計算機統計セミナー　R準備講座

東海大学理学部　山本義郎 *

2016 年 2 月 6 日

1　R の導入

1.1　R とは

R は統計計算と統計グラフ作成機能をもつ、フリーの統計解析ソフトウェアである。**R** は商用ソフトの S-PLUS(数理システム) や S version 4(アイザック) など S 言語 (AT&T ベル研究所の Richard A. Becker, John M. Chambers, and Allan R. Wilks により作られた統計解析やグラフィックスのための言語) にもとづくソフトウェアと関数などについての操作が良く似ており、S を使っている場合には関数の利用法がほぼ同じことから、R には馴染みやすいだろうし、大学では R を使っていたが会社では商用の S-PLUS を用いるということもあるだろう。**R** は S のクローンのフリーウェアと捉えられがちであるが、全くのクローンではなく (もちろん完全なクローンを目指しているわけでもない)、内部的な構成などは S とは全く異なっている。

グラフィックスの機能や GUI(メニューやダイヤログボックス) を利用した対話的な解析機能については、商用ソフトほどの機能があるわけではないので、最初は使いにくいと感じることもあるかもしれないが、解析ソフトウェアとしての機能は、パッケージと呼ばれる追加のライブラリ (関数群) の充実により商用ソフトと遜色ない機能が利用できる。

これまでは、**R** の利用方法などについては、書籍やホームページなど日本語で書かれているものが少なく、ほとんどが英語の情報であったため、英語嫌いのために利用を敬遠する向きがあったが、R の日本語化に取り組まれている中間さん、R のドキュメントの翻訳の中心的な役割をされている間瀬先生などの尽力により、日本人が利用しやすい環境が整い始めている。それに伴い、現在では日本語の書籍なども多く出版されるようになり、使い始めるためのハードルはかなり低くなってきている。

1.2　R のインストールと起動の確認

R は、Windows, Macintosh, Linux(Unix) などの環境で利用することができる。Windows や Mac 版の **R** はインストーラーが利用でき、非常に簡単にインストールできる。Linux についても redhat, debian, vinelinux をはじめ主要なディストリビューションに対するバイナリ (パッケージにより簡単にインストール可能) が用意されている。

RjpWiki (`http://www.okada.jp.org/RWiki/`) において、最新版のインストールについての情報が得られるので、それを参考にすれば簡単にインストールできる。RjpWiki は、Wiki というシ

*yoshiro@yamamoto.name

ステムに慣れていない方には少々わかりにくい構成になっていかと思われるが、そのような方は、インストールに関しては、セミナーの前に配付した資料を参照してください。

2 基本操作

2.1 起動と終了

Linux 版では、ターミナルで R というコマンドを実行することにより **R** の環境となる。「スタート」メニューの R グループから「R 3.2.3」を起動する。Mac OS X では、アプリケーションフォルダある「R」をクリックすることで起動できる。

R を起動すると、**R** システムでは、> (プロンプト) がコマンド (命令) の入力を待っている。例えば、Windows 版の場合 (日本語版) では、次のようなウィンドウが現れる。

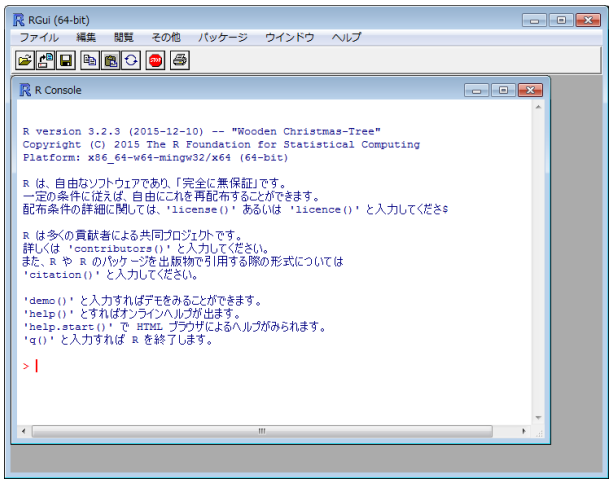

図 2.1: R 起動時のウィンドウ (Windows 版)

ここに、コマンド (式や関数) を入力して実行 ([Enter] キー) することにより、その式を評価した結果が表示され、次のコマンドを待つプロンプトが表示される。コマンドは、関数名 (引数) の形で与えるが、コマンドは大文字小文字を区別することに注意せよ。

非常に簡単な利用例を以下に与える。まず、単に数値を入力すると、その数値 (演算の結果) が返される (表示される)。

```
> 1+2
3
```

R を終了したい場合には、プロンプトに対してコマンド `q()` を実行する。Windows 環境では、ウィンドウの右上の終了ボタンをクリックしても終了できる。

```
> q()
```

このとき、作成した変数を保存するなど作業内容を保存するかの問い合わせがあるので、保存する必要があれば「はい」を選択し、必要なければ「いいえ」を選択する。

2.2 計算機としての利用

R における計算は、上で示した足し算の他にも、演算子 (+, -, *, /, ^(べき乗)) を用いて通常の算術表記による計算が行える。更に、三角関数 (cos(), sin(), tan()) や、平方根 sqrt()、自然対数 log() などの算術関数や、pi($\pi = 3.14\cdots$) などの定数が用意されているので、関数電卓としても利用できる。

統計処理のための関数も数多く用意されており、ある変数に関して統計量を計算したいなどの場合には、変数を c() 関数を使ってデータをベクトルとして扱い、ベクトルに対して関数を適用することにより分析 (計算) を行う。

例えば 10 人の身長のデータの平均は、用意されている統計関数により、次のように、平均を計算する関数 mean() で計算できる。

```
> mean(c(148, 160, 159, 153, 151, 140, 158, 137, 149, 160))
[1] 151.5
```

上記のような計算を行なう場合は、あらかじめデータに <- 演算子を利用して適当な名前を付け、その名前に対して計算をおこなう。(<- 演算子の代わりに= を使って付値をすることができる)

```
> height <- c(148, 160, 159, 153, 151, 140, 158, 137, 149, 160)
> height
 [1] 148 160 159 153 151 140 158 137 149 160
> mean(height)
 [1] 151.5
```

2.3 統計解析関数とオブジェクト

2.3.1 変数の型について

データとして数値以外にも文字も扱えるがその場合には"(ダブルクォート) でくくる。カテゴリカルデータは文字列ベクトルとして定義でき、例えば、性別を "M", "F" とするとき、次のように指定する。

```
> sex <- c("F","M","M","F","F","F","M","F","M","F")
```

このような属性 (質的変数) を利用して属性ごとに統計量を計算したいこともある。tapply() 関数はそのような目的に利用でき、引数として、オブジェクト、属性オブジェクト、統計量をとる。例えば、身長 (height) について性別 (sex) の平均値 (mean) を求めるには、次のようにする。

```
> tapply(height, sex, mean)
       F        M
147.7143 150.1250
```

2.3.2 データフレームについて

通常のデータ解析においては、特定の集団に対して複数の変数が観測されるため、それらのデータをまとめて扱いたいと思う。data.frame() 関数により、複数の変数をひとつのデータフレーム

