

二次元の変数のグラフ

1. 散布図
2. 時系列プロット
3. 発展的内容:
繰り返し処理を用いた重ね書き
4. 参考資料

散布図

散布図：基本 - 二つの連続変数の関係の視覚化 -

二つの連続変数の関係を点で示す最も基本的なグラフ

- ・ 散布図のそれぞれの点は、対応する二つの連続変数の値の組み合わせを表している
- ・ 散布図を見ることで、二つの変数の間のパターンや関連性をおおまかに把握できる

Rでの基本操作：plot() 関数

```
plot(x="散布図のx軸の座標に指定したいベクトル名",  
     y="散布図のy軸の座標に指定したいベクトル名")
```

注意点

- ・ x 軸と y 軸に指定するベクトルの長さは同じである必要がある
- ・ 引数名 x= と y= は省略して書かれることも多い

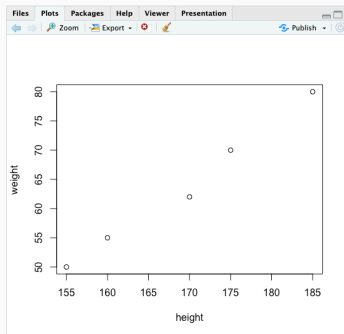
散布図：基本 -例-

あるグループの人々の身長と体重の散布図の作成を想定

```
# 身長データ (単位: cm)
height <- c(155, 170, 175, 160, 185)
# 体重データ (単位: kg)
weight <- c(50, 62, 70, 55, 80)
# 身長をx軸、体重をy軸にした散布図
plot(x = height, y = weight)
```

右下の Plots ウィンドウに
以下のような散布図が表示

演習：上記のコードを実行
してグラフを作成して
みよう



適切なタイトルと軸ラベル: 図の正確な解釈に不可欠

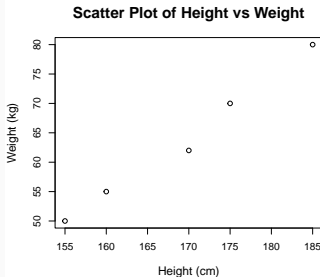
- ・ タイトルとラベルは `plot()` 関数の以下の引数を文字列で指定することで追加
 - ・ `main`: 図のグラフタイトル
 - ・ `xlab`: x 軸ラベル
 - ・ `ylab`: y 軸ラベル
- ・ 文字の大きさは `plot()` 関数の以下の引数を数値で指定することで変更
 - ・ `cex.main`: タイトルの大きさ¹
 - ・ `cex.lab`: 軸ラベルの大きさ
 - ・ デフォルトは 1 で指定する数値は拡大率

¹cex は "character expansion" に由来するらしい

散布図：応用-タイトルとラベルの追加例-

```
# 身長をx軸、体重をy軸にした散布図
plot(x = height, y = weight,
     main = "Scatter Plot of Height vs Weight", # タイトル
     xlab = "Height (cm)", # x軸ラベルの追加
     ylab = "Weight (kg)", # y軸ラベルの追加
     cex.main = 1.5, # タイトルは 1.5 倍のサイズ
     cex.lab = 1.2) # 軸ラベルは 1.2 倍のサイズ
```

演習：上記のコードを実行してグラフにタイトルとラベルを追加してみよう



散布図：応用 - 重ね書き -

複数のグループの散布図に重ね書き：視覚的なグループ間の比較









- ・ `plot()` 関数で散布図を描画した後 `points()` 関数により別の散布図を同じグラフに追加
 - ・ デフォルトでは後から追加した点と最初のプロットの点の区別が難しい
- ・ グループの区別のため、以下の引数で色や形状を変更
 - ・ `col`: 点の色を変更 (例: `col=2`, `col="red"`)
 - ・ `pch`: 形状の変更 (例: `pch=17`, `col="*"`)²

```
points(x="重ね書きのx軸の座標に指定したいベクトル名",  
       y="重ね書きのy軸の座標に指定したいベクトル名",  
       col = 数値 or "色名",  
       pch = 数値 or "文字")
```

²plotting character の略

色の指定



























- ・ col= 数値による指定と色の対応

1	2	3	4	5	6	7	8
							

- ・ 文字列による指定: colors() 関数で一覧を取得できる
- ・ 16 進数カラーコードでも指定できる (col="#DF536B")

