R 言語速習

データの取り扱いと可視化

村田 昇

講義の内容

- データの取り扱い
- 描画の基礎

データフレーム

データ構造

- R に用意されている基本的なデータ構造
  - ベクトル (vector) : 1 次元配列
  - 行列 (matrix) : 2 次元配列
  - 配列 (array) : 多次元配列
  - **データフレーム** (data frame) : 表 (2 次元配列)
- 特殊なもの
  - リスト (list) : オブジェクトの集合

データフレーム

- 複数の個体について，いくつかの属性を集計した表
  - 長さの等しい列ベクトルをまとめたもの
  - 各列のデータ型はバラバラでも良い
- 実データは表形式であることが多いため最も一般的な形式
- 例 : ある小学校の 1 年生の身長・体重・性別・血液型のデータ

データフレームの作成

- 方法はいくつか用意されている
  - 同じ長さのベクトル (同じ型の配列) を並べる
  - データフレームを結合する
  - マトリクスを変換する (全て同じ型の場合)

```
## 同じ長さのベクトルを並べる
## (... <- ...) は代入した結果を表示
(foo <- data.frame(one=c(1,2,3),two=c("AB","CD","EF")))
(bar <- data.frame(three=c("x","y","z"),four=c(0.9,0.5,-0.3)))
## データフレームを結合する
(baz <- cbind(foo,bar)) # column bind
```

練習問題

- 次の表に対応するデータフレームを作成しなさい

	math	phys	chem	bio
A	90	25	65	70
B	80	50	100	50
C	70	75	70	30
D	60	100	40	80
E	50	80	75	100

ファイルの操作

ファイルを用いたデータの読み書き

- 解析においてはデータファイルの操作が必要
  - 整理したデータを保存する
  - 収集されたデータを読み込む
- R で利用可能なデータファイル
  - csv 形式 (comma separated values) : テキストファイル
  - RData 形式 : R の内部表現を用いたバイナリーファイル
  - Excel 形式 : Microsoft Excel ファイル (Files タブから読み込み)

作業ディレクトリの確認と変更

- 作業ディレクトリとファイルに関する注意
  - R の処理は特定のフォルダ (**作業ディレクトリ**) 内で実行される
  - ファイルは作業ディレクトリにあるものとして扱われる
  - 作業ディレクトリ以外のファイルを扱う場合はパスを含めて指定する必要がある
- 作業ディレクトリに関する操作
  - 確認の仕方 : コンソールの上部の表示
  - 変更の仕方 : 主に **Session** メニューを利用

csv 形式の操作 (テキスト)

- 関数 `write.csv()` : csv ファイルの書き出し

```
write.csv(x, file="ファイル名")
## x: 書き出すデータフレーム
## file: 書き出すファイルの名前（作業ディレクトリ下，またはパスを指定）
```

- 関数 `read.csv()` : csv ファイルの読み込み

```
y <- read.csv(file="ファイル名", header=TRUE, row.names=1)
## y: 読み込む変数
## file: 書き出すファイルの名前（作業ディレクトリ下，またはパスを指定）
## header: 1行目を列名として使うか否か
## row.names: 行名の指定（行名を含む列番号/列名または行名のベクトル）
```

- 他の細かいオプションはヘルプを参照

csv 形式の操作 (`package::readr`)

- 日本語などの扱いに問題がある場合に推奨

```
install.packages("readr") # package タブを使ってもよい
```

- 関数 write_csv() : csv ファイルの書き出し

```
write_csv(x, file="ファイル名") # 行名は書き出されない
```

- 関数 read_csv() : csv ファイルの読み込み

```
y <- read_csv(file="ファイル名") # 行名を列から付けるオプションはない
```

- 行名の扱いに違いがあるので注意

RData 形式の操作 (バイナリ)

- 関数 save() : RData ファイルの書き出し

```
save(..., file="ファイル名")
## ...: 保存するオブジェクト名 (複数指定可, データフレーム以外も可)
## file: 書き出すファイルの名前 (作業ディレクトリ下, またはパスを指定)
```

- 関数 load() : RData ファイルの読み込む

```
load(file="ファイル名")
## file: 読み込むファイルの名前 (作業ディレクトリ下, またはパスを指定)
```

