

応用プログラミング3
**第2回 Rプログラミングの基本、
最近の動向について**

専修大学ネットワーク情報学部
田中健太

1. よく使われるデータ構造

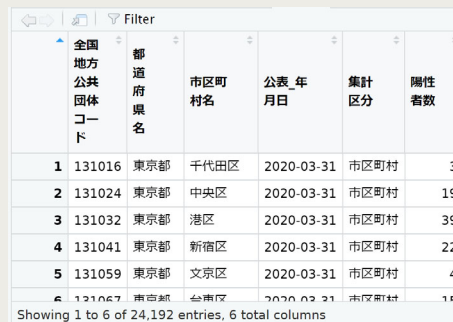
1

前回に引き続き、R言語の基本的な文法を確認しましょう。まず、Rで使われるデータ構造について紹介します。

1.1 データフレーム

- 表構造データは、**データフレーム**として管理される
- ベクトルや行列を **data.frame()** 関数で変換する
- 実際にはファイルから読み込んだデータを格納し、扱うことが多い

```
a <- letters[1:10]
# 1から100の範囲から10個の乱数を抽出 (一様分布)
b <- sample(x = 1:100, size = 10)
df <- data.frame(a, b) # ベクトルaとbをデータフレームの列として結合
df
##   a  b
## 1 a  9
## 2 b 38
## 3 c 86
## 4 d 57
```



	全国 地方 公共 団体 コード	都 道 府 県 名	市区町 村名	公表_年 月日	集計 区分	陽性 者数
1	131016	東京都	千代田区	2020-03-31	市区町村	3
2	131024	東京都	中央区	2020-03-31	市区町村	19
3	131032	東京都	港区	2020-03-31	市区町村	39
4	131041	東京都	新宿区	2020-03-31	市区町村	22
5	131059	東京都	文京区	2020-03-31	市区町村	4
6	131067	東京都	台東区	2020-03-31	市区町村	15

Showing 1 to 6 of 24,192 entries, 6 total columns

2

続いて、データ分析において最も広く使われる構造としてデータフレームを取り上げます。データフレームも行列と同じくn行m列のサイズですが、データフレームでは、列ごとに型を指定できます。数値が入る列、文字列が入る列、といったように列ごとに異なる尺度のデータを格納できます。データフレームは、data.frame() 関数などで作成します。また、このあと紹介するファイルからデータを読み込む read.csv() 関数などを使うと、ファイルの中身がデータフレームとして格納されます。

一般的に、ビジネスにおけるデータ分析の対象となるのはExcelなどで開くことができる、あるいはデータベースに格納された表構造のデータだと思っています。そのようなデータをRで扱うには、データフレームを使います。

1.2 ファイルからの読み込み

- CSVは `read.csv()` 関数、タブ区切り・スペース区切りは `read.table()` 関数で読み込む
- macOSやPosit Cloudの標準文字コードはUTF-8なので、必要に応じて `fileEncoding` オプションで指定する
- 1行目に列名がある場合は `header = TRUE`、ない場合は `header = FALSE` と指定する
- 列の型は自動判定される

3

実際にデータ分析を行う際には、実験や調査で収集したデータがファイルに記録されていて、そのファイルからデータを読み込んでRで処理することが多いでしょう。Rは、標準でテキストファイルを読み込むことができます。コンマ区切りのCSVファイルや、タブ区切りのTSV、スペース区切りのファイルなどがよく用いられます。CSVには `read.csv()` 関数、タブ区切りやスペース区切りには `read.table()` 関数などを使い、内容をデータフレームとして格納します。引数にインターネット上のURLを指定して、ダウンロードして読み込む、ということもできます。

- `read.csv()`, `read.table()` 関数の主要なオプション
 - `header = TRUE / FALSE`: 1行目に列名が含まれているかどうか
 - `sep = "区切り文字"`: 任意の区切り文字を指定する。`read.csv()` では `,`、`read.table()` では、スペース、タブ、改行がデフォルトで指定されている
 - `na.strings = "欠損値を表す文字"`: データファイル中に欠損値がある場合、欠損値を表す文字、記号を指定する。デフォルトは `"NA"` ※ `NA`にも型があるので注意
 - `fileEncoding = "文字コード"`: 入力ファイルの文字コードを指定する。Windowsで作成したファイルは、Shift-JISであることが多いので、`"CP932"` または `"Shift-JIS"` と指定する
 - `skip = スキップする行数`: 先頭から指定した行数を読み込まずスキップする

1.2 ファイルからの読み込み

```
# 疑似日本酒データを読み込む
df <- read.csv("./pseudo_sake_data.csv", fileEncoding = "UTF-8")
tail(df, n = 1) # データの末尾行を出力

##      アルコール分 日本酒度 エキス分 総酸 アミノ酸度 甘辛度 濃淡度 精米歩合
## 998      15.9      3.56  4.14 1.17      1.31 -0.15 -0.84      62
##      容器      タイプ
## 998 茶色瓶 本醸造酒
```

```
# 文字列型の列で値が入っていないセルを欠損値として扱う
df <- read.csv("./small_na_sample.csv", na.strings = "")
df

##      Char Numeric
## 1      A         3
## 2      B        NA
## 3 <NA>         5
```

1.3 データフレームの操作（1）

- 行列と同様、**オブジェクト名[行番号, 列番号]** で参照できる

```
df <- read.csv("./pseudo_sake_data.csv", fileEncoding = "UTF-8")
df[334, ] # データフレームの334行目を表示

##      アルコール分 日本酒度 エキス分  総酸 アミノ酸度 甘辛度 濃淡度 精米歩合
## 334          15.9    10.4    3.99 0.961    0.915 -0.108 -0.977    75
##           容器 タイプ
## 334 紙パック 一般酒
```

```
df[, 4] # データフレームの4列目を表示

## [1] 0.905 1.360 1.320
```

データフレームから任意の行、列を抽出するには、行列の場合と同様、**オブジェクト名[行番号, 列番号]** という指定ができます。この時、行番号または列番号を省略すると、すべての行、すべての列を指定したことになります。

1.3 データフレームの操作 (2)

- オブジェクト名\$列名 や オブジェクト名[["列名"]]
などでも参照できる

```
df[334, 5:6] # 334行目の5列目と6列目を表示  
df$エクス分 # エクス分列を表示  
df$工 # 実は完全一致でなくても参照できてしまう  
df[["エクス分"]] # エクス分列を表示
```

6

また、データフレーム特有の参照方法として、オブジェクト名\$列名 や オブジェクト名[["列名"]] という指定もできます。なお、\$列名 については、完全一致でなくても出力されるので、似た列名が存在する場合には注意が必要です。

1.3 データフレームの操作 (3)

