

# RとRstudio導入

(資料最終更新日：2025-12-02)

資料作成：杉野弘明

Mail >>> [hsugino@yamaguchi-u.ac.jp](mailto:hsugino@yamaguchi-u.ac.jp)

Twitter (現：X) >>> [@hiruandondemo](https://twitter.com/@hiruandondemo)

Instagram >>> [hiroaki\\_teriyaki\\_sugino](https://www.instagram.com/hiroaki_teriyaki_sugino)

# お品書き

本資料はR言語の概要から、R/Rstudioの導入、初步的な使い方までの学習をサポートするための資料です。内容は以下の通り：

- 資料導入
- R入門
- 環境構築
- R基本操作
- データ取り込みとその後の処理

# 資料導入

# 資料の目的と概要

- ・本資料は、Rと呼ばれる代表的な統計分析ツールをこれから始める人が、参考しながら自分のペースで導入できることを目指したものです。
- ・インストール段階から始める方も、インストールはしているけれども、R(とRstudio)を利用して統計解析を本格的に始める準備運動をしたい方も、本資料内容を一読して、スタートを切って頂ければと思います。
- ・資料内の後半では、実際に手を動かして分析コードを書きながら資料を追うことができる実習パートも付けています。合わせてサンプルスクリプトなども用意しているので、そちらも参照されてください。

# まずは、お伝えしておきたいこと

- 人間ができることには制約があるが、末端の複雑な計算をしてくれる関数や出力結果、装置を組み合わせて(End-to-End的な考え方)、応用し、社会課題の解決を行うことができるといったAIやコンピュータにはできないことができる。
- 若い方には、特にそこを強く意識して取り組んで欲しいと思っています。パラダイムやプラットフォームが変わった時に、適応していかなければなりません。適応するためには、まず自分で手を動かして使い込んでみる、ことが重要です。
- 社会調査法やデータサイエンス、それらに関する解析は、現在のコンピューティングの発達に大きな恩恵を受けている部分が大きいです。便利になった時代だからこそ、それらをEnd-to-Endでつなげて、何を皆さんのが分析したいのか、するのかが問われる時代です。一通り手法や技術を身に着けた後には、是非、皆さんの個々の研究課題に立ち返り、どのような分析をするべきなのかを考えていってみてください。

# R入門

# 本資料に関連するソフトウェア

- 1) Word / Excel / Powerpoint

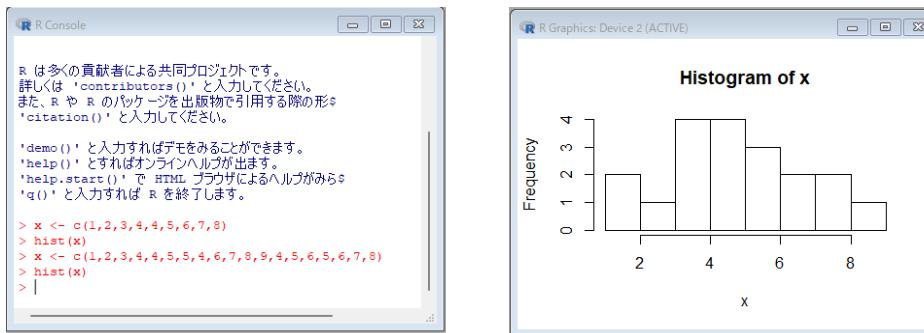
いわゆると知れたMicrosoft社のプロプライエタリ・ソフトウェア。  
レポート作成、表計算や分析用データの整形、  
発表資料作成に使用。

- 2) R + Rstudio

世界中で使用されている統計解析用オープン・ソフトウェア。  
統計解析やグラフなどの作成を行うために使用。

# Rとは何か？

- ・ 統計解析や図の可視化のためのソフトウェア
- ・ 無償のオープンソース・ソフトウェア
- ・ 「パッケージ」と呼ばれる便利な解析機能等をまとめたものにより機能拡張が可能
- ・ プログラムを書いて実行



<https://www.r-project.org/>

# R言語の歴史①

- R言語は、1976年にAT&Tベル研究所で開発された統計処理言語の「S言語」の仕様を受け継ぎ、関数型言語のSchemeを影響を受けて開発されました。オープンソフトウェアとしては1995年に公開されました。
- Rという名前は二人の開発者 Ross Ihaka と Robert Clifford Gentleman のイニシャルに由来します。
- 当時の設計理念、そして現在に至るまで受け継がれている大きなところとして、「持続可能な統計環境」というものがあります。R言語の登場以前は、統計データの処理環境は高額なプロプライエタリ・ソフトウェアばかりでした。これでは、環境や時代の変化によって利用可能な計算資源が変わってしまい、その結果すらソフトウェアの版権保持者とお金に左右されるという結果になりかねませんでした。

## R言語の歴史②

- そこで、オープンソースによって全ての人に開かれておりながら、パッケージと呼ばれる拡張機能により、日々進化し続けるような、つまり「永続的で世界規模の集合知に支えられ、無償でありながら高い信頼に値する統計環境」の実現を目指して、R言語が開発されました。
- 以上のようにオープン性と信頼性において、他のプログラミング言語と比較すると研究に特化しているイメージがあるが、企業など利用されている現場は数多く、現在は書籍やWeb上で参照できる資料も多い。

# 補足：関数型プログラミング言語とは？

- 関数型のプログラミング言語は、**RやHaskell、Python(の一部)、JavaScript(の一部)**が該当する、人工言語の一種です。
- 関数型の前にあった素朴なプログラミング言語（例えば、古いC言語など）は**手続き型**とも呼ばれ、処理の手順を素朴に上から下にどんどん書いていくスタイルのものでした。
- 料理で例えるならば、「まずボールを手元を持ってきて、次の卵を見て、その卵を両手で持って、ボールの上で割って、ゴミ箱を見て、卵の殻をゴミ箱似れて、、、」みたいな感じ

# 補足：関数型プログラミング言語とは？

- しかし、やはりそれだと全てをいちいち書かないといけないので、複雑な手順になってくると、**人間にとってとても大変**でした（コンピューターにとってはどうってことないことではありました）。
- なので、**小さな（もしくは大きな）一定の決まった動きの手順書**ないしはレシピ（これを関数と呼びます）を作り、それらを組み合わせて**複雑な分析や処理を行うタイプ**のプログラミング言語、すなわち関数型のプログラミング言語が生まれました。
- 料理で例えるならば、「卵を割る動きのレシピ（関数）」、「卵を混ぜる動きのレシピ（関数）」、「卵を焼く動きのレシピ（関数）」などを組み合わせて、卵焼きを完成させる、といったイメージです。

# 補足：関数型プログラミング言語とは？

- 関数型プログラミング言語の利点は以下の通りです：
  1. 同じ関数、同じ入力（データ）なら、常に同じ出力になる
  2. データそのものに処理を加えないで、元データを壊すリスクが低い
  3. 繰り返しの処理が簡単になる
  4. 関数があることで、人間にとて分析工程を整理しやすい
- これらの理由から、入力と出力およびコードそのものに**再現性が担保され**、研究や教育の場面で共有する際や、**論文を書く際に重宝される**というわけです。

# 補足：関数の基本構造

- Rやその他のプログラミング言語においても、関数の書き方はそこまで違いはありません。基本的には、関数を書いた後に( )で閉じた中に、引数を入れます。関数が英語などにおける動詞Vだとしたら、引数はその動作の対象や条件となる目的語Oや補語Cのようなものになります。

**関数名(引数1, 引数2, 引数3, ···)**

\* 引数は“ひきすう”と読みます。英語だとargument

\* ただし、言語によっては引数の指定の仕方や、並べ方、区切り方などが違う場合があるので注意してください。

# R言語の特徴など

- 主要な他のプログラミング言語との比較表は以下の通り：

比較ポイント	R	Python	Julia	C
得意なこと	グラフ・統計の分析	汎用性重視(AI・Web接続なども)	数学・物理の計算	コンピュータの細かい動きを作る
覚えやすさ	統計を学ぶ人には親しみやすい	全体的にわかりやすく初心者向き	少し難しいけど、数式に強い人向き	難しい(細かく書かないといけない)
計算スピード	ふつう	ふつう	かなり速い	とても速い
グラフの作りやすさ	とても強い(きれいなグラフが簡単に)	強い(たくさんのライブラリがある)	まあまあ	弱い(全部自分で作らないといけない)
人気と情報の多さ	中くらい(研究者に人気)	とても多い(世界中で使われている)	少なめ(これから人気が出るかも)	多い(昔からある定番の言語)
おすすめな人	統計を勉強するならおすすめ	プログラミング初心者にもおすすめ	理系で計算に興味がある人ならアリ	初心者にはちょっと難しい

# Rの利点

- 1) 自由で開かれていること（オープンソース・ソフトウェア）  
    <－> プロプライエタリ・ソフトウェア (SPSSやExcel)
- 2) 行列計算が得意    <－> Excel (大規模な行列計算は苦手)
- 3) Publication Readyな綺麗な図を比較的容易に描画可能
- 3) スクリプトを書いて残せる
  - 再現性の担保 (研究者としては重要な点)
  - 共同作業
  - 自分のための備忘録
  - スクリプトの再利用

研究発表や論文用の  
データハンドリングや  
可視化はまかせて！



# Rの利点：パッケージの利用

- Rでは、複数の便利な関数がたくさん集まっている「パッケージ」というものを利用します。関数がある種の“魔法”であるとするならば、パッケージはその魔法がたくさん書かれている“魔導書”といったところでしょうか。
- また後程説明を行いますが、Rのようなオープンソース・ソフトウェアの**パッケージ（魔導書）はウェブを介して自由に発注する（手に入れる）ことができます**。このパッケージを発注し、自分のパソコンに入れることを「インストール(install)」と呼びます。
- また、魔導書は発注だけした状態で、**本棚などに積読していても効果を發揮する魔法（関数）を使うことはできません**。RやRstudioを立ち上げる度に、使う魔導書を手元において“開いて”おかなければなりません。この作業を「ライブラリー(library)」と呼びますので、覚えておきましょう。

# Rの利点：パッケージの利用

- R上での魔法の使い方：

インストール：  
ウェブから発注して、自分のパソコンに入れる



\* インストールしただけだと、魔法は使えない



ライブラリー：  
魔導書を開いて、魔法を使える状態にする



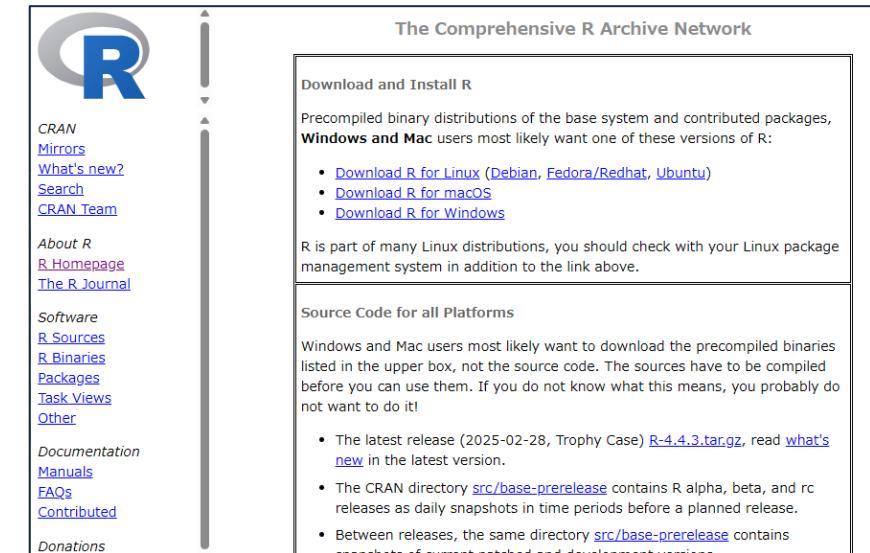
\* 魔法（関数）を使って、いろいろな便利なことを実行する！



# CRAN(シーランorクラン)について

- CRAN : The Comprehensive R Archive Network
- R本体の配布ならびにパッケージの登録と管理を行う場所
- 世界中にミラーサイトが存在している = どこかが潰れても（極端な話本家本元が潰れても、世界に知が残り続ける）

<https://cran.r-project.org/>



# Rstudioについて

- Rをより使いやすくするための統合開発環境(IDE: Integrated Development Environment)。←裏でRを走らせるので、Rもインストールが必要。
- メモ帳機能にもなり、研究そのものがやりやすくなります。
- MacとWindowsでインターフェイスが変わらない。
- 日本語入力が若干苦手で、時々文字化けたりするお茶目なやつです。

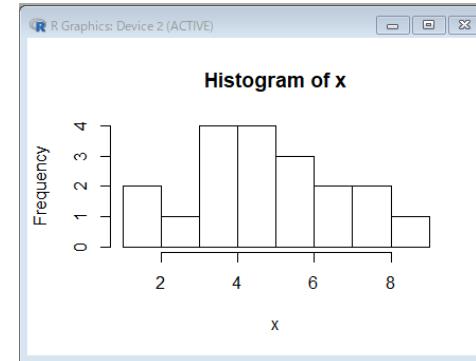
```

R Console
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を
'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。

> x <- c(1,2,3,4,4,5,6,7,8)
> hist(x)
> x <- c(1,2,3,4,4,5,5,4,6,7,8,9,4,5,6,5,6,7,8)
> hist(x)
>

```



```

install.packages("changepoint")
library(changepoint)
x = read.csv("mobile_data_0822_1111.csv", header = TRUE)
View(x)
head(x)
answerm = changepoint(x, method = "BIC")
plot(answerm, xlab="Time", ylab="Value")
print(answerm)

```

The RStudio interface includes:

- Code Editor: Shows the R code used to load data and perform analysis.
- Console: Displays the output of the R code, including the results of the `changepoint` function.
- Plots: A time series plot showing data points and a red horizontal line indicating a change point at approximately Time 40.
- Environment: Shows the current workspace variables and their values.

# Rstudioについて

一昔前（杉野が学生時代だった時）は素朴なRしかなかった

<<<メモ帳にスクリプトを書いて、Rに貼り付けて実行していた



今はめっちゃ開発しやすい！



# Rstudioについて：その他の機能や隠れた利点

- Rstudioの機能まとめ
  - コード入力補完：ほんと、いつもお世話になっております
  - シンタックスハイライト：特定の記号などをわかりやすく異なる色やフォントで表示
  - プロジェクト毎にスクリプトを複数管理することが可能
  - 便利なショートカット完備（Ctrl/Command+Sで保存など、標準的なものも利用可）
- 実はPythonも動かせます。Rstudioの内部で実行まで完結するのでPython使いの方にも実はオススメ

# 統計学とは？

統計学：

知りたい事についてのデータを取得し、それを要約し、またし  
りたいものを推測するための方法論\*

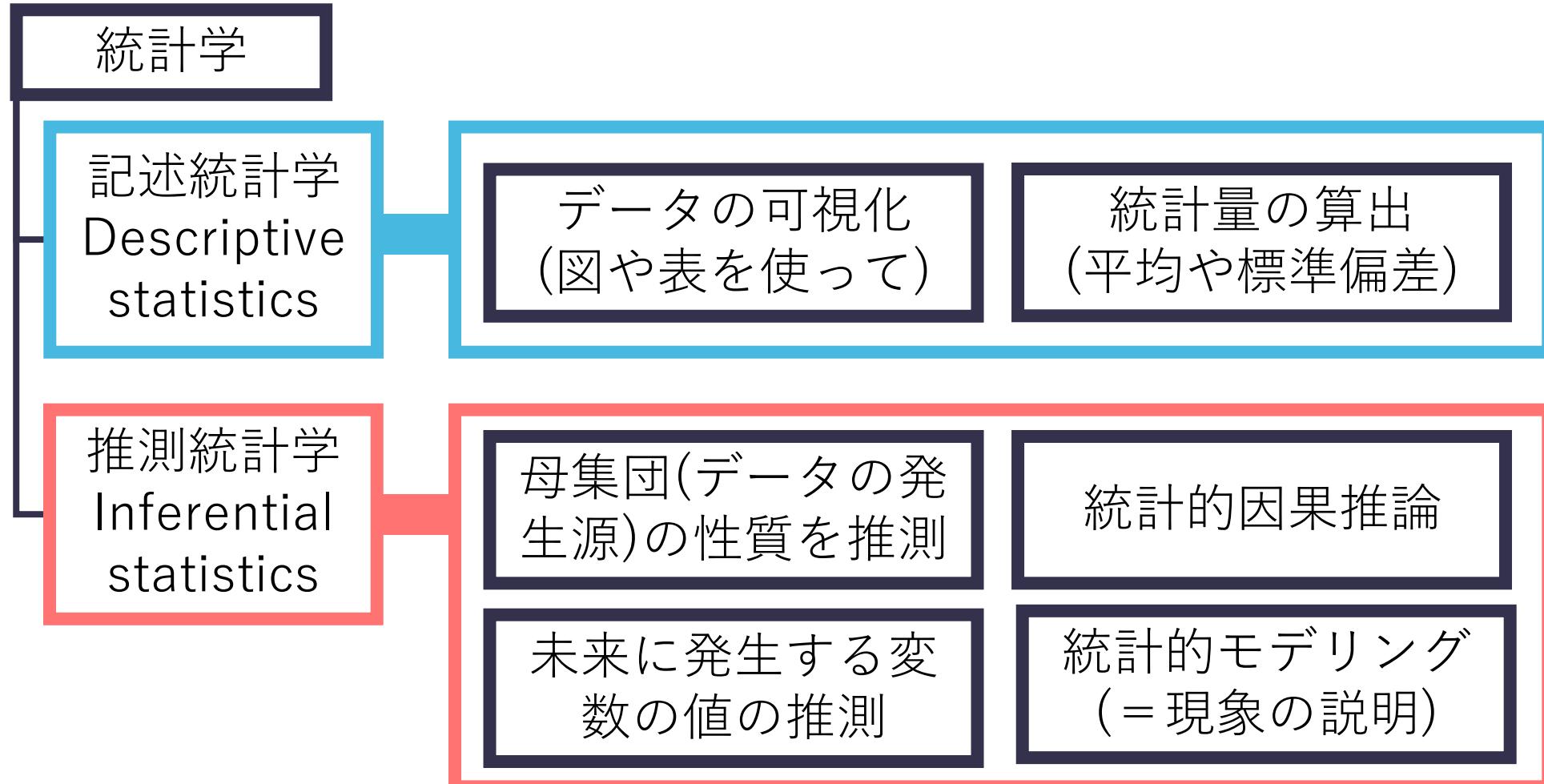
\* データ(data)とは、広義には現実に存在するあらゆる資料や記  
録、調査結果など、情報を符号化したもの全て。

★統計学が想定するデータは、主にカテゴリや数値によってコー  
ディングされた情報のこと。

\* 清水裕士編著(2021)「心理学統計法」

# 統計学とは？

最近は、、、



多变量解析

可視化

機械学習

などなど含めて  
データサイエンス  
と呼ばれますね

# データ入手の重要性

- ところで、学術研究や統計学、データサイエンスと呼ばれるものの根幹になっているデータは、その辺に適当に落ちていたりするのでしょうか？すくなくとも私自身はデータがその辺に転がっているところを見たことがありません。
- 自然界にデータはありません。**データは、世界が有する意味や情報を、誰かの目と脳と手と足と場合によってはその他の部位も動かして、“データ化する”という営みによって存在するようになるものです。
- 皆さんが大学で学ぶ研究という活動は、先人達がありがたくもデータ化してくれたものと、自分自身がデータ化したもの組み合わせながら、新しい知見（これも未来においては貴重なデータになることでしょう）を世に出す行為です。**尊いです。**
- 研究の分野や方法によっては、これまで取られたデータを分析する（これをメタデータ解析と言います）こともありますが、本講義では、**皆さん自身の手で、世界に溢れる情報を量的にデータ化する方法を学んでもらえたら**と思います。

