学術研究のためのオープンソース・ソフトウェア(3)RとRStudio

MIYAZAKI, Kenji / 宮崎, 憲治

(出版者 / Publisher)
法政大学経済学部学会
(雑誌名 / Journal or Publication Title)
経済志林 / The Hosei University Economic Review
(巻 / Volume)
83
(号 / Number)
3
(開始ページ / Start Page)
143
(終了ページ / End Page)
178
(発行年 / Year)
2016-02-26
(URL)
https://doi.org/10.15002/00012756

【研究ノート】

学術研究のためのオープンソース・ソフトウェア（3）：RとRStudio

宮　崎　憲　治

1　はじめに

経済学において，実証論文には必ず統計分析が伴う。多くの研究者はEXCEL でデータを管理・整理し，図表を作成し，Stata といった統計ソフトを利用して，分析をしているであろう。分析結果は WORD で論文を書く場合，EXCEL で図表を作って，貼り付けたすることが多いだろう。

統計ソフトは Stata 以外には SAS があるだろう。また，Stata で標準的に装備していない最新の統計手法を自分で組むために，Matlab という数値計算ソフトを利用する場合もあるだろう。しかしながら EXCEL や Stata や Matlab はオープンソース・ソフトウェアでない。新しいバージョンで作ったデータが古いバージョンで動かないことがある。

こうした問題を避けるために，データはテキストファイルで保存しておくのが望ましい。テキストファルのデータ形式として，csv 形式がある。各行は改行で，各列はコンマで分離されている形式で，最初の行は変数名になっている。これでファイルを管理しておくのが EXCEL 形式より汎用性があり望ましい。

インターネットから入手できる多くのデータはこの csv 形式で入手可能であるが，EXCEL でしかデータ入手可能でないことがある。そうしたフ

ァイルも csv 形式で保存しておいたほうがよい。ただ，csv 形式で保存しようとすると，何度も確認されることがある。また，数値データを桁でコンマを入れる，ある意味おせっかいな機能のため，数値として読み込んでくれないので，注意が必要である。また，EXCEL 図表を作ってもそれをコピーペーストするとミスする可能性がある。それを防ぐためには計算と同時に latex 形式に対応した図表を作成するほうが望ましい。そうした観点も含め，フリーのプログラミング言語として R が注目されている。

この研究ノートでは学術研究のための R について紹介する。以下，R およびデファクトスタンダードになりつつある開発統合環境の RStudio について紹介し，その基本的な使い方として，データ入出力，データ加工,データ分析の仕方について解説する。また latex 形式に出力することを念頭におきながら，R で図表の扱い方を言及する。

2　R

R（あーる）はオープンソース・フリーソフトウェアの統計解析向けのプログラミング言語である。ニュージーランドのオークランド大学の Ross Ihaka と Robert Clifford Gentleman により1986 年に作られた。現在では R Development Core Team によりメンテナンスと拡張がなされている。R は文法的には，統計解析部分は AT&T ベル研究所が開発した S 言語を参考としており，またデータ処理部分は Scheme の影響を受けている。

比較的歴史がある言語であるが，ここ数年で爆発的に人気がでてきた。その理由はいくつかあるが，そのひとつとして Hadley Wickham が devtools という開発を簡単に作れるパッケージを作り多くの便利なパッケージが登場したことによることが大きいだろう。彼自身を多くの使い易いパッケージを開発している[*1]。また, Rでの開発統合環境として RStudio が

[*1] Hadley Wickham がどのようなパッケージを開発してきたかはhttps://github.com/hadley?tab=repositories を見ていただきたい。

2010年12月ごろに登場したことも人気のきっかけになったと考えられる。私自身も RStudio の登場により，本格的に R を使うようになった。

R をインストールするには

```
https://cran.r-project.org/mirrors.html
```

に行き一番近いサーバーを選択し，対応機種のアプリをダウンロードして，実行すればよい。

2.1　実行

R のアイコンをダブルクリックするか，ターミナル上でR と入力すると R は起動する。最初は対話モードとして R コンソールが開き，そこでコマンドを入力するとその結果が出力される。終了するためには，R コンソール上から `q()` とする。

またある程度まとまったことを実行するには，拡張子 R のスクリプトを書いて実行する。実行はターミナル上で

```
R CMD BATCH my_script.R
```

とするか，R を立ち上げてから，R コンソール上で

```
source("test.R")
```

と実行する。

2.2　スクリプトファイル

スクリプトファイルについて，後述のRStudio を導入していないのならこのように記述することを勧める。

```
rm(list = ls())
gc()
setwd('~/Dropbox/project/AAA')
```

一行目はすべての変数を初期化し，二行目はガーベージコレクションを実施し，三行目はこのファイルが存在しているディレクトリを記述している。これをワーキングディレクトリという。~/ はホームディレクトリで mac の場合，/Users/（ユーザー名）/ を表していて，Windows の場合，たとえば C:\Users\（ユーザー名）\を表す。（\を￥と置き換える。）Windows でのディレクトリを示す \ もしくは￥は，/ に変更しなければならない。

ワークングディレクトリがずれているとファイルの読み込みがうまくいかない。どこがワーキングディレクトリなのかを知りたい場合，`getwd()` とする。なお RStudio でプロジェクトとして指定すれば，ワーキングディレクトリが自動的に設定されることになる。

2.3 packages

R は基本的な機能だけでなく，パッケージを追加することでより高度な統計分析が可能である。こうしたパッケージを多くの人が開発して無償で公開し，CRAN（Comprehensive R Archice Network）に登録している。パッケージの導入方法は，

```
install.packages('パッケージ名')
```

を一度実行する。一旦導入すれば，利用するたびに

```
library(パッケージ名)
```

もしくは

```
require(パッケージ名)
```

をスクリプトファイルに書くか，R コンソール上で実行すればよい。`require` のほうが望ましいとされているが，私は昔からの癖で `library` を使ってしまう。

3　RStudio

3.1　RStudio

RStudio は R を実行するための統合環境の一つである。非常に良く出来ていて，現状では R を実行するために必ず入れるべき環境である。ただ現時点ではメニューが日本語に対応していない。また日本語入力も不可能ではないが必ずしも使い心地はよくない。

RStudio の特徴を列挙する:

- エディタと対話モードを自由に行き来できる。
- ソースコードが色分けされて表示できる。
- コマンド履歴をソースコードに付け加えることができる。
- 実行中の変数の中身が簡単に見れる。
- コマンド補完だけでなく，該当のコマンドのヘルプが見れる。
- データの入力やグラフの出力がやりやすくなっている。
- Sweave や knitr に対応している。
- プレゼン用のスライドを作成してくれる。
- SVN や Git などバージョン管理が可能である。
- Html5 をもちいたウェブアプリを簡単に作成してくれる。
- デバック機能もっている.

