

で  を操る方法

～ SAS で R の出力を～

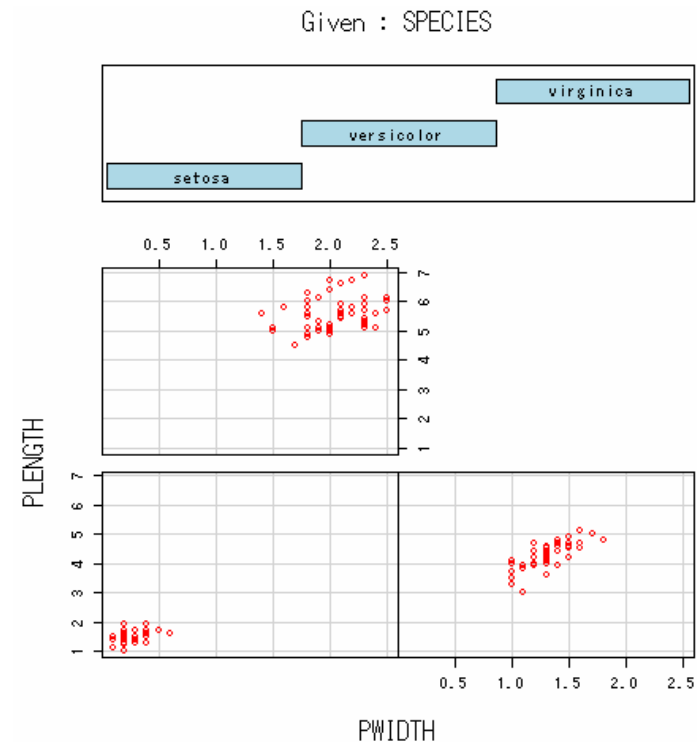
SAS 上で R の出力を得る例 (1)



- R を知らない方でも、SAS 上で R を操作するプログラムがあれば R の出力を得ることが出来る

```
data _null_ ;  
  file aaa ;  
  put 'library(foreign)';  
  put 'library(lattice)';  
  put 'x <- read.xport("C:/iris.xpt)';  
  put 'bmp("C:/test.bmp)';  
  put 'coplot(PLENGTH ~ PWIDTH | SPECIES, x, '  
  put 'col="red", bg="pink", bar.bg=c(fac="light  
  blue"))';  
  put 'dev.off()';  
run ;
```

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
  --no-save < "C:/test.R" > "C:/test.log" ;
```



条件付散布図

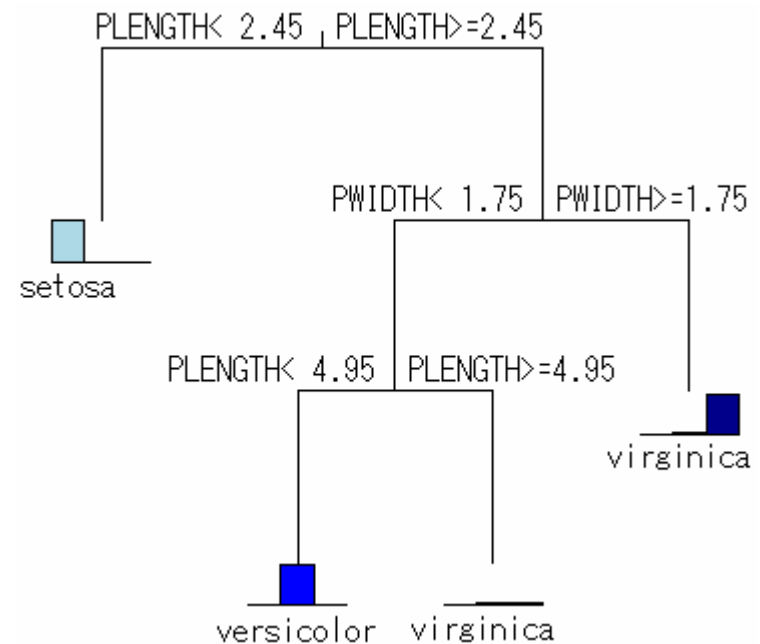
SAS 上で R の出力を得る例 (2)



- R を知らない方でも、SAS 上で R を操作するプログラムがあれば R の出力を得ることが出来る

```
data _null_ ;  
  file aaa ;  
  put 'library(foreign)';  
  put 'library(mvpart)';  
  put 'x <- read.xport("C:/iris.xpt)";  
  put 'result <- rpart(SPECIES~., data=x)';  
  put 'bmp("C:/test.bmp)';  
  put 'par(xpd=T)';  
  put 'plot(result, uniform=T, margin=0.1)';  
  put 'text(result); dev.off()';  
run ;
```

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
  --no-save < "C:/test.R" > "C:/test.log" ;
```



回帰樹 (mvpartパッケージ)

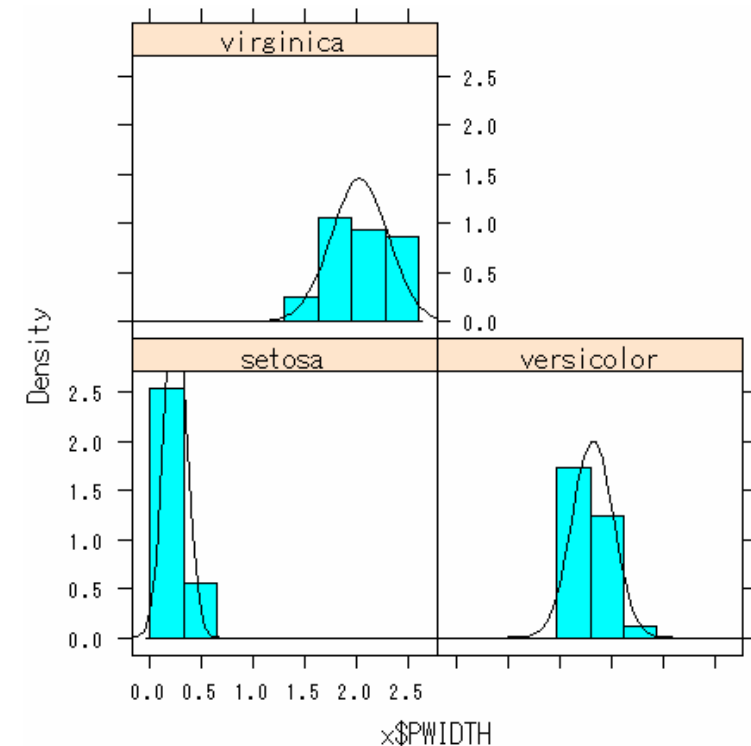
SAS 上で R の出力を得る例 (3)



- R を知らない方でも、SAS 上で R を操作するプログラムがあれば R の出力を得ることが出来る

```
data _null_ ;
  file aaa ;
  put 'library(foreign)';
  put 'library(lattice)';
  put 'x <- read.xport("C:/iris.xpt)';
  put 'bmp("C:/test.bmp)';
  put 'histogram(~PWIDTH | SPECIES, x, ' ;
  put 'type="density", panel=function(x, ...) {' ;
  put 'panel.histogram(x, ...)';
  put 'panel.mathdensity(dmath=dnorm, col=1, ' ;
  put 'args = list(mean=mean(x),sd=sd(x))) } )';
  put 'dev.off()';
run ;
```

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"
  --no-save < "C:/test.R" > "C:/test.log" ;
```



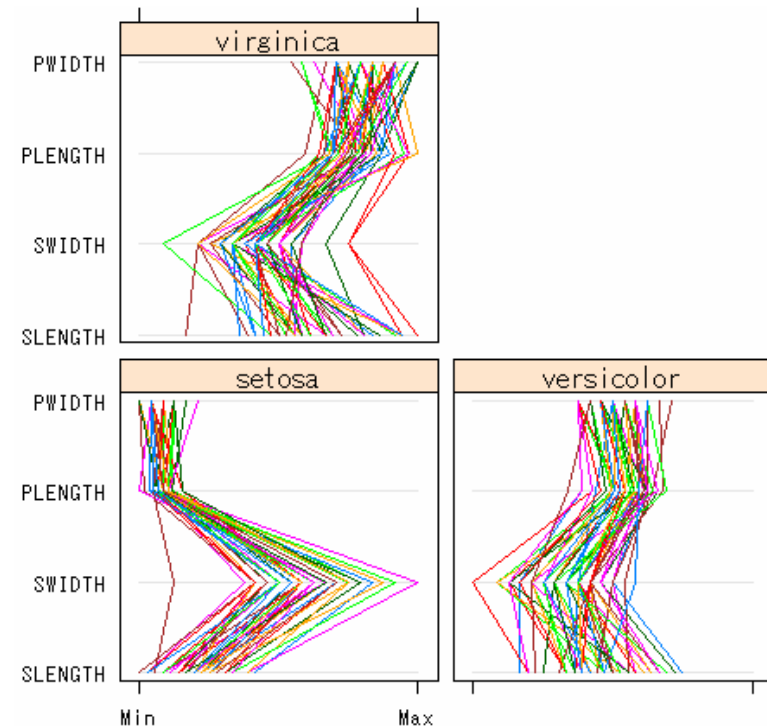
ヒストグラム
(latticeパッケージ)

SAS 上で R の出力を得る例 (4)



- R を知らない方でも、SAS 上で R を操作するプログラムがあれば R の出力を得ることが出来る