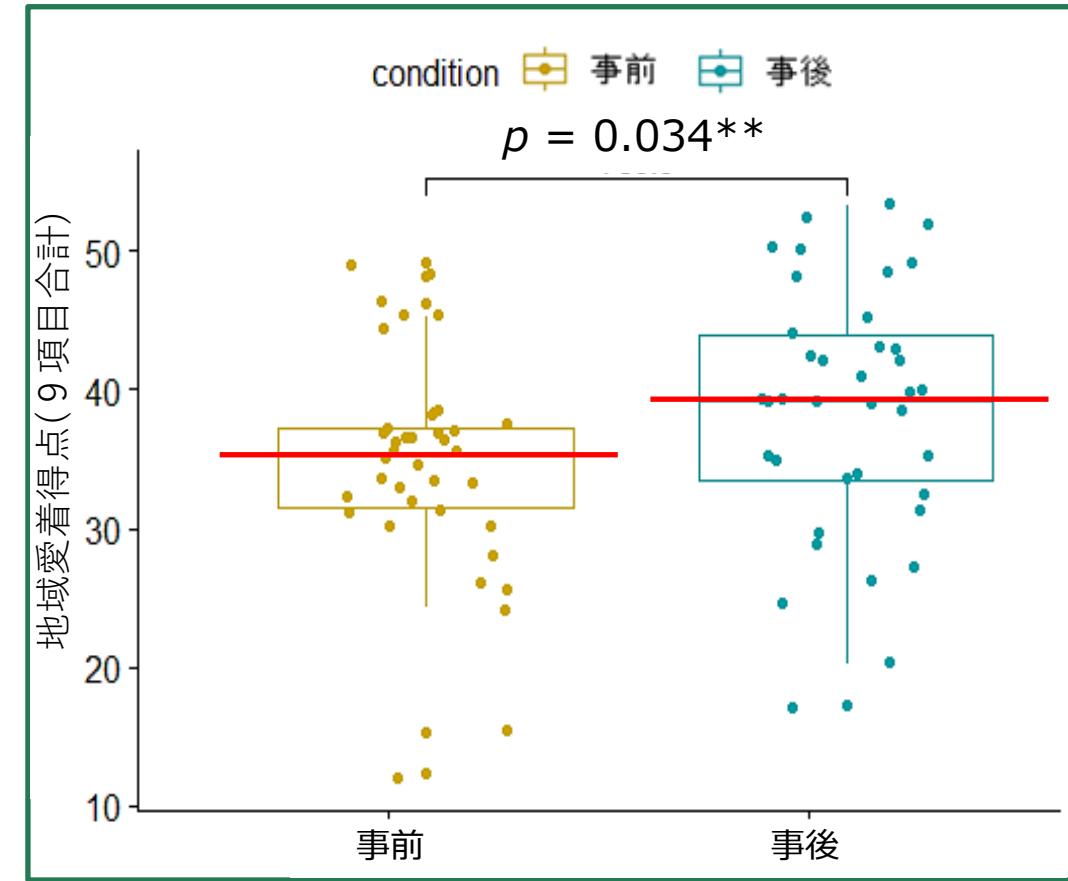
# 心構え

- Rをはじめさまざまな統計ソフトウェアを使いこなすために必要な知識

- ✓ 統計学
- ✓ プログラミング能力
- ✓ ステキな研究題材
- ✓ 諦めない心

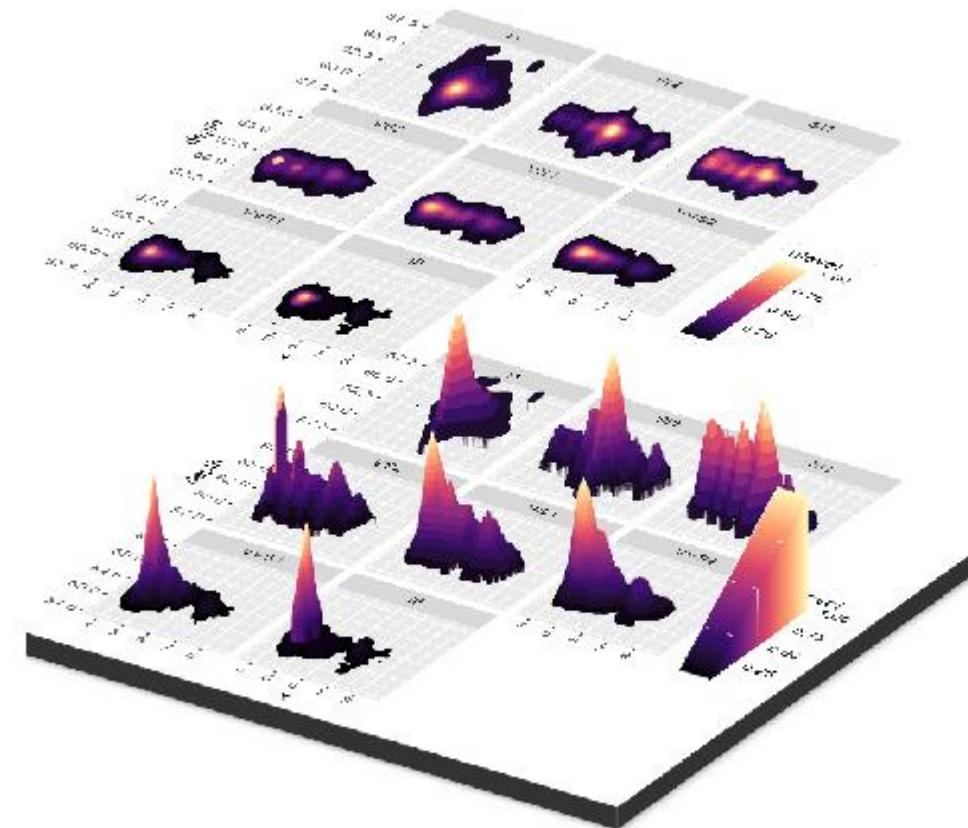
# こんなこともできるよ

- 研究で使える図の作成
- なんか映える図も作れる
- QRコードも作れる
- 簡単なゲームも作れる
- 地図データも使える
- 可視化とかめっちゃ得意



# こんなこともできるよ

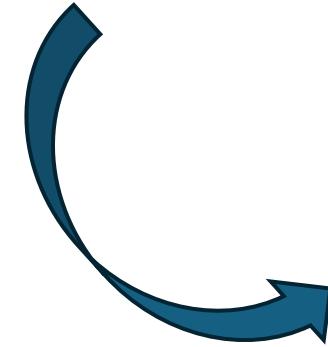
- 研究で使える図の作成
- なんか映える図も作れる
- QRコードも作れる
- 簡単なゲームも作れる
- 地図データも使える
- 可視化とかめっちゃ得意



# こんなこともできるよ

- 研究で使える図の作成
- なんか映える図も作れる
- QRコードも作れる
- 簡単なゲームも作れる
- 地図データも使える
- 可視化とかめっちゃ得意

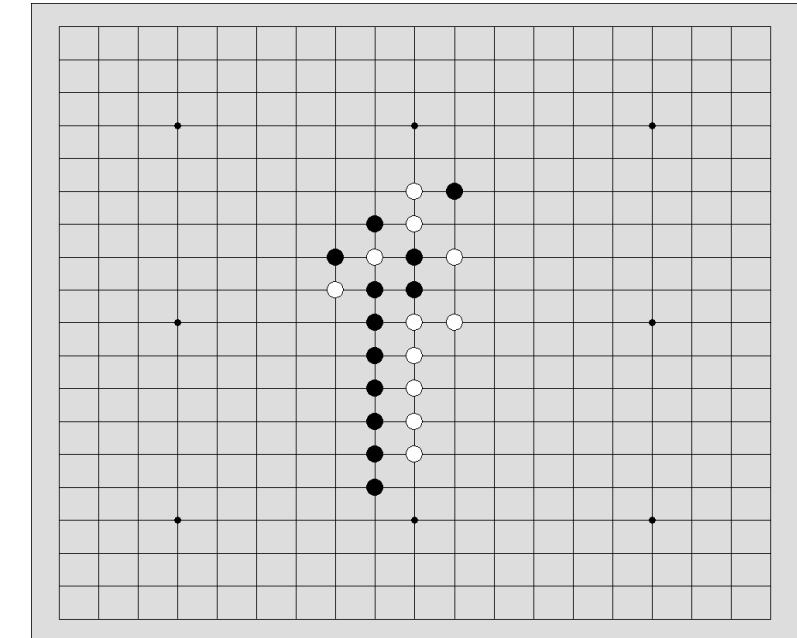
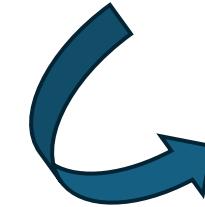
```
1 install.packages("qrcode")
2 library(qrcode)
3 plot(qr_code("https://www.kaggle.com/"))
4
5
```



# こんなこともできるよ

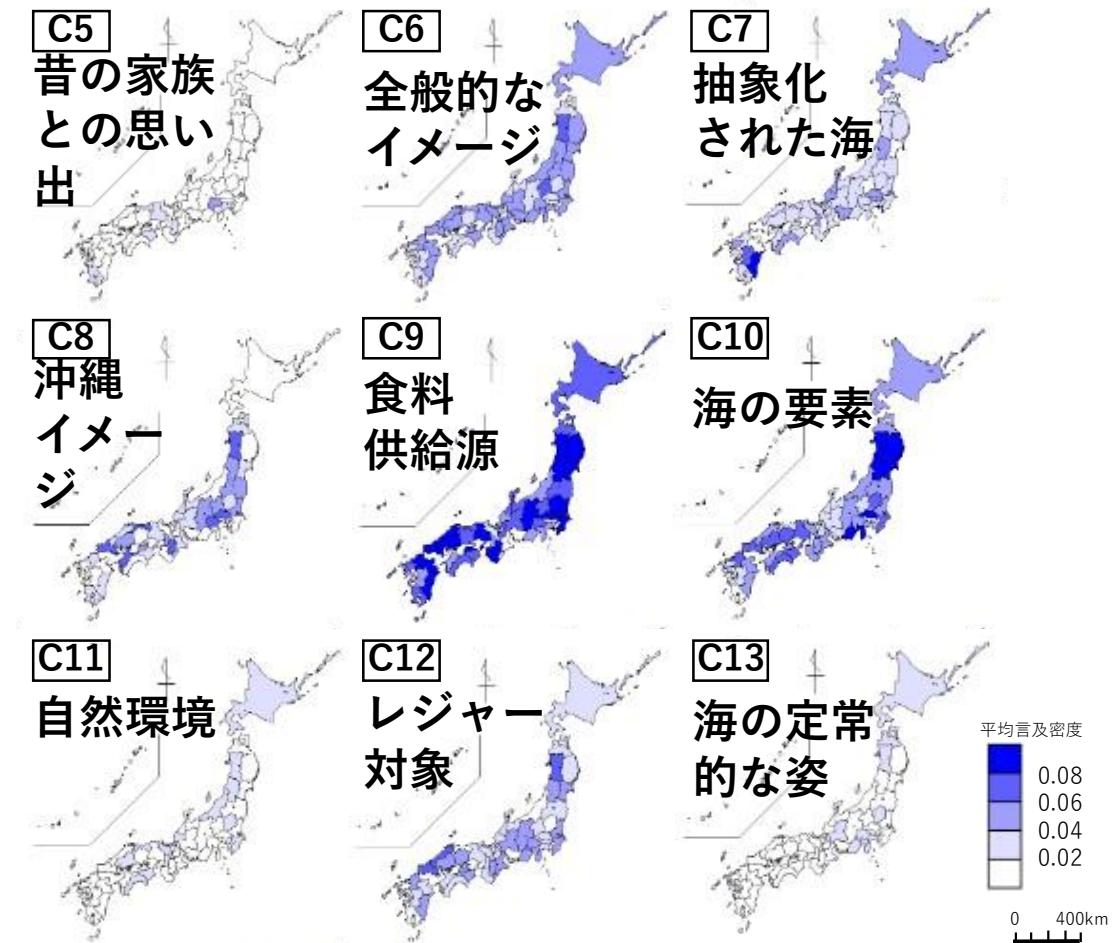
- 研究で使える図の作成
- なんか映える図も作れる
- QRコードも作れる
- 簡単なゲームも作れる
- 地図データも使える
- 可視化とかめっちゃ得意

```
6  
7 install.packages("fun")  
8 library(fun)  
9 gomoku(n=19)  
10 |
```



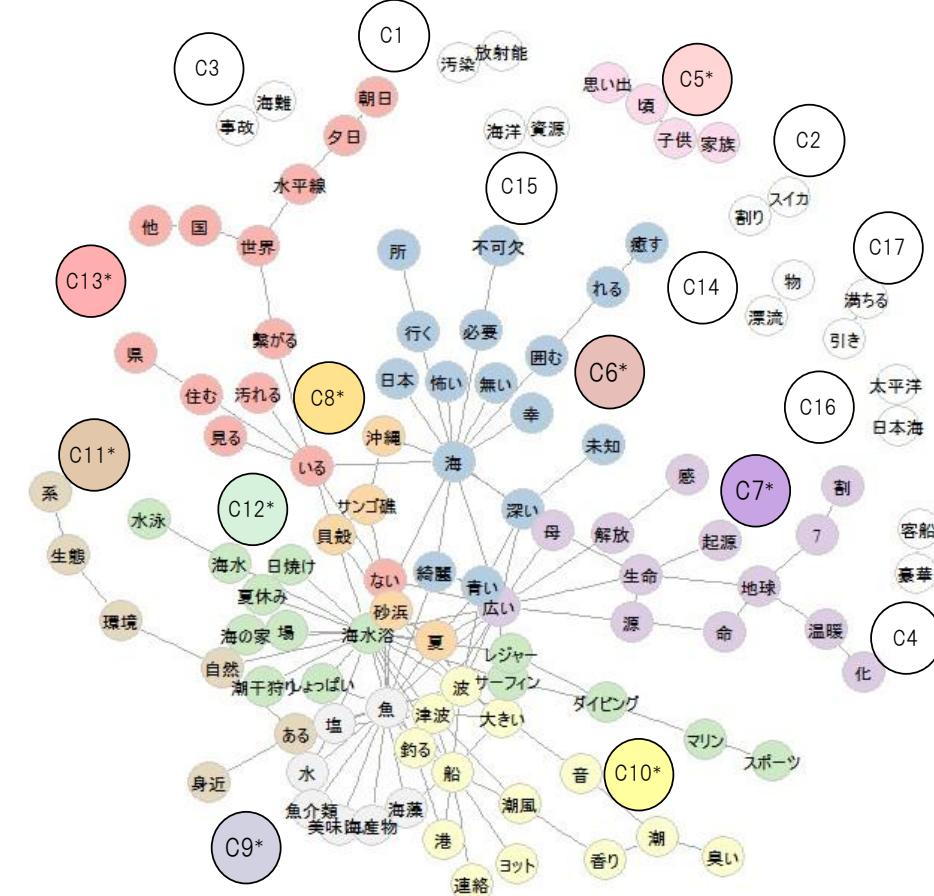
# こんなこともできるよ

- ・研究で使える図の作成
- ・なんか映える図も作れる
- ・QRコードも作れる
- ・簡単なゲームも作れる
- ・地図データも使える
- ・可視化とかめっちゃ得意



# こんなこともできるよ

- ・研究で使える図の作成
- ・なんか映える図も作れる
- ・QRコードも作れる
- ・簡単なゲームも作れる
- ・地図データも使える
- ・可視化とかめっちゃ得意



# 環境構築

# インストールが必要なものの

1. R本体
2. Rstudio
  - Rをより使いやすくするための開発環境
  - 裏で公式のRを走らせるので、Rもインストールが必要です。
  - メモ帳機能にもなり、研究そのものがやりやすくなります。
  - MacとWindowsでインターフェイスが変わらない
  - 日本語入力が若干苦手で、文字化けも多いのが欠点

# インストール時のトラブルシューティング

以下に当てはまる項目がある場合には、次ページを参考に対応を事前に行いましょう。どうやっても自分のコンピュータへのインストールが難しい場合は、Google Colab(oratory)を使うのも手です。

## [ Windows ]

- アカウント名/ユーザー名が日本語である
- アカウント/ユーザーに管理者権限がない
- アプリケーションのデフォルトのインストール先がOneDriveに設定されている

## [ Mac ]

- Mac OSのアップデートを久しく行っていない
- X Codeが最新のものではない

# インストール時のトラブルシューティング

■RやRstudioインストール前のチェック項目に該当があった方や、 実際にインストールができないという方は、 下記を参考にされてください。

- ・ Ryota Mugiyamaさんのウェブサイト 「RとRStudioをインストールするときのつまづきポイントとその対処法へのリンク」 <http://ryotamugiyama.com/2020/08/03/rinstall/>
- ・ Qiita@Maki-Daisuke 「【メモ】 RStudioでパッケージがインストールできない人向けのメモ」 <https://qiita.com/Maki-Daisuke/items/0378626c9bf9971f3822>
- ・ 新保一成先生の研究室ウェブサイト 「日本マイクロソフト社製Microsoft Windowsに固有の問題」 [https://www.fbc.keio.ac.jp/~shimpo/r\\_rstudio\\_problems.html](https://www.fbc.keio.ac.jp/~shimpo/r_rstudio_problems.html)

# 役者をインストールしましょう

- まだR、Rstudioをインストールしていない方は、インストールしましょう。Rstudioが肌に合わない方はRだけでOKですが、私は既にRstudioが無いと生きていけない体になってしまいました。

## [ R ]

- <https://cran.r-project.org/>
- もしくは、ウェブで「R Cran」で検索

## [ Rstudio ] \*Rのインストールが終わった後で

- <https://www.rstudio.com>
- もしくは、ウェブで「Rstudio」で検索

# Rのインストール

Google CRAN R - Google Search

google.com/search?qsrf=ACYBGNQpp6FuL6DTaL5h\_q-xuZ3b31Da5A%3A1573521993695&ei=SQrKXeiJkPhawOZzYHwDQ&q=CRAN+R&oq=CRAN+R&gs\_l=psy-ab.3..35i39j0i203l9.14697.16258..19597...0.1..0.163...

CRAN R

About 20,200,000 results (0.75 seconds)

**CRAN - R Project**  
<https://cran.r-project.org/>

R is part of many Linux distributions, you should check with your Linux package ... CRAN is a network of ftp and web servers around the world that store identical ...

**R for Windows**  
 R-3.6.1 for Windows (32/64 bit).  
 Download R 3.6.1 for Windows ...

**(Mac) OS X**  
 Tools - Old - R for Mac OS X - ...

**Windows**  
 There is also information on third party software available for ...  
 More results from r-project.org »

**Available CRAN Packages for**  
 adeba, Adaptive Density  
 Estimation by Bayesian ...

**CRAN - Contributed Packages**  
 Currently, the CRAN package  
 repository features 15244 ...

**Mirrors**  
<https://nbcgib.uesc.br/mirrors/cran/>, Computational Biology Center ...

People also ask

What is R Cran?

Evernote

個人ノートで類似する結果

Rで項目反応理論 - RjpWiki	R言語 CRAN Task View : 心理モデルや	原油価格推移
2017年5月27日 Rで項目反応理論 http://www.okadajp.org/R R%E3%81%A7%E9%AC% [トップ] Tips紹介 [初級 Q & A   R掲示板   日本 語化掲示板   リンク集] [編集]	2017年2月13日 トライファイルズ 中小 企業経営のためのデータ 分析代行サービス ホー ムサービス 事業概要 ブ ログ サイトマップ お問 い合わせ R言語 CRAN Task View : 心理モデル	2019年10月25日 統計ソフトRの備忘録 2 HomeArchives ② Feb 14 2015 -R 原油価格推 移 gdata, xts, lattice, tseries, vars, MSBVAR, urca, lmtest

See results about

Rnn (Software)  
 Written in: R  
 Original author: Bastiaan Quast

# Rのインストール

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

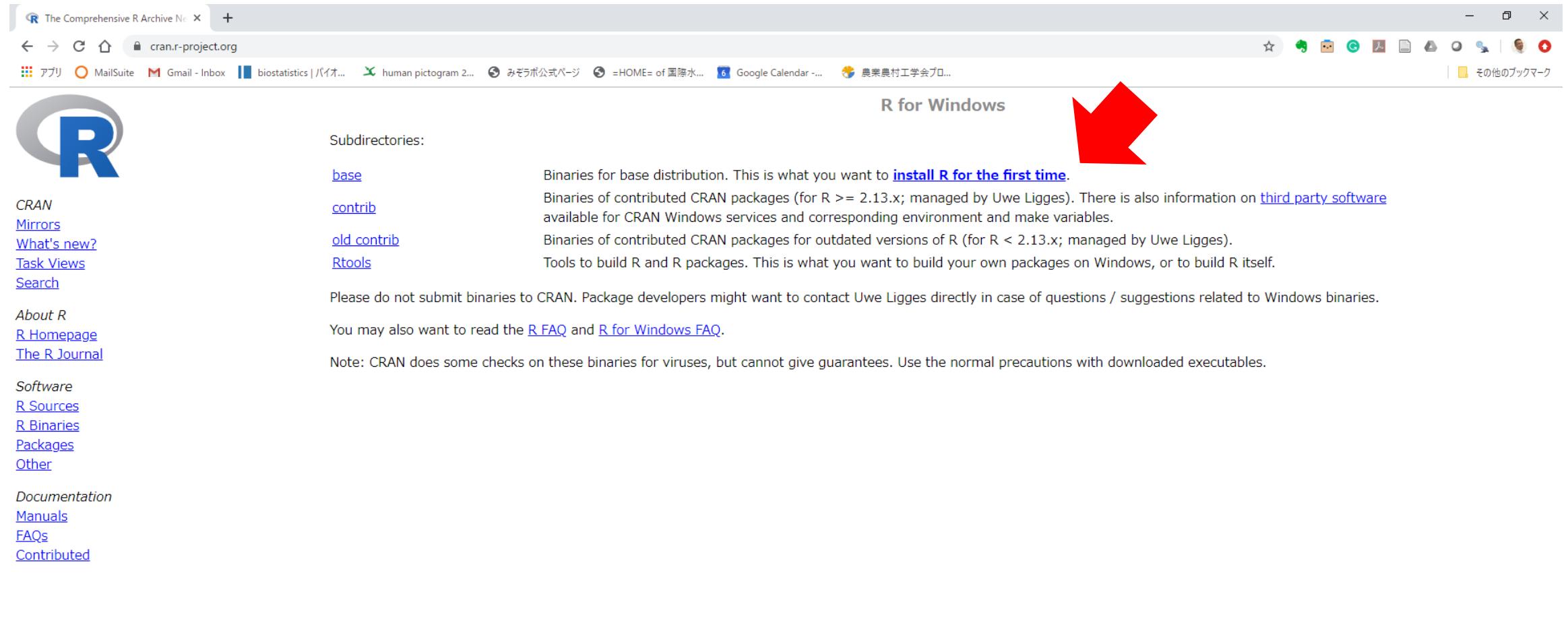
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2019-07-05, Action of the Toes) [R-3.6.1.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](#)

# Rのインストール



The Comprehensive R Archive Network

cran.r-project.org

Subdirectories:

- [base](#) Binaries for base distribution. This is what you want to [install R for the first time](#).
- [contrib](#) 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.
- [old\\_contrib](#) Binaries of contributed CRAN packages for outdated versions of R (for R < 2.13.x; managed by Uwe Ligges).
- [Rtools](#) Tools to build R and R packages. This is what you want to build your own packages on Windows, or to build R itself.

