データ分析基礎
R言語の基礎知識

京都大学 国際高等教育院 附属データ科学イノベーション教育研究センター

せきど ひろと
關戸 啓人

sekido.hiroto.7a@kyoto-u.ac.jp

Rの利用方法1

★ 自分のPCにインストールする方法

★ R本体
  ★ https://cran.r-project.org/
  ★ Windowsでユーザー名が日本語だと動かないことがあるなど，日本語環境に弱いので注意
★ R Studio（必須ではないがおすすめ）
  ★ https://rstudio.com/products/rstudio/download/
  ★ インストールはRをインストールしてからが無難

★ ダウンロードはするがPCに痕跡を残さず利用するポータブル版R，RStudioもある

Rの利用方法2

★PCにインストールせずにブラウザから利用する

★RStudio Cloud（PCにインストールせずにブラウザから利用する）

- ★`https://rstudio.cloud/`
- ★Windowsでユーザー名が日本語の場合はこれを試しても良いかも
- ★ただし，動作がおかしいことがある気がする…
- ★更に，ただし，最近重くなった気がする…

Rの利用方法3

★PCにインストールせずにブラウザから利用する

★Google Colaboratory（PCにインストールせずにブラウザから利用する）

- ★https://colab.research.google.com/notebooks/welcome.ipynb?hl=ja

★通常はPythonを利用できるが，Rも使える（以下にアクセスして新規作成）

- ★https://colab.research.google.com/notebook#create=true&language=r
- ★ただし，R，RStudioと操作感などはかなり違う

R言語の特徴

★ データ解析に特化したプログラミング言語

★ フリーウェア．無償

★ オープンソフトウェア．個人が開発，改良できる

★ プログラミング言語で，自由にロジックが記述可能，汎用性が高い

　★ プログラミングに慣れてない人にはとっつきにくい

★ ソフトウェアR上で記述する．対話式

★ 行列計算，ベクトル計算は速い．そういう統計処理は高速

　★ それ以外のループや条件分岐などを自分で書くととても遅い

★ GUIである程度できるRコマンダーや，統合開発環境のRStudioなどもある

ドキュメント

★ お断り：この授業はWindows版のRのつもりで解説をします

★ Rで`help.start()`と打つとHTMLドキュメントが表示される

★ ヘルプにFAQやPDFのドキュメントがある

- ★ An Introduction to Rの日本語訳（ただし古い）もWebにある
- ★ `https://cran.r-project.org/doc/contrib/manuals-jp/R-intro-170.jp.pdf`

★ その他，検索エンジンで探せば色々と

プログラムの書き方1

★1行1行実行したい文を書く

★1行に複数の文を書きたいなら，；で区切る

★1行ごとに実行結果が表示される（表示されるべきものがあるなら）

★#からその行末まではコメントとして無視される

プログラムの書き方2

★ 複数行に渡ってプログラムを書きたい場合

```
for(i in 1:100){
  if(i %% 2 == 1){
    print(i)
  }
}
```

(1から100までの奇数を出力するプログラム，こう書くべきではない)

