

新・数理/工学
ライブラリ [情報工学 = 2]

Rプログラミング マニュアル[第2版]

—Rバージョン3対応—

間瀬 茂 著

数理工学社

第 2 版まえがき

As for elegance, R is refined, tasteful, and beautiful.
When I grow up, I want to marry R.

– Andy Bunn, R-help (May 2005)

It might surprise many R-help posters, but R has manuals
as well ...

– Uwe Ligges, R-help (January 2006)



プログラミングマニュアルの初版が出版された 2007 年当時、R のバージョンは 2.4, 2.5 であった。2013 年 4 月*¹に R は記念すべきバージョン 3 台に到達した。この間の R の進歩、普及、成熟は真に目覚しかった。今や統計において R はほとんど唯一無二の地位を占めている。R 開発者の当初の目標であった「全ての統計的手法のワークベンチ」は間違いなく既に達成済みである。日本をとってみても、いわゆる R 本の数は訳本を含めて 100 を超えており、従来の統計本のスタイルを一新している。学生自身のパソコンを用いた本格的なデータと高度な統計手法を用いたデータ解析の手軽な実践という点で、統計教育も大きく変えている。何よりも統計の専門家がこぞって使い始めており、若い研究者が最初に使うシステムの地位を獲得している。

一般企業での利用も*²、オープンソースのシステムであることが逆に利用の足かせになっていた当初とは異なり、珍しくなくなっているようである。Google 等の巨大データを所持する企業は、むしろ R の積極的利用を後押ししている。

こうした R の現況は、R が常に手本としてきた S システムの革新性に加え、統計家とユーザが使いやすい統計システムを追求してきた結果であり、誰でも使用でき開発にも参加できるというオープンソースシステムならではの利点の賜物である。逆に、日進月歩の R をフォローするのは容易ではなく、著者もバージョン 3.0 の登場を機に現況の精査を試み、システムが大きく変貌していることに気づいた。多くの新機能の追加と、既存の機能の様々な改良・変更*³は想像以上であった。CRAN に登録されているアドオンパッケージ

*¹ ニュージーランドのオークランド大学の統計の講義用ソフトとして産声を上げた R が、わずか千行程度のプログラムとしてささやかにネットに公開された 1993 年から数えて、丁度 20 年が経過している。

*² 公式 FAQ R FAQ ([9]) では R の商用、コンサルティング利用は自由と明記されている。R の既存コードを組み込んだ商品の販売については GNU GPL の観点から原則ソースコードの公開が義務づけられるが、REvolution R のように付加機能を加えた商用版も幾つか登場している。

*³ R の基本パッケージ中には総計 1,264(内部関数は 488) 個のオブジェクトがある。CRAN から入手

ジ数も 2013 年後半に 5,000 を超え、CRAN 未登録分を含めれば 6,000 を超えると思われる。

主な変更点としては以下が挙げられる：

- **国際化の徹底** 中間栄治氏の R の日本語化作業は、その後 R の国際化につながり、今や R は世界の数多くの言語で使用可能になっており、R 自体にもこうした国際化を支援する機能が加わっている。フリーなシステムであることに加え、言語の壁を取り払うことにより、統計処理の南北問題が二重の意味で解決された。
- **巨大データ** 従来のベクトルの長さの「先天的制限」である $2^{31} - 1$ の壁 (正整数の最大値) を超えることを可能にする “long vectors” 機構の登場。
- **速度向上** 手軽な速度向上の工夫。切り札として、バイトコードコンパイル機能と、マルチコア CPU を持つ計算機の普及を背景にした並列処理機能が R の推奨パッケージとして新たに加わった。実際 R およびパッケージ中の関数はほとんどバイトコンパイル済みである。
- **バイト (raw) 型データ** バイト型データが標準で使えるようになり、それを支援する多くの関数が加わっている。
- **S4 化の推進** 完全なオブジェクト指向機構を備えた S4 (S 言語仕様第 4 版) オブジェクトのサポートが一層充実してきた。
- **開発・サポート・デバッグ体制** R Foundation を中心とする、全世界の多数の統計専門家と熱狂的ユーザからなる支援体制は、進歩と全体的な安定との絶妙なバランス*4をもたらししている。
- **多数の洗練された有益なアドオンパッケージの登場** これらは個人のみならず、コミュニティで開発されているものもある。そうしたパッケージは R を側面から補強している。一部もしくは全体が R の推奨パッケージに取り入れられたものもある。
- **多くの関数・機能の追加** 既存の関数にも新しいオプションが追加され、対応ヘルプ文章および公式マニュアルも相当改訂されている。

こうした変更にも関わらず、従来のような使い方でも支障が起きることはほとんどなく、この本の初版は依然手引きとして有効である。しかしながら、初版の目標であった中級程度のユーザに R のプログラミング機能をできる限り網羅的に紹介する、という趣旨からは、こうした新しい R に触れないまま放置しておくことは許されない、というのが

できるバージョン 3.0.0 の基本、標準そして推奨パッケージのヘルプ文章を網羅した 3,604 頁に及ぶ PDF ファイル [R-fullrefman.pdf](#) がある。こうした目録が必要なほど R システムは巨大になっている。R に関しては「群盲象を撫でる」、「車輪を二度発明する」という故事成語が決して冗談ではない。

*4 R システムの成功はある意味 Linux システムの成功に比肩できるが、開発システムの観点からは相当異なる。Core チームやパッケージ作者を含む R の開発者はピラミッド化されておらず、多分に水平的な関係しか持たず、専門開発者はいない。

今回改訂を志した理由である。

今回の第2版も原則 R のプログラミング機能^{*5}に的を絞っている^{*6}。初版の読者の中には、プログラミングマニュアルと称しながら記述がいわゆる計算機言語のそれとは大きく違うことに戸惑いを覚えた方がいらっしやるようである。R プログラミングの理想は「できれば1命令、さもなければせいぜい1行で」ということにあり、実際それが十分可能なように設計されている。この点ではいわゆるスクリプト言語にあい通ずるものがある。統計解析は試行錯誤の連続からなり、いちいち長いコードを書いて思考中断しているわけにはいかないという、統計解析システム固有の事情が背景にある。更に、多くの統計ユーザにとってプログラミングの手間はなければならないほど良いのであろう。

今回の改訂版の体裁と章立ては初版のそれを基本的に踏襲し、多くの章は変更点の追加にとどめているが、章の構成を変更したものもある。バイト (raw) 型データ、バイトコードコンパイラ、マルチコア CPU 利用の並列処理については特に新しい章を設ける価値があると考えた。初版が目指した何でもかんでも紹介するというスタイルも踏襲しているが、完璧を期すのは困難であり、また一般ユーザが使うことを想定していない関数も多く無駄でもある。内容のソースは第一に各オブジェクトのヘルプ文章と公式マニュアルである。ここに書けなかった内容も多い。是非、必要に応じて参照して欲しい。

初版では R でのプログラミングスタイルに不慣れな読者がまだ多いという視点から Tips を多く載せたが、これらの中には既に常識化しているものも多く、またインターネット上には従来 R の公式メーリングリストでしか得られなかった詳しい情報が集められたサイトが幾つも存在しているという現状から、最小限度に留めることとした。

この本の初版は初心者向きでないと序文で特に断ったにも関わらず、多くの R ユーザから類書のない参考書として高い評価を得てきた。今回の改訂版も同様の評価が得られれば幸いです。


2014年3月

間瀬 茂

^{*5} 現在日本では「R 言語」という呼び方が定着しつつあるが、著者は二重の意味でこれは好ましくないと考えている。何よりも R は統計システム・環境であり、言語機能は即統計解析機能ではなく、単に計算機言語と捉えるのは矮小化していると思うからである。更に R の言語機能は S 言語のフリーな移植 (エンジン) であり、開発者達自身が R の言語機能は S 言語であると明記している。例えば、公式マニュアル *An Introduction to R* ([1]) には “a well developed, simple and effective programming language (called ‘S’) which includes conditionals, loops, user defined recursive functions and input and output facilities. (Indeed most of the system supplied functions are themselves written in the S language.)”, また公式 FAQ 集 *R FAQ* ([9]) には “We can regard S as a language with three current implementations or “engines”, the “old S engine” (S version 3; S-PLUS 3.x and 4.x), the “new S engine” (S version 4; S-PLUS 5.x and above), and R. Given this understanding, asking for “the differences between R and S” really amounts to asking for the specifics of the R implementation of the S language, i.e., the difference between the R and S engines.” と書かれている。

^{*6} 統計機能については [20] 参照。

初版まえがき

 は統計解析言語・システムであり、統計解析用に開発された S 言語・環境のフリーな移植である。現在の R は、S-PLUS 等の S 言語の商用版と比較しても遜色のない水準に達している。遠からず R は統計的手法の共通インフラの地位を占めることは間違いない。実際、日本でも既に R を前提とした出版が 20 冊を越えようとしており、全世界でもブームといえるほど R 使用を前提とした出版が相次いでいる。

このマニュアルの目的は、R の計算機言語としての豊富な機能を、参考コードとその出力例を逐一加えて、紹介することである。全くの初心者というよりは、R を一通り使いこなせる方を読者として想定している。構成は諸機能の索引になることを目指しており、決して系統的ではない。R は通常の計算機言語では考えられないほど多様な関数を最初から備えている。この特徴は、R の本来の機能である対話的統計データ解析処理を容易にするためであり、簡潔なコードを書くことを可能にし、プログラム開発効率が極めて高い。逆に、一般ユーザにはその全貌を把握するのは至難の技で、しばしば「車輪を 2 度発明する」無駄を強いられることになる。著者の経験では、こうした機能があれば便利なのにと感じたことが、実際は既に存在していたことが頻繁にある。また、これまで気づかなかった便利な関数・機能を発見して驚くことが今でも多い。

このマニュアルの内容、特に Tips は、著者を含むしばしば匿名の R ユーザが、R 情報サイト RjpWiki に投稿した記事を基礎材料としている（一部勝手に引用・改編していることをお許し願いたい。無論、最終的な内容の正しさの責任は著者にある）。その意味で、「The R Book」([14]) 同様 RjpWiki の産物といえる。この場を借りて、管理者の岡田昌史氏を含め、RjpWiki を支えている質問者を含む皆さんにお礼を申し上げたい。これらは更に、全世界のユーザと R 開発チームによるメーリングリスト記事・マニュアル・書籍を参考にしていることも多い。R という素晴らしいシステムを作り、改良発展されている R 開発チームとともに、それを陰に陽に支えている全世界の R ユーザにも同時にお礼を申し上げたい。また、著者の研究室の修士院生の千田敏君からは、原稿に対する多くの意見と訂正の指摘を頂いたことを記して感謝します。

2007 年 9 月

間瀬 茂

目 次

第 1 章 序	1
1.1 このマニュアルについて	1
1.2 R キーワード集	3
1.3 R コードの高速化と簡潔化	6
1.4 R 風のコーディングスタイル	8
第 2 章 R オブジェクトのタイプ・保管モード・属性	12
2.1 原子的なオブジェクトと再帰的なオブジェクト	12
2.2 R オブジェクトのタイプ・保管モード・属性	13
2.2.1 R オブジェクトの (内部) 保管モード <code>mode</code> , <code>storage.mode</code> , <code>typeof</code>	13
2.2.2 R オブジェクトの属性	13
2.2.3 データのクラス属性 <code>data.class</code>	14
2.2.4 オブジェクト検査・変換関数 <code>is.xxx</code> , <code>as.xxx</code>	15
第 3 章 整 数	16
3.1 整数型オブジェクト	16
3.1.1 整数の判定 <code>is.integer</code>	16
3.1.2 整数値への変換 <code>as.integer</code>	17
3.1.3 整数の表現範囲	17
3.2 整数の表現	18
3.2.1 整数値の 16 進数・8 進数表現 <code>as.hexmode</code> , <code>as.octmode</code>	18
3.2.2 整数のローマ数字表現 <code>as.roman</code>	18
3.2.3 整数値のビット毎の論理操作	19
第 4 章 倍精度実数と浮動小数点数演算	20
4.1 実数値 (<code>numeric</code> , <code>double</code>)	20
4.1.1 実数への変換と判定 <code>is.numeric</code> , <code>is.double</code> , <code>as.numeric</code> , <code>as.double</code>	20
4.1.2 実数の指数表記	21
4.1.3 16 進数記法 <code>0xnnn</code>	21
4.1.4 非数 <code>NaN</code> と正負の無限大 <code>Inf</code> , <code>-Inf</code>	22
4.1.5 組み込み実数定数 <code>pi</code>	22
4.1.6 単精度属性 <code>Csingle</code> を持つ実数ベクトルの生成と検査 <code>single</code> , <code>as.single</code>	22
4.2 浮動小数点数演算	23
4.2.1 倍精度実数表現に対する IEC 60559 規則, R の組み込みリスト <code>.Machine</code>	23

4.2.2	倍精度浮動小数点数の形式	24
4.2.3	桁落ち・桁溢れ・丸め誤差	25
4.2.4	浮動小数点数演算の落とし穴	26
4.2.5	範囲をカバーする等間隔数列 <code>pretty</code>	26
4.3	実数の丸め・切り捨て	27
4.3.1	実数の丸めと切り捨て <code>round</code> , <code>ceiling</code> , <code>floor</code> , <code>trunc</code> , <code>signif</code> , <code>zapsmall</code>	27
4.3.2	JIS, ISO, IEEE 式 (銀行型) 丸め	27
第5章 複素数		29
5.1	複素数の生成	29
5.1.1	複素数表現 <code>x+yi</code>	29
5.1.2	複素数ベクトルの生成 <code>complex</code>	29
5.1.3	複素数への変換・判定・比較 <code>as.complex</code> , <code>is.complex</code>	30
5.2	複素数の演算	30
5.2.1	複素数の基本関数 <code>Re</code> , <code>Im</code> , <code>Mod</code> , <code>Arg</code> , <code>Conj</code>	30
5.2.2	複素数の四則演算・べき乗	30
5.2.3	複素数の三角・逆三角関数, 双曲線・逆双曲線関数, 指数・対数関数	30
第6章 文字列とその操作		31
6.1	原子オブジェクト文字列 (<code>character</code>)	31
6.1.1	文字列の生成と検査 <code>character</code> , <code>is.character</code>	31
6.1.2	文字列への変換 <code>as.character</code>	32
6.1.3	組み込み文字列 <code>letters</code> , <code>LETTERS</code> , <code>month.name</code> , <code>month.abb</code>	32
6.1.4	16進数・8進数による文字表現	32
6.1.5	Unicode による文字表現	33
6.1.6	引用符なしの文字列出力 <code>noquote</code>	33
6.1.7	制御文字 <code>\b</code> , <code>\n</code> , <code>\r</code>	33
6.1.8	整数値と文字エンコーディング	33
6.2	文字列操作とマッチング	34
6.2.1	文字列を操作する関数ファミリー	34
6.2.2	文字列ベクトルの一部の取り出し・置き換え <code>substr</code> , <code>substring</code>	35
6.2.3	文字列を指定幅に切り詰める <code>strtrim</code>	35
6.2.4	文字列をパターンに従って分解 <code>strsplit</code>	36
6.2.5	文字列の結合 <code>paste</code> , <code>paste0</code>	36
6.2.6	文字数を数える <code>nchar</code>	37
6.2.7	文字の変換 <code>chartr</code> , <code>tolower</code> , <code>toupper</code>	37
6.2.8	数値を表す文字列ベクトルを数値ベクトルに変換 <code>type.convert</code>	38
6.2.9	文字列ベクトルからの対話的選択 <code>select.list</code>	38
6.2.10	文字列の部分的マッチング <code>charmatch</code>	38
6.3	正規表現	39
6.3.1	拡張正規表現	39

6.3.2	基本正規表現	41
6.3.3	Perl 式正規表現	41
6.3.4	部分名でオブジェクトを探す <code>apropos, find</code>	41
6.3.5	UNIX シェル風ワイルドカード指定を正規表現に変換する <code>glob2rx</code>	41
6.3.6	文字列のパターンマッチングと置き換え <code>grep, sub, gsub, regexpr, gregexpr, regexec</code>	42
6.4	文字列 Tips	44
6.4.1	制御文字を用いた <code>cat</code> 関数の出力の制御	44
6.4.2	文字列化による R コードのコメント化	45
6.4.3	文字列・ラベル	45
6.4.4	ローケルと文字列のエンコーディング	46
第 7 章 論理値と条件判断		47
7.1	論理値 <code>TRUE, FALSE</code>	47
7.1.1	論理値への変換と検査 <code>as.logical, is.logical</code>	47
7.1.2	論理値の省略形 <code>T, F</code>	48
7.1.3	論理値ベクトルの生成 <code>logical</code>	48
7.1.4	数値としての <code>TRUE, FALSE</code>	48
7.2	論理演算	49
7.2.1	基本論理演算子	49
7.3	オブジェクトの一致判断	51
7.3.1	オブジェクトの完全一致判断 <code>identical, isTRUE</code>	51
7.3.2	全てが真か? <code>all</code>	52
7.3.3	どれかが真か? <code>any</code>	52
7.3.4	ほとんど等しいか? <code>all.equal, attr.all.equal</code>	52
7.3.5	全てが真でなければ実行中断 <code>stopifnot</code>	53
7.3.6	ベクトル化条件選択 <code>ifelse</code>	53
7.4	論理値添字ベクトルの利用	54
7.4.1	条件を満たす添字ベクトルを得る <code>which, arrayInd</code>	55
第 8 章 因 子		56
8.1	因 子	56
8.1.1	ベクトルを因子にする <code>factor</code>	56
8.1.2	水準パターンを与えて因子を作る <code>gl</code>	58
8.1.3	組み合わせ因子を作る <code>interaction, :</code>	58
8.1.4	因子の水準属性 <code>levels</code>	58
8.1.5	因子の水準数 <code>nlevels</code>	58
8.1.6	因子の添字操作	59
8.1.7	因子水準の並べ替え <code>reorder</code>	59
8.1.8	因子化を避ける <code>I</code>	60
8.1.9	因子化された数値ベクトルを元に戻す	60

第 9 章	バイト型データとビット操作	61
9.1	バイト (raw) 型オブジェクト	61
9.1.1	バイト列を作る・変換する・検査する <code>raw</code> , <code>as.raw</code> , <code>is.raw</code>	61
9.2	バイト型データに対する操作	62
9.2.1	バイト型ベクトルの圧縮と解凍 <code>memCompress</code> , <code>memDecompress</code>	62
9.2.2	バイト型データと文字列データ間の変換 <code>charToRaw</code> , <code>rawToChar</code>	62
9.2.3	バイト型データのビットシフト <code>rawShift</code>	62
9.2.4	バイト型・整数型データのビット列への変換 <code>rawToBits</code> , <code>intToBits</code>	63
9.2.5	ビット列をバイト型・整数型データへパックする <code>packBits</code>	63
9.2.6	バイト型ベクトルのパターンマッチング <code>grepRaw</code>	64
第 10 章	特殊オブジェクト NA, NULL	65
10.1	NA 値	65
10.1.1	NA 値は論理型, 様々な NA 値	65
10.1.2	NA 値を含む数値・論理演算	66
10.1.3	NA 値が含まれるかどうかの検査 <code>is.na</code>	66
10.1.4	NA 値を含むデータの前処理 <code>na.omit</code> , <code>na.fail</code> 等	66
10.1.5	NA 値を置き換える <code>x[is.na(x)] <- y</code>	67
10.2	NULL 値	67
10.2.1	NULL 値を作成・検査する <code>as.null</code> , <code>is.null</code>	67
10.2.2	属性を消す 属性 <code><- NULL</code>	68
第 11 章	繰り返しと条件判断	69
11.1	繰り返し	69
11.1.1	範囲に渡って繰り返す <code>for</code> ループ	69
11.1.2	条件が満たされる限り繰り返す <code>while</code> ループ	70
11.1.3	単純繰り返し <code>repeat</code> ループ	70
11.1.4	次の実行サイクルへ飛ぶ <code>next</code>	70
11.1.5	実行の中断 <code>break</code>	71
11.2	選択実行	71
11.2.1	条件による分岐 <code>if</code> , <code>if else</code>	71
11.2.2	ベクトル化条件分岐 <code>ifelse</code>	71
11.2.3	多重選択 <code>switch</code>	72
11.2.4	メニューによる選択 <code>menu</code>	73
11.3	繰り返しと条件判断 Tips 集	73
11.3.1	<code>for</code> 文の返り値	73
11.3.2	相異なるループ変数による 3 重ループ	74
11.3.3	ベクトル成分の長さ 3 の全ての順列・組み合わせに関してループする	74
11.3.4	<code>for</code> ループの落とし穴	74
11.3.5	<code>if</code> 文で数値を論理値として使う	76
11.3.6	<code>if</code> , <code>ifelse</code> 文の返り値	76

第12章 ベクトル	77
12.1 ベクトルを作る	77
12.1.1 要素を結合してベクトルを作る <code>c</code>	77
12.1.2 規則的なベクトルを作る <code>numeric</code>	77
12.1.3 規則的なベクトルを作る コロン演算子 <code>:</code>	78
12.1.4 規則的なベクトルを作る <code>seq, sequence</code>	78
12.1.5 規則的なベクトルを作る <code>rep</code>	79
12.1.6 長さモードを指定してベクトルを作る <code>vector, as.vector</code>	79
12.1.7 ランダムなベクトルを作る <code>sample, sample.int</code>	80
12.2 ベクトルの添字操作	81
12.2.1 ベクトル要素の抽出と括弧演算子 <code>[]</code>	81
12.2.2 ベクトル要素の置き換えと括弧演算子 <code>[] <-</code>	81
12.2.3 論理値ベクトルによるベクトルの添字操作	81
12.2.4 名前ラベルによるベクトル要素の抽出	82
12.2.5 因子によるベクトル要素の抽出	82
12.2.6 条件に適合するベクトル要素の添字を得る <code>which</code>	82
12.2.7 ベクトル要素の抽出と2重括弧演算子 <code>[[]]</code>	83
12.2.8 名前属性を用いたベクトル要素の抽出	83
12.2.9 添字を用いたベクトルの部分代入の内部的機構	84
12.2.10 ベクトルの一部を取り出す <code>subset</code>	84
12.3 ベクトルを操作する関数	85
12.3.1 ベクトルの長さ <code>length</code>	85
12.3.2 ベクトル要素の逆転 <code>rev</code>	85
12.3.3 累積関数・差分 <code>cumxxx, diff</code>	85
12.3.4 ベクトルから同じ数が引き続く回数とその数を取り出す <code>rle, inverse.rle</code>	86
12.3.5 一意化 <code>unique</code>	86
12.3.6 重複する要素の添字を返す <code>duplicated</code>	86
12.3.7 一部を置き換える <code>replace</code>	87
12.3.8 要素を挿入(付加)する <code>append</code>	87
12.3.9 要素のマッチング <code>match</code>	87
12.3.10 ベクトルの要素に因子グループ毎に関数を適用する <code>ave</code>	87
12.3.11 ベクトルの大小順並べ替え添字ベクトル <code>order, sort.list</code>	88
12.3.12 数値ベクトルを指定した区間に分類する <code>cut</code>	89
12.3.13 ある値が増加数列のどこにあるかを計算 <code>findInterval</code>	89
12.3.14 数値ベクトルの要約 <code>summary, fivenum</code>	89
12.3.15 関数の連続適用によるベクトル生成 <code>sapply</code>	90
12.3.16 ベクトルの外積 <code>outer</code>	90
12.4 long vectors	90
12.5 ベクトル Tips 集	91
12.5.1 ベクトルの内積・ノルムを求める	91

12.5.2	ベクトル要素の順序を他のベクトルの要素の大小に応じて並べ替える . . .	91
12.5.3	ベクトルをリストに変換する	91
12.5.4	数値ベクトルを表す文字列をベクトルに変換する	92
12.5.5	数値ベクトルの対話的入力	92
12.5.6	リストの要素のベクトル化と再リスト化 <code>unlist</code> , <code>relist</code>	92
12.5.7	オブジェクトのシリアル化 <code>serialize</code> , <code>unserialize</code>	93
第 13 章 行 列		94
13.1	行列の生成・検査、行列への変換	94
13.1.1	1つのベクトル・リストから行列を作る <code>matrix</code>	94
13.1.2	行列に変換する・行列かどうか検査する <code>as.matrix</code> , <code>is.matrix</code>	95
13.1.3	次元属性を与えてベクトルを行列に変換する <code>attr</code> , <code>attributes</code>	95
13.1.4	ベクトルの行列への効率的な変形	95
13.1.5	次元を与えて要素が全て 0 のベクトルや行列を作る <code>mat.or.vec</code>	96
13.1.6	データフレームを数値行列に変換 <code>data.matrix</code>	96
13.1.7	属性を持つオブジェクトの生成 <code>structure</code>	96
13.1.8	複数のベクトル・行列をつなげて行列を作る <code>rbind</code> , <code>cbind</code>	97
13.1.9	行列の要約 <code>str</code> , <code>summary</code>	97
13.1.10	文字列行列・リスト行列	98
13.2	行列の次元・次元名属性	98
13.2.1	行列の次元属性 <code>dim</code> , <code>nrow</code> , <code>ncol</code>	98
13.2.2	行列の次元名属性 <code>dimnames</code> , <code>rownames</code> , <code>colnames</code>	99
13.2.3	ベクトル・リストに次元属性を与えて行列を作る <code>dim</code>	99
13.2.4	行列をベクトル化する <code>as.vector</code>	100
13.2.5	行列をベクトルとしてアクセスする <code>as.vector</code>	100
13.3	行列の添字操作	101
13.3.1	行列要素の抽出 括弧演算子 <code>[,]</code>	101
13.3.2	行列要素の置き換え 括弧演算子 <code>[,]</code> <code><-</code>	102
13.3.3	論理値行列による行列の添字操作	102
13.3.4	添字行列で行列要素を取り出す・変更する	103
13.3.5	ある条件に適合する行列要素の添字を得る <code>which</code>	103
13.3.6	行列要素の抽出 2重括弧演算子 <code>[[,]]</code>	104
13.3.7	添字行列による行列要素の抽出	105
13.3.8	行列の一部を取り出す <code>subset</code>	105
13.4	転置行列、対角・三角行列	105
13.4.1	転置行列 <code>t</code>	105
13.4.2	対角行列 <code>diag</code>	106
13.4.3	三角行列 <code>lower.tri</code> , <code>upper.tri</code>	106
13.4.4	行列は対称か <code>isSymmetric</code>	107
13.5	行列の操作	107