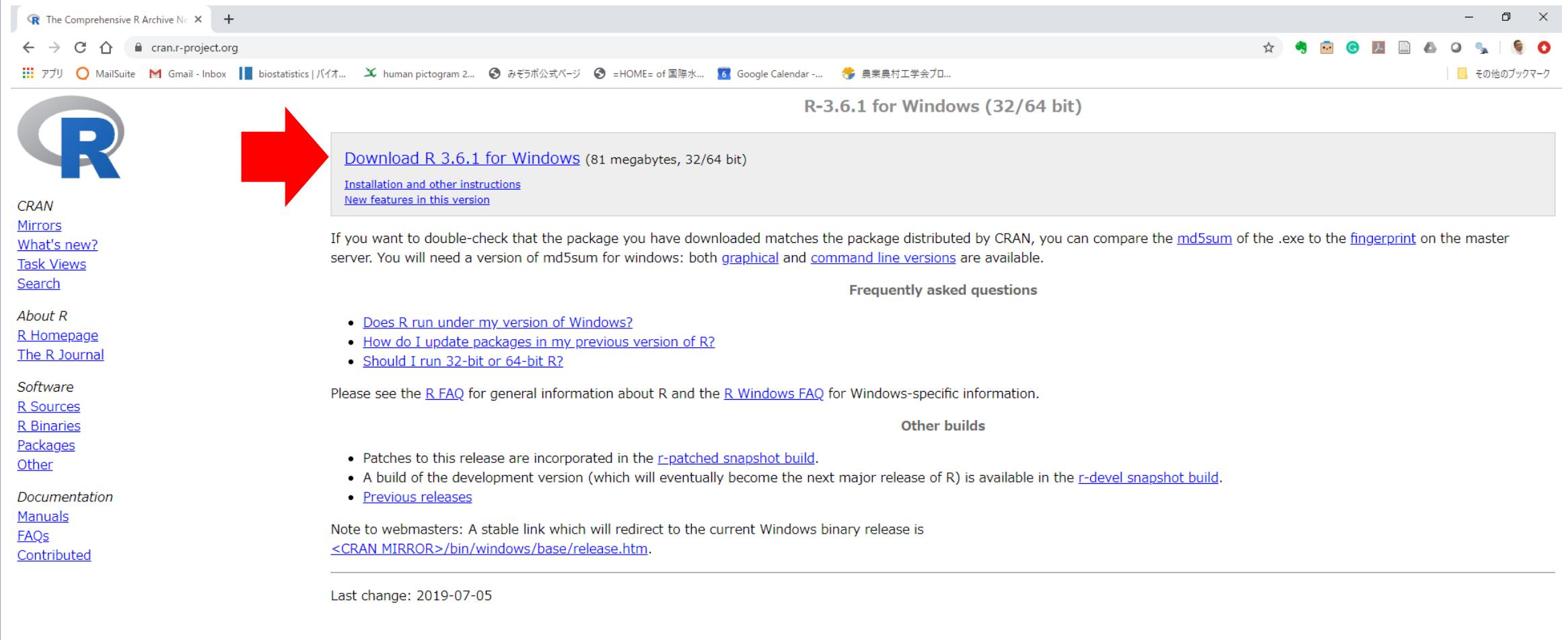
Please do not submit binaries to CRAN. Package developers might want to contact Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

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.

**R for Windows**

# Rのインストール



The Comprehensive R Archive Network

cran.r-project.org

R-3.6.1 for Windows (32/64 bit)

[Download R 3.6.1 for Windows](#) (81 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.htm>.

Last change: 2019-07-05

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のインストール

- ・インストーラーをダウンロード完了
- ・インストーラーを起動して、基本は「はい」や「次へ」、「完了」（インストール中に利用する言語→「日本語」、インストール先はCドライブ内のProgramFiles下に(たぶんデフォルト)、コンポーネントの選択→「利用者向けインストール」、起動時オプション→「いいえ（デフォルトのまま）」）
- ・Windowsの場合、自分が使っているものが、32bitか64bitかだけ確認して、その選択は間違わないようにしてください（「設定」の「システム」で確認可能）。Macの場合は、自分が使っているものがAppleシリコン搭載（M1以降）かどうかでダウンロードするファイルが違うので注意（リンクマーク→「このマックについて」から確認）。

# Rstudioのインストール

Google Search results for "r studio". The search bar is highlighted with a red box. A large red arrow points to the first result, which is the official RStudio website.

Results from rstudio.com

**RStudio - RStudio**  
https://rstudio.com ▾  
Open source and enterprise-ready professional software for data science.  
You visited this page on 11/11/19.

Download RStudio  
RStudio is a set of integrated tools designed to help you be more ...

RStudio IDE  
RStudio is an integrated development environment (IDE) ...

R Packages  
The RStudio team contributes code to many R packages and ...

RStudio Community  
A community for all things R and RStudio.

Books  
In addition to the software we've written at RStudio, we've also ...

RStudio Team  
RStudio's recommended professional data science ...

People also ask

What does R Studio do?

Evernote sidebar:

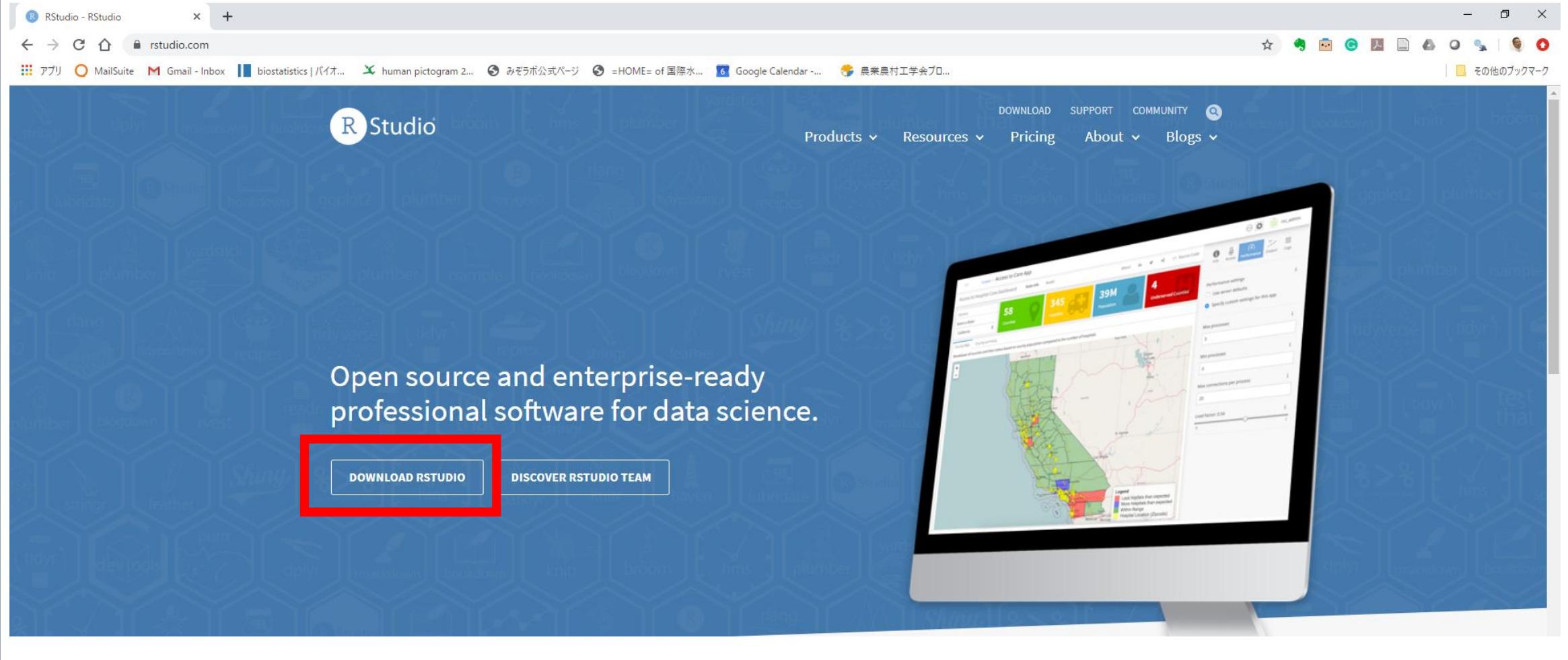
- fillab (2018年1月25日)
- Portal:Complex Systems Digital (2019年1月4日)
- Project Fletcher: Machine Learning (2018年4月12日)

RStudio interface thumbnail:

R Studio

RStudio  
Computer program

# Rstudioのインストール



R Download RStudio - RStudio × + 45

rstudio.com/products/rstudio/download/?utm\_source=downloadrstudio&utm\_medium=Site&utm\_campaign=home-hero-cta

アリ MailSuite Gmail - Inbox biostatistics | バイオ... human pictogram 2... みぞラボ公式ページ =HOME= of 国際水... Google Calendar 農業農村工学会プロ...

その他ブックマーク

# R Studio

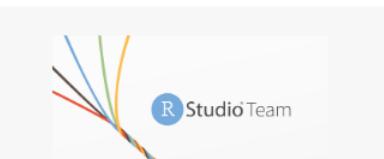
Products Resources Pricing About Blogs

## Download RStudio

### Choose Your Version

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

[LEARN MORE ABOUT RSTUDIO FEATURES](#)



R Studio Team

RStudio's new solution for every professional data science team. RStudio Team includes RStudio Server Pro, RStudio Connect and RStudio Package Manager.

[LEARN MORE](#)

RStudio Desktop	RStudio Desktop	RStudio Server	RStudio Server Pro
Open Source License	Commercial License	Open Source License	Commercial License
<b>Free</b>	<b>\$995 /year</b>	<b>Free</b>	<b>\$4,975 /year</b>
<a href="#">DOWNLOAD</a>	<a href="#">BUY</a>	<a href="#">DOWNLOAD</a>	<a href="#">BUY</a>
<a href="#">Learn more</a>	<a href="#">Learn more</a>	<a href="#">Learn more</a>	<a href="#">Evaluation   Learn more</a>

A large red arrow points to the "DOWNLOAD" button for RStudio Desktop (Open Source License).

ここに入力して検索 ○ ⌘ F Windows Mail Excel PowerPoint Chrome Android Music OneDrive

11:42 2019/11/12

## RStudio Desktop 1.2.5019 - Release Notes

RStudio requires R 3.0.1+. If you don't already have R, download it [here](#).

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

RStudio 1.2 requires a 64-bit operating system, and works exclusively with the 64 bit version of R. If you are on a 32 bit system or need the 32 bit version of R, you can use an [older version of RStudio](#).

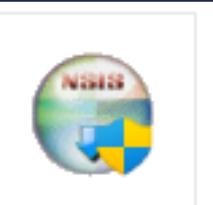
### Installers for Supported Platforms

#### Installers

Installers	Size	Date
RStudio 1.2.5019 - Ubuntu 18/Debian 10 (64-bit)	106.04 MB	2019-11-01
RStudio 1.2.5019 - Debian 9 (64-bit)	106.39 MB	2019-11-01
RStudio 1.2.5019 - Fedora 28/Red Hat 8 (64-bit)	120.89 MB	2019-11-01
RStudio 1.2.5019 - macOS 10.12+ (64-bit)	126.88 MB	2019-11-01
RStudio 1.2.5019 - SLES/OpenSUSE 12 (64-bit)	99.04 MB	2019-11-01
RStudio 1.2.5019 - OpenSUSE 15 (64-bit)	107.09 MB	2019-11-01
RStudio 1.2.5019 - Fedora 19/Red Hat 7 (64-bit)	120.26 MB	2019-11-01
RStudio 1.2.5019 - Ubuntu 14/Debian 8 (64-bit)	96.93 MB	2019-11-01
RStudio 1.2.5019 - Windows 10/8/7 (64-bit)	149.82 MB	2019-11-01
RStudio 1.2.5019 - Ubuntu 16 (64-bit)	104.91 MB	2019-11-01

### Zip/Tarballs

- インストーラーをダウンロード
- インストーラーを起動して、基本は「はい」や「次へ」、「完了」
- 自分が使っているパソコンが、32bitか64bitかだけ確認して、その選択は間違わないようにしてください



RStudio-2024  
.12.1-563.exe

# Rstudioを起動する、その前に、、、

- Rstudio起動の前に、Rによる解析に関するファイルなどを格納するためのフォルダを作りましょう
- フォルダを作る場所と名前は任意ですが、フォルダ名とパスに日本語を含まないように注意してください。
- フォルダ名については、ここでは「RGWD」として進めます

パスというのは、フォルダの場所を示した住所のようなもので、Windowsの場合はフォルダを右クリックして「プロパティ」を開くことで確認することができます。Macの場合はコマンド+クリックして「情報を見る」で確認することが可能です。

# とにかく起動をしましょう

The screenshot shows the RStudio interface. On the left, the script editor contains R code for performing a changepoint analysis on a mobile data dataset. The code includes loading packages, reading a CSV file, visualizing the data, and running the analysis. The output pane shows the results of the analysis, including the number of segments, locations of changes, and summary statistics. On the right, the environment pane lists objects in memory, and a plot pane displays a time series plot with red lines indicating detected changepoints.

```

install.packages("changepoint")
library(changepoint)
# change in variance
x <- read.csv("mobile_data_0822_1111.csv", header = TRUE)
View(x)
x <- x$mobile
ansvar=cpt.var(x)
plot(ansvar)
print(ansvar)
# change in mean
ansmean=cpt.mean(x)
plot(ansmean,cpt.col='blue')
print(ansmean)
# change in mean and variance

```

Created on : Thu Apr 25 20:03:25 2019

summary(.) :

Created Using changepoint version 2.2.2

Changepoint type : Change in mean and variance

Method of analysis : AMOC

Test Statistic : Normal

Type of penalty : MBIC with value, 13.22016

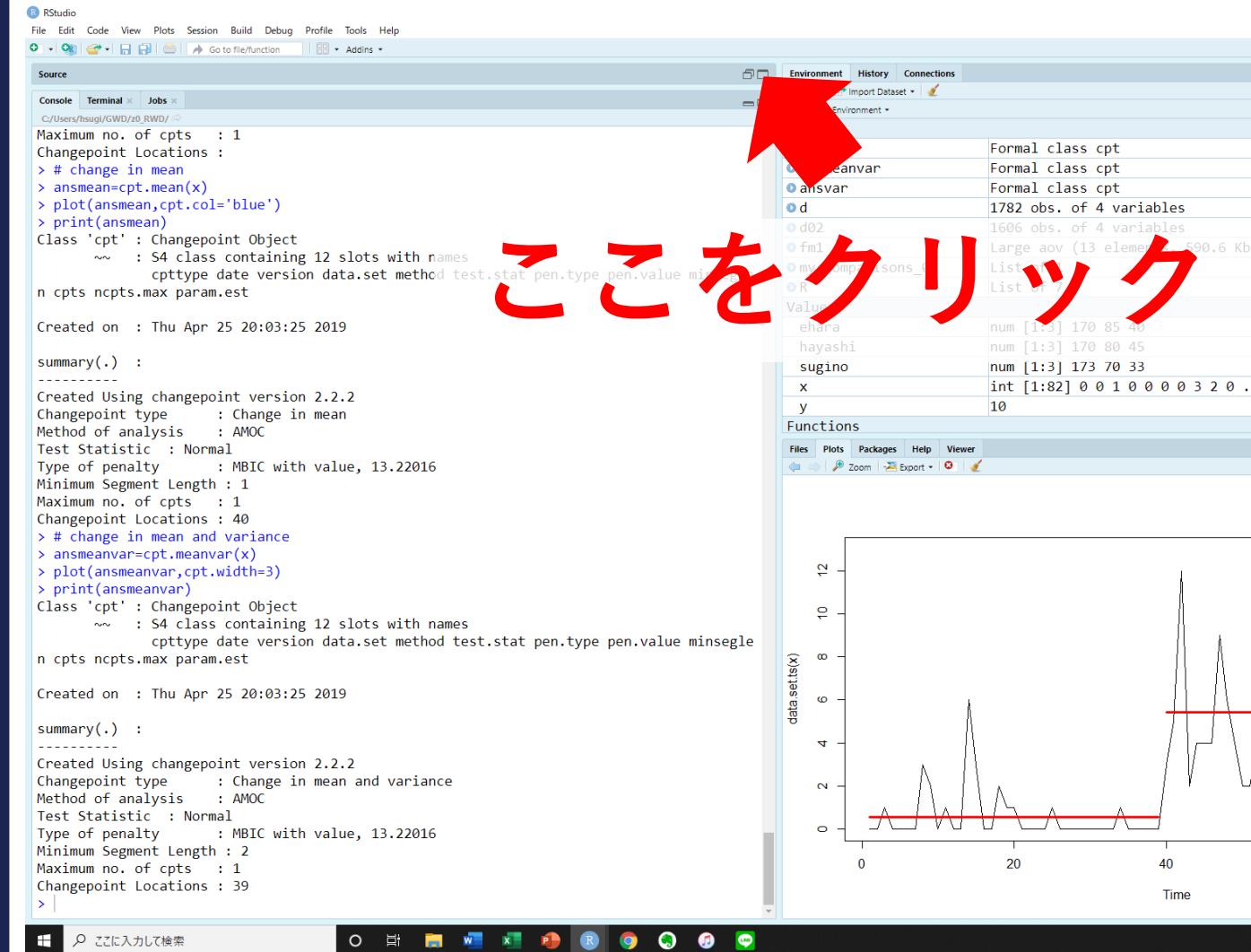
Minimum Segment Length : 2

Maximum no. of cpts : 1

Changepoint Locations : 39

- Rstudio のアイコンを(ダブル)クリックして起動
- Rを起動する必要はありません
- 後に、プロジェクトを作成した場合には、「\* \* \*.Rproj」というプロジェクトファイルから起動することも可能です。

# もし、こういう3分割だったら



The screenshot shows the RStudio interface. In the top-left, the console window displays R code and its output. A large red arrow points to the top-right corner of the Environment pane, where a message reads: "警告: 環境が壊れています。エディタを開いて修正してください。" (Warning: The environment is corrupt. Please open the editor and fix it.) Below this, the Environment pane lists various objects like `ansmeanvar`, `ansvar`, and `d`. On the right, a plot titled "data sets(x)" shows a jagged line graph with a red horizontal line at y=0.

