

改訂3版

R言語

逆引き
ハンドブック

石田 基広◆著

統計解析の定番ツール「R言語」の
基本から活用までを網羅的に解説!

R言語の機能を目的から

探せる!

バージョン
3.3.0
に対応!

 24時間無料でサンプルデータをダウンロードできます。

 C&R研究所

C&R研究所について

C&R研究所は新潟市にある出版社です。ユニークな社風や教育方針は新聞やテレビなどで紹介されたりします。詳細については、次のWebサイトでご覧いただくことができます。

www.c-r.com

また、新潟本社には2代目会社犬「ラッキー」がいます。名刺を持つ正式な社員として広報部に勤務しつつ、セラピードッグとして社内のメンタルヘルスにも貢献しています。

●会社犬「ラッキー」



改訂3版

R言語

逆引き
ハンドブック

石田 基広◆著

■権利について

- 本書に記述されている社名・製品名などは、一般に各社の商標または登録商標です。
- 本書では™、©、®は割愛しています。

■本書の内容について

- 本書は著者・編集者が実際に操作した結果を慎重に検討し、著述・編集しています。ただし、本書の記述内容に関わる運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。

■サンプルについて

- 本書で紹介しているサンプルは、C&R研究所のホームページ (<http://www.c-r.com>)、または筆者のGithubからダウンロードすることができます。ダウンロード方法については、5ページを参照してください。
- サンプルデータの動作などについては、著者・編集者が慎重に確認しております。ただし、サンプルデータの運用結果にまつわるあらゆる損害・障害につきましては、責任を負いませんのであらかじめご了承ください。
- サンプルデータの著作権は、著者およびC&R研究所が所有します。許可なく配布・販売することは強く禁止します。

●本書の内容についてのお問い合わせについて

この度はC&R研究所の書籍をお買いあげいただきましてありがとうございます。本書の内容に関するお問い合わせは、「書名」「該当するページ番号」「返信先」を必ず明記の上、C&R研究所のホームページ(<http://www.c-r.com/>)の右上の「お問い合わせ」をクリックし、専用フォームからお送りいただくか、FAXまたは郵送で次の宛先までお送りください。お電話でのお問い合わせや本書の内容とは直接的に関係のない事柄に関するご質問にはお答えできませんので、あらかじめご了承ください。

〒950-3122 新潟県新潟市北区西名目所4083-6 株式会社 C&R研究所 編集部
FAX 025-258-2801
『改訂3版 R言語逆引きハンドブック』サポート係

III PROLOGUE

Rは統計解析のためのデスクトップアプリケーションとして世界中で広く使われています。Rは数多くのデータ分析手法をサポートし、また、高品質のグラフィックスを作成することもできます。さらにRはプログラミング言語として設計されているため、ユーザー自身が機能を拡張することができます。実際、世界中のRユーザーによって開発された拡張機能（パッケージ）がインターネット上に多数公開されており、誰でも自由に使うことができます。Rの拡張機能はデータ解析やグラフィックス作成にとどまらず、プレゼンテーション作成支援など、ユーザーのさまざまな要望に応じてくれます。

このようにRは多種多様なオプションを提供していますが、その範囲が多岐に及ぶため、全体像をとらえるのは容易ではありません。また、ユーザーが必要とする機能をピンポイントで探し出せないこともあります。Rには詳細なヘルプやドキュメントが用意されていますが、国際的に開発されているアプリケーションであるため、解説は英語で公開することが原則とされています。このため、特に日本のユーザーにとって、Rは敷居が高いと感じられるようです。

本書は「Rで目的とする処理を実行するには何をどうすればよいのか?」に対する答えをセクション見出しとして、これに対する解説と実行例を掲載しています。特に重要な項目についてはコラムとして取り上げました。また、随所に、あまり知られていない便利機能を紹介しています。本書の内容は初心者だけでなく、中級者の参考にもなるはずです。本書が読者のステップアップのお役に立てば幸いです。

本書は改訂2版の再改訂版です。Rは2013年の春にバージョン3.0.0に移行し、新規機能が多数追加されました。また、分析作業を効率化するためのパッケージも新規に公開されました。今回の改訂では、こうした新機能についても解説を加えています。たとえば、高速なデータ処理機能（「dplyr」パッケージ）の解説を新規セクションとして追記したほか、RStudioやプレゼンテーション作成機能についても加筆しました。さらに、全体を通して、解説やコードの見直しを行っています。なお、本書の上級編として2013年に『R言語上級ハンドブック』も刊行されました。そちらも、ぜひ、ご参照ください。

最後になりましたが、本書の執筆をサポートしてくださったC&R研究所の皆さんに、この場をかりて御礼を申し上げます。

2016年5月

著 者

本書について

📌 本書の執筆環境と動作環境

本書では、次のような開発環境を前提にしています。

- R(3.0.0、3.0.1、3.0.2、3.0.3、3.1.0、3.1.1、3.1.2、3.1.3、3.2.0、3.2.1、3.2.2、3.2.3、3.3.0)

上記のRを起動させたOSの環境は次の通りです。

- Windows 7、Windows 8、Windows 10
- Mac OS X 10.7、10.8、10.9、10.10
- Ubuntu 12.04、14.04、15.10

📌 サンプルコードの表記について

本書の表記に関する注意点は、次のようになります。

▶ 「\」(バックスラッシュ)と「¥」の使い分けについて

Rのコンソール上では、Windowsでも「¥」が「\」(バックスラッシュ)で表示されます。本書では、原則としてWindowsのフォルダ構成を表記するときは「¥」、それ以外のサンプルコードなどでは「\」(バックスラッシュ)で表記しています。

▶ サンプルコードの中の▼について

本書に記載したサンプルプログラムは、誌面の都合上、1つのサンプルプログラムがページをまたがって記載されていることがあります。その場合は▼の記号で、1つのコードであることを表しています。

🔴 サンプルファイルのダウンロードについて

本書のサンプルデータは、C&R研究所のホームページからダウンロードすることができます。
本書のサンプルを入手するには、次のように操作します。

- ① 「<http://www.c-r.com/>」にアクセスします。
- ② トップページ左上の「商品検索」欄に「201-3」と入力し、[検索] ボタンをクリックします。
- ③ 検索結果が表示されるので、本書の書名のリンクをクリックします。
- ④ 書籍詳細ページが表示されるので、[サンプルデータダウンロード] ボタンをクリックします。
- ⑤ 下記の「ユーザー名」と「パスワード」を入力し、ダウンロードページにアクセスします。
- ⑥ 「サンプルデータ」のリンク先のファイルをダウンロードし、保存します。

サンプルのダウンロードに必要な ユーザー名とパスワード

ユーザー名 **k3rrev**
パスワード **d9w3p**

※ユーザー名・パスワードは、半角英数字で入力してください。また、「J」と「j」や「K」と「k」などの大文字と小文字の違いもありますので、よく確認して入力してください。

🔴 サンプルファイルの利用方法について

ダウンロードしたファイルはzip形式で圧縮されています。解凍してご使用ください。

サンプルコードのファイルは、開発環境別にWindows用の「Windows」と、Mac/Ubuntu用の「UTF-8」のフォルダに分かれています。

開発環境別のフォルダ内は、CHAPTER別のフォルダが作成されており、その中にはCHAPTERのサンプルコードをまとめたスクリプトファイルのほか、必要なデータなどが収録されています。

スクリプトファイルは、拡張子が「.R」のファイルで提供されます。スクリプトファイルはRのメニューから[ファイル]→[スクリプトを開く]を選択して表示されるダイアログボックスから読み込むことができます。

スクリプトファイルを読み込むと、「Rエディタ」が起動するので、実行するコードの部分を選択して、右クリックで表示されるメニューから[カーソル行または選択中のRのコードを実行]を選択するなどしてコードを実行してください。

また、筆者のGithubからダウンロードすることもできます。RStudioと連携させると、最新の修正を簡単に取り込むことができ便利です(52ページ参照)。

● Windows用

URL https://github.com/IshidaMotohiro/R_Handbook_Sjis

● Mac用 / Linux用

URL https://github.com/IshidaMotohiro/R_Handbook_UTF8

CHAPTER 01 R言語の基礎

001	Rの基礎知識	32
002	Rのインストール	34
003	Rの起動と終了	38
	COLUMN ■ 32bit版と64bit版を選択して起動する	
004	Rコンソールとスクリプトの実行	40
	COLUMN ■ 「history」関数の利用	
005	Windows版のRの画面に関する注意点	43
006	Rの環境設定ファイル	45
	COLUMN ■ Rの起動時と終了時の処理を指定する	
007	Rのセッション用オプションの指定	48
008	作業フォルダの設定	50
009	利用したい機能の検索	51
010	Rの統合環境「RStudio」を使う	52
011	パッケージのインストール	67
	COLUMN ■ : 演算子によるパッケージのアクセス	
012	ヘルプの利用	75
013	ビニエツ(簡易マニュアル)の利用	80
	COLUMN ■ 「demo」関数を使用したデモの確認	
	COLUMN ■ 「example」関数を使用した利用例の確認	
014	Rや統計解析に関する情報の収集	82

CHAPTER 02 オブジェクトの基礎

015	オブジェクトの基礎知識	84
	COLUMN ■ 読みやすいコードの書き方	
016	関数の利用と使用頻度の高い関数	86
017	オブジェクトのタイプとデータ構造	88
018	クラスとオブジェクト	91
019	オブジェクトの生成	94
020	オブジェクトの属性の確認／変更	96
021	言語オブジェクトの作成	99

022	オブジェクトなどのメモリ管理	102
	COLUMN ■ 32bit版と64bit版でのメモリサイズの違い	
	COLUMN ■ メモリの再割り当て	

CHAPTER 03 ベクトルの基礎

023	ベクトルの基礎知識	106
024	演算子とは	109
025	数値ベクトルを初期化する	110
	ONEPOINT ■ 数値ベクトルを初期化するにはデータ型別に用意された関数を使用する	
	COLUMN ■ 初期値を16進法表現で指定する	
	COLUMN ■ あらかじめ要素数が増えるベクトルを初期化するときの注意点	
	COLUMN ■ 複素数での値を指定した初期化	
	COLUMN ■ 関数のまとめ	
026	規則性のある数列を作成する	113
	ONEPOINT ■ 規則性のある数列を作成するには 「:」で初項と終項を指定するか「seq」関数を使用する	
	COLUMN ■ 整数のパターンを繰り返す数列を作成する	
	COLUMN ■ 関数のまとめ	
027	要素を繰り返した数列を作成する	115
	ONEPOINT ■ 要素を繰り返した数列を作成するには「rep」関数を使用する	
	COLUMN ■ 「system.time」関数の出力	
	COLUMN ■ 関数のまとめ	
028	ランダムなベクトルを作成する	117
	ONEPOINT ■ あるベクトルからランダムに抽出するには「sample」関数を使用する	
	COLUMN ■ 復元抽出を指定する	
	COLUMN ■ 抽出条件を指定する	
	COLUMN ■ 関数のまとめ	
029	数値ベクトルを区間分割する	119
	ONEPOINT ■ 数値ベクトルを区間分割するには「cut」関数を使用する	
	COLUMN ■ 「cut」関数の利用法	
	COLUMN ■ 「breaks」引数で指定された区間の下限と上限	
	COLUMN ■ 上限あるいは下限を正しく含める	
	COLUMN ■ 関数のまとめ	
030	標本などのベクトルを標準化する	122
	ONEPOINT ■ 標本などのベクトルを標準化するには「scale」関数を使用する	
	COLUMN ■ 関数のまとめ	
031	数値クラスを変更する	124
	ONEPOINT ■ 整数と実数、複素数のデータ型を明示的に指定するには 「as.～」関数を使用する	
	COLUMN ■ 「identical」関数でオブジェクトとしての同一性を調べる	
	COLUMN ■ 関数のまとめ	

032 数値を文字列に変更する 126

ONEPOINT ■ 数値を文字に変換するには書式指定関数を使用する

COLUMN ■ 「format.info」関数の利用

COLUMN ■ 「sprintf」関数の利用

COLUMN ■ 関数のまとめ

033 数値を比較する 129

ONEPOINT ■ 数値の比較には==演算子や「all.equal」関数を使用する

COLUMN ■ 「all.equal」関数による数値比較

COLUMN ■ 「zapsmall」関数による誤差処理

COLUMN ■ 「all」関数と「any」関数でベクトル行列の各要素を比較する

COLUMN ■ 関数のまとめ

CHAPTER 04 ベクトルの操作

034 文字列ベクトルを作成する 134

ONEPOINT ■ 文字列を要素とするベクトルを生成する

COLUMN ■ 文字列そのものに引用符を含める場合は
「\」(バックスラッシュ)でエスケープする

COLUMN ■ アルファベットや月の名称の定数

COLUMN ■ 関数のまとめ

035 文字列を規則的に合成したベクトルを作成する 137

ONEPOINT ■ 文字列を規則的に合成したベクトルを作成するには
「paste」関数を使用する

COLUMN ■ 結合する文字ベクトルの要素の数が異なる場合

COLUMN ■ 「stringr」パッケージによる文字結合

036 文字列オブジェクトの文字数を数える 139

ONEPOINT ■ 文字列オブジェクトの文字数を数えるには「nchar」関数を使用する

COLUMN ■ 文字列に欠損値「NA」が含まれる場合の処理

COLUMN ■ 関数のまとめ

037 指定位置の文字列を抽出する 141

ONEPOINT ■ 指定の位置の文字列を抽出するには「substr」関数を使用する

COLUMN ■ 文字数カウントの基準

COLUMN ■ 「substr」関数と同様の動きをする「substring」関数

COLUMN ■ 関数のまとめ

038 特定の文字を区切りとして文字列を分割する 144

ONEPOINT ■ 特定の文字を区切りとして文字列を分割するには
「strsplit」関数を使用する

COLUMN ■ 「split」引数で正規表現を使用する

COLUMN ■ 分割結果をリストからベクトル形式に変換する

COLUMN ■ 関数のまとめ

039 文字列を指定の長さに切り詰める 147

ONEPOINT ■ 文字列を指定の長さに切り詰めるには「strtrim」関数を使用する

COLUMN ■ 関数のまとめ

□ 4 0	文字列を指定したパターンで検索する	148
	ONEPOINT ■ 文字列を指定したパターンで検索するには「grep」関数を使用する	
	COLUMN ■ 検索文字列の一致した位置と長さを求める	
	COLUMN ■ 正規表現について	
	COLUMN ■ 関数のまとめ	
□ 4 1	文字列を指定したパターンで置換する	153
	ONEPOINT ■ 文字列を指定したパターンで置換するには「sub」関数や「gsub」関数を使用する	
	COLUMN ■ Windows 版 R における置換の副作用	
	COLUMN ■ 文字列をベクトル単位で指定して置換する	
	COLUMN ■ 関数のまとめ	
□ 4 2	文字列の文字コードを確認する／ 指定の文字コード体系に変更する	156
	ONEPOINT ■ 文字列の文字コードを確認するには「charToRaw」関数を使用し、 指定の文字コード体系に変更するには「iconv」関数や 「enc2utf8」関数を使用する	
	COLUMN ■ R で Unicode を確認する	
	COLUMN ■ R で使用している文字コード体系を確認する	
	COLUMN ■ 指定の文字コードでファイルを保存する	
	COLUMN ■ 関数のまとめ	
□ 4 3	因子ベクトルの基礎知識	160
□ 4 4	因子を作成する	161
	ONEPOINT ■ 因子を作成するには「factor」関数を使用する	
	COLUMN ■ 任意の水準数の因子を作成する	
	COLUMN ■ 因子のラベルを変更する	
	COLUMN ■ 関数のまとめ	
□ 4 5	因子の水準に並び順を定義する	163
	ONEPOINT ■ 因子の水準に並び順を定義するには「relevel」関数を使用する	
	COLUMN ■ 水準の並び順を変更する	
	COLUMN ■ 「reorder」関数の利用	
	COLUMN ■ 関数のまとめ	
□ 4 6	使われていない因子の水準を削除する	167
	ONEPOINT ■ 使われていない因子の水準を削除するには 「droplevels」関数を使用する	
	COLUMN ■ 関数のまとめ	
□ 4 7	因子の水準に大小関係を設定する	168
	ONEPOINT ■ 因子の水準に大小関係を設定するには「ordered」関数を使用する	
	COLUMN ■ コントラストの変更	
	COLUMN ■ 関数のまとめ	
□ 4 8	因子の水準を自由に組み合わせる	171
	ONEPOINT ■ 因子を自由に組み合わせるには「interaction」関数を使用する	
	COLUMN ■ 「interaction」関数の引数「lex.order = TRUE」の効果	
	COLUMN ■ 関数のまとめ	

□ 49	因子の水準ごとに関数を適用する	173
	ONEPOINT ■ 因子の水準ごとに関数を適用するには 「aggregate」「apply」「ave」「by」関数を使用する	
	COLUMN ■ 「aggregate」関数は返回值としてデータフレームを返す	
	COLUMN ■ 「apply」関数は結果を配列として返す	
	COLUMN ■ 「ave」関数は結果を数値ベクトルとして返す	
	COLUMN ■ 「by」関数は「apply」関数のラッパーである	
	COLUMN ■ 関数のまとめ	
□ 50	論理値の基礎知識	178
□ 51	論理ベクトルを作成する	179
	ONEPOINT ■ 論理ベクトルを初期化するには「logical」関数を使用する	
	COLUMN ■ 関数のまとめ	
□ 52	論理ベクトルを計算する	181
	ONEPOINT ■ 論理ベクトルを計算するには「sum」関数などの算術用関数を使用する	
	COLUMN ■ 論理演算の応用	
	COLUMN ■ 関数のまとめ	
□ 53	空のベクトルを初期化する	183
	ONEPOINT ■ 空のベクトルを初期化するには 「vector」関数にデータ型と要素数を指定する	
	COLUMN ■ ベクトル初期化の際に適切な要素数を指定する	
	COLUMN ■ 関数のまとめ	
□ 54	ベクトルの要素数を取得／変更する	185
	ONEPOINT ■ ベクトルの要素数を取得／変更するには「length」関数を使用する	
□ 55	ベクトルの要素に名前を付ける	186
	ONEPOINT ■ ベクトルの要素に名前を付けるには「names」関数を使用する	
	COLUMN ■ 関数のまとめ	
□ 56	ベクトルから要素を抽出する	187
	ONEPOINT ■ ベクトルの要素を抽出するには 「[]」演算子に添字あるいは論理式を指定する	
	COLUMN ■ 「[]」演算子を使った置き換え	
	COLUMN ■ %in%演算子の利用	
□ 57	ベクトルから条件に適合する添字番号を取得する	190
	ONEPOINT ■ 条件に一致する要素の添字番号を取得するには 「which」関数を使用する	
	COLUMN ■ 「any」関数は指定された要素の有無を調べる	
□ 58	ベクトルの要素を並べ替える	191
	ONEPOINT ■ ベクトルの要素を並べ替えるには「sort」関数を使用する	
	COLUMN ■ 「sort.int」関数を使った部分ソート	
	COLUMN ■ 「order」関数は添字を出力する	
	COLUMN ■ 関数のまとめ	
□ 59	ベクトルの要素を置き換える	194
	ONEPOINT ■ ベクトルの要素を置き換えるには「replace」関数か「[]」演算子を使用する	
	COLUMN ■ 欠損値を除くには「is.na」関数と併用する	
	COLUMN ■ 関数のまとめ	

060	ベクトルに要素を追加する	196
	ONEPOINT ■ ベクトル要素を追加するには「c」関数と「append」関数を使用する	
	COLUMN ■ 関数のまとめ	
061	ベクトルの要素の重複を調べる	197
	ONEPOINT ■ ベクトルの要素の重複を調べるには「duplicated」関数を使用する	
	COLUMN ■ 「rle」関数は同じ数値が連続する回数をリストとして返す	
	COLUMN ■ 関数のまとめ	
062	ベクトルの要素の重複を削除する	199
	ONEPOINT ■ ベクトルの要素から重複を削除するには「unique」関数を使用する	
	COLUMN ■ 関数のまとめ	

CHAPTER 05 行列

063	行列オブジェクトの基礎知識	202
064	行列の演算を行う	205
	ONEPOINT ■ 行列の演算を行うには%%演算子や「crossprod」「tcrossprod」「outer」関数などを使用する	
	COLUMN ■ 関数のまとめ	
065	行列の次元ごとに演算を適用する	209
	ONEPOINT ■ 行列の次元ごとに演算を適用するには「sweep」関数や「apply」関数を使用する	
	COLUMN ■ 「sweep」関数	
	COLUMN ■ 「apply」関数	
	COLUMN ■ 「lapply」関数	
	COLUMN ■ 「sapply」関数	
	COLUMN ■ 関数のまとめ	
066	データフレームを行列に変換する	212
	ONEPOINT ■ データフレームを行列に変換するには「as.matrix」関数を使用する	
	COLUMN ■ 演算関数によっては引数として行列を指定する場合がある	
	COLUMN ■ 関数のまとめ	
067	行列に列名と行名を設定する	214
	ONEPOINT ■ 行列に列名と行名を設定するには「colnames」関数と「rownames」関数を使用する	
	COLUMN ■ 「dimnames」関数にはリストを指定する	
	COLUMN ■ 関数のまとめ	
068	行列の属性を確認する	216
	ONEPOINT ■ 行列の属性を確認するには「attributes」関数を使用する	
	COLUMN ■ 次元属性とは	
	COLUMN ■ 行列属性を変えるには「attr」関数を使う	
	COLUMN ■ 関数のまとめ	