点の形状の指定

- ・ pch= 数値による指定と形状の対応

0	1	2	3	4	5	6	7	8
								
9	10	11	12	13	14	15	16	17
								
18	19	20	21	22	23	24	25	
								

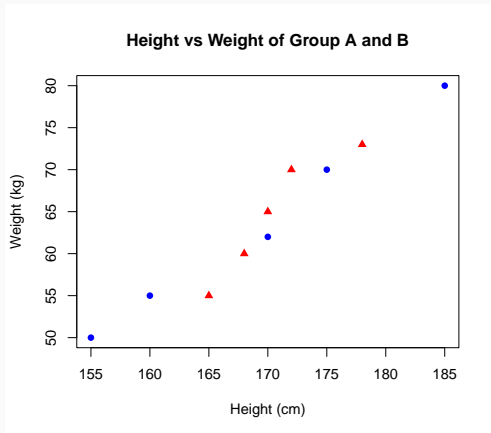
- ・ 21-25 は塗りつぶしが可能な形状
- ・ pch に文字を指定するとその文字がプロットされる (pch="A")

散布図：応用 - 重ね書きの例 -

最初の散布図に別のグループの点を重ね書きする例

```
# グループAの身長と体重
height_A <- c(155, 170, 175, 160, 185)
weight_A <- c(50, 62, 70, 55, 80)
# グループBの身長と体重
height_B <- c(168, 170, 178, 165, 172)
weight_B <- c(60, 65, 73, 55, 70)
# 最初の散布図 (グループA)
plot(x = height_A, y = weight_A,
     main = "Height vs Weight of Group A and B",
     xlab = "Height (cm)", ylab = "Weight (kg)",
     col = "blue", # グループAの点を青色に
     pch = 16) # グループAの点を塗りつぶし丸に
# グループBの点を重ね書き
points(x = height_B, y = weight_B,
      col = "red", # グループBの点を赤色に
      pch = 17) # グループBの点を塗りつぶし三角に
```

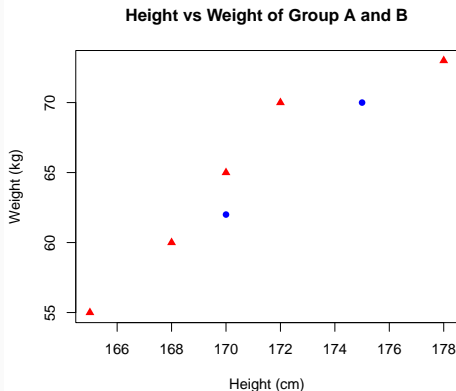
散布図：応用 - 重ね書きの例: 出力-



演習：上記のコードを実行して重ね書きをしたグラフを作成してみよう

散布図：応用 - 重ね書きの注意点: 描画範囲 -

- ・ 重ね書き時の x 軸範囲, y 軸範囲の指定について
 - ・ `plot()` 関数でグラフを描画した場合、後から `points()` 関数で描画しても **描画範囲は変わらない**
- ・ 描画範囲外に点が存在する例:
グループ B を先に `plot()` 関数で描画した場合



- ・ 重ね書きする全ての点が適切に表示されるように
最初に `plot()` 関数で適切な範囲を設定することが必要
 - ・ `xlim`: x 軸範囲を最小値と最大値のベクトルで指定 (`xlim=c(40,80)`)
 - ・ `ylim`: y 軸範囲を最小値と最大値のベクトルで指定 (`ylim=c(140,180)`)

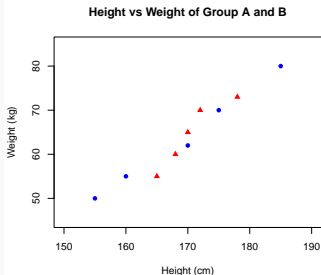
```
plot(..., xlim = x軸の最小値と最大値の長さ2のベクトル,  
       ylim = y軸の最小値と最大値の長さ2のベクトル)
```

散布図：応用 - 軸範囲の指定コード例 -

軸範囲の設定

```
plot(x = height_B, y = weight_B,  
     main = "Height vs Weight of Group A and B",  
     xlab = "Height (cm)", ylab = "Weight (kg)",  
     col = "red", pch = 17,  
     xlim = c(150,190), ylim = c(45,85))  
points(x = height_A, y = weight_A, col = "blue",  
       pch = 16)
```

演習：上記のコードを実行してグラフに軸範囲を設定してみよう



- ・ 複数グループを可視化した散布図に凡例を付けるには `legend()` 関数を使う
- ・ 主な引数
 - ・ `legend`: 表示するラベルの文字列ベクトル
 - ・ `col`: ラベルに対応する点や線の色
 - ・ `pch`: 点の記号
 - ・ `lty`: 線の種類（線も描画する場合）
 - ・ 位置指定: `x=`, `y=` またはキーワード ("`topright`", "`bottomleft`" など)

