1

導入

筆者がRでプログラミングを始めて10年以上になる．この間，R言語の仕組みについて考察し，理解を深めようとするのに十分な時間に恵まれてきた．本書においては，筆者がこの間に身に付けた知識や技術をまとめ，読者が効率的なRプログラミング技法を習得できるように手助けをしたい．ここに至るまで筆者自身も多くのミスや躓きを繰り返してきたが，読者が前轍を踏むことがないよう，さまざまなツールや技術，そして解決のための定石を本書では解説する．これらは読者が遭遇するであろう問題を解決するのにきっと役立つだろう．Rはかなり癖のある言語なので時にイライラさせられるが，それでもRが本質的にはエレガントで美しい言語であり，データ分析や統計処理を実行するのに適した工夫に満ちていることを，本書を通じて示したい．

Rの経験のない読者は，こんな癖のあるプログラミング言語を学んで本当に役に立つのかと疑問に思うかもしれない．こうした疑問に対する筆者なりの回答を以下にいくつか記しておこう．

- Rはフリーでオープンソースのプロジェクトとして開発されており，主要なプラットフォームで利用できる．したがってRを使えば，第三者が簡単に分析を再現できる．
- 多数のパッケージが公開されており，統計モデリングや機械学習，可視化，データの読み込みや前処理のために利用できる．読者が実行しようとするモデルや作成したいグラフィックスは，すでに誰かがパッケージを公開している可能性が高い．そうでなくとも，誰かが試行していて，それを参考にすることができるだろう．
- Rは最先端のツールである．そのため統計学や機械学習の研究者たちが最新の論文の付録としてRのパッケージを公開することが少なくない

ので，最新の統計的技法とその実行方法を即座に試すことができる．

- R はデータ分析特有の処理に狙いを定めた言語である．例えば欠損値への対応や，データを表計算フォーマットで表現する仕組み（データフレーム），さらにはデータからの部分抽出などの方法が用意されている．
- 世界中に R のためのコミュニティがある．つまり R について質問を投げるとエキスパートから回答を得られる場がある．例えば R 全般についての意見交換の場である R-help[1] や，プログラマ向け Q&A サイトである stackoverflow[2] が役に立つ．R-SIG-mixed-models[3] や ggplot2[4] のような特定のテーマに特化したメーリングリストもある[5]．また twitter や LinkedIn に投稿することで R ユーザたちと交流できる．さらには世界各地で R コミュニティが活動している[6]．
- データ分析の結果をプレゼンテーションするための方法が多数用意されている．追加パッケージを導入することで，html や pdf のレポートを簡単に作成できる（例えば **knitr**[7]）．**Shiny**[8] を使うと，インタラクティブな Web アプリケーションを作成することもできる．
- R は関数型プログラミングを土台としている．関数型プログラミングの考え方は，データ分析において遭遇する複雑な課題を解くのにとても役立つ．R に備わった強力で柔軟なツールを使うと，簡潔であるが理解しやすいコードを書くことができる．
- 対話的なデータ分析と統計プログラミングに必須の機能を搭載した統合環境 (IDE) である RStudio[9] が利用できる．
- R はメタプログラミングをサポートしている．R はプログラミング言語であるだけでなく，対話的なデータ分析のための環境でもある．R のメ

---

1) `https://stat.ethz.ch/mailman/listinfo/r-help`
2) `http://stackoverflow.com`
3) `https://stat.ethz.ch/mailman/listinfo/r-sig-mixed-models`
4) `https://groups.google.com/forum/forum/ggplot2`
5) 訳注：SIG は Special Interest Group の略．R 関連ではそれらの一覧を `http://www.r-project.org/mail.html` で確認できる．
6) `http://blog.revolutionanalytics.com/local-r-groups.html`
7) `http://yihui.name/knitr/`
8) `http://shiny.rstudio.com/`
9) `http://www.rstudio.com/products/rstudio/`

タプログラミング機能を通して，驚くほど簡潔で機能的な関数を作成することができる．さらに特定のタスクに適した処理ツール（ドメイン特化言語）を設計するのに適した言語環境が，他ならぬRである．
- C言語やFortran，そしてC++といったパフォーマンスの高い言語とのインターフェイスがRには用意されている．

