

データ処理・グラフ作成のための

# R/RStudioベーシック講座

【はじめてのプログラミング/DX】

# セクション1 はじめに

# コースの概要

## R/RStudioを通じたDXははじめの一步

1. はじめに	R/RStudioについて / 本コースの進め方について
2. RStudio概観	RStudioやデータ分析の全体像について
3. データの用意	既存データを読み込む方法 / 新たにデータを生成する方法
4. データの概観	変数とデータフレーム / 代表値 / 散布度
5. データのグラフ化	グラフ化の論点と整理 / ggplot2パッケージによるグラフ化
6. データの整備	tidy data / dplyrパッケージによるさまざまなデータ操作
7. データのグループ集計	データのグループ化 / グループ集計 / グループ集計の可視化

# なぜR/RStudioか

## R/RStudioは初めてでもとっつきやすいプログラミング言語



### 慣れ親しんだExcel「的」な面

- Rはデータ分析のための言語
- 表計算ソフトであるExcelと親和性



### 無料の優れた統合開発環境RStudio

- RとRStudioをインストールするだけ（難しい環境構築は不要）
- Excelで見慣れた表形式で手軽かつ視覚的にデータを確認できる



### プログラミングの「イメージ」をつかみやすい

- データを通じてどんな処理ができるか具体的にイメージできる
- プログラミングの「メリット」を体感できる

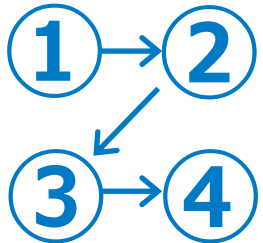
# Rとは

## Excelでのいくつかの悩みを解決できる「データ分析」ツール



### シンプルなファイル管理

- Excel はファイルやシートが乱立しがち...
- R は元データと R のファイルのみでシンプルにファイル管理できる



### 作業プロセスの記録化

- Excel は作業プロセスがブラックボックスになりがち...
- R は作業プロセスが記録されて再現性が担保される



### スピーディな処理

- Excel は複雑で大量な処理で固まりやすい...
- R は複雑で大量な処理でもスピーディに実行できる

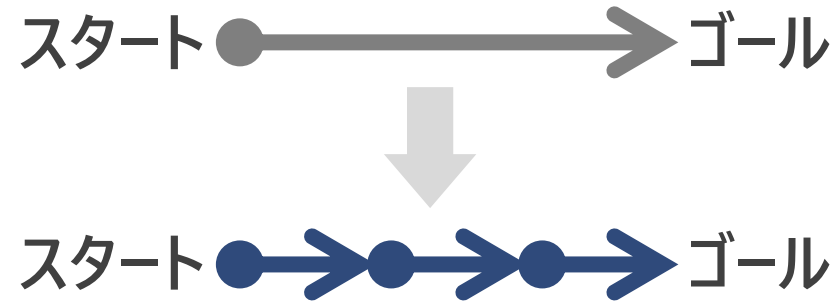
# どうR/Rstudioを学ぶか

ひとつひとつ、くりかえし、復習しながら学んでいく

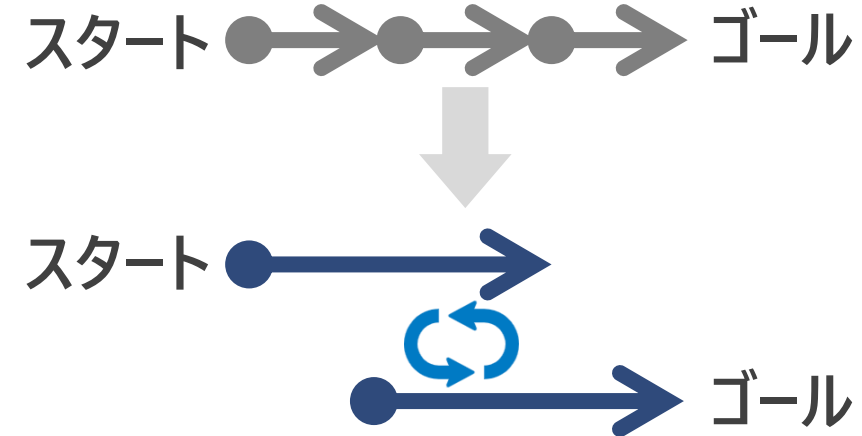
理解→実践



ひとつひとつ順を追って



復習しながら進んでいく



知り尽くす、網羅する

よりも...

豊富な資料と反復で

理解する、繰り返して慣れる

# セクション2

## RStudio概観

# データ分析の全体像

## データ分析プロセスは「お料理」プロセス

材料用意

材料確認

仕込み

調理

盛り付け



データ用意

データ概観

データ整然化

データ解析

資料化

✓ 既存データ

✓ 新規データ

✓ 記述統計

✓ グラフ化

✓ 整然データ

“tidy data”

✓ 集計

✓ 統計的分析

✓ 機械学習

✓ グラフデザイン

✓ プレゼン資料

✓ レポート



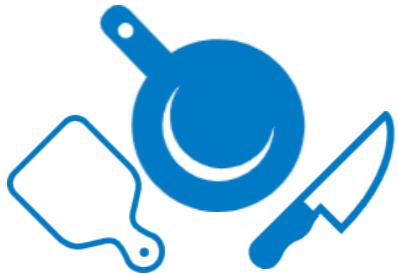
# R/RStudioとは

## RStudioという「キッチン」でR言語というツールを使う



### RStudioはRを動かすための「キッチン」

- RStudioをひらいて、R言語を動かす（統合開発環境）
- （Excelの場合） Excelをひらいて、Excelを動かす（区別がない）



### R言語の「基本ツール」：基本パッケージ

- 基本的なデータ操作をシンプルな関数で実行
- 関数：ExcelのSUM関数やAVERAGE関数などと同様のもの



### R言語の「オプションツール」：無数のオプションパッケージ

- 目的や機能に応じて無数のパッケージが開発・公開されている
- Excelでいうと「分析ツール」や「ソルバー」のイメージ

# 関数とパッケージ

さまざまな関数が開発されパッケージとして公開されている

## 基本パッケージ

デフォルトで使える関数群

- ファイル入出力
- 基本演算
- 確率分布・乱数
- モデリング・分析
- プロット（グラフ）
- データフレーム操作

その他、たくさん!!

## オプションパッケージ

別途インストールして使える関数群

### ● tidyverseパッケージ

#### ● readr/

- 様々なファイル入出力

#### ● dplyrパッケージ

- 様々なデータ操作  
その他、たくさん!!

tidyverse以外にもたくさん!!

## 使い方

2 STEP

① インストール

```
install.packages("パッケージ名")
```

② 呼び出し

```
library(パッケージ名)
```

# RStudioの画面

4つの枠に異なる窓（Pane）が表示されている

The screenshot shows the RStudio interface with four panes highlighted and labeled:

- Source**: ソースコードの記述 (Source code description). The pane shows R code for setting the working directory.
- Environment**: 読み込み・作成したデータの表示 (Display of loaded/created data). The pane shows the Environment pane with a list of objects.
- Console**: コードの実行 (Code execution). The pane shows the R console with the prompt and the output of the code.
- Plots, Help**: 図の表示 (Figure display) and ヘルプの表示 (Help display). The pane shows a bar chart of average annual income by age group and a list of programming languages.

Additional labels and details:

- The **Source** pane also includes a label: **ソースコードの記述** (Source code description).
- The **Environment** pane also includes a label: **読み込み・作成したデータの表示** (Display of loaded/created data).
- The **Plots, Help** pane includes a label: **図の表示** (Figure display) and **ヘルプの表示** (Help display).
- The **Console** pane also includes a label: **コードの実行** (Code execution).
- The **Plots, Help** pane also includes a label: **?関数名** (Function name).

# データの用意

## データを用意する方法は大きく2パターン

### 既存データを読み込む

ファイル形式によって明示的に読み込む



- readrパッケージ

- csvファイルの読み込み

```
read_csv(“ファイル名”)
```



- readxlパッケージ

- Excelファイルの読み込み

```
read_excel(“ファイル名”)
```

### 新しくデータを生成する

直接入力や乱数（シミュレーション）

- 値(変数)を直接入力

```
c(値1, 値2, 値3,...)
```

※c: combine values

※各セルに値を入力するイメージ

- 乱数（例：正規分布）

```
rnorm(サンプルサイズ,  
      平均, 標準偏差)
```

# データの概観

## データの概要を2つの視点から確認

### 変数とデータフレーム

	データフレーム	B
1	身長	体重
2	変数 94	変数 8
3	168.90	65.86
4	185.08	82.31
5	171.29	65.19
6	169.76	74.36
7	170.12	72.80

### 代表値（変数の代表）

# 代表値を返す関数(変数)

`mean(変数)` #平均

`median(変数)` #中央値

### 散布度（変数のばらつき）

# 散布度を返す関数(変数)

`var(変数)` #分散(variance)

`sd(変数)` #標準偏差

(standard deviation)

# データのグラフ化

## データの「分布」をggplot2パッケージでグラフ化

### 基本パッケージ

```
# グラフ関数(変数)  
plot(変数)  
  #棒グラフ/散布図...  
hist(変数)  
  #ヒストグラム  
boxplot(変数)  
  #箱ひげ図
```

### オプションパッケージ

#### ● ggplot2パッケージ

```
ggplot(データ, マッピング) #グラフの下準備
```

+

#### # 各グラフ関数

```
geom_bar(調整値) #棒グラフ  
geom_histogram(同上) #ヒストグラム  
geom_boxplot(同上) #箱ひげ図  
geom_point(同上) #散布図
```

# データの整然化

データを扱いやすい構造にする = 「整然化」する

ハドリー・ウィッカム  
(Hadley Wickham)

「雑然」としたデータ：messy data

	A	B	C	D
1	STORE	APPLE	ORANGE	BANANA
2	A	3	2	2
3	B	5	2	2
4	C	5	6	5
5	D			
6	E			
7	F			
8	G			
9	H	8	3	1
10	I	6	0	1

横長

- 見た目のわかりやすさ◎

- 「変数」が雑然としている

- 分析対象は「変数」...

