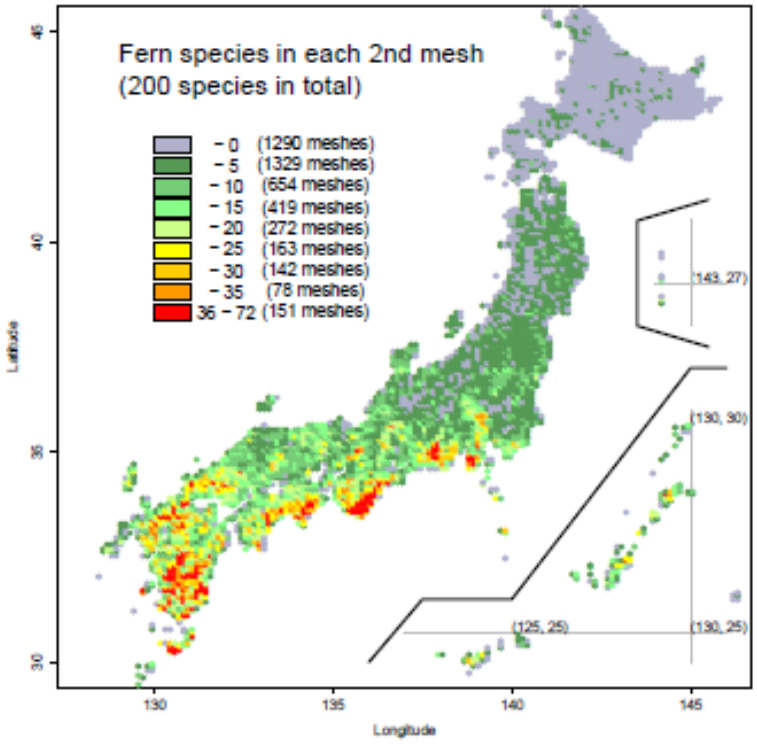
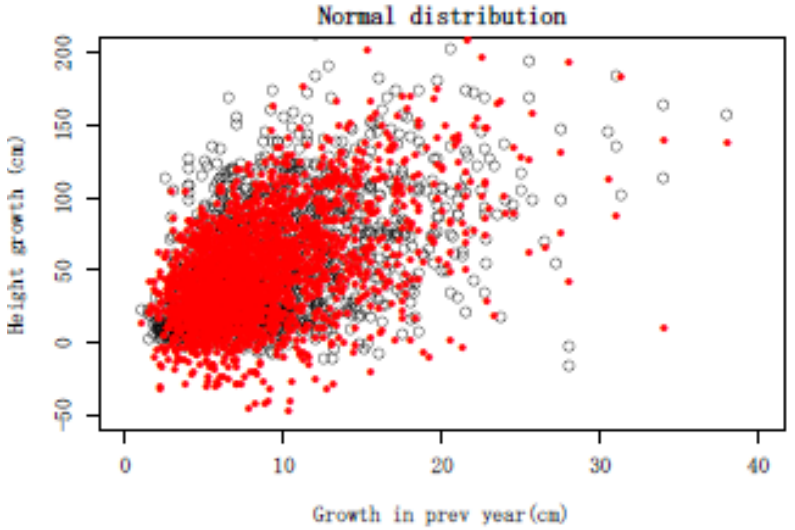
R で自動作図

どんな絵でも，何十回でも，何百枚でも

竹中明夫
（国立環境研）





プログラミング環境
＋実行環境

統計解析関数群

描画関数群

プログラム作図のすすめ

まずは描いてみる

高水準作図関数と低水準作図関数

繰り返し作業をプログラムで

画像の保存

この話の目指すところ

× Rのプログラムで絵が描けるようになる

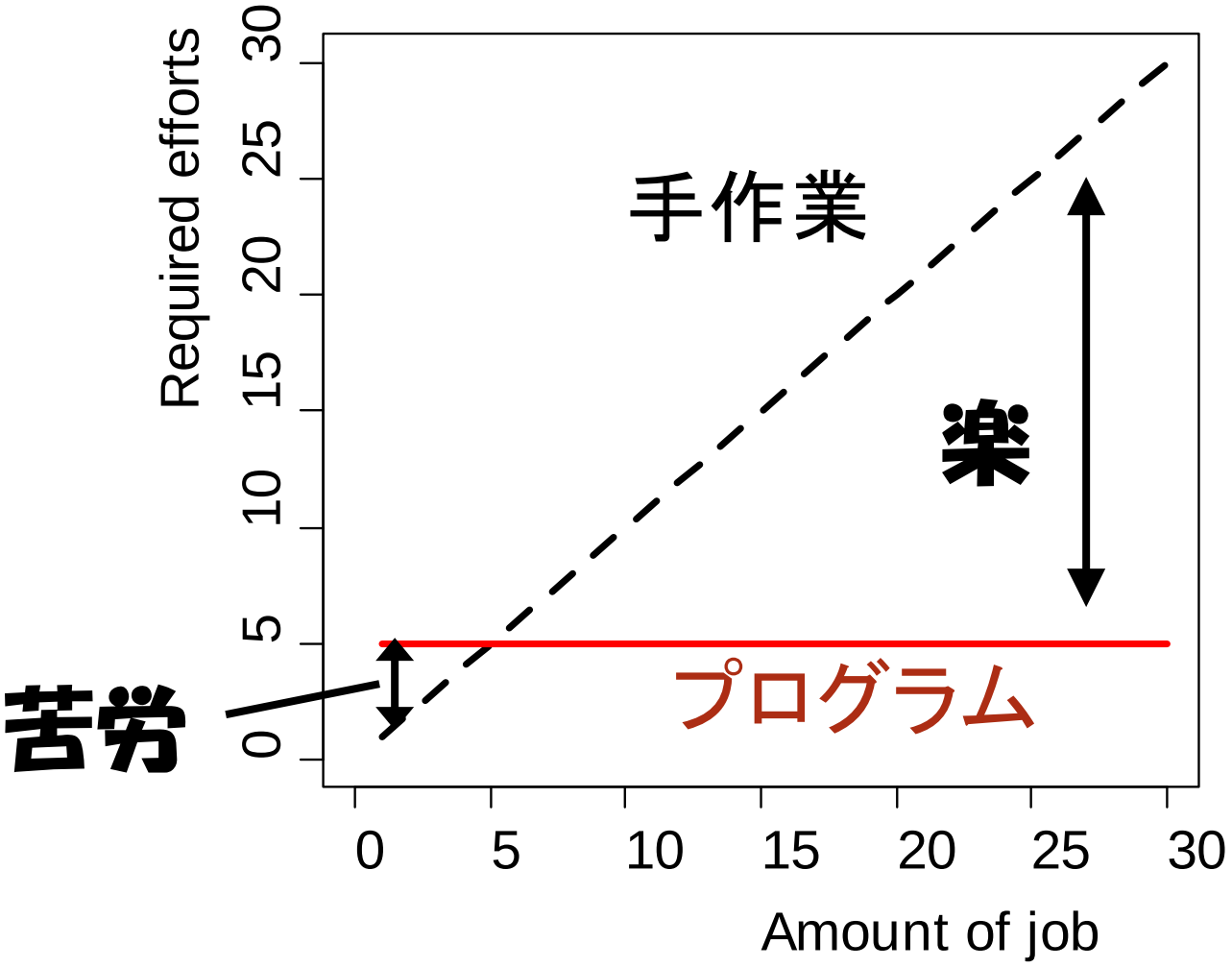
○ Rのプログラムで絵が描けそうな気になる

◎ Rのプログラムで絵を描きたくなる

やる気にさえなれば，世の中にはさまざまな情報がある．

プログラミングの心構え

楽をするための苦労は惜しまない



作図プログラムを書いてしまえば…

- 多量のデータでも片っ端から同じ形式でグラフ化できる
- デザインをちょっと変えての書き直しも簡単
- データにまちがいが見つかってもめげずに描き直せる

片っ端からのグラフ化が最初の一歩

	w	x	y	z
1	2.13	7.89	10.26	8.01
2	1.24	3.57	6.67	3.03
3	2.35	3.86	2.09	8.07
4	3.77	5.33	6.85	3.71
5	3.68	5.37	5.81	2.65
6	2.27	3.05	3.73	2.38
7	2.27	1.99	2.17	2.92
8	1.74	3.78	5.28	8.99
9	1.00	6.53	9.53	2.09

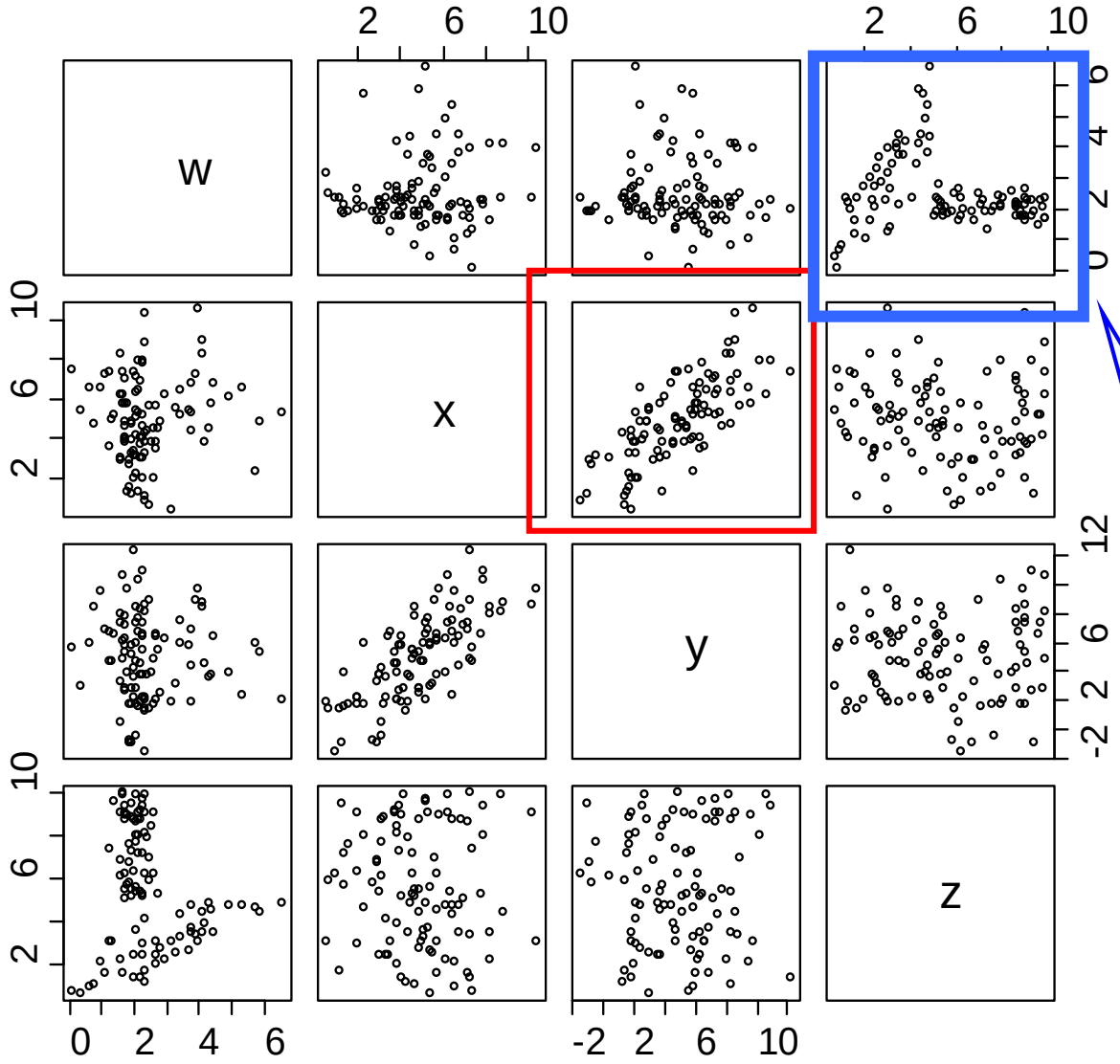
97	4.43	6.83	6.38	3.49
98	2.01	3.16	4.18	8.68
99	1.42	5.18	6.49	9.58
100	4.32	4.46	3.61	4.84

相関係数

	w	x	y	z
w	1.000	0.085	−0.034	−0.138
x	0.085	1.000	0.736	−0.064
y	−0.034	0.736	1.000	−0.008
z	−0.138	−0.064	−0.008	1.000

