

# R Shiny

*Making Graphics Dynamic*

# R SHINY

WEB APPS FOR THE COMMON PERSON

RENDERING:

CONVERT R TO DYNAMIC HTML

# HOW DOES “KNIT” WORK IN RMD?

Shiny functions work similar to other knitr functions that are used to convert your raw R output into HTML objects that make for nice documents.

## Raw R output

```
##
## -----
##           Dependent variable:
##           -----
##           heart.rate
##           (1)      (2)      (3)      (4)
## -----
## caffeine  0.087***  0.080***  0.009    0.037
##           (0.021)  (0.008)  (0.121) (0.047)
##
## gym.time   -1.441***      -1.440***
##           (0.062)          (0.062)
##
## stress.index      0.414    0.228
##           (0.631)    (0.246)
##
## Constant  68.953*** 116.461*** 68.267*** 116.022***
##           (5.454)  (2.942)  (5.568)  (2.982)
## -----
## Observations  100    100    100    100
## R2            0.153  0.872  0.157  0.873
## -----
## Note:                *p<0.1; **p<0.05; ***p<0.01
```



## After conversion to HTML table

	<i>Dependent variable:</i>			
	heart.rate			
	(1)	(2)	(3)	(4)
caffeine	0.087*** (0.021)	0.080*** (0.008)	0.009 (0.121)	0.037 (0.047)
gym.time		-1.441*** (0.062)		-1.440*** (0.062)
stress.index			0.414 (0.631)	0.228 (0.246)
Constant	68.953*** (5.454)	116.461*** (2.942)	68.267*** (5.568)	116.022*** (2.982)
Observations	100	100	100	100
R <sup>2</sup>	0.153	0.872	0.157	0.873

*Note:*  $p < 0.1$ ;  $p < 0.05$ ;  $p < 0.01$

## SIDE NOTE: THIS IS WHAT THE RAW HTML TABLE LOOKS LIKE

```
<table style="text-align:center"><tr><td colspan="5" style="border-bottom: 1px solid
black"></td></tr><tr><td style="text-align:left"></td><td colspan="4"><em>Dependent
variable:</em></td></tr> <tr><td></td><td colspan="4" style="border-bottom: 1px solid black"></td></tr>
<tr><td style="text-align:left"></td><td colspan="4">heart.rate</td></tr> <tr><td style="text-
align:left"></td><td>(1)</td><td>(2)</td><td>(3)</td><td>(4)</td></tr> <tr><td colspan="5"
style="border-bottom: 1px solid black"></td></tr><tr><td style="text-
align:left">caffeine</td><td>0.087<sup>***</sup></td><td>0.080<sup>***</sup></td><td>0.009</td><td>0.037
</td></tr> <tr><td style="text-
align:left"></td><td>(0.021)</td><td>(0.008)</td><td>(0.121)</td><td>(0.047)</td></tr> <tr><td
style="text-align:left"></td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-
align:left">gym.time</td><td></td><td>-1.441<sup>***</sup></td><td></td><td>-
1.440<sup>***</sup></td></tr> <tr><td style="text-
align:left"></td><td></td><td>(0.062)</td><td></td><td>(0.062)</td></tr> <tr><td style="text-
align:left"></td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-
align:left">stress.index</td><td></td><td></td><td>0.414</td><td>0.228</td></tr> <tr><td style="text-
align:left"></td><td></td><td></td><td>(0.631)</td><td>(0.246)</td></tr> <tr><td style="text-
align:left"></td><td></td><td></td><td></td><td></td></tr> <tr><td style="text-
align:left">Constant</td><td>68.953<sup>***</sup></td><td>116.461<sup>***</sup></td><td>68.267<sup>***</
sup></td><td>116.022<sup>***</sup></td></tr> <tr><td style="text-
align:left"></td><td>(5.454)</td><td>(2.942)</td><td>(5.568)</td><td>(2.982)</td></tr> <tr><td
style="text-align:left"></td><td></td><td></td><td></td><td></td></tr> <tr><td colspan="5"
style="border-bottom: 1px solid black"></td></tr><tr><td style="text-
align:left">Observations</td><td>100</td><td>100</td><td>100</td><td>100</td></tr> <tr><td style="text-
align:left">R<sup>2</sup></td><td>0.153</td><td>0.872</td><td>0.157</td><td>0.873</td></tr> <tr><td
colspan="5" style="border-bottom: 1px solid black"></td></tr><tr><td style="text-
align:left"><em>Note:</em></td><td colspan="4" style="text-align:right"><sup>*</sup>p<0.1;
<sup>**</sup>p<0.05; <sup>***</sup>p<0.01</td></tr> </table>
```

# RENDER FUNCTIONS:

## Raw R Version

```
plot( x, y, main="My Plot" )
```

## R Shiny Version

```
renderPlot({  
  plot( x, y, main="My Plot" )  
})
```



Converts this to a shiny object  
that can be updated and  
re-plotted inside a browser.