RStudio の最大の特徴は project 単位でファイル管理することである。この機能を使用すると，ワーキングディレクトリが指定され，そこで R ソース，出力結果などを自然とファイル単位で管理するようになる。あとでスクリプトを変更したりする際に操作が楽になる。また Git などのバージョン管理が可能になる。バージョン管理については別の機会で詳述したい。

3.2 インストールと設定

RStudio をインストールするには

```
http://www.rstudio.com/
```

に行き対応機種のアプリをダウンロードして，実行すればよい。ubuntu を利用しているならサーバー版を用いることができる。

設定は通常デフォルトのままでよいが，もし Windows で実行しているのなら，［Tools］→［Global Options］を実行して，［General］にある［Default Text encodings］の箇所を `UTF-8` とする。

3.3 使い方

使い方はホームページのドキュメンテーションをみればよい。

```
https://support.rstudio.com/hc/en-us/sections/200107586-
Using-RStudio
```

私の使い方は以下である。スクリプトファイルを書きながら，Ctrl+Enter（Mac の場合 CMD+Return）で書いたスクリプトを実行して評価していく。逆に，プロンプトでいくつか試行錯誤しつつ，対話式に実行しながら，有用なファイルを履歴を参照した上で，スクリプトファイルに付け加える。実行後，どのような変数にどのような変数が加わったのかを簡単に確認しながら，プログラムを作成していく。また必要に応じてヘルプを参照していく。これらを繰り返しながらスクリプトファイルを適宜保存しながら作っていく。

さらに実行はソースコードからノートブック・ボタンを押せば，実行結果を html ファイルに自動的に作成してくる。これは knitr とよばれるライブラリを使用している。knitr については別の機会で詳述したい。

4 データ入出力

R にデータを読み込ませ方として，手入力やコピペの他に，外部ファイルからの入力もしくはインターネットからの入手する方法もある。その際

にデータフレームというデータ形式が基本となる。

4.1　データフレーム

R では通常データ分析はデータフレームでおこなわれる。データフレームは同じ長さのベクトルを集めたものである。以下のようにして作成する。

```
x <- rnorm(3)
y <- runif(3)
df <- data.frame(x,y)
```

x は標準正規分布，y は一様分布にしたがう確率変数をそれぞれ 3 つ発生させている。

なおデータフレームの平均や分散は次のようにする必要がある。

```
mean(df$x)
mean(df$y)
var(df$x)
var(df$y)
```

データフレーム名と変数名の間に $ が必要である。

観測値をひとつ追加するは次のようにする。

```
df1 <- rbind(df,c(3,3))
```

`df1` でなく，`df` にすればもとのデータフレームが更新される。

別の変数を追加するには次のようにする。

```
z <- 1:4
df2 <- cbind(df,z)
```

もしくは，

```
df2 <- df
df2$z <- 1:4
```

とする。df2 でなく，df にすればもとのデータフレームが更新され，後者のやりかただと一行で済む。

既存の変数を加工して新たな変数を付け加えるには次のようにすればよい。

```
df3 <- df
df3$w <- df3$x/df3$y
```

もしくは

```
df3 <- df
df3$w <- with(df3,x/y)
```

とする。

変数部分を抜き出すには subset を用いればよい.

```
x <- rnorm (100)
y <- rbinom (100,1,0.5)
df <- data.frame(x,y)
subdf <- subset(df,y==1)
```

x は標準正規分布，y はベルヌーイ分布にしたがう確率変数をそれぞれ 100 個発生させている。とすれば，y が 1 となるデータが抜き出される。

なお，R にはすでにデータが内蔵している。

```
data()
```

で内蔵しているデータを確認できる。リストのなかに例えば cars という名前のデータがあることが確認できる。そして

```
help(cars)
```

とすれば，そのデータについて変数の定義などがわかる。speed と dist という変数があることがわかる。なおデータはすでに使える状態であるが，データの読み込みを明示したければ，

```
data(cars)
```

とすればよい。ライブラリを追加するとデータセットが増える。

データフレームで初学者がよく間違えることは，データフレームの平均値を導出するときにデータフレーム名を指定せずに変数名だけで実施してしまうことである。たとえば mean(cars$speed) とすべきところを mean(speed) としてしまう。事前に attach(cars) とすればよいと昔の R の本には書いてある。ただこうすると Grolemund (2015) など最近の本に述べられているように，R の名前空間がおかしくなる危険があるので推奨されない。Google's R Style Guide (https://google-styleguide.googlecode.com/svn/trunk/Rguide.xml#attach) にも

> The possibilities for creating errors when using attach are numerous. Avoid it.

