Visualización (y otros) avanzada con R

Ecosistema

¿Qué herramientas tenemos a nuestra disposición?

Ecosistema: Librerías gráficas

Graficación

- *graphics:* librería estándar de R
- ggplot2: librería *de facto*, muy popular e intuituva. Implementa «grammar of graphics»[1]
- *plot.ly*: librería de visualización interactiva. Exporta a web. Múltiples lenguajes (p.e: Python)
- *htmlwidgets*: *framework para crear elementos web. Permite utilizar otras librerías web ya existentes (d3, dc, datatables...).*
- *Leaflet*: *estandar* de facto *para visualización geográfica*

[1] Wilkinson, L., & Wills, G. (2005). The grammar of graphics. New York: Springer.

Ecosistema: Documentos complejos

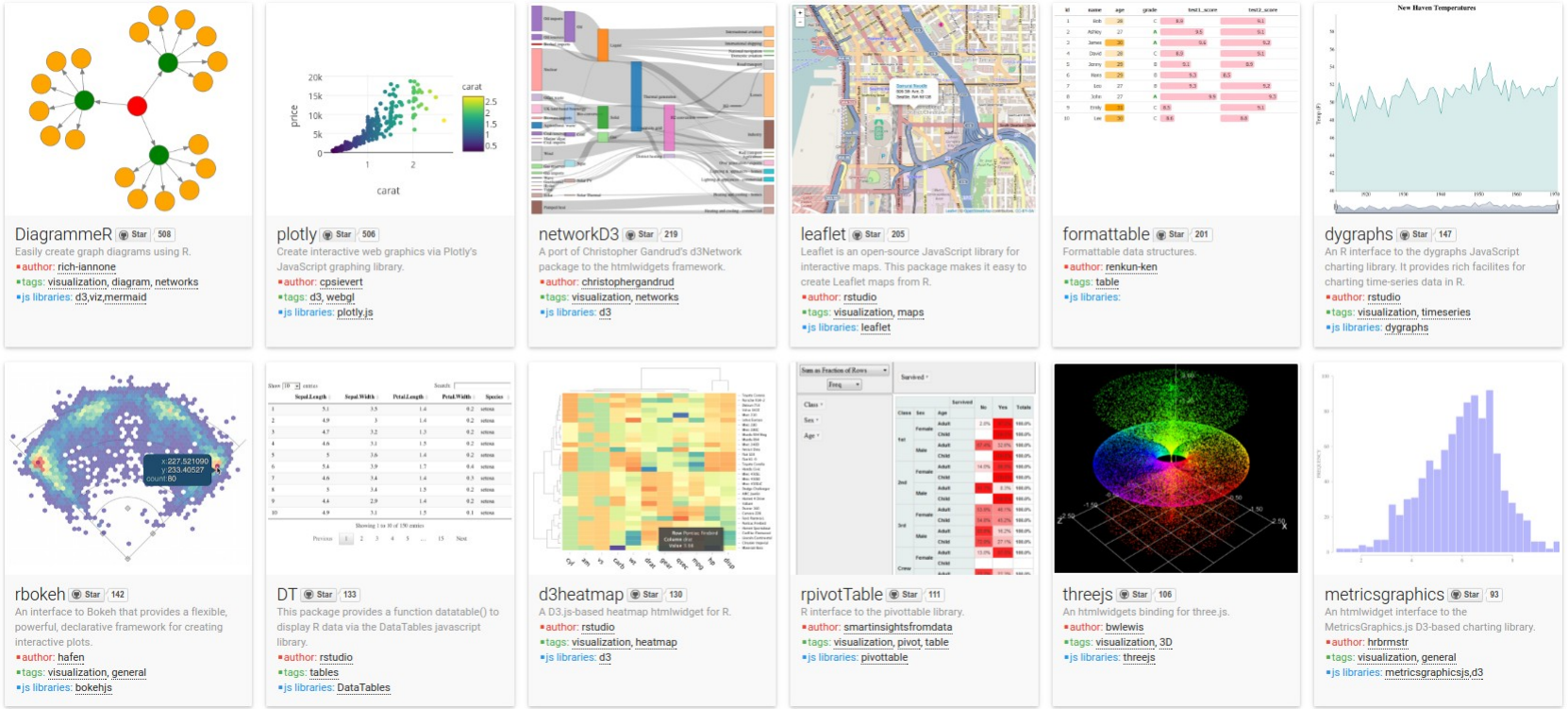
Documentos y dashboards

- knitr: librería para crear documentos (HTML -web-, PDF, DOC) incluyendo código de R, resultados, visualizaciones y htmlwidgets. Muy útil para hacer informes y diapositivas. En Rstudio se le conoce como “RMarkdown”
- shiny: framework para crear aplicaciones web. Muy útil para crear dashboards interactivos y aplicaciones finales.

