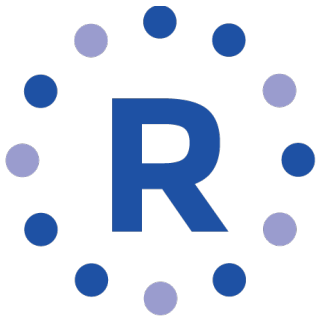
Introducción a Shiny

Cómo crear páginas web usando R

Diego Marfetán Molina

VII Encuentro del Grupo de Usuaries de R en Rosario
18 de Diciembre de 2019



Introducción: ¿Qué es Shiny?

- Shiny es un paquete de `R` que permite construir páginas web directamente desde RStudio
- Una característica importante de las aplicaciones web creadas mediante Shiny es que son **dinámicas e interactivas**
- Esto implica que los usuarios pueden decidir qué datos ver, cómo verlos, y *jugar* con ellos
- En la página web oficial de Shiny existen numerosos ejemplos y tutoriales altamente recomendables para aquellas personas que están dando sus primeros pasos con esta herramienta

Introducción: ¿Cómo instalar Shiny?

- Shiny puede instalarse como cualquier otro paquete de R:

```
install.packages("shiny", dependencies = TRUE)
```

- Una vez instalado, debemos cargarlo para poder empezar a usarlo:

```
library(shiny)
```

- Para que **shiny** funcione correctamente, es necesario tener instalado `R 3.0.2` o cualquier versión posterior (recomendamos estar al día con las actualizaciones, preferentemente `R 3.6.0` o mayor)

Introducción: ¿Cómo funciona Shiny?

- A lo largo de este taller usaremos el término **Shiny App** para referirnos al conjunto de sentencias que generan la página web
- Una Shiny App debe guardarse en un archivo de sentencias de `R`, al cual llamaremos **app.R**
- El archivo app.R posee 3 componentes bien diferenciados:
  1. Interfaz del Usuario
  2. Función *server*
  3. Función *shinyApp*

Estructura de una Shiny App: Interfaz

- La interfaz del usuario (*user interface* o *ui*, por sus siglas en inglés) se encarga de **controlar el aspecto** de la página web
- En general, definir las características de la interfaz es la tarea que más incomoda a quienes están aprendiendo a trabajar con Shiny
- Esto se debe a que muchas de sus herramientas están vinculadas a otros lenguajes de programación, por ejemplo HTML, CSS o JavaScript
- Afortunadamente, las funciones del paquete Shiny facilitan en gran medida nuestro trabajo y no debemos preocuparnos (demasiado) por aprender a utilizar esos lenguajes

Estructura de una Shiny App: Server

- En la sección *server* se escribe el código de `R` que le indica a la app qué debe hacer y cómo debe funcionar
- Aquí se incluyen la lectura y manipulación de datos, el armado de gráficos, el ajuste de modelos, etc.
- Generalmente, las sentencias incluidas dentro de este bloque dependen de un cierto **input** (datos) y devuelven un determinado **output** (resultados, tablas, gráficos, mapas, etc.)

Estructura de una Shiny App: Ejecución

- La parte final es un llamado a la función **shinyApp**, cuyos dos argumentos principales son *ui* y *server*
- Ejecutar esta función da como resultado el lanzamiento de la aplicación, la cual podremos utilizar dentro de RStudio o usando nuestro navegador preferido (Chrome, Firefox, Safari, Explorer, etc.)
- Es importante destacar que, al seguir estos pasos, la aplicación sólo funcionará mientras la sesión de RStudio desde la cual se lanzó siga vigente
- Más adelante veremos cómo subir nuestra aplicación a la web para que cualquier persona, aun las que no poseen `R` instalado, puedan acceder a ella

Ejemplo: datos del Titanic

- Para lanzar nuestra primera app, trabajaremos con un conjunto de datos referido al hundimiento del Titanic
- Estos datos reflejan el porcentaje de pasajeros que sobrevivieron, según género y clase en la que viajaban

```
datos <- read.table("titanic.txt", header = T, sep = "\t")
```