- 複数のデータフレームを同時に扱うことができる

練習問題

- 前の演習で作成したデータフレームを適当なファイルに書き出しなさい
- 書き出したファイルから別の変数に読み込みなさい
- 厚労省からダウンロードしたファイル (pcr_case_daily.csv) を変数 pcr_data に読み込みなさい

データフレームの操作

要素の選択

- 添字の番号を指定する (マイナスは除外)
- 論理値 (TRUE/FALSE) で指定する
- 要素の名前で指定する

```
z <- data.frame(one=c(1,2,3), two=c("AB","CD","EF"), three=6:8)
z[1,2] # 1 行 2 列の要素を選択
z[-c(1,3),] # 1,3 行を除外
z[c(TRUE,FALSE,TRUE),] # 1,3 行を選択
z[,"two"]   # 列名 "two" を選択 (ベクトルになる)
z[["two"]] # 上記と同様の結果となる (1 列の場合しか使えない)
z$two       # 上記と同様の結果となる (1 列の場合しか使えない)
z["two"]    # 列名 "two" を選択 (1 列のデータフレームになる)
z[,c("one","three")] # 列名 "one" と "three" を選択 (データフレームになる)
z[c("one","three")]    # 上記と同様の結果となる
```

部分集合の取得

- 関数 subset() : 条件を指定して行と列を選択

```
subset(x, subset, select, drop=FALSE)
## x: データフレーム
## subset: 行に関する条件
## select: 列に関する条件 (未指定の場合は全ての列)
## drop: 結果が 1 行または 1 列となる場合にベクトルとして返すか否か
```

- オプション subset の条件指定には以下を用いることができる
  - 等号 : == (否定は !=)
  - 不等号 : <,>,<=,>=
  - 論理式 : & (かつ), | (または)

練習問題

- pcr_case_daily.csv から以下の条件を満たすデータを取り出しなさい
  - 関数 subset() の使用例

```
## 国立感染症研究所 (a) の検査件数が 0 でないデータ
subset(pcr_data, subset= a!=0) # subset オプションに条件を指定する

## 検疫所 (b) と民間検査会社 (d) の検査件数データ
subset(pcr_data, select= c(b,d)) # select オプションに列名を指定する
```

  - 医療機関 (f) での検査件数が 2000 を越えたときの国立感染症研究所 (a) と医療機関 (f) のデータ
  - 大学等 (e) と医療機関 (f) でともに検査件数が 2000 を越えたデータ

データフレームの集約

統計量の計算

- データを集約した値 = 統計量
  - 関数 sum() : 総和を計算する
  - 関数 mean() : 平均
  - 関数 max() : 最大値
  - 関数 min() : 最小値
  - 関数 median() : 中央値
  - 関数 quantile() : 分位点
- これ以外にも沢山あるので調べてみよう

行・列ごとの操作

- 関数 apply() : 列または行ごとに計算を行う

```
apply(X, MARGIN, FUN) # 変数名が大文字で定義されている
## X: データフレーム
## MARGIN: 行 (1) か列 (2) かを指定
## FUN: 求めたい統計量を計算するための関数
```

- 例 : 学生の成績表 grade の各教科の平均を計算する

```
apply(X=grade,  # データフレーム
      MARGIN=2, # 列ごとの処理
      FUN=mean) # 処理内容の指定 (関数)
apply(grade, 2, mean) # 上記と同じ (変数名は省略可能)
```

グループごとの操作

- 関数 aggregate() : 各行をいくつかのグループにまとめて計算を行う

```
aggregate(x, data, FUN, ...,
          subset, na.action = na.omit)
## x: 条件式
## data: データフレーム
## FUN: 求めたい統計量を計算するための関数
## subset: データフレームの行に関する条件
## na.action: 欠損値の扱い
```

- 例 : 医療機関 (f) の PCR 件数を各月で集計する

```
aggregate(f ~ month, # 式による集計の指定 x = f ~ month と書いても良い
          data=transform(pcr_data, # データフレームの書き換え
                         month=month(date)),
          FUN=sum)                  # 集計内容の指定 (関数)
```

練習問題

- pcr_case_daily.csv を以下の条件で整理しなさい
  - 各機関での PCR 検査件数の最大値
  - 2020 年の月ごとの各機関での PCR 検査件数の最大値
- datasets::mtcars のデータを以下の条件で整理しなさい
  - 気筒数 (cyl) ごとに排気量 (disp) の最大値，最小値
  - 気筒数 (cyl) とギア数 (gear) ごとの燃費 (mpg) の平均値

