クレジット:
UTokyo Online Education 統計データ解析 Ⅱ 2018 小池祐太

東京大学 THE UNIVERSITY OF TOKYO

CC BY NC ND

統計データ解析 (II) 第 1 回

小池祐太

2018 年 4 月 5 日

- 講義題目: 統計データ解析 II
- 担当: 小池祐太
- 授業の目標, 概要 (詳しくはシラバス参照)
  - 統計ソフトウェア R の基本的な扱いを習得する
  - 大規模データから効果的に情報を抽出し, データの背後に潜む構造を分析するための統計解析手法である「多変量解析」の基本的な解析法に習熟する
    - ★ 解析法の数理科学的側面を理解する
    - ★ R を用いて実際のデータに適用できるようにする
- 月曜 5 限・水曜 5 限に開講される「統計データ解析 II」も同内容の講義

- 線形代数学および微分積分学の知識を講義では使用する (特に前者. 必要なものは講義中に説明する)
- 機材に限りがあるため, 受講希望者多数の場合は後日選抜を実施
  - 受講希望者数確認のため, 配布のアンケートを授業終了時に回収します
  - 2 年生の受講が優先
  - 月曜 5 限・水曜 5 限の方の講義が受講できない事情 (他の講義とかぶる等) があれば考慮する
- 成績評価方法
  - 出席およびレポート

講義計画 (予定)

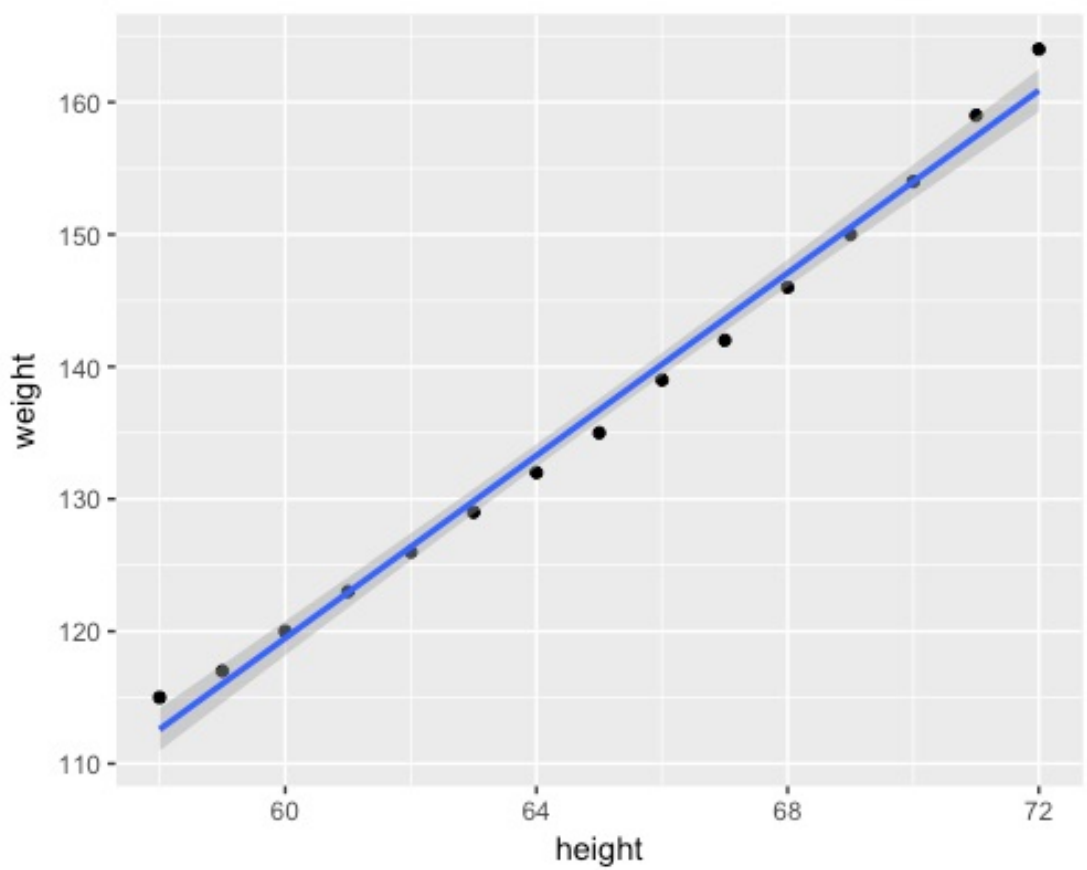
- 統計ソフトウェア R の基本的使用法
- 多変量分布のシミュレーション
- 重回帰分析
- 主成分分析
- 判別分析
- その他の多変量解析法
- 時系列解析入門
  - 講義の最初の数回は, R を使用したことがない人向けに R の使用方法を講義するため, A セメスター講義「統計データ解析 I」と内容が重なります

