第5回 様々なファイル形式の読み込みとデータの書き出し

平成 29 年 10 月 10 日

目 次

1 ここで学ぶ事

- データのやり取りで使われるデータフォーマット（Excel, CSV, Stata，JSON, XML）
- 各フォーマットの読み込み方
- 読み込んだデータを使い易いように整形する方法
- R オブジェクトの管理と保存方法

2 データフォーマット

コンピュータの発達とともにデータが様々なデータが一般に公開されつつある．そうした中でよく使われるデータフォーマットについて説明する．

- Excel ファイル: Microsoft Excel 用のフォーマットで拡張子は「.xls」または「.xlsx」．ネット上で入手可能な国の統計データは主にこのフォーマット．
- CSV ファイル: Comma Separated Values ファイルの略で，表データのセル（フィールド）をコンマで区切ったテキストファイル．普通の テキストファイルなので特別なソフトウェアを必要とせずエディタで開ける．
- Stata ファイル: 商用の統計分析ソフト Stata 用フォーマット．計量経済学でポピュラーなソフトなため，経済学系の研究で使用されたデータはこのフォーマットで公開されることが多い．
- JSON: JavaScript というスクリプト言語で書かれたデータフォーマット．単純なテキストファイルなので，JavaScript 以外のプログラミング言語でもこのフォーマットでデータをやり取りすることが多い．
- XML: XML は Extensible Markup Language の略でデータの構造や名前等の注釈をマークアップ言語で記したテキストファイル．Web ページを記述する HTML は Hyper Text Markup Language の略でやはりマークアップ言語だが，XML もこれに似た記述になる．

R では上のどのフォーマットも読み書きできるが，ここでは Excel，CSV，Stata の 3 つのフォーマットを読み込む方法を学ぶ．

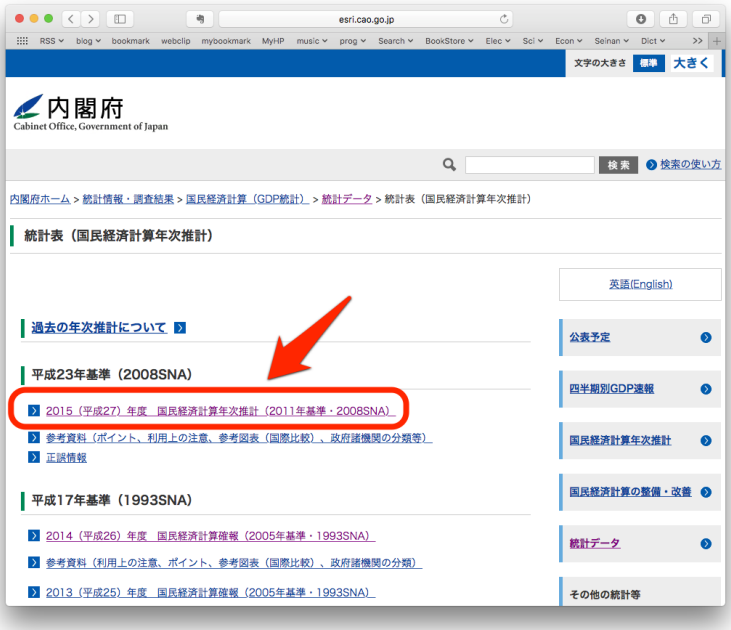
3 Excel フォーマットの読み込み

3.1 GDP データの入手

ここでは GDP 統計を R に読み込む．まずは内閣府のホームページからデータを入手する．

(1) 内閣府のホームページを検索し，「統計データ」$\Longrightarrow$ 「国民経済計算年次推計」のページに入る．

(2) 「平成 23 年基準（2008SNA）」の項目から最新の「2015（平成 27）年度国民経済計算年次推計（2011 年基準・2008SNA）」を選択．



(3) 「フロー編」の「IV. 主要系列」の（1）国内総生産（支出側）から，実質の年度データをダウンロードする．ダウンロードされたファイル名は「27ffm1rn_jp.xls」．

3.2 Excel ファイル読み込みライブラリをインストール

R はインタプリタなので，R 言語で書かれたスクリプト・ファイルを読み込むことによって機能を自由に拡張できる．よく使われる機能は「ライブラリ」として有志によってすでに提供されており，簡単にインストールできるようにパッケージ化されている．readxl パッケージは Excel ファイルの読み込み機能を提供する．

