ESPRESSIONI GENICHE E BIOMARCATORI

Salvatore Alaimo, MSc.

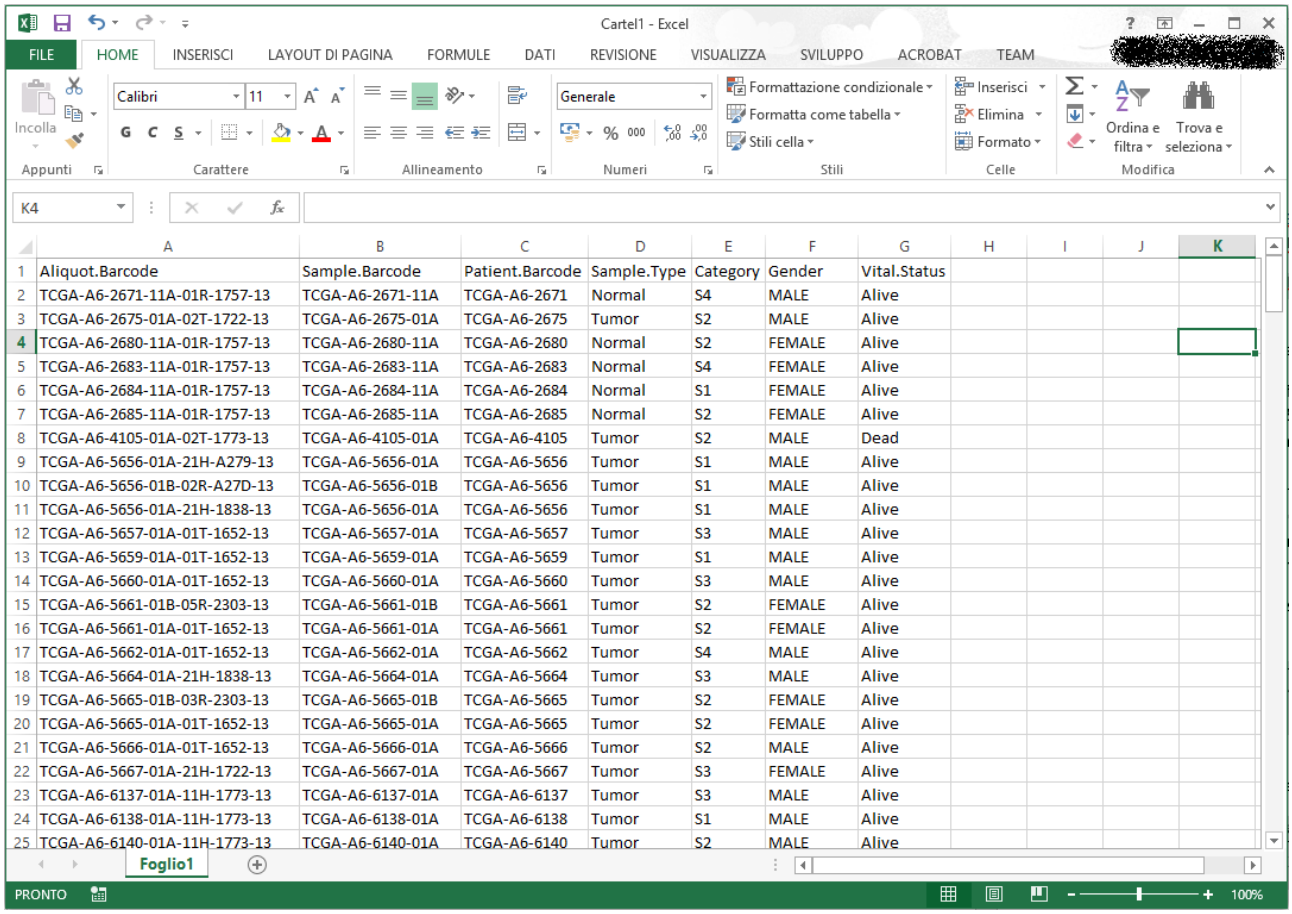
e-Mail: alaimos@dmi.unict.it – Web: http://www.alaimos.com/

SOMMARIO

INTRODUZIONE

In questa esercitazione saranno descritti alcuni dei passi fondamentali per la ricerca di biomarcatori utilizzando gli strumenti forniti dal linguaggio R. In particolare sarà illustrato l'utilizzo dei pacchetti LIMMA e PAMR per la ricerca dei geni differenzialmente espressi partendo da valori di espressione determinati per mezzo di esperimenti NGS.

