

応用プログラミング3
**第15回 RからPythonを使う
/ Shinyによるアプリ開発**

専修大学ネットワーク情報学部
田中健太

1. 課題の振り返り

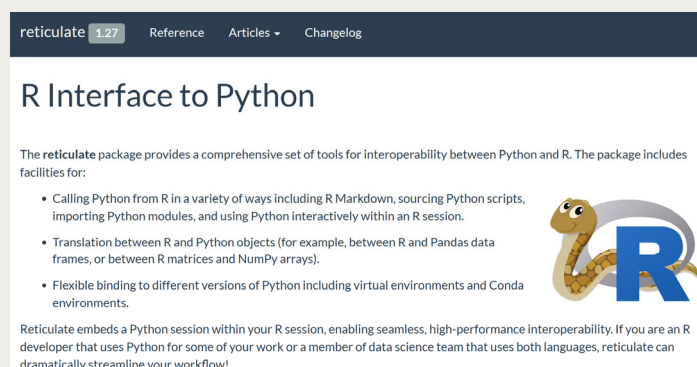
2. RとPython

2

はじめに、RとPythonを組み合わせるための仕組みについて紹介します。

2.1 reticulateパッケージ

- 基本はRだとしても、どうしてもPythonを使いたい場面がある
- ドキュメントはR Markdownで書きたいが、プログラムはPythonで書きたい、Pythonにしかないライブラリを使いたいなど
- **reticulateはRとPythonを連携するためのパッケージ**



3

本講義では、Rでたいいの（ビジネスや研究で求められる）分析ができることを紹介してきました。しかし、近年の動向としてPythonのシェアが大きいため、今後新しい分析手法やそれを実装したライブラリがPythonにのみ提供されることが考えられます。もちろん、それぞれの言語を使い分ければよいのですが、「どうしてもRでやりたい」という場合、RからPythonを呼び出して利用できます。それを実現するのが、**reticulateパッケージ**です。

- Interface to Python • reticulate <https://rstudio.github.io/reticulate/>

reticulateパッケージは、RプログラムからPythonのライブラリや組み込み関数を呼び出して利用したり、R MarkdownにおいてPythonプログラムを記述したチャンクを実行するための機能を提供します。

なお、今回は触れませんが、逆にPythonからRを呼び出すことができる**rpy2**というライブラリもあります。

- 参考: rpy2 · PyPI <https://pypi.org/project/rpy2/>

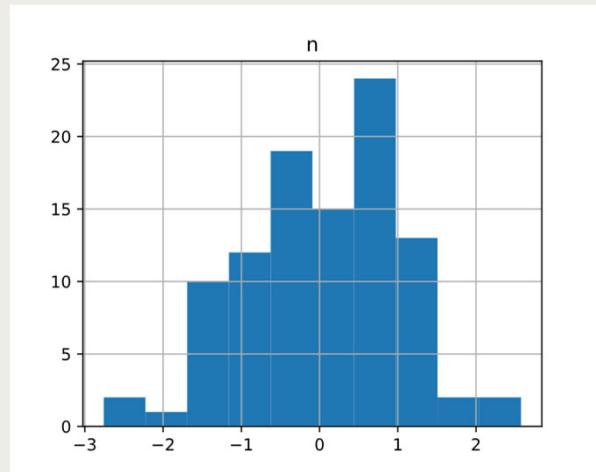
2.2 R MarkdownでPythonを使う

- R Markdownの**チャンク**で `python` と宣言すると、コードがPythonで処理される
- PythonとRの間でオブジェクトのやり取りもできる

```
```{r eval=TRUE, message=FALSE, warning=FALSE, engine="python"}
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

n = np.random.randn(100)
df = pd.DataFrame({'n': n})
df.hist();
plt.show()
```
```



4

はじめに、R MarkdownでPythonを実行するための機能を紹介します。R Markdownをテーマにした第3回の講義でも触れましたが、R MarkdownではR以外にもさまざまな言語のプログラムを実行できます。

```
```{エンジン名 オプション1=値,オプション2=値, ...}
```

プログラム

```
```
```

または

```
```{ オプション1=値,オプション2=値, engine="エンジン名", ...}
```

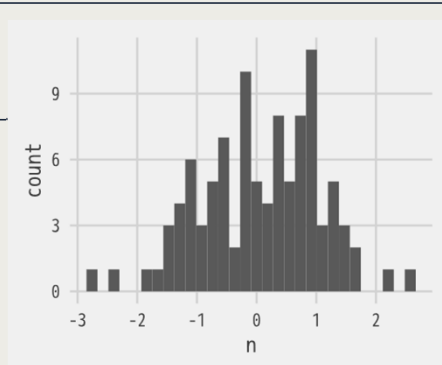
プログラム