このパッケージをインストールするには以下を実行する．readxl パッケージをどこからダウンロードするかサイト一覧が表示されるので日本のサイトを選択する．一旦インストールされてしまえば，次回からこの作業は不要である．

```
> install.packages("readxl")
```

次にライブラリをロードする．これで readxl パッケージの関数が使えるようになる．

```
> library(readxl)
```

Excel ファイルの読み込み関数名は「read_excel()」である．マニュアルに目を通しておこう．

```
> help(read_excel)
```

3.3 Excel ファイルの読み込み

ダウンロードした Excel ファイルを Excel で開くと，3 つのシートがあることがわかる．read_excel() のマニュアルにはシートの指定方法も書いてあるが，デフォルトでは 1 番目のシートを読み込むと記載されている．今回読み込むのは 1 番目のシートなので特にシートの設定は必要ない．read_excel() 関数に以下のように Excel ファイル名を指定すると，読み込むことができる．

```
> gdp <- read_excel("エクセルファイル.xls")  # 実際のファイル名を指定する.
```

3.3.1 作業ディレクトリ

R がファイルを読み込む際，ファイルの場所を指定しない場合，R は「作業ディレクトリ（working directory)」内にファイルを探しに行く．R の作業ディレクトリは getwd() で確認できる．

```
> getwd()                    # 現在の作業ディレクトリを返す.
[1] "/Users/shito/Documents/git-repositories/R-programming-lecnote/handout"
```

作業ディレクトリの変更は setwd() を使う．

```
> setwd("/Users/shito")    # /Users/shito フォルダに作業ディレクトリを変更.
```

上の結果は Mac 上での実行結果だが，例えば Windows の E ドライブ内の seminer フォルダに作業ディレクトリを変更したい場合は以下を実行する．

```
> setwd("E:/seminer")
```

作業ディレクトリを変更しなくてもファイルの位置を直接指定しても良い．例えば今回ダウンロードしたファイル（27ffm1rn_jp.xls）が Windows の E ドライブ内の「基礎演習」フォルダの中にあるなら，

```
> gdp <- read_excel("E:/基礎演習/27ffm1rn_jp.xls")   # Windows の E ドライブ内の基礎演習フォルダ内のファイルを読
```

以下，27ffm1rn_jp.xls ファイルは作業ディレクトリ内の data フォルダの中にあるという前提で話を進める．従って読み込み方は，

```
> gdp <- read_excel("data/27ffm1rn_jp.xls")
```

となる．

3.3.2 読み込む範囲のカスタマイズ

ダウンロードしたファイルを Excel で開いてみると実際のデータは 8 行目から記載され，7 行目に記載の年が列名として利用できる．そこでまず最初の 6 行をスキップするように引数で指示する（skip=6）．データの 1 行目を列名として解釈させるために，名前付き引数 col_names に TRUE を指定する．ちなみに，col_names のデフォルト値は TRUE なのであえて指定しなくても良い．

```
> gdp <- read_excel("data/27ffm1rn_jp.xls", skip=6, col_names=TRUE)
> dim(gdp)     # 読み込まれた行数と列数をチェック
[1] 62 23
```

読み込まれた gdp の中身を全て表示させるとかなりの量になるため，ここでは最初の 1–3 行と 1–4 列を表示させてみる．

```
> gdp[1:3, 1:4]
# A tibble: 3 x 4
                                    X__1   `1994`   `1995`   `1996`
                                   <chr>    <dbl>    <dbl>    <dbl>
1                   l．　民間最終消費支出 245411.3 252396.7 258016.4
2                   （l）家計最終消費支出 241253.3 248033.1 253602.7
3               a．国内家計最終消費支出 237921.6 244309.7 250165.7
```

データの上に「A tibble: 3 x 4」という表示があるが，gdp はデータフレームを拡張した Tibble というデータ形式になっている．Tibble も通常のデータフレームとほとんど扱いが変わらないが少し異なる点もあるので，ここでは普通のデータフレームに変換しておく．

```
> is.data.frame(gdp)          # 一応データフレーム
[1] TRUE
> gdp <- as.data.frame(gdp)   # as.〜でその形式に変換できる．
```