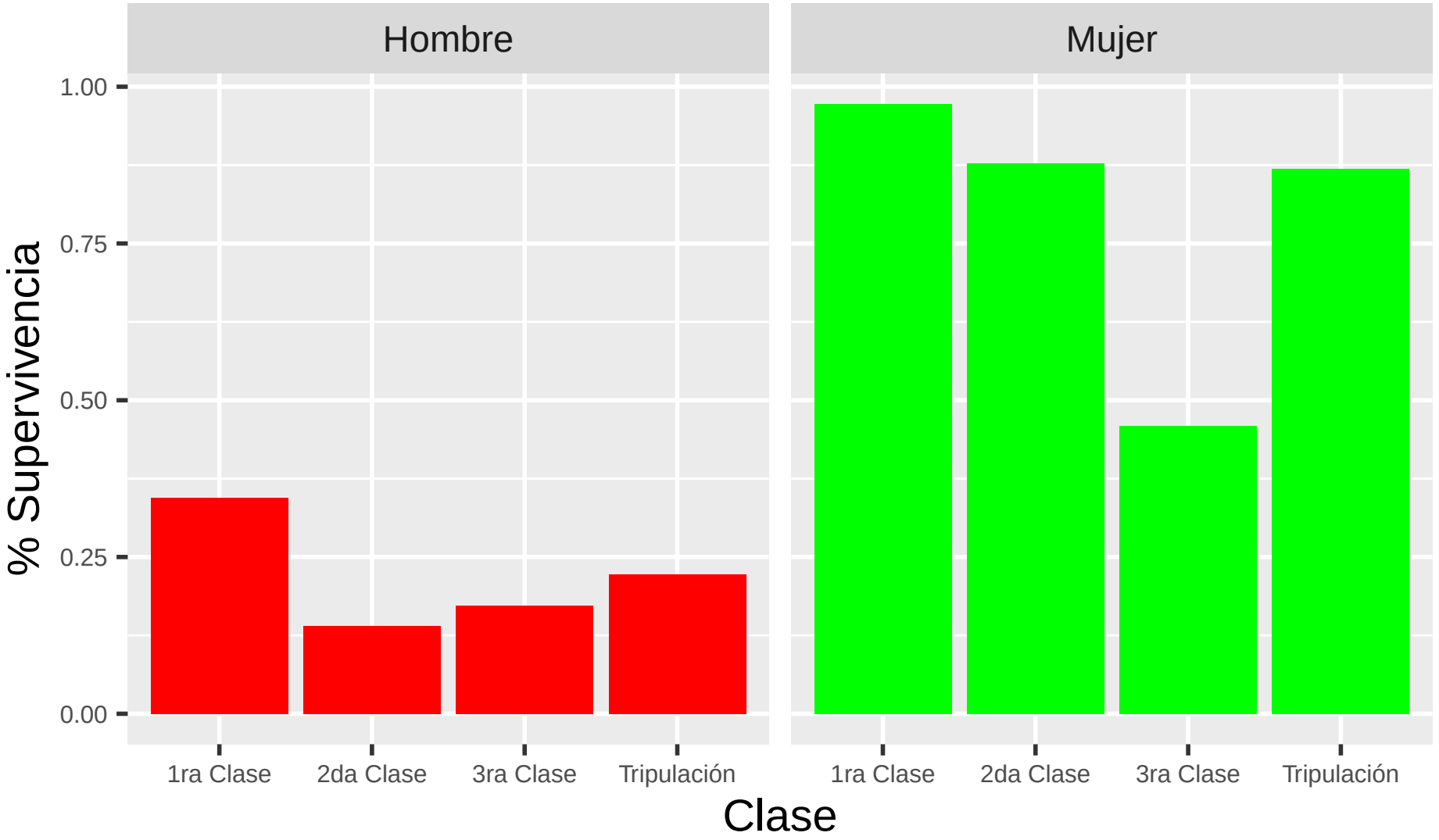
Clase	Sexo	Prop
1ra Clase	Hombre	0.3444444
1ra Clase	Mujer	0.9724138
2da Clase	Hombre	0.1396648
2da Clase	Mujer	0.8773585
3ra Clase	Hombre	0.1725490
3ra Clase	Mujer	0.4591837
Tripulación	Hombre	0.2227378
Tripulación	Mujer	0.8695652