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Source Environment History Connections
C:/Users/sugii/GWD/x0.RWD/ ~
Maximum no. of cpts : 1
Changepoint Locations :
> # change in mean
> ansmean=cpt.mean(x)
> plot(ansmean,cpt.col='blue')
> print(ansmean)
Class 'cpt' : Changepoint Object
  ~~ : S4 class containing 12 slots with names
    cptype date version data.set method test.stat pen.type pen.value minseglen
n cpts ncpts.max param.est
Created on : Thu Apr 25 20:03:25 2019
summary(..) :
-----
Created Using changepoint version 2.2.2
Changepoint type : Change in mean
Method of analysis : AMOC
Test Statistic : Normal
Type of penalty : MBIC with value, 13.22016
Minimum Segment Length : 1
Maximum no. of cpts : 1
Changepoint Locations : 40
> # change in mean and variance
> ansmeanvar=cpt.meanvar(x)
> plot(ansmeanvar,cpt.width=3)
> print(ansmeanvar)
Class 'cpt' : Changepoint Object
  ~~ : S4 class containing 12 slots with names
    cptype date version data.set method test.stat pen.type pen.value minseglen
n cpts ncpts.max param.est
Created on : Thu Apr 25 20:03:25 2019
summary(..) :
-----
Created Using changepoint version 2.2.2
Changepoint type : Change in mean and variance
Method of analysis : AMOC
Test Statistic : Normal
Type of penalty : MBIC with value, 13.22016
Minimum Segment Length : 2
Maximum no. of cpts : 1
Changepoint Locations : 39
>

```

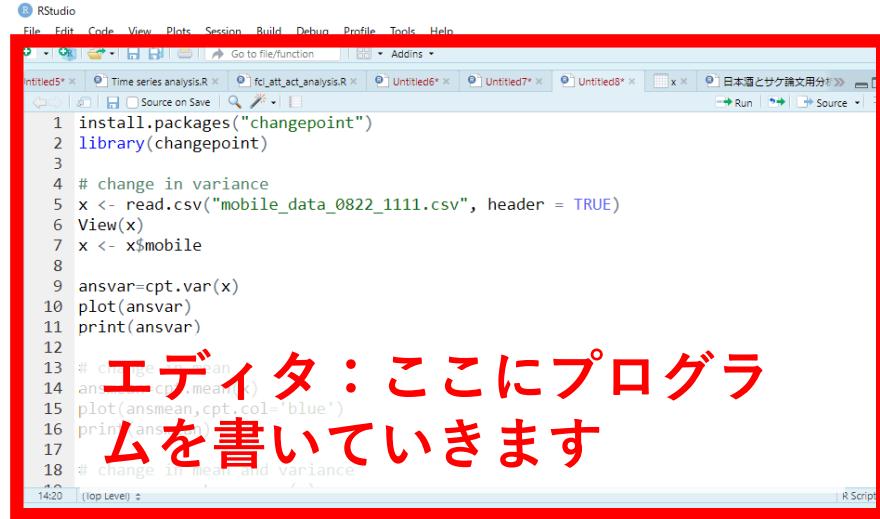
ここをクリック

- 初めてRstudioをインストールした場合に、こういった三分割(エディタがない)の状態になっていることが多いです。落ち着いてコンソールの右上の紙が重なっているようなアイコンをクリックして、エディタを出しましょう。

# とにかく起動をしましよう

## 各ペイン(枠)paneの説明

エディタ  
ペイン



```

install.packages("changepoint")
library(changepoint)

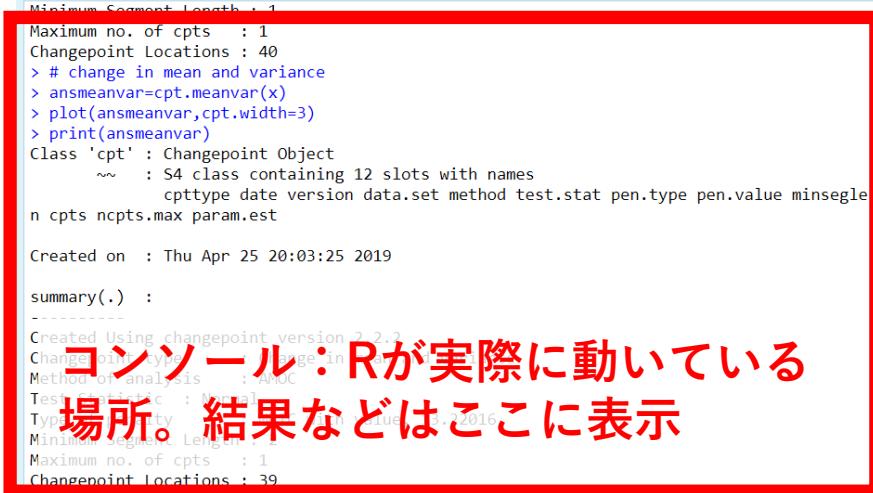
# change in variance
x <- read.csv("mobile_data_0822_1111.csv", header = TRUE)
View(x)
x <- x$mobile
ansvar=cpt.var(x)
plot(ansvar)
print(ansvar)

# change in mean and variance
ansmean=cpt.meanvar(x)
plot(ansmean,cpt.col='blue')
print(ansmean)

```

**エディタ：ここにプログラムを書いていきます**

コンソール  
ペイン



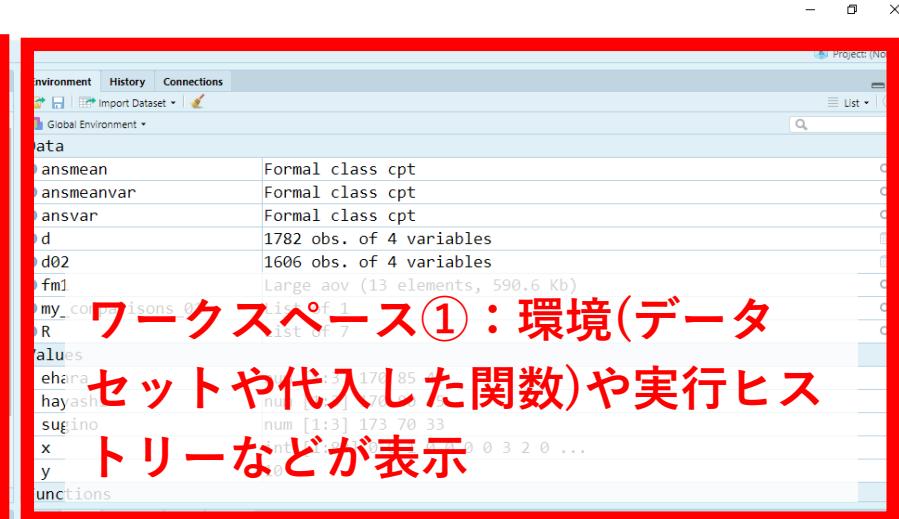
```

Maximum no. of cpts : 1
Changepoint Locations : 40
> # change in mean and variance
> ansmeanvar=cpt.meanvar(x)
> plot(ansmeanvar,cpt.width=3)
> print(ansmeanvar)
Class 'cpt' : Changepoint Object
  ~~~ : S4 class containing 12 slots with names
    cpttype date version data.set method test.stat pen.type pen.value minseglen
n cpts ncpts.max param.est
Created on : Thu Apr 25 20:03:25 2019
summary(.) :
...
Created Using changepoint version 2.2.2
Changepoint type: meanvar
Method of analysis : AIC
Test statistic : tstat
Type of penalty: bdp
Minimum segment length: 1
Maximum no. of cpts : 1
Changepoint Locations : 39

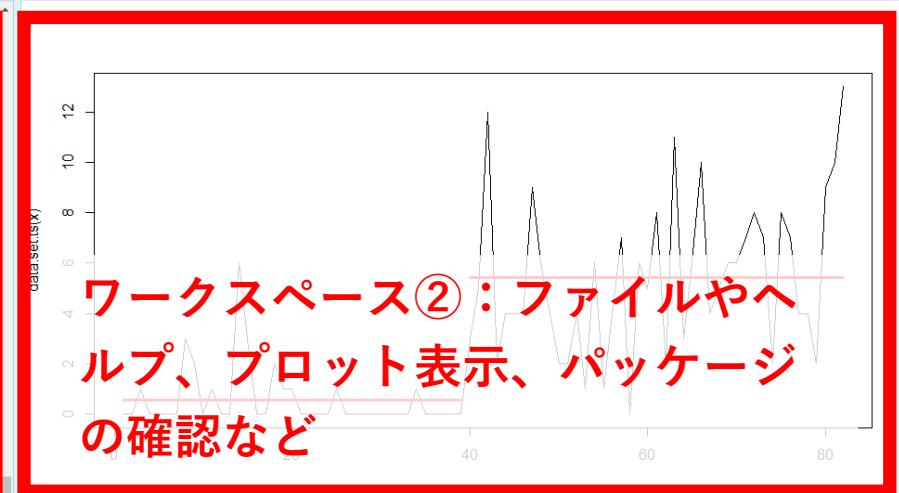
```

**コンソール：Rが実際に動いている場所。結果などはここに表示**

ワークスペース  
ペイン その1



ワークスペース  
ペイン その2



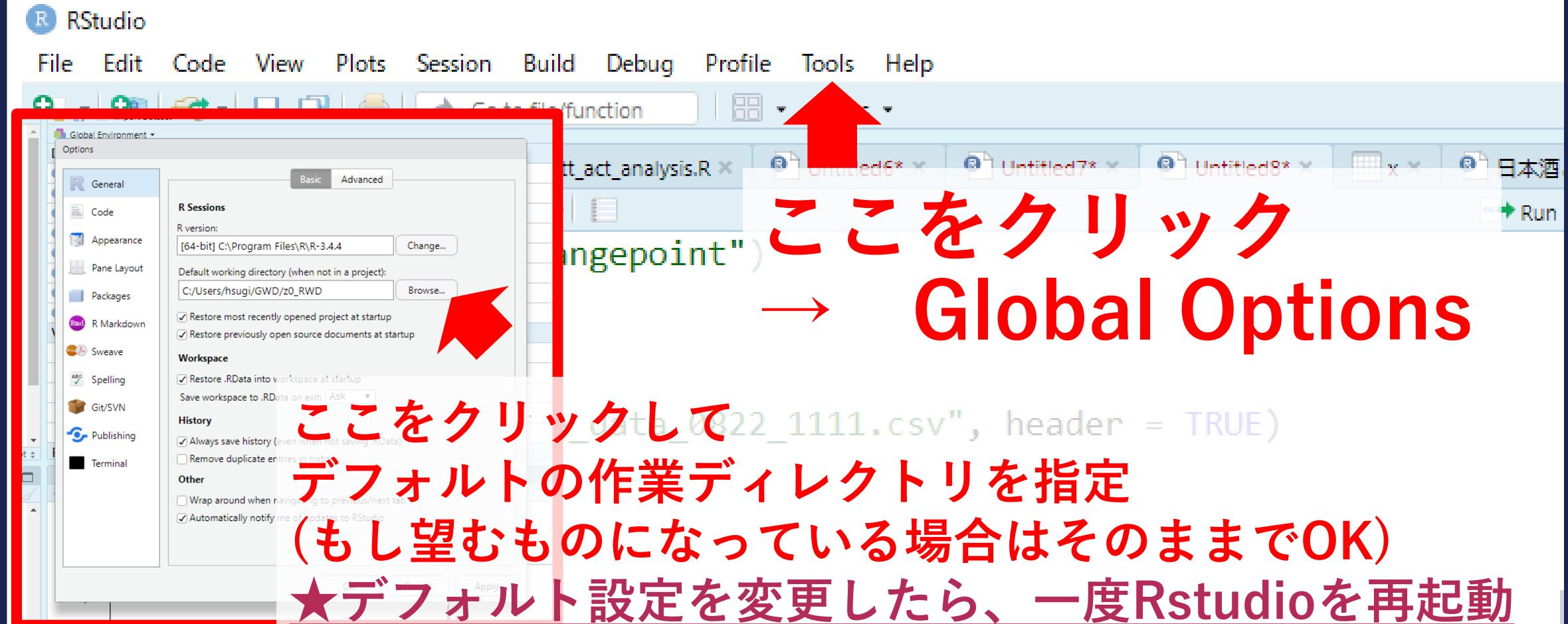
# 各種初期設定

- ・以降のスライドで、下の項目のやり方を説明していきます
  - 1) 作業ディレクトリの確認と設定
  - 2) 新しいプロジェクトの作成
  - 3) Global Optionsの設定
    - ・デフォルトの作業ディレクトリの設定
    - ・Code → Display
    - ・Appearance(フォントやテーマ)
  - 4) 新しいスクリプトの作成

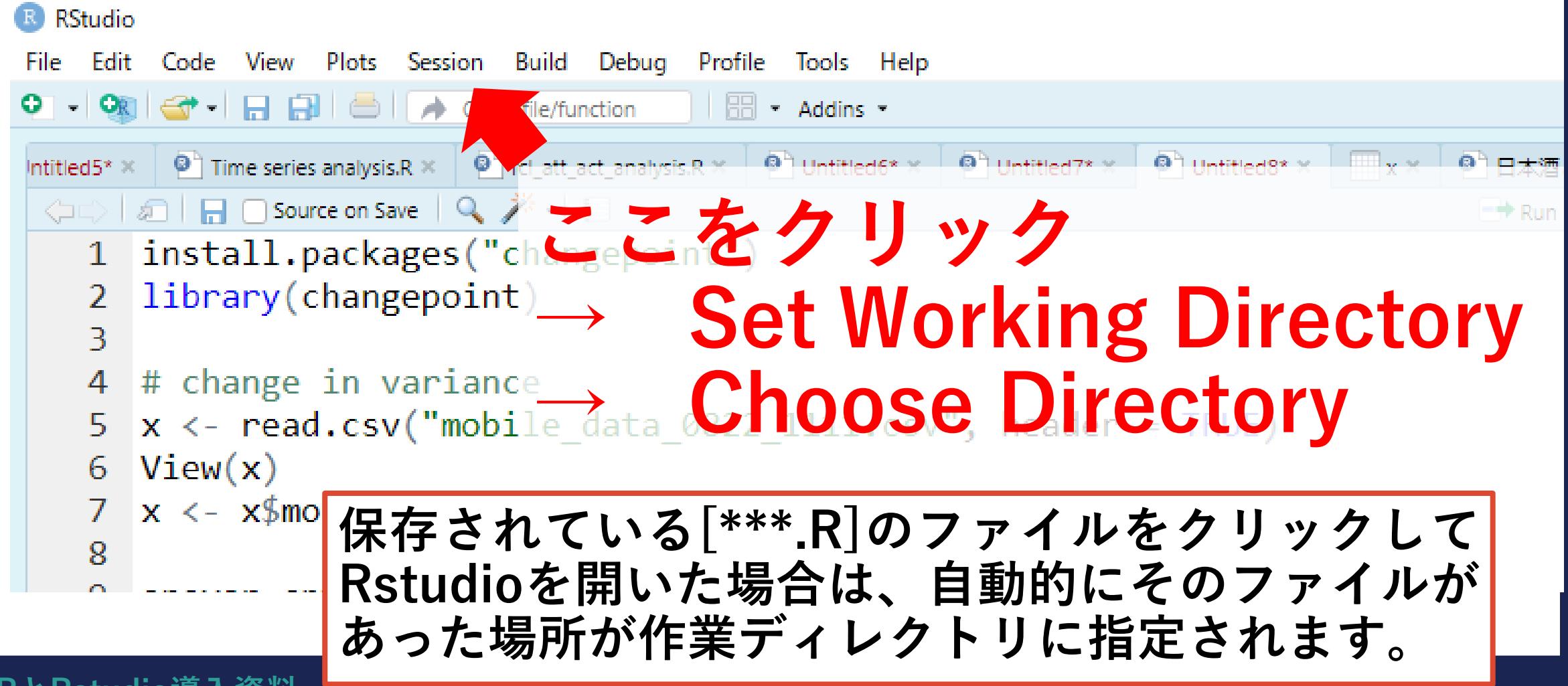
# 作業ディレクトリのデフォルトを設定する

- 先に任意のところにRで作業を行うためのフォルダを作成して頂きましたが、これを今後「おおもとの作業ディレクトリ(General Working Directory)」と呼んでいきます。
- ディレクトリとは、データが格納されている場所を示す“住所”ですが、フォルダ(の名前)とだいたい読み替えることができます。
  - \* ディレクトリにデータを保存 ≒ フォルダにデータを保存
  - \* ディレクトリを指定 ≒ フォルダを指定
- 以降では、「RGWD」という名前で作って頂いたフォルダを“おおもとの作業ディレクトリ”として使用する前提で話を進めますが、基本的に作業ディレクトリは任意の名前・場所で結構です。ここに各種データを保存していきます

# 作業ディレクトリのデフォルトを設定する



# 作業ディレクトリを設定/変更したい場合



RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

Addins

Untitled5\* Time series analysis.R Untitled6\* Untitled7\* Untitled8\* 日本酒

Source on Save Run

```

1 install.packages("changepoint")
2 library(changepoint)
3
4 # change in variance →
5 x <- read.csv("mobile_data_0221_15sec.csv", header=TRUE)
6 View(x)
7 x <- x$mo
8

```

**ここをクリック**

**Set Working Directory**

**Choose Directory**

保存されている`[**.R]`のファイルをクリックして  
Rstudioを開いた場合は、自動的にそのファイルが  
あった場所が作業ディレクトリに指定されます。

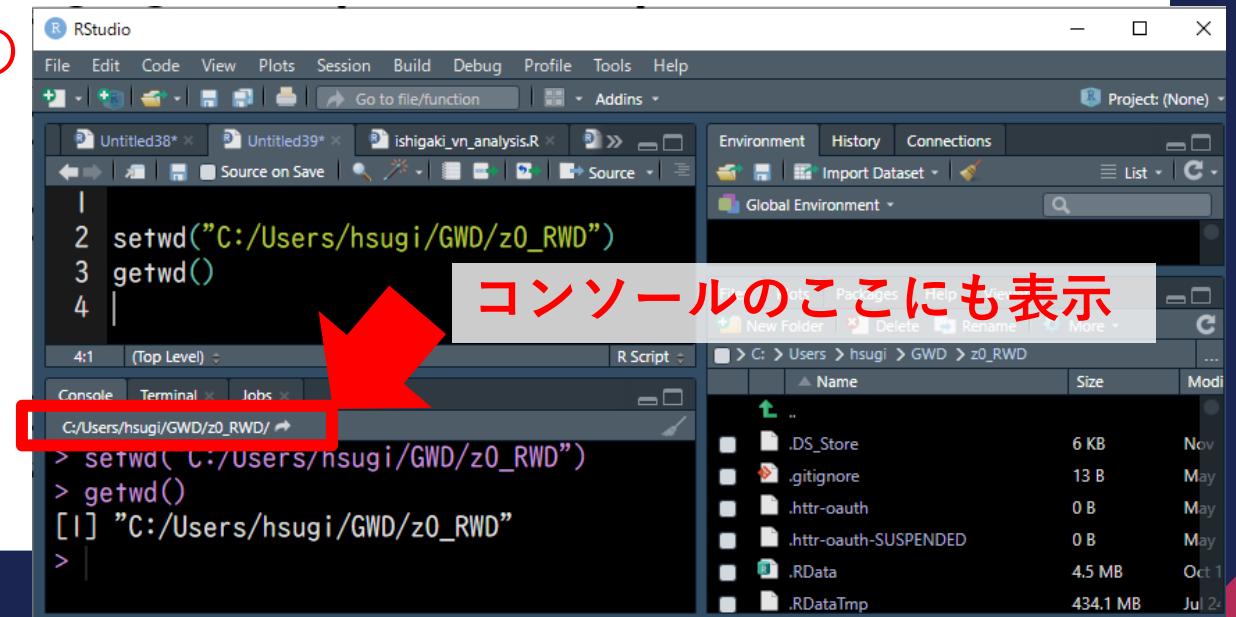
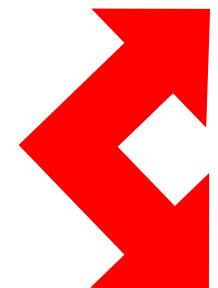
# 作業ディレクトリを設定する/変更する②

