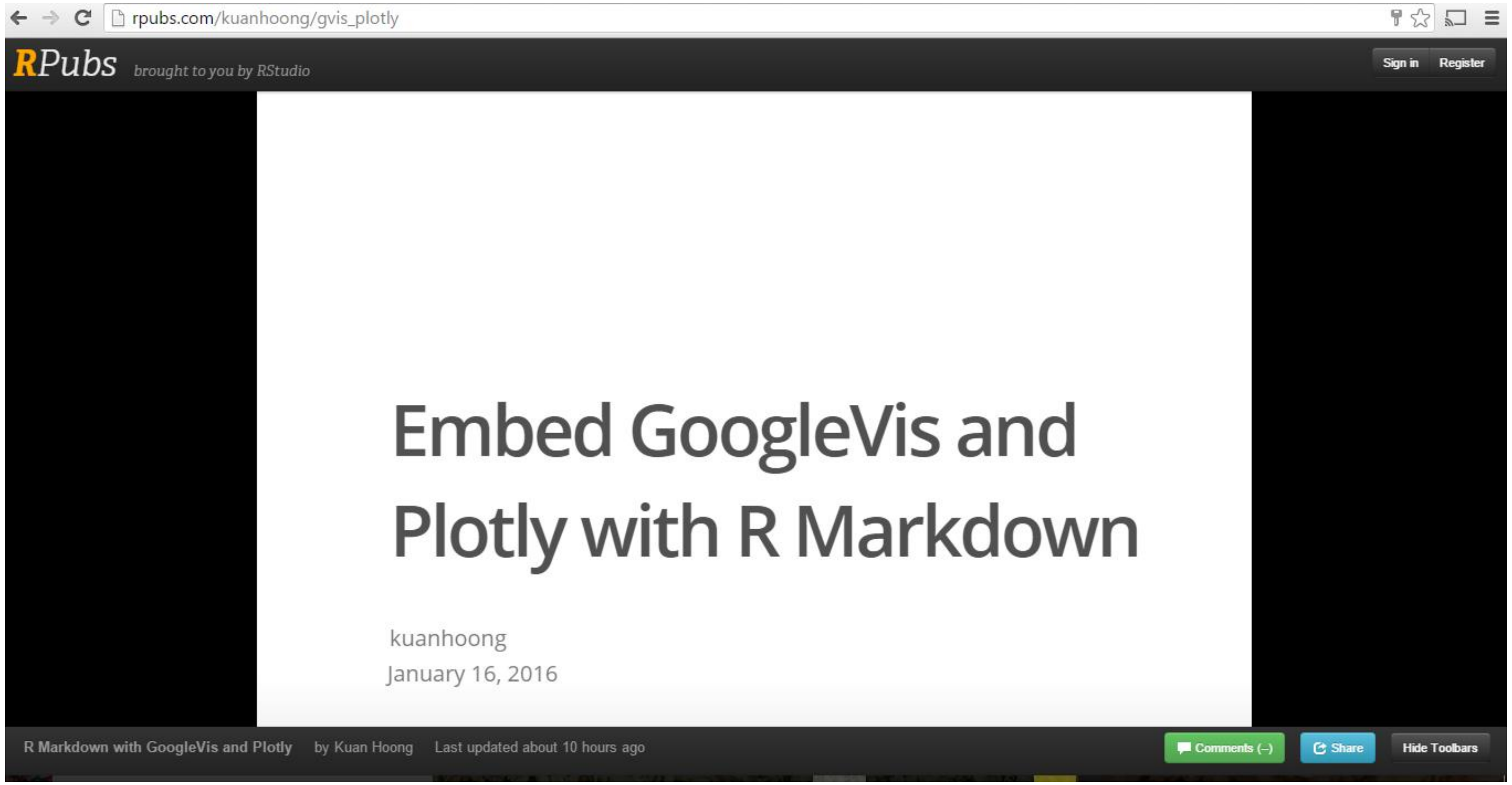
~~~
## Googlevis with R Markdown

```{r, results='asis', message=FALSE, echo=FALSE}
library(googleVis)
df <- data.frame(
loc=c("3.155762:101.716529",
"3.155923:101.712023"),
tip=c("Suria KLCC",
"Mandarin Oriental"))

M2<-gvisMap(df, "loc", "tip",
options=list(
showTip=TRUE, mapType="normal",
enableScrollWheel=TRUE))
print(M2, "chart")
```
~~~



Googlevis with R Markdown

Lastly…

- Sample R Markdown Output from RPubs: http://rpubs.com/kuanhoong/gvis_plotly



Embed GoogleVis and Plotly with R Markdown

kuanhoong
January 16, 2016

R Markdown with GoogleVis and Plotly by Kuan Hoong Last updated about 10 hours ago