Ejemplo: datos del Titanic

- Nuestro objetivo será comparar los porcentajes de supervivencia en los diferentes grupos, dejando que el usuario elija qué género quiere visualizar
- El siguiente código crea un gráfico de barras con la proporción de personas que sobrevivieron a la tragedia, usando el paquete **ggplot2**:

```
ggplot(data = datos,
       aes(x = Clase, y = Prop, fill = Sexo)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Sexo) +
  scale_fill_manual(values = c("red", "green")) +
  ylab("% Supervivencia") +
  theme(legend.position = "none",
        axis.text = element_text(size = 6))
```

Ejemplo: datos del Titanic



Armando nuestra primera app

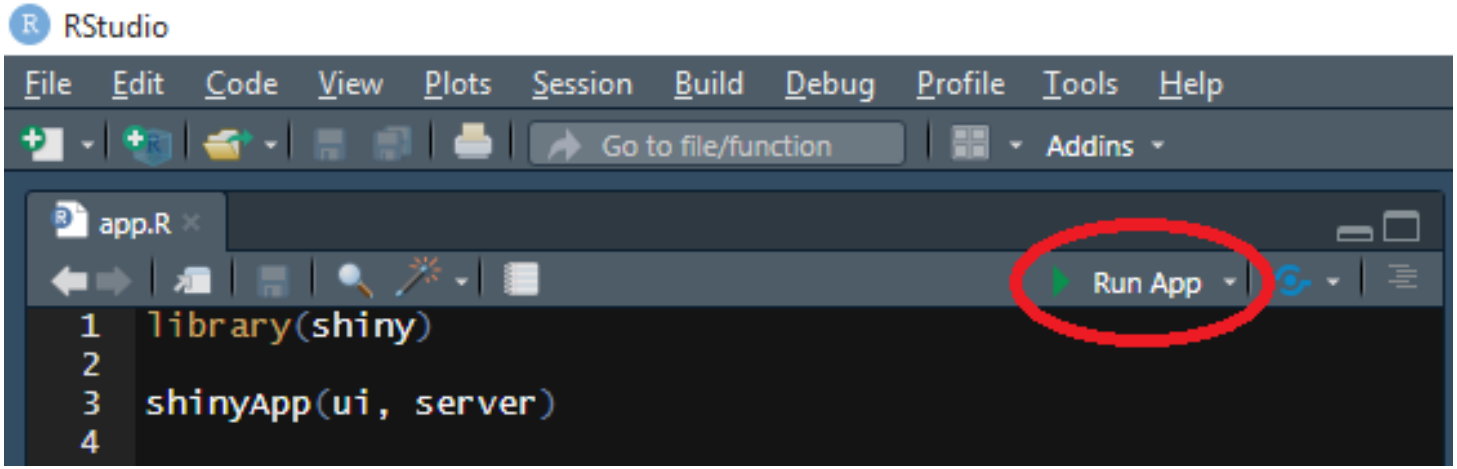
- Ahora que ya tenemos nuestros datos cargados y un objetivo claro, **llegó el momento de construir una Shiny App desde cero**
- Una recomendación importante es guardar cada app en un directorio único, donde no haya apps creadas previamente ni otros archivos innecesarios
- En consecuencia, para empezar a armar nuestra primera app, comenzaremos por crear un nuevo directorio y guardar allí dos archivos:
  1. app.R (por ahora vacío) con codificación UTF-8
  2. titanic.txt

Armando nuestra primera app

- Lo primero que podemos hacer dentro de app.R es cargar los paquetes a utilizar, leer los datos y hacer un llamado a la función *shinyApp*:

```
library(shiny)
library(tidyverse)
datos <- read.table("titanic.txt", header = T, sep = "\t")
shinyApp(ui, server)
```

- Al detectar la función *shinyApp*, RStudio reconoce que nuestro archivo es una aplicación web y agregará la opción **Run App** en la barra de herramientas



Armando nuestra primera app: Interfaz

- El segundo paso consiste en definir el aspecto de la página web
- Algunas de las funciones creadas en Shiny para este propósito son *fluidPage*, *fixedPage*, *fillPage* y *navbarPage*, todas con diferentes estructuras predefinidas
- Comenzaremos utilizando *fluidPage*, la cual permite una gran flexibilidad a la hora de diseñar la aplicación