とある。

なおデータ整理については，tidyr (https://github.com/hadley/tidyr), dplyr (https://github.com/hadley/dplyr) といったパッケージを使えば，より簡単に直感的に扱える。これについて

> https://www.rstudio.com/resources/cheatsheets/

にあるチートシートがよくまとまっている。

4.2 ファクター

R の最大の特徴はファクター（factor）である。因子とも呼ばれている。ファクターはカテゴリーを要素としたベクトルである。これによりカテゴリカルデータを R で簡単に扱える。また，複数のカテゴリーをダミー変数を表現する場合，カテゴリー数から 1 引いた数だけのダミー変数が必要であるが，ファクターだと一つで済む。R では後述する回帰分析でもファクターのまま扱える。作り方は次のようにする。

```
size <-sample(c("H","M","L"),10,replace=T)
fc<-factor(size)
```

またこれに順序もつけることができる。

多くの統計データを外部ファイルから読み込む場合，ダミー変数のままのことが多い。たとえば

```
http://wps.aw.com/wps/media/objects/11422/11696965/
empirical/empex_tb/fertility.xlsx
```

にある `fertility.xlsx` のなかのダミー変数（`black,hispa,othrace`）をファクター型（`race`）への変換するには次のようにする。

```
library(readxl)
df <- read_EXCEL("fertility.xlsx")
df$race <- "white"
df$race[df$black ==1] <- "black"
df$race[df$hispan ==1] <- "hispanic"
df$race[df$othrace ==1] <- "others"
df$race <- as.factor(df$race)
table(df$race)
```

最初の 2 行は EXCEL ファイルを読み込むコマンドであるが, EXCEL の読み込みについては後述する。

逆にファクター型をダミーに変換するは次のように実行する。

```
seed(10)
x <- rnorm (10)
size <-sample(c("H","M","L"),10,replace=T)
df <- data.frame(x,size)
df$dummy1 <- ifelse(df$size=="H" ,1,0)
df$dummy2 <- ifelse(df$size=="M" ,1,0)
head(df)
```

一行目は乱数の再現性のためのコマンドである。

結果は以下のようになる。

```
       x       size    dummy1    dummy2
1  0.8865824   H        1         0
2  0.2675922   H        1         0
3 -0.6596697   H        1         0
4 -0.1201818   M        0         1
5 -0.4707522   M        0         1
6 -0.2218272   H        1         0
```

ところで，文字列から data.frame よりデータフレームを作成すると，自動的にファクターに変換されてしまう。時には文字列のままのほうが望ましいだろう。そうしたいときは stringsAsFactors = FALSE というオプションをつけるか，読み込んだ後に as.character を用いる。なおパッケージ dplyr の data_frame を用いると，文字列は文字列のままデータフレームを作成することができる。

4.3 csvファイル

データ分析の多くの場合, 外部からのデータを読み込ませることになる。データは csv ファイルで扱うのが標準的である。csv ファイルとは，コンマで分離されているテキストデータのことである。たとえば，以下のように書かれている。

```
var1, var2, var3
 3,   40,   4
 1,   30,   4
...
```

一行目は変数名が記述され，二行目以降に数値が含まれている。

読み込むためには次のようにする。

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

そうすると，データフレーム df が作られる。

もし csv ファイルに文字列が含まれるとき，このままだと文字列がファクター型に指定されてしまう。ファクター型についは次小節で説明する。ファクター型の変換を避けるためには

```
df <- read.csv("data.csv",stringsAsFactors= FALSE)
```

とする必要がある。

またデータフレーム df をファイル data.csv に書き込むには次のようにする。

```
write.csv(df,"data.csv",row.names = FALSE)
```

オプションをつけないと，列のインデックスも書き込まれてしまう。

標準の csv 入出力は速度が遅いといわれている。readr (https://

github.com/hadley/readr) というライブラリを導入すれば改善される。

```
library(readr)
df <- read_csv("data.csv")
```

と標準のコマンドの“.”が“_”に変更している。オプションを付けなくても文字列をファクターに変換することなく読み込んでくれる。

なお書き込むには

```
write_csv(df,"data.csv")
```

とする。オプションを付けなくても列のインデックスは無視してくれる。

4.4　EXCELファイル

多くの人はEXCEL でデータ管理しているかもしれない。EXCEL のデータを入力するには，色々なパッケージがあるが，

http://oku.edu.mie-u.ac.jp/~okumura/stat/EXCELdata.html

をみる限り，現在のオススメは readxl (https://github.com/hadley/readxl) である。

```
library(readxl)
df <- read_excel("datafile.xlsx",1)
```

とすればよい。最後の引数の 1 は一枚目のシートを意味する。

特定の名前，たとえば“Revenues”というシートを用いるのなら，

```
df <- read_xlsx("datafile.xlsx",sheet="Revenues")
```

とすればよい。

なお EXCEL ファイルが R に読み込まれやすいように多少整形しておくほうがよいだろう。最初の行に変数名を書くようにして，生データ以外

の情報は別のシートに移しておくなどしておくとよい。

EXCEL のデータを入力するための他の方法として，csv 形式に変換して読みこませればよい。ただし変換のさい 2 点ほど注意することがある。まず EXCEL で桁をしめすコンマのまま保存すると文字列として保存されてしまう。そうしたファイルをread.csv で読み込む数値でなく文字列として読み込まれてしまう。それを避けるには，桁を示すコンマを無しに直して保存しなければならない。なお `read_csv` をもちいるなら数値として読み込んでくれる。

またデータの名前を日本語を英語にしておいたほうがトラブルが少ない。日本語が含まれている場合に，EXCEL 上で csv 形式に変換して保存したときにシフトJIS で保存されてしまう。Linux や Mac では文字コードをUTF-8 にしていて，Windows でも Rstudio を導入したときに最初に文字コードをUTF-8 に設定していたので，そのままの読み込みだと日本語が文字化けになる。それを防ぐためには，`read.csv` を用いるときは

```
df <- read.csv("data.csv",stringAsFactors=FALSE,fileEncoding="SJIS")
```

とし，`read_csv` を用いるときは

```
df <- read_csv("data.csv",locale=locale(encoding = "SJIS"))
```

とする。

4.5　その他のファイル形式

Stata や SAS や SPSS などほかの統計パッケージのファイル形式から読み込むには `haven (https://github.com/hadley/haven)` というパッケージを導入するとよい。たとえば Stata の場合以下のように実行する。

```
library(haven)
df <- read_dta("datafile.dta")
```

昔の本で foreign というライブラリを用いればよいと書いてある場合があるが，foreign は執筆時点では Stata 12 までの保存形式しか対応していない。ただ，私の経験上バージョン12 以下の保存形式だと haven では上手く読み込めないが foreign だと上手くいく場合があった。適宜使い分ければよいだろう。

