リトルブック：R による多変量解析

A Little Book of R For Multivariate Analysis, Release 0.1

2011 年 8 月 31 日

Avril Coghlan

日本語訳　荒木　孝治

2012 年 2 月 17 日

著者：Avril Coghlan（University College Cork, Cork, Ireland）．E メール：a.coghlan@ucc.ie

本ブックレットは，R を用いた時系列解析への簡単な入門書である[*1]．

本ブックレットの pdf 版は，https://github.com/avrilcoghlan/LittleBookofRMultivariateAnalysis/raw/master/_build/latex/MultivariateAnalysis.pdf より取得可能である．

本ブックレットが気に入った人は，“R による生物医学統計学”http://a-little-book-of-r-for-biomedical-statistics.readthedocs.org/ と “R による時系列解析”http://a-little-book-of-r-for-time-series.readthedocs.org/ もあるので参照してほしい．

[*1] 本翻訳に関するコメント・問い合わせ等は，荒木孝治（関西大学商学部，arakit@kansai-u.ac.jp）までお願いします．

目次

第 1 章

R のインストール方法

1.1 R とは

本ブックレットは，多変量解析のために R を利用する方法に関する入門を目的とする．

R（www.r-project.org）は，一般的に用いられるフリーの統計ソフトウェアである．R は，対話モードのみならず簡単なプログラミングにより統計分析を実行することができる．

1.2 R のインストール

R を使うには，R プログラムがコンピュータにインストールされている必要がある．

1.2.1 R が Windows パソコンにインストールされているかどうかを確認する方法

R をコンピュータにインストールする前にすべきことは，R が，例えば前のユーザーによって既にインストールされているかどうかを調べることである．

以下，主に Windows パソコンに関してその方法について説明するが，Macintosh または Linux コンピュータに関しても少し触れる．Windows パソコンの場合，R がコンピュータに既にインストールされているかどうか調べる方法には, 次に示す 2 つがある．

1. “R” のアイコンがコンピュータのデスクトップにあるかどうかを調べる．もしあれば，“R” アイコンをダブルクリックすることにより，R を起動することができる．無いときは，手順 2 に行く．
2. Windows のデスクトップの左下にある “スタート” ボタンをクリックし，ポップアップしたメニューの［全てのプログラム］の上にマウスを移動させる．表示されるプログラムのリストに “R” があるかどうかを確認する．もしあれば，すでに R がインストールされているので，リストから “R”（例えば，R 2.10.0（R X.X.X の X.X.X は R のバージョンを意味する））を選択することによって R をスタートさせることができる．

上記の (1) か (2) により R を起動できたら，既にコンピュータに R がインストールされることを意味する（どちらもだめなときは，R はまだインストールされていない）．インストールされている R のバージョンが古い場合，最新版をインストールする方がよい．最新の R の機能を利用することができるからである．

1.2.2 R の最新版を知る方法

R の最新バージョンは，CRAN（The Comprehensive R Archive Network）`http://cran.r-project.org/` で確認することができる．

“The latest release”（最新のリリース）の他にも（ページの途中に），“R-X.X.X.tar.gz”（例えば，

“R-2.12.1.tar.gz”）のような表示がある．これは，R の最新のバージョンが X.X.X（今の場合，2.12.1）であることを意味する．

R の開発は非常に活発なため，R の新しいバージョンのリリースは定期的に行われている（年におよそ 2 回）．定期的に R の新しいバージョンを確認し，それをインストールすることには価値がある（それにより，ダウンロードした R のパッケージのすべての最新版との互換性を確実にすることができる）．

1.2.3 Windows パソコンへの R のインストール

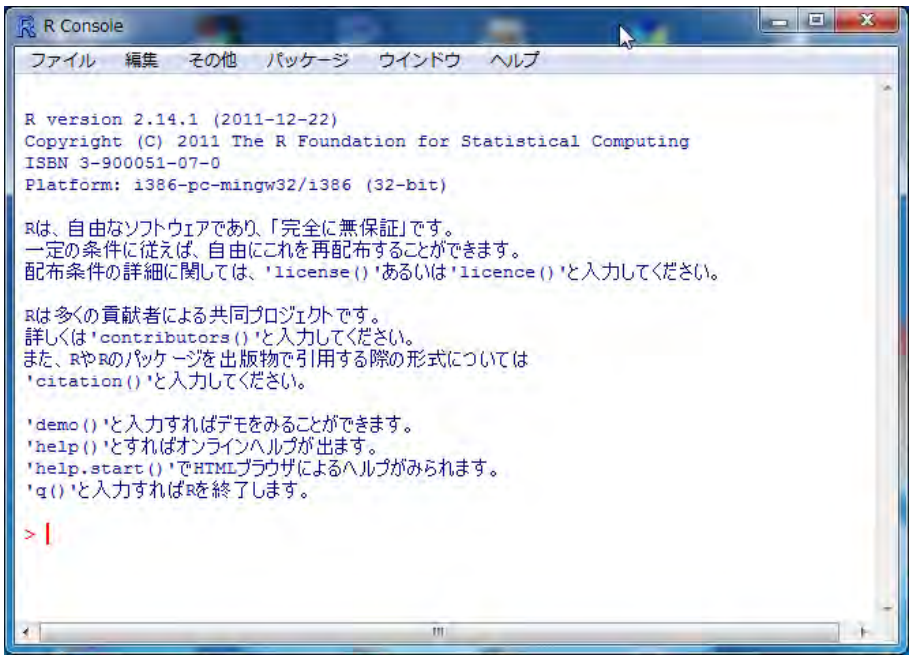
Windows パソコンに R をインストールするには，次の手順を実行する．

1. `http://essrc.hyogo-u.ac.jp/cran/` へ行く[*1]
2. “Download R for Windows” をクリック．
3. “Subdirectories” で “base” をクリック．
4. 次のページに,“Download R 2.12.1 for Windows” (R X.X.X．の “X.X.X” は R のバージョンを示す．例えば，R 2.12.1)[*2]といった表示がある． このリンクをクリック．
5. ファイル “R-2.12.1-win32.exe” をどう処理するか，つまり，保存するか開く（実行する）かを聞かれるので，“ファイルを保存する” を選択し，保存場所をデスクトップに指定する．保存後，ファイルをダブルクリックしてインストールを実行する．
6. インストール中に利用する言語を聞かれるので，英語（English）を選択する[*3]．
7. R のセットアップウイザードが起動するので，下にある “Next”（“次へ”）ボタンをクリックする．
8. “Information”（“情報”）ページが開かれる．“Next”（“次へ”）をクリック．
9. “Select Destination Location”（“インストール先の指定”）ページが開かれる．デフォルトのインストール場所は “C:¥Program Files” である．
10. セットアップウイザードの下にある “Next”（“次へ”）をクリックする．
11. “Select Components”（“コンポーネントの選択”）ページが開かれる．“Next”（“次へ”）をクリック．
12. “Startup options” ページが開かれる．“‘Next”（“次へ”）をクリック．
13. “Select Start Menu Folder” ページが開かれる．“Next” をクリック．
14. “Select Additional Tasks”（“追加タスクの選択”）ページが開かれる．“Next”（“次へ”）クリック．
15. これで R がインストールされる．インストールには 1 分くらいかかる．終了すると “Completing the R for Windows Setup Wizard”（“セットアップウィザードの完了”）が表示される．“Finish”（“完了”）をクリックする．
16. R を起動するには，手順 18 または 19 を実行する．
17. “R” のアイコンがコンピュータのデスクトップに表示されているかどうかを確認する．表示されている場合，R を起動するには “R” をダブルクリックする．表示されていない場合，手順 19 を実行する．
18. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” を選択する．次にプログラムのメニューから “R”（R X.X.X の “X.X.X” は R のバージョンを示し，例えば，R 2.12.1）を選択することにより，R を起動する．
19. R コンソールが表示される．

[*1] （訳注）日本のミラーサイトに変更．原著では，`http://ftp.heanet.ie/mirrors/cran.r-project.org`.

[*2] （訳注）原著では，バージョンが様々なので，表記を “R 2.12.1” に統一した（以下，同様）．

[*3] （訳注）Japanese（日本語）でのインストールを選択できるが，一部文字化けするので，英語でインストールする方がよい．



1.2.4 非 Windows オペレーティングシステム（例えば，Macintosh，または Linux）へのインストール

上記は，Windows パソコンへの R のインストール方法である．非 Windows オペレーティングシステム（OS）を利用するコンピュータ（例えば，Macintosh または Linux）の場合，`http://ftp.heanet.ie/mirrors/cran.r-project.org` から，OS に適切な R インストーラをダウンロードし，`http://ftp.heanet.ie/mirrors/cran.rproject` にある R のインストール方法に従う．

1.3 R のパッケージのインストール

R をインストールするとき，いくつかの標準パッケージも一緒にインストールされる．有用な他の R パッケージ（たとえば，“rmeta” パッケージ）を使う方法を本ブックレットでは説明する．これらの付加的なパッケージは R と一緒にインストールされないので，別にインストールする必要がある．

1.3.1 R のパッケージのインストール法

R を Windows パソコンに（上記のステップに従って）インストールした後，下記の手順でパッケージを追加的にインストールすることができる．

1. R を起動するために，次の手順 2 または 3 を実行する．
2. “R” のアイコンがデスクトップにあるかどうか調べる．あるなら，“R” アイコンをダブルクリックして R を起動する．ない場合，手順 3 を実行する．
3. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” から “R”（R X.X.X の X.X.X は R のバージョンで，例えば，R 2.12.1）を選択することにより，R を起動する．
4. R コンソールが表示される．
5. R を起動すると，R コンソールの上にある “パッケージのインストール” メニューより R のパッケージ（例えば，“rmeta”）をインストールすることができる．どのウェブサイトからダウンロードするかを聞かれるので，“Japan”（または他の国）を選択する．インストール可能なパッケージのリストが表示されるので，そのリストからインストールしたいパッケージ（例えば，

“rmeta”）を指定する．

6. これにより “rmeta” パッケージのインストールが始まる．
7. “rmeta” パッケージのインストールが終了する．この後，R を起動した後，R コンソールに次のコマンドを入力することにより “rmeta” パッケージをロードし，利用することができる．

```
> library("rmeta")
```

Bioconductor（http://www.bioconductor.org）というバイオインフォマティックスのための R パッケージの特殊な集合がある（例えば，“yeastExpData” や “Biostrings” といったパッケージの集まり）．Bioconductor パッケージ群は，Bioconductor に特有の手順（Bioconductor の R パッケージをインストールする方法を参照）によりインストールする必要がある．

1.3.2 Bioconductor のパッケージのインストール法

1.3.1 項に示した手順で，R パッケージの大部分をインストールすることができる．しかし，バイオインフォマティックス用の R パッケージの集合である Bioconductor に関しては，特別な手順に従う必要がある．Bioconductor（http://www.bioconductor.org）は，バイオインフォマティックスのために開発されたパッケージ群である．

1. R を起動するには，次の手順 2 または 3 を実行する．
2. “R” のアイコンがデスクトップにあるかどうか調べる．あるなら，“R” アイコンをダブルクリックして R を起動する．“R” アイコンがない場合，手順 3 を実行する．
3. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” から “R”（R X.X.X の X.X.X は R のバージョンで，例えば，R 2.12.1）を選択することにより，R を起動する．
4. R コンソールが表示される．
5. R を起動した後，R コンソールに次を入力する．

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

6. これにより Bioconductor の主要パッケージ（“affy”，“affydata”，“affyPLM”，“annaffy”，“annotate”，“Biobase”，“Biostrings”，“DynDoc”，“gcrma”，“genefilter”，“geneplotter”，“hgu95av2.db”，“limma”，“marray”，“matchprobes”，“multtest”，“ROC”，“vsn”，“xtable”，“affyQCReport”）がインストールされる．これには少し時間がかかる．
7. 後日，Bioconductor の主要パッケージ以外のパッケージ，例えば，“yeastExpData” をインストールする必要が生じたときは，R コンソールに次を入力する．

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("yeastExpData")
```

8. パッケージをインストールした後，それを利用するには R コンソールに次を入力する．

```
> library("yeastExpData")
```

1.4 R の起動

R を利用するには，先ず R プログラムを起動する．それには，R がインストールされている必要がある．R を起動するには次の手順 1 または 2 を実行する．

1. “R” のアイコンがデスクトップにあるかどうか調べる．あるなら，“R” アイコンをダブルクリッ

クして R を起動する．“R” アイコンがない場合，手順 2 を実行する．

2. コンピュータ・スクリーンの左下にある “スタート” ボタンをクリックし，“すべてのプログラム” から “R”（R X.X.X の “X.X.X” は R のバージョンで，例えば，R 2.12.1）を選択することにより，R を起動する．

これにより，R コンソールという新しいウインドウが表示される．

1.5 R 超入門

R 内で分析を実行するには，R コンソールにコマンドを入力する．R コンソールには次に示す記号が表示されている．

```
>
```

この記号（>）は R のプロンプトである．プロンプトの後ろに，特定の作業をするのに必要なコマンドを入力する．コマンドは，Return キーを入力後に実行される．R をいったん起動すると，コマンドを入力することにより R を利用でき，結果はすぐに表示される．例えば：

```
> 2*3
[1] 6
> 10-3
[1] 7
```

R が生成した全ての変数（スカラー，ベクトル，行列等）はオブジェクトと呼ばれる．R では，変数に値を与えるとき，矢印（<-）を用いる．例えば，値 2*3 を変数 x に与えるには次のようにする．

```
> x <- 2 * 3
```

R のオブジェクトの内容を見るには，その名前を入力するだけでよい．その内容が表示される．

```
> x
[1] 6
```

R には，スカラー，ベクトル，行列，配列，データフレーム，テーブル，リストといった様々な種類のオブジェクトがある．上記のスカラー変数 x は，R オブジェクトの 1 例である．スカラー変数はちょうど 1 つの要素を持つが，ベクトルはいくつかの要素から構成される．c() 関数（c は combine の c）を用いて，ベクトルを作成することができる．例えば値 8，6，9，10，5 の要素から構成される，*myvector* という名前のベクトルを作成するには，次を入力する．

```
> myvector <- c(8, 6, 9, 10, 5)
```

変数 *myvector* の内容を表示するには，その名前を入力するだけでよい．

```
> myvector
[1] 8 6 9 10 5
```

[1] はベクトルの 1 番目の要素を示すインデックスである．ベクトルの任意の要素を抽出するには，角括弧（[ ]）の中に要素のインデックスを与えたものをベクトルの名前の後ろに記載する．例えば，ベクトル *myvector* の 4 番目の要素の値を抽出するには，次のようにする．