もちろんRは完全ではない．そもそもRは，そのほとんどのユーザがプログラマではないという問題に立ち向かう必要がある．これには以下のような困難がある．

- 実務などで目にするRのコードは，ほとんどの場合，作成者の差し迫った問題を解決するために書かれており，決してエレガントとはいえず，また実行速度は遅く，さらには理解するのも困難である．しかし，こうした欠点を補うためにコードを書き直すユーザはほとんどいない．
- 他の言語のユーザたちに比べると，Rのユーザたちは過程ではなく結果を重視する傾向にある．またソフトウェア工学で作業を効率化する実践手法に関する知識も断片的である．例えば，ソースコードを管理するためのツールを利用したり，動作テストを自動化するRプログラマも少ない．
- メタプログラミングは諸刃の剣となる．このトリックによってRの多くの関数はコード量が少なくなっているが，そのために理解しづらくなり，またトラブルの種ともなっている．
- 公開されているパッケージには矛盾した実装が多数見受けられる．R本体ですら例外ではない．Rを使うたびに，20年を超える開発の過程で紛れ込んだ問題に突き当たることになる．またRを習得するのが難しいのは，そのつど覚えなければならない例外にあふれているからでもある．
- Rは特に速いというプログラミング言語ではない．実装が悪ければ，それだけRの実行速度は遅くなる．またRは，たいへんなメモリ喰いとして知られている．

筆者からすると，こうした課題があるからこそ，経験を積んだプログラマたちにはRとそのコミュニティに深くコミットする甲斐があるのではない

だろうか．Rのユーザは，特にデータ分析の再現性を保証するためにも，質の高いコードを書くように心がけて欲しいのだが，そのためのスキルが十分に身に付いていないのも事実だ．本書を通じて，読者がRの単なるユーザから能動的なプログラマに変貌を遂げる，あるいは他言語のプログラマがRに貢献しようという意欲を抱いてくれることに筆者は期待している．

1.1　本書が想定する読者層

本書は，異なる2つの読者層を対象としている．

- Rについて中級程度の技能を有しているが，より深く詳しくRを知りたいと考えており，また種類の異なる課題を解決するのに汎用的に役立つ戦略を新たに身に付けたいと考えているプログラマ
- 他のプログラミング言語には通じているが，Rをまさに学習中であり，R特有の挙動を理解したいと考えるプログラマ

本書の内容を細部に至るまで理解するには，すでにRないし別言語で相当量のコードを書いてきた経験が必要かもしれない．しかし詳細まで熟知している必要はない．ただRで関数がどのように使えるのかは把握している必要はある．さらにいえば，いわゆるapply族の関数群（つまりapply()やlapply()など）の仕組みをまだ十分に理解できていなくとも，ある程度は使えるだけの経験はあったほうがよい．

1.2　本書から読者が得られるもの

本書には，Rプログラミングの上級者ならば身に付けておくべきだと筆者が考える技能を解説している．これは広範囲な問題に汎用的に使える良質のコードを書く技能である．本書を読了したとき，ユーザは以下のような能力を身に付けていることだろう．

- Rの基本原理を把握しているであろう．複雑なデータ型を理解しており，これらを効率的に処理する技術を身に付けているだろう．また関数

の原理について深く理解しており，4つのオブジェクト指向システムの仕組みを再認識し，効率的に利用できるだろう．

- 関数型プログラミングの意味とデータ分析にとっての重要性を認識しているであろう．そして既存のツールの使い方が理解でき，必要があれば自身で関数を作成してツールとして使いこなせる技能を習得しているだろう．
- メタプログラミングの長所と短所を心得ているだろう．これにより，非標準評価の原理を使って関数を作成でき，またコード量の少ないエレガントなコードで重要な処理を回せるようになっているだろう．さらにメタプログラミングの危険性を熟知しており，十分に用心深く利用できるようになっているはずだ．
- R で処理が遅く，またメモリが大量に消費されるケースを予測することができ，パフォーマンスのボトルネックを探り当てるプロファイリングが行えるだろう．そして C++ について必要な知識を身に付けており，R の遅い関数を C++ の高速な関数に実装し直すことができるだろう
- R のコードのほとんどを読んで理解する能力が身に付いているだろう．これらに共通する一般的なテクニックがわかり（自分自身で使うかどうかはまた別だが），他のプログラマの作成したコードを検証できるだろう．