現在の作業ディレクトリの確認と変更は、スクリプト上(エディタで書いたり、直接コンソールで書いたり)で可能

- ・確認 >>> getwd()を実行 \*カッコの中は何もなくてOK
- ・変更 >>> setwd("\*\*\*\*\*")を実行

\*カッコ内は作業ディレクトリの  
パスを""で囲ったもの

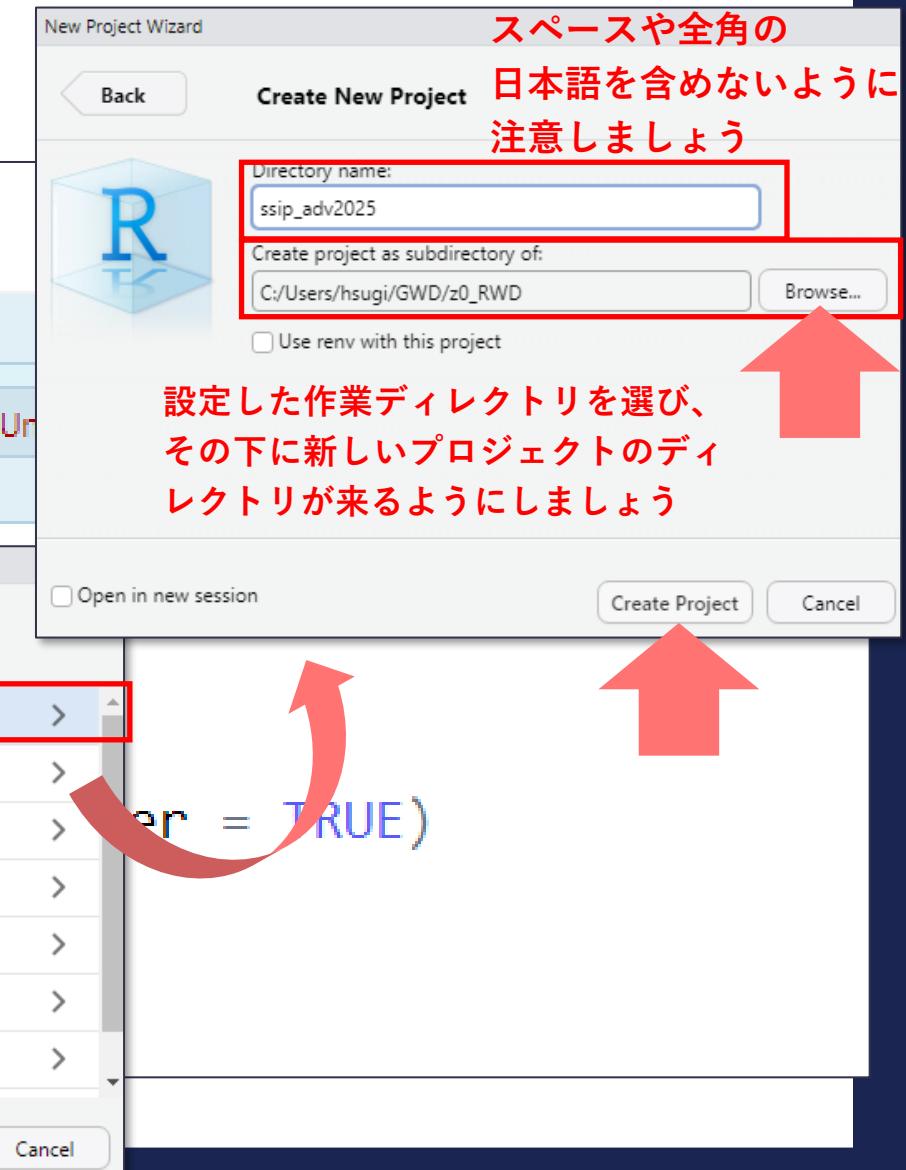
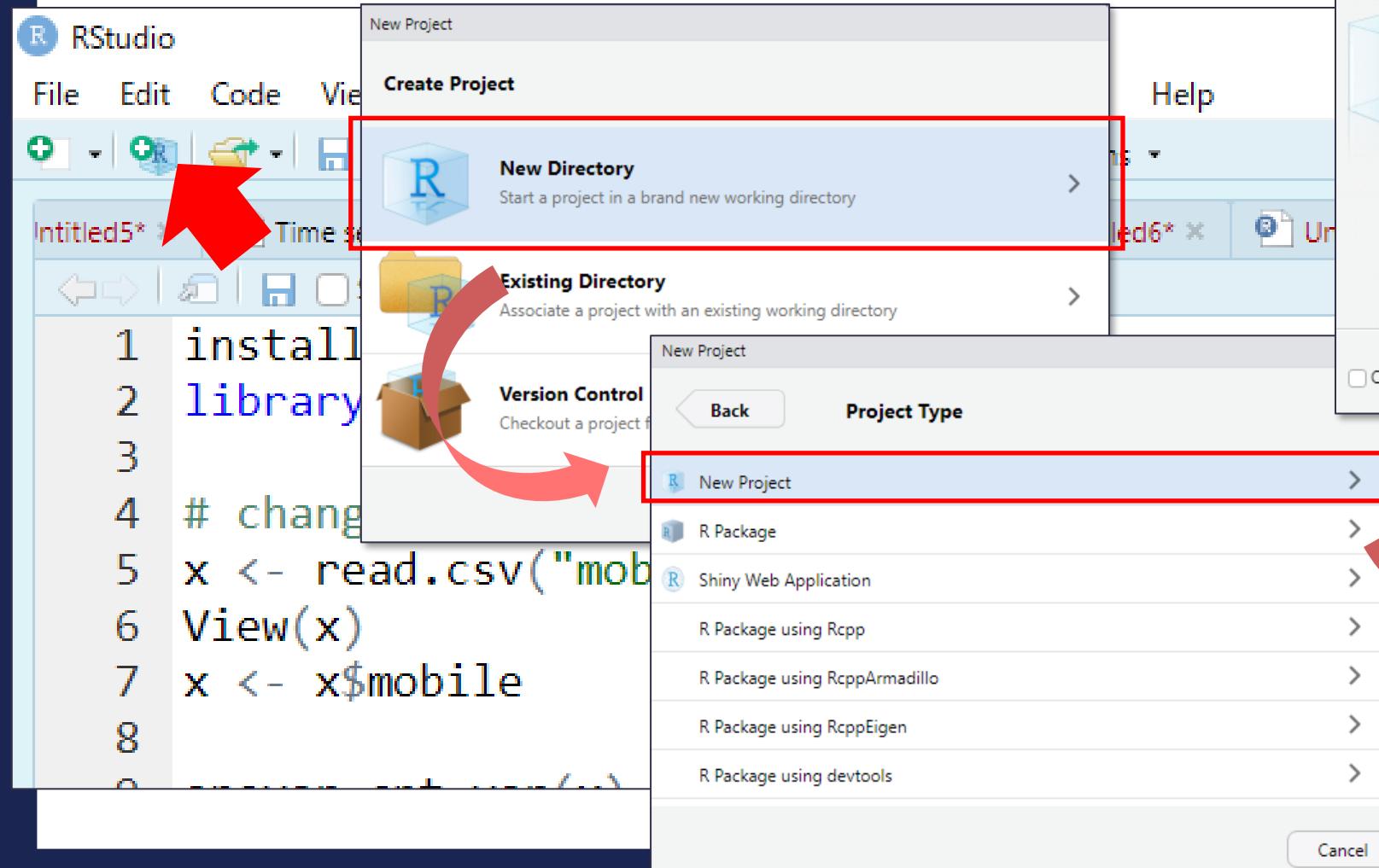
エディタで打って  
Ctrl(Command)+Enterか、  
コンソールに直打ちか



# プロジェクト機能について

- Rstudioに搭載されている便利機能の一つであるプロジェクト機能は最大限活用して、プロジェクト単位で作業を管理していきましょう。
- デフォルトの作業ディレクトリを、先に作成した“おおもとの作業ディレクトリ”に設定できている場合に、下記で説明するプロジェクトの新規作成を行うと、“おおもとの作業ディレクトリ”の中に、プロジェクト単位でフォルダ＝各プロジェクトの作業ディレクトリが作成され、後々にファイル管理などが煩雑にならずに便利です。

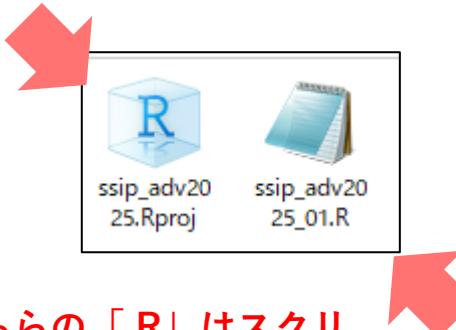
# プロジェクトの新規作成



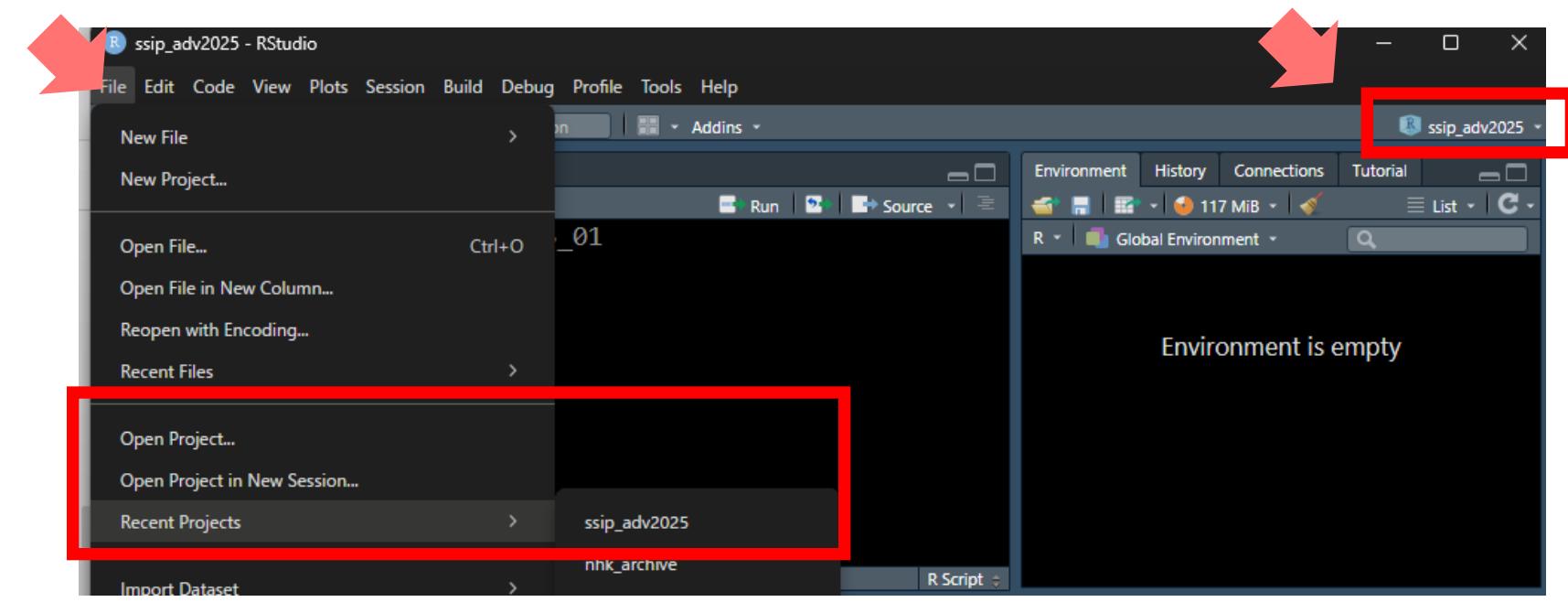
# プロジェクトの切り替えや開き方

- Rstudio起動中にプロジェクトを切り替えたい場合、「Files」→「Open Project」もしくは「Recent Projects」から選択
- もしくは画面右端のアイコンからも選択可能

一度閉じたプロジェクトは作業ディレクトリ中の「.Rproj」から開くことが可能

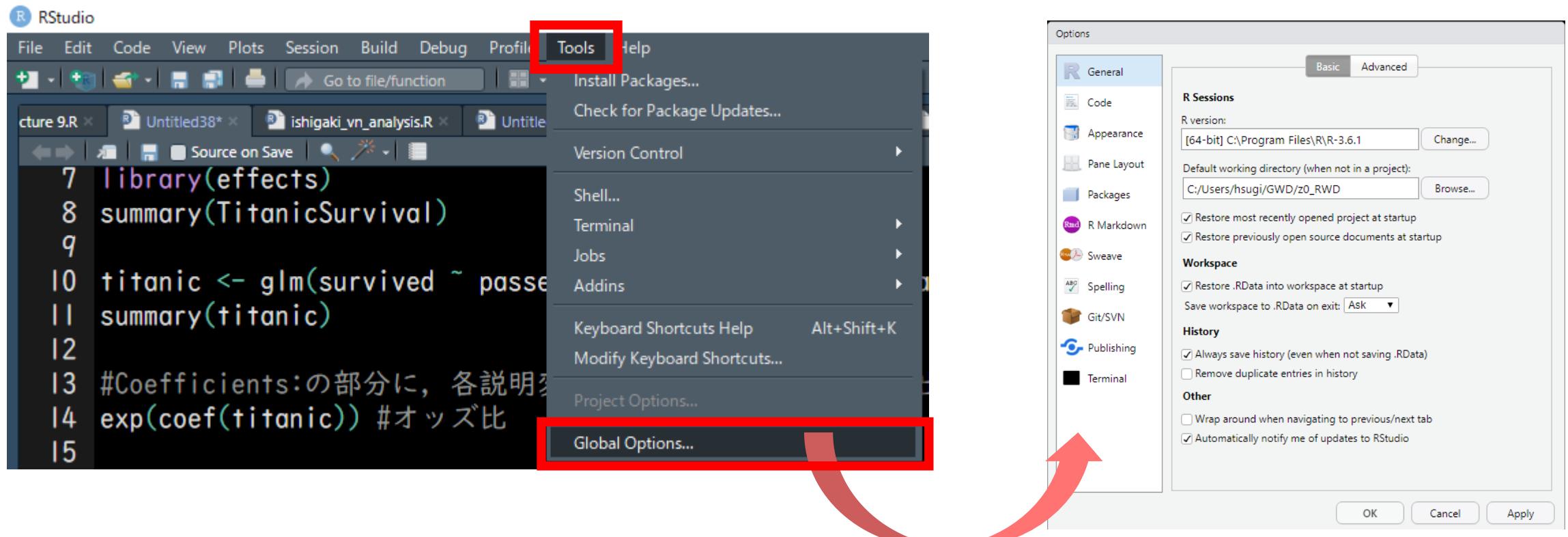


こちらの「.R」はスクリプトのみを開く(確認する)ことができる



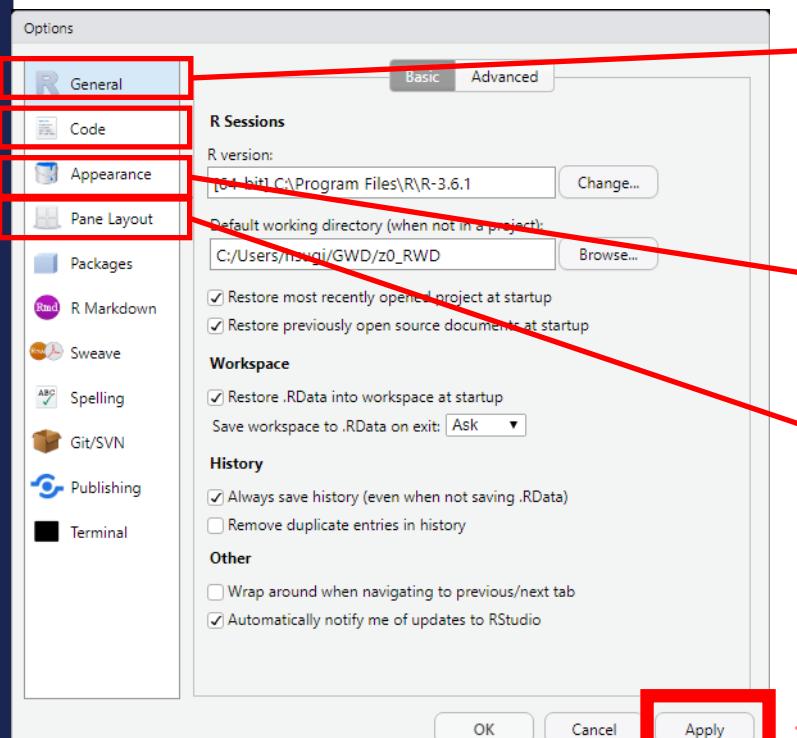
# 補：環境設定をカスタマイズ

- メニューの「Tools」から「Global Options」をクリックすると、環境設定画面を開くことができます。



# 補：環境設定をカスタマイズ

- よく変更する設定を紹介すると、、、

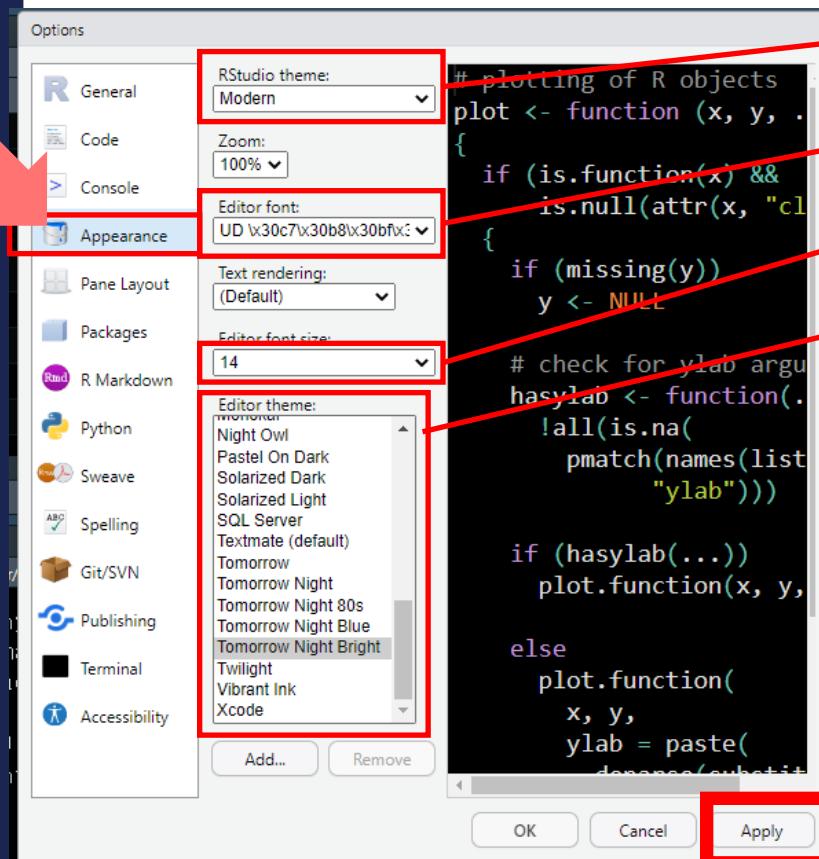


- General : Rのバージョンや作業ディレクトリのデフォルト、ヒストリーやワークスペースの保存についての設定が可能
- Code : エディタ上のコードの見え方を調整。基本は初期設定でOK。「Show whitespace characters」や「Rainbow parentheses」などを試してみても良いかも（好みの問題）
- Appearance : Rstudioのテーマ、フォントとサイズ、エディターのテーマを設定が可能
- Pane Layout : ペイン(枠)のレイアウトやペイン内のタブの設定が可能