069	行列から成分を抽出／置換する	218
	ONEPOINT ■ 行列から成分を抽出／置換するには「」演算子を使用する	
	COLUMN ■ 「subset」関数による列を指定した成分の抽出	
	COLUMN ■ 関数のまとめ	
070	行列を結合する	220
	ONEPOINT ■ 行列に行や列を追加するには「rbind」関数や「cbind」関数を使用する	
	COLUMN ■ 列名や行名の設定	
	COLUMN ■ 関数のまとめ	
071	行列をベクトルに変換する	223
	ONEPOINT ■ 行列をベクトルに変換するには「as.vector」関数を使用する	
	COLUMN ■ 関数のまとめ	
072	対角行列を作成する	224
	ONEPOINT ■ 対角行列を作成するには「diag」関数を使用する	
	COLUMN ■ 単位行列以外の行列を作成する	
	COLUMN ■ 関数のまとめ	
073	三角行列を作成する	225
	ONEPOINT ■ 三角行列を作成するには「lower.tri」関数と「upper.tri」関数を使用する	
	COLUMN ■ 対角成分を含める	
	COLUMN ■ 関数のまとめ	
074	転置行列を作成する	227
	ONEPOINT ■ 転置行列を作成するには「t」関数を使用する	
	COLUMN ■ 関数のまとめ	
075	列や行の周辺和を求める	228
	ONEPOINT ■ 列や行ごとの和や周辺和を求めるには「rowSums」「colSums」「addmargins」などの関数を使用する	
	COLUMN ■ 「addmargins」関数の「FUN」引数に関数を指定する	
	COLUMN ■ 関数のまとめ	
076	固有値分解・特異値分解を適用する	230
	ONEPOINT ■ 固有値分解・特異値分解を適用するには「eigen」関数と「svd」関数を使用する	
	COLUMN ■ 関数のまとめ	
077	疎な行列を効率的なデータ構造に変換する	232
	ONEPOINT ■ 疎な行列を効率的なデータ構造に変換するには基本パッケージ「Matrix」を使用する	
	COLUMN ■ 関数のまとめ	
078	BLASの変更	235

CHAPTER 06 データフレーム

- 079 データフレームの基礎知識 240
- 080 データフレームを作成する 241
- ONEPOINT ■ データフレームを作成するには「data.frame」関数を使用する
- COLUMN ■ データフレーム作成時に列の名前を付ける
- COLUMN ■ 「data.frame」関数に指定された複数のベクトルの長さ(要素数)が異なる場合
- COLUMN ■ 関数のまとめ
- 081 データフレームに列名と行名を追加／変更する 244
- ONEPOINT ■ データフレームに列名と行名を追加／変更するには「rownames」関数や「colnames」関数を使用する
- COLUMN ■ 関数のまとめ
- 082 データフレームの列にアクセスする 246
- ONEPOINT ■ データフレームの列にアクセスするには\$演算子か[]演算子を使用する
- COLUMN ■ 「attach」関数は指定されたデータフレームと同名のコピーを作成する
- COLUMN ■ 「detach」関数は検索パスからデータフレームを取り除く
- COLUMN ■ 「with」関数の利用
- COLUMN ■ Rの「with」関数による処理をパイプ演算子「%>%」に置き換える
- COLUMN ■ 列名の略記
- COLUMN ■ 関数のまとめ
- 083 データフレームに要素を追加する 251
- ONEPOINT ■ データフレームに列や行を追加するには「rbind」関数や「cbind」関数を使用する
- COLUMN ■ 結合するデータフレームに因子がある場合は水準を調整する
- COLUMN ■ 関数のまとめ
- 084 データフレームを結合する 255
- ONEPOINT ■ データフレームを結合するには「merge」関数を使用する
- COLUMN ■ 「all」引数による統合の調整
- COLUMN ■ 2つのデータフレームの列を比較して統合する
- COLUMN ■ 関数のまとめ
- 085 データフレームから一部を抽出する 258
- ONEPOINT ■ データフレームから条件に適合した行と列を抽出するには「subset」関数を使用する
- COLUMN ■ 関数のまとめ
- 086 データフレームの列を変換する 260
- ONEPOINT ■ データフレームの列を変換するには「transform」関数を使用する
- COLUMN ■ 関数のまとめ
- 087 因子ごとに組み合わせを作成する 261
- ONEPOINT ■ 因子ごとに組み合わせを作成するには「expand.grid」関数を使用する
- COLUMN ■ 「KEEP.OUT.ATTRS」引数で次元情報などを調整する
- COLUMN ■ 関数のまとめ

088	データフレームの列や行ごとの合計を求める	263
	ONEPOINT ■ データフレームの列や行ごとの合計を求めるには「rowSums」関数や「colSums」関数を使用する	
	COLUMN ■ 因子の水準別に集計する場合には「rowsum」関数を使用する	
	COLUMN ■ 関数のまとめ	
089	データフレームの列ごとに関数を適用する	265
	ONEPOINT ■ 列または行ごとに関数を適用するには「apply」族の関数を使用する	
	COLUMN ■ 計算結果をリスト形式で返す「apply」関数	
	COLUMN ■ 計算結果をベクトルで返す「sapply」関数	
	COLUMN ■ 計算値の整合性をチェックする「vapply」関数	
	COLUMN ■ 「mapply」関数の利用	
	COLUMN ■ 関数のまとめ	
090	データフレームから欠損値を取り除く	269
	ONEPOINT ■ データフレームから欠損値を含む列を削除するには「complete.cases」関数を使用する	
	COLUMN ■ 関数のまとめ	
091	データフレームの列を水準ごとに分解／統合する	271
	ONEPOINT ■ データフレームの列を複数列に分けるには「unstack」関数を使用する	
	COLUMN ■ 複数列を単独の列にまとめる「stack」関数	
	COLUMN ■ 関数のまとめ	
092	縦長形式のデータフレームを横長形式に変換する	273
	ONEPOINT ■ 縦長形式のデータフレームを横長形式に変換するには「reshape」関数を使用する	
	COLUMN ■ 縦長形式のデータを横長形式に変換する	
	COLUMN ■ 横長形式のデータを縦長形式に変換する	
	COLUMN ■ データを整形できる他のパッケージ	
	COLUMN ■ 関数のまとめ	
093	データフレームを分割する	278
	ONEPOINT ■ データフレームを分割するには「split」関数を使用する	
	COLUMN ■ データフレームを要素とするリストを単一のデータフレームにまとめる	
	COLUMN ■ 関数のまとめ	
094	データフレームを並べ替える	280
	ONEPOINT ■ データフレームを並べ替えるには「order」関数を使用する	
	COLUMN ■ 「arrange」関数による並べ替え	
095	効率的なデータ操作を可能にする「dplyr」パッケージ	282
	COLUMN ■ 「dplyr」パッケージのチートシート	
	COLUMN ■ 関数のまとめ	
096	高速な「data.table」パッケージを使う	291
	COLUMN ■ 関数のまとめ	

CHAPTER 07 リスト／配列／表

097	リスト処理の基礎知識	296
098	リストを作成する	298
	ONEPOINT ■ リストを作成するには「list」関数を使用する	
	COLUMN ■ リストの要素へのアクセス方法	
	COLUMN ■ リストの操作性を高めるパッケージ	
	COLUMN ■ 関数のまとめ	
099	リストをベクトルに変換する	303
	ONEPOINT ■ リストをベクトルに変換するには「unlist」関数を使用する	
	COLUMN ■ 因子を含むリストの統合	
	COLUMN ■ リストの要素からそれらの要素をベクトルとしてリスト化	
	COLUMN ■ 関数のまとめ	
100	リストの要素ごとに演算を行う(「apply」族の応用)	306
	ONEPOINT ■ リストの要素ごとに演算を行うには「apply」系の関数を使用する	
	COLUMN ■ 関数のまとめ	
101	配列(Array)オブジェクトの基礎知識	309
102	配列オブジェクトを作成する	311
	ONEPOINT ■ 配列オブジェクトを作成するには「array」関数を使用する	
	COLUMN ■ 配列へのアクセス	
	COLUMN ■ 配列の次元に名前を付けるには「dimnames」関数を使用する	
	COLUMN ■ 関数のまとめ	
103	配列の次元ごとに計算する(「apply」族の応用)	315
	ONEPOINT ■ 配列の次元ごとに計算するには「apply」族の関数を使用する	
	COLUMN ■ 「MARGIN」引数の指定	
	COLUMN ■ 配列からベクトルの添字でデータを抽出する	
	COLUMN ■ 関数のまとめ	
104	配列に周辺度数を追加する	318
	ONEPOINT ■ 配列に周辺度数を追加するには「addmargins」関数を使用する	
	COLUMN ■ 関数のまとめ	
105	多次元配列を2次元に変換する	321
	ONEPOINT ■ 多次元配列を2次元に変換するには「ftable」関数を使用する	
	COLUMN ■ 「ftable」関数の「row.vars」引数と「col.vars」引数	
	COLUMN ■ 関数のまとめ	
106	配列を転置する	326
	ONEPOINT ■ 配列を転置するには「aperm」関数を使用する	
	COLUMN ■ 関数のまとめ	
107	頻度表を作成する	328
	ONEPOINT ■ 頻度表を作成するには「table」関数を使用する	
	COLUMN ■ データが連続量の場合は「cut」関数と併用する	
	COLUMN ■ 関数のまとめ	

108	分割表を作成する	330
	ONEPOINT ■ 分割表を作成するには「xtable」関数や「table」関数を使用する	
	COLUMN ■ 「xtable」関数のモデル式	
	COLUMN ■ 頻度表に周辺合計を加える	
	COLUMN ■ 関数のまとめ	
109	表をLaTeXやHTMLの形式で出力する	334
	ONEPOINT ■ 表をLaTeXやHTMLの形式で出力するには「xtable」パッケージの「xtable」関数を使用する	
	COLUMN ■ 「xtable」関数はオブジェクトに保存する	
	COLUMN ■ 「Hmisc」パッケージの「latex」関数を利用する	
	COLUMN ■ 関数のまとめ	

CHAPTER 08 関数の作成

110	関数の作成の基礎知識	338
111	関数を定義する	340
	ONEPOINT ■ 関数を定義するには「function」関数を使用する	
	COLUMN ■ 関数に引数を指定する	
	COLUMN ■ 無名関数とは	
	COLUMN ■ ラムダ記法	
	COLUMN ■ 「magrittr」パッケージの演算子	
	COLUMN ■ 関数のまとめ	
112	引数のデフォルト値を設定する	345
	ONEPOINT ■ 引数のデフォルト値を設定するには「function」関数で定義する	
	COLUMN ■ 仮引数とデフォルト値は「formals」関数で確認できる	
	COLUMN ■ 引数の指定順序	
	COLUMN ■ 仮引数名の省略	
	COLUMN ■ ドット引数の利用	
113	引数をチェックする	348
	ONEPOINT ■ 引数の値をチェックするには「missing」関数や「stopifnot」関数を使用する	
	COLUMN ■ 「match.arg」関数を使って実引数を指定する／制限する	
	COLUMN ■ 「pmatch」関数で部分マッチを行う	
	COLUMN ■ 関数のまとめ	
114	関数から複数の値を返す	351
	ONEPOINT ■ 関数から複数の値を返すには明示的にベクトルまたはリストを返り値にする	
	COLUMN ■ 返り値が大きい場合に表示を抑制するには「invisible」関数を使用する	
	COLUMN ■ 関数のまとめ	
115	エラーを処理する	353
	ONEPOINT ■ エラーを処理するには「try」関数を使用する	
	COLUMN ■ 「tryCatch」関数でエラー停止後の処理を制御する	
	COLUMN ■ 関数のまとめ	

1 1 6	引数を取り出して利用する	355
	ONEPOINT ■ 引数をそのまま取り出すには遅延評価を利用する	
	COLUMN ■ 「substitute」関数を使う 関数の実引数をそのまま取り出すことができる	
	COLUMN ■ 言語オブジェクトの利用	
	COLUMN ■ 関数のまとめ	
1 1 7	関数をベクトル化する	358
	ONEPOINT ■ 関数をベクトル化するには「Vectorize」関数を使用する	
	COLUMN ■ デフォルト引数はベクトル化されない	
	COLUMN ■ 関数のまとめ	
1 1 8	警告を抑制する	360
	ONEPOINT ■ 警告を抑制するには「suppressWarnings」関数を使用する	
	COLUMN ■ セッション全体で警告やエラーの表示を設定する	
	COLUMN ■ 関数のまとめ	
1 1 9	条件分岐を設定する	362
	ONEPOINT ■ 条件分岐は「if」関数や「ifelse」関数を使う	
	COLUMN ■ 「if」関数と「else」で複数の条件を分岐する	
	COLUMN ■ 「ifelse」関数では条件部分がベクトル処理される	
	COLUMN ■ 「switch」関数による条件分岐	
	COLUMN ■ 関数のまとめ	
1 2 0	ループ(繰り返し)を設定する	366
	ONEPOINT ■ ループ(繰り返し)を設定するには「for」関数や「while」関数を使用する	
	COLUMN ■ 「for」関数の利用	
	COLUMN ■ 「while」関数の利用	
	COLUMN ■ 「replicate」関数による繰り返し	
	COLUMN ■ 関数のまとめ	
1 2 1	ループを中断する	370
	ONEPOINT ■ ループを抜けるには「next」や「break」を使う	
1 2 2	「do.call」関数を使ってコードを実行する	371
	COLUMN ■ 関数のまとめ	
1 2 3	高階関数	374
	COLUMN ■ 「purrr」パッケージ	

CHAPTER 09 ファイル／データベース

1 2 4	Rによるファイル処理の基礎知識	378
1 2 5	テキストファイルのデータを読み込む	380
	ONEPOINT ■ テキストファイルのデータを読み込むには「read.table」関数を使用する	
	COLUMN ■ 「read.csv」関数や「read.delim」関数でテキストファイルを読み込む	
	COLUMN ■ 文字は因子に変換される	
	COLUMN ■ 「readLines」関数で未定型のテキストファイルを読み込む	
	COLUMN ■ 関数のまとめ	

1 2 6	テキストファイルへデータを書き込む	385
	ONEPOINT ■ テキストファイルへデータを書き込むには 「write.table」関数を使用する	
	COLUMN ■ 関数のまとめ	
1 2 7	Excel形式でデータを読み込み／書き込みする	387
	COLUMN ■ 「RODBC」パッケージの利用方法	
	COLUMN ■ 日本語文字コードの扱い	
	COLUMN ■ 関数のまとめ	
1 2 8	データベースを操作する	391
	ONEPOINT ■ データベースを操作するには「RSQLite」パッケージなどを使用する	
	COLUMN ■ 「RODBC」パッケージによるアクセス	
	COLUMN ■ 「dplyr」パッケージによるデータベース操作	
	COLUMN ■ 関数のまとめ	
1 2 9	文字コードを指定してファイルを読み込む	395
	ONEPOINT ■ 文字コードを指定してファイルを読み込むには 「fileEncoding」引数を指定する	
	COLUMN ■ 表形式のファイルを読み込む関数で「fileEncoding」引数を指定する	
	COLUMN ■ 関数のまとめ	
1 3 0	インターネット上のファイルを読み込む	396
	ONEPOINT ■ インターネット上のファイルを読み込むにはファイル読み込み関数の 「file」引数でアドレスを指定する	
	COLUMN ■ ファイルをダウンロードする	
	COLUMN ■ 青空文庫	
	COLUMN ■ 関数のまとめ	
1 3 1	オブジェクトを保存する／読み込む	399
	ONEPOINT ■ オブジェクトを保存するには「save」関数を使用する	
	COLUMN ■ 保存したオブジェクトの「load」関数による復元	
	COLUMN ■ 「save.image」関数によるオブジェクトの保存	
	COLUMN ■ 「dput」関数と「dget」関数の利用	
	COLUMN ■ 関数のまとめ	
1 3 2	ファイルやフォルダを操作する	402
	ONEPOINT ■ ファイルやフォルダを操作するには「file」で始まる関数を使用する	
	COLUMN ■ ファイルやフォルダを操作する他の関数	
	COLUMN ■ 関数のまとめ	
1 3 3	一時的なフォルダやファイルを作成する	404
	ONEPOINT ■ 一時的なフォルダやファイルを作成するには 「tempfile」関数と「tempdir」関数を使用する	
	COLUMN ■ 一時フォルダ内のファイルの読み込み	
	COLUMN ■ 一時フォルダの削除	
	COLUMN ■ 「textConnection」関数の利用	
	COLUMN ■ 関数のまとめ	

1 3 4	キーボードやクリップボードからデータを読み込む	407
	ONEPOINT ■ キーボードやクリップボードからデータを読み込むには 「scan」「stdin」「fix」関数を使用する	
	COLUMN ■ コンソール入力やファイルから 行列やデータフレームを作る「scan」関数	
	COLUMN ■ スクリプト内での「scan」関数の実行	
	COLUMN ■ クリップボードから入力する方法	
	COLUMN ■ GUIからデータを入力する	
	COLUMN ■ 関数のまとめ	
1 3 5	Rの作業内容をネットワーク越しにやり取りする	412
	ONEPOINT ■ セッションをネットワーク越しにやり取りするには「url」関数を使用する	
	COLUMN ■ 関数のまとめ	
1 3 6	Rのスクリプトコードを読み込む	413
	ONEPOINT ■ Rのスクリプトコードを読み込むには「source」関数を使用する	
	COLUMN ■ 「source」関数はネット上のファイルも指定できる	
	COLUMN ■ 関数のまとめ	
1 3 7	SPSSなどで作成したファイルを読み込む	414
	ONEPOINT ■ SPSSなどで作成したファイルを読み込むには 「foreign」パッケージを使用する	
	COLUMN ■ 書き込みには「write.foreign」関数にソフトウェアを指定する	
	COLUMN ■ 関数のまとめ	

CHAPTER 10 基本統計解析

1 3 8	基本統計関数について	418
	COLUMN ■ 複素数オブジェクトは「+」と「i」で構成される	
1 3 9	乱数の利用	425
1 4 0	要約統計量を求める	426
	ONEPOINT ■ 要約統計量を求めるには「summary」関数を使用する	
	COLUMN ■ 「summary」関数を使う上での注意点	
	COLUMN ■ データフレームのカテゴリ変数の水準ごとに要約関数を適用する	
	COLUMN ■ 5数要約量を返す「fivenum」関数	
	COLUMN ■ 「dplyr」パッケージで列ごとに要約統計量を求めるには 「summarize_each」を適用する	
	COLUMN ■ 要約をもとのデータフレームに追加する	
	COLUMN ■ 関数のまとめ	
1 4 1	数値を丸める	432
	ONEPOINT ■ 数値を丸めるには「round」関数を使用する	
	COLUMN ■ 小数点以下の切り上げには「ceiling」関数を使用し、 切り下げには「floor」「trunc」関数を使用する	
	COLUMN ■ 「zapsmall」関数は数値が「小さい」場合0に丸める	
	COLUMN ■ 関数のまとめ	

1 4 2	データから欠損値を取り除く	435
	ONEPOINT ■ データから欠損値を除くには「na.rm」引数を使用する	
	COLUMN ■ データフレームなどが欠損値を含む場合の処理をセッションオプションとして指定する	
	COLUMN ■ 関数のまとめ	
1 4 3	正規性の検定を実行する	439
	ONEPOINT ■ 正規性の検定には「shapiro.test」「ks.test」関数を使用する	
	COLUMN ■ 「shapiro.test」関数や「ks.test」関数による正規性判定	
	COLUMN ■ 「qqplot」関数による正規性の判断	
	COLUMN ■ 関数のまとめ	
1 4 4	等分散性の検定を実行する	442
	ONEPOINT ■ 等分散性の検定を実行するには2群の比較には「var.test」関数、多群の場合には「bartlett.test」を使用する	
	COLUMN ■ 統計的検定ではモデル式を利用する	
	COLUMN ■ データの先頭行を表示する「head」関数	
	COLUMN ■ 関数のまとめ	
1 4 5	t検定で平均値を比較する	446
	ONEPOINT ■ 正規母集団から抽出された2標本の平均を比較するには「t.test」関数を使う	
	COLUMN ■ 引数「paired」で対応の有無を指定する	
	COLUMN ■ 正規性の仮定	
	COLUMN ■ 引数「var.test」で等分散性の仮定を指定する	
	COLUMN ■ データの変数指定の方法	
	COLUMN ■ 両側検定と片側検定	
	COLUMN ■ 検定の結果の書き方	
	COLUMN ■ 平均値の比較を行う場合の注意事項	
1 4 6	ノンパラメトリックな検定で平均を比較する	451
	ONEPOINT ■ 分布を仮定できないデータの平均(中心位置)を比較するには「wilcox.test」関数を使用する	
	COLUMN ■ 「ウィルコクソンの順位和検定」と「マン・ホイットニーの順位和検定」は本質的に同じもの	
	COLUMN ■ 「クラスカル・ウォリスの順位和検定」の利用	
	COLUMN ■ 関数のまとめ	
1 4 7	比率の検定を実行する	453
	ONEPOINT ■ 比率の差を検定するには「prop.test」関数を使用する	
	COLUMN ■ デフォルトではイエーツの補正が適用される	
	COLUMN ■ 関数のまとめ	
1 4 8	独立性の検定(カイ二乗検定／Fisherの検定)を実行する	456
	ONEPOINT ■ 独立性の検定を実行するには「chisq.test」関数や「fisher.test」関数を使用する	
	COLUMN ■ 2行2列の分割表でイエーツの補正を無効にする	
	COLUMN ■ 対応のある分割表の処理	
	COLUMN ■ Fisherの正確確率は「fisher.test」関数で実行される	
	COLUMN ■ 関数のまとめ	