5　データ分析

R は様々な統計分析ができる。計量経済学で主につかわれるのが回帰分析であり，これを中心に説明していく。

5.1　lm

線形回帰分析は lm というコマンドを使えばよい。

```
data(cars)
fm<-lm(dist~speed,data=cars)
summary(fm)
```

結果は以下になる。

```
Call:
lm(formula = dist ~ speed,data = cars)
Residuals:
   Min      1Q  Median    3Q     Max
-29.069  -9.525  -2.272  9.215  43.201
Coefficients:
            Estimate Std.    Error t value   Pr(>|t|)
(Intercept)  -17.5791   6.7584  -2.601     0.0123 *
speed          3.9324   0.4155   9.464   1.49e-12 ***
```

```
--
Signif.codes:0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘,’ 0.1 ‘ ’ 1
Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared: 0.6511, Adjusted R-squared: 0.6438
F-statistic: 89.57 on 1 and 48 DF, p-value: 1.49e-12
```

dist~speed の左側が被説明変数で，右側が説明変数である。重回帰も分析可能で，例えば x1+x2+x3 とすれば，3つの変数の重回帰が実施可能である。

残差は resid(fm)，予測値は fitted(fm)，共分散行列は vcov(fm) でとることができる。また，残差の標準誤差は summary(fm)$sigma で，自由度は，summary(fm)$df[2] (fm$df.residual) で，決定係数はsummary(fm)$r.squared で，調整済み決定係数は summary(fm)$adj.r.squared でとることができる。

なお二乗項を付けた回帰モデルを実行するには次のコマンドを実行する。

```
fm2 <-lm(dist~speed + I(speed^2),data=cars)
```

$I(.)$ をつけないと正しく実行してくれない。もしくは

```
fm2 <-lm(dist~poly(speed,degree=2,raw=TRUE),data=cars)
```

とする。raw=TRUE としないと正しく実行してくれない。2 変数以上の多項式は polym を用いる。これはホワイトの分散不均一の検定を実行する際に重宝するだろう。

なお複数のモデルを比較する分散分析は anova(fm,fm2) とすればよい。この結果だと以下となり，有意ではない。

```
Analysis of Variance Table
Model 1: dist ~ speed
Model 2: dist ~ speed + I(speed^2)
```

```
  Res.Df RSS Df Sum of Sq F Pr(>F)
1  48 11354
2  47 10825 1 528.81 2.296 0.1364
```

なおこのエフ統計量は，モデル 2 の自乗項についてのティー値の自乗と等しいことに注意されたい。

回帰式にくわえる変数としてファクターを付け加えても線形回帰分析を実行してくれる。ファクターの要素が 2 つのときにはどちらかが 1 をとるダミー変数となる。ファクターの要素が 3 つ以上のときには要素の数から 1 減じた種類のダミー変数を付け加えた回帰式を実行してくれる。

また glm を使えば，ロジットやプロビットモデルの推計や，ポアソン回帰モデルの推計が可能である。詳しくはマニュアルを参照されたい。

5.2　AER

ライブラリ AER をもちいれば，よりさまざまな統計分析が可能になる。Applied Econometrics with R の略である。応用計量経済学に必要な統計パッケージがひと通りまとめられている。横断面データでの回帰分析で実施されるロバスト分散が可能である。たとえば以下のようにすれば，ロバスト分散を用いた推計ができる。

```
library(AER)
data(cars)
fm<-lm(dist~speed,data=cars)
coeftest(fm,vcov=vcovHC)
```

```
t test of coefficients:
             Estimate Std.    Error  t value  Pr(>|t|)
(Intercept)  -17.57909  5.93180  -2.9635  0.004722 **
```

```
speed           3.93241  0.42754   9.1978 3.636e-12 ***
---
Signif.codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Stata のロバスト分散と結果を合わせるには `vcov=vcovHC(fm,type="HC1")` とすればよい。また複数のモデルの比較について，ロバスト分散を使うには `waldtest` を用いればよい。

AER を用いれば，トービットや二段階最小二乗法も推計できる。この作者たちが解説した Kleiber and Zeileis（2008）の一読をすすめる。

たとえば有名な Fair（1978）の「浮気」データを用いた分析は以下のようにして実行される。

```
library(AER)
fm_ols <- lm(affairs ~ age + yearsmarried + religiousness +
  occupation + rating,data = Affairs)
fm_logit <- glm(I(affairs > 0) ~ age + yearsmarried +
  religiousness + occupation + rating,data = Affairs,family
  =binomial)
fm_probit <- glm(I(affairs > 0) ~ age + yearsmarried +
  religiousness + occupation + rating,data = Affairs,family
  = binomial(link = "probit"))
fm_tobit <- tobit(affairs ~ age + yearsmarried +
  religiousness + occupation + rating,data = Affairs)
```

それをまとめたのは表 1 である。この表は後述する `stargazer` によって作成される。

他にも計量分析の拡張するライブラリとしては，ラグ付きデータの扱いが簡単になる `dynlm`，非定常分析の各種検定は `urca`，VAR（多変量自己回帰）モデル作成のための `vars`, GMM（一般化モーメント法）が可能になる

gmm，パネル分析が可能な plm などがある。

R は多くの人がパッケージを開発しているために同じような機能をもつパッケージが複数存在してわかりにくいことがある。またそれぞれのパッケージを利用しようとすると，操作の方法が異なって戸惑うことがある。そうしたことを解消しようと Zelig というパッケージが現在注目をあつめている。

6　図

R は図の作成が簡単にできる。例えば正規分布を表示するには次のようにすればよい。

```
curve(dnorm(x),xlim=c(-3,3),main="Normal Distribution")
```

図 1 （a）のように表示される。また散布図は次のようにする。