```
data _null_ ;  
  file aaa ;  
  put 'library(foreign)';  
  put 'library(lattice)';  
  put 'x <- read.xport("C:/iris.xpt)';  
  put 'bmp("C:/test.bmp)';  
  put 'parallel(~x[1:4] | SPECIES, x)';  
  put 'dev.off()';  
run ;  
  
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
  --no-save < "C:/test.R" > "C:/test.log" ;
```

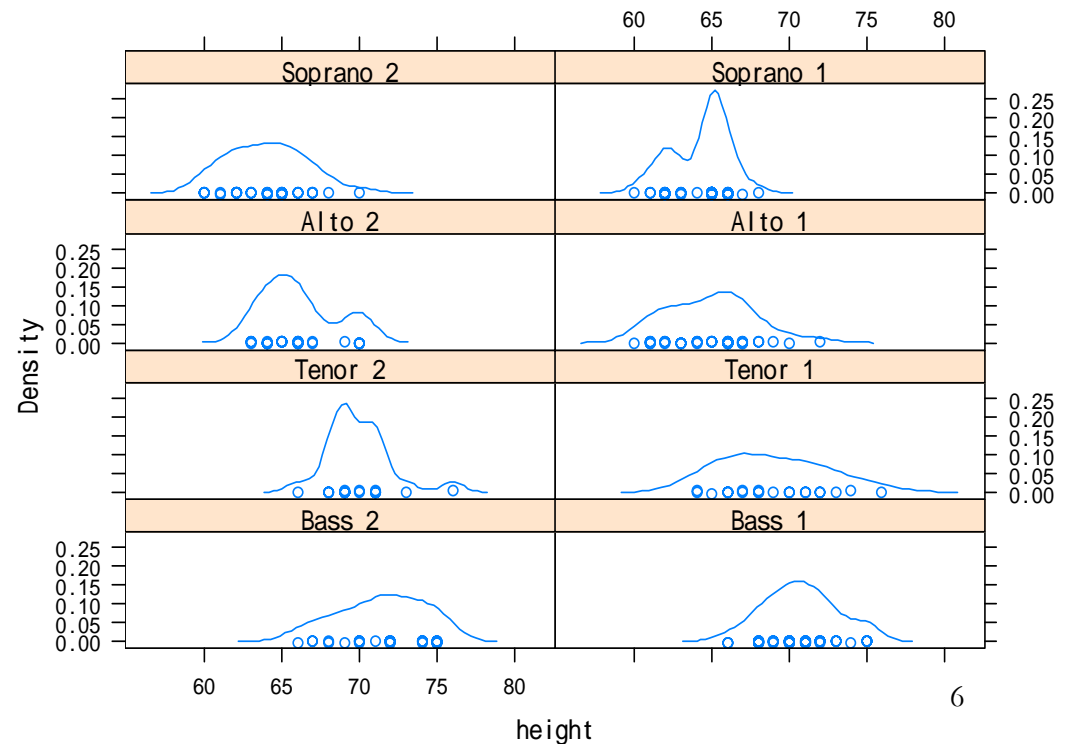
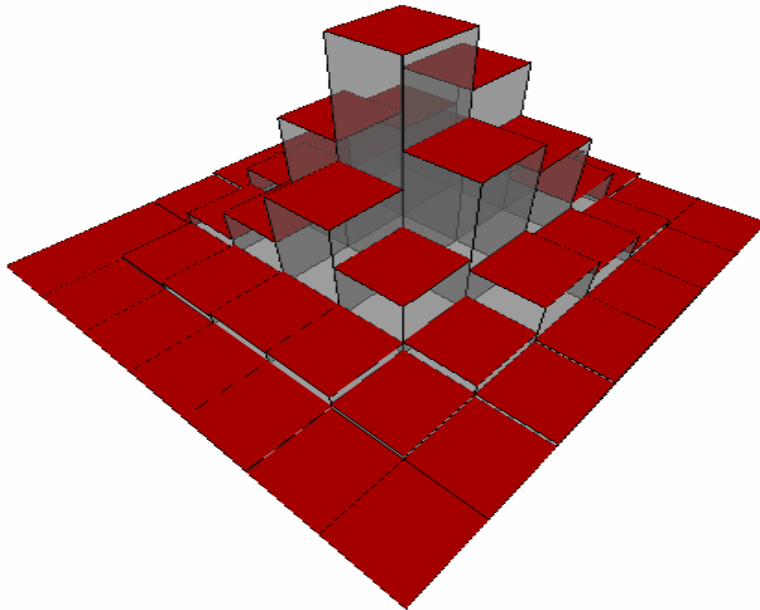


パラレルプロット
(latticeパッケージ)

本日のメニュー



- データフレームのおさらい ←
- SAS から R を操作する(簡単な例を 1 つ)
- SAS から R を操作する(簡単な例を 2 つ)



データフレームとは



- 統計解析を行うデータの形式は様々
 - (R上で) データを手で入力して...
 - テキストファイル, EXCEL, ACCESS, SAS などの形式
- Rでデータ解析を行う際は, データフレームという形式にデータを変換することが多い (見た目は行列)

	A	B	C
1	sex	height	weight
2	F	160	50
3	F	165	65
4	M	170	60
5	M	175	55
6	M	180	70

EXCEL : シート

	sex	height	weight
	F	160	50
	F	165	65
	M	170	60
	M	175	55
	M	180	70

ACCESS : テーブル

	sex	height	weight
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70

SAS : データセット

データフレームの作成



- R でベクトルデータを作成した後、データフレームを作成 (いわゆる手入力)
 - ⇒ 「性別」「身長」「体重」データをベクトルで用意した後、関数 [data.frame\(\)](#) で1つのデータフレームに変換する
- ファイルからデータを読み込んで、データフレームを作成
 - ⇒ 関数 [read.table\(\)](#) などファイルからデータを読み込
 - ⇒ パッケージ RODBC の関数 [odbcConnectXXXXX\(\)](#) でデータファイルにアクセスした後、関数 [sql.Query\(\)](#) でファイルからデータを読み込

データフレームの作成（手入力）



SEX	HEIGHT	WEIGHT
F	158	51
F	162	55
M	177	72
M	173	57
M	166	64

data.frame()



SEX	HEIGHT	WEIGHT
F	158	51
F	162	55
M	177	72
M	173	57
M	166	64

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

```
> height <- c(158, 162, 177, 173, 166)
```

```
> weight <- c( 51, 55, 72, 57, 64)
```

```
> x <- data.frame(SEX=sex, HEIGHT=height,  
                  WEIGHT=weight)
```

データフレームを作成すると・・・



> summary(x) # データフレームの列ごとの特徴を見る

```
SEX          HEIGHT          WEIGHT
F:2  Min.      :158.0  Min.      :51.0
M:3  1st Qu.   :162.0  1st Qu.   :55.0
      Median   :166.0  Median   :57.0
      Mean     :167.2  Mean     :59.8
      3rd Qu.  :173.0  3rd Qu.  :64.0
      Max.     :177.0  Max.     :72.0
```

SEX	HEIGHT	WEIGHT
F	158	51
F	162	55
M	177	72
M	173	57
M	166	64

散布図

> plot(x)

CART (回帰樹)

> library(rpart)

> rpart(SEX ~ ., data=x)

> par(xpd=T); plot(result); text(result)

データフレームの作成 (⇔ .txt)



- 関数 `read.table()` などでテキストファイルからデータを読み込むことが出来る

```
> x <- read.table("mydata.txt",  
                  header=T, sep="," )
```

	sex	height	weight
1	F	158	51
2	F	162	55
3	M	177	72
4	M	173	57
5	M	166	64

```
> x <- read.csv("mydata.txt")
```

```
C:¥  
sex,height,weight  
F,158,51  
F,162,55  
M,177,72  
M,173,57  
M,166,64
```

mydata.txt

データフレームの作成 (RODBC)



- パッケージ **RODBC** 中の関数 **odbcConnectXXXX()** でデータファイルにアクセスした後、関数 **sql.Query()** でファイルからデータを読み込むことが出来る

```
> library(RODBC) # パッケージの呼出
> tmp <- odbcConnectExcel("c:/data00.xls") # データに接続
> tmp <- odbcConnectAccess("c:/data00.mdb") # (Access の場合)
> sqlTables(tmp) # テーブルを表示
> x <- sqlQuery(tmp, "select * from [Sheet1$]") # 読み込み
> x <- sqlQuery(tmp, "select * from [mydata]") # (Access の場合)
> odbcClose(tmp) # 接続を遮断
```

- 他にも **ORACLE** のデータベースや、その他のデータ形式ファイル (**DBASE**, **MySQL**, **PostgreSQL**) からデータフレームを作成することもできる

データフレームの作成 (foreign)



- パッケージ **foreign** の中には、外部データを読み込むための関数が多数用意されている

関数	用途
<code>data.restore()</code>	Read an S3 Binary File
<code>read.dbf()</code>	Read a DBF File
<code>read.dta()</code>	Read Stata Binary Files
<code>read.epiinfo()</code>	Read Epi Info Data Files
<code>read.mtp()</code>	Read a Minitab Portable Worksheet
<code>read.octave()</code>	Read Octave Text Data Files
<code>read.spss()</code>	Read an SPSS Data File
<code>read.ssd()</code>	Obtain a Data Frame from a SAS Permanent
<code>read.systat()</code>	Obtain a Data Frame from a Systat File
<code>read.xport()</code>	Read a SAS XPORT Format Library

データフレームの作成 (foreign)



> library(foreign)

■ SPSS データを R に読み込む例

> x <- read.spss("mydata.sav")

■ SAS データを R に読み込む例

*** SAS データを XPT ファイルに変換 ;