★ 括弧の対応が取れてないなど続きがないと完結しない状況で次の行に行くと，前の行の続きとみなされる

★ 行の最初が+から始まる

★ ただし，次の行に行くと前の行は修正不可能

★ 複雑なプログラムを書くときは非常に困る

★ スクリプト（拡張子.R）を作って読み込ませるなど

★ 打ち間違えなどで前の行の続きとみなされている状態を抜けたいならEscキーを押す

Rに関するファイルについて

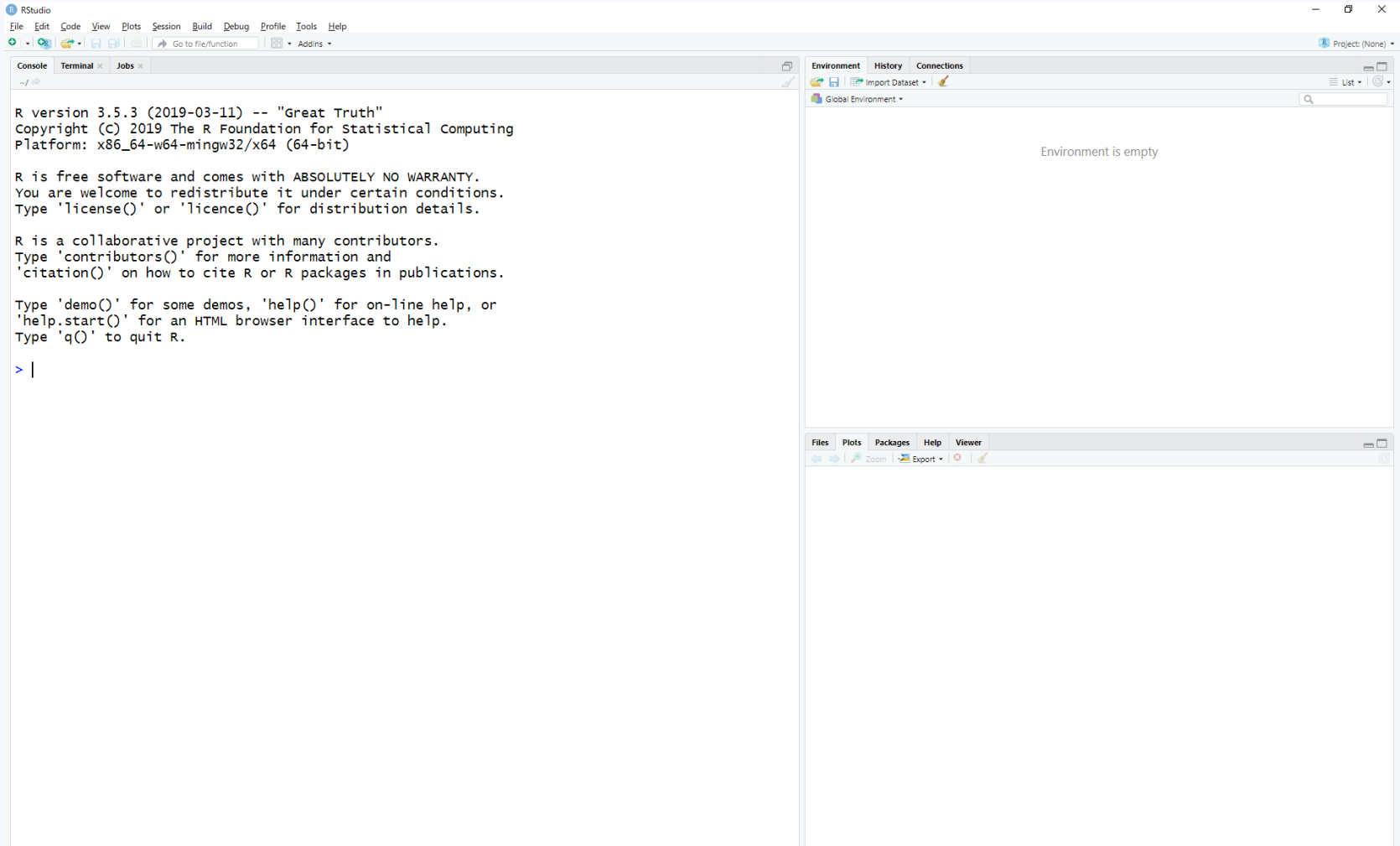
- **filename.R（スクリプト）：Rのプログラムを記述**
  - 書いたスクリプトを一気に実行する場合は source("filename.R")
  - sourceで実行する場合，明示的にprintで出力した結果以外は出力されない
- **filename.RData，filename.Rda：複数（全て）の変数（オブジェクト）の値の保存**
  - 終了時に全変数を ~/.RData に保存，次回起動時に変数の内容を復元してくれる（かも）
  - 全変数を保存：save.image("filename.RData")
  - 変数 x と y を保存：save(x,y,file="filename.RData")
  - 読み込み：load(file="filename.RData")
- **filename.Rds：1つの変数（オブジェクト）の値の保存**
  - 変数 x を保存：saveRDS(x, "filename.Rds")
  - 読み込んで y に代入：y <- readRDS("filename.Rds")

