

Rで学ぶVAR実証分析

時系列分析の基礎から予測まで

プログラム（Rスクリプト）

更新日：2024年5月31日

公開日：2019年12月15日

村尾 博

本書に用いたプログラム（Rスクリプト）およびデータを提供します。ファイル名の後に簡単な説明を書き、それぞれの役割や目的が分かるようにしています。

その他の資料も含めています。それらは目次の2と3に示すものです。

最後にディレクトリ（フォルダ）の表記法に関する説明を行っています。Rスクリプト（プログラム）を書く人にとっては自明な内容ですが、念を押しを兼ね、書いています。

もくじ

1. [本書に用いたプログラム](#)
2. [その他のプログラム](#)
3. [練習用データ](#)
4. [ディレクトリ表記に関する説明](#)

1. 本書に用いたプログラム

これらのファイルは「**Chap_1**」や「**Chap_2**」といったフォルダに入っています。本書に用いたプログラムを含める一方、本書の内容と関連性の強いプログラムも含めています。

「**Chap_1**」や「**Chap_2**」といったフォルダ内にサブフォルダ「**R_prg**」を作り、そこにプログラム（Rスクリプト）を入れています。各フォルダ「**R_prg**」には次の3つのサブフォルダを作っています。

1. data
2. data_temp
3. outputs

個々のフォルダによってサブフォルダ構造を変えることが面倒であることから、「R_prg」と名付けた全てのフォルダに対して同じサブフォルダ構造を採用しています。サブフォルダを使うRスクリプトもあれば、サブフォルダを使わないRスクリプトもあります。

R関数については、「lmtest::bgtest()」といった表記法を用い、R関数が含まれているパッケージ名を示しています。必要に応じてパッケージをインストールしてください。ただし、「base」はRインストール時に自動的にインストールされるパッケージを意味します。したがって別途インストールをする必要はありません。

第1章 Rについて

- `different_Id_process.R` --- 図1.1の作成。`base::matplot()`関数や`base::text()`関数を使ってラベル付き折れ線グラフを作成します。

このプログラムは、前半がインパルス応答分析の基本を示すプログラムになっています。インパルス応答分析の計算カククリが簡単なループ計算であることが示されています。

第3章 時系列分析の基礎

- `roots_vars.R` --- `vars::roots()`関数を用い、VAR過程における固有方程式の固有値 (eigenvalues) を求めます。

第5章 ラグ次数の選択問題

- `lagSelect_VAR.R` --- `vars::VARselect()`関数を用い、VAR過程におけるラグ次数の選択問題に関する情報を得ます。

第6章 単位根検定

- `test_ur_df.R` --- `urca::ur.df()`関数を用い、拡張ディッキー=フラー (ADF) 単位根検定を実行します。
- `test_ur_ers.R` --- `urca::ur.ers()`関数を用い、Elliott-Rothenberg-Stock (ERS) 単位根検定を実行します。この検定の1タイプはADF-GLS検定と呼ばれています。
- `test_CADF_seqTest.R` --- `CADFtest::CADFtest()`関数を用い、Covariate-ADF (CADF) 単位根検定を実行します。帰無仮説が棄却できるまで検定を続け、判断を行なう。
- `test_CADF_noSeqTest.R` --- `CADFtest::CADFtest()`関数を用い、Covariate-ADF (CADF) 単位根検定を実行します。実験的な内容。
- `test_pgff.R` --- `egcm::pgff.test()`関数を用い、Pantula, Gonzales-Farias, Fuller (PGFF) 単位根検定を実行します。この検定はWeighted Symmetric単位根検定とも呼ばれています。
- `test_ur_pp.R` --- `urca::ur.pp()`関数を用い、Phillips-Perron (PP) 単位根検定を実行します。
- `test_ur_kpss.R` --- `urca::ur.kpss()`関数を用い、Kwiatkowski, Phillips, Schmidt, and Shin (KPSS) 単位根検定を実行します。

- `test_ur_zar.R` --- `urca::ur.za()`関数を用い、Zivot and Andrews (ZA) 単位根検定を実行します。未知な1時点で係数パラメータが変化する想定になっています。

第7章 共和分検定

- `test_cajolst.R` --- `urca::ca.jolst()`関数を用い、ヨハンセン共和分検定を実行します。
- `test_cajolst.R` --- `urca::cajolst()`関数を用い、構造変化がある場合のヨハンセン共和分検定を実行します。未知な1時点でレベルシフトが起こる想定になっています。Luetkepohl-Saikkonen-Trenkler 共和分検定と呼ばれています。
- `test_egcm.R` --- `egcm::egcm()`関数を用い、エングル＝グランジャー共和分検定を実行します。

次の共和分検定は本書に記載していません。キーワード的な情報を提供します。

- Phillips-Ouliaris Cointegration Test --- Engle-Granger 検定の一般化。
`tseries::po.test()`で実行できます。

参考文献

- Phillips, P. C. B. and Ouliaris, S. (1990), Asymptotic Properties of Residual Based Tests for Cointegration, *Econometrica* 58, 165-193.