- データフレームから条件を満たす行、列を抽出するには**演算子**を活用する

```
df <- read.csv("./pseudo_sake_data.csv", fileEncoding = "UTF-8")

# 吟醸酒のデータのみ抽出する
df[df[["タイプ"]] == "吟醸酒", ]

##      アルコール分 日本酒度 エキス分 総酸 アミノ酸度 甘辛度 濃淡度 精米歩合
## 648          16.7   -0.154    3.66 1.32    1.490 -0.247 -0.809      12
## 649          15.7    8.360    5.52 1.20    0.862 -0.303 -0.750      57
## 650          16.9    4.890    4.03 1.48    1.210 -0.167 -0.722      48
##
##              容器 タイプ
## 648 リターナブル瓶 吟醸酒
## 649   その他色瓶 吟醸酒
## 650       緑色瓶 吟醸酒
```

7

データフレームからデータを抽出するには、演算子も活用できます。条件を式で指定すると、結果が行ごとに TRUE, FALSE で返り、TRUE となった行だけが出力されます。Rで使用できる演算子は多岐にわたり、自分で定義することすらできますが、一般的に使用されるのは以下のようなものです。

- ==: 左辺と右辺が等しければ TRUE
- !=: 左辺と右辺が等しくなければ TRUE
- >: 左辺が右辺より大きければ TRUE ※文字列を対象とした場合も動作するケースがある
- <: 左辺が右辺より小さければ TRUE
- >=: 左辺が右辺以上であれば TRUE
- <=: 左辺が右辺以下であれば TRUE
- !式: 式が FALSE であれば TRUE (式自体の否定)
- &: 左辺の式と右辺の式が両方とも TRUE ならば TRUE (AND)
- |: 左辺の式と右辺の式のいずれかが TRUE ならば TRUE (OR)

他に、関数として以下の条件判定を行うものがあります。is.*() 関数については、データの型ごとに関数が存在します。

- is.na(): 引数に指定した値が NA ならば TRUE
- is.character(): 引数に指定した値が character 型ならば TRUE

1.3 データフレームの操作 (3)

```
df[df[["アルコール分"]] >= 17, ] # アルコール分が17%以上のデータを抽出する
```

```
##      アルコール分 日本酒度 エキス分  総酸 アミノ酸度 甘辛度 濃淡度 精米歩合
```

```
## 51          17.3    5.54    3.12 1.040    1.59 -0.098 -0.945    73
```

```
...
```

```
##      容器 タイプ
```

```
## 51 茶色瓶 一般酒
```

```
...
```

```
# 複数の条件を評価するには & または | (OR) を使う
```

```
df[df[["タイプ"]] == "吟醸酒" & df[["アルコール分"]] >= 17, ]
```

```
##      アルコール分 日本酒度 エキス分  総酸 アミノ酸度 甘辛度 濃淡度 精米歩合
```

```
## 654          17.0    4.29    4.45 0.852    1.51 -0.558 -0.730    40
```

```
...
```

```
##      容器 タイプ
```

```
## 654 緑色瓶 吟醸酒
```

```
...
```

2. 基本的な集計

9

次に、データを集計する基本的な方法を紹介します。より高度で効率的な方法は、次回以降の講義で紹介しますので、ここではRの組み込み関数による方法を取り上げます。

2.1 データの行数、列数の確認

- データの行数は `nrow()` 関数、列数は `ncol()` 関数で確認できる
- まとめて `dim()` 関数で確認することもできる

```
nrow(df)
## [1] 998
ncol(df)
## [1] 10
dim(df)
## [1] 998 10
```

10

最も基本的なデータの集計は、データの件数を数えることです。表構造のデータについては、行数と列数をカウントします。Rでは、`nrow()` 関数、`ncol()` 関数でそれぞれ行数、列数を取得できます。また、`dim()` (dimension) 関数で、行数と列数をまとめて取得できます。

2.2 質的（カテゴリー）変数の集計

- 性別、年代などのカテゴリー変数について、カテゴリーごとの個数を集計するには `table()` 関数を使う
- 比率を算出したい場合は、`prop.table(table())` 関数を使う

```
df <- read.csv("./pseudo_sake_data.csv", fileEncoding = "UTF-8")
# sort(decreasing = TRUE) 関数で頻度の降順に並べ替え
sort(table(df[["タイプ"]]), decreasing = TRUE)

##
##   一般酒   吟醸酒   純米酒   本醸造酒
##     647     136     125      90

prop.table(sort(table(df[["タイプ"]]), decreasing = TRUE)) * 100

##
##   一般酒   吟醸酒   純米酒   本醸造酒
##  64.830  13.627  12.525   9.018
```

11

次に、データに含まれるカテゴリー変数の個数をカウントします。カテゴリー変数とは、ラベルや質的変数、定性データとも呼ばれ、性別や居住都道府県、年代（10代、20代…）などの離散的な値のことです。Rでは、カテゴリー変数は一般的にcharacter型またはfactor型として扱われます。2020年に公開されたR 4.0からは、`read.*()` 関数で読み込んだテキストファイルでは、文字列はそのままcharacter型として読み込まれます。R 3.xまでは、factor型として読み込まれるので、（古いRが混在するような環境では）違いに注意が必要です。

カテゴリー変数の集計には、`table()` 関数を使います。ベクトルやデータフレームの列などを引数に与えます。すると、カテゴリー変数の水準（ラベル）ごとに個数が出力されます。標準では、水準の文字コード順に出力されますが、頻度順に並べ替えたい時は、`sort()` 関数を組み合わせます。標準は昇順で、`decreasing = TRUE` オプションを指定すると降順になります。

また、データ全体における各水準の比率を得たい場合は、`table()` 関数の実行結果に `prop.table()` 関数を適用すると、全体を1とした時の比率が得られます。

2.3 量的変数の集計

- 量的変数については、まず区間に分割し、個数を集計する
- 区間に分割する単純な方法として `cut()` 関数を使う

```
# 組み込みのairqualityデータセットを使う
data(airquality)

# 1日の平均風速 (mph) を10等分する
# デフォルトでは "始点" より大きく、"終点" 以下
wind_cut1 <- cut(airquality[["Wind"]], breaks = 10)
table(wind_cut1)

## wind_cut1
## (1.68,3.6] (3.6,5.5] (5.5,7.4] (7.4,9.3] (9.3,11.2] (11.2,13.1]
##          4          9         20         37         30         22
## (13.1,15]  (15,16.9] (16.9,18.8] (18.8,20.7]
##          21          7          1          2
```

12

続いて、量的変数（連続データ）の集計について取り上げます。量的変数は、小数点以下も含めて、あらゆる値を取り得るため、数値ごとに個数をカウントすることは無意味です（データ数が少ない場合、幹葉図などで個数をカウントする場合があります）。

量的変数については、まず適切な区間を設定し、個々のデータを区間に割り付けます。そして、その区間に含まれるデータの個数をカウントします。例えば、連続データである体重を、10kg刻みの区間に割り付ける、といったようにです。

区間の設定と割り付けについては、後述する `hist()` 関数でもできますが、ここでは `cut()` 関数を使う例を紹介します。引数に区間に分割したい値を与え、`breaks` オプションに、区間の数（等分する場合）または、区間の境界値を指定します。

