R/Rstudioを用いたプログラミング

講師：遠山 祐太

最終更新：2024-11-16

R & Rstudio 入門

どうしてプログラミングを学ぶのか？

- データを用いた統計・実証分析
  1. データセットを作成する(スクレイピング・クリーニング)
  2. データの記述統計（表・グラフ）
  3. 回帰分析
  4. 分析結果をまとめた図表を作る

- 数値シミュレーション
  - 例：漸近理論（asymptotic theory）はサンプルサイズが充分大きい場合を考える (i.e., $N \to \infty$)
    - 大数の法則（law of large numbers）・中心極限定理（central limit theorem）
    - 漸近近似の精度は？
  - **モンテカルロシミュレーション（Monte Carlo simulation）**とよばれる

どうしてRを使うのか？

- **フリーソフトウェア**
  - Stataは高価
  - Matlabは在学中はライセンスが使えるが卒業後は・・・
- 柔軟さと使いやすさのバランスがちょうど良い
  - Stataでは計量分析がしやすい反面、自分でプログラムを書くのは難しい
  - Matlabはその逆
  - データ構築・回帰分析・機械学習・構造推定など、Rで大体なんでもできる
- ユーザが多い(ネットワーク効果)
  - 日本語でも英語でもネットに資料がある
  - ユーザが開発したパッケージが多い：`tidyverse`、機械学習ツールなど...

とはいえ、最終的には好みの問題

- 選択肢はいろいろ：Stata, Matlab, Python, Julia, etc...
- Pythonも非常にPopular (私はほとんど使ったことない)
- 最終的には皆さん次第（私も未だにStataやMatlabを使っています）
- **この授業ではRを使うことを原則とします。**

R（プログラミング言語全般も）の学び方

- 「手を動かして学ぶ」（**Learning by doing**）しかないです。
- いくつかコード例を紹介するが、**R言語の使い方の講義ではない**ことに注意。
- もしプログラミングでつまづいたら...
  1. まずはググる！エラーメッセージをそのままGoogle検索してみる。
  2. Moodleのフォーラムに質問を投稿する。
  3. オフィスアワーにTAか講師に質問する。

Rに関する便利なオンライン資料

- 森知晴「卒業論文のためのR入門」
  - 一番最初のとっかかりとして。
- Wickham and Grolemund "R for Data Science" https://r4ds.had.co.nz/
  - `tidyverse`の開発者
  - 英語版は無料。 邦訳も入手可能。
- Datacamp https://www.datacamp.com/
- 宋財泫・矢内勇生「私たちのR: ベストプラクティスの探究」https://www.jaysong.net/RBook/

Rに関する書籍資料

- 松村ら「改訂2版 RユーザのためのRStudio[実践]入門」
- 辻・矢吹「実践Data Scienceシリーズ ゼロからはじめるデータサイエンス入門 R・Python一挙両得」

本日の流れ

- RとRstudioの第一歩
  - 環境構築
  - パッケージ
- 分析用フォルダのセットアップ
  - 作業ディレクトリ
  - RStudio project
  - フォルダの構成例
- 基本的な計算（計算機としてのR）
- 統計学の復習を兼ねたプログラミング学習
  - 統計学： 統計的推測と数値シミュレーション
  - プログラミング：変数（variable）、ワークフロー(if文・for文)、関数（function）

はじめよう

RとRStudioのインストール

- RとRStudioとはなにか
  - Rはプログラミング言語
  - RStudioはプログラミングに便利なツールを提供してくれる**統合開発環境（integrated development environment; IDE）**