1 4 9	相関係数を求める／検定する	459
	ONEPOINT ■ 相関係数を求めるには「cor」関数、 検定するには「cor.test」関数を使用する	
	COLUMN ■ 関数のまとめ	
1 5 0	分散分析を実行する	461
	ONEPOINT ■ 分散分析を実行するには「aov」関数にモデル式を使用する	
	COLUMN ■ 「replications」関数で繰り返し数を確認する	
	COLUMN ■ 等分散性の検定	
	COLUMN ■ 分散分析でのモデル記法	
	COLUMN ■ 1元配置の分散分析	
	COLUMN ■ 2元配置の分散分析あるいは対応のある1元配置分散分析	
	COLUMN ■ 誤差項およびネストの指定	
	COLUMN ■ コントラストの設定	
	COLUMN ■ 関数のまとめ	
1 5 1	多重比較を実行する	472
	ONEPOINT ■ 多重比較を実行するには「pairwise.t.test」関数を使用する	
	COLUMN ■ 多重比較の主な手法	
	COLUMN ■ 関数のまとめ	
1 5 2	回帰分析を実行する	475
	ONEPOINT ■ 回帰分析を実行するには「lm」関数を使用する	
	COLUMN ■ 回帰のあてはめを確認する	
	COLUMN ■ 回帰分析の結果は「summary」関数や「plot」関数で確認する	
	COLUMN ■ 回帰を連続して実行する	
	COLUMN ■ 「update」「add1」「drop1」「step」関数によるモデルの修正	
	COLUMN ■ 関数のまとめ	
1 5 3	時系列データの統計量について	485
	COLUMN ■ 関数のまとめ	
1 5 4	検出力を求める	488
	ONEPOINT ■ t検定での検出力や必要な標本数は「power.t.test」関数を使用する	
	COLUMN ■ 関数のまとめ	
1 5 5	モデル式の基礎知識	490

CHAPTER 11 応用統計解析

1 5 6	Rでの応用統計解析	494
1 5 7	一般化線形モデルを実行する	495
	ONEPOINT ■ 一般化線形モデルを実行するには「glm」関数を使用する	
	COLUMN ■ 一般化線形モデルのモデル式	
	COLUMN ■ 関数のまとめ	

158	一般化線形モデルを更新／比較する	498
	ONEPOINT ■ 一般化線形モデルの更新を行うには「update」関数を使用する	
	COLUMN ■ 最適なモデルの選択	
	COLUMN ■ 関数のまとめ	
159	二項分布を仮定してデータに一般化線形モデルを適用する	503
	ONEPOINT ■ 二項分布モデルを指定するにはモデル式を使用する	
160	主成分分析を実行する	506
	ONEPOINT ■ 主成分分析を実行するには「princomp」関数を使用する	
	COLUMN ■ 「princomp」関数による主成分分析	
	COLUMN ■ 「prcomp」関数による主成分分析	
	COLUMN ■ 関数のまとめ	
161	因子分析を実行する	509
	ONEPOINT ■ 因子分析を実行するには「factanal」関数を使用する	
	COLUMN ■ 因子分析とは	
	COLUMN ■ 「GPArotation」パッケージによる因子分析	
	COLUMN ■ 関数のまとめ	
162	対応分析を実行する	515
	ONEPOINT ■ 対応分析を実行するには 「MASS」パッケージの「corresp」関数を使用する	
	COLUMN ■ 分割表の場合の「corresp」関数の引数「nf」の指定	
	COLUMN ■ 多重分割表を作成する場合は「mca」関数を使用する	
	COLUMN ■ 関数のまとめ	
163	クラスター分析を実行する	518
	ONEPOINT ■ 階層的クラスター分析を実行するには 「dist」関数で距離を求めて「hclust」関数を使用する	
	COLUMN ■ データから距離を求める	
	COLUMN ■ 「kmeans」関数による非階層的クラスター分析	
	COLUMN ■ 関数のまとめ	
164	線形判別分析を実行する	522
	ONEPOINT ■ 判別分析を実行するには「lda」関数を使用する	
	COLUMN ■ 「lda」関数の利用	
	COLUMN ■ 判別の結果を確認する	
	COLUMN ■ 新規データの判別	
	COLUMN ■ 交差妥当化	
	COLUMN ■ 関数のまとめ	
165	ニューラルネットワークを実行する	526
	ONEPOINT ■ ニューラルネットワークを実行するには「nnet」パッケージを使用する	
	COLUMN ■ 「nnet」関数の利用	
	COLUMN ■ 関数のまとめ	
166	サポートベクターマシンを実行する	528
	ONEPOINT ■ サポートベクターマシンを実行するには 「kernlab」パッケージを使用する	
	COLUMN ■ 「ksvm」関数の利用	
	COLUMN ■ 関数のまとめ	

1 6 7	決定木を利用する	530
	ONEPOINT ■ 決定木を利用するには「rpart」パッケージを使用する	
	COLUMN ■ 「rpart」関数の利用	
	COLUMN ■ 「rpart」出力のプロット	
	COLUMN ■ 「rpart」関数の枝を刈る	
	COLUMN ■ 関数のまとめ	
1 6 8	自己組織化マップを作成する	534
	ONEPOINT ■ 自己組織化マップを作成するには「kohonen」パッケージを使用する	
	COLUMN ■ 「som」関数の利用	
	COLUMN ■ 関数のまとめ	
1 6 9	ベイジアンネットワーク分析を利用する	536
	ONEPOINT ■ ベイジアンネットワーク分析を実行するには「deal」パッケージを使用する	
	COLUMN ■ 禁則ルールの指定	
	COLUMN ■ 関数のまとめ	
1 7 0	ベイズ統計解析を実行する	540
	ONEPOINT ■ ベイズ統計解析を実行するには「MCMCpack」パッケージを使用する	
	COLUMN ■ 「MCMultinomDirichlet」関数の利用	
	COLUMN ■ 関数のまとめ	
1 7 1	ベイズ法で線形回帰分析を実行する	543
	ONEPOINT ■ ベイズ法で線形回帰分析を実行するには「MCMCregress」関数を使用する	
	COLUMN ■ 関数のまとめ	
1 7 2	MCMCpackパッケージでユーザー定義の関数を シミュレーションする	546
	ONEPOINT ■ ユーザー定義の関数を使ってサンプリングするには「MCMCmetrop1R」関数を利用する	
	COLUMN ■ 関数のまとめ	
1 7 3	「RStan」パッケージ	550
	COLUMN ■ 関数のまとめ	

CHAPTER 12 基本グラフィックス

1 7 4	グラフィックスの基礎知識	558
1 7 5	グラフ領域を設定する	560
	ONEPOINT ■ グラフ領域を設定するには「par」関数を使用する	
	COLUMN ■ 余白を設定するには「par」関数を使用する	
	COLUMN ■ 「box」関数で領域を箱で囲む	
	COLUMN ■ 文字列を追加する「text」関数と「mtext」関数	
	COLUMN ■ 「par」関数でのみ変更可能なパラメータ	
	COLUMN ■ Windows版Rで表示するグラフを確認する	
	COLUMN ■ 関数のまとめ	

176	点と線を描画する	565
	ONEPOINT ■ 点と線を描画するには「plot」関数を使用する	
	COLUMN ■ 関数のまとめ	
177	ダミーの(空白の)散布図を描く	567
	ONEPOINT ■ ダミーの(空白の)散布図を描くには 「plot」関数で引数「type」に「n」を指定する	
	COLUMN ■ 矩形領域を塗りつぶすには「rect」関数を使用する	
	COLUMN ■ 「axis」関数で軸の描画を行う	
	COLUMN ■ メインタイトルとサブタイトルを描画する「title」関数	
	COLUMN ■ 関数のまとめ	
178	データ点を描く	570
	ONEPOINT ■ データをプロットするには「points」関数や「text」関数を使用する	
	COLUMN ■ 「points」関数でデータを描画する	
	COLUMN ■ 「text」関数でプロット領域に文字列を表示する	
	COLUMN ■ 関数のまとめ	
179	日本語フォントの設定	573
180	線を描く	576
	ONEPOINT ■ 線を描くには「lines」関数を使用する	
	COLUMN ■ 線種は「lty」引数に指定する	
	COLUMN ■ 回帰直線の追加	
	COLUMN ■ 「lines」関数のその他の引数	
	COLUMN ■ 関数のまとめ	
181	複雑な図形やグラフを簡単に描く	580
	ONEPOINT ■ 複雑な図形やグラフを簡単に描くには「grid」パッケージを使用する	
	COLUMN ■ 「grid」パッケージの関数による図形の描画	
	COLUMN ■ 作図結果をオブジェクトに代入する	
	COLUMN ■ 関数のまとめ	
182	グリッド・オブジェクトを一括で作成する	583
	ONEPOINT ■ グリッド・オブジェクトを一括で作成するには 「grob」オブジェクトで階層的に作成する	
	COLUMN ■ 関数のまとめ	
183	凡例を表示する	586
	ONEPOINT ■ 凡例を表示するには「legend」関数を使用する	
	COLUMN ■ 関数のまとめ	
184	グラフに数式を表示する	589
	ONEPOINT ■ グラフに数式を表示するには LaTeX風の書式を「expression」関数に指定する	
	COLUMN ■ 「expression」関数の利用	
	COLUMN ■ 「substitute」関数	
	COLUMN ■ 関数のまとめ	
185	グラフにジッターを加える	593
	ONEPOINT ■ グラフにジッターを加えるには「jitter」関数を使用する	
	COLUMN ■ 関数のまとめ	

186	グラフを分割する	595
	ONEPOINT ■ グラフの分割は行列を使って指定する	
	COLUMN ■ 「layout」関数では非対称な配列が可能	
	COLUMN ■ 関数のまとめ	
187	グラフを保存する	598
	ONEPOINT ■ グラフを保存するには 「dev.copy2bmp」「dev.copy2pdf」関数などを使用する	
	COLUMN ■ 日本語を含むグラフをPDFファイルとして保存する	
	COLUMN ■ 複数ページを保存する	
	COLUMN ■ Windows版Rのグラフィックス・ウィンドウのメニューからの保存	
	COLUMN ■ 関数のまとめ	
188	対話的にデータ点を確認する	601
	ONEPOINT ■ 対話的にデータ点を確認するには 「locator」関数や「identify」関数を使用する	
	COLUMN ■ 「locator」関数で対話的に凡例位置を指定できる	
	COLUMN ■ 関数のまとめ	
189	プロットの要素にカラーを設定する	603
	ONEPOINT ■ プロットの要素にカラーを設定するには 「plot」関数の「col」引数を使用する	
	COLUMN ■ 色彩を指定する専用関数の利用	
	COLUMN ■ 「rgb」関数で色を指定する	
	COLUMN ■ 関数のまとめ	
190	バーチャート(棒グラフ)を作成する	607
	ONEPOINT ■ バーチャート(棒グラフ)を作成するには「barplot」関数を使用する	
	COLUMN ■ バーを並べる／積み上げる	
	COLUMN ■ 3D風のバーチャートを作成する	
	COLUMN ■ 関数のまとめ	
191	パイチャート(円グラフ)を作成する	611
	ONEPOINT ■ パイチャート(円グラフ)を作成するには「pie」関数を使用する	
	COLUMN ■ 3D風のパイチャートを作成する	
	COLUMN ■ 関数のまとめ	
192	箱ひげ図を作成する	613
	ONEPOINT ■ 箱ひげ図を作成するには「boxplot」関数を使用する	
	COLUMN ■ データはモデル式で指定する	
	COLUMN ■ ノッチを追加する	
	COLUMN ■ 関数のまとめ	
193	Clevelandのドットチャートを作成する	616
	ONEPOINT ■ Clevelandのドットチャートを作成するには「dotchart」関数を使用する	
	COLUMN ■ 関数のまとめ	
194	ストリップチャートを作成する	617
	ONEPOINT ■ ストリップチャートを作成するには「stripchart」関数を使用する	
	COLUMN ■ 関数のまとめ	

CHAPTER 13 多変量グラフィックス

195	分割表をプロットする	620
	ONEPOINT ■ 分割表をプロットするには 「mosaicplot」「fourfoldplot」「assocplot」などの関数を使用する	
	COLUMN ■ モザイク・プロットを作成する「mosaicplot」関数	
	COLUMN ■ 四重プロットを作成する「fourfoldplot」関数	
	COLUMN ■ 連関プロットを作成する「assocplot」関数	
	COLUMN ■ 関数のまとめ	
196	コプロット(条件付散布図)を描画する	624
	ONEPOINT ■ コプロット(条件付散布図)を描画するには「coplot」関数を使用する	
	COLUMN ■ 条件変数として連続量と離散値を指定した場合の違い	
	COLUMN ■ 関数のまとめ	
197	ヒートマップを作成する	626
	ONEPOINT ■ ヒートマップを作成するには「heatmap」関数を使用する	
	COLUMN ■ カラーの指定	
	COLUMN ■ 関数のまとめ	
198	3次元データをグラフ化する	628
	ONEPOINT ■ 3次元データをグラフ化するには 「image」「persp」「contour」関数を使用する	
	COLUMN ■ 関数のまとめ	
199	「lattice」グラフィックスとは	631
200	3次元散布図を作成する	637
	ONEPOINT ■ 3次元散布図を作成するには 「lattice」パッケージの「cloud」関数を使用する	
	COLUMN ■ 関数のまとめ	
201	3次元平面透視図を作成する	639
	ONEPOINT ■ 3次元平面透視図を作成するには 「lattice」パッケージの「wireframe」関数を使用する	
	COLUMN ■ 関数のまとめ	
202	「ggplot2」パッケージによるグラフィックス	641
	COLUMN ■ 「ggplot2」パッケージのチートシート	
	COLUMN ■ 関数のまとめ	
203	「rgl」パッケージによる3Dグラフィックス	657
204	モーションチャート	659

CHAPTER 14 Rの応用

205	直前の処理結果を表示する	662
	ONEPOINT ■ 直前の処理結果を表示するには「Last.value」オブジェクトを使用する	
206	ガベージコレクションを実行する	663
	ONEPOINT ■ ガベージコレクションを実行するには「gc」関数を使用する	
	COLUMN ■ 関数のまとめ	
207	バッチ処理を実行する	664
	COLUMN ■ バッチ実行時のオプション指定	
	COLUMN ■ 32ビットと64ビットの切り替え	
208	Rの出力を取り込んだLaTeXレポートを生成する	669
	ONEPOINT ■ Rの出力をLaTeXフォーマットに変換するには「Sweave」関数を使用する	
	COLUMN ■ チャンクのオプション指定	
	COLUMN ■ Sweaveファイルの処理はコマンドプロンプトでも実行可能	
	COLUMN ■ 関数のまとめ	
209	日付や時刻をデータとして扱う	673
	ONEPOINT ■ 日付や時間をデータとして扱うにはオブジェクト化する	
	COLUMN ■ 「as.POSIXct」関数と「as.POSIXlt」関数	
	COLUMN ■ C言語風に日付オブジェクトを生成する	
	COLUMN ■ 関数のまとめ	
210	コンソールの表示幅を調整する	677
	ONEPOINT ■ コンソールの表示幅を調整するには「options」関数の「width」引数を使用する	
	COLUMN ■ 関数のまとめ	
211	集合演算を実行する	678
	ONEPOINT ■ 集合演算を実行するには「union」「intersect」「setdiff」などの関数を使用する	
	COLUMN ■ 関数のまとめ	
212	関数の最大化／最小化について	679
	COLUMN ■ 関数を最大化／最小化するには「optimize」関数や「optim」関数を使用する	
	COLUMN ■ 関数のまとめ	
213	オブジェクトのスコープを調べる	683
	ONEPOINT ■ オブジェクトのスコープを調べるには「search」関数を使用する	
	COLUMN ■ サーチパスは「search」関数で確認する	
	COLUMN ■ 「attach」関数はオブジェクトを検索パスに追加する	
	COLUMN ■ 「get」関数は指定された検索パス位置のオブジェクトを取得する	
	COLUMN ■ 関数のまとめ	

2 1 4	環境を確認する	686
	ONEPOINT ■ Rの環境を確認する	
	COLUMN ■ 環境は「environment」関数で確認する	
	COLUMN ■ 「parent.env」関数や「parent.frame」関数はそれぞれ親環境名と実行環境名を返す	
	COLUMN ■ 「sys.frame」関数と「sys.nframe」関数は環境名と番号を表示する	
	COLUMN ■ 「assign」関数は指定された検索パス位置にオブジェクトを生成する	
	COLUMN ■ 関数のまとめ	
2 1 5	環境を指定してオブジェクトに代入する	691
	ONEPOINT ■ 環境をさかのぼってオブジェクトに代入するには<<-演算子を使用する	
2 1 6	デバッグを実行する	693
	ONEPOINT ■ デバッグを実行するには基本デバッグ関数「browser」を使用する	
	COLUMN ■ コードをステップごとに実行するには「debug」関数を使用する	
	COLUMN ■ 「trace」関数を使うと一時的に関数を変更できる	
	COLUMN ■ 関数のまとめ	
2 1 7	R本体のデバッグ(C言語ソースを追う)	698
2 1 8	コードの実行速度を確認する	703
	ONEPOINT ■ コードの実行速度を確認するには「system.time」関数を使用する	
	COLUMN ■ 「rbenchmark」パッケージの利用	
	COLUMN ■ 複数のコードの実行速度を比較する	
	COLUMN ■ 関数のまとめ	
2 1 9	パッケージを作成する	705
	ONEPOINT ■ パッケージを作成するには「package.skeleton」関数でひな形のファイルとフォルダを作成する	
	COLUMN ■ 「DESCRIPTION」ファイルの構成	
	COLUMN ■ 「Read-and-delete-me」ファイルの構成	
	COLUMN ■ 「Rd」ファイルの構成	
	COLUMN ■ 名前空間について	
	COLUMN ■ パッケージ編集後のチェックとインストール	
	COLUMN ■ 「RStudio」によるパッケージ開発	
	COLUMN ■ 「roxygen2」パッケージによるドキュメント生成	
	COLUMN ■ パッケージを公開する前に	
	COLUMN ■ パッケージ開発のチートシート	
	COLUMN ■ 関数のまとめ	
2 2 0	S3クラスを定義する	714
	ONEPOINT ■ S3クラスを定義するには「class」関数でオブジェクトにクラスを設定する	
	COLUMN ■ S3クラスの利用	
	COLUMN ■ 総称関数とメソッドの定義	
	COLUMN ■ 関数のまとめ	

2 2 1	S4クラスを定義する	717
	ONEPOINT ■ S4クラスを定義するには「setClass」関数を使用する	
	COLUMN ■ オブジェクトの初期化は「new」関数で行う	
	COLUMN ■ 「S4」の総称関数は「setGeneric」関数で設定する	
	COLUMN ■ メソッドは「setMethod」関数で定義する	
	COLUMN ■ 「setValidity」関数で検証用のメソッドを指定する	
	COLUMN ■ 関数のまとめ	
2 2 2	Reference Classを定義する	722
	ONEPOINT ■ Reference Classを定義するには「setRefClass」関数を使用する	
	COLUMN ■ 「\$initialize」メソッドの役割	
	COLUMN ■ 「\$fields」メソッドの役割	
	COLUMN ■ 「\$methods」メソッドの役割	
	COLUMN ■ 「\$new」メソッドの役割	
	COLUMN ■ 「\$accessors」メソッドの役割	
	COLUMN ■ 「\$copy」メソッド	
	COLUMN ■ 関数のまとめ	
2 2 3	RのGUIインターフェイス(Rコマンダー)を利用する	727
	ONEPOINT ■ RのGUIインターフェイス(Rコマンダー)を利用するには「library (Rcmdr)」を実行する	
	COLUMN ■ Rcmdrの利用法	
2 2 4	ExcelからのRの利用について	729
	COLUMN ■ 外部アプリケーションの実行	
2 2 5	日本語テキストを解析する	732
	ONEPOINT ■ 日本語テキストを解析するには「RMeCab」パッケージを使用する	
	COLUMN ■ 文字列やファイルから各種の頻度情報を取得してテキストを解析する	
2 2 6	欧米語のテキストデータをマイニングするツール	734
2 2 7	「tm」パッケージで日本語文書行列を操作する	739
2 2 8	EmacsでのRの利用について	742
2 2 9	C言語との連携について	747
2 3 0	「Rcpp」パッケージについて	753
	COLUMN ■ Rcppを利用したパッケージ作成	
	COLUMN ■ 関数のまとめ	
2 3 1	関数のバイトコンパイルについて	756
	COLUMN ■ 関数のまとめ	
2 3 2	並列処理について	758
2 3 3	関数の単体テストを行う	761
	COLUMN ■ 関数のまとめ	
2 3 4	WEBスクレイピング	765
	COLUMN ■ 関数のまとめ	
2 3 5	「Shiny」パッケージでWEBアプリケーションを作成する	773