```
libname out xport "C:/mydata.xpt" ;
```

```
proc copy in=work out=out ;
```

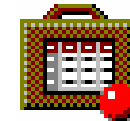
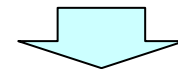
```
select mydata / mt=data ;
```

```
run ;
```

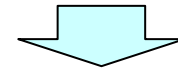
R から SAS の XPT ファイルを読み込む

```
> x <- read.xport("C:/mydata.xpt")
```

	sex	height	weight
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70



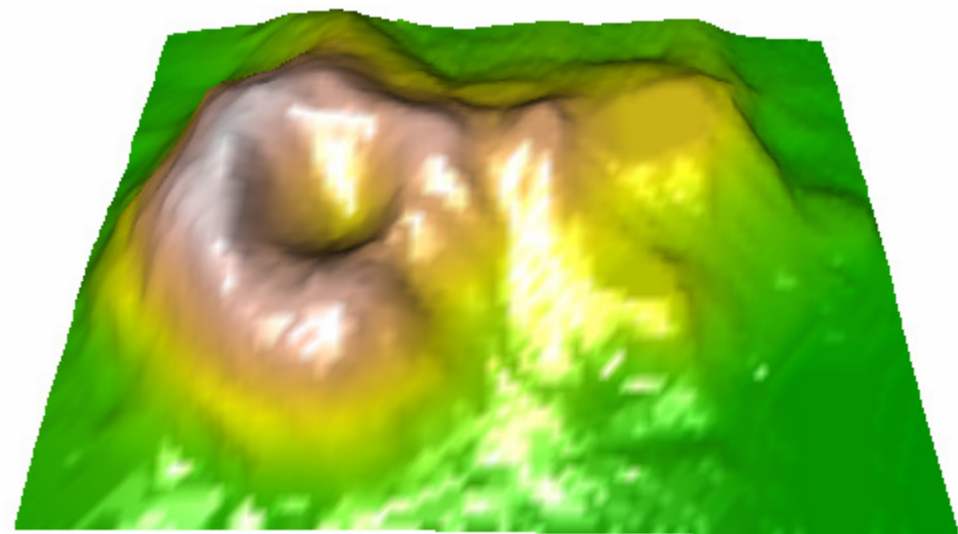
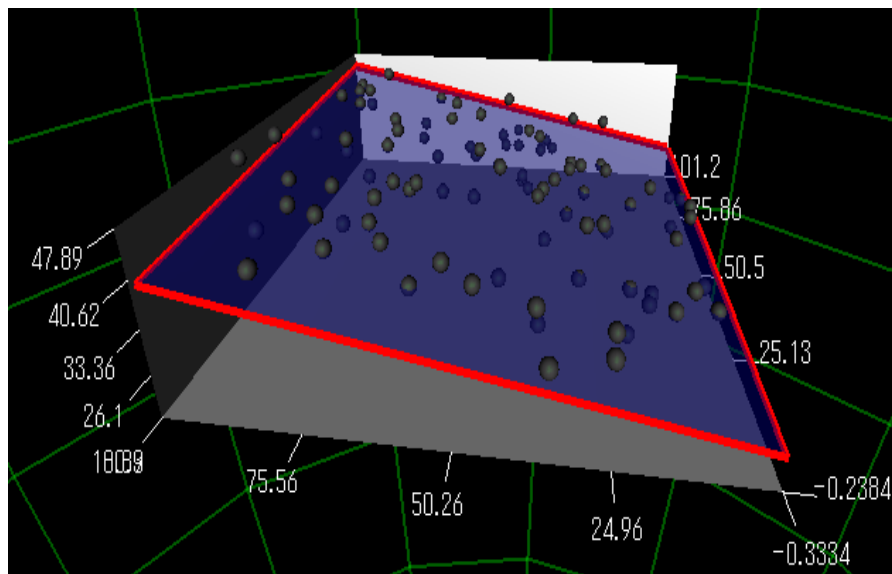
xpt



本日のメニュー



- データフレームのおさらい
- SAS から R を操作する(簡単な例を 1 つ)←
- SAS から R を操作する(簡単な例を 2 つ)



SAS から R を操作する (1)



★ 手順は以下の通り

1. 実行する R のプログラムを用意する

- あらかじめ R のプログラムソースを用意する

- SAS 上で R のプログラムソースを作成する (←今回)

2. x コマンドを用いて DOS を起動し R を実行する

SAS から R を操作する (2)



■ まずはおまじない (R ソースの出力先の指定など)

```
*--- SAS から R を操作する ;  
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data _null_ ;  
  file aaa ;  
  put 'x <- 1:5; sink("C:/out.txt"); mean(x)';  
run ;
```

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
  --no-save < "C:/test.R" > "C:/test.log" ;
```

- Rのファイル名の指定
- オプションの設定
 - ⇒ noxwait : DOSを自動終了
 - ⇒ xsync : 実行されたアプリの処理が終わるまで待機

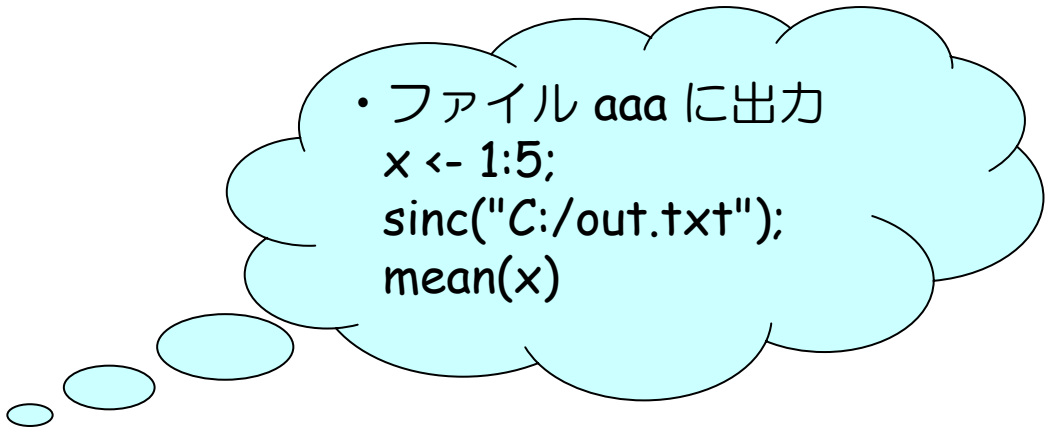
SAS から R を操作する (3)



■ put 文で R プログラムを作成・ファイルに出力する

```
*--- SAS から R を操作する ;  
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data null ;  
file aaa ;  
put 'x <- 1:5; sink("C:/out.txt"); mean(x)';  
run ;
```



• ファイル aaa に出力
x <- 1:5;
sink("C:/out.txt");
mean(x)