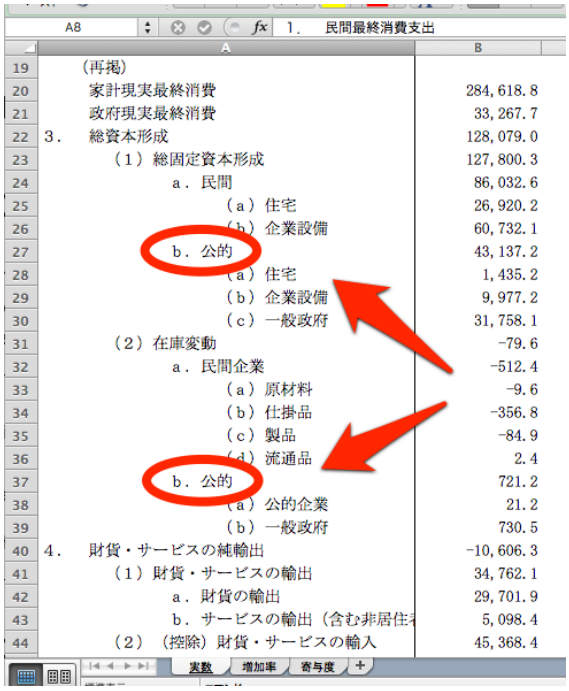
これで Excel ファイルのデータはすべて R で利用可能となるが，このままの形では使いづらいので、行名や列名を設定したり，不要な行や列を削除したり等，形を整えて後々利用し易い形に整形する．

3.4　行名の設定

行名が番号になっているので 1 列目の内容を行名に設定する．

```
> rownames(gdp) <- gdp[, 1]    # <-- エラーが出るはず！
```

上を実行すると「重複した 'row.names' は許されません 」というエラーが出て，最後に重複しているのは「b．公的」という行名だと知らせてくれている．そこで，Excel ファイルを開いて重複している行名を確認してみると確かに同じ行名が存在している．



1 列目に読み込まれた行名を表示させてみると，前後に空白が残っているのが分かる．また Excel の空行も読み込まれて NA となっているのが分かる．read_excel() 関数はデフォルトで前後の空白を取り除く設定（trim=TRUE）になっているのだが，GDP データの項目名の空白は全角なため，空白として認識されなかったのだ．

```
> gdp[,1]        # Excel の行名データを表示．空白が余分に含まれたり NA が存在する．
```

3.4.1　不要な行の削除

まず空行は不要なので NA の行を削除する．

```
> gdp <- gdp[!is.na(gdp[,1]),]        # NA の行を削除
> dim(gdp)                            # 62 行から 57 行に減った
[1] 57 23
```

Excel ファイルを見ると最後の 3 行は注釈で，gdp の 2 列目以降にデータは無く NA なので，最後の 3 行を取り除く．

```
> gdp <- gdp[1:(dim(gdp)[1]-3), ]     # 最後の 3 行を取り除く
> dim(gdp)                            # 57 行から 54 行に減った
[1] 54 23
```

上のコードで最後の 3 行を除いたインデックスを生成するのに 1:(dim(gdp)[1]-3) としているが，括弧を付けないで以下のように書いてしまうと意味が異なるので注意する．プログラムを書くときは，部分部分をプロンプトで実行し結果を確認しながら書くようにすると，このような間違いを犯さずに済む．

```
> 1:dim(gdp)[1]-3     # (1:dim(gdp)[1]) - 3 と同じ意味になってしまう．
 [1] -2 -1  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
[28] 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
> 1:(dim(gdp)[1]-3)   # これが正しいやり方．括弧に注意！
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
[28] 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51
```

3.4.2 行名の準備（空白の削除と重複名の修正）

次に行名の前後に含まれる余計な空白を削除する．文字列の置き換えは gsub() 関数（global substitution）を使う．使い方は，「gsub("置き換えたい文字列", "新文字列", 文字列ベクトル)」である．

```
> cn <- gsub("　", "", gdp[, 1])     # 全角スペースを取り除く
> cn[1:7]                            # 結果の確認．まだ半角スペースが残っている．
[1] "ｌ．民間最終消費支出"
[2] "（ｌ）家計最終消費支出"
[3] "ａ．国内家計最終消費支出"
[4] "ｂ．居住者家計の海外での直接購入"
[5] "ｃ．(控除）非居住者家計の国内での直接購入"
[6] "（再掲）"
[7] "　　家計最終消費支出（除く持ち家の帰属家賃）"
> cn <- gsub(" ", "", cn)            # 半角スペースも取り除く
```