- 多変量解析
  - 「機械学習」で使われる分析手法と多くの関連があり, 原型・入門的方法とも考えられる
  - 訓練データ/学習データからモデルを構築し, データの説明や予測に用いる
- 問題の分類: 説明や予測対象の変数 (目的変数) が訓練データに含まれるか否かによって, 「教師あり」と「教師なし」問題に分類されることが多い
  - 教師あり問題: 目的変数の種類によって更に**回帰**と**判別**に分類される
    - ★ **回帰** 目的変数が量的データ (数値として自然に扱えるデータ. 多くの場合連続的). 例: 重回帰分析
    - ★ **判別** 目的変数が質的データ (区分だけ決められていて数値として自然に扱えないデータ. 通常離散的). 例: 判別分析
  - 教師なし問題:
    - ★ 目的変数が訓練データに含まれておらず, データに隠された構造を発見することが目的
    - ★ 例: 主成分分析

回帰分析

- ある 1 種類の変数/データを別の変数/データ (1 種類もしくは複数) によって説明もしくは予測するための関係式 (回帰方程式) を構成することを目的とする分析法
- 例
  - 身長から体重を予測する
    - ある身長の人がある体重だったときに, それが普通かどうか判定するのに利用できる
  - 築年数, 駅からの距離, 広さ, 間取りで家賃を説明する
    - 新規に家賃設定をする際に利用できる

図 1: 身長と体重のデータに対する回帰分析の例

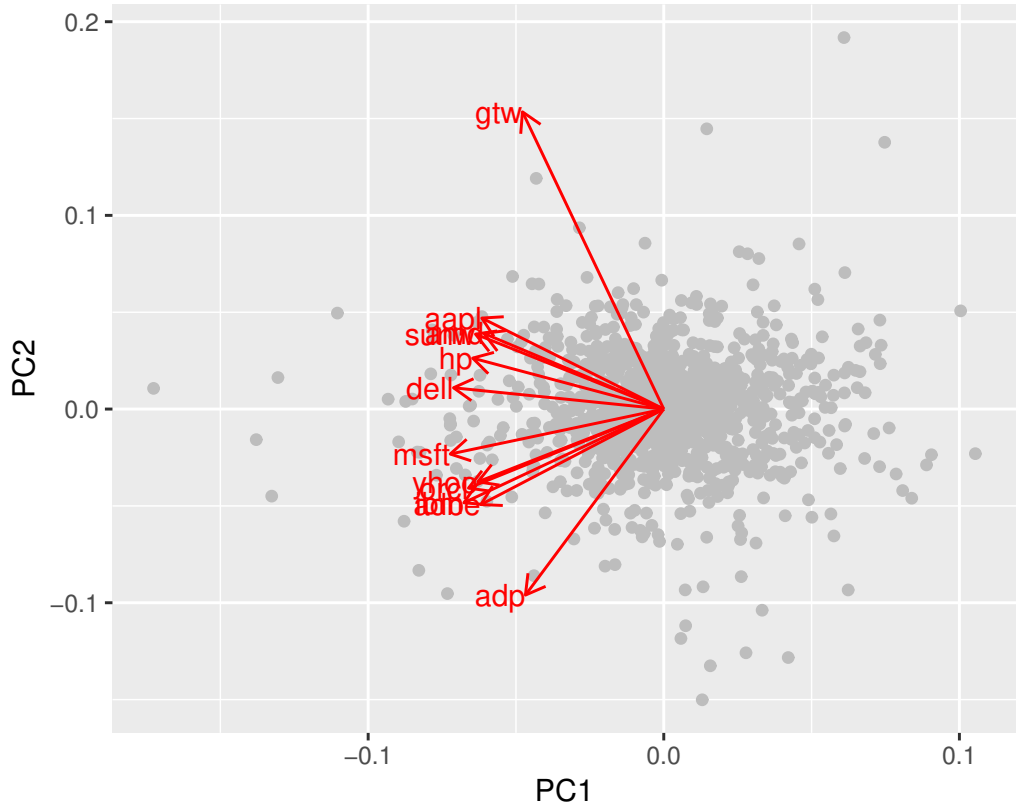


30 代のアメリカ人女性の身長と体重の関係に対する回帰分析．身長によって体重を説明するためのモデルを構築．身長と体重の間の比例関係がうまく説明できている．データは R の **datasets** パッケージ `women` を使用．

