

心理統計の授業中に
GUIのwebアプリを作って遊ぼう

Shiny 入門③

日本心理学会 第85回大会
チュートリアル・ワークショップ
企画者 豊田秀樹・馬景昊
講師 豊田秀樹・馬景昊・堀田晃大



本チュートリアルの流れ

1. Shinyの基本構造
2. 簡単な例の実装
(階級値の数を変更できるヒストグラム)
3. タグ関数 (文章の表示)
4. レイアウト関数 (ボタンなどの操作可能なウィジェット)
5. 関数Reactive (快適な動作のために)
6. デプロイメント (作成したアプリの公開方法)

- 先はShinyが提供されたHTMLのタグを紹介した
- 引き続きはいくつかのUIパーツ（ウィジェット）を紹介する

• ページの全体像

基本的なウィジェット

アクションボタン

ここを押す

アクションボタン今の値は 0

実行ボタン

Submit 実行

日付の範囲入力

2021-07-09

to

2021-07-09

ラジオボタン

- ☒ 1. 回帰分析
- ☐ 2. 因子分析
- ☐ 3. 実験計画

論理値の指定

☐ 決意する!!

入力ファイルの指定

参照

ファイルが選択されていません

選択ボックス

1. 回帰分析 ▼

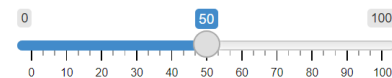
無制限複数選択

- ☒ 1.戸締り
- ☐ 2.消灯
- ☐ 3.火の始末

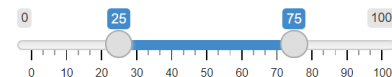
ヘルプ文章

注: ヘルプテキストは、正確にはウィジェットではない。ただし他のウィジェットに付随するテキストを追加する 簡単な方法を提供する。

値が1つのスライダー



値が2つのスライダー



日付の入力

2021-01-01

数値を入力してください

1

テキスト入力

ここに文章を入力して下さい。

- ページのLayoutはsidebarLayout()の代わりにfluidRowを使用している。 fluidRowの中にcolumnを使い、複数の列を作ることができる
- fluidRowとcolumnを組み合わせて、格子構造のページを作ることができる

- Layout はsidebarLayoutやfluidRow以外にも提供されている、詳しくは公式ドキュメント参照
- <https://shiny.rstudio.com/articles/layout-guide.html>

actionButton & submitButton

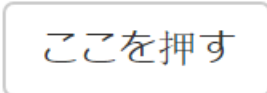
actionButton, アクションボタン : 最初に値が 0 であるアクションボタン、クリックするたびに 1 ずつ増加する。整数が戻り値となる。

■ width 入力幅。'400px' や '100%' 等。

```
h3("アクションボタン"),  
actionButton("action", label = "ここを押す"),  
br(),  
mainPanel(p("アクションボタン今の値は",  
            textOutput("action_num", inline=T))),
```

textoutputに関しては後ほど紹介する

アクションボタン



アクションボタン今の値は 0

submitButton, 送信ボタン : アプリの送信ボタンを作成する。送信ボタンを含むアプリは、入力に変更されたときに出力を自動的に更新せず、ユーザーが明示的に [送信] ボタンをクリックするまで待機する。戻り値は「送信」。

■ width 入力幅。'400px' や '100%' 等。

```
h3("実行ボタン"),  
submitButton("Submit", text = "実行")
```

実行ボタン



- `actionButton`について、 `submitButton`のようにページの変化のタイミングを制御するなどのことができる、時間関係で紹介を割愛する、詳細は公式ドキュメントを参照できる
- <https://shiny.rstudio.com/articles/action-buttons.html>

checkboxInput & checkboxGroupInput

checkboxInput, チェックボックス : 論理値を指定するためのチェックボックスを作成する。戻り値は T または F。

■ value=F 初期値。

■ width 入力幅。'400px' や '100%' 等。

```
column(3,  
  h3("論理値の指定"),  
  checkboxInput("checkbox", label = "決意する!!", value = F)),
```

checkboxGroupInput, チェックボックス/グループ : 複数の選択肢を個別に切り替えるために使用できるチェックボックスのグループを作成する。戻り値は、選択した値の文字ベクトル。