2.3 量的変数の集計

```
# 任意の間隔 (2mphごと) に等分する
# seq(始点, 終点, 刻み) で連続したベクトルを生成する
wind_cut2 <- cut(airquality[["Wind"]], breaks = seq(0, 22, 2))
table(wind_cut2)

## wind_cut2
##  (0,2] (2,4] (4,6] (6,8] (8,10] (10,12] (12,14] (14,16] (16,18]
##      1      4     11     38     27     38     10     17      4      ## (18,20]
## (20,22]
##      1      2
```

3. 組み込み関数による グラフィックス

14

ここからは、Rでグラフィックスを作成する方法について紹介します。この講義では、グラフィックスは主に後述するggplot2パッケージを使って作成しますが、Rには組み込みのグラフィックス関数も多数あります。基本的な使い方を理解しておきましょう。

3.1 グラフィックスによるデータの理解

- データの全体的な傾向を捉えるには、グラフィックスを描くとよい
- Rには、科学的に正確なグラフィックスを描く機能が標準で備わっている

統計をグラフにあわせて

統計は、データを集めて集計しただけでは、単なる数字の集まりであり、そこから何が読み取れるか必ずしも明らかではありません。

統計を作成するときは、必ず、「○○について知りたい!」という目的があるはずですから、得られた結果を、その目的に合わせて上手に使うことが重要です。グラフは、結果を視覚的に表す便利な道具であり、グラフをうまく使うことによって、自分の考えていることを相手に的確に伝えることができます。

グラフにはいくつかの種類があり、それぞれ、得手・不得手があります。自分が伝えたい目的に応じて、適切なグラフを使うことにより、説明力もぐっと高まります。ここでは、そういったグラフの種類やそれぞれの用途、注意点について説明します。

棒グラフ 棒の高さで、量の大小を比較する。	折れ線グラフ 量が増えているか減っているか、変化の方向をみる。	円グラフ 全体の中での構成比をみる。
並列棒グラフ 構成比を比較する。	ヒストグラム データの散らばり具合をみる。	レーダーチャート 複数の指標をまとめてみる。
散布図 2種類のデータの相関をみる。	箱ひげ図 データの散らばり具合をみる。	三角グラフ 3つの量からなる構成比をみる。

出典：総務省統計局「統計学習の指導のために 補助教材」
<https://www.stat.go.jp/teacher/graph.html>

15

データ分析をするには、まず分析対象のデータの全体像を把握することが重要です。データ量が少ない場合、全体を上から下まで眺めることもできますが、生のデータから特徴を把握することは困難です。基本統計量を算出することもできますが、やはり数値から全体像をイメージするには、経験が必要です。

そのため、データの全体像を把握するもっとも容易な手段としてグラフィックス (グラフ) を描きます。グラフィックスには、何を表現したいかという目的に応じて、さまざまな種類があります。適切でないグラフィックスを選択すると、逆に特徴がわかりにくくなり、判断を誤ります。また、軸の範囲や単位によっても、グラフィックスの解釈は変化しますので、ソフトウェア任せではなく、分析者の意思で適切な設定を行う必要があります。

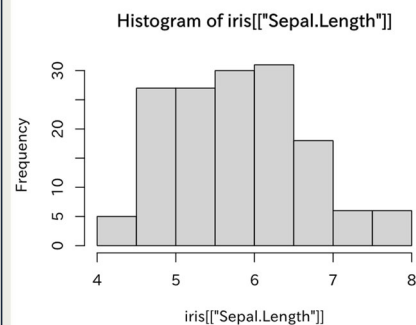
Rには、科学的に正確なグラフィックスを作成するためのさまざまな関数が、標準で用意されています。

3.2 hist() 関数

- `hist(データフレームの列またはベクトル)` とすることで、区間を自動的に決定して集計し、グラフィックスが出力される
- 区間は `breaks オプション` で指定できる
- `オブジェクト名 <- hist(...)` とすることで、区間や度数の情報をオブジェクトに格納し参照できる

```
# 組み込みのirisデータを使用する  
data(iris)
```

```
# Sepal.Length列のデータの分布を  
# ヒストグラムで描画する  
par(family = "IPAexGothic")  
hist(iris[["Sepal.Length"]])
```



16

ヒストグラムは、連続データの分布を表現するためのグラフィックスです。データが全体的にどの範囲に広がっているか、どの辺りに多く観測されているかといった分布形状を一目で把握できます。

Rでは、`hist()` 関数にデータフレームの列やベクトルを与えて作成できます。標準では、区間はRが自動的に決定しますが、`breaks オプション`で任意の区間を指定できます。

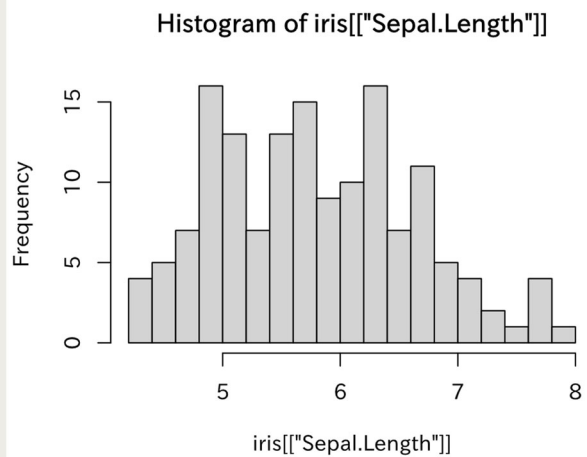
`pretty()` 関数は、引数に与えた数値を、切りの良い間隔で分割してくれるので、`breaks` オプションの引数として活用できます。なお、`breaks` オプションで与える区間には、必ずデータの最小値と最大値が含まれなければなりません。表示範囲の設定は、別途X軸、Y軸のオプションとして指定します。

3.2 hist() 関数

```
# breaksオプションで区間を指定する
# pretty() 関数は数値の範囲を切りの良い間隔で分けてくれる
pretty_breaks <- pretty(range(iris[["Sepal.Length"]]), n = 20)
pretty_breaks

## [1] 4.2 4.4 4.6 4.8 5.0 5.2 5.4 5.6 5.8 6.0 6.2 6.4 6.6 6.8 7.0 7.2 7.4
## [18] 7.6 7.8 8.0

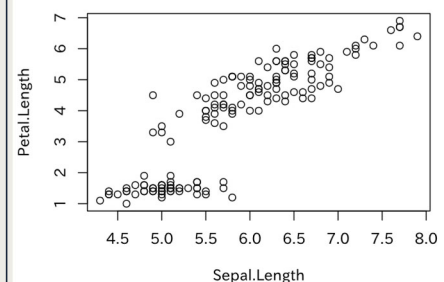
par(family = "IPAexGothic")
hist(iris[["Sepal.Length"]],
     breaks = pretty_breaks)
```