```
> myvector[4]
[1] 10
```

リストは，ベクトルと異なり，数値と記号といった異なる型の要素を保持することができる．リストはまた，ベクトルなどの他の変数を含むこともできる．リストの作成には list() 関数を用いる．

例えば，*mylist* というリストを作るには，次のコマンドを入力する．

```
> mylist <- list(name="Fred", wife="Mary", myvector)
```

リスト *mylist* の内容を表示するには，その名前を入力する．

```
> mylist
$name
[1] "Fred"

$wife
[1] "Mary"

[[3]]
[1]  8  6  9 10  5
```

リストの要素には番号が付けられており，それをインデックスとして参照することができる．リストの要素を抽出するには，リスト名の後ろに，2 重の角括弧（[[ ]]）の中に要素のインデックスを与えたものを記載することにより可能である．だから，2 番目と 3 番目の要素を *mylist* から抽出するには次のようにする．

```
> mylist[[2]]
[1] "Mary"
> mylist[[3]]
[1] 8 6 9 10 5
```

リストの要素に名前を付けることができる．この場合，名前の後ろに “$” をつけ，続いて要素名を記載することでリストの要素を参照することができる．例えば，*mylist*$*name* は *mylist*[[1]] と同じであり，*mylist*$*wife* は *mylist*[[2]] と同じである．

```
> mylist$wife
[1] "Mary"
```

リスト中の名前が与えられた要素の名前を知るには，attributes() 関数を用いて次のようにする．

```
> attributes(mylist)
$names
[1] "name" "wife" ""
```

リスト変数が持つ名前付きの要素を知るために attributes() 関数を用いると，名前付き要素には常に，“$names” というヘッダーが付けられる．よって，リスト変数 *mylist* の名前付き要素の名前が “*name*” と “*wife*” であることがわかる．*mylist*$*name* や *mylist*$*wife* と入力することによってそれらの値を切り出すことができる．

R で利用する別のオブジェクトとして，テーブル変数がある．例えば，学級内の生徒の名前を含む名前のベクトル変数 *mynames* があるとする．table() 関数を利用して，*mynames* 内の同じ名前を持つ子供の数のテーブル変数を作るには次のようにする．

```
> mynames <- c("Mary", "John", "Ann", "Sinead", "Joe", "Mary", "Jim", "John", "Simon")
> table(mynames)
mynames
   Ann    Jim    Joe   John   Mary  Simon Sinead
     1      1      1      2      2      1      1
```

関数 table() によって作ったテーブル変数に，名前 “*mytable*” をつけて保存するには次を入力する．

```
> mytable <- table(mynames)
```

テーブル変数の要素にアクセスするには，リストの要素にアクセスするときと同様に，二重の角括弧（[[ ]]）を使う．例えば，*mytable* の 4 番目の要素（“John” という子供の数）にアクセスするには，

次を入力する．

```
> mytable[[4]]
[1] 2
```

表の 4 番目の要素の名前（“John”）を用いてテーブル要素の値を知ることもできる．

```
> mytable[["John"]]
[1] 2
```

R の関数は通常，引数を必要とする．引数は，関数に渡される入力変数（オブジェクト）であり，それは，演算を実行するときに利用される．例えば，log10() 関数は数を渡され，その数の，底が 10 の対数の値を計算する．

```
> log10(100)
[1] 2
```

R では，help() 関数を用いて特定の関数についてヘルプ画面を参照することができる．例えば，log10() 関数についてヘルプを参照したいとき，次を入力する．

```
> help("log10")
```

help() 関数を利用するとき，ウィンドウまたはウェブページが出現し，ヘルプを求める関数に関する情報が表示される．

関数の名前に確信がないが，その名前の一部がわかっているとき，help.search() と RSiteSearch() 関数を使って関数名を探すことができる．help.search() 関数は，興味があるトピックに関連する関数がすでにインストールされて（インストールされているパッケージのどれかに入って）いるかどうかを探す．RSiteSearch() 関数は，興味があるトピックに関連した関数を，すべての R 関数（まだインストールされていないパッケージに含まれるものを含んで）の中から探す．

例えば，標準偏差を計算する関数があるかどうかを知りたいとき，次を入力することにより，関数に関する記述の中から “deviation”（偏差）という語を含むすべてのインストールされた関数を探すことができる．

```
> help.search("deviation")
Help files with alias or concept or title matching
'deviation' using fuzzy matching:

genefilter::rowSds
                    Row variance and standard deviation of
                    a numeric array
nlme::pooledSD      Extract Pooled Standard Deviation
stats::mad          Median Absolute Deviation
stats::sd           Standard Deviation
vsn::meanSdPlot     Plot row standard deviations versus row
```

見つけた関数の中に，“stats” パッケージ（R のインストールとともにインストールされる基本パッケージ）の中に sd() 関数があり，これにより標準偏差を計算することができることがわかる．

上記の例では，help.search() 関数は関連した関数（sd()）を見つけた．help.search() で期待していたものを見つけることができない場合，RSiteSearch() 関数を使うことにより，R のウェブサイトで記述されているすべての関数の中から，興味があるトピックに関連するものを見つけることができるかもしれない．

```
> RSiteSearch("deviation")
```

RSiteSearch() 関数の結果は，R 関数の記述への，ならびにそれらの関数に関する R メーリングリスト議論へのヒット結果である．

R では，スカラーとベクトルといったオブジェクトを利用して計算することができる．例えば，ベクトル myvector の値の平均を計算する（すなわち，8，6，9，10，5 の平均）ために，mean() 関数を使うことができる．

```
> mean(myvector)
[1] 7.6
```

mean()，length()，print()，plot() といった R の組み込み関数を利用することができる．これに対し，よく利用する機能を持つ関数を独自に作成することもできる．例えば，入力された数の 2 乗に 20 を加えるための関数を作成するには次のようにする．

```
> myfunction <- function(x) { return(20 + (x*x)) }
```

return() 関数は，計算値を返す機能を持つ．この関数を入力した後，関数を使うことができる．例えば，異なる入力値（例えば，10，25）に対してこの関数を使うことができる．

```
> myfunction(10)
[1] 120
> myfunction(25)
[1] 645
```

R を終了するには次を入力する．

```
> q()
```

1.6 リンクと参考文献

さらに学習するためのリンクを紹介する．

R へのより詳細に入門するための良いオンライン・チュートリアルが，“Kickstarting R” というウェブサイト（cran.rproject.org/doc/contrib/Lemon-kickstart）にある．

別の素晴らしい，もう少し詳細なチュートリアルが，“Introduction to R” というウェブサイト（cran.rproject.org/doc/manuals/R-intro.html）にある．

1.7 謝辞

Friedrich Leisch と Phil Spector には，R のインストールに記述に関して非常に有益なコメントと提案をいただいた．感謝する．

1.8 連絡

訂正や改良に関する提案があれば，著者（Avril Coghlan）のメールアドレス（a.coghlan@ucc.ie）まで送付していただければ幸いである．

1.9 ライセンス

第 2 章

R による多変量解析

2.1 多変量解析

本ブックレットは，R を用いて簡単な多変量解析，特に，主成分分析（PCA）と線形判別分析（LDA）を実行する方法を示す．

読者は，多変量解析の入門的な知識を持つことを前提としている．なお，多変量解析そのものではなく，R を用いて多変量解析を実践する方法を説明することを目的とする．

多変量解析を知らなかったり，本ブックレットで説明する概念についてもっと知りたかったりする人には，Open University 刊行の本 “Multivariate Analysis”（製品コード $M249/03$）を参照することを勧める．

本ブックレットでは，UCI Machine Learning Repository（`http://archive.ics.uci.edu/ml`）からのデータセットを例として用いる．

本ブックレットの pdf バージョンは，`https://github.com/avrilcoghlan/LittleBookofRTimeSeries/raw/master/_build/latex/MultivariateAnalysis.pdf` より取得することができる．

本ブックレットを気に入った人は，“R を用いた生物医学統計”（`http://alittle-book-of-r-for-biomedical-statistics.readthedocs.org/`）や “R を用いた時系列解析”（`http://a-little-book-of-r-for-time-series.readthedocs.org/`）を参照してほしい．

2.2 多変量解析データの R への読み込み

多変量データを分析する際に必要なことは，先ずそれを R に読み込み，図に描くことである．read.table() 関数を用いて R にデータを読み込むことができる．

たとえば，ファイル `http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data` にある “*wine.data*” というデータセットは，イタリアの同一地域で栽培された 3 品種のぶどうから製造されたワインの 13 の化学成分である．

データセットの一部を次に示す．

```
1,14.23,1.71,2.43,15.6,127,2.8,3.06,.28,2.29,5.64,1.04,3.92,1065
1,13.2,1.78,2.14,11.2,100,2.65,2.76,.26,1.28,4.38,1.05,3.4,1050
1,13.16,2.36,2.67,18.6,101,2.8,3.24,.3,2.81,5.68,1.03,3.17,1185
1,14.37,1.95,2.5,16.8,113,3.85,3.49,.24,2.18,7.8,.86,3.45,1480
1,13.24,2.59,2.87,21,118,2.8,2.69,.39,1.82,4.32,1.04,2.93,735
...
```

行が 1 本のワインのデータである．第 1 列はワインの品種（ラベルは，1，2，3）であり，続く 13 列は，サンプルの 13 の化学成分である．列はカンマで区切られている．

このファイルを read.table() 関数を用いて R に読み込むとき，read.table() の中で引数`"sep="`を指定し，列がカンマで区切られていることを指定する．次のようにする．

```
> wine <- read.table("http://archive.ics.uci.edu/ml/machine-learning-databases/
+  wine/wine.data", sep=",")
> wine
    V1    V2   V3   V4   V5  V6   V7   V8   V9  V10       V11   V12  V13  V14
1    1 14.23 1.71 2.43 15.6 127 2.80 3.06 0.28 2.29  5.640000 1.040 3.92 1065
2    1 13.20 1.78 2.14 11.2 100 2.65 2.76 0.26 1.28  4.380000 1.050 3.40 1050
3    1 13.16 2.36 2.67 18.6 101 2.80 3.24 0.30 2.81  5.680000 1.030 3.17 1185
4    1 14.37 1.95 2.50 16.8 113 3.85 3.49 0.24 2.18  7.800000 0.860 3.45 1480
5    1 13.24 2.59 2.87 21.0 118 2.80 2.69 0.39 1.82  4.320000 1.040 2.93  735
...
176  3 13.27 4.28 2.26 20.0 120 1.59 0.69 0.43 1.35 10.200000 0.590 1.56  835
177  3 13.17 2.59 2.37 20.0 120 1.65 0.68 0.53 1.46  9.300000 0.600 1.62  840
178  3 14.13 4.10 2.74 24.5  96 2.05 0.76 0.56 1.35  9.200000 0.610 1.60  560
```

これにより，178 本のワインのサンプルに関するデータが，‘*wine*’ という名前で読み込まれる．

2.3 多変量データのグラフ化

多変量データセットを読み込んだ後，次に行うべきことはデータを図に描くことである．

2.3.1 散布図行列

多変量データをプロットする際によく用いられる手法は，散布図行列である．これは，変数のペアの全組み合わせの散布図を描くものである．これを実行するには，パッケージ “car” にある “scatterplotMatrix()” 関数を利用することができる．そのためには，あらかじめ “car” パッケージをインストールしておく必要がある（R のパッケージのインストールに関しては，1.3 節の「R のパッケージのインストール法」を参照のこと）．

“car” パッケージをインストールした後，次を入力して “car” パッケージをロードする．

```
> library("car")
```

次に，“scatterplotMatrix()” 関数を用いて図を描く．

プロットしたい変数を指定する．例えば，最初の 5 つの化学成分のみをプロットしたいとする．これらは，データセット “wine” の $2-6$ 列目に保存されている．これらを “*wine*” から切り出すには，次のようにする．

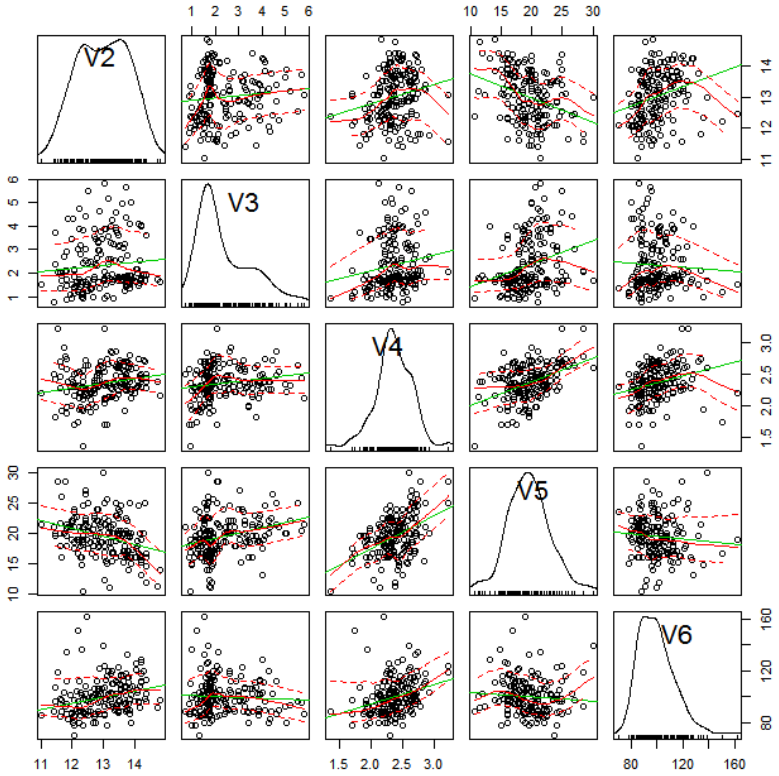
```
> wine[2:6]
      V2   V3   V4   V5  V6
1  14.23 1.71 2.43 15.6 127
2  13.20 1.78 2.14 11.2 100
3  13.16 2.36 2.67 18.6 101
4  14.37 1.95 2.50 16.8 113
5  13.24 2.59 2.87 21.0 118
  ...
```

これら 5 つの変数の散布図行列を描くには，次を入力する．