第8章 攪乱項に関する仮説検定

- `test_serial.R` --- `vars::serial.test()`関数を用い、系列無相関の検定を実行します。
- `test_arch.R` --- `vars::arch.test()`関数を用い、均一分散の検定を実行します。
- `test_normality.R` --- `vars::normality.test()`関数を用い、正規分布の検定を実行します。

第9章 推定と識別問題

- `est_VAR_vars.R` --- `vars::VAR()`関数を用い、誘導VARモデルを推定します。
- `est_VEC_urca.R` --- `urca::cajorls()`関数を用い、誘導VECモデルを推定します。
- `est_SVAR_vars.R` --- `vars::SVAR()`関数を用い、構造VARモデルを推定します。
- `est_SVEC_vars.R` --- `vars::SVEC()`関数を用い、構造VECモデルを推定します。
- `est_restrict_vars.R` --- `vars::restrict()`関数を用い、制約付き誘導VARモデルを推定します。

第10章 係数パラメータに関する仮説検定

グランジャー因果性検定は、[該当セクション](#)へ入れています。

構造安定性の検定

- `test_stability_vars.R` --- `vars::stability()`関数を用い、構造安定性の検定を実行します。

他の検定

- `test_chow_svars.R` --- `svars::chow()`関数を用い、構造変化に関するChow検定を実行します。
- `test_js_svars.R` --- `svars::js.test()`関数を用い、B型構造モデル特有の検定を実行します。
- `test_ablptest_urca.R` --- `urca::ablptest()`関数を用い、VECモデル特有の検定を実行します。

第11章 インパルス応答分析

- `irf_VAR_vars.R` --- `vars::irf()`関数を用い、直交化インパルス応答関数に基づく分析を行ないます。使用データ「Canada」はVECや構造VECに適したデータです。そのような意味で、使用データ「Canada」と誘導VARとの組み合わせは、よい組み合わせではありません。

次は筆者が作成したRスクリプトです。一般化インパルス応答関数と直交化インパルス応答関数に基づき、2つの方法で実行します。モデルは誘導VARモデルです。使用データの名称も合わせて示しておきます。

- `irf_two_methods_VAR.R` --- 筆者が作成したRスクリプト
- `data_KPSW.csv` --- データファイル（csvフォーマット）

データ「data_KPSW.csv」の出所については、[練習用データ](#)を参照してください。

インパルス応答分析のためのコンピュータコードを書く場合は、次の2つのアプローチがあり、結果的には同等な値になります。

1. VARアプローチ：VAR（Vector Autoregression）表現に基づきコンピュータコードを書く。
2. VMAアプローチ：当該VARモデルのVMA（Vector Moving Average）表現に基づきコンピュータコードを書く。

教科書に示されているのはVMAアプローチのみです。しかし、自分でコンピュータコードを書く場合は、VARアプローチのほうが容易であると思います。次のRスクリプトは、これら2つのアプローチが同じ結果になることを示します。

- `irf_VAR_vs_VMA.R` --- VARアプローチとVMAアプローチとの同等性を示します。

第12章 推定後のモデル変換

- `irf_vec2var_vars.R` --- パッケージ「vars」の`vec2var()`関数と`irf()`関数を使うのが特徴的です。推定はVECモデルで行ない、モデル推定後にレベルVARへ変換し、そのレベルVARに基づくインパルス応答分析を実行します。使用パッケージは「urca」と「vars」であり、そこに含まれている`ca.jo()`関数、`vec2var()`関数、`irf()`関数を使います。

- **irf_SVEC_vars.R** --- パッケージ「vars」のSVEC()関数とirf()関数を使うのが特徴的です。いわゆる構造VECモデルに基づくインパルス応答分析を行います。使うパッケージは「urca」と「vars」であり、そこに含まれているca.jo()関数、SVEC()関数、irf()関数を使います。
- **irf_SVAR_vars.R** --- パッケージ「vars」のSVAR()関数とirf()関数を使うのが特徴的です。いわゆる構造VARモデルに基づくインパルス応答分析を行います。使用データ「Canada」と構造VARとの組み合わせは、よい組み合わせではありません。