ファイルの指定方法

- ★ 作業フォルダにあるファイル filename についてはファイル名で良い
  - ★ `read.table("filename")`
- ★ 一般的には，絶対パスか作業フォルダからの相対パスで記述する
  - ★ `read.table("datdir/filename")`
  - ★ `read.table("C:/Users/admin/Documents/datdir/filename")`
  - ★ `read.table("C:\\Users\\admin\\Documents\\datdir\\filename")`
- ★ 作業フォルダの確認：`getwd()`
- ★ 作業フォルダの設定：`setwd("C:/Users/admin/Documents")`
- ★ ファイルをエクスローラーを使って選びたい：`read.table(file.choose())`

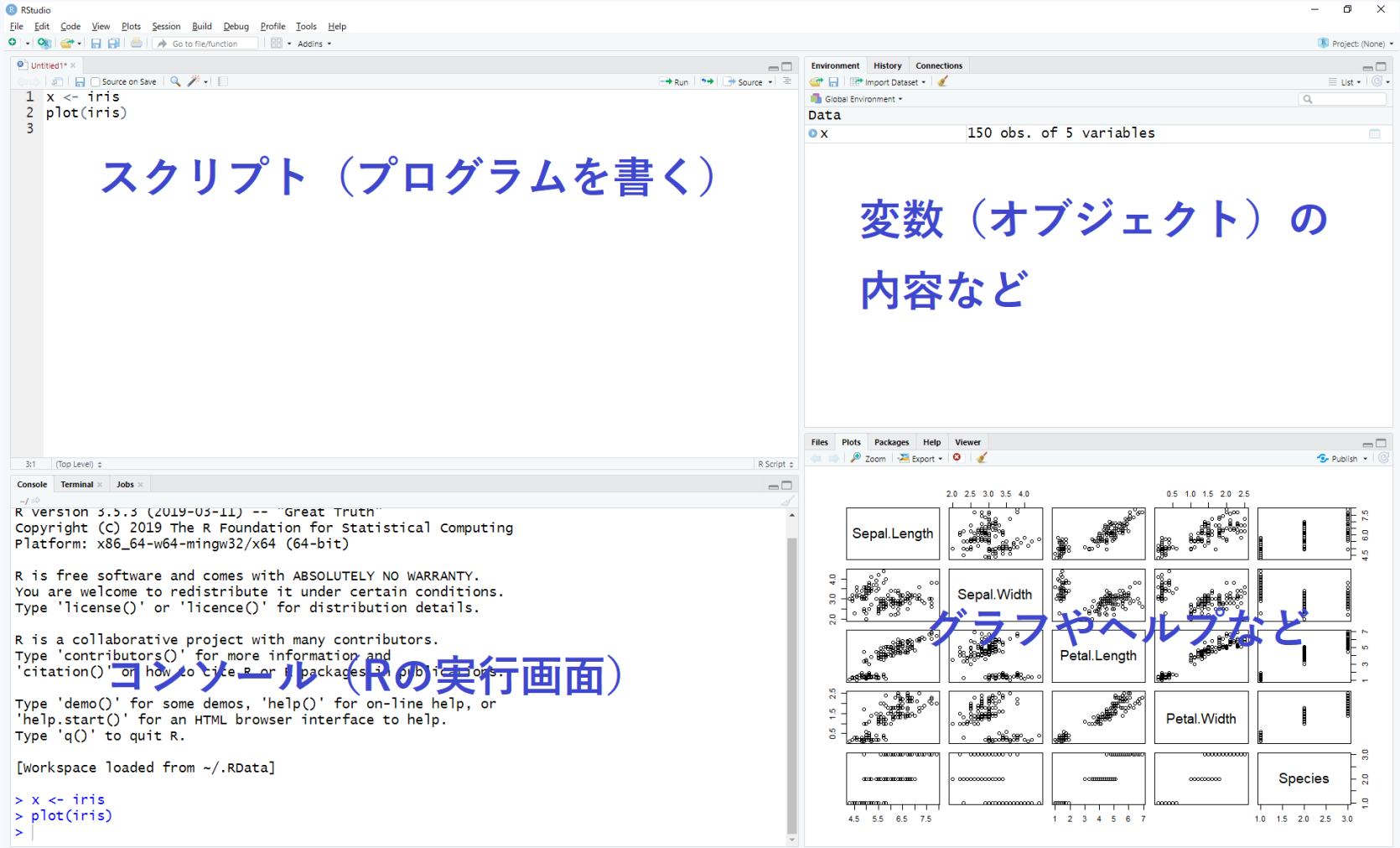
R Studioの画面

★起動時



R Studioの画面

★スクリプトを表示したとき



R Studioの場合

- ★左上のスクリプトを書き，そのスクリプトの内容を1行（ブロック？）ずつ実行することができる
  - ★実行したい行にカーソルを合わせRunを押す

変数

★ 変数は付値 (代入) されることによって定義される

- ★ 変数名は，アルファベット，数字からなり，最初の文字は数字であってはならない
- ★ 変数名は，大文字，小文字は区別される
- ★ また，以下の名前を使用することはできない：`break`, `else`, `for`, `function`, `if`, `in`, `next`, `repeat`, `return`, `while`, `TRUE`, `FALSE`.

★ 変数の代入には主には `x <- 1` という形式を用いる

- ★ `1 -> x` も同じ意味
- ★ `x = 1` もほぼ同じ意味
- ★ `x <<- 1` も似たような意味
- ★ `assign("x",1)` もほぼ同じような意味

★ 変数を削除するときは `rm(x)`

★ 定義されている変数の一覧 `objects()`

演算

- ★ +: 足し算
- ★ -: 引き算
- ★ *: 掛け算
- ★ /: 割り算
- ★ %/%: 整数の割り算
- ★ ** または ^: べき乗
- ★ %%: 余り (modulo)
- ★ %*%: 行列積
  - ★ 演算の順番は，普段の数学的なものとだいたい直感的に同じ順番で行われる
  - ★ 怪しいときは () を明示的につけると良い

ベクトル