APPENDIX データセットの一覧

236 データセットの一覧	778
---------------------	-----

● 索引	784
------------	-----

● 参考文献	799
--------------	-----

CHAPTER 01

R言語の基礎

Rの基礎知識

III Rとは

Rはフリーで使える統計解析環境です。ここで「ソフトウェア」とは呼ばずに、「環境」というのは、Rは機能が固定されたソフトウェアではなく、ユーザー自身による自由な拡張が可能ですからです。

また、Rは統計解析に特化した関数や、グラフィックス作成をサポートする関数を多数備えています。同時にオブジェクト指向のプログラミング言語でもあります。

Rは、SおよびS-PLUSという統計解析環境、あるいは、その思想を前身としています。Rの開発プロジェクト(The R Project for Statistical Computing)は1992年にRoss IhakaとRobert Gentlemanによって開始され、翌年に最初のバイナリが公開されました。続いて1997年にR開発チーム(R Development Core Team)が結成され、その翌年にはRのアーカイブネットワークであるCRAN(The Comprehensive R Archive Network)が設立されています。Rおよび付属パッケージはCRANを中心に公開されています。

RはGPLというライセンス形態で公開されています。これは単にRは無料で使えるということではなく、Rは商用が目的であっても自由に使えるということですが、詳細はRのプロジェクトサイトを参照してください。

- Rの開発プロジェクト

- URL <http://www.r-project.org/>

- CRAN

- URL <http://cran.r-project.org/>

III Rでできること

Rの主な用途は、データの解析と可視化(グラフィックス作成)です。

Rが登場する以前、データ分析(統計解析)といえば、非常に高価な専用ソフトウェアを使うのが当たり前でした。しかしながら、Rには市販の専用ソフトに勝るとも劣らない多くの機能が備わっています。

平均値の検定(t検定)のような初歩的な分析から、マルコフ連鎖モンテカルロ法によるシミュレーションのような高度な解析まで、Rだけで実現することが可能です。

さらにRには豊富なグラフィックス機能が備わっており、データの概要を確認するためのシンプルな棒グラフから、3Dグラフィックスを作成してマウス操作により描画角度を自由に調整するなど、多彩なグラフィックス機能を満喫できます。もちろん、プレゼンや出版物のために高品質の画像(PNGなど)やPDFファイルを作成することも可能です。それどころか、プレゼンテーション資料をR上で作成し、そのままインターネット上に公開することも可能です。

また、Rはコンピュータのバックグラウンドで動作させることも可能であり、夕方にデータを与えて実行しておき、翌朝、その結果を確認するような使い方も可能です。

III R 3.0.0以降の新機能について

2013年4月に、Rはバージョン3の太台に乗りました。大きな変更点は、R本体に関していえば、64bit版でLong Vectorがサポートされたことでしょう。R 3.0.0以前ではベクトルで利用可能な要素数は $2^{31}-1$ 、すなわち2,147,483,647に限定されていました。これは添字が32bitに限定されていたからです。R 3.0.0からは64bit環境においては 2^{52} までの添字を指定できるようになりました。ただし、行列や配列ではそれぞれの次元サイズは現在も $2^{31}-1$ に限定されています。

Windows64bit版Rでは、メモリ制限のデフォルトは搭載されたRAM量に限定されていますが、これは実行時オプションとして「-max-mem-size」を指定することで変更可能です。

また、Rのパッケージにはビネット(vignette)と呼ばれる解説用の小冊子を付録として同封することができますが、R 3.0.0からは「knitr」パッケージを変換エンジンとして指定できるようになりました。knitrについては、本書では60ページで解説しています。最近のR界では、レポートにRのスクリプトを組み込み、これをRでPDFやHTML、あるいはプレゼンテーション用スライドに変換するパッケージの開発が進んでいます。その代表が「knitr」パッケージであり、R 3.0.0でサポートされるようになりました。

その他、集計を行う「aggregate」関数でモデルが指定できるようになるなどの改良が加えられています。

なお、Rはバージョン2.15.1からニックネームが付けられ、起動画面にも表示されるようになりました。R 3.0.0以降のニックネームは下表のようになります。

Rのバージョン	ニックネーム
R 3.0.0(2013-04-03)	Masked Marvel
R 3.0.1(2013-05-16)	Good Sport
R 3.0.2(2013-09-25)	Frisbee Sailing
R 3.0.3(2014-03-06)	Warm Puppy
R 3.1.0(2014-04-10)	Spring Dance
R 3.1.1(2014-07-10)	Sock it to Me
R 3.1.2(2014-10-31)	Pumpkin Helmet
R 3.1.3(2015-03-09)	Smooth Sidewalk
R 3.2.0(2015-04-16)	Full of Ingredients
R 3.2.1(2015-06-18)	World-Famous Astronaut
R 3.2.2(2015-08-14)	Fire Safety
R 3.2.3(2015-12-10)	Wooden Christmas-Tree
R 3.2.4(2016-03-10)	Very Secure Dishes
R 3.3.0(2016-05-03)	Supposedly Educational

Rのインストール

III Rのダウンロードとインストール

WindowsやMacにRをインストールする場合は、CRANあるいはCRANのミラーサイトからRをダウンロードします。公式サイトでは日本のミラーサイトとして、次の2つが登録されています。

- 統計数理研究所

URL <http://cran.ism.ac.jp/>

- 山形大学

URL <http://ftp.yz.yamagata-u.ac.jp/pub/cran/>

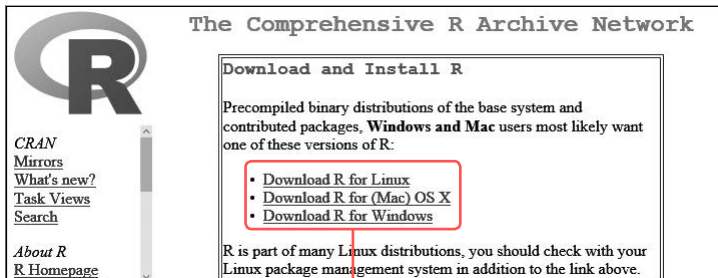
Rは年に数回バージョンアップが行われます。本書は、2013年4月3日に公開されたR 3.0.0以降のバージョンをもとに説明しています。

III Windowsへのインストール

本書の執筆時点でのRの最新バージョンR 3.3.0は、Windows XP以降であれば動かすことができます。ただし、Windows XPは、すでにMicrosoft社によるサポートが打ち切られており、利用するべきではないでしょう。

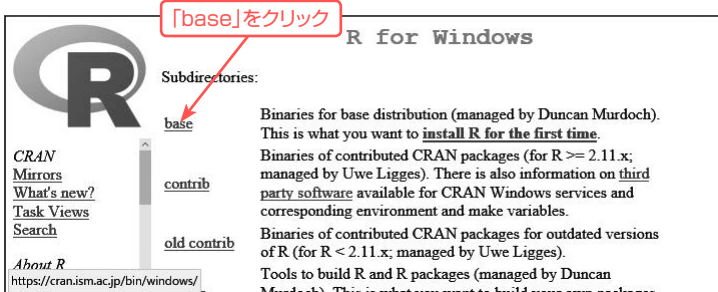
ちなみに、R 2.12.0以降のバージョンでは、32bit版と64bit版の両方のWindowsに対応しています。64bit版のWindowsにR 3.3.0をインストールすると、デスクトップに「R i386 3.3.0」と「R x64 3.3.0」と2つのアイコンが表示され、どちらからもRを起動することができます。

CRANのミラーサイトはいずれも同じデザインで統一されており、左と右のフレームに分割されています。右フレーム上の「Download and Install R」のセクションに「Linux」「Mac OS X」「Windows」と表示されたリンクがあります。



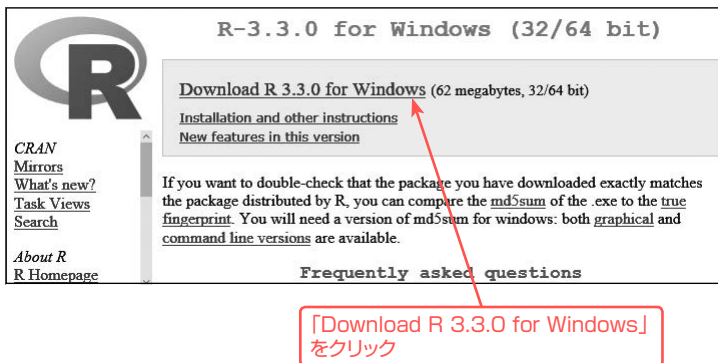
使用するOSにより選択する
リンクが異なる

たとえば、「Windows」をクリックすると「R for Windows」というページが表示され、そこに「base」と書かれたリンクがあります。その下の「contrib」は個別パッケージのバイナリファイルが置かれたリンクです。



「base」をクリックすると、大きく「Download R *** for Windows」というリンクが表示された画面が表示されます（「*」の部分はバージョンを表す数字）。

このリンクをクリックすると、Rと基本パッケージのセットをダウンロードできます。



インストールは、ダウンロードしたファイルをダブルクリックして実行します。インストーラが起動して、インストールの完了まで何度か確認を求めるダイアログボックスが表示されますが、基本的には[OK]ボタンや[続ける]ボタンをクリックしていきます。

また、環境によってはダイアログボックスのメッセージが文字化けすることがあります。これを回避するためには、インストール開始直後の画面で言語として「English」を選択します。この言語選択はインストール作業中の設定であり、R本体での言語設定ではありません。

なお、Windows版では異なるバージョンのRを複数インストールすることが可能です。


Macへのインストール

Mac OS X 10.6(Snow Leopard)以降で動作します。Mac OS 8.6から9.2、Mac OS X 10.1の場合、R 1.7.1であれば動く可能性があります。OS X 10.4(Tiger)やPowerPC版Macは、下記のURLに登録されているバイナリファイルからインストールすれば動きます。

URL <https://cran.ism.ac.jp/bin/macosx/old/>

本書執筆時点の最新のバージョンであるR 3.3.0は、Mac OS X 10.9(Mavericks)以降で動作します。

ダウンロードは、CRANのサイトの「Download and Install R」のセクションから「Mac OS X」を選びます。「R-*.*.pkg(latest version)」を選んでダウンロードし、ダブルクリックしてインストールします。



R for Mac OS X

This directory contains binaries for a base distribution and packages to run on Mac OS X (release 10.6 and above). Mac OS 8.6 to 9.2 (and Mac OS X 10.1) are no longer supported but you can find the last supported release of R for these systems (which is R 1.7.1) [here](#). Releases for old Mac OS X systems (through Mac OS X 10.5) and PowerPC Macs can be found in the [old](#) directory.

CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)
[The R Journal](#)

Software
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)

R 3.3.0 "Supposedly Educational" released on 2016/05/03

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type
`md5 R-3.3.0.pkg`
in the *Terminal* application to print the MD5 checksum for the R-3.3.0.pkg image. On Mac OS X 10.7 and later you can also validate the signature using
`pkgutil --check-signature R-3.3.0.pkg`

Files:

R-3.3.0.pkg
MD5-hash: [1274d0f9a12731d328125774c639a](#)
SHA1:
hash: [a0ae5823cb826d228d7791c9946c634a888cc48](#)
(ca. 71MB)

R 3.3.0 binary for Mac OS X 10.9 (Mavericks) and higher, signed package. Contains R 3.3.0 framework, R.app GUI 1.68 in 64-bit for Intel Macs, Tcl/Tk 8.6.0 X11 libraries and Tinfo 5.2. The latter two components are optional and can be omitted when choosing "custom install", it is only needed if you want to use the `tcltk` R package or build package documentation from sources.

「R-3.3.0.pkg」をクリック

インストールすると、「アプリケーション」フォルダにRのアイコンが作成されます。

Linuxへのインストール

Linuxの代表的なディストリビューションであるUbuntuには、Rのバイナリファイルが用意されています。Synapticを起動してr-baseを検索するか、あるいはターミナルから「`sudo apt-get install r-base`」を実行することでインストールすることができます。ただし、古いバージョンのUbuntuでは、最新のRおよび関連パッケージをデフォルトでは利用できません。次のURL

に記述された方法に従って、リポジトリのデータベースを更新することで、最新のバージョンを利用できるようになります(アクセスエラーが生じる場合は「https」の部分を「http」に変更してみてください)。

● UBUNTU PACKAGES FOR R

URL <http://cran.r-project.org/bin/linux/ubuntu/README>

Ubuntu 14.04の場合の設定方法は次のようになります。

まず、「/etc/apt/sources.list」ファイルを、ターミナルで「`sudo gedit /etc/apt/sources.list`」などとして管理者権限で開き、次の行を最後尾に追記します。

```
deb https://cran.ism.ac.jp/bin/linux/ubuntu trusty/
```

そして、ターミナルで次のコマンドを実行します。

```
$ sudo apt-get update
$ sudo apt-get install r-base
```

なお、Ubuntuに開発環境を導入していない場合、パッケージの導入でエラーが生じることがあります。次のコマンドを実行すると、Rのパッケージなどのインストールに必要な開発環境が自動的にインストールされます。

```
$ sudo apt-get install r-base-dev
```

コンソールで「R」と入力して「Enter」キーを押すと、Rが起動します。

▶ ソースからのコンパイル

UbuntuでRそのものをソースからコンパイルするには開発環境が必要です。最低限、次のコマンドでインストールしておきます。

```
$ sudo apt-get install build-essential gfortran libreadline-dev xorg-dev
```

34ページの図のCRANのトップページにあるリンク「The latest release (201* - * - *) : R-*.tar.gz」をクリックし、ダウンロードしたファイルを解凍して、コンパイルします。

```
$ tar xzf R-*.tar.gz
$ cd R-*.
$ ./configure --enable-R-shlib
$ make
$ make check
$ sudo make install
```

LaTeXなどの環境が整っていない場合はマニュアルの生成が行えないなど、いくつかの警告が表示されますが、Rの起動には支障ありません。

Rの起動と終了

III Rの起動

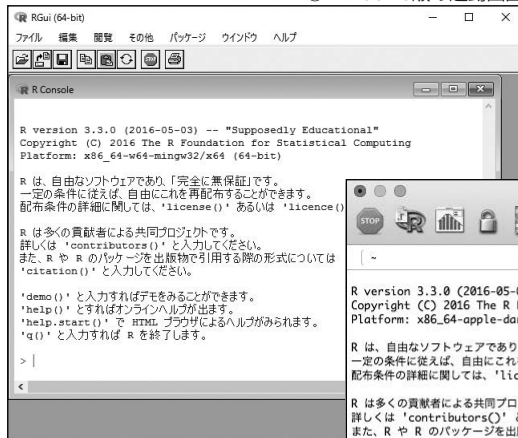
Rの起動はWindows版であれば、インストール時に自動的に作成されるデスクトップ上の「R」のアイコンをダブルクリックします。または、[スタート]メニューから[すべてのプログラム]→[R]→[R 3.3.0]を選択しても起動することができます。

Windows版では、R 2.12.0以降からは32bit版と64bit版の2つのアイコンが表示されます。64bit版は、「R x64 3.3.0」のようにファイル名に「x64」が追加されています。

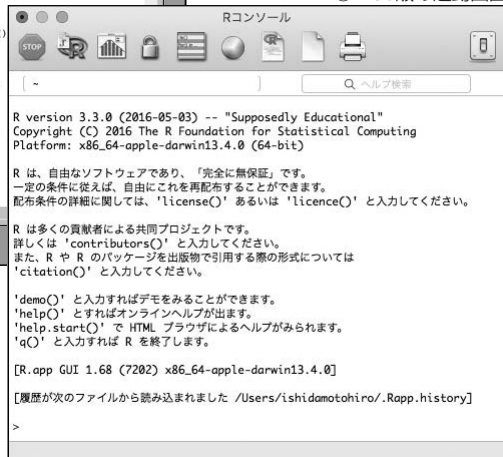
Macの場合はユーザーのホームにある「アプリケーション」フォルダに「R」のアイコンが表示されます。なお、Mac版の場合のデフォルトのインストール先は「/Library/Frameworks/R.framework」以下になります。

ちなみに、Rを起動してから終了するまでを「セッション」と呼びます。

●Windows版の起動画面



●Mac版の起動画面



III Rの終了

終了方法そのものはWindowsやMacの他のソフトウェアと同じです。ただし、Rの場合、終了時に「作業スペースを保存しますか」というダイアログボックスが表示されます(R付属のエディタでスクリプトを編集中は、さらにスクリプトを保存するかどうかを確認するダイアログボックスも表示される)。本書では、「作業スペース」は保存しないことをお勧めします。

作業スペースを保存しない理由

Rではセッション中に、データを表すオブジェクトや関数が多数作成されます。通常、オブジェクトは「作業スペース」内に保存されます。「作業スペースを保存しますか」というダイアログボックスは、これらのオブジェクトを次回起動する際に復元するかどうかを確認するために表示されます。

「はい」を選ぶと、デフォルトのフォルダ(これは「getwd」関数で確認できる)に、「RData」ファイルが作成されます。このファイルにセッション中に生成されたオブジェクトがまとめて保存(記録)されています。このファイルは次にRを起動した際に、自動的に読み込まれるため、起動と同時に前回のセッションを続けて作業できるようになります。

ただし、このファイルは場合によっては非常に大きなサイズとなります。Rはオブジェクトをすべてメモリに展開するため、前回のセッションを不用意に読み込むとPCのメモリを圧迫することになります。さらに、前回のセッションで作成されたオブジェクトが指し示す値をセッション終了の直前に変更していたにもかかわらず、そのことを忘れて新しいセッションを継続したため、まったく予期しない(あるいは間違った)結果が出力される可能性もあります。

Rはプログラム言語であり、処理はコード(命令)を書いて実行することが基本です。これらのコードをまとめたファイルを「スクリプト」と呼びます。Rにはスクリプトファイルを読み込むメニューがあり、また、スクリプトに記述したコードを直接Rに送って実行する機能もあります(413ページ参照)。セッション中の操作をすべてスクリプトにまとめて記述しておけば、終了時に作業スペースを保存しなくとも、次のセッションで同じ操作を繰り返すことで前回利用していたオブジェクトを再現できます。また、スクリプトの範囲を指定して実行することも可能です。

セッション中のオブジェクトは「RData」とは別の名前を指定して保存できます。別名で保存すると、次回にRを起動した際に、前回のオブジェクトが自動的に再現されることはありません。

COLUMN 32bit版と64bit版を選択して起動する

WindowsのRには、32bitと64bitの両方のバイナリが同梱されています。OS側で64bitに対応しているのであれば、通常は64bit版を利用して問題ありません。ただし、Rの追加パッケージの中には、64bit版Rには対応していないものもあります。

コマンドライン版のRを起動する場合、Windowsでは「C:\Program Files\R\R-3.3.0\bin\R.exe」を実行する際に、「--arch」引数を追加します。「--arch 32」または「--arch 64」と実行バイナリを指定することができます(668ページ参照)。

関連項目 ▶▶▶

- オブジェクトを保存する／読み込む p.399
- Rのスクリプトコードを読み込む p.413

Rコンソールとスクリプトの実行

III Rコンソールの基礎

Rを起動すると、コンソールというウィンドウが表示されます。基本的にRへの命令はコンソールにコードを入力し、「Enter」キーを押すと実行されます。コードが長くなる場合は、改行を挟んで記述することができます。たとえば、「iris」データ(778ページ参照)で、「かくの幅(Sepal.Width)」を「かくの長さ(Sepal.Length)」と「花弁の幅(Petal.Width)」と「花弁の長さ(Petal.Length)」で回帰するモデル式は次のように長くなります(回帰については475ページ参照)。

```
lm (Sepal.Width ~ Sepal.Length + Petal.Width + Petal.Length, data = iris)
```

このような場合、次のように適当な位置で改行し、残りを次行に送ることもできます。

```
> lm (Sepal.Width ~ Sepal.Length +  
+     Petal.Width + Petal.Length,  
+     data = iris)
```

Call:

```
lm(formula = Sepal.Width ~ Sepal.Length + Petal.Width + Petal.Length,  
    data = iris)
```

Coefficients:

(Intercept)	Sepal.Length	Petal.Width	Petal.Length
1.0431	0.6071	0.5580	-0.5860

コードを未完のまま改行すると、コンソールの先頭に「+」が表示されます。これは、ユーザーにコードの残りを入力するように促すサインです。残りのコードの入力をキャンセルしたい場合は、Windows版とMac版であれば「Esc」キーを押します。Ubuntuなどでターミナル上で実行している場合は、「Ctrl」+「C」キーを押します。

また、コンソール画面に入力した内容は履歴として残るので、矢印キーを使うことで、以前の入力呼び出すことができます。

R 2.11.0からは、コード補完機能が備わっています。コード補完機能とは、たとえば、「iris」データ名の「ir」まで入力したところで「Tab」キーを押すと、コンソールが「iris」と残りを補ってくれる機能です。複数の候補がある場合は、ダイアログが表示されます。