として扱うことができる。

```
> mydata <- data.frame(height,sex)
> mydata
   height sex
1     148   F
2     160   M
3     159   M
4     153   F
5     151   F
6     140   F
7     158   M
8     137   F
9     149   M
10    160   F
```

データフレーム内の変数は $ を使って mydata$height のようにしてアクセスでき、通常の変数として取り扱い可能である。

```
> mydata$height
 [1] 148 160 159 153 151 140 158 137 149 160
> mean(mydata$height)
[1] 151.5
>
```

Excel などで作成したファイルを **R** に取り込むことができる。Excel ファイルそのままなども取り込むことも可能だが、まずは基本的な以下のような変数が縦に並び、空白で値が区切られ、変数名がついているデータ (ファイル名は example1.dat，Excel ではテキスト形式として保存することにより作成可能) が c:\tmp にあるとする (example1.dat を c:\tmp に保存すると以下のコマンドにより実行できる)。

```
#ファイル (example1.dat) の中身
height weight sex
148 41 "M"
160 49 "M"
159 45 "M"
153 43 "M"
151 42 "M"
140 29 "M"
158 49 "M"
137 31 "M"
149 47 "M"
160 47 "M"
151 42 "M"
157 39 "M"
157 48 "M"
144 36 "M"
139 32 "M"
139 34 "M"
149 36 "F"
142 31 "F"
150 43 "F"
139 31 "F"
161 47 "F"
140 33 "F"
```

```
152 35 "F"
145 35 "F"
156 44 "F"
147 38 "F"
147 30 "F"
151 36 "F"
141 30 "F"
148 38 "F"
```

このようなテキストデータを、データフレームとしての読み込むためには、read.table 関数により次のようにする。

```
exdata1 <- read.table(file="c:/tmp/example1.dat", header=T)
```

read.table() 関数の引数として、file=引数はデータファイル名を指定する。この指定でフォルダ (ディレクトリ) に\(￥記号) ではなく/(スラッシュ) を用いる) を指定していることに注意せよ。この引数は

```
exdata1 <- read.table("c:/tmp/example1.dat", header=T)
```

のように第一引数として指定すれば file=を省略することもできる。**R** の関数には、このように名前つきの引数と名前なしの引数が使えるため、慣れてきたら名前なし引数を使い、何を指定しているか確実にするためには名前つきの引数指定を行えばよい。このファイルには変数名が先頭行にあるので、引数として header=TRUE (T でもよい) をつけることにより、変数名を取り込んでいる。

データフレームのデータを、解析対象にする場合には、その都度データフレーム名に$を付け exdata1$height などとするのは面倒なので、attach 関数を用いて指定した変数名を探すリスト (検索リストと呼ぶ) に登録することで、height だけでアクセスできるようにするとよい。データフレームの中の変数と直接作成した変数の名前が重複する場合は、データフレームの変数の方が優先順位が低くなるので、先ほど作成した変数を削除しておく。まず ls() 関数で、現在付値した変数を確認し、rm() 関数で削除する。その後、attach() 関数で、検索リストに exdata1 を追加する。

```
> ls()
[1] "exdata1" "height" "mydata" "sex"
> rm(height, sex)
> attach(exdata1)
```

検索リストから削除したい場合は、detach() 関数で検索リストから削除したいデータフレーム名を指定する。

2.3.3 基本統計量の計算 (再)

R には多くの関数があるが、特に基本的なデータの要約のための関数として mean(), median(), max(), min(), var(), sd() などがある。ここで、分散は不偏分散であることに注意せよ。

```
> mean(height)
[1] 149
> var(height)
[1] 53.51724
> sd(height)
[1] 7.315548
```

度数分布表を作成する（集計する）ためには、table() 関数を使う。単純に table() 関数を使った場合には、各値ごとにその度数を集計するので、質的データ (カテゴリカルデータ) に対してはそのまま適用する。量的データに対しては、データ値ごとに集計するのではなく通常は度数分布表を作ることが要求されるが、そのためには cut() 関数を使い指定した区間に入るデータの数を集計する方法により度数分布表が作成できる（他の方法として、後述するヒストグラムを作成する関数を利用する方法などもある)。

```
> table(sex)
sex
 F  M
14 16
> table(height)
height
137 139 140 141 142 144 145 147 148 149 150 151 152 153 156 157 158 159 160 161
  1   3   2   1   1   1   1   2   2   2   1   3   1   1   1   2   1   1   2   1
> table(cut(height,seq(135,165,by=5)))
(135,140] (140,145] (145,150] (150,155] (155,160] (160,165]
        6         4         7         5         7         1
```

この例に用いた seq() 関数は、初期値から終端値までの by=で指定された間隔の数列を生成する関数である。

R では関数を適用するデータの型により振る舞いが異なるので、そのことを意識して使えると効率的である。例えば、データフレームに対しては以下のように、要約統計量を簡単に求めることができる。

```
>  summary(exdata1)
     height          weight      sex
 Min.   :137.0   Min.   :29.00   F:14
 1st Qu.:142.5   1st Qu.:33.25   M:16
 Median :149.0   Median :38.00
 Mean   :149.0   Mean   :38.70
 3rd Qu.:155.3   3rd Qu.:43.75
 Max.   :161.0   Max.   :49.00
```

ここで、Min.(最小値)、Max.(最大値)、1st Qu.(第1四分位数)、3rd Qu.(第3四分位数)、Median(中央値)、Mean(平均値) である。

2.4 グラフ表示

一変数データのグラフ表現としては、ヒストグラムを表示する関数 hist() や 箱ひげ図をプロットする boxplot() などがある。hist() は引数の指定により、かなり自由度が高い利用ができる。以下に、何も指定しなかった場合と、breaks= の値を変えて階級数を変更した場合を実行してみる。breaks= にスカラー値 (定数) を与えると、その数に近い階級数の適当なヒストグラムを作成

する、区切りが 0 や 5 などのキリがよくなるとは限らないので、そのように区間を指定したい場合は、breaks=に区切り値のリストを与える。

```
> hist(height)
> hist(height,breaks=3)
> hist(height,br=c(136,140,144,148,152,156,160,170))
```

