

本講座の目的

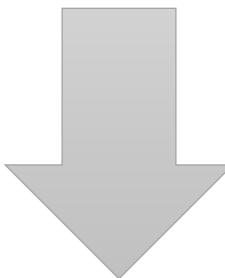
データの前処理のほぼすべてに対応できること

本講座の対象者

- ExcelやPythonのNumpy, Pandasで前処理に挫折した方
- baseRでの前処理に限界を感じた方
- tidyverseは知ってるけど、使い方がよくわからない方
- その他、少しでも前処理を楽にしたい方

ブログ講座 : <https://datasciencemore.com/category/ds-lecture/r-preprocess-lecture/>

なんでデータの前処理が対象なの？？



前処理は、データ分析プロジェクトのどのような仕事でも
大部分を占める重要な部分だから！！

①入力フェーズ

②処理フェーズ

③出力フェーズ

多くのウェイト
を占める！！

データ

前処理

アルゴリズム

可視化アプリ

成果物

資料作成

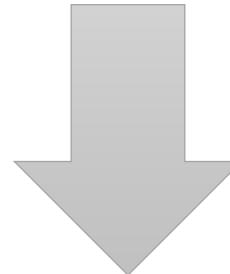
モデリング

AIシステム

論文

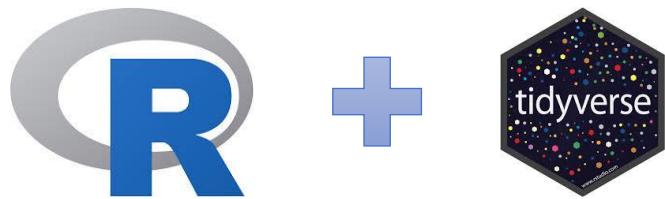
本講座の特徴

- とにかく現場主義！！
- コーディングはリアルタイム形式！！
- コードだけでなく、イメージも！！



前処理スキル向上！！

スーパーデータ
分析環境



データ分析環境

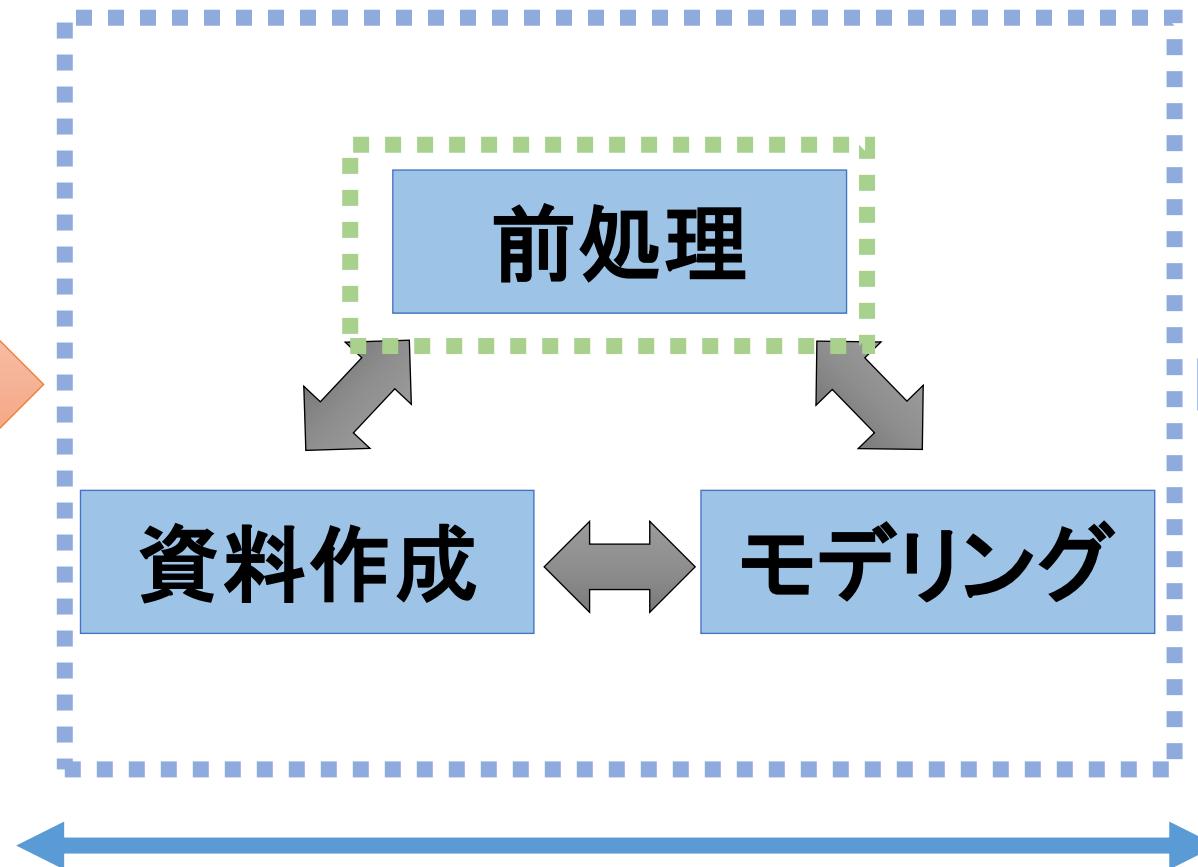
ノーマル環境



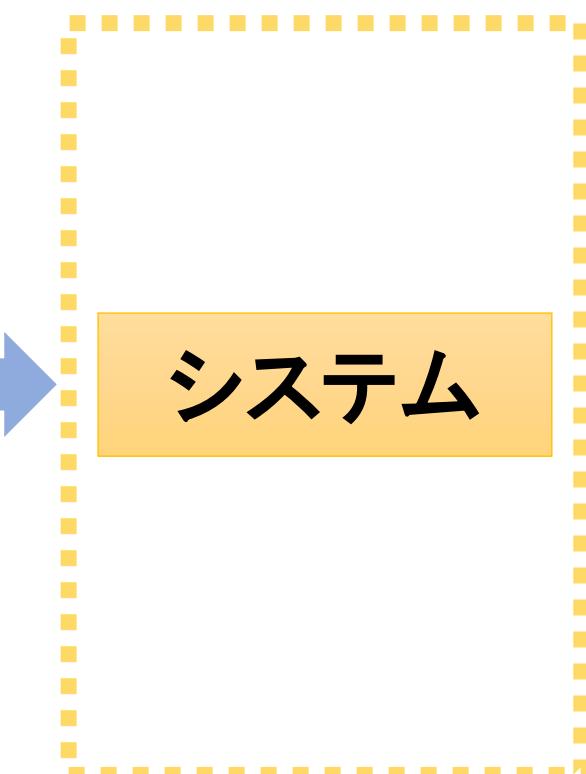
①入力フェーズ



②処理フェーズ



③出力フェーズ



Python: できるけど苦手
R :得意

Python: 得意
R :不可₆

df_member

| No | gender | height |
|----|--------|--------|
| 1 | M | 165 |
| 2 | F | 150 |
| 3 | F | 170 |
| 4 | M | 175 |
| 5 | F | 165 |
| 6 | M | 195 |
| 7 | M | 180 |

①gender = F
②height > 160
を抽出！！



| No | gender | height |
|----|--------|--------|
| 1 | M | 165 |
| 2 | F | 150 |
| 3 | F | 170 |
| 4 | M | 175 |
| 5 | F | 165 |
| 6 | M | 195 |
| 7 | M | 180 |

```
library(tidyverse)
df_member %>%
filter( gender == "F", height > 160 )
```

```
import pandas as pd
df_member[(df_member["gender"] == "F") & (df_member["height"] > 160)]
```



VS



| No. | 分類 | R | Python |
|-----|-------|---|--------|
| 1 | 前処理 | ○ | △ |
| 2 | 可視化 | ○ | × |
| 3 | システム化 | × | ○ |
| 4 | 情報量 | △ | ○ |

R前處理講座

1. 環境構築



2. baseR



3. tidyverse



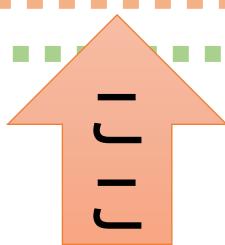


時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|--|-------------|
| 1 | tibble | A dark blue hexagonal logo with the word "TIBBLE" at the top and a stylized bar chart graphic below it. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagonal logo with the word "dplyr" at the top and a colorful rocket ship graphic below it. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagonal logo with the word "tidyr" at the top and a colorful geometric pattern graphic below it. | tidyデータ処理 |
| 4 | ggplot2 | A light gray hexagonal logo with the word "ggplot2" at the top and a small line plot graphic below it. | 可視化 |
| 5 | stringr | A green hexagonal logo with the word "stringr" at the top and a small violin graphic below it. | 文字列処理 |
| 6 | readr | A dark blue hexagonal logo with the word "readr" at the top and a small document and grid icon graphic below it. | 入出力処理 |
| 7 | forcats | A brown hexagonal logo with the word "forcats" at the bottom and a small illustration of two cats above it. | ファクター処理 |
| 8 | purrr | A light gray hexagonal logo with the word "purrr" at the top and a small cat's head graphic below it. | 繰り返し処理 |

R前處理講座

1. 環境構築



2. baseR



3. tidyverse



Docker

Dockerってなんですか？？



Dockerってのは環境構築を楽にするツールのことだよ！





Docker準備

1. Docker Hubの登録

- ① <https://hub.docker.com/>へ移動
- ② Docker ID, Email, Passwordの登録

2. Dockerのインストール

- ① <https://www.docker.com/products/docker-desktop>へ移動
- ② Docker Desktop Installer.exeをダウンロード＆インストール

- 1.コマンドプロンプト(Macの方はターミナル)を起動する.
- 2.docker loginと打ってEnterを押します.
- 3.ユーザ名とパスワードを聞かれることがあるので,
ご自身のDockerHubのユーザ名とパスワードを入力してください. (聞かれないこともあります.)
- 4.ログインに成功するとLogin Succeededと表示されます.

管理者: コマンドプロンプト

```
Microsoft Windows [Version 10.0.19041.501]
(c) 2020 Microsoft Corporation. All Rights Reserved.

C:\Users\Administrator>docker login
Authenticating with existing credentials...
Login Succeeded
```

4 ログインに成功すると
Login Succeededと表示される.

2 docker loginと入力
し, Enterを押す.

3 ユーザ名, パスワードを聞かれる時
があるので, その時はご自身の
DockerHubのユーザ名とパスワード
を入力する.

管理者: コマンドプロンプト

Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\Administrator> docker pull rocker/tidyverse:4.0.3

4.0.3: Pulling from rocker/tidyverse

da7391352a9b: Already exists

14428a6d4bcd: Already exists

2c2d948710f2: Already exists

5bfe9ae7dbf4: Pull complete

b2e9294d1fca: Pull complete

1cbf10622697: Pull complete

3cb93fbffbf4f: Pull complete

1d4c21f7e6cd: Pull complete

Digest: sha256:b434972c07ed4b5

Status: Downloaded newer image for docker.io/rockertidyverse:4.0.3

1 docker pull rocker/tidyverse:4.0.3
と入力し, Enterを押す.

2 docker imagesと入力し, Enterを押す.

3 rocker/tidyverseと表示さればOK

| REPOSITORY | TAG | IMAGE ID | CREATED | SIZE |
|------------------|-------|--------------|-------------|--------|
| rocker/tidyverse | 4.0.3 | 82e99b98c2a9 | 3 weeks ago | 2.47GB |

3 rocker/tidyverseと表示さればOK

- ご自身のデスクトップにr-preprocess-lectureというフォルダ名のフォルダを作成する。
- コマンドプロンプト上でdocker run -p 8787:8787 -v ○○○○:/home/rstudio/r-preprocess-lecture -e PASSWORD=×××× rocker/tidyverse:4.0.3と入力しEnterを押す。
※○○○○と××××には、それぞれ以下を入力する。
○○○○: 1. で作成したフォルダの絶対パス
××××: 適当なパスワード(ご自身でわかればなんでもいいです。)

管理者: コマンドプロンプト - docker run -p 8787:8787 -v C:\Users\Administrator\Desktop\r-preprocess-lecture:/home/rstudio/r-preprocess-lecture -e PASSWORD=a0572208 rocker/tidyverse:4.0.3

Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

```
C:\Users\Administrator>docker pull rocker/tidyverse:4.0.3
4.0.3: Pulling from rocker/tidyverse
da7391352a9b: Already exists
14428a6d4bcd: Already exists
2c2d948710f2: Already exists
5bfe9ae7d6f4: Pull complete
b2e9294d1fcfa: Pull complete
1cbf10622697: Pull complete
3cb93fbbbf4f: Pull complete
1d4c21f7e6cd: Pull complete
Digest: sha256:b434972c07ed4b57dd1ac79620c9599580a37fb2dd72/a/bcdt4900t42c77
Status: Downloaded newer image for rocker/tidyverse:4.0.3
docker.io/rocker/tidyverse:4.0.3
```

```
C:\Users\Administrator>docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
rocker/tidyverse    4.0.3   c92-001-00   2 days ago   2.47GB
```

```
C:\Users\Administrator>docker run -p 8787:8787 -v
[s6-init] making user provided files available...
[s6-init] ensuring user provided files have correct perms...exited 0.
[fix-attrs.d] applying ownership & permissions
[fix-attrs.d] done.
[cont-init.d] executing container initialization...
[cont-init.d] userconf: executing...
[cont-init.d] userconf: exited 0.
[cont-init.d] done.
[services.d] starting services
[services.d] done.
```

2 docker run -p 8787:8787 -v ○○○○:/home/rstudio/r-preprocess-lecture -e PASSWORD=×××× rocker/tidyverse:4.0.3
と入力しEnterを押す。

1. で作成したフォルダの絶対パスを入力する。

適当なパスワードを入力する。



2

UsernameとPasswordに次のように入力し,
Sign Inを押す。

Username:rstudio

Password:先ほどの××××のパスワード

Sign in to RStudio

Username:

Password:

Stay signed in when browser closes

You will automatically be signed out after 60 minutes of inactivity.

Sign In

R RStudio Server x +

localhost:8787

File Edit Code View Plots Session Build Debug Profile Tools Help

rstudio Project: (None)

Console Terminal Jobs

R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |

Environment History Connections Tutorial

Import Dataset Global Environment

Environment is empty

Files Plots Packages Help Viewer

New Folder Upload Delete Rename More

Home

| | Name | Size | Modified |
|--|----------------|------|-----------------------|
| | .Rhistory | 0 B | Dec 28, 2020, 3:00 PM |
| | r-base-lecture | | |
| | workspace | | |

RStudio

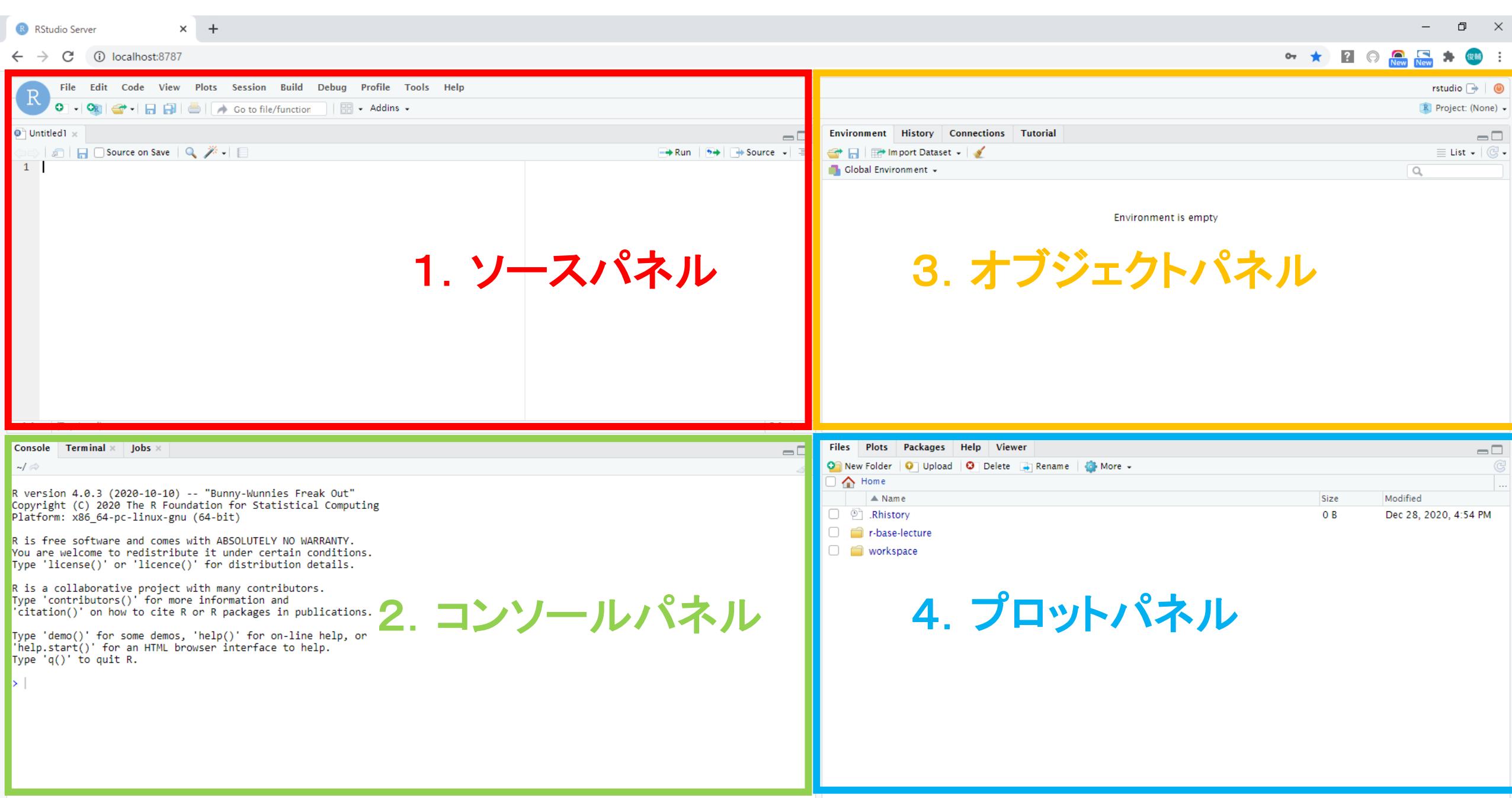
RStudioってなんですか？



RStudioは、Rを使用するためのIDE（統合開発環境）のことだよ。

RStudioは、RのIDEのデファクトスタンダードで、とても高機能なんだ！！





RStudio Server

localhost:8787

R

File Edit Code View Plots Session Build Debug Profile Tools Help

Untitled1 * Go to file/function Addins

Source on Save

1 1 + 1
2
3 test = 1 + 1
4

4:1 (Top Level) R Script

Console Terminal Jobs

~/

R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 1 + 1
[1] 2
> test = 1 + 1
>

Environment History Connections Tutorial

Import Dataset Global Environment

Values test 2

Files Plots Packages Help Viewer

New Folder Upload Delete Rename More

Home .Rhistory r-base-lecture

© 2021 shun

rstudio Project: (None)

23

RStudio Server

localhost:8787

R

File Edit Code View Plots Session Build Debug Profile Tools Help

Untitled1 * Go to file/function Addins

Run Source

Import Dataset

Global Environment

Values test 2

Environment History Connections Tutorial

Source on Save

1 1 1

2 test = 1 + 1

3

4

3:1 (Top Level) R Script

Console Terminal Jobs

~/

R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 1 + 1
[1] 2
> test = 1 + 1
> |

Files Plots Packages Help Viewer

New Folder Upload Delete Rename More

Home .Rhistory r-base-lecture

Name Size Modified

0 B Dec 28, 2020, 4:54 PM

testを選択し、Ctrl + Enter

プロジェクト1

プロジェクト1
の資料

プロジェクト
ファイル1

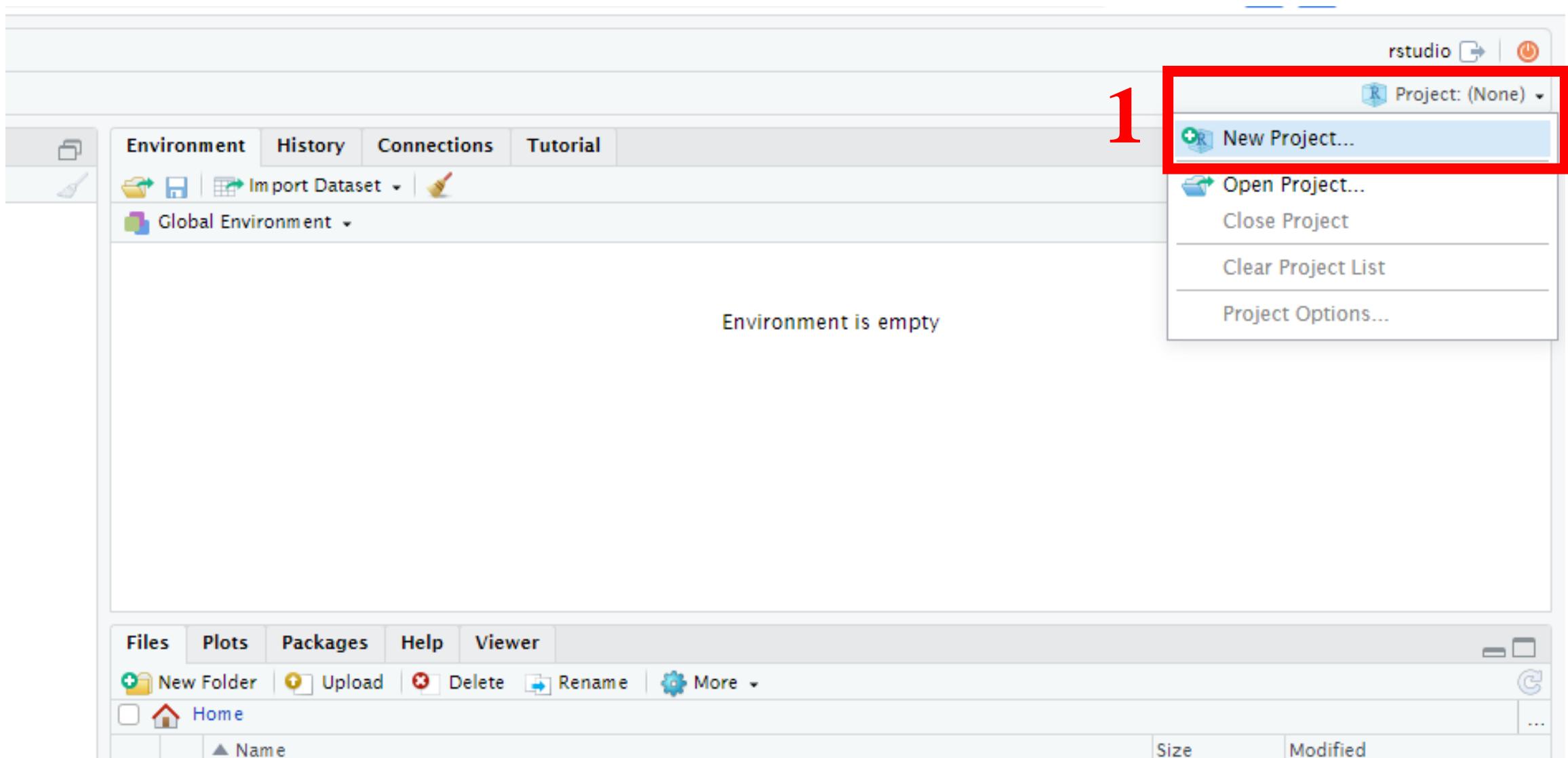
プロジェクト2

プロジェクト2
の資料

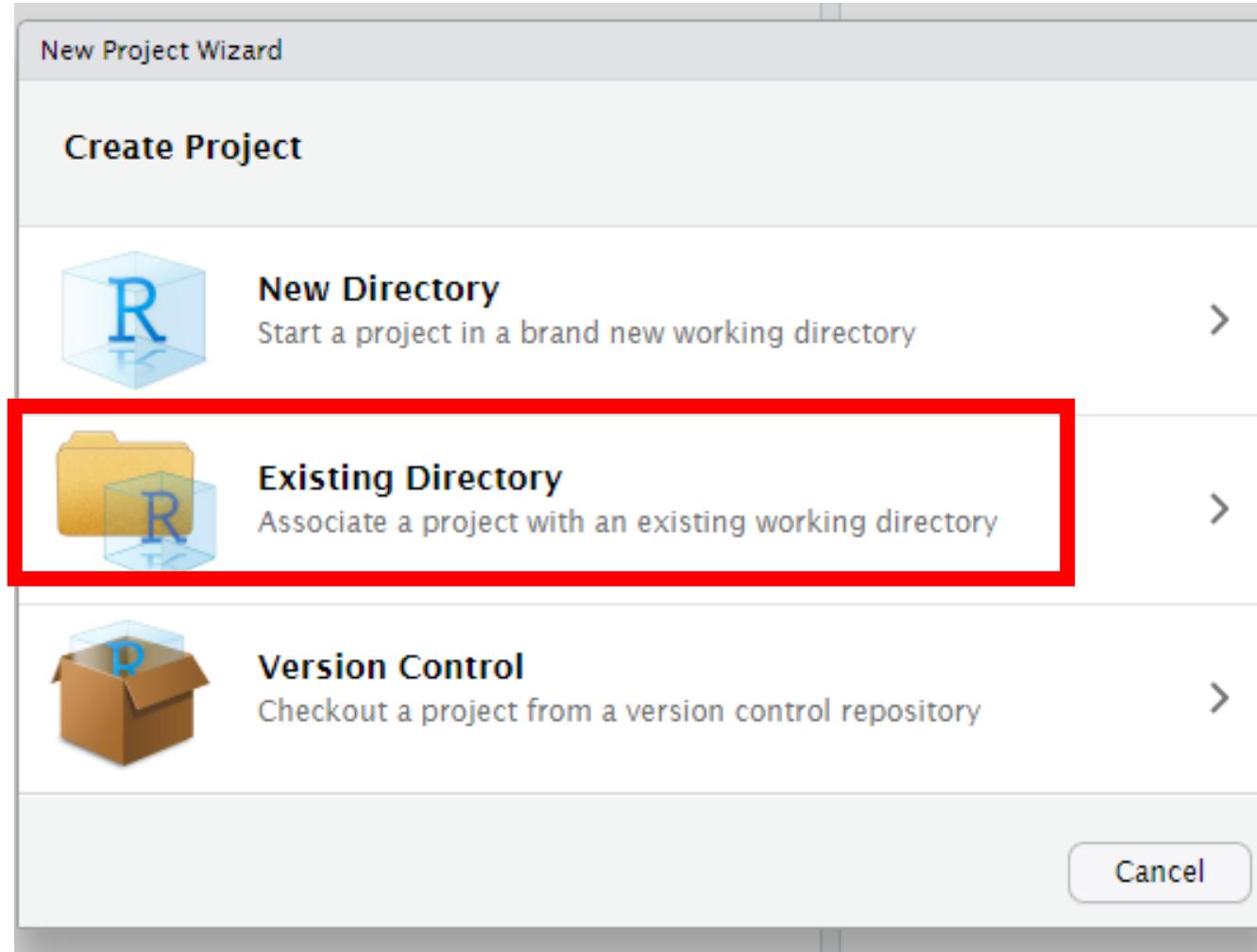
プロジェクト
ファイル2

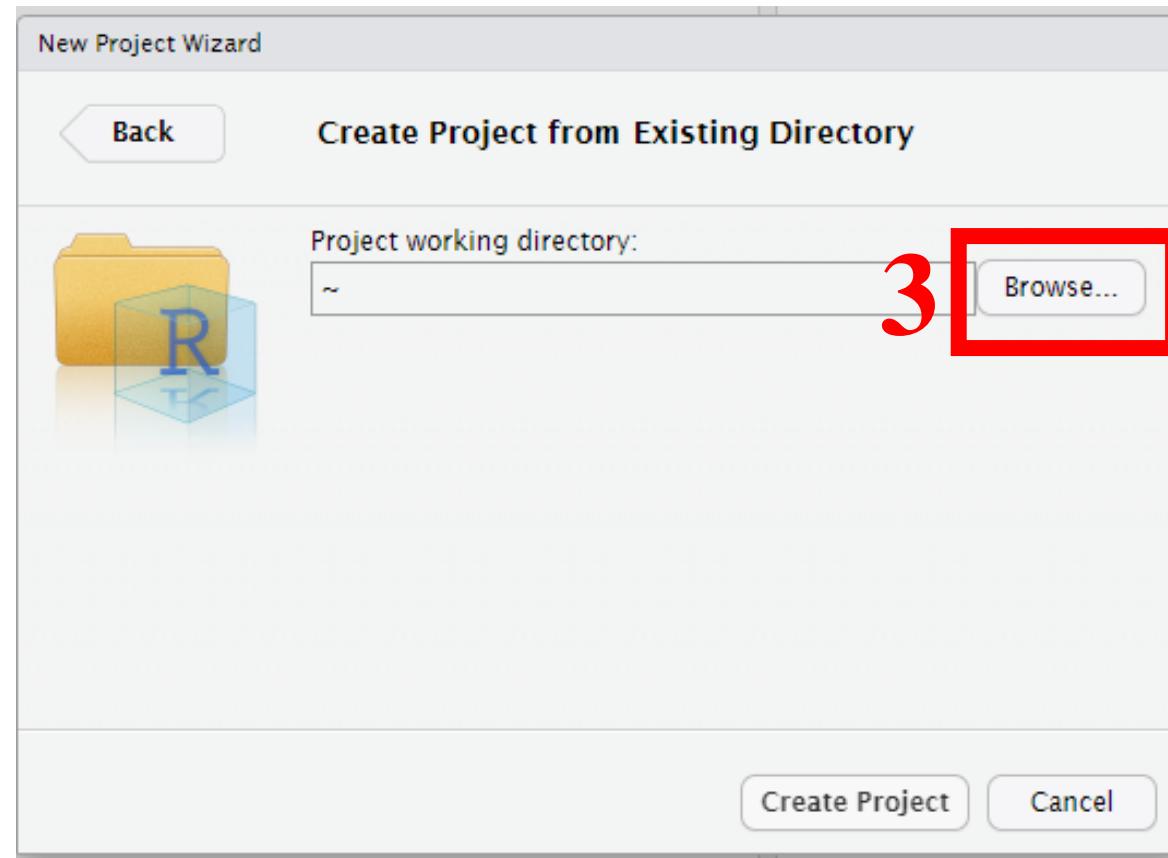
プロジェクトファイルのメリット

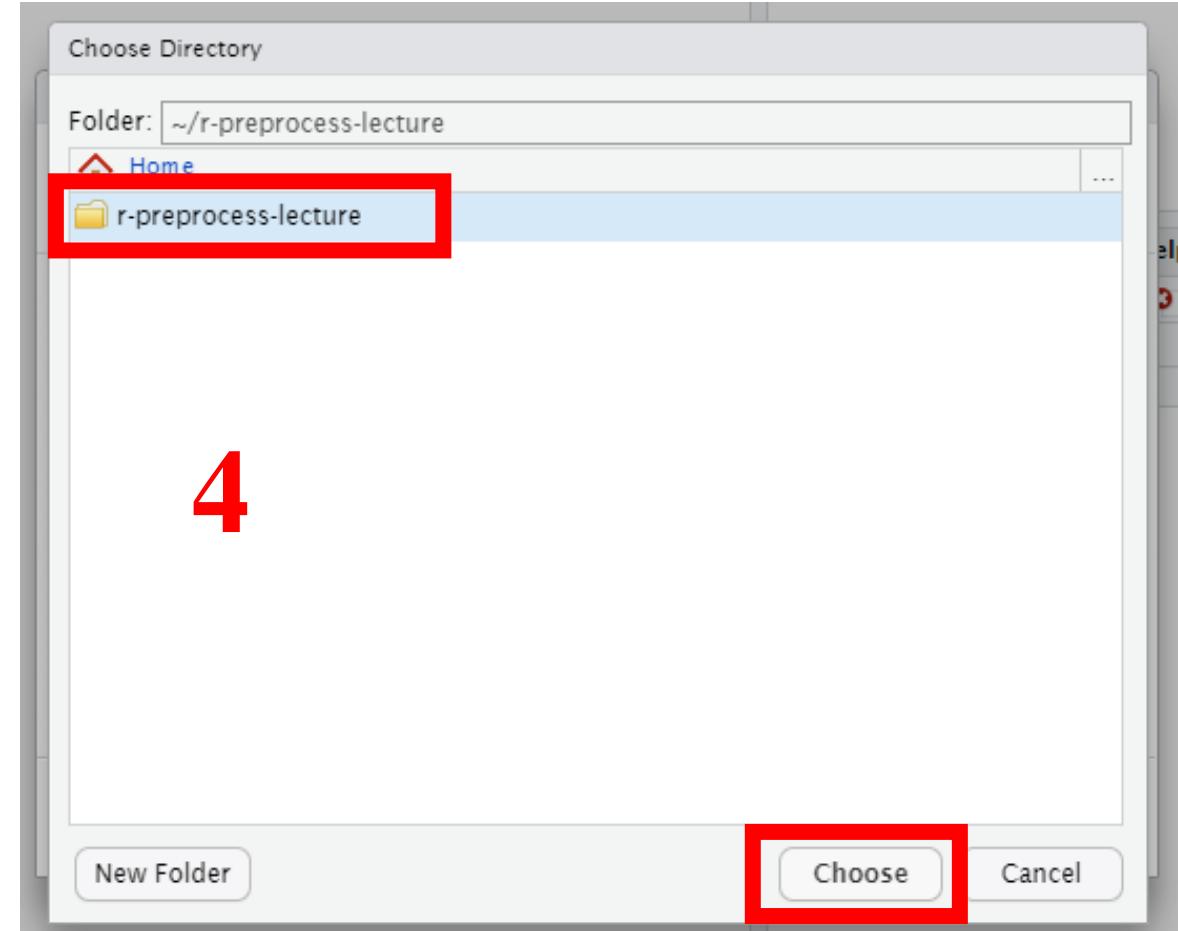
- ・ワーキングディレクトリをプロジェクトファイルが格納されているフォルダにしてくれる。
- ・gitの連携が楽にできる。
- ・資料の管理が楽。
- ・複数のプロジェクトの管理が楽。