Per iniziare abbiamo bisogno di due file contenenti le espressioni geniche e le descrizioni dei campioni. Il primo file è generato dall'analisi dei risultati NGS, mentre il secondo è costruito all'inizio degli esperimenti e descrive i singoli campioni. In generale possono essere aperti in Excel, avendo una struttura a "foglio di calcolo".



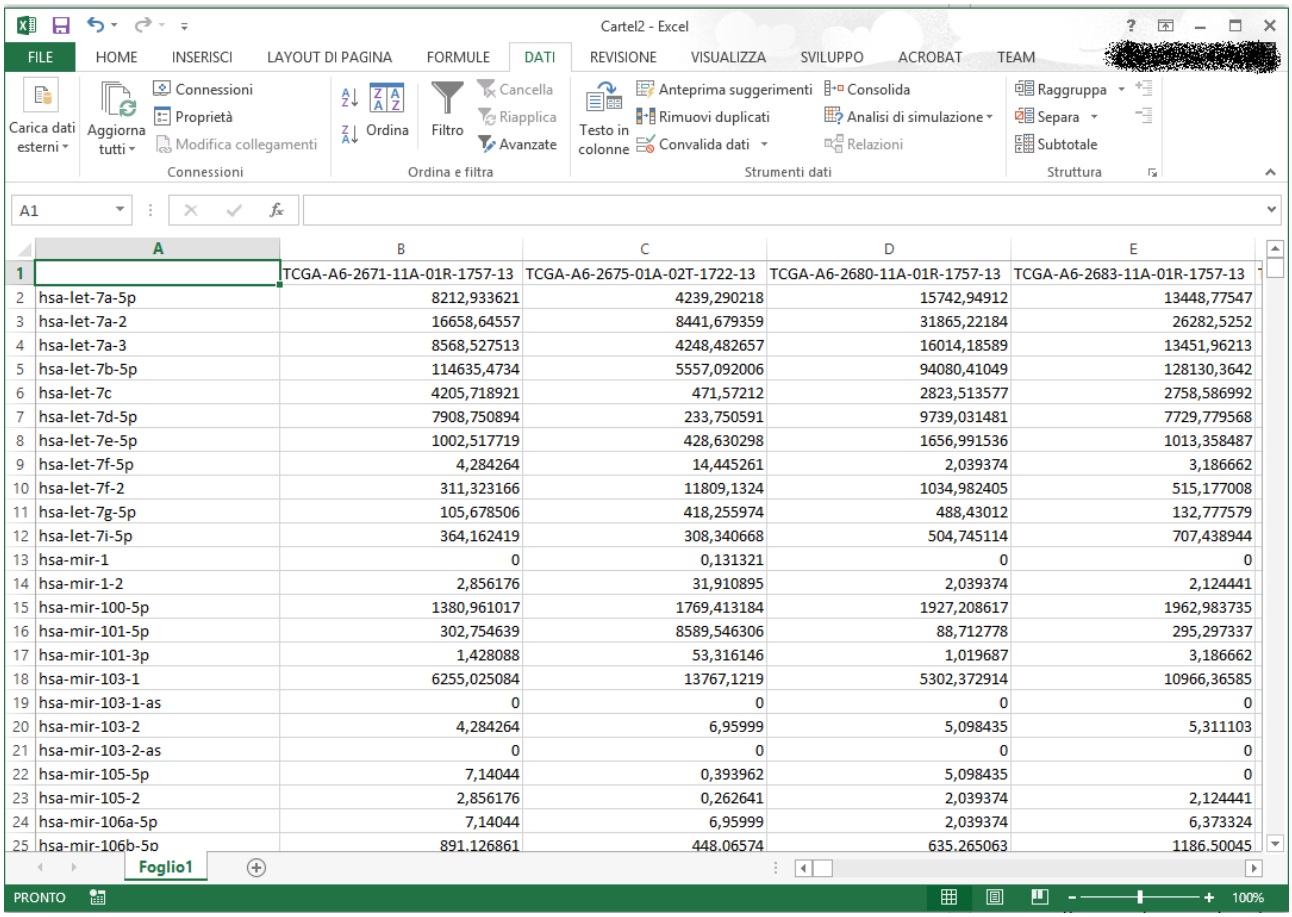
	A	B	C	D	E	F	G
1	Aliquot.Barcode	Sample.Barcode	Patient.Barcode	Sample.Type	Category	Gender	Vital.Status
2	TCGA-A6-2671-11A-01R-1757-13	TCGA-A6-2671-11A	TCGA-A6-2671	Normal	S4	MALE	Alive
3	TCGA-A6-2675-01A-02T-1722-13	TCGA-A6-2675-01A	TCGA-A6-2675	Tumor	S2	MALE	Alive
4	TCGA-A6-2680-11A-01R-1757-13	TCGA-A6-2680-11A	TCGA-A6-2680	Normal	S2	FEMALE	Alive
5	TCGA-A6-2683-11A-01R-1757-13	TCGA-A6-2683-11A	TCGA-A6-2683	Normal	S4	FEMALE	Alive
6	TCGA-A6-2684-11A-01R-1757-13	TCGA-A6-2684-11A	TCGA-A6-2684	Normal	S1	FEMALE	Alive
7	TCGA-A6-2685-11A-01R-1757-13	TCGA-A6-2685-11A	TCGA-A6-2685	Normal	S2	FEMALE	Alive
8	TCGA-A6-4105-01A-02T-1773-13	TCGA-A6-4105-01A	TCGA-A6-4105	Tumor	S2	MALE	Dead
9	TCGA-A6-5656-01A-21H-A279-13	TCGA-A6-5656-01A	TCGA-A6-5656	Tumor	S1	MALE	Alive
10	TCGA-A6-5656-01B-02R-A27D-13	TCGA-A6-5656-01B	TCGA-A6-5656	Tumor	S1	MALE	Alive
11	TCGA-A6-5656-01A-21H-1838-13	TCGA-A6-5656-01A	TCGA-A6-5656	Tumor	S1	MALE	Alive