```
```
```

とします。この時、**エンジン名**に `python` と指定すると、内部でreticulateパッケージが呼び出され、プログラムをPythonで実行し、結果が出力されます。matplotlibなどによるグラフィックスも、Rの場合と同様に出力されます。なお、Jupyter Labなどでは、Pandasの `plot()` メソッドなどを実行するだけで画面にグラフが出力されますが、Rから呼び出す場合は、`plt.show()` と明示する必要があるようです。

2.2 R MarkdownでPythonを使う

```
df <- py$df  
ggplot(df, aes(x = n)) + geom_histogram()
```



```
library(flextable)  
flextable(head(df, 5))
```

| n |
|---------|
| 0.9269 |
| -0.2400 |
| -0.5867 |
| -0.1093 |
| 0.8351 |

5

R Markdownに限らず、reticulateパッケージを介して実行したPythonプログラムで作成されたオブジェクトは、py というRオブジェクトに自動的に格納されます。例えば、前のページで作成したPandasデータフレーム df は、py\$df としてアクセスできます。また、この際Pythonのオブジェクト (pandas.DataFrame) と等価なRオブジェクト (data.frame) に変換されます。オブジェクト間の対応関係や注意点はドキュメントに記載されています。

- "Object Conversion" https://rstudio.github.io/reticulate/articles/calling_python.html#object-conversion
- "Type conversions" <https://rstudio.github.io/reticulate/index.html#type-conversions>

もちろん、Pythonのすべてのオブジェクトが変換されるわけではないため、Rにオブジェクトを渡して利用するには、reticulateが変換可能な形式にしておく必要があります。

2.3 Rプログラム中でPythonを使う

- reticulateパッケージの各種関数でRとPythonを組み合わせ利用できる
- `import()` 関数でPythonのライブラリをインポートできる
- インポートしたライブラリの関数や作成したオブジェクトのメソッドは `オブジェクト名$関数・メソッド名()` で利用できる
- 組み込み関数は `import_builtins()` で読み込む

| <pre>np <- import("numpy", convert = FALSE) pd <- import("pandas", convert = FALSE) dic <- dict(id = as.character(1:10), value = sample(1:100, 10)) df <- pd\$DataFrame(dic) res <- df\$describe() py_to_r(res) %>% flextable()</pre> | <table><thead><tr><th>value</th></tr></thead><tbody><tr><td>10.00</td></tr><tr><td>56.00</td></tr><tr><td>22.36</td></tr><tr><td>17.00</td></tr><tr><td>42.25</td></tr><tr><td>55.00</td></tr><tr><td>75.50</td></tr><tr><td>87.00</td></tr></tbody></table> | value | 10.00 | 56.00 | 22.36 | 17.00 | 42.25 | 55.00 | 75.50 | 87.00 |
|---|--|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| value | | | | | | | | | | |
| 10.00 | | | | | | | | | | |
| 56.00 | | | | | | | | | | |
| 22.36 | | | | | | | | | | |
| 17.00 | | | | | | | | | | |
| 42.25 | | | | | | | | | | |
| 55.00 | | | | | | | | | | |
| 75.50 | | | | | | | | | | |
| 87.00 | | | | | | | | | | |

6

次に、Rプログラム (*.R) の中でPythonを呼び出して利用する方法を紹介します。PythonのライブラリをRで使いたい場合、`import()` 関数で読み込むことができます。ライブラリの一部 (モジュール) を読み込む場合も、Pythonプログラムと同様 `ライブラリ名.モジュール名` とすればよいです。ただし、`from sklearn.linear_model import LogisticRegression` といったコードに対応するような、クラスの読み込みはできないようです。

読み込んだライブラリが提供する関数は、`オブジェクト名$関数名()` として実行できます。関数を用いて作成したPythonオブジェクトのメソッドも、同様に `オブジェクト名$メソッド名()` として実行できます。Pythonの組み込み関数を使いたい場合は、`import_builtins()` 関数が利用できます。

なお、`import()` 関数ではデフォルトで `convert = TRUE` とオプションが指定されており、これはPythonの処理結果のオブジェクトを自動的にRのオブジェクトに変換します。しかし、例えばPandasでデータフレームを作成し、それをScikit-Learnで前処理、モデリングするといった時には、Pythonオブジェクトのまま扱いたいです。この場合、`convert = FALSE` とオプションを指定すると変換が行われません。

