ex6: Clustering

ex6-1

データセットSato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_small.txt(61遺伝子x8遺伝子型）を使って、ユークリッド距離を使った場合とコサイン係数を使った距離でクラスタリングした時の違いを調べなさい（クラスタリング結果の違いの可視化はdendextendライブラリーを使うのが便利である）。このデータはシロイヌナズナ変異体 にバクテリアを感染させた際の発現プロファイルを取り、野生型との比（log2)をとったものである。PR-1は*P.syringae*に対する防御応答の主要なホルモンであるサリチル酸を介したシグナル伝達経路のマーカー遺伝子である。

コサイン係数での距離、クラスタリングは下記 のカスタム関数を用いてよい。また、heatmapおよびheatmap.2(gplotsライブラリー）では引数に Rowv=as.dendrogram(“行のクラスタリング結果”), Colv=as.dendrogram(“列のクラスタリング結果”)と指定することで任意のクラスタリング結果でヒートマップを描くことができる。

準備

```
library(colorspace)
library(dendextend)
library(dendextendRcpp)
library(gplots)

inputMatrix <- read.delim("~/data/MS/Sato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_small.txt",
               header=TRUE, row.name=1)
heatmapColors <- colorpanel(10, low="blue", mid="white", high="orange")
```

実行

1. まずはユークリッド距離を使ってヒートマップを描いてみる。

```
heatmap.2(as.matrix(inputMatrix),
          scale="none",           # 発現量比のスケーリング無し
          trace="none",           # heatmap.2デフォルトのトレースをキャンセル
          # ヒートマップのマス目設定
          sepcolor="black", colsep=0:ncol(inputMatrix), rowsep=0:nrow(inputMatrix),
          sepwidth=c(0.01, 0.01),
          density.info="none",    # ヒストグラム
          col=heatmapColors,
          cexRow=(0.2 + 1/log10(nrow(inputMatrix)))/3*2,
          RowSideColors=ifelse(rownames(inputMatrix)=="At2g14610", "magenta", "grey")
)
```

2. 次にコサイン係数（ベクトルの角度）を距離尺度としたヒートマップを描いてみる。コサイン係数は関数が無いので自作する。

```
cosine.coef <- function(x,y) {
    a <- sum(na.omit(x * y)) / sqrt( sum(na.omit(x)^2) * sum(na.omit(y)^2) )
    return(a)
}

# making a distance table between columns using uncentered Pearson correlation
cosine.table <- function(x) {
    numberOfPoints <- ncol(x)
    columnNames    <- colnames(x)
    distanceTable  <- matrix(data = NA, nrow = numberOfPoints, ncol = numberOfPoints,
                             dimnames = list( columnNames, columnNames )
                            )

    for ( i in 1:(numberOfPoints-1) ) {
        for ( j in (i+1):numberOfPoints ) {
```

```
            v1 <- x[ , i]
        v2 <- x[ , j]
        d  <- 1 - cosine.coef(v1, v2)
        distanceTable[i, j] <- d
        distanceTable[j, i] <- d
    }
    }

    for ( i in 1:numberOfPoints ) { distanceTable[i, i] <- 1 } # fill the diagonal
        return(distanceTable)
}
```

3. 上記関数とas.dist関数を使ってコサイン係数の距離行列を求め、hclust関数でクラスタリングを実行する。

```
# 行のクラスタリング
rowClusters <- hclust(as.dist(cosine.table(as.matrix((t(inputMatrix))))))
# 列のクラスタリング
colClusters <- hclust(as.dist(cosine.table(as.matrix((inputMatrix)))))
```

4. ヒートマップを描く。Rowv, Colv引数にはas.dendrogram関数を介して上記クラスタリング結果を渡す。

```
heatmap.2(as.matrix(inputMatrix), Rowv=as.dendrogram(rowClusters), Colv=as.dendrogram(colClusters),
scale="none", trace="none", sepcolor="black", colsep=0:ncol(inputMatrix), rowsep=0:nrow(inputMatrix),
sepwidth=c(0.01, 0.01), density.info="none", col=heatmapColors,
cexRow=(0.2 + 1/log10(nrow(inputMatrix)))/3*2,
RowSideColors=ifelse(rownames(inputMatrix)=="At2g14610", "magenta", "grey"))
```