★ c(2,4,7) などでベクトルを作れる

- ★ 3:7 は c(3,4,5,6,7) と同じ
- ★ 7:3 は c(7,6,5,4,3) と同じ
- ★ seq(3,7) は c(3,4,5,6,7) と同じ
- ★ seq(3,7,2) は c(3,5,7) と同じ

★ x <- 1:3; y <- 6:8 として

- ★ c(x,10,y) は c(1,2,3,10,6,7,8) と同じ

★ ベクトルの要素の置換は replace が使える

★ ベクトルの要素の挿入は append が使える

ベクトルの演算

★ ベクトルの演算は要素ごとに行われる

★ ベクトルの長さが違う場合は，短いベクトルが周期的に拡張されて適応

★ 2 などは長さ 1 のベクトルと思う (c(2) と同じ)

★ x[k] でベクトルの k 番目の要素

★ k がベクトルなら，x[k] が表すベクトルの i 番目の要素は，ベクトルの k[i] 番目の要素

★ x<-1:3 で x[c(TRUE,FALSE,TRUE)] で c(1,3) になる

  ★ TRUE に該当する要素のみ取り出される

★ x==2 で各要素が条件を満たすかどうかのベクトルが得られる

★ x[x%%2==0] でベクトル x の要素のうち偶数の要素のみを取り出すことができる

行列

★ 行列を作るには，
`matrix(ベクトル, nrow=行数, ncol=列数, byrow=TRUE)`
とすれば良い（ベクトルの長さと行数があれば列数はわかる，のように1つは省略可）

- ★ `byrow=TRUE` の場合は，1行1列，1行2列，1行3列の順番でベクトルの要素を埋める
- ★ `byrow=FALSE` の場合，もしくは指定しないと，1行1列，2行1列，3行1列の順番でベクトルの要素を埋める

★ 行列（ベクトル）`x, y` を並べて行列を作る：`rbind(x, y)`, `cbind(x, y)`

★ 行列 `x` に対して

- ★ `x[2,4]` で2行4列成分を取り出す
- ★ `x[2,]` で2行目からなるベクトルを取り出す
- ★ `x[,4]` で4列目からなるベクトルを取り出す
- ★ `apply(x, 1, mean)` で各行の平均からなるベクトルを求める
- ★ `apply(x, 2, mean)` で各列の平均からなるベクトルを求める

条件分岐

★ 以下のように条件分岐が書ける

```
if(条件文A){
  条件文Aが真のとき
}else if(条件文B){
  条件文Aが偽で，条件文Bが真のとき
}else if(条件文C){
  条件文A，Bが偽で，条件文Cが真のとき
}else{
  条件文A，B，Cが偽のとき
}
```