「整然」としたデータ：tidy data

	A	B	C
1	STORE	FRUITS	SALES
2	A	APPLE	3
3	A		0
4	A		2
5	A	GRAPE	1
6	B	APPLE	5
7	B		0
8	B	ORANGE	1
9	C	ORANGE	0
10	C	BANANA	5
11	C	GRAPE	0
12	C	BANANA	5
13	C	GRAPE	0

縦長

- 個々の“観測”が1つの行をなす

- 個々の“変数”が1つの列をなす

- 個々の“観測パターン”が1つの表をなす

# データの集計・解析・レポート化

## データの集計・解析やRmarkdownという機能によるレポート化

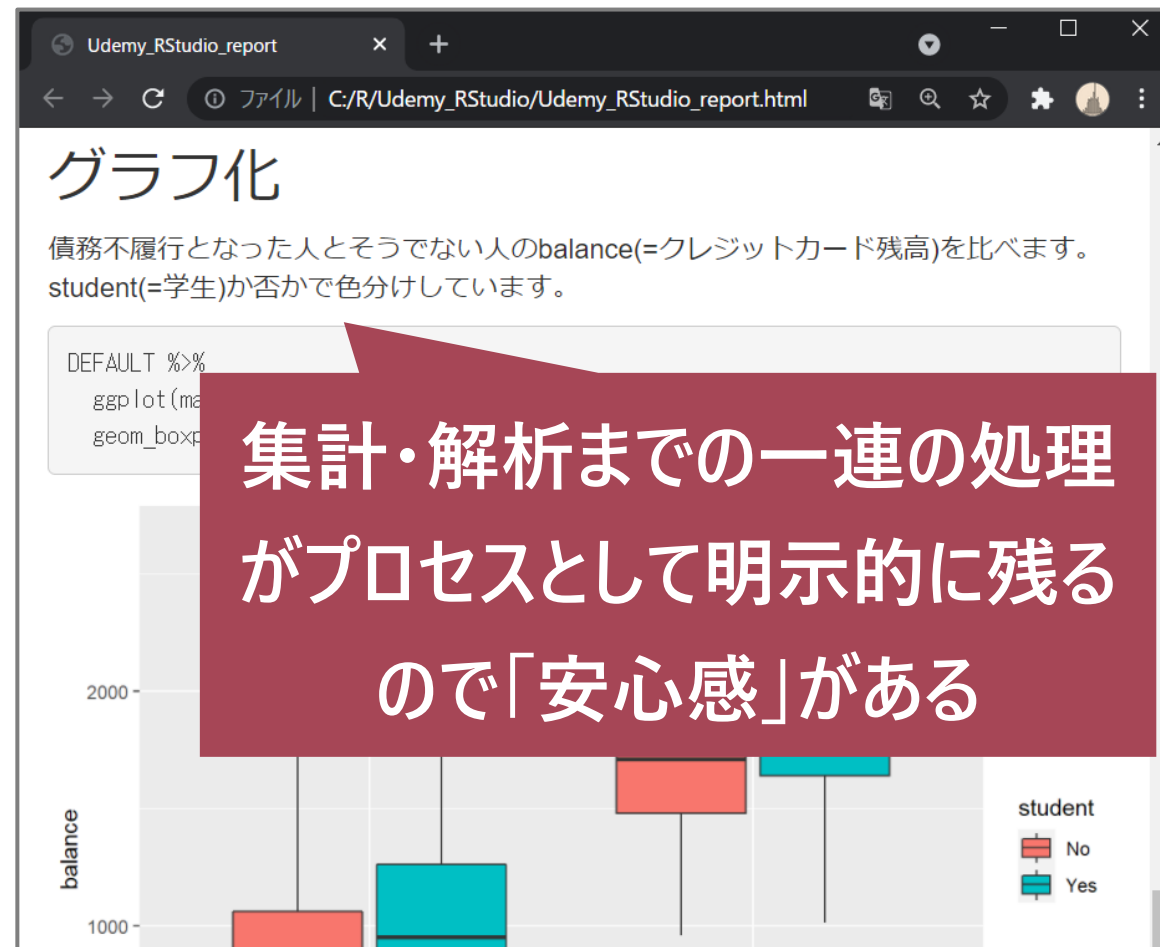
### グループ別の集計

- 「年代別」「部門別」の「残業時間」
- 「月別」の「残業●時間以上の割合」
- 「20XX年」の「部門別」の「離職率」

### 統計的な解析

- 「年齢」と「残業時間」の相関分析
- 「残業時間」を目的変数とする回帰分析
- 「離職有無」を目的変数とする分類分析
- 「アンケート回答結果」によるクラスタリング

### レポート化





# セクション3 データを用意

# tidyverseパッケージ

## 整然化データのための優れたパッケージ群

### ● tidyverseパッケージ

#### ● readrパッケージ

- 様々なファイル入出力

#### ● readxlパッケージ

- Excelファイル入出力

#### ● dplyrパッケージ

- 様々なデータ操作

#### ● tidyrパッケージ

- tidy dataに変形する

#### ● ggplot2パッケージ

- 様々なグラフ作図

#### ● stringrパッケージ

- 文字列の操作

#### ● lubridateパッケージ

- 日付型データの操作

その他にも様々な  
パッケージを内包

# ディレクトリの指定

はじめに作業するディレクトリを指定する

## ✓ ディレクトリとは？



```
> PC > ローカル ディスク (C:) > R > test
```

➤ 東京都千代田区丸の内 ● 丁目 ● 番...

## ✓ ディレクトリの設定と確認

設定



```
setwd("C:/R/test")
```

確認



```
getwd()
```

※ wd:working directory



の中に居場所が  
セットされる

# データの前提

コンピュータが読み込みやすいデータとなっているか

1行目 1列目スタート

	A	B	C
1			
2		身長	体重
3		178.94	84.28
4		168.90	65.86
5		185.08	82.31
6		171.29	65.19
7		169.76	74.36
8		170.13	72.80
9		164.99	62.61
10		173.97	73.98

1行目は列名(変数名)

各変数(列)の変数名		
1	身長	体重
2	178.94	84.28
3	168.90	65.86
4	185.08	82.31
5	171.29	65.19
6	169.76	74.36
7	170.13	72.80
8	164.99	62.61
9	173.97	73.98

余計なデータはNG

	B	C	D
	体重		メモ
94	84.28		山田編集
90	65.86		2021年7月
08	82.31		※取扱注意
29	65.19		
76	74.36		
13	72.80		
99	62.61		
17	73.98		

分析対象外の  
データはNG

# CSVファイルの読み込み

readrパッケージを呼び出し、read\_csv関数で読み込む

testdata.csv



	A	B
1	身長	体重
2	178.94	84.28
3	168.90	65.86
4	185.08	82.31
5	171.29	65.19
6	169.76	74.36
7	170.13	72.80
8	164.99	62.61

```
read.csv(file = “ファイル名.csv”)
```

→ 「普通」に読み込む...

readrパッケージ

- パッケージのインストールと呼び出し

```
# パッケージのインストールと読み込み
```

readrパッケージも含まれる

```
install.packages(“tidyverse”)
```

```
library(readr)
```

```
library(tidyverse)
```

readrパッケージ

```
read_csv(file = “ファイル名.csv”)
```

# エンコーディングの指定

## read\_csv関数のなかでlocale引数でエンコーディング指定

エンコーディングの確認

✓ データの符号化の方式 (UTF-8 / Shift\_JIS(系))

readrパッケージ

```
guess_encoding(file = “ファイル名.拡張子”)
```

エンコーディングを指定して読み込み

readrパッケージ

```
read_csv(file = “ファイル名.csv”,  
         locale = locale(encoding = “UTF-8”))
```

※関数という道具を「どう使うか」を引数で指定

# 読み込みデータの格納

Rでは"<-"という代入記号でデータをオブジェクトに格納する

エンコーディングを指定して読み込み

readrパッケージ

```
read_csv(file = "ファイル名.csv",  
          locale = locale(encoding = "UTF-8"))
```

エンコーディングを指定して読み込み、オブジェクト(箱)に格納

readrパッケージ

```
HAKO  <- read_csv(file = "testdata.csv",  
               locale = locale(encoding = "UTF-8"))
```



# Excelファイルの読み込み

Excelファイルはreadxlパッケージのread\_excel関数で読み込める


	A	B
1	身長	体重
2	178.94	84.28
3	168.90	65.86
4	185.08	82.31
5	171.29	65.19
6	169.76	74.36
7	170.13	72.80
8	164.99	62.61
9	172.07	72.00

パッケージのインストールと呼び出し

```
# パッケージのインストールと読み込み
install.packages("tidyverse")
library(readxl)
```

readxlパッケージも含まれる

readxlパッケージ

```
HAKO <- read_excel("testdata.xlsx",
   sheet = 1)
```

sheet引数でsheetを指定(番号 or "シート名")



# サンプリングデータの生成

sample関数で値(x)とサンプルサイズ(size)を指定して生成

```
sample <- sample(x = 値, size = サイズ, replace = TRUE/FALSE)
```

	A
1	SAMPLE
2	男性
3	女性
4	女性
5	男性
6	男性
7	

複数の値を(ベクトルに)まとめる

```
c(1, 2, 3, 4, 5) = c(1:5)
```

```
c("男性", "女性", "その他")
```

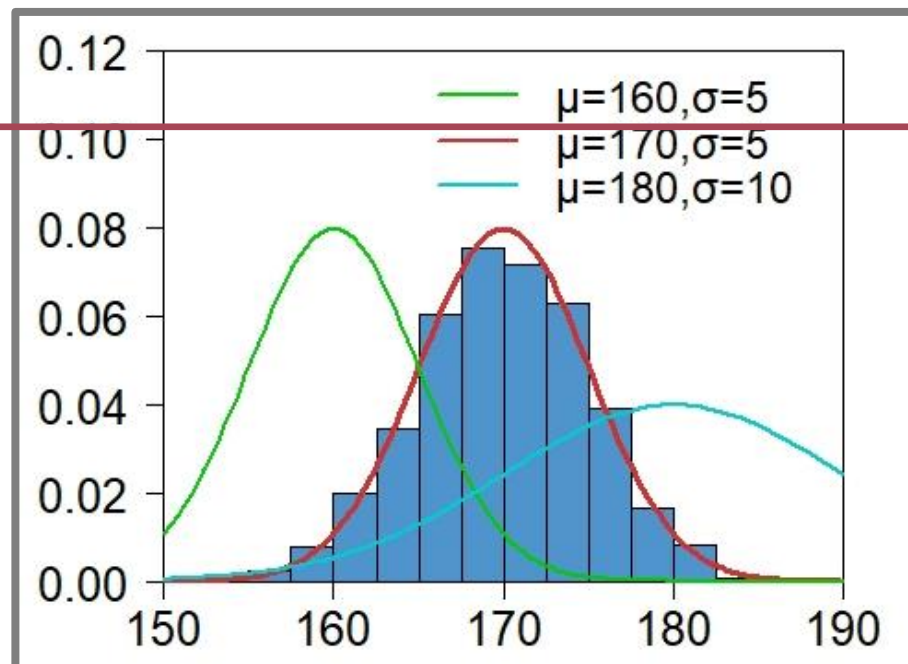
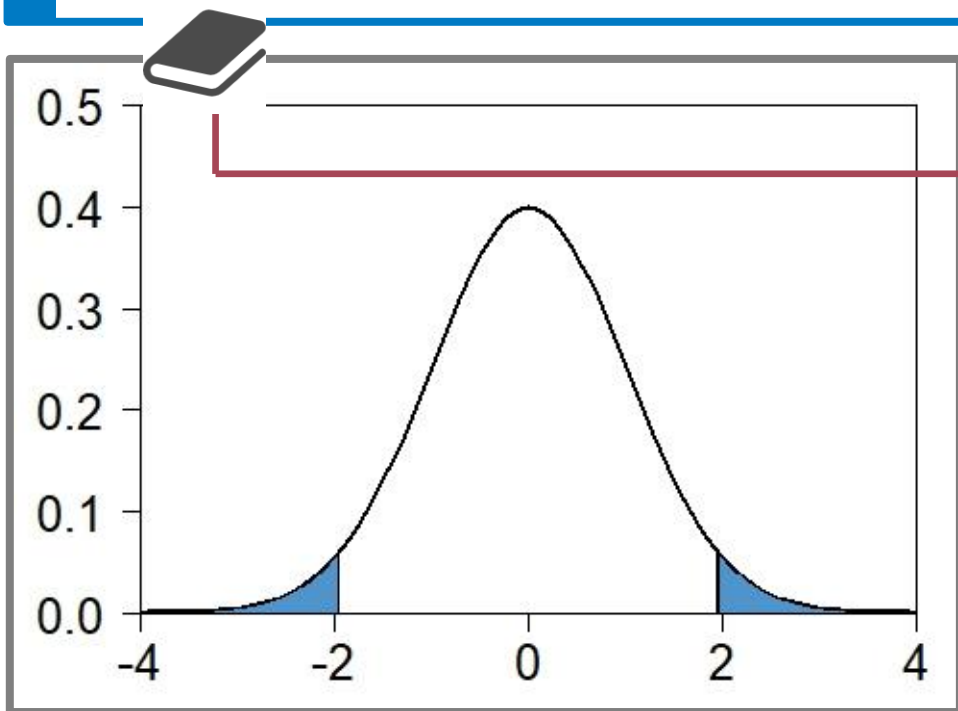
TRUE : 重複OK  
FALSE : 重複NG

```
sample <- sample(x = c("Yes", "No"),  
size = 10, replace = TRUE)
```