■ choices チェックボックスを表示する値のリスト。リストの要素に名前が付いている場合、値ではなくその名前がユーザーに表示される。この引数を指定する場合は、値は文字列にする必要がある。

■ inline T の場合は選択項目を水平にレンダリングする。規定値は F。

■ width 入力幅。'400px' や '100%' 等。

■ selected 最初に選択する必要がある値 (存在する場合)。

```
column(3,  
  checkboxGroupInput("checkGroup",  
    label = h3("無制限複数選択"),  
    choices = list("1.戸締り" = 1,  
      "2.消灯" = 2, "3.火の始末" = 3),  
    selected = 1)),
```

論理値の指定

☐ 決意する!!

無制限複数選択

- ☒ 1.戸締り
- ☐ 2.消灯
- ☐ 3.火の始末

dateInput

dateInput, 日付選択用カレンダー：クリックすると、ユーザーが日付を選択するためにクリックできるカレンダーを表示するテキスト入力を作成する。

- **value** 開始日。既定値は現在の日付。
- **min** 許可される日付の最小値。Date オブジェクトまたは文字列。yyyy-mm-dd
- **max** 許可される日付の最大値。Date オブジェクトまたは文字列。yyyy-mm-dd
- **format** ブラウザーに表示する日付の形式。デフォルトは"yyyy-mm-dd"
- **startview** 入力オブジェクトが最初にクリックされたときに表示される日付の範囲。「month」(規定値)、「year」、または「decade」にすることができる。
- **weekstart** 週の始まりはどの曜日かを指定する。0 (日曜日, 規定値) から 6 (土曜日) までの整数である必要がある。
- **language** 月名と曜日名に使用する言語。

```
column(3,  
  dateInput("date",  
    label = h3("日付の入力"),  
    language = "ja",  
    value = paste(substr(Sys.Date(), 1, 4), "-01-01", sep="")))
```

日付の入力

2021-01-01|

« 2021年01月 »

| | | | | | | |
|----|----|----|----|----|----|----|
| 日 | 月 | 火 | 水 | 木 | 金 | 土 |
| 27 | 28 | 29 | 30 | 31 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 1 | 2 | 3 | 4 | 5 | 6 |

dateRangeInput & fileInput

dateRangeInput, 日付の範囲選択用のカレンダー : 日付の範囲入力を表示する。

- start 最初の開始日
- end 最初の終了日
- min 最後の開始日
- max 最後の終了日

format, startview, weekstart, language は dateInput() と同じ。

```
column(3,  
  dateRangeInput("dates", label = h3("日付の範囲入力"))),
```

fileInput, 入力ファイルの指定 : アップロードするファイルを指定する。

- accept = NULL, サーバーが予期するファイルの種類に関するヒントをブラウザーに与える”固有のファイル型指定子”の文字ベクトル。
- width = NULL, 入力の幅
- buttonLabel = "Browse...", ボタンで使用されるラベル。
- placeholder = "No file selected" ファイルがアップロードされる前に表示するテキスト。

```
column(3,  
  fileInput("file", label = h3("入力ファイルの指定"),  
    accept = ".csv", # csvの入力を促す。  
    buttonLabel = "参照",  
    placeholder = "ファイルが選択されていません"  
  ),
```

日付の範囲入力

| | | |
|------------|----|------------|
| 2021-07-09 | to | 2021-07-09 |
|------------|----|------------|

入力ファイルの指定

| | |
|----|----------------|
| 参照 | ファイルが選択されていません |
|----|----------------|

helpText & numericInput

helpText, インพุットフォームに追加できるヘルプテキスト

```
column(3,  
  h3("ヘルプ文章"),  
  helpText("注: ヘルプテキストは、正確にはウィジェットではない。",  
    "ただし他のウィジェットに付随するテキストを追加する",  
    "簡単な方法を提供する。")),
```

numericInput, 数値入力用フィールド : 数値を入力する。

- min 最小値
- max 最大値
- width 入力の幅。'400px' や'100%' 等。
- step 増分（必要ならば）

```
column(3, |  
  numericInput("num",  
    label = h3("数値を入力してください"),  
    value = 1, width="50%"))
```