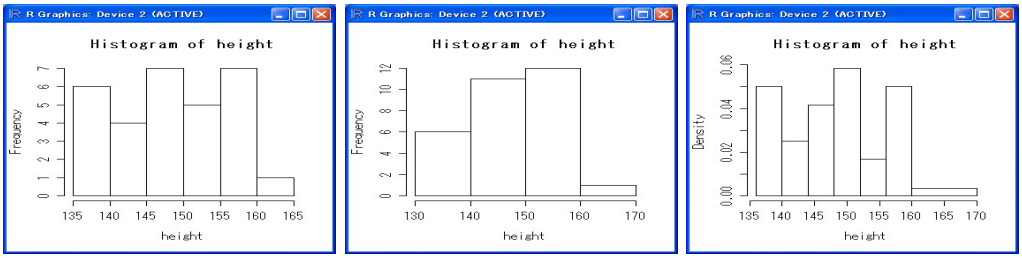

図 2.2: ヒストグラムの作成

さらに、hist() を使ってできることを試してみる。グラフの描画単位はパネルとなっていて、1 つのパネルに複数のグラフを配置する「複数図表モード」もあり、多様な表現ができる。par() 関数により、複数描画モードの指定も含め、グラフィックスのパラメータが変更されるが、この変更は、その後のグラフィックスの描画の全てに影響するため、par() 関数を使う場合には、変更前の内容を保存しておき、グラフの利用が終わったところで元に戻すと安心である。

```
> op <- par(mfrow=c(2,1)) #複数グラフモード、op にグラフパラメータを保存
> hist(height)
> hist(height, br=6,col="lightblue", border="pink")
> par(op)                  #グラフパラメータを元に戻す
> str(hist(height,plot=F))
List of 4
 $ breaks     : num [1:7] 135 140 145 150 155 160 165
 $ counts     : int [1:6] 6 4 7 5 7 1
 $ intensities: num [1:6] 0.0400 0.0267 0.0467 0.0333 0.0467 ...
 $ mids       : num [1:6] 138 143 148 153 158 ...
> h <- hist(height,plot=F)
> h$counts
[1] 6 4 7 5 7 1
```

この例で、hist() が単にヒストグラムを描くだけでなく、ヒストグラムを構成する階級の情報をもっていて、それを参照することができることがわかる。

要約プロットしては、stem(height) 関数により幹葉図 (ステム＆リーフプロット) が得られるが、これはグラフィックスデバイスではなく、通常のコマンドウィンドウに表示される。

```
> stem(height)

  The decimal point is at the |

  136 | 0
  138 | 000
  140 | 000
  142 | 0
  144 | 00
  146 | 00
  148 | 0000
  150 | 0000
  152 | 00
  154 |
  156 | 000
  158 | 00
  160 | 000
```

boxplot() は箱ひげ図 (ボックスプロット) を作成する、このボックスプロットは外れ値が○で表示されるタイプのものである。質的変数を与えることにより、層別の箱ひげ図を作成することもできる。次の例は、身長に関する単なる箱ひげ図 (左) と、性別による層別の箱ひげ図 (右) を作成する。

```
> par(mfrow=c(1,2)) #複数グラフモード
> boxplot(height, main="height")
> boxplot(height~sex, main="height~sex")
```

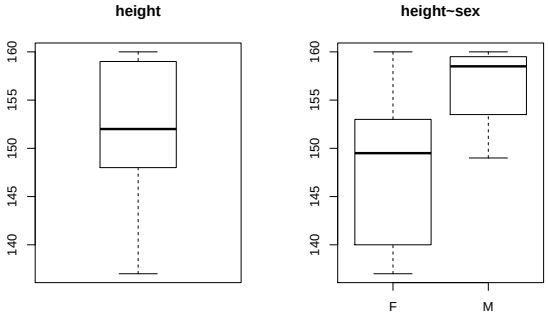

図 2.3: 箱ひげ図 (左) と層別箱ひげ図 (右)

R において利用できる統計グラフの多くは plot() 関数で作成できるが、この関数の振る舞いはデータ (データフレーム) の型に大きく依存しており、与えるデータによって、インデックスプロットや散布図などをはじめとする様々なグラフが作成できる。

```
> plot(height)
> plot(seq(1,length(height)),height,type="b",xlab="")
#上と同じインデックスプロットを、折れ線の散布図として描く方法
> plot(height,weight)  #散布図
```

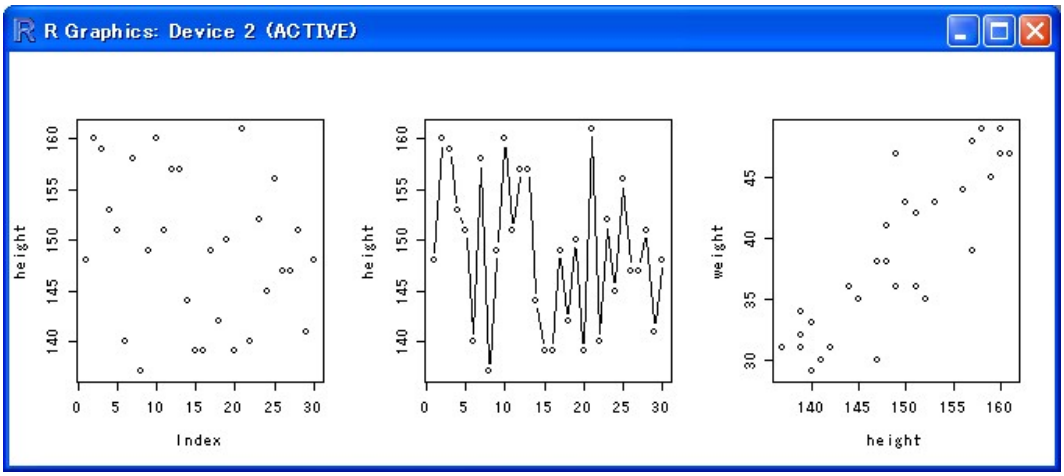

図 2.4: plot 関数によるグラフ作成

このようなプロットを作成するのに便利な方法について「応用編」でも紹介する。

データフレームに対するグラフや、データフレームにおける変数に関するグラフは、カテゴリカルデータである (factor 型など因子型もしくは text 型である) 場合には、振る舞いや指定が異なるので、うまく利用できるようになることが **R** でのグラフ作成の上達のコツとなる。

```
> plot(exdata1)  #データフレームをプロットする
> par(mfrow=c(1,2))
> plot(exdata1$sex)  #データフレーム内の 1 変数をプロットする
> plot(weight~height, data=exdata1)  #データフレーム内の変数のプロット
```

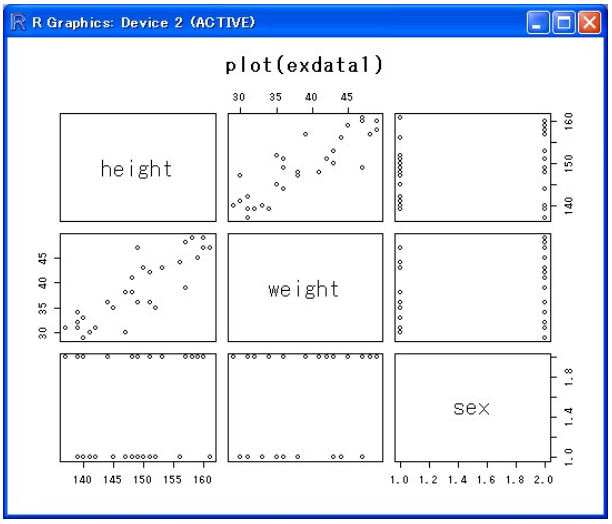

図 2.5: plot 関数によるグラフ作成 (その 1)