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
--no-save < "C:/test.R" > "C:/test.log";
```

SAS から R を操作する (4)

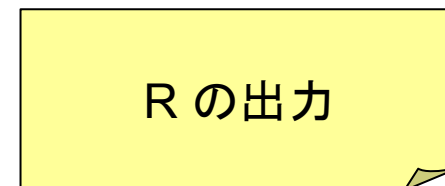
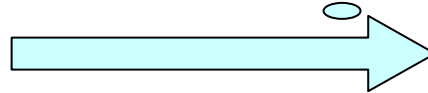
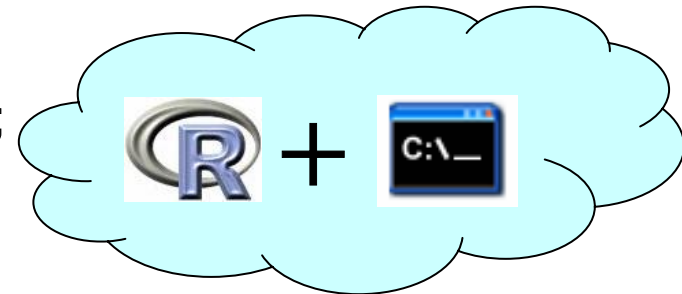


■ x コマンドを用いて DOS を起動し, R を実行する

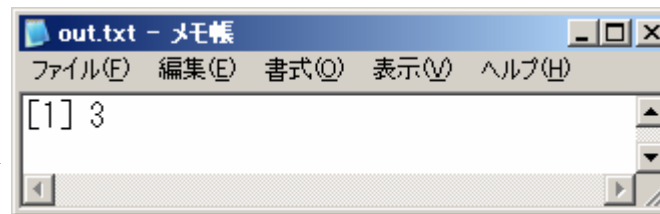
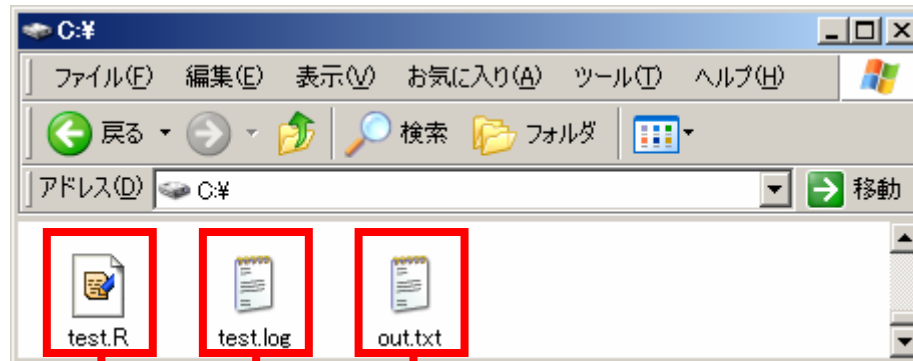
```
*--- SAS から R を操作する ;  
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data _null_ ;  
  file aaa ;  
  put 'x <- 1:5; sink("C:/out.txt"); mean(x)';  
run ;
```

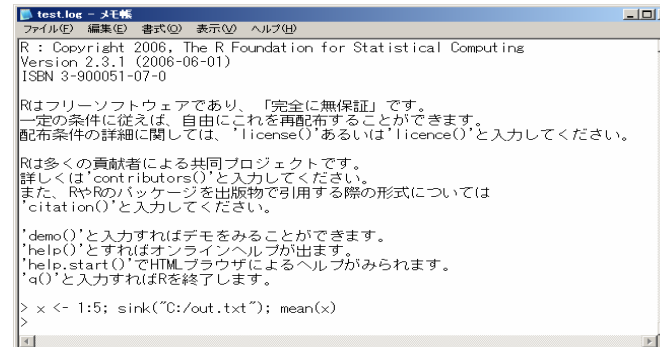
```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
--no-save < "C:/test.R" > "C:/test.log";
```



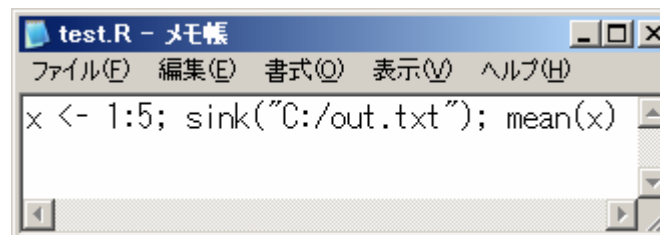
SAS から R を操作する (出力結果)



計算結果



ログファイル



プログラムファイル

★ <寄り道> 複数のプログラムを実行する

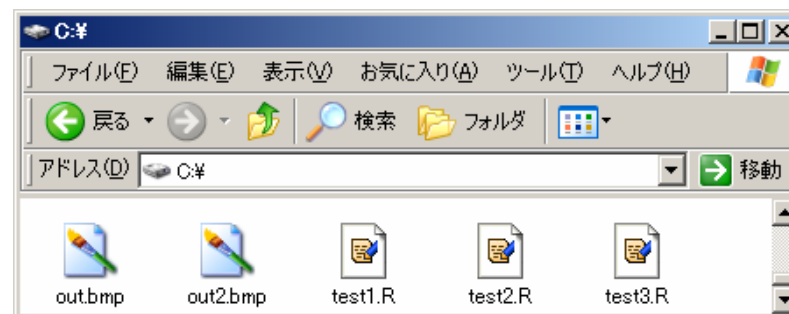


```
%let a='C:/test1.R' ;  
%let b='C:/test2.R' ;  
%let c=%str(C:/test3.R) ;  
filename aaa &a. ;  
filename bbb &b. ;  
filename ccc "&c" ;
```

複数のファイルを実行
するプログラムソース
を作成すればよい

```
data _null_ ;  
  file aaa ;  
  put 'x <- 1:5; bmp("C:/out.bmp"); plot(x); dev.off()' ;  
run ;  
data _null_ ;  
  file bbb ;  
  put 'x <- 1:10; bmp("C:/out2.bmp"); plot(x); dev.off()' ;  
run ;  
data _null_ ;  
  file ccc ;  
  put "source(&a.); source(&b.)" ;  
run ;
```

```
options noxwait xsync ;  
x "'C:/Program Files/R/R-2.3.1/bin/R.exe' --no-save < &c " ;
```



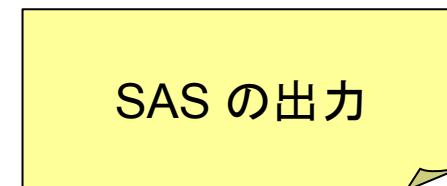
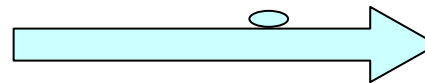
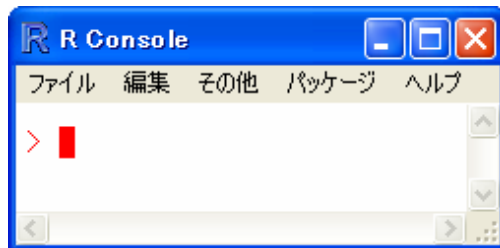
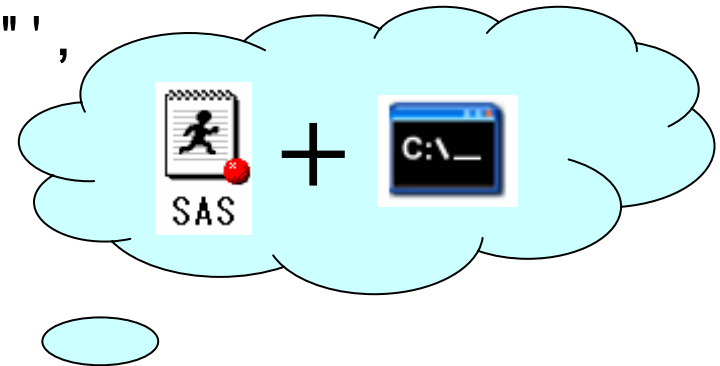
★ <寄り道> R から SAS を操作する



■ 関数 `system()` を用いる

R から SAS を操作する

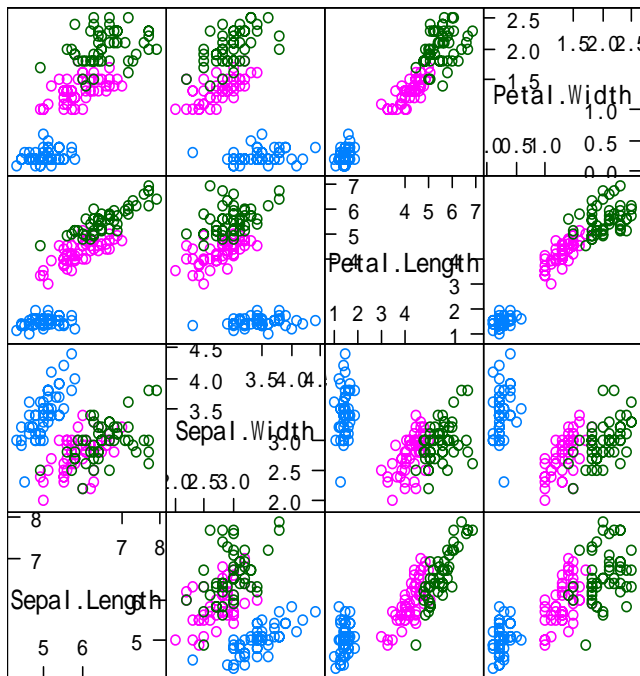
```
> cat("proc print data=sashelp.class;¥n",  
+     file="C:/test.sas")  
> cat("run;¥n", file="C:/test.sas",  
+     append=T)  
> system(paste(' "C:/SASV8/nls/ja/sas.exe" ',  
+              '"C:/test.sas" ',  
+              '-log C:/test.log  
+              -print C:/test.lst '),  
+        wait = F)
```



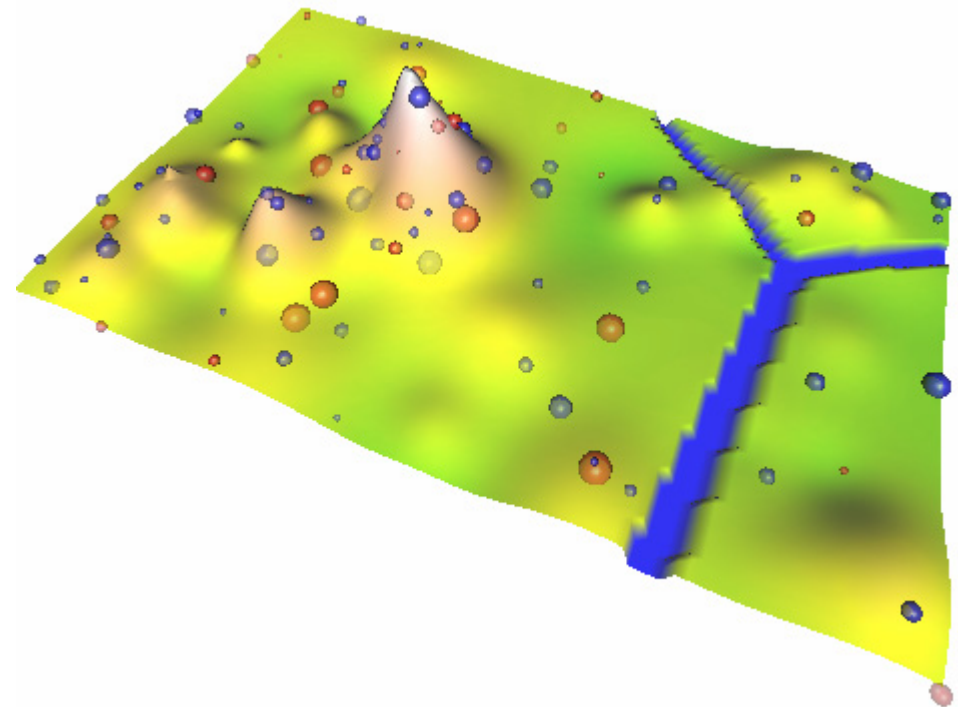
本日のメニュー



- データフレームのおさらい
- SAS から R を操作する(簡単な例を 1 つ)
- SAS から R を操作する(簡単な例を 2 つ)←



Scatter Plot Matrix



SAS から R を操作する (1-1)



- ★ フィッシャーのアヤメの分類データの解析
 - データは「C:/iris.xls」にあると仮定する
 - SAS 上で R のソースを作成し対散布図を作成する
- 1. SAS 上で R のソースを作成
- 2. x コマンドを用いて DOS を起動し R を実行する

	A	B	C	D	E
1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
2	5.1	3.5	1.4	0.2	setosa
3	4.9	3	1.4	0.2	setosa
4	4.7	3.2	1.3	0.2	setosa
5	4.6	3.1	1.5	0.2	setosa
6	5	3.6	1.4	0.2	setosa
7	5.4	3.9	1.7	0.4	setosa
8	4.6	3.4	1.4	0.3	setosa
9	5	3.4	1.5	0.2	setosa
10	4.4	2.9	1.4	0.2	setosa

- Sepal.Length: がく片の長さ
- Sepal.Width: がく片の幅
- Petal.Length: 花びらの長さ
- Petal.Width: 花びらの幅
- Species: アヤメの種類

SAS から R を操作する (1-2)



- まずはおまじない (R ソースの出力先の指定など)