強い正の相関
をひとつ発見

グラフにしてみると…



ここにもあやしげなパターンが！

作図プログラムを書いて
しまえば

あー直せ，こー直せと言われても大丈夫

- 多量のデータでも片っ端から同じ形式でグラフ化できる
- デザインをちょっと変えての書き直しも簡単
- データにまちがいが見つかってもめげずに描き直せる

作図プログラムを書いてしまえば

締め切り間際にはとくに重要

- 多量のデータでも片っ端から同じ形式でグラフ化できる
- デザインをちょっと変えての書き直しも簡単
- データにまちがいが見つかってもめげずに描き直せる

プログラム作図のすすめ

まずは描いてみる

高水準作図関数と低水準作図関数

繰り返し作業をプログラムで

画像の保存

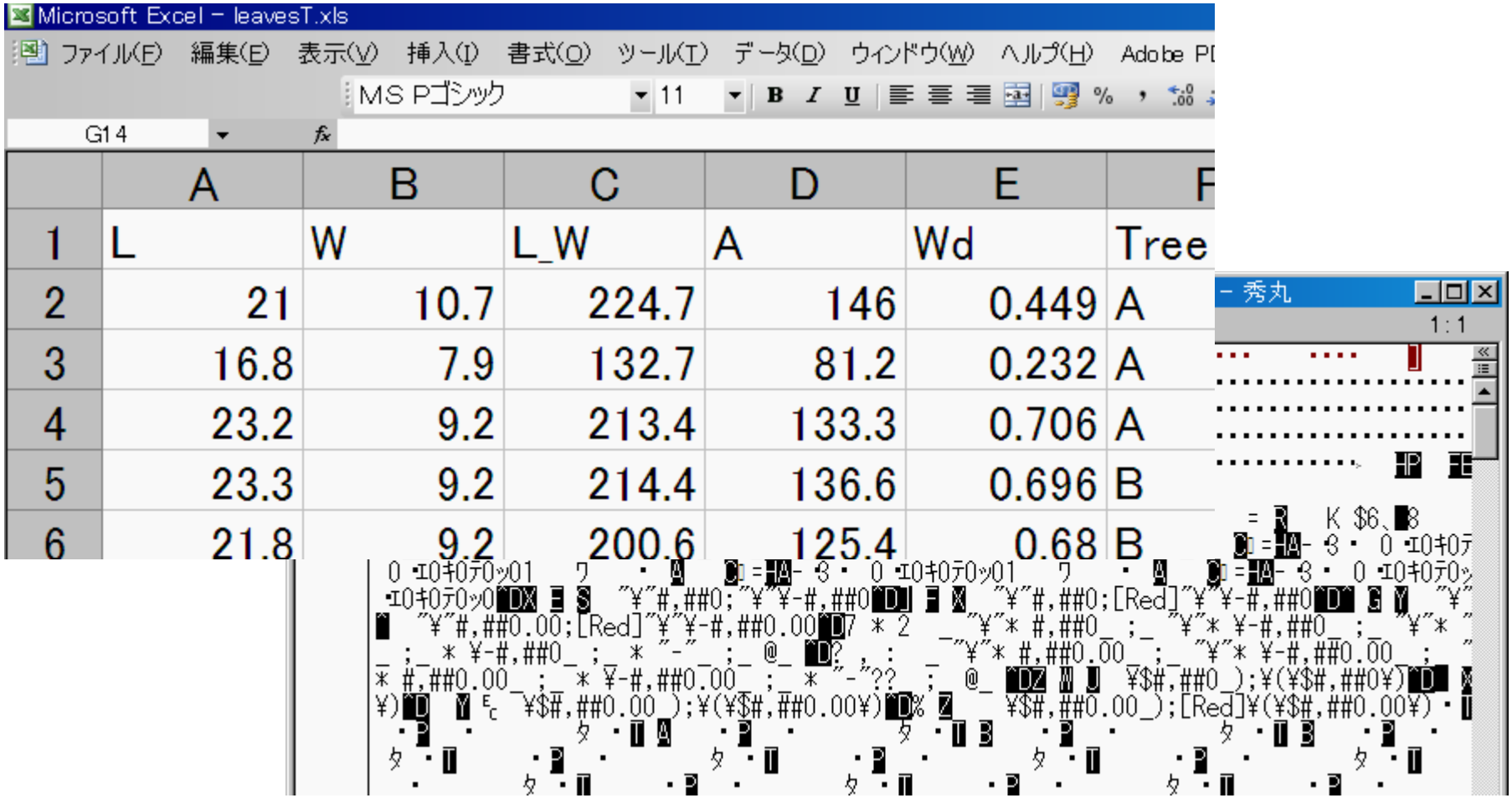
「Rで自動作図」に必要な準備

- R のインストール
- 作業用ディレクトリの作成
- 自分のデータ
- テキストエディタ

タブかカンマ区切りのテキストファイル

Rに組込まれたエディタもあり(スクリプトの編集…). 私はシェアウエアの秀丸を愛用. サクラエディタ, K2Editorなど無料のものも多数. これらはプログラミング支援機能も充実)

Excelのファイルは，データそのものの他に，さまざまな情報が Excel だけが知っている方法で書き込まれている．



Microsoft Excel - leavesT.xls

ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ツール(T) データ(D) ウィンドウ(W) ヘルプ(H) Adobe PD

MS Pゴシック 11

G14

	A	B	C	D	E	F
1	L	W	L_W	A	Wd	Tree
2	21	10.7	224.7	146	0.449	A
3	16.8	7.9	132.7	81.2	0.232	A
4	23.2	9.2	213.4	133.3	0.706	A
5	23.3	9.2	214.4	136.6	0.696	B
6	21.8	9.2	200.6	125.4	0.68	B

- 秀丸

1:1

テキストファイルは，文字で書かれた（人間がそのまま読める）情報のみが入っている


D:¥DeskTopFolder¥学会セミナー発表¥学会発表¥2008¥自由集会¥demo¥leaves.txt（上書き禁止...

ファイル(F) 編集(E) 表示(V) 検索(S) ウィンドウ(W) マクロ(M) その他(O) 1:1

	L	W	L_W	A	Wd	Tree
2	21	10.7	224.7	146	0.449	A
3	16.8	7.9	132.7	81.2	0.232	A
4	23.2	9.2	213.4	133.3	0.706	A
5	23.3	9.2	214.4	136.6	0.696	B
6	21.8	9.2	200.6	125.4	0.68	B
7	10	4	40	23.9	0.064	C
8	21.3	8.4	178.9	109.3	0.521	C
9	10.8	3.3	35.6	18.3	0.049	A
10	12.1	5.3	64.1	33.7	0.11	B
11	17.3	6.7	115.9	70	0.17	C
12	12.5	6.6	82.5	54.7	0.132	C
13	11.3	4.9	55.4	34.6	0.118	A
14	49.5	22.8	1128.6	747.3	3.887	A
15	34.8	13.2	459.4	275.3	1.507	B
16	37.3	15.3	570.7	327.8	1.473	C

Excel で，テキストファイル形式で保存（一度に一枚のシートのみ）

（メニュー）**ファイル**

→「**名前を付けて保存**」

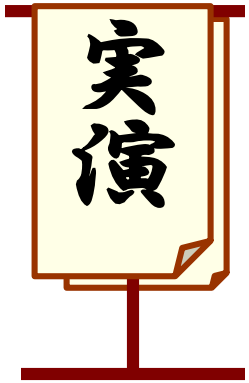
・（ファイルの種類）

テキスト（タブ区切り）（*.txt）

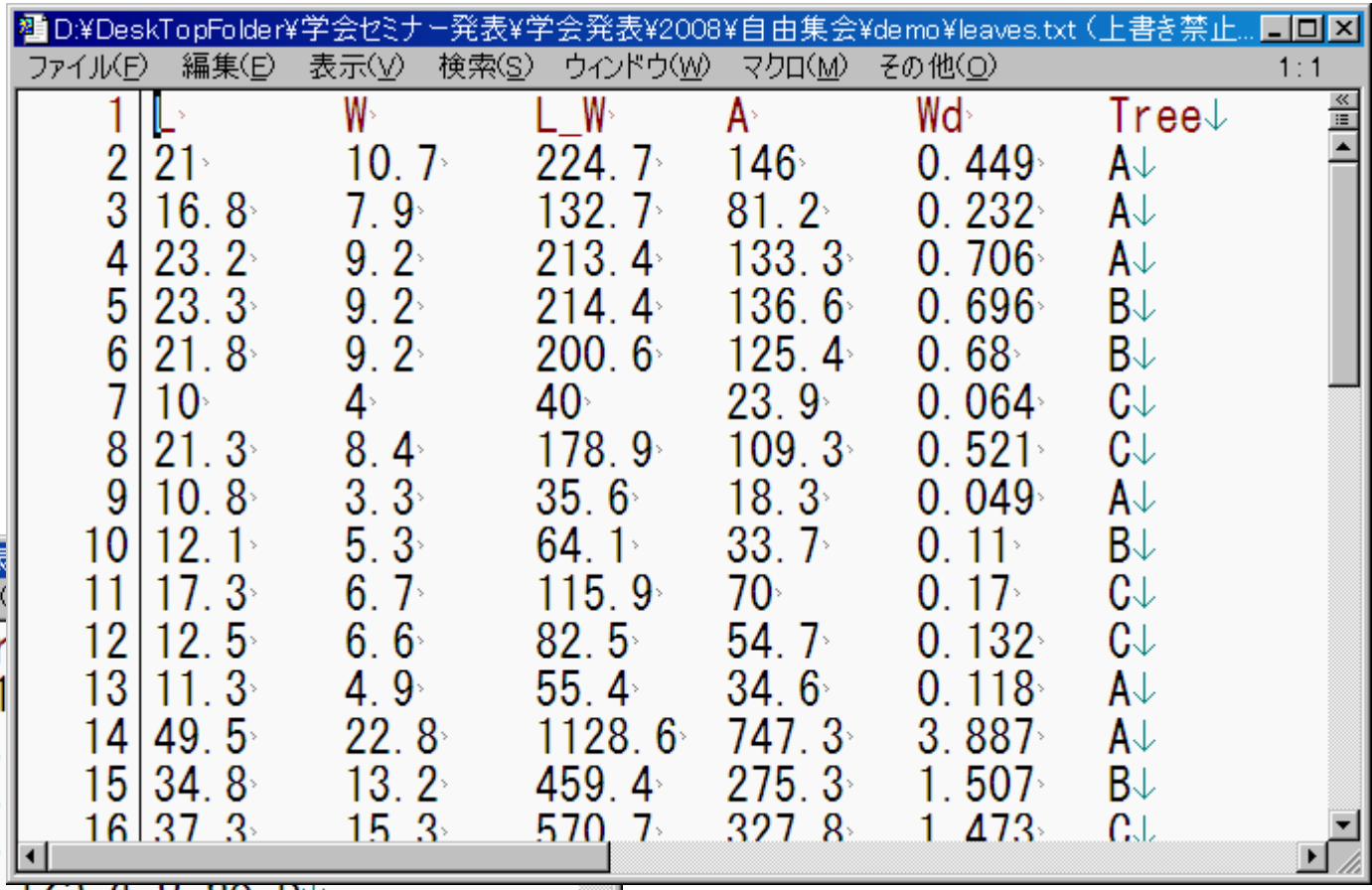
ないしは

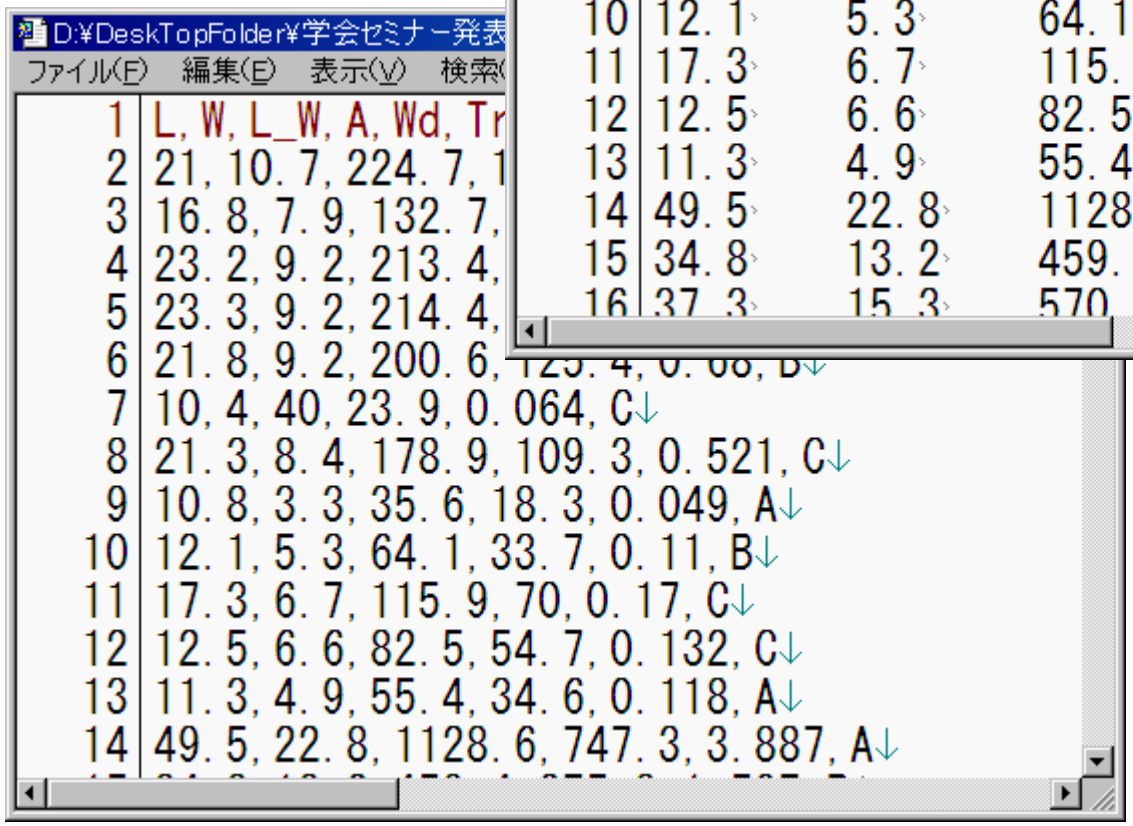
CSV（カンマ区切り）（*.csv）

どちらもテキストファイル



タブ区切り





カンマ区切り (CSV)