ヘルプ文章

注: ヘルプテキストは、正確にはウィジェットではない。ただし他のウィジェットに付随するテキストを追加する 簡単な方法を提供する。

数値を入力してください

radioButtons & selectInput

radioButtons, ラジオボタン : 多肢選択のラジオボタンを作る。

- choices チェックボックスを表示する値のリスト。リストの要素に名前が付いている場合、値ではなくその名前がユーザーに表示される。この引数を指定する場合は、値は文字列にする必要がある。
- selected 最初に選択される値。指定しない場合は、既定での最初の項目。
- inline T の場合は選択項目を水平にレンダリングする。規定値は F。
- width 入力の幅。'400px' や '100%' 等。

```
column(3,  
  radioButtons("radio", label = h3("ラジオボタン"),  
    choices = list("1. 回帰分析" = 1, "2. 因子分析" = 2,  
      "3. 実験計画" = 3), selected = 1)),
```

selectInput, セレクトボックス : 多肢選択のセレクトボックスを作る。

- 引数はラジオボタンと同じ。

```
column(3,  
  selectInput("select", label = h3("選択ボックス"),  
    choices = list("1. 回帰分析" = 1, "2. 因子分析" = 2,  
      "3. 実験計画" = 3), selected = 1)),
```

ラジオボタン

- ☒ 1. 回帰分析
- ☐ 2. 因子分析
- ☐ 3. 実験計画

選択ボックス

1. 回帰分析 ▲

1. 回帰分析

2. 因子分析

3. 実験計画

sliderInput & textInput

sliderInput, スライダーバー : スライダーバーを作る

- min 表示するスライダーの最小値
- max 表示するスライダーの最大値
- step 増分
- width 入力幅。'400px' や '100%' 等。
- value 長さ 1 の数値ベクトルは、通常のスライダーを作成する。長さ 2 の数値ベクトルは、ダブルエンドレンジスライダーを作成する。min と max の間に値が収まらない場合は警告が表示される。

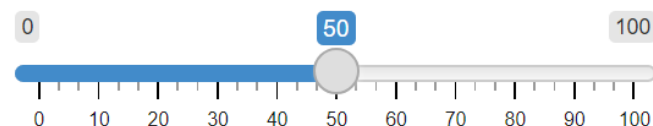
```
column(3,  
  sliderInput("slider1", label = h3("値が1つのスライダー"),  
    min = 0, max = 100, value = 50),  
  sliderInput("slider2", label = h3("値が2つのスライダー"),  
    min = 0, max = 100, value = c(25, 75))  
)
```

textInput, テキスト入力用フィールド : テキスト入力する。

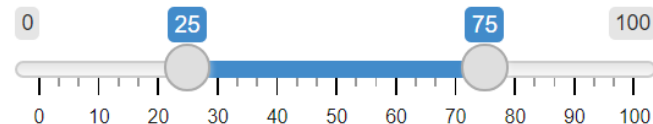
- value 規定値の文章。

```
column(3,  
  textInput("text", label = h3("テキスト入力"),  
    value = "ここに文章を入力して下さい。"))
```