また、明示的にPythonオブジェクトをRオブジェクトに変換する、逆にRオブジェクトをPythonオブジェクトに変換する場合には、それぞれ `py_to_r()` と `r_to_py()` 関数を使います。

2.3 Rプログラム中でPythonを使う

- Pythonスクリプトを `py_run_file()` 関数で実行できる

```
# pyscript_example.py
import numpy as np
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.linear_model import LogisticRegression
wine = load_wine()
lr = LogisticRegression(penalty="elasticnet", solver="saga", l1_ratio=0.5)
model = lr.fit(wine.data, wine.target)
```

```
py_run_file("./pyscript_example.py")
py$model$coef_ %>%
  as_tibble(.name_repair = "minimal") %>%
  setNames(., py$wine$feature_names) %>%
  flextable()
```

| alcohol | malic_acid | ash | alkalinity_of_ash | magnesium | total_phenols | flavonoids | nonflavonoid_phenols | proanthocyanins | color_intensity | hue | od280/od315_of_diluted_wines | proline |
|-----------|------------|------------|-------------------|-----------|---------------|------------|----------------------|-----------------|-----------------|------------|------------------------------|-----------|
| -0.006530 | -0.001873 | -0.0011758 | -0.016340 | -0.040006 | -0.0002074 | 0.0008597 | -0.00034976 | -0.0002386 | -0.003066 | -0.0003225 | -0.0002108 | 0.005893 |
| 0.004543 | -0.001255 | 0.0007153 | 0.010298 | 0.030600 | 0.0017145 | 0.0024086 | 0.00009426 | 0.0013653 | -0.005568 | 0.0010906 | 0.0028093 | -0.004500 |
| 0.001952 | 0.003151 | 0.0004252 | 0.006006 | 0.009291 | -0.0014719 | -0.0033035 | 0.00022026 | -0.0010915 | 0.008669 | -0.0007329 | -0.0025634 | -0.001358 |

7

Pythonプログラムをファイルにまとめたスクリプトも、Rから実行できます。ここでは、Pythonでロジスティック回帰を行うプログラムを、`pyscript_example.py` というファイルに記述しています。これを実行するには、`py_run_file()` 関数を使います。結果は `py` オブジェクトに代入されているので、Rで利用できます。こちら、必要に応じて `convert = FALSE` オプションを指定します。

- Run Python code — `py_run` • reticulate
https://rstudio.github.io/reticulate/reference/py_run.html

3. 研究結果の公開・ 成果の共有

8

ここからは、がらっと話が変わり、データ分析の結果をどのように公開、共有するかということについて述べます。

3.1 論文・レポートからWebへ

- 2010年代後半から、研究成果は論文だけでなく、WebサイトやGitHubリポジトリとして公表することが増えてきた
- 成果をいち早く共有し、世界中の研究者、エンジニアと議論することでよりよいものを生み出す「オープンサイエンス」が加速している

オープンサイエンスとは、ICTを活用して科学を変容させることであり、現在研究データを含めた研究成果をインターネットの上で広く共有する科学の進め方に注目が集まっている。こうした新しい科学の在り方は、科学者の自発的な取り組みとして展開されるのが

日本学術会議，2020，「提言：オープンサイエンスの深化と推進に向けて」

9

データ分析の結果は、自分だけが知っていればよいものではなく、誰かに共有する必要があります。研究の文脈では、世界中の研究者に、自分の成果を論文などで発信します。ビジネスにおいても、基本的に誰かの指示でデータ分析をするので、指示をした上司なり先輩なり顧客に結果を報告する必要があります。

報告の方法として、古くから論文やレポートを作成し、紙に印刷してきました。しかし近年では、PDFをはじめとした電子ファイルで配布したり、広く成果を公表したい（社内であっても）場合、レポートをHTMLで作成し、Webサイト（ページ）として公開することが増えています。（大学生の皆さんにとっては、もはやそれが当たり前かもしれませんが）

また、分析の再現性（第3回で触れた "Reploducible Research"）のため、使用したデータやプログラムを、GitHubリポジトリなどとして公開することも多くなりました。

- 参考: 提言「オープンサイエンスの深化と推進に向けて」 | 日本学術会議
<https://www.scj.go.jp/ja/info/kohyo/kohyo-24-t291-1-abstract.html>