R では関数のグラフの表示ためには、1 変数関数のグラフを作図する関数 curve がある。例え

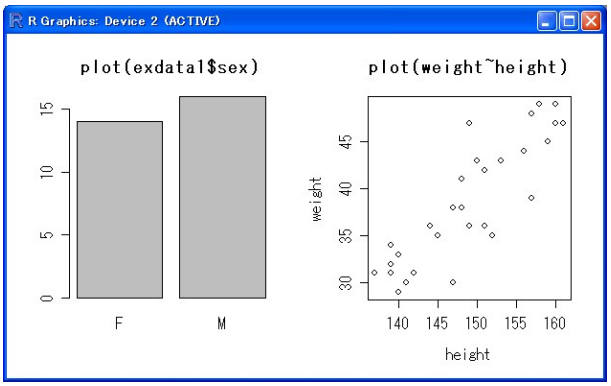

図 2.6: plot 関数によるグラフ作成 (その 2)

ば、正弦関数 $y = \sin(x)$ の区間 $(0, 2\pi)$ のグラフは次のように作図する。(x の関数については x を省略することができる。)

```
> curve(sin,0,2*pi)
```

更に余弦関数を追加するには，curve 関数に範囲は指定せず、add=オプションを利用し、先のグラフと区別するために色を指定することで実現できる。

```
> curve(cos, add= TRUE, col="red")
```

2.5 分布関数，パーセンタイル，乱数

R には特定の分布に従う確率変数に対して、分布関数、確率 (密度) 関数分布、乱数などを求める関数がある。関数名は，表 2.1

表 2.1: 確率分布に関する関数名の規則

関数名	関数の働き
d 分布名	指定した値に対して、分布名の分布の確率 (密度) 関数の値を返す
p 分布名	指定した値に対して、分布名の分布の累積分布関数の値を返す
q 分布名	指定した値に対して、分布名の分布の分位点 (パーセンタイル) を返す
r 分布名	分布名の分布の乱数を指定した数だけ返す

これらの関数は分布を特定するパラメータを指定することができる。例えば、正規分布の分布名は norm であるので、標準正規分布にしたがう X について確率密度関数 $\phi(x)$, 分布関数 $\Phi(x)$、パーセンタイル $\Phi^{-1}(x)$ について、$\phi(x)$, $\Phi(2)$, $\Phi^{-1}(0.975)$ はそれぞれ、次のように求めることができる。

```
> dnorm(2,mean=0,sd=1)
[1] 0.05399097
> pnorm(2,0,1)
[1] 0.9772499
> qnorm(0.975)
[1] 1.959964
```

上の例でわかるように、省略可能なパラメータもあるので、詳しくはヘルプを参照のこと。分布名の一覧を表 2.2 に与えた。

表 2.2: 主要な確率分布に関する分布名

分布名	分布
binom	二項分布
hyper	超幾何分布
pois	ポアソン分布
nbinom	負の二項分布
geom	幾何分布
multinom	多項分布
unif	一様分布
norm	正規分布
gamma	ガンマ分布
exp	指数
beta	ベータ分布
chisq	χ^2 分布
f	F 分布
t	t 分布
lnorm	対数正規分布
weibull	ワイブル分布
cauchy	コーシー分布

2.6 検定関数

R には仮説検定のための関数も用意されている。関数名は，表 2.3。

mydate の男女の身長の違いについて、t 検定で差があるかどうか検定するには、次のようにする。

```
> t.test(height~sex,data=mydata)

        Welch Two Sample t-test

data:  height by sex
t = -1.9369, df = 7.9726, p-value = 0.0889
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -18.260740   1.594074
sample estimates:
mean in group F mean in group M
       148.1667        156.5000
```

表 2.3: 仮説検定に関する関数

関数名	検定
t.text()	t 検定
var.text()	F 検定
chisq.text()	chi^2 検定
prop.text()	母比率の検定
binom.text()	二項検定
fisher.text()	フィッシャーの正確確率検定
wilcox.text()	ウィルコクソンの順位和検定
friedman.text()	フリードマンの順位和検定
kruskal.text()	クラスカル・ウォリスの検定
ks.text()	コロモゴルフ・スミルノフの検定
mcnemar.text()	マクネマー検定

2.7 統計モデル

R は S と同様に統計モデル（回帰モデル、分散分析、一般化線形モデル、非線形回帰モデルなど）の分析が実行できる。

例として、車の運転者と自転車の運転者との自転車レーンの効果について、自転車運転者と道路の中心線との距離 travel と自転車運転者と通過車両との距離 separation とについて 10 人の自転車運転者について調査した結果を考察する (bicycle.csv :LISP-STAT, 1997, 共立出版)。このデータに対して、回帰モデルのあてはめによる解析を行う。モデルとしてまず、separation を目的（従属）変量、travel を説明（独立）変量とした線形回帰モデルを、次のように、関数 lm を使って解析する。

```
> bicycle <- read.table("bicycle.csv", header=T, sep=",")
         #CSV 形式 (カンマ区切り) なのでセパレータを sep="," として指定
> lm(separation ~ travel, data=bicycle)

Call:
lm(formula = separation ~ travel, data = bicycle)

Coefficients:
(Intercept)       travel
    -2.1825       0.6603
```

回帰モデルの結果から、このモデルに対する回帰係数が切片項 (Intercept) が -2.1825、1 次の項の係数 (travel) が 0.6603 であることがわかる。lm() の結果はオブジェクトであり、あとでこのモデルを扱うために lm() の結果を適当な名前として付値することもできる。慣例的に、”データフレーム名”.”モデル” として bicycle.lm のようにオブジェクト名をつけることが多いが、データや解析がわかりやすくしているだけで、オブジェクト指向にそった命名規則の制約があるわけではない。

回帰モデルオブジェクトに対しては summary() を適用することにより、モデルの結果についての詳細な情報が得られ、そのオブジェクトに対して回帰係数を表示するように問い合わせることなどもできる。これを利用することにより、回帰直線を作成することができる。

```
> bicycle.lm <- lm(separation ~ travel, data=bicycle)
> summary(bicycle.lm)

Call:
lm(formula = separation ~ travel, data = bicycle)

Residuals:
     Min       1Q   Median       3Q      Max
-0.76990 -0.44846  0.03493  0.35609  0.84148

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.18247    1.05669  -2.065   0.0727 .
travel       0.66034    0.06748   9.786 9.97e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5821 on 8 degrees of freedom
Multiple R-Squared: 0.9229,     Adjusted R-squared: 0.9133
F-statistic: 95.76 on 1 and 8 degrees of freedom,       p-value: 9.975e-006
> coef(bicycle.lm)
(Intercept)      travel
 -2.1824715   0.6603419
> plot(bicycle)
> names(bicycle.lm)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"
> abline(bicycle.lm$coef)
```

関数 names は、オブジェクトがどのような情報をもっているか確認できる (多様な関数であるがシンプルな利用としてこのように使える)。lm の結果のオブジェクトには上記のような情報があり、

例えば回帰係数は bicycle.lm$coef(coef は coefficients の省略) によっても得られる。関数 abline は、表示されている散布図に、切片と傾きを指定した直線を描くので、この関数の引数として bicycle.lm$coef を与えている。