補足

- より詳細なデータの取り扱いについては以下を参照して下さい
  - 講義ノート
    - R 言語の基礎 第 3 章 (pp29-52)
  - 統計データ解析 I スライド
    - 第 1 講 R の基本的な操作
    - 第 3 講 データの整理と集計

描画の基礎

データの可視化

- データ全体の特徴や傾向を把握するための直感的で効果的な方法
- R 言語には極めて多彩な作図機能が用意されている
- 基本となるのは関数 plot()
- 描画関連の関数は色，線種や線の太さ，図中の文字の大きさなどを指定することができる

基本的な描画 (ベクトル)

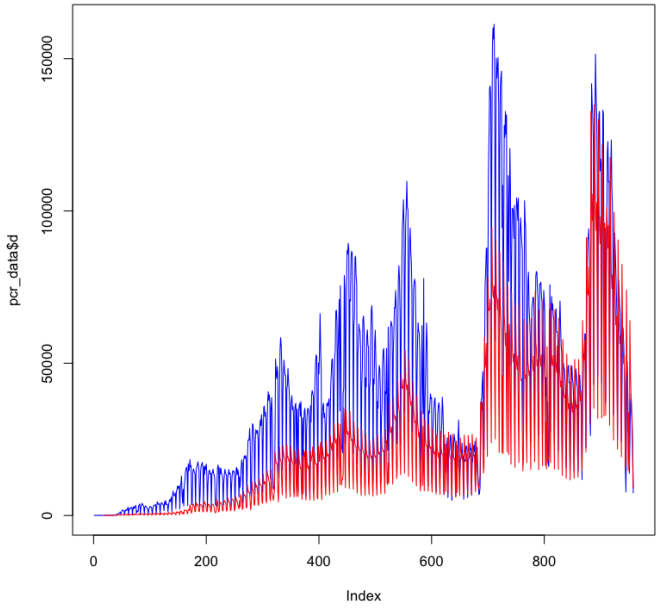
- ベクトルデータを左から等間隔で描画

```
plot(x, type="p", xlim=NULL, ylim=NULL,
     main=NULL, xlab=NULL, ylab=NULL, ...)
## x: ベクトル
## type: 描画タイプ．既定値は "p"(点プロット)．"l"(折れ線) など指定可
```

```
## xlim: x軸の範囲．既定値は自動的に決定
## ylim: y軸の範囲．既定値は自動的に決定
## main: 図のタイトル．既定値はタイトルなし
## xlab: x軸のラベル名．既定値は"Index"
## ylab: y軸のラベル名．既定値はxのオブジェクト名
## ...: 他のオプション．以下に例示．詳細は help(par) を参照
## col: 色の指定．"red"や"blue"など．指定可能な色は colors() を参照
## pch: 点の形．詳細は help(points) を参照
## cex: 文字の大きさ．既定値の何倍にするかを指定
## lty: 線のタイプ．実線・破線などを記号・数字で指定．詳細は help(par) を参照
## lwd: 線の太さ．数字で指定
```

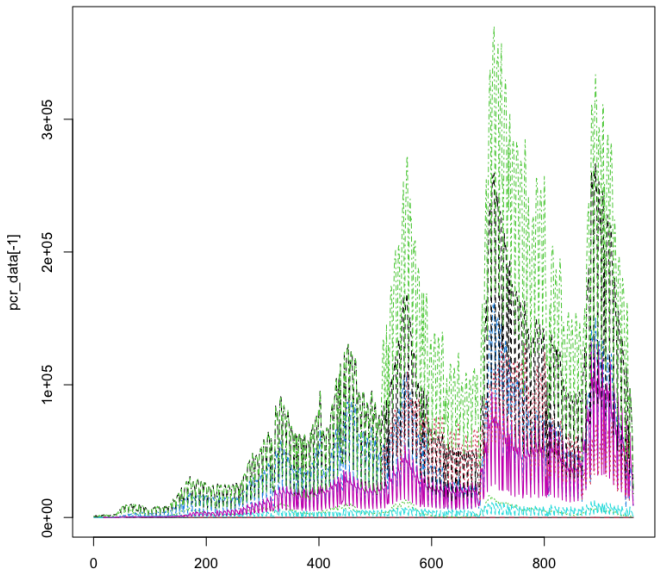
- 例：民間検査会社 (d) と医療機関 (f) の検査件数の推移を視覚化する

```
plot(pcr_data$d, # データフレームからD列のベクトルを抽出
     type="l", col="blue") # 線での描画と色を指定
lines(pcr_data$f, col="red") # 線を重ね描き
```