3.3 plot() 関数

- **plot() 関数は、与えるオブジェクトの型によって振る舞いを変える「総称（ジェネリック）関数」**
- 2列からなる行列、データフレームを与えると、**散布図**を描画する
- (plot() だけではないが) cex や col などのオプションで見た目をカスタマイズできる

```
# irisデータのSepal.Length,  
# Petal.Lengthの散布図を描画する  
par(family = "IPAexGothic")  
plot(iris[, c("Sepal.Length", "Petal.Length")])
```



18

次に、ともに連続データである2変数の、分布や関係性を表現する**散布図**を作成するための**plot() 関数**を取り上げます。実際には、plot() 関数は、引数として与えられたデータの性質(型、オブジェクトのクラスなど)により振る舞いを変え、さまざまなグラフィックスを描画する「**総称関数**」と呼ばれるものです。そのため、plot() = 散布図、というわけではありませんので注意してください。また、plot() 関数に限りませんが、Rの組み込みグラフィックス関数では、cex = サイズ や col = 色名 などのオプションで、グラフィックスの見た目をカスタマイズできます。以下に、オプションの一部を列挙します。

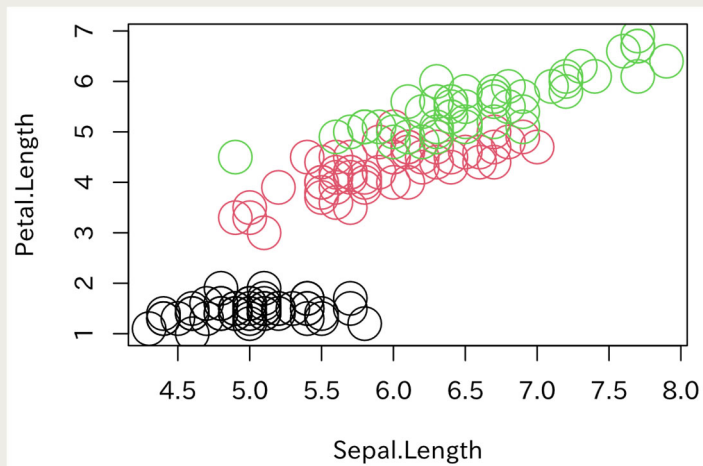
- cex, cex.main, cex.lab, cex.axis: それぞれ対象のデータ点やラベル文字列のサイズを指定する。デフォルトは1。main はタイトル、lab は軸ラベル、axis は軸目盛。数値は、単位のない相対的なもの。
- col: データ点や線の色を指定する。標準で palette() 関数に色と番号の対応リストが8色分格納されており、それらについては、col = 2 といったように番号で指定可能。それ以外は、colors() 関数で定義されている色名や、rgb(Rの値, Gの値, Bの値, 透明度) といったかたちで指定できる。また、学会誌の体裁やユニバーサルデザインの観点で、複数のパレットがパッケージとして提供されているので、それらも活用できる。
- lwd: 線の太さ(幅)を指定する。デフォルトは1。数値は、単位のない相対的なもの。
- main, xlab, ylab: それぞれ、タイトル、X軸ラベル、Y軸ラベルを指定する。
- xlim, ylim: それぞれ、X軸、Y軸の表示範囲を指定する。最小値と最大値からなるベクトルを引数に与える。

3.3 plot() 関数

データ点のサイズを大きくし、Speciesの値で色分けする

```
par(family = "IPAexGothic")
```

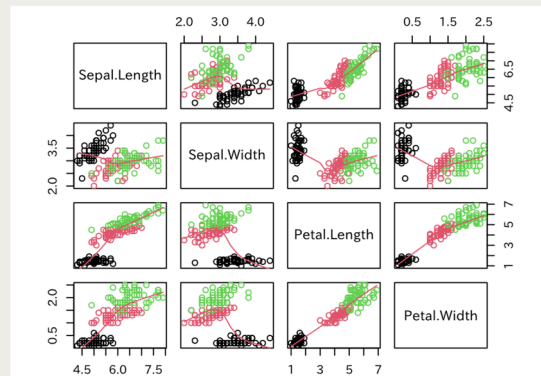
```
plot(iris[, c("Sepal.Length", "Petal.Length")], cex = 3, col =  
iris[["Species"]])
```



3.4 pairs() 関数

- 組み合わせごとの散布図を**ペアプロット**という
- **pairs() 関数**に3列以上のデータを与える

```
# 5列目 (Species) は数値ではないため除外  
# panel.smooth でデータ点の近似曲線を描画する  
par(family = "IPAexGothic")  
pairs(iris[, -5], panel = panel.smooth, col = iris[["Species"]])
```



20

ペアプロットは、複数の列からなるデータフレームに対して、散布図を組み合わせごとに作成したものです。**pairs() 関数**の引数に数値データのみからなるデータフレームを与えます。plot() 関数に複数列を渡しても、ペアプロットは描画できますが、pairs() 関数のほうが、さまざまなカスタマイズができます。

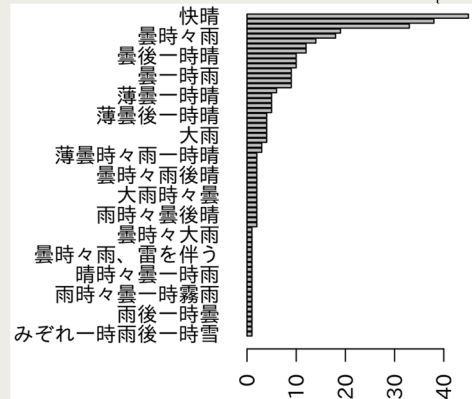
例えば、panel オプションには散布図の上に重ねて描画するさまざまな要素を指定できます。panel.smooth は、データの近似曲線を描画します。

なお、現在では、今後紹介するggplot2パッケージを使ってペアプロットを作成することがほとんどだと思いますので、pairs() 関数のオプションについて、あまり深く調べる必要はないでしょう。

3.5 barplot() 関数

- 棒グラフは `barplot()` 関数で描画する
- `barplot()` 関数はまず `table()` 関数などで集計し、その結果を与える必要がある

```
# 気象庁Webサイトよりダウンロードした2022年の気象データを使用する
df <- read.csv("./tokyo_weather_2022.csv")
# 日中の天気概況を集計し、頻度で並べ替える
tbl <- sort(table(df[["天気概況_昼"]]))
# ラベルがはみ出さないために
# マージンを調整する
par(mar = c(4, 14, 2, 2),
    family = "IPAexGothic")
# horiz = TRUE で横置きに、las = 2 で
# ラベルも横向きに
barplot(tbl, horiz = TRUE, las = 2)
```



21

棒グラフは、質的データの度数や、質的データの水準（ラベル）ごとに集計した連続データの統計量（平均など）を表現するために使います。棒グラフは、`barplot()` 関数を使って作成します。これまでの関数と異なり、生データではなく、`table()` 関数などで作成した集計値を引数に与える必要があります。

グラフィックスは、Rが設定したグラフィックス領域の中に描画されますが、ラベルの文字数やサイズによっては、領域に収まらないことがあります。その場合、グラフィックス領域のサイズや、上下左右の余白との間隔を調整します。`par()` (parameter) 関数は、グラフィックスパラメータを指定する関数です。さまざまな要素を指定できますが、`mar` オプションでは、グラフィックスと余白とのマージンを調整できます。数値は、それぞれ下、左、上、右の余白に対応しています。

4. パッケージによる機能拡張

4.1 パッケージについて

- Rは組み込み関数だけでなく、パッケージを使って機能拡張できる
- 世界中のユーザーがパッケージを無償で公開してくれている

Contributed Packages

Available Packages

Currently, the CRAN package repository features 19925 available packages.