El mundo de la visualización web y htmlwidgets

htmlwidgets: breve mención

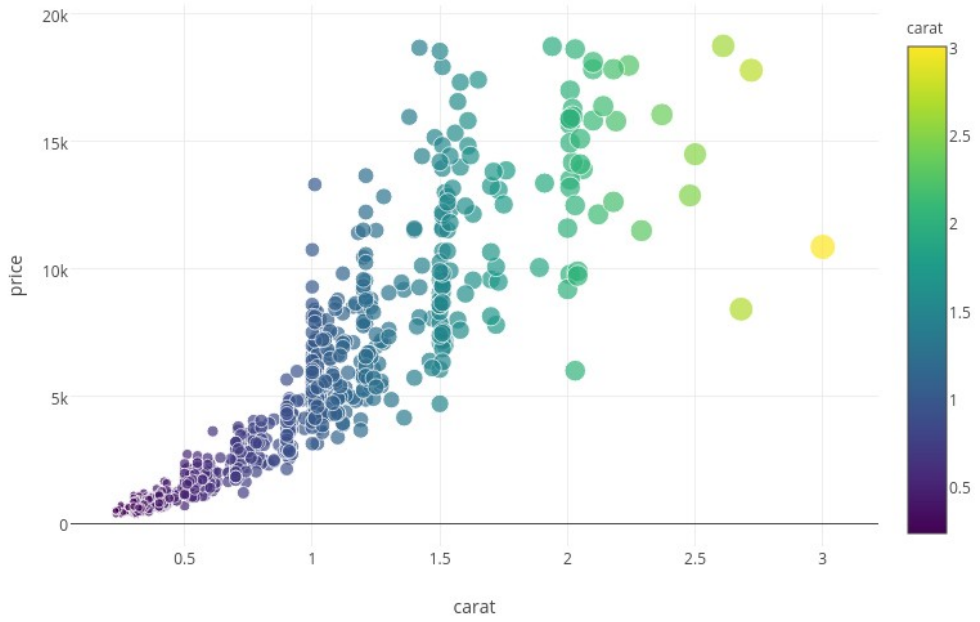
Permite utilizar librerías de javascript (D3, dc, bokeh, threejs…)
Hay muchos paquetes desarrollados: gallery.htmlwidgets.org
Funciona muy bien con Shiny y knitr



Plot.ly

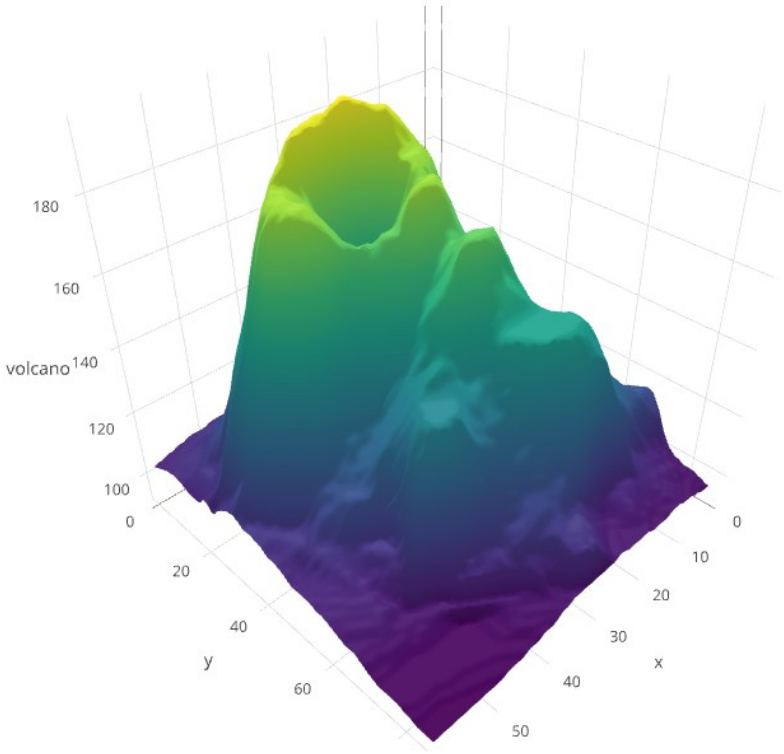
Mi recomendación si buscas visualización web con un buen equilibrio entre esfuerzo y resultado.
Sencilla

- Muy visual y estética con buena interacción.
- Usa htmlwidgets
- Funciona con ggplot2, R, Python, Scala,… Ejemplos