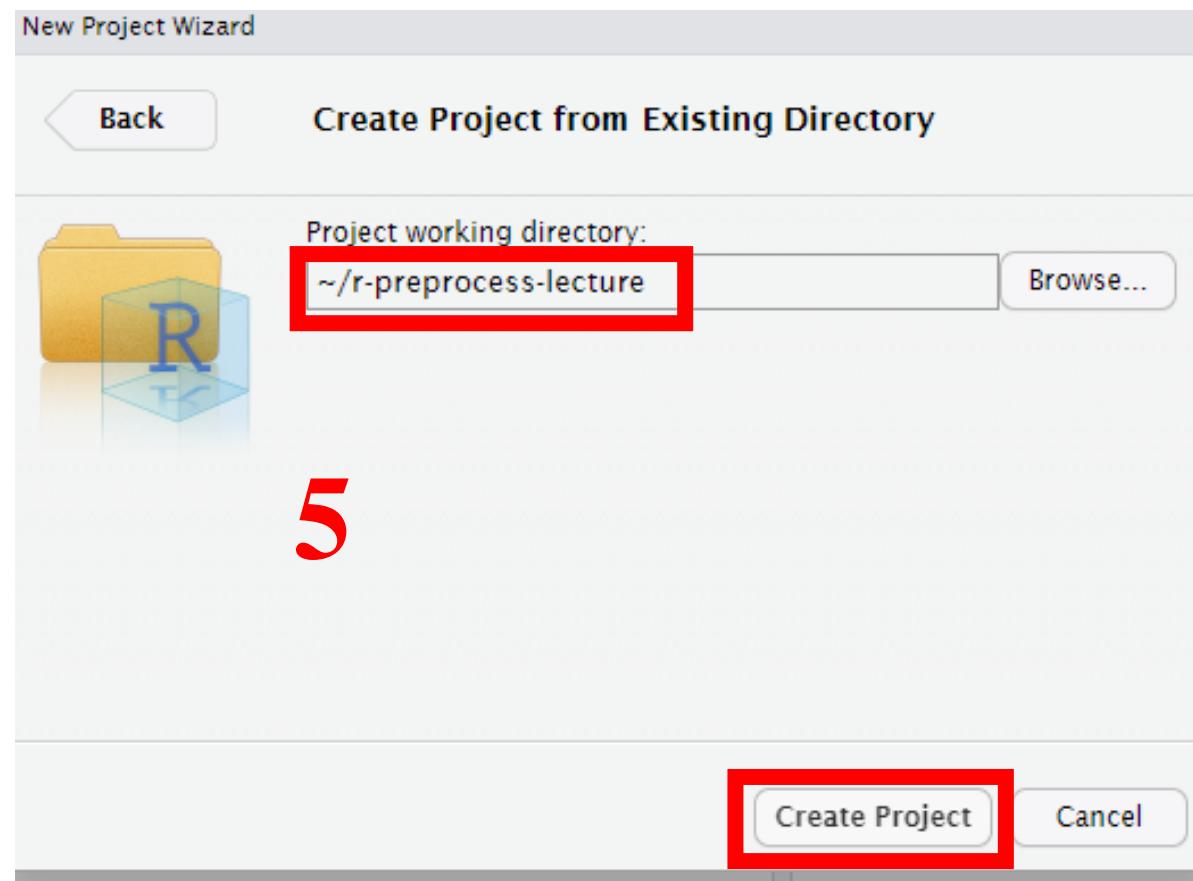
2

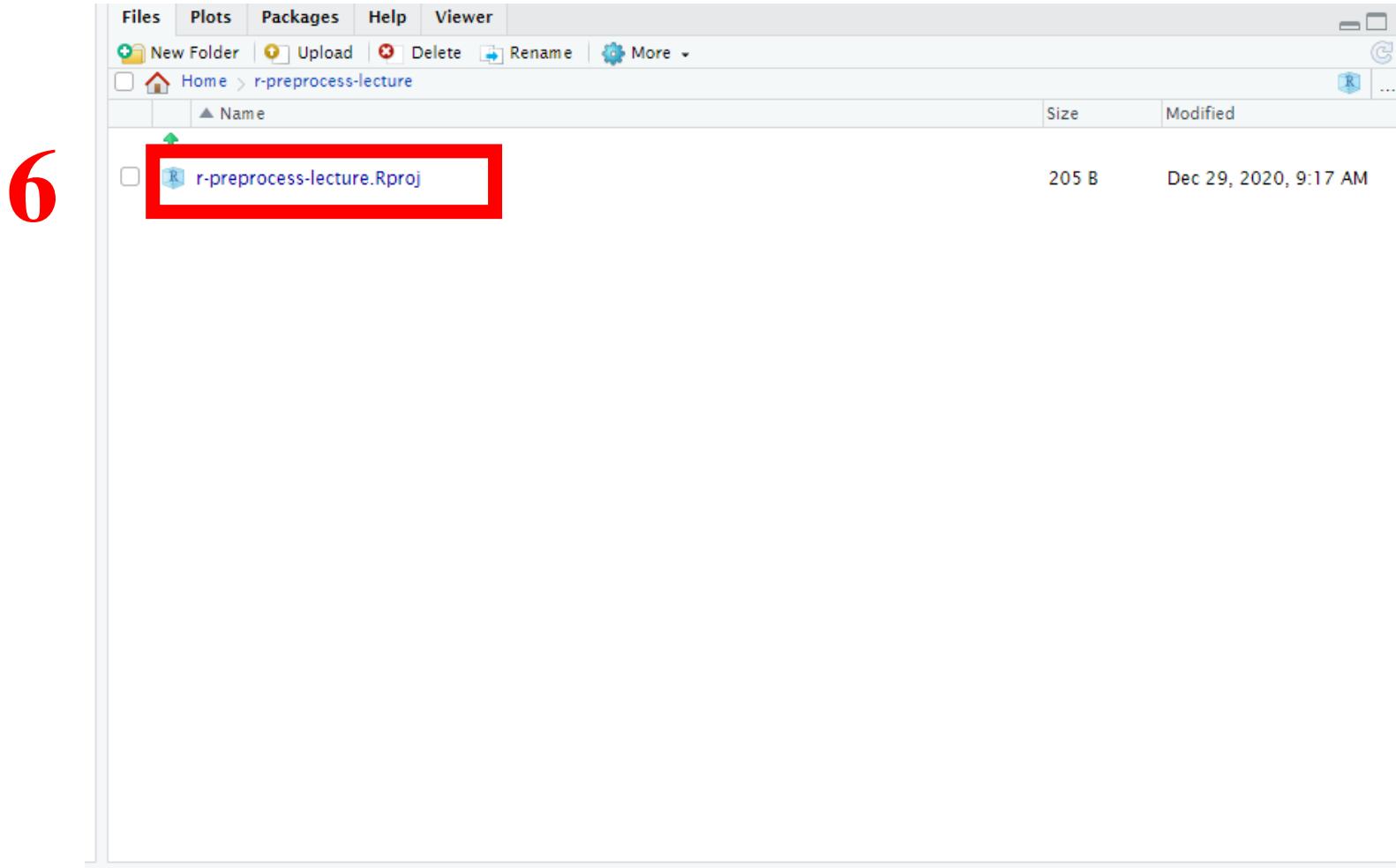






4





CRAN
GitHub
など

パッケージ1

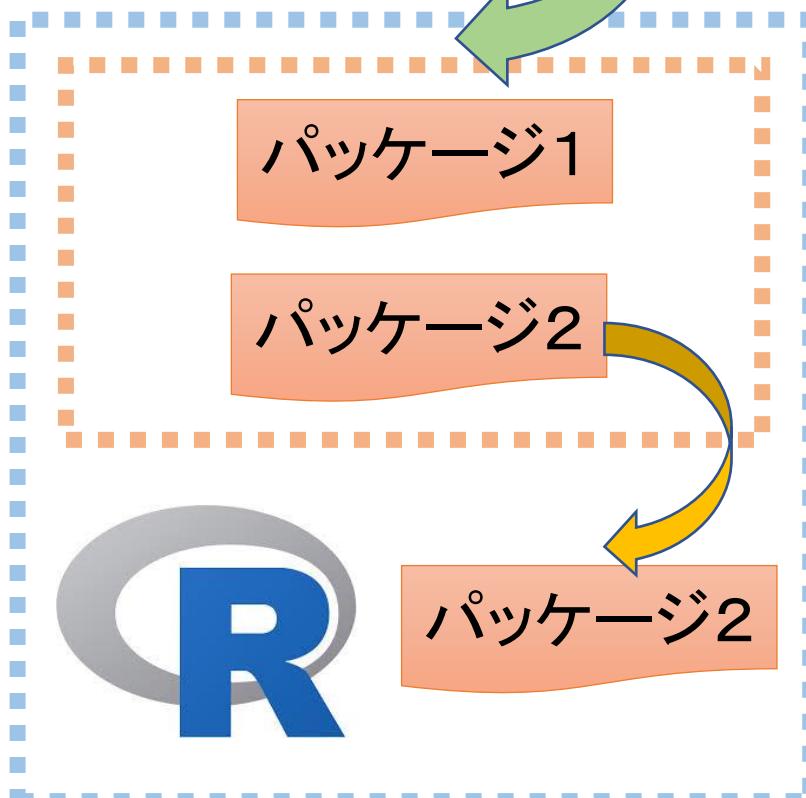
パッケージ2

パッケージ3

パッケージ4

```
install.pakages("パッケージ1")  
install.pakages("パッケージ2")
```

1. インストール



```
library(パッケージ2)
```

2. 呼び出し

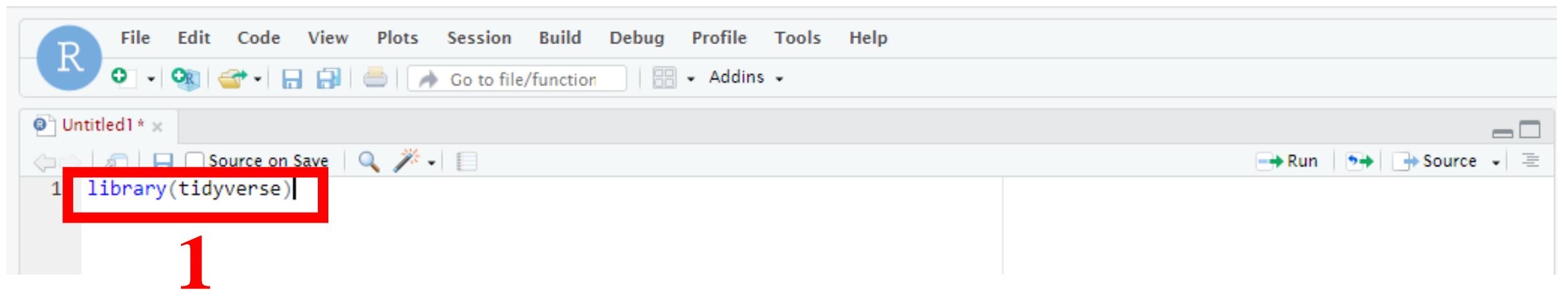
.Rprofile

.Rprofileってなんですか？？



.Rprofileは、Rが起動したときに自動的に読み込むファイルのことだよ。
ここに初期設定によく使う処理を書いておけば自動的に処理してくれるから楽なんだ♪

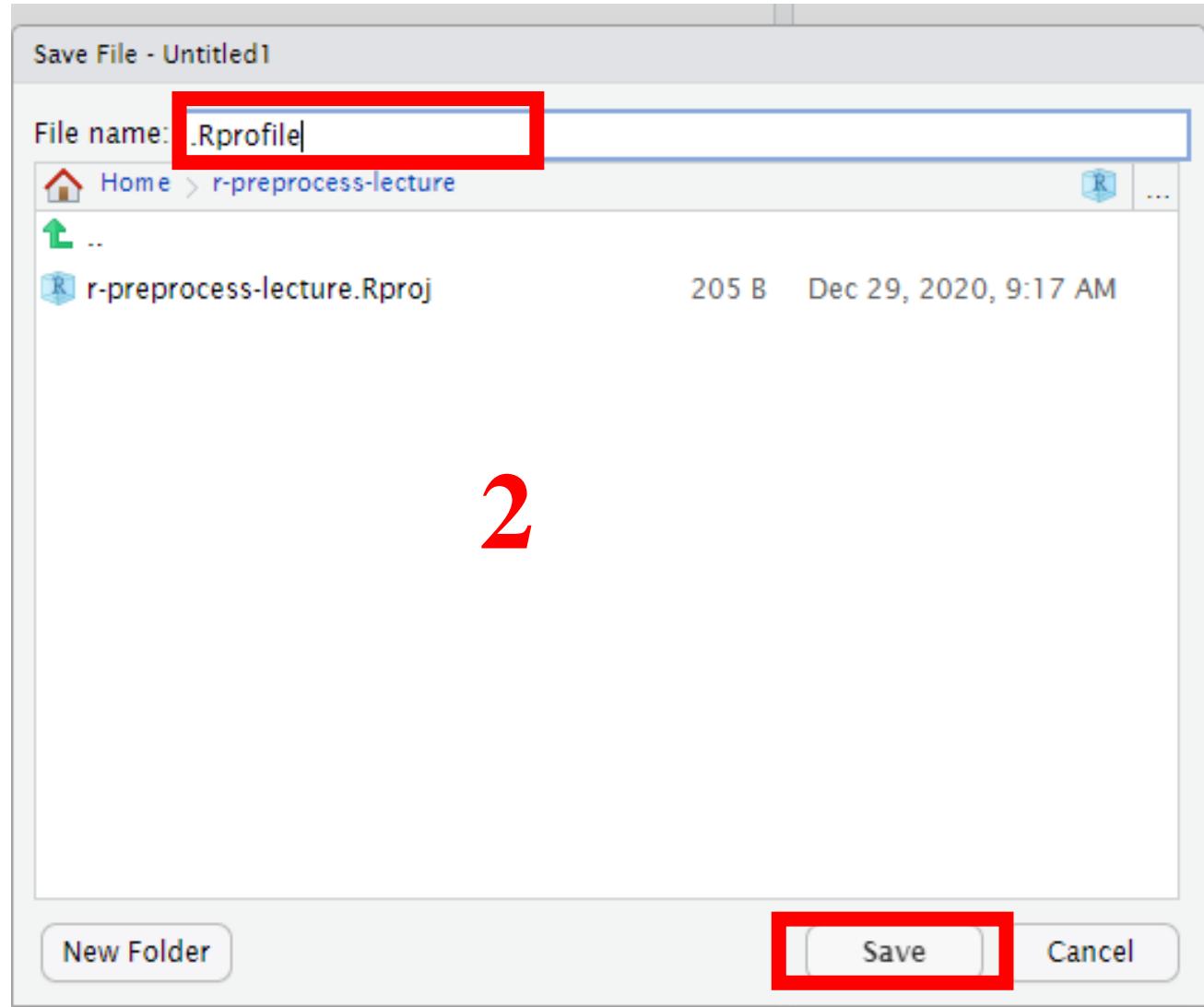




The screenshot shows the RStudio IDE interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with various icons for file operations like Open, Save, and Print. The main code editor window has a tab labeled "Untitled1 *". The code area contains the following line of R code:

```
1 library(tidyverse)
```

A large red box highlights the entire line of code "library(tidyverse)". A large red number "1" is positioned below the code editor window.



The screenshot shows the 'Files' view in RStudio. The top menu bar includes 'Files', 'Plots', 'Packages', 'Help', 'Viewer', and various icons for file operations like 'New Folder', 'Upload', 'Delete', 'Rename', and 'More'. Below the menu is a breadcrumb navigation bar showing the path: Home > r-preprocess-lecture. The main area displays a table of files in the 'r-preprocess-lecture' folder. The columns are 'Name', 'Size', and 'Modified'. Two files are listed: 'r-preprocess-lecture.Rproj' (205 B, Dec 29, 2020, 9:17 AM) and '.Rprofile' (18 B, Dec 29, 2020, 11:08 AM). The '.Rprofile' file is highlighted with a red box.

| | Name | Size | Modified |
|--------------------------|----------------------------|-------|------------------------|
| <input type="checkbox"/> | .. | | |
| <input type="checkbox"/> | r-preprocess-lecture.Rproj | 205 B | Dec 29, 2020, 9:17 AM |
| <input type="checkbox"/> | .Rprofile | 18 B | Dec 29, 2020, 11:08 AM |

3

Docker imageの更新

1. 更新内容をDoker imageにする.
2. Docker Hubにリポジトリを作成する.
3. 更新したDocker imageを作成したりポジトリにpushする.

1. 更新内容をDoker imageにする

- ①コマンドプロンプト上でdocker psと入力し, Enter(Dockerにログインしていない場合は、ログインしてください。)
- ②表示されるCONTAINER IDを記録
- ③docker commit ②のCONTAINER ID ○○○○/r-preprocess-lectureと入力し, Enter
※○○○○は、各自のDocker Hubのアカウント名
- ④docker imagesと入力し, Enter
- ⑤○○○○/r-preprocess-lectureと表示されたらOK

2. Docker Hubにリポジトリを作成する

- ① [Docker Hub](#)に移動
- ② Create Repositoryをクリック
- ③ r-preprocess-lectureと入力し, Createをクリック

3. 更新したDocker imageを作成したり ポジトリにpushする.

①コマンドプロンプトに

docker push ○○○○/r-preprocess-lecture
と入力し, Enter

※○○○○は, 各自のDocker Hubのアカウント名

開発環境の再現方法

大体何でもOK 例:8888

```
docker run -p 8787:8787 -v ★★★★:/home/rstudio/r-preprocess-lecture -e PASSWORD=××××〇〇〇〇/r-preprocess-lecture
```

★★★★:「r-preprocess-lecture」フォルダの絶対パス

××××:適当なパスワード

〇〇〇〇:各自のDocker Hubのアカウント名

開発環境の再現方法

```
docker run -p 8787:8787 -v ★★★★:/home/rstudio/r-preprocess-lecture -e PASSWORD=×××× shun113/r-preprocess-lecture
```

★★★★:「r-preprocess-lecture」フォルダの絶対パス
××××:適当なパスワード

CONTAINER IDを記録

②

10. docker psと入力し, Enter

Microsoft Corporation. All rights reserved.

C:\\$Users\\$Administrator>docker ps

| CONTAINER ID | IMAGE | COMMAND | CREATED |
|--------------|------------------------|---------|-------------------|
| 45f1636c6df5 | rocker/tidyverse:4.0.3 | "/init" | About a month ago |

各自のDocker Hubのアカウント名

C:\\$Users\\$Administrator>docker commit 45f1636c6df5 /r-preprocess-lecture

sha256:81044cab0c443a20bfb50f2b2dbde45f1fcbbe0dedb3c814788251dd2d30d0b

C:\\$Users\\$Administrator>docker images

| IMAGE | IMAGE ID | CREATED | SIZE |
|-----------------------|---|-------------|--------|
| /r-preprocess-lecture | 81044cab0c443a20bfb50f2b2dbde45f1fcbbe0dedb3c814788251dd2d30d0b | 3 weeks ago | 2.4/GB |

3 docker commit ②のCONTAINER ID ○○○○/r-preprocess-lectureと入力し, Enter

4 docker imagesと入力し, Enter

5 ○○○○/r-preprocess-lectureと入力されたらOK



Welcome to Docker Hub

Download and Take a Tutorial

Get started by downloading Docker Desktop, and learn how you can build, tag and share a sample image on Hub.

[Get started with Docker Desktop](#)



Create a Repository

Push container images to Docker Hub



Create an Organization

Manage Docker Hub repositories with your team

②

Create a Repositoryをクリック

Access the world's largest library of container images

各自のDocker Hubのアカウント名

Explore Repositories Organizations Get Help ▾ shun113 ▾ 

Using 0 of 1 private repositories. [Get more](#)

Pro tip

You can push a new image to this repository using the CLI

```
docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname
```

Make sure to change *tagname* with your desired image repository tag.

Create Repository

r-preprocess-lecture

Description

Visibility

Using 0 of 1 private repositories. [Get more](#)



Public 

Public repositories appear in Docker Hub search results



Private 

Only you can view private repositories

③

Build Settings (optional)

Autobuild triggers a new build with every git push to your source code repository. [Learn More](#).

Please re-link a GitHub or Bitbucket account



We've updated how Docker Hub connects to GitHub and Bitbucket. You'll need to re-link a GitHub or Bitbucket account to create new automated builds. [Learn More](#)



Disconnected



Disconnected

Cancel

Create

Create & Build

各自のDocker Hubのアカウント名

```
C:\Users\Administrator>docker push [REDACTED]/r-preprocess-lecture
Using default tag: latest
The push refers to repository [docker.io/[REDACTED]/r-preprocess-lecture]
09fc6185706f: Pushed
f2bbfd2a4f55: Mounted from rocker/r-base:3.6.1
3818991d0c97: Mounted from rocker/r-base:3.6.1
3ebef4ab53f2: Mounted from rocker/r-base:3.6.1
a6989916ca61: Mounted from rocker/r-base:3.6.1
055e0b537fa6: Mounted from rocker/r-base:3.6.1
f6253634dc78: Pushed
9089f84dbbe9: Pushed
baccd3af13903: Pushed
latest: digest: sha256:27a71613c997b982480b2a481ebf9b72de675fe6ad354dd73c2730b3b2d2c17d size: 221
C:\Users\Administrator>
```

docker push ○○○○/r-preprocess-lectureと入力し、クリック



[CRAN](#)
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

[About R](#)
[R Homepage](#)
[The R Journal](#)

[Software](#)
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

[Documentation](#)
[Manuals](#)
[FAQs](#)
[Contributed](#)

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2020-10-10, Bunny-Wunnies Freak Out) [R-4.0.3.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are [available here](#). Please read about [new features and bug fixes](#) before filing corresponding feature requests or bug reports.
- Source code of older versions of R is [available here](#).
- Contributed extension [packages](#)

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the [R project homepage](#) for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN [mirror](#) nearest to you to minimize network load.

Submitting to CRAN



CRAN

Mirrors

[What's new?](#)

[Task Views](#)

[Search](#)

About R

[R Homepage](#)

[The R Journal](#)

Software

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

Documentation

[Manuals](#)

[FAQs](#)

[Contributed](#)

R for Windows

Subdirectories:

[base](#)

[contrib](#)

[old_contrib](#)

[Rtools](#)

Binaries for base distribution. This is what you want to [install R for the first time](#).

Binaries of contributed CRAN packages (for R >= 2.13.x; managed by Uwe Ligges). There is also information on [third party software](#) available for CRAN Windows services and corresponding environment and make variables.

Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).

Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the [R FAQ](#) and [R for Windows FAQ](#).

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.



CRAN

[Mirrors](#)

[What's new?](#)

[Task Views](#)

[Search](#)

[About R](#)

[R Homepage](#)

[The R Journal](#)

[Software](#)

[R Sources](#)

[R Binaries](#)

[Packages](#)

[Other](#)

[Documentation](#)

[Manuals](#)

[FAQs](#)

[Contributed](#)

R-4.0.3 for Windows (32/64 bit)

[Download R 4.0.3 for Windows](#)

(35 megabytes, 32/64 bit)

[Installation and other instructions](#)

[New features in this version](#)

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the [md5sum](#) of the .exe to the [fingerprint](#) on the master server. You will need a version of md5sum for windows: both [graphical](#) and [command line versions](#) are available.

Frequently asked questions

- [Does R run under my version of Windows?](#)
- [How do I update packages in my previous version of R?](#)
- [Should I run 32-bit or 64-bit R?](#)

Please see the [R FAQ](#) for general information about R and the [R Windows FAQ](#) for Windows-specific information.

Other builds

- Patches to this release are incorporated in the [r-patched snapshot build](#).
- A build of the development version (which will eventually become the next major release of R) is available in the [r-devel snapshot build](#).
- [Previous releases](#)

Note to webmasters: A stable link which will redirect to the current Windows binary release is
<<CRAN MIRROR>/bin/windows/base/release.html>.

Last change: 2020-10-10



セットアップに使用する言語の選択

×



インストール中に利用する言語を選んでください。

日本語



OK

キャンセル



情報

続行する前に以下の重要な情報を読みください。

セットアップを続行するには「次へ」をクリックしてください。

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

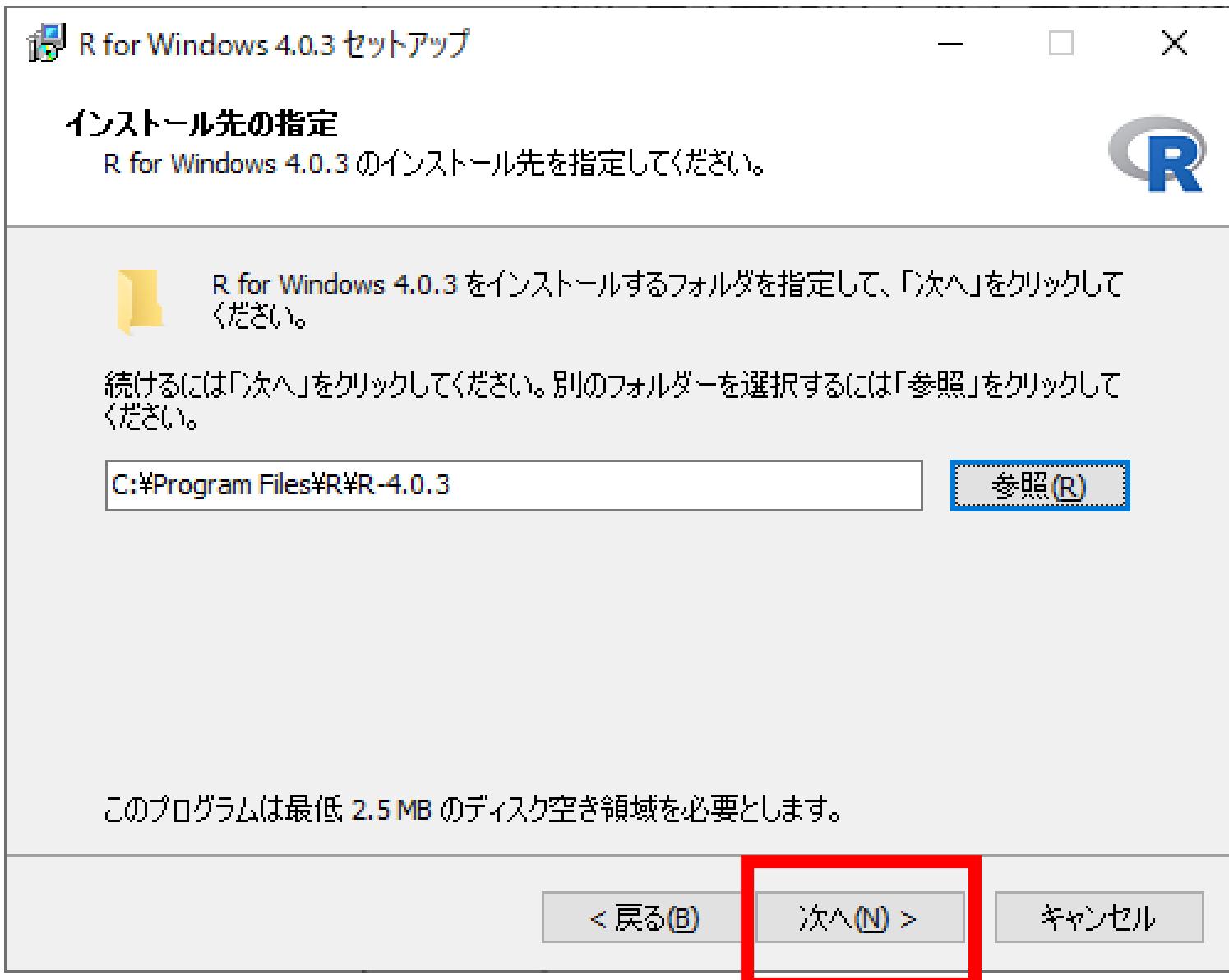
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

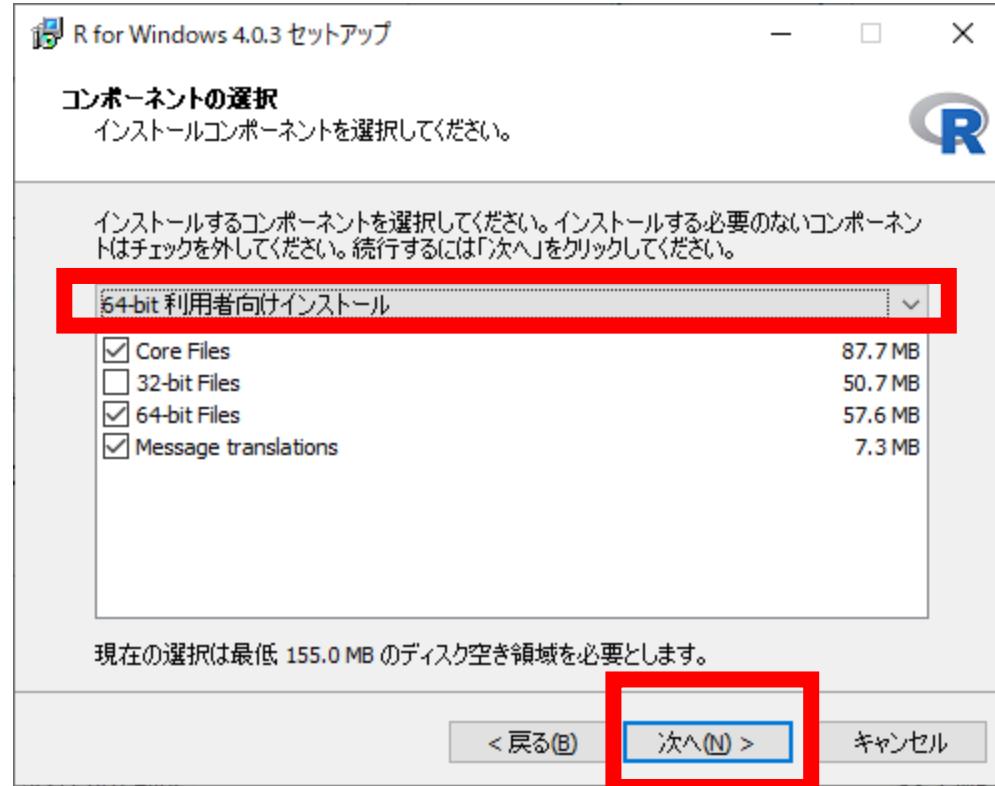
Preamble

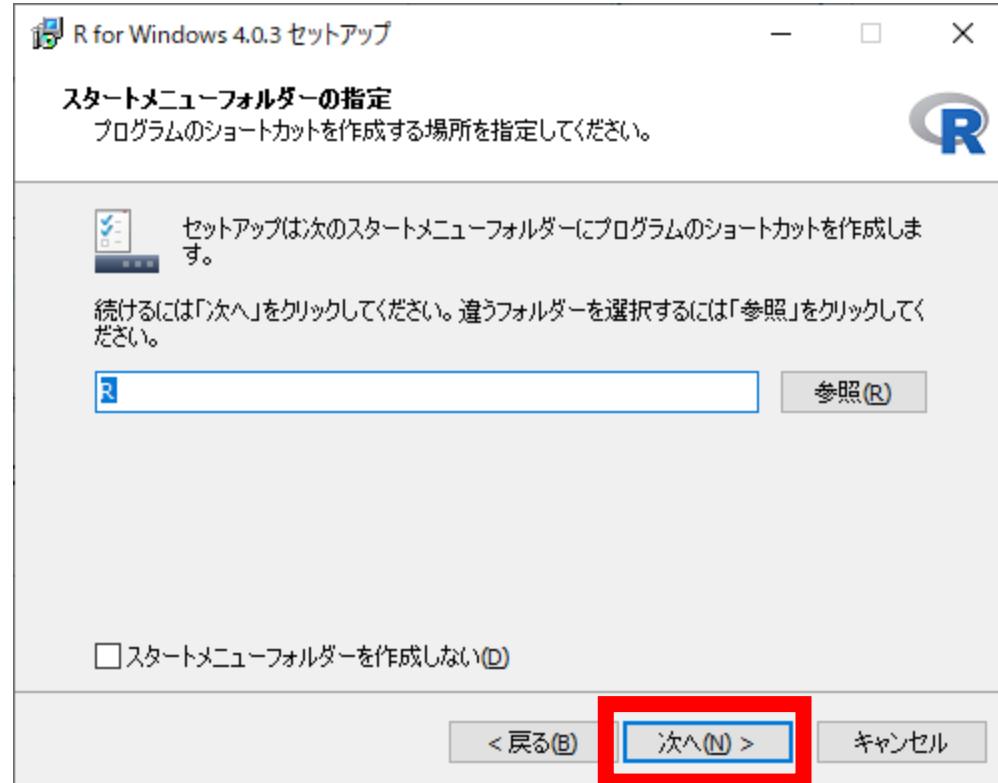
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public