- 例：検査件数の推移を視覚化する

```
## 複数のデータを同時に描画する方法も用意されている
matplot(pcr_data[-1], # データフレームから1列目を取り除いたデータフレームを作成
        type="l") # 線での描画を指定，色も個別に指定できる
```



基本的な描画 (散布図)

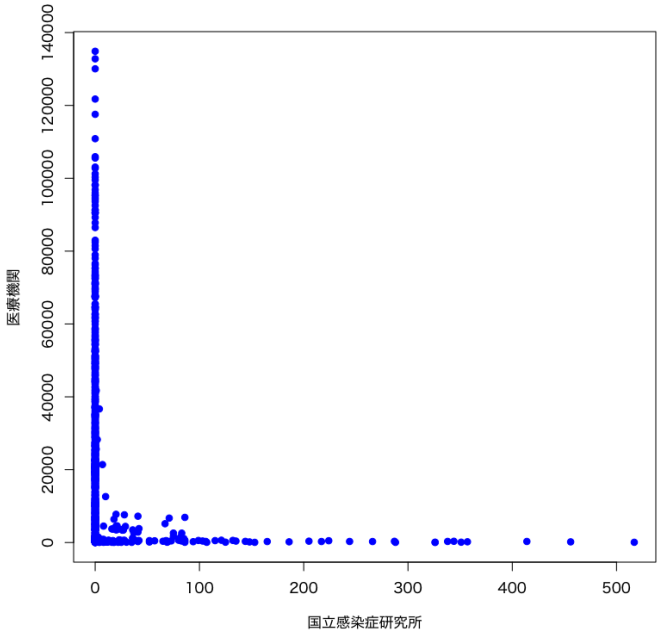
- 点 $(x_1, y_1), \ldots, (x_N, y_N)$ を平面上に描画
  - 2 つの同じ長さのベクトル $x_1, \ldots, x_N$ と $y_1, \ldots, y_N$ を与える
  - $x_1, \ldots, x_N$ と $y_1, \ldots, y_N$ を持つデータフレームを与える

```
plot(x, y, ...)
## x: 1種類目のデータ x_1,...,x_N
## y: 2種類目のデータ y_1,...,y_N
## ...: "ベクトルの描画"と同じオプションが利用可能

## データフレーム x の変数 A, B の散布図を作成する場合 (こちらの書き方を推奨)
plot(B ~ A, data=x, ...)
```

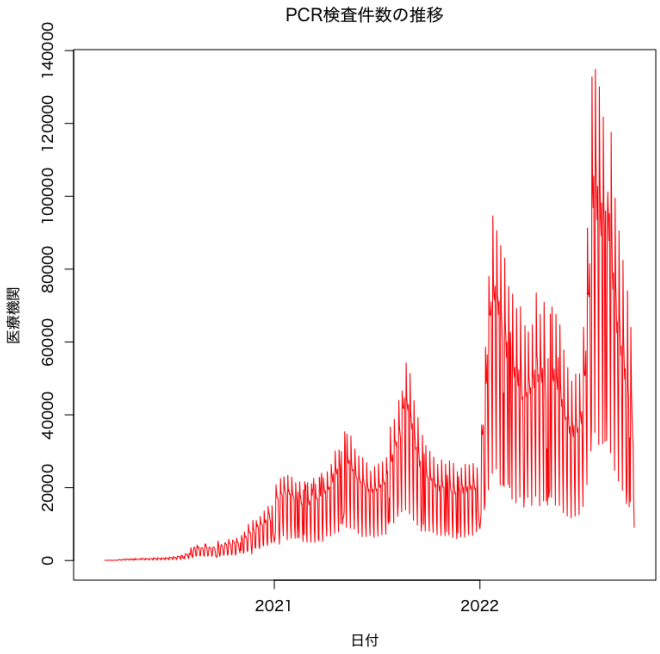
- 例 : 国立感染症研究所 (a) と医療機関 (f) の検査件数の関係を視覚化する

```
## MacOSかどうか調べて日本語フォントを指定する
if(Sys.info()["sysname"]=="Darwin"){par(family="HiraginoSans-W4")}
plot(f ~ a, data=pcr_data, # y軸=f, x軸=a で散布図を作成
     col="blue", pch=19, # 色と形を指定
     xlab=pcr_colname[2], ylab=pcr_colname[7]) # 軸の名前を指定
```