Plot.ly

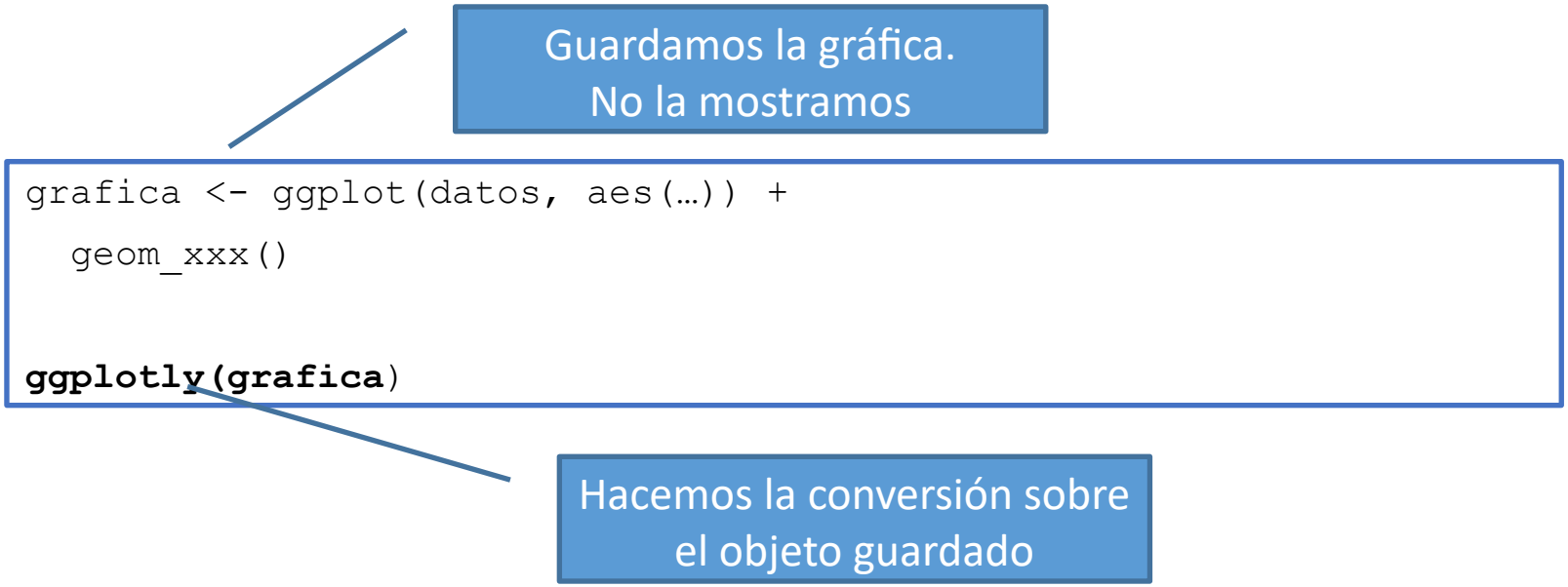
Ejemplo volcano



Plot.ly: convertir gráficos de ggplot

```
grafica <- ggplot(datos, aes(…)) +
  geom_xxx()

ggplotly(grafica)
```

Plot.ly: convertir gráficos de ggplot



Plot.ly: Ejercicio 11

Tomad el scatterplot con la regresión cuadrática (ejercicio 10, último apartado) y convertidlo a plot.ly

```
grafica <- ggplot(datos, aes(…)) +
  geom_xxx()


ggplotly(grafica)
```

Plot.ly: Ejercicio 11

```
grafica <- ggplot(mpg,
        aes(x=cty, y=displ)) +
  geom_point() +
  facet_wrap(~class, ncol=3) +
  stat_smooth(method = "lm",
formula= y~poly(x,2))
ggplotly(grafica)
```

En una línea:

- El resultado es mucho mejor estéticamente
- Texto más legible
- Interactividad
- Menú de acciones
- Respuesta más rápida al resize.

Knitr

¿Qué es?

- Es una librería de R que “cose” (o knit) documentos.
  - Coser es combinar fragmentos de texto con fragmentos de código de R
  - Exporta en HTML (web) con lo cual podemos incrustar cualquier htmlwidget
  - Esto significa… documentos interactivos

- Es MUY recomendable empezar a usarlo. Los resultados son positivos y tienen una visibilidad mucho mayor.

¿Qué es?

- Usos claros:
  - Reporting repetitivo (informes trimestrales de…)
  - Documentación para entregar al cliente final tras un estudio
  - Forma de compartir resultados de un análisis con compañeros (aunque se usa más el notebook para esto)
- Es útil aprender escritura programática.
  - Que el texto que se escriba se “programe” de manera que el texto puede cambiar según lo que pase.

Shiny

¿Qué es?

- Es un framework (o librería o entorno de trabajo...) que nos permite hacer aplicaciones de datos interactivas. Es decir:
  - Aplicaciones que responden ante las acciones del usuario
    - Ejemplo típico: dashboard
  - Aplicaciones que realizan cálculos (modelos, modificaciones de los datos) en vivo
- Cierra el ciclo de valor de un producto. No lo infravaloréis.
  - Que un DS/analista pueda hacer un producto final es una grandísima ventaja
    - (más fácil vender un producto que un modelo)
- Se entiende mucho mejor con ejemplos. Galería

Estructura de una interfaz de Shiny

- Hay dos archivos en todo proyecto de Shiny:
  - ui.R
  - server.R

- ui.R: Abreviatura de *user interface*. Contiene la definición de la interfaz (estructura: dónde está cada cosa)
- *server.R:* Contiene todos los cálculos y genera el contenido que aparece dentro de la estructura definida en ui.R

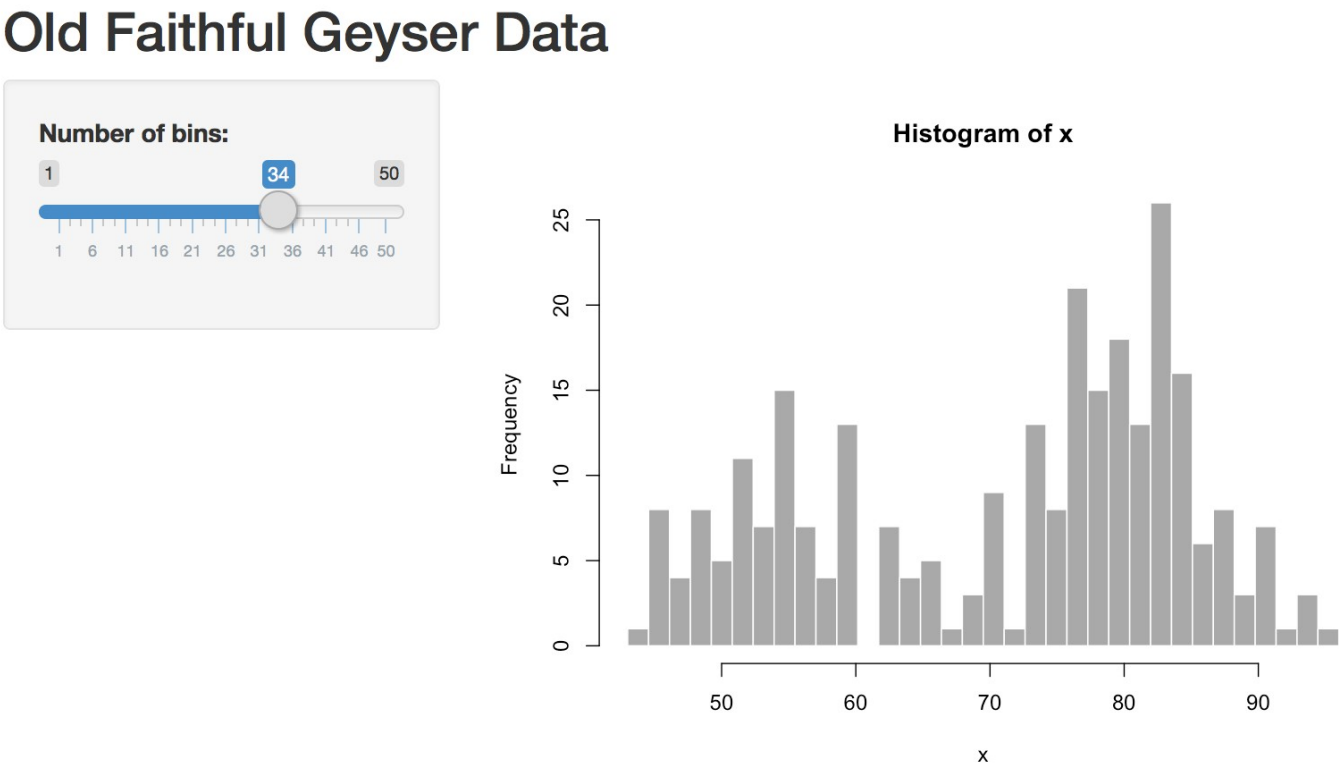
Tarea: Cread un nuevo proyecto de Rstudio y cread esos dos ficheros (vacíos)