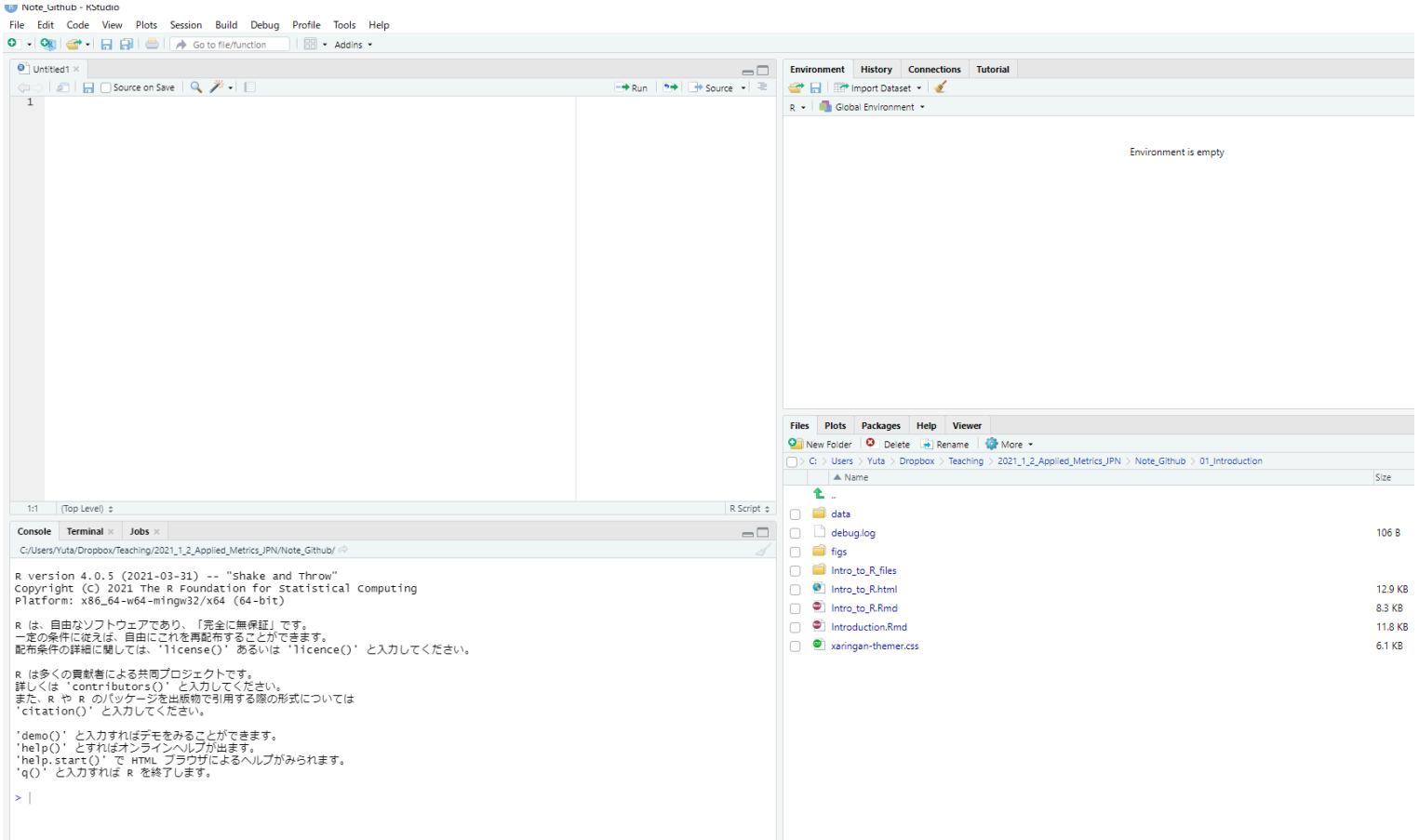
- RとRStudioのインストール
  1. R: https://www.r-project.org/
  2. RStudio: https://www.rstudio.com/

- 詳細についてはこちら

チートシート

- RStudio開発チームは様々な"cheatsheets"を提供してくれている。
- 本講義で役立ちそうなもの：
  - RStudio IDE Cheatsheet
  - Base R
  - Data Wrangling
- 他の便利なもの：
  - estimatr：回帰分析
  - stata2r：RでStataを実行
  - R Markdown

RStudioの概観


Note_Github - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Untitled1
Source on Save
Run
Source
1
1:1
(Top Level)
R Script
Console
Terminal
Jobs
C:/Users/Yuta/Dropbox/Teaching/2021_1_2_Applied_Metrics_JPN/Note_Github/
R version 4.0.5 (2021-03-31) -- "Shake and Throw"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R は、自由なソフトウェアであり、「完全に無保証」です。
一定の条件に従えば、自由にこれを再配布することができます。
配布条件の詳細に関しては、'license()' あるいは 'licence()' と入力してください。
R は多くの貢献者による共同プロジェクトです。
詳しくは 'contributors()' と入力してください。
また、R や R のパッケージを出版物で引用する際の形式については
'citation()' と入力してください。
'demo()' と入力すればデモをみることができます。
'help()' とすればオンラインヘルプが出ます。
'help.start()' で HTML ブラウザによるヘルプがみられます。
'q()' と入力すれば R を終了します。
>
Environment
History
Connections
Tutorial
Import Dataset
R
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
C: Users Yuta Dropbox Teaching 2021_1_2_Applied_Metrics_JPN Note_Github 01_Introduction
Name
Size
..
data
debug.log
106 B
figs
Intro_to_R_files
Intro_to_R.html
12.9 KB
Intro_to_R.Rmd
8.3 KB
Introduction.Rmd
11.8 KB
xaringan-themer.css
6.1 KB

4つの枠（Pane）

1. Source：コードを書く場所
2. Console：コードを走らせる場所
3. Environment/History: 変数の一覧 ／ Consoleで走らせたコードの履歴
4. Files/Plots/Packages/Help: ファイル一覧、など

Source (ソース)

- ここにコードを書く。
- コードをスクリプトファイルとして保存する。(`.R`ファイル)
- Runをクリックして全てのコードを読み込む。
- `SHIFT + ENTER`（Windows）で選択済みの部分を読み込む。

Console (コンソール)

- コードが動くところ
- Source Pane（枠）でRunをクリックするとコードが回りだす。
- ここに直接コードを入力することもできる。

Packages (パッケージ)