III 効率的なコードの実行方法

しかしながら、Rコンソールへの入力は快適とはいえません。特にWindows版では、コードに日本語などが含まれている場合、カーソル位置がずれる不具合があります。

また、セッション中はコマンド履歴をたどれるとはいえ、古い履歴の場合は矢印キーを何度も繰り返し押す必要があります。

別の機会や別のPCで、同様のセッションを実行する必要がある場合には、コードは独立したファイルとして保存する方が適切です。

Windows版であれば、[ファイル]メニューから[新しいスクリプト]を選択することで「Rエディタ」というウィンドウが開きます。ここにコードを入力して、コードのどこかにカーソルをあわせるか、コードを範囲指定します。そして右クリックして表示されるメニューから[カーソル行または選択中のRコードを実行]を選択するか、「Ctrl」+「R」キーを押すと、そのコードがRコンソールに送られて実行されます。

Mac版の場合には、[ファイル]メニューから[新規文書]を選択することでスクリプト作成ウィンドウが表示されるので、コードのどこかにカーソルをあわせるか、コードを範囲指定した上で、「Command」+「Enter」キーを押すと、そのコードが実行されます。

なお、Rのコードを記述したスクリプトには拡張子として「.R」（あるいは「.r」）を付ける慣習があります。「source」関数に引数としてファイル名を指定して実行すると、コンソールからスクリプトの内容をまとめて実行することもできます。

III プロンプトの表示の変更

Rのコンソールでは、入力を促す「>」や「+」が表示されます。これは「options」関数(48ページ参照)で変更できます。

```
> # コンソールのプロンプトを変更する
> # 現在のプロンプトを確認する
> options ("prompt")
$prompt
[1] "> "

> options ("continue")
$continue
[1] "+ "

> # デフォルトの表示
> plot (1:10, 1:10,
+       main = "プロンプト変更前")

> # プロンプトをishida>とput+に変更する
> op <- options (prompt = "ishida> ", continue = "put+")
ishida> options(continue = "put: ")
ishida> plot (1:10, 1:10, main = "プロンプトを変更")
ishida> plot (1:10, 1:10,
put+   main = "プロンプトを変更")
ishida> # 元に戻す
ishida> options(op)
```

「options」関数の「prompt」属性と「continue」属性には、文字列を設定することができます。この変更はセッションを終了するまで有効です。これをデフォルトの設定に変更したい場

合は、環境設定ファイルで指定します(45ページ参照)。ただし、Emacsなどのエディタを介して間接的にRを操作している場合にプロンプトを変更すると、エディタごとフリーズすることがあるので注意してください。

COLUMN 「history」関数の利用

Rではセッション中に実行したコマンドを履歴として残しておく機能があります。コンソールで「history()」と実行すると、Windows版では新たにウィンドウが表示され、これまでのセッションで実行したコードが一覧として表示されます。デフォルトでは25行分のコードのみ表示されますが、「max = Inf」と引数を指定すると保存されているすべての履歴が表示されます。

記憶する履歴のサイズはデフォルトでは512行ですが、環境変数「R_HISTSIZE」に任意の整数を指定することで変更できます。たとえば、「.Renviron」ファイルに次のように追記すると5000行の記憶が可能となります。

```
R_HISTSIZE=5000
```

これはRのコンソールで「Sys.getenv ("R_HISTSIZE")」を実行して確認することができます。また、履歴はキーボードの矢印キーの上ないし下を押すとコンソールでたどることができます。

履歴は、本書では推奨しませんが、Rの終了時に「作業スペースの保存」を選択すれば「.Rhistory」ファイルとして保存されます。作業スペースを保存したくないが、履歴を残したい場合は、「savehistory」関数を実行すると「.Rhistory」ファイルが更新されます。

ユーザーのホームフォルダの「.Rprofile」に次のように追記しておくと、Rの終了時に自動的に履歴を保存します。

```
.Last <- function ()  
  if(interactive ()) try (savehistory ("Rhistory"))
```

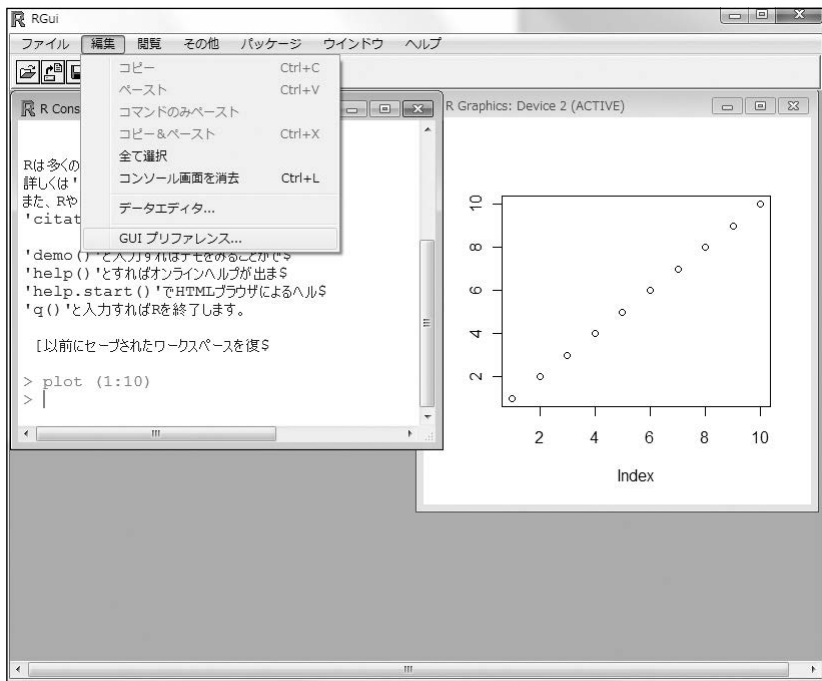
関連項目 ▶▶▶

- Rの環境設定ファイル p.45
- Rのセッション用オプションの指定 p.48
- オブジェクトを保存する／読み込む p.399
- Rのスクリプトコードを読み込む p.413

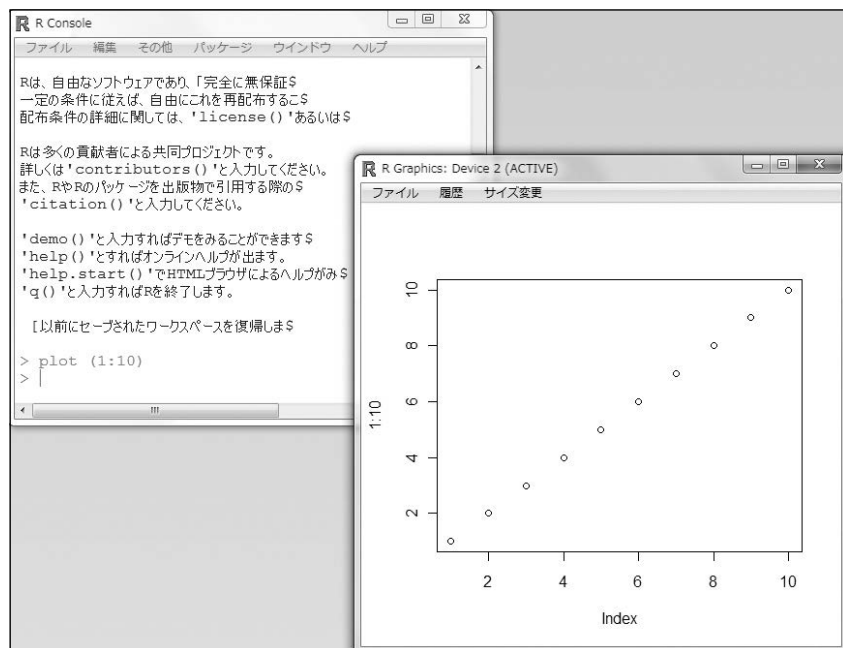
Windows版のRの画面に関する注意点

III Windows版固有の設定

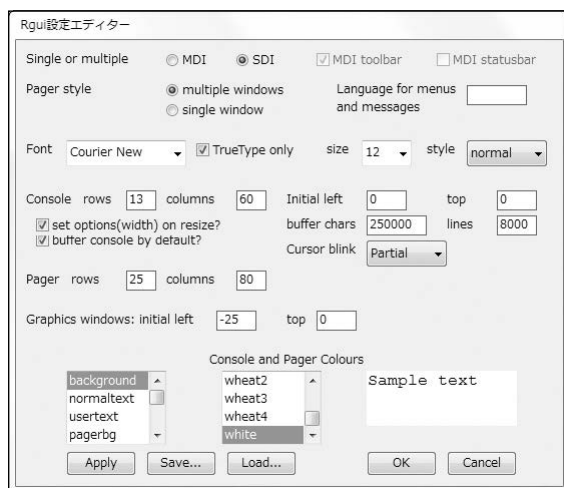
Windows版Rを起動すると、アプリケーション・ウィンドウの内部に、さらにコードを入力するためのRコンソール・ウィンドウがあります。プロットを作成すると、やはりアプリケーション・ウィンドウの内部にプロット・ウィドウが表示されます。これを「MDI(Multiple Document Interface)」スタイルといいます。



これに対してRコンソール・ウィンドウやプロット・ウィンドウが、それぞれ独立して表示されるスタイルを「SDI(Single Document Interface)」といいます。



「SDI」に変更するには、[編集]メニューから[GUIプリファレンス]を選択すると表示される「Rgui設定エディター」ダイアログボックスで[SDI]をONにし、[Save]ボタンをクリックします。これはユーザーのホームフォルダに「Rconsole」というファイルを作成します。そして[OK]ボタンをクリックします。すると、この設定はRを再起動後に有効になると表示されます。



なお、古いバージョンのRでは、日本語が文字化けすることもあります。この場合も「Rgui設定エディター」で[Font]に「MS Gothic」などの日本語を表示可能なフォントを選択します。

Rの環境設定ファイル

2種類の環境設定ファイル

Rを起動すると、デフォルトで作業用フォルダやパッケージのインストール用フォルダなどが設定されています。これらの設定は変更することができます。そのためには環境設定ファイルを作成する必要があります。環境設定のファイルには2種類のファイルがあります。

- ライブラリの保存先などの環境変数を指定する「Renviron」
- CRANのミラーサイトやグラフィオブションなどをRのコードで指定した「Rprofile」

さらに、それぞれについて全体(site)の設定とユーザーごとの設定があります。

「Renviron」の保存場所

R全体の設定では、環境変数「R_ENVIRON」で指定されたファイルを参照します(70ページ参照)。「R_ENVIRON」が未設定であれば、コンソール画面で「R.home ()」を実行して表示されるRのホームフォルダの「etc」フォルダ内の「Renviron.site」を参照します。

一方、ユーザーごとの設定は、環境変数「R_ENVIRON_USER」で指定されたファイルを参照します。「R_ENVIRON_USER」が未設定の場合は、ユーザーのホームフォルダ(「C:¥Users¥ishida」など)にある「.Renviron」を参照します。

なお、32bitと64bitの両方のバイナリが同封されているWindows版Rでは、「.Renviron.i386」または「.Renviron.x64」を作成すると、それぞれの設定が「.Renviron」より優先されます。

「Rprofile」の保存場所

「Rprofile」の設定でも全体の設定ファイルを指示する「R_PROFILE」を参照し、これが未設定であれば、Rのホームフォルダ内の「etc」フォルダの「Rprofile.site」を参照します。

また、ユーザーごと設定は、環境変数「R_PROFILE_USER」で指定されたファイルを参照します。これが未設定の場合はユーザーのホームフォルダの「.Rprofile」を参照します。

一般的な起動時の設定場所

一般的には、Rの起動時の設定を指定するにはユーザーのホームフォルダに「.Renviron」と「Rprofile」を作成します。

環境変数の一覧は、Rコンソールで「Sys.getenv ()」を実行することで確認することができます。

III 「.Renviron」の設定

必要があれば、「.Renviron」には環境設定を記述します。たとえば、ライブラリのパス(追加パッケージの場所)を次のように指定します。「=」の前が環境変数、後ろがその設定値です。

```
R_LIBS_USER=C:/Users/ishida/Documents/R/win-library/3.3
R_LIBS_SITE=C:/Program Files/R/R-3.3.0/library
```

1行目はユーザー固有のライブラリ保存先であり、2行目はサイトの保存先です。ただし、ここに例示しているのはWindowsにおけるデフォルト設定です。そもそも通常はユーザーが「.Renviron」を作成する必要はありません。

III 「.Rprofile」の設定

「.Rprofile」には、CRANのサイトやフォント設定など、Rの起動時に実行するRのコードを記述します。なお、Windowsの標準エディタ「メモ帳」などで作成して保存する場合は「.txt」などの拡張子が自動的に付与されないように注意してください(保存ファイル名を引用符で囲んで指定する)。

Windowsで典型的な設定をRjpWikiの関連ページから引用します。

```
options (repos="https://cran.ism.ac.jp")
setHook (packageEvent ("grDevices", "onLoad"),
  function (...) grDevices::pdf.options (family="Japan1"))
setHook (packageEvent("grDevices", "onLoad"),
  function (...) grDevices::ps.options (family="Japan1"))
```

最初の行はパッケージのインストールなどでアクセスするCRANのサイトを設定しています。

2行目以降は、Rで日本語を含むPDFファイルやPostScriptファイルを作成する際に、文字化けが起こらないようにする設定です。

なお、筆者の作成した設定ファイルを公開しています。Rのコンソールで以下を実行すると、グラフで日本語を扱う場合に必要になる設定が書かれた「.Rprofile」をダウンロードし、ユーザーのホームフォルダに保存します(同名のファイルが存在している場合は「dot.Rprofile-日付.txt」という名前で保存され、新たにダウンロードされた「.Rprofile」に置き換えられます)。新しい設定ファイルは、Rを起動し直すと有効になります。

```
source("http://rmecab.jp/R/Rprofile.R")
```


COLUMN

Rの起動時と終了時の処理を指定する

「Rprofile」にRを起動直後と終了直前の処理を記述しておくことができます。起動時に実行したいコードは、次の「{ }」内に記述します。

```
.First <- function () { }
```

また、終了時に実行したいコードは、次の場所に記述します。

```
.Last <- function () { }
```

次のように「Rprofile」内に記述すると、Rの起動時にR開発者によるメッセージをランダムに表示することができます（ただし、「fortunes」パッケージをインストールしている場合）。終了時には履歴を保存します。

```
.First <- function () {
  tmp <- suppressWarnings (require ("fortunes", quietly = TRUE))
  if (tmp) {
    print (fortunes::fortune())
  }
}
.Last <- function()
  if (interactive()) try (savehistory (".Rhistory"))
```

起動時の処理では「require」関数で「fortunes」を読み込んでいますが、成功すると「TRUE」が返り値となるので、これを「tmp」に保存しています。成功した場合は、「fortunes」パッケージの「fortune」関数を「print」してメッセージを表示します。「::」はパッケージ内のオブジェクトにアクセスするための演算子です。

Rのセッション用オプションの指定

III 「options」関数の利用

コンソール画面の表示幅や小数点の表示桁数など、Rを実行中(セッション)の設定を変更するには「options」関数を利用します。

「Rprofile」に記述しておく、R起動時の設定になります。

```
> # デフォルトの表示幅
> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S"
[20] "T" "U" "V" "W" "X" "Y" "Z"
> getOption("width")
[1] 80
> # 表示幅を変更
> options(width = 60)
> LETTERS
[1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N"
[15] "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
> # 数値の表示幅
> pi
[1] 3.141593
> getOption("digits")
[1] 7
> # 数値の表示桁数を変更
> options(digits = 16)
> pi
[1] 3.141592653589793
> # 指数表示
> 10^5
[1] 100000
> getOption("scipen")
[1] 3
> options(scipen = 1)
> 10^5; 10^6
[1] 100000
[1] 1e+06
> options(scipen = 3)
> 10^7
[1] 10000000
> # 起動時にロードするパッケージ
> (x <- getOption("defaultPackages"))
[1] "datasets" "utils" "grDevices" "graphics"
[5] "stats" "methods"
```



```
> # 以下を環境設定ファイル「.Rprofile」に記載しておく
> # 「lattice」パッケージも起動時にロードされる
> options (defaultPackages = c (x, "lattice"))
> # CRANミラーにデフォルトのサーバーを設定
> # これにより、インストール時の確認が不要になる
> options(repos = "https://cran.ism.ac.jp")
```

作業フォルダの設定

作業フォルダとは

作業フォルダとは、Rの実行中に起点となるフォルダのことで、データファイルなどを作業フォルダに保存しておく、フォルダへのパスを指定することなく、ファイル名だけでデータファイルを読み込むことができます。

```
> read.csv ("Chapter01/myData.csv")
```

データファイルが作業フォルダにない場合、ファイルの保存先のパスを指定する必要があります。

```
> read.csv ("C:/Users/ishida/Documents/R_Handbook_***/Chapter01/myData.csv")
```

現在の作業フォルダは、「getwd()」を実行すると確認できます。変更する場合には、「setwd("C:/working")」などと実行します。

起動時の作業フォルダの指定

作業フォルダをRの起動と同時に指定したい場合は、「.Rprofile」に次のように書き加えます。

```
setwd ("C:/Users/ishida/Documents/R_Handbook_***")
```

また、Windows版の場合は、デスクトップ上のアイコンを右クリックし、プロパティで「作業フォルダ」を指定しても構いません。

●「作業フォルダ」の指定



利用したい機能の検索

キーワードを使って利用したい機能を検索する

利用したい機能をキーワードで検索するには、「apropos」関数を使用します。キーワードには、単語あるいは単語の綴りの一部を指定します。

たとえば、キーワードとして「chi」を指定すると、単語の中にchiを含むキーワードがすべて抽出されます。

```
> # 「chi」(カイ)を文字列の一部に含む関数を表示する
> apropos ("chi")
[1] ".Machine"      "ChickWeight"  "chickwts"     "chisq.test"   "dchisq"
[6] "pchisq"        "qchisq"       "rchisq"
```

検索結果の絞り込み

「apropos」関数の「mode」引数にオブジェクトのモードを指定すると、検索結果を絞り込むことができます。

```
> # 引数「mode」に「関数」を指定して検索結果を絞る
> apropos ("chi", mode = "function")
[1] "chisq.test" "dchisq"      "pchisq"      "qchisq"      "rchisq"
```

なお、「mode」には、「mode」関数の出力となる文字列を指定することができます。詳細は88ページを参照してください。

「find」関数による検索

「find」関数は、指定したキーワードで環境やパッケージ名を検索します。実行例では、「simple.words = FALSE」のオプション指定を行って、部分一致で検索を実行しています。このオプションを外すと、「chi」に完全一致するオブジェクトを検索することになりますが、Rには「chi」というオブジェクトは存在しないので、検索結果は0個になります。

```
> # 「simple.words」引数で部分一致を指定
> find ("^chi", simple.words = FALSE)
[1] "package:stats"  "package:datasets"
```

正規表現を使った検索

正規表現と組み合わせることで柔軟な検索が可能になります。上の例では「^」を利用して行頭を指定しています。正規表現については150ページを参照してください。

Rの統合環境「RStudio」を使う

III RStudioについて

Rで作業する場合、データを読み込み、オブジェクトを操作し、コードを書いて統計量を算出する、あるいはグラフィックスを作成し、保存するという作業を繰り返します。

こうしたルーティンに伴う手間はできるだけ軽減したいものです。RStudioはコード補完機能やオブジェクト操作性に優れているだけでなく、プロジェクト管理やプレゼンテーションの作成と公開まで、Rの可能性をフルに引き出す機能を備えています。

ここでは、RStudioのインストール方法と基本機能の解説を行います。なお、RStudioのサイトには詳細なチュートリアルがあります。

● RStudio - RStudio IDE

URL <http://www.rstudio.com/ide/>

上記のページ下の「Learn more about RStudio」に各種ドキュメントやチュートリアルへのリンクが表示されています。英文ですが、サンプルコードもあり、眺めるだけでも参考になるでしょう。

III RStudioのインストール

RStudioにはデスクトップ版(Windows、Mac、Linux)とサーバー版がありますが、いずれも下記のURLから配布ファイルをダウンロードします。デスクトップ版はダブルクリックするだけでインストールが完了します。ただし、事前にR本体(R 2.11.1以降)をインストールしておく必要があります。