第13章 インパルス応答分析の区間推定

筆者が作成したRスクリプトです。ブートストラップ法のRスクリプトは長いので2つのファイルに分けています。1番ショック(s1)から3番内生変数(y3)へのインパルス応答分析といった意味で「y3s1」といった情報をファイル名に含めています。

- **boot_y3s1_1of2.R** --- ブートストラップ法のパート1。ブートストラップ複製を得るまで。
- **boot_y3s1_2of2.R** --- ブートストラップ法のパート2。ブートストラップ複製からグラフを描くまで。
- **data_Stock.csv** --- データファイル (c s v フォーマット)

次も筆者が作成したRスクリプトです。ブートストラップ法の主要部分を関数の形で書いている点が特徴的です。配列も使っています。そのような意味で上のRスクリプトよりも上級編といった位置づけになっています。

- **boot_myF_1of2.R** --- ブートストラップ法のパート1。ブートストラップ複製を得るまで。
- **boot_myF_2of2.R** --- ブートストラップ法のパート2。ブートストラップ複製からグラフを描くまで。
- **data_Stock.csv** --- データファイル (c s v フォーマット)

データ「data_Stock.csv」の出所については、[練習用データ](#)を参照してください。

第14章 予測誤差の分散分解

- **fevd_vec2var_vars.R** --- vars::vec2var()関数とvars::fevd()関数を使うのが特徴的です。推定はVECモデルで行ない、モデル推定後にレベルVARへ変換し、そのレベルVARモデルに基づいて予測誤差の分散分解を実行します。使用パッケージは「urca」と「vars」であり、そこに含まれているca.jo()関数、vec2var()関数、fevd()関数を使います。

次は筆者が作成したRスクリプトです。直交化インパルス応答関数に基づきつつ、2つの異なった方法で実行します。モデルは誘導VARモデルです。使用データの名称も合わせて示しておきます。

- **fevd_two_methods_VAR.R** --- 筆者が作成したRスクリプト。モデルは誘導VARモデル。
- **data_Stock.csv** --- データファイル (c s v フォーマット)

データ「data_Stock.csv」の出所については、[練習用データ](#)を参照してください。

第15章 グランジャー因果性検定

- `test_causality_vars.R` --- `vars::causality()`関数を使い、グランジャー因果性検定を実行します。 `causality()`関数は（１）グランジャー因果性検定、（２）グランジャー瞬間的因果性検定といった2つのタイプの検定を行います。

第16章 ヒストリカル分解

- `hd_id_svars.R` --- `svars::hd()`関数と`svars::id()`関数を用い、ヒストリカル分解を実行します。 `hd()`関数はVMA表現と配列を使っている特徴があります。

本書第1版の発行後に`hd()`関数の修正がありました。Rスクリプト「`hd_id_svars.R`」は、それを反映する内容へ変更しました。

次は本書の発行後に筆者が作成したRスクリプトです。 `hd()`関数を使わずに、その内容をRコードで書くといった形になっています。配列（3次元の行列）を使用するか否か、VMA表現を使用するか否かなどに基づき、3つのタイプを書いています。使用データの名称も合わせて示しておきます。

- `hd_SVAR_v1.R` --- VMA表現に基づかず、VAR表現に基づいています。行列を使い、配列を使わない特徴があります。
- `hd_SVAR_array_v1.R` --- VAR表現と配列を使っている特徴があります。
- `hd_VMA_v1.R` --- VMA表現と配列を使っている特徴があります。 `id()`関数のコードに最も近い形式になっています。
- `data_Stock.csv` --- データファイル（`c s v`フォーマット）

データ「`data_Stock.csv`」の出所については、[練習用データ](#)を参照してください。

第17章 その他のVAR分析

予測

- `predict_vec2var_vars.R` --- `vars::predict()`関数を使い、予測を実行します。推定はVECモデルで行ない、モデル推定後にレベルVARへ変換し、そのレベルVARに基づいて予測を行います。

2. その他のプログラム

本書の内容に対して「付録的」「2次的」なRスクリプトを含めています。何が「付録的なのか」「2次的なのか」の線引きは難しいですが、そのようなRスクリプトを含めています。

追加のRパッケージを必要とするRスクリプトが含まれているので、その点に注意してください。

2.1 最尤推定法

VARモデルに関する対数尤度関数は複雑であり、Rスクリプトとして書くのは手間暇がかかります。スマートな「抜け道」はないものかと探ると、見つかりました。それは

concentrated likelihood function（集約尤度関数）を使うといったアイデアです。

concentrated likelihood functionの考え方は次のようになっています。

concentrated likelihood functionを最大化する最尤推定量は、元々の尤度関数を最大化する最尤推定量でもある。したがって興味のあるパラメータだけを推定する場合に適しています。「AB型モデル」と呼ばれる構造VARモデルの場合は、 \mathbf{A} 行列と \mathbf{B} 行列のみの推定に興味があります。