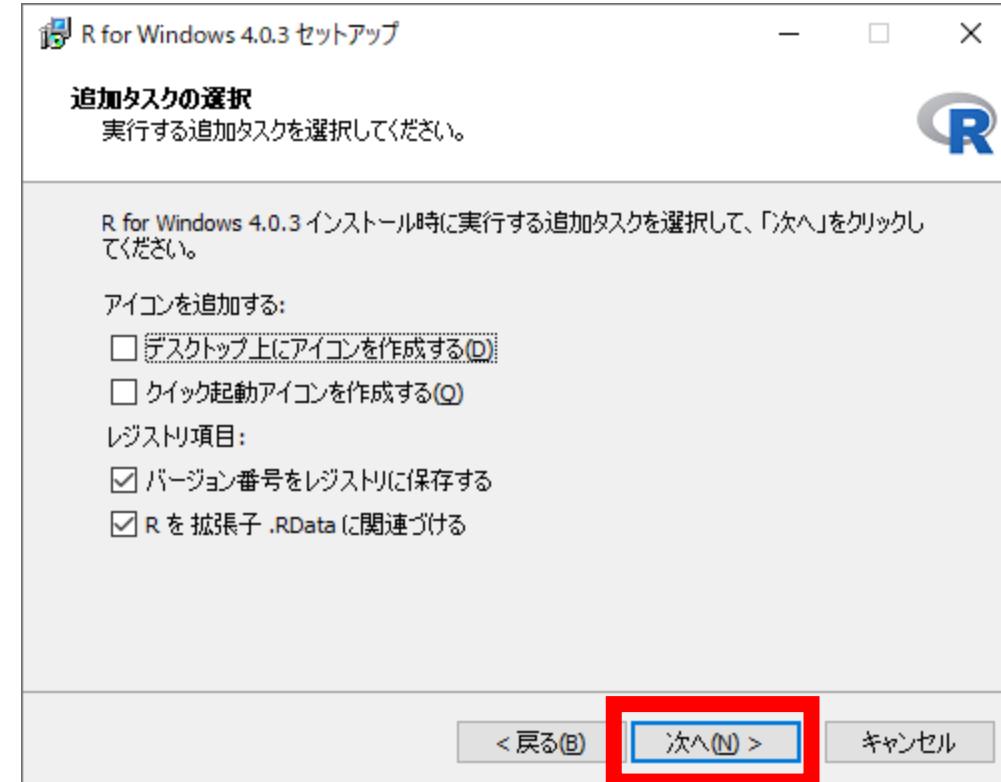
次へ(N) >

キャンセル











RStudio Desktop 1.3.1093 - Release Notes

1. Install R. RStudio requires R 3.0.1+.
2. Download RStudio Desktop. Recommended for your system:

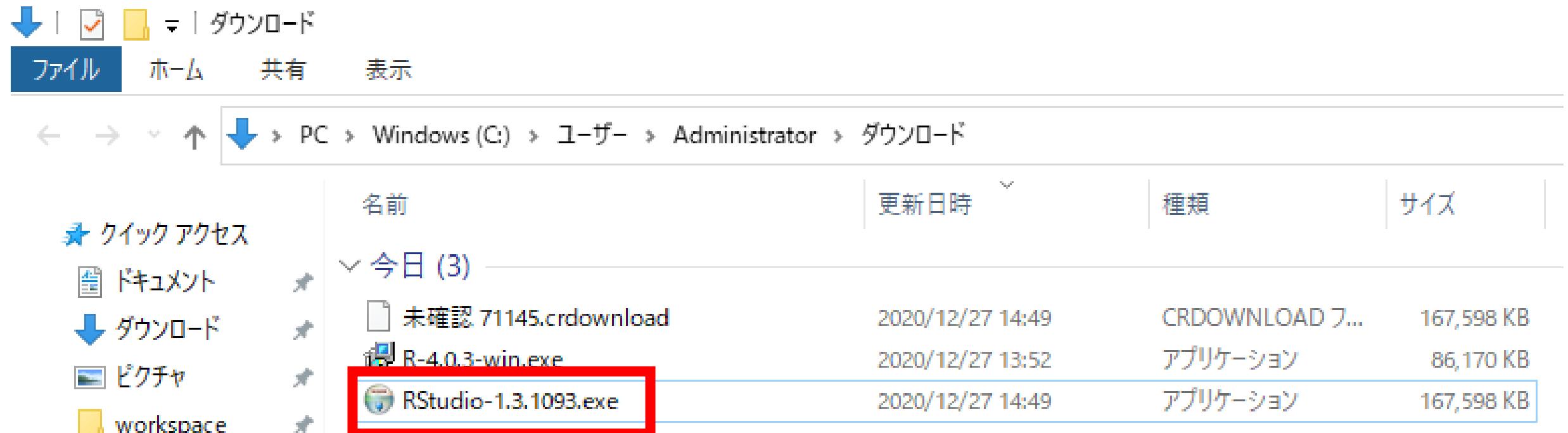


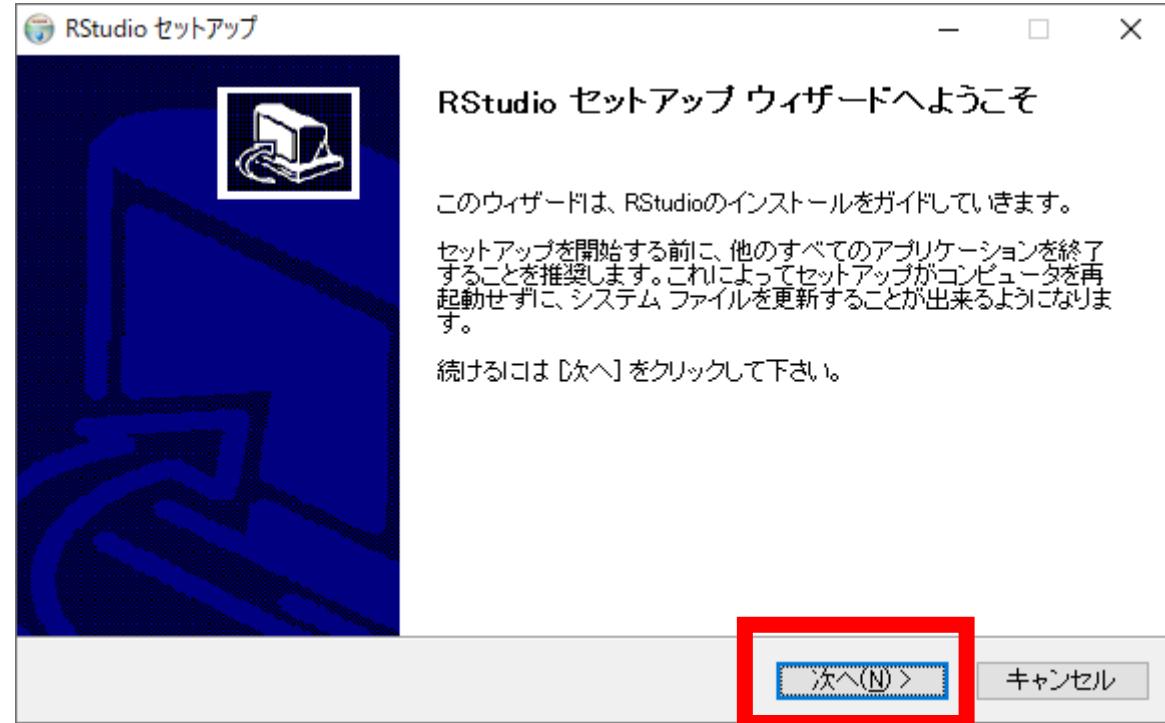
All Installers

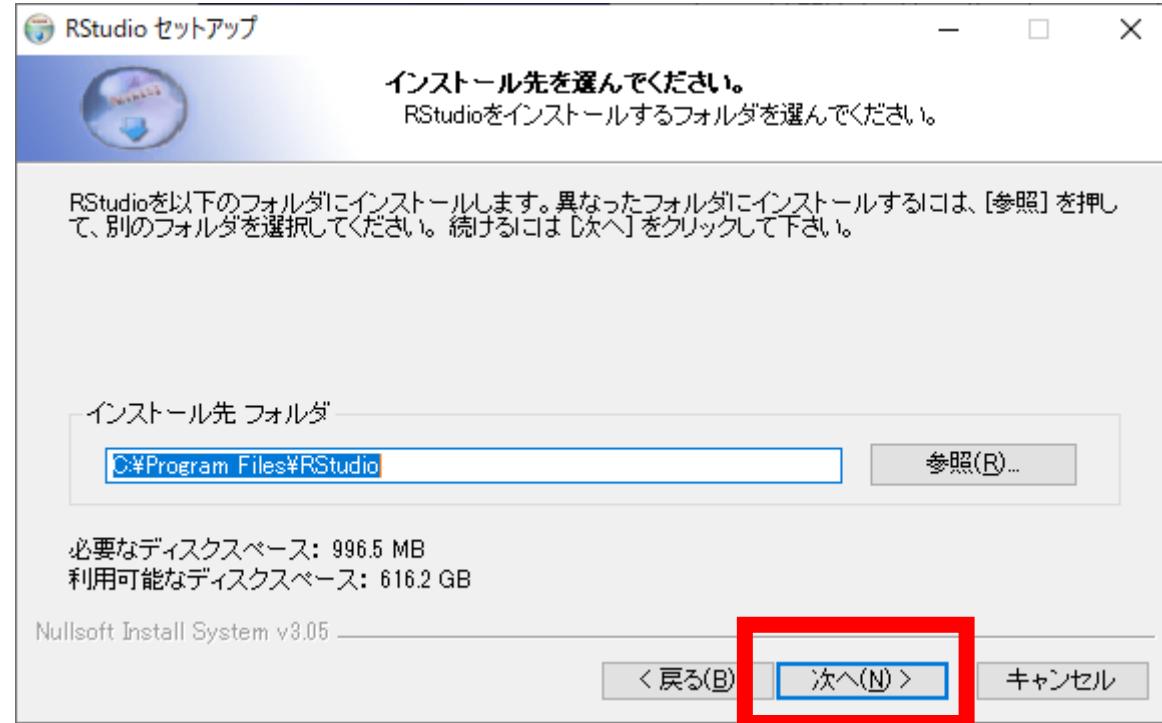
Linux users may need to import RStudio's public code-signing key prior to installation, depending on the operating system's security policy.

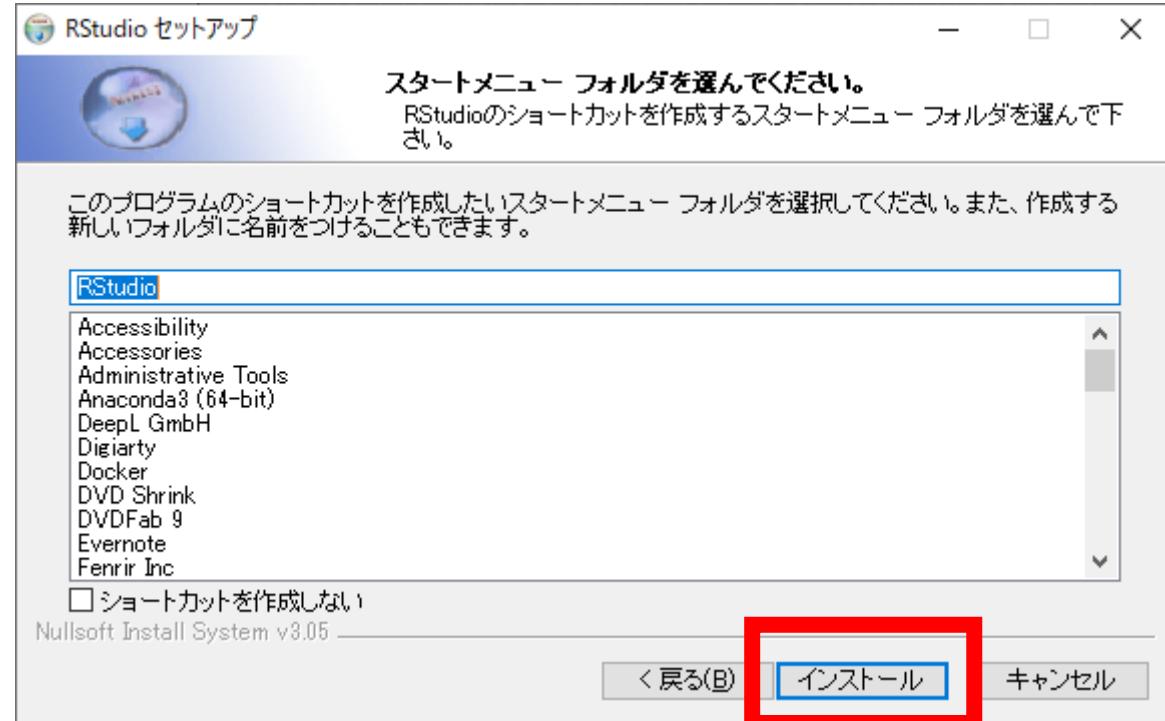
RStudio requires a 64-bit operating system. If you are on a 32 bit system, you can use an older version of RStudio.

| OS | Download | Size | SHA-256 |
|---------------------|--|-----------|----------|
| Windows 10/8/7 | RStudio-1.3.1093.exe | 171.62 MB | 62b9e60a |
| macOS 10.13+ | RStudio-1.3.1093.dmg | 148.66 MB | bdc4d3a4 |
| Ubuntu 16 | rstudio-1.3.1093-amd64.deb | 124.33 MB | 72f05048 |
| Ubuntu 18/Debian 10 | rstudio-1.3.1093-amd64.deb | 126.80 MB | ff222177 |
| Fedora 19/Red Hat 7 | rstudio-1.3.1093-x86_64.rpm | 146.96 MB | ed1f6ef8 |

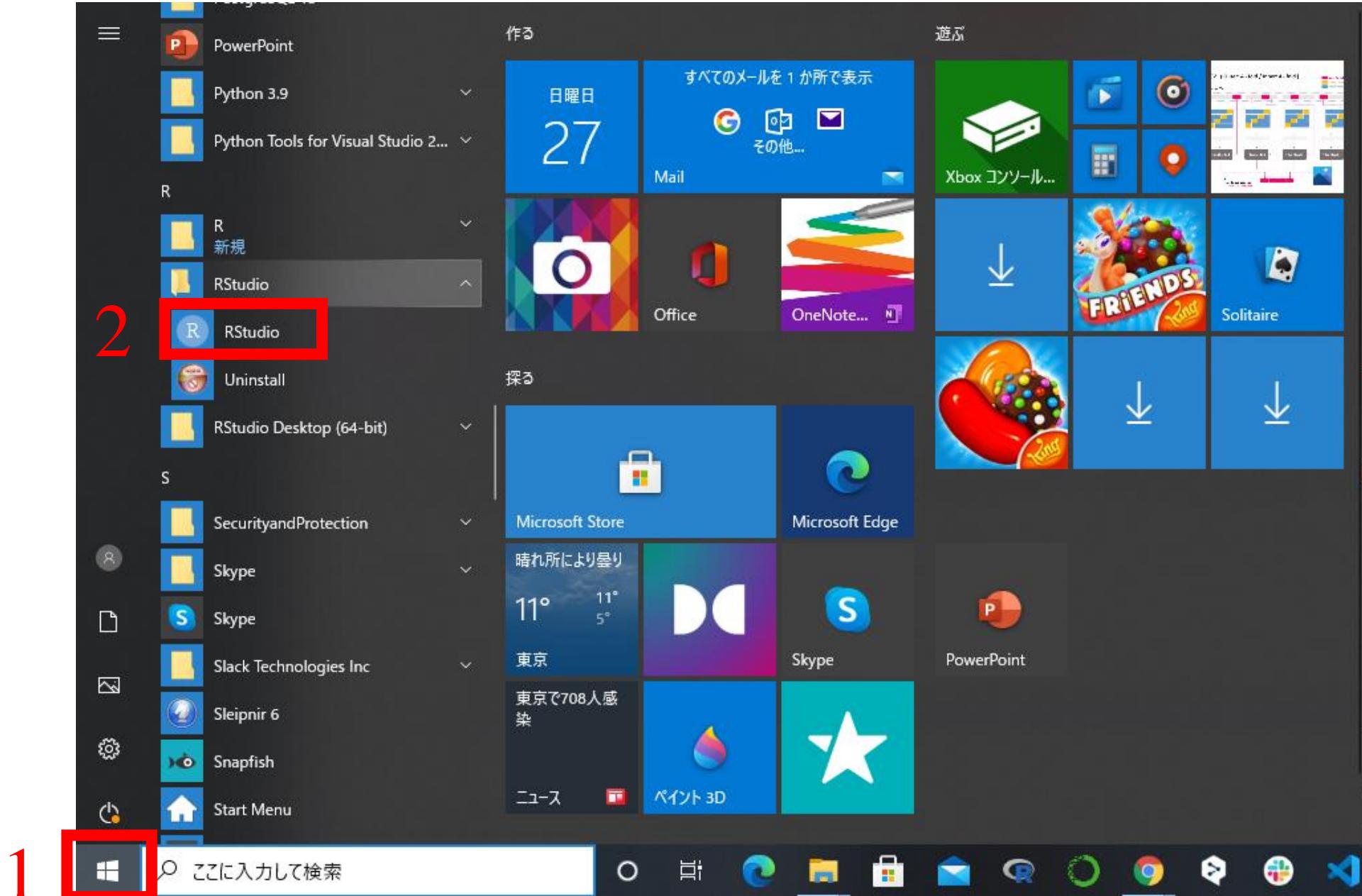












RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

+ Go to file/function Addins

Console Terminal Jobs

C:/Users/Administrator/workspace/

```
R version 3.6.2 (2019-12-12) -- "Dark and Stormy Night"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R は、自由なソフトウェアであり、「完全に無保証」です。
一定の条件下で、自由にこれを再配布することができます。
配布条件の詳細については、「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。
```

Environment History Connections Tutorial

Import Dataset

Global Environment

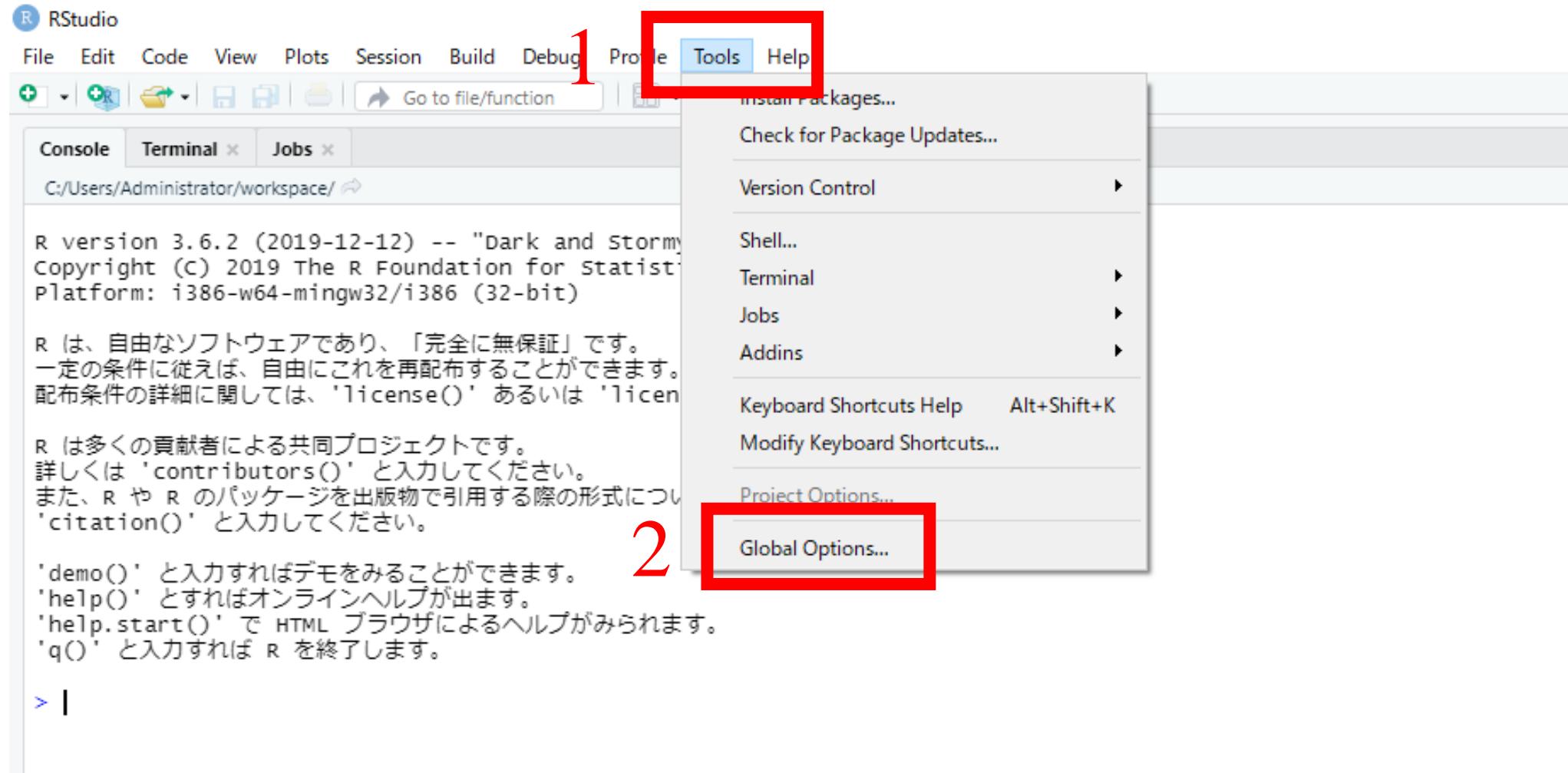
Environment is empty

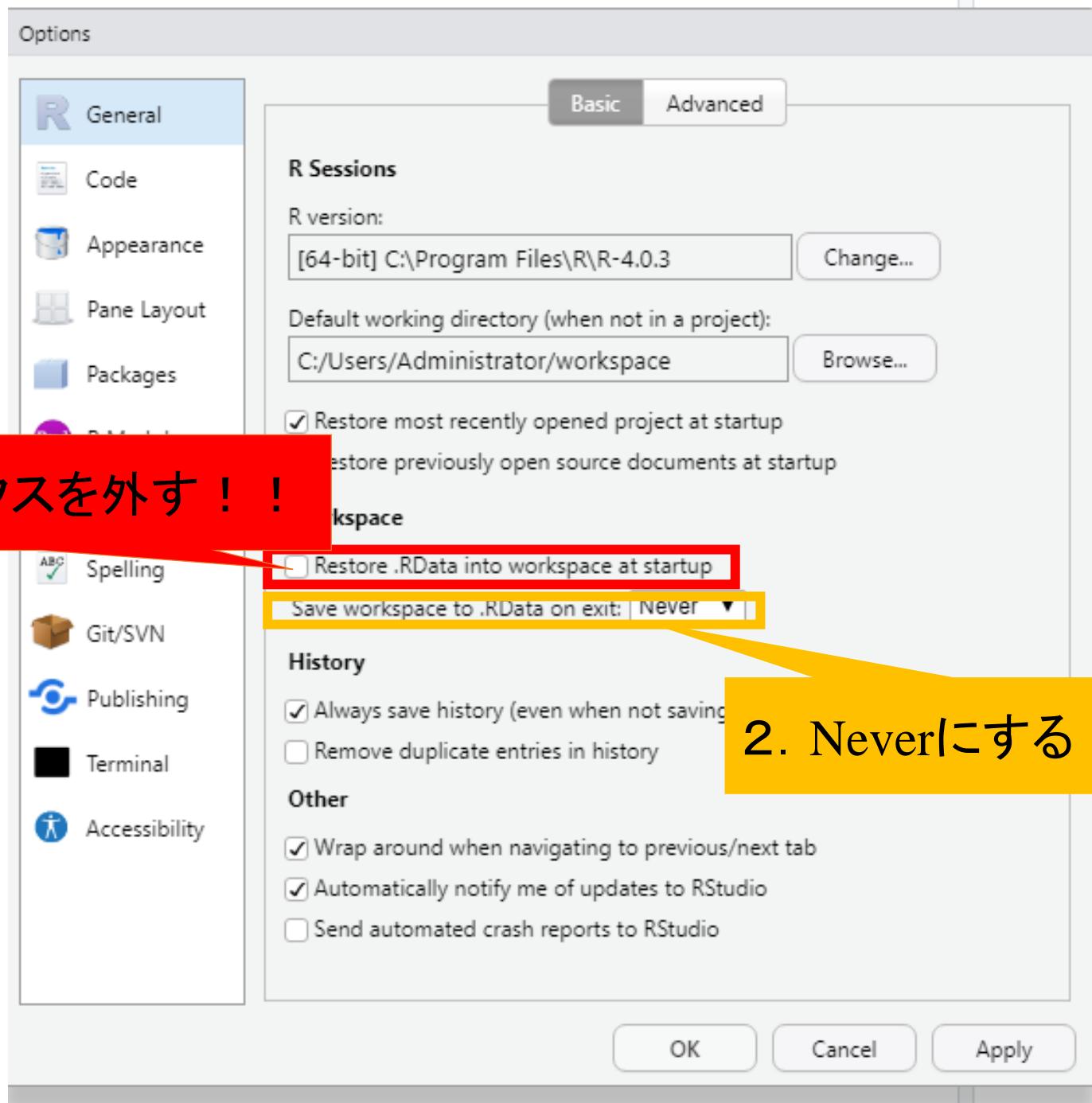
Files Plots Packages Help Viewer

Zoom Export

© 2021 shun

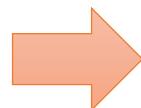
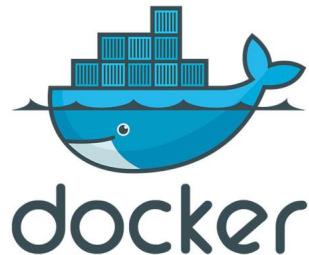
66



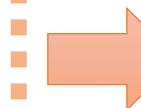


R前處理講座

1. 環境構築



2. baseR



3. tidyverse





| No. | 内容 |
|-----|------------|
| 1 | データ型 |
| 2 | ベクトル |
| 3 | 論理値ベクトルの処理 |
| 4 | リスト |
| 5 | データフレーム |
| 6 | 関数 |

| データ型 大分類 | データ型 | 意味 | 具体例 |
|-------------------------|--------------------|-----------------|------------------------------|
| 数値型 numeric (num) | integer (int) | 整数 | 1 200 9999 |
| | double (dbl) | 小数 | 1.36 3.14 33.9 |
| 文字列型 | character (chr) | 文字列 | “apple” “fish” “TOEIC” |
| | factor (fct) | 文字列を数値化したもの | apple fish TOEIC |
| | order (ord) | factorに順序を付けてもの | apple fish TOEIC |
| 論理值型 | logical (lgl) | 論理値 | TRUE FALSE |

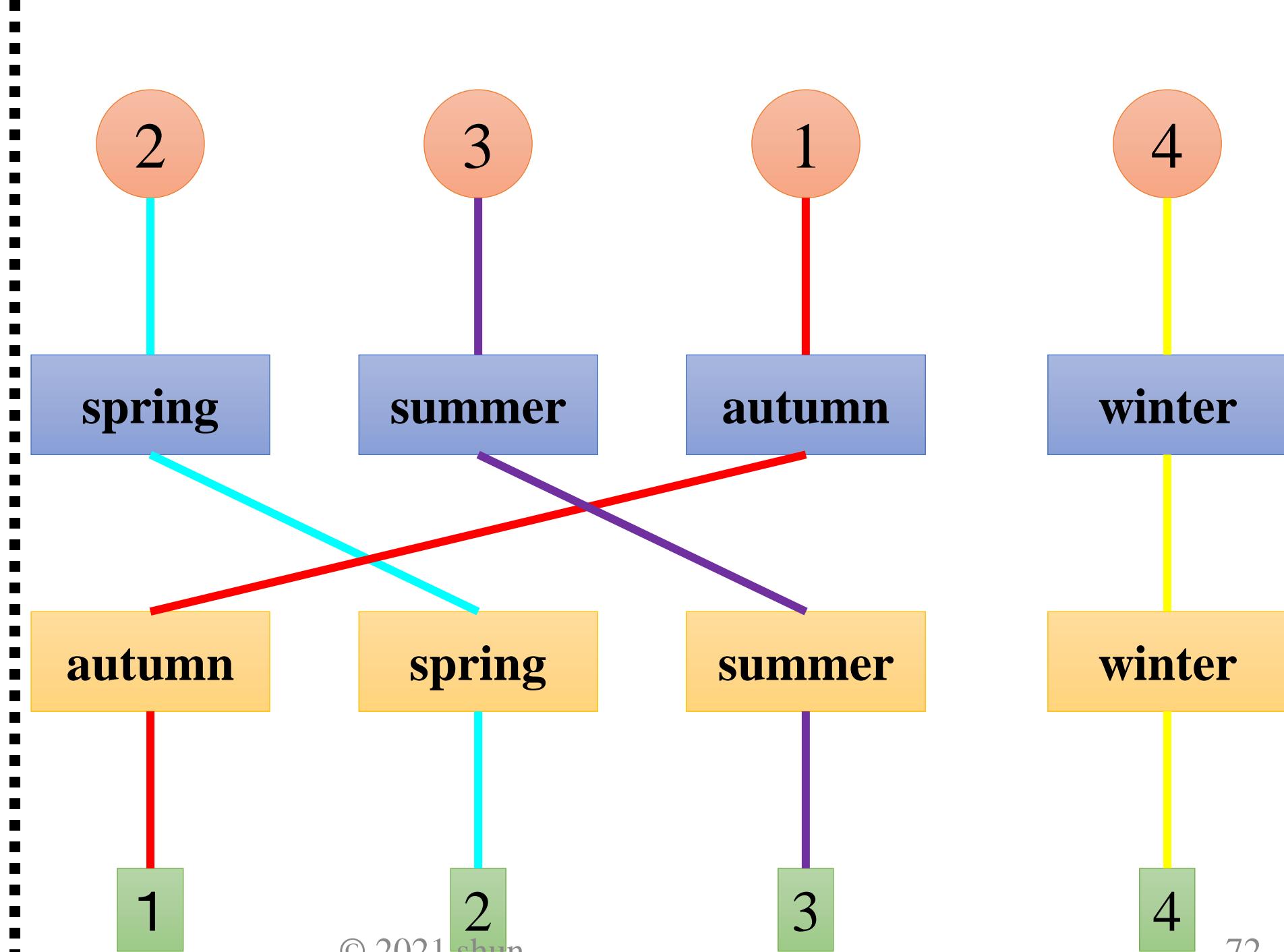
見た目はほとんど
変わらない

コンピュータの認識

表示

水準(Levels)

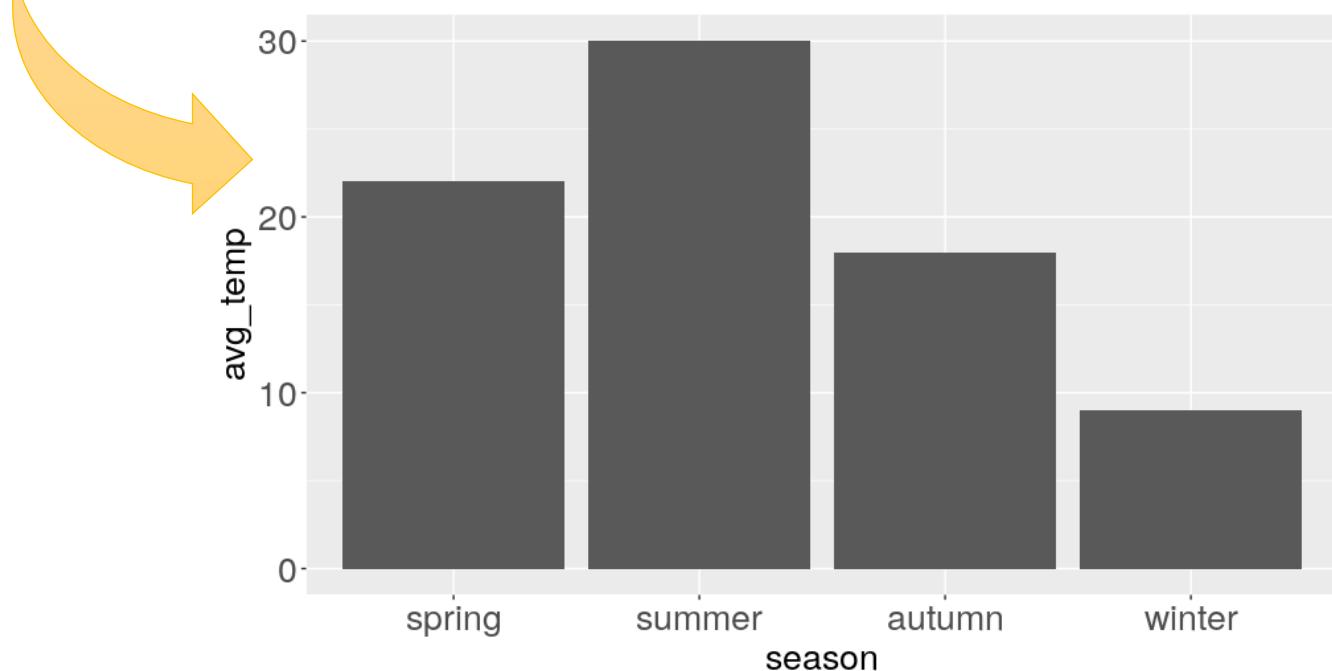
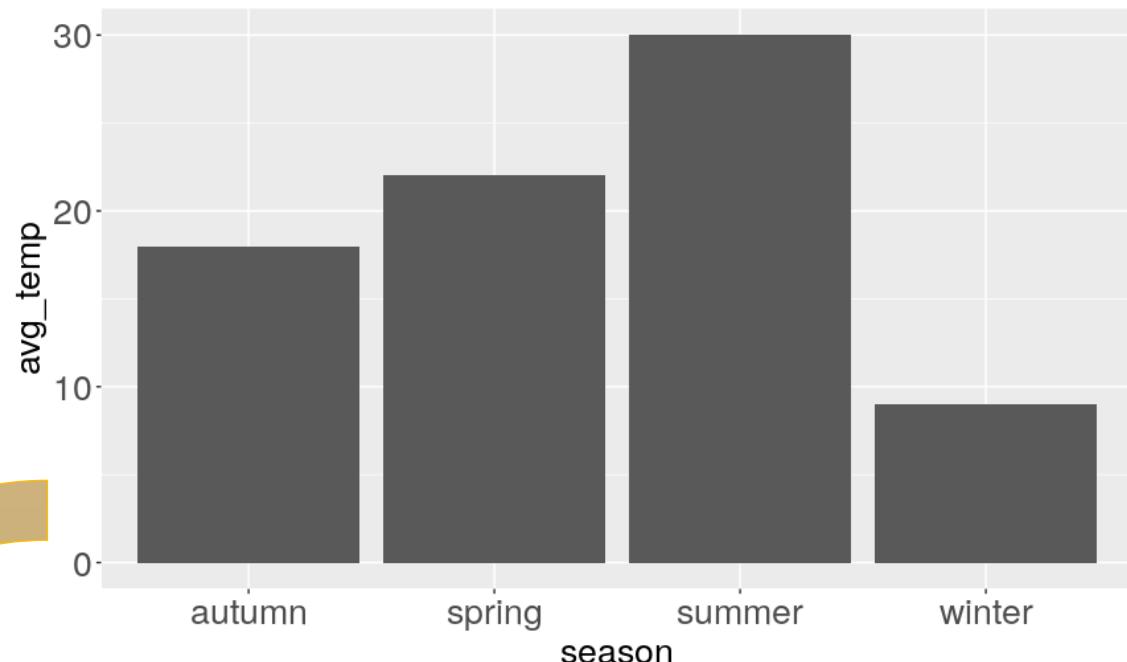
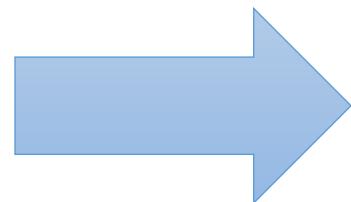
ラベル