そのような、インターネットを介した研究成果の公開にもRやR Markdownは適しています。

3.2 非専門家への情報提供

- 分析結果や知見を非専門家にもわかりやすく伝えるために、**Webベースのダッシュボード**が使われる
- ビジネスにおいても、経営層などの意思決定権者の判断を支援するために、ダッシュボードが作成される



総務省統計局 "統計ダッシュボード"
<https://dashboard.e-stat.go.jp/>



jig.jp 福野泰介氏, "新型コロナウイルス対策ダッシュボード"
<https://www.stopcovid19.jp/>

10

また、近年では分析結果をできるだけ早く、可能ならばリアルタイムに提供することも求められています。特にビジネスでは、分析結果をもとに経営判断、意思決定を行うため、「エラい人がすぐにぱっと見てわかる」かたちでの情報提供が必要です。

研究においても、例えば昨今の感染症に関する最新の知見などは、研究者だけでなく多くの一般人(非専門家)が知りたいものです。そのため、研究成果をわかりやすく、リアルタイムに公表できる仕組みが求められます(そのようにすることで、非専門家が専門的な内容を勘違いして捉えたり、悪意のある他者が都合の良い部分だけを切り取ってデマなどに悪用することも増えてきました)。

そこで、分析結果をわかりやすく可視化し、Webの仕組みを通じて情報提供する**ダッシュボード**が作成されています。分析作業を可能な限り自動化し、結果をWebアプリとして表示する仕組みを構築し、社内や外部に向けて公開します。

今回の講義の後半では、ダッシュボードを容易に開発できるRパッケージを紹介します。

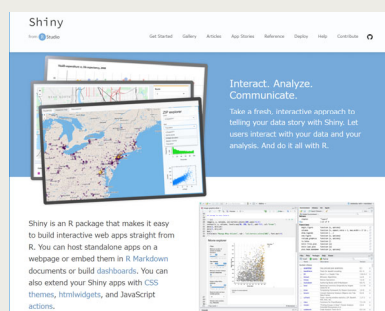
4. Shinyによる Webアプリ開発

11

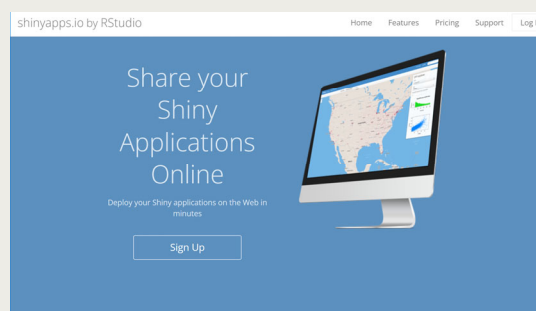
ここから、RでWebアプリを開発するための、Shinyパッケージについて紹介します。

4.1 Shinyパッケージ

- Rによる分析結果を、そのままWebアプリとして可視化できるフレームワーク
- ggplot2をはじめとした強力な可視化機能や機械学習機能を組み込んだアプリを開発できる
- アプリはRを導入した独自サーバーやshinyapps.ioでホスティングできる



<https://shiny.rstudio.com/>



<https://www.shinyapps.io/>

12

第2回の講義でも近年のトレンドとして触れましたが、ShinyはRでWebアプリを開発するためのフレームワークです。Rの強力なデータ分析機能を組み込んだWebアプリを、Rの「作法」で作成できます。アプリエンジニアではないデータ分析者が、他の言語を学習せずとも、分析結果を共有するダッシュボードを開発できるため、利用事例が増えてきています。Posit社のページで、さまざまなShinyアプリが紹介されています。