```
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data _null_ ;  
  file aaa ;  
  put 'library(RODBC)';  
  put 'tmp <- odbcConnectExcel("C:/iris.xls)';  
  put 'sqlTables(tmp)';  
  put 'x <- sqlQuery(tmp,"select * from [mydata$])';  
  put 'odbcClose(tmp)';  
  put 'bmp("C:/test.bmp)';  
  put 'plot(x)';  
  put 'dev.off()';  
run ;
```

- Rのファイル名の指定
- オプションの設定

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
  --no-save < "C:/test.R" > "C:/test.log" ;
```

SAS から R を操作する (1-3)



■ まずはおまじない (R ソースの出力先の指定など)

```
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data null ;  
file aaa ;  
put 'library(RODBC)';  
put 'tmp <- odbcConnectExcel("C:/iris.xls)';  
put 'sqlTables(tmp)';  
put 'x <- sqlQuery(tmp,"select * from [mydata$])';  
put 'odbcClose(tmp)';  
put 'bmp("C:/test.bmp)';  
put 'plot(x)';  
put 'dev.off()';  
run ;
```

- iris.xlsの読み込み
- 散布図を作成した後
test.bmpに書き込み

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
--no-save < "C:/test.R" > "C:/test.log" ;
```

SAS から R を操作する (1-4)



- まずはおまじない (R ソースの出力先の指定など)

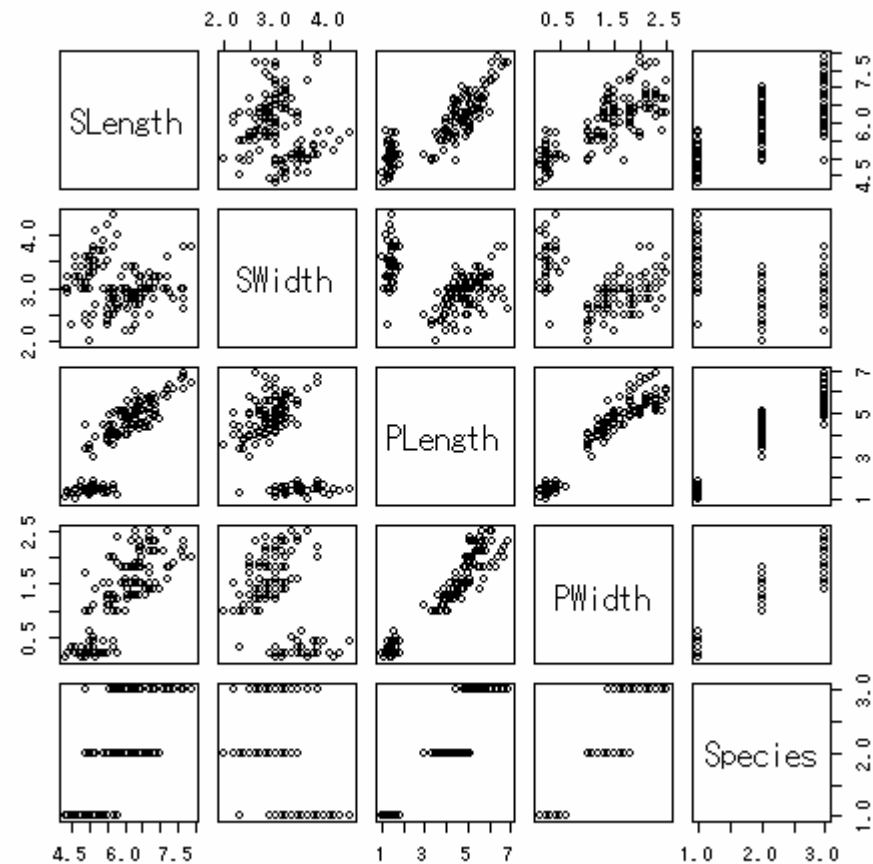
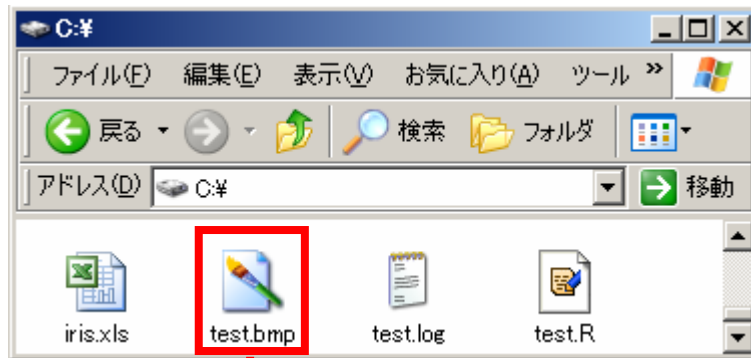
```
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data _null_ ;  
  file aaa ;  
  put 'library(RODBC)';  
  put 'tmp <- odbcConnectExcel("C:/iris.xls)';  
  put 'sqlTables(tmp)';  
  put 'x <- sqlQuery(tmp,"select * from [mydata$])';  
  put 'odbcClose(tmp)';  
  put 'bmp("C:/test.bmp)';  
  put 'plot(x)';  
  put 'dev.off()';  
run ;
```

- xコマンド実行
- ログファイルの作成

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
--no-save < "C:/test.R" > "C:/test.log" ;
```

SAS から R を操作する (1-結果)



SAS から R を操作する (2-1)



- ★ フィッシャーのアヤメの分類データの解析
 - データ「iris」はSASのWORKにあると仮定する
 - SAS 上で R のソースを作成し回帰樹を作成する
- 1. SAS 上でデータ「iris」を XPT ファイルに変換する
- 2. SAS 上で R のソースを作成
- 3. x コマンドを用いて DOS を起動し R を実行する

	SLength	SWidth	PLength	PWidth	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

- SLength: がく片の長さ
- SWidth: がく片の幅
- PLength: 花びらの長さ
- PWidth: 花びらの幅
- Species: アヤメの種類
(変数名は8文字まで...)