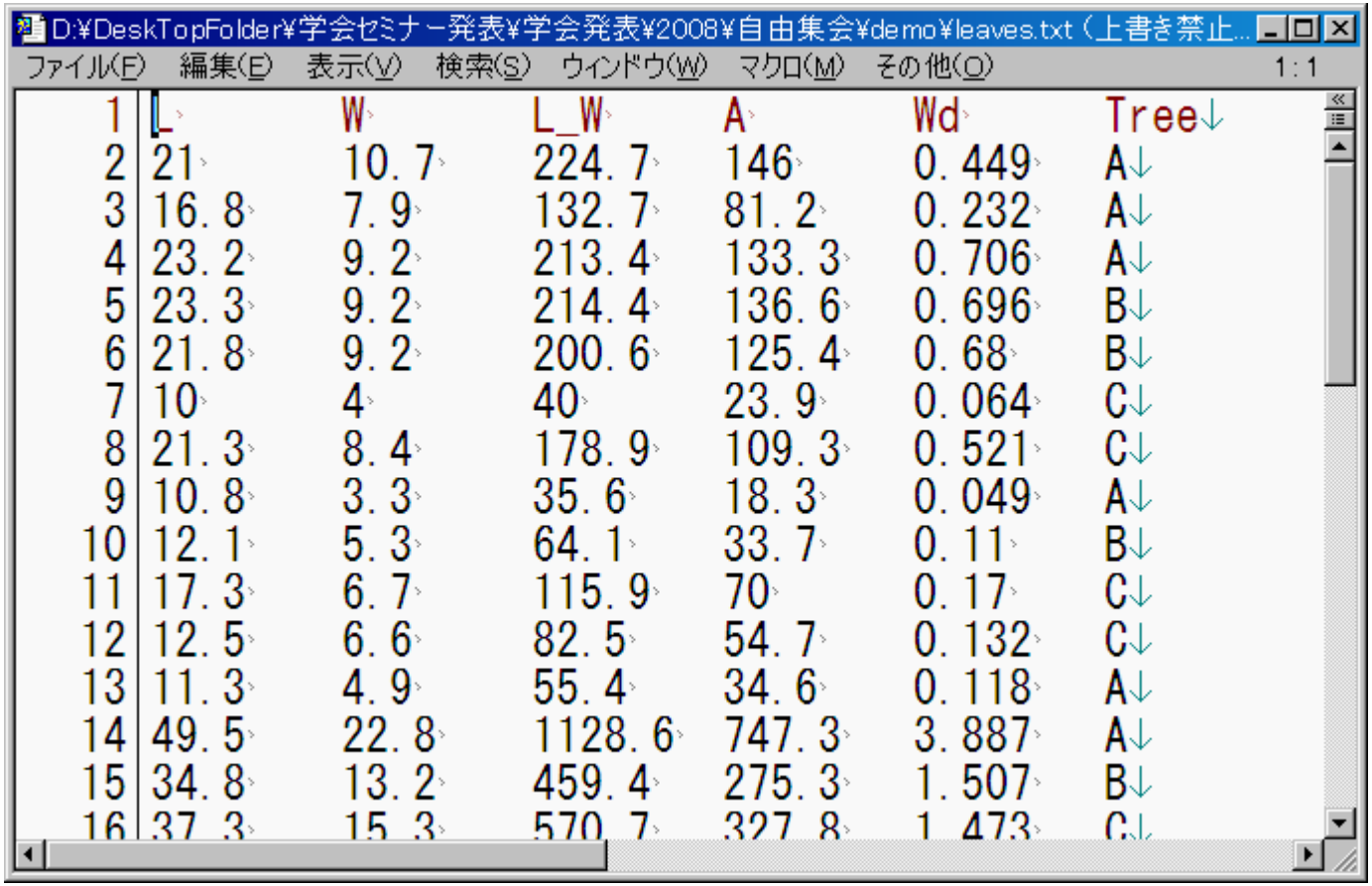
二通りの描き方

- Rの入出力画面で描画関数を呼び出してグラフを描く
- 描画に必要な命令を書き込んだファイルを作り，これを呼び出してグラフを描く

まずは一枚描いてみる

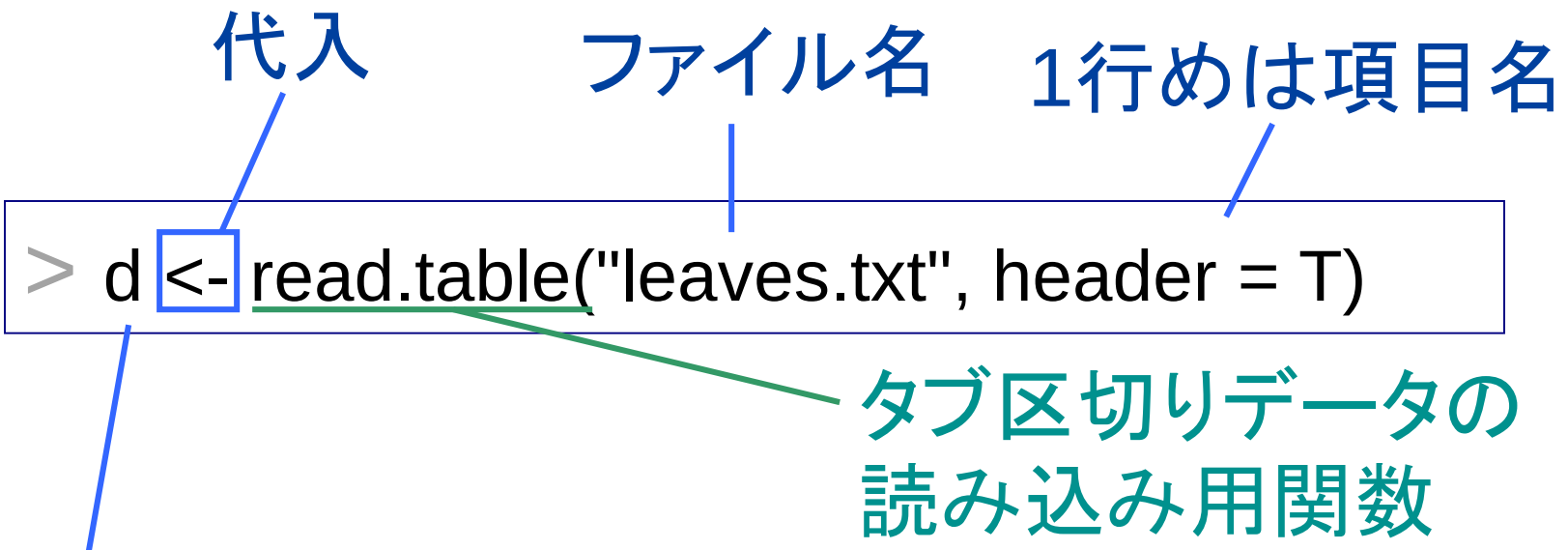
leaves.txt

葉の長さ，幅，面積，乾燥重量，個体



	L	W	L_W	A	Wd	Tree
2	21	10.7	224.7	146	0.449	A
3	16.8	7.9	132.7	81.2	0.232	A
4	23.2	9.2	213.4	133.3	0.706	A
5	23.3	9.2	214.4	136.6	0.696	B
6	21.8	9.2	200.6	125.4	0.68	B
7	10	4	40	23.9	0.064	C
8	21.3	8.4	178.9	109.3	0.521	C
9	10.8	3.3	35.6	18.3	0.049	A
10	12.1	5.3	64.1	33.7	0.11	B
11	17.3	6.7	115.9	70	0.17	C
12	12.5	6.6	82.5	54.7	0.132	C
13	11.3	4.9	55.4	34.6	0.118	A
14	49.5	22.8	1128.6	747.3	3.887	A
15	34.8	13.2	459.4	275.3	1.507	B
16	37.3	15.3	570.7	327.8	1.473	C

Rの入出力画面でデータの読み込み



変数 d は，データの塊.