# 正規分布データの生成

`rnorm`関数でサンプルサイズ( $n$ )とパラメータ(mean, sd)を指定して生成

```
normal <- rnorm(n = サイズ, mean = 平均, sd = 標準偏差 )
```



	A
1	normal
2	178.94
3	168.90
4	185.08
5	171.29
6	169.76

```
shincho <- rnorm(n = 100, mean = 167, sd = 6)
```

# CSVファイルの書き出し

write\_csv関数でデータフレームとファイル名を指定して書き出し

readrパッケージ

```
write_csv(x = データフレーム, file = “ファイル名.csv”)
```

## 復習

### ✓ ディレクトリの設定と確認

設定

```
setwd(“C:/R/test”)
```

確認

```
getwd()
```

※ wd:working directory



の中に居場所が  
セットされる

居場所のフォルダ内に保存される

```
write_csv(x = SHINCHO, file = “SHINCHO.csv”)
```

# セクション4

## データの概観

# サンプルデータの紹介

## ISLRパッケージのクレジットカード顧客の債務不履行データ

### ● ISLRパッケージ

```
install.packages("ISLR")  
library(ISLR)  
DEFAULT <- as_tibble(Default)  
#tibbleというデータ形式で格納
```

### ● tidyrパッケージ

< Defaultデータ >

- ✓ クレジットカード顧客データ
- ✓ 10000サンプル(行)
- ✓ 4変数(列)

	default	student	balance	income
1	債務不履行 (Yes/No)	学生 (Yes/No)	債務残高 (実数)	収入 (実数)
2	No	Yes	817.18041	12106.135

# データフレーム

行と列からなるデータのあつまり = Excelで見慣れた「表」

## サンプル

- ひとりひとり、  
ひとつひとつ  
= 構成単位

## 行

	身長	体重	性別
	178.94	84.28	男性
3	168.90	65.86	男性
4	185.08	92.31	男性
5	171.29	85.12	男性
6	169.76	74.36	男性

## 列

## 変数・属性

- ひとりひとりのサンプルを特徴づける
- 異なる変数は異なる列に格納

# 変数

変数はデータフレームにおける「列」で、それぞれに名前がある

	A	B	C
1	身長	体重	性別
2	178.94	84.28	男性
3	168.90	65.86	男性
4	185.08	82.31	男性
5	171.29	65.19	男性
6	169.76	74.36	男性
7	170.13	72.80	男性
8	164.99	62.61	男性
9	173.97	73.98	男性

データフレームの列名を確認する

データフレーム 

`colnames(データフレーム)`

`str(データフレーム)`

データフレームの列名を変更する

`colnames(データフレーム) <- 列名ベクトル`

dplyrパッケージ

`rename(データフレーム, 新名前 = 旧名前)`

`rename(HAKO, height = 身長)`

# 変数の型

変数に含まれるそれぞれの値は「同じ型」である必要

変数には型があり、データ操作に大きな影響を与える

データ型	値の例
数値型 : numeric	1, -1, 1.1, 1.2...
整数型 : integer<int>	1, -1, 100, 1000...
実数型 : double<dbl>	168.5, 170.2, 166.9...
文字列型 : character<chr>	“営業部”, “人事部”, “財務部”, ...
因子型 : factor<fct>	営業部, 人事部, 財務部...
論理値型 : logical<lgl>	TRUE(1), FALSE(0)...
日付型 : date<date>	2021-01-01, 2021-01-02...



# 変数型の変換

データを読み込む際や読み込んだ後に型を変換できる

データ型

readrパッケージ

数値型 : numeric

整数型 : integer

実数型 : double

文字列型 : character

因子型 : factor

論理値型 : logical

日付型 : Date

HAKO <- read\_csv(“ファイル名.csv”,  
col\_types = “idcflD”)

データフレーム\$列名 <- as.XXX(データフレーム\$列名)

[データフレーム+\$+列名]で  
データフレームの列を指定

as.numeric  
as.integer  
as.double...

HAKO\$age <- as.character(HAKO\$age)

# 基本演算子

Excel同様の記号で基本的な演算等を行える

記号	意味	使用例
^	べき乗	2^2, 2^5, 2+10^2
:	±1の等差数列	1:10, 51:100, 10:-10, 5+10・20
*, /	掛け算、割り算	1*2, 2*4/8, 3-2/6
+, -	足し算、引き算	1+2, 3+5.5, 3-1/2
<, <=	大小比較	0<1, 2>=4, 1+1+1>2
==, !=	等しい、等しくない	1==1, 2!=2.1, “a”==“a”
&,	かつ、または	1>0&2>3, 1>0 2>3

値が返る

TRUE(1)  
FALSE(0)

# 行数・列数

データが何行何列のデータなのかを確認する

データフレームの行数・列数を確認する

データフレーム 

`dim(データフレーム)`

※ `dim:dimension(次元)`

`length(データフレーム$列名)`

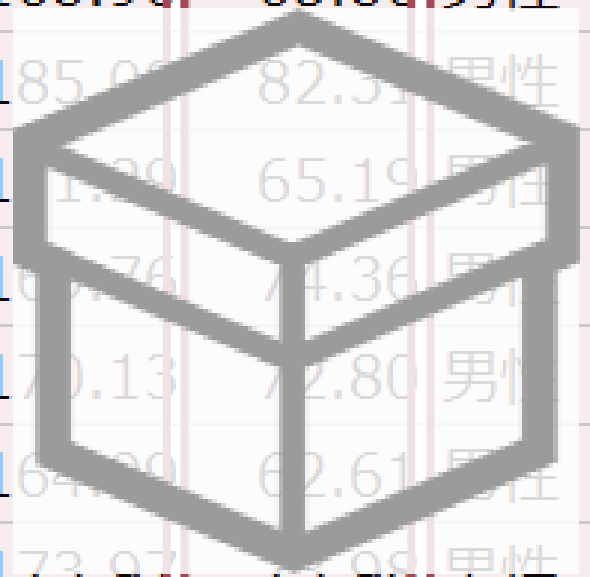
```
Console Terminal x Jobs x
R 4.1.0 · C:/R/Udemy_RStudio/
> dim(DEFAULT)
[1] 10000      4
> length(DEFAULT$default)
[1] 10000
```

```
Console Terminal x
R 4.1.0 · C:/R/Udemy_RStudio/
> Default
default
1      No
2      No
3      No
4      No
5      No
6      No
7      No
8      No
9      No
10     No
11     No

Console Terminal x Jobs x
R 4.1.0 · C:/R/Udemy_RStudio/
> DEFAULT
# A tibble: 10,000 x 4
  default student balance income
  <fct>    <fct>      <dbl>  <dbl>
1 No      No        730.   44362.
2 No      Yes       817.   12106.
3 No      No       1074.   31767.
4 No      No        529.   35704.
5 No      No        786.   38463.
6 No      Yes        920.    7492.
7 No      No        826.   24905.
8 No      Yes        809.   17600.
9 No      No       1161.   37469.
10 No     No          0    29275.
# ... with 9,990 more rows
```

# 変数の代表値

その変数(列)を代表する値を確認する



	身長	体重	性別
1	178.94	84.28	男性
2	168.90	65.86	男性
3	185.00	82.31	男性
4	171.20	65.19	男性
5	167.76	74.36	男性
6	170.13	72.80	男性
7	164.00	62.61	男性
8	173.97	73.98	男性

変数(列)の代表値を確認する

```
mean(データフレーム$列名) #平均  
median(データフレーム$列名) #中央値  
min(データフレーム$列名) #最小値  
max(データフレーム$列名) #最大値  
sum(データフレーム$列名) #合計  
※table(データフレーム$列名) #集計
```