```
> scatterplotMatrix(wine[2:6])
```

作成した散布図行列の対角位置には各変数，今の場合，5 つの濃度（変数 $V2, V3, V4, V5, V6$）のヒストグラムが描かれている．

非対角位置には，5 つの変数の散布図が描かれている．例えば，1 行 2 列目は，$V3$ を x 軸，$V2$ を y 軸とする散布図である．



2.3.2　データ点をグループでラベル付けした散布図

散布図行列内に興味深い散布図があるとき，グループ（今の場合，品種別）でラベル付けしたより詳細な散布図を作成したくなることがある．

例えば，上記の散布図行列で，4 行，3 列目にあるのは，$V5$ が x 軸，$V4$ が y 軸の散布図である．この散布図より，変数 $V4$ と $V5$ の間に正の相関があることがわかる．

よって，2 つの変数 $V4$，$V5$ の散布図をグループ（品種）のラベルを付けて描くことにより，より詳細にこれらの間の関係を調べることができる．2 つの変数の散布図を作成するには，関数 “plot” を用いる．変数 $V4$ と $V5$ の値は，“*wine*” の $V4$，$V5$ 列に保存されているので，*wine*\$$V4$，*wine*\$$V5$ でアクセスすることができる．よって，次を入力することにより散布図をプロットすることができる．

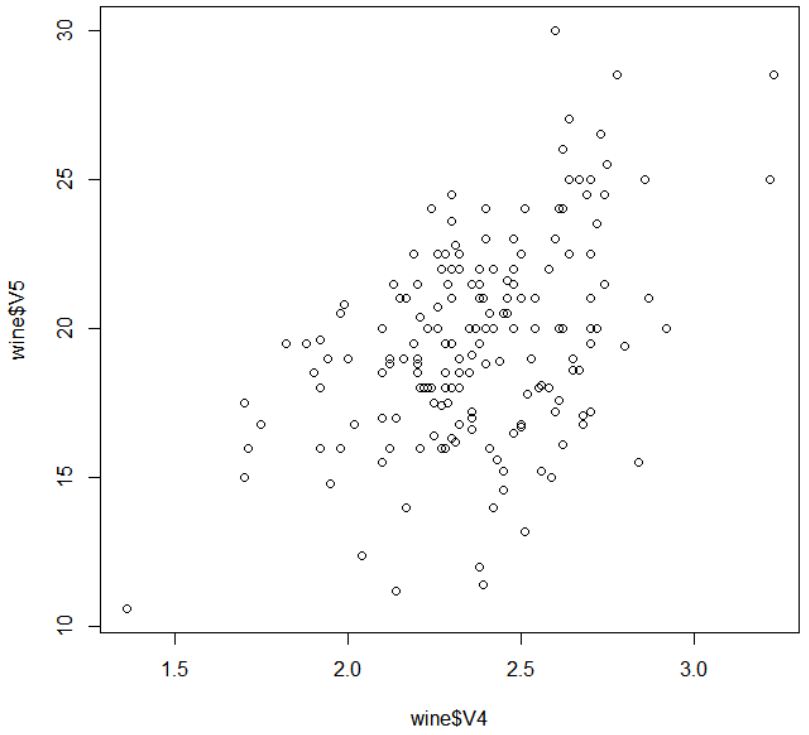
```
> plot(wine$V4, wine$V5)
```

データ点にグループ（今の場合，品種）をラベルとして付けたい場合，“text” 関数を利用して，データ点の横にテキストをプロットすることができる．本例では，ワインの品種は “*wine*” の $V1$ 列に保存されているので，次を入力する．

```
> text(wine$V4, wine$V5, wine$V1, cex=0.7, pos=4, col="red")
```

“text” 関数のヘルプを参照すると，“pos=4” オプションによりデータ点の記号の右にテキストを表示できることがわかる．“cex=0.5” オプションにより，テキストの大きさをデフォルトの大きさの半分に，“col=red” オプションにより，テキストの色を赤に変更している．上記のコマンドにより，次に示す図を作成することができる．

この散布図より，品種 2 のワインは品種 1 と比べて，$V4$ の値が低いことがわかる．



2.3.3 プロファイルプロット

他の有益なプロットとして，“profile plot”（プロファイルプロット）がある．これは，サンプルの各変数の値をプロットすることにより，各変数の変動を示すものである．

“makeProfilePlot()” 関数を用いて，プロファイルプロットを作成することができる．この関数は “RColor-Brewer” ライブラリを必要とする．この関数を利用するには，“RColor-Brewer” をあらかじめインストールしておく必要がある（R パッケージのインストール法については，「R パッケージのイ

ンストール法」を参照のこと).

```
> makeProfilePlot <- function(mylist, names)
  {
     require(RColorBrewer)
     # 変数の数の取得
     numvariables <- length(mylist)
     # 変数の数に対応したランダムな色の選択
     colours <- brewer.pal(numvariables, "Set1")
     # 変数の最小値と最大値を取得:
     mymin <- 1e+20
     mymax <- 1e-20
     for (i in 1:numvariables)
     {
        vectori <- mylist[[i]]
        mini <- min(vectori)
        maxi <- max(vectori)
        if (mini < mymin) { mymin <- mini }
        if (maxi > mymax) { mymax <- maxi }
     }
     # 変数のプロット
     for (i in 1:numvariables)
     {
        vectori <- mylist[[i]]
        namei <- names[i]
        colouri <- colours[i]
        if (i == 1) { plot(vectori,col=colouri,type="l", ylim=c(mymin,mymax)) }
        else { points(vectori, col=colouri, type="l") }
        lastxval <- length(vectori)
        lastyval <- vectori[length(vectori)]
        text((lastxval-10),(lastyval), namei, col="black", cex=0.6)
     }
  }
```

この関数を利用するには，まずこれをコピーし，R にペーストしておく必要がある．この関数の引数は，プロットしたい変数の名前を含むベクトルと，変数の値自体を含むリスト変数である．

例えば，最初の 5 つの変数（列 $V2$, $V3$, $V4$, $V5$, $V6$）のプロファイルプロットを作成するには，次を入力する．

```
> library(RColorBrewer)
> names <- c("V2", "V3", "V4", "V5", "V6")
> mylist <- list(wine$V2, wine$V3, wine$V4, wine$V5, wine$V6)
> makeProfilePlot(mylist, names)
```

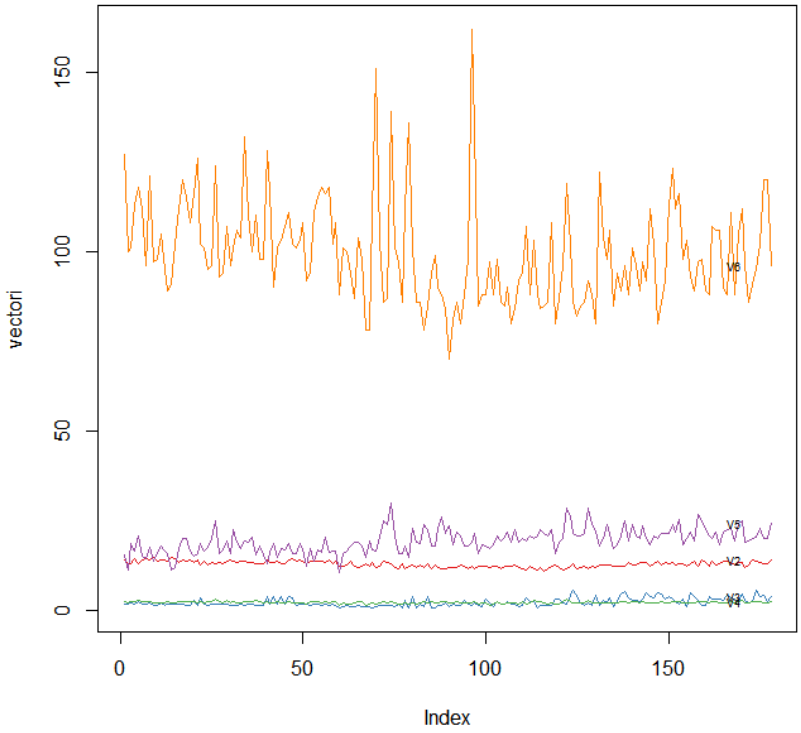
プロファイルプロットから，変数 $V6$ の平均と標準偏差は他の変数のものよりかなり大きいことがわかる．

2.4 多変量データの要約統計量の計算

多変量データセットの各変数の平均や標準偏差といった要約統計量の計算も必要である．

これは，R の “mean()” と “sd()” 関数を用いて簡単に実行できる．例えば，*wine* データセットの 13 個の変数の平均と標準偏差を計算したいとする．これらは “*wine*” の $2-14$ 列目に保存されているので，次を入力する[*1]．

[*1] （訳注）R のバージョンによっては，“警告メッセージ： mean(<data.frame>) is deprecated. Use colMeans() or




```
> mean(wine[2:14])
        V2         V3         V4         V5         V6         V7         V8
13.0006180  2.3363483  2.3665169 19.4949438 99.7415730  2.2951124  2.0292697
        V9        V10        V11        V12        V13        V14
 0.3618539  1.5908989  5.0580899  0.9574494  2.6116854 746.8932584
```

これにより，$V2$ の平均は 13.0006180，$V3$ の平均は 2.3363483 であること等がわかる．

同様に，13 個の変数の標準偏差を求めるには次を入力する．

```
> sd(wine[2:14])
 Use sapply(*, sd) instead.
        V2         V3         V4         V5         V6         V7         V8
 0.8118265  1.1171461  0.2743440  3.3395638 14.2824835  0.6258510  0.9988587
        V9        V10        V11        V12        V13        V14
 0.1244533  0.5723589  2.3182859  0.2285716  0.7099904 314.9074743
```

変数の標準偏差はかなり異なるので，変数を比較するには標準化する方がよいかもしれない－ $V14$ の標準偏差は 314.9074743 なのに，V9 の標準偏差は 0.1244533 にすぎない．だから，変数を比較するには，平均が 0，標準偏差が 1 になるように標準化する必要がある．変数を標準化する方法は，後で説明する．

2.4.1 グループ別の平均と分散

グループ別に，例えば，品種別に平均や標準偏差を求めたいことも多い．品種データは “*wine*” の列 “$V1$” に保存されている．

品種 2 のデータのみを抽出するには次のようにする．

```
> cultivar2wine <- wine[wine$V1=="2",]
```

---

sapply(*, mean) instead. ” というメッセージが表示される．これは mean() 関数が廃止予定のため，colMeans() または sapply() 関数を用いることを推奨しているからである．この警告を回避したい場合，例えば，`colMeans(wine[2:14])` とする．

これを用いて，13 個の化学成分の品種 2 に対する濃度の平均と標準偏差を計算することができる．

```
> mean(cultivar2wine[2:14])
       V2         V3         V4         V5         V6         V7         V8         V9
12.278732   1.932676   2.244789  20.238028  94.549296   2.258873   2.080845   0.363662
      V10        V11        V12        V13        V14
 1.630282   3.086620   1.056282   2.785352 519.507042
> sd(cultivar2wine[2:14])
        V2          V3          V4          V5          V6          V7          V8
 0.5379642   1.0155687   0.3154673   3.3497704  16.7534975   0.5453611   0.7057008
        V9         V10         V11         V12         V13         V14
 0.1239613   0.6020678   0.9249293   0.2029368   0.4965735 157.2112204
```

同様にして，品種 2，品種 3 の化学成分の濃度の平均，分散を計算することができる．しかし，次に示す関数 “printMeanAndSdByGroup()” を用いる方が便利だろう．これは，データセット内の各グループに対する層別した平均と標準偏差を計算する関数である．

```
> printMeanAndSdByGroup <- function(variables, groupvariable)
  {
     # 変数の数の取得
     variables <- as.data.frame(variables)
     numvariables <- length(variables)
     # グループ変数の取り得る値の数の取得
     groupvariable2 <- as.factor(groupvariable[[1]])
     levels <- levels(groupvariable2)
     numlevels <- length(levels)
     for (i in 1:numlevels)
     {
        leveli <- levels[i]
        levelidata <- variables[groupvariable==leveli,]
        groupsize <- nrow(levelidata)
        print(paste("グループ", leveli, "グループの大きさ:", groupsize))
        print(paste("グループ", leveli, "平均:"))
        print(mean(levelidata))
        print(paste("グループ", leveli, "標準偏差:"))
        print(sd(levelidata))
     }
  }
```

関数 “printMeanAndSdByGroup()” を利用するには，先ずそれをコピーし，R コンソールに貼り付けておく必要がある．この関数の引数は，平均と標準偏差を計算したい変数と，グループ情報を保存している変数である．例えば，3 つの異なる品種別に 13 個の化学成分の各々の平均と標準偏差を計算するには，次のコマンドを利用する．

```
> printMeanAndSdByGroup(wine[2:14], wine[1])
[1] "グループ 1 グループの大きさ: 59"
[1] "グループ 1 平均:"
 Use colMeans() or sapply(*, mean) instead.
        V2          V3          V4          V5          V6          V7          V8
 13.744746    2.010678    2.455593   17.037288  106.338983    2.840169    2.982373
        V9         V10         V11         V12         V13         V14
  0.290000    1.899322    5.528305    1.062034    3.157797 1115.711864
[1] "グループ 1 標準偏差:"
 Use sapply(*, sd) instead.
         V2          V3          V4          V5          V6          V7
 0.46212536  0.68854886  0.22716598  2.54632245 10.49894932  0.33896135
```

```
         V8          V9          V10          V11          V12          V13
 0.39749361  0.07004924  0.41210923  1.23857281  0.11648264  0.35707658
        V14
221.52076659
[1] "グループ 2 グループの大きさ: 71"
[1] "グループ 2 平均:"
 Use colMeans() or sapply(*, mean) instead.
       V2         V3         V4         V5         V6         V7         V8         V9
 12.278732   1.932676   2.244789  20.238028  94.549296   2.258873   2.080845   0.363662
      V10        V11        V12        V13        V14
  1.630282   3.086620   1.056282   2.785352 519.507042
[1] "グループ 2 標準偏差:"
 Use sapply(*, sd) instead.
        V2          V3          V4          V5          V6          V7          V8
 0.5379642   1.0155687   0.3154673   3.3497704  16.7534975   0.5453611   0.7057008
        V9         V10         V11         V12         V13         V14
 0.1239613   0.6020678   0.9249293   0.2029368   0.4965735 157.2112204
[1] "グループ 3 グループの大きさ: 48"
[1] "グループ 3 平均:"
 Use colMeans() or sapply(*, mean) instead.
        V2          V3          V4          V5          V6          V7          V8
 13.1537500   3.3337500   2.4370833  21.4166667  99.3125000   1.6787500   0.7814583
        V9         V10         V11         V12         V13         V14
  0.4475000   1.1535417   7.3962500   0.6827083   1.6835417 629.8958333
[1] "グループ 3 標準偏差:"
 Use sapply(*, sd) instead.
        V2          V3          V4          V5          V6          V7          V8
 0.5302413   1.0879057   0.1846902   2.2581609  10.8904726   0.3569709   0.2935041
        V9         V10         V11         V12         V13         V14
 0.1241396   0.4088359   2.3109421   0.1144411   0.2721114 115.0970432
```

“printMeanAndSdByGroup()” 関数は，各グループに含まれるサンプル数も表示する．今の例では，品種 1 のサンプルは 59 本，品種 2 は 71 本，品種 3 は 48 本あることがわかる．

2.4.2 変数のグループ間変動とグループ内変動

特定の変数に対するグループ内での分散を計算したい場合（例えば，特定の化学成分の濃度），次の “calcWithinGroupsVariance()” 関数を利用することができる．