13.5.1	行列に対する四則演算	107
13.5.2	行列の列のスケール化 <code>scale</code>	108
13.5.3	行列の次元にベクトルを適用 <code>sweep</code>	109
13.5.4	重複した行・列の検査と一意化 <code>duplicated</code> , <code>unique</code>	109
13.5.5	リストの各要素に指定関数を適用した結果をベクトル・行列の形で返す <code>sapply</code>	109
13.5.6	行列にその周辺和をつけ加えた行列を作る <code>addmargins</code>	110
13.5.7	行列の行和・列和・行平均・列平均 <code>rowSums</code> , <code>colSums</code> , <code>rowMeans</code> , <code>colMeans</code>	110
13.5.8	グルーピングによる行和 <code>rowsum</code>	110
13.5.9	行列の各行の最大要素位置 <code>max.col</code>	111
13.5.10	行番号・列番号からなる行列を生成 <code>col</code> , <code>row</code>	111
13.6	行列に対する各種積	112
13.6.1	行列積・クロス積 <code>%*%</code> , <code>crossprod</code> , <code>tcrossprod</code>	112
13.6.2	行列・配列の外積 <code>%o%</code> , <code>outer</code>	112
13.6.3	行列のクロネッカー積 <code>%x%</code> , <code>kronecker</code>	113
13.7	距離行列 <code>dist</code>	114
13.8	行列 Tips 集	116
13.8.1	行列をベクトルに変換する	116
13.8.2	整数行列の保管モードを整数型にする	116
13.8.3	行列のグラフィックス表示 <code>plot</code> , <code>matplot</code> , <code>symnum</code>	116
13.8.4	大きな行列の一部を見る	117
13.8.5	行列の全体としての同等性を検査	117
13.8.6	実数行列の誤差範囲内での同等性を検査	117
13.8.7	要素全てが 0(全てが 1) の行列を作る	117
13.8.8	NA 値を 0 で一括置き換え	118
13.8.9	対称行列の効率的計算	118
13.8.10	数値データフレームを行列に変換する	118
13.8.11	行列の要素の自乗の総和	118
13.8.12	複数ベクトルの要素の全ての組み合わせからなる行列	119
13.8.13	行列の次元名を機械的につける	119
13.8.14	行列・配列のコンパクトな表示	119
13.8.15	完全なケース (NA を含まない) の行だけを取り出す	120
13.8.16	同じ行列を何度も作成する手間	120
13.8.17	行列の一般化内積	120
13.8.18	ベクトルから上・下三角行列を作る	121
13.8.19	行列を逆対角線に関して反転する	121
13.9	粗・密行列 パッケージ <code>Matrix</code>	121

第 14 章 配 列		122
14.1	配列の生成と操作	122
14.1.1	配列の生成 <code>array</code>	122
14.1.2	1 重鉤括弧演算子 <code>[]</code> による配列の要素の取り出し	123
14.1.3	1 重鉤括弧演算子 <code>[]</code> による配列の要素の取り出し <code>drop=FALSE</code> オプション	123
14.1.4	2 重鉤括弧演算子 <code>[[]]</code> による配列の要素の取り出し	123
14.1.5	論理値による配列の要素の取り出し・置き換え	124
14.1.6	配列の次元に名前属性をつける <code>dimnames(x) <-</code>	124
14.1.7	配列次元名の自動生成 <code>provideDimnames</code>	124
14.1.8	配列の次元 <code>dim, nrow, ncol, NROW, NCOL</code>	125
14.1.9	配列の次元の変更 <code>dim(x) <-</code>	125
14.1.10	配列のスライス添字 <code>slice.index</code>	125
14.1.11	配列の一般化転置 <code>aperm</code>	125
14.1.12	配列のあるマージンに関数を適用 <code>apply</code>	126
14.1.13	配列のマージンに周辺和を加える <code>addmargins</code>	126
14.1.14	配列の行和・列和・行平均・列平均 <code>colSums, rowSums, rowMeans, colMeans</code>	127
14.1.15	配列から長さ 1 の次元を取り除く <code>drop</code>	127
14.1.16	配列の重複するマージンを検査する・取り除く <code>duplicated, anyDuplicated, unique</code>	127
14.1.17	配列のベクトル化とリスト化 <code>as.vector, as.list</code>	128
14.2	配列 Tips 集	128
14.2.1	配列中の要素の並べかた	128
14.2.2	配列にダミー次元を加える	129
14.2.3	配列をベクトルとしてアクセスする	129
14.2.4	配列のマージン毎の一致検査	129
14.2.5	任意オフセットを持つ行列・配列を真似る	130
第 15 章 リ ス ト		131
15.1	リストを生成する	131
15.1.1	リストを生成する <code>list, as.list, is.list</code>	131
15.1.2	ベクトルをリストに変換 <code>as.list</code>	132
15.1.3	リストの連結 <code>c</code>	132
15.1.4	リストの成分を取り出す <code>[], [[]], \$</code>	132
15.1.5	リストの行列化	133
15.1.6	リスト成分の消去	133
15.1.7	関数のリスト返り値	134
15.1.8	<code>for</code> ループのリスト範囲	134
15.1.9	関数仮引数リスト <code>alist</code>	134
15.2	リストを操作する関数	134

15.2.1	リストをベクトルに変換 <code>unlist</code>	134
15.2.2	リストを再帰的に変更する <code>modifyList</code>	135
15.3	リスト Tips 集	135
15.3.1	関数リストの行列化	135
15.3.2	データとその解析関数の一括リスト化	136
15.3.3	リストの成分名を関数引数で与える	136
15.3.4	関数引数にリストとその成分名を与える	137
15.3.5	リストを行列に変換	137
15.3.6	リストに同一の変数ラベルを与える	137

第 16 章 データフレーム 138

16.1	データフレーム	138
16.1.1	データフレームを作る <code>data.frame</code>	138
16.1.2	データフレームを変形する <code>transform</code>	139
16.1.3	データフレームに対する 1 重・2 重鉤括弧演算子 <code>[]</code> , <code>[[]]</code>	140
16.1.4	データフレームの一部を取り出す <code>subset</code>	140
16.1.5	2 つのデータフレームを結合する <code>merge</code>	141
16.1.6	2 つのデータフレームを横・縦に結合する <code>cbind</code> , <code>rbind</code>	142
16.1.7	数値データフレームの行和・列和・行平均・列平均 <code>colSums</code> , <code>rowSums</code> , <code>rowMeans</code> , <code>colMeans</code>	143
16.1.8	数値データフレームのグルーピングした列和 <code>rowsum</code>	143
16.1.9	リスト成分ベクトルを因子に応じて整列化する <code>stack</code> , <code>unstack</code>	144
16.1.10	経時観測データフレームを横長・縦長形式に相互変換 <code>reshape</code>	144
16.1.11	データフレームを因子でグループ化し関数を適用 <code>by</code>	146
16.1.12	ケースを加える・置き換える	147
16.1.13	データフレームを数値行列に変換 <code>data.matrix</code>	148
16.2	データフレームを用いた作業	148
16.2.1	データフレームを環境として登録・削除する <code>attach</code> , <code>detach</code>	148
16.2.2	データから作られた環境中で R 表現式を評価する <code>with</code>	150
16.2.3	不要な水準を取り除く <code>droplevels</code>	151
16.3	データフレーム Tips 集	151
16.3.1	データフレームを吟味する	151
16.3.2	データフレームの変数名を簡略化する	152
16.3.3	データフレームの重複するケースを検査する・取り除く <code>duplicated</code> , <code>anyDuplicated</code> , <code>unique</code>	153
16.3.4	データフレームに新しい変数を加える	153
16.3.5	ファイルデータを登録する <code>attach</code>	153
16.3.6	定数値である変数を除く	154
16.3.7	完全なケースだけを取り出す <code>complete.cases</code>	154

16.3.8	ベクトル・因子の全ての組み合わせからなるデータフレームを作る <code>expand.grid</code>	154
16.3.9	2つのデータフレームを横に結合する	154
16.3.10	データフレームの変数成分を取り出す速度比較	155
16.3.11	データフレームに同一の変数ラベルを与える	155
16.3.12	エディタを使ったオブジェクトの編集 <code>edit</code> , <code>fix</code> , <code>data.entry</code>	155
16.3.13	文字列変数を因子化しない	156
16.4	アドオンパッケージ <code>data.table</code>	157
16.4.1	データテーブルの操作の例	157

第 17 章 関 数 162

17.1	関数の書き方の基本	162
17.2	関 数 名	163
17.2.1	引用符で囲む必要がある関数名	163
17.2.2	関数名に使えない・好ましくない名前	163
17.2.3	名前のない関数	164
17.2.4	関数名に関するその他注意	164
17.3	関数の仮引数	164
17.3.1	関数仮引数リスト	164
17.3.2	省略時既定値つき仮引数	165
17.3.3	仮引数名は既存変数名と同じでも良い	166
17.3.4	その他の仮引数	166
17.3.5	仮引数リストを取り出す・設定する <code>formals</code> , <code>args</code>	166
17.3.6	仮引数のマッチング <code>match.arg</code>	167
17.4	実 引 数	167
17.4.1	実引数の遅延評価	167
17.4.2	仮引数名の入力の省略	168
17.4.3	実引数リストを取り出す <code>substitute</code>	168
17.4.4	実引数が存在するかどうかチェック <code>missing</code>	169
17.4.5	モデル式とチルダ演算子 ~	169
17.5	関 数 本 体	170
17.5.1	関数本体を取り出す・設定する <code>body</code>	171
17.5.2	関数の再帰的定義 <code>Recall</code>	171
17.5.3	関数内部での関数定義	171
17.6	関数返り値	172
17.6.1	暗黙の返り値	172
17.6.2	返り値の指定 <code>return</code>	172
17.6.3	リスト・ベクトル返り値	173
17.6.4	不可視返り値 <code>invisible</code>	173
17.6.5	返り値のベクトル化	174

17.6.6	返り値の活用	174
17.6.7	関数オブジェクトを返り値にする関数	175
17.7	関数のエラー・終了処理	175
17.7.1	エラー処理 <code>stop</code>	175
17.7.2	エラー処理 <code>stopifnot</code>	176
17.7.3	警告メッセージ <code>warning</code>	176
17.7.4	メッセージを作る・抑制する <code>message</code> , <code>suppressMessages</code>	176
17.7.5	終了処理 <code>on.exit</code>	176
17.8	関数のデバッグ	177
17.8.1	万能デバッグ関数 <code>cat</code>	177
17.8.2	永続付値を使って関数中の変数をチェックする	177
17.8.3	デバッグ用関数 <code>browser</code>	178
17.8.4	デバッグ用関数 <code>debug</code>	178
17.8.5	デバッグ用関数 <code>recover</code>	179
17.8.6	デバッグ用関数 <code>dump.frames</code> , <code>debugger</code>	180
17.8.7	デバッグ用関数 <code>trace</code> , <code>traceback</code>	180
17.9	その他	181
17.9.1	プリミティブな関数かどうかを検査する <code>is.function</code> , <code>is.primitive</code>	181
17.9.2	ソースコード・ファイルの処理 <code>removeSource</code> , <code>srcfile</code>	181
17.9.3	Lisp, Reduce 風構文を持つ関数 <code>Reduce</code> , <code>Filter</code> , <code>Find</code> , <code>Map</code> , <code>Negate</code> , <code>Position</code>	182
17.9.4	他言語で書かれたサブルーチンの利用	183
17.10	関数 Tips 集	185
17.10.1	関数の実行速度計測 <code>system.time</code>	185
17.10.2	オブジェクト名にマッチする関数を返す関数 <code>match.fun</code>	186
17.10.3	関数中の未定義オブジェクトを固定 <code>local</code>	186
17.10.4	計算機環境で中身が変わる関数の定義	186
17.10.5	コードのボトルネックの発見 <code>Rprof</code> , <code>summaryRprof</code>	187
17.10.6	関数のソースコードを見る <code>methods</code> , <code>getS3method</code> , <code>getMethod</code>	187
17.10.7	バックチック記号 ‘	189
17.10.8	R の構文を引数として関数に渡す	189
17.10.9	既存関数の仮引数の省略時既定値を変更する	190
17.10.10	スカラー値関数のベクトル化	190
17.10.11	関数を関数内で作る方法	190
17.10.12	関数構成要素を並べたリストから関数を作る <code>as.function</code>	191
17.10.13	関数と実引数リストを与え関数呼び出しを作成 <code>call</code> , <code>do.call</code>	191
17.10.14	エラーメッセージの出力を抑制する <code>suppressWarnings</code>	191
17.10.15	エラーが起きても中断しない <code>try</code>	192
17.10.16	R のオブジェクト・命令を表す文字列を評価実行 <code>eval(parse(text=...))</code>	193

17.10.17	付値演算子 <- と =	194
17.10.18	奇妙な関数 丸括弧関数・波括弧関数	194
17.10.19	デバッグ用引数を持つ関数	195

第 18 章 R の数値関数 196

18.1	2 項型数値演算子	196
18.2	初等数値関数	197
18.2.1	三角関数・hyperbolic 関数ファミリ	197
18.2.2	対数・指数関数ファミリ	197
18.3	超越関数	197
18.3.1	ガンマ関数ファミリ	197
18.3.2	ベッセル関数ファミリ	198
18.4	擬似乱数	198
18.4.1	擬似乱数発生器	198
18.4.2	その他の擬似乱数発生器	200
18.4.3	確率分布	200
18.5	その他の関数	201
18.5.1	組み合わせ論的関数	201
18.5.2	符号・絶対値・平方根	202
18.5.3	数値ベクトルに対する逐次処理関数	202
18.5.4	丸め関数	202
18.5.5	集合演算	203
18.5.6	順序に関する関数 <code>sort</code> , <code>order</code> , <code>rank</code> , <code>xtfrm</code>	204
18.5.7	基本統計処理関数ファミリ	206

第 19 章 apply 関数ファミリ 207

19.1	apply 関数ファミリ	207
19.1.1	配列のマージンに関数を適用 <code>apply</code>	207
19.1.2	複数回の計算結果をリストで返す <code>lapply</code> , <code>sapply</code> , <code>replicate</code>	209
19.1.3	複数引数に多変数関数を多重適用 <code>mapply</code> , <code>Map</code>	210
19.1.4	スカラー引数を持つ関数をベクトル化 <code>Vectorize</code>	210
19.1.5	因子グループ毎に関数を適用 <code>tapply</code>	211
19.1.6	データフレームに対する <code>tapply</code> 関数 <code>by</code>	212
19.1.7	オブジェクトをグループに分けて要約する <code>aggregate</code>	212
19.1.8	再帰的にリストに関数を適用 <code>rapply</code>	213
19.1.9	環境中の変数に関数を適用 <code>eapply</code>	214
19.1.10	出力書式指定の <code>apply</code> 関数 <code>vapply</code>	214
19.2	apply 関数ファミリ Tips	215
19.2.1	apply 関数ファミリのループ機能だけを使う	215

第 20 章 作表関数	216
20.1 作表関数	216
20.1.1 分割表 <code>table</code> , <code>tabulate</code>	216
20.1.2 クロス集計 <code>xtabs</code>	217
20.1.3 フラットな集計表 <code>fable</code>	218
20.1.4 フラットな分割表の読み書き <code>write.fable</code> , <code>read.fable</code>	218
20.1.5 その他 <code>prop.table</code> , <code>margin.table</code> , <code>addmargins</code>	219
20.2 作表関数 Tips 集	219
第 21 章 暦日・時間	220
21.1 暦日・時間用のクラス	220
21.1.1 <code>date</code> クラス "Date"	220
21.1.2 <code>date-time</code> クラス "POSIXlt", "POSIXct"	221
21.2 暦日・時間用の関数	223
21.2.1 時間差 <code>difftime</code>	223
21.2.2 <code>date-time</code> クラスオブジェクトと文字列間の変換	223
21.2.3 ジュリアン通日	224
21.2.4 タイムゾーンとサマータイム	225
第 22 章 入出力	226
22.1 標準入出力	226
22.1.1 標準出力 <code>cat</code>	226
22.1.2 標準出力 <code>print</code>	227
22.1.3 標準出力 <code>show</code>	227
22.1.4 標準出力 <code>sprint</code> , <code>sprintf</code>	228
22.1.5 コンソールからの入力 <code>readline</code>	229
22.1.6 ファイルへの出力 <code>write</code>	230
22.1.7 ファイルへの出力 <code>sink</code>	231
22.1.8 ファイルへの出力 <code>capture.output</code>	232
22.2 コネクション	232
22.2.1 テキストコネクション <code>textConnection</code>	232
22.2.2 コネクションを作る・開く・閉じる	233
22.2.3 その他	234
22.3 データやコードを読み込む・書き出す	234
22.3.1 データをベクトルやリストに読み込む <code>scan</code>	234
22.3.2 表形式のファイルを読み込みデータフレームにする <code>read.table</code>	235
22.3.3 読み込んだファイルの欄数を数える <code>count.fields</code>	237
22.3.4 幅固定欄ファイルをデータフレームにする <code>read.fwf</code>	237
22.3.5 R コードをファイルやコネクションから読み込む <code>source</code>	237
22.3.6 <code>save</code> 関数で保存されたデータセットを再読み込み <code>load</code>	238
22.3.7 R オブジェクトを保存する <code>save</code> , <code>save.image</code>	238

22.3.8	データセットを読み込む・一覧表示する <code>data</code>	238
22.3.9	R オブジェクトのテキスト表現 <code>dump</code>	239
22.3.10	テキストファイルに書き出す・読み出す <code>dput</code> , <code>dget</code>	239
22.4	ファイル・ディレクトリ操作	239
22.4.1	ファイルやディレクトリを操作する関数	239
22.5	R オブジェクトの整形	240
22.5.1	R オブジェクトの整形 <code>format</code>	240
22.5.2	R オブジェクトの整形 <code>formatC</code>	240
22.5.3	R オブジェクトの整形 <code>encodeString</code>	241
22.5.4	R オブジェクトの整形 <code>noquote</code>	241
22.5.5	R オブジェクトの引用符による文字列化 <code>shQuote</code> , <code>sQuote</code> , <code>dQuote</code>	241
22.6	入出力 Tips 集	242
22.6.1	R セッションの標準出力をファイルに落す	242
22.6.2	現在の作業環境をファイルにセーブし、それを次回に復元 <code>save.image</code>	243
22.6.3	自前の関数定義を保存し、次回使えるようにする <code>save</code> , <code>source</code>	243
22.6.4	様々な外部形式ファイルを R に読み込む	243
22.6.5	複数のファイルを一度に読み込む	244
22.6.6	複数のファイルを一括してリストに読み込む	244
22.6.7	データのセーブとロード	244
22.6.8	先頭文字を与えて一時ファイル名を作る <code>tempfile</code>	245
22.6.9	コンソール出力を消す	246
22.6.10	<code>for</code> ループ中からのコンソール出力	246
第 23 章 R の起動・終了, バッチ処理, 環境変数		247
23.1	R の起動と終了	247
23.1.1	R の起動と終了のメカニズム	247
23.2	バッチ処理	249
23.3	環境変数	250
23.3.1	環境変数の確認・設定	251
第 24 章 パッケージ		252
24.1	パッケージ	252
24.1.1	パッケージのインストール・更新・削除	252
24.1.2	パッケージの読み込み	254
24.1.3	パッケージの自動読み込み <code>autoload</code>	256
24.1.4	フック <code>setHook</code>	256
24.2	パッケージに関する情報を得る	257
24.2.1	パッケージに関する情報を得る <code>help(package=xxx)</code>	257
24.2.2	パッケージ中のファイルに関する情報を得る <code>system.file</code>	257
24.2.3	インストール済パッケージに関する情報 <code>installed.packages</code>	257
24.2.4	インストール済のパッケージのパスを得る <code>find.package</code> , <code>path.package</code>	258

24.2.5	デモ <code>demo</code> , ビニエツト <code>vignette</code> , タスクビュー <code>Task Views</code>	258
24.2.6	パッケージ中のデータの読み込み <code>data</code>	259
24.3	パッケージの作成 <code>package.skeleton</code>	260
第 25 章	R のメモリ管理用関数	261
25.1	ガベージコレクション	261
25.1.1	ガベージコレクション関数 <code>gc</code>	261
25.2	メモリ関連関数	262
25.2.1	使用メモリ量の制御とプロファイリング	262
25.2.2	R 起動時のメモリ量の制御用のコマンドラインオプション	262
25.2.3	Cons Cell メモリ量の使用状況 <code>memory.profile</code>	263
25.2.4	メモリ量の使用状況 <code>Rprofmem</code>	263
25.2.5	オブジェクトの内部的コピー状況 <code>tracemem</code>	263
25.2.6	オブジェクトのメモリサイズ <code>object.size</code>	264
第 26 章	R の内部時計に関連する関数	265
26.1	内部時計を利用した関数	265
26.1.1	プログラムの実行時間を計測 <code>system.time</code>	265
26.1.2	R の起動以来の経過時間 <code>proc.time</code>	265
26.1.3	指定秒数アイドルリング <code>Sys.sleep</code>	266
26.1.4	計算時間・R セッション継続時間を制限 <code>setTimeLimit</code> , <code>setSessionLimit</code>	266
26.1.5	ガベージコレクションに必要なだった時間 <code>gc.time</code>	266
第 27 章	環 境	267
27.1	環境・検索パス・スコープ規則	267
27.1.1	環 境	267
27.1.2	名前空間	268
27.1.3	関数のクロージャ環境・評価環境	269
27.1.4	スコープ規則・レキシカルスコープ	270
27.1.5	関数引数	271
27.1.6	予約オブジェクト	272
27.2	環境を操作する関数	273
27.2.1	検索パス <code>search</code> , <code>searchpaths</code>	273
27.2.2	環境を取得・設定・検査・作成する	273
27.2.3	環境へアクセスするための関数ファミリー	274
27.2.4	指定環境へソースコードを読み込む <code>sys.source</code>	276
27.2.5	データから環境を作る <code>attach</code> , <code>with</code>	277
27.2.6	トップレベルの環境を見つける <code>topenv</code>	277
27.2.7	永続付値 <code><<-</code> , <code>->></code>	277
27.2.8	環境への付値 <code>assign</code>	278
27.2.9	環境中の変数の抽出と設定 <code>[.[.]]</code> , <code>\$</code>	279

27.2.10	変数を指定環境中で探し、その値を返す <code>get, mget</code>	279
27.2.11	与えられた名前の変数が存在するか? <code>exists</code>	279
27.2.12	名前空間を直接操作する関数ファミリー	280
27.2.13	パッケージ中の変数にアクセスする、2重・3重コロン演算子 ::, :::	281
27.2.14	環境中の変数に関数を適用 <code>eapply</code>	281
27.2.15	環境にリスト成分を付値 <code>list2env</code>	281

第 28 章 言語オブジェクト 282

28.1	言語オブジェクト	282
28.1.1	言語オブジェクト	282
28.1.2	代入 <code>substitute</code>	284
28.1.3	呼び出しの即時実行 <code>do.call</code>	286
28.2	言語オブジェクトを操作する関数	286
28.2.1	呼び出しオブジェクトを作る <code>call</code>	286
28.2.2	表現式オブジェクトを作る <code>expression</code>	287
28.2.3	名前オブジェクトを作る <code>as.symbol</code>	287
28.2.4	R 表現式を指定環境で評価する <code>eval, evalq, local</code>	287
28.2.5	関数呼び出しの実行 <code>do.call</code>	288
28.2.6	その他 <code>substitute, quote</code>	288
28.2.7	テキストから表現式を作る <code>parse</code>	289
28.2.8	R のオブジェクト・命令を表す文字列を評価実行 <code>eval(parse(text=...))</code>	289
28.2.9	引数をフルネームで置き換えた呼び出しを返す <code>match.call</code>	290
28.2.10	表現式を文字列に変換 <code>deparse</code>	291
28.2.11	関数引数の評価を強制する <code>force</code>	291
28.2.12	直近の評価結果を記録する隠し変数 <code>.Last.value</code>	292
28.2.13	R の予約語	292

第 29 章 クラスとメソッド 293

29.1	オブジェクトとその属性	293
29.1.1	オブジェクト属性の確認・付加・変更 <code>attr, attributes</code>	294
29.2	ク ラ ス	294
29.2.1	S3 クラス	295
29.3	S4 クラス	295
29.3.1	S4 クラスの定義	296
29.3.2	クラススロットの操作	298
29.3.3	クラスの拡張と継承	299
29.3.4	クラススロットのプロトタイプ	300
29.3.5	クラスの継承関係を検査 <code>is, extends, setIs</code>	300
29.3.6	ユニオンクラス <code>setClassUnion</code>	301

29.3.7	S4 クラスでの S3 クラスの利用 <code>setClass</code> , <code>setOldClass</code>	301
29.4	総称的関数	302
29.4.1	S3 総称的関数	303
29.4.2	S4 総称的関数 <code>standardGeneric</code> , <code>setGeneric</code>	303
29.5	メソッド	305
29.5.1	パッケージに含まれるメソッドの扱い	305
29.5.2	独自の S3 メソッドを定義する	305
29.5.3	メソッドを処理する関数ファミリー	306
29.5.4	S4メソッド	306
29.5.5	次のメソッドを適用する <code>callNextMethod</code>	308
29.6	グループ総称的関数とグループメソッド	308
29.7	S3 グループ総称的関数とグループメソッド	308
29.8	S4 グループ総称的関数とグループメソッド	309
29.8.1	メソッドを処理する関数	311
29.8.2	総称的なプリミティブ・内部関数	313


第 30 章 情報を得る 314

30.1	R のオブジェクトに関する情報	314
30.1.1	オブジェクトの構造を見る <code>str</code>	314
30.1.2	どんなオブジェクトがあるか <code>ls</code>	315
30.1.3	重複するオブジェクトを一覧する <code>conflicts</code>	315
30.1.4	オブジェクトの一覧とその構造の同時表示 <code>ls.str</code> , <code>lsf.str</code>	316
30.1.5	R の組み込みオブジェクトのヘルプ文章を見る・参考コードを実行する <code>help</code> , <code>example</code>	317
30.1.6	デモ <code>demo</code> , ビニエツト <code>vignette</code> , タスクビュー <code>Task Views</code>	318
30.1.7	必要な情報を探す <code>RSiteSearch</code>	318
30.1.8	R news を読む <code>readNEWS</code>	319
30.2	使用中の R・計算機に関する情報	319
30.2.1	使用中の R に含まれる関数に関する情報	319
30.2.2	オプションを見る・設定する <code>options</code> , <code>getOption</code> , <code>.Options</code>	320
30.2.3	使用中の計算機・OS・R のバージョン情報を得る <code>sessionInfo</code>	320
30.2.4	使用中の R の機能の確認 <code>capabilities</code> , <code>.Platform</code>	321
30.2.5	R のホームディレクトリを表示する <code>R.home</code>	321
30.2.6	作業ディレクトリを得る・変える <code>getwd</code> , <code>setwd</code>	321
30.2.7	その他 <code>license</code> , <code>contributors</code> , <code>Sys.which</code>	321
30.2.8	R のホームディレクトリに関する情報を得る <code>R.home</code>	322
30.3	そ の 他	322
30.3.1	R の関連ドキュメントを表示する <code>RShowDoc</code>	322
30.3.2	必要な情報を探す	322