1.3 メタテクニック

R プログラマとしてスキルを磨くのに役立つメタテクニックが2つある．ソースコードを読むことと，科学的な思考法を身に付けることだ．

ソースコードを読むのが重要なのは，自分自身がより良いコードを書けるようになるからだ．まずは自分自身がよく利用しているパッケージや関数のソースコードを眺めることから始めるといいだろう．こうしたコードには，自身でも応用したくなる部分があるだろうし，やがてコードの質を向上させるキモのようなものが感覚的にわかるようになるだろう．こうしたソースコードの中には，必要性が理解できず，自分の好みにも合わない部分もあるかもしれない．こうしたコードであっても，自身にとって良い，あるいは悪

いコードを見分ける感覚を育てるには役に立つ．

科学的な思考法はRを習得する上で大変重要である．筋道をつけて考え，仮説を構築し，実験を立案して実行する．そして結果を記録する．こうした経験の積み重ねが役に立つ．問題に突き当たって他人に助力を求める場合でも，手順を正確に説明できるようになっているからだ．正しい解決策を知ったときには，自分自身の世界観を抵抗なく更新する準備ができているようになる．筆者個人も，課題を他人に順序立てて説明してみると（これにはstackoverflowに投稿されている「再現性のヒント[10)]」を参照するといいだろう），自分自身で解決策を見つけてしまう場合が多い．

1.4　推奨される文献

Rは比較的新しい部類の言語なので，参照できる情報源もまだ豊富とはいえない．筆者自身はむしろ他のプログラミング言語に関する情報源から学ぶことが多かった．またRには関数型言語とオブジェクト指向言語(OO)の2つの側面がある．これらがRにどのように組み込まれているのかを知ることで，他のプログラミング言語に関する既存の知識がブラッシュアップされて，自身の技能のどこに改善の余地があるかを自覚できるだろう．

Rのオブジェクト体系の仕組みを理解するにあたっては，Harold AbelsonとGerald Jay Sussmanによる *The Structure and Interpretation of Computer Programs* (SICP)[11)]がとても役に立つ．これは簡潔ではあるが深い技術書である．一読して，筆者はようやく自身でオブジェクト指向システムをデザインできそうな気になった．同書は，Rのオブジェクト指向システムで標準となっている総称関数スタイルとその利点，さらには弱点を筆者が理解する道標となってくれた．SICPは関数型プログラミングについても多くのページを割いており，シンプルな関数を作成し，これらを結合することで強力な機能が実現できることを解説している．

---

10) `http://stackoverflow.com/questions/5963269/how-to-make-a-great-r-reproducible-example`

11) 訳注：ハロルド・エイブルソン著，和田英一訳『計算機プログラミングの構造と解釈』翔泳社，2014年．

Rを他のプログラミング言語と比較した場合のトレードオフを知るには，Peter van Roy と Sef Haridi による *Concepts, Techniques and Models of Computer Programming*[12] を読むのが一番だ．同書によって，筆者はRの**コピー修正** (copy-on-modify) セマンティクス[13] が，コードを論理的に考察する手助けとなることに気が付いた．ただ現在の実装は効率的とはいえないのだが，これは改善可能だろう．

より良いプログラマになりたいのならば，Andrew Hunt と David Thomas による *The Pragmatic Programmer*[14] を読むべきだ．同書には，どのプログラミング言語でも役立つアドバイスが豊富にあり，より良いプログラマとなる近道だろう．

1.5 助言を得る

Rの作業で行き詰まり，解決策を見い出せないような場合に頼るべき情報源として，本書執筆時点で推奨できるサイトが2つある．stackoverflow と R-help メーリングリスト[15] である．どちらも優れた回答を期待できる情報源ではあるが，それぞれに独自の文化と規約があるので，注意が必要だ．質問を投稿するのであれば，少し時間を割いてサイト規約に目を通しておくべきだ．