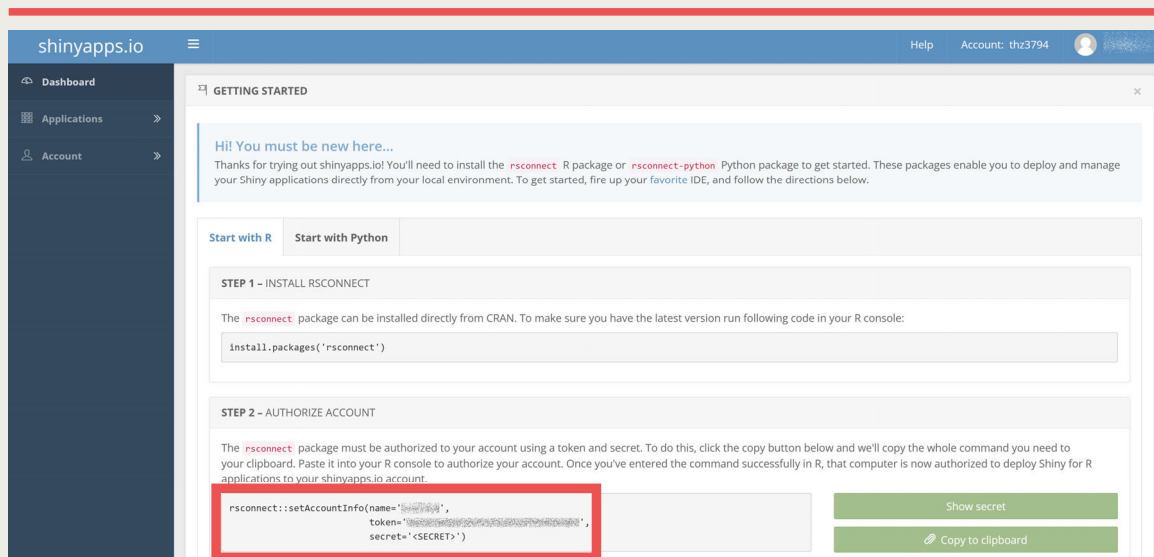
- 参考1: Shiny - Gallery <https://shiny.rstudio.com/gallery/>

Shinyアプリは、当然Rがインストールされたコンピューターでしか動きません。そのため、自前の環境でアプリを動かす場合、一般的なWebサーバーの設定に加え、Rや必要なパッケージをインストールし、さらにShiny Serverというアプリケーションサーバーを導入する必要があります。

- 参考2: Shiny Server - Posit <https://posit.co/products/open-source/shinyserver/>

また、Posit社が運営するShinyアプリのホスティングサービス shinyapps.io を利用することもできます。ある程度までは無償で使用でき、Rコンソールからrsconnectパッケージを使い、直接アプリをアップロードすることができます。

4.1 Shinyパッケージ



shinyapps.io

Help Account: thz3794

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 or `rsconnect-python` Python package to get started. These packages enable you to deploy and manage your Shiny applications directly from your local environment. To get started, fire up your favorite IDE, and follow the directions below.

Start with R Start with Python

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. Paste it into your R console to authorize your account. Once you've entered the command successfully in R, that computer is now authorized to deploy Shiny for R applications to your shinyapps.io account.

```
rsconnect::setAccountInfo(name='<token>',  
  token='<token>',  
  secret='<secret>')
```

Show secret

Copy to clipboard

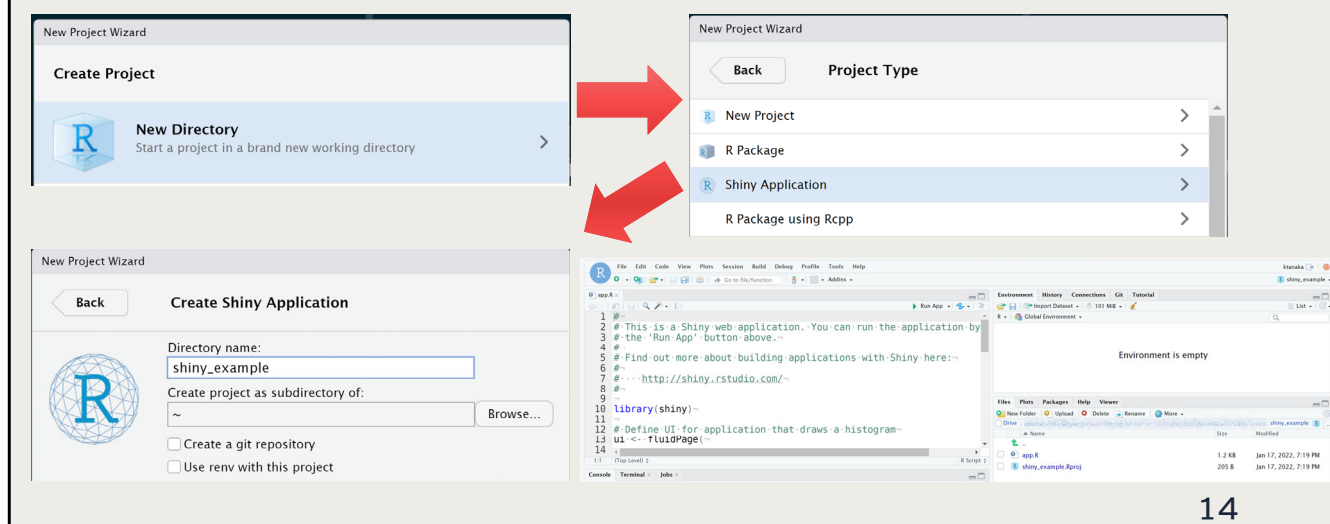
```
library(rsconnect)
```

```
setAccountInfo(name = "アカウント名", token = "トークン", secret = "シーク  
レット") # 初回設定
```

```
deployApp("ディレクトリ")
```