設定を変更した後に、「Apply」を押して適用させることを忘れないようにしてください

# 補：環境設定をカスタマイズ

- 「Appearance」の設定について



- Rstudio全体のテーマの設定・変更
- エディターのフォント設定・変更
- エディターのフォントサイズの設定・変更
- エディターのテーマの設定・変更

もしエディターのテーマを外部から取り込んでみたい人は、下記のリンクから、いろいろな外部テーマを探すことができます！

Githubサイト：<https://github.com/max-alletsee/rstudio-themes>

日本語で見たい場合

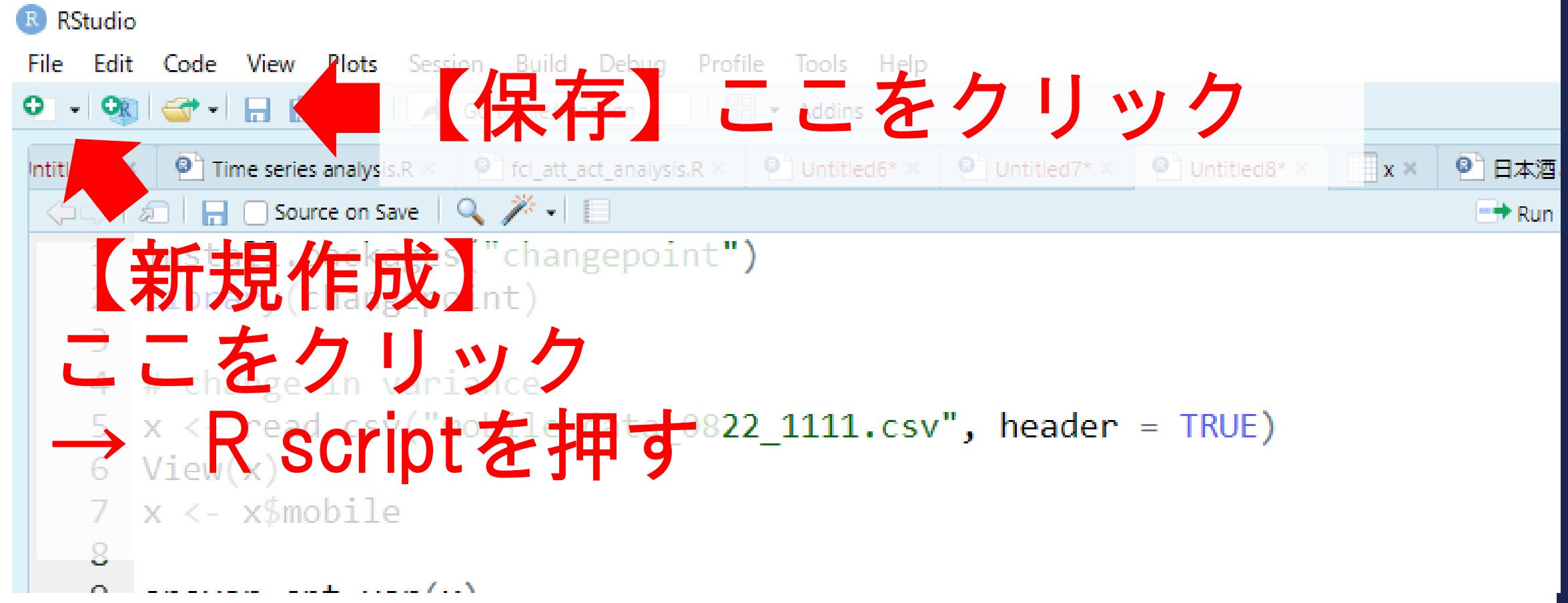
<https://www.wenyanet.com/opensource/ja/5fed8af32d68a502b8478097.html>

```
#例えば「EvangelionUnit01」のテーマを導入したい場合は、下記をコピーして、エディタにペーストして実行
rstudioapi::addTheme("https://raw.githubusercontent.com/takemal-studio/RStudio_EvangelionUnit01_theme/master/EvangelionUnit01.rstheme", apply = TRUE)
#---- もしダウンロードのみを行う場合は、apply=FALSEに
#手動変更は：Tools - Global Options - Appearance - Editor theme
```

# スクリプトの作成と保存について

- Rstudioの中で、実際のプログラムの実行はコンソールペインで行われる
  - コンソールに記述したコードは残らないため、エディタペインにてスクリプトとして記述し、保存しておくことを推奨
  - ただし、一時的に実行したいコードについてはコンソールに直接記述して試したりするのはOK
- 
- 【新規作成】メニューの「File」→「New File」（Ctrl/Command + Shift + N）または次のスライドで示すアイコンをクリックすることで新規作成
  - 【保存】メニューの「File」→「Save」（Ctrl/Command + S）または次のスライドで示すアイコンのクリックで保存

# スクリプトの作成と保存用アイコンの位置



# コマンドやプログラムの実行について

- ・スクリプトをコンソール、またはエディタ(メモ帳)に書きこんで、コマンドを実行することで解析や図表の描画を行う
- ・コマンドやプログラムの実行について、基本はコンソールではEnter、エディタではCtrl(Command) + Enter、もしくはエディタ右上の「Run」

★1行を実行する場合は、その行のどこにカーソルがあっても、  
Ctrl(Command)+Enterで実行が可能。複数の行のプログラムを一気に実行  
したい場合には、Ctrl(Command)+Shift+カーソル(→や←)もしくはマウス  
のドラッグで範囲指定した上で、Ctrl (Command) + Enterもしくはエディタ  
右上の「Run」

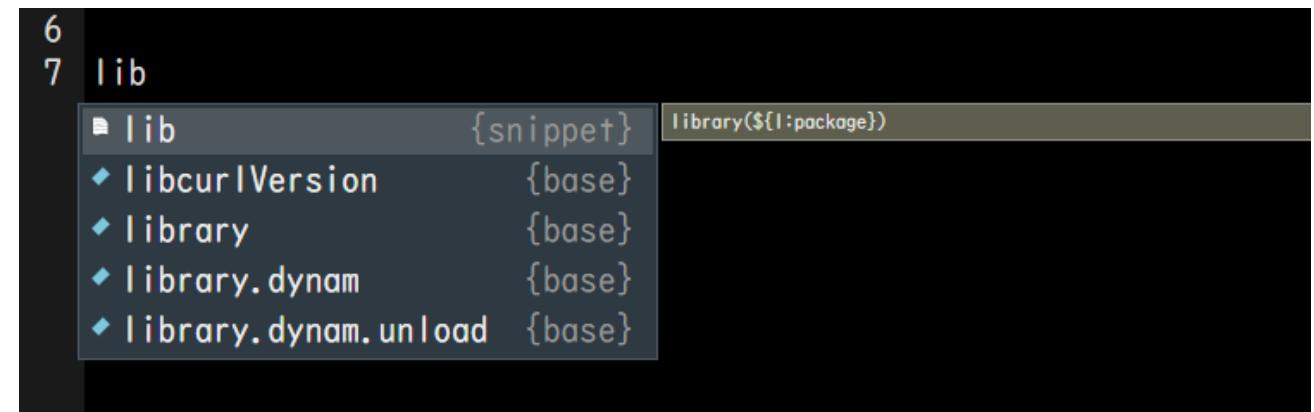
# Rにおけるプログラムを書く際の注意点

- ・残す必要のあるものは、エディタ・ペインで書いて残すのが安全
- ・大文字、小文字は区別されます
- ・できるだけ英語での打ち込みが安全
- ・エディタで書いたものを保存したい場合には、エディタの左上にあるフロッピーディスクのアイコンを押して保存。これまで保存してなかったスクリプトに関してはファイル名を指定する。文字コードの指定を求められたら、適宜指定してから保存。
- ・最新版が保存されている場合は、スクリプトの色が「黒」になっている。スクリプト名が「赤」になっている場合は最新の保存状態からどこかが変更されている。
- ・Ctrl + Sのショートカットでも保存可能。

# Rstudioにおける補完とsnippet(スニペット)の活用

- Rそのものではなく、Rstudioを使う利点は、この補完機能が大きいところを占めます。関数名を入力している途中で、候補が表示されます。
- 例えば、library()を書きたい時に、「lib」まで打つと、こんな感じで候補が表示されます。

候補が表示されたら、  
上下キーで選択し、Tab  
キーもしくはEnterキー  
で補完してもらいま  
しょう



# Rstudioにおける補完とsnippet(スニペット)の活用

- また、候補にスニペットが含まれる場合は、それも表示してくれます。スニペットとは本来「断片」という意味で、プログラミング言語においては、定型的なコード、もしくはそれを簡略なコマンドで呼び出せるようにした機能のことを指します。例えば、繰り返し作業をさせる関数である「for(){}」の入れ子構造や変数入力の位置を、forとエディタで打って {snippet} と書いてあるものを選択すると、まるっと定型が表示されて便利です。

5  
6 for|

6 for {snippet}

for (\${1:variable} in \${2:vector}) {  
 \${0}

6 for (variable in vector) {  
 7  
 8 }

6 vec <-NULL  
7  
8 for (i in 1:5) {  
 9 temp <- i  
 10 vec <- c(vec, temp)  
 11 }

Tabキーで進めていきましょう。  
定型の中で準備された入力箇所に、  
Tabキーで飛ぶことができます。

# 実習課題

- ・自分のパソコンにRとRstudioをインストールする。
- ・Rstudioが問題なく立ち上がり、4分割画面にすることができるようになる。
- ・作業ディレクトリのデフォルトを設定する。
- ・新規のプロジェクトおよびスクリプトを立ち上げることができるようになる。
- ・「Tools」→「Global Options」からテーマやフォントサイズなどを自分好みに変更し、自分の好みに合う環境設定を行ってください。
- ・エディタに「print("Nice try!")」と打ち込み、Ctrl + Enterを押す
  - \* print()のカッコの中はどんな言葉でも構いません。

# R基本操作

# Rの基礎的な使い方：算術演算子演算子と数学的関数)

- よく使う演算子
- よく使う数学的関数：()の中に数値などを入れます

記号	意味
+	足し算
-	引き算
*	掛け算
/	割り算
^	累乗

記号	意味
<code>sqrt()</code>	$\sqrt{\phantom{x}}$ の計算
<code>abs()</code>	絶対値
<code>log()</code>	自然対数
<code>signif(x,a)</code>	$x$ を 有効桁で $a$ 桁まで丸める
<code>trunc()</code>	整数部分を求める
<code>round()</code>	四捨五入

エディタで適当に計算式を入力 & 実行して練習してみましょう。

# Rの基礎的な使い方：論理演算子

- Rで使う主な論理演算子を紹介しておきます。

記号	意味
<もしくは>	通常の不等号
<=もしくは>=	以上、以下
==	イコール
!=	ノットイコール

\* 補足：Rでの論理値ベクトルはTRUE、FALSE、NAの三種  
 「=」は関数内の引数の指定に、「<-」は代入に使用する  
 紛らわしいが、きちんと使い分けをしましょう

# Rの基礎的な使い方：演算子系まとめ練習

#算術演算子の利用

#足し算や引き算

1+1

65-5

#掛け算

32\*3

#割り算

10/2

#演算の優先順位の確認

$(2*4)/3(1+4*5)$  #これはエラーが出る

$(2*4)/3*(1+4*5)$  #カッコに掛ける場合は\*が必要

#論理演算子の利用

#結果に論理値

#TRUE, FALSE, NAを返す

#不等号

$10000000 > 10000001$

#以上や以下

$234 >= 233$

$234 <= 232$

#イコールやノットイコール

$12 == 12$

"Osugi" == "MuraiP"

"Sugino" != "Sugii"

# Rの基礎的な使い方(代入, セミコロン, #)

- Rでは、オブジェクト(イメージは箱や枠)に「<-」を使って代入(格納)し、そのオブジェクトを繰り返し使ったり、後の計算をやりやすくしたりすることができます。\* 「=」も使えますが、関数内の様々な指定(引数(ひきすう))にも使われる所以、代入(格納)には「<-」を使うのがお勧めです。
- 「;」を使うと1行に複数の処理を並列させることができます。
- また、「#」を使うと、それ以降の言葉はコメントとなり、プログラムの実行には関わらなくなります。エディタに以下を打って、練習してみましょう。

```
x <- 20; y <- 15; z <- x+y; w <- z-45 #暗算してみましょう  
(w+x)*z/10+(x*y)
```

# オブジェクトには意味のある名前をつける

- データや統計に用いる変数を格納したオブジェクトを使うと、同じ処理を複数回実行したりする際の手間を省くことが可能
- 処理が複雑になると、どの変数が何の結果や値を持つのかがわかりにくく混乱することも多いため(経験談(笑))、オブジェクトにはわかりやすく意味のある名前をつけることが重要

【悪い例】 aとか、 dとか、 testとか、 fawoeioihjwf

【良い例】 mean\_heightとか、 df\_healthとか、 data\_test01

\*Rでの一般的なコーディングスタイルを学ぶには：  
tidyverse style guide(<https://style.tidyverse.org/index.html>)を参照

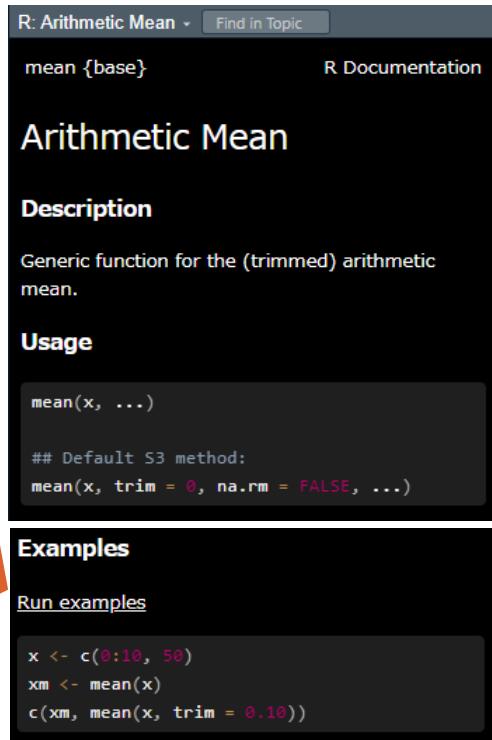
# Rの基礎的な使い方(関数の利用)

- Rには便利な関数がたくさん用意されています。作業ディレクトリの設定の際にも出てきた「getwd()」や「setwd()」もそれにあたります。ここでは基本的に「関数名()」と表記します。
- カッコの中にはそれぞれの関数を実行するために必要な変数や数値を指定します。これらを引数と呼びます。(関数名(引数名 = 値)の形で表記・実行)
- 「?関数」でヘルプが見れますので、使い方に困ったら確認しましょう。

```
getwd()  
x <- c(1, 2, 3, 4, 5, 6) #c()を使うと、複数の値をまとめることができます  
mean(x)  
?mean()
```

# Rの基礎的な使い方(引数について)

- 「?関数」でヘルプを見ると、関数を実行するときに必要となる「引数(関数に渡す値)」に何があるのか、またそれをどのように指定するのかが書かれています。新しい関数を覚える際には、このヘルプであったり、Rdocumentation(<https://www.rdocumentation.org/>)や、関数が入っているパッケージの各Vignetteなどを確認しながら、どのような引数を渡すべきかなどを勉強するようにしましょう。
- また、R4.2.0からはヘルプページ下部にある「Run examples」でExampleの実行結果を確認できるようになっています。



R: Arithmetic Mean Find in Topic R Documentation

**Arithmetic Mean**

**Description**

Generic function for the (trimmed) arithmetic mean.

**Usage**

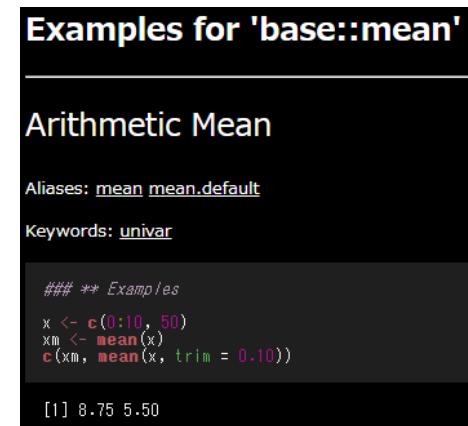
```
mean(x, ...)

## Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)
```

**Examples**

[Run examples](#)

```
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))
```



**Examples for 'base::mean'**

---

**Arithmetic Mean**

**Aliases:** `mean` `mean.default`

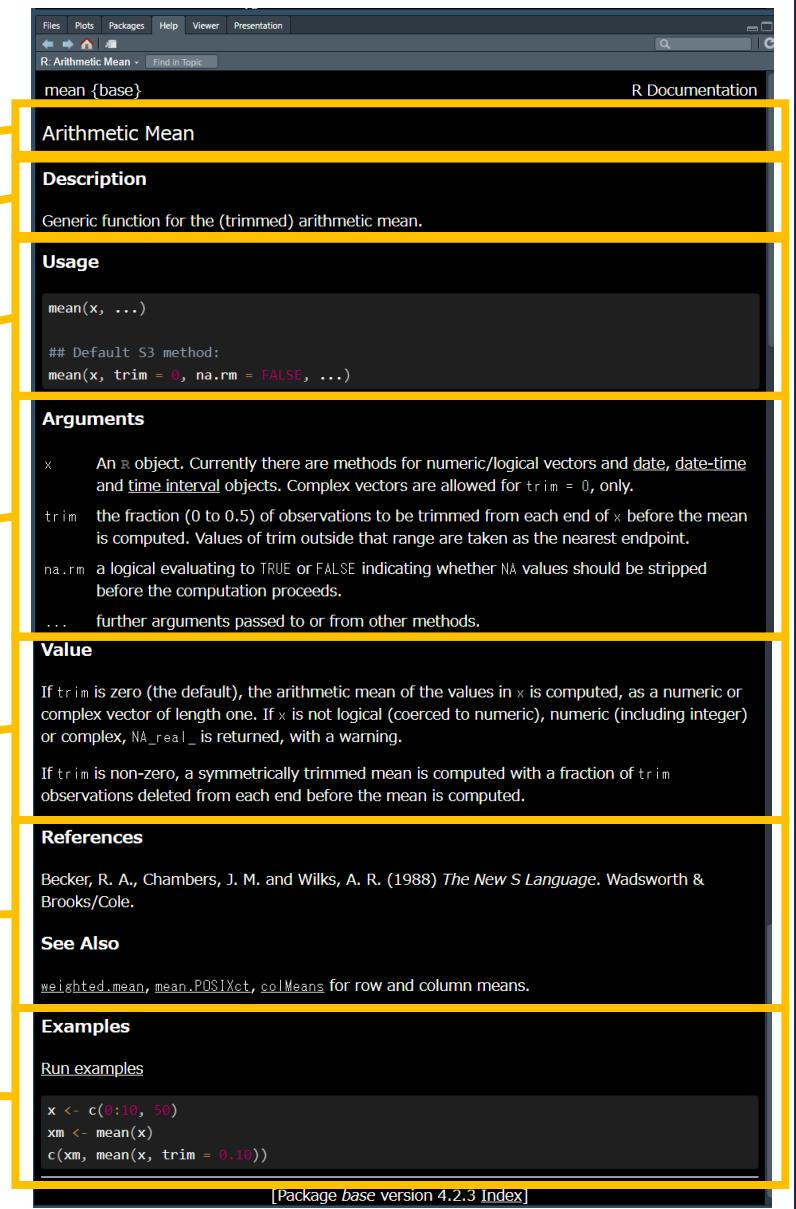
**Keywords:** `univar`

```
### ** Examples
x <- c(0:10, 50)
xm <- mean(x)
c(xm, mean(x, trim = 0.10))

[1] 8.75 5.50
```

# ヘルプドキュメントの読み方

- 関数名
- 関数の説明
- 利用方法
- 引数の説明
- 関数の挙動詳細や説明
- 参考文献など
- 関数を使った処理例



# Rで扱うデータ（数字と変数）の型

- R上で扱うデータ（数字と変数）の型を覚えてもらいたい
- なぜ？
- 実践上、関数によって受け付ける数字と変数の型が違うので、エラーが出た時に真っ先に疑うべきなのは、データ内の数字や変数の型が、その関数で扱うことができるものになっていない可能性
- `str(データ名)` でチェックできます。データを読み込んだら、`View(データ名)`や`head(データ名, n = ?)`でチェックすると共に数字と変数の型もチェック。

# Rで扱うデータの型

データの型	説明	定数の例
<b>integer</b>	整数型	1, -1, 20300, ...
<b>numeric</b>	実数型（整数も含む）	3.1415, -0.001, 125.00012, 1, 1e-10, ...
<b>complex</b>	複素数型	1i, 1 - 4.5i, 3+0i, complex(re=a,im=b), ...
<b>character</b>	文字型	"A", "2014/10/14", "", ...
<b>logical</b>	論理型	TRUE, FALSE, T, F, NA, ...
<b>factor</b>	要素型	一定の数を持つカテゴリ(male, female等)
<b>NULL</b>	データが空	NULL

# Rで扱うオブジェクトの型

オブジェクトの型	説明	Rで試しに代入してみる
vector	ベクトル(スカラー)	vector <- c(1, 2, 3, 4, 5) もしくは c(1:5)
matrix	行列 (変数名、行名がない)	matrix <- matrix(c(1:8), nrow =2)
dataframe	ベクトルのセットが形成する行列 (変数名、行名がある)	dataframe <- data.frame(attend = c("yes", "no", "yes"), gender = c("male", "male", "female"), height = c(155, 163, 172))
list	異なる種類の複数オブジェクトをまとめたもの	list <- list(name = "お試しリスト", jp_en = data.frame(jp=c("バナナ", "ライト", "写真"), en=c("banana", "light", "picture")), data_matrix = matrix(c(1, 5, 8, 10), nrow = 2), gender_vec = c("m", "f", "f", "m", "f"))