data.frameという「クラス」の「オブジェクト」

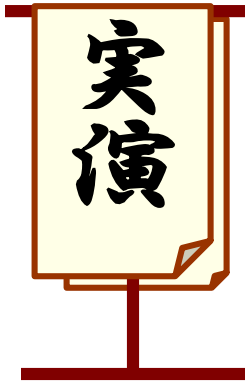
この扱いに慣れることがRの第一歩

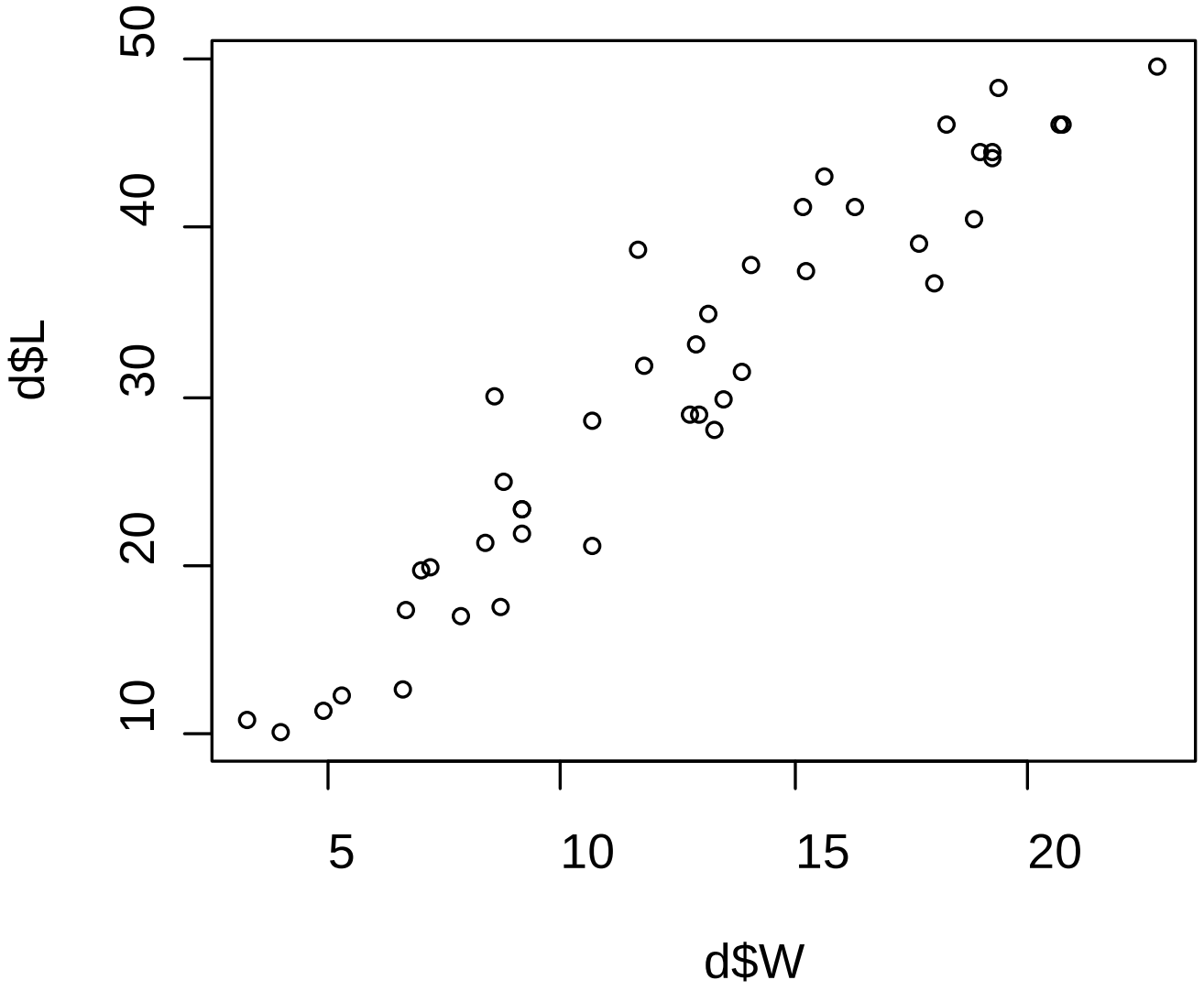
x と y を指定してグラフを描く

から後ろは無視．コメントを書くのに使う

```
> plot(d$W, d$L)  # 長さ vs 幅の撒布図
```

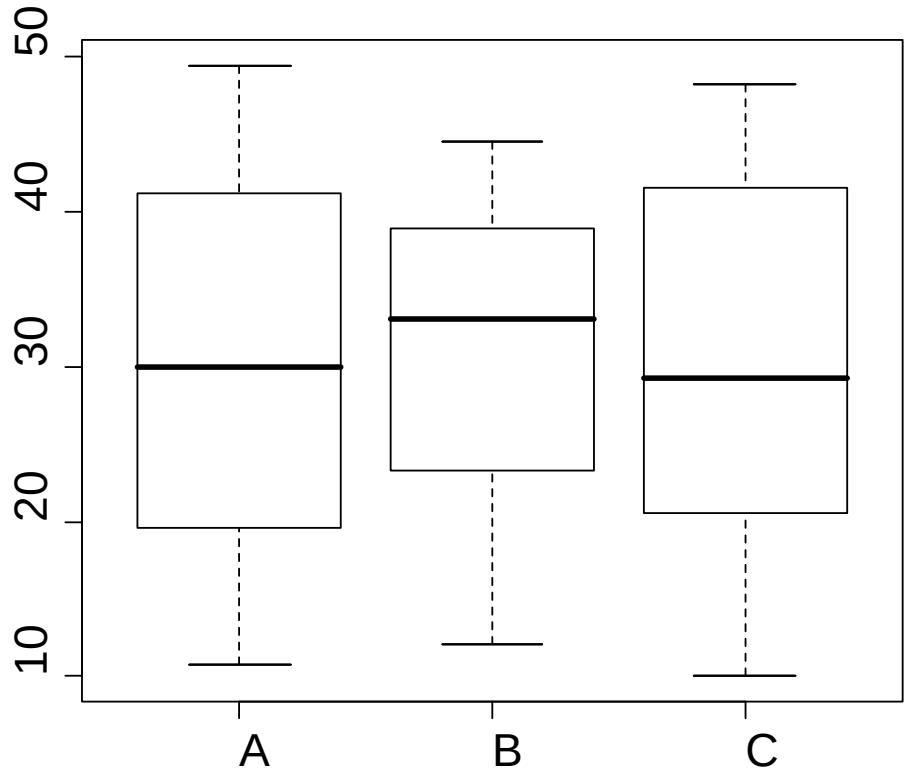
データフレーム d の，W という名前の列

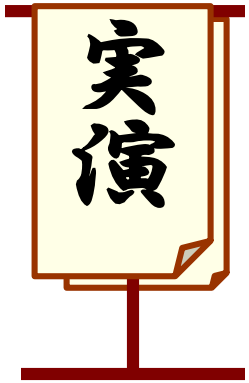



d$L
10
20
30
40
50
5
10
15
20
d$W

x と y を指定してグラフを描く

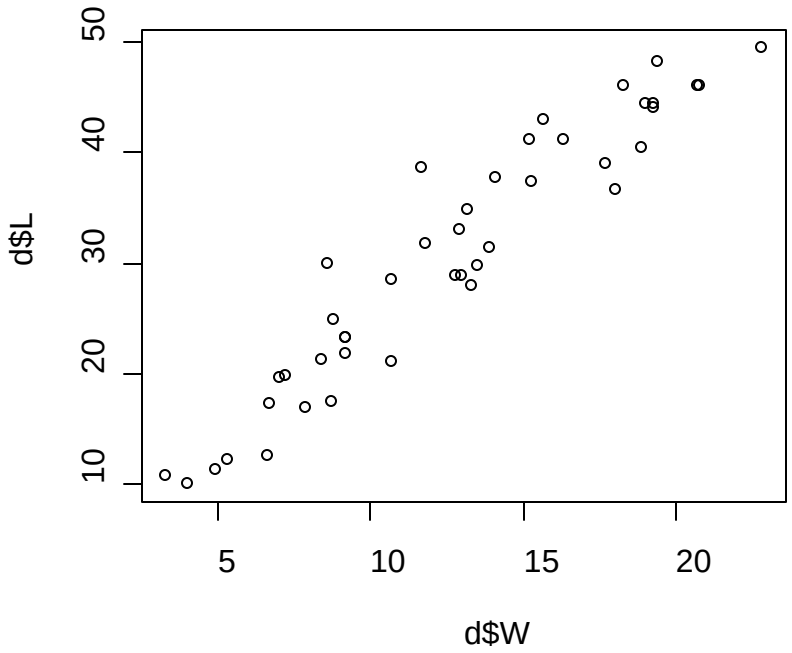
```
> plot(d$Tree, d$L)  # 木毎の葉の長さ
```

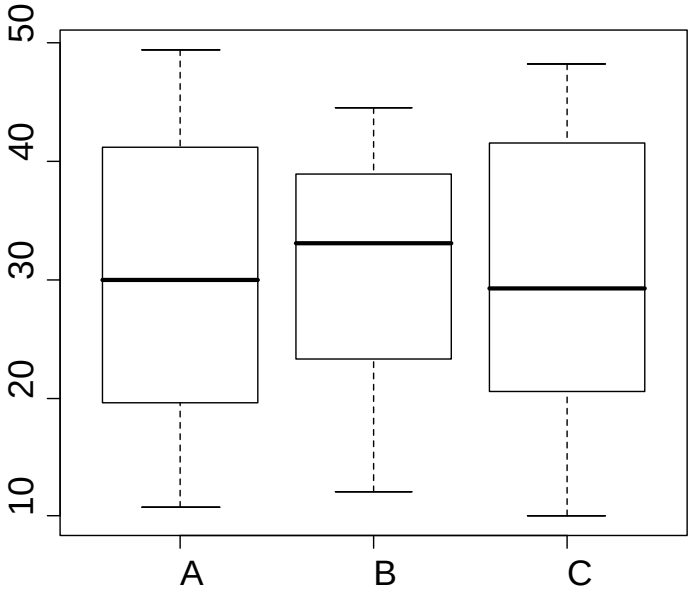




```
> plot(d$W, d$L)  # 長さ vs 幅の撒布図
```

```
> plot(d$Tree, d$L)  # 木毎の葉の長さ
```





plot()に渡すデータの型が違うと，描かれるグラフも違う.

データの型

- 実数
- 複素数
- 論理値
- 文字列
- （因子）
- その他のクラス

さらに…

- ベクトル（1次元）
- 行列（2次元）
- 配列（多次元）
- リスト（異種データを束ねたもの）

```
x1 <- 13
x2 <- c(1, 3, 5)
y <- c("A", "BB")
```

※数値ひとつも長さ1のベクトル

おいおい慣れます．

期待した動作をしないとき，型の問題を疑ってみよう．

関数 plot () に渡すデータの型が違うと，描かれるグラフも違う．

- 渡されたデータの型により，実際の作業を行う描画関数が選択される．
- 描画関数のなかで，データの型により違う動作が選択される．

↑とりあえずは気にしない． plot() **が魔法の呪文でないことだけ分かればＯＫ．**

- Rの入出力画面で描画関数を呼び出してグラフを描く
- 描画に必要な命令を書き込んだファイルを作り，これを呼び出してグラフを描く

組込みのスクリプト編集エディタないしはお好きなテキストエディタで

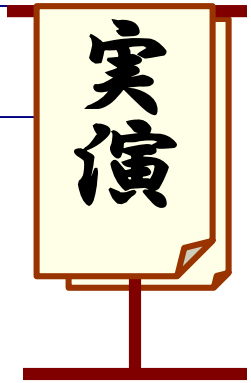
プログラムを書いたテキストファイルを用意し…

plot01.R　(テキストファイル)

```
d <-read.table("leaves.txt", header = T)
plot(d$W, d$L)
```