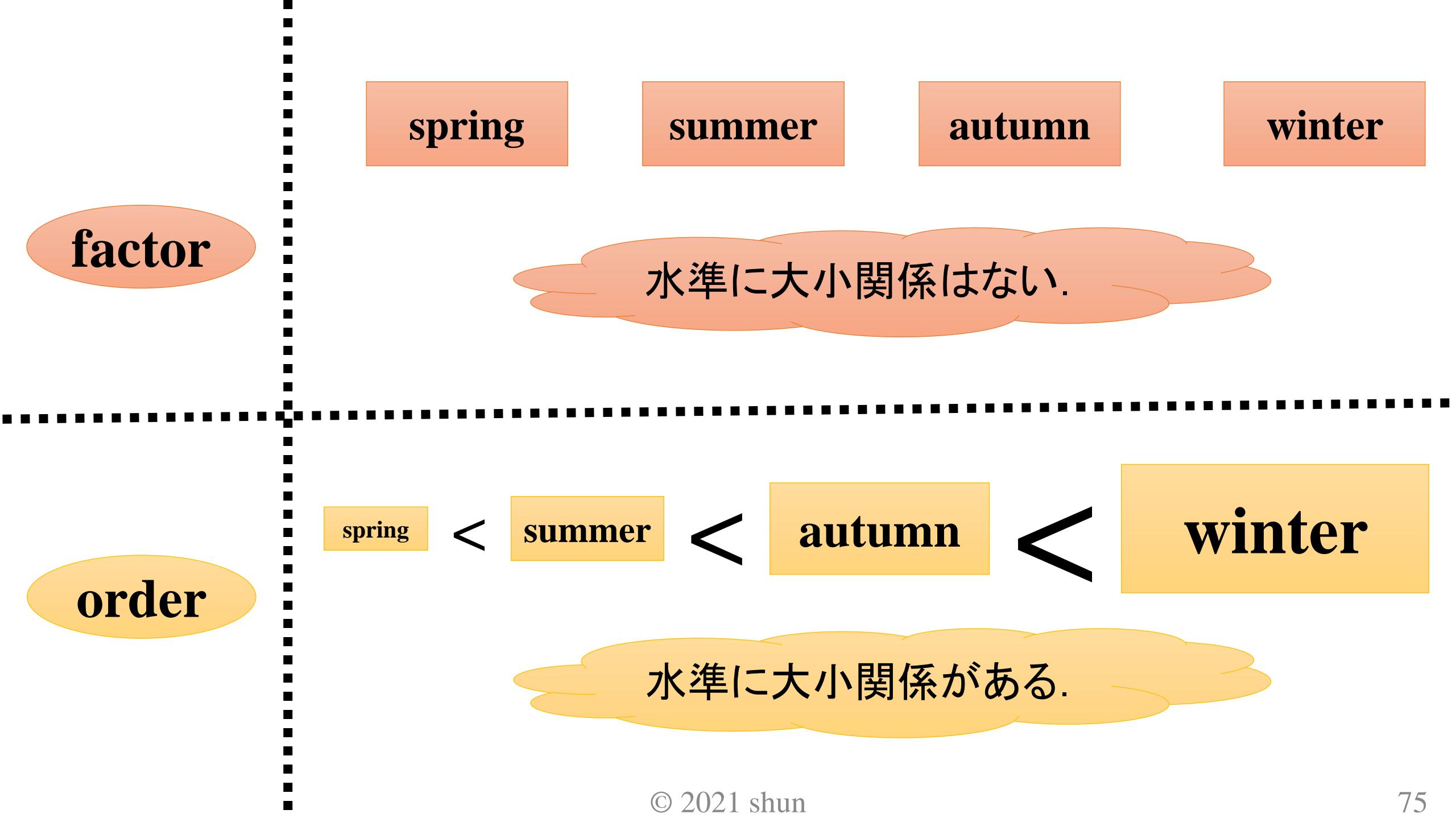


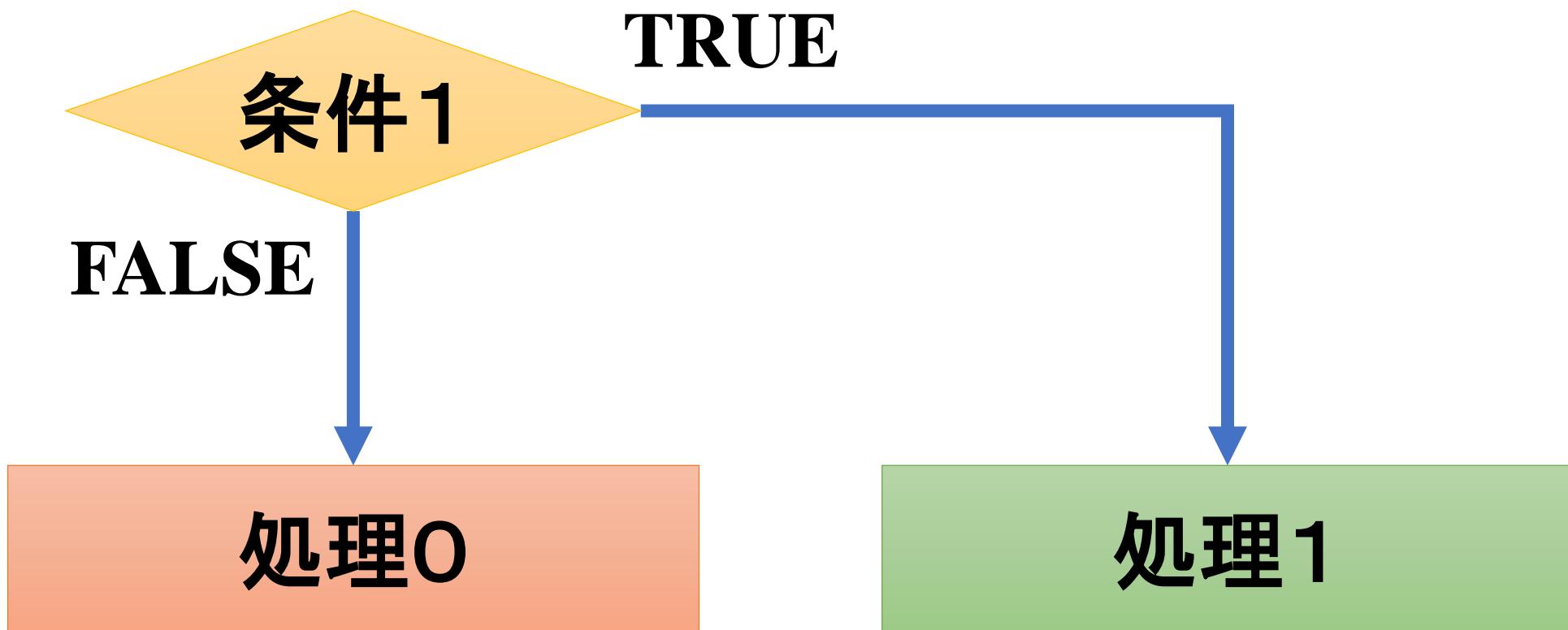
factorを利用する理由

1. メモリの節約
2. 機械学習をする際の前処理
3. 水準を利用した前処理

| season | avg_temp |
|--------|----------|
| spring | 22 |
| summer | 30 |
| autumn | 18 |
| winter | 9 |







ベクトル

ベクトルってなんですか？？



ベクトルってのは、同じデータ型のデータをひとまとめにした集合体のことだよ。



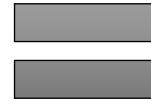
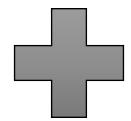
| | | | | | |
|------------|------|------|-------|-------|--------|
| int | 1 | 5 | 8 | 13 | 4 |
| chr | “a” | “B” | “CD” | “EFg” | “hiJK” |
| lgl | TRUE | TRUE | FALSE | TRUE | FALSE |

int_a

| |
|----|
| 1 |
| 5 |
| 8 |
| 13 |
| 4 |

int_b

| |
|----|
| 2 |
| 3 |
| 5 |
| 10 |
| 98 |



| |
|-----|
| 3 |
| 8 |
| 13 |
| 23 |
| 102 |

| |
|----|
| 1 |
| 5 |
| 8 |
| 13 |



| |
|---|
| 2 |
| 3 |

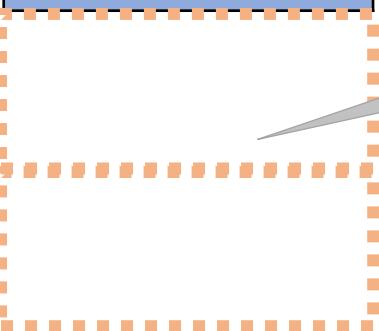


before

| |
|----|
| 1 |
| 5 |
| 8 |
| 13 |



| |
|---|
| 2 |
| 3 |



足りない...

after

| |
|----|
| 1 |
| 5 |
| 8 |
| 13 |



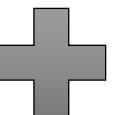
| |
|---|
| 2 |
| 3 |
| 2 |
| 3 |

リサイクル

上のベクトル
で補完！！

before

| |
|----|
| 1 |
| 5 |
| 8 |
| 13 |



| |
|---|
| 2 |
| 3 |
| 9 |

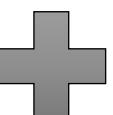
足りない...

| |
|---|
| 2 |
| 3 |
| 9 |
| 2 |

リサイクル

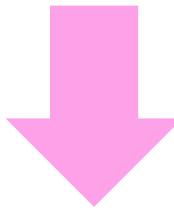
after

| |
|----|
| 1 |
| 5 |
| 8 |
| 13 |



上のベクトルで補完！！
収まりきらなかった部分
(3と9)は消える！！

| | | | | |
|---|---|---|----|---|
| 1 | 5 | 8 | 13 | 4 |
|---|---|---|----|---|



| | | | | |
|---|--|---|----|--|
| 1 | | 8 | 13 | |
|---|--|---|----|--|

1. インデックスで指定する.
2. 名前で指定する.
3. 論理値で指定する.

R インデックス

1

2

3

4

5

| | | | | |
|---|---|---|----|---|
| 1 | 5 | 8 | 13 | 4 |
|---|---|---|----|---|

Python インデックス

0

1

2

3

4

| | | | | |
|---|---|---|----|---|
| 1 | 5 | 8 | 13 | 4 |
|---|---|---|----|---|

T

F

T

T

F



| | | | | |
|---|--|---|----|--|
| 1 | | 8 | 13 | |
|---|--|---|----|--|

| No. | 関数 | 意味 |
|-----|----------|--------|
| 1 | sum | 総和 |
| 2 | mean | 平均 |
| 3 | median | 中央値 |
| 4 | max | 最大値 |
| 5 | min | 最小値 |
| 6 | quantile | クオンタイル |
| 7 | var | 分散 |
| 8 | sd | 標準偏差 |

a,b,n:int型スカラー
x,y,z:num型スカラー
vec:ベクトル

| 関数 | 使い方 | 意味 |
|-----|-----------------------|-------------------|
| : | a : b | aからbまで等差が1の等差ベクトル |
| seq | seq(x, y, length = n) | xからyまでの長さnの等差ベクトル |
| | seq(x, y, by = z) | xからyまで等差がzの等差ベクトル |
| rep | rep(vec, times = n) | vecをn回繰り返す. |
| | rep(vec, length = n) | vecを長さnになるまで繰り返す. |
| | rep(vec, each = n) | vecの要素ごとにn回繰り返す. |

まとめ

1. ベクトルってなに？？
2. ベクトルの四則演算
3. ベクトルのリサイクル
4. ベクトルの要素抽出
5. ベクトルの便利関数
6. 規則的なベクトルの作成

論理値ベクトルの処理

論理値ベクトルの処理ってなんですか？？



論理値ベクトルは、比較演算子によって条件の判定結果として作成されるんだ。

そして、論理値ベクトルに対し、論理演算子を使用して計算をすることで、様々な条件を論理値ベクトルで表現できるんだ。

この論理値ベクトルの作成と計算を総称して論理値ベクトルの処理って呼んでいるよ。

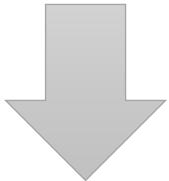
？？？



コンテンツ

- 1.論理値ベクトルの作成
- 2.論理値ベクトルの計算

どのように論理値ベクトルを作成するのか？？



条件を満たしたかどうかでTRUEと
FALSEを判定し、その判定結果を論理値
ベクトルとする。

気温は5°C以上か？

季節は春か？

年齢は25歳以下か？

※ xとyは、ベクトルを表す。

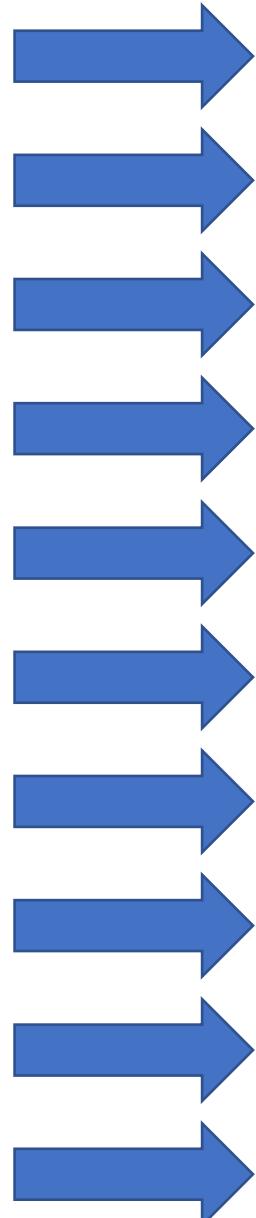
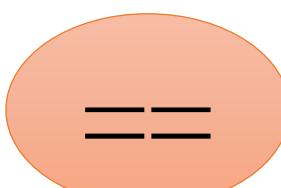
| 比較演算子 | 構文 | 意味 |
|--------------------|------------------------|--------------|
| <code>==</code> | <code>x == y</code> | xとyは、等しいか？ |
| <code>!=</code> | <code>x != y</code> | xとyは、等しくないか？ |
| <code>>=</code> | <code>x >= y</code> | xはy以上か？ |
| <code>></code> | <code>x > y</code> | xはyより大きいか？ |
| <code><=</code> | <code>x <= y</code> | xはy以下か？ |
| <code><</code> | <code>x < y</code> | xはyより小さいか？ |
| <code>%in%</code> | <code>x %in% y</code> | xはyに含まれているか？ |

X

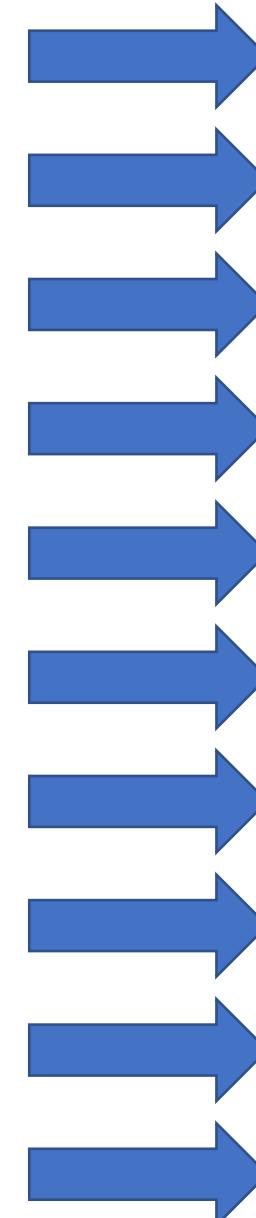
| |
|----|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |

y

| |
|---|
| 1 |
| 3 |
| 4 |
| 7 |
| 2 |
| 1 |
| 3 |
| 4 |
| 7 |
| 2 |



| |
|------------|
| 1と1は等しいか？ |
| 2と3は等しいか？ |
| 3と4は等しいか？ |
| 4と7は等しいか？ |
| 5と2は等しいか？ |
| 6と1は等しいか？ |
| 7と3は等しいか？ |
| 8と4は等しいか？ |
| 9と7は等しいか？ |
| 10と2は等しいか？ |



| |
|-------|
| TRUE |
| FALSE |
| FASLE |
| FALSE |
| FALSE |
| FALSE |

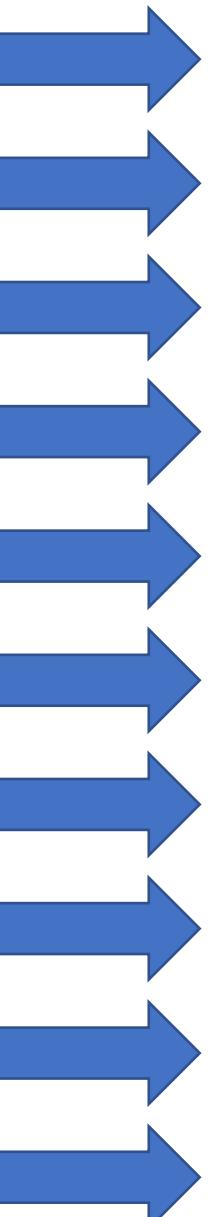
y

| |
|---|
| 1 |
| 3 |
| 4 |
| 7 |
| 2 |

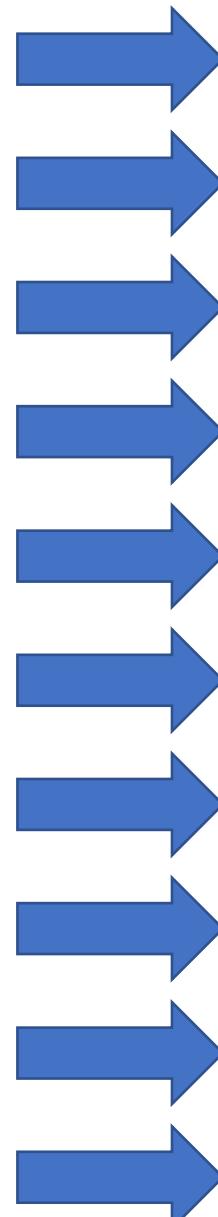
% in %

X

| |
|----|
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |



| |
|---------------|
| 1は, yの中にあるか？ |
| 2は, yの中にあるか？ |
| 3は, yの中にあるか？ |
| 4は, yの中にあるか？ |
| 5は, yの中にあるか？ |
| 6は, yの中にあるか？ |
| 7は, yの中にあるか？ |
| 8は, yの中にあるか？ |
| 9は, yの中にあるか？ |
| 10は, yの中にあるか？ |



| |
|-------|
| TRUE |
| FALSE |
| FALSE |
| TRUE |
| FALSE |
| FALSE |

論理値ベクトルの計算



論理値ベクトルは、組み合わせたり、反転させたり、TRUEの有無を確認したり、いろいろな計算方法があるんだ。

論理値ベクトルを組み合わせるというのは、2つの論理値ベクトルに対して、とある計算をすることで、1つの新たな論理値ベクトルを作成することをいうよ。

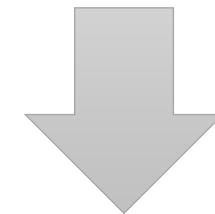
論理値ベクトルの計算?
どういうこと?



キャベツのサイズ

| |
|----|
| 7 |
| 13 |
| 5 |
| 25 |
| 38 |
| 57 |
| 18 |
| 9 |
| 32 |
| 28 |

10以上30以下のサイズを
抽出したい！！



- ① 10以上のキャベツかどうかを判定する.
- ② 30以下のキャベツかどうかを判定する.
- ③ ①と②を両方満たしたキャベツを市場に出せると判定する.

キャベツのサイズ

| |
|----|
| 7 |
| 13 |
| 5 |
| 25 |
| 38 |
| 57 |
| 18 |
| 9 |
| 32 |
| 28 |

①10以上か？

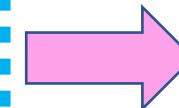
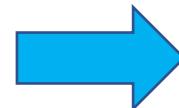
| |
|---|
| F |
| T |
| F |
| T |
| T |
| T |
| T |
| F |
| T |
| T |

②30以下か？

| |
|---|
| T |
| T |
| T |
| T |
| F |
| F |
| T |
| T |
| F |

③市場出荷OKか？

| |
|---|
| F |
| T |
| F |
| T |
| F |
| F |
| T |
| F |
| F |
| T |



※ x と y は、論理値ベクトルを表す。

| 論理演算子 | 構文 | 意味 | 返り値 |
|-------|--------------------|---|---------|
| & | $x \& y$ | 両方TRUE \Rightarrow TRUE それ以外 \Rightarrow FALSE | 論理値ベクトル |
| | $x y$ | 1つがTRUE \Rightarrow TRUE 全部FALSE \Rightarrow FALSE | 論理値ベクトル |
| xor | $\text{xor}(x, y)$ | 1つがTRUE, 1つがFALSE \Rightarrow TRUE 両方ともTRUEもしくはFALSE \Rightarrow FALSE | 論理値ベクトル |
| ! | $! x$ | TRUE \Rightarrow FALSE FALSE \Rightarrow TRUE | 論理値ベクトル |
| any | $\text{any}(x)$ | 1つ以上TRUE \Rightarrow TRUE 全部FALSE \Rightarrow FALSE | 論理値スカラー |
| all | $\text{all}(x)$ | 全部TRUE \Rightarrow TRUE 1つ以上FALSE \Rightarrow FALSE | 論理値スカラー |

まとめ

- 1.論理値ベクトルの作成
- 2.論理値ベクトルの計算

リスト

リストってなんですか？？



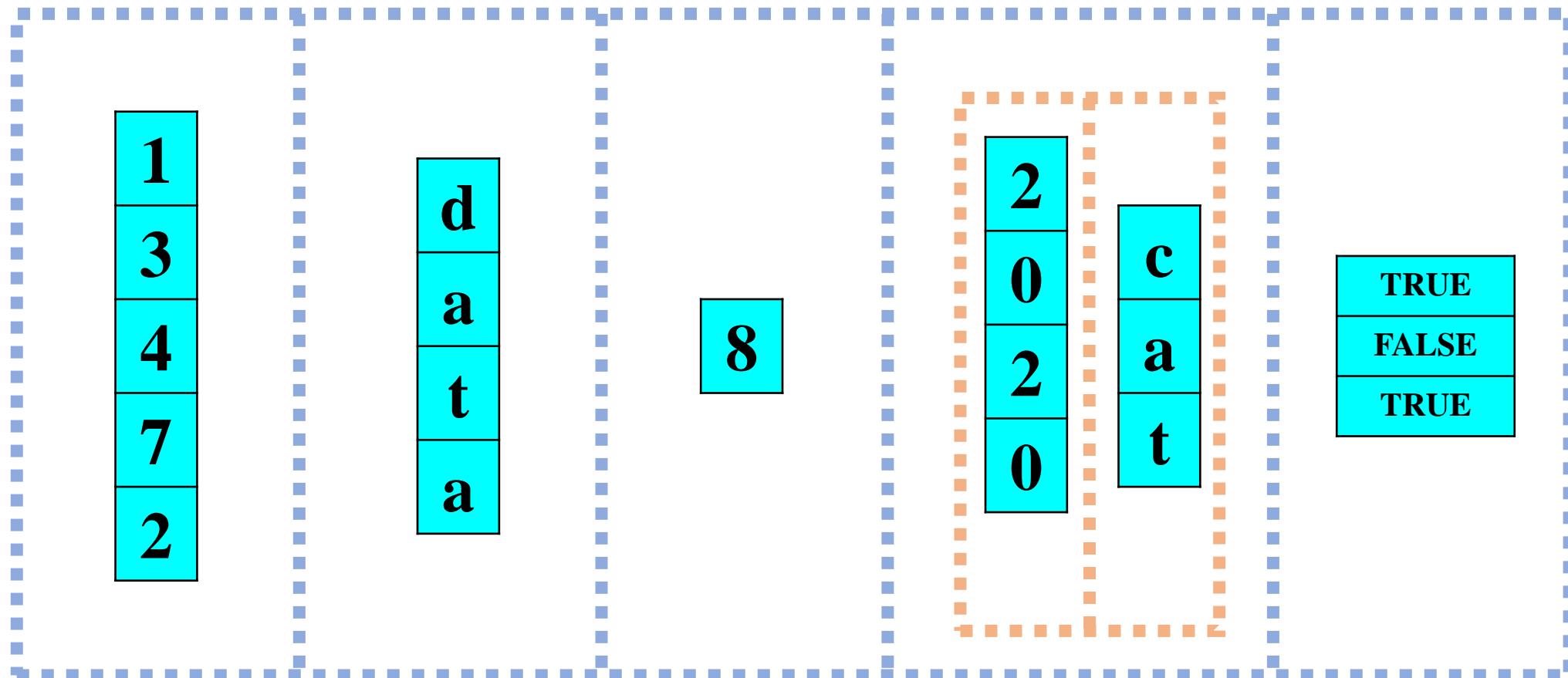
リストっていうのはベクトルと同様、データをひとまとめにした集合体のことだよ。

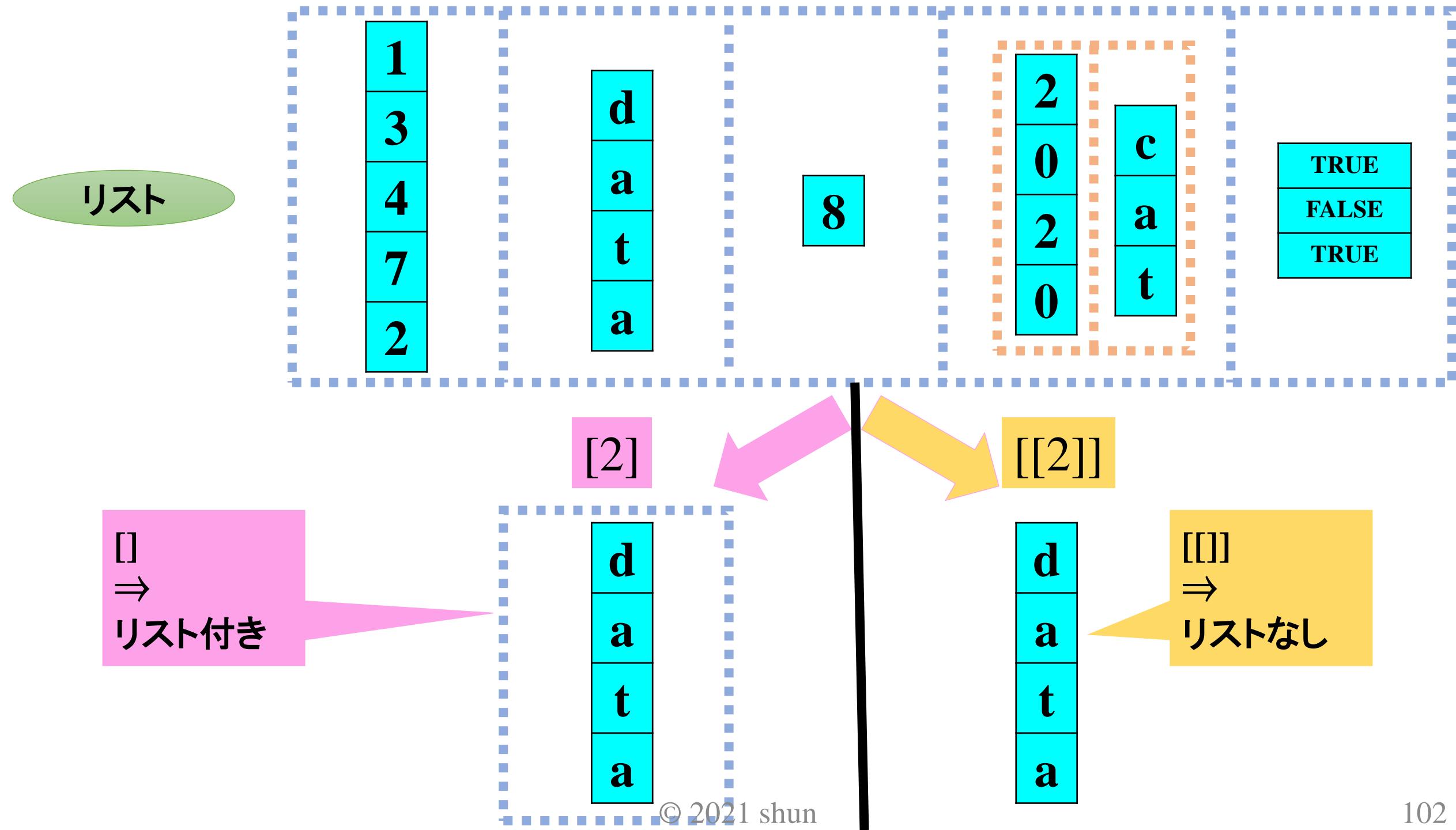
リストは、ベクトルと違って、同じデータ型じゃなくてもOKなんだ。

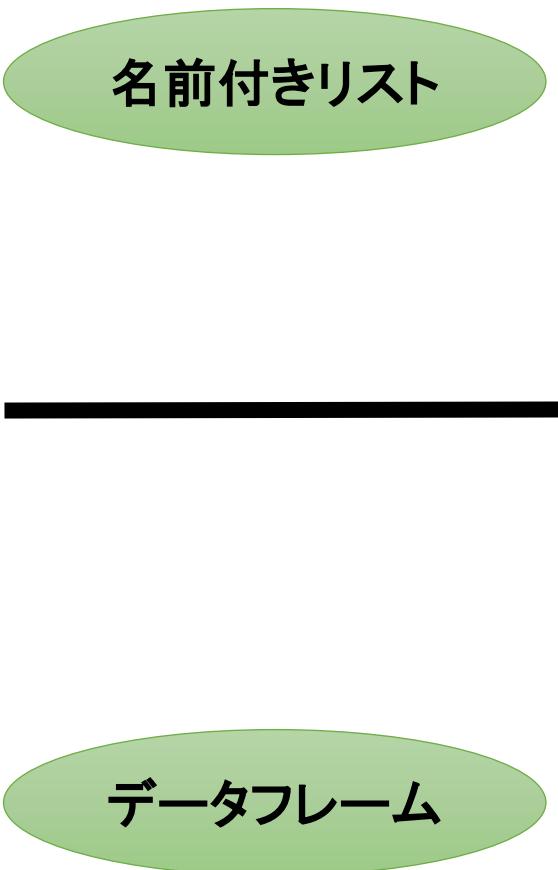
ベクトル



リスト







| one | two | three | four | five |
|-----------|----------------------|-------|--------------|---------------------|
| [4, 7, 2] | ['d', 'a', 't', 'a'] | [8] | [2, 0, 2, 0] | [TRUE, FALSE, TRUE] |



| one | two | three | four | five |
|-----------|-----------------|---------------------------|-----------------------------|---------------------|
| [4, 7, 2] | ['a', 'b', 'c'] | ['asuka', 'yuuki', 'mai'] | ['japan', 'china', 'india'] | [TRUE, FALSE, TRUE] |

まとめ

- 1.リストの要素抽出
- 2.名前付きリスト

データフレーム

データフレームって何ですか？？



データフレームは行数がどの列も同じ名前付きリストのことだよ。

| season | avg_temp |
|--------|----------|
| spring | 22 |
| summer | 30 |
| autumn | 18 |
| winter | 9 |

名前付きリスト

| one | two | three | four | five |
|-------------|------------------|-------|------------------|-------------|
| 4 7 2 | d a t a | 8 | 2 0 2 0 | c a t |
| | | | | |
| | | | | |
| | | | | |

データフレーム

| one | two | three | four | five |
|-------------|-------------|-----------------------|-------------------------|-----------------------|
| 4 7 2 | a b c | asuka yuuki mai | japan china india | TRUE FALSE TRUE |
| | | | | |
| | | | | |
| | | | | |