一方、サーバー版を導入すると、ブラウザからRStudioの機能を使えるようになります。『R言語上級ハンドブック』の338ページに導入方法が紹介されています。

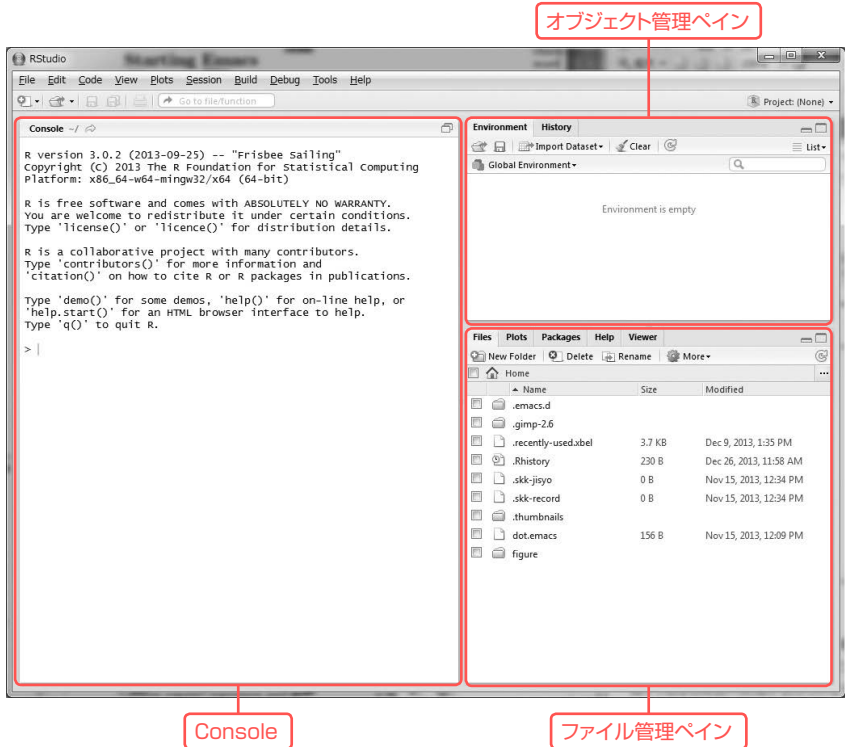
● RStudio - Home

URL <http://www.rstudio.com/>

III RStudioの構成

RStudioを起動すると、初期の状態ではウィンドウ内は3つのペインに分割されています。デフォルトでは左にConsole、右上にオブジェクト管理ペイン、右下にはファイル管理ペインが表示されています。これらの配置は変更することもできます。デフォルトの設定を変更する場合は、[Tools]メニューから[Global Options]を選択し、「Options」ダイアログボックスの[Pane Layout]で指定します。また、[Tools]メニューから[Project Options]を選択し、後述するプロジェクト単位でレイアウトを調整することもできます。

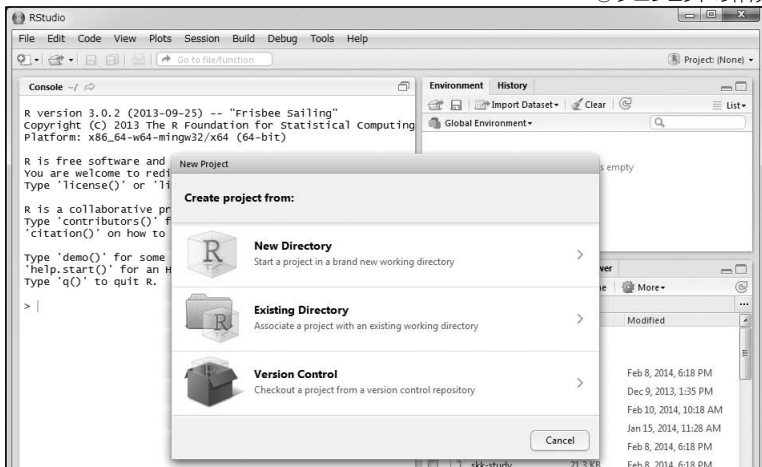
なお、RStudioのバージョンによってはConsoleに表示されているR起動メッセージが英語になっているかもしれません。ただし、この状態でも日本語は正しく表示されますし、ヘルプメッセージも日本語で表示されます。もし、ヘルプメッセージが英語で表示される場合は「Sys.setenv(LANGUAGE="ja")」を実行します。



III プロジェクトの作成

起動直後の状態のまま使い始めることも可能ですが、作業内容ごとにプロジェクトという単位で管理するのがベストです。ウィンドウ右上にある[Project: (None)]をクリックし、[New Project]を選択します。

● プロジェクトの作成



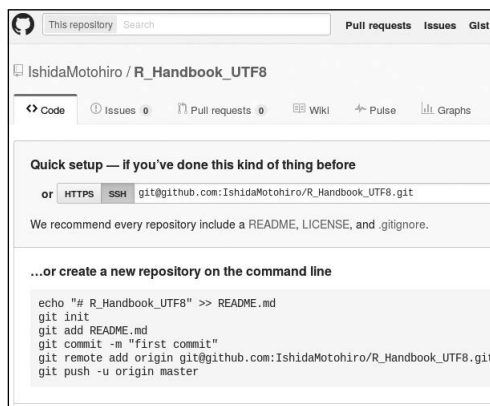
選択肢が3つありますが、通常は上の「New Directory」を選択します。「Version Control」というのは、Gitなどのバージョン管理システムを使ってプロジェクトを管理するためのオプションです(後述)。次に進むと「Empty Project」「R Package」「Shiny Web Application」が選択できます。「Empty Project」は空のプロジェクトです。「R Package」はパッケージを作成するためのプロジェクト、「Shiny Web Application」はRをブラウザで操作するスクリプト作成をサポートしています。いずれもRの拡張機能ですが、RStudioを使うことで開発が容易になります。

ここでは、「Empty Project」を選択します。適当なプロジェクト名とプロジェクトを作成するフォルダを指定します。プロジェクト名は英数半角文字のみを使うのが無難です。海外で作成されたパッケージなどを利用する場合、日本語などの全角文字で構成されたフォルダやファイルを読み込めないことがあります。

なお、プロジェクトは一度、設定したら、明示的に閉じない限り、次回、RStudioを起動した際もデフォルトのプロジェクトとして開かれます。プロジェクトを閉じる、他のプロジェクトで作業する、あるいはプロジェクトごとにオプション設定を変更する場合は、右上のプロジェクト名をクリックして、必要な項目を選びます。

▶ プロジェクトの管理

GitHubなどを使ってプロジェクトの管理を行うことができます。「<http://github.com/>」にアクセスし、「ダッシュボード」で「新しいリポジトリ」を用意します。たとえば、RStudioで「test」というプロジェクトを作成しているとします。そこでGitHub上で同じ名前のリポジトリを作成します。基本的にはGithubのレポジトリ設定画面で一番上にある「Repository name」を入力するだけで構いません。作成すると、ユーザーが次に行う手順(下図参照)を示したウィンドウが表示されるので、内容をコピーしておきます。



RStudioのプロジェクトを置いてあるパソコンにGit管理のためのソフトウェアをインストールします。Windowsの場合は「msysgit」(<http://code.google.com/p/msysgit/>)が簡単でしょう。Macでは、「MacPorts」(<http://www.macports.org/>)か、「Homebrew」(<http://mxcl.github.com/homebrew/>)を使ってインストールしてください。Ubuntuであれば、「`sudo apt-get install git`」としてインストールします。

Git管理用の鍵を作成します。Windowsでmysysgitを使う場合はGit Bashを起動します。LinuxまたはMacではターミナルを使います。次のように入力して鍵を作成します。

```
ssh-keygen -t rsa -C "メールアドレス"
```

上記のコマンドを実行すると、ユーザーのホームフォルダの「.ssh」というフォルダに「id_rsa」と「id_rsa.pub」の2つのファイルが作成されています(ただし、隠しフォルダなのでWindowsのデフォルトの設定や、MacのFinderだと表示されない)。前者は秘密鍵、後者は公開鍵です。

GitHubにログインして公開鍵を登録します。Account Settingsで左の「SSH Keys」をクリックし、「Add SSH Key」を選択します。Windowsの場合は付属のメモ帳などでファイル形式を「すべてのファイル」に変えて、「id_rsa.pub」を選択します。ファイルの内容をすべて範囲指定してコピーし、これをGitHubの画面にペーストして「Add key」をクリックします。

再びターミナルでRStudioのプロジェクトがあるフォルダへ移動します。プロジェクトのパスが「C:\Users\ishida\Documents\test」であれば、次のようにして移動します。

```
cd /d C:/Users/ishida/Documents/test
```

プロジェクトをGitHubサーバー上のレポジトリに登録します。次のように実行していきます(GitHubでレポジトリを作成直後に表示された手順)。ユーザー名を登録し、フォルダを初期化します。そして既存のファイルの登録、変更の設定、サーバーとの接続、最後にファイルの反映を行っています。

```
git config --global user.name "Motohiro ISHIDA"
git config --global user.email ishida-m@ias.tokushima-u.ac.jp
git init
git add .
git commit -m 'first commit'
git remote add origin git@github.com:IshidaMotohiro/R_Handbook_UTF8.git
git push -u origin master
```

最後にGitHubに登録したユーザー名とパスワードを入力します。これでGitHubへの登録は完了です。RStudioでファイルなどを編集すると、右上のペインのGitというタブで、更新されたファイルが青いMのラベル付きになります。左の[Staged]をONにして[Commit]ボタンをクリックします。新しいウィンドウが現われるので、右の[Comment Message]に変更点などのメモを書いて、右下の[Commit]ボタンをクリックします。変更を登録したというウィンドウが新規に現れますが、これは閉じて先のウィンドウ右上の[Push]ボタンをクリックし、ユーザー名とパスワードを入力します。GitへのPushに成功すれば、その旨を表示した簡単なダイアログが表示されます。

逆に、Githubに公開されているプロジェクトを取り込みたい場合は次のようにします。

- 1 RStudioを起動し、右上のプロジェクトから「New Project」を選択します。
- 2 次のダイアログで「Version Control」を選択します。続いて「Git」を選択します。
- 3 一番上の「Repository URL:」にGitHubで管理されているレポジトリのSSH欄の内容を入力

します。

④ その下の2つの欄は(同名のフォルダがなければ)デフォルトの設定のままでよいでしょう。

「Create Project」をクリックすると、指定されたURLからレポジトリがダウンロードされます。ファイルに修正を加えたら「Commit」し、そして「Push」という作業を繰り返します。

Githubにssh公開鍵を登録しているにもかかわらず、RStudio左上の「Git」タブを操作するとアカウントとパスワードが求められる場合は、以下の手順を解決できることが多いです。

- ① [Tools]→[Global Options]→[Git/Svn]でSSHキーのフォルダにGithubに登録した鍵が含まれているか確認します。
- ② 右上の「Git」タブの歯車ボタンを押して、一番下の[Shell]を選択します。
- ③ 起動したShellで以下のように入力します(ユーザ名とレポジトリ名は置き換える)。

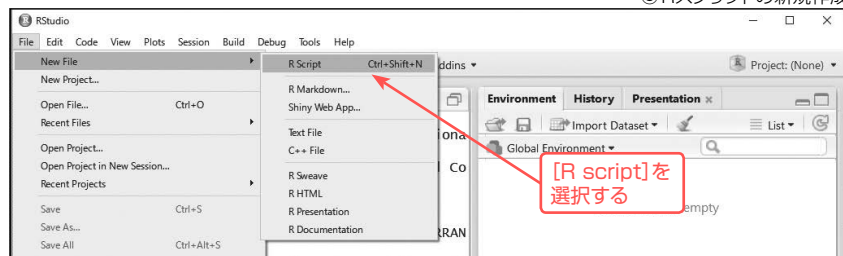
```
git config remote.origin.url git@github.com:ユーザ名/レポジトリ名.git
```

- ④ Shellを閉じます。

III Rスクリプトの作成

Consoleペインにコードを入力して実行することも可能ですが、通常は、別にスクリプトファイルを用意して、それにコードを記述して実行します。[File]メニューから[New File]を選択(あるいはメニュー下左端の白いアイコンをクリック)すると、RStudioがサポートするファイル種類の一覧を確認することができます。[R script]というのが通常のRスクリプトになります。

●Rスクリプトの新規作成

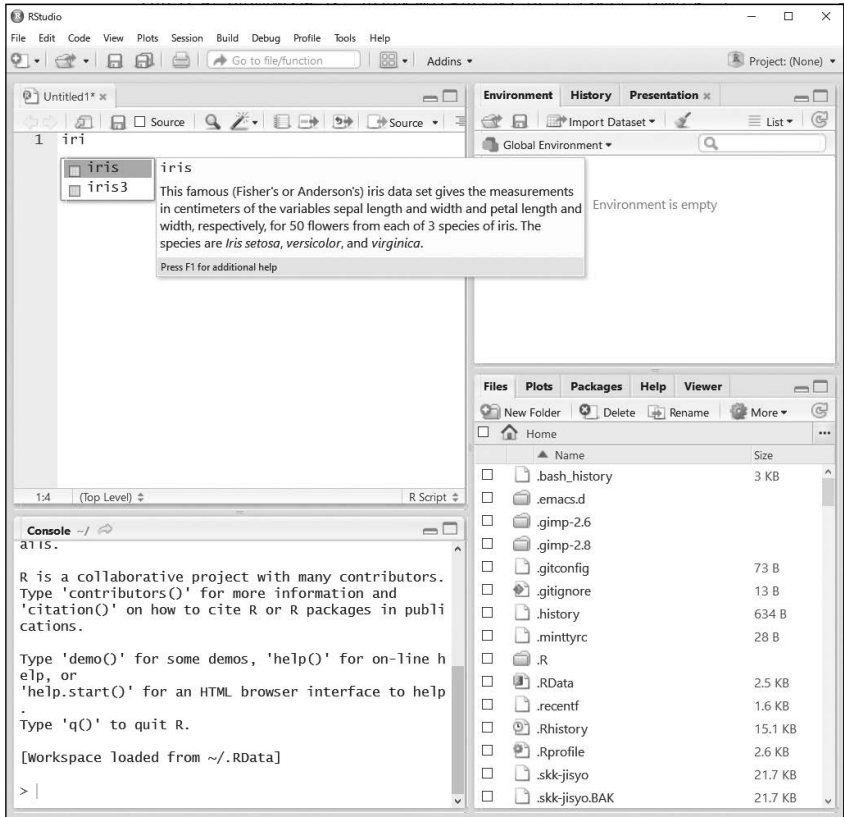


左上にスクリプトファイル用のペインが追加されます。ここにRのコードを書いて実行していくのがRStudioの基本的な使い方です。入力を行うとファイル名を示すタイトルが赤い文字に変わり、「*」(アスタリスク)が付記されます。これはファイルが修正されたことを表しています。新規ファイルは適当なタイミングで保存しておきます。[File]メニューから[Save As]を選択します。なお、日本語を含むファイルを保存する場合は、文字コードに注意が必要です。デフォルトではOSの文字コードに設定されるはずですが、WindowsではShift_JIS(CP932)であり、MacではUTF-8です。データ操作で問題が生じることはないですが、後述するMarkdownを使ったプレゼンテーション作成する場合にはUTF-8で作成する方がトラブルは少なくなります。ファイルの文字コードを変更する場合は、[File]メニューから[Save with Encoding]を選択し、ダイアログで「UTF-8」を選択します。

▶ コード補完

スクリプト上にRの関数やオブジェクトの名前を途中まで入力して「Tab」キーを押すと、コード補完機能が働き、入力候補とその簡易ヘルプが表示されます。いずれかの候補にカーソルをあわせて「F1」キーを押すと、詳細なヘルプも表示されます。

● 補完機能を使って表示させたヘルプ



コードを入力したら上のバーにある「Run」をクリックするか、または「Ctrl」キー（「Command」キー）を押しながら「Enter」キーを押すと、カーソルのあるコードがConsoleにコピーされて実行されます。

▶ 関数抽出

コード範囲を指定すると、これらをまとめて関数として定義する機能があります。コード内に値の定義されていない変数がある場合は、仮引数として「function」関数の引数に挿入してくれます。たとえば、次のようなコードを書きます。

```
tmp <- sample(1:6, x, replace = TRUE)
hist(tmp)
```

単に1から6までの整数から「x」個を復元抽出し、これをヒストグラムにするだけのコードですが、これを範囲指定してペインのバーにあるペンのようなアイコンをクリックし、「Extract Function」を選択します。すると関数名を指定するように促されるので、適当に名前を付けます。これで、範囲指定された箇所が「function」定義に取り込まれます。

```
dice <- function (x) {
  tmp <- sample(1:6, x, replace = TRUE)
  hist(tmp)
}
```

「x」が仮引数として設定されていることに注意してください。

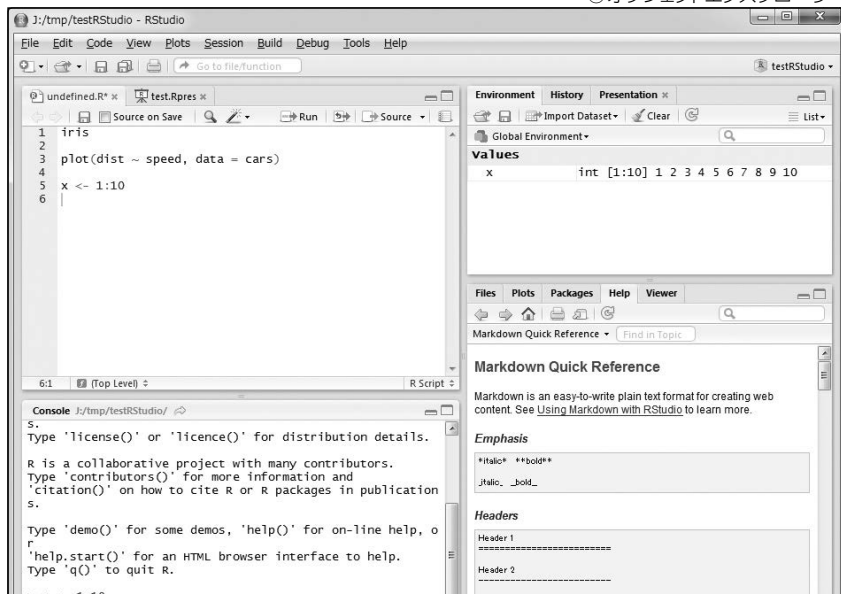
▶ 履歴

実行したコードは右上のペインの「History」タブに履歴として残ります。履歴にあるコードをダブルクリックするとConsoleにペーストされるので、「Enter」キーを押すとただちに実行できます。また、「Shift」キーを押しながら「Enter」キーを押すと、左上のペインに表示されているファイルに挿入することができます。

▶ オブジェクト管理

オブジェクトを生成すると、右上の管理ペインの「Environment」タブにオブジェクト名とデータの概要が表示されます。

● オブジェクトエクスプローラー



デフォルトでは作業スペースに登録されたオブジェクトのみが表示されます。

[Global Environment] ボタンをクリックすると、各環境にあるオブジェクトを確認できます（環境については686ページを参照）。

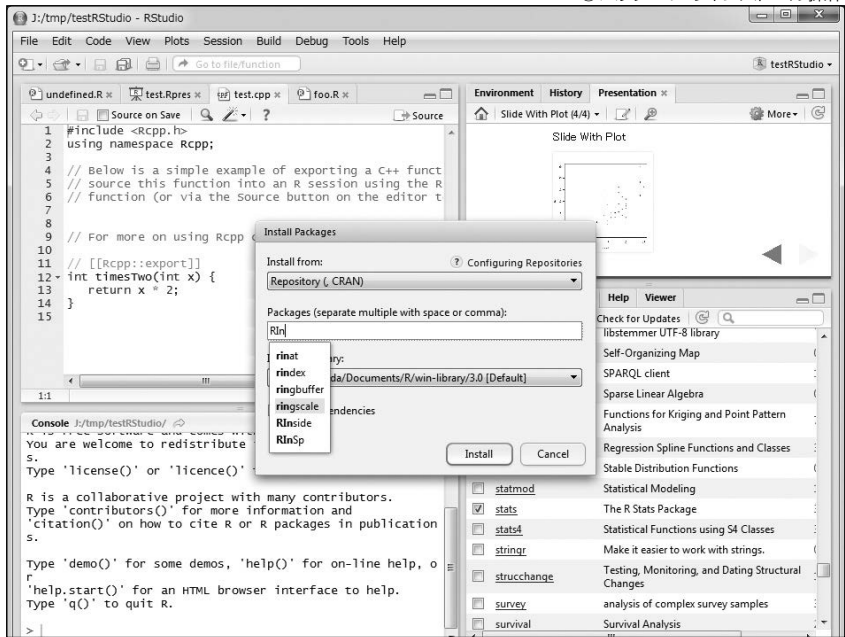
III プロットの作成

プロットを作成するコードの場合、右下のペインの「Plots」タブにプロットが表示されます。このタブの「Export」ボタンをクリックすると、画像をPDF、または任意の画像フォーマットで保存することができます。その際、画像サイズなどの調整が可能です。ただし、日本語を含むプロットをPDFで保存する場合、あらかじめ日本語フォントの設定が必要になります。詳細は598ページを参照してください。

III パッケージ管理

右下のペインにある「Packages」タブではパッケージのインストール、ロード、さらにはアップデートなどを行うことができます。インストール機能でも、パッケージ名を指定する際にコード補完が効きます。

● パッケージのインストール操作



III データの取り込み

RStudio右上のオブジェクト管理ペインからCSVファイルを読み込むことができます。[From Text File]を選択するとダイアログボックスが開くので、ファイルを指定します。RStudioがデータ構造を解釈して適当な形式のデータフレームに変換するので、修正の必要がなければ[Import]ボタンをクリックします。指定された名前のオブジェクトがオブジェクト管理ペインに追加されます。なお、データに日本語が含まれる場合、文字化けなどの問題がないか慎重にチェックしてください。たとえば、列名に日本語を使っているデータファイルを読み込むとすると、ダイアログボックス上ではデータがすべて「undefined」と表示されていることがあります。実際