- オープンソースのRにとっての大きな強みはパッケージが豊富なこと。
  - パッケージからユーザが開発した便利な関数を呼び出せる。
- パッケージをインストールするには`install.packages()`を利用する。
  - 料理本を買ってきて、棚にしまっておくようなもの。
- パッケージを使いたければ、インストール後に`library()`で読み込む。

例

- 図作成のためのggplot2を紹介するが、**いまインストールしないこと！**(時間がかかるので)
- ggplot2は次のコマンドでインストールできる。

```
install.packages("ggplot2")
```

- パッケージがインストールされたら、使う前にlibrary()で読み込む。

```
library("ggplot2")
```

- Rをいったん閉じると、読み込んだパッケージはすべて閉じられる。
- Rをまた開くときは、パッケージを再インストールする必要はないが、使いたいものをlibrary()で再読み込みする必要がある。

tidyverse

- tidyverseは、主にHadley Wickamsによって開発されたパッケージ群のこと。
  - dplyr・readr・ggplot2・readr・stringrなどなど
- tidyverseはデータ分析におけるデファクトスタンダード
- 本講義ではtidyverseをインストールする。（時間がかかるので注意）

```
install.packages("tidyverse")
```

分析用フォルダのセットアップ

分析用フォルダの作り方

- データ分析には様々なファイル：スクリプト（コード）、データ、アウトプット（画像、表）
- ポイント：ファイルを機能ごとに分けて整理すると良い。
- 作り方は結局人それぞれになるが、以下では参考例を理由とともに紹介する。

分析フォルダの一例

```
プロジェクト/
├── data/
│   └── data.csv
├── script/
│   ├── script.R: スクリプトファイル
│   └── function.R: 関数を保存したファイル
├── output/
│   ├── figure(.jpg, .pdf)
│   └── table(.tex, .csv)
├── script.R / master.R
├── markdown_report.RMD
└── readme.md
```

- `data`: データを入れる。**絶対に触らない(上書きしない)フォルダ**
- `script`: スクリプト(.Rファイル)を入れる。
  - もしファイル一つで済むならば、直下の`script.R` だけでもOK
  - 直下の`master.R` において、`script`内のフォルダをどの順番で回すかを指定する。
- `output`: 分析のアウトプット。主に図表

続：分析フォルダの一例

```
プロジェクト/
├── data/
│   └── data.csv
├── script/
│   ├── script.R: スクリプトファイル
│   └── function.R: 関数を保存したファイル
├── output/
│   ├── figure(.jpg, .pdf)
│   └── table(.tex, .csv)
├── script.R / master.R
├── markdown_report.RMD
└── readme.md
```

- `markdown_report.RMD`: Rmarkdownによる分析レポート。
- `readme.md`: 分析フォルダに関する説明ファイル。マークダウンファイルでも、テキストファイルでも、ワードでもなんでもOK。

ファイル・フォルダ管理における重要ルール

- どのファイルを上書きしてOKか否かをはっきりさせる。
  - 元(ソース)となるデータファイル（エクセルやCSV）は絶対に上書きしない。

- 分析を一から再現できるよう、コードを整理する。(**再現性**)
  - 仮にoutputフォルダが空になっても、dataとscriptがあればmaster.Rを回すことでoutputを再現できる。

(時間があれば/おいおい) 応用例

- Git/Githubによるファイル管理 (特に変更履歴)
- 元データから、分析のためにデータ自体の加工が必要な場合
  - 例えば、`data_raw` と `data_cleaned` に分ける。
- 一時的な作業ファイルを置いておく `temp` フォルダの作成
- 分析内容が巨大になった場合：記述統計、回帰分析、シミュレーション分析、などなど
  - 個人的には、分析のサブタスクごとにフォルダを作るとやりやすい（が人による）
- markdown report と scriptの使い分け

プログラミングの基本：作業ディレクトリ

- **作業ディレクトリ（working directory）**はシステムがその時点で参照しているフォルダ。
  - 外部のファイルを読み込む際に重要。(後述)
  - なお、ディレクトリとフォルダは意味がほぼ同じ。
- 作業ディレクトリは次のコマンドで確認できる。

```
getwd()
```

```
## [1] "C:/Users/taho1/OneDrive/ドキュメント/GitHub/Applied_Econometrics_JPN/02_R_Programming"
```

- 作業ディレクトリを変更する場合は、次のように入力する。

```
setwd("ディレクトリのパス")
```

オススメ：RStudio におけるプロジェクトの作成

- Rstudioでは分析作業用のフォルダを**プロジェクト** として設定できる。
- 方法：左上の左から 2 つ目の緑のプラス(Create a project)をクリック (授業で手本)
- `XXXX.Rproj` をクリックしてRStudioを起動すると、作業ディレクトリがプロジェクトフォルダに自動で設定される。
- (今日は割愛) `here` パッケージの活用

基本的な演算

基本的な計算

手始めとして、Rを単純な計算機として使ってみよう。

四則演算

数式	R	出力
$3+2$	`3 + 2`	5
$3-2$	`3 - 2`	1
$3 \cdot 2$	`3 * 2`	6
$3/2$	`3 / 2`	1.5

```
1 + 3
```

```
## [1] 4
```