まとめ

1. データフレームってなに？？
2. データフレームの要素抽出
3. tibble

関数

関数ってなんですか？？

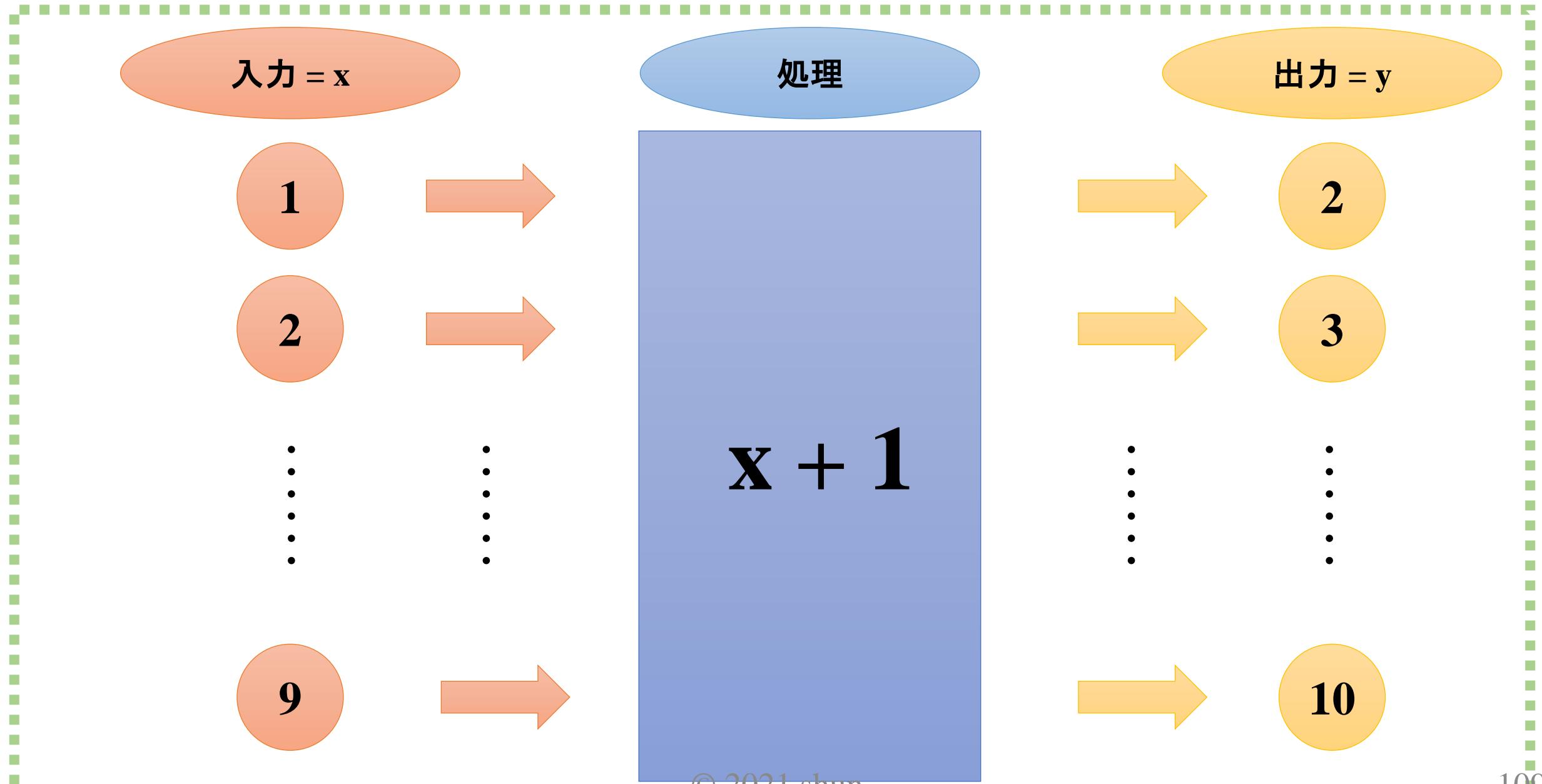


関数は、入力に対してあらかじめ定めた処理を適用することで出力を返す命令のことだよ。

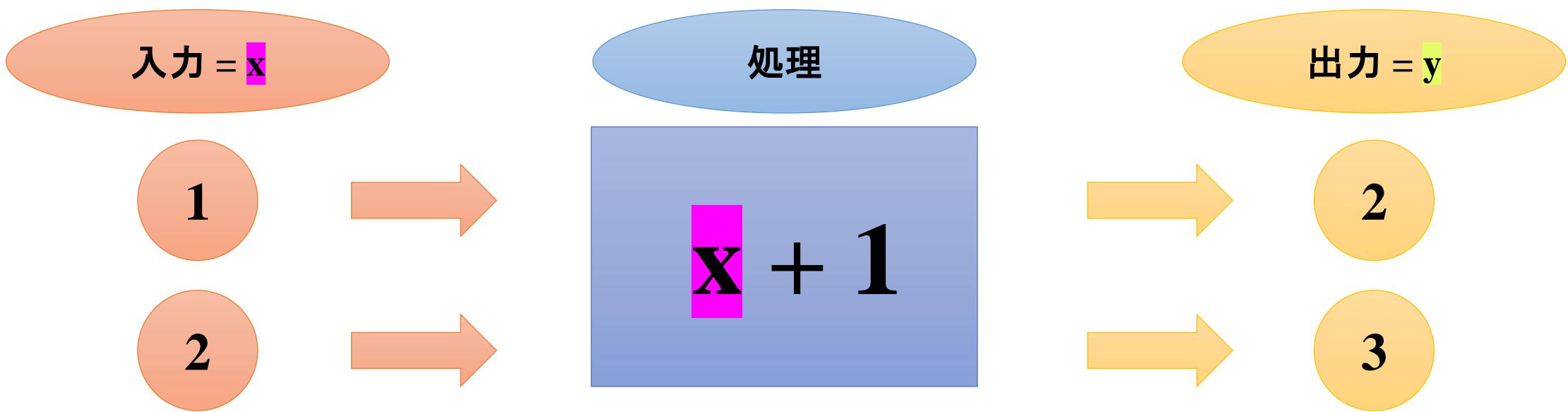
？？



関数



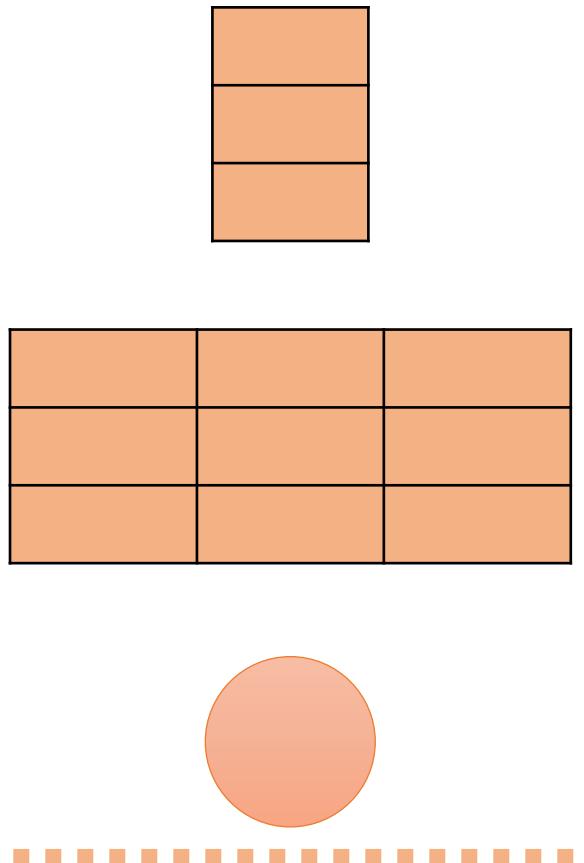
関数 : add_1



```
add_1 = function(x){  
    y = x + 1  
    return(y)  
}
```

関数

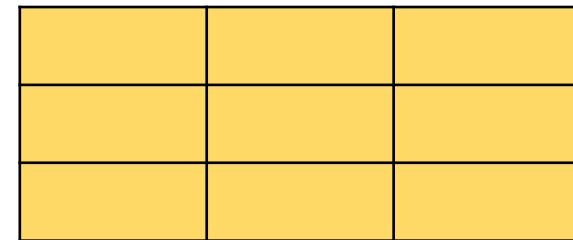
入力



処理



出力



関数を使う理由

関数の書き方についてはわかりました。



でも、関数って使わなくてもよくないですか？
関数使わなくても書けそうだし。 . .



関数を使う理由は主に2点。

- ・コードをきれいにするため。
- ・コードの修正を簡単にするため。

```
# 散布図出力 関数を使用する場合  
make_scatter = function(df, type){  
  df_Species =  
    df %>%  
    tibble() %>%  
    filter(Species == type)  
  scatter =  
    df_Species %>%  
    ggplot(  
      mapping = aes(x = Sepal.Length, y =  
Petal.Length)  
    ) +  
    geom_point() +  
    geom_smooth(method = "lm", se =  
FALSE) +  
    theme(  
      axis.text = element_text(size = 25),  
      axis.title = element_text(size = 25)  
    )  
  return(scatter)  
}
```

```
make_scatter(iris, "setosa")  
make_scatter(iris, "versicolor")  
make_scatter(iris, "virginica")
```

関数の 外部ファイル化

外部ファイル化！！

```
# 散布図出力 関数を使用する場合  
make_scatter = function(df, type){  
  df_Species =  
    df %>%  
    tibble() %>%  
    filter(Species == type)  
  scatter =  
    df_Species %>%  
    ggplot(  
      mapping = aes(x = Sepal.Length, y =  
Petal.Length)  
    ) +  
    geom_point() +  
    geom_smooth(method = "lm", se =  
FALSE) +  
    theme(  
      axis.text = element_text(size = 25),  
      axis.title = element_text(size = 25)  
    )  
  return(scatter)  
}
```

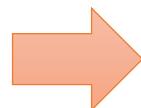
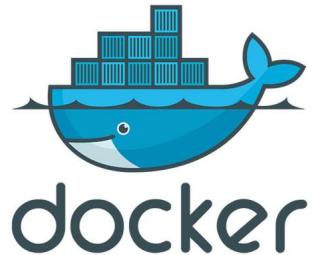
```
make_scatter(iris, "setosa")  
make_scatter(iris, "versicolor")  
make_scatter(iris, "virginica")
```

まとめ

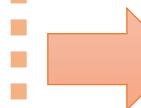
1. 関数ってなに？？
2. 関数の作成
3. 関数を使う理由
4. 関数の外部ファイル化

R前處理講座

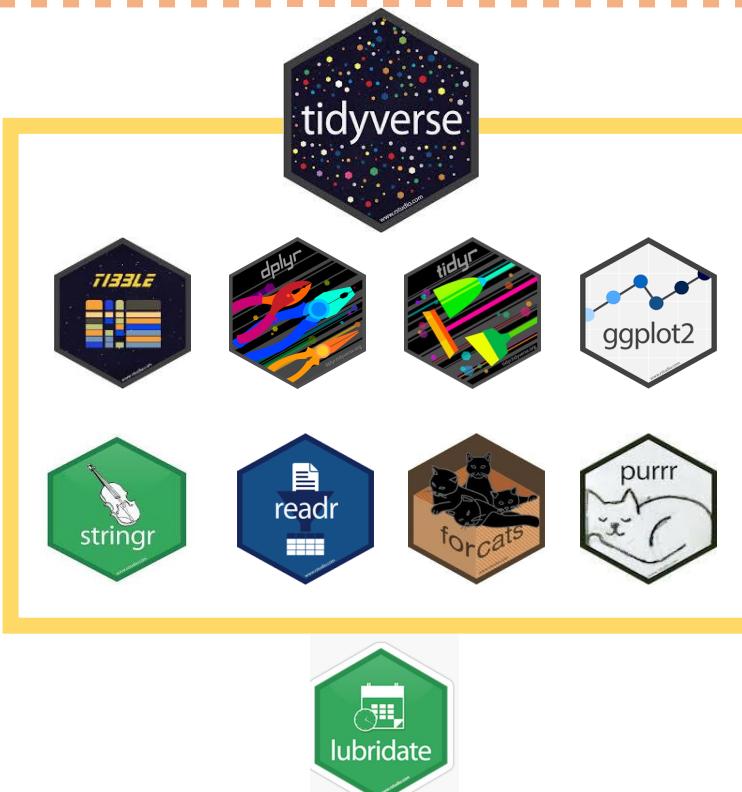
1. 環境構築



2. baseR



3. tidyverse





時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|---|-------------|
| 1 | tibble | A dark blue hexagonal logo for tibble, featuring the word "TIBBLE" at the top and a stylized bar chart icon below it. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagonal logo for dplyr, featuring a colorful rocket ship icon. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagonal logo for tidyr, featuring a colorful geometric pattern of lines and shapes. | tidyデータ処理 |
| 4 | ggplot2 | A light gray hexagonal logo for ggplot2, featuring a line plot with three data points. | 可視化 |
| 5 | stringr | A green hexagonal logo for stringr, featuring a white violin icon. | 文字列処理 |
| 6 | readr | A blue hexagonal logo for readr, featuring a document icon. | 入出力処理 |
| 7 | forcats | A brown hexagonal logo forforcats, featuring two black cats inside a cardboard box. | ファクター処理 |
| 8 | purrr | A light gray hexagonal logo for purrr, featuring a white cat icon. | 繰り返し処理 |

パイプ処理

パイプ処理ってなんですか？？



パイプ処理っていうのは、%>%を使って処理の順番どおりにコードを記述することだよ。

%>%っていう見慣れない文字のことをパイプって呼んでいるよ。

？？？



txhousing

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

①

city,year,mont
h,salesを選択

select

②

salesが100以
上の行を抽出

filter

| | | |
|--|--|--|
| | | |
| | | |
| | | |
| | | |
| | | |

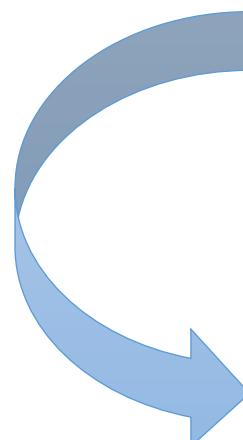
パイプなし

```
filter(  
  select(txhousing, city, year,  
  month, sales),  
  sales >= 100  
)
```

パイプあり

```
txhousing %>%  
  select(city, year, month,  
  sales) %>%  
  filter(sales >= 100)
```

$$f(\mathbf{x}, \mathbf{y}) \iff \mathbf{x} \%>\% f(\mathbf{y})$$



```
filter(  
  select(txhousing, city, year,  
month, sales),  
sales >= 100  
)
```

```
filter(  
  txhousing \%>%  
  select(city, year, month,  
sales),  
sales >= 100  
)
```

```
txhousing \%>%  
select(city, year, month,  
sales) \%>%  
filter(sales >= 100)
```

まとめ

- 1.パイプ処理ってなに？？
- 2.パイプなしとパイプありの比較
- 3.パイプ処理の仕組み

tibble

tibbleってなんですか？？



tibbleってのはね、普通のデータフレームの進化版だと思ってくれればOK。

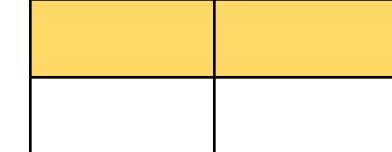
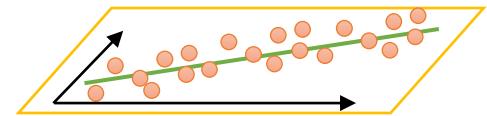
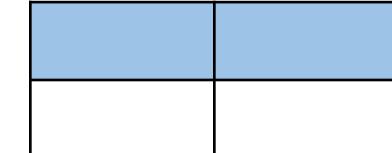
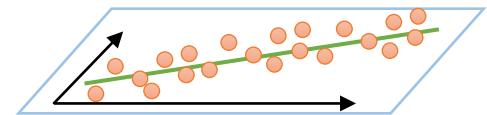
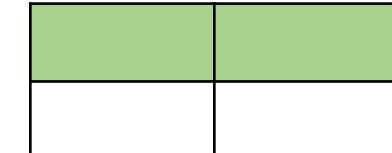
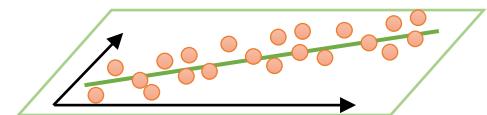
tidyverseは、データフレームではなくてtibbleを使用すること前提だから、基本はtibbleを使つていこう！

tibble

- ①データフレームの進化版
- ②パッケージ
- ③関数

tibbleの便利機能

どんなオブジェクトでもtibbleの中に格納できる！！

| int | dbl | chr | data | fig |
|-----|-------|-----|---|---|
| 5 | 3.124 | “a” |  |  |
| 10 | 8.425 | “c” |  |  |
| 76 | 5.32 | “E” |  |  |

まとめ

- 1.tibbleってなに？？
- 2.tibbleの作成
- 3.tibbleの便利機能

{dplyr}データフレーム処理

{dplyr}ってなんですか？？



{dplyr}は、データフレームの処理に特化したパッケージだよ。

{dplyr}には前処理に必要不可欠な便利関数がたくさんあるんだ！！



時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|---|-------------|
| 1 | tibble | A hexagonal logo for tibble, showing a dark blue background with a grid of colored squares and the word "tibble" at the top. | データフレームの進化版 |
| 2 | dplyr | A hexagonal logo for dplyr, featuring a colorful illustration of various data manipulation tools like a magnifying glass, a wrench, and a gear. | データフレーム処理 |
| 3 | tidyr | A hexagonal logo for tidyr, showing a dark blue background with a grid of colored lines and the word "tidyr" at the top. | tidyデータ処理 |
| 4 | ggplot2 | A hexagonal logo for ggplot2, featuring a line graph with three blue dots connected by a line, with the word "ggplot2" at the bottom. | 可視化 |
| 5 | stringr | A hexagonal logo for stringr, featuring a green background with a white icon of a violin and the word "stringr" at the bottom. | 文字列処理 |
| 6 | readr | A hexagonal logo for readr, featuring a blue background with a white icon of a document and the word "readr" at the bottom. | 入出力処理 |
| 7 | forcats | A hexagonal logo forforcats, featuring a brown background with a white icon of two cats and the word "forcats" at the bottom. | ファクター処理 |
| 8 | purrr | A hexagonal logo for purrr, featuring a white background with a black cat icon and the word "purrr" at the bottom. | 繰り返し処理 |

ここ！！



| No. | 関数 | 用途 |
|-----|----------|--------------------|
| 1 | filter | 行の抽出 |
| 2 | select | 列の抽出 |
| 3 | mutate | 列の追加 |
| 4 | join系 | キー結合 |
| 5 | bind系 | 縦横結合 |
| 6 | group_by | グルーピング |
| 7 | rowwise | 行処理 |
| 8 | その他 | ソート, 重複削除, 列処理などなど |

データフレーム

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

データフレーム %>%
filter(条件)

filter

条件を満たした行が
抽出される。

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

filterの内部処理

- ① 条件を判定し、行ごとに論理値ベクトル(TRUE, FALSE)を作成する。
- ② ①の論理値ベクトルをデータフレームに記録する。
- ③ ②でTRUEであった行のみを抽出する。

df

| No | gender | height |
|----|--------|--------|
| 1 | M | 165 |
| 2 | F | 150 |
| 3 | F | 170 |
| 4 | M | 175 |
| 5 | F | 165 |
| 6 | M | 195 |
| 7 | M | 180 |

```
df %>%  
filter(  
  gender == "F")
```

filter

①

| gender |
|--------|
| M |
| F |
| F |
| M |
| F |
| M |
| M |

| 条件 |
|-------|
| False |
| True |
| True |
| False |
| True |
| False |
| False |

| 判定結果 |
|-------|
| False |
| True |
| True |
| False |
| True |
| False |
| False |

②

| No | gender | height |
|----|--------|--------|
| 1 | M | 165 |
| 2 | F | 150 |
| 3 | F | 170 |
| 4 | M | 175 |
| 5 | F | 165 |
| 6 | M | 195 |
| 7 | M | 180 |

| 判定結果 |
|-------|
| False |
| True |
| True |
| False |
| True |
| False |
| False |

③

| No | gender | height |
|----|--------|--------|
| 2 | F | 150 |
| 3 | F | 170 |
| 5 | F | 165 |

| 判定結果 |
|------|
| True |
| True |
| True |

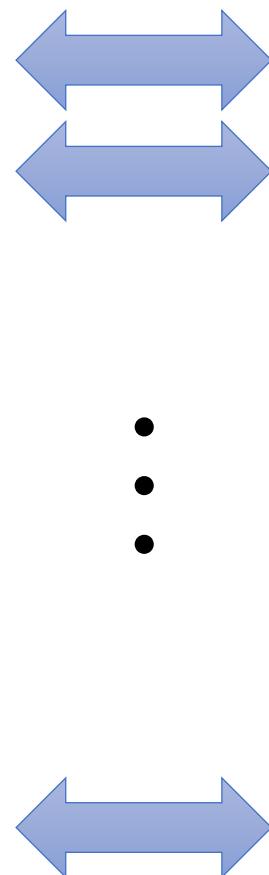
左側が抽出される

※ xとyは、ベクトルを表す。

| 比較演算子 | 構文 | 意味 |
|--------------------|------------------------|--------------|
| <code>==</code> | <code>x == y</code> | xとyは、等しいか？ |
| <code>!=</code> | <code>x != y</code> | xとyは、等しくないか？ |
| <code>>=</code> | <code>x >= y</code> | xはy以上か？ |
| <code>></code> | <code>x > y</code> | xはyより大きいか？ |
| <code><=</code> | <code>x <= y</code> | xはy以下か？ |
| <code><</code> | <code>x < y</code> | xはyより小さいか？ |
| <code>%in%</code> | <code>x %in% y</code> | xはyに含まれているか？ |

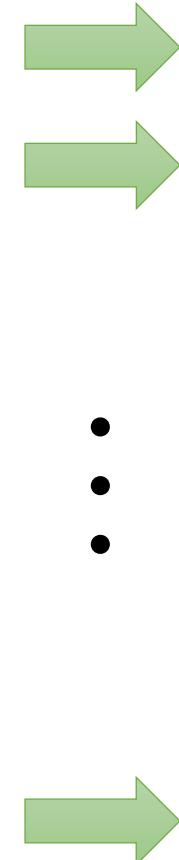
ベクトル1

| |
|---|
| I |
| J |
| J |
| I |
| H |
| H |
| J |
| J |



ベクトル2

| | |
|---|---|
| I | J |
| I | J |
| I | J |
| I | J |
| I | J |
| I | J |
| I | J |
| I | J |



論理値ベクトル

| |
|-------|
| TRUE |
| TRUE |
| TRUE |
| TRUE |
| FALSE |
| FALSE |
| TRUE |
| TRUE |

↓

ここが
作成

IかJはあるか？

ある⇒TRUE
ない⇒FALSE

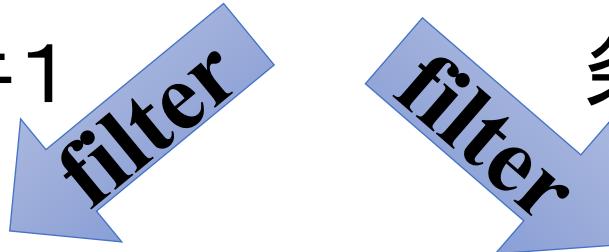
※ x と y は、論理値ベクトルを表す。

| 論理演算子 | 構文 | 意味 | 返り値 |
|-------|--------------------|---|---------|
| & | $x \& y$ | 両方TRUE \Rightarrow TRUE それ以外 \Rightarrow FALSE | 論理値ベクトル |
| | $x y$ | 1つがTRUE \Rightarrow TRUE 全部FALSE \Rightarrow FALSE | 論理値ベクトル |
| xor | $\text{xor}(x, y)$ | 1つがTRUE, 1つがFALSE \Rightarrow TRUE 両方ともTRUEもしくはFALSE \Rightarrow FALSE | 論理値ベクトル |
| ! | $! x$ | TRUE \Rightarrow FALSE FALSE \Rightarrow TRUE | 論理値ベクトル |
| any | $\text{any}(x)$ | 1つ以上TRUE \Rightarrow TRUE 全部FALSE \Rightarrow FALSE | 論理値スカラー |
| all | $\text{all}(x)$ | 全部TRUE \Rightarrow TRUE 1つ以上FALSE \Rightarrow FALSE | 論理値スカラー |

データフレーム

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

条件1 条件2



| 条件1論理値 | 条件2論理値 |
|--------|--------|
| TRUE | FALSE |
| FALSE | FALSE |
| TRUE | TRUE |

AND

| 複合論理値 |
|-------|
| FALSE |
| FALSE |
| TRUE |



ここが
作成

データフレーム

データフレーム %>% select(条件)

select

条件で指定した列が
抽出される。

selectの内部処理

- ① 条件を判定し、列ごとに論理値ベクトル(TRUE, FALSE)を作成する。
- ② ①の論理値ベクトルからTRUEの列の列番号を取得する。
- ③ ②で取得した列番号の列を抽出する。

```
df %>%  
  select(  
    Sepal.Length, Species  
)
```

| 1 | 2 | 3 | 4 | 5 |
|--------------|-------------|--------------|-------------|---------|
| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |

select

①

| 1 | 2 | 3 | 4 | 5 |
|--------------|-------------|--------------|-------------|---------|
| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
| TRUE | FALSE | FALSE | FALSE | TRUE |

判定結果

②

1, 5

③

| 1 | | 5 |
|--------------|--|---------|
| Sepal.Length | | Species |

| No. | 指定方法 | 関数 | 説明 |
|-----|--------|-------------|------------------------------------|
| 1 | 直接指定 | | 列名を直接指定する。 |
| 2 | 範囲指定 | : | 列名と列名の間に:を入れることで、それらの間の列名すべてを抽出する。 |
| 3 | 間接指定 | - | 列名の前に-をつけることで、その列名を抽出から除く。 |
| 4 | 前方一致 | starts_with | 列名の前方部分を指定して、一致する列を抽出する。 |
| 5 | 後方一致 | ends_with | 列名の後方部分を指定して、一致する列を抽出する。 |
| 6 | 部分一致 | contains | 列名に含まれる文字列を指定して、一致する列を抽出する。 |
| 7 | 正規表現一致 | matches | 正規表現で一致する列を抽出する。 |
| 8 | 文字列指定 | all_of | 列名を文字列で指定する。 |
| 9 | 全列抽出 | everything | 全列を抽出する。 |
| 10 | データ型指定 | where | 指定したデータ型を抽出する。 |

データフレーム

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

データフレーム %>%
mutate(条件)

mutate

条件で指定した列が
データフレームの右側
に挿入される。

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

データフレーム

```
データフレーム %>%  
mutate(  
  列名1 = ベクトル1,  
  列名2 = ベクトル2  
)
```

mutate

元のデータフレームの右側に
列名1とベクトル1,
列名2とベクトル2
が追加される.

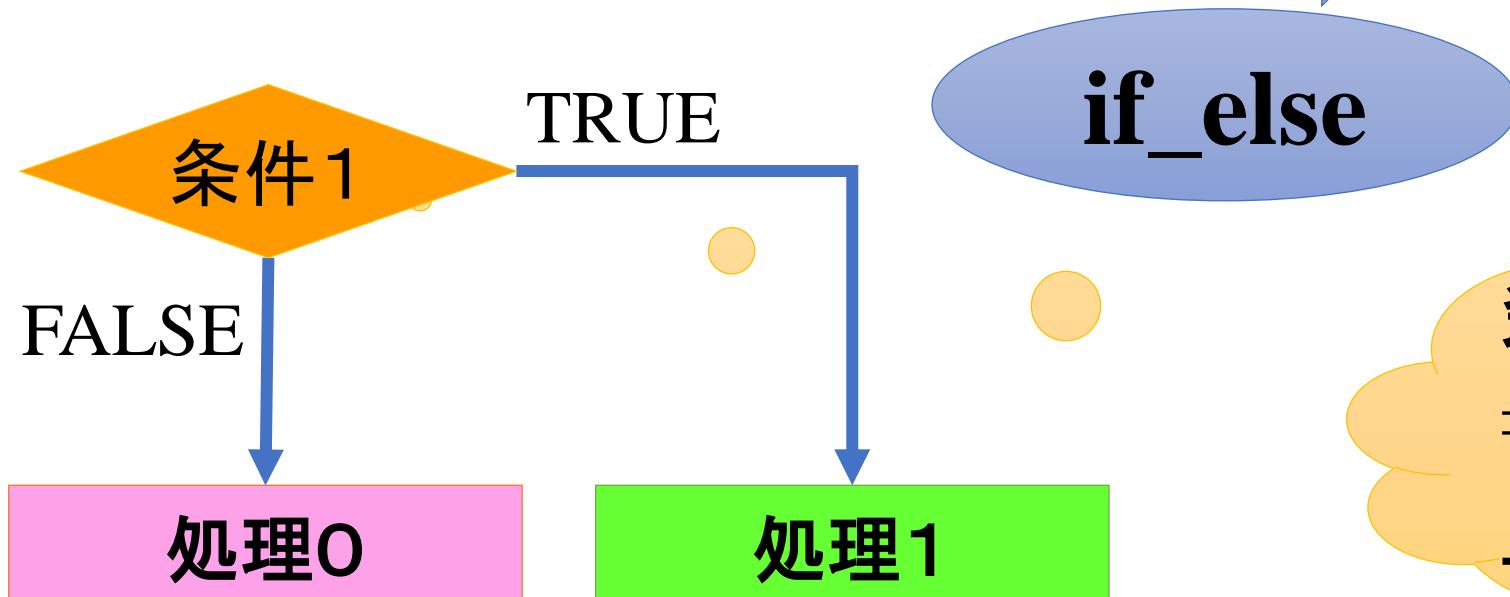
データフレーム

| | | | |
|--|--|--|--|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

データフレーム %>%

```
mutate(列名 =  
if_else(  
条件1,  
処理1, (条件1 = TRUE),  
処理0(条件1 = FALSE)  
)  
)
```

mutate



| | | | | 列名 |
|--|--|--|--|-----|
| | | | | 処理1 |
| | | | | 処理0 |
| | | | | 処理1 |
| | | | | 処理1 |
| | | | | 処理0 |

条件が1つ
⇒
単数条件

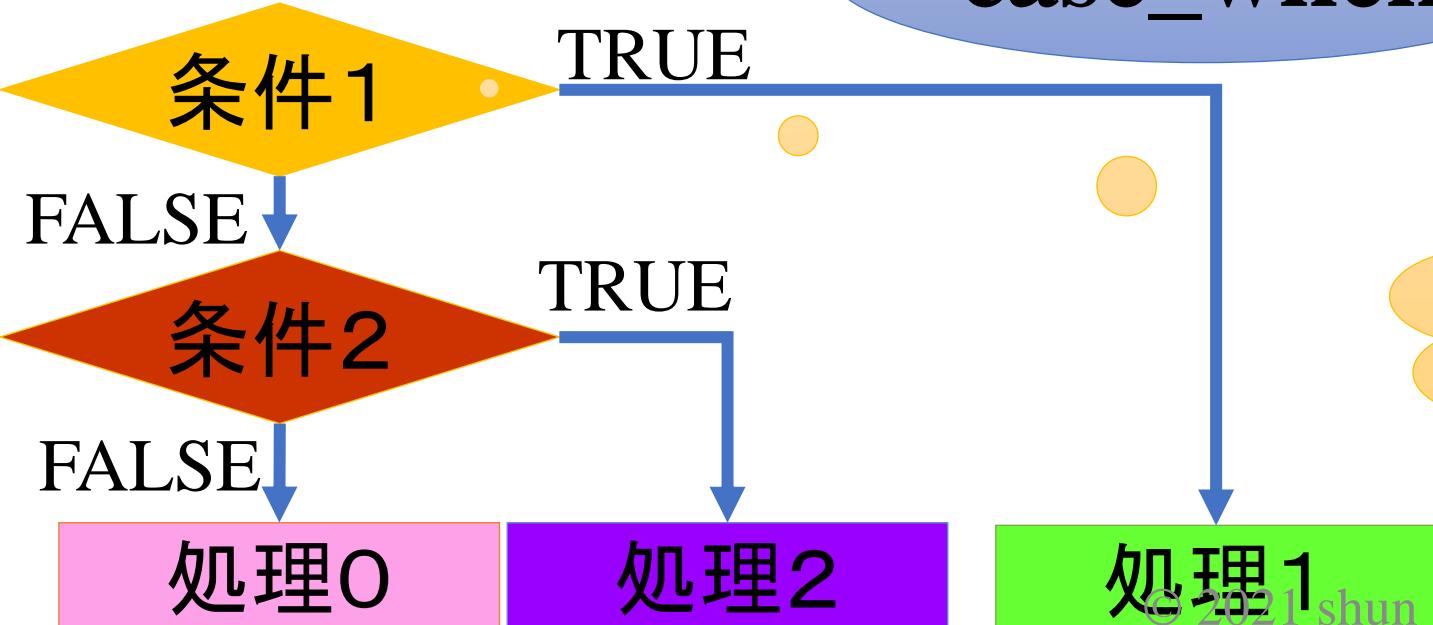
データフレーム

| | |
|--|--|
| | |
| | |
| | |
| | |
| | |

データフレーム %>%
mutate(列名 =
case_when(
条件1 ~ 处理1(条件1 = TRUE),
条件2 ~ 处理2(条件2 = TRUE),
TRUE ~ 处理0(条件1, 2 = FALSE)
)
)

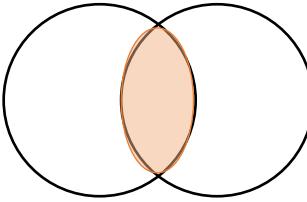
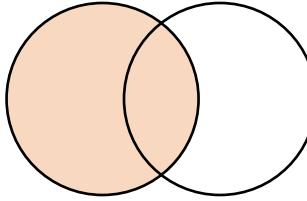
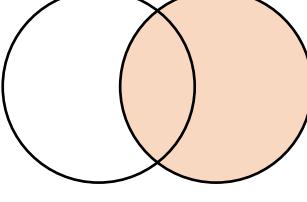
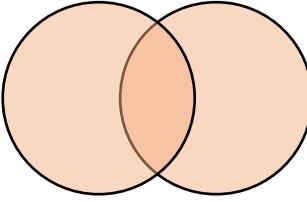
mutate

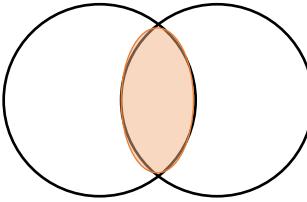
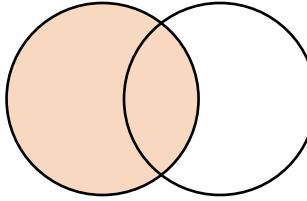
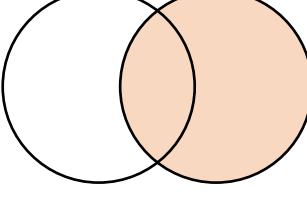
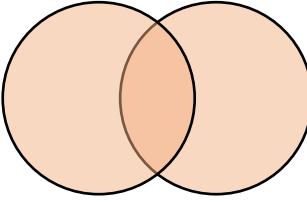
case_when