SAS から R を操作する (2-2)




- まずはおまじない (R ソースの出力先の指定など)

```
libname out xport "C:/iris.xpt" ;  
proc copy in=work out=out ; select iris / mt=data ;  
run ;
```

```
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data _null_ ;  
  file aaa ;  
  put 'library(foreign); library(rpart)';  
  put 'x <- read.xport("C:/iris.xpt)';  
  put 'result <- rpart(SPECIES ~ ., data=x)';  
  put 'bmp("C:/test.bmp)';  
  put 'par(xpd=T); plot(result); text(result)';  
  put 'dev.off()';  
run ;
```

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
  --no-save < "C:/test.R" > "C:/test.log" ;
```



XPTファイルの作成

SAS から R を操作する (2-3)



■ まずはおまじない (R ソースの出力先の指定など)

```
libname out xport "C:/iris.xpt" ;  
proc copy in=work out=out ; select iris / mt=data ;  
run ;
```

```
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

- Rのファイル名の指定
- オプションの設定

```
data _null_ ;  
  file aaa ;  
  put 'library(foreign); library(rpart)';  
  put 'x <- read.xport("C:/iris.xpt)';  
  put 'result <- rpart(SPECIES ~ ., data=x)';  
  put 'bmp("C:/test.bmp)';  
  put 'par(xpd=T); plot(result); text(result)';  
  put 'dev.off()';  
run ;
```

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
  --no-save < "C:/test.R" > "C:/test.log" ;
```

SAS から R を操作する (2-4)



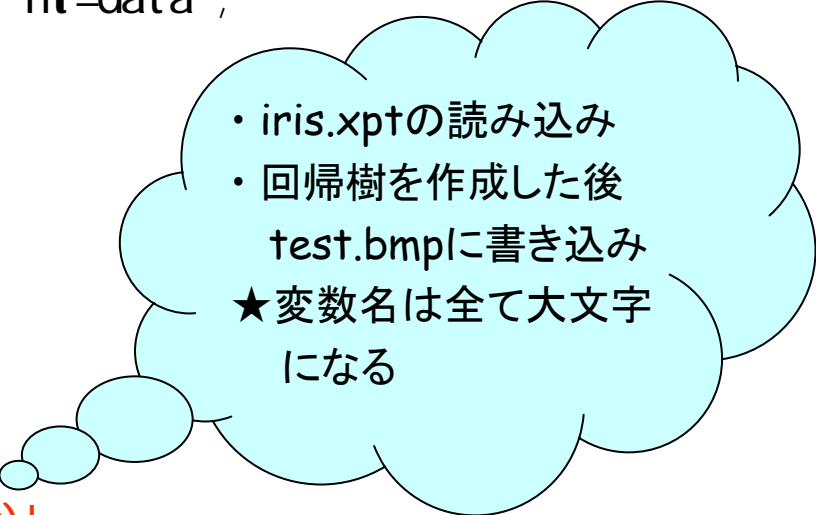
■ まずはおまじない (R ソースの出力先の指定など)

```
libname out xport "C:/iris.xpt" ;  
proc copy in=work out=out ; select iris / mt=data ;  
run ;
```

```
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data null ;  
file aaa ;  
put 'library(foreign); library(rpart)';  
put 'x <- read.xport("C:/iris.xpt)';  
put 'result <- rpart(SPECIES ~ ., data=x)';  
put 'bmp("C:/test.bmp)';  
put 'par(xpd=T); plot(result); text(result)';  
put 'dev.off()';  
run ;
```

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
--no-save < "C:/test.R" > "C:/test.log" ;
```

- 
- iris.xptの読み込み
 - 回帰樹を作成した後
test.bmpに書き込み
 - ★変数名は全て大文字
になる

SAS から R を操作する (2-5)



■ まずはおまじない (R ソースの出力先の指定など)

```
libname out xport "C:/iris.xpt" ;  
proc copy in=work out=out ; select iris / mt=data ;  
run ;
```

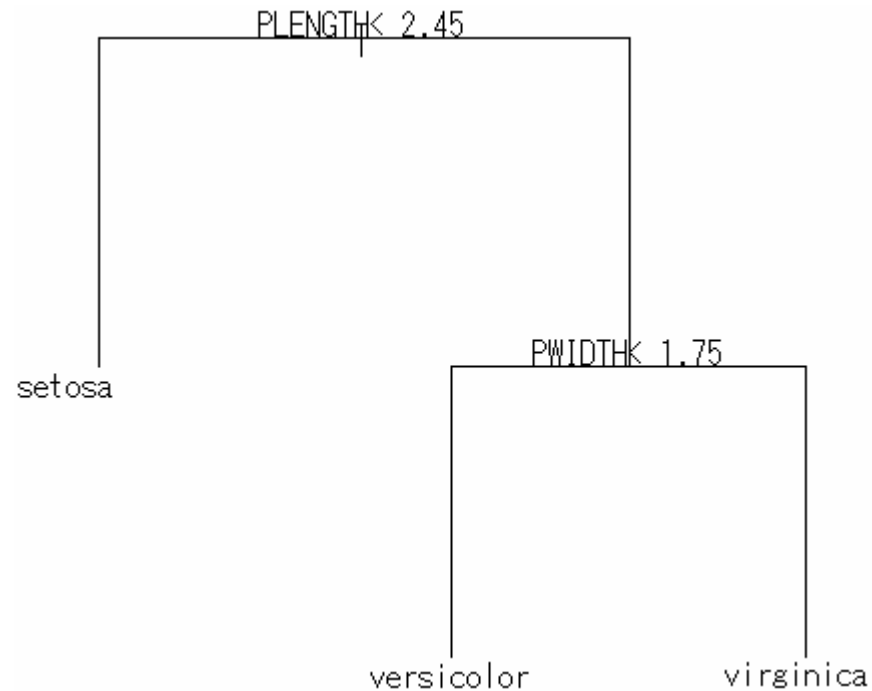
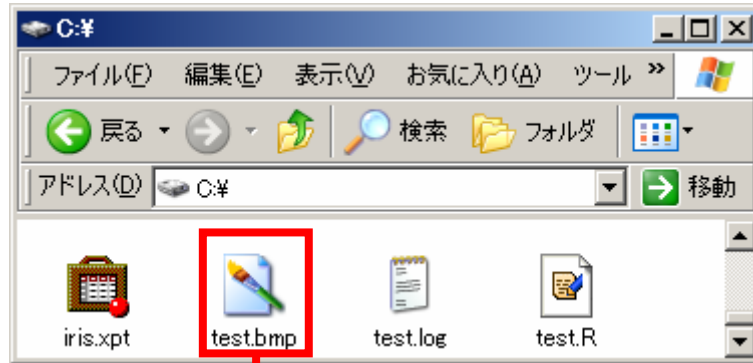
```
filename aaa "C:/test.R" ;  
options noxwait xsync ;
```

```
data _null_ ;  
  file aaa ;  
  put 'library(foreign); library(rpart)';  
  put 'x <- read.xport("C:/iris.xpt")';  
  put 'result <- rpart(SPECIES ~ ., data=x)';  
  put 'bmp("C:/test.bmp)';  
  put 'par(xpd=T); plot(result); text(result)';  
  put 'dev.off()';  
run ;
```

- xコマンド実行
- ログファイルの作成

```
x "C:/Program Files/R/R-2.3.1/bin/R.exe"  
--no-save < "C:/test.R" > "C:/test.log" ;
```

SAS から R を操作する (2-結果)



本日のメニュー



- データフレームのおさらい
 - データフレームとは
 - データフレームの作成方法
- SAS から R を操作する(例 1)
 - SAS 上で R のプログラムソースを作成する
 - x コマンドを用いて DOS を起動し R を実行する
- SAS から R を操作する(例 2)
 - EXCELデータを用意して散布図を作成する方法
 - SASデータを用意して回帰樹を作成する方法
 - ⇒ SAS では解析できない(≡機能追加にお金がかかる...)
場合には、SAS 上で R を操作する方法が有効な場合も
 - ⇒ R を知らない方でも、SAS 上で R を操作する
プログラムがあれば R の出力を得ることが出来る

SAS 上で R を操作するマクロ (1)



■ SAS 上で R を操作するマクロの例(条件付散布図)

```
%macro Histogram(INDIR, INDATA, FORMULA, OUTPASS);  
  %let FORMULA2=%upcase(&FORMULA);  
  %let PASS   =%sysfunc(tranwrd(%trim(&OUTPASS), ¥, /));  
  libname out xport "C:/tmp.xpt" ;  
  proc copy in=&INDIR out=out ; select &INDATA / mt=data ;  
  run ;  
  filename aaa "C:/tmp.R" ; options noxwait xsync ;  
  data _null_ ;  
    file aaa ;  
    put 'library(foreign); library(lattice)';  
    put 'x <- read.xport("C:/tmp.xpt)';  
    put "bmp('&PASS)";  
    put "histogram(%trim(&FORMULA2), x)";  
    put 'dev.off()';  
  run ;  
  x "C:/Program Files/R/R-2.3.1/bin/R.exe" --no-restore --no-save < "C:/tmp.R" ;  
  x 'cd c:¥' ;  
  x 'del tmp.xpt tmp.R ' ;  
%mend Histogram ;
```

•引数の文字を大文字に
•引数の「¥」を「/」に変換するコマンド

•INDIR : SASデータのライブラリ
•INDATA : SASデータ名
•FORMULA : モデル式
•OUTPASS : 出力先のパス

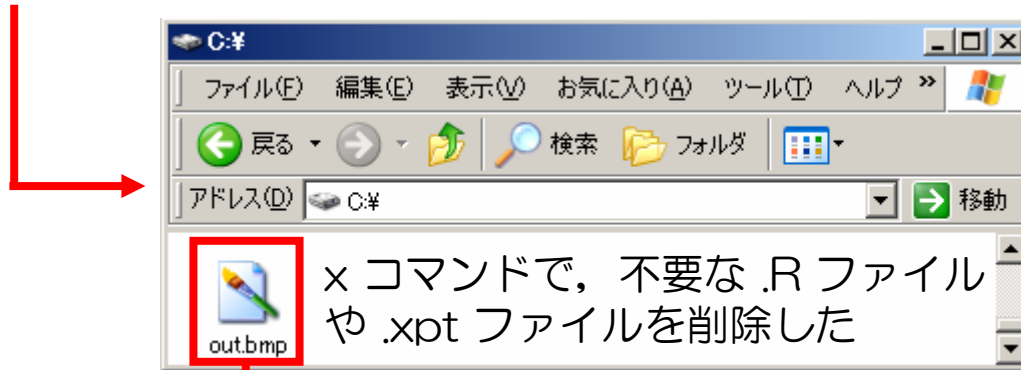
.Rdataを復元せずに
実行するコマンド

SAS 上で R を操作するマクロ (1)