主成分分析

- 多数の変数/データが与えられた際に, 変数/データのもつ構造を効率的に記述できるような少数個の特徴量を構成することを目的とする分析法
- 例
  - 複数銘柄からなる株価の時系列データから, 市場全体の変動を記述する総合指標を作成する
  - 野球選手の打撃成績 (打率, 本塁打数, 打点, . . .) から, 打者としての特徴を記述する指標を作成する

図 2: 株価収益率データに対する主成分分析の例

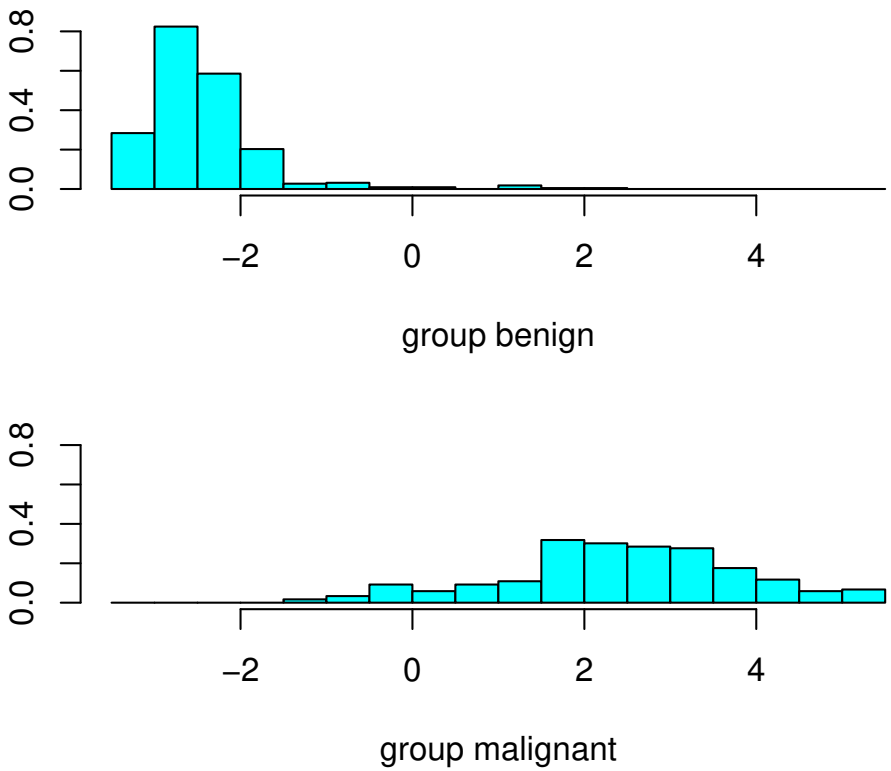


12 の IT 企業の株式リターンに対して主成分分析して得られた結果を図示．横軸は第一主成分，縦軸は第二主成分を表す．全企業の第一主成分が同符合 (負) であることから，第一主成分は市場全体を表す成分と考えられる．第二主成分は正の方にハードウェア系，負の方にソフトウェア/サービス系が含まれる傾向にある．データは `http://web.stanford.edu/~xing/statfinbook/data.html` の “12 stock returns” を使用

判別分析

- ある個体が複数のクラスのいずれかに属するとき, その個体の属性 (特徴量) からどのクラスに属するか予測するモデルを構築することを目的とする分析法
- 例
  - 食道がんを患っている人とそうでない人を, 年齢, 飲酒量, 喫煙度から判別する
  - 花の種類を, その花の花弁の幅・長さおよびがく片の幅・長さから判別する

図 3: 乳がんの生検データに対する判別分析の例



乳がんが良性か悪性かを, 9 種類の生検結果に基づいて判別するモデルを構築し, 判別式の値を良性の患者と悪性の患者のそれぞれについて図示した. 判別式の値の分布が上下で比較的きれいに分離していることから, ある程度よい判別結果が得られていることがわかる. データは R の **MASS** パッケージ `biospy` を使用.