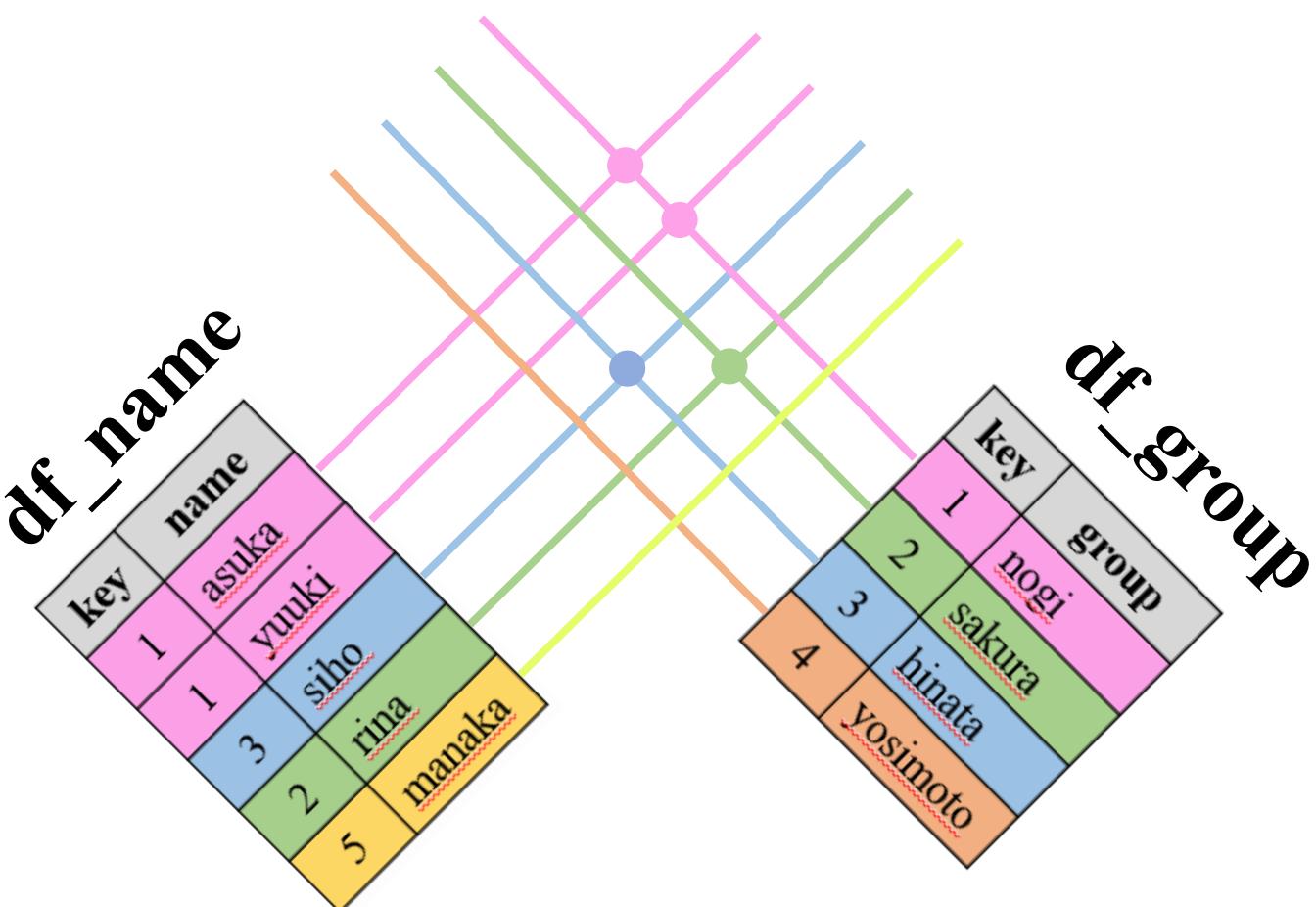
条件が2つ以上
⇒
複数条件

| 列名 |
|-----|
| 処理1 |
| 処理0 |
| 処理2 |
| 処理1 |
| 処理2 |

| No. | キー結合の種類 | 使い方 | イメージ |
|-----|--------------------|-------------------------------|---|
| 1 | インナージョイン (内部結合) | <code>inner_join(x, y)</code> |  |
| 2 | レフトジョイン (左結合) | <code>left_join(x, y)</code> |  |
| 3 | ライトジョイン (右結合) | <code>right_join(x, y)</code> |  |
| 4 | フルジョイン (外部結合) | <code>full_join(x, y)</code> |  |

| No. | キー結合の種類 | 使い方 | イメージ |
|-----|--------------------|-------------------------------|---|
| 1 | インナージョイン (内部結合) | <code>inner_join(x, y)</code> |  |
| 2 | レフトジョイン (左結合) | <code>left_join(x, y)</code> |  |
| 3 | ライトジョイン (右結合) | <code>right_join(x, y)</code> |  |
| 4 | フルジョイン (外部結合) | <code>full_join(x, y)</code> |  |

inner_join

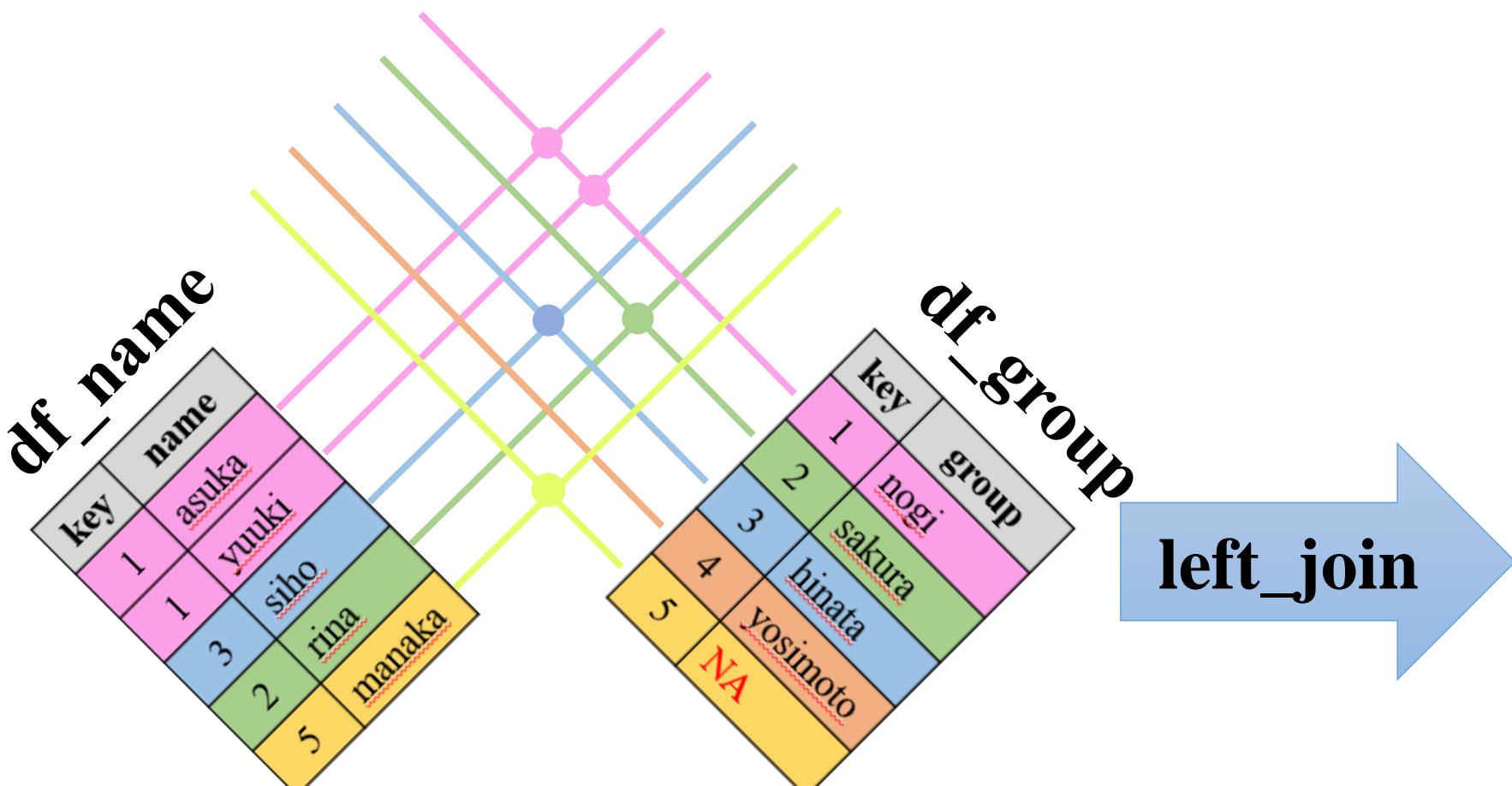


inner_join

`df_inner_join`

| key | name | group |
|-----|-------|--------|
| 1 | asuka | nogi |
| 1 | yuuki | nogi |
| 3 | siho | hinata |
| 2 | rina | sakura |

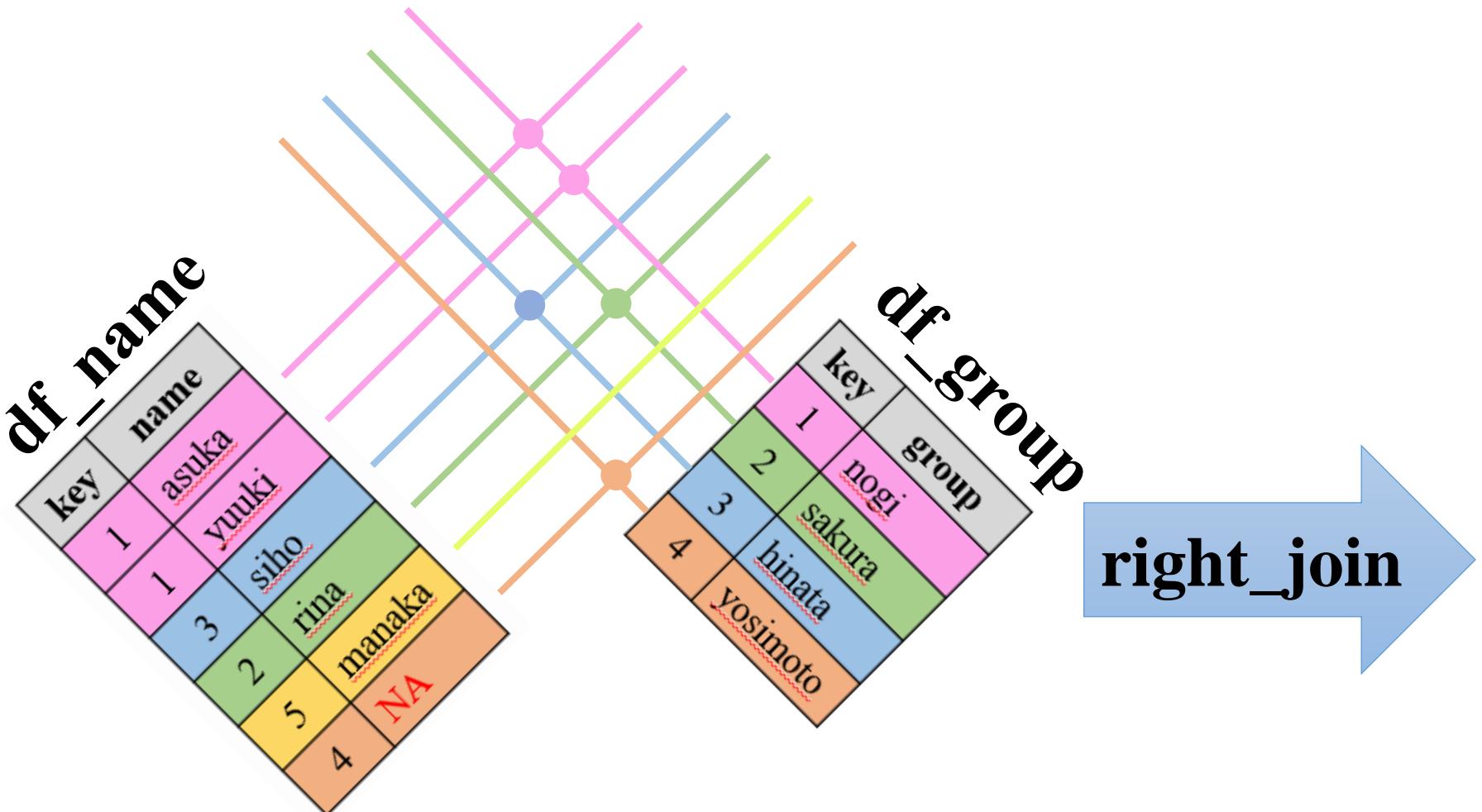
left_join



`df_left_join`

| key | name | group |
|-----|--------|--------|
| 1 | asuka | nogi |
| 1 | yuuki | nogi |
| 3 | siho | hinata |
| 2 | rina | sakura |
| 5 | manaka | NA |

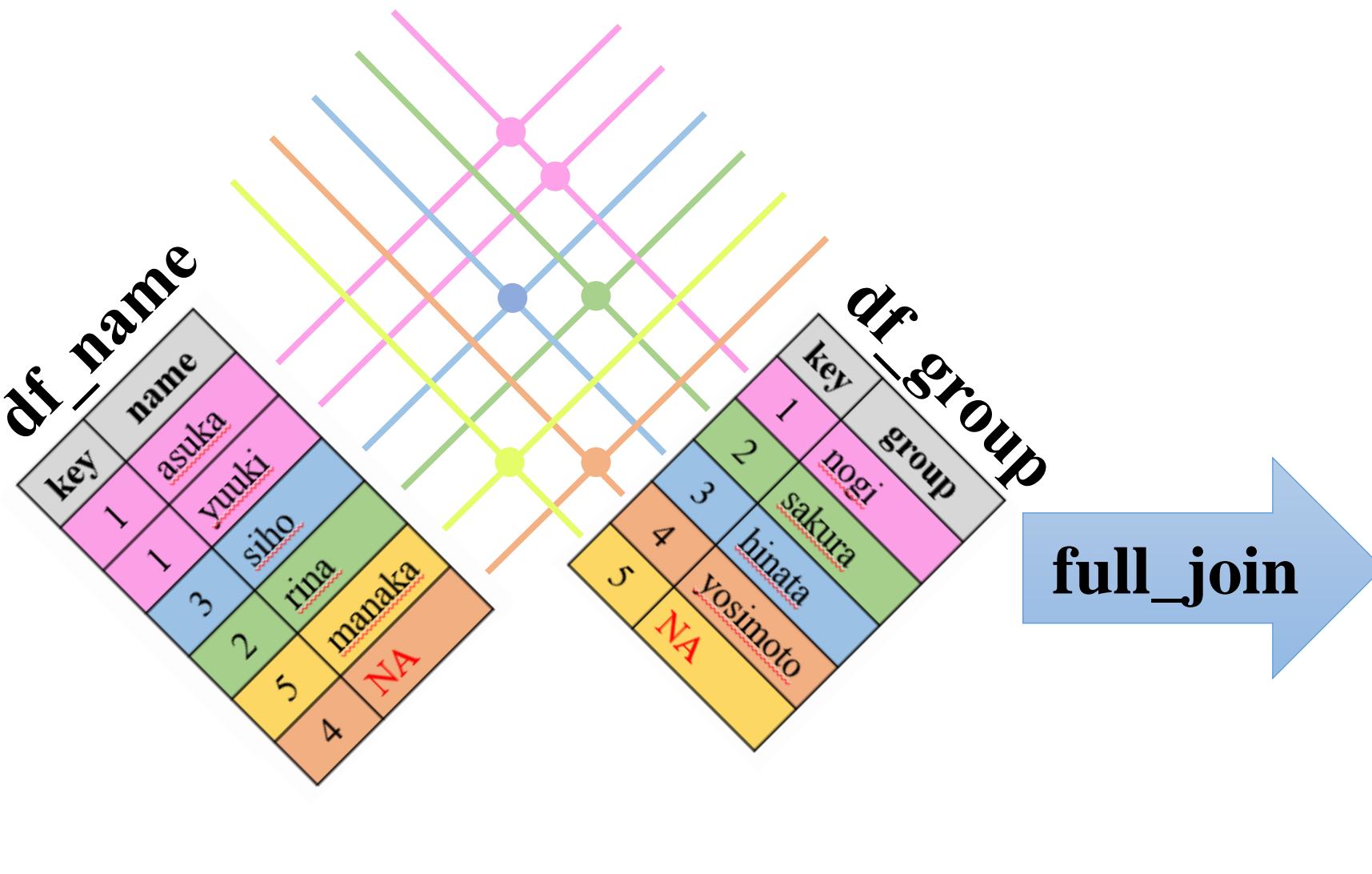
right_join



`df_right_join`

| key | name | group |
|-----|-------|----------|
| 1 | asuka | nogi |
| 1 | yuuki | nogi |
| 3 | siho | hinata |
| 2 | rina | sakura |
| 4 | NA | yosimoto |

full_join



`df_full_join`

| key | name | group |
|-----|--------|----------|
| 1 | asuka | nogi |
| 1 | yuuki | nogi |
| 3 | siho | hinata |
| 2 | rina | sakura |
| 5 | manaka | NA |
| 4 | NA | yosimoto |

df_name_1

| key | name |
|-----|-------|
| 1 | asuka |
| 2 | rina |



df_name_2

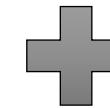
| key | name |
|-----|--------|
| 1 | hinako |
| 2 | yui |
| 3 | kyouko |

1.bind_rows



df_name

| key | name |
|-----|--------|
| 1 | asuka |
| 2 | rina |
| 1 | hinako |
| 2 | yui |
| 3 | kyouko |



df_group

| group |
|--------|
| nogi |
| sakura |
| nogi |
| sakura |
| hinata |

2.bind_cols

| key | name | group |
|-----|--------|--------|
| 1 | asuka | nogi |
| 2 | rina | sakura |
| 1 | hinako | nogi |
| 2 | yui | sakura |
| 3 | kyouko | hinata |

df_name_group

group_by: グルーピング

グルーピングって何ですか？



グルーピングっていうのは、データを特定の条件でまとめることだよ。



| class | gender | height |
|-------|--------|--------|
| a | M | 162 |
| b | F | 150 |
| c | F | 168 |
| c | M | 173 |
| a | F | 162 |
| c | M | 198 |
| b | M | 182 |
| a | F | 154 |
| c | M | 175 |
| b | M | 160 |
| a | F | 172 |

①
グルーピング

group_by
(class)

| class | gender | height |
|-------|--------|--------|
| a | M | 162 |
| a | F | 162 |
| a | F | 154 |
| a | F | 172 |
| b | F | 150 |
| b | M | 182 |
| b | M | 160 |
| c | F | 168 |
| c | M | 173 |
| c | M | 198 |
| c | M | 175 |

②
計算

summarise
(mean=
mean(height))

| class | mean |
|-------|-------|
| a | 162.5 |
| b | 164 |
| c | 178.5 |

| No. | 関数 | 使い方 | 例 | 意味 |
|-----|----------|-------------------|------------------------|------|
| 1 | max | max(列名) | max(height) | 最大値 |
| 2 | mean | mean(列名) | mean(height) | 平均 |
| 3 | median | median(列名) | median(height) | 中央値 |
| 4 | min | min(列名) | min(height) | 最小値 |
| 5 | quantile | quantile(列名, 分位点) | quantile(height, 0.25) | 分位点 |
| 6 | sd | sd(列名) | sd(height) | 標準偏差 |
| 7 | n | n() | n() | 個数 |

rowwise: 行処理

行処理って何ですか？？



行処理は、データフレームの行ごとに処理をしていくことだよ。

行処理は、データフレームにrowwiseを適用することでできるようになるよ。

| | x | y | z |
|---|----|----|----|
| ① | x1 | y1 | z1 |
| ② | x2 | y2 | z2 |
| ③ | x3 | y3 | z3 |
| ④ | x4 | y4 | z4 |

func(x1,y1,z1)

func(x2,y2,z2)

func(x3,y3,z3)

func(x4,y4,z4)

| out |
|------|
| out1 |

| |
|------|
| out2 |
|------|

| |
|------|
| out3 |
|------|

| |
|------|
| out4 |
|------|

diamonds

| | x | y | z |
|---|------|------|------|
| ① | 3.95 | 3.98 | 2.43 |
| ② | 3.89 | 3.84 | 2.31 |
| ③ | 4.05 | 4.07 | 2.31 |
| ④ | 4.2 | 4.23 | 2.63 |

```
diamonds %>%  
rowwise() %>%  
mutate( Mean = mean(c(x, y, z)) )
```

mean(c(3.95, 3.98, 2.43))

mean(c(3.89, 3.84, 2.31))

mean(c(4.05, 4.07, 2.31))

mean(c(4.2, 4.23, 2.63))

Mean

3.45

3.35

3.48

3.69

| class | gender | height |
|-------|--------|--------|
| a | M | 162 |
| b | F | 150 |
| c | F | 168 |
| c | M | 173 |
| a | F | 162 |
| c | M | 198 |
| b | M | 182 |
| a | F | 154 |
| c | M | 175 |
| b | M | 160 |
| a | F | 172 |

①
グルーピング

group_by
(class)

| class | gender | height |
|-------|--------|--------|
| a | M | 162 |
| a | F | 162 |
| a | F | 154 |
| a | F | 172 |
| b | F | 150 |
| b | M | 182 |
| b | M | 160 |
| c | F | 168 |
| c | M | 173 |
| c | M | 198 |
| c | M | 175 |

②
計算

summarise
(mean=
mean(height))

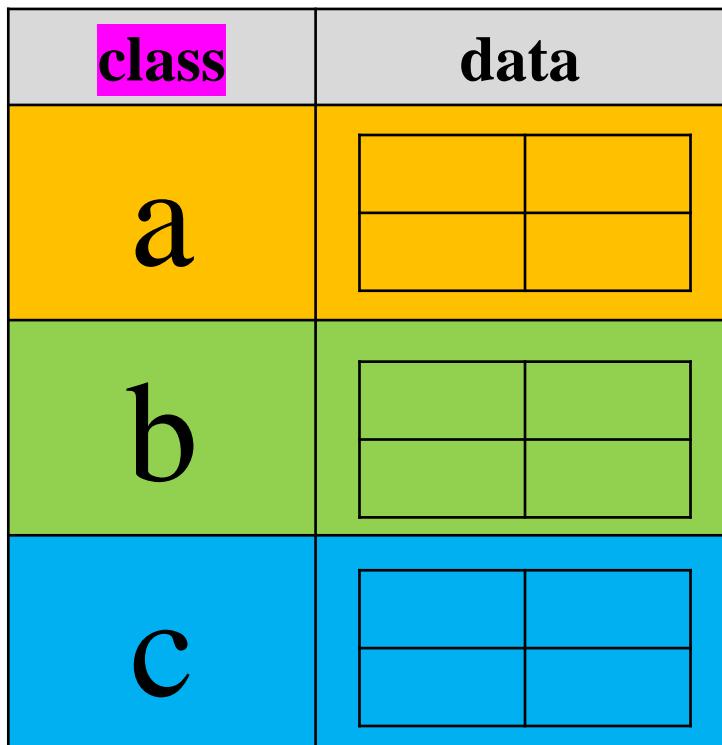
| class | mean |
|-------|-------|
| a | 162.5 |
| b | 164 |
| c | 178.5 |

| class | gender | height |
|-------|--------|--------|
| a | M | 162 |
| b | F | 150 |
| c | F | 168 |
| c | M | 173 |
| a | F | 162 |
| c | M | 198 |
| b | M | 182 |
| a | F | 154 |
| c | M | 175 |
| b | M | 160 |
| a | F | 172 |

ネスト

nest_by
(class)

group_nest
(class)



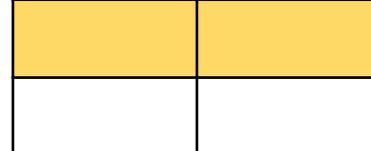
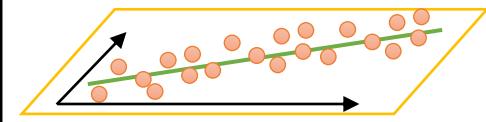
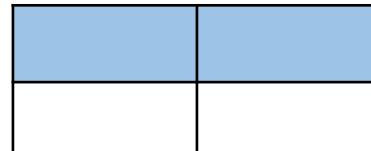
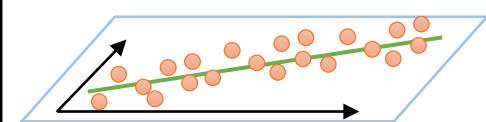
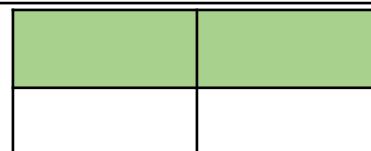
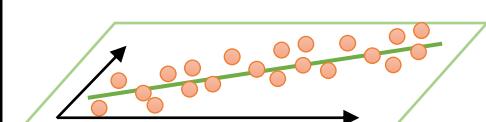
| gender | height |
|--------|--------|
| M | 162 |
| F | 162 |
| F | 154 |
| F | 172 |

| gender | height |
|--------|--------|
| F | 150 |
| M | 182 |
| M | 160 |

| gender | height |
|--------|--------|
| F | 168 |
| M | 173 |
| M | 198 |
| M | 175 |

iamondsをcutでネストしたデータに

- ① 散布図(x:carat y:price)を作成する関数を設定
- ② ①で作成した関数を使用し、グラフを作成
- ③ 単回帰モデル(x:carat y:price)を作成

| cut | data | func | fig | model |
|-----------|--|--------|---|---------|
| Fair |  | func_1 |  | model_1 |
| Good |  | func_2 |  | model_2 |
| Very Good |  | func_3 |  | model_3 |

その他

| No. | 関数 | 意味 |
|-----|----------|------|
| 1 | arrange | ソート |
| 2 | distinct | 重複削除 |
| 3 | across | 列処理 |

データフレーム

| col | | | | | |
|-----|--|--|--|--|--|
| 6 | | | | | |
| 4 | | | | | |
| 1 | | | | | |
| 3 | | | | | |
| 2 | | | | | |
| 5 | | | | | |
| 8 | | | | | |
| 7 | | | | | |

データフレーム %>%
arrange(col)



指定した列をソート
する。

| col | | | | |
|-----|--|--|--|--|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |

データフレーム

| col | | | | |
|-----|--|--|--|--|
| 1 | | | | |
| 2 | | | | |
| 1 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 2 | | | | |
| 1 | | | | |

データフレーム %>%
distinct(col,
.keep_all = TRUE)

指定した列の重複を
削除する。



distinct

| col | | | | |
|-----|--|--|--|--|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

データフレーム

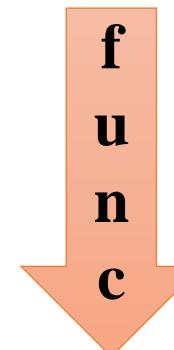
| x | | y |
|----|--|----|
| x1 | | y1 |
| x2 | | y2 |
| x3 | | y3 |
| x4 | | y4 |

データフレーム %>%
across(列, func)

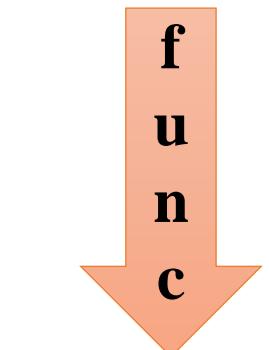


| x |
|----|
| x1 |
| x2 |
| x3 |
| x4 |

| y |
|----|
| y1 |
| y2 |
| y3 |
| y4 |



out x



out y

{dplyr}データフレーム処理

{dplyr}ってなんですか？？



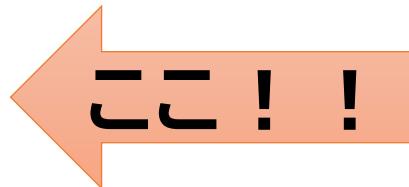
{dplyr}は、データフレームの処理に特化したパッケージだよ。

{dplyr}には前処理に必要不可欠な便利関数がたくさんあるんだ！！



時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|--|-------------|
| 1 | tibble | A dark blue hexagonal logo for tibble, featuring a grid of colored squares and the word "tibble" at the top. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagonal logo for dplyr, featuring a stylized orange and yellow arrow-like shape. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagonal logo for tidyr, featuring a colorful geometric pattern of lines and shapes. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagonal logo for ggplot2, featuring a line graph with three blue dots connected by a line. | 可視化 |
| 5 | stringr | A green hexagonal logo for stringr, featuring a white icon of a violin. | 文字列処理 |
| 6 | readr | A blue hexagonal logo for readr, featuring a white icon of a document and a small chart. | 入出力処理 |
| 7 | forcats | A brown hexagonal logo forforcats, featuring a cartoon illustration of two black cats in a box. | ファクター処理 |
| 8 | purrr | A white hexagonal logo for purrr, featuring a white icon of a cat's head. | 繰り返し処理 |





| No. | 関数 | 用途 |
|-----|----------|--------------------|
| 1 | filter | 行の抽出 |
| 2 | select | 列の抽出 |
| 3 | mutate | 列の追加 |
| 4 | join系 | キー結合 |
| 5 | bind系 | 縦横結合 |
| 6 | group_by | グルーピング |
| 7 | rowwise | 行処理 |
| 8 | その他 | ソート, 重複削除, 列処理などなど |

データフレーム

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

データフレーム %>%
filter(条件)



条件を満たした行が
抽出される。

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

データフレーム

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

データフレーム %>%
select(条件)

select

条件で指定した列が
抽出される。

| | | | | |
|--|---|---|---|---|
| | ■ | | ■ | |
| | | ■ | | ■ |
| | | | | |
| | | | ■ | |
| | | | | ■ |
| | | | | |
| | | | | ■ |
| | | | | |
| | | | | ■ |
| | | | | |
| | | | | ■ |
| | | | | |

データフレーム

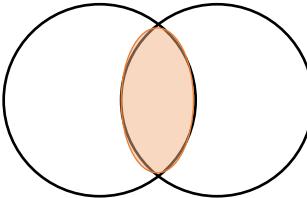
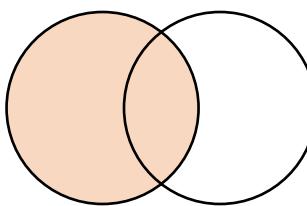
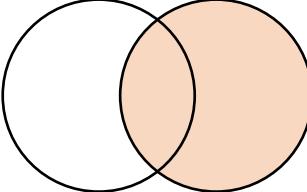
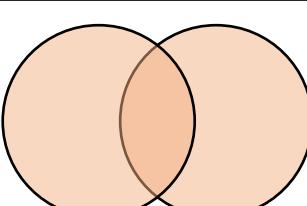
| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

データフレーム %>%
mutate(条件)