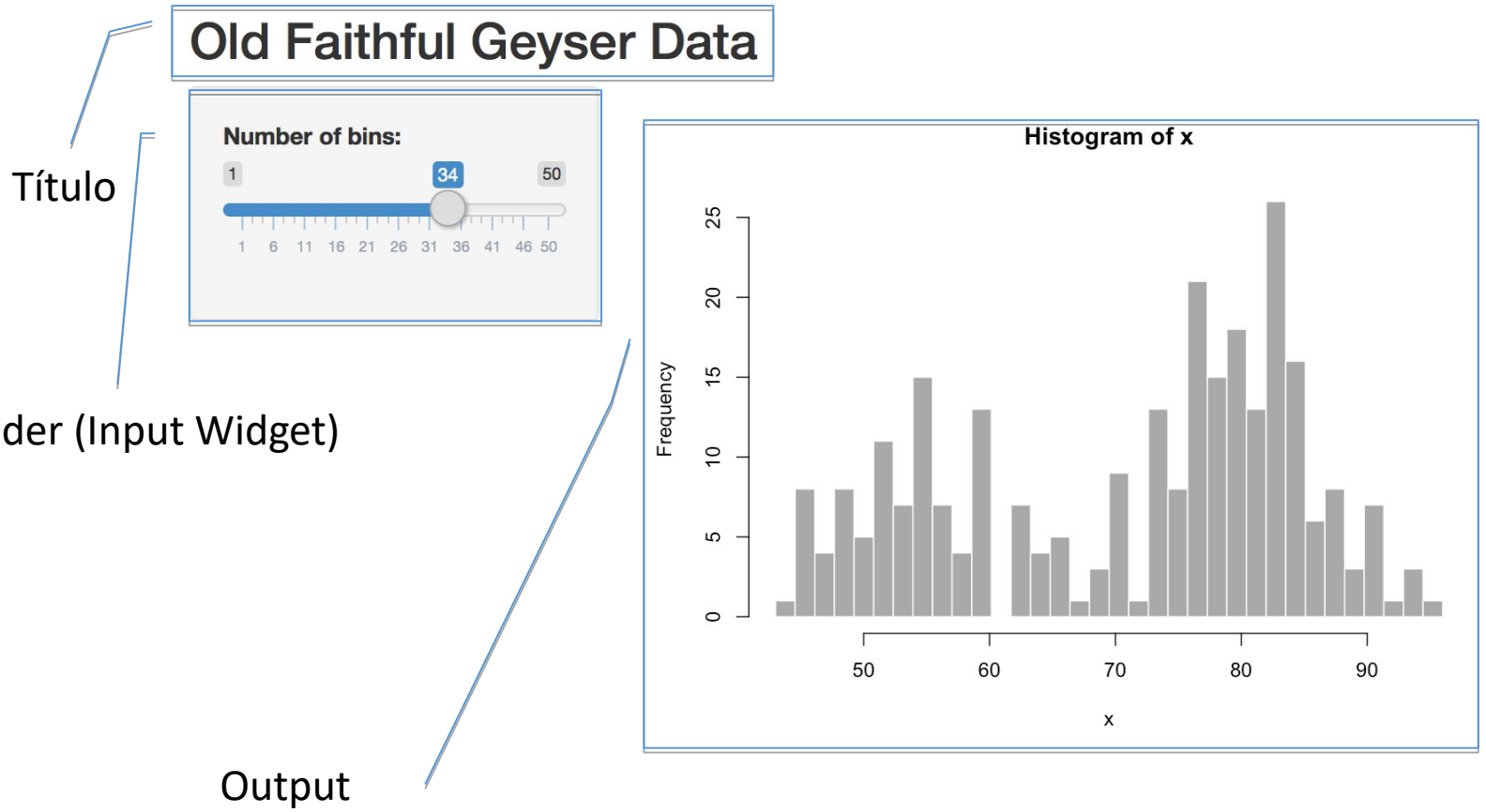
server.R

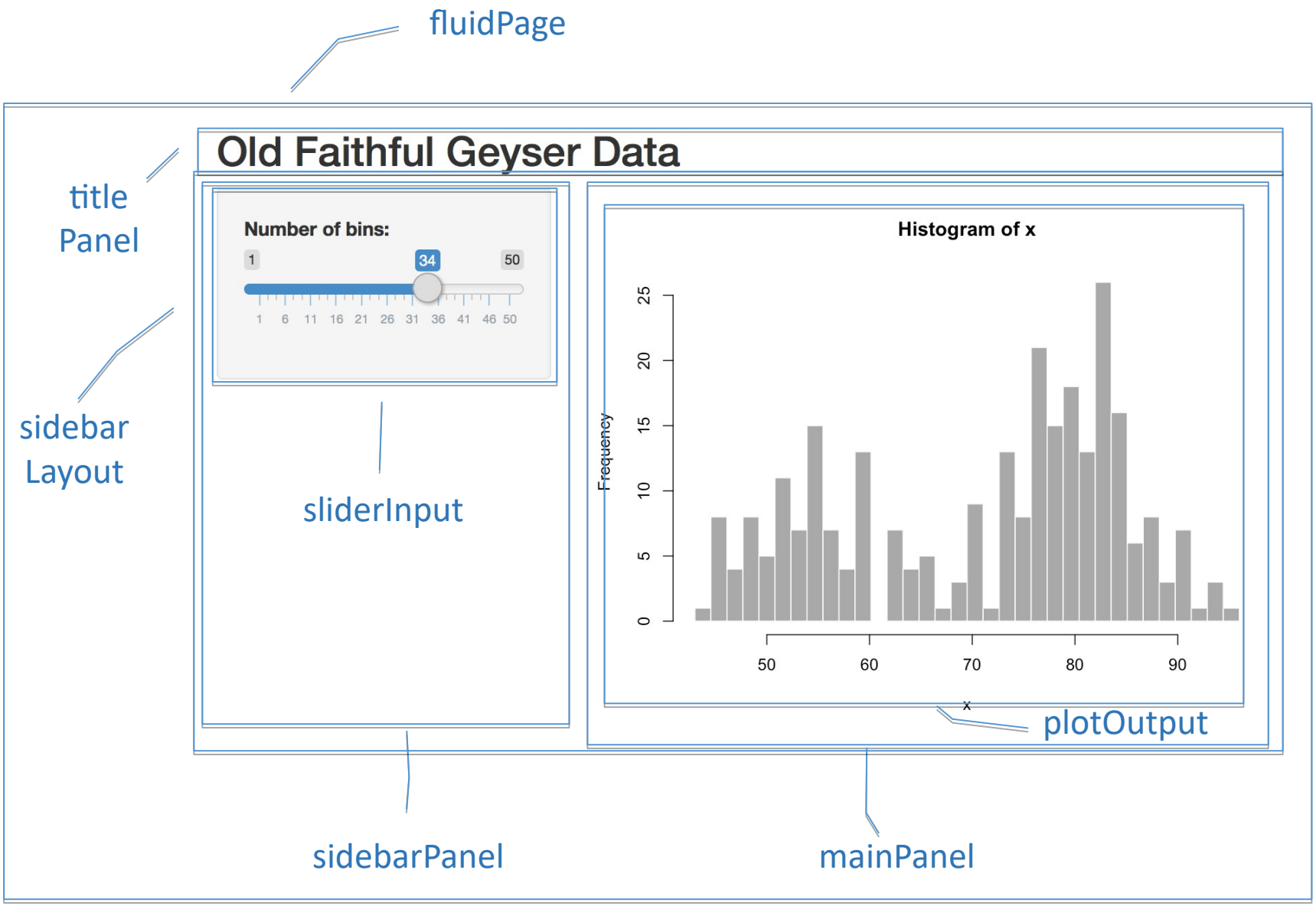
```
library(shiny)

shinyServer(function(input, output)