ここで，いくつかアドバイスをしておこう．

- 問題が生じた際に利用していたRとパッケージが最新のバージョンであることを確認しよう．読者が遭遇した問題はすでに修正されているかもしれないからだ．
- 問題を再現できるサンプル（再現可能な例）を用意しよう．実はサンプルの準備中に自身で問題の解決方法を発見できることも少なくない．

---

[12] 訳注：セイフ・ハリディ著，羽永洋訳『コンピュータプログラミングの概念・技法・モデル』翔泳社，2007年．

[13] 訳注：本書第18章「メモリ」を参照されたい．

[14] 訳注：アンドリュー・ハント著，村上雅章訳『達人プログラマー – システム開発の職人から名匠への道』ピアソンエデュケーション，2007年．

[15] `https://www.r-project.org/mail.html`

- 類似の質問がすでに投稿されていないかを確認しよう．すでに回答が示されていることも多いので，時間の節約になるだろう．

1.6　謝辞

筆者はこれまでR-helpメーリングリストや，最近ではstackoverflow[16]の回答者たちから多くの助言を得てきた．すべての名前を挙げることはできないが，ここでは特にLuke Tierney, John Chambers, Dirk Eddelbuettel, JJ Allaire, Brian Ripley らに，貴重な時間を割いて筆者の誤解を正してくれたことに感謝したい．

本書はオープンソースの著作であり，Github[17]上でも公開されており，また twitter[18]を通じて広報してきた．本書の成立はコミュニティあってのものである．多くの方々が草稿に目を通し，誤植を修正し，内容についても改善点を指摘してくれた．こうした助力がなければ本書は完成していない．特に本書を通読し，多くのミスを指摘してくれたPeter Li に感謝したい．もちろん，他の多くの方々にも，この機会に感謝を伝えたい．紙面の許す範囲で，名前（あるいはアカウント）を挙げる．

Aaron Schumacher, @aaronwolen, Aaron Wolen, @absolutelyNoWarranty, Adam Hunt, @agrabovsky, @ajdm, @alexbbrown, @alko989, @allegretto, @AmeliaMN, @andrewla, Andy Teucher, Anthony Damico, Anton Antonov, @aranlunzer, @arilamstein, @avilella, @baptiste, @blindjesse, @blmoore, @bnjmn, Brandon Hurr, @BrianDiggs, @Bryce, @Carson, @cdrv, Ching Boon, @chiphogg, Christopher Brown, @christophergandrud, C. Jason Liang, Clay Ford, @cornelius1729, @cplouffe, Craig Citro, @crossfitAL, @crowding, @crtahlin, Crt Ahlin, @cscheid, @csgillespie, @cusanovich, @cwarden, @cwickham, Daniel Lee, @darrkj, @Dasonk, David Hajage, David LeBauer, @dchudz, dennis feehan, @dfeehan, Dirk Eddelbuettel, @dkahle, @dlebauer, @dlschweizer, @dmontaner, @dougmi-

[16] R関連の質問は `http://stackoverflow.com/questions/tagged/r` で参照できる．
[17] `https://github.com/hadley/adv-r/`
[18] `https://twitter.com/hadleywickham`