Armando nuestra primera app: Interfaz

- Dado que estamos dando nuestros primeros pasos con Shiny, vamos a enfocarnos en construir una página web simple, conformada por dos paneles
- El primer panel (*sidebarPanel*) se ubicará a la izquierda y contiene los comandos que nos permiten controlar el output
- El segundo panel (*mainPanel*) es el de mayor tamaño y contiene los resultados que queremos visualizar

Armando nuestra primera app: Interfaz

- Podemos armar la estructura recién descripta usando la función *sidebarLayout* dentro de *fluidPage*
- El esquema de la interfaz luce así:

```
fluidPage( #Estructura general de la web
  titlePanel(), #Título de la web
  sidebarLayout( #Función para crear paneles
    sidebarPanel(), #Panel de la izquierda
    mainPanel() #Panel principal
  )
)
```

Armando nuestra primera app: Interfaz

- Dentro del *sidebarPanel* nos gustaría tener una lista de opciones donde poder elegir qué pasajeros mostrar
- En Shiny, este tipo de listas se construyen con la función *checkboxGroupInput*
- Llamaremos a nuestra lista *"MiLista"* y le asignamos las opciones *Mujer* y *Varón*

```
checkboxGroupInput(
  inputId = "MiLista", #Etiqueta del elemento
  label = "Sexo a Visualizar", #Título de la Lista
  choices = c("Mujer", "Hombre"), #Lista de categorías
  selected = "Mujer" #Valor elegido por defecto
)
```

Armando nuestra primera app: Interfaz

- El otro elemento a definir es el gráfico que mostraremos en el panel principal
- Mediante el siguiente código le avisamos a `R` que dentro del *mainPanel* habrá un objeto llamado *MiGrafico*
- La función que utilizamos es *plotOutput*, la cual genera visualizaciones en aplicaciones web

```
mainPanel(
  plotOutput("MiGrafico")
)
```

Armando nuestra primera app: Interfaz

- Uniendo todo lo dicho anteriormente, nuestra interfaz luce así:

```
interfaz <- fluidPage(
              titlePanel("Mi Primer Shiny App!!!"),
              sidebarLayout(
                sidebarPanel(
                  checkboxGroupInput(
                    inputId = "MiLista",
                    label = "Sexo a Visualizar",
                    choices = c("Mujer", "Hombre"),
                    selected = "Mujer")
                ),
                mainPanel(
                  plotOutput("MiGrafico")
                )
              )
            )
```

Armando nuestra primera app: Server

- Ahora podemos ocuparnos de definir la sección *server*
- Este bloque estará constituido por una función con dos argumentos: **input** y **output**:
  1. en **input** se guardarán los valores que pueden cambiar según el deseo del usuario, en este caso la variable *Sexo* almacenada en el objeto *MiLista*
  2. en **output** se guardarán los resultados que dependen de los valores elegidos en **input**, en este caso los gráficos almacenados en el objeto *MiGrafico*

Armando nuestra primera app: Server

- Mientras se esté ejecutando la Shiny App, el valor `input$MiLista` se actualizará de acuerdo a qué categorías de la variable seleccionemos en la pantalla
- El *truco* que utilizaremos es filtrar el conjunto de datos original de acuerdo al valor que en cada momento tome `input$MiLista`, y armar nuestro gráfico en base al nuevo dataset

```
datos_filtro <- datos %>% filter(Sexo %in% input$MiLista)

ggplot(datos = datos_filtro,
       aes(x = Clase, y = Prop, fill = Sexo)) +
  geom_bar(stat = "identity") +
  facet_wrap(~ Sexo) #+ etc
```

Armando nuestra primera app: Server

- Agregando la función *renderPlot* para generar el gráfico, la función del server resulta:

```
MiServer <- function(input, output) {

  output$MiGrafico <- renderPlot({

    datos %>% filter(Sexo %in% input$MiLista) %>%
    ggplot(aes(x = Clase, y = Prop, fill = Sexo)) +
      geom_bar(stat = "identity") +
      facet_wrap(~ Sexo) +
      scale_fill_manual(values = c("red", "green")) +
      ylab("% Supervivencia") +
      theme(legend.position = "none",
            axis.text = element_text(size = 6))
  })
}
```