変数(列)の要約を確認する

```
summary(データフレーム) #各変数の要約
```

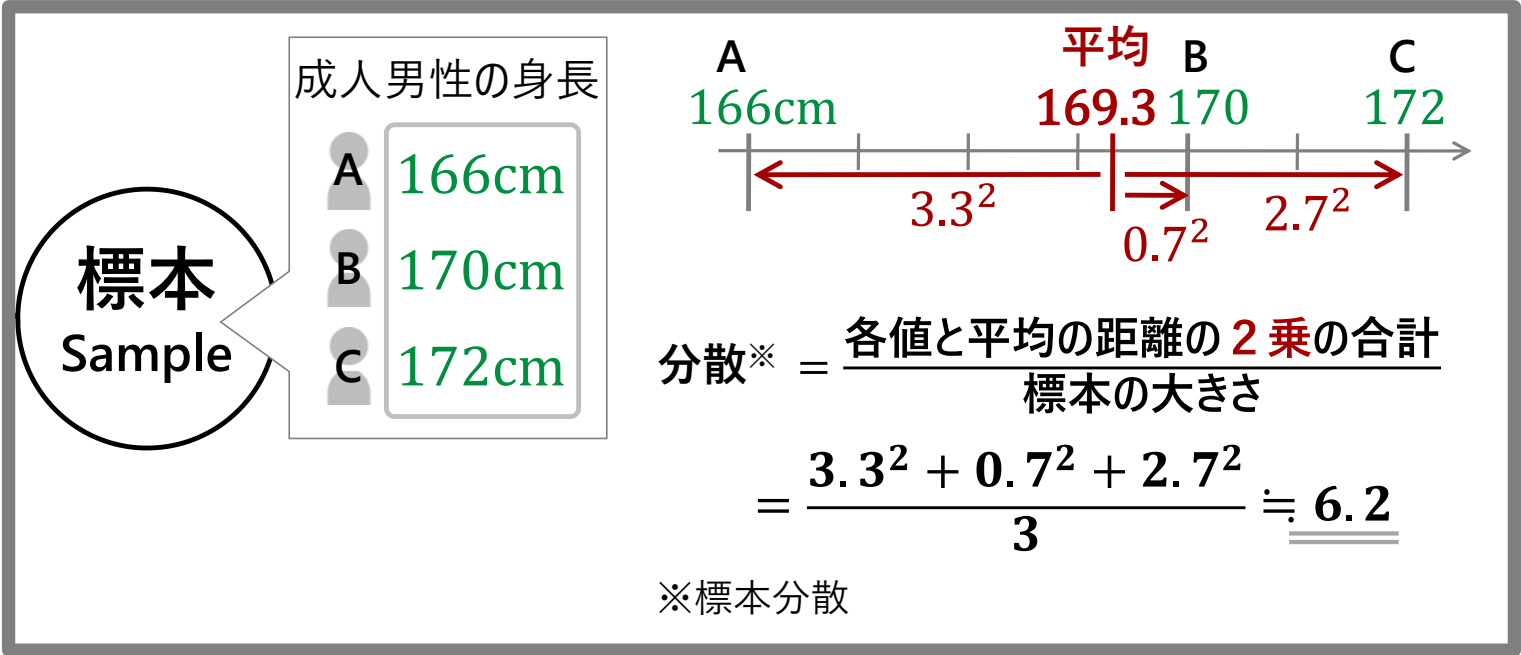
# 変数の散布度

その変数(列)の散布度(ばらつき度合い)を確認する

	身長	体重	性別
1			
2	178.94	84.28	男性
3	168.90	65.86	男性
4	185.0	82.31	男性
5	171.29	65.19	男性
6	169.76	74.36	男性
7	170.13	72.80	男性
8	164.29	62.61	男性
9	173.97	75.96	男性

変数(列)の散布度を確認する

```
var(データフレーム$列名) #不偏分散  
sd(データフレーム$列名) #標準偏差
```



# 変数の演算

変数と変数の演算は各々の変数の値の数（サンプルサイズ）に注意

記号	
^	値が返る
:	
*, /	
+, -	
<, <=	TRUE(1) FALSE(0)
==, !=	
&,	

## 1. 変数と1つの値の演算

- ✓ 変数のすべての値に1つの値が演算される

## 2. 変数と変数の演算（値の数が等しい場合）

- ✓ 順番通りに演算される

## 3. 変数と変数の演算（値の数が異なる場合）

- ✓ 順番通りに演算され、値の数が少ない変数は繰り返されて演算される

# セクション5 データのグラフ化

# (補足) 変数のタイプの整理

## 数量を表す量的変数(連続/離散変数)とカテゴリを表すカテゴリ変数

量的変数 ※数量を表す	連続変数	変数のとりうる値が連続的 (例) 長さ、重さ、時間、(近似的に)金額...
	離散変数	変数のとりうる値が離散的 (例) 件数、個数、年齢、サイコロの目...
カテゴリ変数(質的変数) ※数量を表すものでない		カテゴリ変数(質的変数) (例) 年代、性別、所属、Yes/No...

以降のスライドやレクチャーでは、シンプルな整理や簡潔な説明を優先し、上記の「離散変数」と「カテゴリ変数」を一括りにしてご説明しておりますが、厳密な変数タイプの整理は上記の通りとなります。混乱させてしまう部分がありましたら申し訳ありません。どうぞよろしくお願いいたします。



# グラフ化の論点

グラフ化の論点は「変数タイプ」と「軸」と「グループ」の3つ

## 変数タイプ

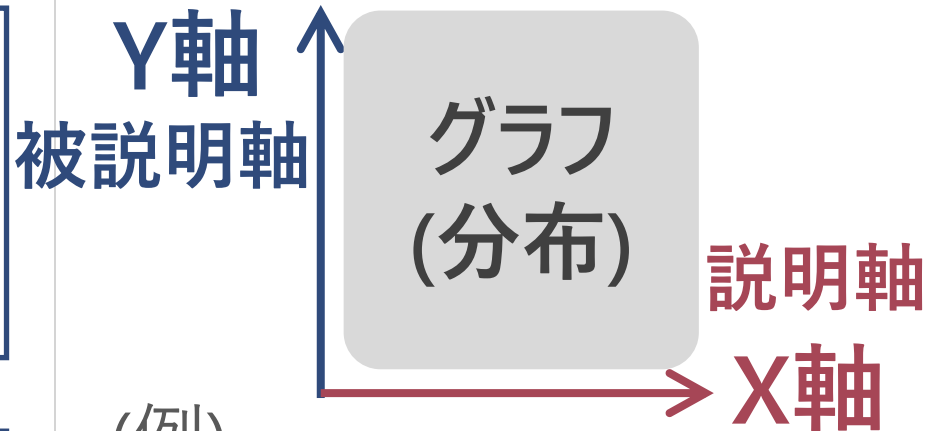
### 連続変数

- 連続数値データ
- 例: 基本給、残業時間、残業代...

### カテゴリ/離散変数

- カテゴリ「的」データ
- 例: 年代、性別、所属、評価...

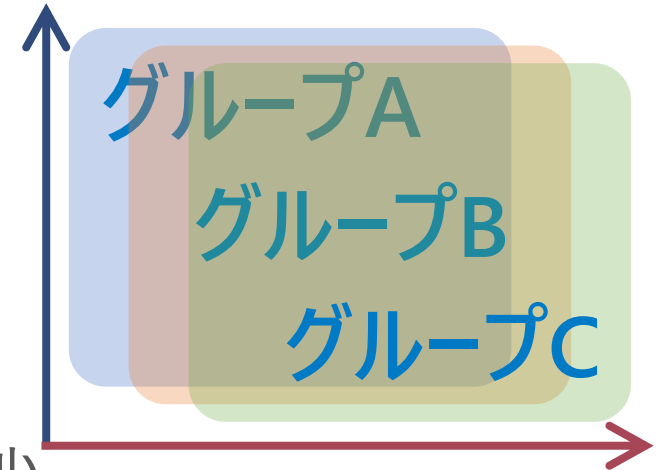
## グラフの軸



(例)

- 年代で給与を説明
- 所属で残業時間を説明
- 残業時間で評価を説明

## グループ



(例)

- 残業時間で評価を説明するグラフを所属部門別に可視化

# サンプルデータの紹介（再掲）

## ISLRパッケージのクレジットカード顧客の債務不履行データ

### ● ISLRパッケージ

```
install.packages("ISLR")  
library(ISLR)  
DEFAULT <- as_tibble(Default)  
#tibbleというデータ形式で格納
```

### ● tidyパッケージ

< Defaultデータ >

- ✓ クレジットカード顧客データ
- ✓ 10000サンプル(行)
- ✓ 4変数(列)

	default	student	balance	income
1	債務不履行 (Yes/No)	学生 (Yes/No)	債務残高 (実数)	収入 (実数)
2	No	Yes	817.18041	12106.135

# グラフ化の整理

主にどのグラフを確認するかX軸とY軸で整理すると考えやすい

Y軸 \ X軸	カテゴリ/離散変数	連続変数
カウント数	<b>棒グラフ&lt;bar&gt;</b> X:default Y:カウント数	<b>ヒストグラム&lt;histogram&gt;</b> X:balance Y:カウント数
カテゴリ/ 離散変数	<b>散布図&lt;point&gt;</b> X:student Y:default      X:balance Y:default	
連続変数	<b>箱ひげ図&lt;boxplot&gt;</b> X:default Y:balance	<b>散布図&lt;point&gt;</b> X:income Y:balance

# ggplot2パッケージ

## グラフの下地に図形をレイヤーとして重ねるイメージ

### 基本パッケージ

```
# グラフ関数(変数)
plot(変数)
  #棒グラフ/散布図...
hist(変数)
  #ヒストグラム
boxplot(変数)
  #箱ひげ図
```

### オプションパッケージ

#### ● ggplot2パッケージ

```
ggplot() #グラフの下地用意
```

+

```
# グラフ関数(データ, マッピング)
geom_bar(データ, マッピング) #棒グラフ
geom_histogram(同上) #ヒストグラム
geom_boxplot(同上) #箱ひげ図
geom_point(同上) #散布図
```

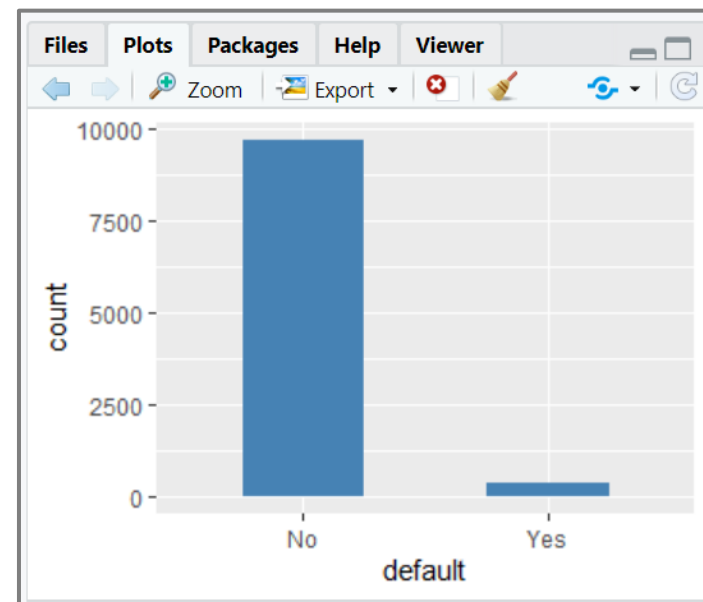
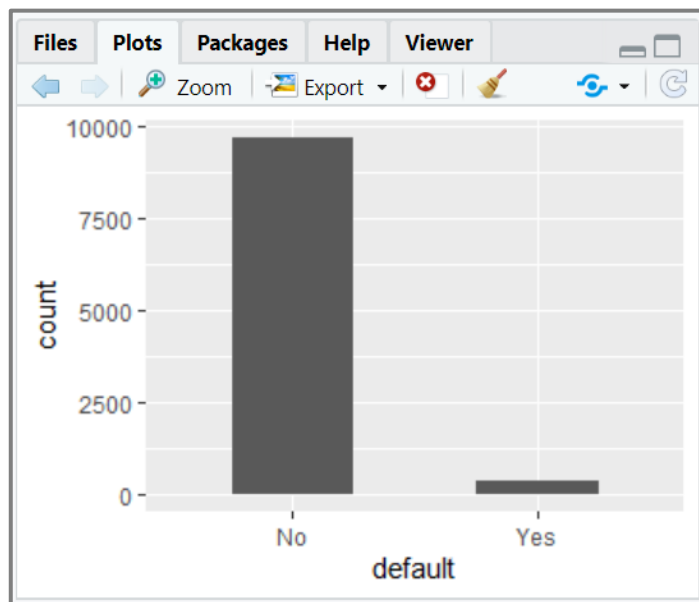
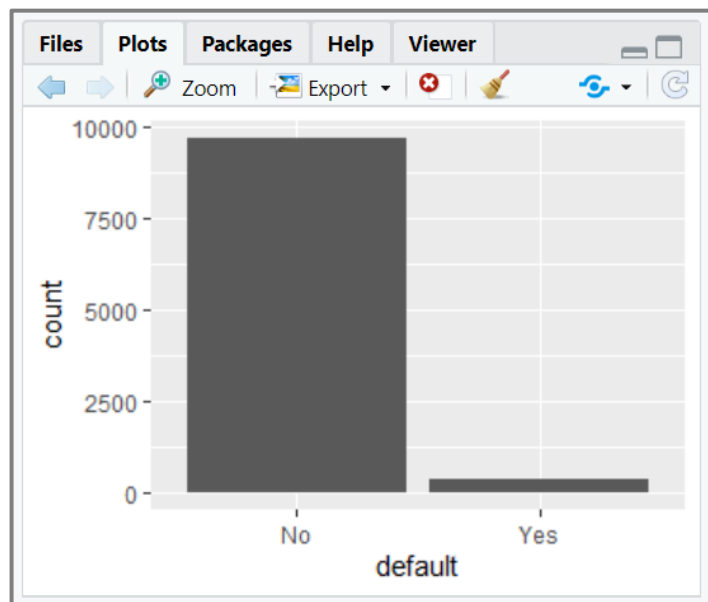
# 棒グラフ