```
plot(dist~speed,data=cars)
```

散布図に回帰直線を付け加えるには次のようにする。

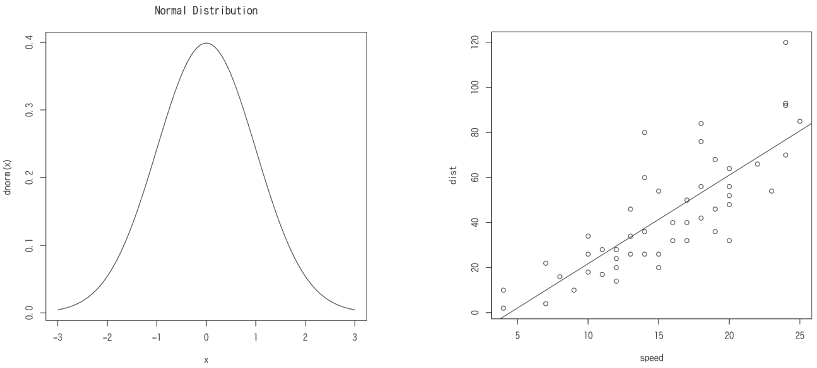
```
fm <- lm(dist~speed,data=cars)
plot(dist~speed,data=cars)
abline(fm)
```

図 1 （b）のように表示される。

また直感的にあつかえるパッケージとして ggplot2 (https://github.com/hadley/ggplot2) が注目されている。ウィッカム（2012）で作者自身が解説本を執筆している。ggplot2 が人気になるまで lattice がよく使われていた。このパッケージは mosaic という統計教育のためのパッケージ群に含まれており，まだまだ便利である。lattice についてショーカー（2012）で作者自身が解説本を執筆している。

	Dependent variable:			
	affairs	I(affairs>0)		affairs
	OLS	*normal*	*probit*	*Tobit*
	(1)	(2)	(3)	(4)
age	−0.050**	−0.007**	−0.022**	−0.179**
	(0.022)	(0.003)	(0.010)	(0.079)
yearsmarried	0.162***	0.018***	0.060***	0.554***
	(0.037)	(0.005)	(0.017)	(0.135)
religiousness	−0.476***	−0.054***	−0.184***	−1.686***
	(0.111)	(0.015)	(0.052)	(0.404)
occupation	0.106	0.012	0.038	0.326
	(0.071)	(0.009)	(0.033)	(0.254)
rating	−0.712***	−0.088***	−0.273***	−2.285***
	(0.118)	(0.016)	(0.053)	(0.408)
Constant	5.608***	0.788***	0.977***	8.174***
	(0.797)	(0.106)	(0.365)	(2.741)
Observations	601	601	601	601
R^2	0.131			
Adjusted R^2	0.124			
Log Likelihood		−318.335	−307.295	−705.576
Akaike Inf.Crit.		648.670	626.591	
Residual Std.Error	3.087(df=595)			
F Statistic	18.004***(df=5;595)			
Wald Test				67.707***(df=5)
Note:				*p<0.1;**p<0.05;***p<0.01

表 1 ：Fair（1978）のデータの分析比較



（a）正規分布　　（b）散布図

図 1 ：基本グラフィックス

6.1　日本語表記の図

なお日本語を表記したいときに，次のようにすればよい．

```
par(family="Japan1GothicBBB")
curve(dnorm(x), xlim=c(-3,3),main="正規分布")
```

Mac の場合 `par(family="HiraKakuProN-W3")` などとすればヒラギノフォントが利用できる。フォントの選択によっては警告がでる場合や出力されない場合がある。たとえば `IPA` フォントは警告がでるが出力されるようである。

`ggplot2` のとき日本語の表記させ方が基本グラフィックスと少し違うので注意が必要である。

```
library(ggplot2)
theme_set(theme_gray(base_family="IPAGothic"))
qplot(cars$speed,cars$dist,main="散布図")
```

さらに `lattice` などにも表記するには表示するためには

`https://oku.edu.mie-u.ac.jp/~okumura/stat/Rprofile.html`

にあるように，設定ファイル `.Rprofile` を付け加える必要がある。なお一行目は RStudio を用いているなら上記のファイルを以下に変更する必要がある。

```
options(repos=c(CRAN="https://cran.rstudio.com/"))
```

またこの設定だと図が `lattice` の図が白黒になることも注意されたい。

6.2　図の保存

図を保存するには次のようにすればよい。

```
pdf("fig0.pdf",width=7, height=5)
curve(dnorm(x),xlim=c(-3,3),main="Normal Distribution")
dev.off()
```

保存した PDF ファイルを latex に挿入するには次のようにすればよい。

```
\begin{figure}[ht]
\centering\caption{figtitle}\label{fig1}
\includegraphics{fig0.pdf}
\end{figure}
```

6.3 日本語表記の図の保存

日本語が含まれる図を作成保存するためには次のようにする。

```
pdf("fig1.pdf",width=7,height=5,family="Japan1GothicBBB")
curve(dnorm(x),xlim=c(-3,3),main="正規分布")
dev.off()
```

とする。

ただ先の例だとフォントが埋め込まれていない。フォントを埋め込むには次のように cairo ドライバを使用する。

```
cairo_pdf("fig2 -1.pdf",width=7,height=5)
par(family="IPAGothic")
curve(dnorm(x),xlim=c(-3,3),main="正規分布")
dev.off()
```

とする。fig2-1.pdf が保存される。

ggplot2 でフォントを埋め込むには次のようにする。

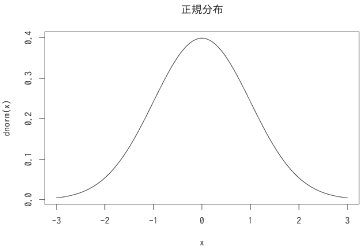
```
cairo_pdf("fig2 -2.pdf",width=7,height=5)
library(ggplot2)
theme_set(theme_gray(base_family="IPAGothic"))
qplot(speed,dist,data=cars,main="散布図")
dev.off()
```

とする。fig2-2.pdf が保存される。

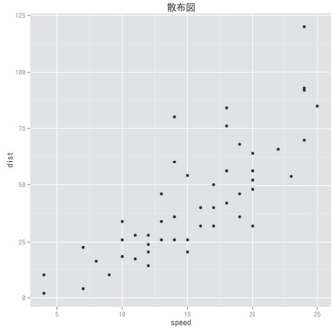
なお複数の図を並べて表示するには subfig というパッケージを用いる。