Rの入出力画面から呼び出す.

```
> source("plot01.R")
```



プログラム作図のすすめ

まずは描いてみる

高水準作図関数と低水準作図関数

繰り返し作業をプログラムで

画像の保存

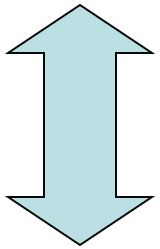
Ｒの作図関数には高水準と低水準がある

高水準作図関数：一枚の絵を作る．呼び出すと座標系が設定される．

低水準作図関数：高水準作図関数で描いた図に点や線や字を書き加える．既存の座標系に従う．

なんら自虐的あるいは侮蔑的な意味はこめられていない

高水準作図関数でお手軽に作図

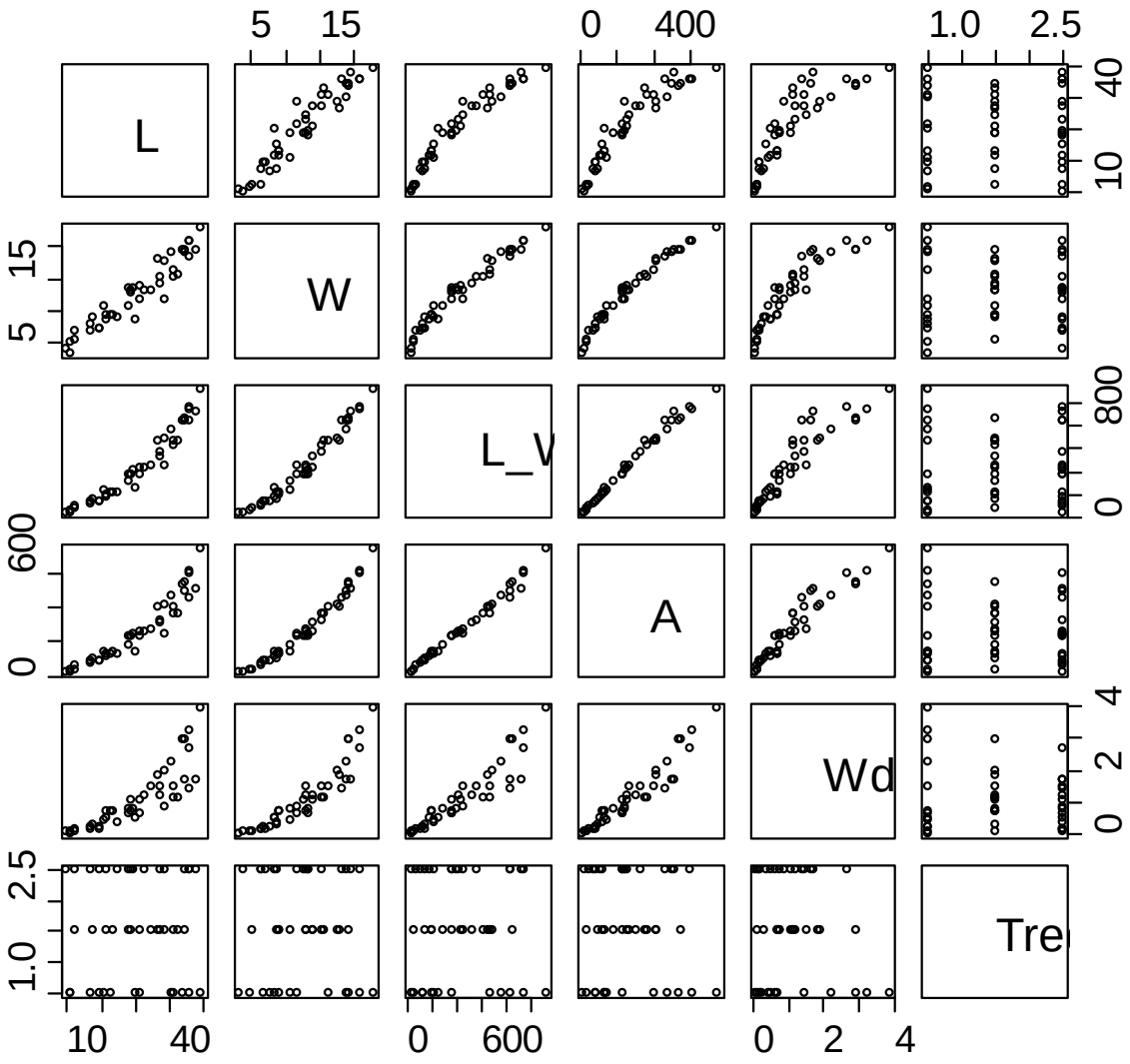


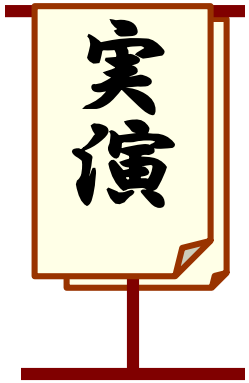
好きなように組み合わせて

低水準作図関数で自由に作図

```
d <-read.table(“leaves.txt”, header = T)
plot(d)
```

高水準





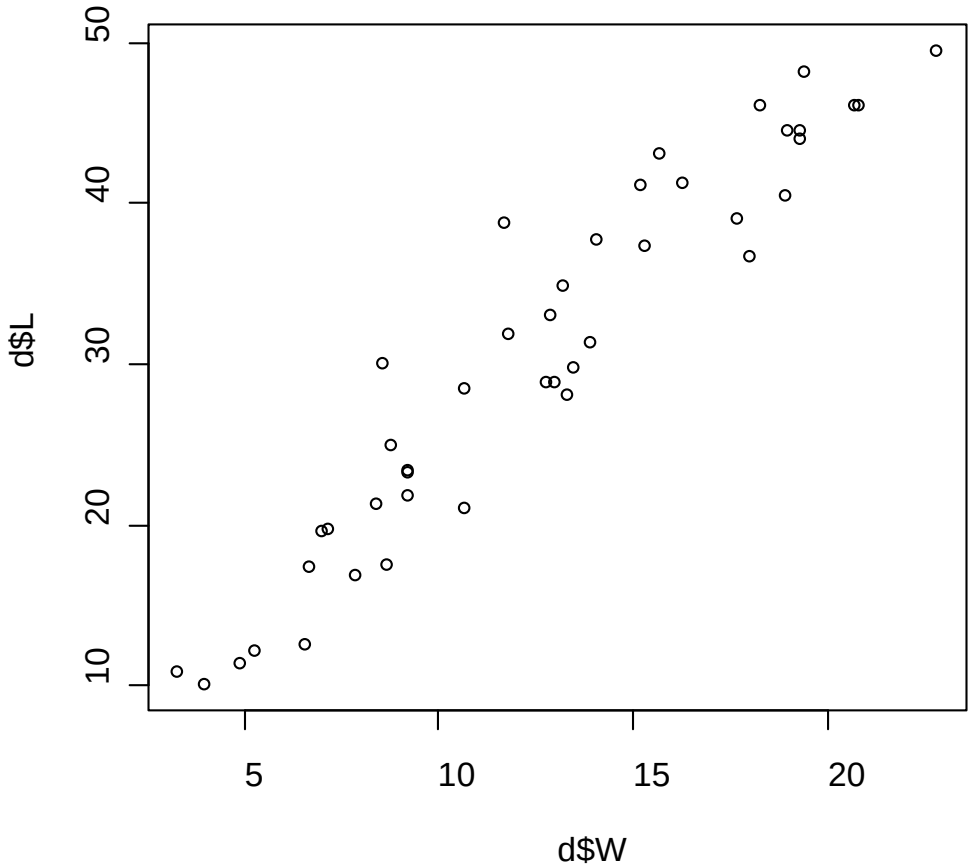
適当な高水準作図関数を使えば楽ができる.

＋　きめ細かい指定は

- 描画関数呼び出し時のオプション指定
- グローバルな描画パラメータの指定 (par)

```
plot(d$W, d$L)
```

高水準



そっけない…

描画関数でよく使うオプション指定の例

col　色

pch 点のタイプ

cex 図中の字や点の大きさ

lty　線のタイプ

lwd 線の太さ

xlim, ylim　軸の最小・最大

log　対数軸の指定

...

※詳しくはマニュアル・ヘルプなど参照

par() 関数で設定するオプションの例

mar, グラフの余白

mfcol, 画面の分割（1ページに複数の絵を描く）

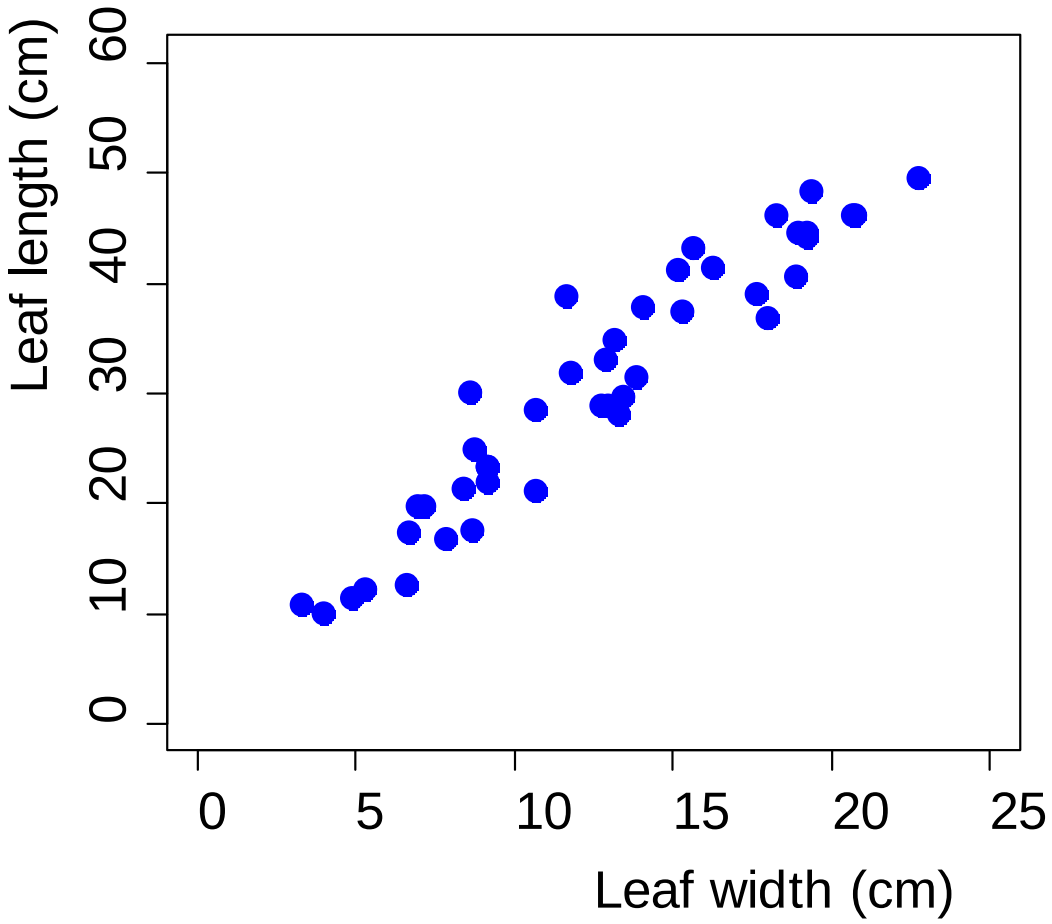
...

例:

```
par(mar = c(5, 6, 2, 2))
par(mfcol = c(2, 2))
```

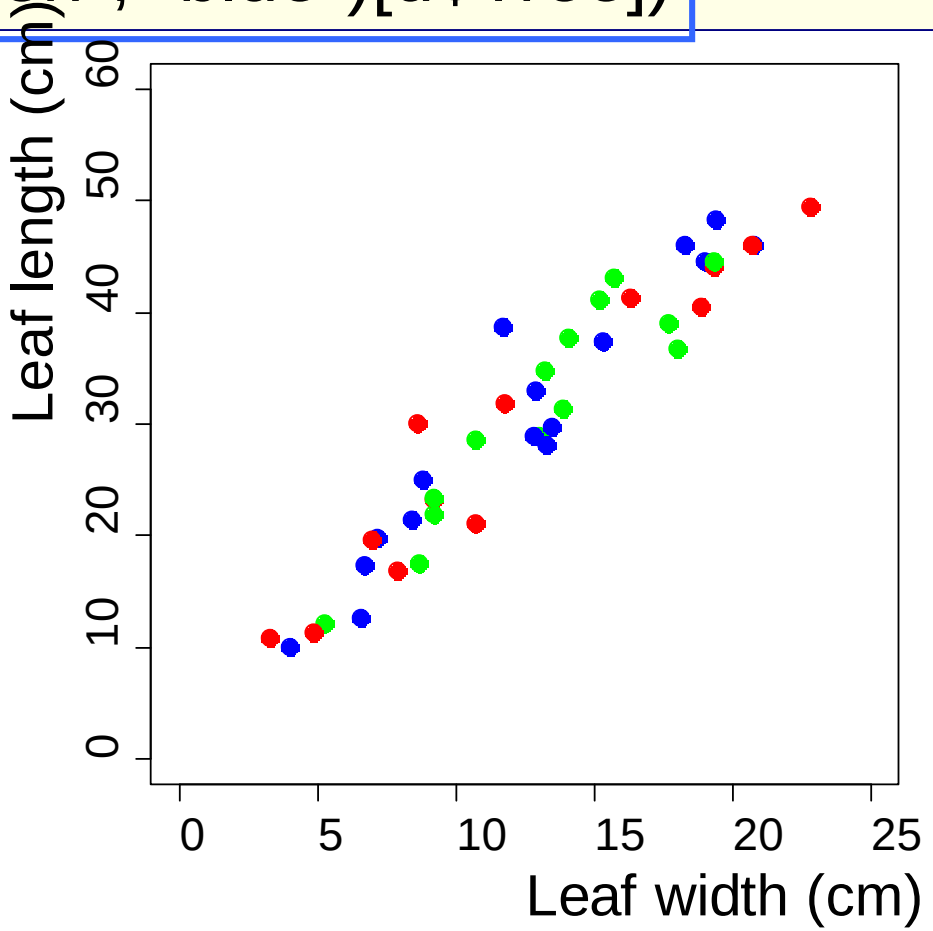
※詳しくはマニュアル・ヘルプなど参照

```
par(mar=c(5, 6, 2, 2))

plot(d$W, d$L, xlab = "Leaf width (cm)", ylab = "Leaf length (cm)", cex = 2, pch = 20, col = 4, cex.lab = 1.5, cex.axis = 1.5, xlim =c(0, 25), ylim = c(0, 60))
```



```
plot(d$W, d$L, xlab = "Leaf width (cm)",
 ylab = "Leaf length (cm)", cex = 2, pch = 20,
 xlim =c(0, 25), ylim = c(0, 60),
col = c("red", "green", "blue")[d$Tree])
```

これで，Treeの種類(A, B, C)別に色分け



ちょっと面倒ですが，一応ご説明．

```
> c("red", "green", "blue")[1]
[1] "red"
> c("red", "green", "blue")[c(1, 3, 2)]
[1] "red"   "blue"  "green"
> d$Tree
 [1] A A A B B C C A B C C A A B C C B B A A A B B
B C C C A B C C A B C C B C A A C
Levels: A B C
> as.numeric(d$Tree)
 [1] 1 1 1 2 2 3 3 1 2 3 3 1 1 2 3 3 2 2 1 1 1 2 2 2 3
3 3 1 2 3 3 1 2 3 3 2 3 1 1 3 3 2 2
```

3要素ベクトルの1番め

添字もベクトルに

d$Treeは因子

数値(ベクトル)として評価

すぐに納得できなくてもよいです．

```
> c("red", "green", "blue")[c(1, 3, 2)]
[1] "red"   "blue"  "green"
> as.numeric(d$Tree)
 [1] 1 1 1 2 2 3 3 1 2 3 3 1 1 2 3 3 2 2 1 1 1 2 2 2 3
3 3 1 2 3 3 1 2 3 3 2 3 1 1 3 3 2 2
> c("red", "green", "blue")[d$Tree]
 ″red″  ″red″  ″green″ ″green″ ″blue″  ″blue″  ″red″
″green″  ″blue″  ....
```

d$Treeは因子

ベクトルXの添字に因子ベクトルを書くと，各因子のレベルに対応した（ベクトルXの）要素が並んだベクトルが得られる．

```
plot(d$W, d$L, xlab = "Leaf width (cm)",
 ylab = "Leaf length (cm)", cex = 2, pch = 20,
 xlim =c(0, 25), ylim = c(0, 60),
col = c("red", "green", "blue")[d$Tree])
```