tarotonda, @dpatschke, @duncandonutz, @EdFineOKL, @EDiLD, @eipi10, @elegrand, @EmilRehnberg, Eric C. Anderson, @etb, @fabian-s, Facundo Muoz, @flammy0530, @fpepin, Frank Farach, @freezby, @fyears, @garrettgman, Garrett Grolemund, @gavinsimpson, @gggtest, Gken Eraslan, Gregg Whitworth, @gregorp, @gsee, @gsk3, @gthb, @hassaad85, @i, Iain Dillingham, @ijlyttle, Ilan Man, @imanuelcostigan, @initdch, Jason Asher, @jasondavies, Jason Knight, @jastingo, @jcborras, Jeff Allen, @jeharmse, @jentjr, @JestonBlu, @JimInNashville, @jinlong25, JJ Allaire, Jochen Van de Velde, Johann Hibschman, @johnbaums, John Blischak, @johnjosephhorton, John Verzani, Joris Muller, Joseph Casillas, @juancentro, @kdauria, @kenahoo, @kent37, Kevin Markham, Kevin Ushey, @kforner, Kirill Mller, Kun Ren, Laurent Gatto, @Lawrence-Liu, @ldfmrails, @lgatto, @liangcj, Lingbing Feng, @lynaghk, Maarten Kruijver, Mamoun Benghezal, @mannyishere, @mattbaggott, Matthew Grogan, @mattmalin, Matt Pettis, @michaelbach, Michael Kane, @mjsduncan, @Mullefa, @myqlarson, Nacho Caballero, Nick Carchedi, @nstjhp, @ogennadi, Oliver Keyes, @otepoti, Parker Abercrombie, @patperu, Patrick Miller, @pdb61, @pengyu, Peter F Schulam, Peter Lindbrook, Peter Meilstrup, @philchalmers, @picasa, @piccolbo, @pierreroudier, @pooryorick, @ramnathv, Ramnath Vaidyanathan, @Rappster, Ricardo Pietrobon, Richard Cotton, @richardreeve, R. Mark Sharp, @rmflight, @rmsharp, Robert M Flight, @RobertZK, @robiRagan, Romain Franois, @rrunner, @rubenfcasal, @sailingwave, @sarunasmerkliopas, @sbgraves237, @scottko, @scottl, Scott Ritchie, Sean Anderson, Sean Carmody, Sean Wilkinson, @sebastian-c, Sebastien Vigneau, @shabbychef, Shannon Rush, Simon O'Hanlon, Simon Potter, @SplashDance, @ste-fan, Stefan Widgren, @stephens999, Steven Pav, @strongh, @stuttungur, @surmann, @swnydick, @taekyunk, @talgalili, Tal Galili, @tdenes, @Thomas, @thomasherbig, @thomaszumbrunn, Tim Cole, @tjmahr, Tom Buckley, Tom Crockett, @ttriche, @twjacobs, @tyhenkaline, @tylerritchie, @ulrichatz, @varun729, @victorkryukov, @vijaybarve, @vzemlys, @wchi144, @wibeasley, @WilCrofter, William Doane, Winston Chang, @wmc3, @word-

nerd, Yoni Ben-Meshulam, @zackham, @zerokarmaleft, Zhongpeng Lin.

1.7　本書での表記

本書を通じて関数はタイプライター体で`f()`のように丸括弧を添えて表記している．変数名や関数の引数，あるいはフォルダパスについても，`g`や`h/`のようにタイプライター体を利用している[19]．

比較的大きなコードブロックでは実行命令とその出力が混在している．そこで出力にはコメント記号 (`#`) を加えているので，本書の電子版[20]に掲載されたコードをコピーして実行する妨げにはならないはずだ．出力のコメントには`#>`を利用し，通常のコメント (`#`) とは区別している．

1.8　本書の公開にあたって

本書は RStudio[21] で R markdown パッケージ[22] を使って書かれている．Rmarkdown を html や pdf に変換するのには **knitr** パッケージ[23] と pandoc[24] を利用した．サイトへの公開では jekyll[25] と bootstrap[26] を利用し，Amazon の S3[27] との連携には travis-ci[28] を活用している．本書の全ソースは Github[29] で公開している．なお原書でコード表示には inconsolata フォント[30] を利用した．

[19] 訳注：翻訳では原書のスタイルを踏襲しているが，パッケージについてはゴシック体を採用するなど，フォントスタイルについては多少変更を加えた．

[20] `http://adv-r.had.co.nz`

[21] `https://www.rstudio.com/products/rstudio/`

[22] `http://rmarkdown.rstudio.com/`

[23] `http://yihui.name/knitr/`

[24] `http://pandoc.org/`

[25] `http://jekyllrb.com/`

[26] `http://getbootstrap.com/`

[27] `http://aws.amazon.com/jp/s3/`

[28] `https://travis-ci.org/`

[29] `https://github.com/hadley/adv-r`

[30] `http://levien.com/type/myfonts/inconsolata.html`