```
\begin{figure}[ht]
\subfloat[基本グラフィックス]{
\includegraphics[width =.45\linewidth]{fig2-1.pdf}
}
\subfloat[ggplot2]{
\includegraphics[width =.45\linewidth]{fig2-2.pdf}
}\ caption{2つを並べて作図}\label{fig:subplots}
\end{figure}
```

上記のようにすれば図 2 のように表示される。左が fig2-1.pdf で右が fig2-2.pdf である。



（a）基本グラフィックス



（b）ggplot2

図２：２つ並べて作図

lattice だと以上のやり方ではうまくいかない。先ほどの .Rprofile で lattice で日本語表示ができるようにしたあとで

https://sites.google.com/site/jibuhoshina/programs/pdf_in_r

にあるように

```
quartz(type="pdf",width=8,height=8,file="fig.pdf")
library(lattice)
xyplot(dist ~ speed,data=cars,main="散布図")
dev.off()
```

とする。ただこのやりかたは Mac 専用であり, 他の機種でどのようにするのか不明である。

6.4 tikzDevice

宮﨑（2015）で紹介したように latex 上で作図する方法として tikz がある。これに即した図を R で作成するには, tikzDevice (https://github.com/yihui/tikzDevice) を用いる。これによって latex のコマンドで数式などを綺麗に埋め込むことができる。例えば以下のようにして tikzfig.tex を作成する。

```
library(tikzDevice)
options(tikzDefaultEngine='xetex')
tikz("tikzfig.tex",width=7,height=5)
curve(dnorm(x),xlim=c(-3,3),main="$N(\\mu,\\sigma^2)$")
dev.off()
```

日本語をつかうために latex のエンジンを xetex とする。また latex の記号を入力する際にバックスラッシュを 2 つ重ねる必要があることに注意されたい。たとえ画面上で文字化けしていても，lattice で日本語表記の

図が作成可能である。ただ古い機種だとコンパイルにかなり時間がかかる。

先に作った tikzfig.tex を読み込む際は次のようにする。

```
\documentclass{bxjsarticle}
\usepackage{xltxtra}
\usepackage{zxjatype}
\usepackage[ipa]{zxjafont}
\usepackage{tikz}
\begin{document}
\begin{figure}[ht]
\centering\caption{figtitle}\label{fig1}
\input{tikzfig}
\end{figure}
\end{document}
```

latex 側の tex ファイルのプレアンブル部に\usepackage{tikz} を付け加えておく必要がある。なお日本語まじりの文章も実行中に警告が出るが出力可能である。

7　表

7.1　原則

表を R で作成し，それを読み込むようにすればよい。まず sink() を用いて tab1.tex を保存する。

```
sink("tab1.tex")
cat("& Estimate & Std.~error & $t$ statistic & $p$ value \\\\")
cat("\\hline")
```

	Estimate	Std.error	t statistic	p value
(Intercept)	-17.579	6.758	-2.601	0.012
speed	3.932	0.416	9.464	< 0.001

表２：Results

```
x <- summary(fm)$coefficients
x[] <- as.character(round(x,digits = 3))
x[,4] <- ifelse(as.numeric(x[,4]) < 0.001,"$<$ 0.001",x[,4])
cat(paste(rownames(x),"&",
apply(x,1,paste,collapse = " & "),"\\\\ \n"))
cat("\\hline")
cat("\\end{tabular}")
sink()
```

それを読み込むようにするには次のようにする。

```
\begin{table}[ht]
\centering
\input{tab1}
\caption{回帰分析結果}
\end{table}
```

そうすると，表２のようになる。

ただ，いささか面倒なのである程度自動化したパッケージをいくつか紹介する。

7.2 xtable

オブジェクトがデータフレームや行列なら，ライブラリ `xtable` を使えば簡単である。`sink` コマンドを用いなくてもファイル保存が可能である。

```
library(xtable)
print(xtable(fm),floating=FALSE,comment=FALSE,file="tab2.tex")
```

そして保存したファイルを読み込むようにする。表 2 とほぼ同じになる。

オブジェクトが対応していれば簡単に出力してくれる。対応していない場合でもオブジェクトを matrix に変更すれば表として表記が可能である。たとえば AER の coeftest などは対応していないので以下のようにすればよい。

```
library(AER)
hc <- vcovHC(fm,type="HC1")
ct <- coeftest(fm,vcov = hc)
ct <- as.matrix(ct[,])
colnames(ct)<-c("係数","標準誤差","t 値","P 値")
rownames(ct)<-c("切片","スピード")
print(xtable(ct),floating=FALSE,comment=FALSE,file="tab3.tex")
```

なお, 注意付の表の作成にはlatex のパッケージ threeparttable をもちいる。

```
\begin{table}\centering
\caption{xtablen による回帰分析結果(注釈付き)}
\label{tab:3}
\begin{threeparttable}
\input{tab3}
\begin{tablenotes}\footnotesize
\item[＊] 標準誤差はロバスト分散による。
\end{tablenotes}
\end{threeparttable}
\end{table}
```

	係数	標準誤差	t 値	P 値
切片	-17.58	5.66	-3.11	0.00
スピード	3.93	0.41	9.66	0.00

標準誤差はロバスト分散による。

表３：注釈付きの分析結果

その結果が表３である。

`xtable` より細かく指定するには `Hsmic` というライブラリを導入して，そこの `latex` コマンドを扱えばよい。

7.3 stargazer

複数のモデルの表を書くにはいくつかのパッケージがある。例えば `texreg`, `stargazer`, `apsrtable` などがある。わたしはメインに `texreg` を使っているが，最近 `stargazer` をよく用いている。

まず `stargazer` をみてみる。`stargazer` について以下のサイトが参考になる。

`http://jakeruss.com/cheatsheets/stargazer.html`

`AER` に内蔵している Stock and Watson（2011）のデータを用いて使用例をしめそう。まず実証分析の前に用いる基本統計量は以下のコマンドで簡単に表現できる。