Armando nuestra primera app: Ejecución

- Ahora que ya tenemos listas la interfaz y el server, sólo falta agregar la sentencia que permite ejecutar la aplicación:

```
shinyApp(ui = interfaz, server = MiServer)
```

- Esta línea de código debe figurar siempre dentro del archivo app.R, respetando los nombres asignados a la interfaz y al server

Armando nuestra primera app: Ejecución

- Existen 3 maneras diferentes de lanzar la aplicación:

1. Apretar el botón **Run App** en la barra de herramientas de RStudio, donde podemos elegir si queremos ejecutarla dentro de RStudio o en un navegador externo
2. Entrar a app.R, seleccionar todo el código y ejecutarlo
3. Usar la función runApp en la consola de RStudio, eligiendo la ruta donde está ubicado el directorio que contiene la app:

```
runApp("C:/Mis Documentos/MiApp")
```

- **Importante:** para detener la app y poder seguir utilizando `R`, es necesario apretar el botón rojo "STOP" en RStudio

Agregando detalles a mi Shiny App

- Si bien ya aprendimos a ejecutar una aplicación web desde RStudio, aún tenemos mucho camino por recorrer dentro del universo Shiny
- Dado que el tiempo disponible en este taller es limitado y la cantidad de opciones que ofrece Shiny es enorme, recomendamos una vez más consultar el tutorial oficial
- Ahora nos enfocaremos en agregar ciertos detalles básicos a nuestra web y recorrer algunos ejemplos muy interesantes que pueden servir de inspiración para crear apps más complejas

Agregando detalles a mi Shiny App

- Es común querer agregar títulos, subtítulos y texto en la sección de interfaz, ya sea para explicar el funcionamiento de la app, describir las variables incorporadas, citar la fuente de los datos, etc.
- Aquí es donde entra en juego el lenguaje HTML, el cual utilizaremos indirectamente a través de ciertas funciones del paquete *shiny*
- Algunas de las más utilizadas son:
  - **h1:** título principal
  - **h2:** subtítulo
  - **p:** párrafo de texto
  - **br:** salto de línea
  - **strong:** texto en negrita
  - **em:** texto en cursiva
  - **img:** agrega una imagen

Agregando detalles a mi Shiny App

- En general, ubicaremos las funciones antes mencionadas dentro del *mainPanel*, cada una separada de la siguiente por una coma:

```
mainPanel(
  h1("Hundimiento del Titanic"),
  h2("Comparando el % de Supervivencia según Clase y Sexo"),
  p("Texto que describe los datos y cómo funciona la app"),
  img(src = "Sticker_R.png", height = 100),
  br(),
  em(strong("Texto importante en negrita y cursiva")),
  plotOutput("MiGrafico")
)
```

- Para tener en cuenta: cuando adjuntamos una imagen, esta debe estar ubicada en una carpeta llamada **www** dentro del directorio donde está guardada la app

Agregando detalles a mi Shiny App

- Las listas desplegables del estilo *multiple choice* que vimos hasta ahora no son los únicos elementos (widgets) que podemos utilizar en una shiny app
- Otros widgets populares son:
  - sliderInput: barra horizontal que permite elegir un valor numérico dentro de un rango determinado
  - numericInput: celda donde se puede escribir un valor numérico
  - textInput: celda donde se puede escribir cualquier texto
  - fileInput: permite subir un archivo desde mi PC para ser utilizado por la app
- Al usar cualquiera de estos widgets debemos definir su ID (nombre interno dentro de R) y su etiqueta (título que se mostrará en la web)
- Para aquellos que involucran variables numéricas, es necesario definir los valores mínimos y máximos aceptados, además del valor por defecto

Agregando detalles a mi Shiny App

- Ejemplo: queremos construir un histograma a partir de datos generados al azar con distribución Normal
- El usuario puede elegir el tamaño de muestra, la cantidad de barras y el título del gráfico
- El sidebarPanel incluye un llamado a cada uno de estos widgets, todos con su correspondiente ID, etiqueta (*label*) y valores por defecto:

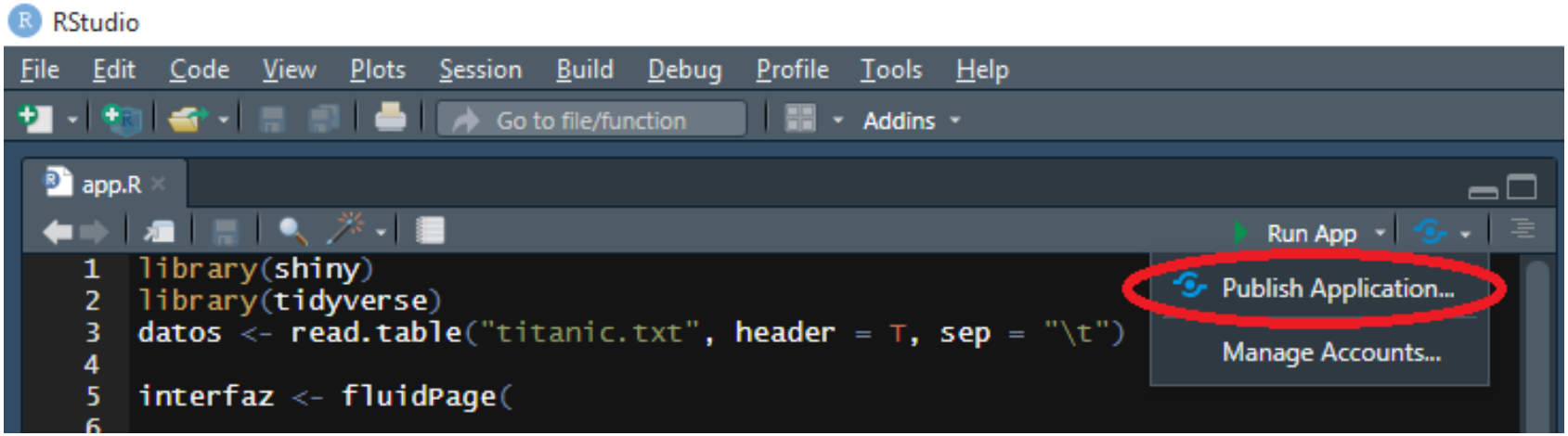
```
sidebarPanel(
  sliderInput(inputId = "n", value = 100, min = 1, max = 500,
              label = "Tamaño de Muestra"),
  numericInput(inputId = "bins", value = 20,
               label = "Cantidad de barras"),
  textInput(inputId = "tit", value = "Histograma Muestral",
            label = "Título del Gráfico"),
)
```

Subir mi Shiny App a la web

- Una pregunta muy común es: ¿cómo compartir una shiny app con otra persona, para que pueda usarla sin necesidad de estar ejecutándola desde RStudio?
- RStudio ofrece tres respuestas a esta pregunta, cada una orientada a diferentes perfiles de usuaries:
  - shinyapps.io, el servicio de hosting **gratuito** de RStudio
  - Shiny Server, otro hosting gratuito orientado a usuarios más avanzados (sólo para Linux)
  - RStudio Connect, servicio pago que ofrece mayor seguridad y atención personalizada

Subir mi Shiny App a la web

- Si estamos dando nuestros primeros pasos con Shiny, la opción más recomendable es sin dudas **shinyapps**
- Para utilizar este servicio, es necesario instalar el paquete *rsconnect*, crear una cuenta personal en la web de shinyapps.io y conectarla con RStudio (ver tutorial)
- Usando el botón de **Publish Application**, podremos lanzar páginas web desde nuestra propia PC para que cualquier persona con conexión a Internet pueda visualizar la shiny app que armamos con R



Subir mi Shiny App a la web

- Las aplicaciones que construimos a lo largo de esta clase están disponibles online en los siguientes links:
  - diego.shinyapps.io/taller1
  - diego.shinyapps.io/taller2
  - diego.shinyapps.io/taller3
- Aplicación para visualizar la ciudad de origen de los estudiantes de la UNR:
  - diego.shinyapps.io/GeoRefUNR

Shiny Apps interesantes

- Nube de palabras de obras de Shakespeare
- App que genera reportes para descargar
- App con gráficos interactivos
- App protegida por contraseña que genera reportes para una compañía
- Juego de la Memoria con stickers de R
- App para visualizar tweets más relevantes en una conferencia
- Concurso de las mejores shiny apps

Fin del Taller

MUCHAS GRACIAS!!!