4.2 Shinyアプリの基本構成

- Shinyアプリは、**app.R** ファイルに一般的なデータ分析プログラムと同じように記述する
- 以前は、ui.R と server.R というファイルに分けていた
- RStudioの "New Project" から作成するのが便利

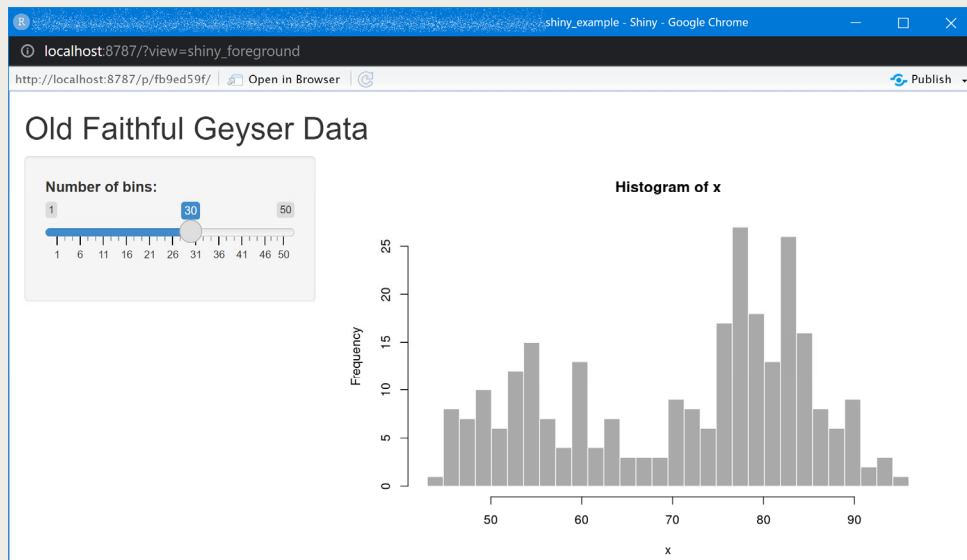


ここから、Shinyアプリの作成方法について、概要を紹介します。Shinyアプリは、**app.R** というファイル名で作成します。この中に、画面表示 (UI) とロジックを記述します。少し前までは、UIを ui.R に、ロジックを server.R に分けて記述するように、公式ドキュメントやさまざまな書籍などで書かれていましたが、現在は1つのファイルにまとめるようです。

RStudioの場合、メニューバーから "File" - "New Project" を選択し、"Shiny application" を選べば、ある程度テンプレートが用意された状態でプロジェクトフォルダーが作成されます。実際のプログラミングは他のエディターでするとしても、最初のセッティングはRStudioで行うと便利でしょう。

4.3 Shinyアプリの実行

- ひとまず、サンプルとして開いた app.R を実行する
- RStudioの "Run App" ボタンや `shiny::runApp("ファイル名")` で実行する



15

まずはテンプレートとして用意されたアプリを実行してみましょう。RStudioでは、エディターペインに "Run App" というボタンがあるので、クリックしてください。すると、ヒストグラムとビンの数を調整するスライダーが表示されます。スライダーを動かすことで、リアルタイムにヒストグラムが再描画されます。

app.R のプログラムは、この講義を通じてある程度Rに慣れた目からは、それほど特殊で難しいものには見えないと思います。前述のように、ui オブジェクトにUI、server() 関数にロジックが記述されています。アプリ自身を実行する、他の言語での `__main__()` 関数にあたるのは、`shinyApp(ui = ui, server = server)` という記述です。このように、Shinyアプリの構造はとてもシンプルなものになっています。

今回は、具体的なShinyアプリのプログラミングについては取り上げませんが、ドキュメントとチュートリアルが充実しており（英語ですが）、学習は比較的容易だと思います。

- 参考1: Shiny - Tutorial <https://shiny.rstudio.com/tutorial/>
- 参考2: Mastering Shiny <https://mastering-shiny.org/index.html>
- 参考3: Getting Started with Shiny <https://ourcodingclub.github.io/tutorials/shiny/>