累乗

数式	R	出力
3^2	`3 ^ 2`	9
$2^{(-3)}$	`2 ^ (-3)`	0.125
$100^{1/2}$	`100 ^ (1 / 2)`	10
$\sqrt{100}$	`sqrt(100)`	10

特別な定数

数式	R	出力
π	`pi`	3.1415927
e	`exp(1)`	2.7182818

対数

- 自然対数を表すために $\ln$ と $\log$ を置き換え可能なものとして使うことに注意。
- Rには`ln()`がないので、自然対数を表すには代わりに`log()`を用いる。

数式	R	出力
$\log(e)$	`log(exp(1))`	1
$\log_{10}(1000)$	`log10(1000)`	3
$\log_2(8)$	`log2(8)`	3
$\log_4(16)$	`log(16, base = 4)`	2

三角関数

数式	R	出力
$\sin(\pi/2)$	`sin(pi / 2)`	1
$\cos(0)$	`cos(0)`	1

Helpの見方

- Rを計算機として使うにあたって、さまざまな関数を見た： `sqrt()`・`exp()`・`log()`・`sin()`
- Rの関数についての説明を見るには、関数名の前に？マークをつければよい。

```
?log
?sin
?paste
?lm
```

プログラミングの基本

概要

- 題材：母平均の統計的推定

- 手を動かしながら、以下の点について見ていこう
  - 作業ディレクトリ（working directory）
  - 変数（variable）
  - ワークフロー：if文・for文
  - 関数（function）定義

演習：母平均の統計的推定

- 統計学・計量経済学では、**母集団（population）**に関して、**データ(標本)**を用いた推測を行う。

- 母平均に関する**統計的推定（statistical estimation）**
  - ある確率変数 Y を考える。 Y はある分布に従っている。
  - 母平均＝期待値＝ $E[Y]$ を知りたい。
  - 母集団から N 個の観測を**無作為抽出（random sampling）**し、 **標本** $\{Y_i\}_{i=1}^N$ を得る。
  - **標本平均（sample mean）**： $\bar{Y} = \frac{1}{n}\sum_{i=1}^n Y_i$

- 図解：

用語の整理：推定対象・推定量・推定値

- **推定対象** (estimand): 推定したいと考えている、母集団の特徴
  - 例：母平均

- **推定量** (estimator) : データ（サンプル）を所与として、目標とするパラメタを推定するルール。
  - いわば、データから推定したい対象への関数。
  - 例：標本平均
  - サンプルがランダム(確率的)であるため、推定量も**確率変数** になる。

- **推定値** (estimate): データを推定量に代入したときに得られる値。
  - 例：データに基づいて計算された標本平均の値

なぜ標本平均？

- $\bar{Y}$ が母平均の良い推定量（estimator）なのはなぜだろうか？
- 理論的理由１：
- 理論的理由２：

標本平均(という推定量)の分布とは何か？

- 理論的に「標本平均の期待値」は計算できる。しかし、直観的な理解は非常に難しい。
- 理由：データ分析において、我々が得るデータ(標本)は、抽出されたものただ一つ。
  - そのデータから計算した標本平均は推定**値** (not 推定量)
- ゆえに、「標本平均が分布を持っている」という点を理解しづらい。
  - (恐らく頻度論的統計学がわかりづらい理由の根幹)
- 図：標本平均の「分布」

数値シミュレーションで標本平均の性質を調べる

- まず、なんらかの変数の母集団分布を用意する。
- 母集団から無作為に標本を抽出し、対応する標本平均を計算する。
- この手続きを繰り返し、標本平均を多数得る。-> 標本平均の分布が得られる。

Step 1 (/4) : パッケージの読み込み

- `install.packages("")`で`tidyverse`をインストールする。

```
library("tidyverse")
```

Step 2 (/4)：母集団を用意

- PUMS of U.S. Cencus 2000の所得と年齢のデータを用いる。
  - PUMS: Public Use Microdata Sample
  - `data_pums_2000.csv`が、data フォルダに入っていることを確認する。
- RにPUMSを読み込む前に、**変数**について説明。

プログラミングの基本その２：変数

- **変数（variable）**とは、データを保存できる名前付きの入れ物のこと。
- 例えば、変数は次のように定義できる。

```
X <- 10
```

- ここで、変数Xは後ほど参照することができる。

```
X
```

```
## [1] 10
```

```
2*X
```

```
## [1] 20
```

データの型 (Data Type)

- 変数は多種多様なデータを格納できる。
  - 数字（scalar）
  - ベクトル（vector）
  - 行列（matrix）
  - リスト（list）
  - データフレーム（data frame）
- これらをデータ型と呼ぶ。

PUMSデータの読み込み

- readrパッケージ（tidyverseに含まれる）のread_csv()関数を使って読み込む。

```
pums2000 <- read_csv("data/data_pums_2000.csv")
```

- 注意点１：
  - read_csv()の引数（argument）においてファイルのパスを指定する。
  - 作業ディレクトリにおけるdataフォルダのdata_pums_2000.csvファイルを探す。
  - このような参照をするパスの表記法を**相対パス（relative path）**とよぶ。
  - 絶対パス（absolute path）を用いて参照することもできるが、相対パスを強く推奨する。