5. ユークリッド距離、コサイン係数を使ったクラスタリング結果の違いをdendextendパッケージに含まれる関数を使って可視化する。

```
rowClusters1 <- as.dendrogram(hclust(as.dist(dist(as.matrix((inputMatrix))))))
rowClusters2 <- as.dendrogram(hclust(as.dist(cosine.table(as.matrix(t(inputMatrix))))))
rowDendrogramList <- dendlist(rowClusters1, rowClusters2)
png("tanglegram_row.png", width=480*4, height=480*2, res=200)
tanglegram(rowDendrogramList, common_subtrees_color_branches = TRUE,
columns_width= c(10,3,10), lab.cex=0.6, lwd=2, main_left="Euclidian", main_right="Cosine")
dev.off()
```

ex6-2

同じデータセットを主成分分析を用いて解析しなさい。この演習ではAt2g14610(PR-1)の主成分得点、npr1-1, sid2-2の負荷量などサリチル酸シグナル伝達経路に注目して主成分分析の結果を評価しなさい。

1. 主成分分析を行うpca関数を定義する。

```
pca <- function(dat)                    # データ行列
{
    if (is.null(rownames(dat))) rownames(dat) <- paste("#", 1:nrow(dat), sep="")
    dat <- subset(dat, complete.cases(dat))     # 欠損値を持つケースを除く
    nr <- nrow(dat)                 # サンプルサイズ
    nc <- ncol(dat)                 # 変数の個数
    if (is.null(colnames(dat))) {
        colnames(dat) <- paste("X", 1:nc, sep="")
    }
    vname <- colnames(dat)
    heikin <- colMeans(dat)             # 各変数の平均値
    bunsan <- apply(dat, 2, var)            # 各変数の不偏分散
    sd <- sqrt(bunsan)              # 各変数の標準偏差
    r <-cor(dat)                    # 相関係数行列
```

```
    result <- eigen(r)              # 固有値・固有ベクトルを求める
    eval <- result$values               # 固有値
    evec <- result$vectors              # 固有ベクトル
    contr <- eval/nc*100                # 寄与率（%）
    cum.contr <- cumsum(contr)          # 累積寄与率（%）
    fl <- t(sqrt(eval)*t(evec))         # 主成分負荷量
    fs <- scale(dat)%*%evec*sqrt(nr/(nr-1))     # 主成分得点
    names(heikin) <- names(bunsan) <- names(sd) <-
        rownames(r) <- colnames(r) <- rownames(fl) <- colnames(dat)
    names(eval) <- names(contr) <- names(cum.contr) <-
        colnames(fl) <- colnames(fs) <- paste("PC", 1:nc, sep="")
    return(structure(list(mean=heikin, variance=bunsan,
            standard.deviation=sd, r=r,
            factor.loadings=fl, eval=eval,
            evec=evec, nr=nr,  # added for subsequent PCA projection
            contribution=contr,
            cum.contribution=cum.contr, fs=fs), class="pca"))
}

# print メソッド
print.pca <- function(  obj,                # pca が返すオブジェクト
            npca=NULL,          # 表示する主成分数
            digits=3)           # 結果の表示桁数
{
    eval <- obj$eval
    nv <- length(eval)
    if (is.null(npca)) {
        npca <- sum(eval >= 1)
    }
    eval <- eval[1:npca]
    cont <- eval/nv
    cumc <- cumsum(cont)
    fl <- obj$factor.loadings[, 1:npca, drop=FALSE]
    rcum <- rowSums(fl^2)
    vname <- rownames(fl)
    max.char <- max(nchar(vname), 12)
    fmt1 <- sprintf("%%%is", max.char)
    fmt2 <- sprintf("%%%is", digits+5)
    fmt3 <- sprintf("%%%i.%if", digits+5, digits)
    cat("\n主成分分析の結果\n\n")
    cat(sprintf(fmt1, ""),
        sprintf(fmt2, c(sprintf("PC%i", 1:npca), "  Contribution")), "\n", sep="", collapse="")
    for (i in 1:nv) {
        cat(sprintf(fmt1, vname[i]),
            sprintf(fmt3, c(fl[i, 1:npca], rcum[i])),
            "\n", sep="", collapse="")
    }
    cat(sprintf(fmt1, "Eigenvalue"),   sprintf(fmt3, eval[1:npca]), "\n", sep="", collapse="")
    cat(sprintf(fmt1, "Contribution"), sprintf(fmt3, cont[1:npca]), "\n", sep="", collapse="")
    cat(sprintf(fmt1, "Cum.contrib."), sprintf(fmt3, cumc[1:npca]), "\n", sep="", collapse="")

}
# summary メソッド
summary.pca <- function(obj,                # pca が返すオブジェクト
            digits=5)           # 結果の表示桁数
{
    print.default(obj, digits=digits)
}

# plot メソッド
plot.pca <- function(obj,               # pca が返すオブジェクト
             which=c("loadings", "scores"), # 主成分負荷量か主成分得点か
             pc.no=c(1,2),          # 描画する主成分番号
             ax=TRUE,                   # 座標軸を描き込むかどうか
             label.cex=0.6,         # 主成分負荷量のプロットのラベルのフォントサイズ
                    markers=NULL,
                ...)                # plot に引き渡す引数
{
    which <- match.arg(which)

    if (which == "loadings") {
        d <- obj$factor.loadings
```

```
    }
    else {
    d <- obj$fs
    }

    label <- sprintf("PC%i", pc.no)
    plot(d[, pc.no[1]], d[, pc.no[2]], xlab=label[1], ylab=label[2], ...)

    if (which == "loadings") {
        labelPosition <- ifelse(d[, pc.no[1]] < 0, 4, 2)
        if (is.null(markers) == FALSE){
            for (marker in markers){
                 points(x=d[marker, pc.no[1]], y=d[marker, pc.no[2]], col="magenta", pch=16)
            }
        }

        text(d[, pc.no[1]], d[, pc.no[2]], rownames(obj$factor.loadings),
             pos=labelPosition, cex=1  #label.cex
            )
    }

    if (which == "scores" && is.null(markers) == FALSE){
        for (marker in markers){
             points(x=d[marker, pc.no[1]], y=d[marker, pc.no[2]], col="magenta", pch=16)
             #text(x=d[marker, pc.no[1]], y=d[marker, pc.no[2]], labels=marker, col="red")# At2g14610
        }
    }

    abline(h=0, v=0)
}
```