- 例 : 医療機関 (f) の検査件数の推移を視覚化する

```
if(Sys.info()["sysname"]=="Darwin"){par(family="HiraginoSans-W4")}
## x 軸を日付とすることで日付と検査数の関係を表すことも可能
plot(f ~ as.Date(date), data=pcr_data, # 線で描画する
     type="l", col="red", # 色と形を指定
     xlab=pcr_colname[1], ylab=pcr_colname[7], # 軸の名前を指定
     main="PCR 検査件数の推移")
```



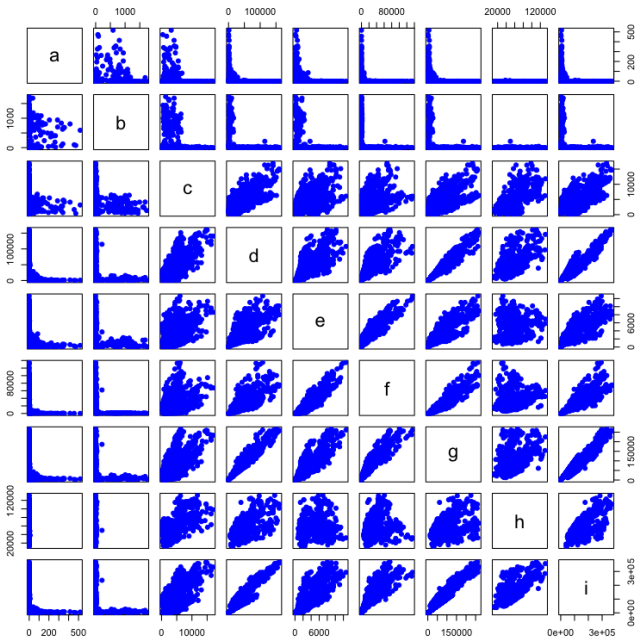
基本的な描画 (散布図行列)

- データフレームの 2 つの列同士の散布図を描画

```
plot(x, ...)
## x: データフレーム (数値データのみにすること)
## ...: "ベクトルの描画"と同じオプションが利用可能
```

- 例：各検査機関での件数の関係を視覚化する

```
plot(pcr_data[-1], col="blue", pch=19) # データフレームから 1 列目を除いて描画
```



図の保存

- RStudio の機能を使う (少数の場合はこちらが簡便)
  - 右下ペイン **Plots** タブから **Export** をクリック
  - 形式やサイズを指定する
  - クリップボードにコピーもできる
- コマンドで実行する (多数の場合はこちらで処理)
  - 関数 `pdf()`
  - 関数 `png()`
  - 関数 `dev.copy()`

  などを参照

練習問題

- `pcr_case_daily.csv` を用いて以下の描画を行いなさい
  - 検疫所 (`b`)，地方衛生研究所. 保健所 (`c`)，民間検査会社 (`d`) における検査件数の推移
  - 民間検査会社 (`d`)，大学等 (`e`)，医療機関 (`f`) での検査件数の関係 (散布図)

さまざまなグラフ

ヒストグラム

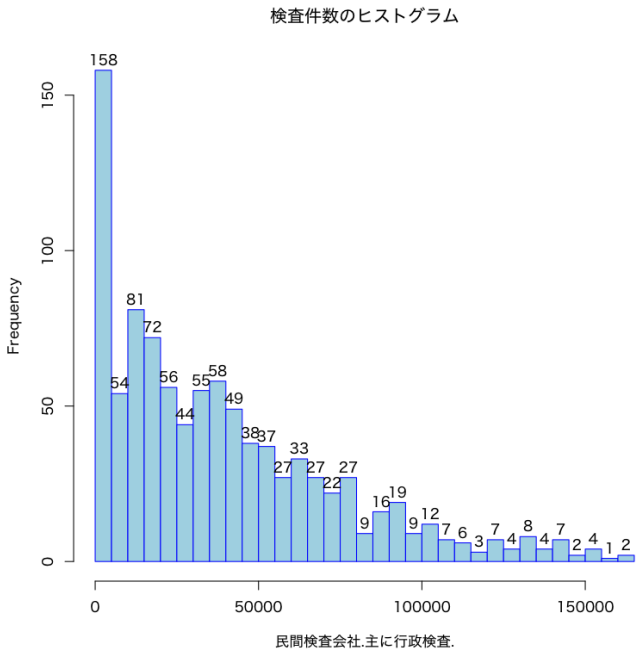
- データの値の範囲をいくつかの区間に分割し，各区間に含まれるデータの個数を棒グラフにした図