## 離散変数の値(カテゴリ)別のカウント数を可視化する

- ggplot2パッケージ



```
ggplot(data = DEFAULT, mapping = aes(x = default)) + #グラフの下地用意  
geom_bar(width = 0.5, fill = "steelblue") #幅と塗色
```



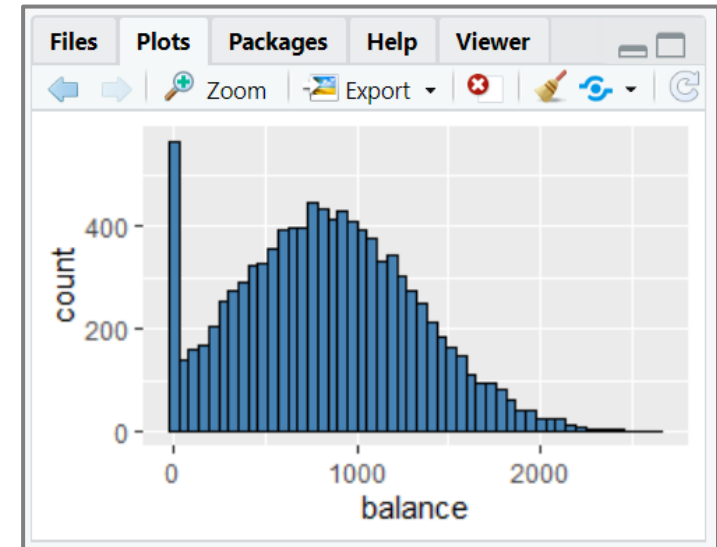
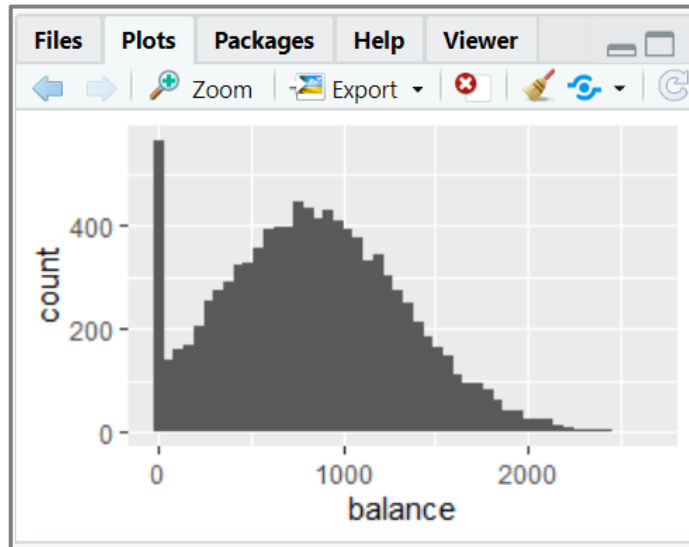
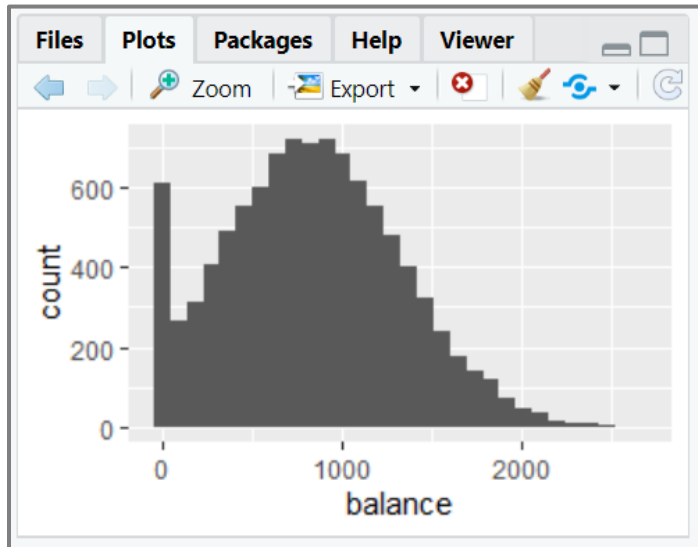
# ヒストグラム

## 連続変数の値別のカウント数を確認する

- ggplot2パッケージ



```
ggplot(data = DEFAULT,
       mapping = aes(x = balance)) +  
  geom_histogram(bins = 50, #棒の数  
                fill = "steelblue", colour = "black") #塗色と枠線色
```



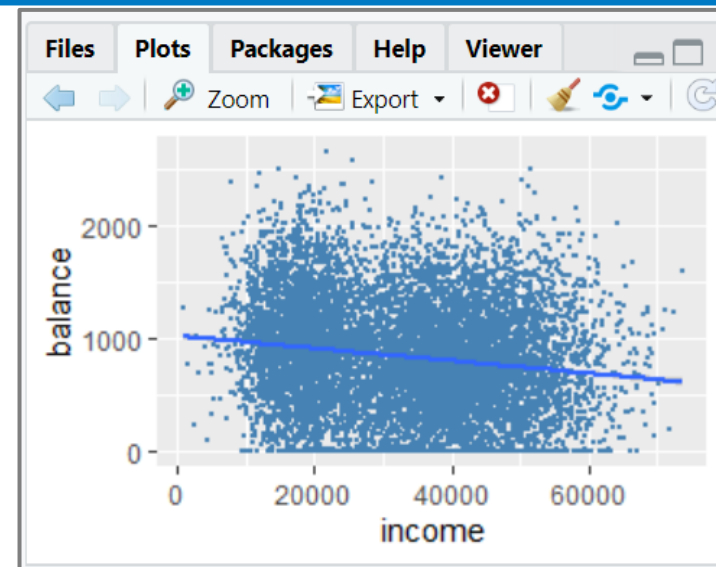
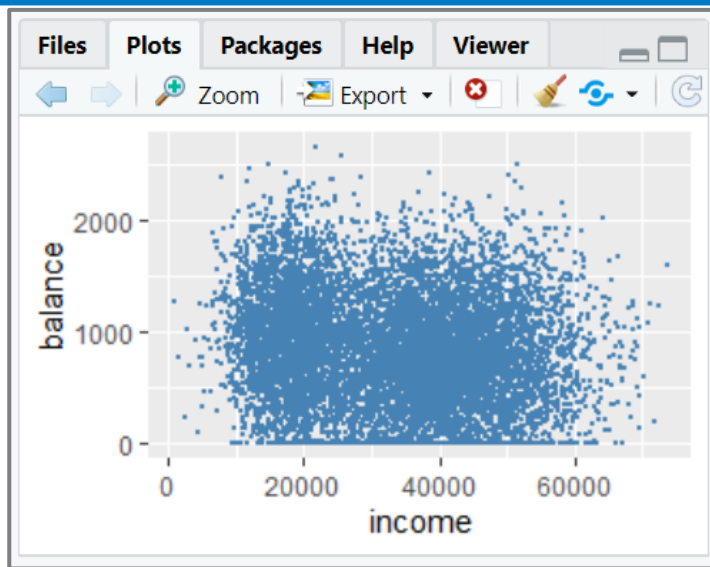
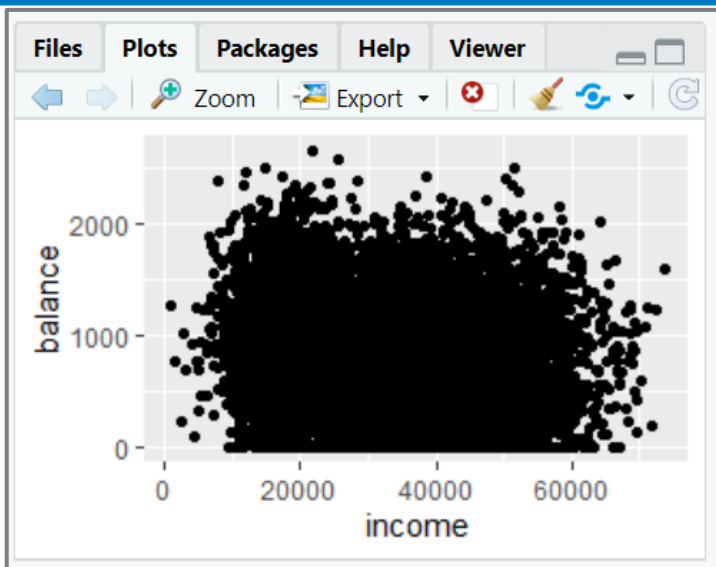
# 散布図①

## 2つの連続変数の分布を同時に確認する

- ggplot2パッケージ



```
ggplot(data = DEFAULT, mapping = aes(x = income, y = balance)) +  
  geom_point(colour="steelblue", size=0.5) + #点の色と大きさ  
  geom_smooth(method = "lm") #近似線(lm:linear model)
```