値が1つのスライダー



値が2つのスライダー



テキスト入力

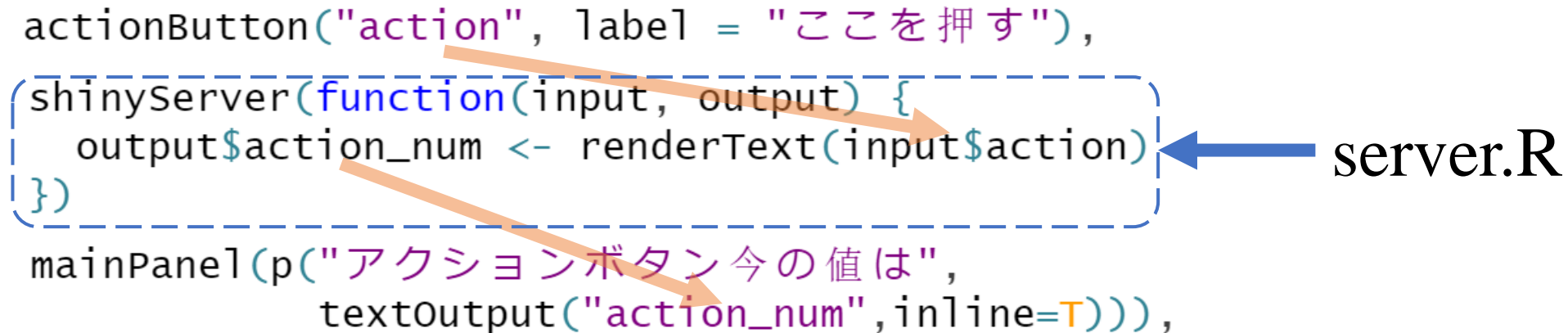
ここに文章を入力して下さい。

textoutput

- 先程はrenderPlotとplotOutputを使用して、図を表示した、図以外に、他の対象も表示することができる、先の
- `mainPanel(p("アクションボタン今の値は",textOutput("action_num",inline=T)))`
- はテキストを表示する

```
actionButton("action", label = "ここを押す"),  
(shinyServer(function(input, output) {  
  output$action_num <- renderText(input$action)  
}))  
mainPanel(p("アクションボタン今の値は",  
            textOutput("action_num",inline=T))),
```

server.R



| render***関数 | 対象 | ***Output 関数 |
|--------------------|-------------|--------------------|
| renderPlot | プロット | plotOutput |
| renderText | テキスト | textOutput |
| renderVerbatimText | テキスト | verbatimTextOutput |
| renderImage | 画像 | imageOutput |
| renderTable | 表 | tableOutput |
| renderUI | ユーザインターフェース | uiOutput |
| renderHtml | HTML | htmlOutput |

- 以上を踏まえて、データを読み、変数のヒストグラムを生成するAppを作成してみる

ui.R

```
library(shiny)
library(ggplot2)
shinyUI(fluidPage(
  titlePanel("ファイルと変数を指定してヒストグラムを描く"),
  sidebarLayout(
    # サイドパネルの描画
    sidebarPanel(
      # ファイル名と拡張子を指定してデータを入力
      fileInput("file1", h6("拡張子が csv か datのファイルを選択"),
        accept = c(".csv", ".dat"),
        buttonLabel = "参照",
        placeholder = "ファイルは未選択"
      ),
      # ファイルのコーディング選択、文字化け対応
      selectInput("select", label = h3("ファイルのコーディング"),
        choices = list("shift-jis" = "shift-jis", "UTF-8" = "UTF-8"),
        selected = "shift-jis"),
      # ヘッダーがあるか否かの論理値の入力
      checkboxInput("header", "ヘッダーがあればチェック", TRUE),
      # セパレータを多肢選択で選ぶ
      radioButtons("sep", "区切り文字",
        choices = c(カンマ = ",",
                    セミコロン = ";",
                    タブ = "\t"),
        selected = ","),
      # データセットを示すときに全部示すか、ヘッドだけかの選択
      radioButtons("disp", "データセットの閲覧",
```

ファイルと変数を指定してヒストグラムを描く

拡張子が csv か datのファイルを選択

参照 ファイルは未選択

ファイルのコーディング

shift-jis ▼

☒ ヘッダーがあればチェック

区切り文字

☒ カンマ

☐ セミコロン

☐ タブ

データセットの閲覧

☒ 最初

☐ 全部

変数を選んでください:

▼

server.R

```
shinyServer(function(input, output) {  
  df <- reactive({  
    return(read.csv(input$file1$datapath,  
                    header=input$header,  
                    sep=input$sep,  
                    fileEncoding = input$Encoding))  
  })  
}
```