散布図：応用－凡例の追加例－

散布図の作成

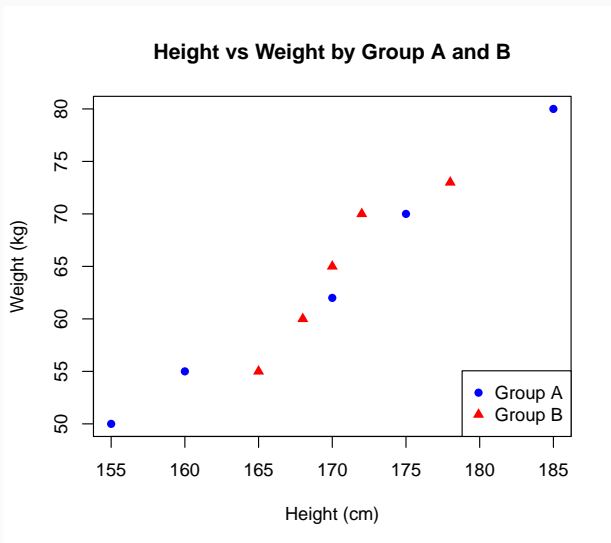
```
plot(x = height_A, y = weight_A,  
     col = "blue", pch = 16,  
     main = "Height vs Weight by Group A and B",  
     xlab = "Height (cm)", ylab = "Weight (kg)")  
points(x = height_B, y = weight_B,  
       col = "red", pch = 17)
```

凡例の追加

```
legend("bottomright", # 右下に追加  
      legend = c("Group A", "Group B"), # ラベル  
      col = c("blue", "red"), # 点の色  
      pch = c(16, 17)) # 点の形状
```

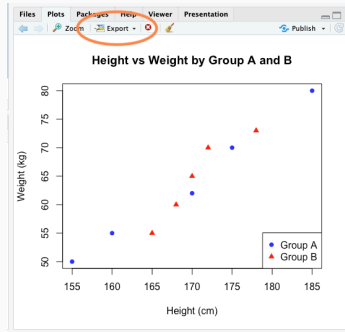
散布図：応用－凡例の追加例－

演習： 上記のコードを実行して、凡例付き散布図を作成してみよう



散布図：応用 - RStudio GUI による保存-

- ・ 画面右下の「Plots」パネル上部の **Export** ボタンをクリック
- ・ メニューから Save as Image または Save as PDF を選択
- ・ フォーマット・幅・高さを指定
- ・ 保存先・ファイル名を指定
- ・ Save をクリックすると保存



Save Plot as PDF

PDF Size: (Device Size) 5.60 × 4.95 inches

Orientation: ☐ Portrait ☒ Landscape

Options: ☐ Use cairo_pdf device (requires X11)

Directory: .../第4回/講義資料作成

File name: Rplot

☐ View plot after saving

Preview Save Cancel

演習問題 -使用データ-

- ・ 使用データ : anorexia_df (MedDataSets パッケージ)
- ・ データ概要 :
 - ・ 拒食症患者を対象としたランダム化比較試験 (RCT) のデータ
 - ・ 被験者数 : 72 名
 - ・ 3 群 (標準ケア、認知行動療法、家族療法) の割り付け
 - ・ 各患者の介入前後の体重変化を記録
- ・ 変数
 - ・ Prewt : 介入前体重 (kg)
 - ・ Postwt : 介入後体重 (kg)
 - ・ Treat : グループ
 - ・ Cont : 対照群 (標準ケア)
 - ・ CBT : 認知行動療法群
 - ・ FT : 家族療法群
- ・ 今回の演習では「対照群」と「認知行動療法群 (CBT)」を比較

介入前体重と介入後体重の関係を表す散布図を対照群と認知行動療法群で点の色、形状を変えて作成してください。

1. MedDataSets からデータセット `anorexia_df` の読み込み
2. 各群のベクトル作成 : `Prewt_Cont`, `Postwt_Cont`,
`Prewt_Treat`, `Postwt_Treat`
3. `plot()` で対照群を描画
4. `points()` で CBT 群を重ね書き
5. `legend()` で凡例を追加

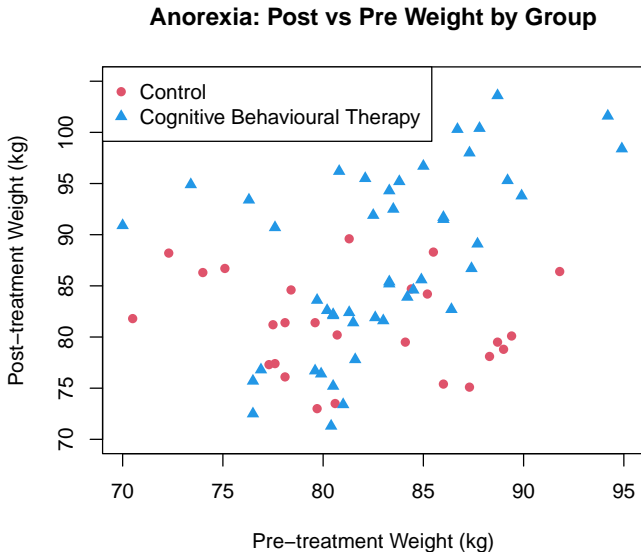
```
# 1. データの読み込み
install.packages("MedDataSets") # 初回のみ
library(MedDataSets)
data("anorexia_df")
df <- anorexia_df

# 2. 各群のベクトルを作成
Prewt_Cont <- df$Prewt[df$Treat == "Cont"]
Postwt_Cont <- df$Postwt[df$Treat == "Cont"]
Prewt_Treat <- df$Prewt[df$Treat == "CBT"]
Postwt_Treat <- df$Postwt[df$Treat == "CBT"]
```