12	TCGA-A6-5657-01A-01T-1652-13	TCGA-A6-5657-01A	TCGA-A6-5657	Tumor	S3	MALE	Alive
13	TCGA-A6-5659-01A-01T-1652-13	TCGA-A6-5659-01A	TCGA-A6-5659	Tumor	S1	MALE	Alive
14	TCGA-A6-5660-01A-01T-1652-13	TCGA-A6-5660-01A	TCGA-A6-5660	Tumor	S3	MALE	Alive
15	TCGA-A6-5661-01B-05R-2303-13	TCGA-A6-5661-01B	TCGA-A6-5661	Tumor	S2	FEMALE	Alive
16	TCGA-A6-5661-01A-01T-1652-13	TCGA-A6-5661-01A	TCGA-A6-5661	Tumor	S2	FEMALE	Alive
17	TCGA-A6-5662-01A-01T-1652-13	TCGA-A6-5662-01A	TCGA-A6-5662	Tumor	S4	MALE	Alive
18	TCGA-A6-5664-01A-21H-1838-13	TCGA-A6-5664-01A	TCGA-A6-5664	Tumor	S3	MALE	Alive
19	TCGA-A6-5665-01B-03R-2303-13	TCGA-A6-5665-01B	TCGA-A6-5665	Tumor	S2	FEMALE	Alive
20	TCGA-A6-5665-01A-01T-1652-13	TCGA-A6-5665-01A	TCGA-A6-5665	Tumor	S2	FEMALE	Alive
21	TCGA-A6-5666-01A-01T-1652-13	TCGA-A6-5666-01A	TCGA-A6-5666	Tumor	S2	MALE	Alive
22	TCGA-A6-5667-01A-21H-1722-13	TCGA-A6-5667-01A	TCGA-A6-5667	Tumor	S3	FEMALE	Alive
23	TCGA-A6-6137-01A-11H-1773-13	TCGA-A6-6137-01A	TCGA-A6-6137	Tumor	S3	MALE	Alive
24	TCGA-A6-6138-01A-11H-1773-13	TCGA-A6-6138-01A	TCGA-A6-6138	Tumor	S1	MALE	Alive
25	TCGA-A6-6140-01A-11H-1773-13	TCGA-A6-6140-01A	TCGA-A6-6140	Tumor	S2	MALE	Alive

Figura 1 - Esempio di Descrizione

I due file hanno tipicamente una struttura simile alla seguente:

- Nel file di descrizione troviamo una colonna che contiene gli identificativi di ogni singolo campione seguita da alcune colonne che ne descrivono le caratteristiche. Nell'esempio di figura 1, la prima colonna contiene l'identificativo univoco del campione, la seconda e la terza ci forniscono un codice relativo al tipo di campione e al paziente (non ci interessano in questa sede), la quarta colonna descrive il tipo di campione (Tumorale o Normale), le colonne successive ci danno informazioni sulla categoria del tumore, il genere del paziente e se esso è vivo o meno. Per una analisi semplice basta sapere solo se i campioni sono tumorali o normali. In esempi più avanzati useremo anche le altre informazioni.
- Nel file dei profili di espressione troviamo in ogni riga un particolare gene o miRNA (nell'esempio di figura 2 solo miRNA), in ogni colonna vi è il profilo di espressione di un paziente.

Questi file devono essere salvati in formato testo per poter essere analizzati tramite R. Supponiamo negli esempi successivi di avere due file di testo: "description.txt" per le descrizioni, e "expression.txt" per i profili di espressione.



Cartel2 - Excel

FILE HOME INSERISCI LAYOUT DI PAGINA FORMULE DATI REVISIONE VISUALIZZA SVILUPPO ACROBAT TEAM

Carica dati esterni | Aggiorna tutti | Connessioni | Proprietà | Modifica collegamenti — Connessioni
Ordina | Filtro | Cancella | Riapplica | Avanzate — Ordina e filtra
Testo in colonne | Anteprima suggerimenti | Rimuovi duplicati | Convalida dati | Consolida | Analisi di simulazione | Relazioni — Strumenti dati
Raggruppa | Separa | Subtotale — Struttura

A1

	A	B	C	D	E
1		TCGA-A6-2671-11A-01R-1757-13	TCGA-A6-2675-01A-02T-1722-13	TCGA-A6-2680-11A-01R-1757-13	TCGA-A6-2683-11A-01R-1757-13
2	hsa-let-7a-5p	8212,933621	4239,290218	15742,94912	13448,77547
3	hsa-let-7a-2	16658,64557	8441,679359	31865,22184	26282,5252
4	hsa-let-7a-3	8568,527513	4248,482657	16014,18589	13451,96213
5	hsa-let-7b-5p	114635,4734	5557,092006	94080,41049	128130,3642
6	hsa-let-7c	4205,718921	471,57212	2823,513577	2758,586992
7	hsa-let-7d-5p	7908,750894	233,750591	9739,031481	7729,779568
8	hsa-let-7e-5p	1002,517719	428,630298	1656,991536	1013,358487
9	hsa-let-7f-5p	4,284264	14,445261	2,039374	3,186662
10	hsa-let-7f-2	311,323166	11809,1324	1034,982405	515,177008
11	hsa-let-7g-5p	105,678506	418,255974	488,43012	132,777579
12	hsa-let-7i-5p	364,162419	308,340668	504,745114	707,438944
13	hsa-mir-1	0	0,131321	0	0
14	hsa-mir-1-2	2,856176	31,910895	2,039374	2,124441
15	hsa-mir-100-5p	1380,961017	1769,413184	1927,208617	1962,983735
16	hsa-mir-101-5p	302,754639	8589,546306	88,712778	295,297337
17	hsa-mir-101-3p	1,428088	53,316146	1,019687	3,186662
18	hsa-mir-103-1	6255,025084	13767,1219	5302,372914	10966,36585
19	hsa-mir-103-1-as	0	0	0	0
20	hsa-mir-103-2	4,284264	6,95999	5,098435	5,311103
21	hsa-mir-103-2-as	0	0	0	0
22	hsa-mir-105-5p	7,14044	0,393962	5,098435	0
23	hsa-mir-105-2	2,856176	0,262641	2,039374	2,124441
24	hsa-mir-106a-5p	7,14044	6,95999	2,039374	6,373324
25	hsa-mir-106b-5p	891,126861	448,06574	635,265063	1186,50045

Foglio1

PRONTO 100%

Figura 2 - Esempio di profili di espressione

NOTA 1: per salvare un foglio di Excel in un file testuale occorre accedere al menu "File" e selezionare "Salva con Nome". Quando sarà richiesto il nome del file bisogna accertarsi di aver selezionato l'opzione "Testo (con valori delimitati da tabulazioni)" nella casella "Salva Come".