- 棒グラフの幅が区間, 面積が区間に含まれるデータの個数に比例するようにグラフを作成
- データの分布を可視化するのに有効 (値の集中とばらつきを調べる)

```
hist(x, breaks="Sturges", freq=NULL)
## x: ベクトル
## breaks: 区間の分割の仕方を指定. 数字を指定するとそれに近い個数に等分割
## freq: TRUE を指定すると縦軸はデータ数,
##       FALSE を指定すると縦軸はデータ数/全データ数. 既定値は TRUE
## ...: plot で指定できるオプションが利用可能
```

- 例 : 民間検査会社 (d) での検査件数の分布を視覚化する

```
if(Sys.info()["sysname"]=="Darwin"){par(family="HiraginoSans-W4")}
hist(pcr_data$d, breaks=25, labels=TRUE, # ビンの数と度数表示を指定
     col="lightblue", border="blue", # 中と境界の色を指定
     main="検査件数のヒストグラム", xlab=pcr_colname[5]) # 軸の名前を指定
```



箱ひげ図

- データ散らばり具合を考察するための図
  - 太線で表示された中央値 (第 2 四分位点)
  - 第 1 四分位点を下端・第 3 四分位点を上端とする長方形 (箱)
  - 中央値から第 1 四分位点・第 3 四分位点までの 1.5 倍以内にあるデータの最小の値・最大の値を下端・上端とする線 (ひげ)
  - ひげの外側のデータは点で表示
- 複数のデータの分布の比較の際に有効

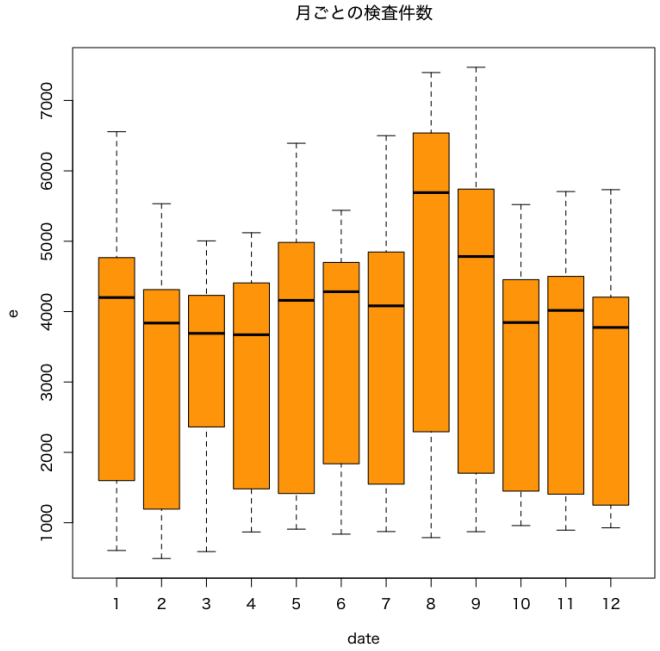
```
boxplot(x, ...)
## x: ベクトルまたはデータフレーム
##       ベクトルに対しては単一の箱ひげ図
##       データフレーム対しては列ごとの箱ひげ図
## ...: plot と同様のオプションを指定可能

## x の変数 B を変数 A(質的変数; 性別, 植物の種類など) で分類する場合
```

```
boxplot(B ~ A, data=x, ...)
```

- 例：月ごとの大学等 (e) での検査件数の分布 (分位点) を視覚化する

```
if(Sys.info()["sysname"]=="Darwin"){par(family="HiraginoSans-W4")}
boxplot(e ~ date,
        data=transform(subset(pcr_data, year(date)==2021),
                       date=month(date)),
        col="orange", main="月ごとの検査件数")
```