R コード例 -過食症データの散布図作成-

```
# 3. 対照群を plot() で描画
plot(Prewt_Cont, Postwt_Cont,
     col = 2, # 赤
     pch = 16, # 塗りつぶし丸
     main = "Anorexia: Post vs Pre Weight by Group",
     xlab = "Pre-treatment Weight (kg)",
     ylab = "Post-treatment Weight (kg)",
     xlim = c(70,95),ylim = c(70,105))
# 4. CBT 群を points() で重ね書き
points(Prewt_Treat, Postwt_Treat,
       col = 4, # 青
       pch = 17) # 塗りつぶし三角
# 5. 凡例の追加
legend("topleft",
      legend = c("Control", "Cognitive Behavioural
                  Therapy"),
      col = c(2, 4), pch = c(16,17))
```

出力例 -過食症データの散布図作成-



時系列プロット

時系列プロット：基本 - データの時間的変化の視覚化 -

データの時間的な変化を線や点で示すグラフ

- ・ 患者のバイタルデータ（心拍数、血圧など）、薬の血中濃度、感染症の患者数など、多くのデータが時系列データとして扱われる

Rでの基本操作：plot() 関数 - グラフの種類の指定 (type = "l") -

```
plot(x = "x軸(時間軸)の座標に指定したいベクトル名",  
     y = "y軸の座標に指定したいベクトル名",  
     type = "l") # type="l": 折れ線グラフ (line)
```

注意点

- ・ x 軸のデータは数値以外に、時間を表す形式（Date 型、POSIXct/lt 型など）でも良い
- ・ y 軸のデータは数値ベクトル

plot() 関数の引数 type について

type 引数の主な指定:

- ・ "p": 点 (points)
- ・ "l": 線 (lines)
- ・ "b": 点と線 (both)
- ・ "c": 点を描かず線だけを描く (connected lines)
"b" から "p" を除いたもの
- ・ "o": 点と線を重ねて表示 (overplotted)
- ・ "h": 各点から x 軸への垂直線 (histogram-like vertical lines)
- ・ "s": 階段状のグラフ (steps),
y 軸の値が x 軸の値から次の x 軸の値まで同じ
- ・ "S": もう一つの階段状のグラフ,
y 軸の値が x 軸の値から前の x 軸の値まで同じ
- ・ "n": 何も描画しない (no plotting)
後から要素を追加する場合に便利

時系列プロット：基本 -例-

ある患者の入院中の心拍数の推移を可視化

```
# 日付データ
```

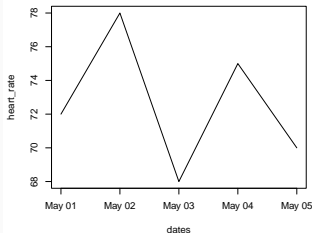
```
dates <- as.Date(c("2025-05-01", "2025-05-02", "  
2025-05-03", "2025-05-04", "2025-05-05"))
```

```
# 心拍数データ（単位：回/分）
```

```
heart_rate <- c(72, 78, 68, 75, 70)
```

```
# 日付をx軸、心拍数をy軸にした時系列プロット（折れ線グラフ）
```

```
plot(x = dates, y = heart_rate, type = "l")
```

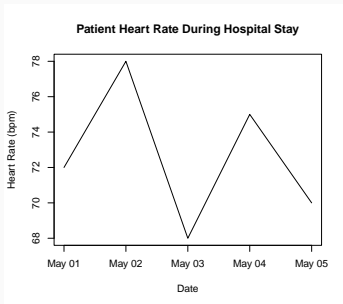


時系列プロット：応用 - 線の種類と点の追加 -

散布図と同様に、タイトル、ラベルを設定できる

```
# 日付をx軸、心拍数をy軸にした時系列プロット
plot(x = dates, y = heart_rate, type = "l",
     main = "Patient Heart Rate During Hospital Stay",
     xlab = "Date",
     ylab = "Heart Rate (bpm)" )
```