1. R 言語の概要
2. 基本的な使い方
3. パッケージのインストール
4. データ構造

R 言語とは

- 統計計算のための言語と環境の総称
- オープンソース・フリーソフトウェア
- 「パッケージ」のインストールによって容易に機能拡張できる
  - パッケージの開発は日進月歩で進んでおり, 最新の技術や方法が簡単に導入できることも多い
- R の本体, およびパッケージは, 開発プロジェクトのサイト R Project

  `http://www.r-project.org/`

  のメニューにある CRAN の中にあるミラーサイトからダウンロードできる
- パッケージのインストールについては, R の機能を用いてインストールすることも可能 (後述)

RStudio

- RStudio
  - RStudio 社により開発・公開されている, R によるデータ解析や統計計算・パッケージ開発等を支援するための統合開発環境 (IDE)
- RStudio は独自の対話型操作環境を提供しており, OS に依存せずに同様に操作できるため, 本講義では RStudio を用いて説明を行う

起動と終了

- RStudio を起動すると, 標準では 4 つの枠 (ペインと呼ばれる) で区切られたウィンドウが立ち上がる
  - 左上: エディタ
  - 左下: コンソール
  - 右上: 作業環境にある変数・コマンドの履歴
  - 右下: パッケージ・グラフィックス・ヘルプの表示
- 詳細は以下順を追って説明

起動と終了

- コンソール (左下のペイン)
  - R においてプログラムを実行するためにコマンドを入力するための ウィンドウ
- 例えば, コンソール上で終了を指示するコマンド

  ```
  q()
  ```

  を入力すれば, R を終了させることができる
  - 終了できない場合は, OS の機能で強制終了する (Mac の場合, 画面左上の Apple マークに「強制終了」の項目がある)

起動と終了

- R 終了前に以下のメッセージが表示される場合がある:

```
Save workspace image?  [y/n/c]:
```

これは, これまでの作業によって生成された変数などをセーブするかどうか尋ねている (セーブした場合次回起動時に読み込まれる)

  - `y` を入力: セーブする (yes の略)
  - `n` を入力: セーブしない (no の略)
  - `c` を入力: R の終了をキャンセルする (cancel の略)

エディタ

- コンソール上に入力したコマンドは直ちに実行されてしまうため, 複雑なコマンドを書く場合や, 後からコマンドの一部を修正したい場合に不便
- RStudio では, 左上のペインが「エディタ」となっており, まとまったコマンドを記述してファイルとして保存・実行できるようになっている
- コマンドを記述したファイル (ソースファイルと呼ばれる) は, RStudio 上部の「File」から「Save」を選択して保存することができる
  - ファイル名を入力する必要がある
  - 拡張子は通常「.R」(「.r」も可) を利用

エディタ

- ファイルの保存はショートカットキー「Ctrl」＋「S」または「command」＋「S」でも可能
- 保存したソースファイルに記述したコマンドの実行は, RStudio のエディタ右上部にある「Source」をクリック
  - 「Ctrl」＋「Shift」＋「S」または「command」＋「Shift」＋「S」でも可
- 上述の方法で実行した場合, コードの実行のみが行われ, 実行されたコードは出力されない. 実行したコードの出力 (“エコーする” という) も行いたい場合は, RStudio のエディタ右上部にある「Source」の右側の ▼ をクリックして, 「Source with Echo」を選択
  - 「Ctrl」＋「Shift」＋「Enter」または「command」＋「Shift」＋「Enter」でも可

エディタ

- コンソールから実行するには, `source("ファイル名")` を入力
- 厳密には, `source("実行したいファイルのある場所 (ディレクトリ)")` を実行する必要がある
  - `source("ファイル名")` と入力した場合は, 現在 R が作業している場所 (作業ディレクトリ) にあるファイルだとみなされる
  - 作業ディレクトリは, コンソール上部に記載されている (もしくは `getwd()` を実行すると確認できる)
- エコーしたい場合は `source("ファイル名", echo = TRUE)` とする
- 新規ファイルの作成は, RStudio 上部の「File」から「New File」を選択し, 更に「R Script」を選択 (もしくは「Ctrl」+「Shift」+「N」)
- なお, #以降に入力された文字列は実行されないため (コメントアウトと呼ぶ), コマンドに対するコメントを残す際に有用