この cn を gdp の行名に設定したいのだが，その前に重複している行名がないかチェックする．duplicated() 関数はベクトルを引数に取り，同じ値の箇所を TRUE とするベクトルを返す．例えば，1 から 10 までのベクトルの 5 番目と最後の 1 を設定し，duplicated() 関数でチェックすると，2 回目以降に 1 が出現する 5 番目と 10 番目が TRUE となる．

```
> n <- 1:10           # 1 から 10 までのベクトルを用意
> n[c(5, 10)] <- 1    # 5 番目と 10 番目に 1 をセット
> n
 [1] 1 2 3 4 1 6 7 8 9 1
```

```
> duplicated(n)        # 重複箇所の 5 番目と 10 番目が TRUE
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE
> n[duplicated(n)]     # 2 回目以降に出現の重複要素を抽出できる.
[1] 1 1
```

duplicated() 関数を用いて cn から重複している要素を抽出する.

```
> cn[duplicated(cn)]
[1] "（再掲）"       "（a）住宅"     "（b）企業設備" "b．公的"
```

Excel ファイルを開いて確認すると「(再掲)」「(a) 住宅」「(b) 企業設備」「b．公的」の行が重複している．ここではとりあえず「(再掲)」行をデータから削除する．それに合わせて cn からも削除する.

```
> gdp <- gdp[cn!="（再掲）", ]
> cn <- cn[cn!="（再掲）"]
> dim(gdp)                          # 2 行削除され 55 行に減った
[1] 52 23
```

次に，2回目に出現する「(a) 住宅」と「(b) 企業設備」の名前を変更する．これらには親カテゴリの「公的」にちなんで「(公的)」を後ろに加えることにする.

```
> (1:length(cn))[duplicated(cn)]              # 重複している箇所のインデックス番号.
[1] 18 19 27
> n <- (1:length(cn))[duplicated(cn)][1:2]    # 最初の 2 つがこれから変更する箇所.
> cn[n]                                       # 変更前
[1] "（a）住宅"     "（b）企業設備"
> cn[n]<- paste(cn[n], "（公的）", sep="")    # paste で後ろに（公的）を追加
> cn[n]                                       # 変更後
[1] "（a）住宅（公的）"     "（b）企業設備（公的）"
```

3.4.3 練習問題

重複している残りの行名「b．公的」を，Excel ファイルと見比べながら重複しないように変更し，変更後の cn を gdp の行名に設定せよ.

3.4.4 列名の変更

列名は文字列でなくてはならないので，読み込まれた年の数値は文字列に変換されている.

```
> colnames(gdp)
 [1] "1994" "1995" "1996" "1997" "1998" "1999" "2000" "2001" "2002" "2003" "2004"
[12] "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015"
```

例えば$記号で 1994 年の列にアクセスしようとするとエラーになる．

```
> gdp$1994            # <-- エラー！
```

以下のように 1994 をダブルクォーテーションで囲って文字列に変換しなければならない．

```
> gdp$"1994"
```

これでは不便なので頭に Y を加えて列名が文字列であることを R に分からせるようにする．

```
> paste("Y", colnames(gdp), sep="")      # これを列名に設定する．
 [1] "Y1994" "Y1995" "Y1996" "Y1997" "Y1998" "Y1999" "Y2000" "Y2001" "Y2002" "Y2003"
[11] "Y2004" "Y2005" "Y2006" "Y2007" "Y2008" "Y2009" "Y2010" "Y2011" "Y2012" "Y2013"
[21] "Y2014" "Y2015"
> colnames(gdp) <- paste("Y", colnames(gdp), sep="")
> gdp$Y1994                              # これで$を使ってうまくアクセスできる
 [1] 245411.3 241253.3 237921.6   3841.3    219.6 204145.9  36801.1   4189.9  72309.6
[10] 284618.8  33267.7 128079.0 127800.3  86032.6  26920.2  60732.1  43137.2   1435.2
[19]   9977.2  31758.1    -79.6   -512.4     -9.6   -356.8    -84.9      2.4    721.2
[28]     21.2    730.5 -10606.3  34762.1  29701.9   5098.4  45368.4  31786.3  12578.1
[37]       NA 426575.4 -11537.5  19152.4 445727.8   3802.3  14738.6  10936.3 449530.1
[46] 446080.7 331308.9 114933.2 421220.0 236999.7  34446.7  45303.1
```