演習：上記のコードを実行して時系列データのプロットを試みよう









時系列プロット：応用 -重ね書き-

- ・ 散布図と同様に `plot()` 関数の後に `points()` 関数により重ね書きができる
 - ・ デフォルトでは後から追加した点と最初のプロットの点の区別が難しい
- ・ グループの区別のため、以下の引数で色や線の種類を変更
 - ・ `col`: 点の色を変更 (例: `col=2`, `col="red"`)
 - ・ `lty`: 線の種類の変更 (例: `lty=2`, `lty="dotted"`)³

```
points(x="重ね書きのx軸の座標に指定したいベクトル名",  
       y="重ね書きのy軸の座標に指定したいベクトル名",  
       col = 数値 or "色名",  
       lty = 数値 or "文字")
```

³line type の略

- ・ lty= の指定と線の種類の対応

0	blank	
1	solid	
2	dashed	
3	dotted	
4	dotdash	
5	longdash	
6	twodash	

時系列プロット：応用 -重ね書きの例-

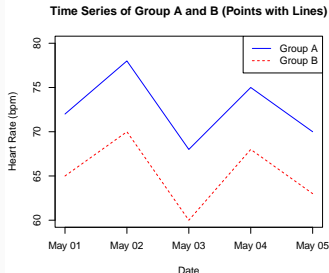
```
# グループAの日付と値
dates_A <- as.Date(c("2025-05-01", "2025-05-02", "
    2025-05-03", "2025-05-04", "2025-05-05"))
values_A <- c(72, 78, 68, 75, 70)
# グループBの日付と値
dates_B <- as.Date(c("2025-05-01", "2025-05-02", "
    2025-05-03", "2025-05-04", "2025-05-05"))
values_B <- c(65, 70, 60, 68, 63)
# 最初の時系列プロット（グループA、線で表示）
plot(x = dates_A, y = values_A, type = "l",
     main = "Time Series of Group A and B (Points with
         Lines)",
     xlab = "Date", ylab = "Heart Rate (bpm)",
     col = "blue", lty = 1, # グループAの線を青色の実線に
     ylim = c(60,80)) # y軸範囲の設定
```


時系列プロット：応用 - 重ね書きの例 続き -

グループBの時系列データを点を線で重ね書き

```
points(x = dates_B, y = values_B, type = "l",  
       col = "red", lty = 2) # グループBの線を赤色の破線に  
# 凡例の追加  
legend("topright", legend = c("Group A", "Group B"),  
       col = c("blue", "red"), lty = c(1, 2))
```

演習：上記のコードを実行
して時系列データの重ね書
きをしてみよう



演習問題 - 使用データ -

- ・ 使用データ : epil (MASS パッケージ)
- ・ データ概要 :
 - ・ 59 人のてんかん患者を対象とした臨床試験データ
 - ・ 各被験者について、介入後 2 週間ごとの発作回数 (計 4 時点) を記録
 - ・ 介入はプログアバイド (progabide) とプラセボ
- ・ 変数
 - ・ subject : 被験者 ID
 - ・ period : 観察時期 (1~4)
 - ・ seizure : 発作回数
 - ・ treatment : 治療群 (placebo / probabide)
- ・ 今回の演習では「被験者 1」と「被験者 2」のデータを比較

以下の手順に従って、時系列プロットを作成してください。

1. `epil` データから被験者 1 と 2 のデータを取り出す
2. `plot()` 関数で被験者 1 のデータを青の実線で描画
3. `points()` 関数で被験者 2 のデータを赤の破線で重ねて描画
4. `legend()` で凡例を追加

```
# データの読み込み
install.packages("MASS") # 初回のみ
library(MASS)
data("epil")

# 被験者1と2のデータを抽出
df1 <- epil[epil$subject==1,]
df2 <- epil[epil$subject==2,]
```

(発展)

- ・ データ抽出は `subset(" データセット名", " 条件")` でも可能
 - ・ 条件は論理式 (例: `subject==1`) で指定
 - ・ 論理式で変数名はそのまま使用できる (\$ マーク不要)

```
df1 <- subset(epil,subject==1)
```

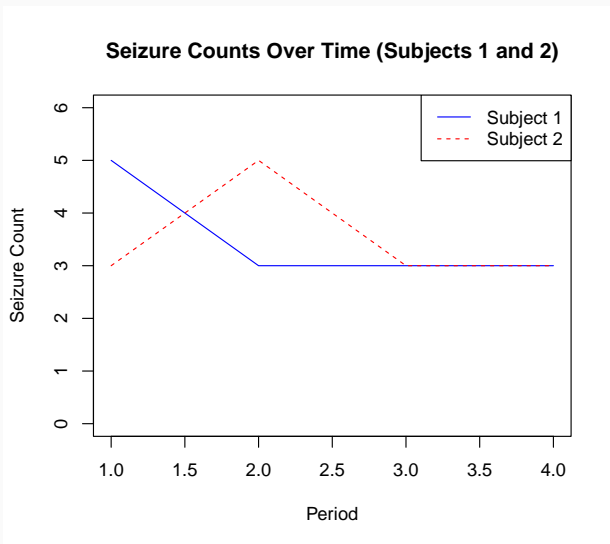
R コード例 -被験者 1 と 2 の発作回数のプロット-

```
# 被験者1をプロット (青の実線)
plot(df1$period, df1$y, type = "l",
     col = "blue", lty = 1,
     ylim = c(0, 6),
     xlab = "Period", ylab = "Seizure Count",
     main = "Seizure Counts Over Time (Subjects 1 and 2)
")

# 被験者2を重ね書き (赤の破線)
points(df2$period, df2$y, col = "red", type="l", lty =
2)

# 凡例の追加
legend("topright",
     legend = c("Subject 1", "Subject 2"),
     col = c("blue", "red"), lty = c(1, 2))
```