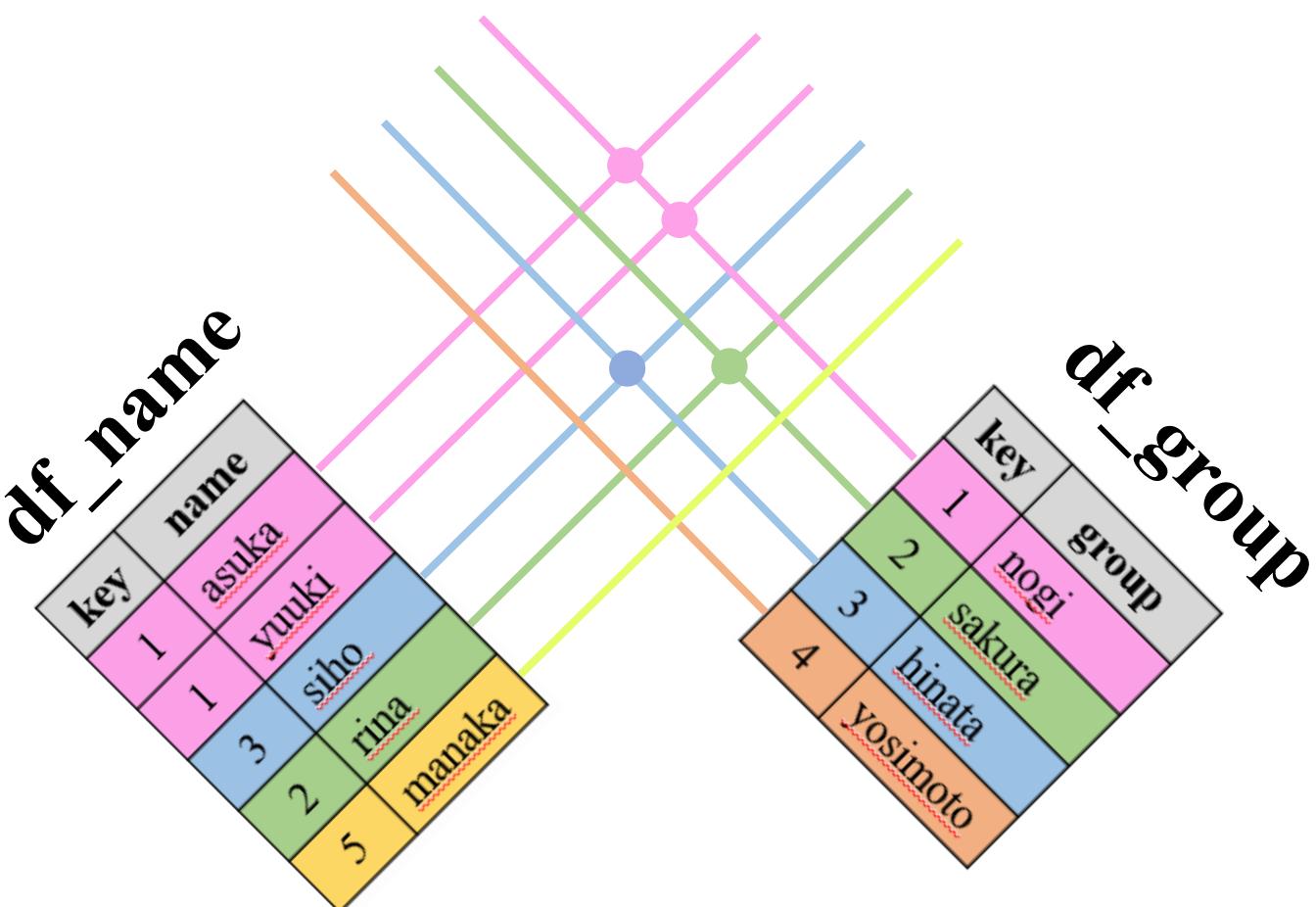
mutate

条件で指定した列が
データフレームの右側
に挿入される。

| | | | | | |
|--|--|--|--|--|--|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| No. | キー結合の種類 | 使い方 | イメージ |
|-----|--------------------|-------------------------------|---|
| 1 | インナージョイン (内部結合) | <code>inner_join(x, y)</code> |  |
| 2 | レフトジョイン (左結合) | <code>left_join(x, y)</code> |  |
| 3 | ライトジョイン (右結合) | <code>right_join(x, y)</code> |  |
| 4 | フルジョイン (外部結合) | <code>full_join(x, y)</code> |  |

inner_join

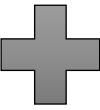


`df_inner_join`

| key | name | group |
|-----|-------|--------|
| 1 | asuka | nogi |
| 1 | yuuki | nogi |
| 3 | siho | hinata |
| 2 | rina | sakura |

df_name_1

| key | name |
|-----|-------|
| 1 | asuka |
| 2 | rina |



df_name_2

| key | name |
|-----|--------|
| 1 | hinako |
| 2 | yui |
| 3 | kyouko |

1.bind_rows



df_name

| key | name |
|-----|--------|
| 1 | asuka |
| 2 | rina |
| 1 | hinako |
| 2 | yui |
| 3 | kyouko |



df_group

| group |
|--------|
| nogi |
| sakura |
| nogi |
| sakura |
| hinata |

2.bind_cols

| key | name | group |
|-----|--------|--------|
| 1 | asuka | nogi |
| 2 | rina | sakura |
| 1 | hinako | nogi |
| 2 | yui | sakura |
| 3 | kyouko | hinata |

df_name_group

group_by: グルーピング

グルーピングって何ですか？



グルーピングっていうのは、データを特定の条件でまとめることだよ。



| class | gender | height |
|-------|--------|--------|
| a | M | 162 |
| b | F | 150 |
| c | F | 168 |
| c | M | 173 |
| a | F | 162 |
| c | M | 198 |
| b | M | 182 |
| a | F | 154 |
| c | M | 175 |
| b | M | 160 |
| a | F | 172 |

①
グルーピング

group_by
(class)

| class | gender | height |
|-------|--------|--------|
| a | M | 162 |
| a | F | 162 |
| a | F | 154 |
| a | F | 172 |
| b | F | 150 |
| b | M | 182 |
| b | M | 160 |
| c | F | 168 |
| c | M | 173 |
| c | M | 198 |
| c | M | 175 |

②
計算

summarise
(mean=
mean(height))

| class | mean |
|-------|-------|
| a | 162.5 |
| b | 164 |
| c | 178.5 |

rowwise: 行処理

行処理って何ですか？？



行処理は、データフレームの行ごとに処理をしていくことだよ。

行処理は、データフレームにrowwiseを適用することでできるようになるよ。

diamonds

| | x | y | z |
|---|------|------|------|
| ① | 3.95 | 3.98 | 2.43 |
| ② | 3.89 | 3.84 | 2.31 |
| ③ | 4.05 | 4.07 | 2.31 |
| ④ | 4.2 | 4.23 | 2.63 |

```
diamonds %>%
rowwise() %>%
mutate( Mean = mean(c(x, y, z)) )
```

mean(c(3.95, 3.98, 2.43))

mean(c(3.89, 3.84, 2.31))

mean(c(4.05, 4.07, 2.31))

mean(c(4.2, 4.23, 2.63))

Mean

3.45

3.35

3.48

3.69

その他

| No. | 関数 | 意味 |
|-----|----------|------|
| 1 | arrange | ソート |
| 2 | distinct | 重複削除 |
| 3 | across | 列処理 |

データフレーム

| col | | | | | |
|-----|--|--|--|--|--|
| 6 | | | | | |
| 4 | | | | | |
| 1 | | | | | |
| 3 | | | | | |
| 2 | | | | | |
| 5 | | | | | |
| 8 | | | | | |
| 7 | | | | | |

データフレーム %>%
arrange(col)



指定した列をソート
する。

| col | | | | |
|-----|--|--|--|--|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |

データフレーム

| col | | | | |
|-----|--|--|--|--|
| 1 | | | | |
| 2 | | | | |
| 1 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 2 | | | | |
| 1 | | | | |

データフレーム %>%
distinct(col,
.keep_all = TRUE)

指定した列の重複を
削除する。



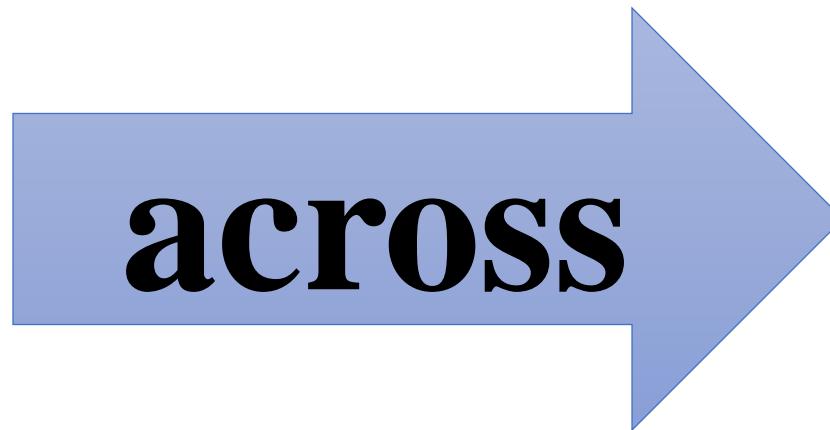
distinct

| col | | | | |
|-----|--|--|--|--|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

データフレーム

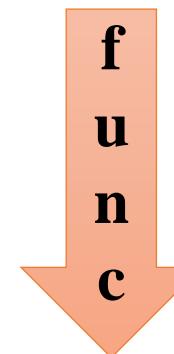
| x | | y |
|----|--|----|
| x1 | | y1 |
| x2 | | y2 |
| x3 | | y3 |
| x4 | | y4 |

データフレーム %>%
across(列, func)

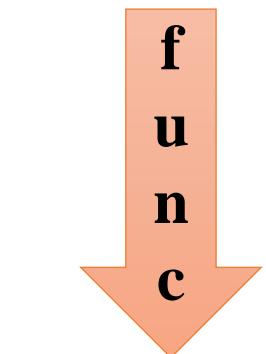


| x |
|----|
| x1 |
| x2 |
| x3 |
| x4 |

| y |
|----|
| y1 |
| y2 |
| y3 |
| y4 |



out x



out y

{tidyverse} tidyデータ処理



{tidyverse}は、 tidyデータの処理に特化したパッケージだよ。

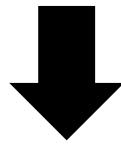
tidyデータっていうのは、コンピュータにわかりやすいようにきれいに整理されたデータのことだよ。

日本語だと整然データといったりもするね。

{tidyverse}ってなんですか？？



果物



| お店 | みかん | りんご |
|----|-----|-----|
| A | 100 | 150 |
| B | 70 | 90 |
| C | 120 | 80 |



お店

値段



tidyデータ！！

| お店 | 果物 | 値段 |
|----|-----|-----|
| A | みかん | 100 |
| A | りんご | 150 |
| B | みかん | 70 |
| B | りんご | 90 |
| C | みかん | 120 |
| C | りんご | 80 |



お店



果物



値段



時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|--|-------------|
| 1 | tibble | A dark blue hexagon with a grid of colored squares at the bottom, with the word "tibble" written vertically above it. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagon with a stylized orange and yellow gear-like shape, with the word "dplyr" written vertically above it. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagon with a colorful geometric pattern of lines and shapes, with the word "tidyr" written vertically above it. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagon with a black outline, containing a small blue line plot with three data points, with the word "ggplot2" written vertically above it. | 可視化 |
| 5 | stringr | A green hexagon with a white icon of a violin, with the word "stringr" written vertically below it. | 文字列処理 |
| 6 | readr | A blue hexagon with a white icon of a document and a bar chart, with the word "readr" written vertically below it. | 入出力処理 |
| 7 | forcats | A brown hexagon with a white icon of two cats, with the word "forcats" written vertically below it. | ファクター処理 |
| 8 | purrr | A white hexagon with a black outline, containing a white icon of a cat's head, with the word "purrr" written vertically below it. | 繰り返し処理 |

ここ！！



| No. | 関数 | 用途 |
|-----|--------------|------|
| 1 | pivot_longer | 縦変換 |
| 2 | pivot_wider | 横変換 |
| 3 | separate | 列の分割 |
| 4 | unite | 列の結合 |

pivot系：縦横変換

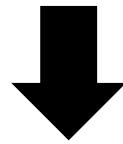
縦横変換って何ですか？



データフレームの内容を変えないで、形を変えることだよ。

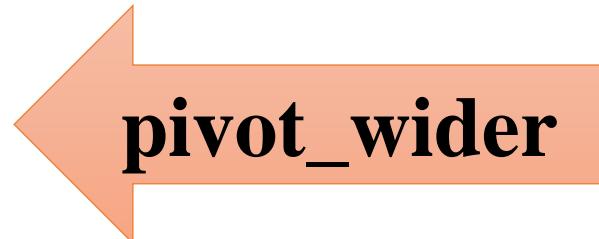
縦型 (long型) にする場合と横型 (wide型) にする場合があるから、合わせて縦横変換って呼んでるよ。

果物



| お店 | みかん | りんご |
|----|-----|-----|
| A | 100 | 150 |
| B | 70 | 90 |
| C | 120 | 80 |

pivot_longer



お店

値段

| お店 | 果物 | 値段 |
|----|-----|-----|
| A | みかん | 100 |
| A | りんご | 150 |
| B | みかん | 70 |
| B | りんご | 90 |
| C | みかん | 120 |
| C | りんご | 80 |

お店

果物

値段

df_wide

| store | orange | apple |
|-------|--------|-------|
| A | 100 | 150 |
| B | 70 | 90 |
| C | 120 | 80 |

```
df_wide %>%  
pivot_longer(  
  cols = c(orange, apple),  
  names_to = "fruit",  
  values_to = "price")  
)
```

pivot_longer

pivot_wider

```
df_long %>%  
pivot_wider(  
  names_from = fruit,  
  values_from = price)  
)
```

df_long

| store | fruit | price |
|-------|--------|-------|
| A | orange | 100 |
| A | apple | 150 |
| B | orange | 70 |
| B | apple | 90 |
| C | orange | 120 |
| C | apple | 80 |

separate unite: 列の分割と結合

列の分割と結合ってなんですか？



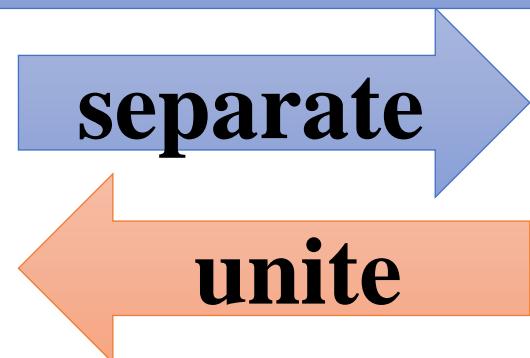
列の分割と結合ってのは、既存の列を分割したり、複数の列を結合することだよ。



df_unite

| No. | date |
|-----|---------|
| 1 | 2019-1 |
| 2 | 2020-3 |
| 3 | 2017-12 |
| 4 | 2018-9 |
| 5 | 2015-4 |

```
df_unite %>%  
separate(  
  col = date,  
  into = c("year", "month"),  
  sep = "-"  
)
```



```
df_separate %>%  
  unite(  
    col = date, year, month ,  
    sep = "-"  
)
```

df_separate

| No. | year | month |
|-----|------|-------|
| 1 | 2019 | 1 |
| 2 | 2020 | 3 |
| 3 | 2017 | 12 |
| 4 | 2018 | 9 |
| 5 | 2015 | 4 |

{ggplot2}可視化

{ggplot2}ってなんですか？？



{ggplot2}は可視化に特化したパッケージのことだよ！

可視化はデータをグラフにすることだよ。

データをグラフにすることで、そのデータがどんな傾向があるのか把握しやすくなるんだ！

データフレーム

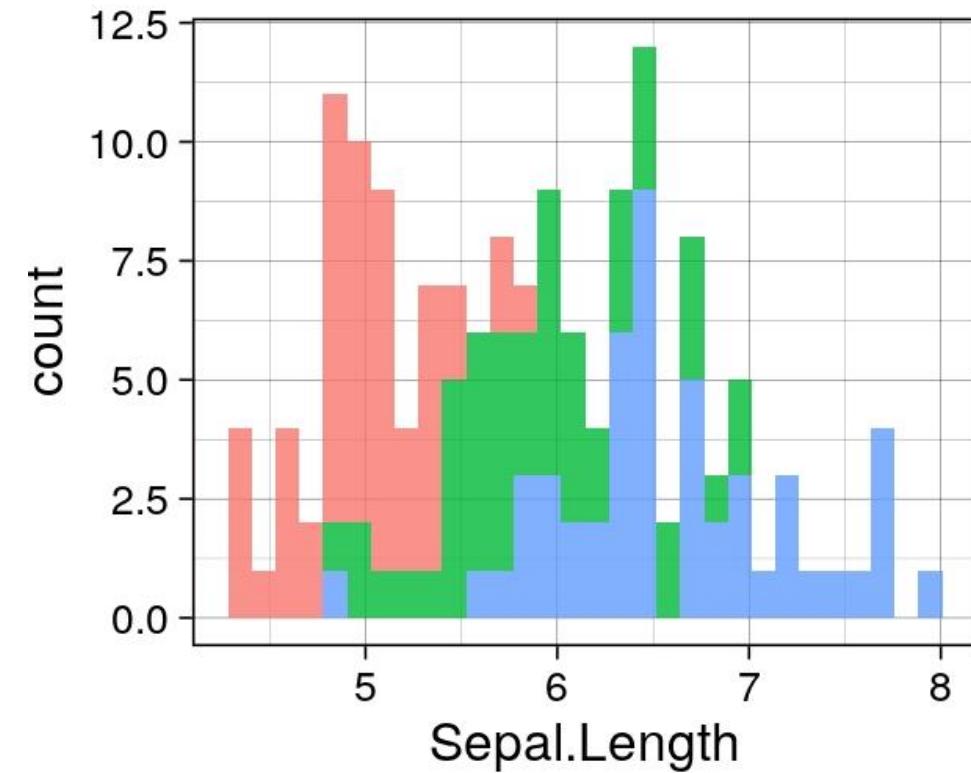
| Sepal.Length | Species |
|--------------|------------|
| 5.2 | setosa |
| 5.6 | versicolor |
| 6.7 | virginica |
| 4.9 | setosa |
| 5.5 | setosa |
| 7.2 | virginica |
| 5.7 | versicolor |

Species

setosa
versicolor
virginica

グラフ

可視化



傾向がわかりづらい...

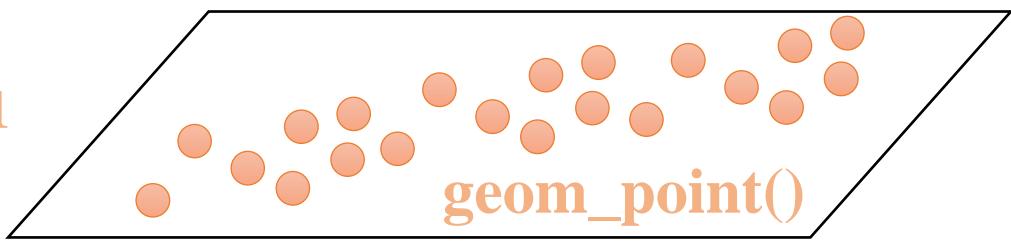
傾向がわかりやすい！！

1.キャンバス



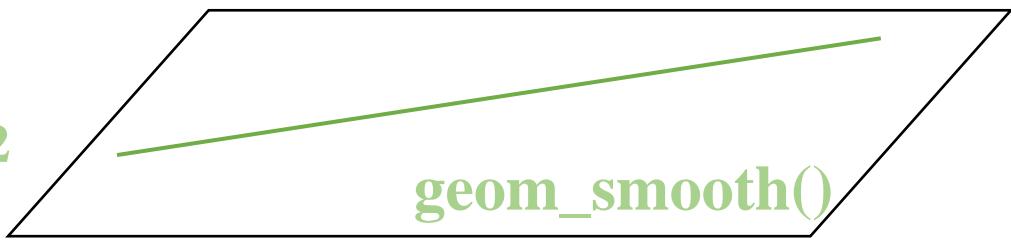
+

2.グラフ_1



+

2.グラフ_2

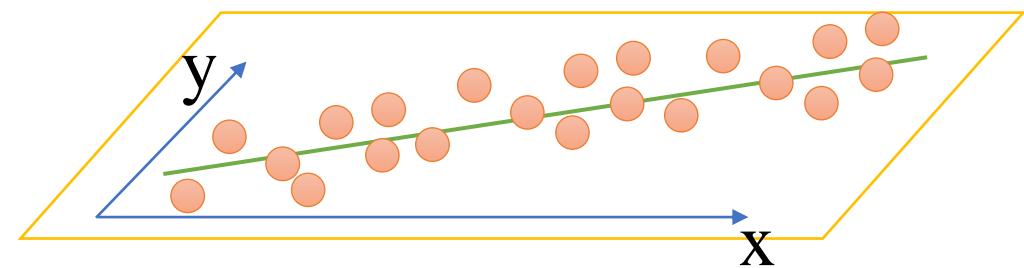


+

3.体裁



全体





時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|---|-------------|
| 1 | tibble | A dark blue hexagon with a grid of colored squares at the bottom, with the word "tibble" in white at the top. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagon with a stylized orange and yellow gear-like shape, with the word "dplyr" in white at the top. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagon with a colorful geometric pattern of lines and shapes, with the word "tidyr" in white at the top. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagon with a black outline, containing a small blue line plot with three data points, with the word "ggplot2" in white at the bottom. | 可視化 |
| 5 | stringr | A green hexagon with a white icon of a violin, with the word "stringr" in white at the bottom. | 文字列処理 |
| 6 | readr | A blue hexagon with a white icon of a document and a bar chart, with the word "readr" in white at the bottom. | 入出力処理 |
| 7 | forcats | A brown hexagon with a white icon of two cats, with the word "forcats" in white at the bottom. | ファクター処理 |
| 8 | purrr | A white hexagon with a black outline, containing a white icon of a cat's head, with the word "purrr" in white at the bottom. | 繰り返し処理 |

ここ！！

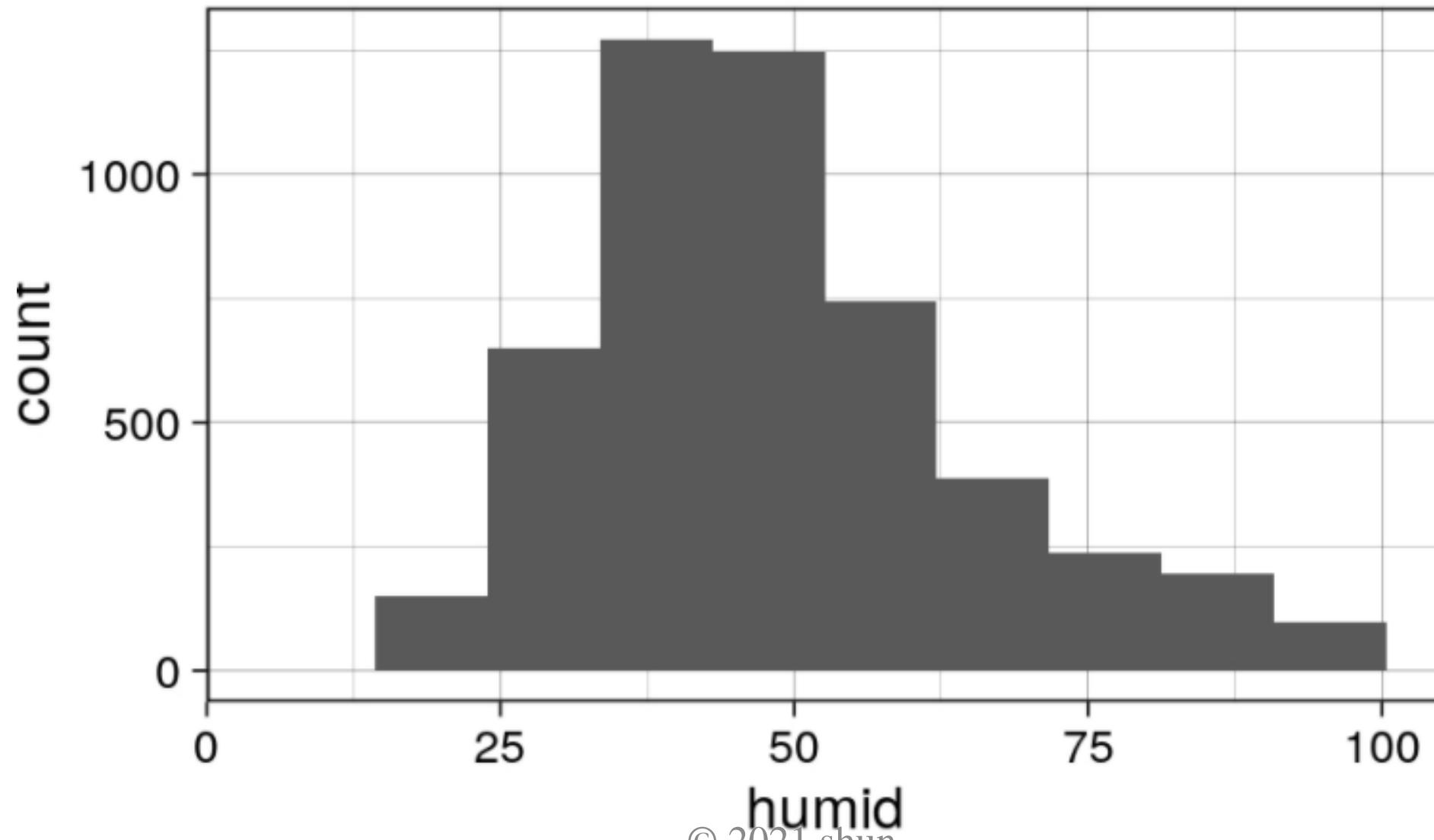


| No. | 関数 | 用途 |
|-----|----------------|--------|
| 1 | ggplot | キャンパス |
| 2 | geom_point | 散布図 |
| 3 | geom_smooth | 回帰曲線 |
| 4 | geom_histogram | ヒストグラム |
| 5 | geom_line | 折れ線グラフ |
| 6 | theme | 体裁 |

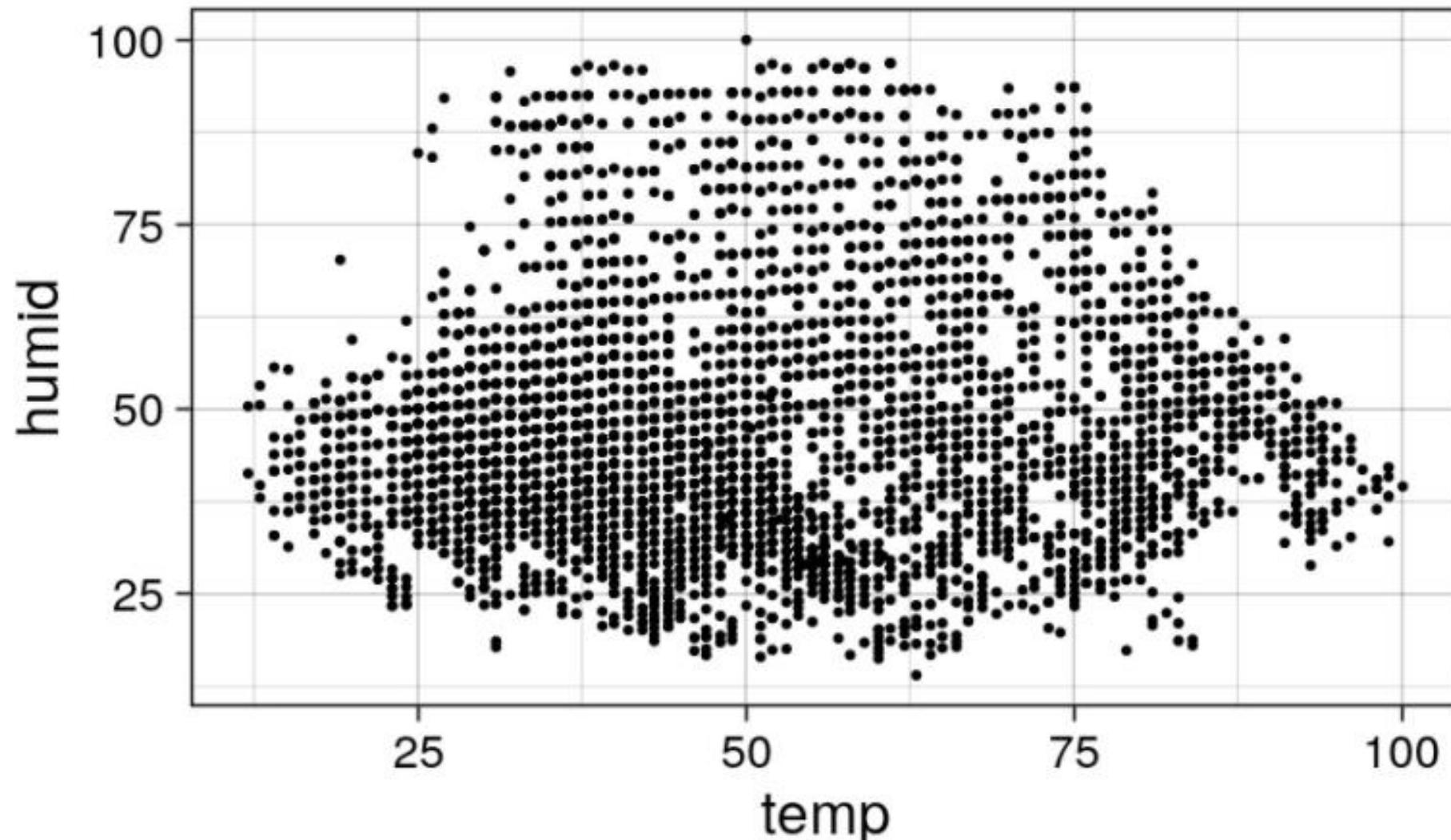
{ggplot2}可視化(具体例)

1. ヒストグラム
2. 散布図
3. トレンドグラフ

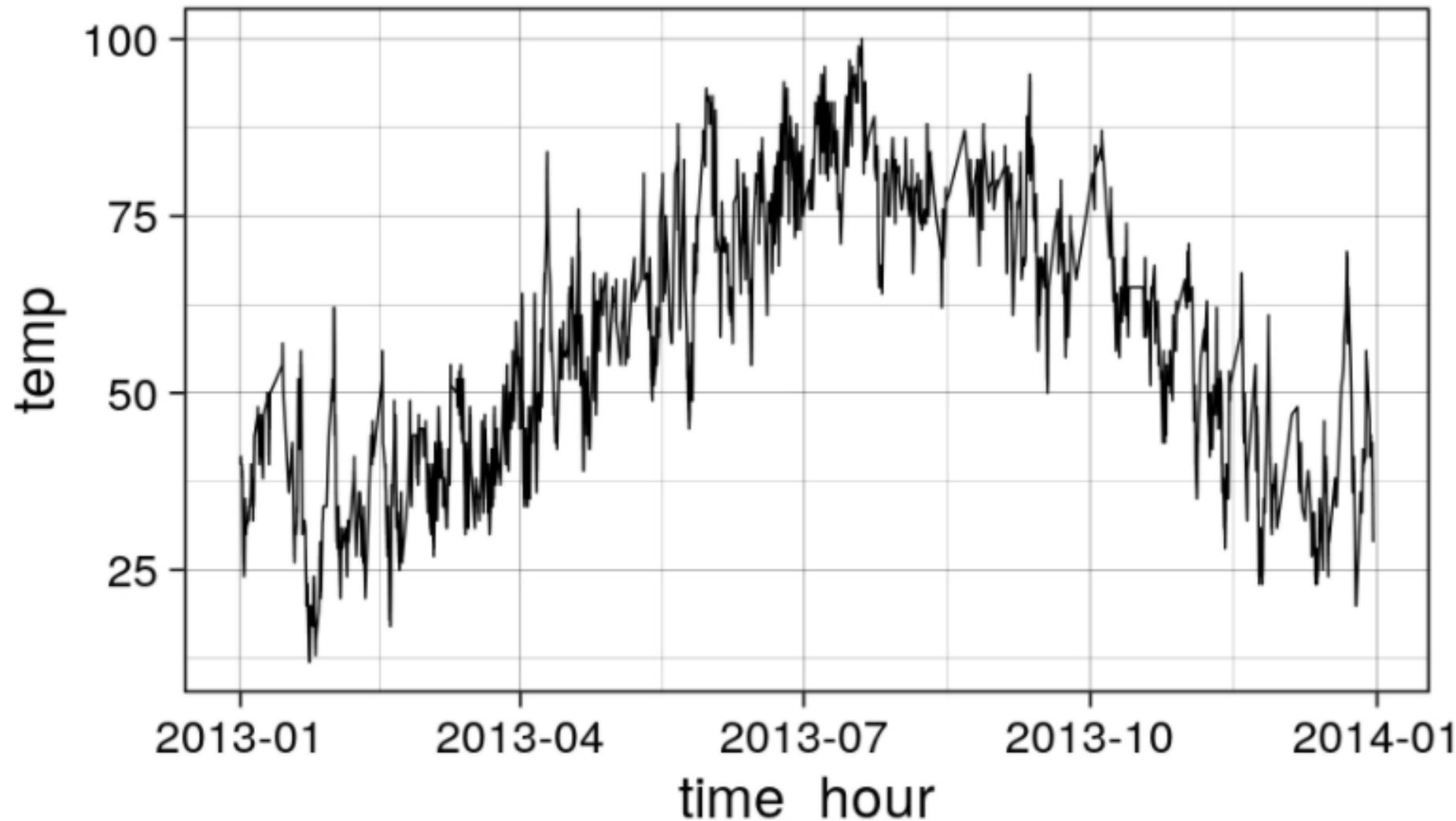
ヒストグラム



散布図



トレンドグラフ



{stringr}文字列処理

{stringr}って何ですか？？



{stringr}は、文字列処理に特化したパッケージのことだよ。

文字列には色々な処理が考えられるんだ。

例えば、文字列を連結してファイル名にしたり、文字列にマッチするところを探したり、などなど。.