上の 1994 年のデータの中に NA が含まれている．このデータの行名を表示させてみると空文字列であることがわかる．前に行名が NA の行を削除したが，全角文字だけの行名が含まれていて NA とならなかったのだろう．この行を削除する．

```
> rownames(gdp)[is.na(gdp$Y1994)]           # データが NA の行名は空の文字列
[1] ""
> gdp <- gdp[rownames(gdp) != "", ]         # 行名が空文字列の行を gdp から削除
> gdp$Y1994
 [1] 245411.3 241253.3 237921.6   3841.3    219.6 204145.9  36801.1   4189.9  72309.6
[10] 284618.8  33267.7 128079.0 127800.3  86032.6  26920.2  60732.1  43137.2   1435.2
[19]   9977.2  31758.1    -79.6   -512.4     -9.6   -356.8    -84.9      2.4    721.2
[28]     21.2    730.5 -10606.3  34762.1  29701.9   5098.4  45368.4  31786.3  12578.1
[37] 426575.4 -11537.5  19152.4 445727.8   3802.3  14738.6  10936.3 449530.1 446080.7
[46] 331308.9 114933.2 421220.0 236999.7  34446.7  45303.1
```

これで読み込まれた Excel のデータを，綺麗なデータフレームに整えることができた．最初の 3 行と 4 列を表示させてみよう．

```
> gdp[1:3, 1:4]
                          Y1994    Y1995    Y1996    Y1997
1．民間最終消費支出      245411.3 252396.7 258016.4 255521.7
（1）家計最終消費支出    241253.3 248033.1 253602.7 251163.4
a．国内家計最終消費支出 237921.6 244309.7 250165.7 248216.8
```

4 データの保存

ここでは作成したデータフレームをファイルに保存する方法を学ぶ．R で作成されたあらゆるデータは「オブジェクト」と呼ばれる．ls() 関数は作成されたオブジェクトの一覧を表示する．

```
> ls()        # オブジェクト一覧を表示（list の略）
[1] "cn"  "gdp" "n"
```

オブジェクトは save() 関数を使ってバイナリのまま保存できる．次回以降のセッションで load() 関数を使ってバイナリデータをロードできる．データファイルの拡張子は「.RData」を使う．

```
> save(gdp, file="ch05-GDP.RData")   # gdp を ch05-GDP.RData ファイルに保存.
```

複数のオブジェクトを一つのファイルにまとめて保存したい場合，最初にオブジェクトを列挙する．

```
> save(gdp, cn, file="ch05-GDP.RData")   # gdp と cn がまとめて保存される.
```

5 データのロード

保存したデータは load() 関数を使って読み込む．実際にロードされたかどうかを確認するために，一旦，gdp と cn を rm() 関数（remove）を使って削除しておこう．

```
> rm(gdp, cn)     # gdp と cn を remove
> ls()            # gdp と cn が無くなった
[1] "n"
```

load() 関数で保存しておいたデータを読み込む．

```
> load("ch05-GDP.RData")
```

読み込まれたかどうか確認してみる．

```
> ls()
[1] "cn"  "gdp" "n"
> gdp[1:3, 1:4]
                          Y1994    Y1995    Y1996    Y1997
１．民間最終消費支出      245411.3 252396.7 258016.4 255521.7
（１）家計最終消費支出    241253.3 248033.1 253602.7 251163.4
ａ．国内家計最終消費支出 237921.6 244309.7 250165.7 248216.8
```

今後，実習や宿題で作成したデータは全てファイルに保存しておこう．

6　おまけ—GDP データを使ってみよう

せっかく R から使いやすいよう GDP のデータフレームを作成したので，ここで少しだけデータを使って分析してみよう．

国の家計簿である国民所得統計では GDP（国内総生産）は GDI（国内総所得）と GDE（国内総支出）と等しい[1]．これらの名前を行名から探すと「５．国内総生産（支出側）」と「国内総所得」が 37 行目と 40 行目にそれぞれ見つかる．

```
> rownames(gdp)[c(37, 40)]
[1] "５．国内総生産（支出側）" "国内総所得"
```

試しに 2015 年のこれらの行を抽出してみる．

```
> gdp[c("５．国内総生産（支出側）", "国内総所得"), "Y2015"]
[1] 517195.3 523706.6
```

等しいはずの両者が違う値になっている．そこで，もう一度 Excel ファイルを開いてデータの定義を探すと，我々が削除した最後の 3 行に「国内総所得＝国内総生産＋交易利得」という記述がある．従って，この統計表では GDE に交易利得を加えたものを国内総所得として計上していることがわかる．