{…})
```

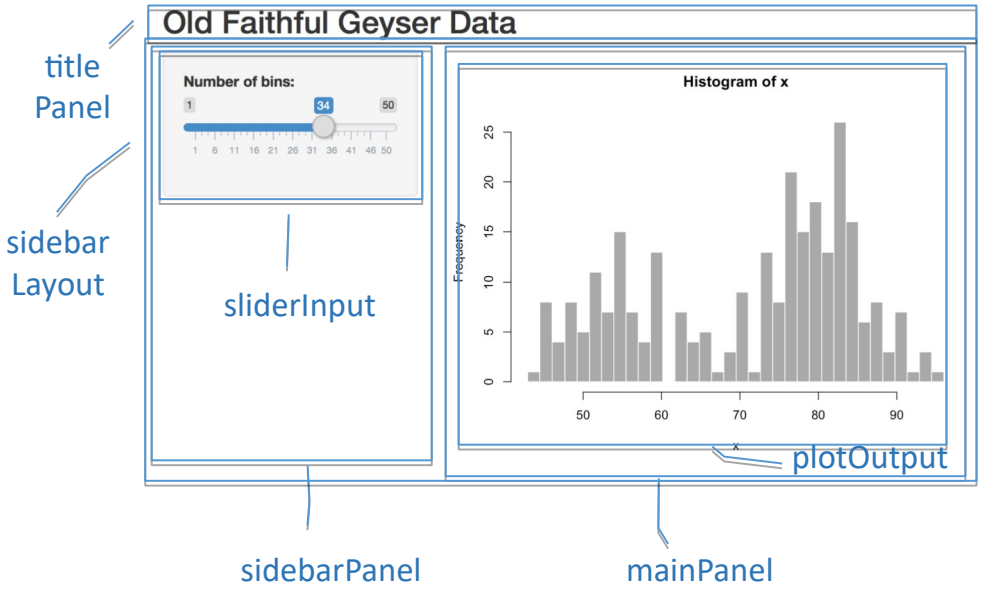
Tarea: Escribe esta función en server (quitad los puntos suspensivos)