棒グラフ

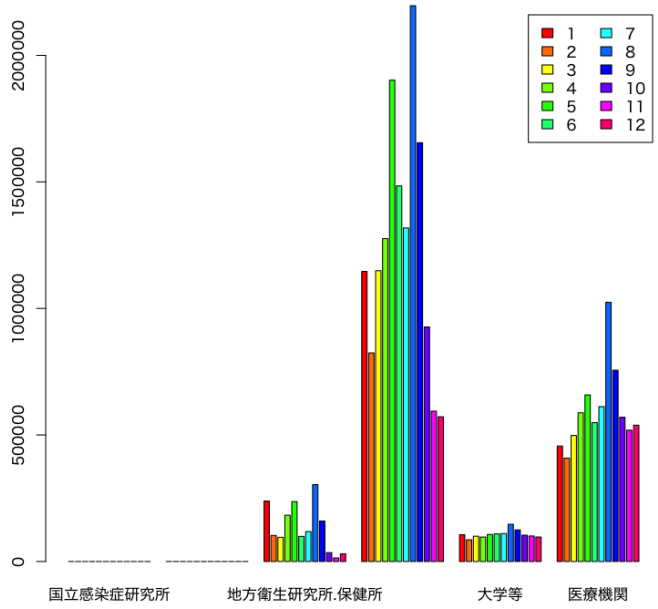
- 項目ごとの量を並べて表示した図
- 縦にも横にも並べられる

```
barplot(x, width=1, space=NULL, beside=FALSE,
        legend.text=NULL, args.legend=NULL, ...)
## x: ベクトルまたは行列 (データフレームは不可)
## width: 棒の幅
## space: 棒グラフ間・変数間のスペース
## legend.text: 凡例
## beside: 複数の変数を縦に並べるか・横に並べるか
## args.legend: legend に渡す引数
## ...: plot で指定できるオプションが利用可能
```

- 例：機関 (医療機関まで) ごとの月の検査件数の推移を視覚化する

```
if(Sys.info()["sysname"]=="Darwin"){par(family="HiraginoSans-W4")}
foo <- aggregate(. ~ date, # 集計したデータを保存
                 transform(subset(pcr_data,
                                  subset = year(date)==2021,
                                  select = 1:7),
                           date=month(date)),
                 sum, na.action=na.pass)
barplot(as.matrix(foo[-1]), col=rainbow(12), # 作成した月の色を利用
        names.arg=pcr_colname[2:7], # 変数名を日本語で表示
        beside=TRUE, space=c(.3,3), # 横並びの指定とスペースの設定
```

```
        legend.text=foo[,1], args.legend=list(ncol=2)) # 凡例の指定
```



グラフィクス環境の設定

- グラフィクス関数には様々なオプションがある
- 共通の環境設定のためには関数 `par()` を用いる
  - 複数の図の配置: `mrow`, `mcol`
  - 余白の設定: `margin`
  - 日本語フォントの設定: `family`
  - 他多数 (`?par` を参照)
  - より進んだグラフィクスの使い方の例は `demo("graphics")`, `example(関数名)` を参照

練習問題

- 適当なデータに対してグラフの作成を行ってみよう
  - PCR 検査件数データ (`pcr_case_daily.csv`)
  - 東京都の気候データ (`tokyo_weather.csv`)
  - R 言語に用意されているデータ (関数 `data()` で一覧表示)

補足

- より詳細なグラフの描画については以下を参照して下さい
  - 講義ノート
    - R 言語の基礎 第 4 章 (pp53-70)
  - 統計データ解析 I スライド
    - 第 4 講 データの可視化

疑似乱数

疑似乱数とは

- コンピューターで生成された数列のこと
- 完全にランダムに数字を発生されることは不可能
- R の既定値は “Mersenne-Twister 法” (`?Random` 参照)
- 数値シミュレーションにおいて再現性が要請される場合には，乱数の “シード値” を指定して再現性を担保 (関数 `set.seed()`)

基本的な乱数

- **ランダムサンプリング** : 与えられた集合の要素を無作為抽出することで発生する乱数
- **二項乱数** :「確率 p で表がでるコインを n 回投げた際の表が出る回数」に対応する乱数
- **一様乱数** : 決まった区間 (a, b) からランダムに発生する乱数
- **正規乱数** : 平均 μ, 分散 σ^2 の正規分布に従う乱数

乱数を生成する関数

- 関数 `sample()` : ランダムサンプリング
- 関数 `rbinom()` : 二項乱数
- 関数 `runif()` : 一様乱数
- 関数 `rnorm()` : 正規乱数