*NOTA 2: nel testo che segue, tutto quello che è racchiuso in un box con sfondo grigio corrisponde a un comando o un output di R. Se un testo in tale box è preceduto dal simbolo "**>** ", esso corrisponde ad un comando che deve essere digitato al prompt dei comandi di R per eseguirlo.*

PREPARAZIONE DELL'AMBIENTE R

Prima di poter iniziare è necessario installare tutti i pacchetti R che useremo.

1. Installare BioConductor

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

2. Installare i pacchetti "limma" e "pamr"

```
> biocLite("limma")
> biocLite("gplots")
> biocLite("pamr")
```

CARICAMENTO DEI DATI

Per caricare i file di testo contenente le descrizioni e i valori di espressione, usiamo il comando R "**read.table**"

```
> description <- read.table("description.txt", header=TRUE,
sep="\t")
> expressions <- read.table("expression.txt", header=TRUE, sep="\t",
check.names=FALSE)
```

In questo modo sono state create due variabili "**description**" e "**expressions**" che contengono rispettivamente le tabelle delle descrizioni e dei valori di espressione. Quando si importa il dataset di espressione, l'uso del parametro "**check.names=FALSE**" è fondamentale per garantire che i nomi dei campioni non vengano modificati da R quando letti dal file di testo.

Affinché limma funzioni correttamente è necessario che i nomi delle righe della tabella delle descrizioni corrispondono a quelle delle colonne della tabella dei valori di espressione. Per fare ciò, basta ricordarsi che nella prima colonna della tabella di esempio ci sono gli identificativi univoci dei campioni, che corrispondono a quelli del dataset di espressione.

Il seguente comando fa quanto detto sopra:

```
> rownames(description) <- as.vector(description[,1])
```

In questo modo stiamo dicendo ad R che i nomi delle righe della tabella descrizione sono contenute nella prima colonna della tabella. La funzione "**as.vector**" serve per assicurarci che l'oggetto contenente i nomi sia nel formato voluto da R.

ANALISI DI BASE CON LIMMA

Limma consente di eseguire una analisi dei geni differenzialmente espressi tra due categorie di pazienti in modo estremamente semplice.

Dopo aver caricato i dati basta eseguire i seguenti passaggi:

1. Caricare la libreria limma

```
> library("limma")
```

2. Costruire una matrice di modello: si costruisce a partire dalle descrizioni e identifica le caratteristiche dei dati in un formato comprensibile da limma.

```
> design <- model.matrix(~0+Sample.Type, data=description)
```

Con questo comando stiamo costruendo una matrice di modello che chiamiamo "design" dicendo che ci interessa suddividere tutti i campioni in base al tipo (Normale o Patologico).

3. Assegnare dei nomi più adeguati alla matrice di modello: per capire meglio cosa si sta facendo è necessario assegnare dei nomi più chiari per noi.

```
> colnames(design)
[1] "Sample.TypeNormal" "Sample.TypeTumor"
> colnames(design) <- c("Normal", "Tumor")
> colnames(design)
[1] "Normal" "Tumor"
```

L'assegnazione dei nomi non ha alcun effetto sui risultati finali ma ci permette di capire meglio su cosa stiamo lavorando.

4. Normalizzare i count dell'NGS usando la matrice di modello.

```
> normalized.expressions <- voom(expressions, design)
```

Con il comando "voom" si normalizzano i valori di espressione prodotti da un esperimento NGS per l'utilizzo con "limma". Il comando usa la matrice costruita al passo precedente per sapere come sono organizzati i dati.

5. Addestrare un modello limma sui dati.

```
> initial.model <- lmFit(normalized.expressions, design)
> limma.model <- eBayes(initial.model)
```

Con i due comandi sopra indicati si costruisce un modello matematico che limma usa per calcolare i geni differenzialmente espressi.

6. Estrarre i geni differenzialmente espressi.

```
> results <- topTable(limma.model, number=nrow(expressions),
adjust.method="BH", coef="Tumor", p.value=0.01, lfc=1)
> View(results)
```

Con il comando "**topTable**" si estraggono i geni differenzialmente espressi, nello specifico per i pazienti tumorali (indicando **coef="Tumor")**. In questa istruzione è possibile filtrare i risultati per p-value e per logFoldChange. Nello specifico è stato selezionato un p-value massimo di **0.01** (**p.value=0.01**) e un logFoldChange minimo di **1** (**lfc=1**). Se uno o entrambi i parametri non sono specificati R restituirà tutti i geni.

