R, RStudioのいろは

FOSS4G北海道 ハンズオンデイ:
モダンな方法で学ぶ、Rによる地理空間情報データの処理

瓜生 真也 @u_ribo

2017年6月30日

概要

ハンズオンで利用するR, RStudioについて、紹介を兼ねたおさらいをします。

Rについて

- オープンソースのプログラミング言語
- パッケージ（ライブラリ）による機能拡張が充実
- (当初は) 学術・研究領域で活用される
  - 書籍がたくさん出ている

Rについて

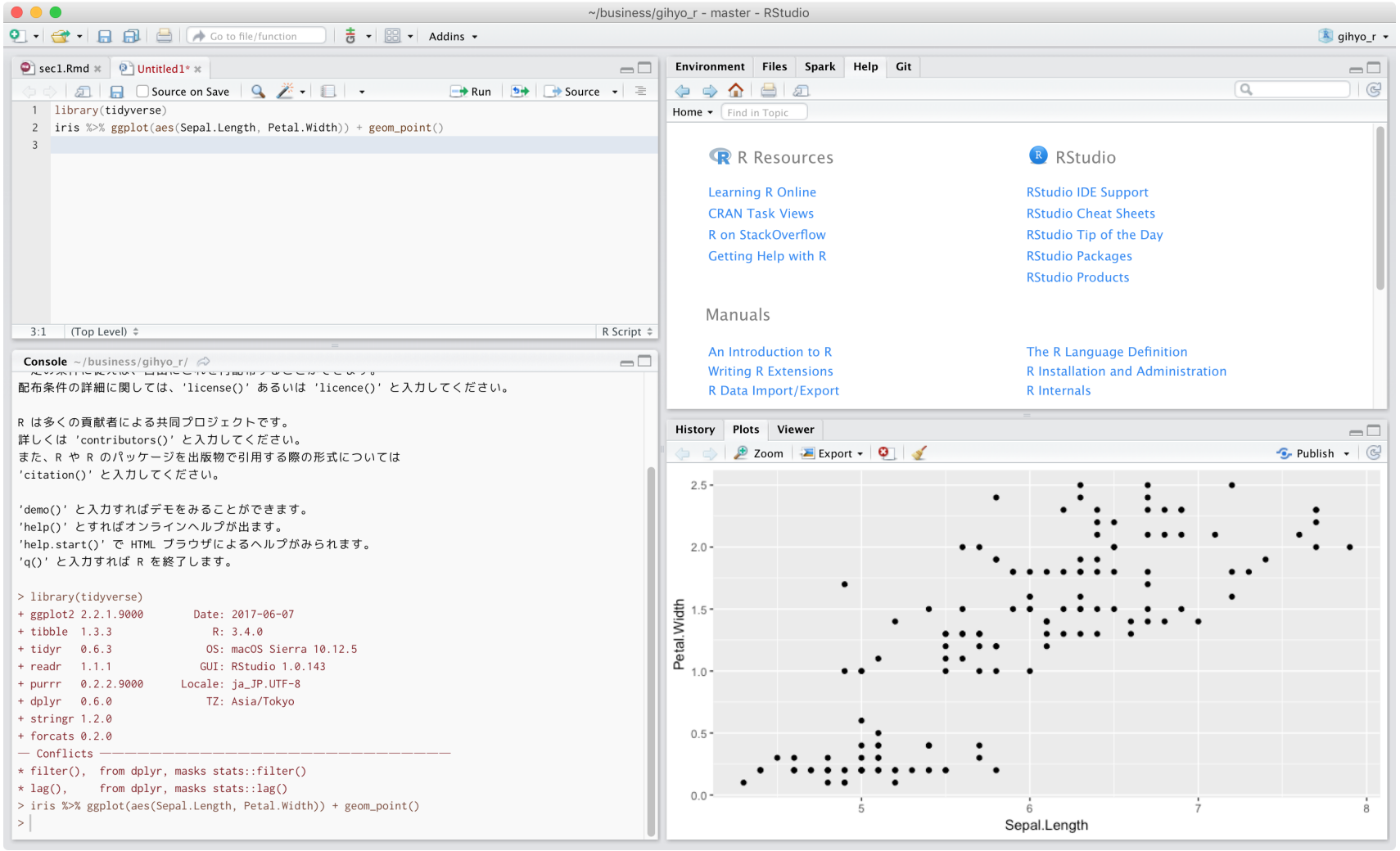
- オープンソースのプログラミング言語
- パッケージ（ライブラリ）による機能拡張が充実
- (当初は) 学術・研究領域で活用される
    - 書籍がたくさん出ている