「AB型モデル」に分類される構造VARモデル（SVAR）において、 \mathbf{A} 行列と \mathbf{B} 行列に焦点を置いたconcentrated likelihood function（集約尤度関数）は、次のように表現できます。

$$\log L_c(\mathbf{A}, \mathbf{B}) = -\frac{K \times T}{2} \log 2\pi + \frac{T}{2} \log(\det(\mathbf{A})^2) - \frac{T}{2} \log(\det(\mathbf{B})^2) - \frac{T}{2} \text{tr}(\mathbf{A}'(\mathbf{B}^{-1})' \mathbf{B}^{-1} \mathbf{A} \hat{\Sigma}_u)$$

ただし、 $(\mathbf{B}^{-1})' = (\mathbf{B}')^{-1}$ であり、 $\hat{\Sigma}_u$ は誘導モデルにおける共分散行列 Σ_u の最尤推定量です。このconcentrated likelihood functionは、Rパッケージ「vars」のSVAR()関数に関するマニュアルに記載されています。

上に示すconcentrated likelihood function ($\log L_c$) がどれほど有用かを述べておきたいと思います。元々のモデルが複雑であれば、それに対応する対数尤度関数も複雑になります。一方、上に示すconcentrated likelihood function ($\log L_c$) は、 $K \times K$ の行列 $\hat{\Sigma}_u$ を入力（引数）とし、係数行列 \mathbf{A} と \mathbf{B} に関する最尤推定量を出力する道具になっています。 \mathbf{A} と \mathbf{B} に含まれている未知パラメータを決定するのに必要な情報は $\hat{\Sigma}_u$ だけであり、他の情報はいらないと言っています。コンピュータコードを書く者の視点から見れば、「快刀乱麻を断つような工夫」「目から鱗が落ちるような工夫」になっています。

入力（引数）となる $\hat{\Sigma}_u$ は容易に得ることができます。誘導モデルの環境ではOLS推定法やGMM推定法などが備わっており、最尤推定量 $\hat{\Sigma}_u$ と等しい推定量は容易に得ることができます。つまり、最尤推定法だからといって誘導共分散行列 Σ_u を推定するのに最尤推定法を用いる必要はありません。数値的に等しいのであれば簡単な方法を用いれば良いのです。

例えば簡単な回帰モデル

$$y_i = x_i' \beta + u_i, \quad u_i \sim i.i.d. N(0, \sigma^2)$$

$$\text{for } i = 1, 2, \dots, T$$

において、我々は次のことを知っています。OLS推定法から得た残差2乗和 RSS を用い、分散 σ^2 の不偏推定量 S^2 は $S^2 = \frac{RSS}{T-k}$ といった形で計算します。ただし、 k は RSS を得るのに使った推定係数の個数($k = \dim(\beta)$)です。一方、分散 σ^2 の最尤推定量 $\hat{\sigma}^2$ は $\hat{\sigma}^2 = \frac{RSS}{T}$ といった形になります。つまり、自由度を調整するだけで分散 σ^2 の最尤推定量 $\hat{\sigma}^2$ は簡単に得ることができます。この考え方は係数推定量が最尤推定量と同じになるのであればOLS推定量であろうがGMM推定量であろうが関わりなく広く適用できます。この考え方を $K \times K$ の残差2乗和に適用し、 $K \times K$ の最尤推定量 $\hat{\Sigma}_u$ を求めれば良いのです。

ここでは`maxLik::maxLik()`関数と`base::optim()`関数を用い、構造VARモデル（SVAR）を推定するためのRスクリプトを書きました。なお、「base」はRインストール時に自動的にインストールされるパッケージを意味します。したがって別途インストールをする必要はありません。

1. `est_SVAR_maxLik.R` --- `maxLik::maxLik()`関数を使って構造VARモデル (SVAR) を推定する。
2. `est_SVAR_optim.R` --- `base::optim()`関数を使って構造VARモデル (SVAR) を推定する。

このRスクリプトは、フォルダ「`SVAR_ML`」に入っており、データセット「[Stock](#)」を使います。

2.2 残差回帰

次の回帰では残差回帰の理論通りの結果が得られます。残差回帰のテクニックを推定に適用するものであり、具体的にはリンゴの需要関数を推定します。理論通りの結果が得られる具体例でもって残差回帰を実感してください。