ベクトルの添字にベクトル

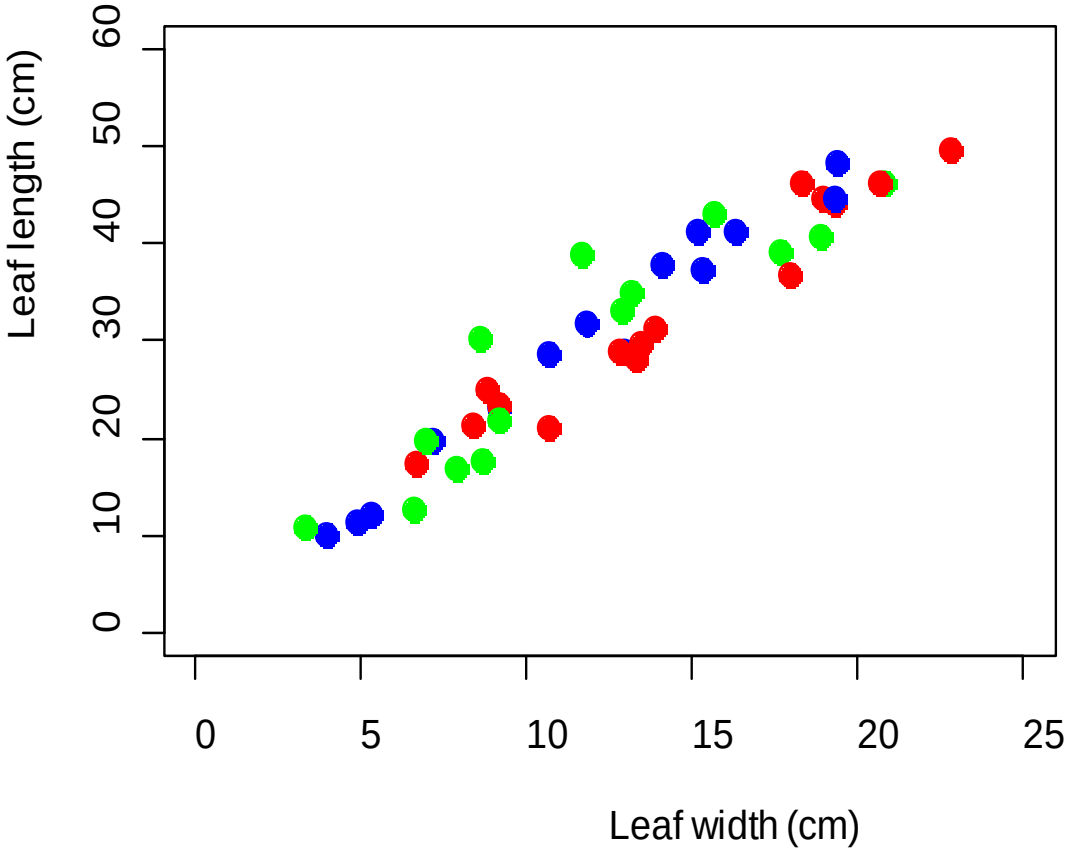
```
col = c("red", "red", "red", "green", "green", "blue",
"blue", "red","green" ...)
```

[c(1,1,1,2,2,3,3,3,1,2,..)]

それを添字に

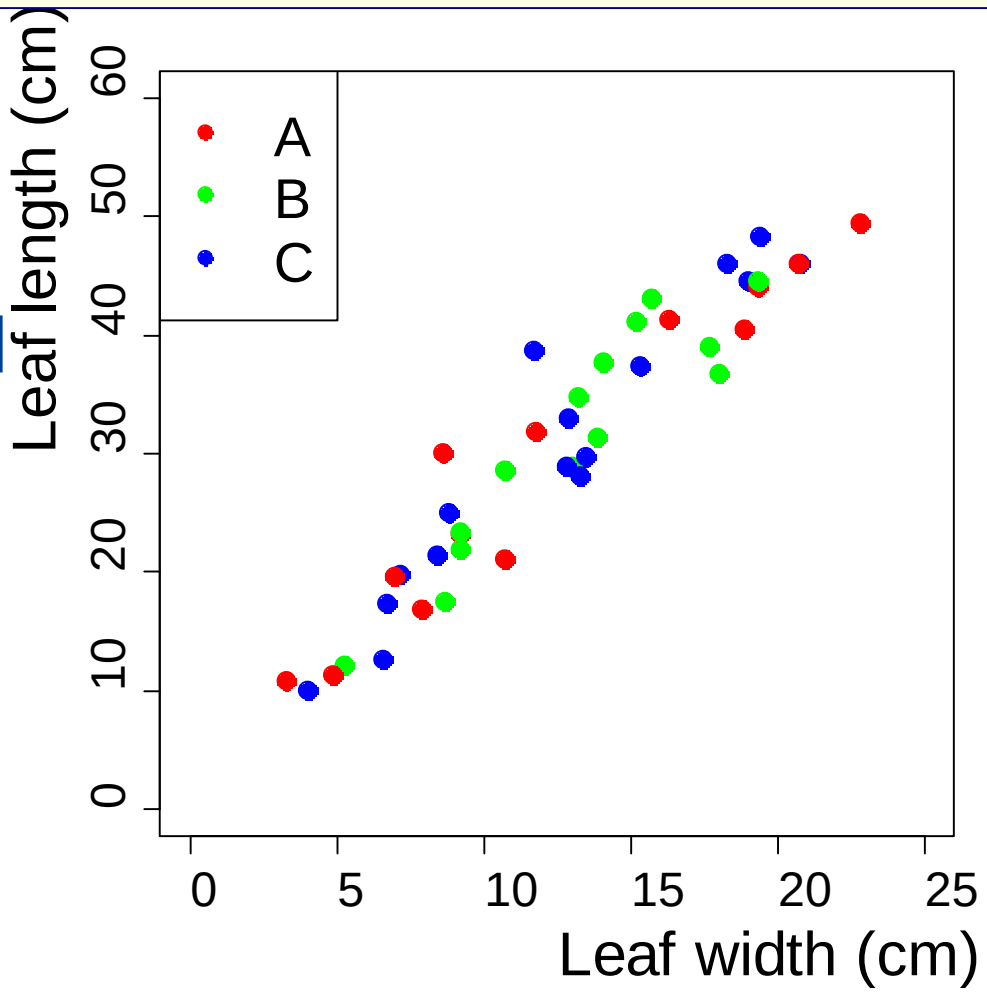
```
plot(d$W, d$L, xlab = "Leaf width (cm)",
 ylab = "Leaf length (cm)", cex = 2, pch = 20,
 xlim =c(0, 25), ylim = c(0, 60),
col = c("red", "green", "blue")[d$Tree])
```

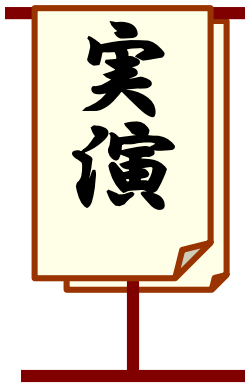
ベクトルの添字にベクトル



```
plot(d$W, d$L,... ,
    col = c("red", "green", "blue")[d$Tree])
legend("topleft", levels(d$Tree),
  col=c("red", "green", "blue"), pch = 20, cex = 1.8)
```

凡例を書き加える低水準作図関数



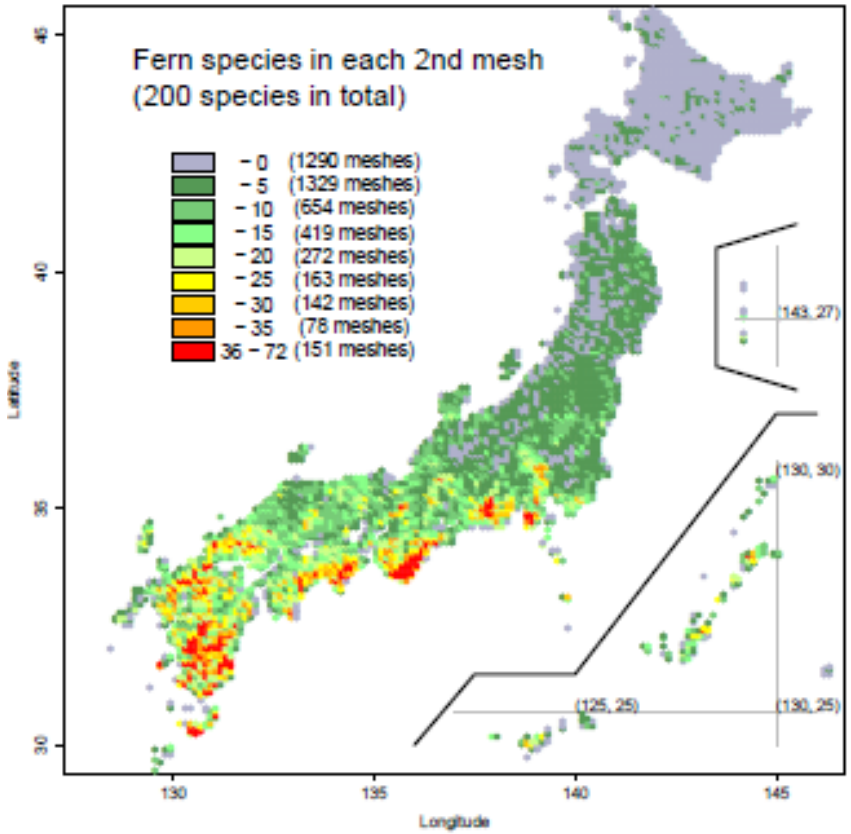


こんな定跡もあります

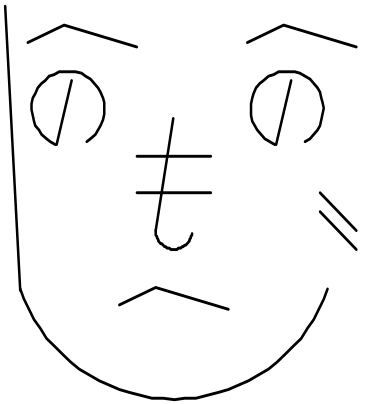
```
#  高水準で枠だけ描く(座標系が設定される)
plot(0, 0, type = 'n', xlim = c(0, 10),
      ylim = c(0, 100), xlab = "...", ylab = "...")


# そのあと，線を引いたり点を打ったり多角形を描いたり
文字を書いたり.
lines(x, y, …)
points(x, y, …)
polygon(x, y, ...)
text(x, y, "...")
```

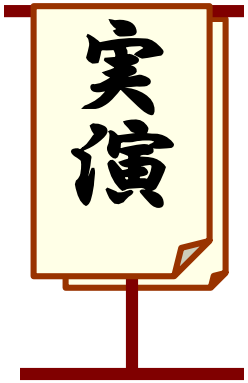
注意：高水準関数でグラフを描いたあと，続けて高水準関数を呼び出しても，普通は上書きできません（※curve()　など例外あり)

低水準作図関数を組み合わせれば（原理的には）なんでも書ける.



これも ***R*** で描きました





プログラム作図のすすめ

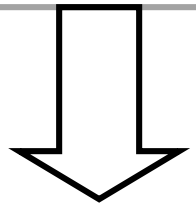
まずは描いてみる

高水準作図関数と低水準作図関数

繰り返し作業をプログラムで

画像の保存

- Rの入出力画面で描画関数を呼び出してグラフを描く
- 描画に必要な命令を書き込んだファイルを作り，これを呼び出してグラフを描く



プログラム中で作業の繰り返しを指示すれば，何枚もの絵を一度に描ける

今さら言うまでもありませんが,

単純作業の繰り返しはコンピューターがもっとも得意とするところ.