- 注意点２：
  - 変数pums2000はdata_pums_2000.csvのデータを格納する。
  - この種の変数を**データフレーム（data frame）**とよぶ。

母集団を決める

- このデータを**母集団**として扱う。

```
# pop_incはベクトル
pop <- as.vector(pums2000$INCTOT)
```

- 母集団の平均と標準偏差

```
# 母集団の平均所得
pop_mean = mean(pop)
pop_mean
```

```
## [1] 30165.47
```

```
# 母集団の所得の標準偏差
pop_sd   = sd(pop)
pop_sd
```

```
## [1] 38306.17
```

母集団の所得分布（米ドル単位）

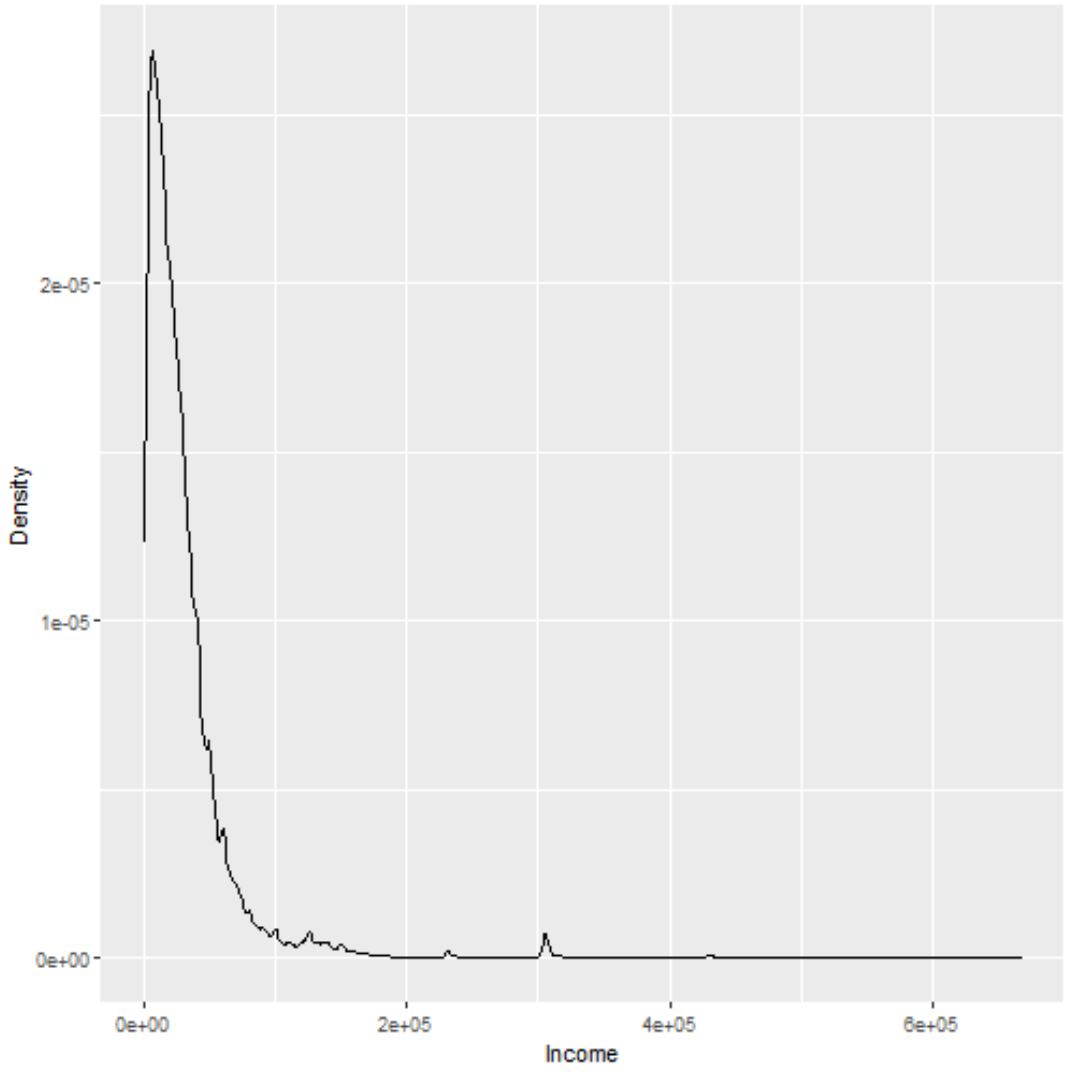
- ggplot2を使おう。

```
fig <- ggplot2::qplot(pop, geom = "density",
      xlab = "Income",
      ylab = "Density")
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

- 注意：
  - qplot()はggplotパッケージの関数。
  - パッケージ名::関数名は関数を呼び出す方法の一つ。
  - 変数figは図を格納している。

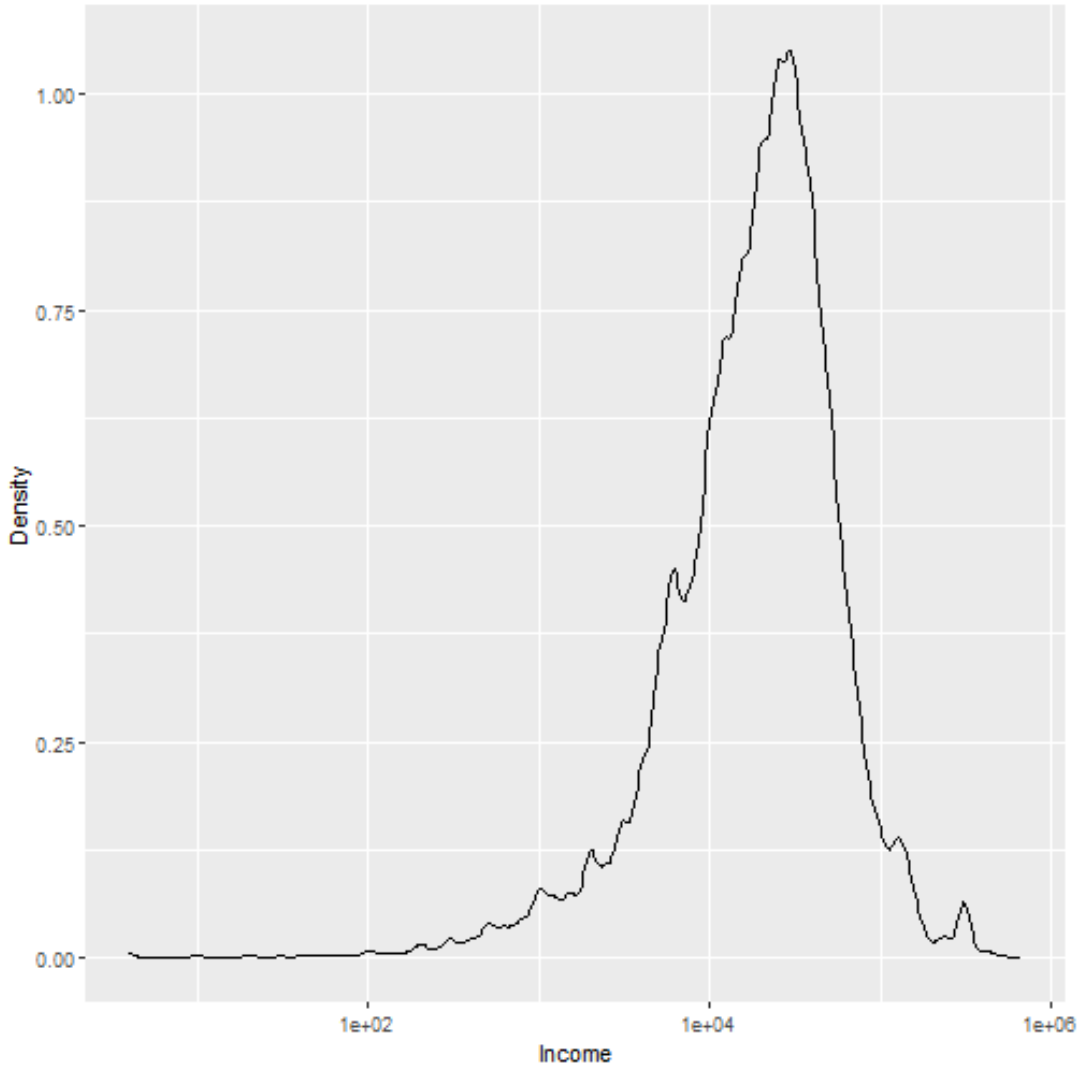
```
plot(fig)
```



- 分布の裾が長い。
- 対数スケールでプロットしてみよう。