基本的な使い方: 式の入力

- 四則演算や一般的な関数はほぼ直感に沿った文法で計算を実行できる
- +: 加算, -: 減算, *: 乗算, /: 除算, ^あるいは**: ベキ乗
- 例 コンソール上で

  ```
  1*2+3^2
  ```

  を実行すると, $1 \cdot 2 + 3^2$ を計算して結果である 11 が出力される
- 実行例 `calc.r`

基本的な使い方: 関数の実行

- 一般に f という関数があったとすると, f は以下のような書式で実行できる:

```
f(arg1 = value1, arg2 = value2, ...)
```

ここで arg1, arg2, ... は引数の名前を表し, value1, value2, ... はそれぞれの引数に渡す値を表す

- 引数名は省略可能. つまり, 上のコマンドは以下と同値:

```
f(value1, value2, ...)
```

- 例 1 正弦関数の計算は関数 sin で実行できる. $\sin(\pi/2)$ を計算したい場合は, コンソール上で以下を実行:

```
sin(x = pi/2) または sin(pi/2)
```

基本的な使い方: 関数の実行

- 例 2 対数関数の計算は関数 `log` で実行できる. 底を b とする x の対数 $\log_b(x)$ の計算は以下の書式で実行:

  ```
  log(x, b)
  ```

  他にも, 引数名を省略せずに

  ```
  log(x = x, base = b)
  ```

  と書いたり, 引数 `base` と `x` を入れ替えて

  ```
  log(base = b, x = x)
  ```

  としてもよい (`log(b, x)` は `log(x = b, base = x)` の意味となるので不可)
- また, 引数 `base` の値を指定しなくても, `base` には既定値として `exp(1)` (自然対数の底) が指定されているため, `log(x)` は x の自然対数を計算する

基本的な使い方: 関数の実行

- 引数が存在しない, もしくはすべての引数に対し既定値が設定されている関数もある. 例えば f がそのような関数だった場合, コンソール上で

  ```
  f()
  ```

  を実行すると引数=既定値として関数の内容が実行される
- 例 R を終了する関数 q

ヘルプ機能

- 上述のように R には関数が数多く用意されており, 各関数の詳細 (機能, 引数名, 引数の既定値, 実行例など) を記述したヘルプファイルが用意されている
- 一般に, 関数 `f` のヘルプファイルにアクセスするには, コンソール上で以下のコマンドを実行:

  `help(f)`　　または　　`?f`

  - 右下のペインにヘルプファイルが表示される
- 例 関数 `log` のヘルプファイルを表示:

  `help(log)`　　または　　`?log`

ヘルプ機能

- ヘルプファイルには実行例もついているが, 実行例を実際に走らせるにはコンソール上で以下のコマンドを実行 (関数 f の場合):

```
example(f)
```

- 関数の正確な名前が分からない場合, キーワードを渡すことで関連するトピックをもつ関数を検索できる
- 例えば, xxx というキーワードに関連するトピックを検索したい場合, コンソール上で以下のコマンドを実行:

```
help.search(xxx)      または      ??xxx
```

ヘルプ機能

- 例 定積分を数値的に計算する関数があるかどうか調べたいとする. 積分は英語で integration であることから,

  ```
  help.search(integration)
  ```

  または

  ```
  ??integration
  ```

  を実行してみればよい
- なお, RStudio では, 右下のペインの “Help” タブの右上にある検索バーに関数名を入力することでヘルプファイルにアクセスすることが可能

基本的な使い方: 数の扱い

- R では実数及び複素数を取り扱うことができ, 指数表記にも対応している
- また, 無限大や不定な数など特殊なものを扱うこともできる
- 実行例 `numbers.r`

基本的な使い方: 変数への代入

- 文字列を変数名として, 数値等を保持することができる. また, 変数をそのまま計算に用いることもできる
- 文字列 `xxx` を変数名とする変数に `a` という値を代入するには, コンソール上で以下のコマンドを実行:

```
xxx <- a      または      xxx = a
```