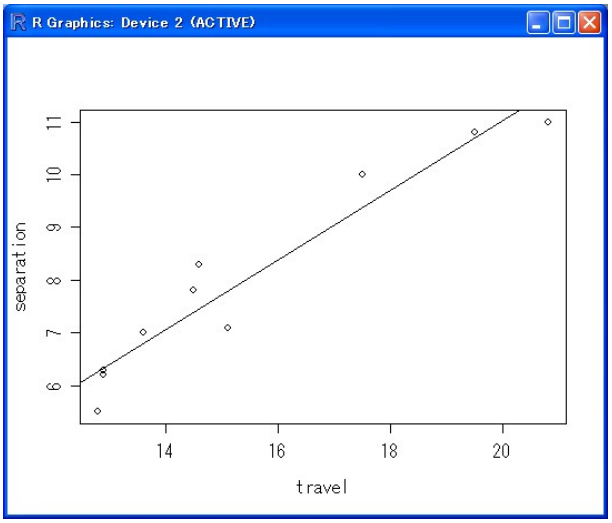

図 2.7: 回帰直線

3 R で解析を行うための詳細

これまで紹介した内容で、ある程度どのように R を利用するかについては理解できたのではないだろうか。本章では、更に R の詳細についての理解を深め、実際に解析を行う際に把握しておきたい R の特徴について解説する。

本節以降の内容については、ページ数の関係で詳細な記述は、最新の情報を追加した内容をホームページにおいて提供する。

3.1 デモとヘルプ

R でどんなことができるのか、その際のコマンドは何かについて概観したい場合には、デモが利用できる。デモは、demo(”トピック名”) で実行できる。demo() とするとトピック名が表示されるので、トピック名を確認して特定のデモを見るとよい。例えば、どのようなグラフが作成できるか確認するために、graphics のデモを実行する。

```
> demo()
> demo(graphics)
```

R には沢山の関数があるが、それら全ての使用方法を覚えておくことは不可能だろう。そこで、関数の使用方法などについては、ヘルプの参照が必須となる。ヘルプには 2 種類あり、その一つの方法である help() 関数は、引数として関数名を指定することによりヘルプが表示される。

```
> help(plot)
```

関数名を完全には覚えていないが、一部わかる際には、指定した文字を含むで関数名を表示する apropos() 関数や、指定したキーワードに関係する関数の説明を表示する help.search() 関数を用いる。これらの関数は、キーワードとして文字を指定することから、キーワードをダブルクォート ("") でくくる必要がある。Windows の GUI におけるメニューを利用した場合、これらは「ヘルプ」メニューから実施できる。

```
> apropos(plot)
> help.search("plot")     #plot という文字列の検索のため""でくくる
```

3.2 データについて

3.2.1 変数の型について

R で扱うことのできるデータの型は、数値 (numeric) 型とテキスト (text) 型の他に、論理型 (logical) がある。論理型は値として TRUE(T) または FALSE(F) のみを取り、関数の引数としても用いられる。数値としては、実数の他にも複素数もあり、無限大を表す Inf(数値積分で用いられる) や、数でない NAN(0 で割った結果など) や欠損値を表す NA などが利用できる。

3.2.2 ベクトル

すでに、変数は c() 関数を用いてベクトルとして用いることを説明しているが、規則的なベクトルを生成する方法としていくつか方法がある。コロンを数値で挟むと、2 つの整数に対する増分が 1 の数列を生成でき, より複雑な数列は sep() 関数で増分を指定できる。

数値や繰返しのパターンが rep() 関数で生成できる。

```
> 1:5     #seq(1,5)
[1] 1 2 3 4 5
> 5:1     #seq(5,1)
[1] 5 4 3 2 1
> seq(2,10,3)  #seq(from=2,to=10,by=3)
[1] 2 5 8
> rep("F",3)
[1] "F" "F" "F"
> rep(1:3,c(2,3,4))
[1] 1 1 2 2 2 3 3 3 3
```

ベクトルに対して作用できる関数として、統計量を計算する mean, median, var, cor や min, max, range などの他、和 (sum)、累積和 (cumsum) などの要素の集計関数と並べ替え関係の (sort, rank, order) などがある。これらの関数をうまく使えるようになると効率的なプログラミングに結びつく。

```
> uriage <-c(243,224,253,335,314)
> (kokusei <- uriage/sum(uriage))
[1] 0.1775018 0.1636231 0.1848064 0.2447042 0.2293645
> (ruisekihi<- cumsum(uriage)/sum(uriage))
[1] 0.1775018 0.3411249 0.5259313 0.7706355 1.0000000
```

上の例では、付値に対してカッコ () で囲んでいるが、このようにすると、その結果を表示するので、変数変換した結果を確認したい場合にはこのようにするとよい。

ベクトルの要素はインデックス (要素番号) を用いてアクセスでき、大きさが同じ真偽 (TRUE or FALSE) のベクトルにより真 (T) に対応するケースのみ取り出すことができる。

```
> height
 [1] 148 160 159 153 151 140 158 137 149 160 151 157 157 144 139 139 149 142 150
[20] 139 161 140 152 145 156 147 147 151 141 148
> height[3:5]
[1] 159 153 151
> height[c(1,3,7)]
[1] 148 159 158
> height[c(-1,-3,-7)]
 [1] 160 153 151 140 137 149 160 151 157 157 144 139 139 149 142 150 139 161 140
[20] 152 145 156 147 147 151 141 148
> weight>45
 [1] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE
[13]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
[25] FALSE FALSE FALSE FALSE FALSE FALSE
> height[weight>45]
[1] 160 158 149 160 157 161
> height[sex=="F"]
 [1] 149 142 150 139 161 140 152 145 156 147 147 151 141 148
```

3.2.3 リストと配列

ベクトルをまとめて扱う方法として、リストを作成する方法と配列や行列を利用する方法がある。リストは、異なる形式のベクトルを要素としてもつことができ、list() 関数で複数のベクトルを一つのリストとしてまとめることができる。array() 関数はベクトルから多次元の配列をつくることができ、matrix() 関数で行列を作ることができる。

```
> c(height,weight)
 [1] 148 160 159 153 151 140 158 137 149 160 151 157 157 144 139 139 149 142 150
[20] 139 161 140 152 145 156 147 147 151 141 148  41  49  45  43  42  29  49  31
[39]  47  47  42  39  48  36  32  34  36  31  43  31  47  33  35  35  44  38  30
[58]  36  30  38
> in1 <- list(height,weight)
> in1
[[1]]
 [1] 148 160 159 153 151 140 158 137 149 160 151 157 157 144 139 139 149 142 150
[20] 139 161 140 152 145 156 147 147 151 141 148

[[2]]
 [1] 41 49 45 43 42 29 49 31 47 47 42 39 48 36 32 34 36 31 43 31 47 33 35 35 44
[26] 38 30 36 30 38

> in2 <- array(c(height,weight), dim=c(30, 2))
> in2
      [,1] [,2]
 [1,]  148   41
 [2,]  160   49
(略)
[30,]  148   38
> in3 <- matrix(c(height,weight), ncol=2)
> in3
      [,1] [,2]
 [1,]  148   41
 [2,]  160   49
(略)
[30,]  148   38
```