Ejercicio: Old Faithful Geyser Data



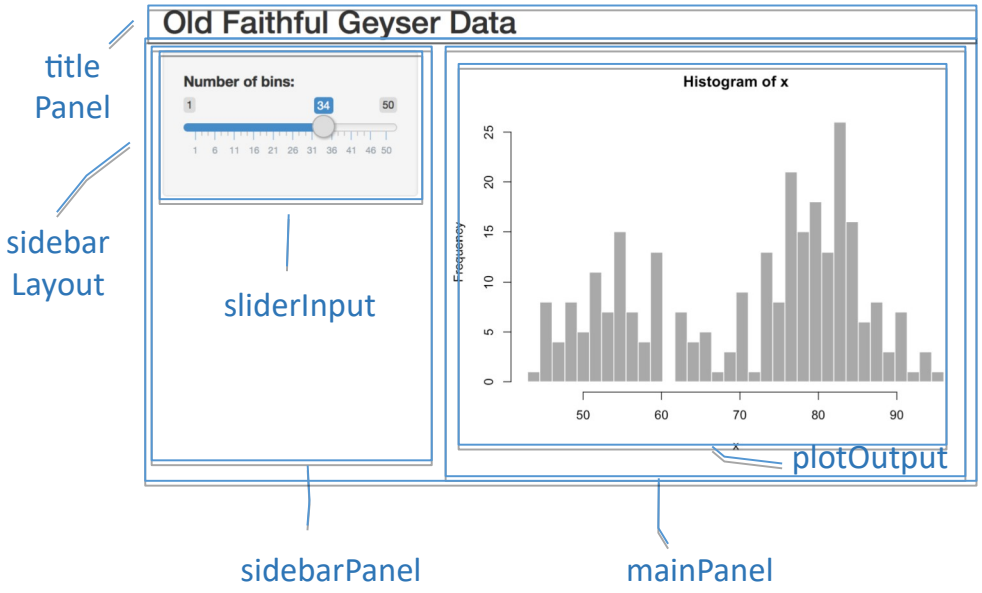

Old Faithful Geyser Data
Number of bins:
1
34
50
1 6 11 16 21 26 31 36 41 46 50
Histogram of x
Frequency
0
5
10
15
20
25
50
60
70
80
90
x
Título
Slider (Input Widget)
Output


fluidPage
Old Faithful Geyser Data
title Panel
Number of bins:
1
34
50
1 6 11 16 21 26 31 36 41 46 50
sidebar Layout
sliderInput
Histogram of x
Frequency
0 5 10 15 20 25
50 60 70 80 90
x
plotOutput
sidebarPanel
mainPanel



- Cualquier UI tiene una estructura jerárquica
  - Esta estructura siempre empieza con shinyUI

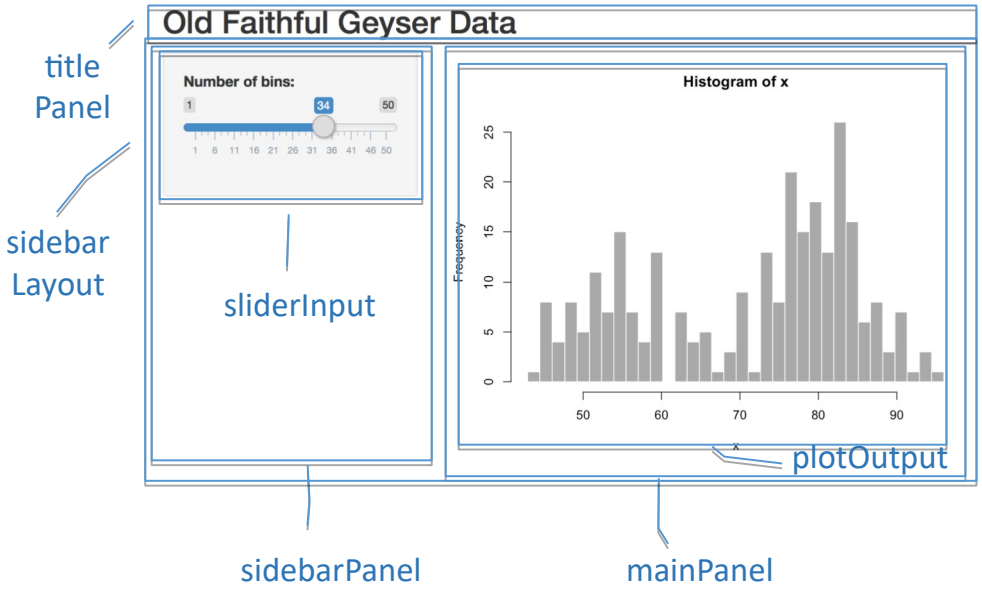
- shinyUI
  - fluidPage
    - titlePanel
    - sidebarLayout
      - sidebarPanel
        - sliderInput
      - mainPanel
        - plotOutput



- shinyUI
  - fluidPage
    - titlePanel
    - sidebarLayout
      - sidebarPanel
        - sliderInput
      - mainPanel
        - plotOutput

