政治学方法論 I

第 3 回：再現性のある研究の実施法

矢内 勇生

神戸大学　法学部/法学研究科

2014 年 10 月 15 日

今日の内容

1 再現性のある研究
- 再現性 (reproducibility)
- 再現性の確保

2 R で再現性のある研究を実施する
- 記録！　記録！　記録！
- 可読性 (readability) の高いコーディング
- 文芸的プログラミング入門

再現性のある研究とは何か

再現性のある研究 (reproducible research)

科学的主張（特に、この授業ではデータ分析の結果）を研究成果として公表する際、他人が研究の内容を検証できるよう、データや分析に利用したコンピュータコードを一緒に公開するような研究

- 検証に晒されることによって、手続きの正しさが守られ易い
- 言葉では説明が難しいような研究の手続きも、実際に体験することで理解できる
- 研究内容をより明快に、深く理解できる

2 つの再現性

- 研究で求められる再現性
  1. 著者以外が、同じデータ、同じ手続きを利用して、著者と同じ結果を再現できる
  2. 異なるデータに同じ手続きを適用し、同じ（同様の）結果を再現できる
- 目標は 1、2 ともに確保すること
- 2 が確保できないものは、反証される（通常科学）
- 1 が確保できないものは、科学の作法に反する（非科学）
- ただし、複雑系などの研究からは異論も

再現性の確保に求められるもの

- 研究過程の詳細な記録
- 記録のうち、研究の再現に十分な部分の公開
- 使用したデータの公開
- 使用したコード（R のスクリプトなど）の公開
  - 他人が読んでも理解できる可読性の高いコード
  - 十分なドキュメンテーション

再現性を確保するメリット

- 作業の洗練化
- 共同研究を促進
- 引用が増える（データを公開した場合特に）
- 知の蓄積の促進
- 研究コミュニティへのサービス（→ 評判）

→ 自分のためになることが多い

研究過程の記録

研究過程は**すべて記録せよ!**

- どうやってデータを集めたか
- データセットをどうやって作ったか
- データをどうやって分析したか
- 分析結果をどう解釈したか（なぜそのように考えたのか）
- それぞれの日付

RStudio のプロジェクト機能を利用する

- １つのプロジェクト（論文）に使うファイルはまとめて管理する
- R を使うなら、RStudio の Project を利用するのが便利
- Project の作り方：File → New Project
  - 既存のフォルダに Project を使うとき：Existing Directory を選択
  - 新規フォルダを作成するとき：New Directory を選択
- プロジェクトにわかり易い名前をつける：拡張子は “.Rproj”
- Project を使うと、Project のあるフォルダが自動的に作業ディレクトリに指定される：setwd() を使う必要がない

R スクリプトの書き方：RStudio の場合

- プロジェクトを指定する：File → Open Project
- 新しいスクリプトを作成する：File → New File → R Script
- 名前をつけて保存する：R スクリプトの拡張子は ".R"
- 中身はテキスト：テキストエディタで開ける
- コメントは "#" を使って加える
- ファイルの先頭に、ファイルの説明を書く
- ctrl（または cmd）+ enter で現在行のコードを実行できる

R スクリプトに書き込むべき内容

- ファイル名
- フォルダ名
- R スクリプトの目的
- 入力ファイルと出力ファイル（ない場合は省略）
- 作成日と作成者
- 修正日と修正者（可能なら、version control を使うほうがよい）
- コードに対するコメント

記録！ 記録！ 記録！

R スクリプトの例

```
##############################################
##  example.R
##  wd: ~/classes/rm1/
##  Purpose: R スクリプトの書き方を説明する
##  Datasets used:
##      data/fake-data-01.csv
##      data/fake-data-02.dta
##  Created: 10/14/2014 Yuki Yanai
##  Last Modified: 10/15/2014 YY
##############################################

##  作業ディレクトリを指定する
setwd("~/classes/rm1/")

##  作図のために ggplot2 パッケージを読み込む
require(ggplot2)
```

R スクリプトを書く際に考えるべきこと

- 読み易いか：適切なスペース、改行、字下げ（ブロック化）
  実行速度を多少犠牲にしても、読みやすい・理解し易いコードを書くべき（再現性を重視する場合。製品を作る場合はまた別）
- 変数の名付け方に一貫性があるか：（例）linear.model or linearModel
- 来週, 来月, 来年, 5 年後, . . . に読んでも理解できるか
- 他人が読んでも理解できるか
- コメントが少な過ぎないか（多すぎるのは問題ない！）
  目安：コード全体の 30〜70%はコメント
- 他の言語（Stata の do ファイルなど）でも考え方は同じ