出力例 -発作回数の時系列プロット-



発展的内容:

繰り返し処理を用いた重ね書き

for 文による繰り返し処理の基本

for 文は、**繰り返し処理 (loop)**を行うための構文

- ・ 同じ処理を複数のデータに対して繰り返したいときに使用
- ・ 基本構文:

```
for (i in 1:5) {  
  # i を使った処理をここに書く  
}
```

- ・ この例では、i が 1 から 5 まで順に値を取り
- ・ 中括弧 {} 内の処理が 5 回繰り返される

for 文の簡単な例 – 1~5 の二乗を表示–

次のコードは、1~5 の数を順に取り出し、それぞれの 2 乗を表示します。

```
for (i in 1:5) {  
  print(i^2) # iの2乗を表示  
}
```

出力結果：

```
[1] 1  
[1] 4  
[1] 9  
[1] 16  
[1] 25
```

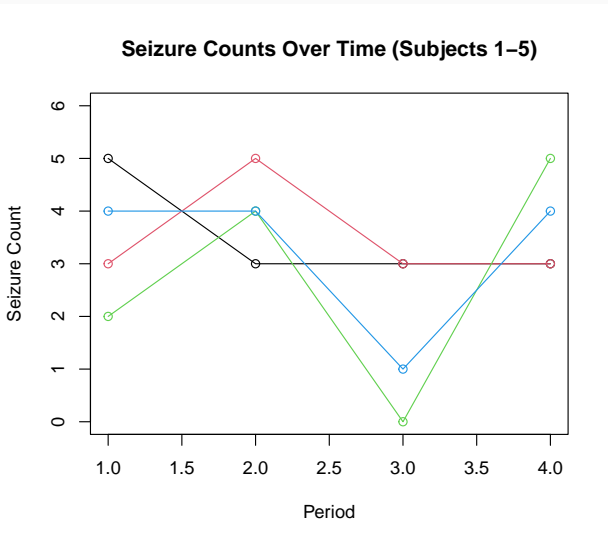
コード例：被験者 1～5 の発作回数を時系列で重ね書き

- ・ 最初に `plot()` で軸・範囲・ラベルだけを描画 (`type = "n"`)
- ・ `for` 文を使って、被験者 1～5 を `points()` で 1 人ずつ重ね書き

```
# 空のプロットを用意 (type = "n")
plot(NA, type = "n", xaxt = FALSE,
     xlim = c(1, 4), ylim = c(0, 6),
     xlab = "Period", ylab = "Seizure Count",
     main = "Seizure Counts Over Time (Subjects 1-5)")

# 被験者1～5のデータを順に描画
for (i in 1:5) {
  dfi <- subset(epil, subject == i)
  points(dfi$period, dfi$y, type = "o", col = i)
}
```

出力例 -被験者 1~5 の発作回数を時系列で重ね書き-



 **演習** : 上記のコードを実行して重ね書きをしてみよう

データセット epil の視覚化

1. epil データセットにおいて placebo 群と progabide 群で色と線の種類を変えて全ての被験者の発作回数の period 1-4 の推移を重ね書きしてください
2. epil データセットにおいて placebo 群と progabide 群で色と線の種類を変えて各群の被験者の発作回数のピリオドごとの平均の推移を重ね書きしてください

 **演習** : 繰り返し処理や層別の処理による重ね書き 

參考資料

ts クラス: 時系列データ専用のデータ構造

- ・ 主に等間隔の時系列データ（例：月次、四半期、年次）に適用
- ・ 開始時点や頻度などの時系列情報を内部に保持
- ・ グラフ描画や時系列解析の関数で自動的に利用される

(参考) ts() 関数の構文と使用例

ts() 関数: ベクトルや行列に時系列情報を付加し、ts クラスのオブジェクトを作成

```
ts(data, start = 開始時点, frequency = 頻度)
```