```
> calcWithinGroupsVariance <- function(variable, groupvariable)
  {
     # find out how many values the group variable can take
     groupvariable2 <- as.factor(groupvariable[[1]])
     levels <- levels(groupvariable2)
     numlevels <- length(levels)
     # 各グループに対する平均と標準偏差:
     numtotal <- 0
     denomtotal <- 0
     for (i in 1:numlevels)
     {
        leveli <- levels[i]
        levelidata <- variable[groupvariable==leveli,]
        levelilength <- length(levelidata)
        # グループ i に対する平均と標準偏差:
        meani <- mean(levelidata)
        sdi <- sd(levelidata)
```

```
        numi <- (levelilength - 1) * (sdi * sdi)
        denomi <- levelilength
        numtotal <- numtotal + numi
        denomtotal <- denomtotal + denomi
    }
    # グループ内分散の計算
    Vw <- numtotal / (denomtotal - numlevels)
    return(Vw)
}
```

この関数を利用するには，先ず，R にコピー&ペーストしておく．例えば，変数 $V2$ のグループ内分散を計算するためには，次を入力する．

```
> calcWithinGroupsVariance(wine[2], wine[1])
[1] 0.2620525
```

$V2$ のグループ内分散は 0.2620525 である．

次に示す “calcBetween-GroupsVariance()” 関数を用いて，特定の変数（例えば，$V2$）のグループ間分散を求めることができる．

```
> calcBetweenGroupsVariance <- function(variable, groupvariable)
{
    # グループ変数が取る値の数の取得
    groupvariable2 <- as.factor(groupvariable[[1]])
    levels <- levels(groupvariable2)
    numlevels <- length(levels)
    # 全体平均の計算:
    grandmean <- mean(variable)
    # 各グループの平均と標準偏差の計算:
    numtotal <- 0
    denomtotal <- 0
    for (i in 1:numlevels)
    {
        leveli <- levels[i]
        levelidata <- variable[groupvariable==leveli,]
        levelilength <- length(levelidata)
        # get the mean and standard deviation for group i:
        meani <- mean(levelidata)
        sdi <- sd(levelidata)
        numi <- levelilength * ((meani - grandmean)^2)
        denomi <- levelilength
        numtotal <- numtotal + numi
        denomtotal <- denomtotal + denomi
    }
    # グループ間分散の計算
    Vb <- numtotal / (numlevels - 1)
    Vb <- Vb[[1]]
    return(Vb)
}
```

R にこの関数をコピー&ペーストしておくと，これを利用して $V2$ といった変数のグループ間分散を求めることができる．

```
> calcBetweenGroupsVariance (wine[2], wine[1])
[1] 35.39742
```

$V2$ のグループ間分散は 35.39742 である．

グループ間分散をグループ内分散で割ることにより変数の “分離度” を計算することができる．$V2$ の分離度は，次により求めることができる．

```
> 35.39742/0.2620525
[1] 135.0776
```

多変量データセット内の全ての変数の分離度を計算したいなら，次の関数 “calcSeparations()” を用いる．

```
> calcSeparations <- function(variables,groupvariable)
  {
     # 変数の数の取得
     variables <- as.data.frame(variables)
     numvariables <- length(variables)
     # 変数名の取得
     variablenames <- colnames(variables)
     # 各変数の分離度の計算
     for (i in 1:numvariables)
     {
        variablei <- variables[i]
        variablename <- variablenames[i]
        Vw <- calcWithinGroupsVariance(variablei, groupvariable)
        Vb <- calcBetweenGroupsVariance(variablei, groupvariable)
        sep <- Vb/Vw
        print(paste("変数", variablename, "Vw=", Vw, "Vb=", Vb, "分離度=",sep))
     }
  }
```

例えば，13 個の化学成分の濃度の分離度を求めるには，次を入力する．

```
> calcSeparations(wine[2:14],wine[1])
[1] "変数 V2 Vw= 0.262052469153907 Vb= 35.3974249602692 分離度= 135.0776242428"
[1] "変数 V3 Vw= 0.887546796746581 Vb= 32.7890184869213 分離度= 36.9434249631837"
[1] "変数 V4 Vw= 0.0660721013425184 Vb= 0.879611357248741 分離度= 13.312901199991"
[1] "変数 V5 Vw= 8.00681118121156 Vb= 286.41674636309 分離度= 35.7716374073093"
[1] "変数 V6 Vw= 180.65777316441 Vb= 2245.50102788939 分離度= 12.4295843381499"
[1] "変数 V7 Vw= 0.191270475224227 Vb= 17.9283572942847 分離度= 93.7330096203673"
[1] "変数 V8 Vw= 0.274707514337437 Vb= 64.2611950235641 分離度= 233.925872681549"
[1] "変数 V9 Vw= 0.0119117022132797 Vb= 0.328470157461624 分離度= 27.5754171469659"
[1] "変数 V10 Vw= 0.246172943795542 Vb= 7.45199550777775 分離度= 30.2713831702276"
[1] "変数 V11 Vw= 2.28492308133354 Vb= 275.708000822304 分離度= 120.664018441003"
[1] "変数 V12 Vw= 0.0244876469432414 Vb= 2.48100991493829 分離度= 101.3167953903"
[1] "変数 V13 Vw= 0.160778729560982 Vb= 30.5435083544253 分離度= 189.972320578889"
[1] "変数 V14 Vw= 29707.6818705169 Vb= 6176832.32228483 分離度= 207.920373902178"
```

グループ（品種）を最もよく分離する変数は，$V8$ である（分離度は 333.9）[*2]．後で議論するが，線形判別分析（LDA）の目的は，グループを最もよく分離する変数の線形結合を求めることにある．これにより，個々の変数が達成するベストの分離（ここでは，分離度 233.9 の変数 $V8$）よりも良くなることが期待される．

2.4.3 2 つの変数のグループ間共分散とグループ内共分散

異なるグループからのサンプリング単位を記述する変数（例えば，異なる品種のワイン）を持つ多変量データセットの場合，変数のペアのグループ内共分散やグループ間分散に関心があることもよく

[*2] （訳注）出力中，Vb はグループ間分散，Vw はグループ内分散．

ある．

これは，次の関数を用いることにより可能である．

```
> calcWithinGroupsCovariance <- function(variable1, variable2, groupvariable)
  {
     # グループ変数が取る値の数の取得
     groupvariable2 <- as.factor(groupvariable[[1]])
     levels <- levels(groupvariable2)
     numlevels <- length(levels)
     # 各グループの変数 1 と変数 2 の共分散の計算:
     Covw <- 0
     for (i in 1:numlevels)
     {
        leveli <- levels[i]
        levelidata1 <- variable1[groupvariable==leveli,]
        levelidata2 <- variable2[groupvariable==leveli,]
        mean1 <- mean(levelidata1)
        mean2 <- mean(levelidata2)
        levelilength <- length(levelidata1)
        # このグループに対する共分散の計算:
        term1 <- 0
        for (j in 1:levelilength)
        {
           term1 <- term1 + ((levelidata1[j] - mean1) * (levelidata2[j] - mean2))
        }
        Cov_groupi <- term1 # このグループに対する共分散
        Covw <- Covw + Cov_groupi
     }
     totallength <- nrow(variable1)
     Covw <- Covw / (totallength - numlevels)
     return(Covw)
  }
```

例えば，変数 $V8$ と $V11$ のグループ内共分散を計算するには，次を入力する．

```
> calcWithinGroupsCovariance(wine[8], wine[11], wine[1])
[1] 0.2866783
> calcBetweenGroupsCovariance <- function(variable1, variable2, groupvariable)
  {
     # グループ変数の取り得る値の数の取得
     groupvariable2 <- as.factor(groupvariable[[1]])
     levels <- levels(groupvariable2)
     numlevels <- length(levels)
     # 全体平均の計算
     variable1mean <- mean(variable1)
     variable2mean <- mean(variable2)
     # グループ間共分散の計算
     Covb <- 0
     for (i in 1:numlevels)
     {
        leveli <- levels[i]
        levelidata1 <- variable1[groupvariable==leveli,]
        levelidata2 <- variable2[groupvariable==leveli,]
        mean1 <- mean(levelidata1)
        mean2 <- mean(levelidata2)
        levelilength <- length(levelidata1)
        term1 <- (mean1 - variable1mean) * (mean2 - variable2mean) * (levelilength)
```

```
        Covb <- Covb + term1
    }
    Covb <- Covb / (numlevels - 1)
    Covb <- Covb[[1]]
    return(Covb)
  }
```

例えば，変数 $V8$ と $V11$ のグループ間共分散を計算するには，次を入力する．

```
> calcBetweenGroupsCovariance(wine[8],wine[11],wine[1])
[1] -60.41077
```

変数 $V8$ と $V11$ のグループ間共分散は -60.41 であり，グループ内共分散は 0.29 である．グループ内共分散がプラスなので（0.29），$V8$ と $V11$ はグループ内では正の関係がある．つまり，同一グループの個体において，$V8$ の値が大きくなると $V11$ の値も大きくなり，逆も成り立つ．グループ間共分散がマイナスであるため（-60.41），グループ間では $V8$ と $V11$ は負の相関がある．よって，$V8$ の平均が高いグループは，$V11$ の平均は低くなり，逆も成り立つ．

2.5 多変量データの相関の計算

多変量データセットの変数間の相関が有意かどうかを知りたいことがある．

変数のペアの線形（ピアソン）相関係数を求めるには，R の “cor.test()” 関数を用いるとよい．例えば，化学成分の濃度の最初の 2 つの変数 V2 と V3 の相関係数を求めるには，次を入力する．

```
> cor.test(wine$V2, wine$V3)

        Pearson's product-moment correlation

data:  wine$V2 and wine$V3
t = 1.2579, df = 176, p-value = 0.2101
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.05342959  0.23817474
sample estimates:
       cor
0.09439694
```

出力より，相関係数は 0.094（出力中の cor の値）であり，これは非常に弱い相関である．さらに，無相関の検定（相関係数が 0 と異なって有意かどうかの検定）の P 値は，0.21 である．これは 0.05 より遙かに大きく（この値は統計的有意性の判断のための基準値である[*3]），相関は 0 ではないということへの証拠としては非常に弱い．

変数がたくさんある場合，変数の各ペアの相関係数を求めるために “cor.test()” を用いることができるが，最も高い相関を持つ変数のペアを知りたいだけのときもある．これには，次に示す “mosthighlycorrelated()” 関数を用いる．

関数 “mosthighlycorrelated()” は，データセット内の変数のペアの線形相関係数を大きさの順に表示する．これにより，最も相関の大きな変数のペアを知ることができる．

```
> mosthighlycorrelated <- function(mydataframe, numtoreport)
  {
     # 相関係数の計算
     cormatrix <- cor(mydataframe)
     # 対角位置と下側三角位置の相関を 0 に設定
```

[*3] （訳注）有意水準のこと

```
    # これにより，最大には含まれない:
    diag(cormatrix) <- 0
    cormatrix[lower.tri(cormatrix)] <- 0
    # 行列の次元と行名の取得:
    numrows <- nrow(cormatrix)
    therownames <- rownames(cormatrix)
    # 相関の最大の取得
    sorted <- sort(abs(cormatrix), decreasing=TRUE)
    for (i in 1:numtoreport)
    {
       corri <- sorted[i]
       # find the pair of variables with this correlation
       for (j in 1:(numrows-1))
       {
          for (k in (j+1):numrows)
          {
             corrjk <- cormatrix[j,k]
             if (corri == abs(corrjk))
             {
                rowname <- therownames[j]
                colname <- therownames[k]
                print(paste("i=", i, "変数", rowname, "と", colname, "相関係数=", corrjk))
                            }
          }
       }
    }
 }
```

この関数を利用するには，これを R にコピー&ペーストしておく必要がある．引数として，表示したい相関係数の数を与える（例えば，上位の 10 個や上位の 20 個の値を表示するよう指定する）．

例えば，13 個の化学成分の濃度の相関係数を求め，上位の 10 個を表示するには，次を入力する．

```
> mosthighlycorrelated(wine[2:14], 10)
[1] "i= 1 変数 V7 と V8 相関係数= 0.864563500095115"
[1] "i= 2 変数 V8 と V13 相関係数= 0.787193901866952"
[1] "i= 3 変数 V7 と V13 相関係数= 0.699949364791186"
[1] "i= 4 変数 V8 と V10 相関係数= 0.652691768607515"
[1] "i= 5 変数 V2 と V14 相関係数= 0.643720037178213"
[1] "i= 6 変数 V7 と V10 相関係数= 0.612413083780036"
[1] "i= 7 変数 V12 と V13 相関係数= 0.565468293182659"
[1] "i= 8 変数 V3 と V12 相関係数= -0.561295688664945"
[1] "i= 9 変数 V2 と V11 相関係数= 0.546364195083704"
[1] "i= 10 変数 V8 と V12 相関係数= 0.54347856648999"
```

最も大きな線形相関係数を持つ変数のペアは，$V7$ と $V8$ である（相関係数は約 0.86）．

2.6 変数の標準化

単位が異なり，分散も非常に異なる変数を比較したい場合，先ず変数を標準化する．

例えば, ワイン・サンプルの 13 の化学成分の濃度の標準偏差は, $V9$ の 0.1244533（分散は 0.01548862）から $V14$ の 314.9074743（分散は 99166.72）とかなり幅広い．これは分散でみると $6,402,554$ 倍の違いがある．

そのため，ワインのサンプルの主成分分析（PCA : Principal Component Analysis）を行う場合，標準化しない変数を用いることはよくない．なぜなら，第 1 主成分は，最大分散を持つ変数 $V14$ によっ

て支配されてしまうからである．

だから，先ず変数を標準化し（平均を 0，分散を 1），標準化された変数に対して主成分分析を適用する方がよい．これにより，オリジナルのデータの変動の最良の低次元表現を与える主成分を見つけることが可能となる．“scale()” 関数を用いて変数を標準化することができる．

例えば，ワインサンプルの 13 個の変数を標準化するには次を入力する．