R スクリプトのメリット・デメリット

メリット

- そのまま R で分析できる

```
## スクリプト全体を一気に実行する
source("example.R")
```

- ファイルを作るのが簡単

デメリット

- 長い説明を書くのに向いていない
- 文書と分析結果（図や表など）を一緒に見ることができない
- コードのハイライト（色づけ）以外はただのテキストファイル

可読性の高いコードを書く！

- コンピュータコードは正しく動けばよいというものではない
- 可読性の高いコードが良いコード（*ceteris paribus*）
  - コードの維持・改訂・再利用が容易になる
  - 共同研究の促進
  - 研究の透明性が高まる
- 可読性：読み易いかどうか

可読性を高めるポイント（1）：コメント

コメントをたくさん書く！

- 他人が書いたコードを読むとき、どんな情報を知りたいか考える
- 例：平均値を求める関数を定義する

```
calc.mean <- function(x){## 平均値を計算する関数
  ## 引数：x = 数値ベクトル
  ## 返り値: mean.x = の算術平均x

  n <- length(x)  ## ベクトルの長さを計算する
  sum.x <- sum(x) ## ベクトルに含まれる数値を合計する
  mean.x <- sum.x / n ## 合計を個数で割る

  return(mean.x)  ## 算術平均を返す
}
```

可読性を高めるポイント（2）：字下げによるブロック化

コードのまとまりがわかるように、字下げ (indent) してブロックを作る

- 悪い例

```
for(i in 1:n){
for(j in 1:k){
x[i, j] <- mean(rnorm(10))
}}
```

- 良い例

```
for(i in 1:n){   # 行列x の行に対するループ
  for(j in 1:k){   # 行列x の列に対するループ
    x[i, j] <- mean(rnorm(10))
  }
}
```

可読性を高めるポイント（3）：適切なスペース・改行

スペースと改行を適切に使う事が重要

- 悪い例

```
a<-(1+2)*4+5-8
plot(x,y,xlim=c(1,10),ylim=c(-5,5),xlab="x␣軸のラベル",ylab="y␣軸のラベル",main="図のタイトル")
```

- 良い例

```
a <- (1 + 2)*4 + 5 - 8
plot(x, y, xlim=c(1, 10), ylim=c(-5, 5),
     xlab="横軸のラベル", ylab="縦軸のラベル",
     main="図のタイトル")
```

文芸的プログラミングとは何か

文芸的プログラミング (literate programming)

自然言語（日本語, 英語など）によるコンピュータコードの説明と、実行されるコード（コンピュータ言語）をひとつの文章として記述するプログラミングのスタイル

- Donald Knuth（TEX の開発者）によって提唱された
- 1 つのファイルを書くことで、文書作成とデータ分析を同時に行える

RStudio を使った文芸的プログラミング

1. Project を指定する
2. File → New File → R Markdown
3. ファイル名を付けて保存する：拡張子は ".Rmd"
4. タイトル等の基本情報を書く
5. コードの説明や結果の解釈などは文章で書く：その際、マークダウンという記法を利用する
6. コードをコードチャンクの中に書き込む
7. Knit HTML ボタンを押して、HTML ファイルを作成する

R マークダウン：基本情報の書き方

まず、タイトルや著者などの基本情報を書き込む：基本情報の開始と終了は 3 つのハイフン（RStudio で新規 Rmd ファイルを作ると、必要な情報を入力するよう求められるので、指示どおり進めばよい）

```
---
title: "RStudio␣による文芸的プログラミング入門"
author: "矢内　勇生"
date: "10/15/2014"
output:
    html_document
    theme: united
    highlight: tango
    toc: true
---
```

R マークダウン：文章の書き方

通常通り文章を書けばよい

- 見出しは “#” で：“#” の数が少ないほど、見出しのレベルが上
- “*” または “_” で挟むとイタリック
- ‘**” または “__” で挟むと太字
- “***”’ または “___” で挟むと太字のイタリック
- 箇条書き
  - 番号なしは “-”（ハイフン）
  - 番号付きは “1.”, “2.” など
  - 入れ子にするときは字下げ
- リンク：[リンク先の名前](URL)
- 画像: ![画像の代わりに表示するもの](ファイル名)

R マークダウン：コードチャンクの書き方

- コードチャンクの開始：「```{r チャンク名, チャンクオプション }」
- コードチャンクの終了：「```」
- 上の 2 つの間に R のコードを書く
- チャンク名はユニークにする必要がある
- チャンクオプションは、必要な場合のみ指定する

R マークダウン：文章の中（コードチャンクの外）にコードを書く

- 説明のためにコード自体を見せたいとき：「‘」で挟む：（例）‘sessionInfo()‘
- 文章の中に、コードの実行結果を載せたいとき
  - （入力例）この変数の分散は ‘r var(x)‘ です
  - （出力例）この変数の分散は 30.8 です

来週の内容

- 分析結果の提示法
- 図表の作り方
- ggplot2 の使い方
- etc.