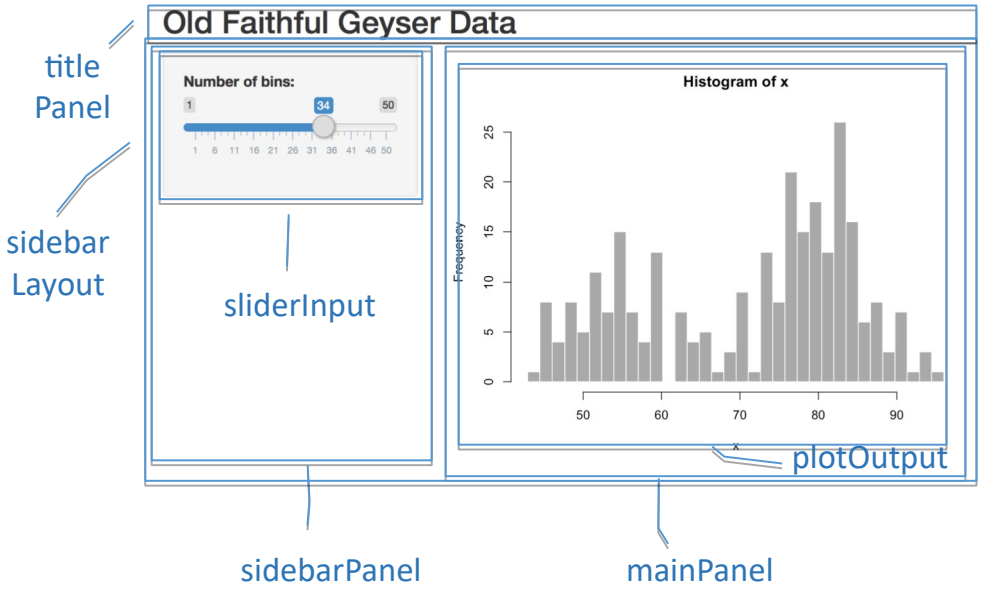
¿Cómo se escribe la estructura en ui.R?

```
library(shiny)

shinyUI(…)
```



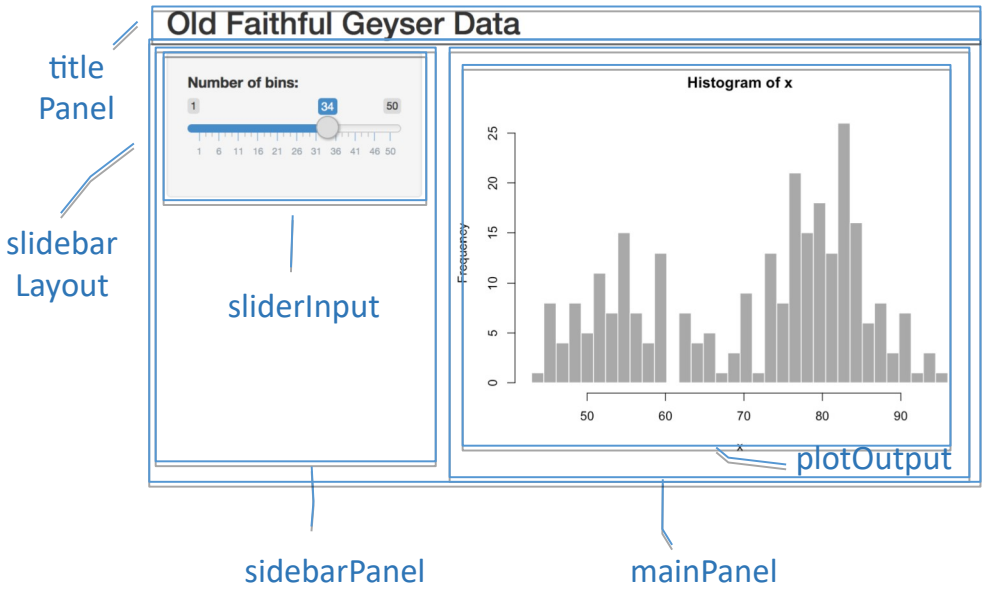
- shinyUI
  - fluidPage
    - titlePanel
    - sidebarLayout
      - sidebarPanel
        - sliderInput
      - mainPanel
        - plotOutput

```
library(shiny)

shinyUI(fluidPage(…))
```



- shinyUI
  - fluidPage
    - titlePanel
    - sidebarLayout
      - sidebarPanel
        - sliderInput
      - mainPanel
        - plotOutput

```
library(shiny)

shinyUI(fluidPage(titlePanel(...),sidebarLayout(...)))
```

