R と exametrika パッケージの導入

Kosugitti

目次

0.1 自己紹介

- 小杉考司(こすぎこうじ)
- 専修大学人間科学部　教授　博士(社会学)
- 担当講義 ;心理学データ解析基礎, 心理学データ解析応用
- 専門分野
  - 心理尺度の作り方, 使い方
  - 多変量解析(因子分析, 多次元尺度構成法), 統計モデリング
  - 統計パッケージ開発；exametrika
- 関わった書籍
  - 数値シミュレーションで読み解く統計のしくみ〜R でためしてわかる心理統計
  - 研究論文を読み解くための多変量解析入門基礎篇: 重回帰分析からメタ分析まで
  - 言葉と数式で理解する多変量解析入門など

0.2 R の紹介

- **R 言語**はオープンソースの統計解析環境
  - 豊富な統計手法とグラフィックス機能
  - 無料で利用可能で継続的に発展中
  - 拡張パッケージが充実(CRAN, Bioconductor 等)
  - データサイエンス・統計分析の標準ツールの一つ
- **RStudio** は R をより使いやすくする統合開発環境(IDE)
  - コード編集、実行、可視化が一画面で完結

- プロジェクト管理機能で作業を整理
- Rmarkdown/Quarto による再現可能な分析レポート作成

0.3　R のはじめかた

~~0. SPSS や SAS などの統計ソフトをアンインストールします~~

1. `CRAN`（しーらん）と検索します。`The Comprehensive R Archive Network` というサイトが出てくるはずです。
2. 自分の OS/CPU に合ったページから，最新版をダウンロードします。現在は R4.4.2 になります。
3. 指示に従ってインストール！「次へ」を連打するだけでいいです。簡単ですね！

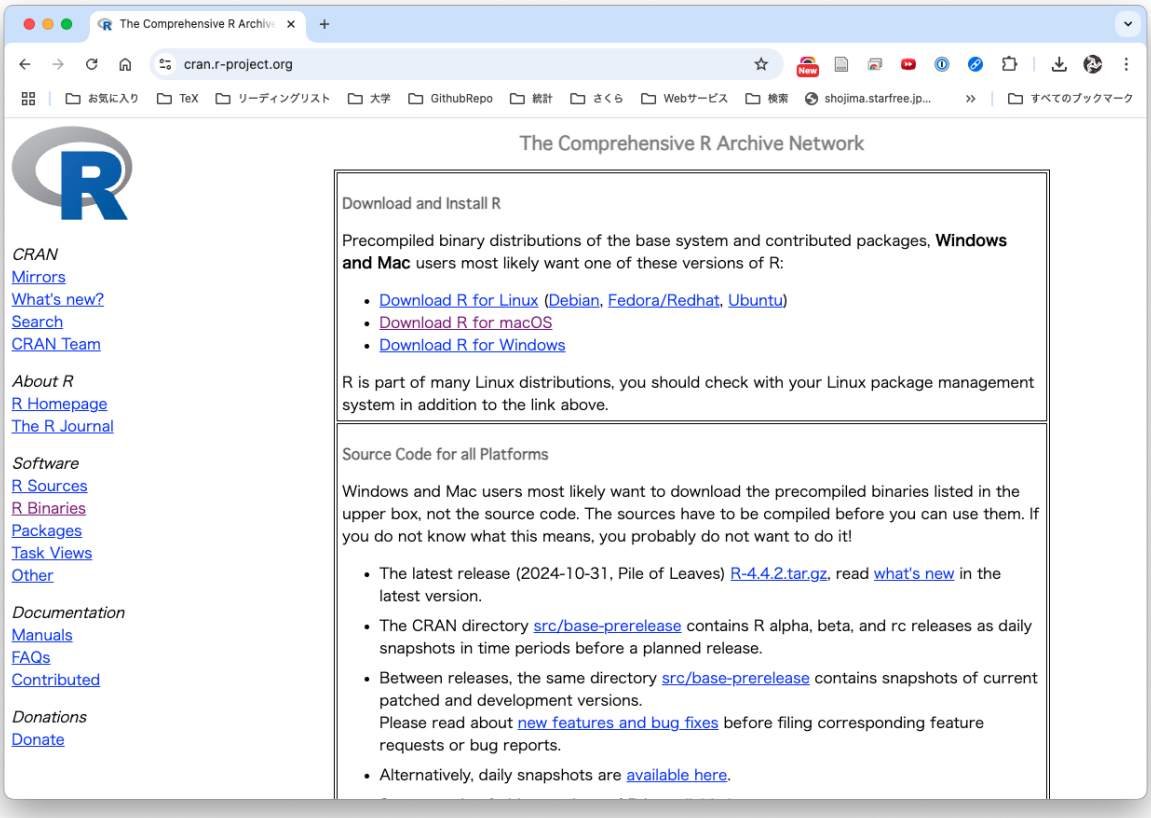
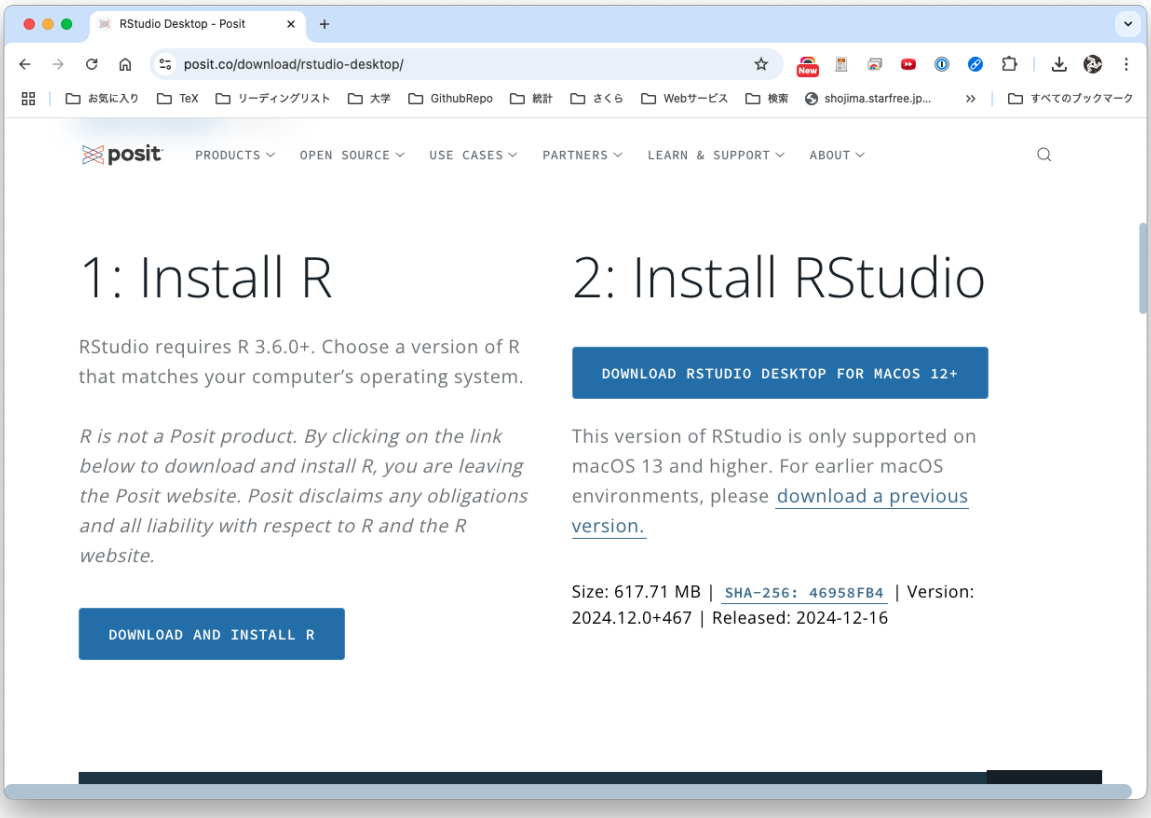


図 1: cran のページ

0.3.1 RStudio も使いましょう

1. `RStudio` で検索します。`RStudio Desktop` あるいは Posit 社が出てきます。
2. `Install RStudio` から RStudio Desktop をダウンロードしてインストールしましょう。