```
library(AER)
data("CASchools",package = "AER")
CASchools$stratio <- with(CASchools,students/teachers)
CASchools$score <- with(CASchools,(math + read)/2)
library(stargazer)
stargazer(CASchools[,c("stratio","english","lunch","calworks")],
header=FALSE,out="tab6.tex",float=FALSE)
```

Statistic	N	Mean	St.Dev.	Min	Max
stratio	420	19.640	1.892	14.000	25.800
english	420	15.768	18.286	0.000	85.540
lunch	420	44.705	27.123	0.000	100.000
calworks	420	13.246	11.455	0.000	78.994

表４：stargazer による基本統計量

そこで作成したファイル tab6.tex を latex で読みこめば表 4 を作成してくれる。

また複数の回帰式を表にまとめるには以下のようにすればよい。

```
fm1 <- lm(score ~ stratio,data = CASchools)
fm2 <- lm(score ~ stratio + english,data = CASchools)
fm3 <- lm(score ~ stratio + english + lunch,data = CASchools)
fm4 <- lm(score ~ stratio + english + calworks,data = CASchools)
fm5 <- lm(score ~ stratio + english + lunch + calworks,data
= CASchools)
stargazer(fm1,fm2,fm3,fm4,fm5,out="tab7.tex",df=FALSE,
header=FALSE,float=FALSE)
```

そこで作成したファイル tab7.tex を latex で読みこめば表 5 となる。Stock and Watson（2011）の表7.1と比較すると係数は再現しているが，標準誤差が同じでない。これは標準誤差をロバスト分散にもとづいていないからである。

Stock and Watson (2011) の表7.1 と同じようなことをしようとすれば以下のようにする.

```
rse1 <- sqrt(diag(vcovHC(fm1,type = "HC1")))
rse2 <- sqrt(diag(vcovHC(fm2,type = "HC1")))
rse3 <- sqrt(diag(vcovHC(fm3,type = "HC1")))
```

	Dependent variable:				
	score				
	(1)	(2)	(3)	(4)	(5)
stratio	−2.280***	−1.101***	−0.998***	−1.308***	−1.014***
	(0.480)	(0.380)	(0.239)	(0.307)	(0.240)
english		−0.650***	−0.122***	−0.488***	−0.130***
		(0.039)	(0.032)	(0.033)	(0.034)
lunch			−0.547***		−0.529***
			(0.022)		(0.032)
calworks				−0.790***	−0.048
				(0.053)	(0.061)
Constant	698.933***	686.032***	700.150***	697.999***	700.392***
	(9.467)	(7.411)	(4.686)	(6.024)	(4.698)
Observations	420	420	420	420	420
R^2	0.051	0.426	0.775	0.629	0.775
Adjusted R^2	0.049	0.424	0.773	0.626	0.773
Residual Std.Error	18.581	14.464	9.080	11.654	9.084
F Statistic	22.575***	155.014***	476.306***	234.638***	357.054***
Note:					*p<0.1;**p<0.05;***p<0.01

表５：stargazer による回帰分析結果

```
rse4 <- sqrt(diag(vcovHC(fm4,type = "HC1")))
rse5 <- sqrt(diag(vcovHC(fm5,type = "HC1")))
stargazer(fm1,fm2,fm3,fm4,fm5,out="tab8.tex",df=FALSE,
se=list(rse1,rse2,rse3,rse4,rse4),header=FALSE,float=FALSE)
```

そうすれば表 6 となる。

7.4 texreg

次に texreg について解説する。stargazer は簡単に出力してくれるが，細かなカスタマイズしようとすると面倒である。そうしたとき texreg を使うようにしている。それでも満足できなければ最初のように自分でプログラムを作成する。

再び Stock and Watson（2011）のデータを用いて使用例を示そう。回帰

	Dependent variable:				
			score		
	(1)	(2)	(3)	(4)	(5)
stratio	−2.280***	−1.101**	−0.998***	−1.308***	−1.014***
	(0.519)	(0.433)	(0.270)	(0.339)	(0.339)
english		−0.650***	−0.122***	−0.488***	−0.130***
		(0.031)	(0.033)	(0.030)	(0.030)
lunch			−0.547***		−0.529
			(0.024)		
calworks				−0.790***	−0.048
				(0.068)	(0.068)
Constant	698.933***	686.032***	700.150***	697.999***	700.392***
	(10.364)	(8.728)	(5.568)	(6.920)	(6.920)
Observations	420	420	420	420	420
R^2	0.051	0.426	0.775	0.629	0.775
Adjusted R^2	0.049	0.424	0.773	0.626	0.773
Residual Std.Error	18.581	14.464	9.080	11.654	9.084
F Statistic	22.575***	155.014***	476.306***	234.638***	357.054***
Note:					*p<0.1;**p<0.05;***p<0.01

表６：stargazer による回帰分析結果（ロバスト標準誤差使用）

分析をおこなったあと，回帰式を並べるには以下のようにすればよい。

```
library(texreg)
texreg(list(fm1,fm2,fm3,fm4,fm5),file="tab4.tex",table = FALSE)
```

この結果は表７である。Stock and Watson（2011）の表7.1の係数の推定値を再現できている。

ただこのままだと，係数の標準誤差が再現できていない。また日本語表記にしたいなど表の細かい部分を変更したいときがある。この texreg は自分でカスタマイズできるという利点がある。この場合，library(texreg) 以下を次のようにして，新しい抽出のやり方を定義し直す。