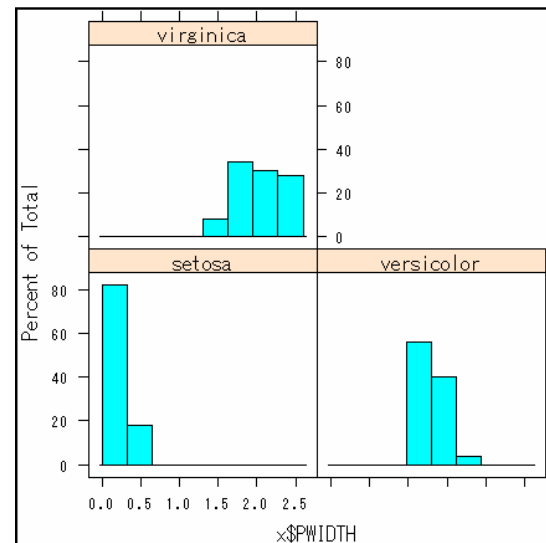
■ SAS 上で R を操作するマクロの例(条件付散布図)

```
%Histogram(WORK, IRIS, ~PwIdTh|Species, C:¥out.bmp) ;
```



× コマンドで, 不要な .R ファイル
や .xpt ファイルを削除した

引数中のモデル式には
大文字と小文字が混在
しているが, エラーは
出ない!!!



c:/out.bmp

SAS 上で R を操作するマクロ (2)



■ SAS 上で R を操作するマクロの例(散布図)

```
%macro Plot(X,Y,MAIN,SUB,XLAB,YLAB,TYPE=P,INDIR=WORK,INDATA=,OUTPASS=C:\%out.bmp);  
  %let PASS =%sysfunc(tranwrd(%trim(&OUTPASS),%,/));  
  %let X2=%upcase(&X);   %let Y2=%upcase(&Y);  
  %let TYPE2=%lowcase(&TYPE);  
  libname out xport "C:/tmp.xpt" ;  
  proc copy in=&INDIR out=out ; select &INDATA / mt=data ;  
  run ;  
  filename aaa "C:/tmp.R" ; options noxwait xsync ;  
  data _null_ ;  
    file aaa ;  
    put 'library(foreign); x <- read.xport("C:/tmp.xpt")';  
    put "bmp('&PASS)";      *--- type=p,l,b,c,o,h,s,S,n ;  
    put "plot(x=x$&X2, y=x$&Y2, type='&TYPE2', main='&MAIN',";  
    put "sub='&SUB', xlab='&XLAB', ylab='&YLAB', data=x)";  
    put 'dev.off()';  
  run ;  
  x "C:/R/R-2.4.0/bin/R.exe" --no-restore --no-save < "C:/tmp.R" ;  
  x 'cd c:%' ;  
  x 'del tmp.xpt tmp.R ' ;  
%mend Plot ;
```

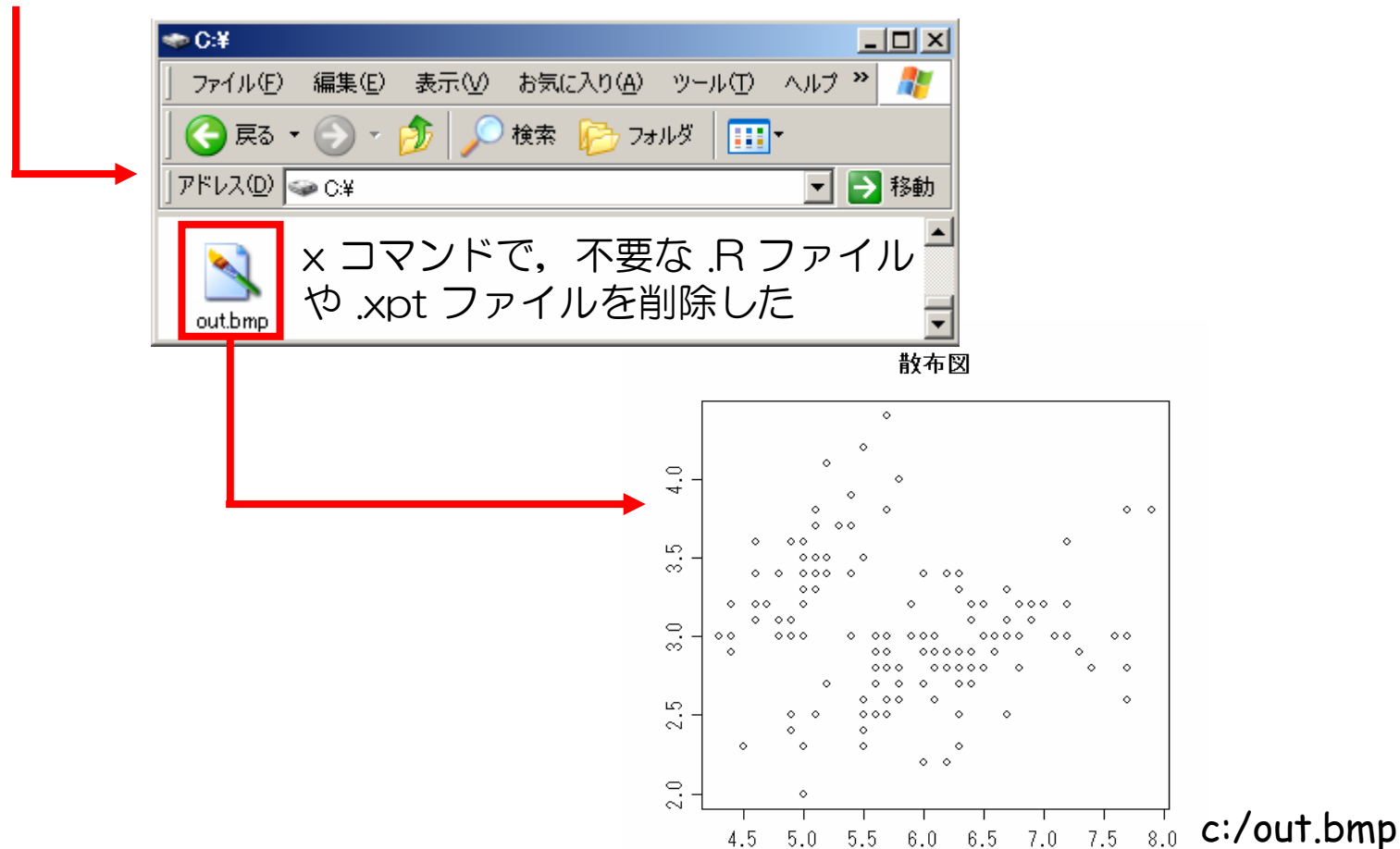
- X : X軸の変数名
- Y : Y軸の変数名
- MAIN : タイトル
- SUB : サブタイトル
- XLAB : X軸ラベル
- YLAB : Y軸ラベル
- TYPE : シンボル
- INDIR : ライブラリ
- INDATA : データ名
- OUTPASS : 出力先

SAS 上で R を操作するマクロ (2)



■ SAS 上で R を操作するマクロの例(散布図)

```
%Plot(X=SLENGTH, Y=SWIDTH, MAIN=散布図, INDIR=WORK, INDATA=IRIS, OUTPASS=C:¥out.bmp) ;
```



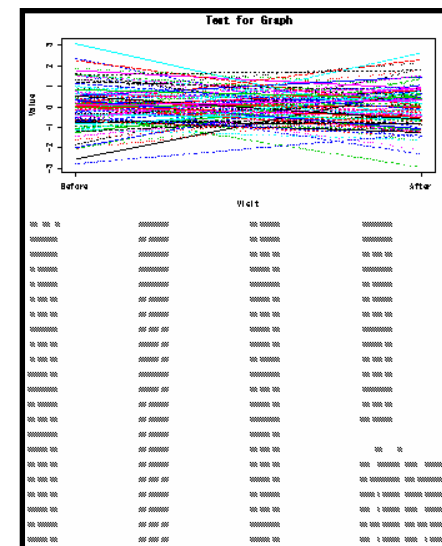
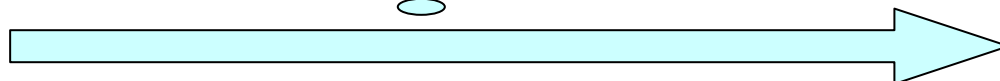
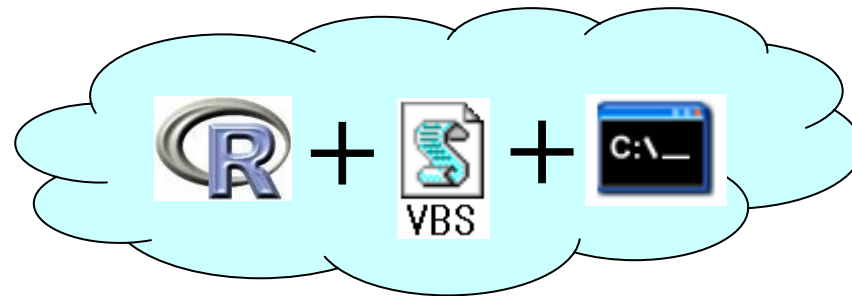
SAS 上で R を操作するマクロ (★)