拡張子が csv か datのファイルを選択

参照

ファイルは未選択

ファイルのコーディング

shift-jis

☒ ヘッダーがあればチェック

区切り文字

☒ カンマ

☐ セミコロン

☐ タブ

ui.R

```
fileInput("file1", h6("拡張子が csv か datのファイルを選択"),  
          accept = c(".csv", ".dat"),  
          buttonLabel = "参照",  
          placeholder = "ファイルは未選択"),  
checkboxInput("header", "ヘッダーがあればチェック", TRUE),  
radioButtons("sep", "区切り文字",  
             choices = c(カンマ = ",",  
                         セミコロン = ";",  
                         タブ = "\t"),  
             selected = ","),  
selectInput("Encoding", label = h3("ファイルのコーディング"),  
            choices = list("shift-jis" = "shift-jis",  
                           "UTF-8" = "UTF-8"),  
            selected = "shift-jis"),
```

reactiveについて

- 一般的には、ウィジェットの変数が変わるたびに、`render`関数の物がすべて再計算される（表示をリアルタイム更新のため）
- しかし、変数の変化に影響されない計算もある、例えば今回、ヒストグラムに関連する変数を変える際、データを再度読み込む必要がない（`MCMC`とか、場合によって計算がかなり時間がかかる）
- そのため、`reactive({ . . . })`関数が用意されている

- reactiveの中にある変数が変更される際に、 reactiveの中の計算を再度行われるが、 逆に、 reactiveの中にない変数が変更しても、 reactiveの中の計算が再計算されない

server.R

```
df <- reactive({  
  return(read.csv(input$file1$datapath,  
                  header=input$header,  
                  sep=input$sep,  
                  fileEncoding = input$Encoding))  
})  
output$contents <- renderTable({  
  req(input$file1)  
  if(input$disp == "head") { return(head(df()))  
  }else { return(df()) }  
})
```

ui.R

```
checkboxInput("header", "ヘッダーがあればチェック", TRUE),
```

headerの有無が変更する際、データが再度読み込まれる

データセットを示すときに全部示すか、ヘッドだけかの選択

```
radioButtons("disp", "データセットの閲覧",  
             choices = c(最初 = "head",  
                          全部 = "all"),  
             selected = "head"),
```

dispの値が変わっても、データが再度読み込まれない

server.R

```
output$contents <- renderTable({  
  req(input$file1)  
  if(input$disp == "head") { return(head(df()))  
}else { return(df())  
})
```

ui.R

req:対象が存在するか否かの
チェック、対象が存在しない場
合、分析を中止する

```
# データセットを示すときに全部示すか、ヘッドだけかの選択  
radioButtons("disp", "データセットの閲覧",  
  choices = c(最初 = "head",  
              全部 = "all"),  
  selected = "head"),
```

データセットの閲覧

- ☒ 最初
- ☐ 全部

server.R

```
output$html.select <- renderUI({  
  if(is.character(input$file1$datapath)) {  
    # データセットが指定される後の変数選択ボックス  
    return(varSelectInput("variable", "変数を選んでください:",  
                          data=df()))  
  } else {  
    # データセットが指定される前の表示  
    return(varSelectInput("variable", "変数を選んでください:",  
                          data=""))  
  }  
})
```

varSelectInput, 変数を選択 : データ フレームから変数を選択する

- data データ フレーム。列名を選択肢として取得するために使用する
- selected = NULL 初期値
- selectize = TRUE セレクトボックス「Selectize.js」を使うか否か
- width 入力の幅。'400px' や '100%' 等。
- size 選択ボックスに表示する項目の数。数値を大きくすると、高いボックスになる。

ui.R

```
# ヒストグラムを描く変数を選択する  
uiOutput("html.select")
```

変数を選んでください:

変数を選んでください:

Petal.Length

X

Sepal.Length

Sepal.Width

Petal.Length

Petal.Width

if(is.character(input\$file1\$datapath))
ファイルが存在すれば

server.R

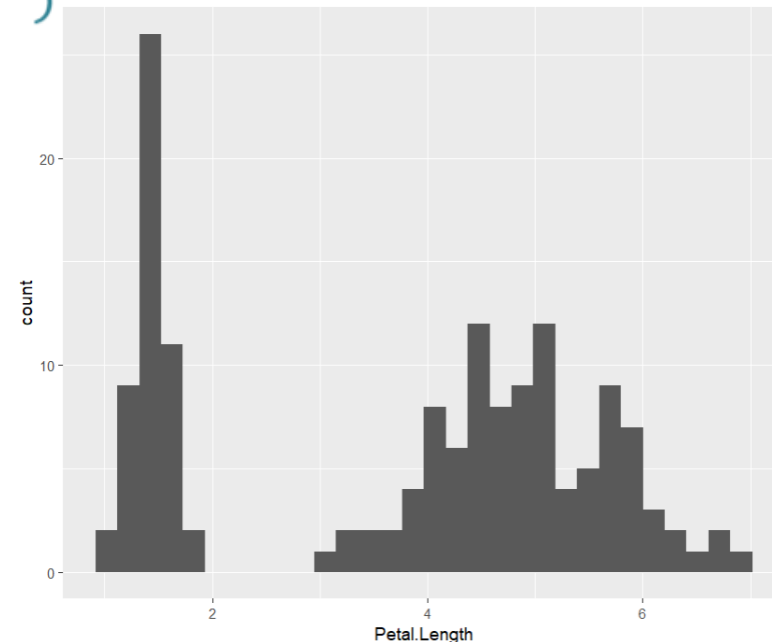
```
output$contents <- renderTable({
  req(input$file1)
  if(input$disp == "head") { return(head(df()))
} else {
  return(df())
}})

output$hist_plot <- renderPlot({
  if(is.character(input$file1$datapath)) {
    # 変数が指定された後のヒストグラム
    ggplot(df(), aes(!!input$variable)) + geom_histogram()
  }
})
```

| X | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.10 | 3.50 | 1.40 | 0.20 | setosa |
| 2 | 4.90 | 3.00 | 1.40 | 0.20 | setosa |
| 3 | 4.70 | 3.20 | 1.30 | 0.20 | setosa |
| 4 | 4.60 | 3.10 | 1.50 | 0.20 | setosa |
| 5 | 5.00 | 3.60 | 1.40 | 0.20 | setosa |
| 6 | 5.40 | 3.90 | 1.70 | 0.40 | setosa |

ui.R

```
# メインパネルの描画
mainPanel(
  # データフレームの描画
  tableOutput("contents"),
  # ヒストグラムの描画
  plotOutput("hist_plot")
)
```



デプロイメント

- Appを作り、次はどのようにユーザへ提供するのが問題である。
- Shinyは大きく2つのデプロイメント方法がある

ソースコード提供

- メリット：提供が簡単（かも）
- デメリット：RとShinyの環境が必要

サーバーを利用

- メリット：RとShinyの環境が不必要、ブラウザさえあれば利用可能
- デメリット：サーバが必要（無料サービスあり）、デプロイメントも少し手間がかかる

ソースコード提供

- UBSなどの直接引き渡すや、Google driveを利用して提供する方法もあるが、githubなどを利用して、Appのzipファイルを提供すると、ShinyのrunUrl()関数を利用することもできる

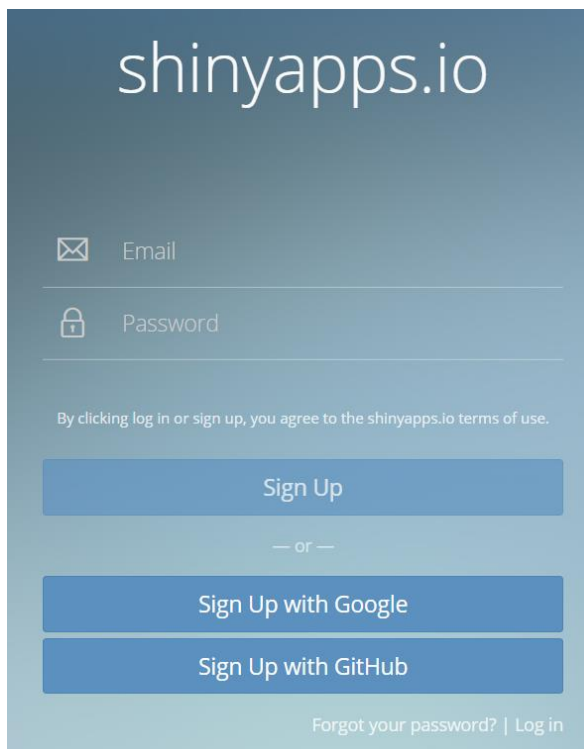
```
shiny::runUrl("https://github.com/tyosem/shiny\_example/raw/main/study05.zip")
```