ANALISI AVANZATA CON LIMMA

Oltre ad una semplice analisi Caso/Controllo, limma consente anche di estrarre i geni differenzialmente espressi in situazioni più complesse, come ad esempio in dataset con casi multipli.

1. Dopo aver caricato limma, occorre costruire una motrice di modello che tiene conto di tutte le classi del problema.

```
> design <- model.matrix(~0+Category, data=description)
```

2. Occorre quindi assegnare dei nomi più adeguati alle colonne della matrice modello per lavorare con più facilità sui dati.

```
> colnames(design)
[1] "CategoryN"  "CategoryS1" "CategoryS2" "CategoryS3" "CategoryS4"
> colnames(design) <- c("N", "S1", "S2", "S3", "S4")
> colnames(design)
[1] "N"  "S1" "S2" "S3" "S4"
```

3. Normalizzare i count dell'NGS usando la matrice precedentemente ottenuta.

```
> normalized.expressions <- voom(expressions, design)
```

4. Addestrare un primo modello limma sui dati.

```
> initial.model <- lmFit(normalized.expressions, design)
> limma.model <- eBayes(initial.model)
```

5. Definire un elenco di contrasti, ovvero un elenco di test Caso/Controllo da eseguire in batteria.

```
> contrasts <- makeContrasts(S1-N, S2-N, S3-N, S4-N, levels=design)
```

Con la sintassi del comando precedente si definiscono i test caso/controllo basandosi sulle classi definite nella matrice di modello. Scrivere A-B indica che la classe A è il caso mentre B è il controllo. Si possono usare espressioni più complesse come ad esempio (A+C)-B che indica che le classi A e C sono caso mentre B è il controllo.

6. Addestrare un modello limma a partire da quello calcolato precedentemente e che tiene conto dei contrasti.

```
> contrasts.model <- eBayes(contrasts.fit(limma.model, contrasts))
```

7. Estrarre i geni differenzialmente espressi tra tutti i test o quelli di un singolo test.
   a. Per estrarre i geni differenzialmente espressi tra tutti i test:

```
> results.all <- topTableF(contrasts.model,
number=nrow(expressions), adjust.method="BH", p.value=0.01,
lfc=1.5)
> View(results.all)
```

   b. Per estrarre i geni differenzialmente espressi in un solo test:

```
> results.S1.vs.N <- topTable(contrasts.model, coef=1,
number=nrow(expressions), adjust.method="BH", p.value=0.01,
lfc=1.5)
> View(results.S1.vs.N)
```

Con il parametro "**coef**" si seleziona il singolo test. Con i parametri "**p.value**" e "**lfc**" si applica un filtro sul p-value massimo e sul logFoldChange minimo.

VISUALIZZAZIONE DI UNA HEATMAP

Supponendo di usare i risultati dell'analisi precedente, possiamo costruire una heatmap dei geni differenzialmente espressi nel seguente modo.

1. Carichiamo le librerie necessarie generare la heatmap

```
> library("Biobase")
> library("gplots")
```

2. Calcoliamo la heatmap
   a. Estraiamo l'elenco dei geni differenzialmente espressi

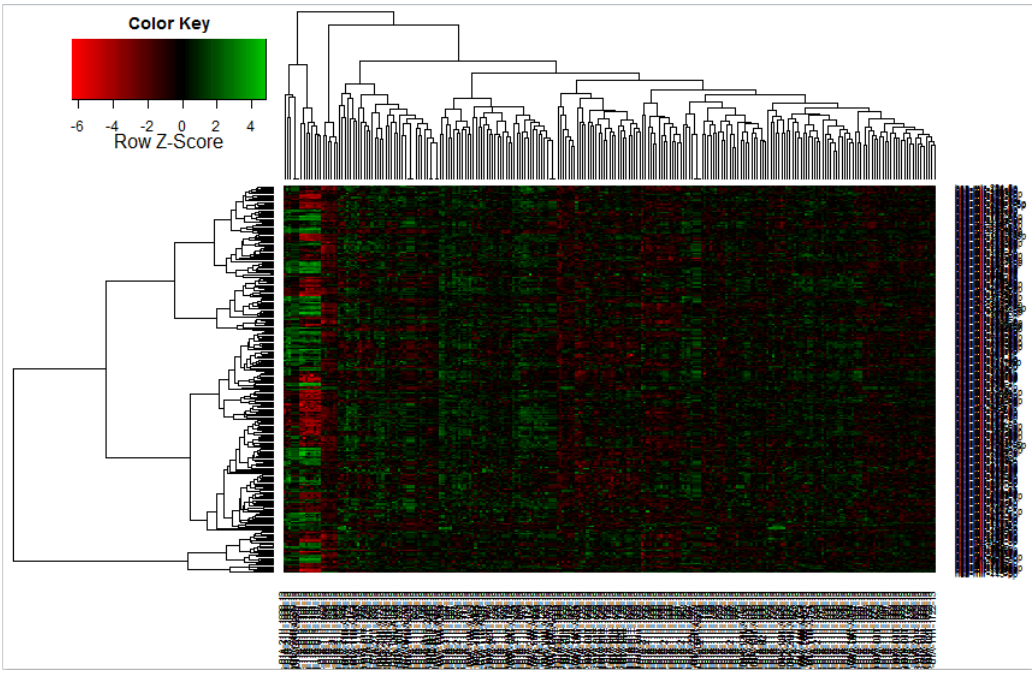
```
> de.genes <- rownames(results.all)
```