23

ユーザーが作成したRの関数を取りまとめたものが、パッケージです。Rには標準でもさまざまなデータ分析のための関数が用意されていますが、自分で作成することもできます。

そして、世界中のRユーザーが、自分の目的のために作成した関数をせっくくなら使って、と公開しているものがパッケージです。パッケージが提供する関数を使うことで、標準ではできない分析ができたり、標準では作成できないグラフィックスを作成できます。

パッケージは、ユーザーの個人ホームページなどで公開されていることもありますが、大抵の広く使われるパッケージは、CRANに登録されています。登録には審査があり、それをパスしたパッケージが、2万個近く登録されています。

この後、CRANに登録されているパッケージを、手元のRにインストールする方法などを紹介していきます。

4.2 パッケージの読み込み

- パッケージは `library()` 関数で読み込む
- パッケージ名はクォーテーションで囲まない
- Rを起動するたびに実行する必要がある

```
vec <- sample(0:9, 20, replace = TRUE)
vec
nnzero(vec) # エラーになる

## Error in detach("package:Matrix", unload = TRUE): invalid 'name'
argument
## [1] 9 3 3 4 9 5 3 0 1 9 0 6 3 1 3 1 8 3 3 5
## Error in nnzero(vec): could not find function "nnzero"
```

```
library(Matrix) # パッケージを読み込む
nnzero(vec) # ゼロでない値の数を出力する

## [1] 18
```

24

パッケージは、コンピューターにインストールしただけでは、使用できる状態にはなりません。使用したいタイミングでRに読み込む必要があります。パッケージの読み込みは、`library()` 関数で行います。引数に、読み込みたいパッケージの名前を、ダブルクォーテーションで囲まずに指定します。

`library()` 関数で読み込んだパッケージは、「現在起動しているRでのみ」有効です。そのため、Rを終了し、再度起動した際には、都度パッケージを読み込む必要があります。Rの起動時に読み込まれる設定ファイル (`.Rprofile`) に記述しておくという手もありますが、記述したこと、自動で読み込まれることを忘れて、意図しない動作に困惑することもあるので、あまりおすすめしません。

4.3 パッケージのインストール

- 標準添付でないパッケージは、インストールする必要がある
- CRANに登録されているパッケージは、`install.packages()` 関数でインストールできる
- macOS, Linuxではソースコードからのコンパイルが行われるため、開発環境が必要

```
# 統計学の学習のためのTeachingDemosパッケージを
```

```
# インストール
```

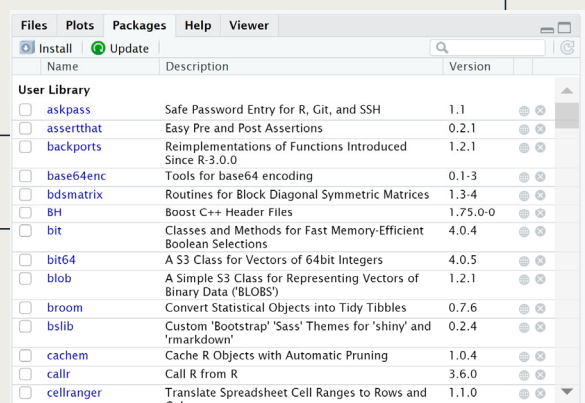
```
install.packages("TeachingDemos")
```

```
library(TeachingDemos)
```

```
# サイコロを振るシミュレーションのための
```

```
# dice() 関数を実行
```

```
dice(ndice = 3, plot.it = TRUE)
```



25

パッケージの中にはRのインストール時に標準で添付されるものも多くありますが、インターネットや書籍で紹介されている便利なパッケージは、多くの場合別途インストールする必要があります。パッケージの多くは、Rユーザーの有志が(まずは自分の仕事を効率化するために) 開発していますが、Rコミュニティの審査を経て、CRANに登録することができます。CRANに登録されたパッケージについては、`install.packages()` 関数でダウンロード・インストールできます。Windowsではあらかじめコンパイルされたバイナリパッケージがインストールされますが、macOSやLinuxでは、ソースコードからのコンパイルが行われるので、Cコンパイラ (gccなど) や関連するライブラリをインストールしておく必要があります。

なお、CRANは世界中にミラーサーバー (リポジトリ) が存在するため、`install.packages()` 関数の実行時に接続するサーバーを指定できます。日本では、東京都立川市の独立行政法人・統計数理研究所がミラーサーバーを提供しているので、"Japan" のサーバーを選択するとよいでしょう。あるいは、"O-Cloud" というクラウドサーバーは、世界中のどこからでも高速に接続できます。

4.4 GitHub上のパッケージ

- 近年は、GitHubでパッケージを公開することも多い
- GitHub上のパッケージは、remotesパッケージまたはdevtoolsパッケージの `install_github()` 関数でインストールする

```
# remotesパッケージはCRANからインストールする
install.packages("remotes")

# 筆者のリポジトリから、「ほぼ何もしない」パッケージをインストールする
remotes::install_github("tetlabo/nothing")

library(nothing)

# ただ、"Hello, World!" と返すだけの関数
hello()

## [1] "Hello, world!"
```

26

CRANでは、審査を経た「安全な」パッケージが公開されていますが、開発者としては、さまざまなドキュメントを作成したり、レビュアーとのやり取りを（英語で）行ったりする負担と、それに伴うスピード感の低下が気になります。そこで、近年では、GitHubにパッケージのソースコードを公開することが増えています。開発したものを即座に公開できる利便性と、（有用性があれば）ユーザーがバグ報告をしてくれたり、共同開発者として名乗り出てくれるといったメリットがあります。

GitHubに公開されているパッケージをインストールするには、remotesパッケージの `install_github()` 関数を使います。引数には、GitHubのリポジトリ名 ("ユーザー名/リポジトリ名") を指定します。なお、GitHubに公開されているのはソースコードなので、OSを問わず、インストール時にはコンパイルが行われます。そのため、WindowsではRToolsというコンパイラなどをまとめたツールをインストールしておく必要があります。

- RTools: <https://cran.r-project.org/bin/windows/Rtools/>

なお、Rに限りませんが、GitHubには、誰でも、不完全でもソースコードを公開できますので、インストールしたパッケージが期待した通りに動作しなかったり、Rの動作が不安定になることもありますので、注意して導入してください。