データ分析を行う人の「道具」

RStudio

- Rの統合開発環境 (IDE)
- Rと同じくマルチプラットフォーム (Windows, Mac, Ubuntuで動く)
- プロジェクト機能をはじめ、Rを実行する上で便利な機能が備わる

RStudioの画面


~/business/gihyo_r - master - RStudio
Addins
gihyo_r
sec1.Rmd
Untitled1*
Source on Save
Run
Source
1 library(tidyverse)
2 iris %>% ggplot(aes(Sepal.Length, Petal.Width)) + geom_point()
3
3:1
(Top Level)
R Script
Console ~/business/gihyo_r/
配布条件の詳細に関しては、'license()' あるいは 'licence()' と入力してください。
R は多くの貢献者による共同プロジェクトです。
詳しくは 'contributors()' と入力してください。
また、R や R のパッケージを出版物で引用する際の形式については
'citation()' と入力してください。
'demo()' と入力すればデモをみることができます。
'help()' とすればオンラインヘルプが出ます。
'help.start()' で HTML ブラウザによるヘルプがみられます。
'q()' と入力すれば R を終了します。
> library(tidyverse)
+ ggplot2 2.2.1.9000 Date: 2017-06-07
+ tibble 1.3.3 R: 3.4.0
+ tidyr 0.6.3 OS: macOS Sierra 10.12.5
+ readr 1.1.1 GUI: RStudio 1.0.143
+ purrr 0.2.2.9000 Locale: ja_JP.UTF-8
+ dplyr 0.6.0 TZ: Asia/Tokyo
+ stringr 1.2.0
+ forcats 0.2.0
── Conflicts ──
* filter(), from dplyr, masks stats::filter()
* lag(), from dplyr, masks stats::lag()
> iris %>% ggplot(aes(Sepal.Length, Petal.Width)) + geom_point()
>
Environment
Files
Spark
Help
Git
Home
Find in Topic
R Resources
Learning R Online
CRAN Task Views
R on StackOverflow
Getting Help with R
RStudio
RStudio IDE Support
RStudio Cheat Sheets
RStudio Tip of the Day
RStudio Packages
RStudio Products
Manuals
An Introduction to R
Writing R Extensions
R Data Import/Export
The R Language Definition
R Installation and Administration
R Internals
History
Plots
Viewer
Zoom
Export
Publish
Petal.Width
Sepal.Length
0.0
0.5
1.0
1.5
2.0
2.5
5
6
7
8

Rproject

- Rを実行する作業環境
  - コードやデータ、プロットした図のデフォルトの保存先
- プロジェクトに応じて切り替えると良い

Rproject

- Rを実行する作業環境
  - コードやデータ、プロットした図のデフォルトの保存先
- プロジェクトに応じて切り替えると良い

新規プロジェクトの作成

メニューバーのFileから...

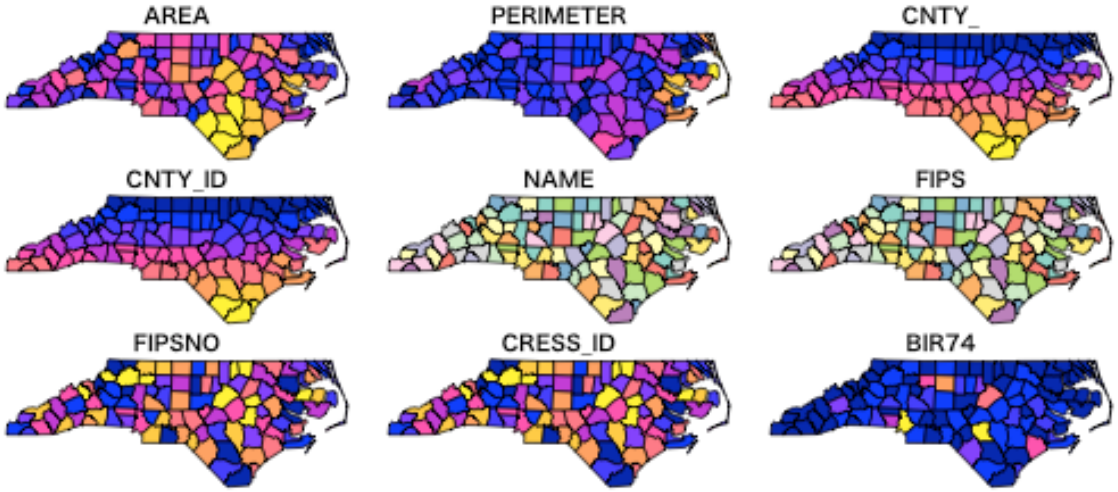
やってみよう

最初の3行のRコード

たったの3行で地図が描ける

```
        (sf)
nc <- read_sf(system.file("shape/nc.shp", package = "sf"))
```

```
plot(nc)
```