第 31 章	バイトコードコンパイラパッケージ compiler	324
31.1	バイトコードコンパイラ	324
31.1.1	バイトコンパイル関数 <code>cmpfun</code> , <code>compile</code>	324
31.1.2	JIT バイトコンパイル <code>enableJIT</code> , パッケージのバイトコンパイル <code>compilePKGS</code>	326
第 32 章	並列処理パッケージ parallel	327
32.1	クラスタの作成と操作	327
32.1.1	R における並列処理機能	327
32.1.2	コア数の確認・クラスタ生成と終了 <code>detectCores</code> , <code>makeCluster</code> , <code>stopCluster</code>	328
32.1.3	親プロセス中のオブジェクトを子プロセスに移出 <code>clusterExport</code>	329
32.1.4	並列計算の時間計測	329
32.1.5	<code>sapply</code> と <code>mapply</code> 関数の並列版 <code>parSapply</code> , <code>clusterMap</code>	330
32.2	自動化並列計算関数	330
32.2.1	自動化並列計算関数 <code>pvec</code>	330
32.2.2	非同期的自動化並列計算 <code>mcpParallel</code> , <code>mccollect</code>	332
32.2.3	自動化並列計算関数 <code>mclapply</code> , <code>mcmapply</code> , <code>mcMap</code>	333
32.2.4	子プロセス中で特定のパッケージを利用する	333
32.2.5	子プロセスの負荷の均衡 (load balancing)	334
32.3	子プロセスの擬似乱数	334
32.3.1	子プロセスの擬似乱数系列の設定	334
32.3.2	L'Ecuyer の並列化擬似乱数発生器 <code>RNGkind("L'Ecuyer-CMRG")</code>	335
32.4	R の新しい実装 <code>parR</code> (a pretty quick version of R)	336
付録	R を用いた講義デモ用の関数	337
参考文献		339
索引		342

第 1 章

序

 はかつてのベル研究所で開発された S 言語を基本とした対話的統計解析環境・システムである。単なる計算機言語システムではなく、対話的統計解析を効率的に行えるように、過剰とも思える豊富なシステム関数、様々な統計データの保存・編集機能や高品位のグラフィックス機能を持つ。また、豊富な組み込みデータセットを備えており、各関数毎に詳細なヘルプ文章と例示用コードを備えている。

1.1 このマニュアルについて


R は本体だけを取っても、全部で千数百あまりの関数・データセットおよび関連ドキュメント・コードの巨大な集合体である。どのような関数・データセットがあるか、それらの関連は容易には窺い知ることができない。関数は、実際のデータと解析目的の多様性に応じ、多数のオプション引数を持ち、同じ関数から微妙に、場合によれば全く異なった結果を得ることができる。R はその膨大な関数と本格的な組み込みデータセット毎に、詳細なヘルプ文章、更に例示用コードを持つ。R の使用に熟達する王道は、これらのヘルプ文章を繰り返し読み、参考コードを吟味^{*1}することであろう。しかしながら、全て英語で書かれたヘルプ文章および例示コードの理解には、語学的な障壁に加え、広範囲に渡る統計手法と、計算機言語としての R の機構に関するかなりの知識を既に持っていることが前提となる。

著者は R の 1 ユーザであり、計算機言語一般にも必ずしも詳しくはない。また、R が持つ広範な機能の実際の実装方法にも決して詳しくない。結果として多くの誤解が含まれている可能性がある。日本語訳語だけを取っても、必ずしも各分野で慣用的なもの（それら自体が混乱していることは別としても）になっているとは限らないであろう。著者は、こうした欠陥を持つにせよ、このマニュアルが R のプログラミング機能を一般ユーザが理解する助けになると期待し、また確信している。

^{*1} 実際、RjpWiki に寄せられる質問の過半数は、関連ヘルプ文章を良く読めば答えが得られることを強調しておきたい。

第2章

R オブジェクトのタイプ・ 保管モード・属性

 の全てのデータ型の基本は原子的 (atomic) なオブジェクトである。全てのオブジェクトは内部的表現法を表す保管モードを持ち、更に多くはその素性を表す属性を持つ。こうしたオブジェクトに対しては、そのタイプ・モード・属性を検査・取り出し・設定するための関数ファミリーがある。

2.1 原子的なオブジェクトと再帰的なオブジェクト

R の全てのデータ型の基本は原子的なオブジェクトである。原子的なデータ型は、論理値 (logical)、整数 (integer)、倍精度実数 (numeric, double)、複素数 (complex)、文字列 (character)、バイト (raw)、そして NA (Not Available)、Inf、NaN と NULL である。それぞれの原子型については後の章で解説する。

原子的なデータ型に対し、リストやデータフレーム等は再帰的 (recursive) なデータ型と呼ばれる。R オブジェクトは原子的であるか再帰的であるかのいずれかである。R オブジェクトが原子的かどうかは関数 `is.atomic` で判定できる。原子的な要素を持つベクトル (従って行列・配列) も原子的なデータ型とされる。特殊値 NA, NaN, Inf, NULL も原子的とされる。R オブジェクトが再帰的かどうかは関数 `is.recursive` で判定できる。関数やモデル式等も再帰的オブジェクトとされる。


```
# 整数・実数・文字列ベクトルは原子的
> is.atomic(1:5,sqrt(1:5),c("a","b"))
[1] TRUE TRUE TRUE
# 行列は原子的
> is.atomic(matrix(1:4,2,2))
[1] TRUE
# 組み込み関数は再帰的
> is.recursive(sin)
[1] TRUE
# リストは再帰的
> is.recursive(list(1:3,runif(3)))
```

```
[1] TRUE
# モデル式は再帰的
> is.recursive(y~x)
[1] TRUE
# 私製関数は再帰的
> is.recursive(function(x) x)
[1] TRUE
# 表現式は再帰的
> is.recursive(expression(x+1))
[1] TRUE
```

第3章

整数

3.1 整数型オブジェクト

 の原子オブジェクトの基本は整数 (integer) である。R では整数値は必要に応じて倍精度実数値に強制変換されるため、特に整数であることを意識する局面は少ない。コンソール表示においても両者の区別は困難である*1。但し、ベクトルの添字は整数でなければならず、また整数値ベクトルのサイズは対応する実数値ベクトルよりも保存サイズが少なくなる。1:3 等で作られる数列は整数値ベクトルになる。特に整数であることを指定するには表現 1L のように記号 L をつける。

3.1.1 整数の判定 `is.integer`

整数かどうかの判定には `is.integer(x)` 関数を用いる。x が全て整数からなるときのみ TRUE を返す。

```
# 整数値 1L
> is.integer(1L)
[1] TRUE
# 単なる 1 は実数 1.0
> is.integer(1)
[1] FALSE
> identical(1,1.0)
[1] TRUE
> identical(1L,1)
[1] FALSE

[1] FALSE
> is.integer(1:10)
[1] TRUE
# 整数値 1L
> is.integer(1L)
[1] TRUE
# 1L に強制変換
> is.integer(as.integer(1))
[1] TRUE
```


実数が実質整数値であるかどうかを判定するには以下のような関数を用いる。

```
> is.Integer <- function(x)
  abs(x-round(x))
  < .Machine$double.eps^0.5
> is.Integer(1)
[1] TRUE
> is.Integer(seq(1,3,by=1/2))
[1] TRUE FALSE TRUE FALSE TRUE
```

*1 因子も見かけ上整数として表示されることがあるので紛らわしい。56 頁参照。

第 4 章

倍精度実数と浮動小数点数演算

には倍精度実数 (double)^{*1}を表す原子オブジェクトがある。整数は文脈に応じて倍精度実数に変換される。倍精度実数を引数に取る数値計算関数については第 18 章 (196 頁) にまとめて紹介する。

4.1 実数値 (numeric, double)

4.1.1 実数への変換と判定 `is.numeric`, `is.double`, `as.numeric`, `as.double`

オブジェクトが実数型^{*2}かどうかは関数 `is.double` で判定できる。関数 `as.numeric` と `as.double` は数値として解釈できるものを倍精度実数化する


```
# 整数値も Inf も numeric
> is.numeric(c(1,1L,Inf))
[1] TRUE
# 数値と解釈できないものは NA とされ警告が出る
> as.numeric(c(1,1L,"a",NA,NaN,Inf))
[1] 1 1 NA NA NaN Inf
警告メッセージ:
強制変換により NA が生成されました
# 整数値も倍精度実数化される
> is.integer(as.numeric(1L))
[1] FALSE
# as.numeric と同じ
> as.double(c(1,1L,"a",NA,NaN,Inf))
[1] 1 1 NA NA NaN Inf
警告メッセージ:
強制変換により NA が生成されました
# 整数値も倍精度実数化される
> is.integer(as.numeric(1L))
[1] FALSE
警告メッセージ:
強制変換により NA が生成されました
# 長さ 3 の実数の零ベクトル
> double(3)
[1] 0 0 0
> x <-factor(sample((1:10)/10,200,
                    replace=TRUE))
# 因子は見かけ上整数値ベクトルだが...
> str(x)
Factor w/ 10 levels "0.1","0.2","0.3",...
 9 4 7 3 4 1 6 4 1 6 ...
# 実際は数値とは見なされない
> is.numeric(x)
[1] FALSE
# 整数値も Inf も numeric
> is.numeric(c(1,1L,Inf))
[1] TRUE
```

^{*1} 倍精度実数は実際は 10 進法で 15, 16 桁の表現を持つが、コンソールへは既定で 7 桁だけが表示される。

^{*2} 歴史的な理由から R では用語 `numeric`, `double`, `real` が混在している。判定関数 `is.numeric` は整数値もしくは倍精度実数値かどうかの判定を行い、倍精度実数かどうかの判定を行う関数 `is.double` とは異なる。

第5章

複素数

 の複素数 (complex) は原子オブジェクトである。複素数は2つの倍精度実数の対で表現される。複素数に関する基本的演算とその三角関数や指数関数が提供されている。

5.1 複素数の生成

5.1.1 複素数表現 $x+yi$

1つの複素数は $1+2i$, $0+2i$, $1+0i$ のような表現を持つ^{*1}。

```
# 実数の 1
> str(1)
num 1
# 複素数の 1
> str(1+0i)
cplx 1+0i
# 実数 1 と複素数 1+0i
# は等しいとされる
> 1 == 1+0i
[1] TRUE
# 厳密な判断では異なる
# オブジェクト
> identical(1,1+0i)
[1] FALSE
# これは正しいが
> 2i
[1] 0+2i
# 次はエラー。
# (-2)*1iか-2iとする
> (-2)i
エラー: 予想外のシンボルです in "(-2)i"
# 虚部が NA の複素数の
# つもり
> 1+NAi
エラー: オブジェクト 'NAi' がありません
# NA*(1i) とすれば良い
> NA*(1i)
[1] NA
# もしくは
> complex(1,1,NA)
[1] NA
# NA_complex_ である
> str(
  complex(1,NA,1))
cplx NA
> is.na(
  complex(1,NA,1))
[1] TRUE
> complex(1,NaN,1)
[1] NaN+1i
> complex(1,1,NaN)
[1] 1+NaNi
> complex(1,Inf,1)
[1] Inf+1i
> complex(1,1,-Inf)
[1] 1-Inf
```

5.1.2 複素数ベクトルの生成 `complex`


関数 `complex` は複素数ベクトル生成関数である。

```
# 実部・虚部ベクトルを与えて
# 複素数ベクトルを作る
> complex(2,1:3,2:4)
[1] 1+2i 2+3i
# 絶対値・角度ベクトルを与える
> complex(2,mod=1:2,arg=pi*(2:3)/4)
[1] 0.000000+1.000000i -1.414214+1.414214i
```

^{*1} 実数値変数 x に対し xi , $x*i$, $x*(i)$ という表現はできない。 $x*(1i)$ または $complex(1,0,x)$ とする必要がある。 1 と $1+0i$, 0 と $0i$ は R オブジェクトとしては異なる。実部と虚部のどちらか、もしくは双方が `NA`, `NaN`, `Inf`, `-Inf` であっても良い。但し直接 `NAi`, `Inf` のように記述することはできない。

第6章

文字列とその操作

 の文字列 (character)^{*1} は原子オブジェクトである。文字列 (ベクトル) はデータとして登場するばかりではなく、プログラムの一部としても頻繁に登場する。R には文字列を操作する豊富な関数ファミリーが用意されている。

6.1 原子オブジェクト文字列 (character)

文字列は 2 重引用符で挟んで `"abc"` のように表される^{*2}。1 重引用符 (右引用符) `'` で挟んで `'abc'` としても良い。左引用符 `'` は使えない。2 重引用符自体を含む文字列を作るには 1 重引用符で囲めば良い。 `\` としても良い。

```
# "abc"と'abc'は
# 同じオブジェクト
> identical("abc",'abc')
[1] TRUE
# 2重引用符自体を含む文字列
# 2重引用符は\"で表示
> 'ab\"c'
```

```
[1] "\\ab\\c\\"
> is.character('ab\"c')
[1] TRUE
> identical('ab\"c',
            "\\ab\\c\\")
[1] TRUE
# 逆も真である
```

```
> is.character("'abc'")
[1] TRUE
# 一重引用符は
# そのまま表示される
> "'abc'"
[1] "'abc'"
```

6.1.1 文字列の生成と検査 character, is.character

関数 `character` は空文字列 `""` からなる指定された長さの空文字列ベクトルを作る。関数 `is.character` は引数が文字列 (ベクトル) であるかどうかを検査する。

```
# 長さ4の空文字列ベクトル
> character(4)
[1] "" "" "" ""
# 空文字列
```

```
> character(0)
character(0)
> str(character(0))
chr(0)
```


```
> is.character(NA)
[1] FALSE
```

^{*1} 定数値文字列というニュアンスのストリング (string) という呼び方もある。また、出力やコード上では文字列と区別が難しい名前 (name) という概念がある。これは名前ラベル・引数ラベル・オブジェクト名等に使われるもので、文字列と区別される。

^{*2} データフレームの文字列変数はコンソールへの出力では 2 重引用符なしで表示される。

第7章

論理値と条件判断

でも条件判断はプログラム言語としての中核機能であるが、Rらしいプログラミングスタイルはむしろ `if` 文等の条件判断文を表に出さず、論理値ベクトルを仲介とするベクトル操作や条件判断を内包する関数 (探せば結構ある) を用いて行うことにある。

7.1 論理値 TRUE, FALSE

論理値 `TRUE`, `FALSE` は R の原子オブジェクトである。 `NA` 値も論理オブジェクトオブジェクトであり、論理値ベクトル中に含まれる可能性がある。

```
# 論理値は原子オブジェクト
> is.atomic(TRUE)
[1] TRUE
# モードは logical
> mode(TRUE)
[1] "logical"
# 型も logical
> typeof(TRUE)
[1] "logical"
# 32 ビットで表現
> object.size(TRUE)
[1] 32
```


7.1.1 論理値への変換と検査 `as.logical`, `is.logical`

数値 `x` は論理値が必要とされる文脈では、関数 `as.logical` を用いて論理値 `TRUE`, `FALSE` に変換される。実際には整数と実数の `0` が `FALSE` に、それ以外の数値は正負の無限大を含み `TRUE` に変換される。数値以外では文字列 `"T"`, `"TRUE"`, `"True"`, `"true"` は `TRUE` に変換され、文字列 `"F"`, `"FALSE"`, `"False"`, `"false"` は `FALSE` に変換される。それ以外は全て `NA` に変換される。関数 `is.logical` は論理値かどうか検査する。

```
# 数値 0 は FALSE に、1 は TRUE に変換される
> as.logical(c(0L,0,0.0,1L,1,1.0))
[1] FALSE FALSE FALSE TRUE TRUE
# NA は NA に変換される
> as.logical(c(2.3,-10,NA,NULL,Inf,-Inf))
[1] TRUE TRUE NA TRUE TRUE
# TRUE に変換される 4 種類の文字列
> as.logical(c("T","TRUE","True","true"))
[1] TRUE TRUE TRUE TRUE
# FALSE に変換される 4 種類の文字列
> as.logical(
  c("F","FALSE","False","false"))
[1] FALSE FALSE FALSE FALSE
# NA 値はなぜか論理値とされる
> is.logical(NA)
[1] TRUE
# 整数としての論理値
> c(TRUE,FALSE) * pi
[1] 3.141593 0.000000
```

第 8 章

因 子

 の因子 (factor) は整数値ベクトルの一種類と考えられるが、その真の値は対応する水準ベクトルにより間接的に表現される。同じ値を持つものをグループ化し 1 つの水準値で代表させることにより、操作が簡単になり、必要メモリも減らせる。データフレームの文字列変数は原則因子として扱われ、グループ化される。因子は統計モデル関数で特に重要*1になる。

8.1 因 子

8.1.1 ベクトルを因子にする factor


factor はベクトルを因子にする。S との互換性のために同値な関数 **ordered** がある。データフレーム等を実行する関数の内部で、変数を因子化するのに使われる。検査・変換関数 **is.factor**, **is.ordered**, **as.factor** がある。もし **ordered=TRUE** ならば因子水準は順序づけられていると見なされる。これはクラス属性だけの違いであるが、モデル当てはめ関数等では全く異なった扱いがされる。基本的には、もし **x[i]** が **levels[j]** に等しければ **x[i]** の水準は **j** とされる。**exclude** で与えられた値は水準に含められない。水準値集合 **levels** は既定でデータ値から決まるが、もし独自に与えた **levels** 中不在の値は **NA** とされる。数値 **x** に対し **exclude=NULL** とすると **NA** 値は特殊水準 **"NA"** とされ、水準の最後の値とされる。返り値はクラス属性 **"factor"** を持つオブジェクトで、属性 **"levels"**、モード **character** を持つ **x** と同じ長さの整数値コードの集合である。順序づけられた場合はクラス属性 **c("ordered", "factor")** を持つ。

因子の実際の解釈はコードと水準集合双方に依存する。水準集合が同じ 2 つの因子の比較には **as.numeric(levels(f))[f]** で数値ベクトル化することが勧められる。因子水準は既定でソートされるが、ソートはロケールに依存する。例え少しでも同じ値が頻繁に繰り返されている文字列データは、因子として扱えば必要メモリ量が減る。32 ビット機では **n** バイトの文字列の保存には **28+8*ceiling((n+1)/8)** バイトが必要だが、そうした

*1 **help(contrasts)** 参照。

第9章

バイト型データとビット操作

 に比較的新しくつけ加わったデータ型がバイト (**raw**) 型^{*1}である。計算機で処理されるデータは基本的にバイト列で表現されている。そのうちアスキー文字等可視化できるものだけからなり、テキストとして意味を持つものは限られる。ファイルも一般にバイト列として扱うことになる。

9.1 バイト (raw) 型オブジェクト

1つのバイト型データは16進数(0,1,2,...,9,a,b,c,d,e,f)の対^{*2}で表現される。一方でバイト型データのビット列(2進法)表現があり、00,01(それぞれ2進法の0,1を表す)の列で表現される。1つのバイト型データは長さ32のビット列で表現される。

```
# 255 は 16 進法では ff
> x <- as.raw(255); x
[1] ff
```

```
> str(x)
raw ff
```

9.1.1 バイト列を作る・変換する・検査する raw, as.raw, is.raw

関数 **raw** は指定された長さの0バイト00からなるベクトルを作る。**is.raw** はバイト列かどうかを検査する。**as.raw** は整数をバイト列に変換する。

```
# 長さ 2 の 0 バイトベクトル
> xx <- raw(2); xx
[1] 00 00
# 40 のバイト (16 進法) 表現
> xx[1] <- as.raw(40)
```


```
# 文字列 "A" のバイト表現
> xx[2] <- charToRaw("A")
# 16 進数の対の並びで表現
> xx
[1] 28 41
```

^{*1} そのまま RAW 型と書いたり、ロー型と訳されている例があるが、ここでは意識してバイト型と訳す。

^{*2} 21 頁参照。

第 10 章

特殊オブジェクト NA, NULL

 に存在する特殊オブジェクト NA, NULL についてまとめておく*1。NA は「不明だが何かある」ことを意味するが、NULL は「そもそも何もない」ことを意味する。これらは便利でもあるが、反面 R の操作を困難にする主要な原因の 1 つになっている。特に一連のプログラム実行中に登場する際には、エラーに悩まされることもある。

10.1 NA 値

NA(Not Available) は欠損値を表現する原子 R オブジェクトである。オブジェクトの初期化の際は、各要素を良くある 0 ではなく NA と置くと後で確認しやすい。

```
# 行列の要素を NA 値で初期化
> matrix(NA,2,2)
     [,1] [,2]
[1,] NA  NA
[2,] NA  NA
```

10.1.1 NA 値は論理型, 様々な NA 値

NA 値は、同時に論理型を持つ*2。R 2.5.0 より各種原子型オブジェクトの欠損値を表す専用オブジェクト、NA_integer_, NA_real_, NA_character_, NA_complex_ が導入された。それ以前は全ての型に対応する NA と as.character(NA) で得られる文字列型の欠損値だけが存在した。いずれも is.na 関数では TRUE となるが、NA との identical 関数による厳密な一致比較では FALSE となる。

```
# NA は論理型を持つ
> typeof(NA)
[1] "logical"

# モードは "logical"
> mode(NA)
[1] "logical"


# 保管モードは "logical"
> storage.mode(NA)
[1] "logical"
```

*1 浮動小数点数演算に関する特殊オブジェクトの NaN (非数, Not a Number) と Inf (無限, Infinity, Infimum) については第 4 章 (20 頁) で扱う

*2 比較演算等では、比較不能場合が生じると NA 値を返すことが多い。NA 値自体を論理値として扱うことで、全体としての演算を取りあえずエラーにしないようにできる。

第 11 章

繰り返しと条件判断

 は多くの計算機言語と同じような制御命令のセットを持つが、より多様で柔軟^{*1}である。

11.1 繰り返し

11.1.1 範囲に渡って繰り返す for ループ

`for(arg in range) expr` 文^{*2}はループ範囲 `range` の各要素 `arg` に対して式 `expr` を実行する。 `for` と `()` の間には空白を置いても置かなくても良い。 R ではループ範囲 `range` にベクトル・行列・リスト・データフレーム・因子が可能である。 行列・配列は実際はベクトル `as.vector(range)` としてアクセスされる。 空の範囲では一度も実行されない。

```
# ループ範囲にベクトルを
# 取る
> for (i in 1:3) cat(i, "\n")
1
2
3
# ループ範囲に文字ベクトルを
# 取る
> for(i in c("a", "b", "1"))
  cat(i, "\n")
a
b
1
# 行列ループ範囲 (実際は
# ベクトルとしてアクセス)
> X <- matrix(1:4, c(2,2))

> for(i in X) cat(i, "\n")
1
2
3
4
# リストループ範囲
> X <- list("a", 1:4, 3)
> for (i in X) cat(i, "\n")
a
1 2 3 4
3
# ループ範囲に
# データフレームを取る
> x <- data.frame(
  a=3:5, b=letters[1:3],
  c=LETTERS[7:9])


> x
  a b c
1 3 a G
2 4 b H
3 5 c I
# 文字列は因子に変換
> for(i in x)
  cat(i, "\n")
3 4 5
1 2 3
1 2 3
> Y <- list(c=rnorm(5),
  d=letters[1:3])
# 入れ子リスト
> X <- list(a=1:3, b=X)
```

*1 繰り返しや条件判断はプログラミング言語の中核を占める制御構造であり、R もいわゆる Algol 系言語の制御文を全て持つ。しかしながら、インタプリタ言語である R では、そうした制御文の使用は目立って実行速度を遅くしたり、コードが複雑になりやすい。R らしいコードは、むしろそれら（あからさまには）使わないことにある。

*2 ループ処理は一般に実行速度を遅くするボトルネックになりやすい。またコードが長くなりがちである。 `apply` 関数ファミリー (第 19 章) の使用や、ベクトル・行列・配列・データフレーム用に用意されている各種専用高速関数の使用を考えると良い。

第12章

ベクトル

 のデータ構造ではベクトル (vector) は、リストと並んで最も基本的なもので、ベクトルを引数に取り、返回值もベクトルとなる膨大な関数ファミリー*1がある。ベクトルは、数値・文字列・バイト値等、同じ型を持つ要素を1次元的に並べたもので、各要素は括弧演算子 `[]` で取り出すことができる。単一の数値・文字列・論理値も長さ1のベクトルである。

12.1 ベクトルを作る

12.1.1 要素を結合してベクトルを作る `c`

ベクトルを作る基本関数は `c`*2で、任意個の引数を取り、それを連結したベクトルを返す。引数の型が異なるときは、全てを表現できる型に強制変換される。

```
# 数値とベクトルを結合
> c(1,2,runif(1))
[1] 1.0000 2.0000 0.8775
# 文字列ベクトル
> c("A","B","CDE")
[1] "A" "B" "CDE"
# 数値と文字列が混在すると
# 文字列ベクトルになる
> c(1:3,"A")
[1] "1" "2" "3" "A"
```

`recursive=TRUE` オプションはリストの成分を再帰的に並べたベクトルを作る。

```
> c(list(c(1,2),7),rec=TRUE)
[1] 1 2 7
# 名前ラベルを継承する
> c(list(A=c(B=1,C=2),B=c(E=7)),rec=TRUE)
A.B A.C B.E
1 2 7
```

12.1.2 規則的なベクトルを作る `numeric`


関数 `numeric(n)` は0が `n` 個並んだ数値ベクトルを作る。同等な関数として `integer`, `character`, `logical` 等がある。初期値等を作るときに便利である。

*1 行列と配列は次元属性を持つベクトルに他ならず、行列・配列用の関数も適用できる可能性がある。

*2 結合 (combine, concatenate) 関数。

第 13 章

行 列

 の行列*1は行列次元属性 (および次元名属性) を持つベクトルもしくはリスト*2であり、行列として添字操作ができる。従って R における行列の概念は、単なる数値行列に留まらず、文字列ベクトルやリストも行列化できる。