- 本日紹介した手法を活用すると様々な出力が！

〔例〕 SAS を実行するだけで解析結果のWORDファイル
を出力する場合

⇒ 例えば「R」と「VBScript」と「MS-DOS」を
組み合わせることで実現できる



WORDファイル

★ <寄り道> SAS ⇒ XML ⇒ R (1)

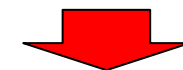


- パッケージ XML と SASXML を使用する
(作者: Duncan Temple Lang 氏 <duncan@research.bell-labs.com>)
- まず, SAS データセット ⇒ XML に変換する

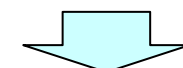
* --- SAS 上のコマンド;

```
libname xxx xml "C:/demo.xml"
      xmltype=oimdbm xmlschema=yes ;
proc format ;
  value sexf 1="Female"
           2="Male" ;
data demo(label="Test") ;
  input id name $ sex ;
  format sex sexf. ; label sex="Gender" ;
  cards;
      111 ABC 1
      222 DEF 2
      333 GHI 1
      444 JKL 2
;
proc copy in=work out=xxx ; select demo ; run;
```

	sex	height	weight
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70



.xml



★ <寄り道> SAS ⇒ XML ⇒ R (2)



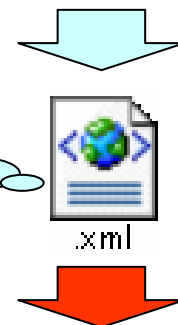
- 次に XML ⇒ R へ読み込む
 - パッケージ **SASXML** の中に入っているソース `eventSAS.S`, `sas.S`, `sasDataset.R` を実行する
 - その後, 以下を実行する

```
> library(XML)
> x <- sas("C:/demo.xml")
> x$DEMO
```

	id	name	sex
1	111	ABC	Female
2	222	DEF	Male
3	333	GHI	Female
4	444	JKL	Male

XMLだとSAS V6の
制約を受けない!
(例:変数名は8文字迄)

	sex	height	weight
1	F	160	50
2	F	165	65
3	M	170	60
4	M	175	55
5	M	180	70



★ <寄り道> R ⇒ XML ⇒ R

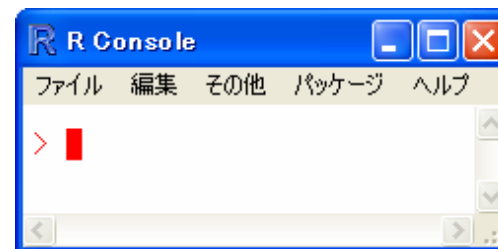
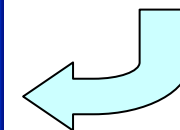
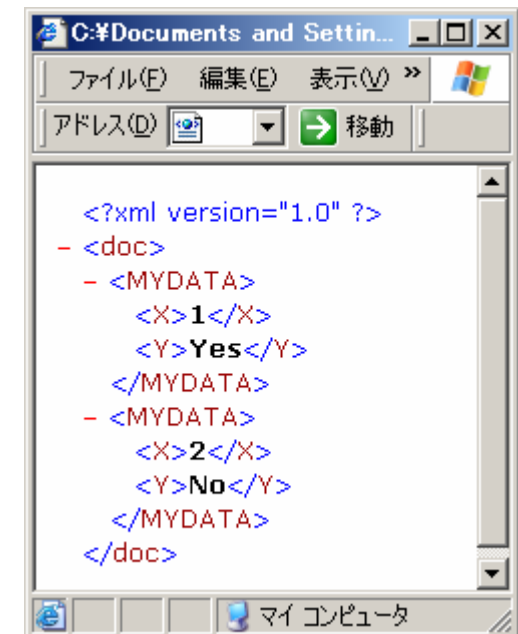
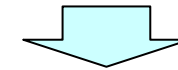
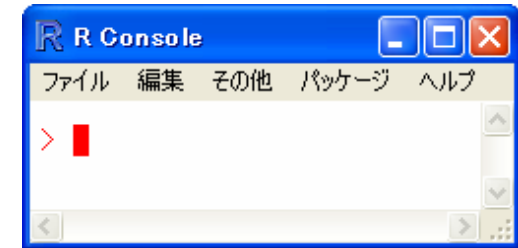


```
> ### R XML へ出力 (要パッケージ XML)
> tmp <- xmlOutputDOM()
> tmp$addTag("MYDATA", close=F)
> tmp$addTag("X", 1); tmp$addTag("Y", "Yes")
> tmp$closeTag() # MYDATA
```

```
> tmp$addTag("MYDATA", close=F)
> tmp$addTag("X", 2); tmp$addTag("Y", "No")
> tmp$closeTag() # MYDATA
> saveXML(tmp$value(), file="C:/mydata.xml")
```

```
### XML R へ入力 (要パッケージ SASXML)
> x <- sasXMLDataSet("C:/mydata.xml")
> x$data$MYDATA
```

```
   X   Y
1  1  Yes
2  2  No
```



参考文献 & 謝辞

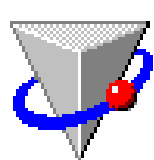


■ 参考文献

- THE R BOOK 4章 (岡田昌史 他; 九天社)
- データ解析環境 R 5章 (舟尾, 高浪; 工学社)
- パッケージ「RODBC」「foreign」のヘルプ

■ 発表資料作成の際にお世話になった人

- 北西 由武さん (塩野義製薬)
- 高浪 洋平さん (武田薬品工業)



で  を操る方法

～ SAS で R の出力を～

終

で を操る方法

～ データフレームに関するおまけ ～

★ データへのアクセス方法(1)



コマンド	機能
<code>x\$列名, x["列名"], x[["列名"]]</code>	指定した列データを表示
<code>x[2], x[[2]]</code>	2 番目の列データを表示
<code>x[3, 2], x[[3, 2]]</code>	3 行 2 列目のデータを表示
<code>x[[3,"列名"]], x[[3,"列名"]]</code>	指定した列の 3 行目のデータを表示
<code>x[c(1, 2)]</code>	1 列目と 2 列目のデータを表示
<code>x[c(3, 4),]</code>	3 行目と 4 行目のデータを表示
<code>x[,c(T,F,T)]</code>	論理ベクトル <code>c(T,F,T)</code> が TRUE となっている列を表示
<code>x[SEX=="F",]</code>	性別が F (女性) である行を表示
<code>x[,SEX=="F" & WEIGHT>50]</code>	性別が F (女性) かつ体重が 50kg より大きい行を表示

★ データへのアクセス方法(2)



■ データフレーム[行番号, 列番号] で指定する

`x[c(1,3,5),]`

1,3,5行目にアクセス

```
sex height weight
1  F    158    51
3  M    177    72
5  M    166    64
```

`x[,c(1,3)]`

1,3列目にアクセス

```
sex weight
1  F     51
2  F     55
3  M     72
4  M     57
5  M     64
```

データフレーム x

SEX	HEIGHT	WEIGHT
F	158	51
F	162	55
M	177	72
M	173	57
M	166	64

★ データへのアクセス方法(3)



■ データフレーム\$列名 で指定する

```
> x$height # 身長データ
```

```
[1] 158 162 177 173 166
```

```
> x$height <- NULL # 身長を削除
```

```
> x
```

```
  sex weight
1   F     51
2   F     55
3   M     72
4   M     57
5   M     64
```

データフレーム x

SEX	HEIGHT	WEIGHT
F	158	51
F	162	55
M	177	72
M	173	57
M	166	64

★ データの加工・抽出(1)



コマンド	機能
<code>ncol(x)</code>	<code>x</code> の列数 (変数の数) を求める
<code>nrow(x)</code>	<code>x</code> の行数 (データ数) を求める
<code>names(x)</code>	<code>x</code> の列名を表示する
<code>rbind(x,y)</code>	<code>x</code> と <code>y</code> を縦に並べて結合する
<code>cbind(x,y)</code>	<code>x</code> と <code>y</code> を横に並べて結合する
<code>data.frame(x,y)</code>	<code>x</code> と <code>y</code> を横に並べて結合する
<code>merge(x,y)</code>	<code>x</code> と <code>y</code> を併合 (マージ) する。 ※通常は引数に <code>all=T</code> を指定し、データを全て残す。 <code>all=T</code> を指定しなければデータの共通部分が結果として返される。

★ データの加工・抽出(2)



コマンド	機能
<code>head(x, n=a)</code>	先頭から a 行だけ抽出する
<code>tail(x, n=b)</code>	末尾から b 行だけ抽出する
<code>na.omit(x)</code>	NA を含む行を削除する
<code>transform(x, y=ベクトル)</code>	データフレーム x に新たな列 y を追加する
<code>subset(x, 条件式)</code>	条件式に合う行のみを抽出する
<code>subset(x, 条件式, ベクトル)</code>	ベクトルで指定した列に対し, 条件式に合う行のみを抽出する