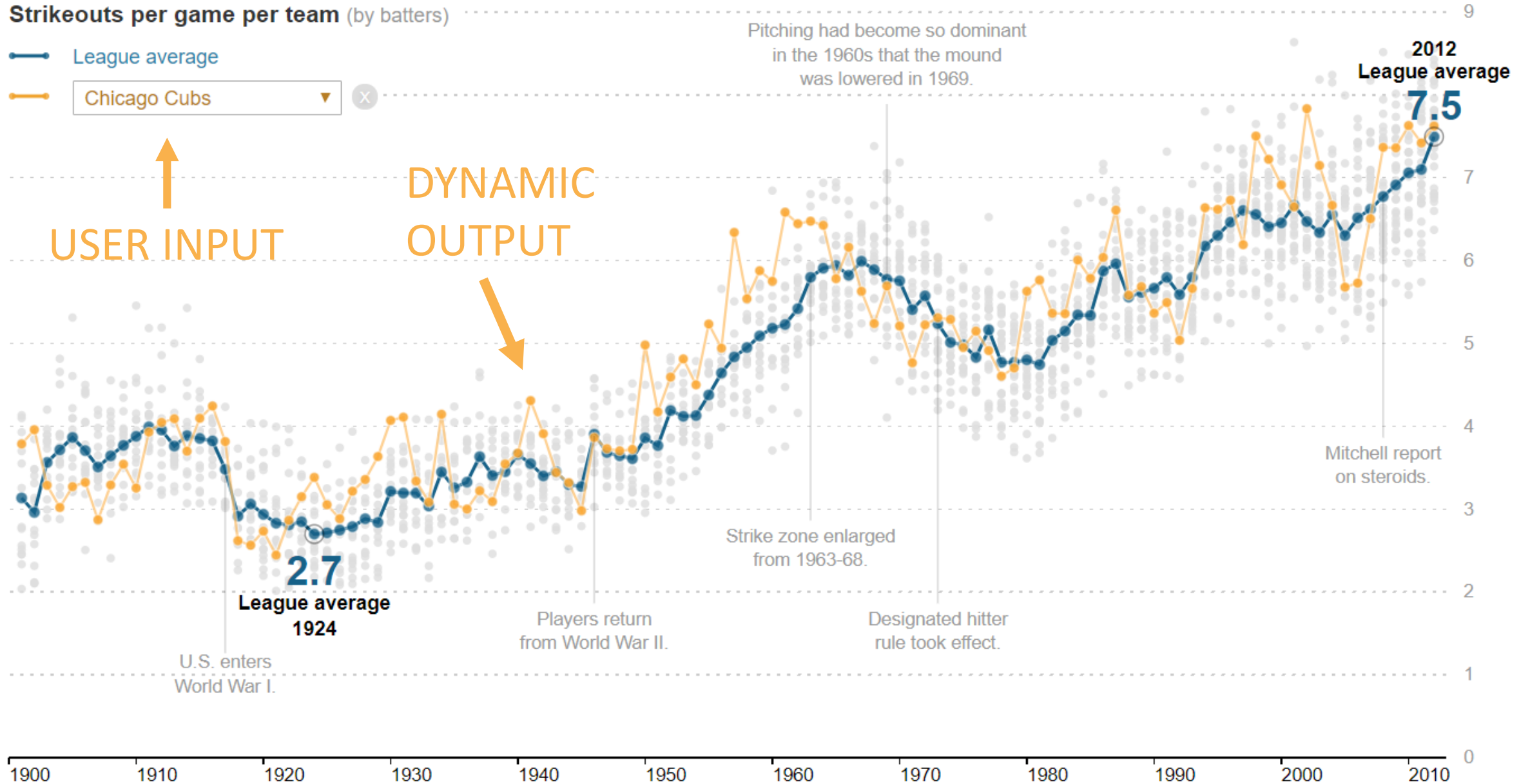
# Strikeouts on the Rise

There were more strikeouts in 2012 than at any other time in major league history.

Strikeouts per game per team (by batters)

League average

Chicago Cubs



# RENDER FUNCTIONS:

The output functions take R code and “render” it as HTML objects that can be used in web browsers in order to display your dashboard. Shiny functions add some javascript features that allow output to be updated in real-time inside a browser.

<b>Output Functions</b>	<b>Creates</b>
renderImage	image
renderPlot	plot
renderTable	table
renderText	text

Note that HTML creates static text, tables, and images in web documents. Any time you are doing something active on a webpage (other than clicking a link), you are using the javascript language. It was created as a way to make web pages more interactive and responsive.