   b. Estraiamo i valori di espressione per ogni gene

```
> de.expressions <- normalized.expressions$E[de.genes,]
```

   c. Mostriamo la heatmap

```
> heatmap.2(de.expressions, col=redgreen(100), scale="row",
key=TRUE, symkey=FALSE, density.info="none", trace="none")
```



3. Se si desidera selezionare solo uno o più gruppi di pazienti:
   a. Estraiamo l'elenco dei geni differenzialmente espressi

```
> de.genes <- rownames(results.all)
```

   b. Estraiamo i pazienti a cui siamo interessati usando come guida il file di descrizione.

```
> group1 <- as.vector(
description$Aliquot.Barcode[description$Category == "S1"])
> group2 <- as.vector(
description$Aliquot.Barcode[description$Category == "N"])
> samples <- as.vector(na.omit(c(group1, group2)))
```

   c. Assegniamo ai due gruppi dei colori per riconoscerli meglio nella heatmap

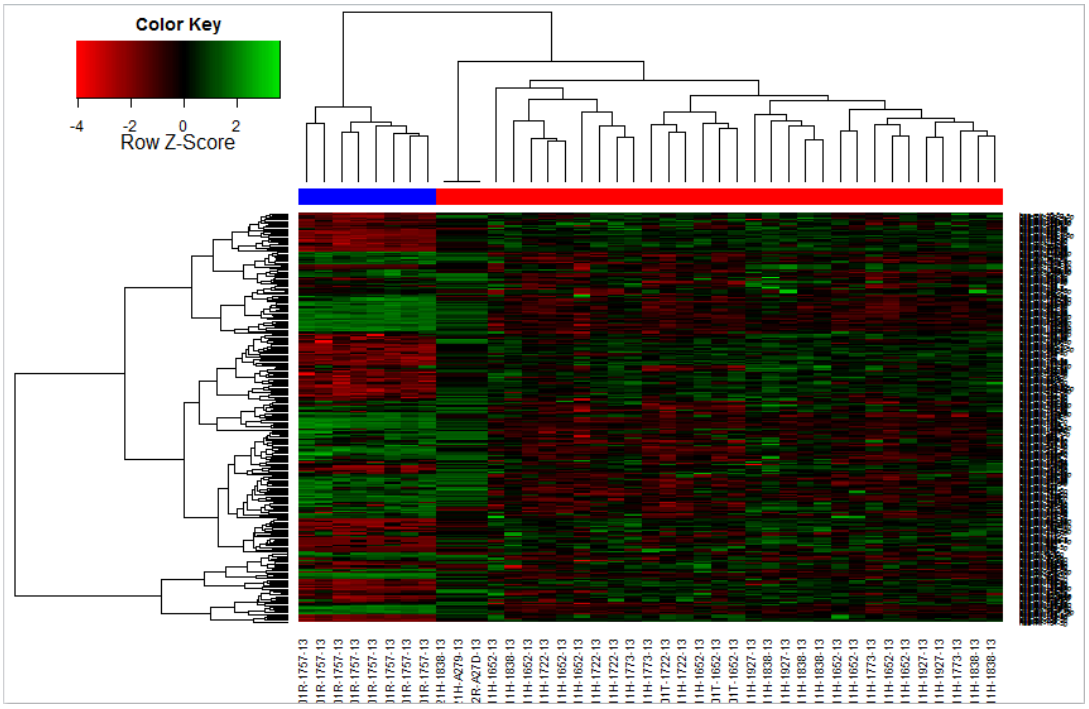
```
> samples.colors <- character(length(samples))
> names(samples.colors) <- samples
> samples.colors[group1] <- "red"
> samples.colors[group2] <- "blue"
```