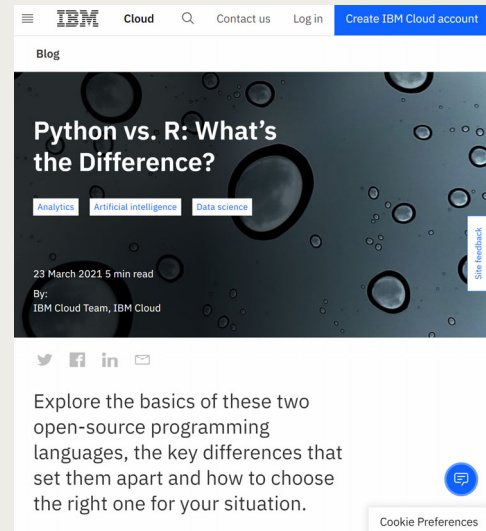
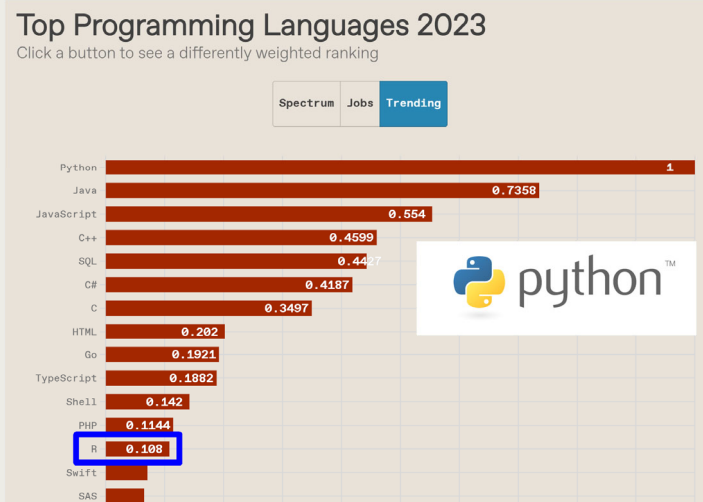
5. 2020年代の動向

27

ここからは、2023年中頃における、Rの現状とトレンドを紹介します。とはいえ、本講義のターゲットは私大文系（ネットワーク情報学部が文系かどうかは議論がありますが）の学生ですので、Rをカジュアルに使用するユーザーの立場から見た動向を取り上げます。

5.1 RかPythonか？

- **ビジネスにおけるニーズは圧倒的にPython**
- 一方でアカデミックではRのシェアも大きい
- **対話的なデータ分析にはRの方が適しているのでは**



28

広く使われているデータ分析ツールとして、(1) Microsoft Excel, (2) SAS, (3) SPSS, (4) R, (5) Pythonが挙げられます。Excelは最も広く使われていますが、できることはデータの集計や簡単な可視化に留まりますので、他のツールとは位置付けが異なります。SASとSPSSは、それぞれ商用製品で、歴史があります。大学の研究室や情報センターなどには導入されていることが多いですが、高額なので民間企業でのシェアはそれほど多くありません。RとPythonは無償で使用できるため、大学、企業を問わず利用が広がっています。

データ分析のためのプログラミング言語としては、古典的な数値計算言語としてFORTRAN、モダンな科学技術計算のための言語としてJuliaなども使われていますが、RとPythonが二大巨頭と言ってよいでしょう。実際には、**ビジネスにおいては、2010年代後半から、Pythonのシェアが圧倒的に大きく、データ分析ツールの主流になっています。**Pythonは、データ分析専門ではない、汎用言語なので、エンジニアがシステムへの組み込み前提で使うなら、便利です。一方で、研究者やビジネスパーソンがちょっとした集計、可視化をしたい時には大げさではないかと思います。

Rは後ほど紹介するように、GUIも使用でき、ビジネスシーンでの対話的なデータ分析には適しているのではないかと考えます。また、RMarkdownという仕組みが非常に強力なので、研究や、**データを基にした意思決定など、分析とレポートがセットになるケースでは、Rのほうが便利であると言えるでしょう。**

5.1 RかPythonか？

Python vs. R: Which is right for you?

Choosing the right language depends on your situation. Here are some things to consider:

- **Do you have programming experience?** Thanks to its easy-to-read syntax, Python has a learning curve that's linear and smooth. It's considered a good language for beginning programmers. With R, novices can be running data analysis tasks within minutes. But the complexity of advanced functionality in R makes it more difficult to develop expertise.
- **What do your colleagues use?** R is a statistical tool used by academics, engineers and scientists without any programming skills. Python is a production-ready language used in a wide range of industry, research and engineering workflows.
- **What problems are you trying to solve?** R programming is better suited for statistical learning, with unmatched libraries for data exploration and experimentation. Python is a better choice for machine learning and large-scale applications, especially for data analysis within web applications.
- **How important are charts and graphs?** R applications are ideal for visualizing your data in beautiful graphics. In contrast, Python applications are easier to integrate in an engineering environment.

Note that many tools, such as Microsoft Machine Learning Server, support both R and Python. That's why most organizations use a combination of both languages, and the R vs. Python debate is all for naught. In fact, you might conduct early-stage data analysis and exploration in R and then switch to Python when it's time to ship some data products.

- プログラミング経験があればPythonの習得は容易
- Rは初心者が手軽に実行できるが高度な処理を実現するにはスキルが求められる

- 一人だけ他と違うソフトを使ってもしょうがないので、同僚（社内、業界）が使っているほうに合わせる

- Rは統計学的分析や探索的データ分析に適している
- Pythonは機械学習や大規模システム開発に向く

- Rは高度なデータ可視化機能を有する
- Pythonはシステムへの分析機能の組み込みに適する

29

ここでは、IBMが公開している技術ブログの記事である "Python vs. R: What's the Difference?" (<https://www.ibm.com/cloud/blog/python-vs-r>) を取り上げます。特にこの記事に権威があったり、新しい視点が含まれるわけではないですが、RとPythonを客観的に比較したわかりやすい記事です。

記事の結論部分をスライドに示していますが、結局は「場合による」ということになります。RとPython、どちらの文法が手になじむか（筆者はもちろんRです）、周囲が何を使っているか、データ分析+レポートだけで完結するのか、分析結果を機能としてITシステムに組み込む計画なのかで、Rが適するケース、Pythonが適するケースは変わってきます。

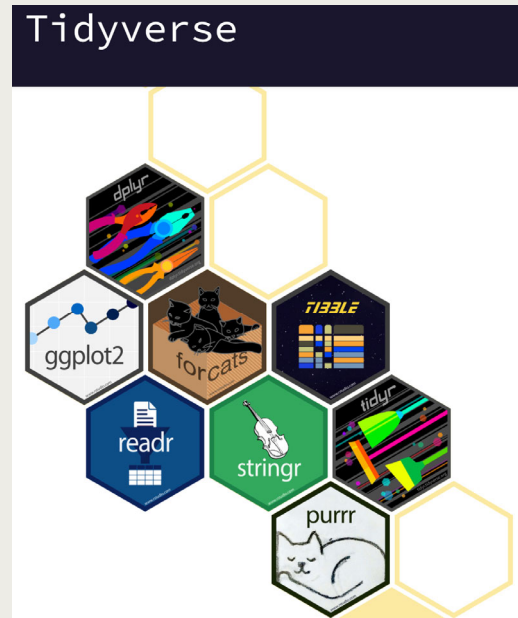
なお、就職活動という観点で見ると、「Rが使えます」よりも「Pythonが使えます」のほうが圧倒的にアピールポイントになります。多くの企業では「データ活用、DXといえばPython」という雰囲気です。人事が捉えているので、加点を稼ぐことができるでしょう。