- なお, R では, 変数や関数, および関数の実行結果等を総称して「オブジェクト」と呼ぶ
- 実行例 `variables.r`

基本的な使い方: 変数への代入

- 変数名は自由に決めて用いることが出来る (例: `x`, `y`, `abc` など)
- しかし, `sin`, `log`, `pi` など R の仕様として使われているものは, 用いることができないわけではないが混乱を招く元なので使わないほうがよい
- 参考 以下の文字は R 起動時からすでに特定の機能を与えられているので, 値を代入する際は注意が必要

  `c q t C D F I T`

  - それぞれの機能はヘルプを参照

基本的な使い方: 変数への代入

- これまでの作業で生成した変数に関する情報は, 右上のペインの “Environment” タブで確認できる
- また, これまでの作業でコンソール上で打ち込んだコマンドは, 右上のペインの “History” タブで確認できる
- 更に, コンソール上で上下キーを打つことで, 以前に実行したコマンドを再表示できる

パッケージのインストール

- R では, その機能を拡張するために多数のパッケージが用意されている
- 従って, 初期設定で関数が実装されていなくても, その関数を実装しているようなパッケージがすでに開発されていることが多い
- パッケージのインストールには以下の 2 通りの方法がある:
  - RStudio の機能を利用する方法
  - コンソールから行う方法

パッケージのインストール

- パッケージ **tseries** のインストール手順 (RStudio の機能を利用)
  1. 右下ペインの “Package” タブをクリック
  2. 左上あたりに “Install” という表示があるのでそれをクリック
  3. 新規ウィンドウが現れるので, “Package” のフォームにインストールしたいパッケージ名 (ここでは tseries) を入力し, 下部の “Install” をクリック
  4. パッケージがインストールされる

パッケージのインストール

- インストール済みのパッケージは右下ペインの “Package” タブに表示される
  - パッケージ名の左側のボックスをチェックすると, そのパッケージがロードされて, パッケージに含まれる関数などが利用可能になる
- 左上部の “Update” をクリックすると, インストール済みパッケージを最新版に更新できる

パッケージのインストール

- パッケージ **tseries** のインストール手順 (コンソールから)
  1. R のコンソール上で `install.packages("tseries")` を実行
  2. パッケージをダウンロードするためのサイト (CRAN のミラーサイト) を選ぶことを要求された場合は, 適当なものを選ぶ (Japan のミラーサイトがよい)
  3. パッケージがインストールされる
- インストールしたパッケージはコンソール上で `library(パッケージ名)` か `require(パッケージ名)` を実行することでロードされる (今の場合 `library(tseries)` または `require(tseries)` を実行)

データ構造

- R には, 以下のようなデータ構造が用意されている (代表的なもの):
  - ベクトル (vector)
  - 行列 (matrix)
  - リスト (list)
  - データフレーム (data frame)

データ構造: ベクトル

- R におけるベクトルは, 「スカラー値の集合」と解釈される (多くの場合順序が意味を持つが, 順序が無関係な場合も稀にある)
- R オブジェクトは基本的にはベクトルとして扱われる (スカラー値は長さ 1 のベクトルとして扱われる)
- スカラー値として扱われるものには, 実数や複素数以外に, 文字列 (“” で囲まれた文字. “x” など) や論理値 (TRUE, FALSE) などが含まれる
- 実行例 `scalar.r`

データ構造: ベクトル

- 要素 a, b, ... からなるベクトルは以下で生成できる:

```
c(a,b,...)
```

  - a, b, ... は数値や文字列が混同していてもよい
- ベクトル x の i 番目の要素は x[i] で取り出せる (注: ベクトルの添え字は 1 から始まる)
- x[c(1,3,4)] のように添え字もベクトルで指定すると, 指定された添え字に対応する要素からなるベクトルが取り出せる (この場合ベクトル c(x[1],x[3],x[4)])
- ベクトル x の長さは length(x) で取り出せる

データ構造: ベクトル

- 実数 x から y (x<y) まで 1 ずつ増加していく要素を持つベクトルは x:y で生成できる
  - x>y の場合は x から y まで 1 ずつ減少していく要素を持つベクトルとなる
- より一般に, 実数 x から y (x<y) まで a ずつ増加していく要素を持つベクトルは seq(x,y,by=a) で生成できる
- ベクトル x を n 回繰り返した要素をもつベクトルは rep(x,n) で生成できる
  - 従って, rep(x,n) の長さは length(x) × n となる
- ベクトル x の末尾にベクトル y をつなげたベクトルは c(x,y) で, x の要素を反転したベクトルは rev(x) で生成できる
- 実行例 vector.r