13.1 行列の生成・検査、行列への変換

13.1.1 1つのベクトル・リストから行列を作る `matrix`

関数 `matrix` は行列を生成する R の基本関数であり、要素であるベクトル `data` と、それぞれ行数、列数を表す 2 つの正整数値 `ncol`, `nrow` を引数に取る。 `byrow=TRUE` ならばベクトル要素は列順に、 `byrow=FALSE` (既定) ならば行順に埋められる。 `dimnames` 引数は次元名属性を与える長さ 2 のリストで、それぞれ行名と列名を与える。もし引数 `nrow` か `ncol` のいずれかが与えられないと、 `data` の長さ等からもう一方が推測される。もし `data` の要素数が行列全体を埋めるだけないときは `data` の要素がリサイクル規則が適用される。もし `data` の長さが 0 ならば、その型に応じ `NA` や `NULL` で埋められる。

<pre># 列順で行列にする > matrix(1:8,nrow=2,ncol=4) [,1] [,2] [,3] [,4] [1,] 1 3 5 7 [2,] 2 4 6 8 # 行順で行列にする > matrix(1:8 nrow=2,ncol=4,</pre>	<pre>byrow=TRUE) [,1] [,2] [,3] [,4] [1,] 1 2 3 4 [2,] 5 6 7 8 # ベクトル 1:5 のリサイクル # 使用. 警告が出る > matrix(1:5,nrow=2,ncol=4)</pre>	<pre> [,1] [,2] [,3] [,4] [1,] 1 3 5 2 [2,] 2 4 1 3 Warning message: 行列のデータ長 [5] が行数 [2] を 整数で割った、もしくは掛けた値 ではありません</pre>
---	--	--


次元が 0 の行列も構文的には許される。

*1 注意：R プログラミングのコツの 1 つは、ベクトル・行列・配列に対する操作を、その要素をいちいち取り出すループ操作・条件判断で行わないことにある。これはコードを冗長にし、しばしば極度の低速化を招く。専用の高速内部関数を用いベクトル・行列・配列そのものを操作するべきである。ベクトル・行列・配列を論理値添字で一括して操作することも Rらしいスタイルである。

*2 実体はベクトルもしくはリストに過ぎないことは注意に値する。コンソールへの表示や各種操作では、この属性を参考に“行列風”に操作するだけである。データフレームは外見上行列と似ているが、両者は厳密に区別する必要がある。データフレームはリストであり、一見行列風に表示されるが、数値と文字列等が混在可能である。一方行列の要素は全て同じ型でなければならない (リストの行列化を除く)。

第 14 章

配 列

 の配列 (array) は次元属性を持つ R オブジェクトであり、特別な場合として行列を含む。配列の操作は、多くの点で行列の操作に似ている。この章では、特に次元数が 3 以上の狭義の配列に関する関数を紹介する。

14.1 配列の生成と操作

14.1.1 配列の生成 array

関数 `array(data=NA, dim=length(data), dimnames=NULL)` はベクトルもしくはリスト `data` から配列を生成する。次元属性 `dim` は長さ 1 以上の正整数値ベクトルである。各次元の名前ラベルである次元名属性 `dimnames` を加えることができる。`data` は列順 (より左の添字が最も早く変化する) で埋められ、要素数が不足するならばサイクル規則が適用され、もし余分ならば切り捨てられる。1 次元配列はベクトルとは区別され、関数によっては異なった扱いを受ける可能性がある。

行列に関する操作・注意の多くが配列に対しても有効である。

```
> array()
[1] NA
> array(0)
[1] 0
# dim=c(1) の 1 次元配列
# ベクトルではない
> str(array(0))
num [1(1d)] 0
# dim=c(1,3) とされる
> array(0,3)
[1] 0 0 0
# 2 次元配列 matrix(0,3,2) で
```

```
# ある
> array(0,c(3,2))
      [,1] [,2]
[1,]  0  0
[2,]  0  0
[3,]  0  0
# 3 次元配列
> array(0,c(2,2,2))
, , 1
      [,1] [,2]
[1,]  0  0
[2,]  0  0
```

```
, , 2
      [,1] [,2]
[1,]  0  0
[2,]  0  0
> x <- array(1:12,c(2,3,2))
# 次元は長さ 3
> dim(x)
[1] 2 3 2
# 総要素数
> length(x)
[1] 12
```


要素が文字列の配列。

```
# "y","z"は切り捨てられる
> array(letters[1:24],
      c(2,3,4))
, , 1
      [,1] [,2] [,3]
[1,] "a" "c" "e"
[2,] "b" "d" "f"
--途中省略--
```

```
, , 4
      [,1] [,2] [,3]
[1,] "s" "u" "w"
[2,] "t" "v" "x"
```

第 15 章

リスト

 のリスト (list)^{*1} は任意の相異なる型の R オブジェクトを一括して保存できる便利なデータ構造である。

15.1 リストを生成する

15.1.1 リストを生成する `list`, `as.list`, `is.list`

リストを作る基本関数は `list` であり、任意個数の R オブジェクトを引数に取る。各成分には名前ラベルを `name=value` の形で指定できる。リストの成分にはリスト自体も可能である。リスト `x` の各成分には、2 重括弧演算子と添字番号を用いて `x[[i]]` とするか、名前つき成分なら `$` 演算子とその名前 (を表す文字列) を用いて `x$name` もしくは 2 重括弧演算子を用いて `x[["name"]]` とする。検査・変換関数 `is.list`, `as.list` がある。

<pre># 空リスト > list(NULL) NULL # NA を用いた初期化 > list(NA) [[1]] [1] NA # 0 を単一成分とする # リスト > list(0) [[1]]</pre>	<pre>[1] 0 # 2つの空成分を持つ # リスト > list(NULL,NULL) [[1]] NULL [[2]] NULL # 2つの空成分を持つ # リスト > vector("list",2)</pre>	<pre>[[1]] NULL [[2]] NULL # 名前のない成分を持つ # リスト > x <- list(1,2); x [[1]] [1] 1 [[2]] [1] 2</pre>	<pre># 第1成分 > x[[1]] [1] 1 # 数値と文字列のリスト > x <- list(1,"2") > x [[1]] [1] 1 [[2]] [1] "2"</pre>
--	---	---	--


名前つき・なし成分を持つリスト。

<pre>> x <- list(a=1:3,b="abc", NA); x \$a [1] 1 2 3 \$b</pre>	<pre>[1] "abc" [[3]] [1] NA # x[[1]],x[["a"]] でも良い > x\$a</pre>	<pre>[1] 1 2 3 > x[["b"]] [1] "abc"</pre>
--	--	--

^{*1} R には総称的ベクトルであるリストと、言語オブジェクト中で使われる Lisp 言語風の `dotted-pair list`, `alist` という特殊リストがある。また、R の統計データに対する中心的なデータ構造であるデータフレーム (第 16 章参照) は、行列風な外見を持つリストであり、数値・文字データを一括して保存できる。

第 16 章

データフレーム

 のデータフレーム (data frame) は多変量データを表現する汎用的なオブジェクトで、行列風の外見を持つリスト^{*1}である。各列は 1 つの変数の値を表す同じ長さの数値ベクトル、文字列ベクトル (引用符なしで表示される) 等である。行列と異なり、異種の変数を同時に格納できる。各行 (case) は 1 つの対象 (例えば 1 人の被験者) に関するひとかたまりのデータ項目を並べたものと考えられる。行と列には内容を示す名前ラベルがつくことが多い。R の多くの統計処理、グラフィックス関数は引数としてデータフレームを想定し、専用のメソッド関数を持つ。その際、モデル式を用いて変数間の関係を表現することが可能になる。処理データをデータフレームの形にまとめ、それを必要に応じて変形することは、R システムで統計解析を行う必須の基本技術となる。

16.1 データフレーム

16.1.1 データフレームを作る `data.frame`


関数 `data.frame` はデータフレームを作る。データフレームは同じ長さの変数のリストで、クラス属性 "`data.frame`" を持つ。文字列変数は関数 `I` で保護されない限り、因子 (整数値による内部的表現) に変換される。行列やリスト (各成分が同じ長さのベクトル) は関数 `as.data.frame` を用い、その列や成分が変数であるデータフレームに変換できる。`data.frame` に渡される各変数は、同じ長さでなければならない。しかし、原子的なベクトル・因子、`I` で保護された文字列ベクトルは、必要ならサイクル規則が適用される。(列の) 長さが同じならデータフレームであっても良い。もし列 (変数) 名が与えられないと、適切な名前を持つ最初の成分から合成されたり、1 から始まる整数値とされる^{*2}。

^{*1} 残念ながら、この事実がデータフレームを対象とする操作を一般的に低速にする。157 頁参照。

^{*2} データフレームの変数・ケース名の自動生成関数 `provideDimnames` については 124 頁を参照。

第 17 章

関 数

 は対話的なシステムであり、試行錯誤をしながら命令を順次打ち込んでいくやり方が基本であるが、多少複雑な処理は、たとえ 1 回限りの作業でも関数としておくと、デバッグ・改良・実行が容易になる。R の真の能力は適切な関数を書く^{*1}ことにより発揮される。R には自前で関数を書くための便利な機能が豊富に用意されており、コーディングの効率が高い。一度定義された関数は、システム関数と区別なく使用することができる。

17.1 関数の書き方の基本

R の関数定義の基本書式は

```
関数名 <- function(仮引数リスト) 関数本体
```

である。仮引数リスト^{*2}はカンマで区切った仮引数列である。仮引数名=省略時既定値の形式で、仮引数の省略時既定値を与えることもできる。特別な仮引数として... (ellipsis, dot-dot-dot) 引数も利用できる。仮引数リストは空でも良い。関数本体は関数の仮引数を含む R の実行文である。実行文が 1 つの場合は括弧 { } で囲む必要はない。

<pre># 仮引数 x,y を持ち名前が # foo という関数の定義 > foo <- function(x,y) { s <- x+y;t <- x-y return(sqrt(s^2+t^2))} > foo(1,2)</pre>	<pre>[1] 3.162278 # 括弧の前には任意個数の # 半角空白をおいて良い > foo (1,2) [1] 3.162278 > foo (1,2)</pre>	<pre>[1] 3.162278 # 仮引数 x に対する実引数はベク # トル 1:3. 返り値は自然にベク # トル化されている > foo(1:3,2) [1] 3.1622 4.0000 5.0990</pre>
--	---	--

関数本体は仮引数・局所変数以外にシステム定数や既に定義されている変数・関数を含んでも良い。


<pre>> a <- pi/2 # 外部変数を含む関数例 > foo <- function(x,y)</pre>	<pre> a*sqrt(x^2+y^2) # 仮引数のない関数例 > foo <- function()</pre>	<pre> sample(1:100,5) > foo() [1] 8 91 39 11 85</pre>
--	--	---

^{*1} 実際、R の関数の多くは特に高速さを要求される基本関数や、他の言語で書かれた数値関数ライブラリを除けば、S 言語 (の R 版) で書かれている。

^{*2} `function` と丸括弧の間には任意個数の半角空白を置いて良い。

第 18 章

R の数値関数

 は通常の計算機言語ではあり得ないほどの数値関数を備えている。これらは R 同様、長年蓄積・改良されてきた信頼性の高いオープンソース数値ライブラリ等を取り込んだものである。多くは、Fortran や C, C++ 等の言語で書かれており、R はこれらを内部的に呼び出す。以下では、線形代数、最適化関連関数*1 を除いた関数等を紹介する。関数は原則としてベクトル化されており、結果もベクトルになる。複素数引数を許し、結果も複素数になる関数もある。

18.1 2 項型数値演算子

`x + y`, `x - y`, `x * y`, `x / y` 加減乗除
`x^y` 冪乗
`x**y` 冪乗 (将来廃止される可能性がある)

`x %% y` `x` を `y` で割った余り。非負で `y` 未満
`x %/% y` いわゆる整数商で値は整数


```
> x
[1] -1 0 1 2 3 4 5 6 7 8
> x %% 5
[1] 4 0 1 2 3 4 0 1 2
> x %/% 5
[1] -1 0 0 0 0 0 1 1 1 1
# x を復元
> (x%/%5)*5 + (x%%5)
[1] -1 0 1 2 3 4 5 6 7 8
> x <- 1+0.5i; y <- 3+2i
# 複素数も問題なし
> c(x+y,x-y,x*y,x/y,x^y)
[1] 4.000000+2.500000i
[2] -2.000000-1.500000i
[3] 2.000000+3.500000i
[4] 0.307692-0.038461i
[5] -0.023927+0.552381i
# (-8)^(1/3)=-2 にはならない!
> (-8)^(1/3)
[1] NaN
# 複素数として計算
# x^3=1 の 3 つの複素数解の 1 つ
> (-8+0i)^(1/3)
[1] 1+1.732051i
```

`x %% y` は適当な整数 n で $0 \leq x - ny < y$ となるときの値 $x - ny$ を表す。そのときの整数 n が `x %/% y` になる。 1^y , y^0 は常に 1 になる。 x , y が正負の `Inf` のときも、可能なら合理的な値が返る。これらの演算子は総称的関数であり、オブジェクトのクラスに応じたメソッド関数を定義できる。

*1 汎用の最適化関数 `optim`, `nlm`, 線形制約つき最適化関数 `constrOptim`, 1 変数最適化関数 `optimize`, 1 変数関数の根を求める `uniroot`, 実・複素多項式の零点を見つける `polyroot`, 数式微分関数 `deriv` 等。

第 19 章

apply 関数ファミリー

には `apply` 関数ファミリーという、ベクトル・配列・データフレーム・リストにある処理を繰り返し適用した結果を、再びベクトル・配列・データフレーム・リストとして一括して返す関数ファミリーがある。処理関数を適切に定義することで複雑な処理を行うことができる。R プログラミングの基本精神「ベクトル化」を効率的に実践でき、またループ処理を隠蔽^{*1}するため簡潔なコードを書くことができ、独立して紹介する価値がある。R を使うメリットを感じさせる関数ファミリーである。以下は R 3.0.1 にある、そうした関数^{*2}の一覧である。

<code>aggregate</code>	<code>dendrapply</code>	<code>mapply</code>	<code>sapply</code>	<code>vapply</code>
<code>apply</code>	<code>eapply</code>	<code>replicate</code>	<code>tapply</code>	
<code>by</code>	<code>lapply</code>	<code>rapply</code>	<code>Vectorize</code>	

19.1 apply 関数ファミリー

19.1.1 配列のマージンに関数を適用 `apply`


関数 `apply(X, MARGIN, FUN, ...)` は配列 (ベクトル・行列を含む) `X` の次元 (ベクトルで複数指定できる) `MARGIN` に関数 `FUN` を適用して得られる値を、ベクトル・配列・またはリストとして返す。もし `FUN` の各適用が同じ長さ `n` のベクトルを返すならば、`apply` の返り値は次元 `c(n, dim(X)[MARGIN])` の配列になる。但し、`MARGIN` が長さ 1 ならベクトル、さもなければ次元 `dim(X)[MARGIN]` の配列を返す。結果は一旦 `as.vector` で型変換されるので、例えば因子は文字列配列になる。

^{*1} `lapply`, `vapply`, `sapply`, `rapply` 関数は内部関数を用いて計算するため、処理内容によっても異なる可能性があるが、単なるループ処理よりは計算速度自体が向上すると思われる (但し、使用環境に依存する可能性がある)。それ以外もバイトコンパイルされている分計算速度は向上すると思われる。209 頁を参照。

^{*2} 関連する関数 `Reduce`, `Filter`, `Map` については 182 頁を参照。またデンドログラムの各ノードに再帰的に関数を適用する `dendrapply` 関数がある。

第20章

作表関数

には、データを分割表 (contingency table) 形式に集計する関数が幾つかある。表オブジェクトは属性 "table", "xtabs", "ftable" 等を持つ整数値配列である。

20.1 作表関数

20.1.1 分割表 table, tabulate

`table` は因子水準の組み合わせ毎にオブジェクトを集計して、表にする。関数 `tabulate` はその実働関数で、正整数値ベクトルを表化する。`as.table`, `is.table` は表への変換・検査関数である。`as.data.frame` は属性 "table" を持つ整数値配列をデータフレームに変換する。もし `dnn` が与えられないと `dimname names` が内部的に計算される。... 中の引数が名前を持てば、それが使われる。

返り値は属性 "table" を持つ整数値配列である分割表 (contingency table) である。`table` と `xtabs` には要約関数 `summary` のメソッドがあり、 χ^2 検定も行う。


```
> table(rpois(100,5))      wool L M H      quantile(Temp),Month)
0 1 2 3 4 5 6 7 8 9 10    A 9 9 9      Month
1 1 10 22 19 12 11 13 6 4 1  B 9 9 9      5 6 7 8 9
# 組み込みデータフレームを # 組み込みデータフレームを (56,72] 24 3 0 1 10
# 使った例. 変数 wool, tension # 使った例. 変数 Temp を (72,79] 5 15 2 9 10
# の組み合わせで2元分類 # クォンタイル値で分類 (79,85] 1 7 19 7 5
> with(warpbreaks,         # Monthとともに2元分類 (85,97] 0 5 10 14 5
      table(wool,tension)) > with(airquality,
      tension              table(cut(Temp,
```

オプション `deparse.level` による、次元名と次元名の名前の違い。

```
> a <- letters[1:3]      c 0 0 1      sample(a)
> b <- sample(a)        > table(a,b,deparse.level=0) a a b c
# 既定 deparse.level=1   a b c      a 1 0 0
> table(a,b)            a 1 0 0      b 0 1 0
a a b c                 b 0 1 0      c 0 0 1
a 1 0 0                 c 0 0 1      > b <- factor(
b 0 1 0                 > table(a,b,deparse.level=2) rep(c("A","B","C"),10)
```

第21章

暦日・時間

には日付(年月日, date)オブジェクトを表すクラス `"Date"` と、日付・時間(date-time)オブジェクトを表すクラス `"POSIXlt"`、`"POSIXct"` がある。これらは使用ロケールやタイムゾーンにより表現が異なる可能性がある。日付は内部的にはある基準日付からの経過日数で表されている。

21.1 暦日・時間用のクラス

21.1.1 `date` クラス `"Date"`

クラス `"Date"` は暦日(年月日, date)オブジェクトを表し、1970-01-01 以来の経過日数(負の値は過去に遡る)で表現される。内部的には実数で表現されるが、表示の際は整数値とされる。日数との加減算、比較演算が可能である。`format` や `plot` 関数は `"Date"` クラス用のメソッド関数を持つ。`as.Date` は数値を年月日オブジェクトに変換する。`weekdays`、`months`、`quarters` はそれぞれ曜日・月・四半期名を返す。

```
# 今日の日付
# 表示の際は使用ロケールに合わせて整形される
> today <- Sys.Date(); today
[1] "2013-06-11"
> str(today)
Date[1:1], format: "2013-06-11"
# "Date"クラス
> class(x)
[1] "Date"
# クラス属性 "Date" を与える
> start <- 0; class(start) <- "Date"
> start
[1] "1970-01-01"
# 等差数列
> weeks <- seq(today, len=3, by="1 weeks")
> weeks
```


```
[1] "2013-06-11" "2013-06-18" "2013-06-25"
# 曜日・月・四半期
> c(weekdays(today), months(today),
quarters(today))
[1] "火曜日" "6月" "Q2"
# 年月日オブジェクトに対する数値演算
# 1000 日後
> z <- today+1000; z
[1] "2016-03-07"
# 年月日オブジェクトに対する比較演算
> z < c("2010-08-21", "2017-1-1")
[1] FALSE TRUE
# 文字列を年月日オブジェクトに変換
> x <- as.Date("2007-1-1"); x
[1] "2007-01-01"
```

```
# 過去の閏秒のあった日付
> as.Date(.leap.seconds)
[1] "1972-07-01" "1973-01-01"
```

```
[3] "1974-01-01" "1975-01-01"
--途中省略--
[25] "2012-07-01"
```

第22章

入出力

 のデータ・プログラムの入力・出力用の機能を解説する。出力には、コンソールへの出力 (標準出力)、ファイルへの出力、そして他の命令へのリダイレクトがある。入力には、コンソールからの直接の出力 (標準入力)、コード・データファイルの読み込み、がある。R およびそのアドオンパッケージには特殊な形式のファイル (Excel ファイル, SAS 等の統計システム出力, 等) への入出力を可能にする関数がある*1 がここでは触れない。

22.1 標準入出力

22.1.1 標準出力 `cat`


関数 `cat(...,file="",sep=" ",fill=FALSE,labels=NULL,append=FALSE)` は引数のオブジェクトを連結して標準出力やファイルに出力する。現在、原子的なベクトル (リストは駄目) と名前だけが処理できる。数値等は文字列に変換された後連結されて出力される。既定 (`fill=FALSE`) では、改行文字 `"\n"` を置いたときだけ改行することを注意しよう。 `print` 関数等と比べると、 `cat` は最小限の整形 (システムオプション `width`, `digits`, `scipen` 等に依存) しかない。246 頁を参照。

```
# 分離記号"/", 改行されない
> cat(letters[1:10],sep="/")
a/b/c/d/e/f/g/h/i/j>
# 改行するが行末にも"/"が入る
> cat(letters[1:10],"\\n",sep="/")
a/b/c/d/e/f/g/h/i/j/
# 改行オプション fill=TRUE を使う
> cat(letters[1:10],sep="/",fill=TRUE)
a/b/c/d/e/f/g/h/i/j
# ファイル"test.txt"に出力
> cat(letters[1:10],file="test.txt",
      sep="/",fill=TRUE)
# 同じことを sink を使うと以下の3行と同値
> sink("test.txt")
> cat(letters[1:10],sep="/",fill=TRUE)
> sink()
# 幅 width 文字で改行, 行頭ラベルつき
> cat(paste(letters[1:3],1:10),fill=TRUE,
      labels=paste("{",1:3,"}:",sep=""))
{1}: a 1 b 2 c 3 a 4 b 5 c 6 a 7
{8}: b 8 c 9 a 10
```

*1 例えばパッケージ `foreign` 参照。

第 23 章

R の起動・終了，バッチ処理，環境変数

 の起動と終了の仕組みは普通意識することもないが，結構複雑である．この章では R の起動と終了，そして関連環境変数に関する話題を紹介する．

23.1 R の起動と終了


23.1.1 R の起動と終了のメカニズム

R の起動と終了の仕組みは次のようなステップで行われる (プラットフォーム依存の可能性がある)．

- `--no-environ` がコマンドラインオプションとして与えられない限り，R は設定すべき環境変数をサイトとユーザファイルから探す．サイト用ファイルの存在位置は，環境変数 `R_ENVIRON` が設定されていればそれが，さもなければ `R_HOME/etc/Renviron.site` (工場出荷値) が使われる．ユーザ用ファイルは，環境変数 `R_ENVIRON_USER` が設定されていればそれが，さもなければ現在のディレクトリかユーザのホームディレクトリにあるファイル `.Renviron` が (もし存在すれば) 使われる．
- `--no-site-file` がコマンドラインオプションとして与えられない限り，R はサイト用起動時プロファイルを記述したファイルを探す．このファイルは環境変数 `R_PROFILE` が指示する．もしこれがなければ既定値は (もし存在すれば) `R_HOME/etc/Rprofile.site` とされる．これに書かれたコードは R の基本パッケージに読み込まれる．基本パッケージ中のオブジェクトをうっかり上書きしないように注意すべきであり，以下の例が示すように実行可能オブジェクトに対しては `local` 関数の使用が勧められる．
- `--no-init-file` がコマンドラインオプションとして与えられない限り，R はユーザ用起動時プロファイルを記述したファイルを探す．このファイルは環境変数 `R_PROFILE_USER` が指示する．もしこれがなければ既定値は (もし存在すれば) 現在のディレクトリかユーザのホームディレクトリにある `.Rprofile` とされる．こ

第 24 章

パッケージ

 は無数にあるパッケージを読み込むことにより様々な機能をつけ加えることができる。パッケージには R 起動時に必ず読み込まれる必須パッケージ, R 本体に必ず付属する推奨パッケージ, そして全世界のユーザが作成した 6 千余りのアドオンパッケージ*1がある。この章ではパッケージに関する話題*2を紹介する。

24.1 パッケージ

パッケージのインストールや確認を行う関数を紹介する。パッケージの名前空間を直接操作する関数ファミリーについては 280 頁を, パッケージ中の変数に直接アクセスする, 2重・3重コロン演算子 `::`, `:::` については 281 頁を参照されたい。

24.1.1 パッケージのインストール・更新・削除

以下はパッケージ (もしくはパッケージバンドル) のインストール, 更新を R 内部からインターネット経由で行う*3関数ファミリーである。

<code>update.packages</code>	<code>new.packages</code>	<code>contrib.url</code>	<code>.packages</code>
<code>available.packages</code>	<code>download.packages</code>	<code>installed.packages</code>	
<code>old.packages</code>	<code>install.packages</code>	<code>remove.packages</code>	

```
> install.packages(c("XML_0.99-5.tar.gz",
  ".../Interfaces/Perl/RSPerl_0.8-0.tar.gz"),
  repos=NULL,
  configure.args=c(XML='--with-xml-config=xml-config',
  RSPerl="--with-modules='IO Fcntl'"))
--インストール情報省略--
# インストール済みパッケージ情報
> installed.packages()
      Package      LibPath                               Version Priority
```


*1 R 自体と同じくほとんどがオープンソースであるが, 一部商用使用禁止のものもある。

*2 R の起動と終了時におけるパッケージの処理については 247 頁を参照。

*3 Linux 系 OS ではソースからその場でコンパイルする, Windows や MacOSX の場合はコンパイル済みのバイナリ版が使われる。現在のパッケージはバイトコンパイル済みのものが多いようである。

第 25 章

R のメモリ管理用関数

 は実行時のメモリを管理・監視するための幾つかの関数を持つ。

25.1 ガベージコレクション

作業中に新規にオブジェクトを作ったり、消去したりを繰り返すと空きメモリは細断され、作業効率が落ちる。ガベージコレクション機能はメモリ上のオブジェクトを必要なら一続きに移動・整理し、使いやすくする。

25.1.1 ガベージコレクション関数 `gc`

大きなオブジェクト、多数のオブジェクトを消去した場合等、空きメモリは細断され効率が悪くなる。R は必要に応じメモリ中のオブジェクトを自動的に再配置するが、これを手動で強制実行する関数が `gc`^{*1} である。関数 `gcinfo` は自動ガベージコレクションの際に、出力を行うかどうかを選択する。関数 `gctorture` は開発者向けであり、更に詳細なガベージコレクションを行い、メモリリークの有無のチェックが可能になる。