サーバーを利用

- R studio社は、www.shinyapps.ioというサイトで、Shiny Appをデプロイメントするサービスを提供されている、無料のプランは制約が多いが、一応使える


| FREE | STARTER | BASIC | STANDARD | PROFESSIONAL |
|--|--|---|---|--|
| \$0 /month | \$9 /month (or \$100/year) | \$39 /month (or \$440/year) | \$99 /month (or \$1,100/year) | \$299 /month (or \$3,300/year) |
| New to Shiny? Deploy your applications for FREE. | More applications. More active hours! | Take your users to the next level! | Password protection? Authenticate your users! | Professional has it all! Personalize your domains. |
| 5 Applications | 25 Applications | Unlimited Applications | Unlimited Applications | Unlimited Applications |
| 25 Active Hours | 100 Active Hours | 500 Active Hours | 2,000 Active Hours | 10,000 Active Hours |
| ✓ Community Support | ✓ Premium Email Support | ✓ Performance Boost | ✓ Authentication | ✓ Authentication |
| ✓ RStudio Branding | | ✓ Premium Email Support | ✓ Performance Boost | ✓ Account Sharing |
| | | | ✓ Premium Email Support | ✓ Performance Boost |
| | | | | ✓ Custom Domains |
| | | | | ✓ Premium Email Support |

- ここで、簡単に無料プランを利用して、**APP**をデプロイメント方法を紹介する。まずはユーザー登録



The image shows a sign-up form for shinyapps.io. At the top, the text 'shinyapps.io' is displayed in a large, white, sans-serif font against a dark blue background. Below this, there are two input fields: 'Email' with an envelope icon and 'Password' with a lock icon. Both fields have light blue borders. Under the password field, a small line of text reads: 'By clicking log in or sign up, you agree to the shinyapps.io terms of use.' Below this text is a large blue button labeled 'Sign Up'. Underneath the button is a separator consisting of a horizontal line with the text '— or —' in the center. Below the separator are two more blue buttons: 'Sign Up with Google' and 'Sign Up with GitHub'. At the bottom right of the form, there is a link that says 'Forgot your password? | Log in'.

- できれば、APPアクセスためのURLを設定する

 ACCOUNT SETUP

Let's get started

You'll need an account before you can deploy any applications. Account names can contain letters, numbers and hyphens, but can't start with a hyphen or a number, and can't end with a hyphen. Pick an account name below to proceed.

https://

tyosem

.shinyapps.io

Save

- すると、この画面に進むはず、そして指示通りすれば

shinyapps.io

Help Account: [redacted]

Dashboard

Applications >

Account >

GETTING STARTED

Hi! You must be new here...

Thanks for trying out shinyapps.io! You'll need to install the `rsconnect` R package to get started. The `rsconnect` package enables you to deploy and manage your Shiny applications directly from your R console. To get started, fire up your favorite IDE, and follow the directions below.

STEP 1 - INSTALL RSCONNECT

The `rsconnect` package can be installed directly from CRAN. To make sure you have the latest version run following code in your R console:

```
install.packages('rsconnect')
```

STEP 2 - AUTHORIZE ACCOUNT

The `rsconnect` package must be authorized to your account using a token and secret. To do this, click the copy button below and we'll copy the whole command you need to your clipboard. Just paste it into your console to authorize your account. Once you've entered the command successfully in R, that computer is now authorized to deploy applications to your shinyapps.io account.

```
rsconnect::setAccountInfo(name=[redacted]  
  token=[redacted]  
  secret='<SECRET>')
```

Show secret

Copy to clipboard

In the future, you can manage your tokens from the [Tokens](#) page the settings menu.

- `install.packages("rsconnect")`して、

STEP 2 – AUTHORIZE ACCOUNT

The `rsconnect` package must be authorized to your account using a token and secret. To do this, click the copy button below and we'll copy the whole command you need to your clipboard. Just paste it into your console to authorize your account. Once you've entered the command successfully in R, that computer is now authorized to deploy applications to your shinyapps.io account.

```
rsconnect::setAccountInfo(name='[REDACTED]',  
                           token=[REDACTED],  
                           secret='<SECRET>')
```

Show secret

 Copy to clipboard

In the future, you can manage your tokens from the [Tokens](#) page the settings menu.