練習問題

- ヘルプを用いて以下の関数を調べよ
  - 関数 `sample()`
  - 関数 `rbinom()`
  - 関数 `runif()`
  - 関数 `rnorm()`
  - 関数 `set.seed()`
- 以下の試行を実装してみよ
  - サイコロを 10 回振る
  - 4 枚のコインを投げたときの表の枚数

モンテカルロ法

モンテカルロ法とは

- 乱数を使った統計実験
- 計算機上でランダムネスを実現 (擬似乱数)
- ランダムネスから導かれる種々の数学的結果を観察

例： 中心極限定理

- 定理

  $X_1, X_2, \ldots$ を独立同分布な確率変数列とし，その平均を μ，標準偏差を σ とする．このとき，すべての実数 $a < b$ に対して

  $$P\left(a \leq \frac{\sqrt{n}(\bar{X}_n - \mu)}{\sigma} \leq b\right) \to \frac{1}{\sqrt{2\pi}} \int_a^b e^{-\frac{x^2}{2}} dx \quad (n \to \infty)$$

  が成り立つ．

- 直感的には “多数の独立な確率変数の和はほぼ正規分布に従う” ことを主張している

```
## 確率変数の分布の設定 (例 : 区間 [-1,1] の一様乱数)
myrand <- function(n) { # n 個の乱数を生成
  return(runif(n,min=-1,max=1))
}
## 標本平均の計算
mymean <- function(n) { # n 個のデータで計算
  return(mean(myrand(n)))
}
```

```
## Monte-Carlo 実験
set.seed(123) # 実験を再現したい場合はシードを指定する
mu <- 0; sigma <- sqrt(1/3) # 理論平均と標準偏差
mc <- 5000 # 実験の繰り返し回数
for(n in c(1,2,4,8,16)){ # n を変えて実験
  xbars <- replicate(mc, mymean(n)) # mc 回実験し標本平均を記録
  hist(xbars, breaks=25, freq=FALSE, # 分布を表示
       col="orchid", border="slateblue",
       xlab=expression(bar(X)), main=paste0("n=",n))
  thdist <- function(x){dnorm(x,mean=mu,sd=sigma/sqrt(n))}
  curve(thdist, add=TRUE, col="orange", lwd=2) # 理論曲線を重ねる
}
```

例： コイン投げの賭け

- A と B の二人で交互にコインを投げる．最初に表が出た方を勝ちとするとき，A と B それぞれの勝率はいくつとなるか？
- コイン投げは関数 `sample()`, `rbinom()` などを用いて模擬できる

```
sample(0:1,1) # 0 と 1 が入った壺からから 1 つ選ぶ
rbinom(1,size=1,prob=0.5) # 表裏が等確率で出る 1 枚のコインを 1 回投げる
```

```
## コイン投げの試行 (いろいろな書き方があるので以下は一例)
my_trial <- function(){
  while(TRUE){ # 永久に回るループ
    if(rbinom(1,size=1,prob=0.5)==1){return("A")} # A が表で終了
    if(rbinom(1,size=1,prob=0.5)==1){return("B")} # B が表で終了
    ## どちらも裏ならもう一度ループ
  }
}
```

```
## Monte-Carlo 実験
set.seed(8888) # 実験を再現したい場合はシードを指定する
mc <- 10000 # 実験回数を設定
my_data <- replicate(mc,my_trial())
## 簡単な集計
table(my_data)     # 頻度
```

```r
table(my_data)/mc # 確率（推定値）
```

練習問題

- 以下の簡単な双六ゲームの実験を行ってみよう
  - ゴールまでのます目は 100
  - さいころを振り出た目の数だけ進む
  - ゴールに辿り着くまで繰り返す
  - さいころを振る回数の分布は?

補足

- より詳細な確率シミュレーションについては以下を参照して下さい
  - 講義ノート
    - R 言語の基礎 第 5 章 (pp71-82)
      - Buffon の針
      - Monty Hall 問題
      - 秘書問題 (最適停止問題)

      などの実装例がある
  - 統計データ解析 I スライド
    - 第 5 講 確率シミュレーション

次回の予定

- **第 1 回: 回帰モデルの考え方と推定**
- 第 2 回: モデルの評価
- 第 3 回: モデルによる予測と発展的なモデル