```
# `log`オプションで対数をとる軸を指定
fig2 <- qplot(pop, geom = "density",
      xlab = "Income",
      ylab = "Density",
      log = "x")
```

```
plot(fig2)
```



Step 3 (/4)：無作為標本と標本平均の計算

- 無作為標本（random sample）から作成した標本平均が、真の母平均にどれだけ近いか調べてみよう。
- 母集団から N 個の観測を含む無作為標本を取り出し、標本平均 $\bar{Y}$ を計算する。
- 例えば、これを200回くり返し、200個の標本平均を得る。

無作為抽出のしかた

- まず、結果の再現性（reproductivity）を担保するために、**乱数（random number）**の**シード値（seed）**を設定しなければならない。

```
set.seed(12345)
```

- 変数popから100個の観測で構成される標本を取り出す。

```
test <- sample(x = pop, size = 100)
```

- 変数testに母集団から無作為抽出された100個の観測を格納する。標本平均は次のように計算できる。

```
mean(test)
```

```
## [1] 25756.1
```

プログラミングの基本その３：for文

- 同じ手続きをくり返したかったら？`for`文を使おう！
- `for`文を使う前に：

```
Nrep = 200
result1 <- numeric(Nrep)
```

- `result1`は長さ200で全ての要素が0のベクトルである。

- キモの部分はここ

```
# 結果を格納するベクトル
result1 <- numeric(Nrep)

# ループ
for (i in 1:Nrep ){
    test <- sample(x = pop, size = 100)
    result1[i] <- mean(test)
}
```

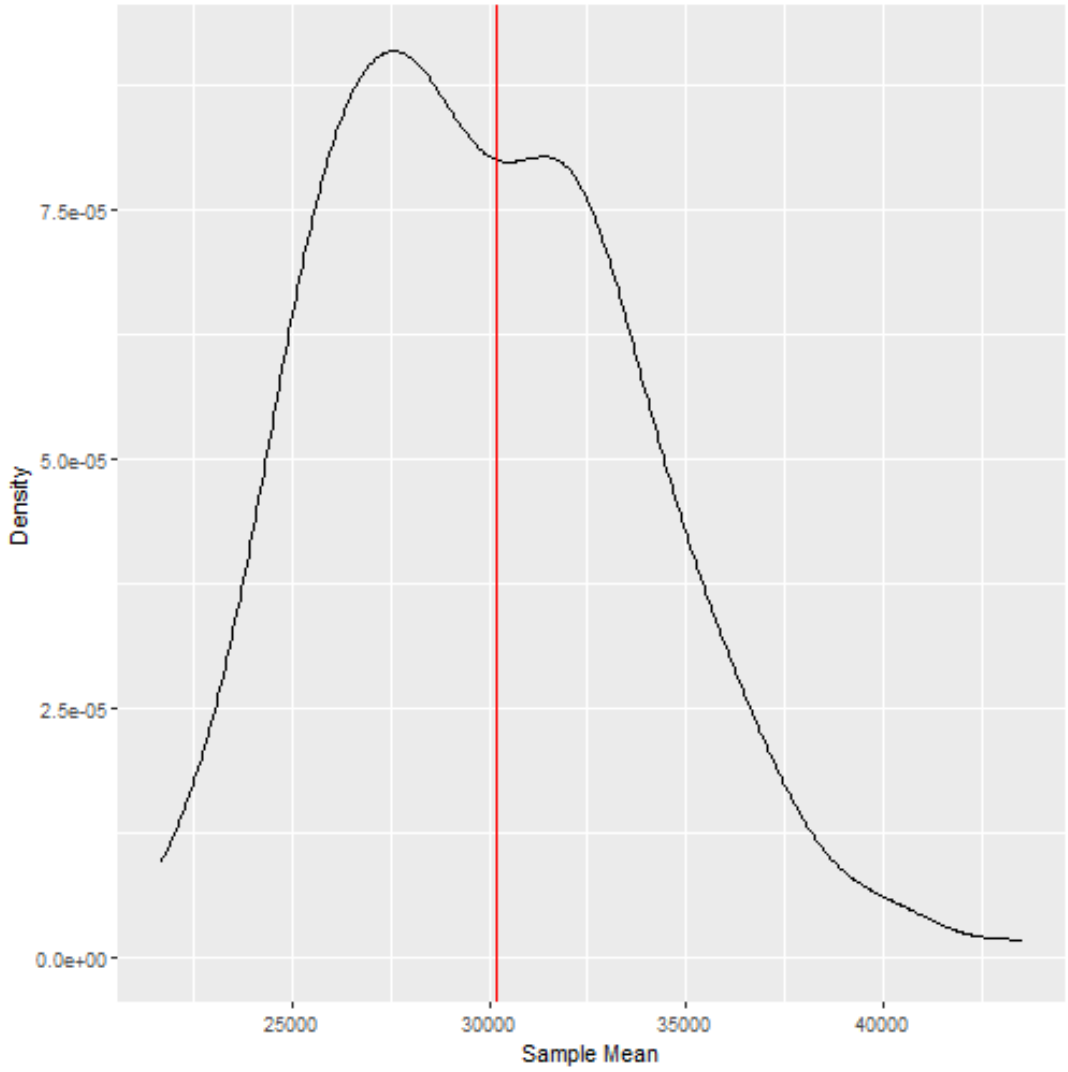
- `result1`には、200個の無作為標本から計算された200個の標本平均が格納されている。
- プログラミングのコツ：ループの結果を格納するベクトルを先に用意しておく。さもないと計算が極めて遅くなる。

Step 4 (/4)：標本平均の分布

- 標本平均の分布はどんな形をしているだろうか？

```
fig3 <- qplot(result1, geom = "density",
      xlab = "Sample Mean",
      ylab = "Density") +
  geom_vline(xintercept = pop_mean, color = "red")
```

```
plot(fig3)
```


7.5e-05
5.0e-05
2.5e-05
0.0e+00
Density
25000
30000
35000
40000
Sample Mean

プログラミングの基本その４：関数

- サンプルサイズが大きくなったら？例えば、 $N = 100, 200, 400$ ではどうだろうか？
- **関数（function）**を使って確かめる。
- 関数を定義するには、以下を特定する：
  1. 関数名
  2. 引数（argument）
  3. コード
  4. 出力（output）

関数定義

- f_samplemean関数