# ベクトル、データフレーム、リストの補足説明

- 複数の値をまとめたもの → ベクトル
- 複数のベクトル（変数）をまとめたもの → データフレーム
  - \* 行列形式でまとめるので、各ベクトルの行（と行数）は共通
  - \* `data.frame(ベクトル, ベクトル, ...)`で作成可能
  - \* データフレーム\$変数名、でベクトル内の変数にアクセス可能
- 複数の異なるオブジェクトをまとめたもの → リスト
  - \* 長さが違うベクトル同士や、ベクトルとデータフレームの混在OK
  - \* リスト\$オブジェクト名、でリスト内のオブジェクトにアクセス可能

# Rで扱うオブジェクトの使用・応用

- `df <- as.data.frame(オブジェクト名)` \*`as.data.frame` とか `as.matrix`とかで、その形式で保存させることができる
- データフレームやリストに入っている要素にアクセスして取り出したいときは、「\$」を使う（例：データフレーム名\$コラム名 → `dataframe$gender` とか `list$name` とか）
- **発展**：Rを使いこなしていく中で、データハンドリング用の「tidyverse」というパッケージを利用する際に、データフレーム型の進化版として「tibble」型を扱うこともある。`tibble`型はデータフレーム型と比較して格納できるオブジェクトの内容が増え、表示も視覚的にわかりやすい形となる。大きな違いとして、`tibble`型には行名が無い。`tidyverse{}`パッケージを使う際には、ぜひ`tibble`型のオブジェクトを使いこなしてみてください。

# Rの基礎的な使い方(ベクトルや行列)

- 複数の値をまとめたもの = ベクトル(統計に用いるデータの基本単位。Rで言うところのベクトルを統計では変数と呼ぶ)も利用できます。
- 行列も作ることができます。一度ベクトルを作った上で、matrix()を使います。引数はmatrix(ベクトル, nrow = \*, ncol = \*)です。
- 以下をエディタに打って、練習してみましょう。

```
x <- c(1, 2, 3, 4, 5, 6); y <- c(2, 4, 6, 8, 10, 12)
z <- x*y
z
matrix(z, nrow = 2, ncol = 3) #matrix(z, 2, 3)と略記も可。左列から順に上から下
matrix(z, nrow = 2, ncol = 3, byrow = T) #byrow = Tで上行から順に左から右
```

# ベクトル、データフレーム、リスト

```
#データフレームの作成練習
gender <- c("m", "f", "m", "f", "f")
name <- c("Sugii", "Nagai", "Sugino", "Yamamoto", "Ueda")
building <- c("fgss", "edu", "edu", "fgss", "edu")
floor <- c(3, 1, 2, 2, 1)

df <- data.frame(gender, name, building, floor)
df

#データフレーム内の変数にアクセスするためには$を使用
df$gender

#リストの作成練習
mylist <- list(
  colors = c("red", "blue", "green"),
  num = c(1, 3, 5, 6, 8, 8, 9, 10),
  data = df
)

mylist
mylist$data #リスト内のオブジェクトにアクセスするためには$を使用
mylist$data$name #こうすると、リスト内のDF内の変数にアクセス可能
```

# オブジェクト内の参照

- ・オブジェクト内の値を参照したい場合は、[ ]や[[ ]]を使用
- ・[[ ]]は名前が付いた要素内の値を直接参照する
- ・[ ]や[[ ]]内には数値を入れると位置の参照する  
例：floor[2] とすると、floorベクトル内の2番目を参照
- ・[ ]や[[ ]]内に文字列を入れると、要素の名前を参照する  
例：mylist[floor] とすると、mylist内のfloorを参照  
mylist\$floor[3]は、mylist内のfloor変数の3番目を参照

# オブジェクト内の参照

```
#オブジェクト内の要素を参照
```

```
building[2] #buildingベクトル内の2番目の要素を参照
```

```
#名前付きの数値ベクトルを作成
```

```
fruits <- c(banana = 230, grape = 560, strawberry = 450)
```

```
fruits
```

```
length(fruits) #ベクトル内の要素数を確認
```

```
names(fruits) #ベクトル内の要素に与えられた名前を確認
```

```
fruits[2] #要素の位置を指定して参照
```

```
fruits["banana"] #要素の名前を指定して参照
```

```
fruits[[3]] #要素の値だけ返したい場合
```

```
fruits[3] #前行との参照結果を比較
```

```
#リスト内のオブジェクト内の要素を参照
```

```
mylist$data
```

```
mylist["data"] #前行との参照結果を比較
```

```
mylist$data$name[3]
```

# パイプ演算子の活用

- パイプ演算子：左辺の値を右辺の第一引数に渡す処理
  - x |> f() \*これはf(x)と同じ処理
- Rのバージョン4.1.0から、以前はdplyrという特別なパッケージを導入した上で利用されていた「パイプ演算子」がデフォルトで利用可能となりました。
- dplyrパッケージで使われていたパイプ演算子 → %>%
- 現在デフォルトで使用できるパイプ演算子 → |>

# パイプ演算子の活用

- 通常の連續した処理工程では、各行に処理を行い、オブジェクトに格納して、を繰り返す。部品を作つて箱詰めして、その部品を使って製品を作つて箱詰めして、を繰り返すイメージ。
  - > **メリット：中間オブジェクトを残して結果を参照可能**
  - > **デメリット：不要な中間オブジェクトを発生させる可能性**
- パイプ演算子を使用すると、各工程で処理されたものを次の行程に流していく。製造ラインが整備された工場のイメージ。
  - > **メリット：処理の流れがわかりやすく、コード可読性も高い**
  - > **デメリット：中間オブジェクトが残らない(最終アウトプットのみ参照可能)**

# パイプ演算子の活用

```
#パイプ演算子の活用
```

```
mylist$data$gender |> table() #genderの度数分布表を作成
```

```
#パイプ演算子を利用する際、改行と字下げを行うことで可読性が高まる
```

```
rnorm(5000, mean = 0, sd = 1) |>  
  matrix(ncol = 2) |>  
  hist()
```

```
#上の処理をパイプ演算子無しで行おうとすると、、、
```

```
rnorm_vec <- rnorm(5000, mean = 0, sd = 1)  
rnorm_mat <- matrix(rnorm_vec, ncol = 2)  
hist(rnorm_mat)
```

# さらにRの扱いに慣れる

更に練習してみましょう。

```
#ある5人の国語と数学の成績について、下記の計算を練習  
kokugo <- c(70, 53, 64, 82, 48) #国語の成績の変数  
sugaku <- c(51, 49, 86, 88, 71) #数学の成績の変数  
  
#各個人の平均点  
heikin <- (kokugo + sugaku) / 2  
  
#全体の平均点  
mean(heikin)
```

```
#ある3人の身長・体重・年齢のマトリクスおよびデータフレームを作成してみましょう。
```

```
hayashi <- c(170, 60, 35)
```

```
sugino <- c(173, 70, 23)
```

```
ehara <- c(175, 85, 42)
```

```
hayashi[1:3]
```

```
data_hwa1 <- cbind(hayashi, sugino, ehara) #ベクトルを列方向にくっつける *行方向にくっつけるrbindもある
```

```
data_hwa2 <- t(data_hwa1) #行列を転置（行と列を入れ替える）
```

```
colnames(data_hwa2) <- c("height", "weight", "age") #列名を入れこむ
```

```
data_hwa3 <- as.data.frame(data_hwa2) #マトリクスをデータフレームに変換
```

```
data_hwa3 #分析に利用できるデータフレームの完成
```

```
str(data_hwa2)
```

```
str(data_hwa3) #3変数、3観測値のデータフレームであることを確認
```

```
data_hwa2[1,3] #マトリクスの場合、マトリクス[行,列]で任意の値にアクセスできる
```

```
data_hwa3[1,3] #データフレームでも可能
```

```
data_hwa3$height #データフレームの場合は、変数を$で呼び出すことが可能
```

```
mean(data_hwa3$height) #身長の平均
```

```
mean(data_hwa3$weight) #体重の平均
```

# さらにRの扱いに慣れる

## ちょっとしたテクニック

```
#オブジェクトの確認やリセット(プロジェクトを新しく始める時などに有効)  
objects() #現在のオブジェクト一覧を確認する  
rm(list = ls(all=TRUE)) #現在のオブジェクトをすべてリセットする  
cat("¥014") #コンソールにある文字を全て消去  
if(dev.cur() > 1) dev.off() # 「Plots」 にある図を全て消去
```

# パッケージの利用 = 無限の可能性

- Rでは、解析や図表の描画が可能な関数のパッケージ(複数の関数やデータセットの集まり)を必要に応じて追加できるのが強みです。パッケージを利用することで、Rで出来ることは無限に拡張されていきます。日々新しいパッケージが生まれていますので、たくさんのパッケージに触れ、それらを組み合わせて自分なりの分析や可視化ができるようになります。
- 現在20,000を超えるオフィシャルパッケージがCRANに登録されています。
- CRAN以外にも、下記のサイト等でパッケージが配布されています：
  - GitHub(<https://github.com/>)
  - Bioconductor(<https://www.bioconductor.org/>)
  - R-Forge(<https://r-forge.r-project.org/>)
  - R-universe(<https://r-universe.dev/search>)

# パッケージのインストールとライブラリー

- Rでは、基礎的な解析を行うことが可能なプリセットのプログラムが準備されていますが、解析や図表の描画が可能な関数のパッケージを必要に応じて追加できるのが強みです。
- パッケージのインストールを行う場合は、`install.packages()`を使って、引数に必要なパッケージ名を“\* \* \* \*”で指定してインストール。
- `dependencies = T`を引数に指定することで、付属で必要となる全ての他のパッケージもインストールしてくれる

```
install.packages("psych", dependencies = T)
```

# パッケージのインストールとライブラリー

- ・パッケージのインストールは一度行えば、パッケージやRそのものをアンインストールしたりしない限り、その後は実行しなくても良いが、パッケージのロードについては以下のように、RやRstudioを起動する毎に行う必要性がある。
- ・install.packages()はアマゾンでの書籍購入、library()は本を開くイメージ

```
library(psych) #インストールしたpsychパッケージをロード  
require(psych) #こちらでも可能
```

# パッケージのインストールとライブラリー

The screenshot shows the RStudio interface. On the left, the code editor displays R code for calculating averages and performing arithmetic operations. The Environment pane shows variables like hayashi, mean, x, y, wpost, wpre, z, year\_from, year\_to, yearfrom, yearto, and year. A red box highlights the Environment pane. On the right, the Packages pane is open, titled 'Functions'. It has tabs for Files, Plots, Packages, Help, and Viewer, with 'Packages' selected. An arrow points to the 'Update' tab. Below the tabs, a table lists packages in the 'User Library': assertthat, backports, BH, and cli. Each entry includes a checkbox, the package name, a brief description, and its version.

Name	Description
assertthat	Easy Pre and Post Assertions
backports	Reimplementations of Backwards Compatible Functions
BH	Boost C++ Header File Reader
cli	Helpers for Developing Command Line Interfaces

右下のウインドウにある Packages タブをクリックすると、インストールされたパッケージが確認できる。パッケージ名左のチェックボックスをクリックすることでロードする事が可能。またタブ下の「Update」でパッケージの更新を、「Install」で新しいパッケージのインストールをインターフェースで行うことができる。

# パッケージに会いに行く

- 新しいパッケージに出会いたい時、下記のウェブサイトを参照してみてください
- CRAN Task Views (<https://cran.r-project.org/web/views/>):  
分野ごとに関連するパッケージをリスト化したものを  
例：Agriculture、GraphicalModels、MissingDataなど
  - CRANberries (<https://dirk.eddelbuettel.com/cranberries/>):  
CRANに登録されたパッケージの情報発信サイト

# 実習課題

- 一度Rstudioを閉じて、もう一度Rstudioを立ち上げましょう
- リセット系のファンクションを試してみましょう
- 四則演算、ベクトル、行列を練習しましょう
- {psych}パッケージをインストール＆ライブラリーし、ワークスベースペインのパッケージタブでチェックボックスを確認しましょう

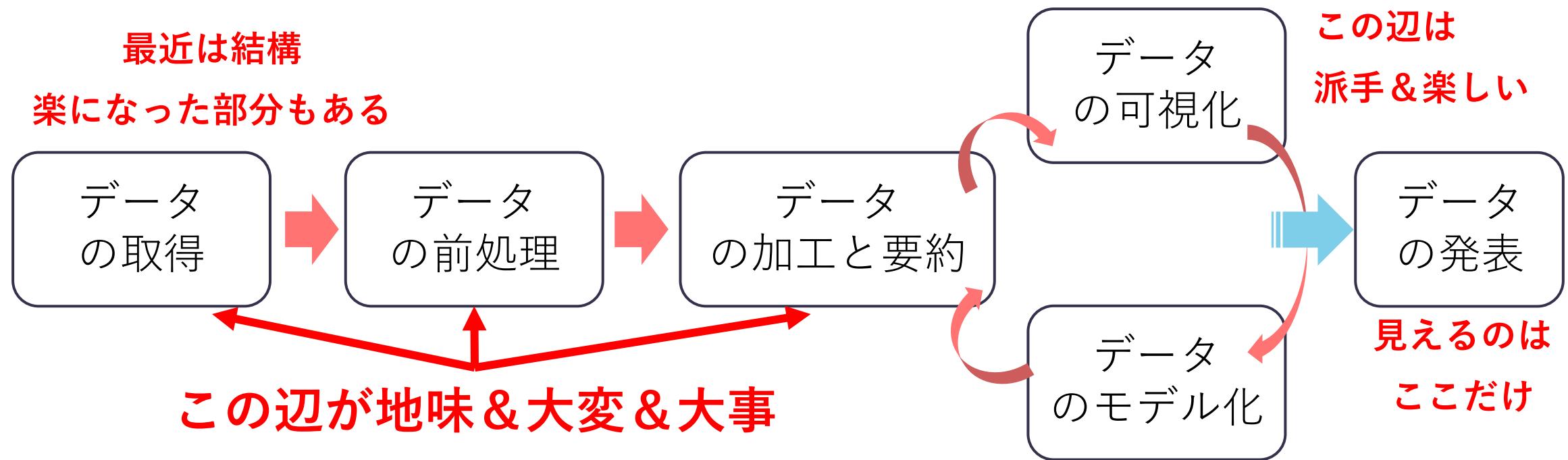
\* このパッケージは後々使うので、確実に行ってください

## データ取り込みとその後の処理

# データ解析の面白いところ辛いところ

データの解析はおおまかに下記のような道筋です。

面白いところもあれば、辛いところもありますが、頑張っていきましょう！



<https://r4ds.had.co.nz/introduction.html> 訳・改

# 解析前の環境設定テンプレ

- まずは前回の復習をしながら、環境設定から、データインポート、記述統計、ヒストグラムなどの作成までの流れを追っていきましょう。
- 「#----」を使うと、スクリプト上のセクションを作成することが可能です。

```
#---- 各種環境設定 -----
```

```
### 環境やプロット、コンソールをクリアしたい場合は以下を実行(任意)
rm(list = ls(all=TRUE))      #グローバル環境(作成されているオブジェクトや変数)をクリア
if(dev.cur() > 1) dev.off()   #プロット欄の図があればクリア
cat("¥014")                  #コンソールのクリア
```

# 解析前の環境設定テンプレ

- 作業ディレクトリを確認し、もし意図したところではない場合は、設定しましょう。作業ディレクトリを設定したら、「list.files()」でディレクトリ内のファイルチェックを忘れずに。

```
### 作業ディレクトリの設定
getwd()          #現在の作業ディレクトリを確認
#setwd("****")  #もし作業ディレクトリのパスが意図したものでなかった場合は、この行の頭の#を消して、意図したパスに変更
myworkingdirectory <- getwd() #作業ディレクトリのパスを後で使うように取っておきます
list.files() #現在の作業ディレクトリ内に存在するファイルのチェック

#ファイルの書き出し時などに使用する情報(ファイルのプレフィックスとスクリプト稼働日)を先に取っておきます
outinfo1 <- "r_practice" #書き出し用ファイルのプレフィックス(接頭辞)の定義 (任意のものを設定)
outinfo2 <- Sys.Date()    #作業日の定義
#outinfo2 <- "2024-12-04" #マニュアルで作業日のデータを取る場合

#再現性の担保のために、乱数のシードを設定しておく
set.seed(123456789)
```

# データの読み込み（0）

- 先のlist.files()で、本資料と同時に配布している「faostat\_practice\_01.csv」というファイルが、作業ディレクトリ内にあることを確認してください。もし持っていない方は、こちらからもダウンロード可能です。開いた先を「名前を付けて保存」→
- [https://raw.githubusercontent.com/teriyakisugi/ssipyu2025/refs/heads/main/data/faostat\\_practice\\_01.csv](https://raw.githubusercontent.com/teriyakisugi/ssipyu2025/refs/heads/main/data/faostat_practice_01.csv)
- このファイルは、FAO（国連食糧農業機関）が出しているFAOSTAT (<https://www.fao.org/faostat/en/>) というデータベースから、1961年から2019年までの、日本、中国およびマレーシアにおける「リンゴ」、「バナナ」、「オレンジ」の収穫量 (kg/ha) (1ヘクタール (ha)あたりの生産量 (収穫量)) を取ってきたものです。
- データ中、id列は通し番号、country列は国、item列は生産物、year列は年、yield\_kg\_ha列は1ヘクタールあたりの生産量を表します。

# データの読み込み（1）

- 「faostat\_practice\_01.csv」というファイル(FAOから取ってきたサンプルのデータ)を読み込み、d01に格納する、という練習です。当該ファイルを作業ディレクトリに入れて下記を実行してみましょう。
- 1行目に変数名などが入っている場合は「header = TRUE」を引数に、1列目に各行の名前(例えばサンプルの通し番号など)が入っている場合は「row.names = 1」を引数に指定する。

```
#オーソドックスなデータの読み込み方法 *ファイルを作業ディレクトリに入れた状態で下記を実行  
d01 <- read.csv("faostat_practice_01.csv", header = TRUE, row.names = 1)
```

```
#ファイルのパスをまずGUIで指定して、読み込む方法  
fpath <- file.choose()  
d01 <- read.csv(fpath)
```

# データの読み込み（2）

- 多目的(「Files」や「Plots」などのタブがある)ペインの「Files」タブをクリックすると、作業ディレクトリ内のファイル一覧が表示されます。
- 任意のファイルをクリック→「Import Dataset」をクリック→ファイル読み込みウインドウが出るので、設定を行った後に、「Import」ボタンを押すと取り込みができます。

The screenshot shows the RStudio interface. On the left, the 'Files' panel displays a list of files in the current working directory. A red box highlights the 'classification\_resall\_gyoson-2022-04-14.csv' file, with a callout pointing to it. Another red box highlights the 'Import Dataset...' option in the context menu for this file. On the right, the 'Import Text Data' dialog is open. It shows the file path 'C:/Users/hsugi/GWD/z0\_RWD/gyoson\_insta/classification\_resall-gyoson-2022-04-14.csv' in the 'File/URL:' field. The 'Data Preview' section shows the first 50 entries of the CSV file. The 'Import Options' section has 'Name:' set to 'classification\_resall\_gyoson' and 'Skip:' set to '0'. Red boxes highlight these fields. Callouts point from the 'Name:' field to the text 'Name'にデータフレーム名を入れましょう。' and from the 'Skip:' field to the text 'Skip'は上から何行を飛ばすかの指定です。'.

「Name」にデータフレーム名を入れましょう。デフォルトはファイル名になっています。

「Skip」は上から何行を飛ばすかの指定です。「1」を入れると、1行目が飛ばされて取り込まれます。

\* ファイルのパス

\* データのプレビュー

Red arrows point from the 'Import Options' section to the 'Import' button at the bottom right of the dialog.