```
> standardisedconcentrations <- as.data.frame(scale(wine[2:14]))
```

“scale()” 関数の出力をデータフレームに変換するために，“as.data.frame()” 関数を利用していることに注意．データフレームは，ワインデータセットと同じデータの型である．

“*standardisedconcentrations*” に保存された標準化変数の平均が 0，分散が 1 となっていることを，次の入力により確認することができる．

```
> mean(standardisedconcentrations)
           V2            V3            V4            V5            V6            V7
-8.591766e-16 -6.776446e-17  8.045176e-16 -7.720494e-17 -4.073935e-17 -1.395560e-17
           V8            V9           V10           V11           V12           V13
 6.958263e-17 -1.042186e-16 -1.221369e-16  3.649376e-17  2.093741e-16  3.003459e-16
          V14
-1.034429e-16
```

標準化された変数の平均は全て非常に小さな値であるが，本質的に 0 である．標準化された変数の標準偏差は全て 1 である．

2.7 主成分分析

主成分分析の目的は，多変量データセットの変数の低次元での最適な表現を得ることである．例えば，ワインデータセットには，3 つの異なる品種からのワイン・サンプルの 13 個の化学成分の濃度の変数がある．より少ない数の新しい変数を求め，それらがサンプル間の変動のほとんどを説明できるかどうかを主成分分析により確認することができる．新しい変数は，13 個の化学成分の濃度の全て，またはいくつかの線形結合である．多変量データセットに主成分分析（PCA）を適用するとき，最初のステップは，“scale()” 関数を用いて変数を標準化することである．これは，変数の分散が非常に異なるときに（今の場合そうであるが），必要である．

変数を標準化した後，R の “prcomp()” 関数を用いて主成分分析を実行することができる．

例えば，ワインサンプルの 13 個の変数を標準化し，主成分分析を実行するには次を入力する．

```
> standardisedconcentrations <- as.data.frame(scale(wine[2:14])) # 変数の標準化
> wine.pca <- prcomp(standardisedconcentrations) # PCA の実行
```

“prcomp()” 関数による主成分分析の結果に対する要約情報を，“summary()” 関数を用いて表示することができる．

```
> summary(wine.pca)
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7     PC8
Standard deviation     2.169 1.5802 1.2025 0.95863 0.92370 0.80103 0.74231 0.59034
Proportion of Variance 0.362 0.1921 0.1112 0.07069 0.06563 0.04936 0.04239 0.02681
Cumulative Proportion  0.362 0.5541 0.6653 0.73599 0.80162 0.85098 0.89337 0.92018
                          PC9   PC10    PC11    PC12    PC13
Standard deviation     0.53748 0.5009 0.47517 0.41082 0.32152
Proportion of Variance 0.02222 0.0193 0.01737 0.01298 0.00795
Cumulative Proportion  0.94240 0.9617 0.97907 0.99205 1.00000
```

これは，各主成分の標準偏差，各主成分が説明する分散の寄与率を与える．

主成分の標準偏差は，“prcomp()” 関数の出力の “*sdev*” という要素に保存されている．

```
> wine.pca$sdev
 [1] 2.1692972 1.5801816 1.2025273 0.9586313 0.9237035 0.8010350 0.7423128 0.5903367
 [9] 0.5374755 0.5009017 0.4751722 0.4108165 0.3215244
```

主成分で説明される全分散は，主成分の分散の合計である．

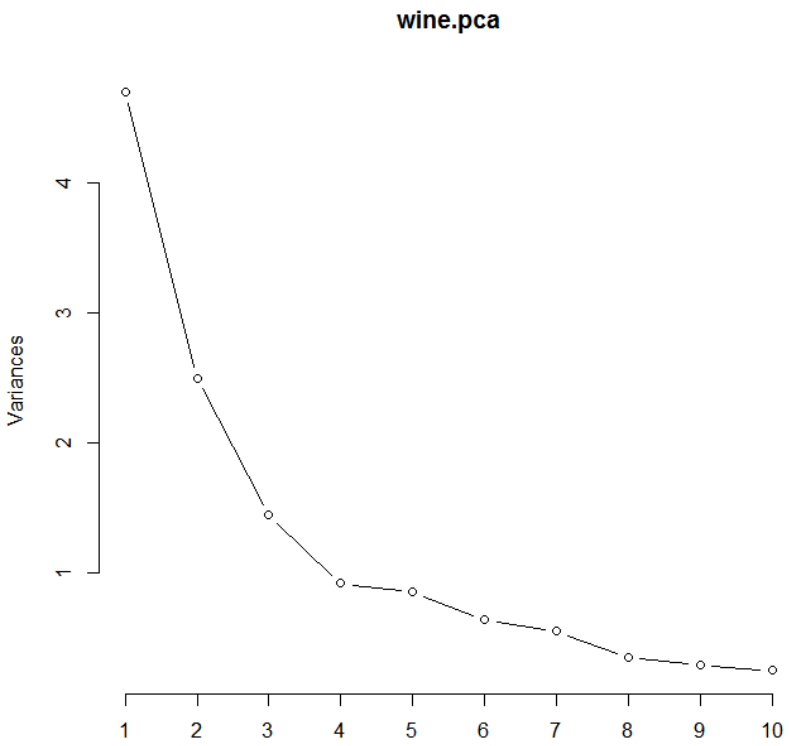
```
> sum((wine.pca$sdev)^2)
[1] 13
```

今の場合，全分散は 13 であり，これは変数の数（13 変数）と等しい．なぜなら，標準化された変数の分散は 1 だからである．全分散は個々の変数の分散の和であり，標準化された変数の分散は 1 なので，全分散は変数の数（今の場合 13）と等しくなる．

2.7.1 保持する主成分数の決定

いくつの主成分を保持すべきかを決定するために，スクリープロットを作成することが多い．

```
> screeplot(wine.pca, type="lines")
```



スクリープロットにおいて，最も明確な変化は第 4 主成分で生じている．これはスクリープロットの “肘” に相当する．ゆえに，最初の 3 つの主成分を保持すればよい．

主成分数を決定する他の方法に，Kaiser（カイザー）基準がある．これは，「分散が 1（標準化変数に対する主成分分析の場合）を超える主成分を保持する」というものである．これは，各主成分の分散を見ることにより確認することができる．

```
> (wine.pca$sdev)^2
 [1] 4.7058503 2.4969737 1.4460720 0.9189739 0.8532282 0.6416570 0.5510283 0.3484974
 [9] 0.2888799 0.2509025 0.2257886 0.1687702 0.1033779
```

分散が 1 を超えているのは，第 1，2，3 主成分である（分散はそれぞれ，4.71，2.50，1.45）．よって，カイザー基準では，上位の 3 つの主成分を保持すればよい．

主成分を保持する数を決定する第 3 の方法は，最小の累積寄与率を保証できる主成分数で決定することである．例えば，少なくとも 80% の累積寄与率が必要なら，上位の 5 つの主成分を保持する必要がある．“summary(wine.pca)” の出力結果より，上位の 5 つの主成分の寄与率の合計は 80.2% である（上位の 4 つの主成分の寄与率の合計は 73.6% なので，これでは充分ではない）．

2.7.2 主成分に対する負荷量

主成分への負荷量は，prcomp() によって返される変数の要素 “rotation” に保存される．これは各主成分の負荷量の行列であり，その行列の第 1 列には第 1 主成分の負荷量，第 2 列には第 2 主成分の負荷量という形になっている．

だから，ワイン・サンプルの 13 の化学濃度の主成分分析において，第 1 主成分の負荷量を得るには，次を入力する：

```
> wine.pca$rotation[,1]
      V2          V3          V4          V5          V6          V7
-0.144329395 0.245187580  0.002051061 0.239320405 -0.141992042 -0.394660845
      V8          V9          V10         V11         V12         V13
-0.422934297 0.298533103 -0.313429488 0.088616705 -0.296714564 -0.376167411
      V14
-0.286752227
```

第 1 主成分は，元の変数とこの負荷量との一次結合である．つまり，$-0.144 * Z2 + 0.245 * Z3 + 0.002 * Z4 + 0.239 * Z5 - 0.142 * Z6 - 0.395 * Z7 - 0.423 * Z8 + 0.299 * Z9 - 0.313 * Z10 + 0.089 * Z11 - 0.297 * Z12 - 0.376 * Z13 - 0.287 * Z14$ であり，ここで，$Z2, Z3, Z4, ..., Z14$ は変数 $V2, V3, V4, ..., V14$ を標準化したものである（よって，平均は 0，分散は 1）．

負荷量の 2 乗和は 1 であり，これは負荷量を求めるときの制約式である．

```
> sum((wine.pca$rotation[,1])^2)
  [1] 1
```

負荷量と変数の値が与えられたときに主成分の値を計算するための独自の関数を定義することができる：

```
> calcpc <- function(variables,loadings)
  {
     # データセット中のサンプル数の取得 t
     as.data.frame(variables)
     numsamples <- nrow(variables)
     # 主成分を保存するベクトルの作成
     pc <- numeric(numsamples)
     # 変数の数の取得
     numvariables <- length(variables)
     # 各サンプルに対する主成分値の計算
     for (i in 1:numsamples)
     {
        valuei <- 0
        for (j in 1:numvariables)
        {
           valueij <- variables[i,j]
           loadingj <- loadings[j]
           valuei <- valuei + (valueij * loadingj)
        }
```

```
        pc[i] <- valuei
    }
    return(pc)
}
```

関数 “calcpc()” を利用して，ワインデータの各サンプルに対する第 1 主成分の値を計算することができる：

```
> calcpc(standardisedconcentrations, wine.pca$rotation[,1])
  [1] -3.30742097 -2.20324981 -2.50966069 -3.74649719 -1.00607049 -3.04167373
  [7] -2.44220051 -2.05364379 -2.50381135 -2.74588238 -3.46994837 -1.74981688
 [13] -2.10751729 -3.44842921 -4.30065228 -2.29870383 -2.16584568 -1.89362947
 [19] -3.53202167 -2.07865856 -3.11561376 -1.08351361 -2.52809263 -1.64036108
 [25] -1.75662066 -0.98729406 -1.77028387 -1.23194878 -2.18225047 -2.24976267
 [31] -2.49318704 -2.66987964 -1.62399801 -1.89733870 -1.40642118 -1.89847087
 [37] -1.38096669 -1.11905070 -1.49796891 -2.52268490 -2.58081526 -0.66660159
...
```

第 1 主成分の値は，“prcomp()” 関数により返される変数 $wine.pca\$x[1]$ に保存されている．これらの値は，上記の計算値と一致しなければならない：

```
> wine.pca$x[,1]
  [1] -3.30742097 -2.20324981 -2.50966069 -3.74649719 -1.00607049 -3.04167373
  [7] -2.44220051 -2.05364379 -2.50381135 -2.74588238 -3.46994837 -1.74981688
 [13] -2.10751729 -3.44842921 -4.30065228 -2.29870383 -2.16584568 -1.89362947
 [19] -3.53202167 -2.07865856 -3.11561376 -1.08351361 -2.52809263 -1.64036108
 [25] -1.75662066 -0.98729406 -1.77028387 -1.23194878 -2.18225047 -2.24976267
 [31] -2.49318704 -2.66987964 -1.62399801 -1.89733870 -1.40642118 -1.89847087
 [37] -1.38096669 -1.11905070 -1.49796891 -2.52268490 -2.58081526 -0.66660159
...
```

一致していることがわかる．

第 1 主成分は，$V8$ (− 0.423)，$V7$ (− 0.395)，$V13$ (− 0.376)，$V10$ (− 0.313)，$V12$ (− 0.297)，$V14$ (− 0.287)，$V9$ (0.299)，$V3$ (0.245)，$V5$ (.239) に対して（絶対値で）大きな負荷量を持つ．$V8$，$V7$, $V13$, $V10$, $V12$, $V14$ に対する負荷量は負で，$V9$, $V3$, $V5$ に対しては正である．よって，第 1 主成分は，濃度 $V8, V7, V13, V10, V12, V14$ と濃度 $V9, V3$, $V5$ との間の対比となっていると解釈できる．

同様に，第 2 主成分に対する負荷量は次のようにして求めることができる．

```
> wine.pca$rotation[,2]
         V2          V3          V4          V5          V6          V7
 0.483651548 0.224930935 0.316068814 -0.010590502 0.299634003 0.065039512
         V8          V9         V10         V11         V12         V13
-0.003359812 0.028779488 0.039301722 0.529995672 -0.279235148 -0.164496193
        V14
 0.364902832
```

これから，第 2 主成分は変数の次の形の線形結合であることが分かる：$0.484 * Z2 + 0.225 * Z3 + 0.316 * Z4 - 0.011 * Z5 + 0.300 * Z6 + 0.065 * Z7 - 0.003 * Z8 + 0.029 * Z9 + 0.039 * Z10 + 0.530 * Z11 - 0.279 * Z12 - 0.164 * Z13 + 0.365 * Z14$．ここで，$Z2, Z3, Z4, ..., Z14$ は変数 $V2, V3, V4, ..., V14$ を標準化したもので，その平均は 0，分散は 1 である．

第 1 主成分の場合と同様，負荷量の 2 乗和は 1 となる．