RStudio は Server 版もあります。サーバを用意すればブラウザ経由で簡単に使える利点があります。


RStudio Desktop - Posit
posit.co/download/rstudio-desktop/
お気に入り
TeX
リーディングリスト
大学
GithubRepo
統計
さくら
Webサービス
検索
shojima.starfree.jp...
すべてのブックマーク
posit
PRODUCTS
OPEN SOURCE
USE CASES
PARTNERS
LEARN & SUPPORT
ABOUT
1: Install R
RStudio requires R 3.6.0+. Choose a version of R that matches your computer's operating system.
R is not a Posit product. By clicking on the link below to download and install R, you are leaving the Posit website. Posit disclaims any obligations and all liability with respect to R and the R website.
DOWNLOAD AND INSTALL R
2: Install RStudio
DOWNLOAD RSTUDIO DESKTOP FOR MACOS 12+
This version of RStudio is only supported on macOS 13 and higher. For earlier macOS environments, please download a previous version.
Size: 617.71 MB | SHA-256: 46958FB4 | Version: 2024.12.0+467 | Released: 2024-12-16

0.3.2 RStudio の起動画面

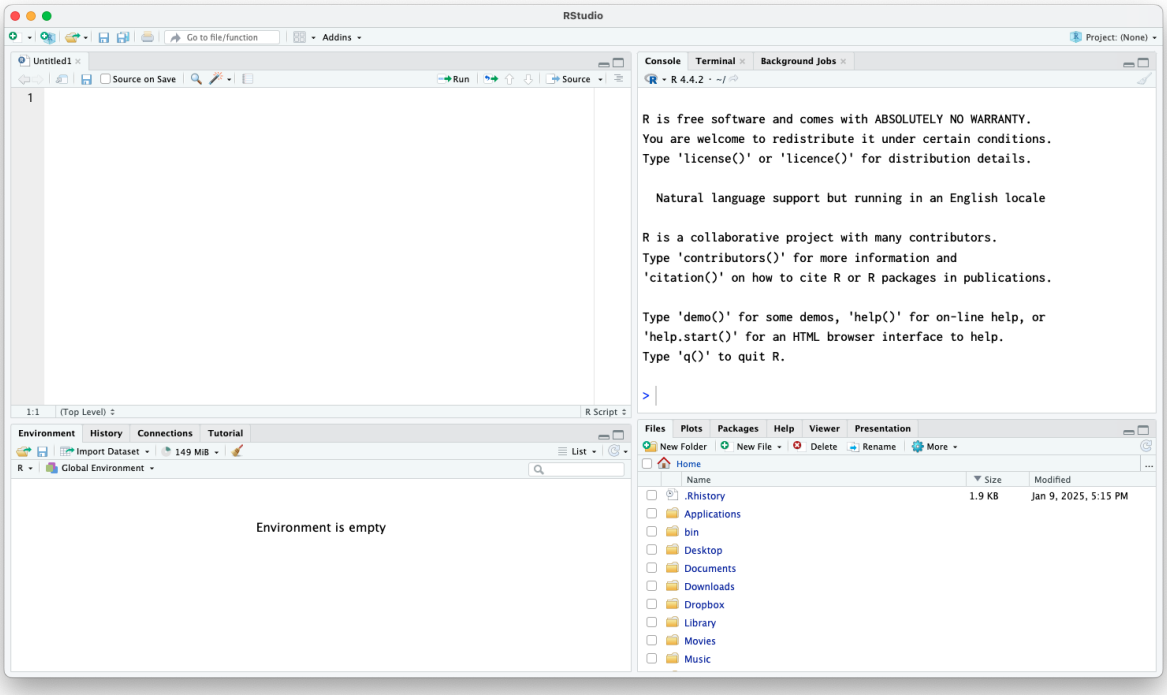


図 2: RStudi 起動画面

- 大きく 4 分割して使います。
- 起動して最初にやるのが「**環境設定**」です。
- メニューバーから，Tools > Global Options と進みます。

0.3.3 オススメ設定

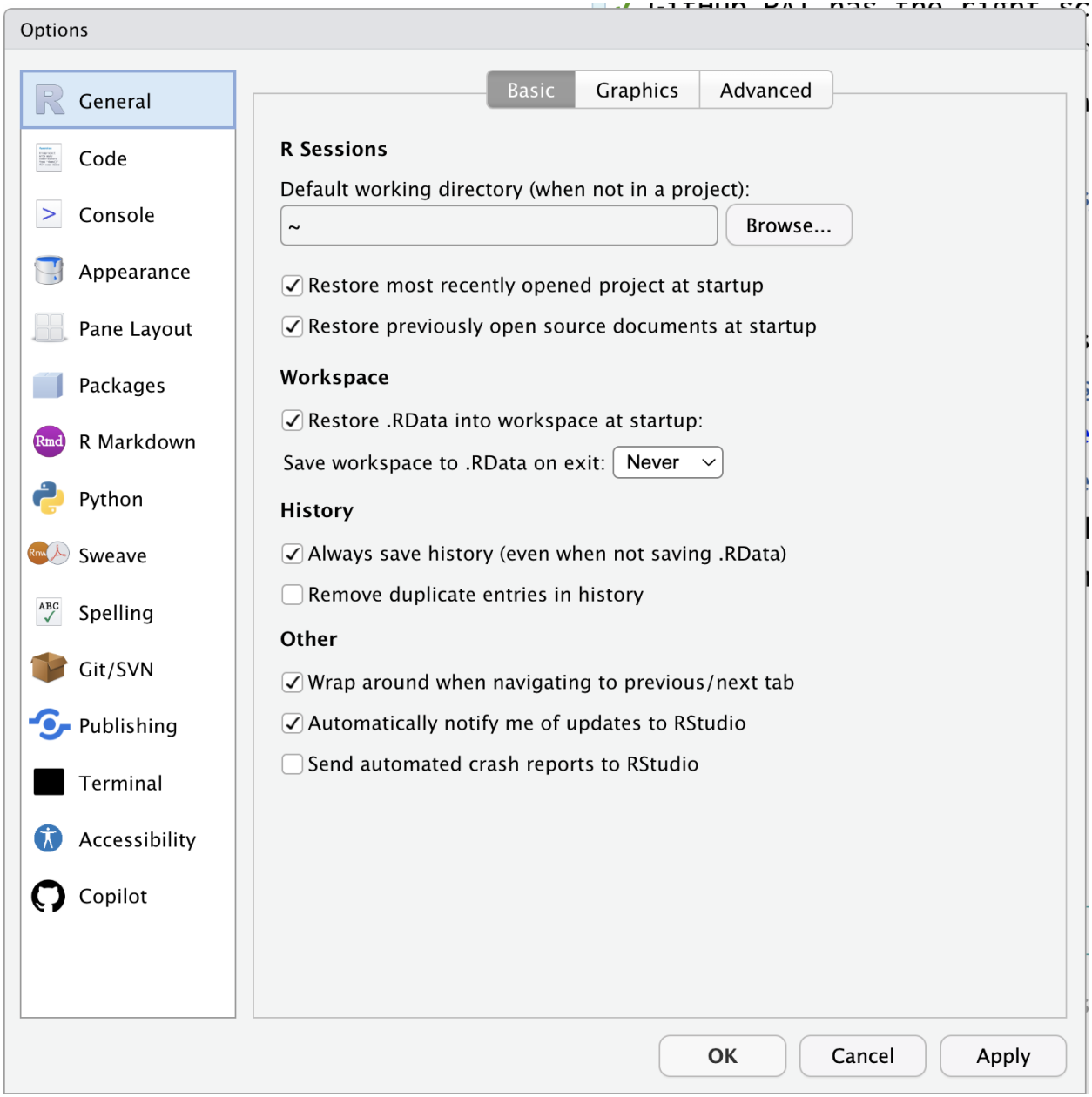


図 3: 環境設定画面

- General > Basic の Wrokspace, `Save Workspace to .RData on exit:` を **never** に
- General > Graphics > Graphics Device の `Backend` を **AGG** に
- Appearance の `Editor Font` を見やすいフォントにしましょう
- Appearance の `Editor Font size` を見やすい大きさにしましょう

おすすめフォント

- Bizin Gothic
- HackGen

0.3.4 オススメ設定 (つづき)