関数の実行時間は、有効メモリ量と空きメモリの配置にも依存する。不要なオブジェクトがあれば削除し、ガベージコレクションを行うべきである。`system.time` 関数は、既定では時間計測の開始前にガベージコレクションを行う。関数 `gc.time` (266 頁参照) はガベージコレクションに必要な時間を計測する。


```
> gc()
Garbage collection 24 = 19+1+4 (level 0) ...
5.9 Mbytes of cons cells used (54%)
2.3 Mbytes of vectors used (37%)
      used (Mb) gc trigger (Mb) max used (Mb)
Ncells 218151  5.9   467875 12.5   350000  9.4
Vcells 250879  2.0   786432  6.0   786432  6.0
```

`reg.finalizer` はガベージコレクション時や、オプションで R 終了時に実行される関数を登録する。

^{*1} `gc` 関数は続けて 2 度実行することが勧められる。292 頁参照。

第 26 章

R の内部時計に関連する関数

には計算機の内部時計を利用した幾つかの関数がある。

26.1 内部時計を利用した関数

26.1.1 プログラムの実行時間を計測 `system.time`

関数 `system.time(expr,gcFirst=TRUE)` は `proc.time` を用いて、表現式 `expr` の実行時間を計測する。コードの高速化のためには不可欠である。既定では計測前にガベージコレクションを行う。計測される時間は順にユーザ時間・システム時間、そして経過時間 (単位秒) である。ユーザ時間は計算に要した CPU 専有時間、システム時間はプロセスの呼び出し時間、そして経過時間は両者の和で計算全体の实時間である。実際には副プロセスに関するユーザ・システム時間*1も計測されている。

```
# ベクトルを逐次拡大するまづいコード
> system.time(for(i in seq(1e5))
  x <- c(x,rnorm(1)))
ユーザ システム 経過
19.668      7.196    27.010

# まとめて生成すれば一瞬
> system.time(x <- rnorm(1e5))
ユーザ システム 経過
0.012      0.000    0.012
```

```
# ある関数 exT の実行時間. 100 回の計測値を 5x100 行列にまとめる
> x <- sapply(1:100, function(i) system.time(exT(1000)))
# ユーザ時間の要約統計量
> c(summary(x[1,]),Sd.=sd(x[1,]))
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.     Sd.
0.264000 0.268000 0.268000 0.268200 0.268000 0.280000 0.002261
```


26.1.2 R の起動以来の経過時間 `proc.time`

関数 `proc.time` は現在稼働中の R の起動以来の経過秒数を与える。5 種類の時間の意味は前節を参照。

*1 普通は 0 となる。Windows では計測できず常に NA とされる。

第27章

環境

 の基本概念である環境 (environment) ・ スコープ規則 (scoping rule) ・ 検索パス (searching path) について簡単に紹介*1する。

27.1 環境・検索パス・スコープ規則

環境*2はオブジェクトのシンボル名とその値の対の集合であるフレーム (frame) と、その囲み (親) 環境 (enclosing environment) へのポインタであるエンクロージャ (enclosure) からなる。ある環境の囲み環境は関数 `parent.env` でアクセスできる。

27.1.1 環境

環境はツリー状になっており、囲み環境が親ノードになる。ツリーのルートノードは `emptyenv` と呼ばれる空の環境であり、R の基本パッケージ (base package) の親ノード (囲み環境) である。

```
# 大局的環境
> globalenv()
<environment: R_GlobalEnv>
# 大局的環境の親環境はパッケージ stats
> parent.env(.GlobalEnv)
attr(,"name")
[1] "package:stats"
attr(,"path")
[1] "/usr/lib/R/library/stats"
# stats の親環境はパッケージ graphics
> parent.env(parent.env(.GlobalEnv))
<environment: package:graphics>
attr(,"name")


[1] "package:graphics"
attr(,"path")
[1] "/usr/lib/R/library/graphics"
# 基本 (base) 環境
> baseenv()
<environment: base>
# 基本環境の親環境は空環境
> parent.env(baseenv())
<environment: R_EmptyEnv>
# 空環境は親環境を持たない
> parent.env(parent.env(baseenv()))
以下にエラー parent.env(env) : 空の環境は親を持ちません
```

*1 主に公式マニュアル「*The R Language Definition*」 ([2]) に基づく。

*2 複数の環境を持つことができるという R の特徴は R. Ihaka と R. Gentleman が R のプロトタイプを創った際、Lisp 言語の一種である Scheme 言語のアイデアを採り入れた (但し一部で誤解されているように、R 自体が Lisp 言語で書かれているわけではない) もので、本来の S 言語にはない独自の特徴である。S-PLUS では環境はフレーム (frame) と呼ばれ、大局的環境と、関数の評価環境しかない。

第 28 章

言語オブジェクト

 には関数型計算機言語としての R の機構の基礎である言語オブジェクト (language object) の概念^{*1}がある。以下の解説は主として公式マニュアル「*The R language definition* (文献 [2])」に基づく。

28.1 言語オブジェクト

28.1.1 言語オブジェクト

計算機言語としての R を構成する 3 つの言語オブジェクト型、呼び出し (call)、表現式 (expression)、そして名前 (name)^{*2}がある。これらはそれぞれ "call", "expression" そして "name" というモードを持つ。関数 `is.language` は引数が言語オブジェクト型であるかどうかを検査する。

呼び出しオブジェクト (未評価表現式とも呼ばれるが、必ずしも適切ではない) は関数名と引数を与えて `call` 関数から作ることができる。

<pre># call 関数で # 呼び出しオブジェクトを作る > c1 <- call("round",10.4)</pre>	<pre>> c1 round(10.5) # eval 関数で解釈実行できる</pre>	<pre>> eval(c1) [1] 10</pre>
--	--	---------------------------------

表現式は呼び出しオブジェクトを並べたリスト風構造であり、リスト風の成分操作ができる。

<pre># 表現式を作る > ex1 <- expression(1+0:5) > str(ex1) expression(1 + 0:5)</pre>	<pre># ex1 は呼び出しオブジェクトで # はない > is.call(ex1) [1] FALSE</pre>	<pre># eval 関数で解釈実行 > eval(ex1) [1] 1 2 3 4 5</pre>
--	---	--


名前オブジェクトは `as.name` 関数を用いて文字列から作ることができる。

^{*1} 一般ユーザにとって特に興味のあるこの章の内容は、`substitute` 関数で実行式から作図関数等における数式ラベル・注釈を作り出す機構であろう。

^{*2} 「シンボル」(symbol) とはほぼ区別なく使われる。シンボルは Lisp 言語での用語で、「名前」は S 言語での同等物である。

第 29 章

クラスとメソッド

 のオブジェクト指向言語としての核となる機構がクラス (class) とメソッド (method) である。現在の R システムとそのパッケージには、S 言語第 3 版の仕様を実装した S3 クラス・メソッドと、S 言語第 4 版の仕様を実装した S4 クラス・メソッド*¹ が混在している。簡易で効果的ではあるが多分に見かけだけのオブジェクト指向機構である S3 クラス・メソッドに対して、S4 クラス・メソッドは本格的なオブジェクト指向機構を実現している。但し、S4 クラス・メソッドは以下で見るように多分に煩雑であり、S3 クラス・メソッドは将来もなくなることはないであろう。

クラスとメソッドに関する詳細はパッケージ等の汎用的ツールを作る際には必須の知識*²となる。一般ユーザにとっても、`str`, `plot`, `print`, `summary` 関数等、同じ関数の適用がオブジェクトに応じて全く異なる挙動を示す仕組みを知ることが R システムを理解する上で欠かせない。この章では R のクラスとメソッドについて、基本常識レベルの紹介を行う。

注意：この章で紹介する以外に R には無数のクラス・総称的関数・メソッドを対象にした関数があるが、多くは一般ユーザには無縁と思われる。`help.search("class", package="base")` 等の命令で検索してみよ。

29.1 オブジェクトとその属性


R の多くのオブジェクトは、自分がどのようなオブジェクトなのかを示す様々な属性 (attribute) を持つ。新しい属性を付加したり、既存の属性を変更することにより新しいオブジェクトを作ることができる。

*¹ 例えば、R 本体に付属するパッケージ `stats4` は S4 クラス・メソッドに基づく統計処理関数からなる。

*² S 言語仕様については S 言語の創始者たちによる [17, 18, 21] が原典である。R システムにおけるクラス・メソッドについては公式マニュアル [1, 2, 3] に解説がある。R News 4/1, 2004, pp. 33 - 36: http://cran.r-project.org/doc/Rnews/Rnews_2004-1.pdf に Thomas Lumley による解説 “*Programmer’s Niche: A Simple Class, in S3 and S4*” がある。

第 30 章

情報を得る

 に関する情報を得る方法についてまとめる。パッケージに関する情報を得る方法は 257 頁を参照されたい。

30.1 R のオブジェクトに関する情報

30.1.1 オブジェクトの構造を見る `str`

関数 `str` は引数で与えられたオブジェクトの構造を簡潔に示す。様々なクラス用のメソッドが用意されており、それぞれにふさわしい表示が自動的に選択される。豊富なオプションを設定するための専用関数 `strOptions` がある。作業中にあるオブジェクトの正体に疑問を感じたら、すかさず `str` 関数で調べることが勧められる。

```
# str 関数自体の構造, 単に関数と表示
> str(str)
function (object,...)
# lm 関数の構造, 引数も表示
> str(lm)
--出力総略--
# 整数ベクトルの構造
> str(1:5)
int [1:5] 1 2 3 4 5
# 実数ベクトルの構造
> str(as.numeric(1:5))
num [1:5] 1 2 3 4 5
# 整数行列の構造
> str(matrix(1:6,2,3))
int [1:2,1:3] 1 2 3 4 5 6
# 整数配列の構造
> str(array(1:8,c(2,2,2)))
int [1:2,1:2,1:2] 1 2 3 4 5 6 7 8


# リストの構造
> str(list(a=1:3,b=rnorm(3)))
List of 2
 $ a: int [1:3] 1 2 3
 $ b: num [1:3] 0.753 0.424 0.839
# 入れ子リストの構造
> str(list(a=1:3,b=list(c=runif(4),d=sin)))
List of 2
 $ a: int [1:3] 1 2 3
 $ b:List of 2
 ..$ c: num [1:4] 0.796 0.206 0.538 0.395
 ..$ d: function (x)
# データフレームの構造
> str(data.frame(a=1:3,b=letters[1:3]))
'data.frame': 3 obs. of 2 variables:
 $ a: int 1 2 3
 $ b: Factor w/ 3 levels "a","b","c":1 2 3
```

以下は線形モデル当てはめ結果の構造を幾つかのオプション引数で示したものである。

```
> str(lsfrit(1:9,1:9))
--結果省略--
# リストの入れ子レベルを制限
> str(lsfrit(1:9,1:9),max.level=1)
List of 4
```

第 31 章

バイトコードコンパイラパッケージ compiler

 の推奨パッケージに比較的新しくつけ加わったバイトコードコンパイラパッケージ `compiler`*1 を紹介する。バイトコードコンパイラは R コードをバイトコード*2 に変換し、実行時間が数割程度早くなる可能性がある。利用には事前に `library(compiler)` を実行しておく。

31.1 バイトコードコンパイラ

31.1.1 バイトコンパイル関数 `cmpfun`, `compile`

パッケージ `compiler` に含まれる関数は以下の通りである。

```
cmpfun
compile
cmpfile
```

```
loadcmp
disassemble
enableJIT
```

```
compilePKGS
getCompilerOption
setCompilerOptions
```

関数 `cmpfun` は関数クロージャの本体をコンパイルし、本体をそれで置き換えた新しいクロージャを返す。`compile` は表現式をバイトコードオブジェクトにコンパイルする。`cmpfile` は入力ファイル `infile` をコンパイルし、出力ファイル `outfile` に書き出す。出力ファイル名を与えないと、入力ファイル名に拡張子 `.Rc` をつけたものとされる。`loadcmp` はコンパイル済みファイルの読み込み*3 をする。`disassemble` はコードの可読表現を与える。

コンパイル時に様々な警告が `cat` 関数を用いて出力される。`options` 引数はコンパイル動作を制御する。現在の 3 つのオプションは、`optimize` (最適化レベル, 0,1,2,3), `suppressAll` (TRUE ならメッセージは出力されない), `suppressUndefined` (TRUE なら未


*1 かつては独立パッケージとして提供されていた。

*2 バイトコードコンパイラはインタプリタ型言語の実行速度を早めるために使われる。本来のコードを R が実行する際、まずそれを構文解析した中間コードに変換し、最後にそれを実行する。バイトコードコンパイラはこの中間コードを直接生成する。これは人間が解釈しやすい形式にはなっていない。

*3 `sys.source` に似るが、既定の読み込み環境はベース環境ではなく、大局的環境になる。

第 32 章

並列処理パッケージ parallel

 の推奨パッケージに比較的新しくつけ加わった並列処理パッケージ `parallel`^{*1} を紹介する。このパッケージはマルチコア CPU を持つ 1 台の計算機で並列処理を行うことを想定している。

32.1 クラスタの作成と操作

最近の CPU はマルチコア化されており、各コアが独立した CPU として動作するため、1 台の計算機を複数の計算機のように使うことができ、並列処理を手軽に実行できるようになった。但し、Windows 系 OS では使える機能に制限がある。

以下は `library(help=parallel)` で得られるパッケージ `parallel` の目次である。それぞれが関連する複数の関数を含む。詳細は各ヘルプ文章を参照されたい。

<code>children</code>	フォークされたプロセスを管理する低水準関数	<code>mclapply</code>	フォークを用いた <code>lapply</code> と <code>mapply</code> の並列版
<code>clusterApply</code>	クラスタを用いた <code>Apply</code> 操作	<code>mcpapply</code>	R 表現式を別個のプロセスで非同期的に評価する
<code>detectCores</code>	CPU のコア数を検出する	<code>nextRNGStream</code>	L'Ecuyer の <code>RngStream</code> (並列版擬似乱数生成器)
<code>makeCluster</code>	並列ソケットクラスタを生成する	<code>pvec</code>	フォークを用いたベクトル <code>Map</code> 関数の並列化
<code>mcaffinity</code>	現在のプロセスの CPU Affinity Mask を得る・設定する	<code>splitIndices</code>	クラスタに作業を分割配分する
<code>mcfork</code>	現在の R をプロセスのコピーをフォークする		

32.1.1 R における並列処理機能


互いに情報をやり取りする親・子プロセスの集まりをクラスタ (`cluster`) と呼ぶ。この章ではマルチコア CPU を持つ計算機でのクラスタを用いた並列処理の概観を与える。並列計算では

1. R を実行している親プロセス (`master process`) が、それ自体独立した R セッション

^{*1} 2つのパッケージ `multicore` と `snow` の内容を受け継ぐ。両者は子プロセスの実装の仕方に違い (`fork` と `socket`) がある。パッケージ `parallel` には解説ビニエットが存在する (`vignette("parallel")` を実行)。

付録

R を用いた講義デモ用の関数

を用いた統計学・データ解析の講義の例は現在では珍しくないようである。自分のパソコンにRをインストールすることで学生が手軽に高度な統計手法を使用できること、本格的な大規模データをインターネット経由で自由に入手できることがこうした風潮を後押ししている。

以下では著者が実際にデータ解析の講義で使用しているプレゼンテーション用の関数 `Demo`*1を紹介する。この関数はアドオンパッケージ `evaluate` に含まれる関数 `evaluate` と `replay` 関数を使い、ソースファイルを予め命令単位に解釈実行した結果をリストにまとめ、それを改めて擬似的に実行しているように見せかけているため、実際に実行する過程をほぼ忠実に再現できる。擬似実行中に作られたオブジェクトは保存されない。途中でアドリブ的に追加実行はできない。コメントおよびコメント行もそのまま表示されるため、説明をコメントとして書き込むことができ、ソースファイル自体をテキスト代わりに使うこともできる。

```
# ソースファイルを一旦実行した結果を逐一リストにまとめ、後から擬似的に実行するデモ用の関数
# リターンキーを押すたびに1命令ずつ実行する
Demo <- function(x) {      # x はソースファイル名
  require(evaluate)      # アドオンパッケージ evaluate を読み込む
  xx <- evaluate(file(x))
  tmp <- function(y) {
    yy <- y
    z <- substr(yy, 1, nchar(yy)-1)
    if(class(yy)[1] == "source")
      yy <- paste("> ", z, sep="")
    else if(class(yy)[1] == "character")
      yy <- z
    replay(yy)
    invisible(readline())
  }
  invisible(lapply(xx, tmp))
}
```

使い方は、Rのソースファイルを `foo.R` とすると `Demo("foo.R")` を実行(取りあえず

*1 関数中の補助関数 `tmp` はステップ毎に必ず入る空白行を取り除くために使われるが、作図関数等ではやはり余分な空白行が入る。

参考文献

- [1] *An Introduction to R*, R の公式マニュアル (R 入門)
CRAN の Documentation – Contributed 頁に和訳がある。
- [2] *The R language definition*, R の公式マニュアル (R の言語仕様)
CRAN の Documentation – Contributed 頁に和訳がある。
- [3] *Writing R Extensions*, R の公式マニュアル (R のパッケージ作成法)
CRAN の Documentation – Contributed 頁に和訳がある。
- [4] *R Data Import/Export*, R の公式マニュアル (データ入出力)
CRAN の Documentation – Contributed 頁に和訳がある。
- [5] *R Installation and Administration*, R の公式マニュアル (R のインストール法)
CRAN の Documentation – Contributed 頁に和訳がある。
- [6] *R Internals*, R の公式マニュアル (R の内部仕様とコーディング標準)。
- [7] *The R Reference Index*, R の公式マニュアル (R 本体と推奨パッケージ中の全てのヘルプ文章を含む)。
- [8] *R Newsletter*, R の公式ニュースレター。1年に数回オンラインで発行される。CRAN から入手可能。
- [9] *R FAQ*, R の公式マニュアル FAQ (しばしば質問される項目に対する回答集)。RjpWiki に和訳がある。
- [10] *R MACOS X FAQ*, R の公式マニュアル (MacOS X 専用の FAQ 集)。RjpWiki に和訳がある。
- [11] *R FAQ*, R の公式マニュアル (Windows 専用の FAQ 集)。RjpWiki に和訳がある。
- [12] *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*, R. Gentleman et al., Springer (2005)。著者 Gentleman は R 開発チームのメンバーで、R 創始者の 1 人。Bioconductor というバイオインフォーマティクス関連の膨大な R パッケージ群開発のリーダー。和訳「R と Biocnductor を用いたバイオインフォーマティクス」, 荒川和春他訳, シュプリンガー・ジャパン (2007)。
- [13] *R Graphics*, P. Murrell, Chapman & Hall (2005)。R のグラフィックスに関する基本的文献。著者は R 開発メンバーの 1 人。
- [14] 「The R Book」, 岡田昌史編, 九天社 (2004)。
- [15] 「R による医療統計学」, P. Dalgaard 著, 岡田昌史監訳, 丸善出版 (2007)。Intro-

- ductory Statistics with R*, P. Dalgaard, Springer-Verlag (2002) の訳。R そのものをターゲットにした初めての本。著者は R の開発メンバーの 1 人。この本で取り上げられたデータとコードは **ISwR** という R のパッケージになって公開されている。
- [16] 「R の基礎とプログラミング技法」, U. リゲス著, 石田基広訳, シュプリンガー・ジャパン (2006)。著者は R 開発メンバーの 1 人。
- [17] J.M. Chambers & J.T. Hastie (eds.), *Statistical Models in S*, Wadsworth & Brooks/Cole (1992)。[S と統計モデル – データ科学の新しい波 –], 柴田里程訳, 共立出版 (1994)。
- [18] Becker, R. A., Chambers, J. M. and Wilks, A. R., *The New S Language*. Wadsworth & Brooks/Cole (1988)。[S 言語 データ解析とグラフィックスのためのプログラミング環境 I, II], 渋谷政昭・柴田里程訳, 共立出版 (1991)。
- [19] 「S-PLUS による統計解析」, W.N. ヴェナブルズ, B.D. リプリー著, 伊藤幹夫他訳, シュプリンガー・フェアラーク東京 (2001)。原著 *Modern Applied Statistics with S-PLUS*, 4th ed., W.N. Venables & B.D. Ripley, Springer (2002)。S-PLUS のみならず R に関しても基本文献。特に統計手法の解説が詳しい。この本で取り上げられたデータとコードは **VR** という R のパッケージになって公開されている。著者リプリーは R 開発メンバーの中心人物。
- [20] 「R 基本統計関数マニュアル」, 著者が R 本体 (バージョン 2.6.2.7) に含まれる全統計関数とそのヘルプ文章をカテゴリー別に分類して翻訳したマニュアル。CRAN の非英語貢献マニュアル頁にある。404 頁の PDF ファイル。
- [21] Chambers, J.M., *Programming With Data: Guide to the S Language*, Springer-Verlag (1998)。[データによるプログラミング], 垂水共之他訳, 森北出版 (2002)。
- [22] Chambers, J.M. *Software for Data Analysis: Programming with R*, Springer-Verlag (2008)。上記の本の R 版。
- [23] RjpWiki. 日本の R ユーザがボランティアで運営している情報サイト。
URL <http://www.okada.jp.org/RWiki/index.php?RjpWiki>。
- [24] CRAN(Complete R Archive Network). R の本拠サイト
URL <http://cran.md.tsukuba.ac.jp/>。
日本の代表的ミラーサイト URL <http://cran.md.tsukuba.ac.jp/>。
- [25] R Wiki. R の公式 Wiki。様々な情報が得られる。
URL <http://wiki.r-project.org/rwiki/>。
- [26] R とパッケージの参考コードの出力画像を一同に集めたサイト「R graphical Manual」。URL はキーワード **R Grapical Manual** でネット検索せよ。
- [27] R 関連の検索エンジン「R Site Search」,
URL <http://finzi.psych.upenn.edu/search.html>。
Firefox 用のプラグインがある。URL <http://wiki.r-project.org/rwiki/>。

- [28] R 関連のもう 1 つの検索エンジン「Rseek」, URL <http://www.rseek.org/>.
- [29] S-Plus/R の関数のカテゴリー化されたリスト「S-Plus/R Function Finder」,
<http://biostat.mc.vanderbilt.edu/s/finder/finder.html>.

索引

まえがき		
初版まえがき	iv	
第2版まえがき	i	
apply 関数ファミリー 第19章		
apply 関数ファミリー	207	
因子グループ毎に関数を適用		
tapply	211	
オブジェクトをグループに分けて要約する aggregate	212	
環境中の変数に関数を適用		
eapply	214	
再帰的にリストに関数を適用		
rapply	213	
出力書式指定の apply 関数		
vapply	214	
スカラー引数を持つ関数をベクトル化 Vectorize	210	
データフレームに対する		
tapply 関数 by	212	
配列のマジックに関数を適用		
apply	207	
複数回の計算結果をリストで返す lapply, sapply, replicate	209	
複数引数に関数を多重適用		
mapply, Map	210	
apply 関数ファミリー Tips	215	
apply 関数ファミリーのルーパ機能だけを使う	215	
R オブジェクトのタイプ・保管モード・属性 第2章	12	
R オブジェクトのタイプ・保管モード・属性	13	
R オブジェクトの属性	13	
R オブジェクトの(内部)保管モード mode, storage.mode, typeof	13	
オブジェクト検査・変換関数 is.xxx, as.xxx	15	
データのクラス属性 data.class	14	
原子的なオブジェクトと再帰的なオブジェクト	12	
R の起動・終了、バッチ処理、環境変数 第23章	247	
R の起動と終了	247	
R の起動と終了のメカニズム	247	
環境変数	250	
環境変数の確認・設定	251	
バッチ処理	249	
R の数値関数 第18章	196	
2 項型数値演算子	196	
擬似乱数	198	
確率分布	200	
擬似乱数発生器	198	
その他の擬似乱数発生器	200	
初等数値関数	197	
三角関数・hyperbolic 関数ファミリー	197	
対数・指数関数ファミリー	197	
その他の関数	201	
基本統計処理関数ファミリー	206	
組み合わせ論的関数	201	
集合演算	203	
順序に関する関数 sort, order, rank, xfrm	204	
数値ベクトルに対する逐次処理関数	202	
符号・絶対値・平方根	202	
丸め関数	202	
超越関数	197	
ガンマ関数ファミリー	197	
ベッセル関数ファミリー	198	
R のメモリ管理用関数 第25章	261	
ガベージコレクション	261	
ガベージコレクション関数 gc	261	
メモリ関連関数	262	
Cons Cell メモリ量の使用状況 memory.profile	263	
R 起動時のメモリ量の制御用のコマンドラインオプション	262	
オブジェクトの内部的コピー状況 tracemem	263	
オブジェクトのメモリサイズ object.size	264	
使用メモリ量の制御とプロファイリング	262	
データ保管に使用するメモリ量の制御	262	
メモリ量の使用状況 Rprofmem	263	
メモリ関連関数		
因子 第8章	56	
因子	56	
因子化された数値ベクトルを元に戻す	60	
因子化を避ける I	60	
因子水準の並べ替え reorder	59	
因子の水準数 nlevels	58	
因子の水準属性 levels	58	
因子の添字操作	59	
組み合わせ因子を作る interaction, :	58	
水準パターンを与えて因子を作る gl	58	
ベクトルを因子にする factor	56	
環境 第27章	267	
環境・検索パス・スコープ規則	267	
環境	267	
関数のクロージャ環境・評価環境	269	
関数引数	271	
スコープ規則・レキシカルスコープ	270	
名前空間(namespace)	268	
予約オブジェクト	272	
環境を操作する関数	273	
与えられた名前の変数が存在するか? exists	279	
永続付値 <<- , >>	277	
環境中の変数に関数を適用 eapply	281	
環境中の変数の抽出と設定 [[.], \$	279	
環境にリスト成分を付値 list2env	281	
環境へアクセスするための関数ファミリー	274	
環境への付値 assign	278	
環境を取得・設定・検査・作成する	273	
検索パス search, searchpaths	273	
指定環境へソースコードを読み込む sys.source	276	
データから環境を作る attach, with	277	
トップレベルの環境を見つける topenv	277	
名前空間を直接操作する関数ファミリー	280	
パッケージ中の変数にアクセス		