リストの要素にアクセスするには、リスト名 [[インデックス]] と指定すればよい。従って、得られたリスト in1 の第一要素のリストを得るには以下のように指定すればよい。リストの要素には名前をつけることができ、名前を使ってアクセスできる。

```
> in1[[1]]
 [1] 148 160 159 153 151 140 158 137 149 160 151 157 157 144 139 139 149 142 150
[20] 139 161 140 152 145 156 147 147 151 141 148
> in1[[1]][3]
[1] 159
> names(in1) <- c("身長","体重")
> in1$身長
 [1] 148 160 159 153 151 140 158 137 149 160 151 157 157 144 139 139 149 142 150
[20] 139 161 140 152 145 156 147 147 151 141 148
> in1$身長 [5]
[1] 151
```

3.2.4 行列とベクトルの演算

行列に対しては、行列の演算 (和、差、積%*%) や行列式の計算ができ、逆行列 (solve()) や固有値固有ベクトル (eigen()) を求めたり、特異値分解 (svd()) やコレスキー分解 (chol()) などをおこなう関数も用意されている。

行列や配列のデータに対しては、cov() 関数を使って分散共分散行列を計算できる。以下では、

共分散行列を求め、求められた行列に対する固有値と固有ベクトルを eigen() 関数により求めている。

```
> cov(in2)
         [,1]     [,2]
[1,] 53.51724 40.79310
[2,] 40.79310 41.73448
> eigen(cov(in2))
$values
[1] 88.842189  6.409535

$vectors
          [,1]       [,2]
[1,] 0.7559557 -0.6546228
[2,] 0.6546228  0.7559557
```

3.3 パッケージの利用

R では、すべての機能があらかじめ実装されているのではなく、必要に応じてパッケージを追加することにより利用できる機能の幅を広げ、パッケージ単位で更新するなど管理の集中化を避けるとともにシステムの肥大化を防ぐ仕組みになっている。

必要な機能を自分で作ることも時には必要であるが、多くの場合、すでに先人が必要な機能を作成し、パッケージとして利用可能となっていることが多いので、どのようなパッケージがあるのか知っておくことも重要である。

パッケージの利用はとても簡単に行うことができる。すでにシステムにインストールされていて、すぐ利用可能なパッケージの一覧は library() を実行することで確認できる。インストールされていないパッケージについては、install.packages(パッケージ名) でインストールできる。インストール済みのパッケージは library(パッケージ名) で利用が可能となる。

例えば、「S- PLUS による統計解析 (Modern Applied Statistics with S-PLUS)」で扱われているデータや関数をまとめた MASS パッケージ (ライブラリ) は、インストールされているので、単に library(MASS) とすれば利用可能となり、パッケージの詳細は library(help="MASS") とすれば確認でき、また HTML ヘルプの packages において関数やデータの詳細を確認することができる。

```
> library(MASS)
> library(help="MASS")
```

「Using R for Introductory Statistics」の関数やデータは UsingR パッケージとしてまとめられている (http://wiener.math.csi.cuny.edu/UsingR/)。これは、インストールされていないので、まずインストールしてから、利用可能とする。

```
> install.packages("UsingR")
also installing the dependency ‘HistData’

 URL ’https://cran.r-project.org/bin/windows/contrib/3.2/HistData_0.7-5.zip’ を
試しています
Content type ’application/zip’ length 252897 bytes (246 KB)
downloaded 246 KB

 URL ’https://cran.r-project.org/bin/windows/contrib/3.2/UsingR_2.0-5.zip’ を試
しています
Content type ’application/zip’ length 2078803 bytes (2.0 MB)
downloaded 2.0 MB

 パッケージ ‘HistData’ は無事に展開され、MD5 サムもチェックされました
 パッケージ ‘UsingR’ は無事に展開され、MD5 サムもチェックされました

 ダウンロードされたパッケージは、以下にあります
       C:\Users\yama_2\AppData\Local\Temp\Rtmpw9K9w1\downloaded_packages
> library(UsingR)
```

R の起動ごとに追加するパッケージはデフォルトの設定に戻るので、必要な際には library コマンドで追加する必要がある (起動時に取り込む設定は可能)。インタラクティブグラフが利用できる iplot や Rcmdr パッケージなど操作の補助となるパッケージと、固有分野における分析のための関数など、さざまざなものが提供されている。

3.4 データの読み込み

3.4.1 scan によるデータの読み込み

次のように値が変数ごとに列として配置されているとき、データをファイルから入力する方法としては、scan() 関数に適切な引数の指定を行う。

```
# example2.dat
148 41
160 49
159 45
153 43
151 42
140 29
158 49
137 31
149 47
160 47
151 42
157 39
157 48
144 36
139 32
139 34
149 36
142 31
150 43
139 31
```

```
161 47
140 33
152 35
145 35
156 44
147 38
147 30
151 36
141 30
148 38
```

引数は file="ファイル名"、sep="区切り文字" などを指定する。空白により固定長に整形してあるデータの場合 sep 引数はいらないが、カンマ区切りの場合は sep=","と指定する。複数の変数を扱い場合には、データフレームを利用するのがいいが、scan 関数で扱う場合には注意が必要である。scan 関数は、改行を区切り記号としてしか扱わないので、2 列 10 行でデータを入力している場合でも

```
> in4 <- scan("example2.dat")
Read 60 items
> in4
 [1] 148  41 160  49 159  45 153  43 151  42 140  29 158  49 137  31 149  47 160
[20]  47 151  42 157  39 157  48 144  36 139  32 139  34 149  36 142  31 150  43
[39] 139  31 161  47 140  33 152  35 145  35 156  44 147  38 147  30 151  36 141
[58]  30 148  38
```

のように 1 つのリストとして読み込まれる。そこで 2 つ目の引数として指定されたデータの型のリストとして、型は数値の場合 0 をラベルの場合""を指定する。

```
> in5 <- scan("example2.dat", list(0,0))
Read 30 lines
> in4
[[1]]
 [1] 148 160 159 153 151 140 158 137 149 160 151 157 157 144 139 139 149 142 150
[20] 139 161 140 152 145 156 147 147 151 141 148

[[2]]
 [1] 41 49 45 43 42 29 49 31 47 47 42 39 48 36 32 34 36 31 43 31 47 33 35 35 44
[26] 38 30 36 30 38
```

ファイルを選択して読み込みたい場合には、file=file.choose() を指定するとよい。

```
> read.table(file=file.choose())
```

Windows の場合には、クリップボードを経由してデータの取り込みが可能である。Excel でデータをコピーしておき、R で以下の命令を実行するだけでよい。

```
exdata1 <- read.table("clipboard",header=TRUE)
```

3.4.2 テキストデータ以外のファイル形式の読み込み

Excel ファイルから直接データを読み込むのは、`library(RODBC)` により ODBC 接続することにより可能であるが、データベースの知識が若干必要となるので、Excel で名前をつけて保存するとして、形式を CSV 形式を指定し、CSV 形式のテキストとして保存し、`read.table()` 関数により取り込むのがよい。