RとRstudio導入資料

# データ内容の確認

- データを読み込んだ後は、読み込まれたデータを確認しましょう。全体の確認、データの先頭と末尾の確認、データ構造とデータの型の確認をしていきます。

```
#データの全体を確認できる(エディタウィンドウに別枠で表示される)
```

```
View(d01) #View()の頭は大文字なことに注意
```

```
#データの確認。「n = ?」を指定して、データの先頭から?行のデータをコンソールに表示
```

```
head(d01, n = 10)
```

```
#データの確認。「n = ?」を指定して、データの末尾から?行のデータをコンソールに表示
```

```
tail(d01, n = 10)
```

```
#データフレームの構造を表示：観測数と変数名一覧、変数の中のデータの型などの一覧を表示
```

```
str(d01)
```

# 分析に進む前のデータ加工

- Rにおいて分析を行う関数には、主に下記2種類のデータのどちらかを引数に取ります。取り込んだ際のデータがどちらであっても、関数に入れる前に適切に加工・変形できるようになっておきましょう。FAOのデータはどっち？

**Tidy Data (整然データ)**

都道府県	時刻	天気
東京	06:00	晴れ
東京	08:00	晴れ
東京	10:00	曇り
大阪	06:00	曇り
大阪	08:00	雨
大阪	10:00	雨

**Value-oriented Data (挟まれた値にフォーカスした行列データ)**

都道府県	06:00	08:00	10:00
東京	晴れ	晴れ	曇り
大阪	曇り	雨	雨

- Tidy Dataの方は、縦1列に1つの変数、横1行に1つの観測、1セルに1つの値。ある意味variable-oriented。機械可読性が高い。
- 右の図はセル内の値にフォーカスされた行列の形。プレゼンテーションや報告書、論文内などの表現に適している。

# 戦略的なデータの整備

- データを整備する際には、Tidy Data(整然データ)の作成を心がけましょう。tidyverse{}パッケージの開発者であるHadley Wickham氏によるデータ分析に対する理想的とされるTidy Dataは以下の特徴を有しているものを指します：

1. 各変数は列をなす
2. 各標本は行をなす
3. 各値はセルである

Wickham(2014), Wickham(2017)

# データの列に対する良い命名法

- コードは他の人が最短時間で理解でき、データの列名については情報量が多く、曖昧性がない形で命名されなければならない (Boswel and Foucher, 2011)
- 変数の在り方による良い命名法は以下の通り：

変数の在り方	良い命名法	説明
ブール変数 (0/1やダミー変数)	is_female, has_kids	何が1なのかを明示して分かりやすく
カテゴリ変数 (ファクター)	prefecture47, country5	カテゴリの数を変数名に入れておく
連続量の bin 化した変数	age_bin5, yield_bin10	bin の数を変数名に入れておく
連続量の変数	yield_kg, height_cm	可能であれば単位も変数名に入れておく

# データのトリミング

- 全体のデータから解析に使う部分や変数を抜き出す方法を紹介します。

# 1 ) 変数行列が連續している場合は、[行,列]で指定して抜き出し

```
d01_subdata1 <- d01[1:177, 1:4] #サンプルデータ中、最初の国データにあたる部分だけ抜き出し
```

# 2 ) 変数名を選択するならsubset関数の利用。selectオプションで引き抜く変数を指定。

```
d01_subdata2 <- subset(d01, select = c("item", "year", "yield_kg_ha"))
```

#complete.cases()で、欠損値がないものだけを抽出することも可能

```
d01_subdata3 <- subset(d01_subdata2, complete.cases(d01_subdata2))
```

#データ名\$変数名の後に「==」や「!=」、「>=」で、指定した変数内で該当行だけ抜くことも可能

```
d01_subdata4 <- subset(d01, d01$item == "Apples")
```

```
d01_subdata5 <- subset(d01_subdata4, d01_subdata4$country == "Japan")
```

# データの記述統計を確認

```
#平均と中央値、最小値、最高値、第一および第三四分位数を手軽に得るためのコマンド
d01_jp_app_sum <- summary(d01_subdata5)
```

```
#全般的な記述統計ならこれでOK。psychパッケージが必要
d01_jp_app_des <- describe(d01_subdata5)
```

#describeで出てくる値の説明  
# 「n」 , 「mean」 , 「sd」 , 「median」 はサンプル数、平均、標準偏差、中央値  
#trimmed = 中間項平均値：データに異常値が混ざっていて、平均値がその値に引きずられてしまう場合の対応策として、データの最大値と最小値付近の値を平均値の計算から除外して出したもの  
#mad = median absolute deviation = 中央絶対偏差：標準偏差の算出時に標本平均の代わりに中央値を用いてロバストな推定を目指した統計量。  
#minとmax、rangeは最小値、最大値、レンジ(データの最小値と最大値の差)  
#skew(歪度)とkurtosis(尖度)は正規分布への近さを調べる指標：データが正規分布に近い場合は、歪度が0付近・尖度が3付近、になる  
#se = standard error = 標準誤差：母集団からある数の標本を選ぶとき、選ぶ組み合わせに依って統計量がどの程度ばらつくかを、全ての組み合わせについて標準偏差で表したもの

# 簡単なグラフで視覚的にデータ確認

#シンプルな表現などは、Rにベースとして備わっている以下の関数(ファンクション)を活用できます:

```
#散布図のプロット    >>> plot()  
#2変数の組み合わせの散布図を行列形式で表現          >>> pairs()  
#箱ひげ図を作成      >>> boxplot()  
#ストリップチャート >>> stripchart()  
#ヒストグラムのプロット >>> hist()  
#密度分布 >>> dd <- density() した後に plot(dd)  
#バーチャート >>> barplot()  
#ラインプロット >>> plot() した後に lines()  
#円グラフ >>> pie()  
#ドットチャート >>> dotchart()  
#プロットにテキストを追加する >>> text()
```

# 簡単なグラフで視覚的にデータ確認

>上記の関数を使う際には、そのほとんどで、以下のような引数を活用してカスタマイズすることができる:

#pch: ポイント等の形(character)を変える。指定する数字(1から25まで)に対応して形が変わる

#cex: ポイント等のサイズを変える(character expansion).通常を1とした時の拡大率. 例: cex = 0.8.

#col: ポイント等の色を変更する. 例: col = "blue".

#frame: TRUE/FALSEで指定して、frame = FALSE になるとプロットパネルの境界線が無くなる。

#main, xlab, ylab: main title、 x/y 軸の名前を付ける 例：main = "＊＊＊＊"で書かれた言葉がタイトルに表示される

#las: x軸のラベルを、 las = 1ですべて水平に、 las = 2ですべて垂直に表示

**#Macで軸ラベルやタイトルに日本語使った時に□□□といった表記になってしまう場合は、図を表示する前に「par(family= “HiraKakuProN-W3” )」を実行してから図を表示するとうまく表示されます。**

# 簡単なグラフで視覚的にデータ確認

#読み込んだデータセットを使って、以下のコードで3つの図を作成してみましょう：

**#(1) 各変数のヒストグラムを作成して、分布を確かめる**

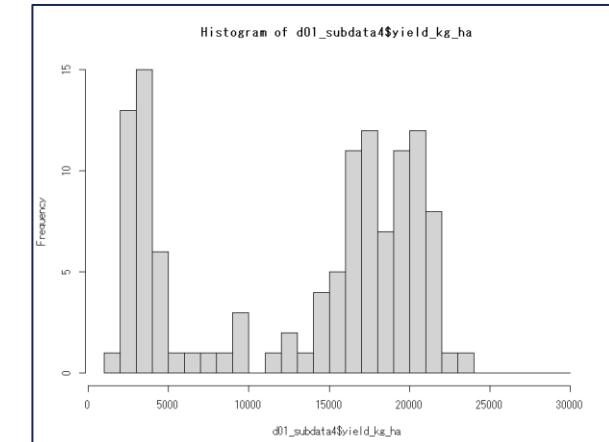
#(2) 一つの変数をx軸に、もう一つの変数をy軸にとった散布図

#(3) データの群ごとの散らばりを見る箱ひげ図

#もしさらに例が欲しい場合はこちら：R base Graphics on STHDA, <http://www.sthda.com/english/wiki/r-base-graphs>

#(1) ヒストグラムの作成

```
hist(d01$yield_kg_ha)
hist(d01_subdata4$yield_kg_ha)
hist(d01_subdata5$yield_kg_ha)
```



# 簡単なグラフで視覚的にデータ確認（1）

```
### ヒストグラムの調整と度数分布表の作成

#ヒストグラムを作る際に、範囲と階級の数を指定したい場合: breaks=seq()の中は左から、○から、×まで、△刻み

hist_data1 <- hist(d01_subdata4$yield_kg_ha); hist_data2 <- hist(d01_subdata4$yield_kg_ha, breaks = seq(1000, 30000, 1000))

#ヒストグラムを作った際に作成されたデータを見てみる

hist_data1$breaks; hist_data2$breaks #breaksは階級を区切る値 x1 < value =< x2

hist_data1$counts; hist_data2$counts #countsは度数

#ヒストグラムのデータを使って、度数分布表を作成する

n <- length(hist_data1$counts) # 階級の数

counts <- hist_data1$counts; breaks <- hist_data1$breaks #階級と度数の情報を格納しておく

class_names <- NULL # 階級の名前格納用に空のオブジェクトを作成

for(i in 1:n) {
  class_names[i] <- paste(breaks[i], "~", breaks[i+1])
}

(frequency_table <- data.frame(class=class_names, frequency=counts)) #A<-Bの格納を、(A<-B)とすると格納と確認を同時にできる
```

	class	frequency
1	0 ~ 2000	1
2	2000 ~ 4000	28
3	4000 ~ 6000	7
4	6000 ~ 8000	2
5	8000 ~ 10000	4
6	10000 ~ 12000	1
7	12000 ~ 14000	3
8	14000 ~ 16000	9
9	16000 ~ 18000	23
10	18000 ~ 20000	18
11	20000 ~ 22000	20
12	22000 ~ 24000	2

# 簡単なグラフで視覚的にデータ確認（2）

#読み込んだデータセットを使って、以下のコードで3つの図を作成してみましょう：

#(1) 各変数のヒストグラムを作成して、分布を確かめる

**#(2) 一つの変数をx軸に、もう一つの変数をy軸にとった散布図**

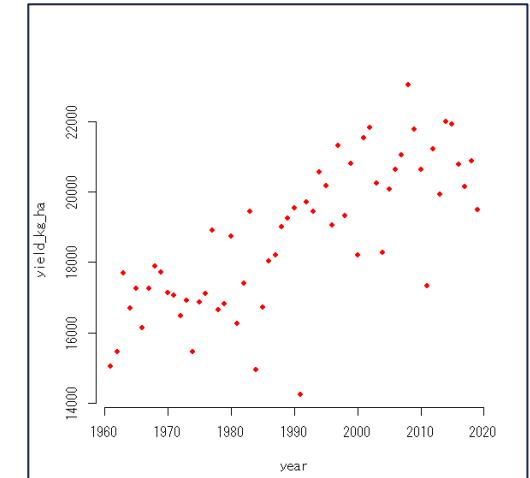
#(3) データの群ごとの散らばりを見る箱ひげ図

#もしさらに例が欲しい場合はこちら：R base Graphics on STHDA, <http://www.sthda.com/english/wiki/r-base-graphs>

#(2) 散布図を作成する

```
plot(d01_subdata5$yield_kg_ha)
```

```
plot(  
  x = d01_subdata5$year, y = d01_subdata5$yield_kg_ha,  
  pch = 19, cex = 0.8, frame = FALSE, col = "red",  
  xlab = "year", ylab = "yield_kg_ha"  
)
```



# 簡単なグラフで視覚的にデータ確認（3）

#読み込んだデータセットを使って、以下のコードで3つの図を作成してみましょう：

#(1) 各変数のヒストグラムを作成して、分布を確かめる

#(2) 一つの変数をx軸に、もう一つの変数をy軸にとった散布図

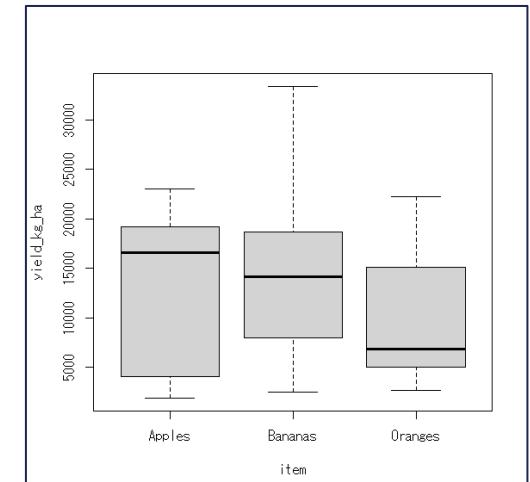
## #(3) データの群ごとの散らばりを見る箱ひげ図

#もしさらに例が欲しい場合はこちら：R base Graphics on STHDA, <http://www.sthda.com/english/wiki/r-base-graphs>

- データの群ごとの散らばりを見るには、箱ひげ図、boxplotが便利
- 箱ひげ図は、箱の中央太線が平均値、箱の上端と下端が第3・第1の四分位、上下の線の先端の値は、それぞれ四分位数 $\pm 1.5 \times (\text{第3四分位数} - \text{第1四分位数})$ になっています。それ以上の所にあるデータは外れ値として、○が表示されています。

#(3) 箱ひげ図を作成する

```
boxplot(yield_kg_ha~item, data=d01) #y軸(量)~x軸(カテゴリー)で指定
```



# データの書き出し

- 最後に、作成したデータをcsvファイルに書き出して、保存しましょう。
- 以下の様にwrite.csv()を使ってデータの書き出しを行います。書き出されたデータは作業ディレクトリに入っているので、確認しましょう。

```
#csvファイルに書き出してみる  
write.csv(d01_subdata5, "d01_jp_app_subdata.csv")  
  
write.csv(d01_jp_app_sum, "d01_jp_app_sum.csv")  
  
write.csv(d01_jp_app_des, "d01_jp_app_des.csv")
```

# 実習課題

- FAOSTATデータをRへインポートした上で、以下を実行してみましょう。
- 日本以外の国の一つの作物に限定したデータを抜き出す
- psychパッケージのdescribe()関数を利用して、記述統計を出す
- 抜き出したデータを使って、簡易的なヒストグラム(yieldのデータを利用)と散布図を作成。散布図はx軸にyear、y軸にyieldのデータを利用。
- 抜き出したデータをcsv形式で書き出し→作業ディレクトリに入っているかどうかを確認
- 余力があれば、データの加工や他の図の作成を試す

# 補：困った時のウェブリファレンス(Rに関して)

\*Rの使い方全般について >>> 私たちのR: ベストプラクティスの探求

(<https://www.jaysong.net/RBook/>)

\*RjpWikiの利活用

(<http://www.okadajp.org/RWiki/>)

\*R Documentation の利活用

(関数の使い方、引数、どのパッケージに属するのか等の情報を得る)

(<https://www.rdocumentation.org/>)

\*seekR: Rに関する情報に特化した検索エンジン

(<http://seekr.jp/>)

# 補：引数について分からなくなったら

★関数を使う際の引数に何を指定したら良いのか分からない時は、、、

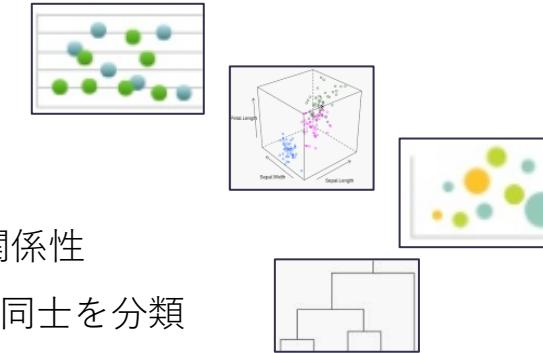
1. コンソールで「??関数名」を入れて実行 → ヘルプが表示される
  - \*だいたい、指定できる引数の説明と、最後に例が付いています！
2. 「RDocumentation」 (<https://www.rdocumentation.org/>) で検索
  - \*パッケージ名からも検索できます
3. 「CRAN R Project」のウェブサイトから、パッケージの**Vignette**を確認
  - \*Googleで「R パッケージ名 vignette」でだいたい引っかかります
  - \*「CRAN R Project」のページ(<https://cran.r-project.org/>)からは、左コラムの「Software」の欄にある「Packages」から「Table of available packages」でリストを表示させて検索(Ctrl + F)

# 補：データや結果の視覚化：表現方法まとめ

- データを使ってできることの中に視覚化がありましたが、その視覚化の中をさらに目的別に細分化し、それぞれの表現方法をまとめました。

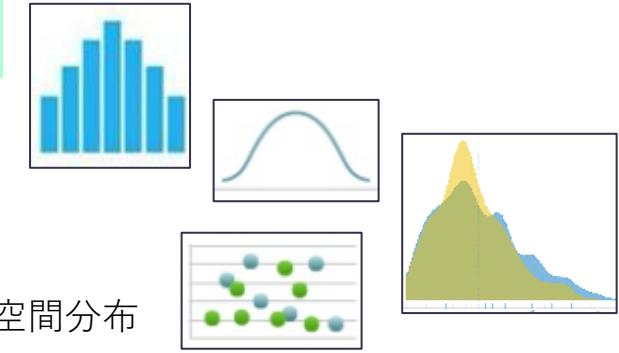
## 1) 関係性を見る

- 二次元散布図：2変数の関係性
- 三次元散布図：3変数の関係性
- バブルチャート：3変数(以上)の関係性
- クラスター図：関係性の強いもの同士を分類



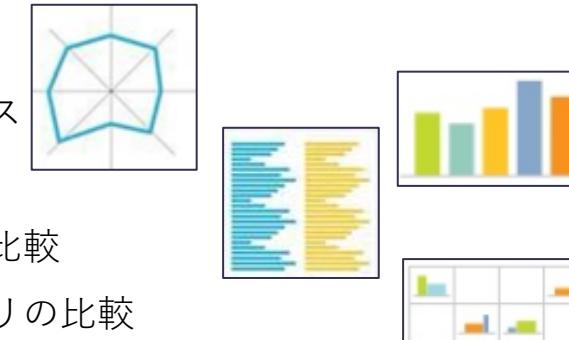
## 2) 分布を見る

- 棒ヒストグラム：少数の分布
- 線ヒストグラム：多数の分布
- 面ヒストグラム：密度分布
- 二次元散布図：2変数の2次元空間分布



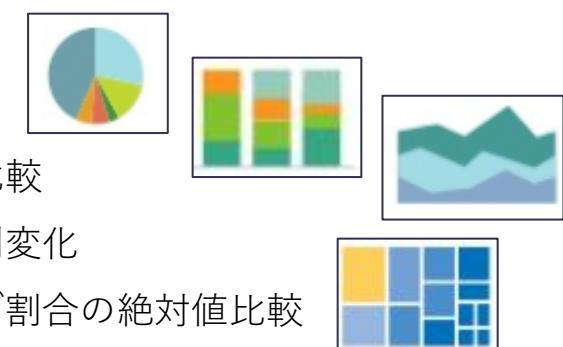
## 3) 比較する

- 蜘蛛の巣チャート：要素バランス
- 線グラフ：時系列の変化の比較
- 水平棒グラフ：少数カテゴリの比較
- 格子配置棒グラフ：多数カテゴリの比較



## 4) 構成を見る

- 円グラフ：静的な構成
- 積み上げ棒グラフ：複数の構成比較
- 積み上げ面グラフ：構成の時系列変化
- ツリーマップ：要素間の積み上げ割合の絶対値比較



# 補：表現方法から逆引きしてRのコードに接近

- Rをはじめとしたプログラミング言語やExcelの使いこなし方を身に着けるには、出力方法や表現方法などを先行研究などからまずイメージして、それに近づけることができる行程を逆引きして練習することが効果的です。以下は個人的な例ですが、参考までに。
1. 「The R Graph Gallery」 (<https://r-graph-gallery.com/>) や 「Data Visualization with R」 (<https://rkabacoff.github.io/datavis/>) 等で見てみる
  2. Googleのイメージ検索で「data visualization cool」などを試す
  3. ChatGPTに尋ねてみる

# 補：ChatGPTを使ったRの練習方法(一例です)

1. 「これからRをえるデータサイエンティストとして振舞ってください」と伝える（役割付け。なくても良いけど、ちょっとした味付けと思って）
2. 「これからデータをお伝えします」と一度送ってから、解析したデータのヘッダーやIDを含む一部をコピペして送る。\*全部を送ると大抵は分量オーバーになるので、サンプルコードを得るのに必要なところだけでOK。
3. （例えば、ヒストグラムを描くサンプルコードが知りたい場合は）「このデータを用いてヒストグラムを描くRのサンプルコードを教えてください」と送り、返ってくるコードをコピーして、自身のR環境で実践してみる。「yield\_kg\_haを対象にヒストグラムを...」や、「必要なパッケージがある場合はそれを明示してください」など必要に応じてカスタマイズするのが吉。
4. うまくいかない場合は、それを伝えて改善方法を探る。うまくいったら「ありがとうございました」と御礼を伝える（＾＾）
5. プラスアルファで「よりテクニカルな図として表現したい場合はどのようにすれば良いですか？」とか「ggplot2以外のパッケージを使ってみたいのですが」などで、さらに拡張！また、出力されたRコードが細切れで散らかってしまった場合は、「上記で出力したRコードを全てまとめてくれませんか？」などを送って返してもらった回答をまとめてコピペすると吉。
6. また、ウェブで見つけた素敵な図のスクショをChatGPTに読み込ませ、自分が持っているデータも読み込ませた上で、自分のデータを基に「スクショのような可視化の方法をR言語で書いてください」とやってみるのも手です。