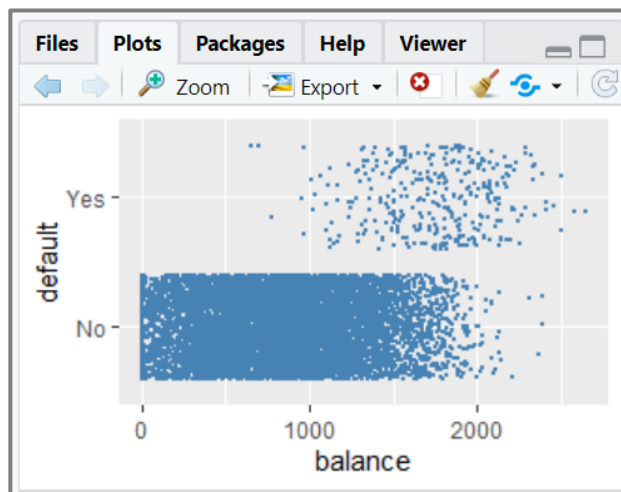
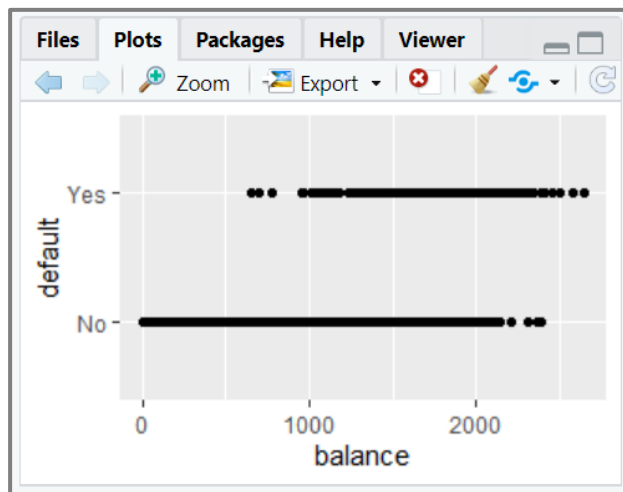
## 散布図②

### 連続変数と離散変数の分布を同時に確認する

- ggplot2パッケージ



```
ggplot(data = DEFAULT,
       mapping = aes(x = balance, y = default)) +
  geom_point(colour = "steelblue", size = 0.5,
            position = position_jitter(height = 0.3))
#position:配置の指定, jitter:ランダム配置, height:高さ
```





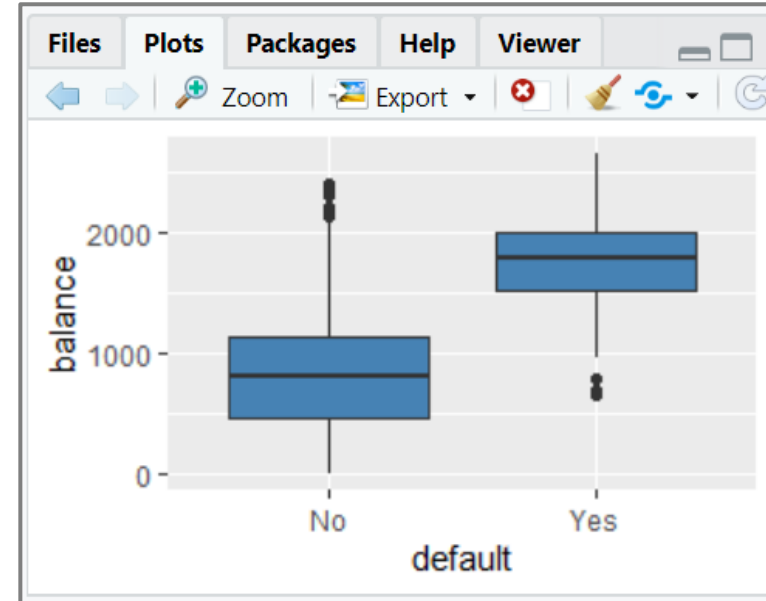
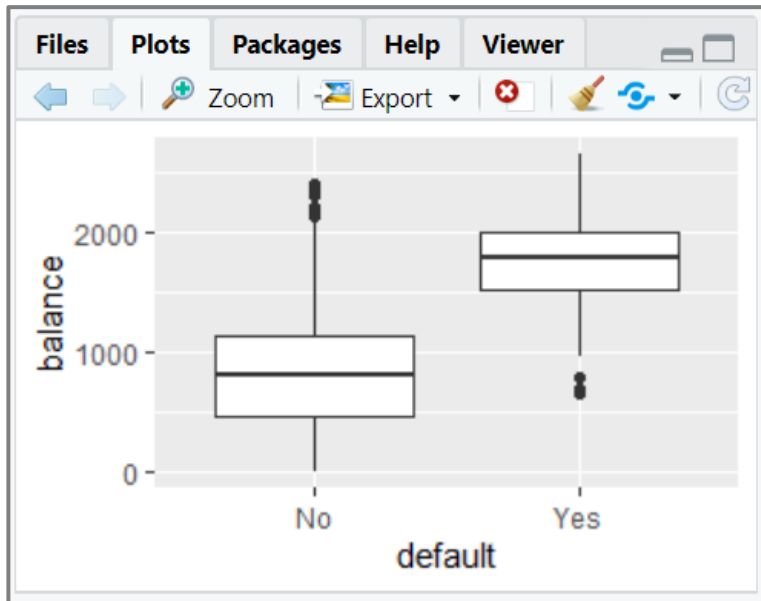
# 箱ひげ図

## 離散変数の値別に連続変数の分布を確認する

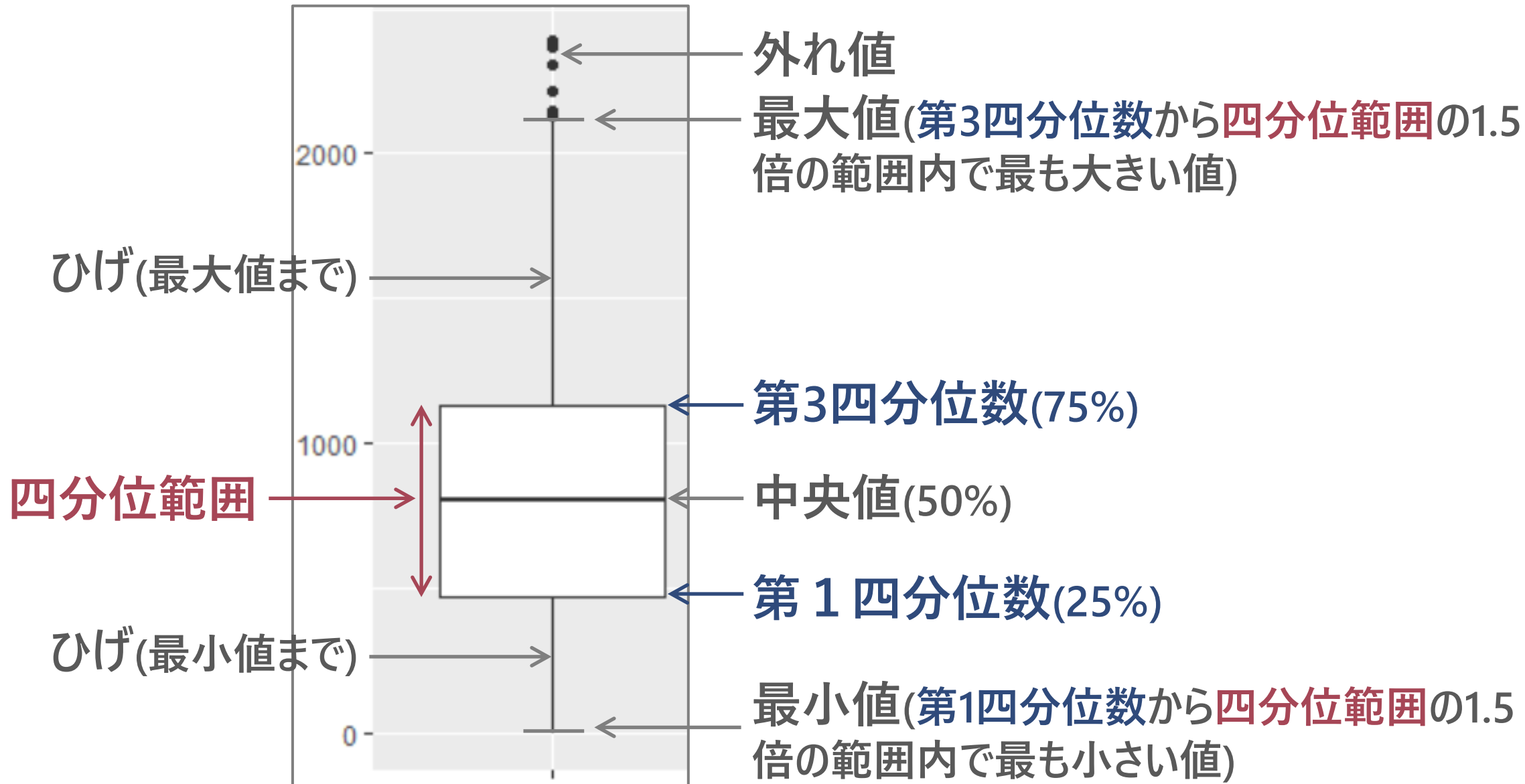
- ggplot2パッケージ



```
ggplot(data = DEFAULT,  
       mapping = aes(x = default, y = balance)) +  
  geom_boxplot(fill = "steelblue")
```



# 箱ひげ図



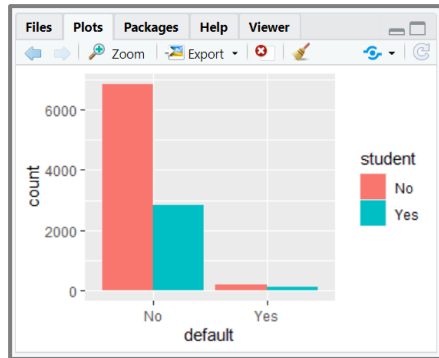
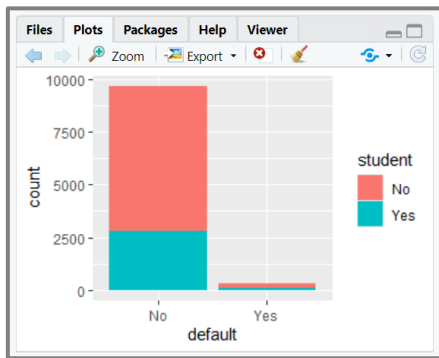
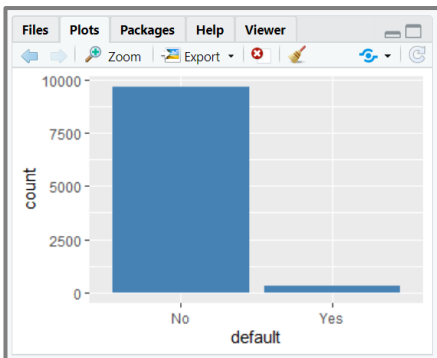
# 棒グラフ（グルーピング）

## 別のカテゴリ変数でグルーピングして棒グラフを確認する

- ggplot2パッケージ

```
ggplot(data = DEFAULT,
       mapping = aes(x = default, fill = student)) +
  geom_bar(width = 0.9, fill = "steelblue",
           position = position_dodge())
```