2. 主成分分析を実行する。

```
PCAresults <- pca(inputMatrix)
```

3. 結果を主成分得点、因子負荷量を用いて評価しなさい。

```
# 主成分得点（scores）から評価
par(mfrow=c(4,7))
for (kPC1 in 1:(ncol(inputMatrix)-1)){
    for (kPC2 in (kPC1+1):ncol(inputMatrix)){
        plot.pca(PCAresults, which="scores", pc.no=c(kPC1, kPC2), cex=0.5, markers="At2g14610")
    }
}

# 因子負荷量（loadings）から評価
par(mfrow=c(4,7))
for (kPC1 in 1:(ncol(inputMatrix)-1)){
    for (kPC2 in (kPC1+1):ncol(inputMatrix)){
        plot.pca(PCAresults, which="loadings", pc.no=c(kPC1, kPC2), cex=0.5,
                 markers=c("npr1.1", "sid2.2"))
    }
}
```

ex6-3

__（発展問題: k-means法はトレーニングコースでは解説していません）__coi1, dde2, jar1, jin1はジャスモン酸シグナル伝達経路、ein2-1, ein3はエチレンシグナル伝達経路、npr1-1, sid2-2はサリチル酸シグナル伝達経路に関わる遺伝子の変異体である。k-means法はデータをいくつのクラスターに分ければよいか見当が付いている場合によいクラスリング方法である。これらの変異体遺伝子発現プロファイルをk-means法を使って解析し、クラスター数の妥当性を考察せよ。kの数は3から始めなさい。同じ処理

例

```
kmeans(t(inputMatrix), centers=3)$cluster
```

を繰り返し、結果の安定性やクラスタリングのされ方を指標に結果を評価しなさい。

ヒント: centers引数、iter.max引数を調整することにより、結果が変化します。

ex6-4

Sato_A_thaliana-P_syringae_arvRpt2_6h_expRatio_full.txt（484遺伝子x22遺伝子型の実データセット）を使って同じ解析をし、より複雑なデータセットを使った場合のクラスタリング結果の違いを検討しなさい。