```
f_samplemean <- function(pop, size){

  Nrep = 200
  result <- numeric(Nrep)

  for (i in 1:Nrep ){
    test <- sample(x = pop, size = size)
    result[i] <- mean(test)
  }

  return(result)
}
```

詳細

- 引数（入力）
  - `pop`：母集団を格納している変数
  - `size`：サンプルサイズ（観測の数）
- コード：`size`個の観測を含む無作為標本を200個抽出し、標本平均を計算する
- 出力：200個の標本平均を格納している`result`

Step 3 と 4 の続き

- 定義された関数を用いて、異なるサンプルサイズの標本平均を作成する。

```
result1 <- f_samplemean(pop, size = 100)
result2 <- f_samplemean(pop, size = 400)
result3 <- f_samplemean(pop, size = 800)
```

- 可視化のためにデータフレームを用意する。

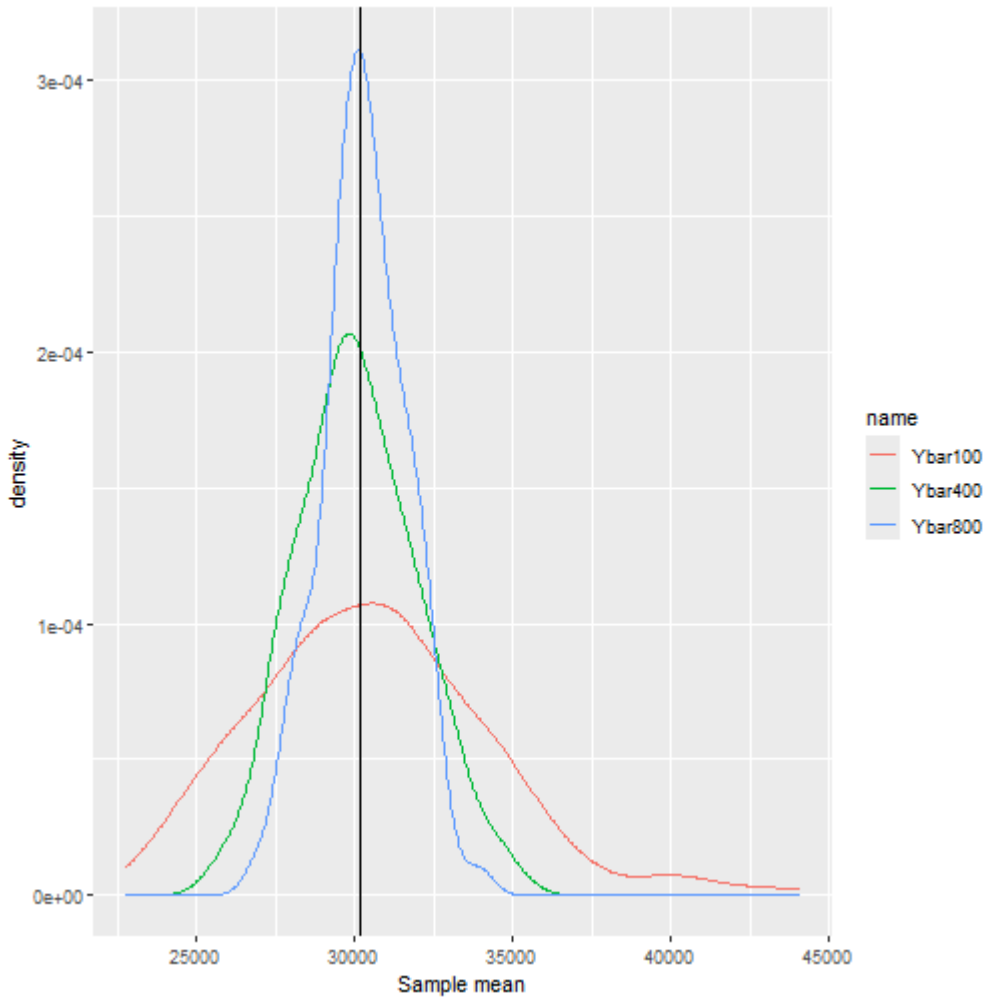
```
result_data <- tibble(  Ybar100 = result1,
                        Ybar400 = result2,
                        Ybar800 = result3)
```

分布の比較

```
# result_data のフォーマットを変更するために pivot_longer を使う
data_for_plot <- pivot_longer(data = result_data, cols = everything() )

# "ggplot2" で図を作成
fig <-
  ggplot(data = data_for_plot) +
  xlab("Sample mean") +
  geom_line(aes(x = value, colour = name ),   stat = "density" ) +
  geom_vline(xintercept=pop_mean ,colour="black")
```

```
plot(fig)
```


3e-04
2e-04
1e-04
0e+00
density
25000
30000
35000
40000
45000
Sample mean
name
Ybar100
Ybar400
Ybar800

練習問題

- **不偏性（unbiasedness）**および**一致性（consistency）**の観点から、シミュレーション結果を検討せよ。