データ構造: 行列

- 行列 スカラー値を長方形状に並べたもの
- すべての要素が a である $m \times n$ 行列は

  `matrix(a,m,n)`

  で生成できる
- より一般に, 長さ mn のベクトル
  $x = (x_{11}, \ldots, x_{m1}, x_{21}, \ldots, x_{2n}, \ldots, x_{m1}, \ldots, x_{mn})$ に対して,

  `matrix(x,m,n)`

  で $m \times n$ 行列 $(x_{ij})_{1 \leq i \leq m, 1 \leq n}$ が生成できる
  - `as.vector` を適用することでベクトル化できる:

    ```
    x = as.vector(matrix(x,m,n))
    ```

データ構造: 行列

- 行列 `X` のサイズは `dim(X)` で取り出せる (長さ 2 のベクトル)
- 逆に, ベクトルに dim 属性を指定してやることで行列に変換できる
  - 例えば, 前頁のベクトル `x` に対して

    ```
    dim(x) <- c(m,n)
    ```

    を実行すれば, `x` は行列 `matrix(x,m,n)` に変換される
- 行列 `X` の行数は `nrow(X)` で, 列数は `ncol(X)` で取り出せる
- 行列 `X` の (i, j) 成分は `X[i,j]` で取り出せる
  - ベクトルの場合と同様に添え字をベクトルで指定することで, 部分行列の取り出しも可能
  - `X[i,]`, `X[,j]` でそれぞれ `X` の第 `i` 行, 第 `j` 列が取り出せる

データ構造: 行列

- 長さが等しい複数のベクトル `x`, `y`, ... を列もしくは行ベクトルとする行列を作成することもできる
  - 列ベクトルの場合 ··· `cbind(x, y, ...)`
  - 行ベクトルの場合 ··· `rbind(x, y, ...)`
- `cbind`/`rbind` は行数/列数が等しい複数の行列を横/縦に結合するのにも使える
- 補足 行列の高次元版のデータ構造として, 配列 (array) が用意されている
- 実行例 `matrix.r`

データ構造: リスト

- リスト
  - 異なる構造のデータをまとめて 1 つのオブジェクトとして扱えるようにしたもの
  - リストの各要素はデータ型・クラスともにバラバラであってもよい
- R のオブジェクト x, y, ... を要素とするリストは `list(x,y,...)` で生成される. リスト L の i 番目のオブジェクトには `L[[i]]` でアクセスできる.

データ構造: リスト

- リストは各成分に名前をつけることができる
  - 方法 1: 作成時に `list(name1 = x, name2 = y, ...)` のように書くと, 各成分に `name.1`, `name.2`, ... といった名前がつく
  - 方法 2: すでに作成してある長さ `n` のリスト `L` の各成分に名前をつける (もしくは名前を変更する) には,

    ```
    names(L) <- c(name.1, name.2, ..., name.n)
    ```

    のようにする
- リスト `L` の名前 `name` の成分は, `L$name` もしくは `L[[name]]` で取り出せる
- 実行例 `list.r`

データ構造: データフレーム

- データフレーム
  - 行列風リスト
  - リストにおいて, 各成分が長さが等しいベクトルであるようなもの
- イメージ
  - 複数の個体について, いくつかの属性を集計したデータ (例えばある小学校の 1 年生の身長, 体重, 性別, 血液型, . . . を集計したデータ)
  - リストの各成分はある属性に関する観測データに対応
  - 個体数は集計項目に関わらず変化しないが, 集計項目によっては定量的データ・定性的データの違いが出てくるので, データ型は変わりうる

データ構造: データフレーム

- 長さが等しいベクトル `x`, `y`, ... を変数とするデータフレームは, `data.frame(x,y,...)` で生成される
  - データフレームはリストなのでリストと同様にして各変数にアクセスできる
  - 他方, データフレームは行数がベクトルの長さ (個体数), 列数が変数の個数 (観測項目の数) の行列と同様にアクセスできる
- 実データは上のような表形式のデータであることが多いので, 実データを R に読み込む際に役に立つデータ構造
- 実行例 `data.frame.r`