- knitr → converts R to HTML when knitting RMD documents
- shiny functions → convert R to javascript when knitting RMD documents



# ANATOMY OF SHINY FUNCTIONS: DATA FLOW

## HTML Doc or Dashboard

Input widgets

collect  
parameters

render functions

sends rendered  
objects back to  
dashboard

## RMD File

Load data  
Load packages

Conduct analysis –  
generate tables and graphs

“dynamic” output means that the user can change something about the tables or graphs by selecting new input parameters

# ANATOMY OF SHINY FUNCTIONS: USER INPUT

## HTML Doc or Dashboard

Input widgets

sends rendered  
objects back to  
dashboard

collect  
parameters

render functions

## RMD File

Load data  
Load packages

Conduct analysis –  
generate tables and graphs

“dynamic” output means that the user can change something about the tables or graphs by selecting new input parameters

# INPUT WIDGETS

Building the user interface to gather user inputs

# Shiny Widgets Gallery

For each widget below, the Current Value(s) window displays the value that the widget provides to shinyServer. Notice that the values change as you interact with the widgets.

## Action button

Action

Current Value:

```
[1] 0  
attr(,"class")  
[1] "integer" "shinyActionButtonValue"
```

[See Code](#)

## Single checkbox

Choice A

Current Value:

```
[1] TRUE
```

[See Code](#)

## Checkbox group

Choice 1  
 Choice 2  
 Choice 3

Current Values:

```
[1] "1"
```

[See Code](#)

## Date input

2014-01-01

Current Value:

```
[1] "2014-01-01"
```

[See Code](#)

## Date range

2019-09-16

to

2019-09-16

Current Values:

```
[1] "2019-09-16" "2019-09-16"
```

[See Code](#)

## File input

Browse...

No file selected

Current Value:

```
NULL
```

[See Code](#)

# STANDARD SHINY WIDGETS (INPUTS)

## Function Name

actionButton

checkboxGroupInput

checkboxInput

dateInput

dateRangeInput

fileInput

helpText

numericInput

radioButtons

selectInput

sliderInput

submitButton

textInput

## Widget

Action Button

A group of check boxes

A single check box

A calendar to aid date selection

A pair of calendars for selecting a date range

A file upload control wizard

Help text that can be added to an input form

A field to enter numbers

A set of radio buttons

A box with choices to select from

A slider bar

A submit button

A field to enter text

Note each function will store different input values:

textInput = a single character element

selectInput = character elements from a list

sliderInput = two numbers in a range

checkboxInput = T / F

# WIDGET COMPONENTS

- **Name** for the widget. The user will not see this name, but you can use it to access the widget's value. The name should be a character string.
- **Label**. This label will appear with the widget in your app. It should be a character string, but it can be an empty string "".

```
actionButton( name="submit", label = "Submit Your Form")
```



Creates an entry at `input$submit`

How you will access the data:

`input$name`

Note that you do not create the input object and assign values at `input$widget_name`. That is done for you by the Shiny package.

# ANATOMY OF SHINY FUNCTIONS: USER INPUT

## HTML Doc or Dashboard

Input widgets

collect  
parameters

render functions

sends rendered  
objects back to  
dashboard

## RMD File

Load data  
Load packages

Conduct analysis –  
generate tables and graphs

User inputs collected from widgets will  
change the data or parameters used in  
the analysis, changing the output

# SHINY FUNCTIONS VS SHINY APPS

Using shiny in RMD documents versus building custom web apps



# USING SHINY IN RMD DOCUMENTS

Please note that if you look up tutorials on R Shiny you will find lots of information about how to build web apps. This was the original design and intended purpose of R Shiny.

HOWEVER, it requires you to build a user interface from scratch, so you need to learn a bunch of functions to lay out a web page and add container for widgets and output. These are useful if you need your final app to be fully customized where you have full control over the entire look and feel.

Shiny widgets have recently been integrated into RMD documents, making them much easier to use. The RMD documents do all of the hard work of creating a web page for us, and we already know how to add graphics. In this step we will just add widgets to gather user inputs.

We are using **Shiny functions()** inside of RMD documents, **we are NOT building Shiny apps.**

# USING SHINY IN RMD DOCUMENTS

```
---  
title: "Lab-04 Dynamic Graphics"  
output: flexdashboard::flex_dashboard  
runtime: shiny  
---
```

Header arguments in RMD

```
```{r}  
selectInput( inputId="my.widget",  
             label="Please select one",  
             choices=c("A","B","C"),  
             selected="B" )  
```
```

Widgets in RMD

```
```{r}  
renderPlot({  
  # code here  
})  
```
```

Render functions in RMD