には正しく読み込まれる可能性が高いので、そのままインポートしてみてください。

テキストファイルの文字コードがOSのデフォルト設定とは異なる場合、あるいはExcel形式のファイルを直接読み込みたい場合は、コードを記述して実行し、取り込みます(378ページ、380ページ、387ページを参照)。

III プレゼンテーションファイルの作成(rmarkdown、knitr)

データ解析ではデータを変えて分析をやり直したり、グラフィックスを作り直したりすることが多いはずです。そこで、コードとレポートを1つのファイルにまとめ、コード部分を一括で実行し、その結果で置き換えることができれば便利です。これを文芸的プログラミングといいます。

RStudioでは3種類の文芸的プログラミングがサポートされています。TeX文書をベースにした「Sweave」、HTMLファイルを生成する「R Markdown」、同じくHTMLファイルをパワーポイント風のスライドに変換する「R Presentation」です。

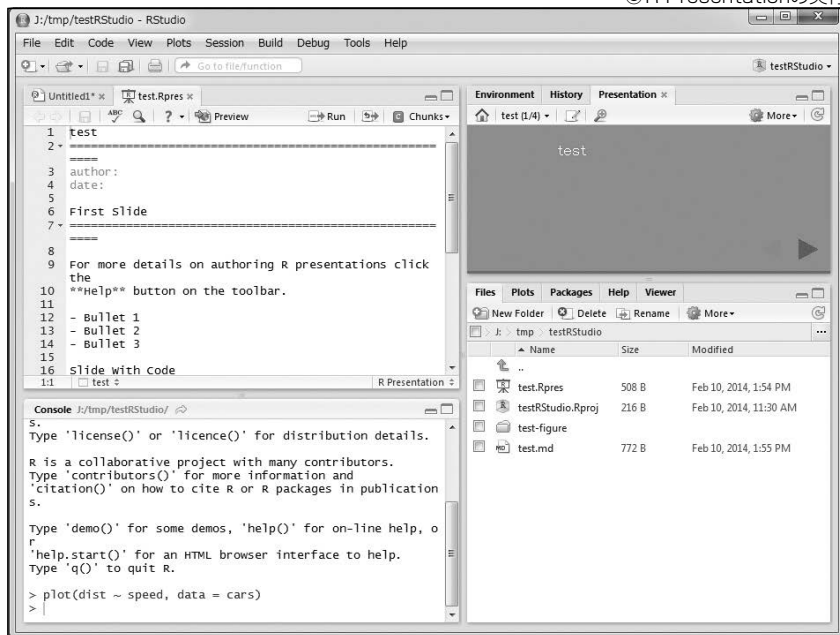
SweaveについてはLaTeXの実行環境が別に必要になります(RStudioでSweaveを利用する方法については『R言語上級ハンドブック』の338ページを参照)。後者のHTML形式の変換ではMarkdownというマークアップ言語でファイルを作成します。この場合、RStudioに「rmakrdown」と「knitr」パッケージが導入されていれば、すぐに使い始めることができます(「rmakrdown」と「knitr」パッケージが導入されていない場合は、RStudioがインストールを促す)。メニューバーの「新規作成」ボタンから[R Markdown]を選択するとHTML形式のファイルを、[R Presentation]を選択するとHTMLでスライドを作成するためのファイルを作成することができます。どちらもMarkdown記法を使ってHTMLファイルを作成するので、基本的な使い方は同じです。

たとえば、[R Markdown]を選択したとすると、サンプルがひな形として記載されたファイルが開きます。このままファイルに適当な名前を付けて保存します。拡張子は自動的に「.Rmd」となるはずです。ここでタブ上部の「Knit HTML」ボタンを押すと「Rmd」ファイルが処理され、新たに現われたウィンドウに処理結果であるHTMLファイルが表示されます。

「Rmd」ファイルにMarkdown形式で指定されたRコードが実行され、HTMLファイルでは、その結果に置き換えられています。グラフィックスを出力する命令があれば、該当の箇所に挿入されます。

「新規作成」ボタンから[R Presentation]を選択すると、拡張子は自動的に「.Rpes」と設定されます。自動生成されるファイルにはサンプルコードが記載されています。また、右上のペインに「Presentaiton」というタブが追加されています。このタブの右下の三角形をクリックするとスライドを遷移させることができるので、試してみてください。あるいはタブの虫眼鏡アイコンをクリックすると別ウィンドウで表示されます。

● R Presentationの実行



4枚程度のスライドですが、これは左上のtest.Rpresファイルに記載されているサンプルコードを変換したものです。

▶ Markdown

プレゼンテーション作成用ファイルではR Markdownと呼ばれる記法でコードやレポートを書き込みます。記法については、メニュー下のバーにある「?」をクリックし、「Markdown Quick Reference」を選択すると、英文ですが、右下のヘルプに代表的な記法のサンプルが表示されるので、参照してみてください(「Rmd」ファイルと「Rpres」ファイルでは記法が少し異なる)。

たとえば、「Rmd」ファイルでは、次の記述はHTMLタグでいうところのヘッダ(H1見出し)になります。

```
Header 1
```

```
=====
```

一方、「Rpres」ファイルでは、これはスライドの区切りになります。

また、見出しの階層は、次のように表されます。

```
Header 2
```

```
-----
```

```
### Header 3
```

```
#### Header 4
```


それぞれHTMLタグでいうところのH2、H3、H4に該当します。

次にRのコードを指定する箇所は次のように書きます。これをコードチャンクといいます。

```
```{r}
summary(cars$dist)
summary(cars$speed)
```
```

ここで「```」はバックチック(バッククォート)と呼ばれる記号を3つ並べています。通常のキーボードだと「Shift」キーを押しながら「@」キーを押して入力します。なお、バックチックは行頭から記述し、前に余計なスペースが挿入されないよう気を付けましょう。

コードチャンクにはさまざまなオプションを指定できます。これは「[r *]」の*の部分にオプション名とその値を「=」でつないで指定します。たとえば、次のように記述します。

```
```{r mychunk1, eval=TRUE, echo=FALSE}
summary(cars)
```
```

チャンクごとに名前を付けることができますが、ここでは「mychunk1」という名前を指定しています。チャンクに名前を付けておくと、HTMLへの変換時にエラーが出た場合に問題の箇所を特定しやすくなります。その後の「eval」はコードチャンク内の式を評価することを、また「echo」はコード部分をHTMLに残すかどうかの指定です。複数のオプションを列記する場合は、「,」(カンマ)で区切ります。

下表に代表的なオプションとそのデフォルト値を示します。詳細はknitr開発者のサイト(<http://yihui.name/knitr/options>, <http://cran.r-project.org/web/packages/knitr/vignettes/knitr-refcard.pdf>)を参照してください。

| オプション | デフォルト値 | 目的 |
|----------------------|--------------------|---|
| eval | TRUE | チャンクを実行する(Rのオプション設定などで使う) |
| echo | TRUE | チャンクそのものを変換後のファイルに残す。なお、「echo=2:3」とするとチャンクの2、3行目だけを実行する |
| results | 'markup/asis/hide' | 出力をマークアップ形式に整形する(markup)、あるいはRの出力そのまま残す(asis)、さもなければ表示しない(hide) |
| warning | TRUE | チャンクコードを実行した際の警告を変換のファイルに残す |
| error | TRUE | チャンクコードを実行した際のエラーを残す |
| comment | '###' | チャンクコードの実行結果の冒頭に付与されるコメント記号を指定する |
| message | TRUE | チャンクコードを実行に伴うメッセージを残す |
| fig.width、fig.height | 7 | グラフィックスのサイズをインチで指定する |
| fig.cap | "" | グラフィックスのキャプションを指定する |
| out.width、out.height | "" | 出力ファイルでのグラフィックスのサイズをピクセルや%で指定する |

なお、デフォルト設定そのものを変えたい場合は、ファイルの最初のチャンクに次のように記述しておきます。

```
```{r echo=FALSE}
opts_chunk$set(warning=FALSE, message=FALSE)
```
```

R Markdownについては、下記のURLに簡便なチートシート(早見表)が用意されています。英文ですが直感的に理解できる構成になっています。非常に便利なので、操作に迷ったら参照するとよいでしょう。

URL <https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf>

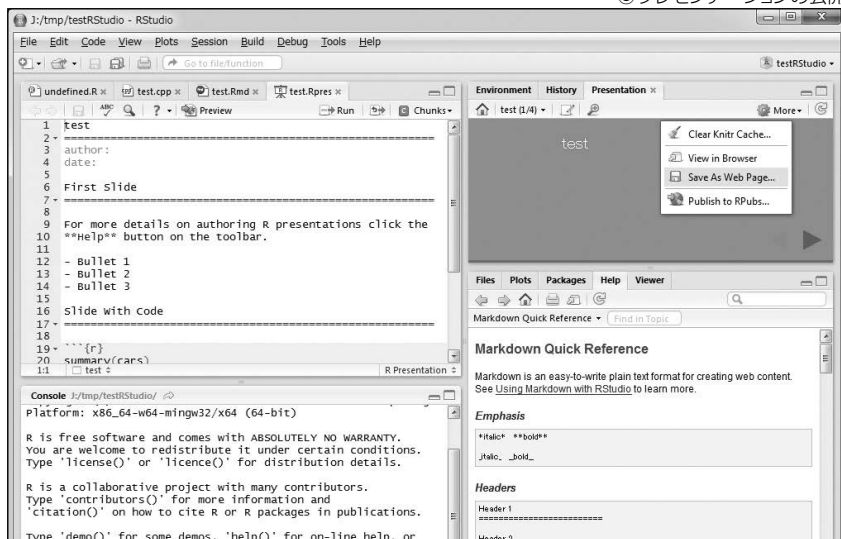
URL <https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf>

▶ プレゼンテーションの公開

「Rmd」ファイルの場合、「Knitr」ボタンをクリックするとすぐにHTMLファイルに変換されます。「R presentation」の場合、右上のペインで「More」をクリックすると、「Clear Knit Cache」「view in Browser」「Save as Web Page」「Publish to RPubS」の選択肢があります。「Save as Web Page」を選択すると、RpresファイルをHTMLに変換したファイルが生成されます。

変換後のHTMLには画像などがすべて含まれており、このファイルとブラウザさえあれば、どこでもプレゼンテーションを表示することができます。ただし、古いブラウザなどではR Presentationの機能を十分に引き出せない場合があるので注意してください。

● プレゼンテーションの公開



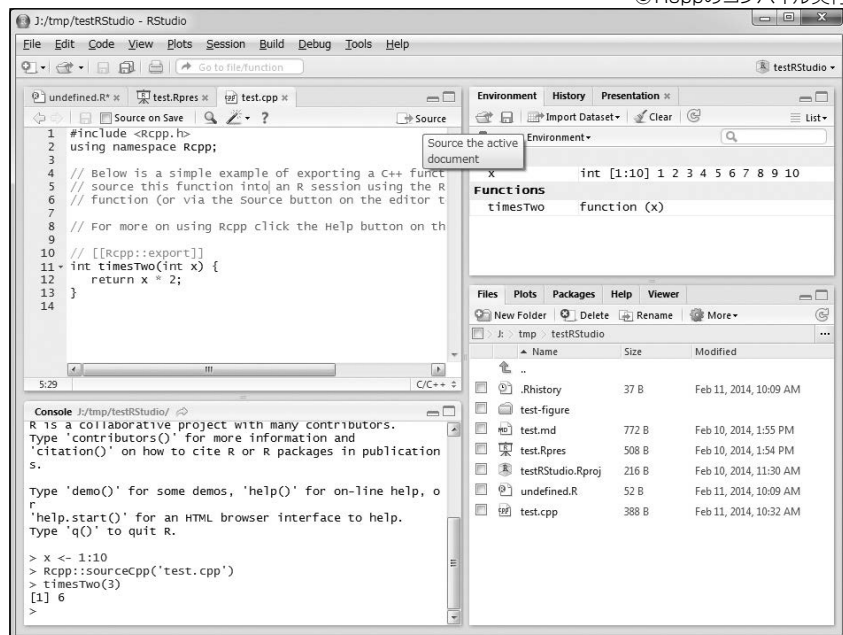
また、いずれの場合も「RPubs」ボタンをクリックすると、「http://rpubs.com/」に自動的に公開してくれます。その際、ユーザー名などの登録(または入力)が必要です。

III Rcppとの連携

RとC++を連携させるRcppパッケージ用のスクリプトをRStudio上で手軽に作成することができます。開発環境が必要ですので、747ページを参考に導入しておきます。

[新規作成]ボタンから[C++ File]を選択し、表示されたファイルを適当な名前で保存しておきます。サンプルコードはこのままコンパイルして実行できます。ペイン右上の「Source」をクリックします。するとRcppパッケージの「sourceCpp」関数にファイル名が渡されてConsoleで実行されます。これはRcppのアトリビュート機能を使っており、C++のソースに書いた関数名をそのままRで実行できるようになります。

● Rcppのコンパイル実行



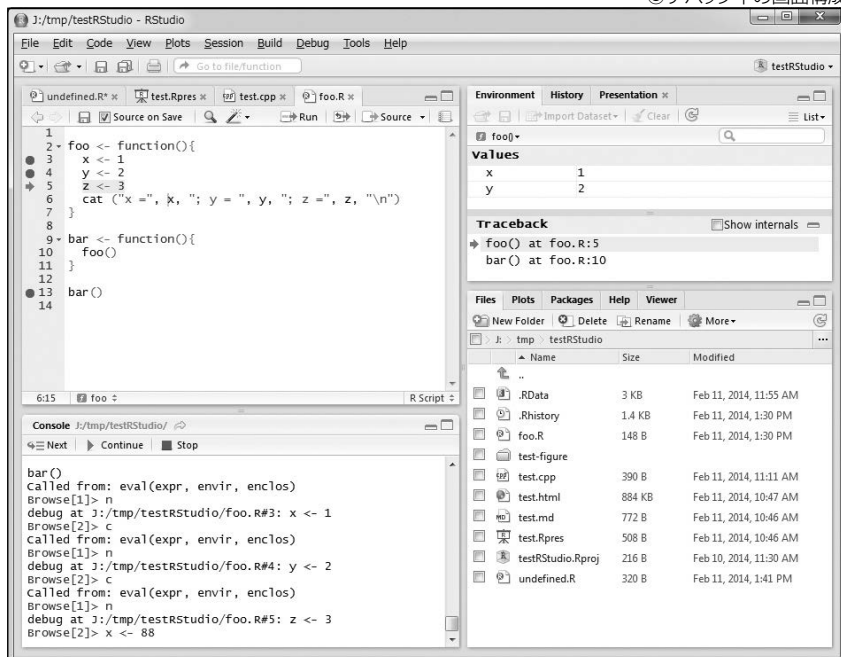
リンクオプションなどを指定する場合はMakefileまたはMakevarsファイルの保存先を指定します。[Build]メニューから[Configure Build Tools]を選択し、設定ダイアログを表示させます。Makefileのひな形はRcppパッケージのインストール先のexamplesフォルダ内にあるOpenMPサンプルなどを参照するとよいでしょう。

III デバッグ機能

RStudioにはデバッグをサポートする機能があります。「debug」関数でデバッグモードを指定した関数は、以降、実行するたびにデバッグモードに入ります。

ブレークポイントを指定して実行する場合、ペイン右の「Source」ボタンをクリックして、そのスクリプトを取り込んでおく必要があります。ブレークポイントを置くにはスクリプトファイルの左端をクリックするか、「Shift」キーを押しながら「F9」キーを押します。

●デバッグ中の画面構成

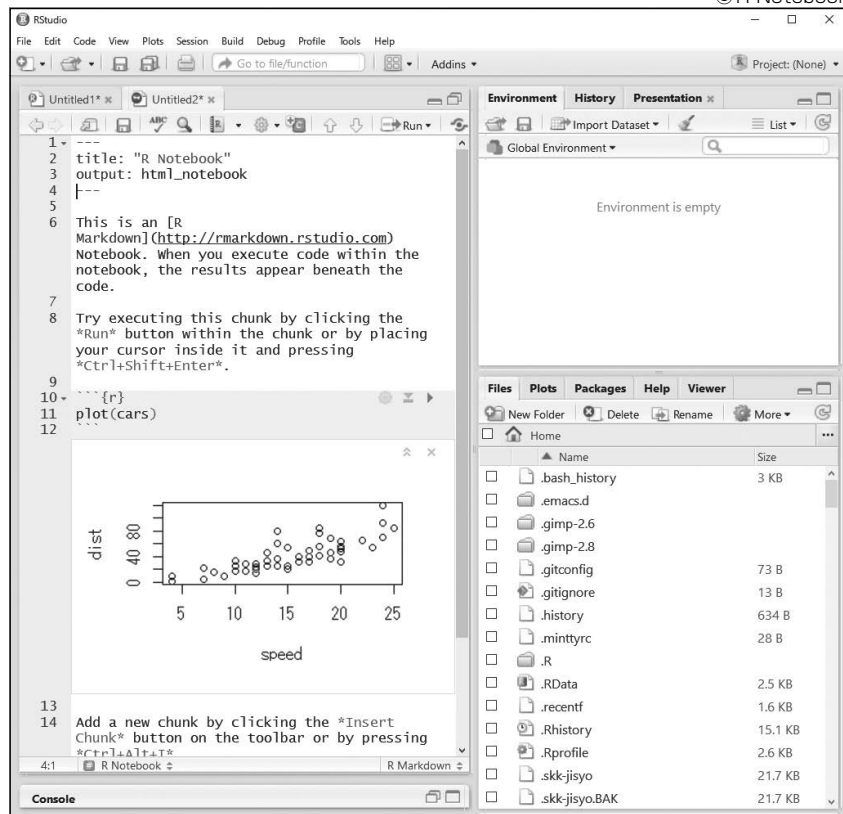


左下のConsole画面ではカーソルが「Browse[]>」に変わっていますが、R本体でデバッグ作業する場合と同じように「n」や「c」のコマンドを入力できます。あるいはタブ上部の「Next」や「Continue」ボタンをクリックして操作を続けることができます。「Next」は現在ハイライトされている行(図の左上で右向きの矢印の行)を実行して停止します。「Continue」は次のブレークポイント直前までのコードを実行して止ります。

また、右上のオブジェクト操作ペインでは環境が関数名(ここでは「foo」)になっており、関数のフレームになっています(フレームについては686ページを参照)。下の「Traceback」は、関数の実行順に積み上がって表示されます(スタック表示)。

III RStudioの機能を調べる

RStudioは機能が豊富であるため、ここで紹介しきれません。また、開発が盛んに進められており、新規機能が次々と追加されています。たとえば、本書の執筆時点(2016年6月)には、R Notebookという機能のプレビュー版が公開されていました。これはHTMLなどに変換する前のRmdファイル内に出力や画像を挿入する機能です。



RStudioの機能については、下記のURLを参照するとよいでしょう。

URL <https://www.rstudio.com/products/rstudio/features/>

幸い、RStudio本体に詳細なヘルプが付随しています。英語ではありますが、図が多用されているので、理解するのは困難ではありません。[Help]メニューから、各種項目にアクセスできます。

また、RStudioを利用する上ではショートカットキーを覚えておくとう便利です。ショートカットキーとは、マウスを使わず、複数のキーを同時に押すことで、特定の機能を実現することを指します。[Tools]メニューから[Keyboard ShortcutsHelp]を選択すると一覧を確認できます。

パッケージのインストール

III パッケージとは

パッケージとは、特定の解析手法やグラフィックス機能を実現することに特化したRのプログラムをまとめたファイルの集合です。基本的なパッケージはR本体とセットでインストールされています。

R本体とセットでインストールされる基本パッケージだけでも、非常に多くの解析技法やグラフィックス作成機能が利用できます。しかし、Rの機能をさらに拡張したり、専門分野別の解析関数、3Dグラフィックスなどの機能を備えたパッケージが、世界中のRユーザーによって開発されています。これらのパッケージはCRANに登録されており、自由にダウンロードして利用することができます。CRANから、必要な機能を備えたパッケージを検索するには、次のサイトを利用すると便利です。

- It's crantastic!