超簡単

最初の3行のRコード (おさらい)

1. 　というパッケージを利用可能にする
2. `read_sf()`関数によりshapefileを読み込み、"nc"という名前のオブジェクトに保存する
3. "nc"を`plot()`関数で描画する

```
        (sf)
nc <- read_sf(system.file("shape/nc.shp", package = "sf"))
plot(nc)
```

Rを扱う上で重要なことば

- オブジェクト、クラス
- 関数、引数、演算子
- パッケージ

オブジェクト

Rで操作する文字、値、関数、変数、データ...
Rで扱う「　　」の全て

- 「もの」なのなので名前をつけられる
- <- は代入演算子
  - `res <- 1+1; res`
  - resに結果を保存。resとして結果を呼び出す
  - `res + 3`
- オブジェクトの種類に応じて名前が付いている
  - クラス

オブジェクトのクラス

オブジェクト名で内容が出力される

```
letters # 文字列ベクトル
# [1] "a" "b" "c" "d" "e" "f" "g" ...

iris # データフレーム
#     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
# 1            5.1         3.5          1.4         0.2     setosa
# 2            4.9         3.0          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            5.0         3.6          1.4         0.2     setosa
# ...

help # 関数
# function (topic, package = NULL, lib.loc = NULL, verbose = getOpti
```

オブジェクトがどのクラスに属するかを`class()`を使って確認

データフレーム

エクセルなどの　　をもった表形式のデータ格納方法

```
class(iris)
```

```
# [1] "data.frame"
```

```
class(nc)
```

```
# [1] "sf"         "data.frame"
```

（行・）列には名前がつく

```
names(iris) # オブジェクトに与えられている名前を取得
```

```
# [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
# [5] "Species"
```

データフレーム

```
head(iris, n = 2) # 先頭行を返す関数
```

```
#   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
# 1          5.1         3.5          1.4         0.2  setosa
# 2          4.9         3.0          1.4         0.2  setosa
```

```
dim(iris) # データフレームのサイズ(行数と列数)を確認
```

```
# [1] 150   5
```

```
# [演算子を使ってデータフレームの各値を参照
iris[2, ] # 2行目
```

```
#   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
# 2          4.9           3          1.4         0.2  setosa
```

```
iris[1:4, "Species"] # Species列の1から4番目の値
```

```
# [1] setosa setosa setosa setosa
# Levels: setosa versicolor virginica
```

データフレーム

```
head(iris, n = 2) # 先頭行を返す関数
```

```
#   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
# 1          5.1         3.5          1.4         0.2  setosa
# 2          4.9         3.0          1.4         0.2  setosa
```

```
dim(iris) # データフレームのサイズ(行数と列数)を確認
```

```
# [1] 150   5
```

```
# [演算子を使ってデータフレームの各値を参照
iris[2, ] # 2行目
```

```
#   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
# 2          4.9           3          1.4         0.2  setosa
```

```
iris[1:4, "Species"] # Species列の1から4番目の値
```

```
# [1] setosa setosa setosa setosa
# Levels: setosa versicolor virginica
```

関数・演算子

Rで色々な処理を実行する役割をもつ

- **関数名(引数){処理内容本体}**という形
  - ユーザが触れるのは関数名と引数
  - 引数は、関数を実行する対象や、関数の挙動を制御するための値を指定する
- `1 + 1`を実行するための `+` も関数の一種（演算子）
- `function()`により定義される（自作関数を書ける）

関数・演算子

Rで色々な処理を実行する役割をもつ

- **関数名(引数){処理内容本体}**という形
  - ユーザが触れるのは関数名と引数
  - 引数は、関数を実行する対象や、関数の挙動を制御するための値を指定する
- `1 + 1`を実行するための `+` も関数の一種（演算子）
- `function()`により定義される（自作関数を書ける）

関数を覚えることでRでできることの幅が広がる

関数への理解を深めるには

help(関数名)

- Description
- Arguments
- Usage
- Examples
- ...

パッケージ

トピックごとに関数をまとめて提供

- インストール時から利用可能
- `library()`を使って呼び出す
  - ex) `library(sf)`
  - `package::function()`という形式でも良い
- CRAN (しーらん、くらん) からインストール
  - インストールされていないパッケージを読み込むとエラー。

それでは改めて

最初の3行のRコード

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

```
        (sf)
nc <- read_sf(system.file("shape/nc.shp", package = "sf"))
plot(nc)
```

よく使う関数

- help()... 関数のドキュメントを表示する
- library()... パッケージの読み込み
- class()... オブジェクトのクラスを確認する