フォルダ「`data_apple_JP`」には次のファイルが入っています。

1. `apple_JP.csv` --- データセット (c s vフォーマット)
2. `info_data_apple_JP.pdf` --- データの説明書 (PDFファイル)
3. `resid_reg_OK.R` --- 残差回帰のためのRスクリプト

第2章や第10章に示した単一方程式回帰モデルは当該データを使っています。2 t ルールを含め、いくつかの関連Rスクリプトを含めています。

4. `reg_2t_rule.R` --- 2tルールのためのRスクリプト
5. `graph_dot_line.R` --- 需要曲線を描くためのRスクリプト

使用するデータは果物関連の時系列データであり、1980年から2004年までの年周期データであり、53個の変数を含んでいます。その中からリンゴ需要関数用の変数を選び出すといった想定 of データセットになっています。他の果物の需要分析にも使えます。

2.3 ディッキー＝フラー (DF) 分布のグラフ

ディッキー＝フラー (DF) 分布に関し、3つのtau分布を可視化するためのプログラムです。

どのような図が得られるかは図 1 に示すとおりです。

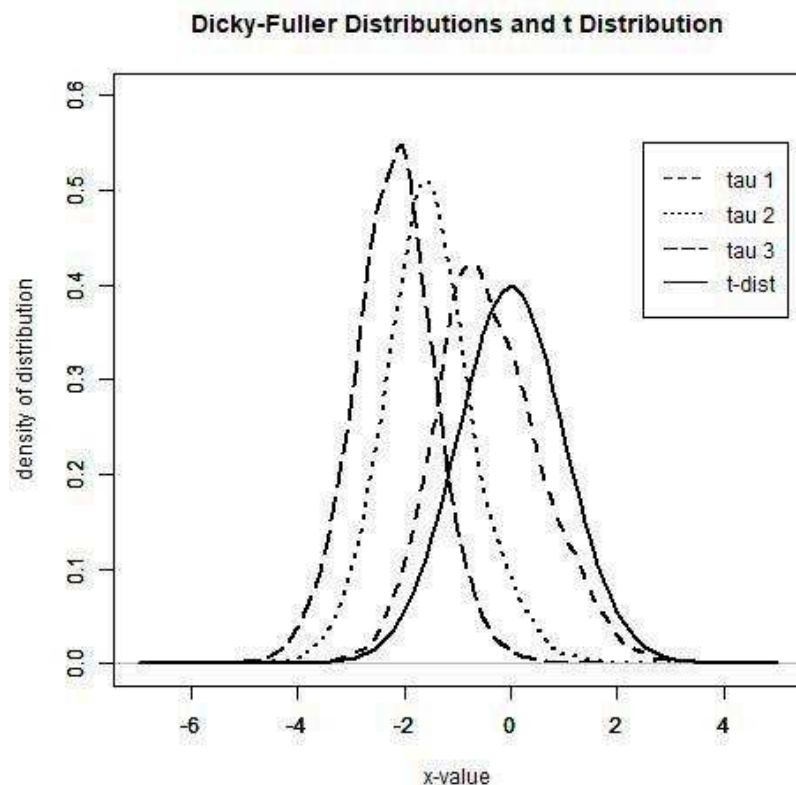


図1：ディッキー=フラー（DF）分布とt分布の比較

次に示す順序で使い、最後のRスクリプトでもって完成したグラフが見られるようになっています。

1. `DF_tau_1.R`
2. `DF_tau_2.R`
3. `DF_tau_3.R`
4. `DF_three_taus.R`

これらはフォルダ「`DF_tau_dist`」に入っています。

図1を得るには次のRパッケージが必要です。

- `lattice` --- 図1に示すような高品質グラフの作成

図1は本書の発行時に欲しいと思っていた図ですが、その当時は見つけることも出来ないし、自分で作成することも出来ませんでした。本書と同時期に発行された下記の書籍（p.206-207）に作成法が記載されていることに気づき、その説明に基づき図1を作成しました。

Enders, Walter (2019), 新谷元嗣・藪友良訳, 『実証のための計量時系列分析』, 有斐閣.