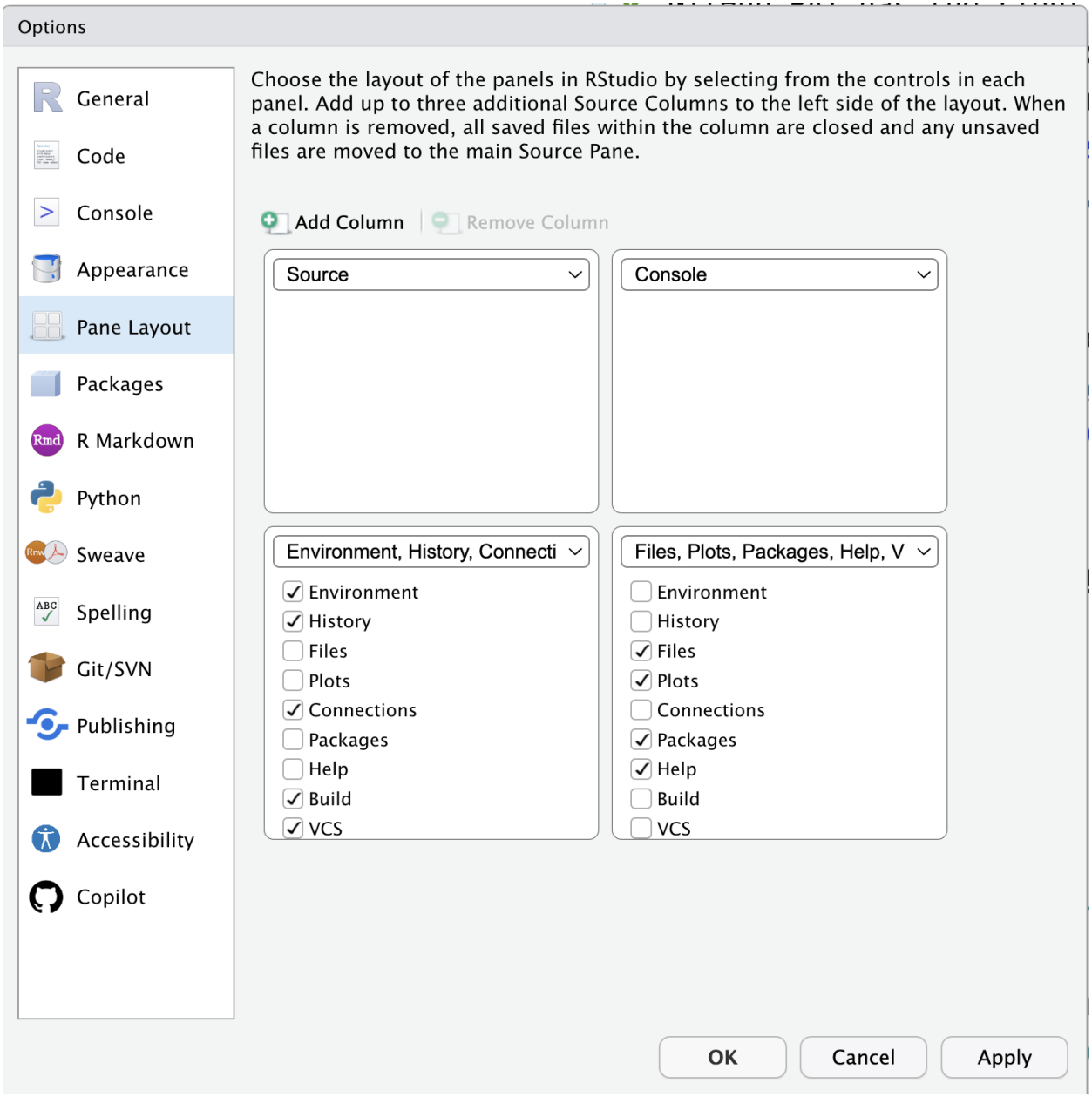


図 4: 環境設定画面その 2

- Pane Layout を
  - **Source** と **Cosole** を横並びに
  - かなりワイドな画面をお使いの方は，`Add Column` で 3 列にして source pane を一列増やそう

- 設定が終わったら **Apply(適用)** ボタンをおして，**OK** で閉じる

0.3.5 RStudio の 4 つの窓

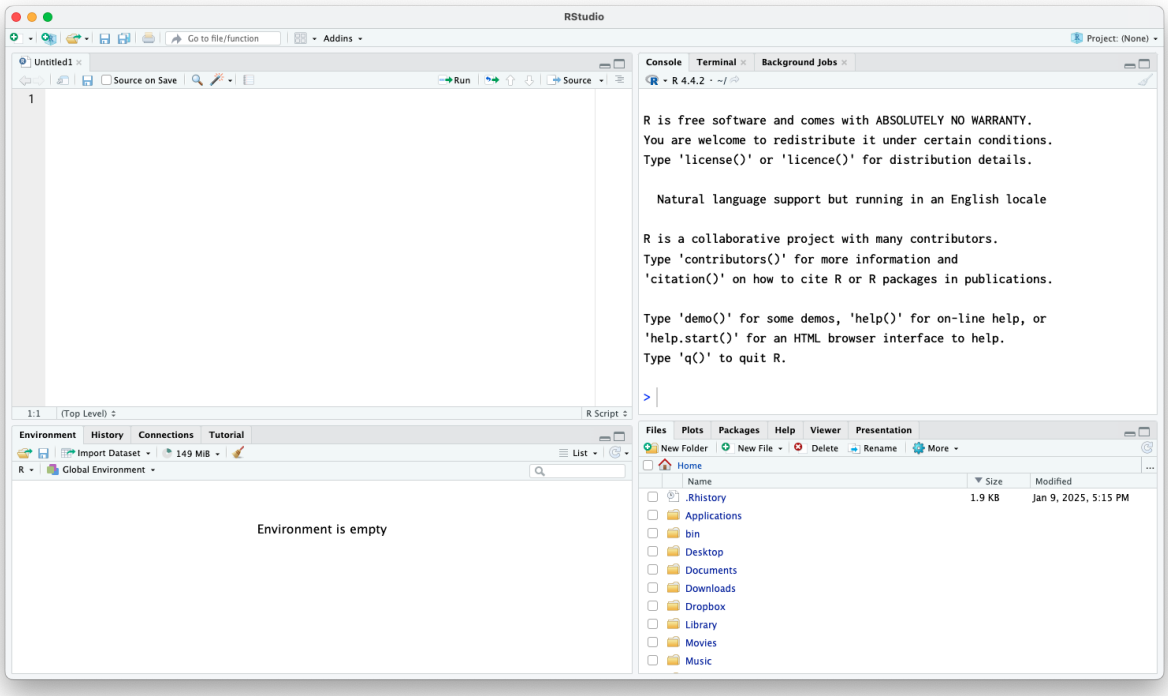


図 5: 4 つのペイン

- `Source` ペインはエディタ領域で，R スクリプトを書く場所。
- `Console` ペインは R エンジン。直接 R コードを書いてもいいし，`Source` から一行ずつ, あるいは `Source` 全体を流し込んで計算を実行する。