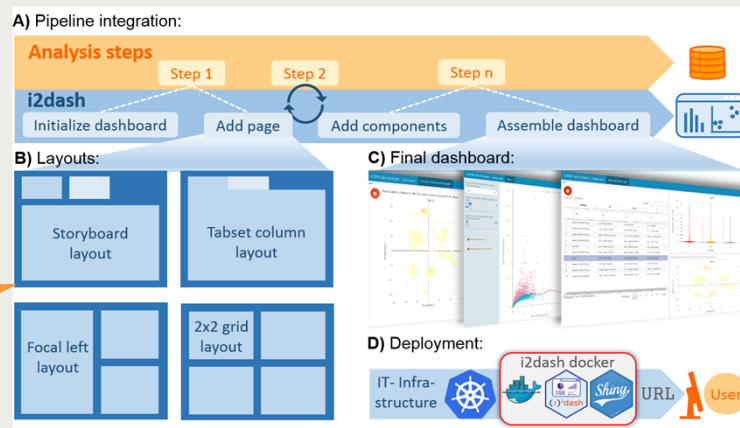
5. i2dashパッケージによる ダッシュボード開発の実践

16

最後に、Shinyをベースにした、より容易にダッシュボードを作成するためのi2dashパッケージを紹介します。

5.1 i2dashパッケージ

- Shinyによるダッシュボードを効率的に開発するフレームワーク
- R MarkdownからShinyアプリを作成できる
- ベースはPosit社によるflexdashboardパッケージ



<https://loosolab.github.io/i2dash/>

17

Shinyを使えば、RだけでWebアプリを開発できます。しかし、レイアウトや入出力のための部品の配置など「何でもできるけどめんどくさい」面もあります。そのため、Shinyの上に被せるかたちで、ダッシュボードを作成することに特化したフレームワークがいくつか存在します。今回は、その中で（筆者が試用した限り）最もシンプルなi2dashパッケージを取り上げます。

- Iterative and Interactive Dashboards • i2dash <https://loosolab.github.io/i2dash/>

i2dashパッケージは、Shiny → flexdashboard → i2dashといった階層構造で作成されています。flexdashboardはPosit社によるパッケージで、ShinyアプリをR Markdownで記述できるようにするものです。i2dashは、flexdashboardの自由度をある程度制約する代わりに、いくつかの関数だけでR Markdownファイルが生成される仕組みです。

- R Markdown Format for Flexible Dashboards • flexdashboard
<https://pkgs.rstudio.com/flexdashboard/>

5.2 i2dashオブジェクト

- ダッシュボードの基本要素が整った i2dashboard オブジェクトが提供されている
- `i2dashboard(title, author, theme, ...)` とダッシュボードの情報を記述する
- `assemble(オブジェクト, "ファイル名.Rmd")` 関数で R Markdown ファイルに出力する
- `rmarkdown::run("ファイル名.Rmd")` で実行する

```
library(i2dash)

dashboard <- i2dashboard(title = "テスト", author = "名前",
  theme = "cosmo", interactive = TRUE, datadir = getwd(), pages = list())
assemble(dashboard, "example.Rmd")
rmarkdown::run("example.Rmd")
```

18

i2dashパッケージは、ダッシュボード全体の情報を保持する i2dashboard オブジェクトを提供します。i2dashboard() 関数の引数に、以下の設定項目を指定します。

- title: ダッシュボードのタイトル
- author: 作者名
- theme: Bootstrapのテーマから cosmo, bootstrap, cerulean, journal, flatly, readable, spacelab, united, lumen, paper, sandstone, simplex, yeti を選択できます (デフォルトは yeti)。
- datadir: データなどを配置するフォルダーのパスを指定します。
- pages: ダッシュボードのページ (タブ) をリストで指定します。後から add_page() 関数で追加できるので、list() と空にしておいてよいです。
- interactive = TRUE / FALSE: TRUE にするとShinyベースのWebアプリ、FALSE にすると静的なHTMLファイルとして出力します。

- The i2dashboard S4 class. — i2dashboard-class • i2dash
<https://loosolab.github.io/i2dash/reference/i2dashboard-class.html>

作成したi2dashboardオブジェクトは、assemble() 関数で R Markdown ファイルに出力できます。ここでは interactive = TRUE としているので、Shinyアプリとして出力されます。作成したダッシュボード (Shinyアプリ) は、rmarkdown::run() 関数で実行できます。