#position:配置の指定, dodge:横に配置  
#「position = "dodge"」という略記も可能



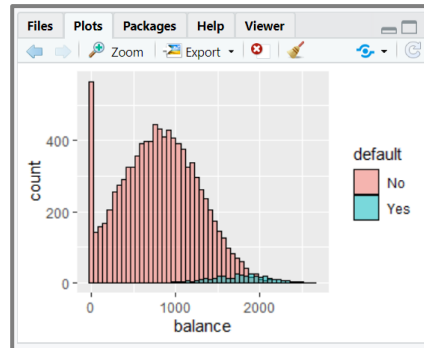
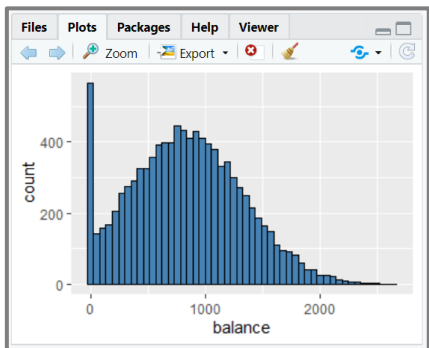
# ヒストグラム（グルーピング）

別のカテゴリ変数でグルーピングしてヒストグラムを確認する

- ggplot2パッケージ



```
ggplot(data = DEFAULT,
       mapping = aes(x = balance, fill = default)) +
  geom_histogram(bins = 50,
                 fill = "steelblue", colour = "black",
                 position = position_identity(), alpha = 0.8)
#identity: 配置調整なし(独立表示), alpha: 透過度
```



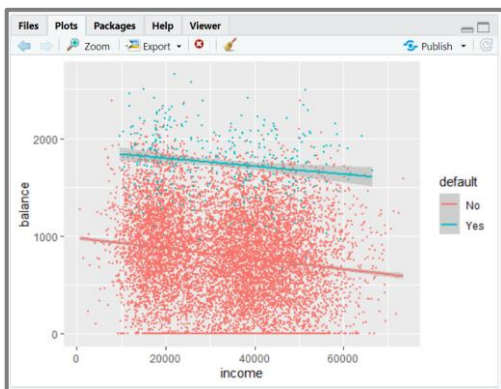
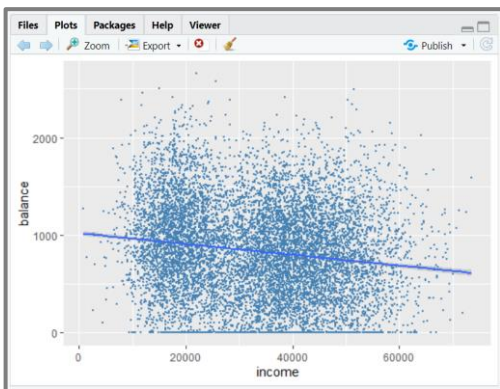
# 散布図（グルーピング）

## 別のカテゴリ変数でグルーピングして散布図を確認

- ggplot2パッケージ



```
ggplot(data = DEFAULT,
       mapping = aes(x = income, y = balance,
                     colour = default)) +
  geom_point(colour = "steelblue", size = 0.5) +
  geom_smooth(method = "lm") #近似線(lm:linear model)
```



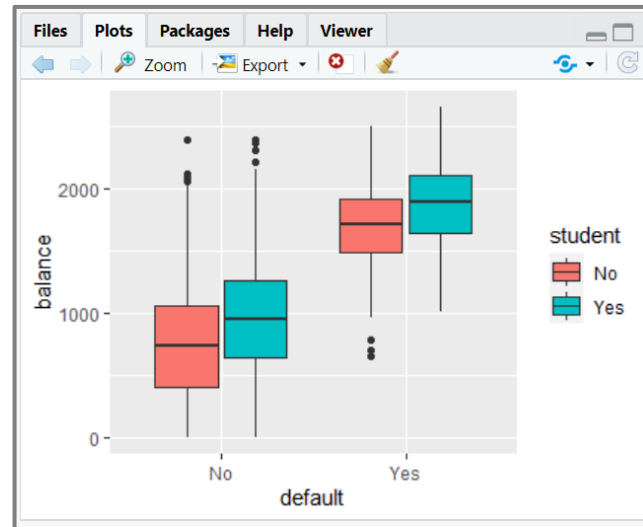
# 箱ひげ図（グルーピング）

別の離散変数でグルーピングして箱ひげ図を確認する

- ggplot2パッケージ



```
ggplot(data = DEFAULT,  
       mapping = aes(x = default, y = balance,  
                      fill = student)) +  
geom_boxplot(fill = "steelblue")
```



# セクション6

## データの整備

# tidy dataとは

tidy dataと呼ばれる列指向のデータ形式に整備する

雑然としたデータ：messy data

				
1	STORE	APPLE	ORANGE	BANANA
2	A	3	2	2
3	B	5	2	5
4	C	5	6	5
5	D			
6	E			
7	F			
8	G			
9	H	8	3	1
10	I	6	0	1


横長

● 見た目のわかりやすさ◎

● 「意味」が雑然としている

● 分析対象は「意味」...

整然としたデータ：tidy data

			
1	STORE	FRUITS	SALES
2	A	APPLE	3
3	A		0
4	A		2
5	A	GRAPE	1
6	B	APPLE	5
7	B		
8	B		
9	B		
10	B		
11	C	ORANGE	0
12	C	BANANA	5
13	C	GRAPE	0

縦長

● 個々の“観測”が1つの行をなす

● 個々の“変数”が1つの列をなす

● 個々の“観測パターン”が1つの表をなす



# tidy dataへの変換

tidyrパッケージのpivot\_longer関数で列指向のデータに変換する

- tidyrパッケージ

```
pivot_longer(data = DATA, cols = 変形対象列名のベクトル,  
             names_to = “キーとなる列”, values_to = “値となる列”)
```

	A	B	C	D		A	キー	値
1	STORE	APPLE	ORANGE	BANANA		STORE	FRUITS	SALES
2	A	3	0	2	2	A	APPLE	3
3	B	5	5	2	3	A	ORANGE	0
4	C	5	6	5	4	A	BANANA	2
5	D	9	1	2	5	A	GRAPE	1
6	E	2	2	3	6	B	APPLE	5

Diagram illustrating the transformation of data from a wide format to a long format using `pivot_longer()` and back to a wide format using `pivot_wider()`.

The left table (wide format) has columns A, B, C, and D. The right table (long format) has columns A, FRUITS (key), and SALES (value). The transformation is indicated by arrows and labels `pivot_longer()` and `pivot_wider()`.

# パイプ演算子

データ操作を行うdplyrパッケージ等では「パイプ演算子」が役に立つ

## ● dplyrパッケージ

`select(DATA, 列1, 列2, 列3, ...)`

## パイプ演算子

`DATA %>% select(DATA, 列1, 列2, 列3, ...)`

`DATA %>%  
select(列1, 列2, 列3, 列4, ...) %>%`

`filter(列1 == 条件値) %>%`

`mutate(追加列 = 値ベクトル)`



<-



%>%



%>%



%>%



# 列の抽出

## dplyrパッケージのselect関数で列を抽出（選抜）

### ● dplyrパッケージ



```
NEWDATA <- DATA %>%
```

```
  select(列1, 列2,...)
```

#多様なselectの方法

```
#select(列1:列3) #select(!列4)
```

```
#select(新しい列名 = 列1) #列を抽出して列名も変更
```

```
  #rename(新しい列名 = 列1) #列名変更のみ
```

```
#select(starts_with("col_")) #select(ends_with("_col"))
```

```
#select(contains("_")) #文字列による列抽出
```

```
#select(where(is.character)) #TRUEの列のみ
```

TRUE  
FALSE

```
is.XXX(変数)
```

```
#is.numeric(変数)
```

```
#is.character(変数)
```

```
#is.factor(変数)...
```

# 列の追加①

## dplyrパッケージのmutate関数で列を追加

### ● dplyrパッケージ



```
NEWDATA <- DATA %>%
```

```
  mutate(追加列1 = 追加する値, 追加列2 = 追加する値,...)
```

#「追加列」を「既存の列名」と同じにすると上書きされる

#多様なmutateの方法

```
#mutate(追加列2 = “Yes”) #すべて同じ値の列
```

```
#mutate(追加列3 = c(1:10000))
```

#異なる値を追加する場合は行数（サンプルサイズ）を同じにする

```
#mutate(追加列4 = 列1 + 列2) #既存の列による計算値を追加
```

# 列の追加②

## dplyrパッケージのmutate関数で列を追加

- dplyrパッケージ



```
NEWDATA <- DATA %>%
```

```
  mutate(追加列1 = if_else(列1 > 0, “Yes”, “No”))
```

```
  #if_else(条件式, TRUEの場合の値, FALSEの場合の値)
```

```
  #変数をカテゴリ化できる
```

```
#3カテゴリ以上に変換する場合
```

```
  #case_when(条件式1 ~ 条件式1がTRUEの場合の値,
```

```
             条件式2 ~ 条件式2がTRUEの場合の値,
```

```
             TRUE ~ 上記どちらもFALSEの場合の値)
```

# 行の抽出

## dplyrパッケージのfilter関数で行を抽出（保持）

- dplyrパッケージ

```
NEWDATA <- DATA %>%
```

```
  filter(列1 == 条件値,...)
```

```
  #条件を「,」でつないで「かつ」、「|」でつないで「または」
```

```
#多様なfilterの方法
```

```
#filter(列1 != 条件値) #filter(列1 >= 条件値)
```

```
#filter(between(列1, 条件値1, 条件値2)) #条件値1以上2以下
```

```
#filter(列1 %in% c(条件値1, 条件値2,...))
```

```
  #いずれかの条件値と同じ行を抽出
```

```
#filter(!is.na(列1)) #その列がNAの行は抽出しない
```

# 行の並べ替え

## dplyrパッケージのarrange関数で行を並べ替え

- dplyrパッケージ

```
NEWDATA <- DATA %>%
```

```
  arrange(列1) #その列の値の小さい順（昇順）に並べ替え
```

#多様なarrangeの方法

```
#arrange(列1, 列2,...)
```

```
#列1で昇順に並べ替えて、列1の値が同じ場合は列2で昇順に並べ替え...
```

```
#arrange(desc(列1)) #大きい順（降順）に並べ替え
```

```
#arrange(desc(列1, 列2))
```

```
#列1で降順に並べ替えて、列1の値が同じ場合は列2で降順に並べ替え...
```

# 文字列の処理①

## stringrパッケージを用いて文字列を処理

- stringrパッケージ

```
str_c(DATA$列1, DATA$列2, sep = “_”)
#列1と列2を連結（sepで指定した文字や空白を間に挿入）
str_detect(string = DATA$列1, pattern = “_”)
#patternで指定した文字列を含むかどうか判定（論理値:TRUE/FALSE）
str_subset(string = DATA$列1, pattern = “_”)
#patternで指定した文字列を含む値（行）を抽出
str_replace(string = DATA$列1, pattern = “_”,
             replacement = “and”) #“_”を“and”に置換
str_trim(string = DATA$列1) #左端と右端の空白を除去
```



## 文字列の処理②

stringrで文字列を処理しながら、dplyrで行・列を操作

● dplyrパッケージ

● stringrパッケージ

```
NEWDATA <- DATA %>%
```

```
  mutate(追加列 = str_c(列1, 列2, sep = " "))
```