★ else if{}は何個あっても良い（0個でも良い）

★ else{}は省略可能

★ 最も単純な形は

```
if(条件文A){
  条件文Aが真のとき
}
```

★ ベクトル版のifelse関数もある

条件文の書き方1

★ 値の比較

- ★ A == B: AとBが等しいとき
- ★ A != B: AとBが等しくないとき
- ★ A > B: BよりAの方が大きいとき
- ★ A < B: AよりBの方が大きいとき
- ★ A >= B: AがB以上のとき（Aの方が大きいか等しい）
- ★ A <= B: AがB以下のとき（Aの方が小さいか等しい）

条件文の書き方2

★ 論理値 (スカラ) の演算

- ★ `A && B`: AとBの両方が真のとき
- ★ `A || B`: AとBの少なくても片方が真のとき

★ 論理値 (ベクトルも可) の演算

- ★ `A & B`: AとBの両方が真のとき
- ★ `A | B`: AとBの少なくても片方が真のとき
- ★ `!A`: Aが偽のとき

ループ

★ forループの書き方は以下の通り

```
for(i in v){
  # 変数iにベクトルvの各要素が1つずつ代入され実行される
}
```

★ repeatループ (無限ループ) の書き方は以下の通り

```
repeat{
  # 無限に実行される
}
```

★ whileループ の書き方は以下の通り

```
while(条件文){
  # 条件文が真である限り実行される
}
```

★ breakはループを終了させる (repeatループでは必須)

★ nextはループの1つのサイクルをここで終了させる

関数

★ 関数 (プログラムの塊) を作るには以下のようにします

```
関数名 <- function(引数){
  処理
  return(関数が返す値)
}
```

★ 変数のスコープ

- ★ 関数の中で新しく代入した変数は，関数内でのみ有効
- ★ 関数の外でも有効であってほしいなら，代入に«-を用いる

★ デフォルト引数

- ★ 引数のデフォルト値を指定するなら=を使う

read.table関数

★ ファイルからデータを読み込むにはread.table()関数が便利なことが多い

★ データを表 (データフレーム型) として読み込む関数

★ 最もシンプルな形式は

```
x <- read.table("hoge.txt")
```

★ hoge.txtに記述されている内容が，列に対しては半角スペース区切りでxに代入される

★ 色々オプションはあり，例えば，

```
x <- read.table("hoge.csv", header=TRUE, sep=",")
```

★ header=TRUE は1行目は列のラベルが書かれていることを意味する

★ sep="," は列の区切りがスペースではなくコンマであることを意味する

★ これは x <- read.csv("hoge.csv") と同値

リストとデータフレーム

★ ベクトルは同じ型（種類）の要素のみからなる

★ ベクトルの2次元版は行列

★ リストは色々な種類の要素が含まれていても良い

★ リストの2次元版はデータフレーム

★ データフレーム型の中身の詳細を知りたい場合は `str()` が便利

★ データフレーム型から行列に変換するには `data.matrix()` を用いる

★ 行列からデータフレーム型に変換するには `data.frame()` を用いる

★ 関数によって，データフレームの方が扱いやすい，行列の方が扱いやすい，等がある

ソート

★ 単純に値を小さい順に並び替えるなら sort 関数を使えば良いが，以下の order 関数の方が便利なことが多い

- ★ sort(c(5,2,9,3)) は c(2,3,5,9) になる

★ order(x) でベクトル x の各要素が k 番目に小さい要素が何番目の要素かを k が小さい順に並べたベクトルを返す

- ★ order(c(5,2,9,3)) は c(2,4,1,3)
- ★ y <- x[order(x)] でベクトル x をソートできる
- ★ y <- x[order(x[,3]),] で行列 x の行を3列目の値が小さい順にソートできる

attribute

★ オブジェクトにはattributeが付与されている場合がある

- ★ 例：データを標準化（平均0，分散1に）する scale 関数は元の平均値，分散を属性として付与する
- ★ 行列も，ベクトルに dim という属性を付与したもの

★ attr(x,"hoge") でオブジェクト x に hoge という名前でつけられている属性を表す（取り出したり代入ができる）

★ attributes(x) で（mode と length 以外の）属性を全て表示する