Rでの繰り返し構造の表現方法

for（変数名 in ベクトル） {処理}

while（条件）{処理}

repeat {処理}

（+ next, break）

※構造化定理

どんな計算手順も，「順次・反復・分岐」の組み合わせで記述できる

```
for (...){    # ファイルごとの繰り返し
    ファイル名を設定する.
    ファイルの内容を読み込む.
    読んだデータを使って絵を描く
}
```

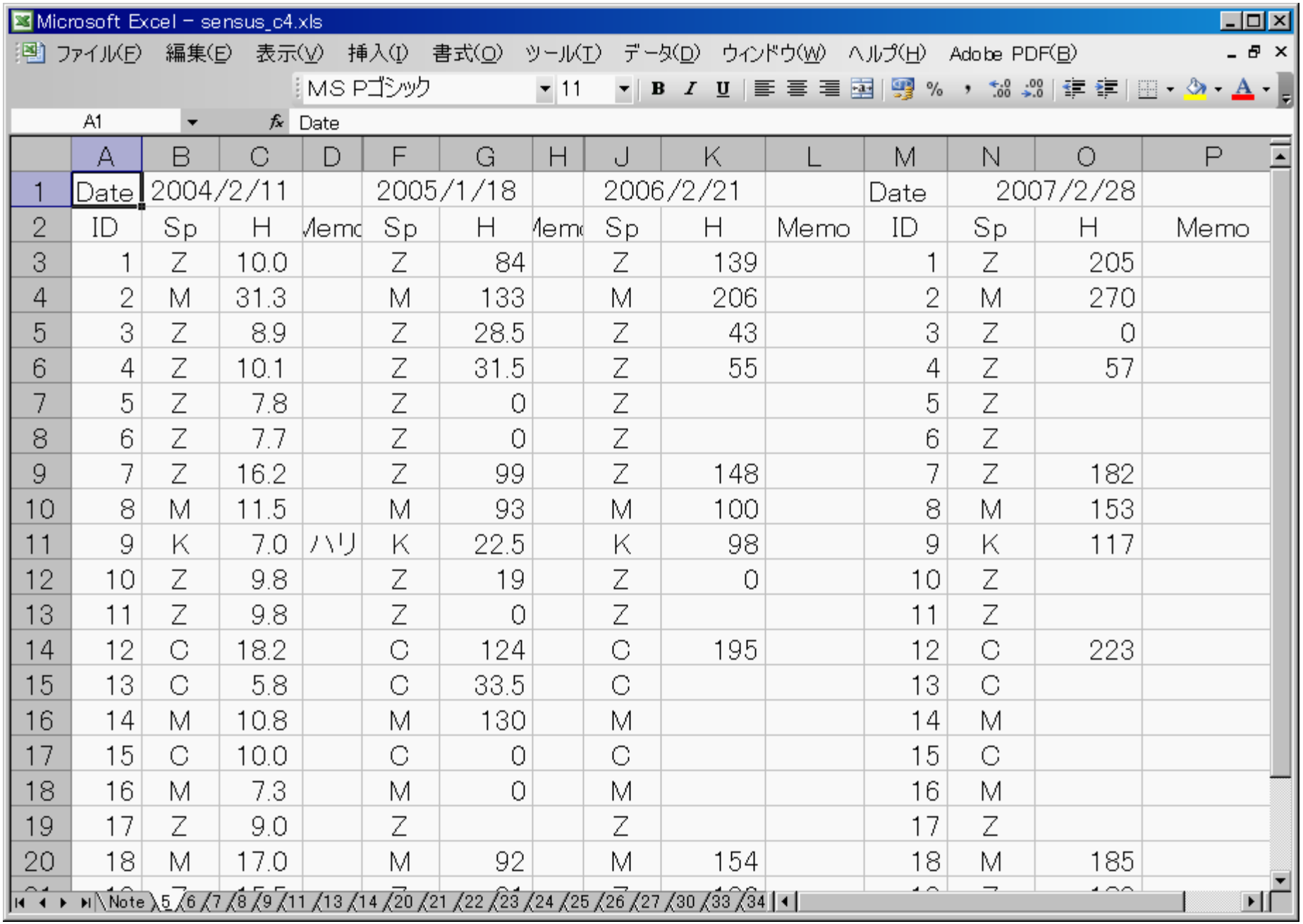
```
for (...){    # ファイルごとの繰り返し
    ファイル名を設定する.
    ファイルの内容を読み込む.
    読んだデータを使って絵を描く
}
```

- ファイル名をプログラム中に書き並べる.
- ファイル名を規則的に生成する
- ディレクトリ中のファイルを調べる
- ファイル名を書き並べたファイルを読む
- . . . .

```
sites <- c("A", "B", "Tenjin", "Hakata")  # 4つのサイト
n.block <- 100                            # 100 ブロック
n.pot <- 50                               # ポットが50個

#  20,000ポットそれぞれのデータファイルを読んで描画
for (site in sites) {
   for (i in 1:n.block) {
      for (j in 1:n.pot) {
         file <-  sprintf("%s_B%03d_P%03d.txt", site, i, j)
         d <- read.table (file, header = T)
          # 描画など…
      }
   }
}
```

たとえば…

90個の方形区内の木本の実生，約3,000個体余りの3年間の高さ成長のグラフ



	A	B	C	D	F	G	H	J	K	L	M	N	O	P
1	Date	2004/2/11			2005/1/18			2006/2/21			Date	2007/2/28		
2	ID	Sp	H	Memo	Sp	H	Memo	Sp	H	Memo	ID	Sp	H	Memo
3	1	Z	10.0		Z	84		Z	139		1	Z	205	
4	2	M	31.3		M	133		M	206		2	M	270	
5	3	Z	8.9		Z	28.5		Z	43		3	Z	0	
6	4	Z	10.1		Z	31.5		Z	55		4	Z	57	
7	5	Z	7.8		Z	0		Z			5	Z		
8	6	Z	7.7		Z	0		Z			6	Z		
9	7	Z	16.2		Z	99		Z	148		7	Z	182	
10	8	M	11.5		M	93		M	100		8	M	153	
11	9	K	7.0	ハリ	K	22.5		K	98		9	K	117	
12	10	Z	9.8		Z	19		Z	0		10	Z		
13	11	Z	9.8		Z	0		Z			11	Z		
14	12	C	18.2		C	124		C	195		12	C	223	
15	13	C	5.8		C	33.5		C			13	C		
16	14	M	10.8		M	130		M			14	M		
17	15	C	10.0		C	0		C			15	C		
18	16	M	7.3		M	0		M			16	M		
19	17	Z	9.0		Z			Z			17	Z		
20	18	M	17.0		M	92		M	154		18	M	185	

集めたデータはまずはグラフ
化して眺めるべきだが…

手作業で3000個体の成長
を90個のブロックごとにグ
ラフにする気になるか？

集めたデータはまずはグラフ化して眺めるべきだが…

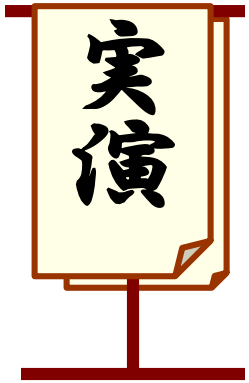
そんな簡単なことはコンピューターにだってできます.

手作業で3000個体の成長を90個のブロックごとにグラフにする気になるか？

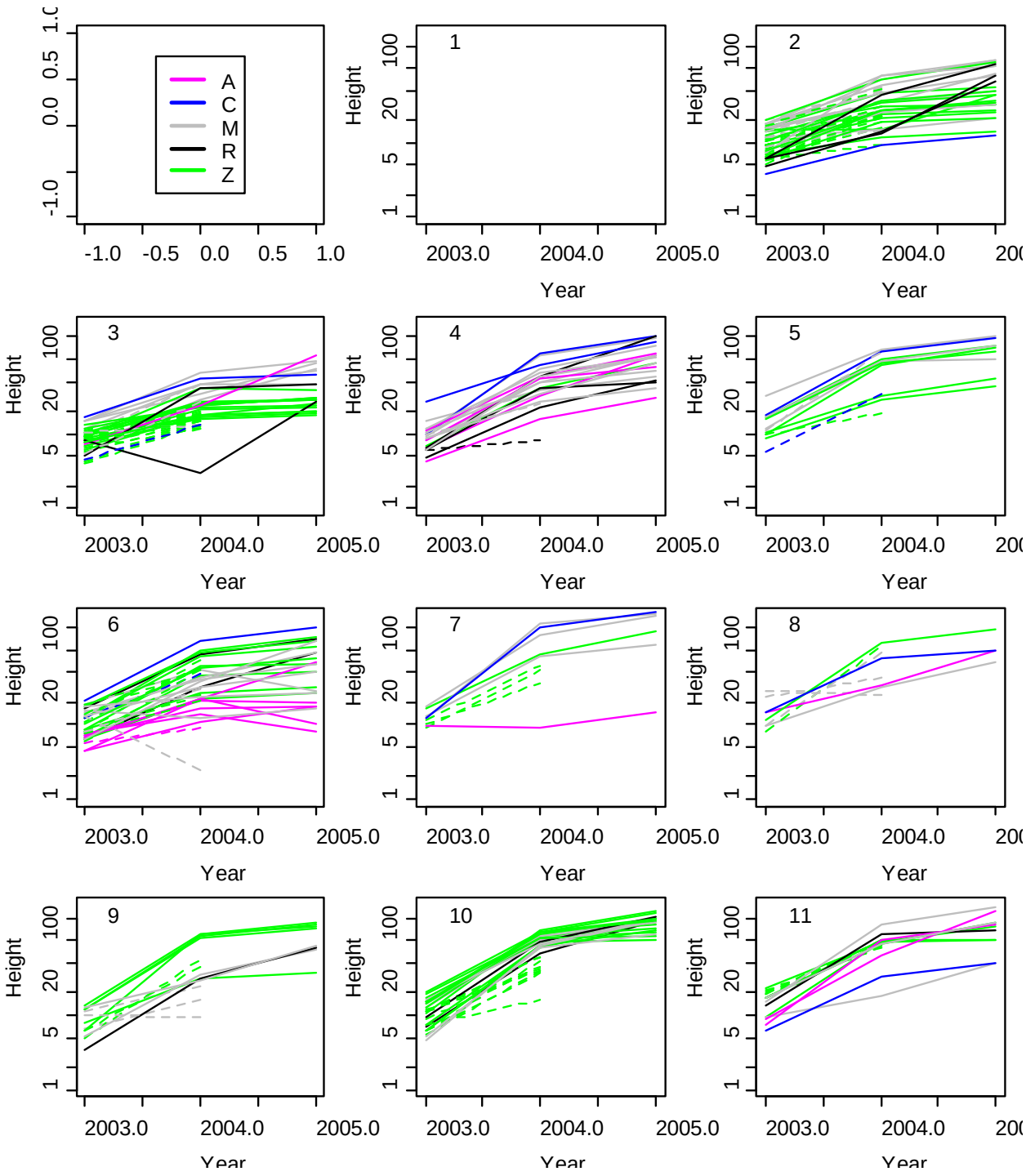
調査データの管理と図化：私の場合

- データを表計算ソフトで入力（久保さんには内緒）
- テキストファイルに変換
- 必要に応じてPerlで整形
- Rで作図

自作の道具も



par(ask = T)


A
C
M
R
Z
1
2
3
4
5
6
7
8
9
10
11
Height
Year
1
5
20
100
2003.0
2004.0
2005.0
-1.0
-0.5
0.0
0.5
1.0

```
#  ブロックごとの，全個体の高さ成長
#   2008-03-07  Rev. TAKENAKA, A.

# 1ページのグラフ数，余白などの設定
par(mfrow = c(4,3))
par(mar =c(4, 4, 1, 1) )
par(mex = 0.80)

# 種ごとの名前と色分け
sp.color <- c("magenta", "blue", "grey", "black", "green")
sp.name <-  c('A', 'C', 'M', 'R', 'Z')

# 凡例用に空のプロットをひとつ.
plot(0, 0, type = 'n',  xlab = "", ylab = "")
legend("center", sp.name, col = sp.color, lwd = 2, lty = 1)
```

```
n.blocks <- 150  #  ブロック番号はたかだかここまで.

for (i.block in 1:n.blocks) {    # 各ブロックごとに…

    file.name = sprintf("_%03d.txt", i.block)
    if (file.access(file.name) == -1) {  #  該当ファイルなし
        next                             #  次のブロックへ
    }
    d <- read.table(file.name, header = TRUE)
    plot(1, 1, type = 'n', xlab = "Year", ylab = "Height", log="y",
         ylim = c(1, 300), xlim = c(2003, 2005))  # 枠だけ描画
     text(2003.2, 250, i.block)  #  ブロック名を書き込む.
    for (i in 1:nrow(d)) {  # 個体ごとに…
            #  ※高さ成長を描く(次項)
    }
}
```

```
for (i in 1:nrow(d)) {  # 個体ごとに…
   if (d$h2[i] > 0) {  # 2年めに生きている
        if (d$h3[i] > 0) {  # 3年めも生きてるなら，実線で
            lines(c(2003, 2004, 2005), c(d$h1[i], d$h2[i], d$h3[i]),
                col = sp.color[match(d$sp[i], sp.name)],
                lwd = 1, lty = 1)
        } else {  # 3年めまでに死亡なら，破線で
            lines(c(2003, 2004), c(d$h1[i], d$h2[i]),
                col = sp.color[match(d$sp[i], sp.name)],
                lwd = 1, lty = 2)
        }
    }
    text(2003.2, 250, i.block)  #  ブロック名を書き込む.
}
```

色指定

線の指定

プログラム作図のすすめ

まずは描いてみる

高水準作図関数と低水準作図関数

繰り返し作業をプログラムで

画像の保存

ベクトル画像とビットマップ画像

ベクトル：図の描き方の情報

PostScript, eps, Windowsメタファイル，PDF，…

- 拡大・縮小してもきれい.
- 多くの場合，データサイズが小さい
- 描画に時間がかかることがある

ビットマップ：画素ひとつひとつの色情報

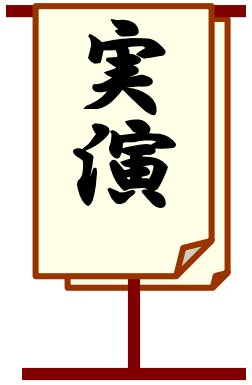
（png, jpeg, bmp, GIF,…）

- 拡大・縮小でみにくくなる.
- データサイズが大きくなりがち（画像によるが）

画像の保存

・描画window 上に描かれた図をメタファイルとして保存するか，クリップボードにコピーし，適当なアプリケーションに渡して保存する.