この文字列のいろんな処理の総称を文字列処理って呼んでるよ。



時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|---|-------------|
| 1 | tibble | A dark blue hexagon with the word "tibble" at the top and a grid of colored squares below it. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagon with the word "dplyr" at the top and a colorful illustration of various data manipulation tools like arrows and a magnifying glass. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagon with the word "tidyr" at the top and a colorful illustration of data structures and flow. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagon with the word "ggplot2" at the bottom and a small line plot with three data points. | 可視化 |
| 5 | stringr | A green hexagon with the word "stringr" at the bottom and an icon of a violin. | 文字列処理 |
| 6 | readr | A blue hexagon with the word "readr" at the bottom and an icon of a document and a grid. | 入出力処理 |
| 7 | forcats | A brown hexagon with the word "forcats" at the bottom and an illustration of two cats. | ファクター処理 |
| 8 | purrr | A white hexagon with the word "purrr" at the bottom and an illustration of a cat's head. | 繰り返し処理 |

ここ！！



| No. | 関数 | 用途 |
|-----|-----------------|-----------|
| 1 | str_length | 長さ取得 |
| 2 | str_c | 連結 |
| 3 | str_sub | 抽出 |
| 4 | str_detect | マッチの真偽 |
| 5 | str_subset | マッチ文字列の抽出 |
| 6 | str_extract | マッチの先頭抽出 |
| 7 | str_extract_all | マッチの全抽出 |
| 8 | str_replace | マッチの先頭置換 |
| 9 | str_replace_all | マッチの全置換 |
| 10 | str_split | マッチの分割 |

`pattern = "a"`
でマッチング

nogi

sakura

hinata

`str_subset`



sakura

hinata

`str_extract`

NA

la

a

`str_extract_all`

character(0)

la a

a a

{readr}入出力処理



{readr}はデータの入出力処理に特化したパッケージのことだよ。

データを読み込んだり、加工したデータを出力したりすることができるよ。

{readr}ってなんですか？？





時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|--|-------------|
| 1 | tibble | A dark blue hexagonal logo for tibble, featuring a grid of colored squares and the word "tibble" at the top. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagonal logo for dplyr, featuring a stylized orange and yellow arrow icon. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagonal logo for tidyr, featuring a colorful geometric pattern of lines and shapes. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagonal logo for ggplot2, featuring a line graph with three blue dots connected by lines. | 可視化 |
| 5 | stringr | A green hexagonal logo for stringr, featuring a white icon of a violin. | 文字列処理 |
| 6 | readr | A blue hexagonal logo for readr, featuring a white icon of a document with a bar chart. | 入出力処理 |
| 7 | forcats | A brown hexagonal logo forforcats, featuring a black cat icon. | ファクター処理 |
| 8 | purrr | A white hexagonal logo for purrr, featuring a white icon of a cat's head. | 繰り返し処理 |





| No. | 関数 | 用途 |
|-----|-----------|-------|
| 1 | write_csv | CSV出力 |
| 2 | read_csv | CSV入力 |
| 3 | write_rds | RDS出力 |
| 4 | read_rds | RDS入力 |

{forcats}ファクター処理

{forcats}ってなんですか？？



{forcats}はファクター処理に特化したパッケージのことだよ。

ファクターでは、普通に文字列にはない水準という概念があることは覚えているかな？

その水準があるために、色々な処理をする必要があるんだ。



時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|--|-------------|
| 1 | tibble | A dark blue hexagonal logo for tibble, featuring a grid of colored squares and the word "tibble" at the top. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagonal logo for dplyr, featuring a stylized orange and yellow arrow icon. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagonal logo for tidyr, featuring a colorful geometric pattern of lines and shapes. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagonal logo for ggplot2, featuring a line graph with three blue dots connected by lines. | 可視化 |
| 5 | stringr | A green hexagonal logo for stringr, featuring a white icon of a violin. | 文字列処理 |
| 6 | readr | A blue hexagonal logo for readr, featuring a white icon of a document with a barcode. | 入出力処理 |
| 7 | forcats | A brown hexagonal logo forforcats, featuring a black cat icon. | ファクター処理 |
| 8 | purrr | A white hexagonal logo for purrr, featuring a white icon of a cat's head. | 繰り返し処理 |

← ここ！！



| No. | 関数 | 用途 |
|-----|--------------|----------|
| 1 | fct_inorder | 水準の順番の変更 |
| 2 | fct_infreq | |
| 3 | fct_rev | |
| 4 | fct_shift | |
| 5 | fct_relevel | |
| 6 | fct_reorder | |
| 7 | fct_recode | 水準の値の変更 |
| 8 | fct_collapse | |
| 9 | fct_lump | |

{purrr}繰り返し処理

{purrr}ってなんですか？？



{purrr}は繰り返し処理に特化したパッケージのことだよ。

繰り返し処理は、行ごとに同じ処理を繰り返す処理のことだよ！

| | x | y | z |
|---|----|----|----|
| ① | x1 | y1 | z1 |
| ② | x2 | y2 | z2 |
| ③ | x3 | y3 | z3 |
| ④ | x4 | y4 | z4 |

func(x1,y1,z1)

func(x2,y2,z2)

func(x3,y3,z3)

func(x4,y4,z4)

| out |
|------|
| out1 |

| |
|------|
| out2 |
|------|

| |
|------|
| out3 |
|------|

| |
|------|
| out4 |
|------|

{dplyr} rowwise

```
# {dplyr} の rowwise を使用
diamonds_with_fig_rowwise =
  diamonds %>%
  nest_by(cut) %>%
  mutate(
    fig =
      list(
        data %>%
          ggplot(aes(x = carat, y = price)) +
          geom_point(color = "orange")
      )
  )
```

{purrr} map

```
# {purrr} の map を使用
diamonds_with_fig_map =
  diamonds %>%
  group_nest(cut) %>%
  mutate(
    fig =
      data %>%
      map(
        ~
        .x %>%
        ggplot(aes(x = carat, y = price)) +
        geom_point(color = "orange")
      )
  )
```



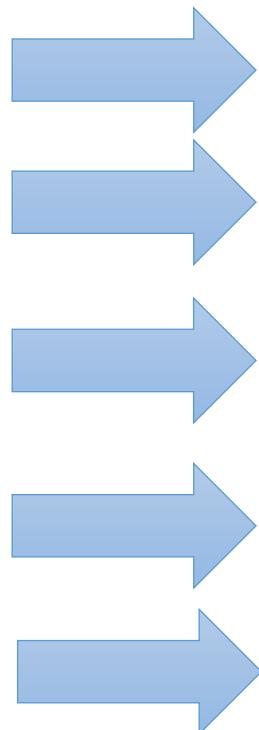
時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|--|-------------|
| 1 | tibble | A dark blue hexagonal logo for tibble, featuring a grid of colored squares and the word "tibble" at the top. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagonal logo for dplyr, featuring a stylized orange and yellow gear-like shape with the word "dplyr" at the top. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagonal logo for tidyr, featuring a colorful 3D bar chart with the word "tidyr" at the top. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagonal logo for ggplot2, featuring a line graph with three blue dots connected by lines, with the word "ggplot2" at the bottom. | 可視化 |
| 5 | stringr | A green hexagonal logo for stringr, featuring a white icon of a violin and the word "stringr" at the bottom. | 文字列処理 |
| 6 | readr | A blue hexagonal logo for readr, featuring a white icon of a document and the word "readr" at the bottom. | 入出力処理 |
| 7 | forcats | A brown hexagonal logo forforcats, featuring a black cat icon and the word "forcats" at the bottom. | ファクター処理 |
| 8 | purrr | A white hexagonal logo for purrr, featuring a white icon of a cat's head and the word "purrr" at the bottom. | 繰り返し処理 |



| No. | 関数 | 用途 |
|-----|-------|-----------|
| 1 | map | 行処理(引数1つ) |
| 2 | pmap | 行処理(引数複数) |
| 3 | pwalk | 行処理(保存) |

| cut | data |
|-----------|--------------|
| Fair | df_Fair |
| Good | df_Good |
| Very Good | df_Very Good |
| Premium | df_Premium |
| Ideal | df_Ideal |



fig_Fair
fig_Good
fig_Very_Good
fig_Premium
fig_Ideal

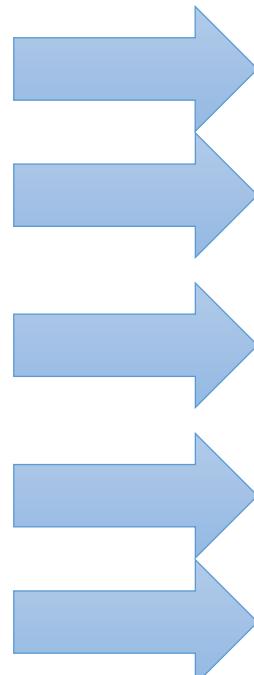
```

diamonds %>%
  group_nest(cut) %>%
  mutate(
    fig =
      data %>%
      map(
        ~
        .x %>%
        ggplot(aes(x = carat, y = price)) +
        geom_point(color = "orange")
      )
    )
  
```

1

2

| cut | data |
|-----------|--------------|
| Fair | df_Fair |
| Good | df_Good |
| Very Good | df_Very Good |
| Premium | df_Premium |
| Ideal | df_Ideal |



fig_Fair
fig_Good
fig_Very_Good
fig_Premium
fig_Ideal

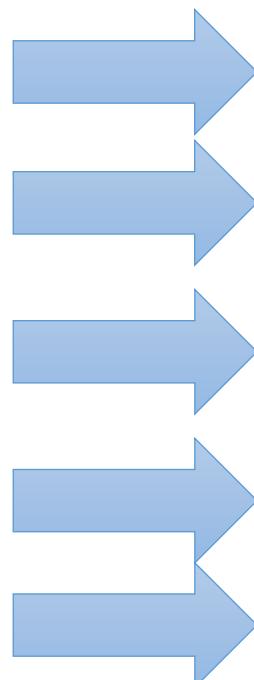
```

diamonds %>%
group_nest(cut) %>%
mutate(
  fig =
  list(cut, data) %>%
  pmap(
    ~
    ..2 %>%
    ggplot(aes(x = carat, y = price)) +
    geom_point(color = "orange") +
    labs(title = ..1)
  )
)
  
```

1

2

| cut | data |
|-----------|--------------|
| Fair | df_Fair |
| Good | df_Good |
| Very Good | df_Very Good |
| Premium | df_Premium |
| Ideal | df_Ideal |



```

diamonds %>%
group_nest(cut) %>%
pwalk(
~ ..2 %>%
write_csv(str_c(..1,
"_data.csv")))
)

```

{lubridate}時間処理

{lubridate}ってなんですか？？



{lubridate}は、時間処理に特化したパッケージのことだよ。

時間処理は簡単そうに見えるけど、実際に処理しようとすると色々考えることが多くて難しいんだ。．。

ややこしい例

1. 通常は1年は365日だが、うるう年は1年は366日
2. 通常は1日は24時間だが、サマータイム時は25時間
3. 時差の影響により、国々の時刻は異なる。

時間型

- 1.date型 : 日付を表す.
- 2.datetime型 : 日付と時刻を表す.

※baseRでは、時刻を表す型として、POSIXc型、
POSIXlt型があるが、これらの使い方は覚えなくてOK



時間処理

| No. | コアパッケージ | ロゴ | 用途 |
|-----|---------|---|-------------|
| 1 | tibble | A dark blue hexagon with a grid of colored squares at the bottom, with the word "tibble" written vertically on the right side. | データフレームの進化版 |
| 2 | dplyr | A dark blue hexagon with a stylized orange and yellow gear-like shape, with the word "dplyr" written vertically on the left side. | データフレーム処理 |
| 3 | tidyr | A dark blue hexagon with a colorful geometric pattern of lines and shapes, with the word "tidyr" written vertically on the left side. | tidyデータ処理 |
| 4 | ggplot2 | A white hexagon with a black outline, containing a small blue line plot with three data points, with the word "ggplot2" written vertically on the right side. | 可視化 |
| 5 | stringr | A green hexagon with a white icon of a violin, with the word "stringr" written vertically on the right side. | 文字列処理 |
| 6 | readr | A blue hexagon with a white icon of a document with a bar chart, with the word "readr" written vertically on the right side. | 入出力処理 |
| 7 | forcats | A brown hexagon with a white icon of two cats in a box, with the word "forcats" written vertically on the right side. | ファクター処理 |
| 8 | purrr | A white hexagon with a black outline, containing a white icon of a cat's head, with the word "purrr" written vertically on the right side. | 繰り返し処理 |

ここ ! !



| No. | 関数 | 用途 |
|-----|---|-------------|
| 1 | ▪ymd ▪ymd_hms | 文字列から時間型を作成 |
| 2 | ▪make_date ▪make_datetime | 要素から時間型を作成 |
| 3 | ▪as_date ▪as_datetime | 時間型の変換 |
| 4 | ▪date ▪ydayなど | 要素の取得 |
| 5 | ▪floor_date ▪round_date ▪ceiling_date | 時間型の丸め処理 |
| 6 | ▪dseconds ▪secondsなど | 時間型の計算 |

y:year(年)
m:month(月)
d:day(日)
h:hour(時)
m:minute(分)
s:second(秒)

date型

ymd("2020/12/17")

datetime型

ymd_hms("2020/12/17 10:04:59")

2017/5/17 13:48:27

date

2017/5/17

2017/5/17 13:48:27

year

2017

2017/5/17 13:48:27

month

5

2017/5/17 13:48:27

day

17

2017/5/17 13:48:27

hour

13

2017/5/17 13:48:27

minute

48

2017/5/17 13:48:27

second

27

2017/5/17 13:48:27

yday

137



2017/5/17 13:48:27

wday

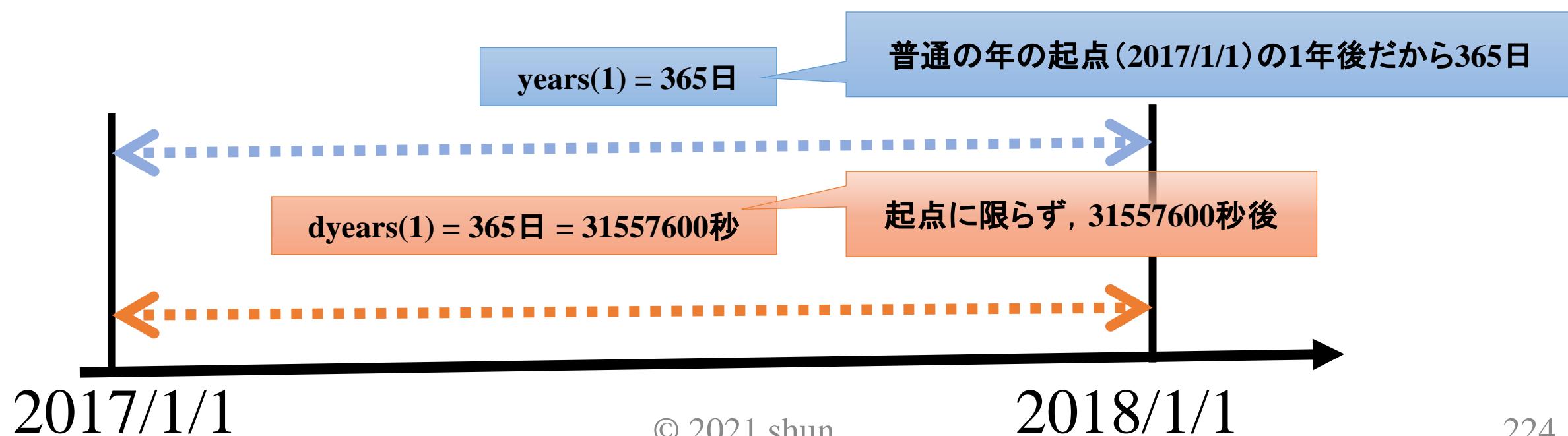
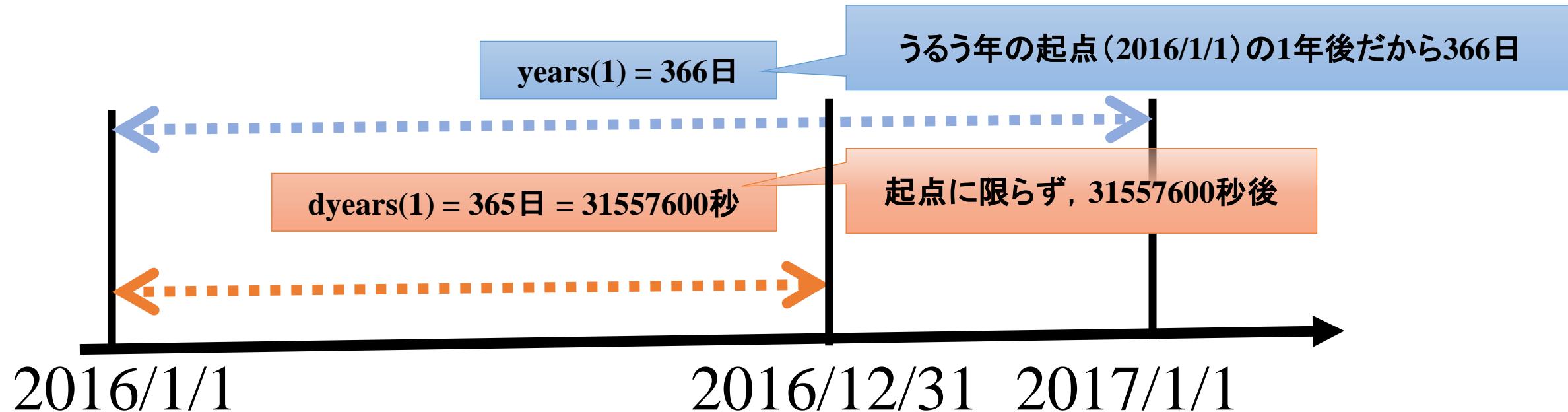
4

Wed (label = TRUE)

今週何日目か

4日





NA処理

NA処理ってなんですか？？



データの欠損値に対する処理のことだよ。

欠損値はNAで表されて、NAのままだと扱いづらいから処理する必要があるんだ。

処理の方法は色々あるから1つずつ見てみよう。

| 性別 | 年齡 | 体重 | 身長 |
|----|----|----|-----|
| 男性 | 35 | 55 | 172 |
| 女性 | 24 | 48 | NA |
| 女性 | 19 | NA | 153 |
| 男性 | 63 | 51 | 164 |
| 男性 | NA | 73 | 178 |

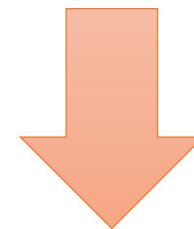


| No. | 関数 | 用途 |
|-----|------------|---------|
| 1 | drop_na | NA削除 |
| 2 | replace_na | NA置換 |
| 3 | fill | NAパディング |

総合演習

目的

今までに学習したことを利用し、実務でよく使用する前処理を経験することで、データの前処理に関する理解を深める！！



実践

映画のデータを使用して、総合演習！！

info_movies

info_platforms

| # | 列名 | 意味 |
|----|-----------------|-------------------------|
| 1 | ID | 主キー |
| 2 | Title | タイトル |
| 3 | Year | 上映年 |
| 4 | Age | 推奨年齢 |
| 5 | IMDb | 映画のデータベースによる点数(0~10で評価) |
| 6 | Rotten_Tomatoes | 映画評論サイトによる点数(0~100%で評価) |
| 7 | Directors | 監督 |
| 8 | Genres | ジャンル |
| 9 | Country | 上映国 |
| 10 | Language | 言語 |
| 11 | Runtime | 上映時間(単位:分) |

| # | 列名 | 意味 |
|---|-------------|----------------------|
| 1 | ID | 主キー |
| 2 | Age | 推奨年齢 |
| 3 | Netflix | Netflixで放映しているか. |
| 4 | Hulu | Huluで放映しているか. |
| 5 | Prime_Video | Prime_Videoで放映しているか. |
| 6 | Disney | Disneyで放映しているか. |

問題

0. デスクトップに「execise」フォルダを作成し、その配下にプロジェクトファイルを作成しましょう。
 1. 以下を実施し、データを整理しましょう。
 - ① info_movies, info_platformsをそれぞれ縦に結合する。
 - ② ①でできたデータを内部結合(inner join)する。
 - ③ ②でできたデータに以下の処理をする。
 - ・欠損値(NA)がある行を削除
 - ・Netflix, Hulu, Prime_Video, Disneyに対し、0, 1以外の行を削除
 - ・Age列をファクターに変換(levelは、7+, 13+, 16+, 18+, allの順)
 - ・IMDb列を10倍
 - ・Rotten_Tomatoes列の%を取って、数値型に変換
 - ④ ③でできたデータをrdsにして保存する。
- ※以降の問題は、1.で作成したデータを使用してください。

問題

2. Directorsごとの映画の作成数を算出し、作成数の降順(大きい順)にソートしましょう。
3. プラットフォームの配信パターンごとにIMDbとRotten_Tomatoesの平均値を算出しましょう。
4. Country列からそれぞれの映画が何か国で上映されたか算出しましょう。

問題

5. 各プラットフォームで放映されている作品のうち, それぞれの推奨年齢が何割を占めているか算出しましょう.
6. IMDb列の大きさにより以下の条件で分類し, それぞれについて, Runtimeのヒストグラムを作成しましょう.
 - ・70以上 : high
 - ・50以上70未満 : middle
 - ・50未満 : low
7. AgeごとにIMDb, Rotten_Tomatoes, runtimeの散布図行列を作成しましょう.
※パッケージggpalsを使用すると楽にできます.

注意事項

- ・これらの問題は、前回までのレクチャーを参考にしたり、ググったりしてもOKです。
- ・一部の問題については、前回までのレクチャーで学習していない部分もあります。そのような問題でもググって自分の答えを探してみましょう。
- ・制限時間の目安は3時間です。
あくまで目安なので、3時間を超えても問題ないです。
- ・解説のレクチャーにすぐにいかないで、まずは自分で考えてやってみることをお勧めします。

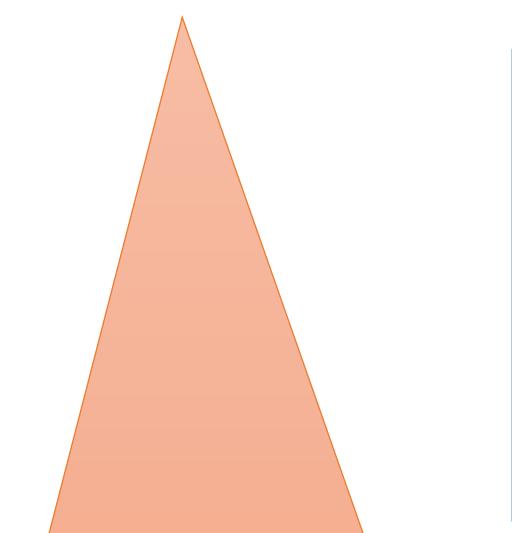
解説:0

0. デスクトップに「execise」フォルダを作成し、その配下にプロジェクトファイルを作成しましょう。

| # | 参考セクション | 参考レクチャー |
|---|---------|---------|
| 1 | 3 | 8 |
| 2 | 4 | 10 |

解説:0

```
docker run -p 8888:8787 -v □□□□:/home/rstudio/exercise  
-e PASSWORD=×××× ○○○○/r-preprocess-lecture
```



□□□□:exerciseフォルダの絶対パス
××××:パスワード
○○○○:DokerHubのID

コマンドプロンプト(ターミナルでこのコマンドを実行)

解説:1

1. 以下を実施し、データを整理しましょう。

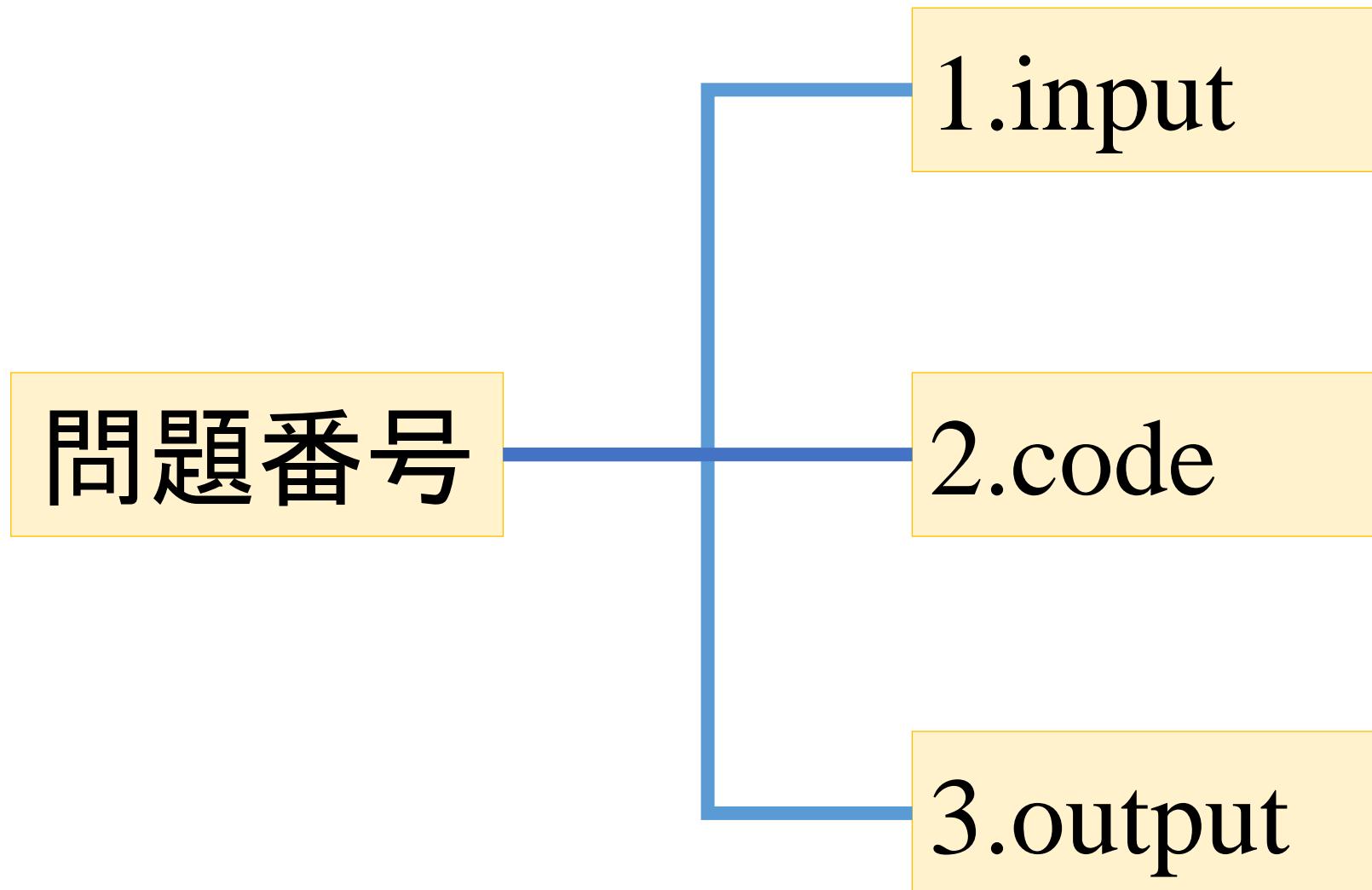
- ① info_movies, info_platformsをそれぞれ縦に結合する。
- ② ①でできたデータを内部結合(inner join)する。
- ③ ②でできたデータに以下の処理をする。
 - ・欠損値(NA)がある行を削除
 - ・Netflix, Hulu, Prime_Video, Disneyに対し、0, 1以外の行を削除
 - ・Age列をファクターに変換(levelは、7+, 13+, 16+, 18+, allの順)
 - ・IMDb列を10倍
 - ・Rotten_Tomatoes列の%を取って、数値型に変換
- ④ ③でできたデータをrdsにして保存する。

※以降の問題は、1.で作成したデータを使用してください。

解説: 1

| # | セクション | レクチャー | パッケージ | 関数 | 用途 |
|----|-------|------------|---------|---------------------------|--------------------|
| 1 | 15 | 56 | tibble | tibble | tibbleの作成 |
| 2 | | | baseR | dir | ファイル名の取得 |
| 3 | 16 | 64, 65 | dplyr | mutate | 列の追加 |
| 4 | 16 | 68 | dplyr | bind_rows | 縦結合 |
| 5 | 16 | 66, 67 | dplyr | inner_join | 内部結合 |
| 6 | 24 | 123 | tidyr | drop_na | NA削除 |
| 7 | 16 | 58, 59, 60 | dplyr | filter | 行の抽出 |
| 8 | 9 | 33 | baseR | %in% | 論理値ベクトルの作成 |
| 9 | 21 | 106 | forcats | fct_relevel | 水準の順番の変更 |
| 10 | 19 | 94 | stringr | str_replace | マッチの先頭置換 |
| 11 | 20 | 99, 100 | readr | ▪ read_csv ▪ write_rds | ▪ CSV入力 ▪ RDS出力 |

補足：フォルダ体系



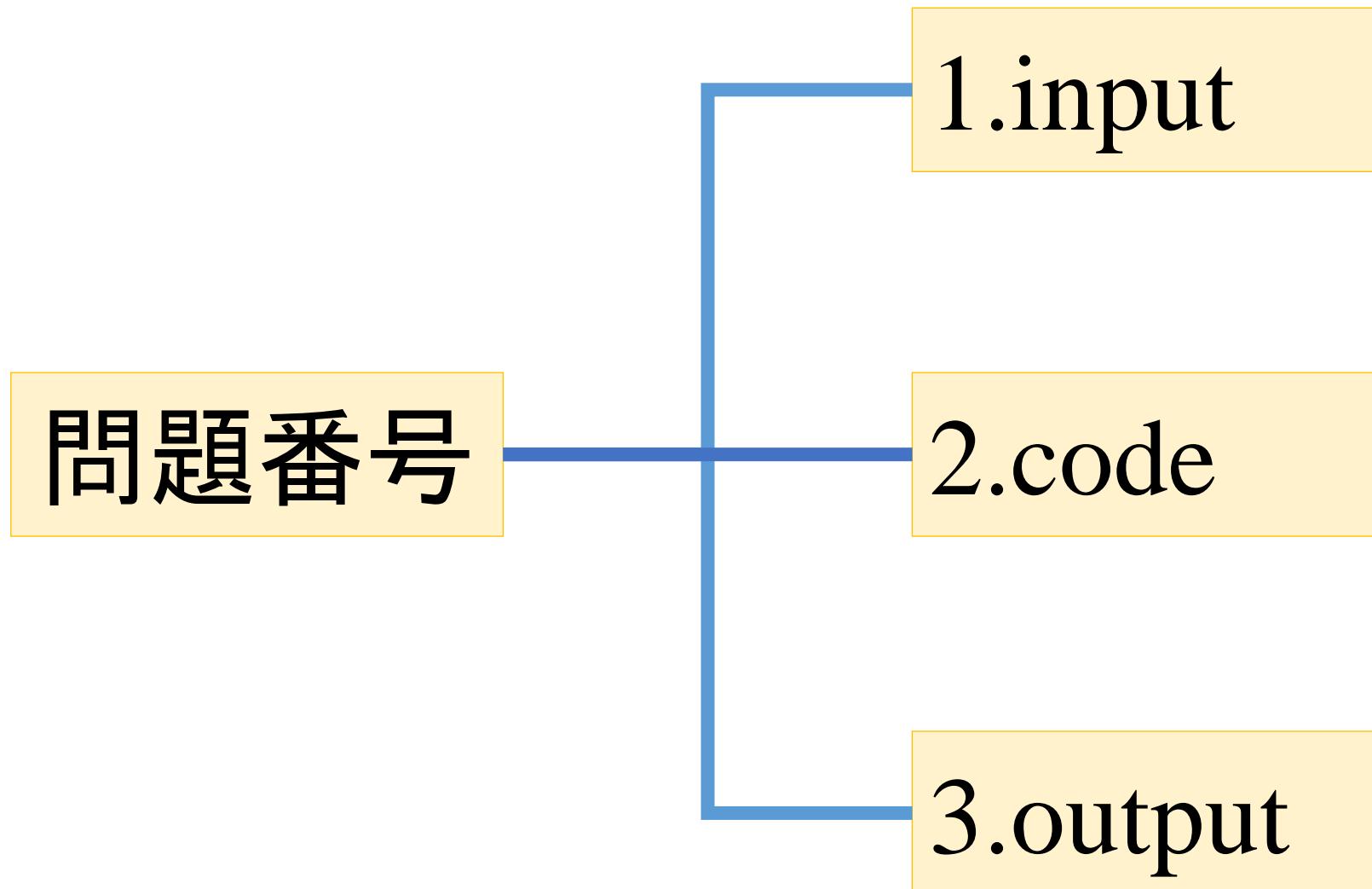
解説:2

2. Directorsごとの映画の作成数を算出し、作成数の降順(大きい順)にソートしましょう。

※Director_1, Director_2のように2人以上の監督がいる場合1人として算出することとしましょう。

| # | セクション | レクチャー | パッケージ | 関数 | 用途 |
|---|-------|--------|---------|------------|--------------|
| 1 | 16 | 72, 73 | dplyr | nest_by | ネストしてrowwise |
| 2 | 16 | 64, 65 | dplyr | mutate | 列の追加 |
| 3 | | | baseR | nrow | データフレームの行数算出 |
| 4 | 16 | 74 | dplyr | arrange | ソート |
| 5 | 19 | 88 | stringr | str_c | 文字列の連結 |
| 6 | | | baseR | dir.create | フォルダの作成 |
| 7 | | | baseR | file.copy | ファイルのコピー |

補足: フォルダ体系



解説:3

3. プラットフォームの配信パターンごとにIMDbとRotten_Tomatoesの平均値を算出しましょう。

| # | セクション | レクチャー | パッケージ | 関数 | 用途 |
|---|-------|--------|-------|----------|--------------|
| 1 | 16 | 72, 73 | dplyr | nest_by | ネストしてrowwise |
| 2 | 16 | 64, 65 | dplyr | mutate | 列の追加 |
| 3 | 8 | 29 | baseR | mean | ベクトルの平均算出 |
| 4 | 16 | 69 | dplyr | group_by | グルーピング |

解説:4

4. Country列からそれぞれの映画が何か国で上映されたか算出しましょう。

| # | セクション | レクチャー | パッケージ | 関数 | 用途 |
|---|-------|------------|---------|-----------|-----------|
| 1 | 16 | 70, 71, 72 | dplyr | rowwise | 行処理 |
| 2 | 16 | 64, 65 | dplyr | mutate | 列の追加 |
| 2 | 19 | 96 | stringr | str_split | マッチの分割 |
| 4 | | | baseR | length | ベクトルの長さ算出 |

解説:5

5.各プラットフォームで放映されている作品のうち, それぞれの推奨年齢が何割を占めているか算出しましょう.

| # | セクション | レクチャー | パッケージ | 関数 | 用途 |
|---|-------|------------|-------|--------------|------|
| 1 | 16 | 70, 71, 72 | dplyr | nest_by | 行処理 |
| 2 | 16 | 64, 65 | dplyr | mutate | 列の追加 |
| 3 | 16 | 61, 62, 63 | dplyr | select | 列の抽出 |
| 4 | 17 | 78 | tidyr | pivot_longer | 縦変換 |
| 5 | 17 | 78 | tidyr | pivot_wider | 横変換 |

解説:6

6. IMDb列の大きさにより以下の条件で分類し、それについて、Runtimeのヒストグラムを作成しましょう。

- ・70以上 :high
- ・50以上70未満 :middle
- ・50未満 :low

| # | セクション | レクチャー | パッケージ | 関数 | 用途 |
|---|-------|------------|-----------|----------------|----------|
| 1 | 16 | 70, 71, 72 | dplyr | nest_by | 行処理 |
| 2 | 16 | 65 | dplyr | case_when | 条件分岐 |
| 3 | 21 | 106 | forcats | fct_relevel | 水準の順番の変更 |
| 4 | 18 | 83 | ggplot2 | geom_histogram | ヒストグラム作成 |
| 5 | 17 | 83 | ggplot2 | geom_density | 密度曲線作成 |
| 6 | 16 | 72 | patchwork | wrap_plots | グラフの並び調整 |

解説:7

7. AgeごとにIMDb, Rotten_Tomatoes, runtimeの散布図行列を作成しましょう。

※パッケージGGallyの関数ggpairsを使用すると楽にできます。

| # | セクション | レクチャー | パッケージ | 関数 | 用途 |
|---|-------|-------|--------|---------|---------|
| 1 | 18 | 84 | GGally | ggpairs | 散布図行列作成 |