```
extract.lm <- function(model) {
  library(lmtest)
```

```
  s <- summary(model)
  names <- rownames(s$coef)
  co <- s$coef[,1]
  hc <- vcovHC(model,type="HC1")
  ct <- coeftest(model,vcov = hc)
  se <- ct[,2]
  pval <- ct[,4]
  ser <- s$sigma
  adj <- s$adj.r.squared
  n <- nobs(model)
  gof <- c(ser,adj,n)
  gof.names <- c("標準誤差","調整済み決定係数","観測数")
  gof.decimal <- c(TRUE,TRUE,FALSE)
  tr <- createTexreg(
  coef.names = names,
  coef = co,
  se = se,
  pvalues = pval,
  gof.names = gof.names,
  gof.decimal = gof.decimal,
  gof = gof
  )
  return(tr)
}
setMethod("extract",signature = className("lm","stats"),
definition = extract.lm)
```

その上で以下のようにすれば，Stock and Watson（2011）の表7.11の結果を日本語に直して再現できる。

	Model 1	Model 2	Model 3	Model 4	Model 5
(Intercept)	698.93***	686.03***	700.15***	698.00***	700.39***
	(9.47)	(7.41)	(4.69)	(6.02)	(4.70)
stratio	−2.28***	−1.10**	−1.00***	−1.31***	−1.01***
	(0.48)	(0.38)	(0.24)	(0.31)	(0.24)
english		−0.65***	−0.12***	−0.49***	−0.13***
		(0.04)	(0.03)	(0.03)	(0.03)
lunch			−0.55***		−0.53***
			(0.02)		(0.03)
calworks				−0.79***	−0.05
				(0.05)	(0.06)
R^2	0.05	0.43	0.77	0.63	0.77
Adj.R^2	0.05	0.42	0.77	0.63	0.77
Num.obs.	420	420	420	420	420
RMSE	18.58	14.46	9.08	11.65	9.08

$^{***}p < 0.001$, $^{**}p < 0.01$, $^{*}p < 0.05$

表７：texreg による回帰分析結果

```
texreg(list(fm1,fm2,fm3,fm4,fm5),
custom.model.names = c("(1)","(2)","(3)","(4)","(5)"),
custom.coef.names =c("切片",
                     "教員一人あたり学生数",
                     "英語学習者割合",
                     "昼食補助者割合",
                     "所得支援家計"),
custom.note = "係数下のカッコ内の数値はロバスト標準誤差を示す。
また＊＊は P 値が 1\\% 以下,＊は 5\\% 以下を示す。",
reorder.coef = c(2, 3, 4, 5, 1),stars = c(0.05, 0.01),
table=FALSE , file="tab5.tex")
```

この結果は表８である。

	(1)	(2)	(3)	(4)	(5)
教員一人あたり学生数	−2.28**	−1.10*	−1.00**	−1.31**	−1.01**
	(0.52)	(0.43)	(0.27)	(0.34)	(0.27)
英語学習者割合		−0.65**	−0.12**	−0.49**	−0.13**
		(0.03)	(0.03)	(0.03)	(0.04)
昼食補助者割合			−0.55**		−0.53**
			(0.02)		(0.04)
所得支援家計				−0.79	−0.05
				(0.07)	(0.06)
切片	698.93**	686.03**	700.15**	698.00**	700.39**
	(10.36)	(8.73)	(5.57)	(6.92)	(5.54)
標準誤差	18.58	14.46	9.08	11.65	9.08
調整済み決定係数	0.05	0.42	0.77	0.63	0.77
観測数	420	420	420	420	420

係数下のカッコ内の数値はロバスト標準誤差を示す。また**はP値が1% 以下, *は５% 以下を示す。

表８：texreg による回帰分析結果（日本語）

８　まとめ

このようにデータ収集して，データ整形して，データ分析して，図表を作成する作業がすべて R で出来ることを示した。紹介したのパッケージのうち，`readr`，`readxl`，`haven`，`tidyr`，`dplyr`，`ggplot2` は Hadley Wickham による。他にもここでは紹介していないが，パッケージ作成のための `devtools` を作成している。彼が R にはたした貢献はとてつもなく大きく，最近の R の人気に間違いなく貢献している。彼の著書 Wickham（2014）は今後の R のため重要な書物になるだろう。なおこの本のソースコードは

```
https://github.com/hadley/adv-r/
```

に公開されている。コンパイルすれば書籍を入手することができる。日本語訳も発売されたようである。

ここでは紹介しなかったが，`gEcon` というパッケージが存在する。これによって一般均衡モデルを扱うことができる。一般均衡モデルについて GAMS や Matlab を使うことが主流であるが，これらは商用ソフトウェア

である。特に DSGE ではDynare というフリーの Matlab のアドオンがよく使われている。この Dynare は一階の条件式から動学モデルを解くことができる。一方 `gEcon` では最大化もしくは最小化問題からモデルを解くことができる。更に一階条件の数式や結果の表をlatex 形式に出力してくれる。詳しくは以下のサイトに訪れていただきたい。

```
http://gecon.r-forge.r-project.org/
```

最後に，学術研究において重要なのはオリジナルデータをどのように分析したかを再現できることである。安易に EXCEL を用いると手順のミス，コピペの失敗などがあるため，使わないほうがよい。コピペを使わず，自動的に論文を作成することを再現可能研究 (reproducible research) と呼ばれている[*2]。この再現可能研究についての重要なパッケージである `knitr` について次回詳述したい。

[*2] プログラムに重点を置くと文芸プログラミング（Literate Programming）と，文章に重点を置くと動的文書（DynamicDocuments）とも呼ばれている。

参考文献

Fair, Ray C。(1978) “A Theory of Extramarital Affairs,” *Journal of Political Economy*, Vol. 86, No. 1, pp.45–61.

Grolemund, Garrett (2015)『RStudio ではじめる R プログラミング入門』, オライリージャパン, 東京, (大橋真也・長尾高弘訳).

Kleiber, Christian and Achim Zeileis (2008) *Applied Econometrics with R*, New York: Springer.

Stock, James H. and Mark W. Watson (2011) *Introduction to Econometrics*: Pearson/Education, 3rd edition.

Wickham, Hadley (2014) *Advanced R*: Chapman and Hall/CRC.

ウィッカム, H.(2012)『グラフィックスのための R プログラミング』, 丸善出版, 東京, (石田基広・石田和枝訳).

ショーカー, D.(2012)『 R グラフィックス自由自在』, 丸善出版, 東京, (石田基広訳).

宮﨑憲治 (2015)「学術研究のためのオープンソース・ソフトウェア (1) : XeLaTeX」,『経済志林』, 第82巻, 第 4 号, 285–321 頁, 3 月.