多数の絵を描くならこちら.



・画像ファイルを描画デバイスとして開いたあと描画を実行すると，ファイルに絵が書き込まれる.

描画デバイスのいろいろ

描画画面

windows(width = …, height = …,)

win.metafile(filename, …)

postscript(filename, …)

pdf(filename, …)

png(filename, …)

jpeg(filename, …)

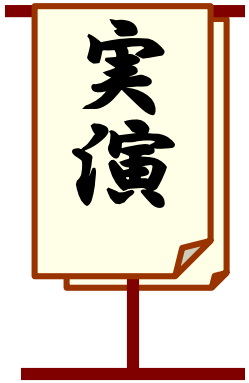
bmp(filename, …)

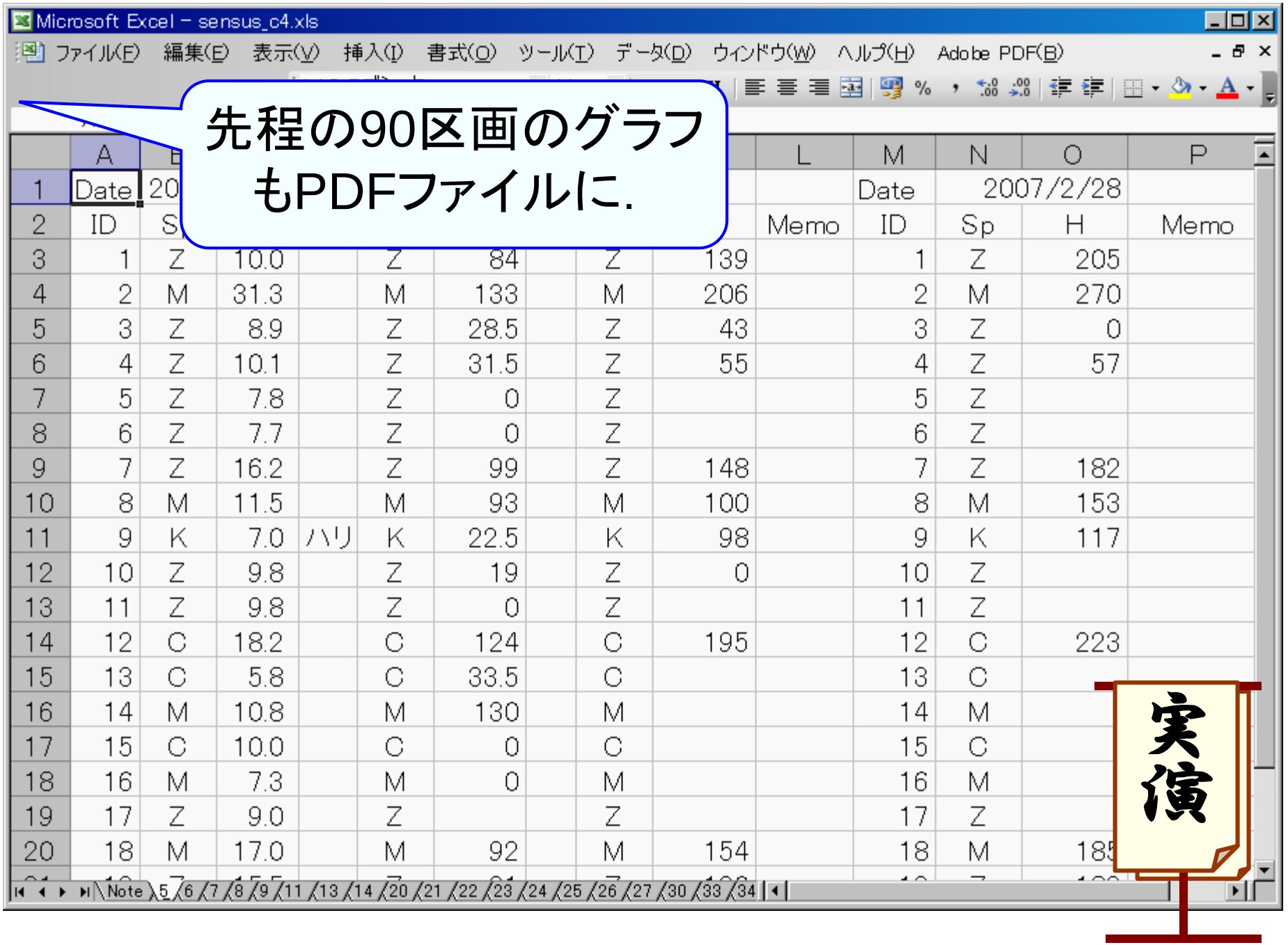
…

ひとつのファイルにたくさんの絵を描けて便利

たとえば

```
> d <- read.table(“leaves.txt”, header = T)
> pdf(“test.pdf”)
> plot(d)
> plot(d$L, d$W)
> plot(d$L ~ d$Tree)
> dev.off()
```




Microsoft Excel - sensus_c4.xls
ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ツール(T) データ(D) ウィンドウ(W) ヘルプ(H) Adobe PDF(B)
先程の90区画のグラフもPDFファイルに.
実演

今日の話は「基礎の基礎」の入門

「基礎の基礎」は拙頁にもあります.

他にもネット上の情報や書籍いろいろあります.

描画関数，描画ライブラリもいろいろあります

R でプログラミング：データの一括処理とグラフ描き

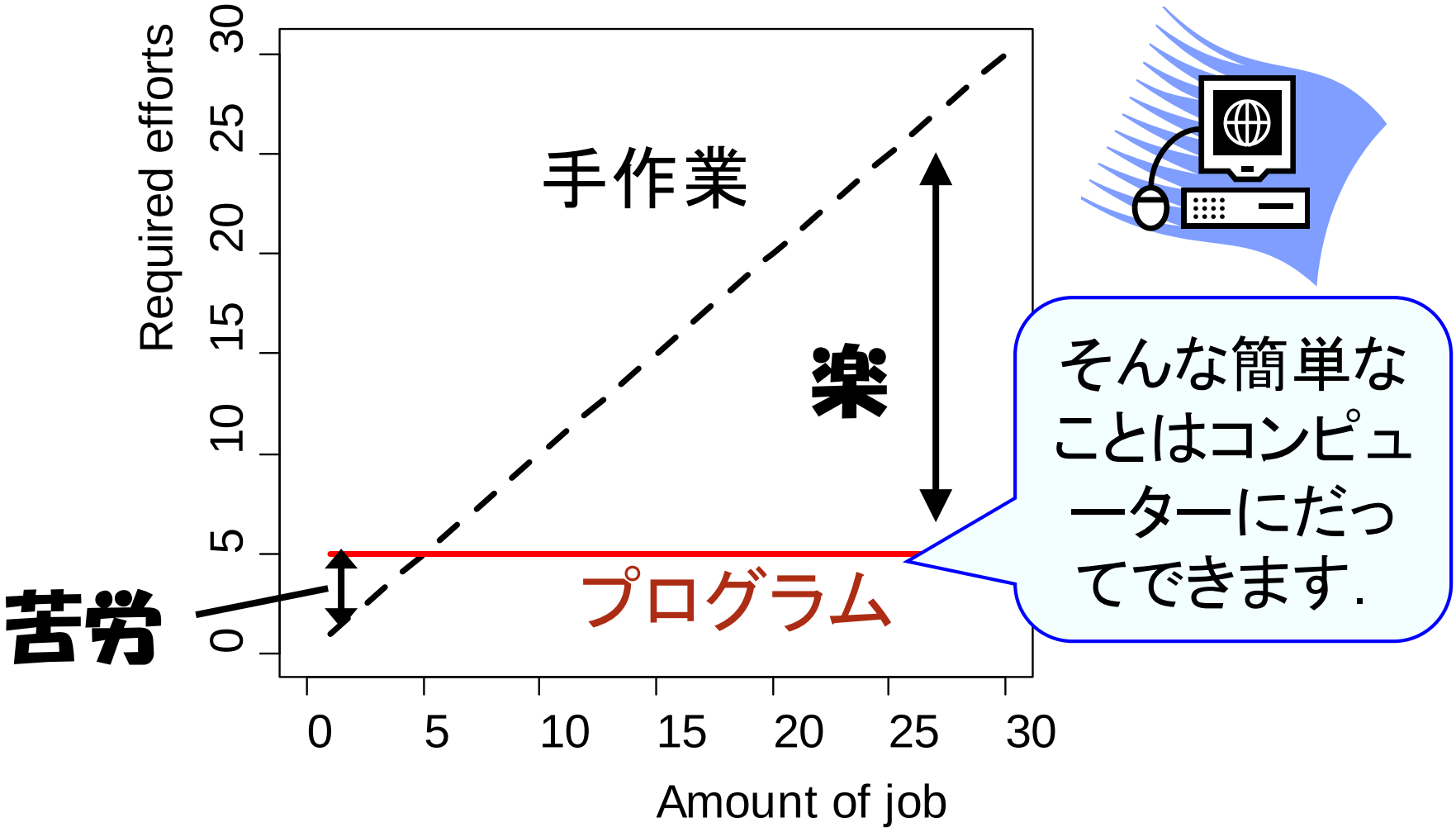
started on 2005-06-06
updated on 2007-04-13

この文書は，フリーの統計解析・作図システム R を使って，データの一括処理と図化のプログラムを書けるようになるためのチュートリアルです． R の経験がまったくなくても読めるように書いています．ただし統計解析手法についての解説はほとんどしていません． 他のページや書籍を見てください．

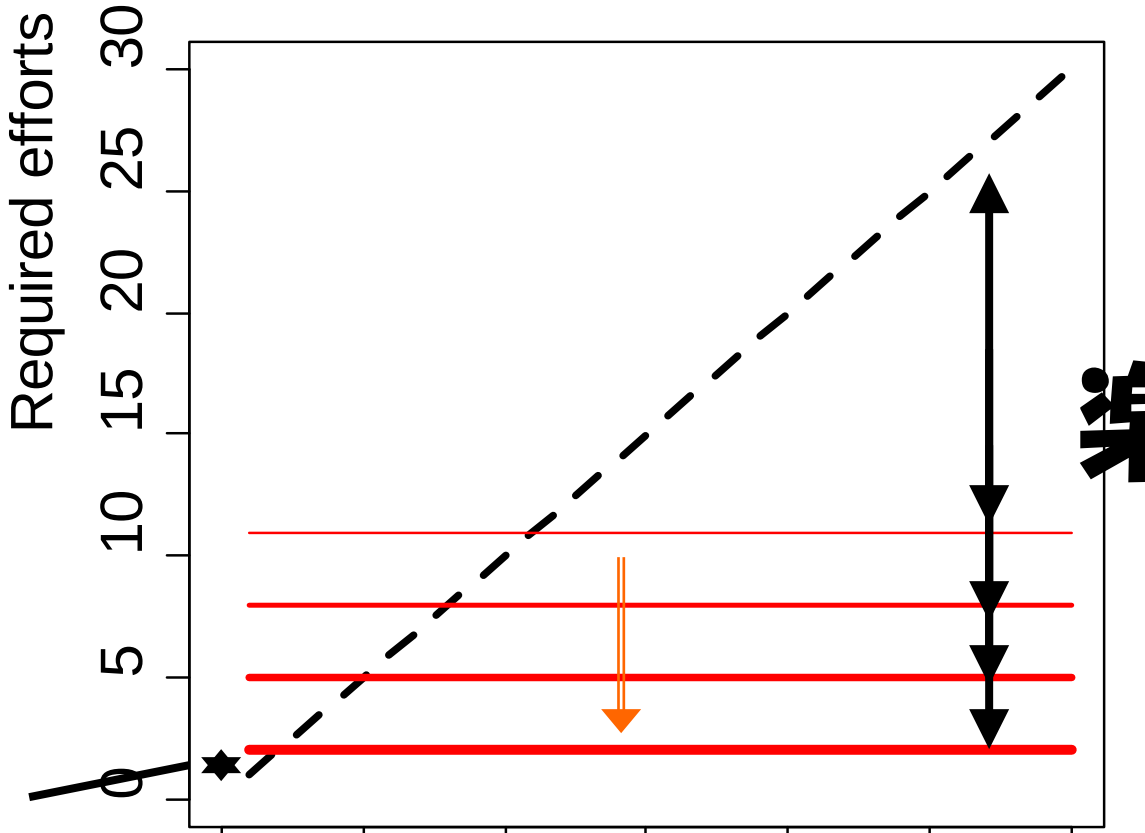
暫定目次

プログラミングの心構え

楽をするための苦労は惜しまない



資産の蓄積と慣れ・習熟で苦労はどんどん小さくなる



楽！

苦労

苦労

苦労…?

楽をするための苦労は惜しまない

清聴多謝