2.4 標準的な分布表

フォルダ「`stat_tables`」には次のRスクリプトが入っています。分布表枠に含める数値情報は、手作業によって変更したほうが良いようになっています。若干の手作業を

加え、完成させてください。その部分をプログラム化すると、煩雑なプログラムになる感じです。

1. `normal_table.R` --- 標準正規分布表を作成します。
2. `chisq_table.R` --- カイ2乗分布表を作成します。
3. `t_table.R` --- t 分布表を作成します。
4. `F_table.R` --- F分布表を作成します。

3. 練習用データ

VAR分析の練習に使えるデータを示します。Rパッケージ「vars」や「svars」に含まれているデータも練習に使えますが、そのようなデータはここで再び示すようなことはしません。学術論文や教科書に使われ、公開されているデータが対象になります。著者が編集したデータも含めています。

3.1 データセット「data_Stock」

データセット「`data_Stock.csv`」はStock and Watson (2001)に使用されたデータです。米国マクロ経済の四半期データであり、（インフレ率、失業率、利子率）からなる3変量を含みます。データ収録期間は1954Q1 - 2000Q4です。その元となるデータは次のウェブページから入手しました。

- <http://www.princeton.edu/~mwatson/publi.html> --- プリンストン大学 Watson博士のウェブページ

このページを開き、次の論文のところへ移動してください。

Vector Autoregressions (with James H. Stock), Journal of Economic Perspectives Fall 2001, Vol. 15, No. 4, pp. 101-116.

この論文に使われている（物価指数GDPD、失業率LHUR、利子率FYFF）のデータ入手し、論文の結果を自分で再現することが考えられます。

物価指数GDPDが四半期データ、失業率LHURと利子率FYFFが月周期データであること、さらに観測期間が異なることから、これらのデータは共通の期間からなる四半期データに編集する必要があります。このようなデータ編集は面倒と思われる場合は、既に編集済みの四半期データ「`data_Stock.csv`」を使うことが考えられます。

「`data_Stock.csv`」を使う場合でも、次の3変量はつくる必要があります。

1. インフレ率 $dp(t) = 400 * \log(GDPD(t)/GDPD(t-1))$ 。価格レベルは $p(t)$ と表記する。
2. 失業率 $u(t) = LHUR(t)$
3. 利子率 $R(t) = FYFF(t)$

インフレ率は年率でパーセント表示にするために400を掛けています。詳しいデータの計算法については論文に記載されています。特にp.102のフットノートが役立ちます。「sample period」は、論文（p.105）に示されている如く、「1960Q1-2000Q4」になります。

この論文はVAR分析の練習資料として有用です。有名なVAR実証分析はVECモデルを使用し、複雑な内容になっている事例が多く、練習用資料に適していないのが実情です。一方、この論文は3変量からなる誘導VARモデルを使い、基本的なVAR分析「インパルス応答分析」「予測誤差の分散分析」「グランジャー因果性検定」を行っています。VAR分析の練習に対する「答え」が示されていること、簡単なVARモデルであることから、この論文はVAR分析の練習に適しています。特に最初の練習資料としては最適でしょう。

3.2 データセット「data_KPSW」

これは King, Plosser, Stock and Watson (1991)に使用されたデータであり、VAR分野では「KPSWデータ」として知られています。米国マクロ経済の四半期データであり、データ収録期間は1947Q1 - 1988Q4です。変数の個数は最大で6個であり、(消費C、投資I、生産量Y、貨幣供給量M、利子率R、インフレ率DP) からなります。本書で使用した3変量データ「[data_KPSW.csv](#)」は別の場所から入手しましたが、元々の出所はKPSWデータです。KPSWデータは次のウェブページから入手できます。

- <http://www.princeton.edu/~mwatson/publi.html> --- プリンストン大学 Watson博士のウェブページ

このページを開き、次の論文のところへ移動してください。

Stochastic Trends and Economic Fluctuations (with Robert King, Charles Plosser, and James Stock), American Economic Review, Vol. 81, No. 4, (September 1991), pp. 819-40.

ダウンロードしたファイルの中には数個のデータセットが含まれていますが、(消費C、投資I、生産量Y、貨幣供給量M、利子率R、インフレ率DP) からなるデータセット「CIYMRDP.PRN」がメインになります。「CIYMRDP.PRN」からデータを読み取り、King et al. (1991)のFigure 1を再現するためのRスクリプトを作成し、ここに提供しています。文言や表現法の差異はありますが、Figure 1が再現されていると言えます。

1. [kpsw_zip](#) --- Watson博士のウェブページからダウンロードしたファイルであり、KPSWデータを含みます。
2. [read_KPSW_data.R](#) --- 「kpsw_zip」に含まれているKPSWデータを読み取り、King et al. (1991)のFigure 1を描きます。
3. [data_KPSW.csv](#) --- 別の場所からダウンロードしたKPSWデータであり、本書11章で使用しました。

これらはフォルダ「[data_KPSW](#)」に入っています。

King et al. (1991)のFigure 1が再現できたので、Figure 2もRでもって再現することを試みました。しかし、R関数の組み合わせだけでもってFigure 2を再現することは難しいようです。したがってFigure 2はRの練習対象にならないと判断しました。もちろん、一から始めるようなRスクリプトを自分で書けば、Figure 2は再現できるでしょう。その方向性へ進む場合でも不明な点があり、Figure 2の再現は容易ではありません。