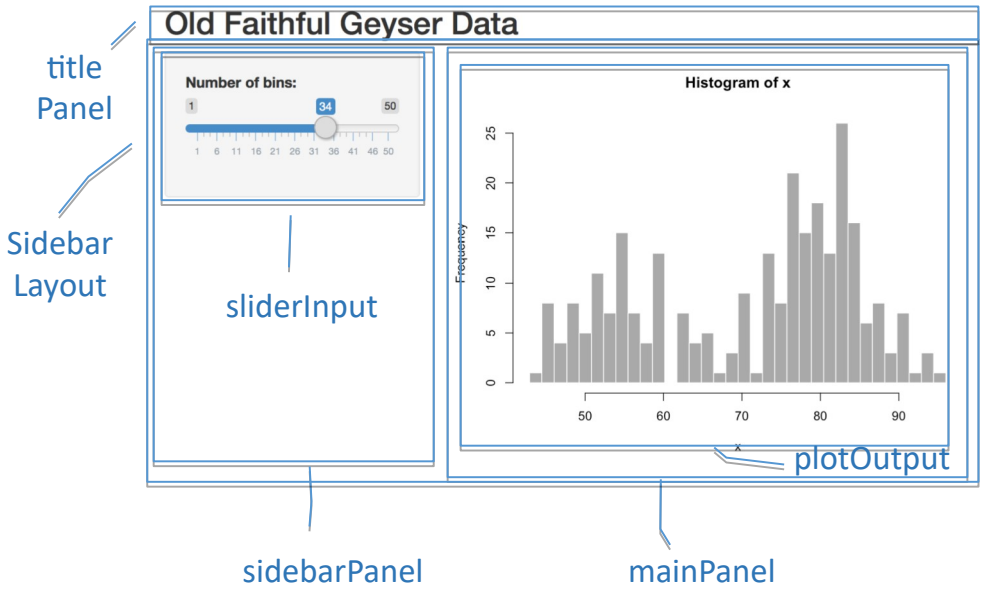


- shinyUI
  - fluidPage
    - titlePanel
    - sidebarLayout
      - sidebarPanel
        - sliderInput
      - mainPanel
        - plotOutput

```
library(shiny)

shinyUI(
    fluidPage(
        titlePanel(...),
        sidebarLayout(...)
    )
)
```

Tarea: Haced todo el árbol del UI pero sin poner los dos elementos en rojo (sliderInput y plotOutput)

Tarea: Poner el titlePanel el parámetro title a title="Old Faithful Geyser Data"



- xxxxPage: Tipo de página: fluida o de ancho fijo
- xxxxPanel: contenedor, no tiene significado especial. Sólo formato.
- xxxxLayout: layout o disposición: es una forma de distribuir a los hijos del layout
- xxxxInput: todo widget diseñado para que el usuario introduzca un valor
- xxxxOutput: placeholder o hueco para introducir una salida u output.

Tarea: Añadir el sliderInput en el lado izquierdo

(Mirad la documentación. Google: “shiny R sliderInput”)

Tiene que llamarse: “bins” y el label debe ser “Número de bines”. Poner correctamente el máximo y el mínimo (1 y 50) y el valor por defecto (5)

Tarea: Añadir la siguiente gráfica en el mainPanel.

1. Tenéis que usar renderPlot y plotOutput
2. La gráfica es:

```
hist(faithful$eruptions)
```

Conectar ui.R y server.R

Ya tenemos una estructura visual (ui.R) y un servidor (server.R), por ahora vacío.

Para que el servidor (server.R) pueda introducir resultados en la interfaz (ui.R) se tiene que usar el par xxxxOuput y renderXxxx.

En ui.R se coloca el hueco a rellenar (xxxxOutput)

En server.R se coloca el “render” con el cálculo correspondiente (renderXxxx) para rellenar

Conectar ui.R y server.R

La pareja output-render tiene un identificador único (un nombre) que los relaciona.

Por ejemplo:

```
ui.R:       renderXxxx("miTexto")
server.R:   output$miTexto <- renderXxxx(...)
```

Tips: siempre en minúscula o en camelCase, sin espacios y fácil de recordar

Conectar ui.R y server.R

En el server.R completas el renderXXXX con el cálculo que quieras:

- Entrenar un modelo
- Hacer una gráfica
- ....

En el ui.R colocas el XXXXoutput en la parte de la interfaz dónde quieras colocar este resultado

Ejemplo básico. Vamos a conectar un pequeño texto.

textOutput("miTexto")

output$miTexto <- renderText(“Hola”)

Recibir interacción del usuario

¿Cómo sabe R entonces qué input del usuario “se conecta a que output” del usuario?

```
shinyServer(
   function(input, output) {
   output$miTexto <- renderText(paste0("Me
encanta R", input$xxxxx))
   }
)
```

Al usar input$XXXXX dentro de renderText en miTexto los estás “enlazando”

Pequeños detalles: frecuencia de actualización

- ¿Cuándo ha cambiado la gráfica?
  - ¿En cuanto muevo el slider?
  - ¿Cuándo suelto el ratón?
  - ¿Al pasar un tiempo?

Resumen

Shiny tiene dos ficheros básicos:

- ui.R: estructura de la interfaz. “Huecos” o placeholders
- server.R: cálculos. Rellenar esos placeholders

Para conectar la interfaz con el servidor:

- Crear un par renderXxxx y xxxxxOutput
  - Tienen que tener el mismo identificador

    output$blah <- renderText(…)

    textOutput(”blah")

Resumen

Hay varios tipos de renders y de outputs, deben ser acordes con el contenido que va a haber en ese contenedor.

En ui.R…

- Hay una estructura jerárquica (árbol) que empieza con shinyUI
- Hay que tener cuidado al cerrar los paréntesis
- Los nodos hijos están dentro de los paréntesis de los padres
- Los nodos hermanos se separan con comas

Resumen

En server.R...

- Hay UNA función dentro de shinyServer
- Recibe input y output
  - Son dos listas
  - En input recibes la respuesta del usuario a los widgets

    p.e: miFuncion(input$blah + 5)
  - En output puedes guardar el contenido para que lo reciban los placeholders/xxxOutput

    p.e: output$other <- renderPlot(ggplot(...) + ...)

Tarea para casa: Reproducir el ejemplo de “Telephones by Region” de la documentación de Rstudio. Usad ggplot2 para la gráfica

http://shiny.rstudio.com/gallery/telephones-by-region.html

- Tips:
  - Cread un NUEVO proyecto en Rstudio
  - Pensad el árbol de la interfaz (muy similar a lo que hemos hecho hoy)
  - Primero haced un servidor vacío y la estructura básica del ui.R
  - Después completadlo

Tenéis la solución en la misma página, pero sin ggplot2 (don’t cheat)

Hay un foro para añadir las dudas o contactadme por correo para cualquier duda. Practicar es la única opción para aprender a usar un lenguaje (formal -R- o natural -Inglés-).

Ejemplo: Interactividad intermedia (resuelto en los adjuntos)