URL <http://crantastic.org/>

CRANに登録されたパッケージは、Rのメニューや関数を使ってダウンロードすることができます。CRAN以外にもRのパッケージを公開しているサイトは、個人のサイトを含め、多数あります。R-Forgeはその代表的なサイトです。

- R-Forge

URL <http://r-forge.r-project.org/>

▶ GitHub

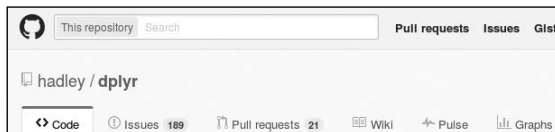
最近では開発がGithubで進められ安定版がCRANで公開されるパッケージや、そもそもCRANに公開されないパッケージも増えてきました。

Githubに公開されているパッケージは次の手順でインストールするのが簡単です。

- ① まず、次のようにして「devtools」パッケージをインストールします。

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

- ② 次に、Githubのユーザー名とレポジトリ名を確認します。たとえば、開発版の「dplyr」パッケージは以下のサイトになります。



- ③ 最後に、次のように「devtools」の「install_github()」を使います。

```
> devtools::install_github("hadley/dplyr")
```

III パッケージのインストール方法

パッケージをインストールするには、次のようにします。

▶ Windowsでのパッケージのインストール(ネットワーク経由)

メニューの「[パッケージ]」→「[パッケージのインストール]」を選択します。ダウンロード先のサイトを尋ねられるので、「Japan(Tokyo)」などを選択(48ページの「options」関数で「reposオプション」を指定した場合は自動的に選択されている)すると、パッケージの一覧が表示されるので、必要なパッケージを選択します。

▶ Macでのパッケージのインストール(ネットワーク経由)

メニューの「[パッケージとデータ]」→「[パッケージインストール]」を選択し、表示されるダイアログボックスで「CRAN(バイナリ)」を選択して「一覧を取得」ボタンをクリックします。ダウンロード先のサイトを尋ねられるので、「Japan(Tokyo)」などを選択すると、パッケージの一覧が表示されます。インストールするパッケージを選んでから、「選択をインストール」ボタンをクリックするとインストールが開始されます。

場所はデフォルト(/Library/Frameworks/R.framework/Versions/3.*/Resources/library)のままでよいでしょう。

▶ Linuxでのパッケージのインストール(ネットワーク経由)

Ubuntuの場合、リポジトリに「r-cran-」で始まるパッケージが用意されていることがあるので、Synapticで検索するか、あるいは「apt-cache search r-cran」などを実行して検索します。

リポジトリが存在する場合は、「sudo apt-get install r-cran-***」でインストールできます。

用意されているパッケージはUbuntuのバージョンごとに異なります。詳細はUbuntuのパッケージ・サイト「<http://packages.ubuntu.com/>」からUbuntuのバージョン(コード)別に「GNU R」パッケージ・リストを確認してください(たとえば、<http://packages.ubuntu.com/trusty/gnu-r/>)。また、「数学」パッケージにもRパッケージが含まれています(たとえば、<http://packages.ubuntu.com/trusty/math/>)。

次のコマンドはUbuntuにリポジトリからRパッケージをインストールする例です。ターミナルを起動して実行します。ここでは3Dグラフィックスをサポートする「rgl」パッケージ(「r-cran-rgl」)をインストールしています。

```
$ sudo apt-get install r-cran-rgl
```

III すべてのプラットフォームで共通の方法(ネットワーク経由)

パッケージのインストールは、Rのコンソールから行うこともできます。たとえば、次のようにダウンロード先のCRANサイトとパッケージ名を指定して実行します。Rを管理者権限で実行していない場合は、ユーザーのホームに用意されたパッケージ用フォルダにインストールが行われます。

```
> options (repos = "https://cran.ism.ac.jp")
> install.packages ("パッケージの名称")
```

最初にCRANのミラーサイトを指定します。「options」関数による環境設定はセッション(Rを起動してから終了するまで)の間だけ有効です。環境設定ファイルを用意すると、起動時のデフォルトとすることができます(45ページ参照)。

また、インストールしようとするパッケージが別のパッケージのインストールを前提としているため、エラーや警告が表示されることがあります。このような依存関係を自動的に解決するには、次のように実行します。

```
> install.packages ("パッケージの名称", dependencies = TRUE)
```

Linux版で「install.packages」を実行すると、デフォルトではソースファイルからインストールします。また前述のように、Ubuntuなどでは一部のパッケージについては、ターミナルから「apt-get install」を実行することで追加できます。

▶ 全パッケージの一括インストール

CRANに登録されている全パッケージをまとめてインストールするには、次のようにします。ただし、相応に時間がかかるので注意してください。

```
> pkgs <- available.packages() # 全パッケージのリストを取得
> install.packages (pkgs [, 1]) # 1列目がパッケージ名
```

III パッケージのインストール先

パッケージのインストール先は、Rを管理者権限で実行したか、ユーザー権限で実行したかによって異なります。

ユーザー権限で実行した場合は、最初の実行時には個人領域にパッケージ専用のフォルダが新規に作成されます。パッケージ保存用のフォルダは、Windows 10/8/7であれば、「C:\Users\¥username¥Documents」に作成されます。Macでは「/Library/Framework/R.framework/Versions/3.3/Resources/library」となります。

ただし、環境変数「HOME」が設定してある場合、たとえばWindowsだと、HOMEで設定されたフォルダの下に「R¥win-library¥3.3」というフォルダが生成され、以降、このフォルダがデフォルトとなります。ライブラリのパスはRで「.libPaths()」を実行すると確認できます。筆者の場合は次のように表示されます。なお、Rでは、フォルダの区切りの「¥」は「/」で表現されるので注意してください。

```
> .libPaths ()
[1] "D:/home/R/win-library/3.3"
[2] "C:/Program Files/R/R-3.3.0/library"
```

すなわち、デフォルトでは[1]の場所が利用されます。パッケージをインストールしたり、読み込む際には、この順番で検索と読み込みが行われます。

管理者権限でRを実行している場合は、次のように「lib」引数にインストール先を文字列で指定すると、「C:/Program Files/R/R-3.3.0/library」にインストールすることができます。あるいは「.libPaths」関数の出力を保存しておいて、次のように利用します。


```
> slibs <- .libPaths ()
> install.packages ("パッケージの名称", lib = slibs [2])
```

なお、Ubuntuでは「apt-get」コマンドでインストールしたパッケージは「/usr/lib/R」以下に置かれます。

III ダウンロードしたパッケージのインストール

あらかじめCRANからパッケージファイルをダウンロードしておいて、これをRのコンソールや、OSのコマンドプロンプトやターミナルからインストールすることも可能です。CRANからダウンロードしたパッケージファイルをインストールするには、次のコマンドを実行します。「repos=NULL」はCRANへアクセスしないことを指定します。

```
> install.packages ("ダウンロードファイルのパス", repos = NULL)
```

また、Windowsであれば「zip」形式のパッケージファイルをダウンロードしてから、メニューの[パッケージ]→[ローカルにあるzipファイルからのパッケージのインストール]を選択します。

Macでは、「tgz」形式のファイルをダウンロードしてから、メニューの[パッケージとデータ]→[パッケージインストーラ]を選択して表示されるダイアログボックスで、[このコンピュータ上のバイナリパッケージ]を選択して、インストールするファイルを指定します。

III OSのコマンドプロンプトからのインストール

Windowsを例に挙げます。ファイルの保存先に移動して次のコマンドを実行します。ここではWindowsのコマンドプロンプトを起動して、「rgl_0.93.991.zip」をインストールしています。なお、コマンドプロンプトからパッケージをインストールするには、環境変数「PATH」にRの実行ファイルの位置が指定されている必要があります。

```
> cd /d C:\Users\¥ishida¥Downloads
> C:\Users\¥ishida¥Downloads> R CMD INSTALL rgl_0.93.991.zip
```

▶ Windowsの環境変数の設定

環境変数とは、利用しているPCにアプリケーションの保存場所や、ユーザーごとのフォルダ位置などを指定する仕組みです。

Windowsの場合、コントロールパネルから「システムとセキュリティ」→「システム」→「システムの詳細設定」の順にクリックすると、「システムのプロパティ」ダイアログボックスが表示されます。ここで「詳細設定」タブにある「環境変数(N)」ボタンをクリックし、上部の「ユーザー環境設定」で「HOME」などの環境変数を「D¥¥home」のように指定します。

また、下部の「システム環境設定」で「PATH」などを指定します。「PATH」にRがインストールされたフォルダを指定しておくコマンドプロンプトでRを起動できます。「PATH」を選択して[編集(I)]ボタンをクリックすると、次ページの上図のようなウィンドウが表示されるので、Rの実行ファイルの位置を「C:\¥Program Files¥R¥R-3.3.0¥bin」のように「bin」まで指定します(バージョン番号は適宜、変更する)。

● 環境変数の設定

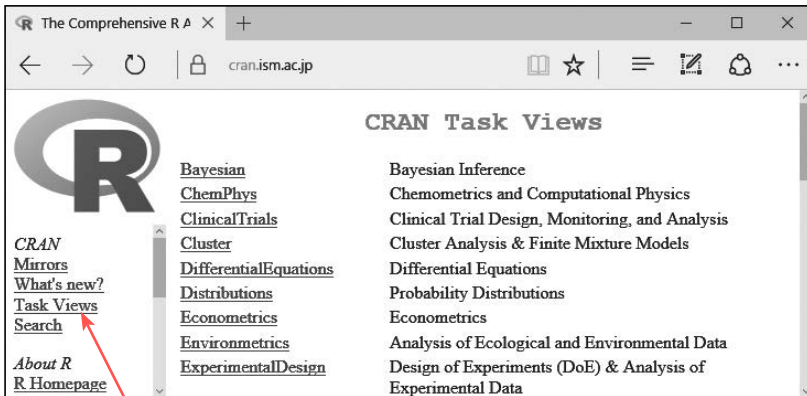


ただし、設定を間違えたり、既存の内容を誤って消してしまったりと、Windowsの動作に大きな影響を与えてしまうため、操作は慎重に行う必要があります。

III パッケージをテーマ(CRAN Task Views)ごとにインストール

CRANには膨大なパッケージが登録されています。これらの中から、テーマあるいはタスクごとにパッケージをまとめたものに「Task Views」があります。

CRANのサイトでは左メニューに「Task Views」というリンクがあります。



「Task Views」を
クリック

「Task Views」のリンクをクリックすると、右フレームにテーマの一覧が表示されます。これを利用すると、関連するパッケージをまとめてインストールすることができます。

たとえば、自然言語処理関係(Natural Language Processing)のパッケージをまとめてインストールするには、次のように実行します。

最初に「ctv」パッケージをインストールします。

```
> install.packages ("ctv")
> library ("ctv")
> install.views ("NaturalLanguageProcessing")
```

アップデートは次のように実行します。

```
> update.views ("NaturalLanguageProcessing")
```

III パッケージのロード

パッケージはWindowsであれば[パッケージ]メニューから[パッケージの読み込み]を選択してロードします。Macであれば[パッケージとデータ]メニューから[パッケージマネージャ]でロードするパッケージをONにします。あるいはコンソールから次のコマンドを実行します。

```
> library ("パッケージの名称")
```

「ロード」とは、パッケージが利用できるようにセッションに読み込むことです。「require ("パッケージの名称")」としても読み込まれますが、「require」関数は、あるパッケージが内部から別のパッケージを読み込む場合などに利用するのが適切です。

III インストールされたパッケージの確認

インストールされたパッケージを確認するには、コンソールで次のように実行します。出力は長くなるので、注意してください。

```
> installed.packages ()  
... 以下略 ...
```

セッションでロードしたライブラリを確認するには、次のように実行します。「.packages」関数は内部で特殊な関数を利用しており、出力をそのまま表示することはありません。表示させたい場合は、次のように全体を括弧で囲みます。

```
> (.packages ())  
[1] "lattice"   "stats"     "graphics"  "grDevices" "utils"     "datasets"  
[7] "methods"  "base"
```

以上の出力は、実行時の状況によって変わります。

III パッケージのアップデート

R本体が更新されると、多くの場合これにあわせてパッケージのアップデートも行われます。RのパッケージはR本体のバージョンに依存するものが多いので、基本的には、再度インストールし直した方がよいでしょう。

Rのメジャーアップデート(たとえば3.2から3.3へのバージョンアップ)では、新たにパッケージ用フォルダが作成されるので、改めてすべてのパッケージをインストールする必要があります。

どうしても古いパッケージ環境をそのまま利用したい場合は、古いバージョンのRで作成されたパッケージのフォルダの中身を、新しいパッケージのフォルダ(「C:\Program Files\R\R-3.3.0\library」など)にそっくりコピーしてしまうか、あるいは環境設定でパッケージの読み込み先として古いパッケージ用フォルダを指定します。その上でパッケージの更新が必要になれば、新しいRのコンソールから次のコマンドを実行しておきます。

```
> update.packages (checkBuilt = TRUE, ask = FALSE)
```

R本体のバージョンアップとは別に、個別にパッケージをアップデートする必要がある場合は、Windowsの場合は[パッケージ]メニューから[パッケージの更新]を選択することもできます。あるいはコンソールから次のコマンドを実行します。

```
> update.packages (ask = "graphics")
```

上記のコマンドを実行すると、更新対象とするパッケージの一覧が表示されるので、確認の上、[OK]ボタンをクリックします。

URLを指定したファイルのダウンロード

Rを使ってファイルをダウンロードすることができます。「download.file」関数にURLを指定します。

```
> download.file("http://rmecab.jp/data.R", destfile = "data.R")
```

「URL」引数に指定できるのは、「https://」「http://」「ftp://」「file://」などのプロトコルで始まるアドレスです。

COLUMN ::演算子によるパッケージのアクセス

パッケージのオブジェクトは「library」でロードしなくとも読むことができます。これには演算子「::」か演算子「:::」を使います。たとえば、「MASS」パッケージには「beav1」というビーバーの体温のデータがありますが、「MASS」パッケージをロードしないと通常はアクセスはできません。ところが「MASS::beav1」とするとアクセスすることができます。

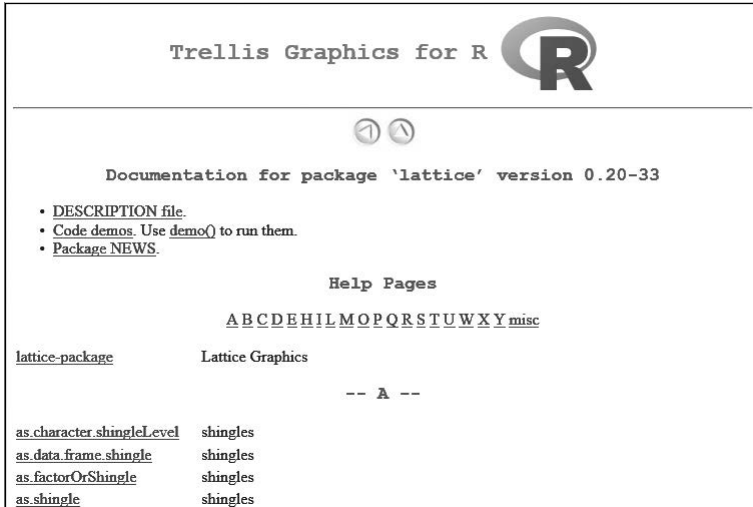
ただし、::演算子ではアクセスできないオブジェクトもあります。こうしたオブジェクトはパッケージの作者が外部に隠蔽することを意図しているものですが、::演算子の代わりに:::演算子を使うとアクセスできます。この仕組みを「名前空間」といいます(710ページ参照)。

ヘルプの利用

III ヘルプの表示

オブジェクトの意味や使い方を知るには、関数名の頭に「?」を付けるか、「help」関数に引数として指定して実行します。

Windowsの場合、デフォルトでは下図のようにブラウザが起動してヘルプが表示されます。関数名などがリンクになっており、クリックすると別のヘルプページが表示されるようになってます。



なお、中央上にある三角形をクリックすると、ヘルプファイル全体のトップページに移動できます。その隣の左向きのアイコンをクリックすると、インストール済みパッケージの一覧とヘルプページへのリンクが表示されます。

MacやLinuxでは、別ウィンドウやコンソールにヘルプが表示されます。Windowsでも、下記で説明するようにhelp_typeオプションに"text"を指定すると、ヘルプの内容が別ウィンドウに表示されます。

```
> # もっとも基本的な実行方法
> ?chisq.test
> # 上と同じ結果
> help ("chisq.test")
> # パッケージの情報をみる
> help (package = "lattice")
```

パッケージ 'lattice' の情報

記述:



```

Package:      lattice
Version:      0.20-27
Date:         2014/02/27
Priority:      recommended
Title:        Lattice Graphics
Author:       Deepayan Sarkar <deepayan.sarkar@r-project.org>
Maintainer:   Deepayan Sarkar <deepayan.sarkar@r-project.org>
Description:   Lattice is a powerful and elegant high-level data
               visualization system, with an emphasis on
               multivariate data, that is sufficient for typical
               graphics needs, and is also flexible enough to
               handle most nonstandard requirements. See ?lattice
               for an introduction.

Depends:      R (>= 2.15.1)
Suggests:     KernSmooth, MASS
Imports:      grid, grDevices, graphics, stats, utils
Enhances:     chron
LazyLoad:     yes
LazyData:     yes
License:      GPL (>= 2)
URL:          http://lattice.r-forge.r-project.org/
Packaged:     2014-02-27 06:16:44 UTC; deepayan
NeedsCompilation: yes
Repository:   CRAN
Date/Publication: 2014-02-27 09:13:27
Built:        R 3.0.3; x86_64-unknown-linux-gnu; 2014-03-08
               23:54:01 UTC; unix

```

... 以下略 ...

「help」関数を使ったパッケージの情報の表示

「package」引数を利用すると、指定されたパッケージの情報が表示されます。そのパッケージがあらかじめロードされている必要はありません。

なお、パッケージのヘルプは「library (help = "lattice")」としても表示されます。

ヘルプの表示方法の変更

ヘルプを表示する方法は「getOption」関数で確認でき、また変更することができます。Rの環境設定には「help_type」という項目があり、ヘルプを参照しようすると、環境設定に従ってヘルプが表示されます。この設定を変更すると、ヘルプの表示形式を変更できます。

```

> # 現在のヘルプの表示方法を確認
> getOption ("help_type")
[1] "html"

```

```
> # 出力方法を指定してヘルプを参照
> help ("chisq.test", help_type = "text")
> # あるいはヘルプの表示方法を変更
> options (help_type = "text")
```

Rの標準設定は「options」関数で確認できますが、この関数を引数なしで実行すると、直接関係のない項目まで数多く表示されてしまいます。そこで「getOption」関数に引数として参照したいオプション、ここでは「help_type」を文字列として指定するのが便利です。なお、「getOption ("help_type")」は、「options ()\$help_type」や「options ("help_type")」としても同じです。

III ヘルプの表示形式を「help」関数実行時に指定

「help_type」引数をいちいち指定するのが面倒な場合は、Rのセッションの最初に「options」関数で「help_type」の設定を変更します。この指定はセッションの間ずっと、つまりRを終了させるまで有効です。Rを起動するたびに毎回実行するのが面倒であれば、環境設定ファイルに指定します(45ページ参照)。

III 「help_type」引数に指定可能なファイルの種類

Rに組み込まれているヘルプファイルには、さまざまなフォーマットが用意されています。指定できる形式は「text」「html」「postscript」「ps」「pdf」です。

なお、「html」を指定した場合はブラウザが起動します。利用されるブラウザは「getOption ("browser")」で確認できます。

III ヘルプファイルの検索

Rおよびパッケージに用意されているヘルプファイルから、「help.search」関数を使ってキーワードを検索することができます。

キーワードは単語あるいは単語の綴りの一部を指定します。たとえばキーワードとして「chi」を指定すると、単語の中にchiを含むキーワードがすべて抽出されます。

▶ 正規表現によるキーワード検索

「help.search」関数では、正規表現による検索が可能です(150ページ参照)。

「help.search ("^chi")」は「chi」で始まるキーワードを含むヘルプファイルの一覧を表示します。ここで「^」は行頭を意味する正規表現です。これは「\\b」を使って「help.search ("\\bchi")」と実行しても同じ結果になります。

```
> help.search ("^chi")
Help files with alias or concept or title matching '^chi' using
regular expression matching:
```

| | |
|-----------------------|---|
| Hmisc::biVar | Bivariate Summaries Computed Separately by a Series of Predictors |
| datasets::ChickWeight | Weight versus age of chicks on different diets |
| datasets::chickwts | Chicken Weights by Feed Type |
| grid::grid.grob | Create a Grid Graphical Object |
| stats::chisq.test | Pearson's Chi-squared Test for Count Data |
| stats::Chisquare | The (non-central) Chi-Squared Distribution |

Type 'PKG::FOO' to inspect entries 'PKG::FOO', or 'TYPE?PKG::FOO' for entries like 'PKG::FOO-TYPE'.

▶ 「fields」引数の指定

第2引数「fields」にはデフォルトで「c ("alias", "concept", "title")」が指定されています。これはヘルプファイル固有の項目設定です。Rのヘルプファイルは拡張子が「.Rd」で終わるTeXに類似したフォーマットなどで生成されます(707ページ参照)。

たとえば、「chisq.test」関数であれば、「chisq.test.Rd」というファイルからヘルプファイルが自動生成されています。もとの「chisq.test.Rd」ファイルの冒頭部分には、次のように記載されています。

```

¥name{chisq.test}
¥alias{chisq.test}
¥concept{goodness-of-fit}
¥title{Pearson's Chi-squared Test for Count Data}
¥description{
  ¥code{chisq.test} performs chi-squared contingency table tests
    and goodness-of-fit tests.
}
```

「help.search」関数で第2引数「fields」に「alias」を指定すると、上の「¥alias{chisq.test}」に記載された文字列を検索することになります。

III 標準で起動するブラウザの変更

HTML形式のヘルプファイルを表示する場合、デスクトップ環境のデフォルトのブラウザが起動します。ブラウザの種類を変更したい場合は「options」関数で引数「browser」に、ブラウザの実行ファイル名(とその絶対パス)を文字列として指定します。

```

> # Windows版Rの場合、標準ではNULLが出力されるが、デスクトップの標準設定のブラウザが利用される
> getOption ("browser")
NULL
> options (browser = "C:/Program Files/Mozilla Firefox/firefox")
```