- スする, 2重・3重コロ
ン演算子 `::, :::`281
変数を指定環境中で探し, そ
の値を返す `get, mget`
.....279
- 関数 第17章162
関数 Tips 集185
R のオブジェクト・命令を表
す文字列を評価実行
`eval(parse(text=...))`
.....193
R の構文を引数として関数に
渡す189
エラーが起きてでも中断しない
`try`192
エラーメッセージの出力を抑
制する
`suppressWarnings`191
オブジェクト名にマッチする
関数を返す関数
`match.fun`186
関数構成要素を並べたリスト
から関数を作る
`as.function`191
関数中の未定義オブジェクト
を固定 `local`186
関数と実引数リストを与え関
数呼び出しを作成 `call,`
`do.call`191
関数のソースコードを見る
`methods, getS3method,`
`getMethod`187
関数の実行速度計測
`system.time`185
関数を関数内で作る方法190
既存関数の仮引数の省略時既
定値を変更190
奇妙な関数 丸括弧関数・波括
弧関数194
計算機環境で中身が変わる関
数の定義186
コードのボトルネックの発見
`Rprof, summaryRprof` 187
スカラー値関数のベクトル化
.....190
デバッグ用引数を持つ関数
.....195
バックチェック記号 `'`189
付値演算子 `<-` と `=`194
関数返回值172
暗黙の返回值172
戻り値の活用174
戻り値の指定 `return`172
戻り値のベクトル化174
関数オブジェクトを返回值に
する関数175
不可視返回值 `invisible` 173
リスト・ベクトル返回值173
関数のエラー・終了処理175
エラー処理 `stopifnot`176
エラー処理 `stop`175
警告メッセージ `warning`
.....176
終了処理 `on.exit`176
メッセージを作る・抑制する
`message,`
- `suppressMessages`176
関数の書き方の基本162
関数の仮引数164
仮引数のマッチング
`match.arg`167
仮引数名は既存変数名と同じ
でも良い166
仮引数リストを取り出す・設
定する `formals, args`
.....166
関数仮引数リスト164
省略時既定値つき仮引数165
その他の仮引数166
関数のデバッグ177
永続付値を使って関数中の変
数をチェックする177
デバッグ用関数 `browser` 178
デバッグ用関数 `debug`178
デバッグ用関数 `dump.frames,`
`debugger`180
デバッグ用関数 `recover` 179
デバッグ用関数 `trace,`
`traceback`180
万能デバッグ関数 `cat`177
関数本体170
関数内部での関数定義171
関数の再帰的定義 `Recall`
.....171
関数本体を取り出す・設定す
る `body`171
関数名163
引用符で囲む必要がある関数
名163
関数名に使えない・好ましく
ない名前163
関数名に関するその他注意
.....164
名前のない関数164
実引数167
仮引数名の入力の省略168
実引数が存在するかどうか
チェック `missing`169
実引数の遅延評価167
実引数リストを取り出す
`substitute`168
モデル式とチルダ演算子
.....169
その他181
Lisp, Reduce 風構文を持つ
関数 `Reduce, Filter,`
`Find, Map, Negate,`
`Position`182
ソースコード・ファイルの処
理 `removeSource,`
`srcfile`181
他言語で書かれたサブルーチ
ンの利用183
プリミティブな関数かどうか
を検査する `is.function,`
`is.primitive`181
- 行列 第13章94
行列 Tips 集116
NA 値を 0 で一括置き換え
.....118
大きな行列の一部を見る117
- 同じ行列を何度も作成する手
間120
完全なケース (NA 値を含ま
ない) の行だけを取り出す
.....120
行列の一般化内積120
行列のグラフィックス表示
`plot, matplot, symnum`
.....116
行列の次元名を機械的につけ
る119
行列の全体としての同源性を
検査117
行列の要素の自乗の総和118
行列・配列のコンパクトな表
示119
行列を逆対角線に関して反転
する121
行列をベクトルに変換する
.....116
実数行列の誤差範囲内での同
等性を検査117
数値データフレームを行列に
変換する118
整数行列の保管モードを整数
型にする116
対称行列の効率計算118
複数ベクトルの要素の全ての
組み合わせからなる行列
119
ベクトルから上・下三角行列
を作る121
要素全てが 0 (全てが 1) の行
列を作る117
行列に対する各種積112
行列積・クロス積 `%*%,`
`crossprod, tcrossprod`
.....112
行列のクロネッカー積 `%x%,`
`kroncker`113
行列・配列の外積 `%o%, outer`
.....112
行列の次元・次元名属性98
行列の次元属性 `dim, nrow,`
`ncol`98
行列の次元名属性 `dimnames,`
`rownames, colnames` ..99
行列の添字操作100
行列をベクトル化する
`as.vector`100
ベクトル・リストに次元属性
を与えて行列を作る `dim`
.....99
行列の生成・検査・行列への変
換94
1 つのベクトル・リストから
行列を作る `matrix`94
行列に変換する・行列かどう
か検査する `as.matrix,`
`is.matrix`95
行列の要約 `str, summary` ..97
次元属性を与えてベクトルを
行列に変換する `attr,`
`attributes`95
次元を与えて要素が全て 0 の
ベクトルや行列を作る
`mat.or.vec`96

- 属性を持つオブジェクトの生成 `structure` 96
- データフレームを数値行列に変換 `data.matrix` 96
- 複数のベクトル・行列をつなげて行列を作る `rbind`, `cbind` 97
- ベクトルの行列への効率的な変形 95
- 文字列行列・リスト行列 98
- 行列の操作 107
- 行番号・列番号からなる行列を生成 `col`, `row` 111
- 行列にその周辺和をつけ加えた行列を作る `addmargins` 110
- 行列に対する四則演算 107
- 行列の各行の最大要素位置 `max.col` 111
- 行列の行和・列和・行平均・列平均 `rowSums`, `colSums`, `rowMeans`, `colMeans` 110
- 行列の次元にベクトルを適用 `sweep` 109
- 行列の列のスケール化 `scale` 108
- グルーピングによる行和 `rowsum` 110
- 重複した行・列の検査と一意化 `duplicated`, `unique` 109
- リストの各要素に指定関数を適用した結果をベクトル・行列の形で返す `sapply` 109
- 行列の添字操作 101
- ある条件に適合する行列要素の添字を得る `which` 103
- 行列の一部を取り出す `subset` 105
- 行列要素の置き換え 括弧演算子 `[,]` 102
- 行列要素の抽出 2 重括弧演算子 `[[,]]` 104
- 行列要素の抽出 括弧演算子 `[,]` 101
- 添字行列で行列要素を取り出す・変更する 103
- 添字行列による行列要素の抽出 105
- 論理値行列による行列の添字操作 102
- 距離行列 `dist` 114
- 粗・密行列 パッケージ `Matrix` 121
- Bunch-Kaufman 分解 121
- Cholesky 分解 121
- LU 分解 121
- QR 分解 121
- Shur 分解 121
- 転置行列, 対角・三角行列 105
- 行列は対称か `isSymmetric` 107
- 三角行列 `lower.tri`, `upper.tri` 106
- 対角行列 `diag` 106
- 転置行列 `t` 105
- クラスとメソッド 第 29 章 293
- S4 クラス 295
- S4 クラスでの S3 クラスの利
用 `setClass`,
`setOldClass` 301
- S4 クラスの定義 296
- クラスロットの操作 298
- クラスロットのプロトタイプ 300
- クラスの拡張と継承 299
- クラスの継承関係を検査 `is`,
`extends`, `setIs` 300
- ユニオンクラス
`setClassUnion` 301
- オブジェクトとその属性 293
- オブジェクト属性の確認・付加・変更 `attr`,
`attributes` 294
- クラス 294
- S3 クラス 295
- S4 クラス 298
- グループ総称的関数とグループ
メソッド 308
- S3 グループ総称的関数とグ
ループメソッド 308
- S4 グループ総称的関数とグ
ループメソッド 309
- 総称的関数 302
- S3 総称的関数 303
- S4 総称的関数
`standardGeneric`,
`setGeneric` 303
- メソッド 305
- S4 メソッド 306
- 総称的なプリミティブ・内部
関数 313
- 次のメソッドを適用する
`callNextMethod` 308
- 独自の S3 メソッドを定義す
る 305
- パッケージに含まれるメソッ
ドの扱い 305
- メソッドを処理する関数
..... 311
- メソッドを処理する関数ファ
ミリ 306
- 繰り返しと条件判断 第 11 章
..... 69
- 繰り返し 69
- 実行の中断 `break` 71
- 条件が満たされる限り繰り返
す `while` ループ 70
- 単純繰り返し `repeat` ループ
..... 70
- 次の実行サイクルへ飛ぶ
`next` 70
- 範囲に渡って繰り返す `for`
ループ 69
- 繰り返しと条件判断 Tips 集
..... 73
- `for` 文の返り値 73
- `for` ループの落とし穴 74
- `if`, `ifelse` 文の返り値 76
- `if` 文で数値を論理値として使
う 76
- 相異なるループ変数による 3
重ループ 74
- ベクトル成分の長さ 3 の全ての
順列・組み合わせに関
してループする 74
- 選択実行 71
- 条件による分岐 `if`, `if else`
..... 71
- 多重選択 `switch` 72
- ベクトル化条件分岐 `ifelse`
..... 71
- メニューによる選択 `menu` 73
- 言語オブジェクト 第 28 章
..... 282
- 言語オブジェクト 282
- 言語オブジェクト 282
- 代入 `substitute` 284
- 呼び出しの即時実行 `do.call`
..... 286
- 言語オブジェクトを操作する関
数 286
- R のオブジェクト・命令を表
す文字列を評価実行
`eval(parse(text=...))`
..... 289
- R の予約語 292
- R 表現式を指定環境で評価す
る `eval`, `evalq`, `local`
..... 287
- 関数引数の評価を強制する
`force` 291
- 関数呼び出しの実行 `do.call`
..... 288
- その他 `substitute`, `quote`
..... 288
- 直近の評価結果を記録する変
数 `Last.value` 291
- テキストから表現式を作る
`parse` 289
- 名前オブジェクトを作る
`as.symbol` 287
- 引数をフルネームで置き換え
た呼び出しを返す
`match.call` 290
- 表現式オブジェクトを作る
`expression` 287
- 表現式を文字列に変換
`deparse` 291
- 呼び出しオブジェクトを作る
`call` 286
- 作表関数 第 20 章 216
- 作表関数 `Tips` 219
- 作表関数 216
- クロス集計 `xtabs` 217
- その他 `prop.table`,
`margin.table`,
`addmargins` 219
- フラットな集計表 `fTable`
..... 218
- フラットな分割表の読み書き
`write.fTable`,
`read.fTable` 218
- 分割表 `table`, `tabulate` 216

- 序 第 1 章 1
 R キーワード集 3
 R コードの高速化と簡潔化 6
 R 風のコーディングスタイル 8
 このマニュアルについて 1
- 情報を得る 第 30 章 314
 R のオブジェクトに関する情報 314
 R news を読む readNEWS 319
 R の組み込みオブジェクトのヘルプ文章を見る・参考コードを実行する help, example 317
 オブジェクトの一覧とその構造の同時表示 ls.str, lsf.str 316
 オブジェクトの構造を見る str 314
 重複するオブジェクトを一覧する conflicts 315
 デモ demo, ビニエット vignette, タスクビュー Task View 318
 どんなオブジェクトがあるか ls 315
 必要な情報を探す RSiteSearch 318
 使用中の R・計算機に関する情報 319
 R のホームディレクトリに関する情報を得る R.home 322
 R のホームディレクトリを表示する R.home 321
 オプションを見る・設定する options, getOptions, .Options 320
 作業ディレクトリを得る・変える getwd(), setwd 321
 使用中の R に含まれる関数に関する情報 319
 使用中の R の機能の確認 capabilities, Platform 321
 使用中の計算機・OS・R のバージョン情報を得る sessionInfo 320
 その他 license, contributors, Sys.which 321
 その他 322
 R の関連ドキュメントを表示する RShowDoc 322
 必要な情報を探す 322
- 整数 第 3 章 16
 整数型オブジェクト 16
 整数値への変換 as.integer 17
 整数の判定 is.integer 16
 整数の表現範囲 17
 整数の表現 18
 整数値の 16 進数・8 進数表現 as.hexmode, as.octmode 18
- 整数値のビット毎の論理操作 151
 ファイルデータを登録する attach 153
 ベクトル・因子の全ての組み合わせからなるデータフレームを作る expand.grid 154
 文字列変数を因子化しない 156
 データフレームを用いた作業 148
 データから作られた環境中で R 表現式を評価する with 150
 データフレームを環境として登録・削除する attach, detach 148
 不要な水準を取り除く droplevels 151
- 特殊オブジェクト NA, NULL 第 10 章 65
 NA 値 65
 NA 値が含まれるかどうかの検査 is.na 66
 NA 値は論理型、様々な NA 値 65
 NA 値を置き換える x[is.na(x)] <- y 67
 NA 値を含む数値・論理演算 66
 NA 値を含むデータの事前処理 na.omit, na.fail 等 66
 NULL 値 67
 NULL 値を作成・検査する as.null, is.null 67
 属性を消す 属性 <- NULL 68
- 内部時計に関連する関数 第 26 章 265
 内部時計を利用した関数 265
 R の起動以来の経過時間 proc.time 265
 ガベージコレクションに必要だった時間 gc.time 266
 計算時間・R セッション継続時間を制限 setTimeLimit, setSessionLimit 266
 指定秒数アイドルング Sys.sleep 266
 プログラムの実行時間を計測 system.time 265
- 入出力 第 22 章 226
 R オブジェクトの整形 240
 R オブジェクトの引用符による文字列化 shQuote, sQuote, dQuote 241
 R オブジェクトの整形 encodeString 241
 R オブジェクトの整形 format 240
 R オブジェクトの整形 formatC 240
 R オブジェクトの整形
- 整数値のローマ数字表現 as.roman 18
- データフレーム 第 16 章 138
 アドオンパッケージ data.table 157
 データテーブルの操作の例 157
 データフレーム 138
 2 つのデータフレームを結合する merge 141
 2 つのデータフレームを横・縦に結合する cbind, rbind 142
 2 つのデータフレームを横に結合する 154
 経時観測データフレームを横長・縦長形式に相互変換 reshape 144
 ケースを加える・置き換える 147
 数値データフレームの行和・列和・行平均・列平均 colSums, rowSums, rowMeans, colMeans 143
 数値データフレームのグルーピングした列和 rowsum 143
 データフレームに対する 1 重鉤括弧演算子 [] 140
 データフレームの一部を取り出す subset 140
 データフレームを因子でグループ化し関数を適用 by 146
 データフレームを数値行列に変換 data.matrix 148
 データフレームを作る data.frame 138
 データフレームを変形する transform 139
 リスト成分ベクトルを因子に応じて整列化する stack, unstack 144
 データフレーム Tips 集 151
 エディタを使ったオブジェクトの編集 edit, fix, data.entry 155
 完全なケースだけを取り出す complete.cases 154
 定数値である変数を除く 154
 データフレームに新しい変数を加える 153
 データフレームに同一の変数ラベルを与える 155
 データフレームの重複するケースを検査する・取り除く duplicated, anyDuplicated, unique 153
 データフレームの変数成分を取り出す速度比較 155
 データフレームの変数名を簡略化する 152
 データフレームを吟味する

noquote.....241
 コネクション.....232
 コネクションを作る・開く・閉じる.....233
 その他.....234
 テキストコネクション **text connection**.....232
 データコードを読み込む・書き出す.....234
 R オブジェクトのテキスト表現 **dump**.....239
 R オブジェクトを保存する **save, save.image**.....238
 R コードをファイルやコネクションから読み込む **source**.....237
save 関数で保存されたデータセットを再読み込み **load**.....238
 データセットを読み込む、一覧表示する **data**.....238
 データをベクトルやリストに読み込む **scan**.....234
 テキストファイルに書き出す・読み出す **dput, dget**.....239
 幅固定欄ファイルでデータフレームにする **read.fwf**.....237
 表形式のファイルを読み込みデータフレームにする **read.table**.....235
 読み込んだファイルの欄数を数える **count.fields**.....237
 入出力 Tips 集.....242
for ループ中からのコンソール出力.....246
 R セッションの標準出力をファイルに落す.....242
 現在の作業環境をファイルにセーブし、それを次回に復元 **save.image**.....243
 コンソール出力を消す.....246
 様々な外部形式ファイルを R に読み込む.....243
 自前の関数定義を保存し、次回使えるようにする **save, source**.....243
 先頭文字を与えて一時ファイル名を作る **tempfile**.....245
 データのセーブとロード.....244
 複数のファイルを一度に読み込む.....244
 複数のファイルを一括してリストに読み込む.....244
 標準入出力.....226
 コンソールからの入力 **readline**.....229
 標準出力 **cat**.....226
 標準出力 **print**.....227
 標準出力 **show**.....227
 標準出力 **sprint, sprintf**.....228
 ファイルへの出力 **capture.output**.....232

ファイルへの出力 **sink**.....231
 ファイルへの出力 **write**.....230
 ファイル・ディレクトリ操作.....239
 ファイルやディレクトリを操作する関数.....239
 倍精度実数と浮動小数点数演算 第 4 章.....20
 実数値 (numeric, double).....20
 16 進数記法 **0xnnn**.....21
 組み込み実数定数 **pi**.....22
 実数の指数表記.....21
 実数への変換と判定 **is.numeric, is.double, as.numeric, as.double**.....20
 単精度属性 **Csingle** を持つ実数ベクトルの生成と検査 **single, as.single**.....22
 非数 **NaN** と正負の無限大 **Inf, -Inf**.....22
 実数の丸め・切り捨て.....27
 JIS, ISO, IEEE 式 (銀行型) 丸め.....27
 実数の丸めと切り捨て **round, ceiling, floor, trunc, signif, zapsmall**.....27
 浮動小数点数演算.....23
 桁落ち・桁溢れ・丸め誤差.....25
 倍精度浮動小数点数の形式.....24
 倍精度実数表現に対する IEC 60559 規則・R の組み込みリスト **.Machine**.....23
 範囲をカバーする等間隔数列 **pretty**.....26
 浮動小数点数演算の落とし穴.....26
 バイト型データとビット操作 第 9 章.....61
 バイト (raw) 型オブジェクト.....61
 バイト型・整数型データのビット列への変換 **rawToBits, intToBits**.....63
 バイト列を作る・変換する・検査する **raw, as.raw, is.raw**.....61
 バイト型データに対する操作.....62
 バイト型ベクトルのパターンマッチング **grepRaw**.....64
 バイト型データと文字列データ間の変換 **charToRaw, rawToChar**.....62
 バイト型データのビットシフト **rawShift**.....62
 バイト型ベクトルの圧縮と解凍 **memCompress, memDecompress**.....62
 ビット列をバイト型・整数型データへパックする **packBits**.....63

バイトコードコンパイラパッケージ **compiler** 第 31 章.....324
 バイトコードコンパイラ.....324
 JIT バイトコンパイル **enableJIT**.....326
 バイトコンパイル関数 **cmpfun, compile**.....324
 パッケージのバイトコンパイル **compilePKGS**.....326
 配列 第 14 章.....122
 配列 Tips 集.....128
 任意オフセットを持つ行列・配列を真似る.....130
 配列中の要素の並べかた.....128
 配列にダミー次元を加える.....129
 配列のマージン毎の一致検査.....129
 配列をベクトルとしてアクセスする.....129
 配列の生成と操作.....122
 1 重括弧演算子 [] による配列の要素の取り出し.....123
 1 重括弧演算子 [] による配列の要素の取り出し **drop=FALSE** オプション.....123
 2 重括弧演算子 [[] による配列の要素の取り出し.....123
 配列から長さ 1 の次元を取り除く **drop**.....127
 配列次元名の自動生成 **provideDimnames**.....124
 配列のあるマージンに関数を適用 **apply**.....126
 配列の一般化転置 **aperm**.....125
 配列の行和・列和・行平均・列平均 **colSums, rowSums, rowMeans, colMeans**.....127
 配列の次元 **dim, nrow, ncol, NROW, NCOL**.....125
 配列の次元に名前属性をつける **dimnames(x) <-**.....124
 配列の次元の変更 **dim(x) <-**.....125
 配列のスライス添字 **slice.index**.....125
 配列の生成 **array**.....122
 配列の重複するマージンを検査する・取り除く **deduplicated, anyDuplicated, unique**.....127
 配列のベクトル化とリスト化 **as.vector, as.list**.....128
 配列のマージンに周辺和を加える **addmargins**.....126
 論理値による配列の要素の取り出し・置き換え.....124

- パッケージ 第 24 章 252
 パッケージ 252
 パッケージのインストール・更新・削除 252
 パッケージの自動読み込み
 autoload 256
 パッケージの読み込み 254
 ブック 256
 パッケージに関する情報を得る 257
 インストール済のパッケージのパスを得る
 find.package, path.package 258
 インストール済パッケージに関する情報
 installed.packages 257
 デモ demo, ビニエット vignette, タスクビュー Task Views 258
 パッケージ中のデータの読み込み data 259
 パッケージ中のファイルに関する情報を得る
 system.file 257
 パッケージに関する情報を得る help(package=xxx) 257
 パッケージの作成
 package.skeleton 260
 複素数 第 5 章 29
 複素数の演算 30
 複素数の基本関数 Re, Im, Mod, Arg, Conj 30
 複素数の三角・逆三角関数, 双曲線・逆双曲線関数, 指数・対数関数 30
 複素数の四則演算・べき乗 30
 複素数の生成 29
 複素数表現 $x+yi$ 29
 複素数ベクトルの生成
 complex 29
 複素数ベクトルの生成 complex 複素数への変換・判定・比較 as.complex, is.complex 30
 付録 R を用いた講義デモ用の関数 337
 並列処理パッケージ parallel 第 32 章 327
 pqR 336
 クラスタの作成と操作 327
 R における並列処理機能 327
 sapply と mapply 関数の並列版 papSapply, clusterMap 330
 親プロセス中のオブジェクトを子プロセスに移出 clusterExport 329
 コア数の確認・クラスタ生成と終了 detectCores, makeCluster, stopCluster 328
 子プロセス中で特定のパッケージを利用する 333
 子プロセスの負荷の均衡 (load balancing) 334
 自動化並列計算関数 330
 自動化並列計算関数
 mclapply, mcmapply, mcMap 333
 自動化並列計算関数 pvec 330
 非同期的自動化並列計算
 mcpParallel, mccollect 332
 並列計算の時間計測 329
 子プロセスの擬似乱数 334
 L'Ecuycr の並列化擬似乱数発生器 RNGkind ("L'Ecuycr-CMRG") 335
 子プロセスの擬似乱数系列の設定 334
 ベクトル 第 12 章 77
 long vectors 90
 ベクトル Tips 集 91
 オブジェクトのシリアル化
 serialize, unserialize 93
 数値ベクトルの対話的入力 92
 数値ベクトルを表す文字列をベクトルに変換する 92
 ベクトルの内積・ノルムを求める 91
 ベクトル要素の順序を他のベクトルの要素の大小に応じて並べ替える 91
 ベクトルをリストに変換する 91
 リストの要素のベクトル化と再リスト化 unlist, relist 92
 ベクトルの添字操作 81
 因子によるベクトル要素の抽出 82
 条件に適合するベクトル要素の添字を得る which 82
 添字を用いたベクトルの部分代入の内部的機構 84
 名前ラベルによるベクトル要素の抽出 82
 名前属性を用いたベクトル要素の抽出 83
 ベクトルの一部を取り出す subset 84
 ベクトル要素の置き換えと括弧演算子 [] <- 81
 ベクトル要素の抽出と 2 重括弧演算子 [[]] 83
 ベクトル要素の抽出と括弧演算子 [] 81
 論理値ベクトルによるベクトルの添字操作 81
 ベクトルを操作する関数 85
 ある値が増加数列のどこにあるかを計算
 findInterval 89
 一意化 unique 86
 一部を置き換える replace 87
 関数の連続適用によるベクトル生成 sapply 90
 数値ベクトルの要約 summary, fivenum 89
 数値ベクトルを指定した区間に分類する cut 89
 重複する要素の添字を返す duplicated 86
 ベクトルから同じ数が引き続く回数とその数を取り出す rle, inverse.rle 86
 ベクトルの外積 outer 90
 ベクトルの大小順並べ替え添字ベクトル order, sort.list 88
 ベクトルの長さ length 85
 ベクトルの要素に因子グループ毎に関数を適用する
 ave 87
 ベクトル要素の逆転 rev 85
 要素のマッチング match 87
 要素を挿入 (付加) する
 append 87
 累積関数・差分 cumxxx, diff 85
 ベクトルを作る 77
 規則的なベクトルを作る
 numeric 77
 規則的なベクトルを作る rep 79
 規則的なベクトルを作る seq, sequence 78
 規則的なベクトルを作る コロ演算子 : 78
 長さとモードを指定してベクトルを作る vector, as.vector 79
 要素を結合してベクトルを作る c 77
 ランダムなベクトルをつくる sample, sample.int 80
 文字列とその操作 第 6 章 31
 原子オブジェクト文字列 (character) 31
 16 進数, 8 進数による文字表現 32
 Unicode による文字表現 33
 引用符なしの文字列出力 noquote 33
 組み込み文字列 letters, LETTERS, month.name, month.abb 32
 制御文字 \b, \n, \r 33
 整数値と文字エンコーディング 33
 文字列の生成と検査 character, is.character 31
 文字列への変換 as.character 32
 正規表現 (regular expression) 39