0.3.6 RStudio の 4 つの窓

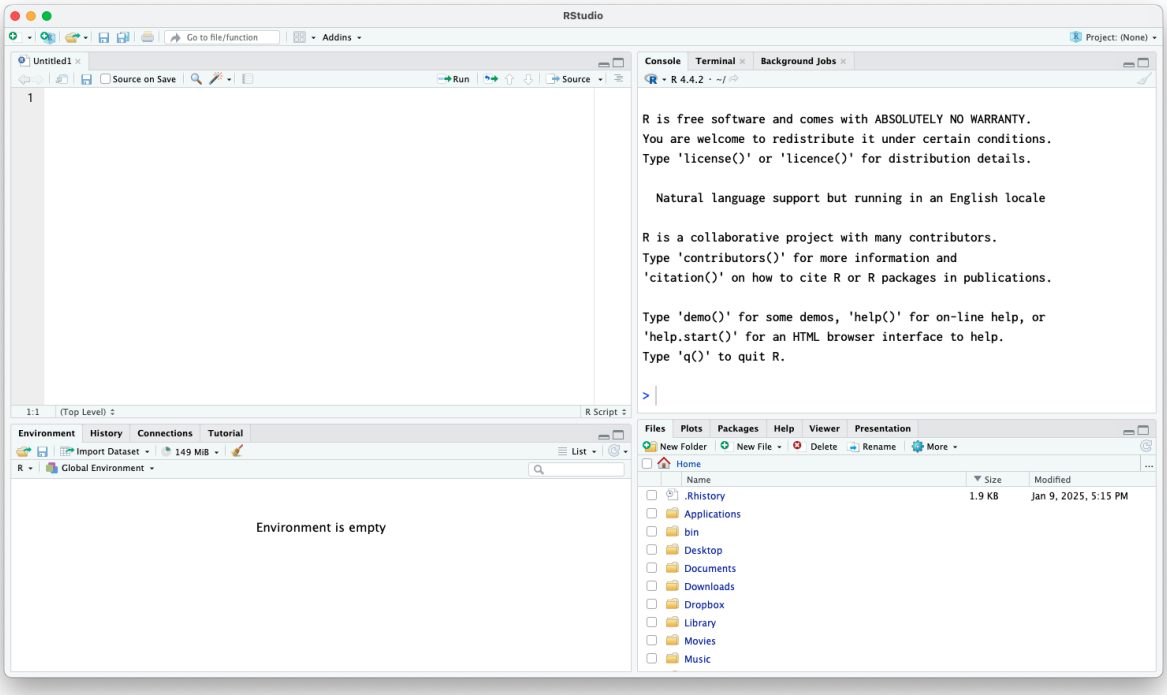


図 6: ペインの確認 1

- `Environment` はメモリに入っている変数・オブジェクトを表示
- `Files` はワーキングディレクトリの表示, 簡単な操作
- `Package` はパッケージ管理 (後述)
- `Plots`,`Viewer` は出力表示

0.3.7 R はプロジェクト管理が基本

- プロジェクト＝フォルダに紐づいた作業環境を作ろう
  - File > New Project から New Directory/Existing Directory/Version Control を選ぶ
    - New Directory; 新しいフォルダで作業開始
    - Existing Directory; 既存のフォルダをプロジェクトと紐付け
    - Version Control; Github レポジトリとプロジェクトを紐付け

プロジェクトにしておくと, 作業フォルダの設定も自動でなされるから, ファイルの読み込みなどでパスの指定が楽になります。

- 今回の春セミ用にプロジェクトフォルダを作りましょう！
  - すでにフォルダに色々まとめている人は，Existing Directory から

  - まだフォルダがない人は，New Directory から

0.4 R をさわってみましょう

0.4.1 はじめの 1 歩

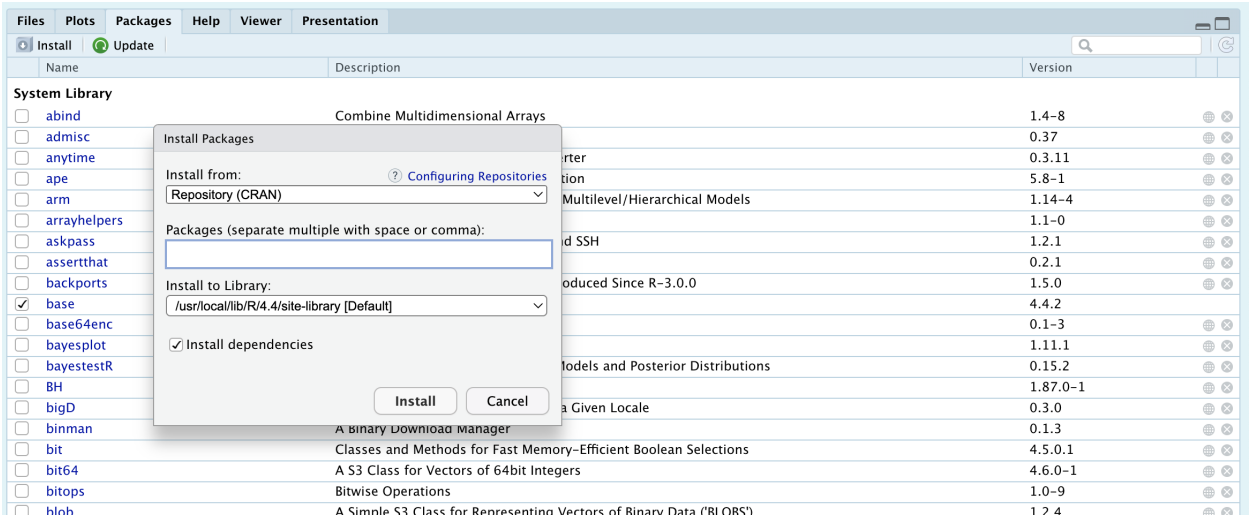
- R はインタプリタ言語＝一問一答
  - Console に>が出ていたら聞く準備ができています。
  - Console に + が出ていたら前の入力が終わってません。
- 直接 Console に書き込むのではなく, スクリプトに書きましょう。
  - File > New File > R Script と進むと無名のスクリプトファイルが開きます
- スクリプトファイルが開いたら, まず次のように書きます。

```
rm(list = ls())
```

- 一行目は呪文のようなものだと思ってください。
  - `rm` という関数は remove を意味していて, 現在 R のメモリにある変数やオブジェクトを除外します。
  - `list=ls()` は「メモリのすべてのオブジェクトリスト」を意味するので, これで環境の初期化になります。

0.4.2 パッケージ

- パッケージは関数のセット。元の R に追加するだけで機能が増えます。
- パッケージは CRAN を通じて公開され, ペインの `Packages` タブで管理できます。



- デフォルトでは CRAN から取ってくることになります。(要ネット環境)
  - Packages のところで `exametrika` と入力してインストールしちゃいましょう。
  - あるパッケージが他のパッケージを必要とすることもあります。これを**依存パッケージ**といいます。
  - RStudio の Packages タブでは `install dependencies` にチェックがあるのがデフォルトです。

    * 依存パッケージがあれば自動的にインストールされます。
    * `exametrika` は `igraph` などに依存していますので, それらが同時に導入されます

0.4.3 パッケージの使い方

- パッケージを使うには `library` と書きます。

```
library(exametrika)
```

```
Loading required package: mvtnorm
```

```
Loading required package: igraph
```

```

Attaching package: 'igraph'
```

```
The following objects are masked from 'package:stats':

    decompose, spectrum
```

```
The following object is masked from 'package:base':

    union
```

- これで `exametrika` パッケージの持つ関数が実行できるようになりました！他のパッケージも同様です。
- パッケージのインストールを毎回する必要はありません。インストールは「手に入れる」ということだからです。
- パッケージの実装 (`library`) はセッション毎に行う必要があります。これは「そうびする」ようなものです。
- R スクリプトの冒頭で `rm(list=ls())` としましたが, 分析に必要なパッケージはスクリプトの最上部にまとめて書いておきましょう。
    - R はインタプリタなので, 逐次的に処理が進みますが, 行ったり来たりしていると「パッケージを読み込んだっけ？」とか「今は何の数字で何の計算をしてるんだっけ？」となってしまいます。
    - 細かいことですが, パッケージは読み込む順番に影響されることがあります。
        * 同じ関数名を異なるパッケージが使っている場合, 後で読み込まれた方が上書きされます。
        * 混同しないように `PackageName::function` のように`::` で明示することがあります。

0.4.4 数値計算の基礎

- スクリプトに四則演算を書いて，Cmd+Enter でコンソールに送ります。
- 複数行選択/Run ボタン/Source ボタンをつかってもいいでしょう。

```
1 + 2
```

```
[1] 3
```

```
3 - 4
```

```
[1] -1
```

```
5 * 6
```

```
[1] 30
```

```
7 / 3
```

```
[1] 2.333333
```

- 出力に [1] とあるのは気にしないでください。
  - R はベクトルで処理します。今回の演算も, 要素が 1 つのベクトルとして考えて処理しています。

0.4.5 数値計算の基礎

- 計算結果を保持する, あるいは名前をつけて管理することができます。
- R は「名前をつけて管理する対象」をすべて**オブジェクト**といいます。

```
a <- 1 + 2
b <- 3 - 4
print(a)
```

```
[1] 3
```

```
print(b)
```

```
[1] -1
```

```
print(a + b)
```

```
[1] 2
```