話が変わりますが、King et al. (1991)のTable 4は、統計解析ソフトRが計算する分散寄与度（RVC）に対応していません。したがってTable 4もRの練習対象になりません。

King et al. (1991) はRの練習に対する「答え」が示されているとは言い難いですが、それでも幾らかの練習に役立ちます。少なくとも単位根検定や共和分検定に対する「答え」は示されています。

3.3 データセット「data_ISLM」

IS-LMモデルのためのデータセットを作成し、「data_ISLM.csv」と名付けました。それは日本マクロ経済の四半期データであり、50個ぐらいの変数が含まれています。時間的なカバー範囲は次のようになっています。

1. 下を示す変数を除き、全ての変数： 1994Q1-2022Q3, 115 time periods.
2. マネーストック関連の変数： 2003Q2-2022Q3, 78 time periods.
3. 消費者物価指数： 2001Q1-2022Q3, 91 time periods.

データの入手源は次のとおりです。

1. GDP統計の変数： 内閣府経済社会総合研究所
2. マネーストックと利子率に関する変数： 日本銀行
3. 消費者物価指数： 政府統計ポータルサイト「e-Stat」
4. 鉱工業生産指数： 経済産業省

データや関連資料はフォルダ「**data_ISLM**」に入っています。

このフォルダには次のファイルが含まれています。

- **data_ISLM.csv** --- データセット(csvフォーマット)
- **データ作業表.xlsx** --- データ編集のためのExcelファイル
- **info_data_ISLM.pdf** --- データの説明書 (PDFファイル)
- **plot_data.R** --- データを読み取り、幾らかのグラフを描くためのRスクリプト

4. ディレクトリ表記に関する説明

プログラミング初心者を対象とし、ディレクトリ（フォルダ）の表記法に関する説明を行います。Rスクリプト（プログラム）を書く人にとっては自明な内容であり、ざーと読み流してください。

まず、ディレクトリも、ファルダも、ファイルの保存場所といった意味では同じです。フォルダは入れ物（容器）といった意味合いが強くなります。一方、ディレクトリはツリー構造を持ったファイルの組織体といった意味合いが強くなります。

ダウンロードしたプログラムが問題なく動くためには、ディレクトリやフォルダと呼ばれるファイル保存場所の表記が重要であり、次の一致が必要です。

プログラムに記載されているディレクトリ（フォルダ） = コンピュータ上の実際のディレクトリ（フォルダ）

これだけです。これだけでは味も素っ気もないので、もう少し詳しく説明します。ここからは具体例をもって説明します。

まず、当サイトからダウンロードしたファイルを入れるための保存場所（フォルダ）を作ります。「book_VAR」と名付けたフォルダを作り、そこにダウンロードしたファイルを入れたとします。フォルダ「book_VAR」の場所は次の場所とします。

```
C:/data_works/time_multi_eqs/book_VAR
```

「C:/data_works/time_multi_eqs」の部分は、自分のコンピュータ環境に応じて選んでください。

ダウンロードしたファイルの中には第11章で用いたプログラム

「[irf_two_methods_VAR.R](#)」が入っています。このプログラム

「[irf_two_methods_VAR.R](#)」は、フォルダ「Chap_11/R_prg」に入っていますから、それを絶対参照の形で表記すると、次のようになります。

```
C:/data_works/time_multi_eqs/book_VAR/Chap_11/R_prg/irf_two_methods_VAR.R
```

第14章で用いたプログラム「[fevd_two_methods_VAR.R](#)」は、フォルダ

「Chap_14/R_prg」に入っていますから、それを絶対参照の形で表記すると、次のようになります。

```
C:/data_works/time_multi_eqs/book_VAR/Chap_14/R_prg/fevd_two_methods_VAR.R
```

繰り返しになりますが、「[R_prg](#)」と名付けた各フォルダに対し、次の3つのサブフォルダを作っています。

1. data
2. data_temp
3. outputs

サブフォルダ「[data_temp](#)」は、研究やプロジェクトが終了すると、削除するデータの保管場所といった意味で作っています。どのサブフォルダが必要かは個々のプログラムによって異なります。説明の単純化のため、全てのフォルダに対し、3つのサブフォルダを作っています。