- Perl 式正規表現..... 41
 UNIX シェル風ワイルドカード指定を正規表現に変換する `glob2rx` 41
 拡張正規表現..... 39
 基本正規表現..... 41
 部分名でオブジェクトを探す `apropos`, `find` 41
 文字列のパターンマッチングと置き換え `grep`, `sub`, `gsub`, `regexpr`, `gregexpr`, `regexec` 42
 文字列 Tips 44
 制御文字を用いた `cat` 関数の出力の制御..... 44
 文字列化による R コードのコメント化..... 45
 文字列・ラベル..... 45
 ローケルと文字列のエンコーディング..... 46
 文字列操作とマッチング..... 34
 数値を表す文字列ベクトルを数値ベクトルに変換 `type.convert` 38
 文字数を数える `nchar` 37
 文字の変換 `chartr`, `tolower`, `toupper` 37
 文字列の結合 `paste`, `paste0` 36
 文字列の部分的マッチング `charmatch` 38
 文字列ベクトルからの対話的選択 `select.list` 38
 文字列ベクトルの一部の取り出し・置き換え `substr`, `substring` 35
 文字列を指定幅に切り詰める `strtrim` 35
 文字列を操作する関数ファミリー 34
 文字列をパターンにしたがって分解 `strsplit` 36
 リスト 第 15 章 131
 リスト Tips 集 135
 関数引数にリストとその成分名を与える 137
 関数リストの行列化 135
 データとその解析関数の一括リスト化 136
 リストに同一の変数ラベルを与える 137
 リストの成分名を関数引数で与える 136
 リストを行列に変換 137
 リストを生成する 131
 for ループのリスト範囲 134
 関数仮引数リスト `alist` 134
 関数のリスト返り値 134
 ベクトルをリストに変換 `as.list` 132
 リスト成分の消去 133
 リストの行列化 133
 リストの成分を取り出す [], [[]], \$ 132
 リストの連結 `c` 132
 リストを生成する `list`, `as.list`, `is.list` 131
 リストを操作する関数 134
 リストを再帰的に変更する `modifyList` 135
 リストをベクトルに変換 `unlist` 134
 暦日・時間 第 21 章 220
 暦日・時間用の関数 223
 date-time クラスオブジェクトと文字列間の変換 223
 ジュリアン通日 224
 時間差 `difftime` 223
 タイムゾーンとサマータイム 225
 暦日・時間用のクラス 220
 date クラス "Date" 220
 date-time クラス "POSIXlt" 221
 "POSIXct" 221
 date-time クラス "POSIXt" 221
 date-time クラス "POSIXt" 221
 date-time クラス "tzzone" 221
 date-time クラスオブジェクトと文字列間の変換 223
 date クラス "Date" 220
 double 20
 expression 282
 for 文の返り値 73
 for ループ中からのコンソール出力 246
 for ループのリスト範囲 134
 for ループの落とし穴 74
 if, ifelse 文の返り値 76
 if 文で数値を論理値として使う 76
 implicit generic 303
 JIS, ISO, IEEE 式 (銀行型) 丸め 27
 JIT バイトコンパイラ `enableJIT` 326
 L'Ecuyer の並列化擬似乱数発生器 `RNGkind` ("L'Ecuyer-CMRG") 335
 Lisp, Reduce 風構文を持つ関数 `Reduce`, `Filter`, `Find`, `Map`, `Negate`, `Position` 182
 long vectors 90
 name 282
 NA 値 65
 NA 値が含まれるかどうかの検査 `is.na` 66
 NA 値を 0 で一括置き換え 118
 NA 値を置き換える `x[is.na(x)] <- y` 67
 NA 値を含むデータの前処理 `na.omit`, `na.fail` 等 66
 NA 値を含む数値・論理演算 66
 NULL 値 65, 67
 NULL 値を作成・検査する `as.null`, `is.null` 67
 numeric 20
 real 20
 Perl 式正規表現 41
 pqr 336

- R_HOME/etc/Renviron.site247
 R_HOME/etc/Rprofile.site247
 R news を読む readNEWS319
 round-to-even method 27
 R オブジェクトの引用符による文字列化 shQuote, sQuote, dQuote241
 R オブジェクトの整形240
 R オブジェクトの整形 encodeString241
 R オブジェクトの整形 format240
 R オブジェクトの整形 formatC240
 R オブジェクトの整形 noquote241
 R オブジェクトの属性 (attribute) 13
 R オブジェクトのテキスト表現 dump239
 R オブジェクトの (内部) 保管モード mode, storage.mode, typeof 33
 R オブジェクトを保存する save, save.image238
 R キーワード集 3
 R 起動時のメモリ量の制御用のコマンドラインオプション262
 R コードの高速化と簡潔化 6
 R コードをファイルやコネクションから読み込む source237
 R セッションの標準出力をファイルに落す242
 R における並列処理機能327
 R のオブジェクト・命令を表す文字列を評価実行 eval(parse(text=...)) 193, 289
 R の関連ドキュメントを表示する RShowDoc322
 R の起動以来の経過時間 proc.time265
 R の起動・終了247
 R の組み込みオブジェクトのヘルプ文章を見る, 参考コードを実行する help, example317
 R の組み込みリスト .Machine23
 R の構文を引数として関数に渡す189
 R のホームディレクトリに関する情報を得る R.home322
 R のホームディレクトリを表示する R.home321
 R の予約語 163, 292
 R 表現式を指定環境で評価する eval, evalq, local287
 R 風のコーディングスタイル 8
 S3 クラス295
 S3 グループ総称的関数とグループメソッド308
 S3 総称的関数303
 S4 クラス295
 S4 クラスでの S3 クラスの利用 setClass, setOldClass301
 S4 クラスの定義296
 S4 グループ総称的関数とグループメソッド309
 S4 総称的関数 standardGeneric, setGeneric303
 S4 メソッド306
 S4 メソッド選択適用 standardGeneric303
 supply と mapply 関数の並列版 papSupply, clusterMap330
 save 関数で保存されたデータセットを再読み込み load238
 statistician's rounding 27
 symbol282
 Unicode による文字表現 33
 UNIX シェル風ワイルドカード指定を正規表現に変換する glob2rx 41
 UTF-8 33
 vcell262
 XDR バイナリ書式 238, 244
 異なるループ変数による 3 ループ74
 Unicode 33
 アスキー文字 32
 与えられた名前の変数が存在するか? exists279
 アドオンパッケージ Oarray130
 アドオンパッケージ data.table157
 アドオンパッケージ evaluate337
 ある値が増加数列のどこにあるかを計算 findInterval 89
 ある条件に適合する行列要素の添字を得る which103
 暗黙の返り値172
 移出 (export)329
 一意化 unique 86
 一部を置き換える replace 87
 因子 56
 因子化された数値ベクトルを元に戻す 60
 因子化を避ける I 60
 因子グループ毎に関数を適用 tapply211
 因子水準の並べ替え reorder 59
 因子によるベクトル要素の抽出 82
 因子の水準数 nlevels 58
 因子の水準属性 levels 58
 因子の添字操作 59
 インストール済のパッケージのパスを得る find.package, path.package258
 インストール済パッケージに関する情報 installed.packages257
 隠蔽 (mask)268
 インポート (import)268
 引用符で囲む必要がある関数名163
 引用符なしの文字列出力 noquote 33
 閏秒 .leap.seconds221
 永続付値 <-、>-277
 永続付値を使って関数中の変数をチェックする177
 エスケープシーケンス 33
 エディタを使ったオブジェクトの編集 edit, fix, data.entry155
 エラーが起きても中断しない try192
 エラー処理 stopifnot176
 エラー処理 stop175
 エラーメッセージの出力を抑制する suppressWarnings191
 エンクロージャ (enclosure)267
 円周率 pi 22
 大きな行列の一部を見る117
 オブジェクト .BaseNamespaceEnv273
 オブジェクト検査・変換関数 15
 オブジェクト属性の確認・付加・変更 attr, attributes294
 オブジェクトの一覧とその構造の同時表示 ls.str, lsf.str316
 オブジェクトの一致判断 51
 オブジェクトのメモリサイズ object.size264
 オブジェクトの内部的コピー状況 tracemem263
 オブジェクトの構造を見る str314
 オブジェクトのシリアル化 serialize, unserialize 93
 オブジェクト名にマッチする関数を返す関数 match.fun186
 オブジェクトをグループに分けて要約する aggregate212
 オプションを見る・設定する options, getOptions, .Options320
 親プロセス (master process)327
 親プロセス中のオブジェクトを子プロセスに移出 clusterExport329
 下位クラス (sub-class)299
 外部形式ファイル243
 返り値の活用174
 返り値の指定 return172
 返り値のベクトル化174
 隠し変数 .Last.value292
 拡張正規表現 39
 確率分布200

囲み (親) 環境 (enclosing environment) 267
 ガベージコレクション関数 gc 261
 ガベージコレクションに必要な
 だった時間 gc.time 266
 仮引数 164
 仮引数のマッチング match.arg 167
 仮引数名の入力省略 168
 仮引数名は既存変数名と同じでも
 良い 166
 仮引数リストを取り出す
 substitute 168
 仮引数リストを取り出す・設定
 する formals, args 166
 環境 .GlobalEnv 273, 296
 環境 (environment) 267
 環境・検索パス・スコープ規則
 267
 環境中の変数に関数を適用
 eapply 214, 281
 環境中の変数の抽出と設定
 [[,], \$ 279
 環境にリスト成分を付値
 list2env 281
 環境の付加 (attach) 273
 環境パッケージ .AutoloadEnv
 256
 環境へアクセスするための関数
 ファミリー 274
 環境への付値 assign 278
 環境変数 250
 環境変数 HOME 250
 環境変数 LC_ALL 250
 環境変数 R_DEFAULT_PACKAGES
 247
 環境変数 R_ENVIRON 247
 環境変数 R_ENVIRON_USER 247
 環境変数 R_HISTFILE 247
 環境変数 R_HOME 250
 環境変数 R_PLATFORM 250
 環境変数 R_PROFILE 247
 環境変数 R_PROFILE_USER 247
 環境変数 TZ 250
 環境変数の確認・設定 251
 環境を取得・設定・検査・作成
 する 273
 環境を操作する関数 273
 関数オブジェクトを返り値にする
 関数 175
 関数返り値 172
 関数仮引数リスト 164
 関数仮引数リスト alist 134
 関数構成要素を並べたリストから
 関数を作る as.function 191
 関数中の未定義オブジェクトを
 固定 local 186
 関数と実引数リストを与え関数
 呼び出しを作成 call, do.call 191
 関数内部での関数定義 171
 関数のエラー・終了処理 175
 関数の書き方の基本 162
 関数の仮引数 164

関数のクロージャ環境・評価環
 境 269
 関数の再帰的定義 Recall 171
 関数の実行速度計測
 system.time 185
 関数のソースコードを見る
 methods, getS3method,
 getMethod 187
 関数のデバッグ 177
 関数のリスト返り値 134
 関数の連続適用によるベクトル
 生成 sapply 90
 関数引数 271
 関数引数にリストとその成分名
 を与える 137
 関数引数の値渡し
 (call-by-value) 271
 関数引数の強制評価 (forcing)
 272
 関数引数の遅延評価 (lazy
 evaluation) 271, 272
 関数引数の評価を強制する
 force 291
 関数本体 170
 関数本体を取り出す・設定する
 body 171
 関数名 163
 関数名に関するその他注意 164
 関数名に使えない・好ましくな
 い名前 163
 関数呼び出し環境 170
 関数呼び出しの実行 do.call
 288
 関数リストの行列化 135
 関数を関数内で作る方法 190
 完全一致判断 51
 完全なケースだけを取り出す
 complete.cases 154
 完全なケース (NA 値を含まな
 い) の行だけを取り出す 120
 ガンマ関数ファミリー 197
 擬似乱数 198
 擬似乱数発生器 198
 規則的なベクトルを作る
 numeric 77
 規則的なベクトルを作る seq,
 sequence 78, 79
 規則的なベクトルを作る コロン
 演算子 : 78
 基礎パッケージ (base package)
 267
 既存関数の仮引数の省略時既定
 値を変更 190
 基底環境 (base environment)
 267
 基本正規表現 41
 基本統計処理関数ファミリー 206
 基本論理演算子 49
 奇妙な関数 丸括弧関数・波括弧
 関数 194
 行番号・列番号からなる行列を
 生成 col, row 111
 行列積・クロス積 crossprod,
 tcrossprod 112
 行列にその周辺和をつけ加えた
 行列を作る addmargins 110

行列に対する各種積 112
 行列に対する四則演算 107
 行列に変換する・行列かどうか
 検査する as.matrix,
 is.matrix 95
 行列の一部を取り出す subset
 105
 行列の一般化内積 120
 行列の各行の最大要素位置
 max.col 111
 行列の行和・列和・行平均・列
 平均 rowSums, colSums,
 rowMeans, colMeans 110
 行列のグラフィックス表示
 plot, matplot, symnum
 116
 行列の次元属性 (dim
 attribute) 98
 行列の次元属性 dim, nrow,
 ncol 98
 行列の次元にベクトルを適用
 sweep 109
 行列の次元名を機械的につける
 119
 行列の生成・検査, 行列への変
 換 94
 行列の全体としての同等性を検
 査 117
 行列の操作 107
 行列の要素の自乗の総和 118
 行列の要約 str, summary 97
 行列の列のスケール化 scale
 108
 行列・配列の外積 %%, outer
 112
 行列・配列の外積 %x%,
 kronexpr 113
 行列・配列のコンパクトな表示
 119
 行列は対称か isSymmetric 107
 行列要素の抽出 2 重括弧演算子
 [,] 104
 行列を逆対角線に関して反転す
 る 121
 行列をベクトル化する
 as.vector 100
 行列をベクトルとしてアクセス
 する as.vector 100, 101
 行列をベクトルに変換する 116
 行列要素の置き換え 括弧演算子
 [,] <- 102
 行列要素の抽出 括弧演算子 [,]
 101
 巨大配列 17
 距離行列 dist 114
 銀行型丸め 27
 空環境 (empty environment)
 267
 組み合わせ因子を作る
 interaction, : 58
 組み合わせ論的関数 201
 組み込み実数定数 pi 22
 組み込み文字列 letters,
 LETTERS, month.name,
 month.abb 32
 組み込みリスト .Machine 23
 クラス (class) 294

- クラス "noquote"241
 クラススロットのプロトタイプ300
 クラススロットの操作 298, 299
 クラス属性 (class attribute) 293, 294
 クラスタ (cluster)327
 クラスタの作成と操作327
 クラス定義の封印 sealClass296
 クラス定義の抹消 removeClass296
 クラスの拡張と継承299
 クラスの継承関係を検査 is, extends, setIs300
 クラスのスロット296
 繰り返し69
 繰り返しと条件判断 Tips 集 73
 ゲルベングによる行和
 rowsum110
 グループ総称的関数とグループ
 メソッド308
 クロージャ (closure) 環境269
 クロス集計 xtabs217
 警告メッセージ warning176
 計算機環境で中身が変わる関数
 の定義186
 計算時間・R セッション継続時
 間を制限 setTimeLimit,
 setSessionLimit266
 桁溢れ (overflow)25
 桁落ち (underflow)25
 経時 (longitudinal) 観測データ
 フレーム144
 経時観測データフレームを横
 長・縦長形式に相互変換
 reshape144
 ケースを加える・置き換える147
 言語オブジェクト282
 言語オブジェクトを操作する関
 数286
 現在の作業環境をファイルに
 セーブし、それを次回に復
 元 save.image243
 検索パス170, 273
 原子的な (atomic) オブジェク
 ト12
 コア数の確認・クラスタ生成と
 終了 detectCores,
 makeCluster, stopCluster328
 公開 (export)268
 コードのボトルネックの発見
 Rprof, summaryRprof187
 コネクション (connection)232
 コネクションを作る・開く・閉
 じる233
 子プロセス (child, worker
 process)327
 子プロセス中で特定のバッケー
 ジを利用する333
 子プロセスの擬似乱数334
 子プロセスの擬似乱数系列の設
 定334
 子プロセスの負荷の均衡 (load
 balancing)334
 コマンドラインオプション262
 コマンドラインオプション
 --no-environ247
 コマンドラインオプション
 --no-init-file247
 コマンドラインオプション
 --no-restore-data247
 コマンドラインオプション
 --no-restore-history247
 コマンドラインオプション
 --no-restore247
 コマンドラインオプション
 --no-site-file247
 コマンドラインオプション
 --vanilla247
 コンソールからの入力
 readline229
 コンソール出力を消す246
 コンテキスト (context)270
 再帰的 (recursive) なオブジェ
 クト12
 再帰的にリストに関数を適用
 rapply213
 作業ディレクトリを得る・変え
 る getwd(), setwd321
 作表関数216
 作表関数 Tips219
 サマータイム225
 様々な外部形式ファイルを R に
 読み込む243
 三角関数・hyperbolic 関数ファ
 ミリ197
 三角行列 lower.tri, upper.tri
 "difftime"106
 時間差 オブジェクト
 "difftime"223
 次元属性 (dim attribute)98,
 99
 次元属性を与えてベクトルを行
 列に変換する
 attr.attributes95
 次元名属性 (dimnames
 attribute)98, 99
 次元を与えて要素が全て 0 のベ
 クトルや行列を作る
 mat.or.vec96
 四捨五入27
 実行の中断 break71
 実数行列の誤差範囲内の同等
 性を検査117
 実数値 (numeric, double)20
 実数の指数表記21
 実数の丸めと切り捨て round,
 ceiling, floor, trunc,
 signif, zapsmall27
 実引数167
 実引数が存在するかどうか
 チェック missing169
 実引数の遅延評価 (lazy
 evaluation)167
 指定環境へソースコードを読み
 込む sys.source276
 指定秒数アイドリング
 Sys.sleep266
 自動化並列計算関数330
 自動化並列計算関数 mclapply,
 mcapply, mcMap333
 自動化並列計算関数 pvec330
 自前の関数定義を保存し、次回
 使えるようにする save,
 source243
 集合演算203
 終了処理 on.exit176
 出力書式指定の apply 関数
 vapply214
 ジュリアン通日 (Julian date)224
 順序に関する関数 sort, order,
 rank, xtfm204
 上位クラス (super-class)299
 条件が満たされる限り繰り返す
 while ループ70
 条件に適合するベクトル要素の
 添字を得る which82
 条件による分岐 if, if else 71
 条件を満たす添字ベクトルを得
 る which, arrayInd55
 使用中の R・計算機に関する情
 報319
 使用中の R に含まれる関数に関
 する情報319
 使用中の R の機能の確認
 capabilities, .Platform
 321
 使用中の計算機・OS・R のパー
 ジョン情報を得る
 sessionInfo320
 使用メモリ量の制御とプロファ
 イリング262
 省略時既定値つき仮引数165
 初期設定ファイル .Rprofile247
 初等数値関数197
 シンボル (symbol)282
 水準パターンを与えて因子を作
 る gl58
 数値データフレームの行和・列
 和・行平均・列平均
 colSums, rowSums,
 rowMeans, colMeans143
 数値データフレームのグルー
 ピングした列和 rowsum143
 数値データフレームを行列に変
 換する118
 数値としての TRUE, FALSE48
 数値に対する「正確な同等性検
 査関数」identical(x,y)
 25
 数値ベクトルに対する逐次処理
 関数202
 数値ベクトルの対話的入力92
 数値ベクトルの要約 summary,
 fivenum89
 数値ベクトルを表す文字列をベ
 クトルに変換する92
 数値ベクトルを指定した区間に
 分類する cut89
 数値を表す文字列ベクトルを数
 値ベクトルに変換
 type.convert38
 スカラー値関数のベクトル化190

スカラー引数を持つ関数をベクトル化 **Vectorize** 210
 スコープ規則 (scoping rule) 170, 270
 スコープ規則・レキシカルスコープ 270
 全てが真か? **all** 52
 全てが真であれば実行中断 **stopifnot** 53
 正規表現 39
 制御文字 `\b, \n, \r` 33
 制御文字を用いた **cat** 関数の出力の制御 44
 整数型オブジェクト 16
 整数行列の保管モードを整数型にする 116
 整数値と文字エンコーディング 33
 整数値の 16 進数・8 進数表現 **as.hexmode, as.octmode** 18
 整数値のビット毎の論理操作 19
 整数値への変換 **as.integer** 17
 整数の表現 18
 整数の表現範囲 17
 整数の判定 **is.integer** 16
 整数のローマ数字表現 **as.roman** 18
 静的スコープ (static scope) 270
 正の機械イプシロン (machine epsilon) **.Machine \$double.eps** 25
 選択実行 71
 先頭文字を与えて一時ファイル名を作る **tempfile** 245
 総称的関数 293
 総称的なプリミティブ・内部関数 313
 添字行列で行列要素を取り出す・変更する 103
 添字行列による行列要素の抽出 105
 添字を用いたベクトルの部分代入の内部的機構 84
 ソースコード・ファイルの処理 **removeSource, srcfile** 181
 属性 `<- NULL` 68
 属性を持つオブジェクトの生成 **structure** 96
 ソケット (socket) 327
 その他の仮引数 166
 粗・密行列 パッケージ **Matrix** 121
 対角行列 **diag** 106
 大局的環境 (global environment) 267
 対称行列の効率計算 118
 対数・指数関数ファミリー 197
 ダイナミックスコープ (dynamic scope) 270
 代入 284
 タイムゾーン属性 "tzzone" 225

他言語で書かれたサブルーチンの利用 183
 多重選択 **switch** 72
 多バイト文字 33
 単位 (chunk) 327
 単純繰り返し **repeat** ループ 70
 単精度属性 **Csingle** 22
 遅延評価 (lazy evaluation) 167
 超越関数 197
 重複した行・列の検査と一意化 **deduplicated, unique** 109
 重複するオブジェクトを一覧する **conflicts** 315
 重複する要素の添字を返す **deduplicated** 86
 直近の評価結果を記録する変数 **.Last.value** 291
 次の実行サイクルへ飛ぶ **next** 70
 次のメソッドを適用する **callNextMethod** 308
 定数値である変数を除く 154
 データから環境を作る **attach, with** 277
 データから作られた環境中で R 表現式を評価する **with** 150
 データセットを読み込む、一覧表示する **data** 238
 データテーブルの操作の例 157
 データとその解析関数の一括リスト化 136
 データのセーブとロード 244
 データフレームに新しい変数を加える 153
 データフレームに対する **tapply** 関数 **by** 212
 データフレームに対する 1 重鉤括弧演算子 `[]` 140
 データフレームに同一の変数ラベルを与える 155
 データフレームの一部を取り出す **subset** 140
 データフレームの重複するケースを検査する・取り除く **deduplicated, anyDuplicated, unique** 153
 データフレームの変数成分を取り出す速度比較 155
 データフレームの変数名を簡略化する 152
 データフレームを因子でグループ化し関数を適用 **by** 146
 データフレームを環境として登録・削除する **attach, detach** 148
 データフレームを吟味する 151
 データフレームを数値行列に変換 **data.matrix** 96, 148
 データフレームを作る **data.frame** 138
 データフレームを変形する **transform** 139

データ保管に使用するメモリ量の制御 262
 データやコードを読み込む・書き出す 234
 データをベクトルやリストに読み込む **scan** 234
 テキストから表現式を作る **parse** 289
 テキストコネクション **text connection** 232
 テキストファイルに書き出す・読み出す **dput, dget** 239
 デバッグ用関数 **browser** 178
 デバッグ用関数 **debug** 178
 デバッグ用関数 **dump.frames, debugger** 180
 デバッグ用関数 **recover** 179
 デバッグ用関数 **trace, traceback** 180
 デバッグ用引数を持つ関数 195
 デモ **demo**, ビニエツト **vignette**, タスクビュー **Task View** 318
 転置行列 **t** 105
 転置行列、対角・三角行列 105
 独自の S3 メソッドを定義する 305
 トップレベルの環境を見つける **topenv** 277
 どれかが真か? **any** 52
 どんなオブジェクトがあるか **ls** 315
 内部時計を利用した関数 265
 長さモードを指定してベクトルを作る **vector, as.vector** 79
 名前 (name) 282
 名前オブジェクトを作る **as.symbol** 287
 名前空間 268
 名前空間を直接操作する関数 **ファミリ** 280
 名前属性を用いたベクトル要素の抽出 83
 (引数の) 名前タグ 271
 名前つきスロット 296
 名前なしスロット **.Data** 296
 名前のない関数 164
 名前ラベルによるベクトル要素の抽出 82
 入出力 **Tips** 集 242
 任意オフセットを持つ行列・配列 130
 倍精度実数 20
 倍精度実数表現に対する IEC 60559 規則 23
 倍精度浮動小数点数の形式 24
 バイト (raw) 型オブジェクト 61
 バイト型・整数型データのビット列への変換 **rawToBits, intToBits** 63
 バイト型データとビット操作 61
 バイト型データと文字列データ間の変換 **charToRaw, rawToChar** 62

バイト型データに対する操作 62
 バイト型ベクトルの圧縮と解凍
 memCompress,
 memDecompress 62
 バイト型ベクトルのパターン
 マッチング grepRaw 64
 バイト型データのビットシフト
 rawShift 62
 バイトコードコンパイラ 324
 バイトコンパイラパッケージ
 compiler 324
 バイトコンパイル関数 cmpfun,
 compile 324
 バイト列を作る・変換する・検
 査する 61
 配列から長さ 1 の次元を取り除
 く drop 127
 配列次元名の自動生成
 provideDimnames 124
 配列中の要素の並べかた 128
 配列にタミー次元を加える 129
 配列のあるマージンに関数を適
 用 apply 126
 配列の一般化転置 aperm 125
 配列の行和・列和・行平均・列
 平均 colSums, rowSums,
 rowMeans, colMeans 127
 配列の次元 dim, nrow, ncol,
 NROW, NCOL 125
 配列の次元に名前属性をつける
 dimnames(x) <- 124
 配列の次元の変更 dim(x) <-
 125
 配列のスライス添字
 slice.index 125
 配列の生成 array 122
 配列の重複するマージンを検査
 する・取り除く
 duplicated,
 anyDuplicated, unique
 127
 配列のベクトル化とリスト化
 as.vector, as.list 128
 配列のマージン毎の一致検査
 129
 配列のマージンに関数を適用
 apply 207
 配列のマージンに周辺和を加え
 る addmargins 126
 配列をベクトルとしてアクセス
 する 129
 バックチェック記号 ‘ 45, 163,
 189
 パッケージ 252
 パッケージに関する情報を得る
 help(package=xxx) 257
 パッケージに含まれるメソッド
 の扱い 305
 パッケージのインストール・更
 新・削除 252
 パッケージの作成
 package.skeleton 260
 パッケージの自動読み込み
 autoload 256
 パッケージのバイトコンパイル
 compilePKGS 326

パッケージ中のデータの読み込
 み data 259
 パッケージ中のファイルに関す
 る情報を得る system.file
 257
 パッケージ中の変数にアクセス
 する, 2重・3重コロン演
 算子 ::, ::: 281
 パッケージの読み込み 254
 バッチ処理 249
 幅固定欄ファイルをデータフ
 レームにする read.fwf
 237
 範囲に渡って繰り返す for ル
 ープ 69
 範囲をカバーする等間隔数列
 pretty 26
 万能デバッグ関数 cat 177
 非アスキー文字 33
 非数 NaN 22
 引数をフルネームで置き換えた
 呼び出しを返す
 match.call 290
 ビット列をバイト型・整数型
 データへバックする
 packBits 63
 必要な情報を探す 322
 必要な情報を探す RSiteSearch
 318
 非同期的自動化並列計算
 mcpParallel, mcCollect
 332
 評価環境 (evaluation
 environment) 269
 表形式のファイルを読み込み
 データフレームにする
 read.table 235
 表現式 (expression) 282
 表現式オブジェクトを作る
 expression 287
 表現式を文字列に変換 deparse
 291
 標準出力 cat 226
 標準出力 print 227
 標準出力 show 227
 標準出力 sprint, sprintf 228
 標準入出力 226
 ファイル .Rdata 247
 ファイル .Renvirom 247
 ファイル .Rhistory 247
 ファイル .Rprofile 247
 ファイル・ディレクトリ操作
 239
 ファイルデータを登録する
 attach 153
 ファイルへの出力
 capture.output 232
 ファイルへの出力 sink 231
 ファイルへの出力 write 230
 ファイルやディレクトリを操作
 する関数 239
 フォーク (fork) 327
 不可視戻り値 invisible 173
 複数回の計算結果をリストで返
 す lapply, sapply,
 replicate 209