他の解析ソフトのデータ (SPSS や SAS、Minitab) などのデータの読み込みは、`foreign` パッケージを利用することにより取り込み可能である。

3.5 関数とプログラミング

R においては、必要な処理を行う関数がない場合や、定形処理を行う場合には、ユーザ関数を作成するとよい。

3.5.1 関数定義

R における関数定義は `funcname <- function(arg1,..) expression` の形で行われ、`funcname` が関数名、`arg1,..` が引数、`expression` が本体である。関数名に括弧を付けずに入力すると関数の内容が表示される。関数 `var` について確認してみると

```
> var
function (x, y = x, na.rm = FALSE, use)
{
    if (missing(use))
        use <- if (na.rm)
            "complete.obs"
        else "all.obs"
    cov(x, y, use = use)
}
```

これにより、関数 `var` が関数 `cov` を利用して作られていることがわかる。

では、新しい関数としてトリム平均 (最大値と最小値を除いた平均) を求める関数 `tmean` を作ってみよう。

```
> tmean <- function (x)
{
    (sum(x)-max(x)-min(x))/(length(x)-2)
}
```

次のように、局所 (ローカル) 変数を使うこともできる (`tsum`, `tlen` が局所変数である)。

```
> tmean2<-tmean
> edit(tmean2)
function (x)
{
  tsum <- sum(x)-max(x)-min(x)
  tlen <- length(x)-2
  tsum/tlen
}
```

R におけるプログラミングでは、制御構造として条件分岐 (if, else, switch) や繰返し (for, while, repeat) が使えるため、通常のプログラム言語と同様のプログラミングが可能である。

一度作成した関数は、edit(関数名) で関数の編集が実行可能である。edit() を使う場合には、関数を上書きすることになるので、あらかじめ関数をコピーし (<-, =を使って)、コピーされた新しい関数を編集するのが安全である。また、自分で作った関数をファイルに保存しておき、source() 関数を使ってファイルからシステムに読み込むことができる。

3.5.2 プログラミングにおける留意点

R では、配列や行列に対する計算に対しては、通常のプログラミングのスタイル (制御構造の利用) を利用せず、できるだけ配列関数を用いたり、行列演算で対処することが重要である。例えば、**R** はベクトルや行列の処理には非常に強く、ベクトルに対する関数が用意されているので、ベクトルの成分の和を for ループで計算せず、sum() 関数を使うのがよい。また、変数変換についても for ループで各要素の値を作用するのではなく、log(height) などのように変数に対する演算を行う。

例として、中央値からの偏差についての計算を考える。

```
> height-median(height)  #中央値からの偏差
 [1]  -1  11  10   4   2  -9   9 -12   0  11   2   8   8  -5 -10 -10   0  -7   1
[20] -10  12  -9   3  -4   7  -2  -2   2  -8  -1
> sum(abs(height-median(height))) #中央値からの絶対偏差の和
[1] 180
> sum((height-median(height))^2) #中央値からの偏差の 2 乗和
[1] 1552
>  (height-median(height))%*%(height-median(height)) #内積で計算
     [,1]
[1,] 1552
```

上の例には基本的なアイディアがある。ベクトルとスカラーの和・差はベクトルの各成分に対して行われる。ベクトルに対して算術関数 (ここでは abs) 関数は各成分に対して作用される。成分の和の計算は、関数 (sum) を用いたり、行列演算で行うことができる。

R におけるプログラミングのコードの最適化については、プログラミングスキルがあがるにつれ意識すればよいが、RjpWiki には「R コード最適化のコツと実例集」などの話題もあり、大変参考になるだろう。

4 R を効果的に利用するために

R はコマンド中心で利用するため、意識して作業の効率化を考えないと、前回の作業の記録がなく、一から始めないといけないなど面倒なことが多い。また、**R** には、GUI を使って作業する方法などもあるので、そのような **R** を使って作業する上で知っておくと作業の効率化が図れたり、利用の幅が広がると思われることについて解説します。

4.1 データや解析結果の保存について

R では、自分で定義した変数などは終了時に「Save workspace image?」と聞かれる。これに対して「はい」と答えることにより **R** 起動ショートカットの作業ディレクトリに .RData という

ファイルに保存され、次回の起動時に自動的にロードされ、利用することが可能となる。ユーザが使用しているオブジェクトは objects() 関数 (古い S の関数 ls() も利用できる) により確認できる、引数なしで全てのオブジェクトを表示し、引数として pat="パターン" を指定することによりパターンを満たす名前だけを表示する。いらなくなったオブジェクトは、rm() 関数で削除する。

```
> objects()
[1] "height" "weight" "xname"
> objects (pat="we*") #we から始まるオブジェクト名
[1] "weight"
> rm("xname")
```

標準出力に表示される結果をファイルに出力する場合には、sink() 関数の引数に保存するファイル名を指定する。その後は、実行結果は標準出力には表示されずに、指定したファイルに書き込まれる。記録をやめ、再度標準出力に表示をうつす場合には、引数なしの sink() を利用する。

```
> sink("yama.log")
> objects()
> table(height)
> mean(height)
> sink()
```

データをファイルに出力するには、write() や write.table() 関数が利用できる。

4.2 データの入力や編集方法

R のデータフレームは、edit(データフレーム名) で、Excel のような表計算モードで編集することができる。

```
> edit(exdata1)
```

関数を作成したり、定形作業を行う場合に、エディタに作業記録を残すのは、**R** を使う上での基本となるが、エディタから直接 R のコマンドを実行できると、作業の効率が上がる。また、実行はできなくても、コマンドと変数名の色分けがされる表示があったり、カッコの対応を与えてくれる補完機能があるだけでも十分作業の効率が上がるだろう。

そのようなことが可能なエディタとして、Linux 環境では Emacs に対する ESS がある。Windows 環境では Meadow というフリーソフトがあり、ESS を利用することができる。しかし Emacs の操作性は慣れるまでは難しいと感じることが多いだろう。

実際には、R での分析には RStudio などの開発環境を利用するのがよい。

4.3 R の諸設定について

R は Linux 版では、ユーザが作業ごとにディレクトリ (フォルダ) を変え、そこから R を起動する作業により自然に作業環境の変更が可能であるが、Windows の場合にはインストールされたフォルダが作業フォルダになり、その場所が意識しにくいので、意図した作業フォルダを利用するようにしたほうがよい (データの読み書きや作業の記録のため)。例えば、作業フォルダを "c:\yama\Rintro" に指定するには図 4.1 のようにする。新たに、設定した場合、過去の履歴は全てクリアされること