ここでは GDP として「５．国内総生産（支出側）」を採用する．まず，期間中の GDP の最大値と最小値を求めてみよう．

```
> max(gdp["５．国内総生産（支出側）",])    # GDP の最大値を抽出
[1] 517195.3
> min(gdp["５．国内総生産（支出側）",])    # GDP の最小値を抽出
[1] 426575.4
```

Excel ファイルを見ると単位は 10 億円なので，景気の良い時と悪い時で 100 兆円近く GDP に差があることがわかる．

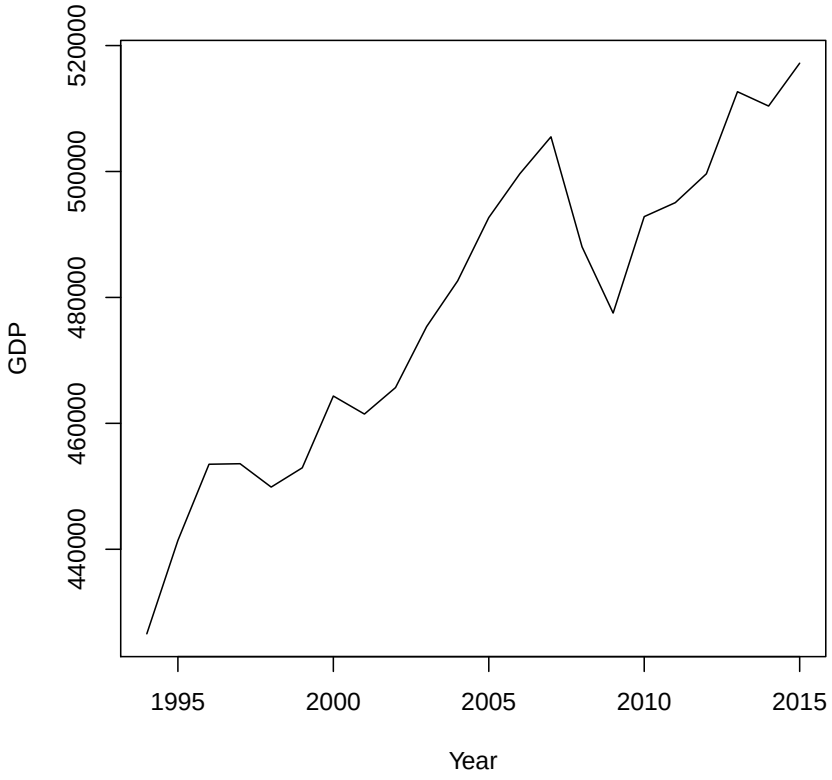
次に，GDP が最大・最小になった年度を調べてみよう．列名が年度だったので，最大値・最小値と等しい真偽値インデックスを生成して列名を抽出する．GDP の値は何度も参照するので，一旦変数 y に格納しておく．

```
> y <- gdp["５．国内総生産（支出側）",]
> colnames(gdp)[y == max(y)]                # y の最大値が y と等しい列名を取得
[1] "Y2015"
> colnames(gdp)[y == min(y)]                # y の最小値が y と等しい列名を取得
[1] "Y1994"
```

GDP が最大なのは一番直近の 2015 年度で，最小は一番古い 1994 年度である．それでは，継続して経済成長してきたのかを見るために，GDP の推移をグラフにしてみよう．

[1] 知らなかった人はマクロ経済学 I を受講しましょう！

```
> plot(1994:2015, y, type="l", xlab="Year", ylab="GDP")
```



plot() 関数は，plot(x, y) で x 座標値と y 座標値のデータを順に受け取り，その点をプロットする．今回，x 軸の座標値は 1994 年から 2015 年なので 1994:2015 で座標値のベクトルを生成している．y 軸の座標値は GDP データなので先ほど設定した変数 y を与えている．

第 2 引数の type="l"は，プロットした点を線（line）で結ぶオプション指定である．これで折れ線グラフができる．xlab と ylab はそれぞれ x 軸と y 軸のラベル（軸名）の設定オプションである．

描かれたグラフを見ると必ずしも後の年度の方が GDP が高いとは限らないことがわかる．4 回ほど前年の GDP を割り込みマイナス成長している．特に 2009 年の落ち込みが最大で，これは 2008 年のリーマンショックが原因と考えられる．