- <-で代入を意味します。ショートカット (ALT と-,option と-) も覚えておこう
- RStudio の Environment タブに保存されているオブジェクトが表示されています。ダブルクリックで確認できます。

```
a <- 5
a + b
```

```
[1] 4
```

同じオブジェクト名なら上書きされることに注意

0.4.6 ベクトル,行列,リスト,データフレーム

- 複数の数字のセット，**ベクトル**は c() でくくることで表現します。
  - 連続した数字はコロン: で表現します。
- 2 次元に並ぶ数字のセット，**行列**は matrix() でつくります。
  - matrix 関数にベクトルを与えるなどします。
- 3 次元以上の数字のセット，**配列**は array() で，dim オプションで各次元の大きさを指定します。
- 数字, 文字, 論理値 (T/F) などが混在するもののセット，**リスト**は list() でつくります。
- リストの中でも矩形に整っている**データフレーム**は，data.frame() でつくります。

分析するときはデータフレームがもっともよく使われます

データフレームの上位互換，tibble という型もあります。これは tibble パッケージを読み込むことで使えるようになります。

0.4.7 ベクトル(Vector)の例

0.4.7.1 数値ベクトル

```
x <- c(1, 2, 3, 4, 5)
print(x)
```

```
[1] 1 2 3 4 5
```

0.4.7.2 文字列ベクトル

```
y <- c("りんご", "みかん", "バナナ")
print(y)
```

```
[1] "りんご" "みかん" "バナナ"
```

0.4.7.3 論理値ベクトル

- R には文字, 数字以外に論理値というのがあります。真/TRUE か偽/FALSE か, を表します。
- 使い方としては, 論理判断の条件で使ったり, オプションの「スイッチオン・オフ」を表す時につかいます。
- 大文字の T や F は論理値を表す特別な用語 (予約語) です。

```
z <- c(TRUE, FALSE, TRUE)
print(z)
```

```
[1]  TRUE FALSE  TRUE
```

0.4.8 行列(Matrix)

- 1 から 9 までの数字で 3×3 行列を作成

```
m1 <- matrix(1:9, nrow = 3, ncol = 3)
print(m1)
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

- 行名と列名を付ける

```
m2 <- matrix(1:9,
    nrow = 3, ncol = 3,
    dimnames = list(
        c("A", "B", "C"),
        c("X", "Y", "Z")
    )
)
print(m2)
```

```
  X Y Z
A 1 4 7
B 2 5 8
C 3 6 9
```

0.4.9 配列(Array)

- 2×3×2 の 3 次元配列を作成

```
arr <- array(1:12, dim = c(2, 3, 2))
print(arr)
```

```
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2
```

```
     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
```

0.4.10 リスト(List)

- 様々な型のデータを含むリストを作成

```r
my_list <- list(
    numbers = c(1, 2, 3),
    text = "Hello",
    logical = TRUE,
    matrix = matrix(1:4, 2, 2)
)
print(my_list)
```

```
$numbers
[1] 1 2 3

$text
[1] "Hello"

$logical
[1] TRUE

$matrix
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

0.4.11 リスト(List)

- リストの要素へのアクセス
  - 名前付きリストなら$マークで呼び出せます

```r
my_list$numbers
```

```
[1] 1 2 3
```

```r
my_list$numbers[3]
```

```
[1] 3
```

```
my_list$matrix[, 2]
```

```
[1] 3 4
```

```
my_list$matrix
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

0.4.12 データフレーム(Data Frame)

- データフレームの作成例
  - データフレームはリストの特殊な型なので, リストを `as.data.frame` 関数で変換しても OK

```
df <- data.frame(
    name = c("田中", "鈴木", "佐藤"),
    age = c(25, 30, 28),
    gender = c("M", "F", "M"),
    height = c(170, 160, 175)
)

print(df)
```

```
  name age gender height
1 田中  25      M    170
2 鈴木  30      F    160
3 佐藤  28      M    175
```

- 要素へのアクセスの仕方はリストと同じです

```
df$age
```

```
[1] 25 30 28
```

0.4.13 データ構造の比較

特徴	ベクトル	行列	配列	リスト	df	Tibble
次元	1 次元	2 次元	n 次元	階層構造	2 次元	2 次元
型の統一	必要	必要	必要	不要	列ごと	列ごと
データ型	単一	単一	単一	複数可	複数可	複数可
主な用途	単純な数列	数値計算	多次元データ	複雑なデータ	データ分析	データ分析

tibble 型はデータフレームの上位互換で，tibble パッケージを使うことで導入できます。主な特徴は次のとおりです。

- 型情報の表示
- 行数と列数の表示
- データの一部のみ表示(大きなデータセット時に便利)

0.4.14 パイプ演算子を活用しよう

- パイプ演算子は、データの処理を順番に繋げてくれる記号
- 左から右へ、データが流れていくイメージ！コードが読みやすく、理解しやすくなる

0.4.14.1 基本的な使い方

- パイプ演算子を使わないでいると？