複数のファイルを一度に読み込
 む 244
 複数のファイルを一括してリス
 トに読み込む 244
 複数のベクトル・行列をつなげ
 て行列を作る rbind, cbind
 97
 複数次数に関数を多重適用
 mapply, Map 210
 複数次数の要素の全ての組
 み合わせからなる行列 119
 複素数対数関数の主値 30
 複素数の演算 30
 複素数の基本関数, Re, Im, Mod,
 Arg, Conj 30
 複素数の三角・逆三角関数, 双
 曲線・逆双曲線関数, 指
 数・対数関数 30
 複素数の四則演算・べき乗 30
 複素数の生成 29
 複素数表現 x+yi 29
 複素数ベクトルの生成 complex
 29
 符号・絶対値・平方根 202
 付値演算子 <- と = 194
 フック 256
 浮動小数点数演算 20, 23
 浮動小数点数演算の落とし穴 26
 負の機械イプシロン (machine
 negative epsilon)
 .Machine
 \$double.neg.eps 25
 部分名でオブジェクトを探す
 apropos, find 41
 不要な水準を取り除く
 droplevels 151
 フラットな集計表 ftable 218
 フラットな分割表の読み書き
 write.ftable,
 read.ftable 218
 プリミティブな関数かどうかを
 検査する is.function,
 is.primitive 181
 フレーム (frame) 267
 プログラムの実行時間を計測
 system.time 265
 分割表 table, tabulate 216
 並列計算 327
 並列計算の時間計測 329
 ベクトル Tips 集 91
 ベクトル・因子の全ての組み合
 わせからなるデータフレー
 ムを作る expand.grid
 154
 ベクトル化条件分岐 ifelse
 53, 71
 ベクトルから同じ数が引き続く
 回数とその数を取り出す
 rle, inverse.rle 86
 ベクトルから上・下三角行列を
 作る 121
 ベクトル成分の長さ 3 の全ての
 順列・組み合わせに関して
 ループする 74
 ベクトルの一部を取り出す
 subset 84
 ベクトルの外積 outer 90

- ベクトルの行列への効率的な変形..... 95
- ベクトルの添字操作..... 81
- ベクトルの大小順並べ替え添字ベクトル `order`, `sort.list`..... 88
- ベクトルの内積・ノルムを求める..... 91
- ベクトルの長さ `length`..... 85
- ベクトルの要素に因子グループ毎に関数を適用する `ave`..... 87
- ベクトル要素の逆転 `rev`..... 85
- ベクトル要素の置き換えと括弧演算子 [] <-..... 81
- ベクトル要素の順序を他のベクトルの要素の大小に応じて並べ替える..... 91
- ベクトル要素の抽出と2重括弧演算子 [[]]..... 83
- ベクトル要素の抽出と括弧演算子 []..... 81
- ベクトル・リストに次元属性を与えて行列を作る `dim`..... 99
- ベクトルを因子にする `factor`..... 56
- ベクトルを操作する関数..... 85
- ベクトルを作る..... 77
- ベクトルをリストに変換 `as.list`..... 132
- ベクトルをリストに変換する..... 91
- ベッセル関数ファミリー..... 198
- 変数 `Autoloaded`..... 256
- 変数 `libPaths`..... 254
- 変数 `Library`..... 254
- 変数を指定環境中で探し、その値を返す `get`, `mget`..... 279
- ほとんど等しいか? `all.equal`, `attr.all.equal`..... 52
- マルチコア CPU..... 327
- 丸め関数..... 202
- 丸め誤差 (rounding error)..... 25, 27
- 未評価の関数呼び出し `call`..... 191
- 未評価の関数呼び出しを実行 `do.call`..... 191
- 未評価の関数呼び出しを実行 `eval`..... 191
- 無限大 `Inf`, `-Inf`..... 22
- 命令 R CMD BATCH..... 249
- メソッド (method)..... 293
- メソッド選択適用 (method dispatch)..... 293
- メソッドを処理する関数..... 311
- メソッドを処理する関数ファミリー..... 306
- メタ文字..... 41
- メッセージを作る・抑制する `message`, `suppressMessages`..... 176
- メニューによる選択 `menu`..... 73
- メモリア量の使用状況 `Rprofmem`..... 263
- 文字の変換 `chartr`, `tolower`, `toupper`..... 37
- 文字列化による R コードのコメント化..... 45
- 文字列行列..... 98
- 文字列操作とマッチング..... 34
- 文字列のエンコーディング..... 46
- 文字列の結合 `paste`, `paste0`..... 36
- 文字列の生成と検査 `character`, `is.character`..... 31
- 文字列のパターンマッチングと置き換え `grep`, `sub`, `gsub`, `regexpr`, `gregexpr`, `regexec`..... 42
- 文字列的部分のマッチング `charmatch`..... 38
- 文字列ベクトルからの対話的選択 `select.list`..... 38
- 文字列ベクトルの一部の取り出し・置き換え `substr`, `substring`..... 35
- 文字列への変換 `as.character`..... 32
- 文字列変数を因子化しない..... 156
- 文字列・ラベル..... 45
- 文字数を数える `nchar`..... 37
- 文字列を指定幅に切り詰める `strtrim`..... 35
- 文字列を操作する関数ファミリー..... 34
- 文字列パターンにしたがって分解 `strsplit`..... 36
- モデル式とチルダ演算子 ~..... 169
- ユニオンクラス `setClassUnion`..... 301
- 要素全てが 0 (全てが 1) の行列を作る..... 117
- 要素のマッチング `match`..... 87
- 要素を結合してベクトルを作る `c`..... 77
- 要素を挿入 (付加) する `append`..... 87
- 呼び出し (call)..... 282
- 呼び出しオブジェクトを作る `call`..... 286
- 呼び出しの即時実行 `do.call`..... 286
- 読み込んだファイルの欄数を数える `count.fields`..... 237
- 予約オブジェクト (promise object)..... 272, 284
- ランダムなベクトルをつくる `sample`, `sample.int`..... 80
- リスト `.Options`..... 320
- リスト `S3PrimitiveGenerics`..... 313
- リスト行列..... 98
- リスト成分の消去..... 133
- リスト成分ベクトルを因子に応じて整列化する `stack`, `unstack`..... 144
- リストに同一の変数ラベルを与える..... 137
- リストの各要素に指定関数を適用した結果をベクトル・行列の形で返す `sapply`..... 109
- リストの行列化..... 133
- リストの成分を取り出す [], [[]], \$..... 132
- リストの成分名を関数引数で与える..... 136
- リストの要素のベクトル化と再リスト化 `unlist`, `relist`..... 92
- リストの連結 `c`..... 132
- リスト・ベクトル返り値..... 173
- リストを再帰的に変更する `modifyList`..... 135
- リストを生成する..... 131
- リストを生成する `list`, `as.list`, `is.list`..... 131
- リストを行列に変換..... 137
- リストをベクトルに変換 `unlist`..... 134
- 累積関数・差分 `cumxxx`, `diff`..... 85
- レキシカルスコープ (lexical scope)..... 270
- 暦日・時間用の関数..... 223
- 暦日・時間用のクラス..... 220
- ロケール (locale)..... 46
- 論理演算..... 49
- 論理演算の優先度..... 49
- 論理型の NA 値..... 65
- 論理値 NA..... 47
- 論理値 TRUE, FALSE..... 47
- 論理値行列による行列の添字操作..... 102
- 論理値添字ベクトルの利用..... 54
- 論理値による配列の要素の取り出し・置き換え..... 124
- 論理値の省略形 T, F..... 48
- 論理値ベクトルによるベクトルの添字操作..... 81
- 関数索引
 - \$..... 132, 133
 - %%..... 196
 - %*%..... 112
 - %/%..... 196
 - %in%..... 203
 - %x%..... 113
 - *..... 196
 - -..... 196
 - ->..... 277
 - 166
 - .C..... 183
 - .Call..... 183
 - .Call.graphics..... 183
 - .colMeans..... 110
 - .colSums..... 110
 - .External..... 183
 - .External.graphics..... 183
 - .find.package..... 254
 - .First..... 247
 - .First.lib..... 254
 - .First.sys..... 247
 - .Fortran..... 183
 - .Internal..... 319
 - .knownS3Generics..... 305
 - .Last..... 247
 - .Last.lib..... 254
 - .Last.sys..... 247
 - .makeMessage..... 176
 - .OldClassesList..... 295
 - .OptRequireMethods..... 247
 - .packages..... 252, 254

- .Platform 319, 321
- .Random.seed 198
- .rowMeans 110
- .rowSums 110 313
- / 196
- : 58
- :: 281
- ::: 281
- <- 194
- <<- 277
- = 194
- & 49
- isTRUE 49
- xor 49
- && 49
- ! 49
- != 49
- <= 49
- < 49
- >= 49
- > 49
- >= 49
- | 49
- || 49
- @ 298
- [] 59, 81, 82, 123, 132, 133, 140
- [,] 101, 102
- [[]] 81, 83, 104, 123, 132, 133, 140
- ‘ 189
- ~ 169
- : 78
- "(" 194
- "f" 194
- "if" 194
- autoload 256
- available.packages 252
- contrib.url 252
- download.packages 252
- getHook 256
- abbreviate 34, 152
- abs 202
- acos 30, 197
- acosh 30, 197
- addmargins 110, 126, 219
- addNA 56
- adist 34
- aggregate 212
- agrep 34
- alist 134, 166
- all 52, 117
- all.equal 52, 117
- any 52, 117
- anyDuplicated 127, 153
- aperm 125
- append 87
- apply 120, 126, 207
- apropos 41, 322
- aregexec 34
- Arg 30
- args 166
- array 122
- arrayInd 55
- as.call 282, 286
- as.character 32
- as.complex 30
- as.data.frame 138, 216
- as.Date 220
- as.difftime 223
- as.double 20
- as.expression 287
- as.formula 169
- as.function 191
- as.hexmode 18
- as.integer 17
- as.list 128, 131, 132, 282
- as.logical 47, 49
- as.matrix 94, 95
- as.name 282, 287
- as.null 67
- as.numeric 20
- as.octmode 18
- as.POSIXct 221
- as.POSIXlt 221
- as.raw 61
- as.relistable 92
- as.roman 18
- as.single 22
- as.symbol 287
- as.table 216
- as.vector 79, 100, 128
- as.xxx 15
- asin 30, 197
- asinh 30, 197
- asS3 296
- asS4 296
- assign 278
- atan 30, 197
- atan2 197
- atanh 30, 197
- attach 148, 153, 277
- attachNamespace 280
- attr 13, 95, 294
- attr.all.equal 52
- attributes 13, 95, 294
- available.views 258, 318
- ave 87
- baseenv 319
- basename 239
- besselI 198
- besselJ 198
- besselK 198
- besselY 198
- beta 197, 200
- binom 200
- birthday 200
- bitAnd 19
- bitwNot 19
- bitwOr 19
- bitwShiftL 19
- bitwShiftR 19
- bitwXor 19
- body 171
- bquote 282
- break 70, 71
- browser 178
- builtins 319, 322
- by 146, 212
- c 77, 132
- call 191, 286
- callNextMethod 308
- capabilities 321
- capture.output 232
- casefold 37
- cat 177, 226, 246
- cauchy 200
- cbind 97, 142
- cbind2 97
- ceiling 27, 202
- char.expand 34
- character 31, 34
- charmatch 34, 38
- charToRaw 62
- chartr 34, 37
- children 327
- chisq 200
- choose 201
- close 233
- closeAllConnections 234
- clusterApply 327
- clusterApplyLB 327, 334
- clusterCall 327
- clusterEvalQ 327
- clusterExport 327, 329, 333
- clusterMap 327, 330, 334
- clusterSetRNGStream 335
- clusterSplit 327
- cmpfile 324
- cmpfun 324
- col 111
- colMeans 110, 127, 143
- colnames 99
- colSums 110, 127, 143
- combn 74, 201
- commandArgs 247
- compile 324
- compilePKGS 324, 326
- complete.cases 120, 154
- complex 29
- conflicts 315
- Conj 30
- contributors 321
- cor 206
- cos 30, 197
- cosh 30, 197
- count.fields 237
- cov 206
- crossprod 112
- cummax 85, 202
- cummin 85, 202
- cumprod 85, 202
- cumsum 85, 202
- cut 89
- cut.Date 220
- data 238, 259
- data.entry 155

- data.frame 138, 154
- data.matrix 96, 118, 148
- Date 220
- debug 178
- debugger 180
- delayedAssign 272
- delimMatch 34
- Demo 337
- Demo0 337
- DemoAux 337
- deparse 284, 291
- detach 148, 277
- detectCores 327, 328
- dget 239
- diag 106, 117
- diff 85, 206
- difftime 221, 223
- digamma 197
- dim 98, 99, 125
- dimnames 99, 124
- dir 239
- dir.create 239
- dirname 239
- disassemble 324
- do.call 191, 286, 288
- download.file 239
- dput 239
- dQuote 241
- drop 127
- dropLevels 151
- dump 239
- dump.frames 180
- dumpMethod 306
- dumpMethods 306
- duplicated
..... 86, 109, 127, 153
- dyn.load 183
- dyn.unload 183
- eapply 214, 281
- edit 155
- emacs 155
- emptyenv 267
- enableJIT 326
- enc2native 46
- enc2utf8 46
- encoded_text_to_latex
..... 33
- encodeString 46, 241
- Encoding 34, 46
- environment 270, 273
- eval
... 191, 193, 282, 287, 289
- eval.parent 193, 289
- evalq 287
- evaluate 337
- example 317
- exists 279
- existsMethod 311
- exp 30, 197, 200
- expand.grid 119, 154
- expm1 197
- expression 287
- extends 300
- F 48
- f 200
- factor 56
- factorial 201
- FALSE 48
- file 233
- file.access 239
- file.append 239
- file.choose 239
- file.copy 239
- file.create 239
- file.exists 239
- file.info 239
- file.link 239
- file.path 239
- file.remove 239
- file.rename 239
- file.show 239
- file.symlink 239
- file_test 239
- Filter 182, 319
- Find 182
- find 41, 322
- find.package 258
- findClass 296
- findFunction 306
- findInterval 89
- findMethod 311
- fivenum 89, 206
- fix 155
- floor 27, 202
- flush 233
- for 69, 74, 134, 246
- force 167, 291
- formals 166
- format 34, 240
- format.Date 220
- format.info 34, 240
- formatC 34, 240
- formula 169
- ftable 218
- functionBody 171
- gamma 197, 200
- gc 185, 261
- gc.time 261, 266
- gcinfo 261
- gctorture 261
- geom 200
- get 279
- getAnywhere 268
- getClass 296
- getClasses 296
- getCompilerOption 324
- getConnection 234
- getElement 81, 82
- geterrmessage 175
- getGenerics 306
- getMethod 187, 311
- getMethods 311
- getOption 320
- getS3method 187
- getSrcLines 181
- gettext 34
- gettextf 228
- getwd 239, 321
- gl 58
- glob2rx 34, 41
- gregexpr 42
- grep 34, 42
- grepRaw 42, 64
- gsub 42
- hasMethod 311
- head 117
- help 257, 317, 322
- help.search 322
- help.start 317
- hist.Date 220
- hyper 200
- I 60, 156, 169
- iconv 34, 46
- iconvlist 46
- identical 51, 117
- identity 194
- if 71, 76
- if else 71
- ifelse 53, 71, 76
- Im 30
- install.views 258, 318
- install.packages 252
- installed.packages 252, 254, 257
- interaction 58
- intersect 203
- intToBits 63
- intToUtf8 33
- inverse.rle 86
- invisible 173, 246
- IQR 206
- is 300
- is.atomic 12
- is.call 286
- is.character 31
- is.complex 30
- is.double 20
- is.element 203
- is.environment 273
- is.expression 287
- is.function 181
- is.integer 16
- is.language 282
- is.list 131
- is.loaded 183
- is.logical 47, 49
- is.matrix 94, 95
- is.na 66, 67
- is.name 287
- is.null 67
- is.numeric 20
- is.primitive 181
- is.raw 61
- is.recursive 12
- is.relistable 92
- is.single 22
- is.symbol 287
- is.table 216
- is.xxx 15
- isClass 296
- isClassUnion 301

- isGeneric306
- isGroup306
- ISOdate 221, 223
- ISOdatetime 221, 223
- isS4296
- isSymmetric107
- isTRUE 51
- julian224
- kronecker113
- l10n_info 46
- lapply209
- lbeta197
- LC_NUMERIC 46
- lchoose201
- length85, 90
- levels 58
- lfactorial201
- lgamma197
- library 254, 322
- license321
- list131
- list.files239
- list2env281
- lnorm200
- load238
- loadcmp324
- loadedNamespaces280
- loadNamespace280
- local 186, 247, 287
- log 30, 197
- log10 30, 197
- log1p197
- log2 30
- logb197
- logical 48
- logis200
- lower.tri95, 106, 121
- ls315
- ls.str316
- lsf.str316
- mad206
- make.names 34
- make.unique 34
- makeCluster 327, 328, 333
- Map 182, 210
- mapply210
- margin.table219
- mat.or.vec 96, 117
- match 87
- match.arg167
- match.call290
- match.fun 120, 186
- matplot116
- matrix 94
- max206
- max.col111
- mc.reset.stream335
- mcaffinity327
- mccapply333
- mcollect332
- mcfork327
- mclapply 327, 333-335
- mcMap333
- mcparallel 327, 332, 334, 335
- mean206
- median206
- memCompress 62
- memDecompress 62
- memory.profile263
- menu 73
- merge141
- message176
- methods187
- MethodsListSelect311
- mget279
- min206
- missing169
- Mod 30
- mode 13
- modifyList135
- mostattributes294
- multinom200
- NA 65
- na.fail 66
- na.omit 66
- NA_integer_ 65
- NA_character_ 65
- NA_complex_ 65
- NA_real_ 65
- namespace268
- nbinom200
- nchar 34, 37
- NCOL125
- ncol 98, 125
- Negate182
- new 296, 306
- new.env 267, 273
- new.packages252
- next 70
- nextRNGStream 327, 335
- nextRNGSubStream335
- nlevels 58
- noquote 33, 241
- norm200
- normalizePath239
- NROW125
- nrow 98, 125
- NULL 67, 133
- numeric 77
- nzchar 37
- object.size264
- objects315
- old.packages252
- on.exit176
- open233
- options 23, 247, 320
- order 88, 204
- ordered 56
- outer 90, 112
- packageEvent256
- packBits 63
- page117
- papSapply330
- parallelLB334
- parApply327
- parCapply327
- parent.env273
- parent.frame274
- parLapply327
- parLapplyLB327
- parRapply327
- parSapply 327, 333
- parSapplyLB327
- parse 193, 289
- paste 34, 36, 119
- paste0 34, 36
- path.expand239
- path.package 254, 258
- pi 22
- pico155
- plot 116, 151
- plot.Date220
- pmatch 34
- pmax206
- pmin206
- pois200
- Position182
- pretty 26
- prettyNum240
- print 227, 246
- proc.time 265, 329
- prohibitGeneric303
- prop.table219
- provideDimnames124
- psigamma197
- pushBack234
- pvec 327, 330, 334
- q247
- quantile206
- quit247
- quote 282, 288
- R.home 321, 322
- r2dtable200
- range206
- rank204
- rapply213
- raw 61
- rawShift 62
- rawToBits 63
- rawToChar 62
- rbind 97, 142
- rbind2 97
- Re 30
- read.csv235
- read.csv2235
- read.delim235
- read.delim2235
- read.ftable218
- read.fwf237
- read.table235
- readBin234
- readChar234
- readline 92, 229
- readNEWS319
- Recall171
- recover179
- Reduce182
- reg.finalizer 247, 261
- regex 34

- regexec 42
- regexpr 42
- regmatches 34
- relist 92
- removeClass 296
- removeGeneric 306
- removeMethods 306
- remove.packages 252, 254
- removeSource 181
- Renviron.site 247
- reorder 59
- rep 79
- rep.int 79
- replace 87, 118
- replay 337
- replicate 209
- require 254, 258
- resetClass 296
- reshape 144
- return 172
- rev 85
- rle 86
- rMWC1019 200
- RNGkind 198, 335
- RNGversion 198, 200
- round 27, 202
- round.Date 220
- row 111
- rowMeans 110, 127, 143
- rownames 99
- rowsum 110, 143
- rowSums 110, 127, 143
- Rprof 187
- Rprofile.site 247
- Rprofmem 263
- RShowDoc 322
- RSiteSearch 318
- Rsitesearch 322
- sample 80, 200
- sample.int 80, 200
- sapply 90, 109, 209
- save 153, 238, 243, 244
- save.image 238, 243
- savehistory 247
- scale 108
- scan 234
- sd 206
- sealClass 296
- search 273
- searchpaths 273
- seek 234
- select.list 38
- selectMethod 311
- seq 74, 78, 221
- seq.Date 220
- seq.int 78
- sequence 78
- serialize 93
- sessionInfo 320
- set.seed 185, 198, 200
- setClass 296, 301, 306
- setClassUnion 301
- setCompilerOptions 324
- setdiff 203
- setequal 203
- setGeneric 303
- setGenericImplicit 303
- setGroupGeneric 303
- setHook 256
- setIs 300
- setkey 157
- setkeyv 157
- setMethod 335
- setOldClass 301
- setReplaceMethod 306
- setSessionLimit 266
- setTimeLimit 266
- setwd 321, 239
- show 227
- showConnections 234
- showMethods 308, 311
- shQuote 241
- sign 202
- signature 306
- signif 27, 202
- signrank 200
- sin 30, 197
- sinh 30, 197
- sink 231, 246
- sink.number 231
- slice.index 125
- slot 298
- sort 204
- sort.list 88, 204
- source 237, 243
- splitIndices 327
- sprint 228
- sprintf 34, 228
- sqrt 202
- sQuote 34, 241
- srcfile 181
- srcfilealias 181
- srcfilecopy 181
- srcref 181
- stack 144
- standardGeneric 303
- stderr 234
- stdin 234
- stdout 234
- stop 175
- stopCluster 328, 333
- stopifnot 53, 176
- storage.mode 13, 17, 116
- str 97, 98, 151, 314
- strftime 223
- strptime 221, 223
- strsplit 34, 36
- strtol 34
- strtrim 34, 35
- structure 96
- strwidth 34
- strwrap 34
- sub 42
- subset 84, 105, 140
- substitute 168, 272, 284, 288
- substr 34, 35
- substring 35
- summary 89, 97, 98, 151
- summaryRprof 187
- SuppDists 200
- suppressMessages 176
- suppressPackageStartupMessages 176
- suppressWarnings 191, 246
- sweep 109
- switch 72
- symnum 34, 116
- sys.call 274
- sys.calls 274
- Sys.chmod 239
- Sys.Date 220
- sys.frame 274
- sys.frames 274
- sys.function 274
- Sys.getenv 251
- Sys.getlocale 46, 251
- sys.nframe 274
- sys.on.exit 274
- sys.parent 274
- sys.parents 274
- Sys.readlink 239
- Sys.setenv 251
- Sys.setFileTime 239
- Sys.setlocale 46, 251
- Sys.sleep 265
- sys.source 276
- sys.status 274
- Sys.time 221
- Sys.umask 239
- Sys.unsetenv 251
- Sys.which 321
- system.file 257
- system.time 185, 261, 265, 329
- T 48
- t 105, 200
- table 216
- tabulate 216
- tail 117
- tan 30, 197
- tanh 30, 197
- tapply 211
- Task View 318
- tcrossprod 112
- tempdir 245
- tempfile 245
- terms 169
- textConnection 232
- tolower 37
- top.env 277
- toupper 37
- trace 180
- traceback 180
- tracemem 263
- transform 139
- trigamma 197
- TRUE 48
- trunc 17, 27, 202
- try 192, 246
- tryCatch 192

- `tukey`200
- `type.convert` 38
- `typeof` 13, 181
- `unif`200
- `union`203
- `unique`
 86, 109, 127, 153, 203
- `unlink`239
- `unlist` 92, 134
- `unloadNamespace`280
- `unserialize` 93
- `unstack`144
- `update.packages`252
- `upper.tri`95, 106, 121
- `url.show`239
- `UseMethod`303
- `utf8ToInt` 33
- `vapply`214
- `var`206
- `vector` 79
- `Vectorize`210
- `vi`155
- `vignette` 258, 318
- `warning`176
- `weekdays`220
- `weekdays.POSIXt`221
- `weibull`200
- `weighted.mean`206
- `which`55, 82, 103
- `while` 70
- `wilcox`200
- `with` 150, 277
- `write`230
- `write.ftable`218
- `writeBin`234
- `writeChar`234
- `writeLine`234
- `xedit`155
- `xemacs`155
- `xtabs`217
- `zapsmall` 27, 202

著者略歴

ま せ し げ る
間瀬 茂

1976年 東京工業大学大学院数学専攻博士課程修了
理学博士

現在 東京工業大学 名誉教授
専門：空間統計学，地球統計学

主要著書

空間データモデリング（共著，共立出版，2001）

工学のためのデータサイエンス入門（共著，数理工学社，
2004）

地球統計学とクリギング法（オーム社，2010）

ベイズ法の基礎と応用（日本評論社，2016）

新・数理/工学
ライブラリ [情報工学=2]

Rプログラミングマニュアル [第2版]
—Rバージョン3対応— (電子版)

2018年 2月 25日 ©

第2版発行

この電子書籍は2014年5月10日第2版発行の同タイトルの底本としています。

著者 間瀬 茂

発行者 矢沢和俊

【発行】

株式会社 数理工学社

〒151-0051 東京都渋谷区千駄ヶ谷1丁目3番25号

編集 ☎ (03) 5474-8661 (代) サイエンスビル

【発売】

株式会社 サイエンス社

〒151-0051 東京都渋谷区千駄ヶ谷1丁目3番25号

営業 ☎ (03) 5474-8500 (代) 振替 00170-7-2387

FAX ☎ (03) 5474-8900

組版 三美印刷

《検印省略》

本書の内容を無断で複写複製することは、著者および
出版者の権利を侵害することがありますので、その場合
にはあらかじめ小社あて許諾をお求め下さい。

ISBN978-4-86481-900-8

サイエンス社・数理工学社の

ホームページのご案内

<http://www.saiensu.co.jp>

ご意見・ご要望は

suuri@saiensu.co.jp まで。