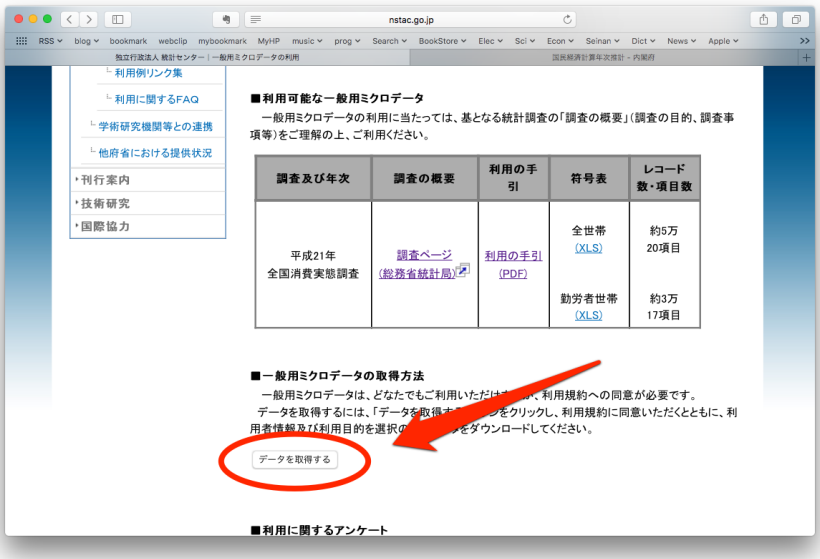
7 CSV フォーマットの読み込み

本節では CSV（Comma Separated Values）フォーマットファイルの読み込み方を学ぶ．CSV ファイルは単なるテキストファイルで汎用性が高いため，多くのデータが CSV フォーマットで提供されている．ここでは，独立行政法人統計センターが学習用に提供している大規模な擬似ミクロデータを読み込んでみる．データ数は約 5 万件に上る．

7.1　データの入手

(1) 「独立行政法人 統計センター」で検索し，以下の URL のページを開く．
http://www.nstac.go.jp/services/ippan-microdata.html

(2) 「データを取得する」ボタンを押して，利用規約に同意してデータをダウンロードする．



(3) ダウンロードファイルを展開すると「ippan_2009zensho」というフォルダが作成される．中に 2 つの csv ファイルと PDF ファイルがある．このうち，今回読み込むのは，「ippan_2009zensho_z_dataset.csv」ファイルである．

7.2　CSV ファイルの読み込み

CSV ファイルの読み込みは read.csv() 関数を使う．いつものように，まずはマニュアルを読んでみよう．

```
> help(read.csv)
```

マニュアルの最初には Usage 項目があり，指定できるオプションが名前付き引数で提供されている．

今回読み込む ippan_2009zensho_z_dataset.csv をエディタで開いてみると，最初の 5 行はコメントが書いてあり，6 行目に列名，7 行目からデータが始まっている．オプションで skip=5, header=TRUE を指定する．

また，Excel ファイルで提供されているこのデータの符号表を見ると，文字コードは Shift_JIS であることが分かる．R のデフォルト文字コードは UTF8 なので，読み込む際には，encoding="Shift_JIS"で文字コードを指定する．

```
> microdata <- read.csv("data/ippan_2009zensho_z_dataset.csv",  # パスは自分の環境に合わせる
+                       skip=5, header=TRUE, encoding="Shift_JIS")
```

```
> dim(microdata)    # 読み込まれたデータ数をチェック
[1] 45811    20
```

読み込まれたデータの最初の数行と最後の数行を表示する際，行番号を指定する代わりに head() 関数と tail() 関数を使うと便利である．

```
> head(microdata)         # 最初の数行を表示する
> tail(microdata)         # 最後の数行を表示する
```

このデータも次回以降使用するので，RData ファイルに保存しておこう．

```
> save(microdata, file="Microdata.RData")  # パスは各自の環境に合わせて変更すること
```

8 Stata フォーマットの読み込み

Stata は計量経済学の商用ソフトでは非常にポピュラーである．研究者が整備した経済データは Stata フォーマットで提供されることが多いので，このファイルの読み込み方も練習しておこう．拡張子が「.dta」のファイルは Stata フォーマットのファイルである．

Stata ファイルは，R のデータファイルのようにプログラムから利用しやすいように整形済みのデータを格納したものなので，オプション等をつけずに単純に読み込めばデータフレームとして読み込まれる．ここでは実際にファイルを読み込むことはしないが，読み込み方だけ以下に説明する．