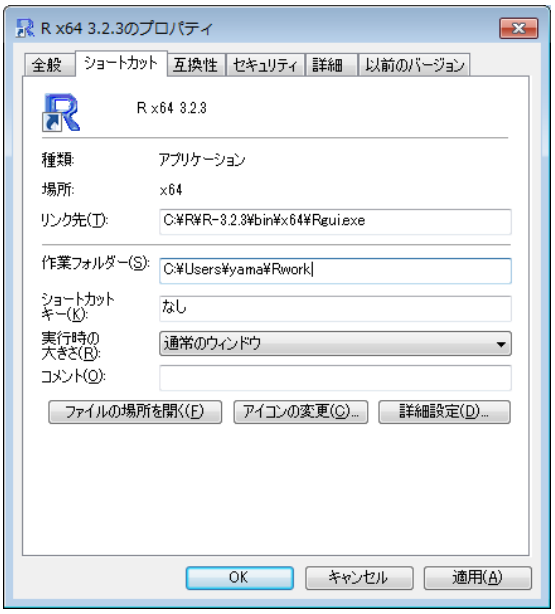

図 4.1: R のショートカットの設定

になるので、過去に作成したデータや関数を使いたい場合には作業フォルダの `.Rdata` を、コマンド履歴も使いたい場合には、`.Rhistory` をコピーすればよい。

英語表記で利用したい場合には、ロケールを変更すればよいので、ターミナルで環境変数を `LANG=C`, `LC_ALL=C` と変更することにより英語モードとなります。Windows の場合には、R のショートカットをコピーし、リンク先に

```
"C:\R\R-3.2.3\bin\x64\Rgui.exe" LANG=C LC_ALL=C
```

と環境変数を並べることにより設定できる。

「スタート」メニューから「コマンドプロンプト」を起動し、Rgui を起動する際に、オプションとして `--help` をつけると Rgui の起動時のパラメータが確認できる (図 4.2)。

```
> cd "C:\R\R-3.2.3\bin\x64"
> C:\R\R-3.2.3\bin\x64\Rgui.exe --help
```

4.4 R コマンダーの利用

R を GUI で利用するための R コマンダー (R Commandar) というパッケージがある。R コマンダーは、メニューから解析やグラフの指定を行うことが可能で、R で何ができるかわからない、R の関数が覚えられない、もっと簡単に使いたいなどの要求に応えるものであるので、初心者はまず気楽に使ってみるとよいでしょう。

R コマンダーを使うにはパッケージ `Rcmdr` をインストールすればよい。

R コマンダーを利用する場合には、SDI タイプで R を起動したほうが便利である。これは、**R** の起動時のオプションに `--sdi` をつけて起動することによりできる。

「フリーソフトウェア R による統計的品質管理入門」では R コマンダーを使ってほとんどコマンドを使うことなく品質管理 (QC) のために R を使っている。また、そのために必要な関数などが

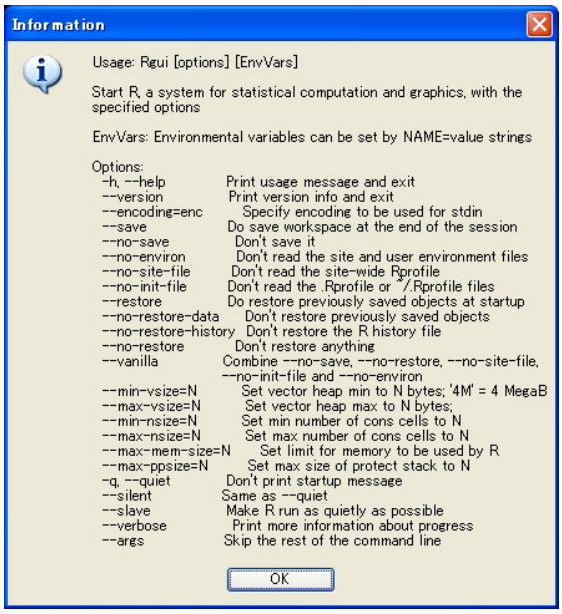

図 4.2: Rgui.exe –help

`http://www.ec.kansai-u.ac.jp/user/arakit/R.html`

で入手できる。

4.5 R に関する情報

R に関する情報源としては、なんといっても日本では RjpWiki

`http://www.okada.jp.org/RWiki/`

につきる。Wiki に慣れていない人にとっては、情報をどのように探せばよいか迷うと思うが、まずはトップページの「主な内容」を中心に調べたい情報があるか調べ、見つからなかったら上のメニューにある「一覧」で見出しを探すか、「単語検索」で探すと欲しい情報にたどり着けるだろう。

R に関する日本語の書籍も増えてきており、各分野において参考になるものがある。経済の分野では、「経済・経営のための統計学」が **R** を用いて統計処理を行っている入門書となっている。「The R Tips」は R を使いこなすための色々な情報が集約されているので、手元に一冊あると重宝するだろう。R でどんなことができるかについては「The R Book」が様々な分野での利用方法について網羅的に扱っている。

R に関する書籍の多くは、その本で利用しているデータや関数を利用できる関数を提供しているので、そのようなものがあるかどうかもテキスト選びの判断材料にするとよい。以下の参考文献には、そのようなサイトに関する情報を付加した。

その他、初心者にとって参考になりそうな Web サイトや参考文献について、以下にまとめておくが、最新の情報は RjpWiki の「R に関する日本語リンク」には多くの利用者により常に最新の情報が更新されているため、参考になる。

参考となるサイト

- 山本の R のページ　http://stat.sm.u-tokai.ac.jp/~yama/R/
- R project のホームページ　http://www.r-project.org/
- RjpWiki　http://www.okada.jp.org/RWiki/
- R-Tips(中央農業総合研究センター・竹澤)
  http://cse.naro.affrc.go.jp/takezawa/r-tips/r.html
- 統計処理ソフトウェア R についての Tips(群馬大・中澤)
  http://minato.sip21c.org/swtips/R.html
- R による統計処理 (群馬大・青木)　http://aoki2.si.gunma-u.ac.jp/R/

参考文献

[1] 山本義郎・藤野友和・久保田貴文 (2015) R によるデータマイニング入門, オーム社.

[2] 山本義郎・飯塚誠也・藤野友和 (2013) 統計データの視覚化 (R で学ぶデータサイエンス 12), 共立出版.

[3] 金 明哲 (2007) R によるデータサイエンス-データ解析の基礎から最新手法まで, 森北出版

[4] U. リゲス著，石田基広 訳 (2006) R の基礎とプログラミング技法, シュプリンガー・ジャパン.

[5] 間瀬 茂, 鎌倉稔成, 神保雅一, 金藤浩司 (2004) 工学のためのデータサイエンス入門—フリーな統計環境 R を用いたデータ解析, 数理工学社.

[6] 舟尾暢男 (2005) The R tips—データ解析環境 R の基本技・グラフィックス活用集　第 2 版, オーム社.

[7] J.H. Maindonald and John Braun (2003) Data Analysis and Graphics Using R: An Example-Based Approach, Cambridge Univ. Press.
(http://wwwmaths.anu.edu.au/~johnm/r-book.html)

[8] Peter Dalgaard (2002) Introductory Statistics With R, Springer Verlag.

[9] John Verzani (2005) Using R for Introductory Statistics, Chapman & Hall.
(http://wiener.math.csi.cuny.edu/UsingR/)