```
result <- sum(sqrt(abs(log(c(1:10)))))
```

- パイプ演算子を使ってみると？

```
result <- c(1:10) |>
    log() |>
    abs() |>
    sqrt() |>
    sum()
```

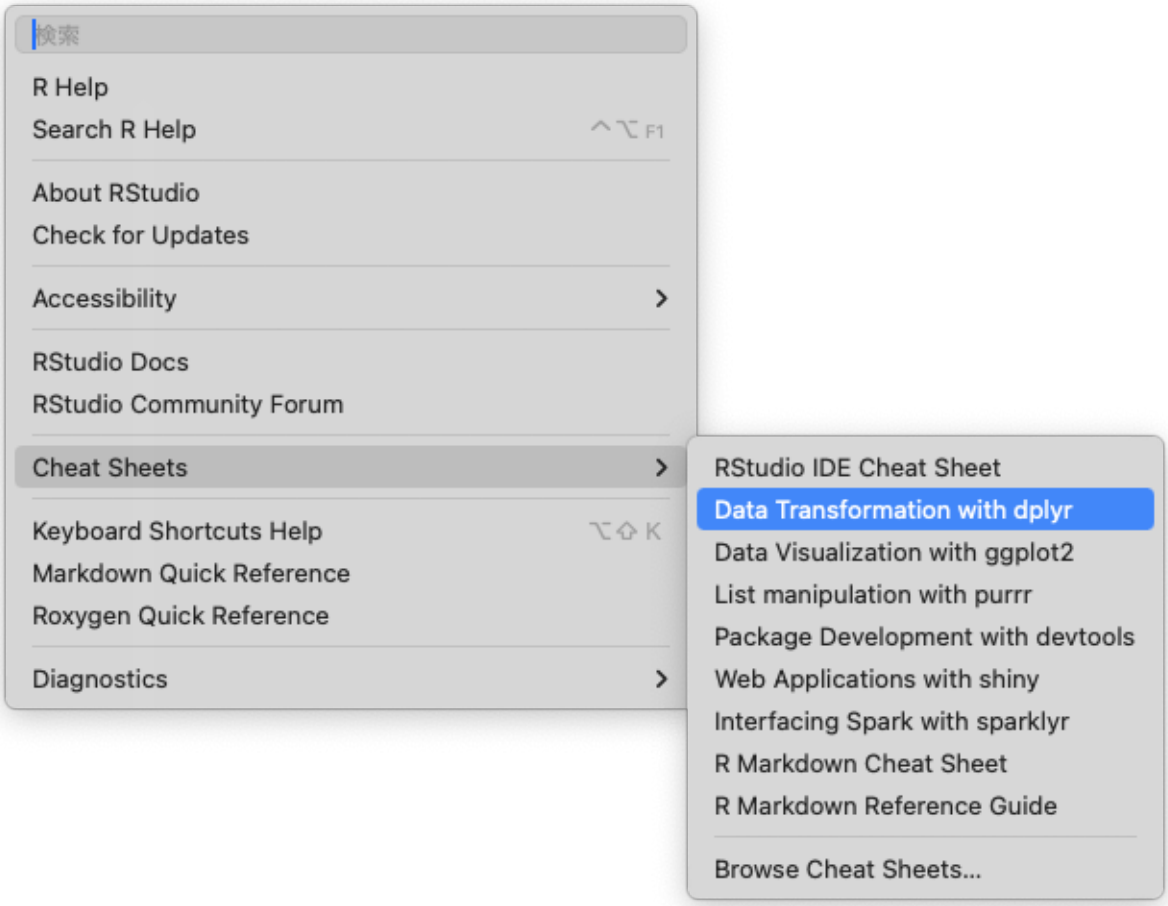
- パイプ演算子はショートカット Ctrl/Cmd + Shift + M で入力できます
  - |>は R4.1 以降使えるようになった，R のもってるパイプ演算子
  - %>% は magrittr パッケージや, それを含んだ tidyverse パッケージで以前から使われていたもの

0.4.15 (余談)tidy な世界

- tidyverse パッケージは, データハンドリングを画期的に簡単にしたパッケージで, これで R のユーザが一気に広がったと言っても過言ではありません。
- tidyverse パッケージはパッケージのパッケージ。
  - 大規模データ用のデータフレーム，tibble
  - パイプ演算子のパッケージ magrittr
  - 描画を綺麗にしてくれるパッケージ ggplot2 などが含まれます
- 専門の書籍も出ています tidyverse パッケージを基本にした改訂 2 版 R ユーザのための RStudio[実践]入門〜tidyverse によるモダンな分析フローの世界

0.4.16　(余談) チートシートを活用しよう

- RStudio のメニューバー，Help> Cheat Sheets と進んでください
- PDF ファイル 1，2 枚分で基本的な使い方を始めとした, 様々なチートシートが現れます！



0.5　具体的にデータを扱ってみよう

0.5.1　データの読み込みと操作

0.5.1.1　CSV ファイルの読み込み

- CSV ファイルは R でもっとも一般的なデータ形式の一つです
- エクセルファイルなどと違って, アプリケーションに依存せず, メモ帳で開くこともできますので, あらゆる OS に対応できます。
- 基本的な読み込み方法は `read.csv()` 関数を使います
- `tidyverse` パッケージを使っている人は，`read_csv()` 関数のほうが細かな調整が効いていいかも

0.5.2 基本的な CSV 読み込み

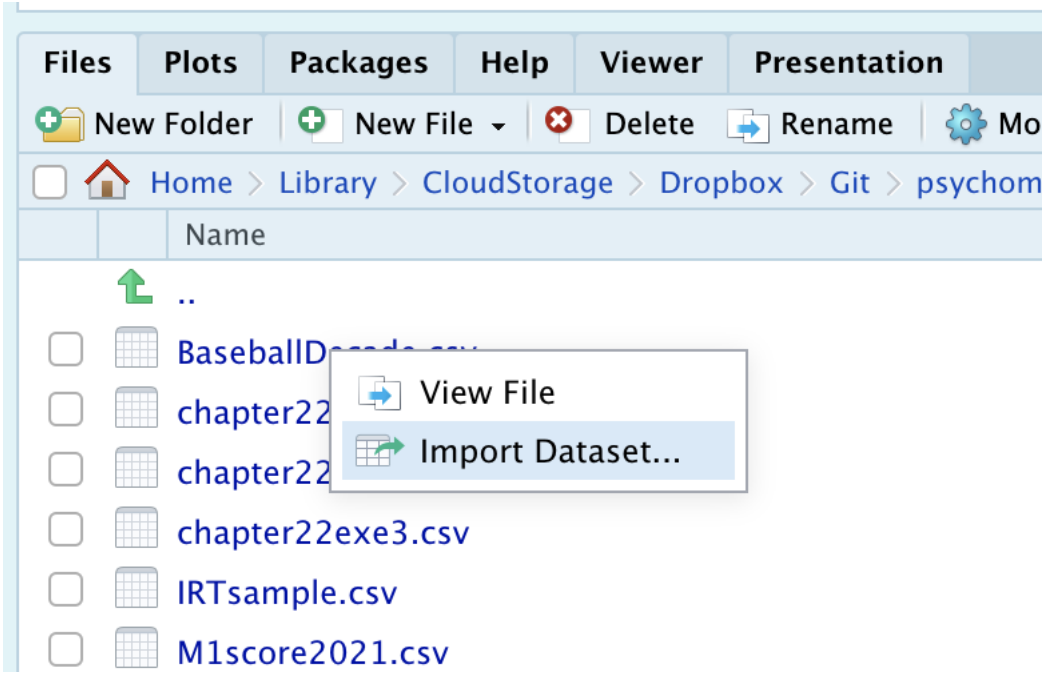
```
data <- read.csv("data.csv")
```

0.5.2.1 日本語を含む CSV ファイルの場合

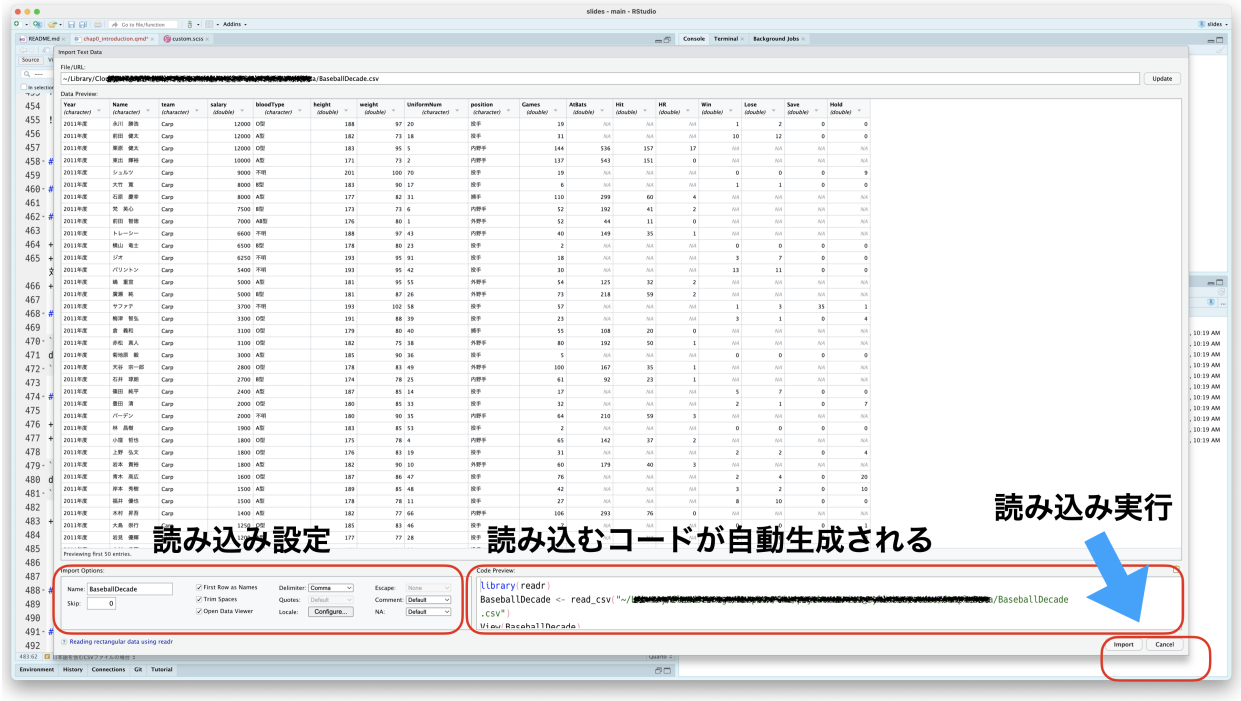
- Windows ユーザ/Excel ユーザは文字化けを起こす可能性があります。
- 世界標準である UTF-8 という文字コードでファイルを管理しましょう

```
data <- read.csv("data.csv", fileEncoding = "UTF-8")
```

- Rstudio の Files タブからファイルを選んで Import Dataset とすると GUI でも操作できます。
  - Excel ファイルを読み込みたい場合は, そちらを使うのもいいでしょう



0.5.3　Import Dataset



0.5.4　サンプルコードを読み込んでみよう

- インターネットから読み込むこともできます！
- 次のコードでサンプルデータを読み込んでみましょう。