一方、Rは、「R？知らないなあ。大学でしか使わないような専門的なソフトでしょ？社会では役に立たないでしょ」といった反応が返ってきてしまうかもしれません。もちろん、データ分析をしっかりと行って、適材適所を理解している会社では、良い反応が得られるでしょう。

また、Pythonはどこの大学でも必修化が進んでいますし、民間の「プログラミングスクール」でも大半はPythonを採用しているので、競争相手が多く、優位性をアピールすることが難しいかもしれません。

5.2 時代はtidyverseファースト

- <https://www.tidyverse.org/>
- tidyverseは、「モダンなRプログラミング」を実現するためのパッケージ群
- 2020年代は、最初からtidyverseを使う前提でRを学んだ方が良いでしょう
- 詳細は第5回あたりから



30

この講義はRをテーマにしていますので、ここからはRを取り巻くトレンドを紹介します。Rの基本として紹介したように、Rの基本的な関数や「古典的」なパッケージを使えば、大抵のデータ分析を行うことができます。しかし、関数やパッケージによって書式や出力が異なるため、プログラムの可読性（見やすさ）や開発効率が低下する、という課題がありました。そこで、近年はより統一されたインターフェースで、見通しの良いプログラミングをするために、tidyverseというパッケージ群が開発され、世界的に広く使われています。

また、tidyverseパッケージ群を使用することを前提に、関連するパッケージも多数開発されており、現在ではRプログラミングにおいて、tidyverseを使うことが事実上の標準（デファクトスタンダード）になっています。

tidyverseについて、詳しくは次回以降の講義で紹介しますが、これからR言語について学習する上では、tidyverseを使うことを前提とした方が良いでしょう。

5.3 tidyverseを活用したプログラムの例

- tidyverseの特徴はパイプ (%>%)
- パイプで処理をつなげていくことで、効率的なプログラミングができる

```
library(tidyverse)
# irisデータからPetal.Length, Species列を選択し、Speciesでグループ化、
# ラベル (品種) ごとに平均を算出してデータフレームに格納する
df <- iris %>% select(Petal.Length, Species) %>%
  group_by(Species) %>%
  summarise(avg = mean(Petal.Length))

df

## # A tibble: 3 × 2
##   Species      avg
##   <fct>      <dbl>
## 1 setosa      1.46
## ...
```

31

tidyverse/パッケージ群の特徴は多岐にわたりますが、代表的なものとして、パイプ (%>%) を使った処理の連結があります。実際には、tidyverseに含まれるmagrittrパッケージの機能ですが、パイプ演算子の左辺で実行した処理の結果が、右辺に記述した次の処理の引数として渡されます。そのため、複雑な処理を次々にパイプでつなげて実現できます。

ある関数の実行結果をオブジェクトに代入し、そのオブジェクトを引数に別の関数を実行し、といったかたちではプログラムの行数が多くなりますし、関数(関数(関数(データ)))といった記述は可読性が下がります。関数(データ) %>% 関数() %>% 関数() とパイプを活用して書けば、見通しの良いプログラムが作成できます。

tidyverseで提供される関数は、ほぼすべてがパイプ演算子に対応しているので、効率的なプログラミングが可能です。

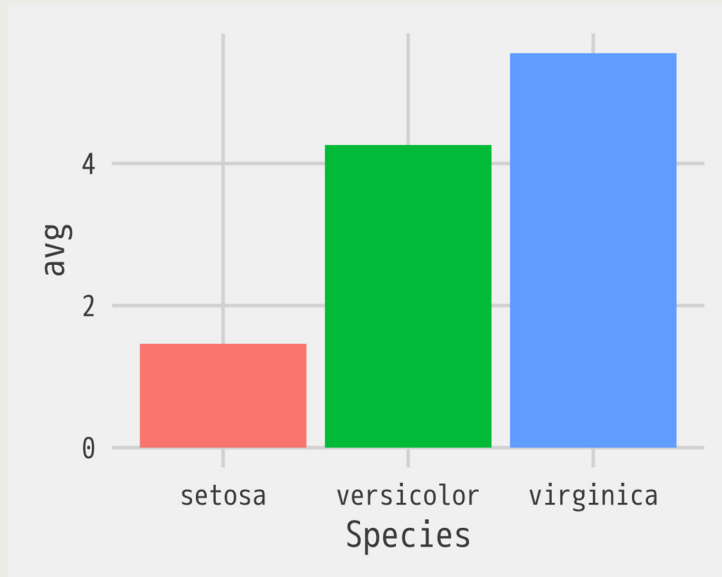
なお、2021年にリリースされたR 4.1.0からは、標準でパイプの機能を持つ演算子 (|>) が搭載されました。これは、Native Pipeと呼ばれます。左から右に処理を流す、という基本的な機能はtidyverseのパイプ演算子と変わりませんが、細かな違いがあります。今後、書籍やWebにおいて目にする機会が増えると思いますので、概要を知っておくとよいでしょう。

- 参考: R 4.1.0で導入された パイプ演算子 (|>)の紹介 / r_native_pipe - Speaker Deck
https://speakerdeck.com/s_uryu/r-native-pipe

5.3 tidyverseを活用したプログラムの例

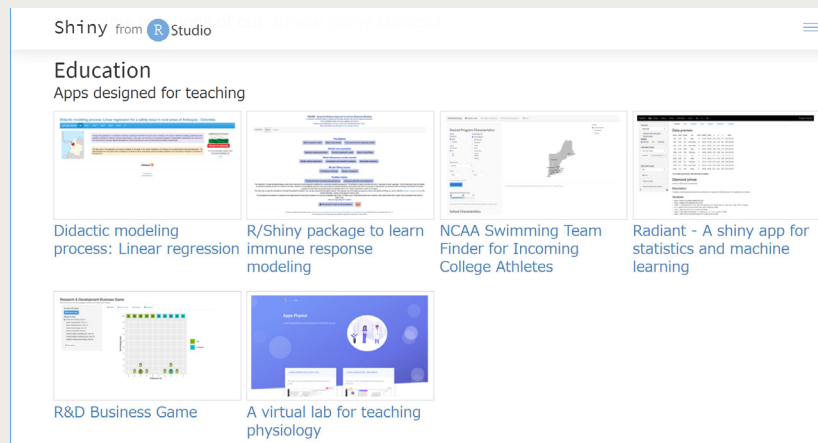
ggplot2パッケージでグラフィックスを描画

```
ggplot(df, aes(x = Species, y = avg, fill = Species)) +  
  geom_bar(stat = "identity") +  
  theme(legend.position = "none")
```