#列1と列2を半角スペースを挿入して連結した文字列の列を追加

```
NEWDATA <- DATA %>%
```

```
  mutate(列1 = str_trim(列1)) #空白を除去して上書き
```

```
NEWDATA <- DATA %>%
```

```
  filter(str_detect(string = 列1, pattern = "_"))
```

#指定した文字列を含む行を抽出

# 欠損値補完

## 変数の欠損値を補完するtidyrパッケージ関数

- tidyrパッケージ

#欠損値を前の値で補完

```
fill(data = データフレーム, 補完する列,...)
```

- tidyrパッケージ

#特定の値で補完

```
replace_na(data = データフレーム,  
replace = list(補完する列1 = 補完する値,...))
```

	A	B	C
1	企業	項目	金額
2	一番商事	売上	1000
3		費用	800
4		利益	200
5	二番物産	売上	1200
6		費用	1100
7		利益	100
8	三番商事	売上	900
9		費用	850
10		利益	50
11			

# データ結合①

2つのデータセットに共通の列をキーとして結合

従業員データセット

従業員ID	氏名	入社年	部門ID
1	斉藤	2019	1
2	鈴木	2019	3
3	高橋	2019	2
4	田中	2020	1
5	木村	2021	99

部門データセット

部門ID	部門	住所
1	営業部	第1ビル
2	経理部	第2ビル
3	人事部	第2ビル
4	開発部	第3ビル

<inner\_join>

従業員ID	氏名	入社年	部門ID	部門	住所
1	斉藤	2019	1	営業部	第1ビル
2	鈴木	2019	3	人事部	第2ビル
3	高橋	2019	2	経理部	第2ビル
4	田中	2020	1	営業部	第1ビル

## データ結合②

### dplyrパッケージのinner\_join関数などで結合

- dplyrパッケージ

```
inner_join(x = DATA1, y = DATA2, by = キーとなる列)
```

#キー列についてxとyに共通する値が存在する行のみ返す

```
left_join(x = DATA1, y = DATA2, by = キーとなる列)
```

#キー列についてxに値が存在する行を返す

```
right_join(x = DATA1, y = DATA2, by = キーとなる列)
```

#キー列についてyに値が存在する行を返す

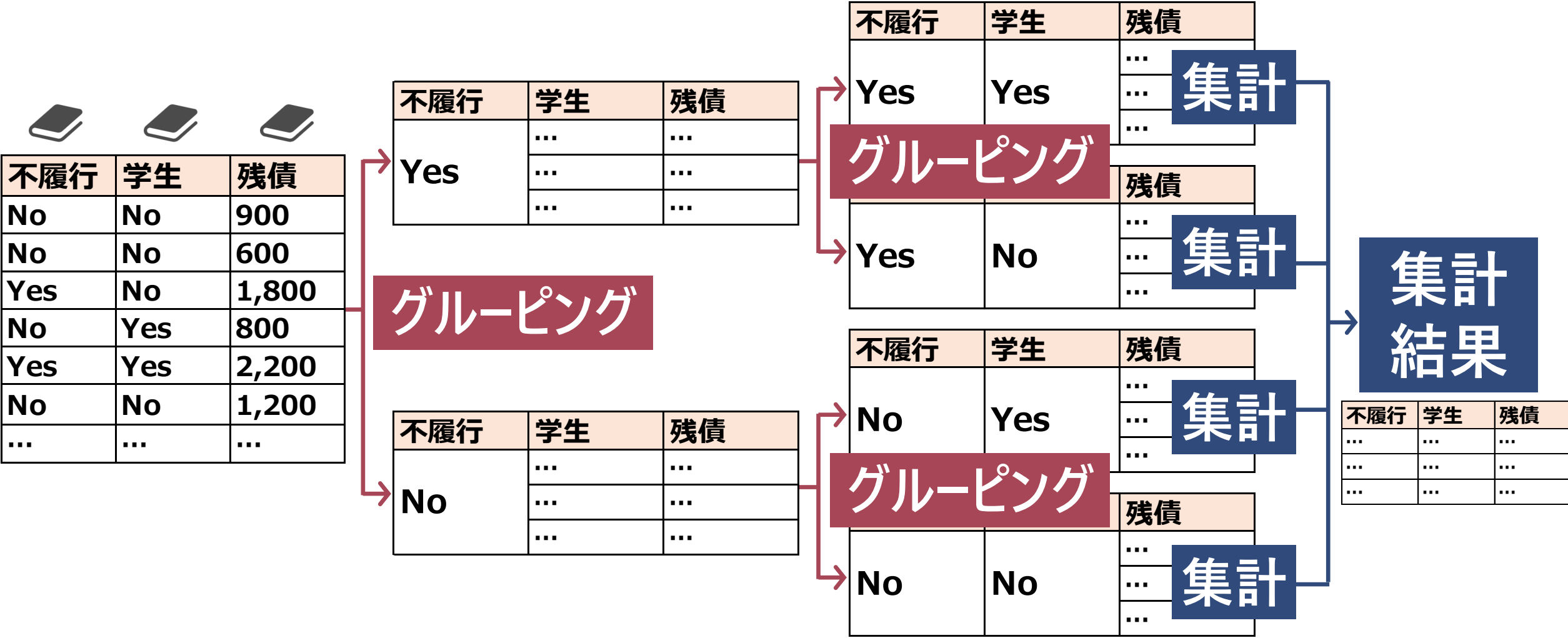
```
full_join(x = DATA1, y = DATA2, by = キーとなる列)
```

#キー列についてxまたはyに値が存在する行を返す

# セクション7 データのグループ集計

# グループ集計のイメージ

グループ別にデータを集計し、グループ別の特徴を確認する



# グルーピング①

## dplyrパッケージのgroup\_by関数でデータフレームをグループ化

- dplyrパッケージ

```
GROUPED_DATA <- DATA %>%
```

```
  group_by(グループ化する列1, グループ化する列2,...)
```

```
  #指定の列でグループ化
```

不履行	学生	残債	不履行	学生	残債
No	No	900	No	No	900
No	No	600	No	No	600
Yes	No	1,800	No	No	600
No	Yes	800	No	No	600
Yes	Yes	2,200	No	No	600
No	No	1,200	No	No	600
...	...	...	No	No	600

見た目は変化なし

不履行	学生	残債	不履行	学生	残債
Yes	...	...	Yes	Yes	...
Yes	...	...	Yes	No	...
Yes	No	1,800	Yes	No	...
No	Yes	800	Yes	No	...
Yes	Yes	2,200	Yes	No	...
No	No	1,200	Yes	No	...
...	...	...	Yes	No	...

中身はグループ化

## グルーピング②

### dplyrパッケージでグルーピング用の新たな変数を追加してグループ化

- dplyrパッケージ

```
GROUPED_DATA <- DATA %>%  
  mutate(追加する列1  
    = if_else(条件式, 真の場合の値, 偽の場合の値)) %>%  
  group_by(追加した列1)  
#カテゴリ変数に変換した列を追加 → 追加した列でグループ化
```

不履行	学生	残債	mutate	不履行	学生	残債	残債1,000未満	グループ化
No	No	900		No	No	900	1,000未満	残債1,000未満
No	No	600		No	No	600	1,000未満	
Yes	No	1,800		Yes	No	1,800	1,000以上	残債1,000以上



# グループ集計（基本）

## dplyrパッケージのsummarise関数でグループ集計

~~mean(GROUPED\_DATA\$列名)~~

~~median(GROUPED\_DATA\$列名)~~

~~sd(GROUPED\_DATA\$列名)~~

~~var(GROUPED\_DATA\$列名)~~

グループ化データでもグループごとに処理されない

### ● dplyrパッケージ



グループ化データをグループごとに処理してくれる

```
GROUPED_DATA %>%
```

```
  summarise(
```

```
    列1_mean = mean(列1),
```

```
    列1_sd = sd(列1), ...
```

```
)
```

```
#グループ化データを集計してグループごとに返す
```

グループ列A	列1_mean	列1_sd
グループ1	...	...
グループ2	...	...

# グループ集計（順位付け）

## dplyrパッケージのsummarise関数のなかでrank関数

```
rank(GROUPED_DATA$列名, ties.method = "min")
```

#変数(列)を順位付け #ties.method:同順位の場合の扱い

ヒット数	average	min	max
120	1	1	1
145	3	2	4
145	3	2	4
145	3	2	4
180	5	5	5

-ヒット数	average	min	max
-120	5	5	5
-145	3	2	4
-145	3	2	4
-145	3	2	4
-180	1	1	1

### ● dplyrパッケージ

```
GROUPED_DATA %>%
```

```
  summarise(列1_rank = rank(列1))
```

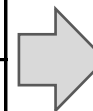
#グループ化データを集計してグループごとに順位を返す

# グループ集計（累積和）

## dplyrパッケージのsummarise関数のなかでcumsum関数

```
cumsum(GROUPED_DATA$列名)  
#前から順番に累積和を計算  
#cumsum:Cumulative Sums
```

HR数	累積和
1	1
7	8
18	26
20	46
10	56



ア・リーグ	
HR数	累積和
20	20
18	38
ナ・リーグ	
HR数	累積和
-	-
-	-
-	-
-	-
-	-
-	-

### ● dplyrパッケージ

```
GROUPED_DATA %>%
```

```
  summarise(列1_cumsum = cumsum(列1))
```

```
  #グループ化データを集計してグループごとに累積和を返す
```

```
  #上位（下位）から累積していく場合はarrange関数で並べ替えてから
```

# グループ集計（累積最小値・最大値）

## dplyrパッケージのsummarise関数のなかでcummin/cummax関数

```
cummin(GROUPED_DATA$列名)  
#前から順番に最小値を更新していく  
cummax(GROUPED_DATA$列名)  
#前から順番に最大値を更新していく
```

### ● dplyrパッケージ

```
GROUPED_DATA %>%
```

```
  summarise(列1_cummin = cummin(列1))
```

```
  #グループ化データを集計してグループごとに「それまでの最小値」を返す
```

```
  #「それまでの最大値」はcummax関数
```

2019年		ここまでの最小値	ここまでの最大値
収益率		0.959	0.959
2020年		ここまでの最小値	ここまでの最大値
収益率		0.574	0.574
	0.918	0.574	0.918
	0.621	0.574	0.918
	-0.65	-0.65	0.918
	-0.359	-0.65	0.918

# グループ集計のグラフ化

集計結果のグラフ化はggplotの“stat”という引数で行う

● dplyrパッケージ

● ggplot2パッケージ

```
DATA %>%
```

```
  group_by(グループ化する列) %>%
```

```
  summarise(列1_mean = mean(列1)) %>%
```

```
  ggplot(mapping = aes(x = グループ化した列, y = 列1_mean)) +  
  geom_bar(stat = “identity”)
```

```
#データ %>% グループ化 %>% 集計 %>% グラフ下地 + 棒グラフ
```



%>%



%>%



%>%



# 以上

(最後まで受講いただき本当にありがとうございました)