次はプログラムに記載されているディレクトリ（フォルダ）構造を変更します。フォルダ「Chap_11/R_prg」内のプログラム「[irf_two_methods_VAR.R](#)」を例にとると、`setwd()`のところの内容を次のように変更します。

```
setwd("C:/book_VAR/data_works/Pesaran_EL/R_prg")
↓
setwd("C:/data_works/time_multi_eqs/book_VAR/Chap_11/R_prg")
```

この操作は、「プログラムに記載されているディレクトリ（フォルダ）」 = 「コンピュータ上の実際のディレクトリ（フォルダ）」となるようにするための変更です。

フォルダ「Chap_11/R_prg」のサブフォルダ「[data](#)」や「[outputs](#)」を絶対参照の形で表示すると、次のようになります。

```
C:/data_works/time_multi_eqs/book_VAR/Chap_11/R_prg/data
C:/data_works/time_multi_eqs/book_VAR/Chap_11/R_prg/outputs
```

一方、working directory（現在位置）から見て、下のフォルダ、上のフォルダ、隣のフォルダといった形でフォルダ指定（一般的にはオブジェクト指定）を行う方式は**相対参照**と呼ばれます。working directory（現在位置）が「Chap_11/R_prg」であるとき、そのサブフォルダ「**data**」や「**outputs**」を相対参照で表記すると、次のようになります。

```
./data
./outputs
```

working directory「Chap_11/R_prg」から見て、「Chap_11/R_prg」内のサブフォルダが「data」や「outputs」といった名前になっているといった意味です。このように相対参照を利用すると、フォルダ指定の表記が簡単できます。フォルダ指定のために、長々と続く絶対参照を書く必要がなくなります。

setwd()でもってworking directoryを指定しない場合は、Rの実行ファイルがある場所がworking directoryになっています。「データが上手く読み取れない」「出力が思うように出力されない」といったトラブルは、相対参照の誤りであるケースが意外と多いものです。このような経験から、私がRスクリプトを書く場合は、setwd()でもってworking directoryを特定化し、それを最初のコマンドとして書くようにしています。つまり、自分の作業場所（現在位置）をコンピュータに伝えてから具体的な作業に入っていきます。このような準備作業は、相対参照に関する指示ミスを少なくします。

次はworking directory「Chap_11/R_prg」内のサブフォルダ「data」や「outputs」の名前や構造を変更した場合の対処法を説明します。

データの保存場所（フォルダ）の名前を「**data**」から「**external_data**」に変更した場合は、プログラム「**irf_two_methods_VAR.R**」における該当部分を変更します。変更前と変更後を示すと、次のような変更を行います。

```
read.csv("./data/data_KPSW.csv")
↓
read.csv("./external_data/data_KPSW.csv")
```

ファイルの保存場所は絶対参照の形で書くことも出来ますが、ここでは相対参照の形で書いています。

画像ファイルの保存場所（フォルダ）を「**outputs**」から「**outputs/irf**」に変更した場合は、プログラム「**irf_two_methods_VAR.R**」における該当部分を変更します。変更前と変更後を示すと、次のような変更を行います。

```
dev.copy(jpeg, file = "./outputs/Persaran_Fig1.jpg")
↓
dev.copy(jpeg, file = "./outputs/irf/Persaran_Fig1.jpg")
```

このような操作も、「プログラムに記載されているディレクトリ（フォルダ）」＝「コンピュータ上の実際のディレクトリ（フォルダ）」となるようにするための変更です。

サブフォルダの相対参照に関し、次の2つは同等です。

```
read.csv("./data/data_KPSW.csv")  
read.csv("data/data_KPSW.csv")
```

一方、working directory（現在位置）から1つ上のディレクトリ（フォルダ）へ移動する場合は「../」の表記を使います。2つ上のディレクトリ（フォルダ）へ移動する場合は「../../」、3つ上のディレクトリ（フォルダ）へ移動する場合は「../../../」の表記を使います。したがってworking directory「[Chap_11/R_prg](#)」から隣のフォルダ「[Chap_14/R_prg](#)」へアクセスする場合は、

```
../../Chap_14/R_prg
```

といった相対参照にします。現在、自分のいる部屋「Chap_11/R_prg」から別の部屋へ移動する場合は、一度、廊下へ出てから別の部屋「Chap_14/R_prg」へ移動する必要があります。コンピュータにおいても同様であり、一度、上のディレクトリ（フォルダ）へ移動し、そこから見える「Chap_14/R_prg」へアクセスするように指示します。1つ上のディレクトリ（フォルダ）なのか、2つ上のディレクトリ（フォルダ）なのかは、個々の状況によって異なりますが、廊下へ出てから別の部屋へ行くという基本は変わりません。

このページの[トップ](#)に戻る。

[オーム社トップページ](#)