- ・ data: 数値ベクトルまたは行列
- ・ start: 開始時点 (例: 年次: 1980、月次: c(2023, 1))
- ・ frequency: 周期 (年次 =1, 四半期 =4, 月次 =12 など)

```
# 月次売上データ (2023年1月~2024年12月まで)
sales <- c(100, 120, 130, 125, 140, 135, 150, 155, 160,
          158, 165, 170, 175, 180, 178, 185, 190, 195,
          200, 198, 205, 210, 215, 220)
# tsオブジェクトとして作成 (開始: 2023年1月、頻度: 月次)
ts_sales <- ts(sales, start=c(2023, 1), frequency=12)
# プロット (横軸に年月の情報が自動表示される)
plot(ts_sales,
     main = "Monthly Sales (2023-2024)", ylab = "Sales")
```

(参考) ts クラスで使える主な関数

ts クラスは多くの専用関数に対応しており、時系列解析の基礎的な操作が簡単に実行可能

- ・ `plot()` : 時系列データのグラフ描画 (自動で時間軸が付く)
- ・ `window()` : 特定の期間を取り出す (例: 2010 年以降)
- ・ `cycle()` : 季節要素の確認 (周期的データに有用)
- ・ `frequency()` : データの周期 (frequency) の取得
- ・ `start()`, `end()` : 開始点・終了点の取得
- ・ `lag()`, `diff()` : ラグ (遅れ) や差分系列の生成
- ・ `aggregate()` : 期間単位での集計 (例: 月次 → 年次)

より高度な解析は `forecast` や `fable` パッケージを使用

(参考) plot() 関数の他の引数

1. 軸の調整
 - ・ `log` : 対数軸の指定 ("`x`", "`y`", "`xy`")
2. 色とサイズ
 - ・ `bg` : プロット記号の背景色 (`pch` = 21 から 25 でのみ反映)
 - ・ `cex` : 点のサイズの倍率 (デフォルト = 1)
3. タイトル、軸ラベル
 - ・ `sub` : グラフサブタイトル
 - ・ `cex.main`, `cex.lab`, `col.lab`, `font.sub` など : 文字のサイズや色
4. 軸や枠の制御
 - ・ `axes` : 軸の描画 (FALSE で非表示)
 - ・ `ann` : ラベル・タイトルの描画 (FALSE で非表示)
 - ・ `frame.plot` : 枠線の表示 (TRUE/FALSE)
 - ・ `xaxt`, `yaxt` : 特定の軸のみ抑制 ("n" で非表示)
5. その他の便利な設定
 - ・ `panel.first`, `panel.last` : プロット前後の実行命令
 - ・ `asp` : 縦横比 (例 : `asp` = 1 で等間隔)
 - ・ `xgap.axis`, `ygap.axis` : 目盛りとプロット領域の隙間設定

(参考) par() 関数とマルチパネル表示

- ・ `mfrow = c(nrow, ncol)` : 描画順が行優先のパネル分割
- ・ `mfcol = c(nrow, ncol)` : 描画順が列優先のパネル分割
- ・ `mar = c(bottom, left, top, right)` : プロット内余白
- ・ `oma = c(bottom, left, top, right)` : 全体外余白

```
# 2列横並びに分割
par(mfrow = c(1,2), mar = c(4,4,2,1))
# 左パネル: 対照群
plot(Prewt_Cont, Postwt_Cont, main = "Control",
     xlab = "Prewt (kg)", ylab = "Postwt (kg)",
     col = 2, pch = 16)
# 右パネル: CBT 群
plot(Prewt_Treat, Postwt_Treat, main = "CBT",
     xlab = "Prewt (kg)", ylab = "Postwt (kg)",
     col = 4, pch = 17)
# リセット
par(mfrow = c(1,1))
```

(参考) `par()` 関数の他の引数の紹介

`par()` 関数では、描画の詳細を制御する多くのオプションが指定可能

- ・ `las` : 軸ラベルの方向 (0: 水平, 1: 垂直, 2: 全て垂直, 3: 全て水平)
- ・ `tck` : 目盛り線の長さ (負の値で内向き)
- ・ `cex`, `cex.axis`, `cex.lab`, `cex.main` : 文字サイズの拡大率
- ・ `col.axis`, `col.lab`, `col.main` : 軸・ラベル・タイトルの色
- ・ `xaxt`, `yaxt` : x 軸/y 軸を描画しない ("n" 指定で非表示)

他にも多くの設定が可能

設定できるパラメータと現在の設定は `par()` で表示できる

```
par()
```

`par()` の設定は R セッション全体に影響するため、必要に応じてリセットするのが推奨される

(参考) plot() の引数のベクトル指定

- ・ col、pch などはグループに対応するベクトルで指定できる⁴
- ・ 辞書となるベクトルを作成しておくにより簡単に指定できる
 - ・ Treat が factor の場合、levels(df\$Treat) の順序に注意
 - ・ 必要に応じて as.character() や relevel() を用いて明示的に制御する
 - ・ ifelse などを用いてそれぞれ色を表すベクトルにしてもよい
- ・ long format のデータで特に有用

```
# 辞書となるベクトルの定義
cols <- c(Cont = "red", CBT = "blue")
pchs <- c(Cont = 16, CBT = 17)
plot(df$Prewt, df$Postwt,
      col = cols[df$Treat],
      pch = pchs[df$Treat],
      xlab = "Pre-treatment Weight",
      ylab = "Post-treatment Weight")
```