- Show secretを押して、secretが表示される

- 最後はAppをアップロードすれば完成

```
library(rsconnect)  
rsconnect::deployApp('._shiny/study05')
```

Appのパス

ファイルと変数を指定してヒストグラムを描く

<https://toyosem.shinyapps.io/study05/>

拡張子が csv か dat のファイルを選択

参照 iris.csv

Upload complete

ファイルのコーディング

shift-jis

☒ ヘッダーがあればチェック

区切り文字

☒ カンマ

☐ セミicolon

☐ タブ

データセットの閲覧

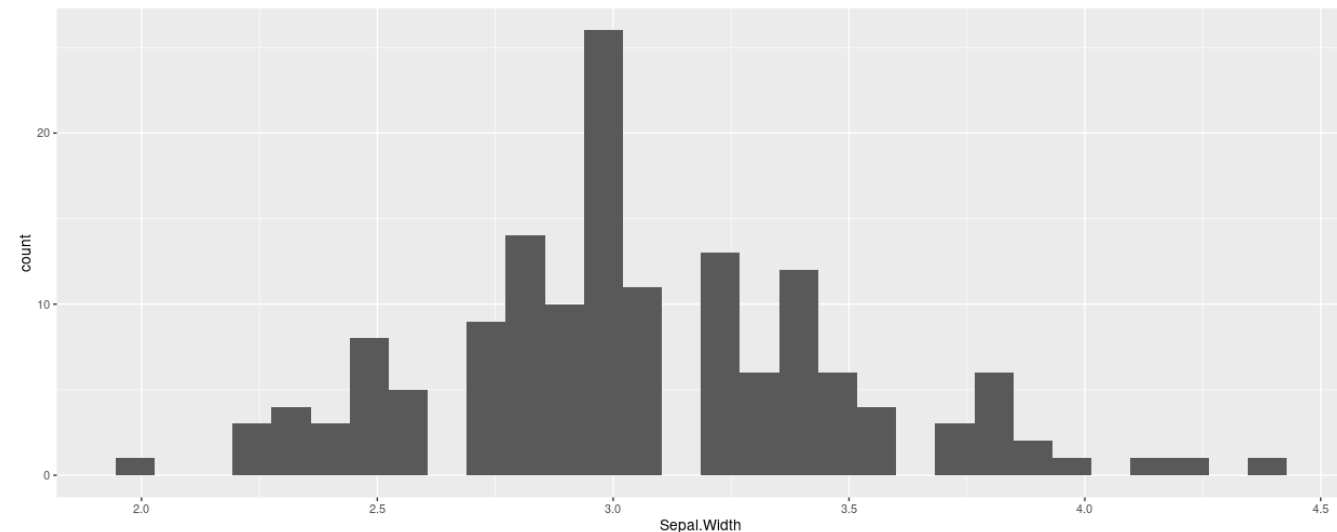
☒ 最初

☐ 全部

変数を選んでください:

Sepal.Width

| X | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.10 | 3.50 | 1.40 | 0.20 | setosa |
| 2 | 4.90 | 3.00 | 1.40 | 0.20 | setosa |
| 3 | 4.70 | 3.20 | 1.30 | 0.20 | setosa |
| 4 | 4.60 | 3.10 | 1.50 | 0.20 | setosa |
| 5 | 5.00 | 3.60 | 1.40 | 0.20 | setosa |
| 6 | 5.40 | 3.90 | 1.70 | 0.40 | setosa |



- もちろん、www.shinyapps.ioを使わずに、自分でサーバーやAWS(Amazon Web Services)などクラウドでデプロイメントした方に対して、R studio社はオープンソースのLinux(Ubuntu 16.04+など)用のShiny Serverを提供されている、こちらを利用して、自分のサーバーでAPPをデプロイメントすることも可能。詳細は公式ページに参照
- <https://www.rstudio.com/products/shiny/shiny-server/>

参考文献

- 梅津 雄一, 中野 貴広 (2018), R と Shiny で作る Web アプリケーション, C&R研究所