```
> sum((wine.pca$rotation[,2])^2)
[1] 1
```

第 2 主成分は，$V11(0.530)$，$V2(0.484)$，$V14(0.365)$，$V4(0.316)$，$V6(0.300)$，$V12(-0.279)$，$V3(0.225)$ に対して高い負荷量を持つ．$V11, V2, V14, V4, V6$，$V3$ に対する負荷量は正で，$V12$ に対しては負である．よって，第 2 主成分は，濃度 $V11, V2, V14, V4, V6$，$V3$ と濃度 $V12$ との間の対比となっていると解釈できる．

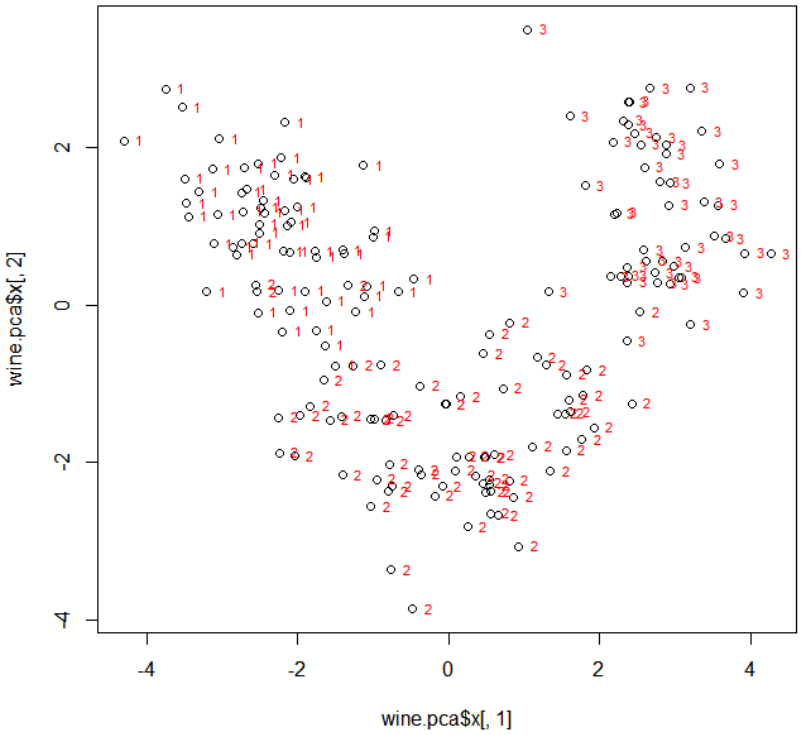
2.7.3 主成分の散布図

主成分得点は，“prcomp()” によって返される “x” という要素に保存される．これは主成分から構成される行列を含む．行列の第 1 列は第 1 主成分の値，第 2 列は第 2 主成分の値である．

だから，今の例において，“wine.pca$x[,1]” は第 1 主成分の値，“wine.pca$x[,2]” は第 2 主成分の値である．

第 1，第 2 主成分得点の散布図を作り，ワイン・サンプルの品種をラベルとしてデータ点に付加することができる：

```
> plot(wine.pca$x[,1],wine.pca$x[,2]) # 散布図の作成
> text(wine.pca$x[,1],wine.pca$x[,2], wine$V1, cex=0.7, pos=4, col="red") # ラベルの付加
```



この散布図の x 軸は第 1 主成分得点，y 軸は第 2 主成分得点である．散布図より，品種 1 のワイン・サンプルが品種 3 のワイン・サンプルより非常に低い第 1 主成分得点を持つことがわかる．したがって，第 1 主成分は品種 1 のワイン・サンプルと品種 3 のサンプルとを分離する．

また，品種 2 のワイン・サンプルが品種 1，3 のワイン・サンプルより非常に高い第 2 主成分の値を持つこともわかる．したがって，第 2 主成分は，品種 2 のサンプルと品種 1，3 のサンプルとを分離する．

よって，最初の 2 つの主成分は，3 つの異なる品種のワイン・サンプルの分離にかなり役立つ．

第 1 主成分を $V8$，7，$V13$，$V10$，$V12$，$V14$ の濃度と $V9$，$V3$，$V5$ の濃度との対比であると解釈した．“printMeanAndSdByGroup()” 関数（上記参照）を用いて，標準化された濃度の品種で層別した平均を表示することにより，このことが正しいかどうかを確認することができる：

```
> printMeanAndSdByGroup(standardisedconcentrations,wine[1])
```

```
[1] "Group 1 Group size: 59"
[1] "Group 1 Means:"
        V2         V3         V4         V5         V6         V7         V8
 0.9166093 -0.2915199  0.3246886 -0.7359212  0.4619232  0.8709055  0.9541923
        V9        V10        V11        V12        V13        V14
-0.5773564  0.5388633  0.2028288  0.4575567  0.7691811  1.1711967
[1] "Group 1 Standard Deviations:"
       V2        V3        V4        V5        V6        V7        V8        V9
0.5692415 0.6163463 0.8280333 0.7624716 0.7350927 0.5416007 0.3979478 0.5628555
      V10       V11       V12       V13       V14
0.7200189 0.5342623 0.5096112 0.5029315 0.7034472
[1] "Group 2 Group size: 71"
[1] "Group 2 Means:"
         V2          V3          V4          V5          V6          V7          V8
-0.88921161 -0.36134241 -0.44370614  0.22250941 -0.36354162 -0.05790375  0.05163434
         V9         V10         V11         V12         V13         V14
 0.01452785  0.06880790 -0.85039994  0.43239084  0.24460431 -0.72207310
[1] "Group 2 Standard Deviations:"
       V2        V3        V4        V5        V6        V7        V8        V9
0.6626591 0.9090742 1.1498967 1.0030563 1.1730101 0.8713912 0.7065071 0.9960462
      V10       V11       V12       V13       V14
1.0519061 0.3989712 0.8878480 0.6994087 0.4992299
[1] "Group 3 Group size: 48"
[1] "Group 3 Means:"
         V2          V3          V4          V5          V6          V7          V8
 0.18862653  0.89281222  0.25721896  0.57544128 -0.03004191 -0.98483874 -1.24923710
         V9         V10         V11         V12         V13         V14
 0.68817813 -0.76413110  1.00857281 -1.20199161 -1.30726231 -0.37152953
[1] "Group 3 Standard Deviations:"
       V2        V3        V4        V5        V6        V7        V8        V9
0.6531461 0.9738258 0.6732065 0.6761844 0.7625055 0.5703767 0.2938394 0.9974790
      V10       V11       V12       V13       V14
0.7142999 0.9968323 0.5006795 0.3832607 0.3654948
```

第 1 主成分は品種 1 と品種 3 を分離するといってよいだろうか？品種 1 において，$V8$ (0.954)，$V7$ (0.871)，$V13$ (0.769)，$V10$ (0.539)，$V12$ (0.458)，$V14$ (1.171) の平均値は，$V9$ (-0.577)，$V3$ (-0.292)，$V5$ (-0.736) の平均値と比較してかなり大きい．品種 3 において，$V8$ (-1.249)，$V7$ (-0.985)，$V13$ (-1.307)，$V10$ (-0.764)，$V12$ (-1.202)，$V14$ (-0.372) の平均値は，$V9$ (0.688)，$V3$ (0.893)，$V5$ (0.575) の平均値と比較してかなり小さい．よって，第 1 主成分は，$V8$, $V7$，$V13$, $V10$, $V12$, $V14$ と $V9$, $V3$, $V5$ との対比であり，第 1 主成分は，品種 1 と品種 3 とを分離する．

第 2 主成分に関しても，これは $V11$, $V2$, $V14$, $V4$, $V6$, $V3$ の濃度と，$V12$ の濃度との対比であると解釈した．品種で層別したこれらの変数の平均値に基づいて，第 2 主成分が品種 2 を品種 1 と 3 分離するといってよいだろうか？品種 1 では，$V11$ (0.203)，$V2$ (0.917)，$V14$ (1.171)，$V4$ (0.325)，$V6$ (0.462)，$V3$ (-0.292) の平均値は，$V12$ (0.458) の平均値とあまり違わない．品種 3 では，$V11$ (1.009)，$V2$ (0.189)，$V14$ (-0.372)，$V4$ (0.257)，$V6$ (-0.030)，$V3$ (0.893) の平均値は，$V12$ (-1.202) の平均値ともあまり違わない．これとは対照的に，品種 2 において，$V11$ (-0.850)，$V2$ (-0.889)，$V14$ (-0.722)，$V4$ (-0.444)，$V6$ (-0.364)，$V3$ (-0.361) の平均値は，$V12$ (0.432) の平均値と比べてかなり小さい．よって，第 2 主成分は $V11$, $V2$, $V14$, $V4$, $V6$, $V3$ と $V9$, $V3$, $V5$ との対比であり，第 2 主成分は，品種 2 と品種 1，3 とを分離する．

2.8 線形判別分析

主成分分析の目的は，多変量データセットが持つ変動の最適な低次元表現を見つけることであった．ワイン・データセットには，3 つの品種からのワインのサンプルを記述する 13 の化学成分の濃度があった．主成分分析を実行することによって，サンプル間の化学濃度の変動の大部分は最初の 2 つの主成分によって捕らえることができることがわかった．このとき各主成分は，13 の化学濃度の一次結合であった．

線形判別分析（LDA）の目的は，我々のデータセットでグループ（ワインの品種）を最も良く分離する変数（13 の化学成分濃度）の一次結合を見つけることにある．線形判別分析は，「正準判別分析」または単に「判別分析」とも呼ばれる．

ワインを栽培品種に分類することを考える．品種は 3 つなので，グループの数 (G) は 3 であり，変数の数は 13（13 の化学薬品の濃度; $p = 13$）である．ワインの品種を分離するときに利用できる判別関数の最大数は $G - 1$ と p の最小値であり，今の場合，2 と 13 の小さい方なので，2 個である．

R の “MASS” パッケージにある “lda()” 関数を用いて，線形判別分析を実行することができる．この関数を利用するには，“MASS” パッケージをインストールしておく必要がある（「R のパッケージをインストールする」を参照のこと）．

例えば，ワイン・サンプルで 13 の化学濃度を用いた線形判別分析を実行するには，次のようにする：

```
> library("MASS") # MASS パッケージのロード
> wine.lda <- lda(wine$V1 ~ wine$V2 + wine$V3 + wine$V4 + wine$V5 + wine$V6 + wine$V7 +
  wine$V8 + wine$V9 + wine$V10 + wine$V11 + wine$V12 + wine$V13 + wine$V14)
```

2.8.1 判別関数の負荷量

ワイン・データに対する判別関数の負荷量を表示するには，次を入力する：

```
> wine.lda
Call:
lda(wine$V1 ~ wine$V2 + wine$V3 + wine$V4 + wine$V5 + wine$V6 +
    wine$V7 + wine$V8 + wine$V9 + wine$V10 + wine$V11 + wine$V12 +
    wine$V13 + wine$V14)

Prior probabilities of groups:
        1         2         3
0.3314607 0.3988764 0.2696629

Group means:
   wine$V2  wine$V3  wine$V4  wine$V5  wine$V6  wine$V7   wine$V8  wine$V9 wine$V10
1 13.74475 2.010678 2.455593 17.03729 106.3390 2.840169 2.9823729 0.290000 1.899322
2 12.27873 1.932676 2.244789 20.23803  94.5493 2.258873 2.0808451 0.363662 1.630282
3 13.15375 3.333750 2.437083 21.41667  99.3125 1.678750 0.7814583 0.447500 1.153542
  wine$V11  wine$V12 wine$V13  wine$V14
1 5.528305 1.0620339 3.157797 1115.7119
2 3.086620 1.0562817 2.785352  519.5070
3 7.396250 0.6827083 1.683542  629.8958

Coefficients of linear discriminants:
                  LD1           LD2
wine$V2  -0.403399781  0.8717930699
wine$V3   0.165254596  0.3053797325
wine$V4  -0.369075256  2.3458497486
```

```
wine$V5   0.154797889 -0.1463807654
wine$V6  -0.002163496 -0.0004627565
wine$V7   0.618052068 -0.0322128171
wine$V8  -1.661191235 -0.4919980543
wine$V9  -1.495818440 -1.6309537953
wine$V10  0.134092628 -0.3070875776
wine$V11  0.355055710  0.2532306865
wine$V12 -0.818036073 -1.5156344987
wine$V13 -1.157559376  0.0511839665
wine$V14 -0.002691206  0.0028529846

Proportion of trace:
   LD1    LD2
0.6875 0.3125
```

この出力は，第 1 判別関数は次の形の変数の線形結合であることを意味する：$-0.403 * V2 - 0.165 * V3 - 0.369 * V4 + 0.155 * V5 - 0.002 * V6 + 0.618 * V7 - 1.661 * V8 - 1.496 * V9 + 0.134 * V10 + 0.355 * V11 - 0.818 * V12 - 1.158 * V13 - 0.003 * V14$．ここで，$V2, V3, ..., V14$ はワインデータに含まれる 13 個の化学薬品の濃度である．便宜上，判別関数の各値は平均の値が 0 になるよう，尺度変換されている（下記参照）．

これらの負荷量は，次に示すように，各グループ（品種）に対する判別関数のグループ内分散が 1 になるように計算されていることに注意．この尺度変換は，lda() 関数が返す変数の “scaling” という要素に保存されている．この要素は，第 1 列が第 1 判別関数の負荷量，第 2 列が第 2 判別分析の負荷量という形の行列である．

例えば，第 1 判別関数の負荷量を取得するには次のようにする：

```
> wine.lda$scaling[,1]
     wine$V2      wine$V3      wine$V4      wine$V5      wine$V6      wine$V7
-0.403399781  0.165254596 -0.369075256  0.154797889 -0.002163496  0.618052068
     wine$V8      wine$V9     wine$V10     wine$V11     wine$V12     wine$V13
-1.661191235 -1.495818440  0.134092628  0.355055710 -0.818036073 -1.157559376
    wine$V14
-0.002691206
```

負荷量と入力変数の値を与えて第 1 主成分スコアを計算する関数を独自に定義することができる：

```
> calclda <- function(variables,loadings)
  {
     # データセット中のサンプル数の取得
     as.data.frame(variables)
     numsamples <- nrow(variables)
     # 判別関数を保存するベクトルの作成
     ld <- numeric(numsamples)
     # 変数の数の取得
     numvariables <- length(variables)
     # 各サンプルに対する判別関数の値の計算
     for (i in 1:numsamples)
     {
        valuei <- 0
        for (j in 1:numvariables)
        {
           valueij <- variables[i,j]
           loadingj <- loadings[j]
           valuei <- valuei + (valueij * loadingj)
        }
```

```
        ld[i] <- valuei
    }
    # 平均が 0 になるように判別関数を標準化:
    ld <- as.data.frame(scale(ld, center=TRUE, scale=FALSE))
    ld <- ld[[1]]
    return(ld)
}
```

関数 calclda() は，データセット内の各サンプルに対する判別関数の値を計算する．例えば，第 1 判別関数の場合，各サンプルに対して式 $-0.403*V2-0.165*V3-0.369*V4+0.155*V5-0.002*V6+0.618*V7-1.661*V8-1.496*V9+0.134*V10+0.355*V11-0.818*V12-1.158*V13-0.003*V14$ を用いて計算する．さらに，calclda() 関数の中で “scale()” 関数を用いて，判別関数の値（例えば第 1 判別関数）を標準化する．その結果，平均値（すべてのワイン・サンプルに対して）は 0 となる．

ワイン・データの各サンプルに対して第 1 判別関数の値を計算するには，関数 calclda() を次のように用いる：