⁴lty

(参考) ベクトル指定を用いた散布図例

下記のコードで p.20 の演習問題と同じ図が書ける

```
plot(df$Prewt, df$Postwt,  
     col = cols[df$Treat],  
     pch = pchs[df$Treat],  
     main = "Anorexia: Post vs Pre Weight",  
     xlab = "Pre-treatment Weight",  
     ylab = "Post-treatment Weight")  
legend("topleft",  
       legend = names(cols),  
       col = cols,  
       pch = pchs)
```

(参考) 色に透明度を指定する

多くの点が重なり合う散布図で視認性向上に使える

- ・ `rgb(r, g, b, alpha)` 例: `rgb(1,0,0,0.4)`
- ・ `adjustcolor("blue", alpha.f=0.3)`

```
plot(df$Prewt, df$Postwt,  
      col = rgb(1,0,0,0.4), pch = 16,  
      xlab = "Pre-treatment Weight",  
      ylab = "Post-treatment Weight")  
# または  
plot(df$Prewt, df$Postwt,  
      col = adjustcolor("blue", alpha.f = 0.5),  
      pch = 16)
```

(参考) abline() による直線の上書き

- ・ `abline(h = y0)`: 水平線
- ・ `abline(v = x0)`: 垂直線
- ・ `abline(a, b)`: 直線 $y = a + bx$
- ・ `abline(lm(y ~ x))`: 回帰直線

```
plot(df$Prewt, df$Postwt, pch = 16, col = "gray")  
# 45度線を追加  
abline(a = 0, b = 1, lty = 2, col = "black")  
# 回帰直線を追加  
fit <- lm(Postwt ~ Prewt, data=df[df$Treat=="Cont", ])  
abline(fit, col = "red", lwd = 2)
```

(参考) `lines()`, `curve()` の紹介

- ・ `lines()` 関数 :

- ・ `points(..., type = "l")` とほぼ同じ用途だが、`lines()` は線専用
- ・ 点の描画はできない (`pch` は使えない)
- ・ 実線・破線や太さ (`lty`, `lwd`) の指定が可能

```
lines(x, y, col = "blue", lty = 2)
```

- ・ `curve()` 関数 :

- ・ 数式や関数を直接指定して関数曲線を描画できる
- ・ 実測値ではなく、理論式やモデル式の描画に便利
- ・ `curve()` は新規描画と重ね書き (`add = TRUE`) の両方に対応

```
curve(log(x + 1), from = 0, to = 10, col = "red")
```


(参考) axis() による軸の設定

- ・ axis(side, at, labels, las, tck)
 - ・ side : 1= 下, 2= 左, 3= 上, 4= 右
 - ・ at : 目盛の位置 (数値ベクトル)
 - ・ labels : 目盛ラベル (文字ベクトル)
 - ・ las : ラベルの向き (0-3)
 - ・ tck : 目盛線の長さ (絶対値または割合)

```
# 軸なしでプロット
plot(df$Prewt, df$Postwt,
      axes = FALSE, ann = FALSE, pch = 16)
# 下軸
axis(1, at = seq(70, 95, by = 5),
      labels = seq(70, 95, by = 5), las = 1, # 横書き
      tck = -0.02)
# 左軸
axis(2, at = seq(70, 105, by = 5), las = 1)
box() # 枠線を追加
```

(参考) pdf(), png() による図の保存

pdf() と dev.off() の間に書いた図が保存できる

- ・ pdf(" ファイル名.pdf", width=, height=)
- ・ png(" ファイル名.png", width=, height=, res=)
- ・ dev.off(): ファイル書き出し終了

```
# setwd(output) ディレクトリの指定
pdf("anorexia_scatter.pdf", width = 6, height = 4)
  plot(df$Prewt, df$Postwt,
        col = cols[df$Treat],
        pch = pchs[df$Treat])
dev.off()

png("anorexia_scatter.png", width = 800, height = 600,
    res = 150)
  plot(df$Prewt, df$Postwt,
        col = cols[df$Treat],
        pch = pchs[df$Treat])
dev.off()
```

(参考) 連続変数とカテゴリカル変数の2変数の図示

- ・ `boxplot()` 関数は、連続変数とカテゴリカル変数の関係を視覚的に示すのに便利

`boxplot(連続変数 ~ カテゴリカル変数, data=データ名)`

- ・ 例: 治療群 (`trt`) ごとの発作回数 (`y`) の分布を比較

```
boxplot(y ~ trt,  
        data = epil,  
        xlab = "Treatment  
               group",  
        ylab = "Number of  
               seizures")
```