```
baseball <- read.csv("https://shorturl.at/X4ctc")
```

- ショート URL の参照先は怪しいところではありません。
  - https://kosugitti.github.io/psychometrics_syllabus/codes/SampleData/BaseballDecade.csv
  - 私の心理統計教育教材サイトに置いてあるサンプルデータです
  - 野球選手の基本情報など，10 年分のデータがあります。
- データの一部 (冒頭) を head 関数で確認してみましょう

```
head(baseball)
```

```
      Year        Name team salary bloodType height weight UniformNum position
1 2011 年度 永川　勝浩 Carp  12000       O 型    188     97         20     投手
2 2011 年度 前田　健太 Carp  12000       A 型    182     73         18     投手
3 2011 年度 栗原　健太 Carp  12000       O 型    183     95          5   内野手
4 2011 年度 東出　輝裕 Carp  10000       A 型    171     73          2   内野手
5 2011 年度   シュルツ Carp   9000      不明    201    100         70     投手
```

```
6 2011年度　大竹　寛 Carp   8000       B型    183     90         17      投手
  Games AtBats Hit HR Win Lose Save Hold
1    19     NA  NA NA   1    2    0    0
2    31     NA  NA NA  10   12    0    0
3   144    536 157 17  NA   NA   NA   NA
4   137    543 151  0  NA   NA   NA   NA
5    19     NA  NA NA   0    0    0    9
6     6     NA  NA NA   1    1    0    0
```

0.5.5 オブジェクトの基本情報

- str 関数, あるいは Environment タブにあるオブジェクト名を開くと, 基本情報が確認できます。

```
str(baseball)
```

```
'data.frame':   6546 obs. of  17 variables:
 $ Year      : chr  "2011年度" "2011年度" "2011年度" "2011年度" ...
 $ Name      : chr  "永川　勝浩" "前田　健太" "栗原　健太" "東出　輝裕" ...
 $ team      : chr  "Carp" "Carp" "Carp" "Carp" ...
 $ salary    : int  12000 12000 12000 10000 9000 8000 8000 7500 7000 6600 ...
 $ bloodType : chr  "O型" "A型" "O型" "A型" ...
 $ height    : int  188 182 183 171 201 183 177 173 176 188 ...
 $ weight    : int  97 73 95 73 100 90 82 73 80 97 ...
 $ UniformNum: int  20 18 5 2 70 17 31 6 1 43 ...
 $ position  : chr  "投手" "投手" "内野手" "内野手" ...
 $ Games     : int  19 31 144 137 19 6 110 52 52 40 ...
 $ AtBats    : int  NA NA 536 543 NA NA 299 192 44 149 ...
 $ Hit       : int  NA NA 157 151 NA NA 60 41 11 35 ...
 $ HR        : int  NA NA 17 0 NA NA 4 2 0 1 ...
 $ Win       : int  1 10 NA NA 0 1 NA NA NA NA ...
 $ Lose      : int  2 12 NA NA 0 1 NA NA NA NA ...
 $ Save      : int  0 0 NA NA 0 0 NA NA NA NA ...
 $ Hold      : int  0 0 NA NA 9 0 NA NA NA NA ...
```

- 何年度のデータか (Year), 選手名 (Name), どのチーム所属か (team), 年俸 (salary) などがあります。
- データの型もわかります
  - chr は文字列型です。四則演算の対象ではありません。
  - int,num は数字です (整数と実数)
  - NA は欠損値を表しています。
- read.csv 関数は読み込んだデータを自動的にデータフレーム型にします。

0.5.6 記述統計量

- summary 関数で要約統計量を算出できます

```
summary(baseball)
```

```
     Year             Name              team              salary
 Length:6546        Length:6546        Length:6546        Min.   :  200
 Class :character   Class :character   Class :character   1st Qu.: 1000
 Mode  :character   Mode  :character   Mode  :character   Median : 2000
                                                          Mean   : 5178
                                                          3rd Qu.: 5700
                                                          Max.   :65000

  bloodType             height          weight      UniformNum
 Length:6546        Min.   :163.0   Min.   : 60   Min.   : 0.00
 Class :character   1st Qu.:177.0   1st Qu.: 78   1st Qu.:16.00
 Mode  :character   Median :180.0   Median : 83   Median :33.00
                    Mean   :180.7   Mean   : 84   Mean   :34.93
                    3rd Qu.:184.0   3rd Qu.: 89   3rd Qu.:52.00
                    Max.   :216.0   Max.   :135   Max.   :99.00

   position            Games            AtBats           Hit
 Length:6546        Min.   :  1.00   Min.   :  0.0   Min.   :  0.00
 Class :character   1st Qu.:  9.00   1st Qu.: 23.0   1st Qu.:  4.00
 Mode  :character   Median : 25.00   Median : 95.0   Median : 21.00
                    Mean   : 40.67   Mean   :165.1   Mean   : 42.53
                    3rd Qu.: 61.00   3rd Qu.:271.0   3rd Qu.: 69.00
                    Max.   :144.00   Max.   :603.0   Max.   :216.00
                                     NA's   :3233    NA's   :3233
       HR              Win              Lose             Save
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.00
 1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.00
 Median : 1.000   Median : 1.000   Median : 1.000   Median : 0.00
 Mean   : 3.967   Mean   : 2.517   Mean   : 2.509   Mean   : 1.27
 3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 4.000   3rd Qu.: 0.00
 Max.   :60.000   Max.   :24.000   Max.   :15.000   Max.   :54.00
 NA's   :3233     NA's   :3307     NA's   :3307     NA's   :3307
      Hold
 Min.   : 0.000
```

```
 1st Qu.: 0.000
 Median : 0.000
 Mean   : 3.511
 3rd Qu.: 3.000
 Max.   :45.000
 NA's   :3307
```

- 行数, 列数を確認して, データのサイズを見ておきましょう

```
NROW(baseball)
```

```
[1] 6546
```

```
NCOL(baseball)
```

```
[1] 17
```

0.5.7 変数毎の要約統計量

- 変数に$でアクセスして, 要約統計量を計算してみましょう。

```
mean(baseball$height)
```

```
[1] 180.7177
```

```
sd(baseball$height)
```

```
[1] 5.613504
```

```
median(baseball$weight)
```

```
[1] 83
```

```
max(baseball$salary)
```

```
[1] 65000
```

```
min(baseball$salary)
```

```
[1] 200
```

```
quantile(baseball$salary)
```

```
   0%   25%   50%   75%  100%
  200  1000  2000  5700 65000
```

0.5.8 因子型をつくってみましょう

- チーム名はたかだか 12 種類です。名義尺度水準＋ラベルの数値である Factor 型にしてみましょう。

```
baseball$team <- as.factor(baseball$team)
summary(baseball$team)
```

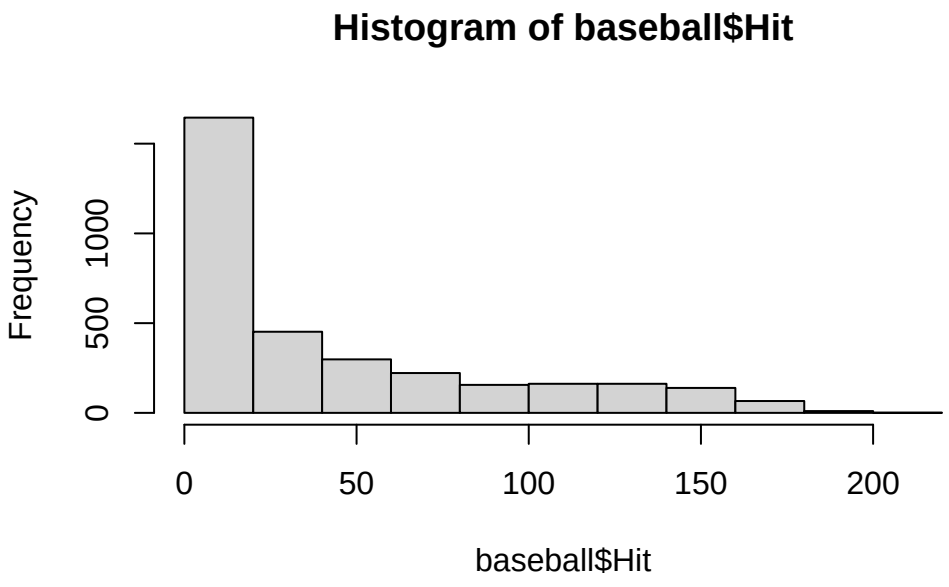
```
    Carp     DeNA  Dragons   Eagles Fighters   Giants    Lions    Lotte
     517      550      573      569      533      524      528      538
   Orix Softbank Swallows   Tigers
     579      525      556      554
```

- 一行目で, 同じ変数に「上書き」していることに注意
- as.factor 関数は変数を Factor 型に変換するものです
  - クラス名, 実験の水準などグループ化変数として扱うのに便利です

0.5.9 可視化してみましょう

- **データは図にする**のが基本です。R の基本関数でも十分綺麗な図が描けます。
  - ヒットの数をヒストグラムにしてみましょう
  - ヒストグラムの関数は hist です