```
> calclda(wine[2:14], wine.lda$scaling[,1])
  [1] -4.70024401 -4.30195811 -3.42071952 -4.20575366 -1.50998168 -4.51868934
  [7] -4.52737794 -4.14834781 -3.86082876 -3.36662444 -4.80587907 -3.42807646
 [13] -3.66610246 -5.58824635 -5.50131449 -3.18475189 -3.28936988 -2.99809262
 [19] -5.24640372 -3.13653106 -3.57747791 -1.69077135 -4.83515033 -3.09588961
 [25] -3.32164716 -2.14482223 -3.98242850 -2.68591432 -3.56309464 -3.17301573
 [31] -2.99626797 -3.56866244 -3.38506383 -3.52753750 -2.85190852 -2.79411996
...
```

第 1 線形判別関数の値は，“predict()” 関数を使って計算することもできるので，上で定義して計算したものと比較することができる．これらは一致しなければならない：

```
> wine.lda.values <- predict(wine.lda, wine[2:14])
> wine.lda.values$x[,1] # 第 1 判別関数の値を表示
          1           2           3           4           5           6           7
-4.70024401 -4.30195811 -3.42071952 -4.20575366 -1.50998168 -4.51868934 -4.52737794
          8           9          10          11          12          13          14
-4.14834781 -3.86082876 -3.36662444 -4.80587907 -3.42807646 -3.66610246 -5.58824635
         15          16          17          18          19          20          21
-5.50131449 -3.18475189 -3.28936988 -2.99809262 -5.24640372 -3.13653106 -3.57747791
         22          23          24          25          26          27          28
-1.69077135 -4.83515033 -3.09588961 -3.32164716 -2.14482223 -3.98242850 -2.68591432
         29          30          31          32          33          34          35
-3.56309464 -3.17301573 -2.99626797 -3.56866244 -3.38506383 -3.52753750 -2.85190852
         36          37          38          39          40          41          42
-2.79411996 -2.75808511 -2.17734477 -3.02926382 -3.27105228 -2.92065533 -2.23721062
...
```

一致していることがわかる．

入力変数を標準化することが必要な主成分分析とは異なり，線形判別分析の場合，入力変数が標準化されているかどうかは重要でない．しかし，線形判別分析で標準化された変数を用いると，線形判別関数の負荷量の解釈がより容易になる．

線形判別分析で入力変数を標準化した場合，グループ内分散は 1，平均は 0 になる．平均を変数の各々の値から引いて，グループ内の標準偏差によって割ることによって，“グループ内で標準化された” 変数を計算することができる．グループ内で標準化された変数の値を計算するために，次に示す関数 “groupStandardise()” を使うことができる：

```
> groupStandardise <- function(variables, groupvariable)
```

```
{
   # 変数の数の取得
   variables <- as.data.frame(variables)
   numvariables <- length(variables)
   # 変数名の取得
   variablenames <- colnames(variables)
   # 変数のグループ内での標準化
   for (i in 1:numvariables)
   {
      variablei <- variables[i]
      variablei_name <- variablenames[i]
      variablei_Vw <- calcWithinGroupsVariance(variablei, groupvariable)
      variablei_mean <- mean(variablei)
      variablei_new <- (variablei - variablei_mean)/(sqrt(variablei_Vw))
      data_length <- nrow(variablei)
      if (i == 1) { variables_new <- data.frame(row.names=seq(1,data_length)) }
      variables_new[‘variablei_name‘] <- variablei_new
   }
   return(variables_new)
}
```

ワイン・サンプルの化学濃度を品種のグループ内で標準化するには，“groupStandardise()” 関数を次のように利用する：

```
> groupstandardisedconcentrations <- groupStandardise(wine[2:14], wine[1])
```

次に，グループ内で標準化された変数に対して線形判別分析を実行するために，lda() 関数を使う：

```
> wine.lda2 <- lda(wine$V1 ~ groupstandardisedconcentrations$V2 + groupstandardisedconcentrations$V3
                           + groupstandardisedconcentrations$V4 + groupstandardisedconcentrations$V5
                           + groupstandardisedconcentrations$V6 + groupstandardisedconcentrations$V7
                           + groupstandardisedconcentrations$V8 + groupstandardisedconcentrations$V9
                           + groupstandardisedconcentrations$V10 + groupstandardisedconcentrations$V11
                           + groupstandardisedconcentrations$V12 + groupstandardisedconcentrations$V13
                           + groupstandardisedconcentrations$V14)

> wine.lda2
...
Coefficients of linear discriminants:
                                          LD1          LD2
groupstandardisedconcentrations$V2  -0.20650463  0.446280119
groupstandardisedconcentrations$V3   0.15568586  0.287697336
groupstandardisedconcentrations$V4  -0.09486893  0.602988809
groupstandardisedconcentrations$V5   0.43802089 -0.414203541
groupstandardisedconcentrations$V6  -0.02907934 -0.006219863
groupstandardisedconcentrations$V7   0.27030186 -0.014088108
groupstandardisedconcentrations$V8  -0.87067265 -0.257868714
groupstandardisedconcentrations$V9  -0.16325474 -0.178003512
groupstandardisedconcentrations$V10  0.06653116 -0.152364015
groupstandardisedconcentrations$V11  0.53670086  0.382782544
groupstandardisedconcentrations$V12 -0.12801061 -0.237174509
groupstandardisedconcentrations$V13 -0.46414916  0.020523349
groupstandardisedconcentrations$V14 -0.46385409  0.491738050
...
```

グループ内で標準化された変数を用いて求めた第 1 判別関数において，（絶対値で）大きな負荷量を持つ変数は，$V8$（-0.871），$V11$ (0.537)，$V13$（-0.464），$V14$（-0.464），$V5$ (0.438) となってい

る．元の（標準化されていない）変数に対する負荷量ではなくグループ内で標準化された変数に対して計算される負荷量を解釈することには意味がある．グループ内で標準化された変数を用いて求めた第 1 判別関数において，大きな（絶対で）負荷量を持つ変数は，$V8$ (-0.871), $V11$ (0.537)，$V13$ (-0.464)，$V14$ (-0.464)，$V5$ (0.438) である．$V11$, $V5$ に対する負荷量は正であるが，$V8$, 13, $V14$ に対する負荷量は負である．したがって，判別関数は $V8$, $V13$, $V14$ と $V11$, $V5$ との対比を表すと考えることができる．分離度を尺度として，グループを最もよく分離する変数は $V8$（分離度 233.93)，$V14$ (207.92)，$V13$ (189.97)，$V2$ (135.08)，$V11$ (120.66) であることを既に見た．これらの多くは，線形判別関数で大きな負荷量を持つ変数と同じである（$V8$ に対する負荷量は-0.871, V14に対しては-0.464, V13に対しては-0.464, V11に対しては 0.537）．

変数 $V8$ と $V11$ は負のグループ間共分散（-60.41）を持ち，正のグループ内共分散（0.29）を持つことがわかっている．2 変数のグループ間共分散とグループ内共分散が逆の符号を持つとき，それらを単独で利用するより一次結合を用いる方がよりよく分離することを意味する．

このように，グループ間とグループ内の共分散の符号が異なる 2 変数 $V8$, $V11$ があり，これらを個別に利用するときに最も大きくグループを分離するとき，これらが第 1 判別関数において最大の負荷量を持つ 2 変数となることは意外ではない．

標準化しない変数に対する負荷量と比べて，グループ内で標準化された変数の負荷量の方が解釈が容易になるが，判別関数の値は同じであることに注意．例えば，ワイン・データに対して，標準化しない変数およびグループ内で標準化された変数を使って求める第 1 判別関数の値を計算することができる：

```
> wine.lda.values <- predict(wine.lda, wine[2:14])
> wine.lda.values$x[,1] # 標準化しない変数を用いた第 1 判別関数の値
          1           2           3           4           5           6
-4.70024401 -4.30195811 -3.42071952 -4.20575366 -1.50998168 -4.51868934
          7           8           9          10          11          12
-4.52737794 -4.14834781 -3.86082876 -3.36662444 -4.80587907 -3.42807646
         13          14          15          16          17          18
-3.66610246 -5.58824635 -5.50131449 -3.18475189 -3.28936988 -2.99809262
         19          20          21          22          23          24
-5.24640372 -3.13653106 -3.57747791 -1.69077135 -4.83515033 -3.09588961
         25          26          27          28          29          30
...
> wine.lda.values2 <- predict(wine.lda2, groupstandardisedconcentrations)
> wine.lda.values2$x[,1] # 標準化した変数を用いた第 1 判別関数の値
          1           2           3           4           5           6
-4.70024401 -4.30195811 -3.42071952 -4.20575366 -1.50998168 -4.51868934
          7           8           9          10          11          12
-4.52737794 -4.14834781 -3.86082876 -3.36662444 -4.80587907 -3.42807646
         13          14          15          16          17          18
-3.66610246 -5.58824635 -5.50131449 -3.18475189 -3.28936988 -2.99809262
         19          20          21          22          23          24
-5.24640372 -3.13653106 -3.57747791 -1.69077135 -4.83515033 -3.09588961
         25          26          27          28          29          30
...
```

標準化しない場合の負荷量と標準化した場合の負荷量の大きさは異なるが，第 1 判別関数の値は同じであることがわかる．

2.8.2 判別関数によって達成される分離

各判別関数によって達成される分離を計算するには，判別関数（例えば，第 1 判別関数の場合，$-0.403 * V2 - 0.165 * V3 - 0.369 * V4 + 0.155 * V5 - 0.002 * V6 + 0.618 * V7 - 1.661 * V8 - 1.496 *$

$V9 + 0.134 * V10 + 0.355 * V11 - 0.818 * V12 - 1.158 * V13 - 0.003 * V14$）に変数の値を代入して各判別関数の値を計算し，次に，それらの平均がゼロとなるように判別関数の値を標準化する．

“predict()” 関数を用いて R で実行することもできる．例えば，ワイン・データに対して判別関数の値を計算するには，次を入力する：

```
> wine.lda.values <- predict(wine.lda, standardisedconcentrations)
```

返された変数は，線形判別関数を含む行列である名前付き要素 “x” を持っている：x の第 1 列は第 1 判別関数を，第 2 列は第 2 の判別関数を含む（判別関数が 3 つ以上ある場合は同様）．従って，“calcSeparations()” 関数を利用して，ワイン・データの 2 つの線形判別関数によって達成できる分離度を，グループ間変動とグループ内変動の比として計算する．

```
> calcSeparations(wine.lda.values$x,wine[1])
[1] "変数 LD1 Vw= 1 Vb= 794.652200566216 分離度= 794.652200566216"
[1] "変数 LD2 Vw= 1 Vb= 361.241041493455 分離度= 361.241041493455"
```

上述のように，また，calcSeparations() の出力からわかるように，各判別関数の負荷量は，各グループ（品種）におけるグループ内分散（Vw）が 1 になるように計算されている．calcSeparations() の出力より，第 1（最良の）判別関数によって達成される分離度は 794.7 であり，第 2（2 番目に良い）判別関数によって達成される分離度は 361.2 であることがわかる．

よって，全体の分離度はこれらの合計であり，小数点第 2 位に丸めると 1155.89 となる（$794.652200566216 + 361.241041493455 = 1155.893$）．したがって，第 1 判別関数によって達成される分離度は 68.75%（$794.652200566216 * 100/1155.893$）であり，第 2 判別関数による分離度は 31.25%（$361.241041493455 * 100/1155.893$）である．“*wine.lda*”（lda() 関数によって帰された変数）と入力するときに表示される “proportion of trace” は，各判別関数によって達成される分離度（パーセント）である．

```
> wine.lda
Proportion of trace:
   LD1    LD2
0.6875 0.3125
```

第 1 判別関数のみで 3 つのグループ（品種）をうまく分離するが，第 2 判別関数はそれを改善するので，第 2 判別関数も利用する方がよい．したがって，グループをよりよく分離するには，最初の 2 つの判別関数両者を使う方がよい．

個別の変数（個々の化学濃度）が達成する分離度の最大値は $V8$ の 233.9 であったが，これは第 1 判別関数で達成される 794.7 よりかなり小さい．したがって，判別関数を計算するために複数の変数を使と，単独変数のみで達成できるよりはるかに大きな分離度を達成する判別関数を見つけることができる．

lda() 関数によって返される変数は，“*svd*” という要素を持ち，これは線形判別式変数に対するグループ間標準偏差とグループ内標準偏差の比である．この比は，calcSeparations() を用いて計算した “分離度” の平方根である．だから，“svd” に保存されている値の 2 乗値は，calcSeparations() を使って求めたものと一致する必要がある：

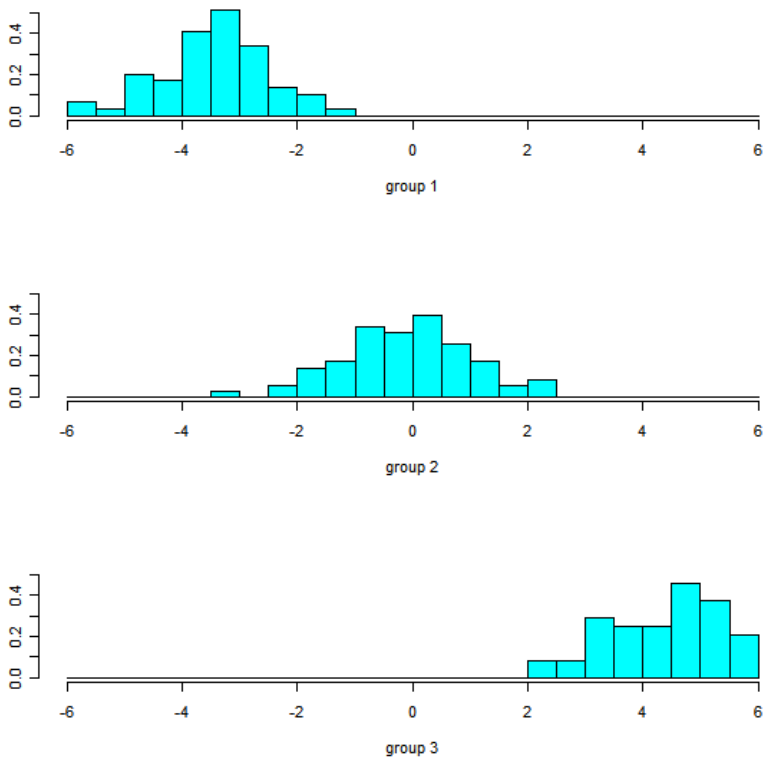
```
> (wine.lda$svd)^2
[1] 794.6522 361.2410
```

2.8.3 LDA 値の積み重ねヒストグラム

線形判別分析（LDA）の結果を示す良い方法に，グループ（本例では品種）で層別した判別関数値の積み重ねヒストグラムを作ることがある．R では “ldahist()” 関数を用いて実行することができる．例

えば，3 つの異なる品種のワイン・サンプルに対して求めた第 1 判別関数の値の積み重ねヒストグラムを表示するには次を入力する：