5.3 ページの追加

- `add_page()` 関数でページ（タブ）を追加できる
- `menu` オプションでドロップダウンリストにもできる

```
dashboard %<>% add_page(page = "contents01", title = "ページ1",  
  layout = "storyboard") %>%  
  add_page(page = "contents02", title = "リスト1", layout = "focal_left",  
    menu = "リスト") %>%  
  add_page(page = "contents03", title = "リスト2", layout = "focal_left",  
    menu = "リスト")  
  
assemble(dashboard, "example.Rmd")  
rmarkdown::run("example.Rmd")
```

19

次に、ダッシュボードにページを追加します。i2dashboard/パッケージは、パイプで処理した結果を自身に上書き代入する (?) `%<>%` という演算子を提供します。`add_page()` 関数には、引数として以下のような情報を指定します。

- `page = "ページID"`: 一意のIDを指定します。
- `title`: 上部のメニューバーに表示するタイトルを指定します。
- `layout`: `focal_left`, `storyboard`, `2x2_grid`, `default` (Tabset?) を指定します。
具体的なレイアウトは、5.1節の図を参照してください。
- `menu`: 同じ名前のページは、リストとしてまとめられます。

5.4 コンテンツの追加

- データやグラフィックスをページに追加できる
- `htmlwidgets`型のオブジェクト、`ggplot2`の出力、プレーンテキスト、画像などが利用できる
- `add_component()` 関数で結果のオブジェクトなどをページに紐づける

```
library(DT)

dashboard %<>% add_component(datatable(iris), page = "contents01")

p <- ggplot(iris, aes(x = Sepal.Length, y = Petal.Length, colour = Species)) +
  geom_point()
dashboard %<>% add_component(p, page = "contents01")

assemble(dashboard, "example.Rmd")
rmarkdown::run("example.Rmd")
```

20

各ページに表示するコンテンツは、`add_component()` 関数で追加します。テキストや画像、`plot()` 関数や`ggplot2`の出力などのグラフィックスを引数に与えます。また、`htmlwidgets`型のオブジェクトは、ページ内で操作可能なインタラクティブなコンテンツとして出力できます。`htmlwidgets`型は、DTパッケージの `datatable()` 関数や、`plotly`パッケージの各種関数で作成できます。他にも、地図を描画する `leaflet`パッケージなどがあります。`htmlwidgets`に対応したパッケージは、以下のプロジェクトサイトにまとめられています。

- `htmlwidgets` for R <https://www.htmlwidgets.org/>

`add_component()` 関数には、表示したいコンテンツと、対象となるページの名前を指定します。コンテンツとして、テキストや画像、`htmlwidgets`型のオブジェクトを出力する自作の関数を指定することもできます。

なお、現状ではいちど追加したコンテンツを削除する方法がない (`remove_component()` 的な関数がない) ため、修正はプログラムをはじめてから実行しなおすことになります。

5.4 コンテンツの追加

```
library(leaflet)
```

```
leaflet_map <- leaflet() %>%
```

```
  addTiles() %>%
```

```
  addMarkers(lng = 139.5552138, lat = 35.6088936,  
            popup = "専修大学生田キャンパス2号館")
```

```
dashboard %<>% add_component(leaflet_map, page = "contents02")
```

```
assemble(dashboard, "example.Rmd")
```

```
rmarkdown::run("example.Rmd")
```

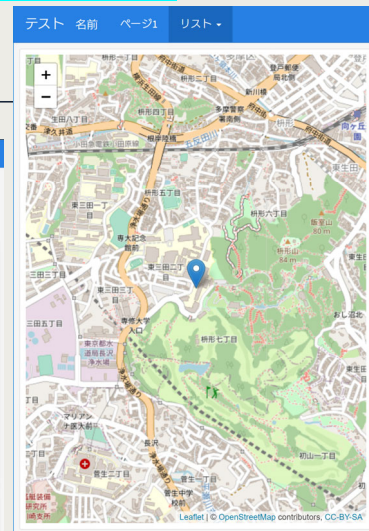
テスト 名前 ページ1 リスト

Show 10 entries

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|----|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 2 | 4.9 | 3 | 1.4 | 0.2 | setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5 | 5 | 3.6 | 1.4 | 0.2 | setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 8 | 5 | 3.4 | 1.5 | 0.2 | setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | setosa |

Showing 1 to 10 of 150 entries

Previous 1 2 3 4 5 15 Next



6. まとめ

6.1 今日の内容

- reticulateパッケージ
- 研究結果の公開・成果の共有
- ShinyによるWebアプリ開発
 - Shinyアプリの基本構成
 - Shinyアプリの開発
 - Shinyアプリの実行
 - Shinyアプリのデプロイ

これで本講義は終了です。半期、受講いただきありがとうございました。