5.4 Shinyによるアプリ開発

- Shinyは、RでWebアプリを開発するためのパッケージ
- データ分析のためのプログラミングとはややアプローチが異なるが、データやグラフィックスをダッシュボードとして表示し、インタラクティブに分析結果の共有ができる



33

従来、データ分析の結果は、論文やレポートとして紙で共有されることが主流でした。しかし、現在ではペーパーレス化や研究やビジネスのスピードが加速したことにともない、PDFやHTMLでやり取りされることが一般的になりました。さらに、ビジネスの場では、毎日変化する指標 (KPI; Key Performance Indicator) をリアルタイムに監視し、意思決定することが求められます。そのため、RやPythonによる分析プログラムと、**ダッシュボード**と呼ばれるWebアプリを連携させ、KPIやその予測値を共有することが増えてきました。

Shinyは、RでダッシュボードなどのWebアプリを開発するためのフレームワークです。Rで分析した結果を、ファイルとして出力して外部のサーバーに転送したり、Webアプリ部分は別の言語で開発する必要がなく、直接ダッシュボードとして表示できます。ゼロからダッシュボードプログラムを構築することもできますが、開発を支援するパッケージが複数あるので、それらを活用することで、比較的容易にデータを可視化し、共有する仕組みが作れます。

なお、Shinyで作成するアプリをLAN内やインターネットに公開するためには、ネットワークやサーバーの知識がある程度必要になります。ShinyによるWebアプリ開発については、本講義の第15回前後で取り上げます。

5.5 開発環境のクラウド化

- Rは基本的にパソコンで使用するのがメインだった
- **RStudio Server**は、RStudioをより高性能なサーバーで動かし、ブラウザ経由でアクセスする
- **Posit Cloud**は、Posit社が管理するクラウドでサービスを提供する
- **shinyapps.io**は、Shinyアプリをホスティングするサービス
- **Google Colab**は、Pythonがメインだが、Rカーネルも使用できる
- 再現可能な研究（Reproducible research）のためにも、クラウドに分析環境を構築することは合理的

34

2010年代後半まで、Rの実行環境は、手元のパソコンであることがほとんどでした。大規模にRを利用している研究室や企業では、**RStudio Server**というサーバー・クライアント型の環境を構築したり、手元で作成したプログラムを転送して、高速な計算機（サーバー）上でRを実行することもあります。いずれにしても、LAN内の閉じた仕組みでした。しかし、機械学習など複雑で多くのCPUやメモリを必要とする分析が主流になったり、テレワークなど業務環境が多様化したこと、組織をまたいだ共同研究開発の広がり（分析コンペのチーム戦化も）などで、インターネット越しにどこからでも利用できる、高速なR実行環境が求められるようになってきました。

そのような背景から、Rをクラウド上で利用できるサービスが提供されています。本講義でも、**Posit Cloud** (<https://posit.cloud/>) をハンズオン環境として使用します。**shinyapps.io** (<https://www.shinyapps.io/>) も、posit.cloudが提供する、Shinyアプリのホスティングサービスです。前述したダッシュボードを、自前でサーバーを用意する必要なく、関係者で共有できます。

また、Googleが提供する「ノートブック」サービスである**Colaboratory** (<https://colab.research.google.com/>) でも、Rを使用できます。標準ではPythonが使用できますが、ノートブックの設定でRカーネル（実行環境）を指定できます。Colabでは、高速な行列演算などが可能なGPGPU（General purpose computing on graphics processing units）やTPU（Tensor processing unit）が使用できます。これにより、ディープラーニングなどの複雑な計算を大抵のパソコンよりも高速に実行できます。

5.5 開発環境のクラウド化

The image displays two web interfaces side-by-side. The left interface is the Posit Cloud homepage, featuring the heading "Friction free data science" and a description of the platform as a browser-based data science environment. It includes "GET STARTED" and "ALREADY A USER? LOG IN" buttons. The right interface is the shinyapps.io homepage, titled "Share your Shiny Applications Online", which promotes deploying Shiny applications to the web. It features a "Sign Up" button and a navigation menu with links for Home, Features, Pricing, Support, and Log In. Both pages include a header with "Log In" and "Sign Up" links.

posit Cloud

Log In Sign Up

Friction free data science

Posit Cloud (formerly RStudio Cloud) lets you access Posit's powerful set of data science tools right in your browser – no installation or complex configuration required.

GET STARTED ALREADY A USER? LOG IN

shinyapps.io by RStudio

Home Features Pricing Support Log In

Share your Shiny Applications Online

Deploy your Shiny applications on the Web in minutes

Sign Up

5.6 データ分析の大規模化とワークフロー管理

- Rは基本的には、統計学（できるだけ少ないサンプルを使い、できるだけシンプルなモデルを作る）のためのツール
- 現在は、機械学習（できるだけ多くのデータを使い、できるだけ精度の高い \Rightarrow 複雑なモデルを作る）アプローチが主流
- データやプログラムが複雑化する中で、分析フローを適切に管理することが求められる（再現性のある研究）
- Rでは、GitHubなどの外部ツールだけでなく、tidymodelsやmlr3といった機械学習フレームワーク、renvやworkflowrなどのワークフロー管理のためのパッケージが使用される

36

近年、機械学習の高度化に伴い、アルゴリズムも、投入するデータも複雑さを増し、データ分析が大規模化する中で、分析のワークフロー（全体像）を適切に管理する必要が生じてきました。また、分析結果を共同作業や他の研究者が再現し、検証できるように、使用したパッケージのバージョンなども管理しなければなりません。

そのような目的で、Rによるワークフローを管理するためのパッケージとして、tidymodelsやmlr3といったフレームワークが利用されています。これらは機械学習を念頭に置いたパッケージですが、より広くデータ分析の工程を管理するパッケージとして、renvやworkflowrなどがあります。また、Rプログラムに限らず、データやドキュメントなどを含めた分析環境全体について、GitHubなどを使いバージョン管理することも一般的です。

フレームワークを用いたデータ分析、機械学習については、本講義の第10回前後で取り上げます。

5.6 データ分析の大規模化とワークフロー管理



6. まとめ

6.1 今日の内容

- Rの概要
 - データフレームの操作
 - 集計
 - 基本的なグラフィックス
 - パッケージ
- 2020年代のトレンド
 - tidyverse
 - 詳しくは以降の授業で

6.2 次回までの課題

- Posit Cloudプロジェクト内の
`02_trend_exercise.R` について、指示に従って
プログラムを作成してください
- 編集したファイルは、ファイル一覧でチェックを
入れ、[more] メニューから [Export] を選択し、
[Download] ボタンを押してダウンロードして
ください
- ダウンロードしたファイルを、Classroomの課題
ページから提出してください
- 提出期限: 2023-10-09 23:59