```
> ldahist(data = wine.lda.values$x[,1], g=wine$V1)
```



ヒストグラムより，品種 1 の第 1 判別関数の値は −6 から − 1 の間に，品種 3 の値は 2 から 6 の間にあり，重なりはない．よって，品種 1 と 3 が第 1 判別関数によってうまく分離されていることがわかる．しかし，訓練データに対する線形判別関数による分離は，過大評価である可能性があることに注意．

第 1 判別関数がどれくらいうまくグループを分離しているかをより正確に評価するには，“検証用データ”，つまり，判別関数を計算するのに用いなかったデータセットに対して求めた品種の結果の積み重ねヒストグラムを作成する必要がある．第 1 判別関数は，品種 1 と 3 を非常によく分離しているが，品種 1 と 2，または品種 2 と 3 の分離はそれほどよく無いことがわかる．

次に，第 2 判別関数の値の積み重ねヒストグラムより，第 2 判別関数がそれらの品種をうまく分離するかどうか調べる：

```
> ldahist(data = wine.lda.values$x[,2], g=wine$V1)
```

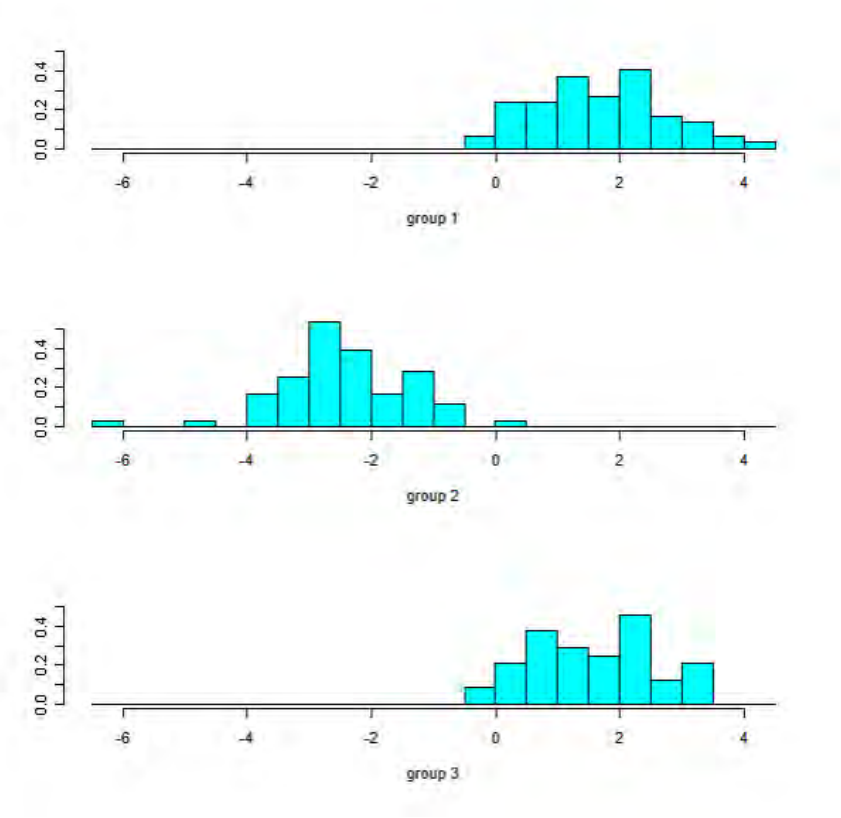
第 2 判別関数が非常にうまく品種 1 と 2 を分離しているが，少し共通部分もあることがわかる．さらに，第 2 判別関数は品種 2 と 3 も非常にうまく分離しているが，少し共通部分があるため，完全ではないことがわかる．

以上より，品種を分離するのに 2 つの判別関数を用いる方がよいことがわかった．

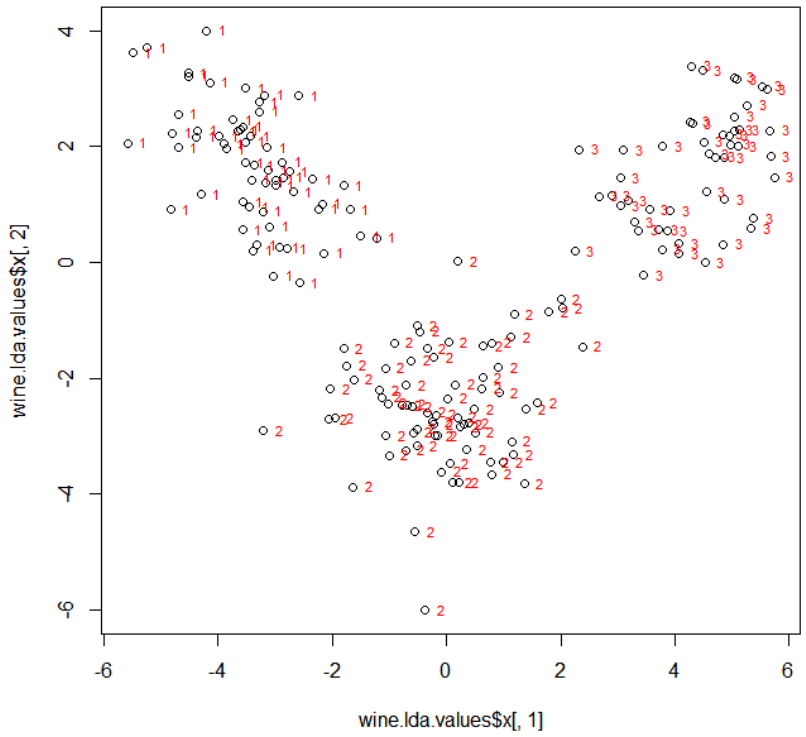
2.8.4 判別関数の散布図

品種のラベルをデータ点に付加した第 1，第 2 判別関数値の散布図を作成するには，次を入力する：

```
> plot(wine.lda.values$x[,1],wine.lda.values$x[,2]) # 散布図の作成
> text(wine.lda.values$x[,1],wine.lda.values$x[,2],wine$V1,cex=0.7,pos=4,col="red") # ラベ
```



ルの付加



最初の 2 つの判別関数の散布図から，3 つの品種からのワインが散布図でよく分離されていることがわかる．第 1 判別関数（x 軸）は品種 1 と 3 を非常によく分離しているが，品種 1 と 2，または品種 2 と 3 の分離は完全ではない．

第 2 判別関数（y 軸）は品種 1 と 2，品種 2 と 3 をかなりよく分離しているが，完全でない．3 つの品種をうまく分離するには，第 1 判別関数と第 2 判別関数の両者を利用するのが最良である．なぜなら，第 1 判別関数は品種 1 と 3 をうまく分離し，第 2 判別関数は品種 1 と 2，品種 2 と 3 をうまく分離するからである．

2.8.5 分類規則と誤分類率

“print-MeanAndSdByGroup()” 関数（上記参照）を用いて，品種で層別した判別関数の平均値を計算することができる：

```
> printMeanAndSdByGroup(wine.lda.values$x,wine[1])
[1] "Group 1 Means:"
LD1 LD2
-3.422489 1.691674
[1] "Group 2 Means:"
LD1 LD2
-0.07972623 -2.47265573
[1] "Group 3 Means:"
LD1 LD2
4.324737 1.578120
```

第 1 判別関数の平均値は，品種 1 では -3.422489，品種 2 では -0.07972623，品種 3 では 4.324737 である．品種 1 と 2 の平均値の中点は $(-3.422489 - 0.07972623)/2 = -1.751108$ であり，品種 2 と 3 のそれは $(-0.07972623 + 4.324737)/2 = 2.122505$ である．

よって，次の分類規則を用いることができる：

- 第 1 判別関数の値 ≤ -1.751108 のとき，品種 1 に分類する．
- 第 2 判別関数の値 > -1.751108 かつ ≤ 2.122505 のとき，品種 2 に分類する．
- 第 1 判別関数の値 > 2.122505 のとき，品種 3 に分類する．

次に示す “calcAllocationRuleAccuracy()” 関数を用いて，この分類規則の正確さを調べることができる：

```
> calcAllocationRuleAccuracy <- function(ldavalue, groupvariable, cutoffpoints)
  {
     # グループ変数が取る値の数を取得
     groupvariable2 <- as.factor(groupvariable[[1]])
     levels <- levels(groupvariable2)
     numlevels <- length(levels)
     # 各グループに対する真陽性（true positives）と偽陰性（false negatives）の計算
     numlevels <- length(levels)
     for (i in 1:numlevels)
     {
        leveli <- levels[i]
        levelidata <- ldavalue[groupvariable==leveli]
        # あるグループが各グループに分類されたサンプル数の計算
        for (j in 1:numlevels)
        {
           levelj <- levels[j]
           if (j == 1)
           {
              cutoff1 <- cutoffpoints[1]
              cutoff2 <- "NA"
              results <- summary(levelidata <= cutoff1)
```

```
        }
      else if (j == numlevels)
      {
         cutoff1 <- cutoffpoints[(numlevels-1)]
         cutoff2 <- "NA"
         results <- summary(levelidata > cutoff1)
      }
      else
      {
         cutoff1 <- cutoffpoints[(j-1)]
         cutoff2 <- cutoffpoints[(j)]
         results <- summary(levelidata > cutoff1 & levelidata <= cutoff2)
      }
      trues <- results["TRUE"]
      trues <- trues[[1]]
      print(paste("グループ",levelj,"に分類されたグループ",leveli,"のサンプル数 : ",
      trues,"(cutoffs:",cutoff1,",",cutoff2,")"))
    }
  }
}
```

例えば，ワイン・データに対する分類規則に基づく第 1 判別関数の正確さを求めるには次を入力する :

```
> calcAllocationRuleAccuracy(wine.lda.values$x[,1], wine[1], c(-1.751108, 2.122505))
[1] "グループ 1 に分類されたグループ 1 のサンプル数 :  56 (cutoffs: -1.751108 , NA )"
[1] "グループ 2 に分類されたグループ 1 のサンプル数 :  3 (cutoffs: -1.751108 , 2.122505 )"
[1] "グループ 3 に分類されたグループ 1 のサンプル数 :  NA (cutoffs: 2.122505 , NA )"
[1] "グループ 1 に分類されたグループ 2 のサンプル数 :  5 (cutoffs: -1.751108 , NA )"
[1] "グループ 2 に分類されたグループ 2 のサンプル数 :  65 (cutoffs: -1.751108 , 2.122505 )"
[1] "グループ 3 に分類されたグループ 2 のサンプル数 :  1 (cutoffs: 2.122505 , NA )"
[1] "グループ 1 に分類されたグループ 3 のサンプル数 :  NA (cutoffs: -1.751108 , NA )"
[1] "グループ 2 に分類されたグループ 3 のサンプル数 :  NA (cutoffs: -1.751108 , 2.122505 )"
[1] "グループ 3 に分類されたグループ 3 のサンプル数 :  48 (cutoffs: 2.122505 ,
```

この結果は次のように “誤判別表” の形にまとめることができる。

	品種 1 に分類	品種 2 に分類	品種 3 に分類
品種 1 を	56	3	0
品種 2 を	5	65	1
品種 3 を	0	0	48

全ワインサンプル 178 ($= 56 + 3 + 5 + 65 + 1 + 48$) のうち，誤分類されたのは 9 ($= 3 + 5 + 1$) 個である．品種 1 の 3 サンプルが品種 2，品種 2 の 5 サンプルが品種 1 に，品種 2 の 1 サンプルが品種 3 に誤分類されている．よって，誤分類率は 9/178，つまり 5.1% である．誤分類率は非常に低く，この分類規則はかなり正確であることが分かる．

しかし，分類規則自体がこのデータ（これが訓練用データ）に基づいて作成されているので，この誤分類率は過小評価となっている可能性がある．分類規則を作成したのとは別の検証用データを用いて誤分類率を計算すると，誤分類率のより正確な推定値を得ることができる．

2.9 リンクと参考文献

さらに学習するためのリンク・参考文献を紹介する．

R をより詳細に知るには，“Kickstarting R” ウェブサイト（cran.rproject.org/doc/contrib/Lemon-kickstart）に良いオンライン・チュートリアルを参照のこと．

別のチュートリアルとしてもう少し詳細なものが，“Introduction to R” ウェブサイト（cran.rproject.org/doc/manuals/R-intro.html）にある．

多変量解析を学ぶには，Open University 発行の本 “Multivariate analysis”（製品コード M249/03. Open University のショップより購入可）を推薦する．

多変量解析のための R の使用法に関しては，“Use R!” シリーズの 1 冊である Everitt and Hothorn 著の “An Introduction to Applied Multivariate Analysis with R” がある．

2.10 謝辞

本ブックレットの例の多くは，Open University のショップで入手できる優れた本，“Multivariate Analysis”（製品コード M249/03）より着想を得た．

また，本ブックレットの例で利用したデータセットを利用できるようにしてくれている UCI Machine Learning Repository（http://archive.ics.uci.edu/ml）に感謝する．

2.11 連絡

本書に関して間違いや提案があれば私（Avril Coghlan）にメール（アドレス a.coghlan@ucc.ie）を送付してもらえるとありがたい．

第 3 章

謝辞

本書を作成する際の Sphinx（http://sphinx.pocoo.org），異なるバージョンの管理のための github（https://github.com/），ビルドし，ディストリビュートするための readthedocs（http://readthedocs.org/）の使い方について支援を受けたことに対して，Noel O'Boyle に感謝する．

第 4 章

連絡

本書に関して間違いや提案があれば私（Avril Coghlan）にメール（a.coghlan@ucc.ie）を送付していただければ幸いである.

第 5 章

ライセンス