```
hist(baseball$Hit)
```

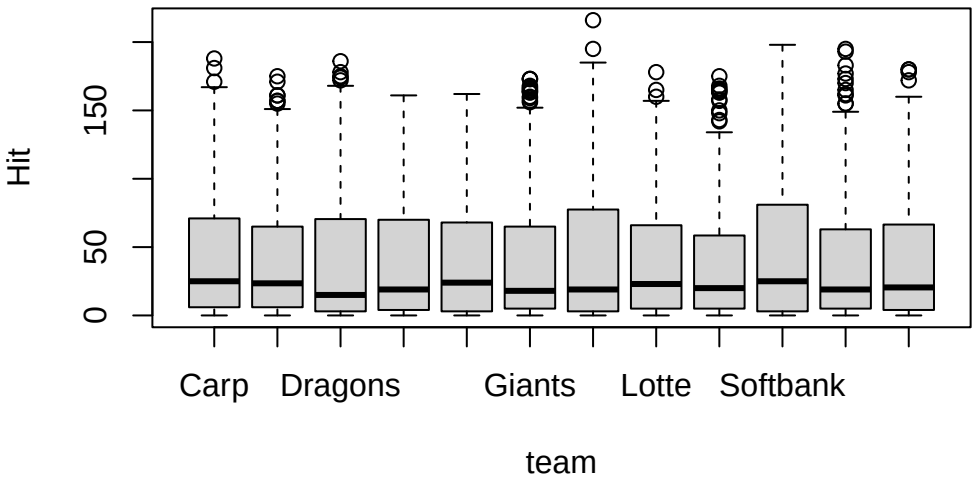


0.5.10 可視化してみましょう

- **データは図にする**のが基本です。

  - チーム毎のヒット数の違いを見てみましょう
  - ボックスプロット (箱ひげ図) の関数は boxplot です
    * x 軸が Factor 型になっています

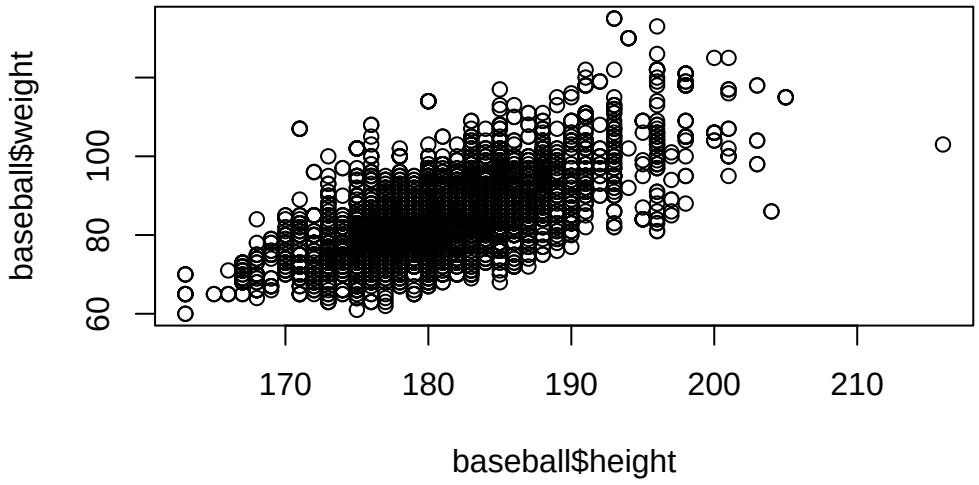
```
boxplot(Hit ~ team, data = baseball)
```



0.5.11 可視化してみましょう

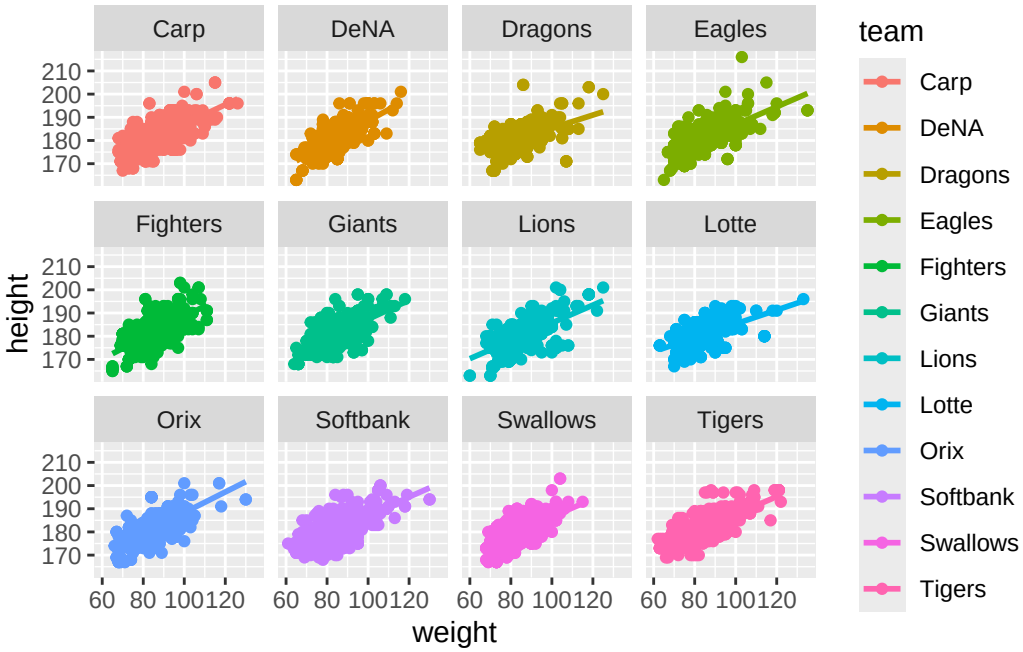
- **データは図にする**のが基本です。
  - 散布図を書いてみましょう
  - 散布図は plot 関数に x 軸と y 軸変数を指定します

```
plot(baseball$height, baseball$weight)
```



0.5.12 (余談)ggplot による出力は,より綺麗です

```
library(ggplot2)
baseball |>
    ggplot(aes(x = weight, y = height, color = team)) +
    geom_point() +
    geom_smooth(formula = "y ~ x", method = "lm", se = FALSE) +
    facet_wrap(~team)
```



0.6 Enjoy!

- 以上で入門コースは終了です
- この講義では `exametrika` パッケージの他は, 基本的に R の基本関数だけで分析できるようにご案内します。
- 実は `ggexametrika` パッケージというのもあります→こちら

0.7 Advanced Topics

0.7.1 Quarto/Rmarkdown：文芸的プログラミング

0.7.1.1 Rmarkdown とは

- Markdown 書式というプレーンな文書作成文法 + チャンクと呼ばれる R コードの結合
- 文書を作成 (レンダリング) するときは，R の計算を実行してその結果を文書内に反映させる
- コピペ汚染がなく，RStudio で執筆と分析が統合, これだけで完結できます。

0.7.1.2 Quarto とは

- 次世代の R Markdown です。私のスライドも Quarto で作られています
- マルチ言語対応(R, Python, Julia 等)
- ePub, PDF など出力も多様

0.7.2 Quarto/Rmarkdown：文芸的プログラミング

0.7.2.1 基本的な使い方

1. File → New File で Quarto Document / R Markdown を選択
2. Title や Author を入力, 出力書式 (HTML,PDF,Word) などを選ぶ画面が出ます
3. サンプル文書・コードが書いてあるファイルが生成されます

これを Rneder/Knit することでファイルが出力されます。