Stata ファイルの読み込みに使う read.dta() 関数は，foreign パッケージで提供されているので，まずはパッケージのロードから行う．

```
> library(foreign)
```

パッケージが提供する関数の一覧は以下で得ることができる．

```
> library(help=foreign)         # foreign パッケージが提供する関数一覧を表示
> help(read.dta)                # read.dta のマニュアルを読む．
```

Stata ファイルを読み込むには単純にデータファイルへのパスを read.dta() 関数に渡せば良い．

```
> stata.data <- read.dta("ファイル名.dta")    # Stata の dta ファイルを読み込む
```

9 実習

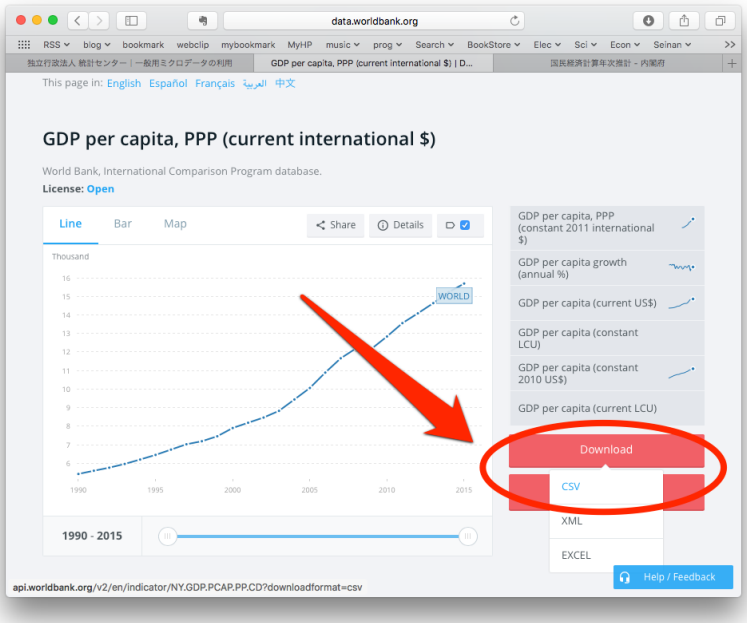
世界各国の GDP を比較するために世界銀行が提供する GDP 統計をダウンロードし R に読み込む．円表示の GDP とドル表示の GDP では比較できないため，各国の GDP を比較する際には，通常ドルに通貨単位をそろえる．ただし，日本人にとっての 10 ドルと，最貧国の国民にとっての

10 ドルでは価値が異なる．そこで GDP をドルに変換する際には，単純な外国為替レートではなく，購買力が等しくなるように調整した為替レート，Purchasing Power Parity Rate（PPP レート）を用いて変換する．

幾つかの期間が PPP レートを推計し，通貨単位を揃えた GDP を発表している．ここでは，世界銀行が提供する GDP データを用いる．人口の多い国と少ない国とで GDP を比較しても意味がないので，条件を揃えて一人当たり GDP（GDP per capita）を使用する．

(1) 「World Bank GDP ppp per capita」で検索し，
http://data.worldbank.org/indicator/NY.GDP.PCAP.PP.CD
のページを開く．

(2) 右にある「Download」をクリックし「CSV」ファイルを選択しダウンロードする．



(3) API_NY というフォルダ内の「`API_NY.GDP.PCAP.PP.CD_DS2_en_csv_v2.csv`」がデータファイルである．各自でファイルを観察し適切なオプションを設定して R に読み込み，ppp という名前の変数に格納せよ．

(4) 読み込んだデータ数を表示せよ．

(5) ppp の行名を 1 列目の国名に置き換え，不要になった 1 列目を削除せよ．

(6) 「Indicator.Name」と「Indicator.Code」の列は不要なので削除せよ．

(7) 2015 年の最も豊かな国と，最貧国と，日本の一人当たり GDP を抽出し比較せよ．

(8) 上で求めた最富裕国と最貧国の一人当たり GDP が，日本の一人当たり GDP の何倍か求めよ．

(9) ppp を `WorldBank_GDP.RData` ファイルに保存せよ．

(10) ワークスペースから ppp を削除し，ワークスペースのオブジェクト一覧を表示せよ．

(11) `WorldBank_GDP.RData` ファイルからデータを読み込み，`ppp` オブジェクトの次元を表示せよ．