d. Estraiamo i valori di espressione a cui siamo interessati

```
> de.expressions <- normalized.expressions$E[de.genes, samples]
```

e. Mostriamo la heatmap

```
> heatmap.2(de.expressions, col=redgreen(100), scale="row",
key=TRUE, symkey=FALSE, density.info="none", trace="none",
ColSideColors=samples.colors)
```



VALUTAZIONE DI UN BIOMARCATORE

L'estrazione dei geni differenzialmente espressi non è sufficiente a stabilire quali sono possibili biomarcatori. Un metodo semplice per valutare possibili geni biomarcatori e la relativa qualità consiste nell'utilizzo di boxplot.

1. Usando i risultati dell'analisi precedente, selezioniamo i valori di espressioni relativi all'elemento per cui dobbiamo valutarne la qualità come biomarcatore.

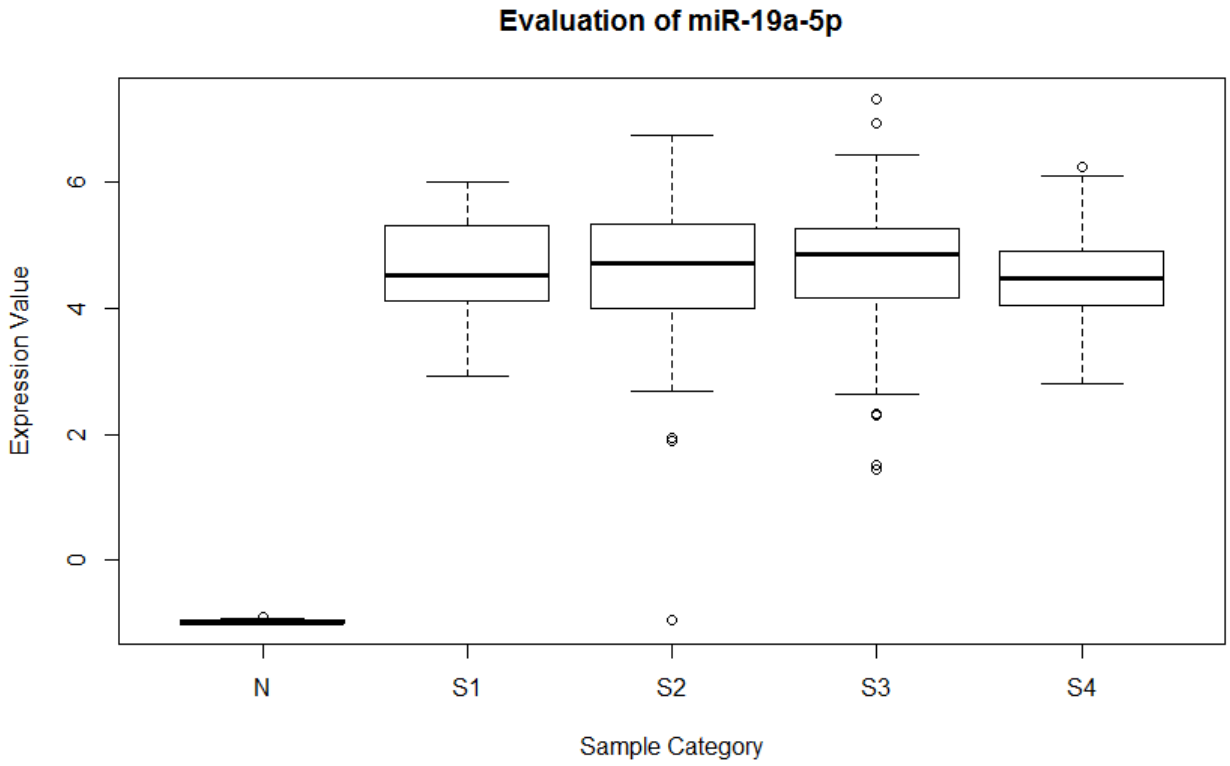
```
> gene       <- "hsa-mir-19a-5p"
> exp.values <- as.vector(normalized.expressions$E[gene,
rownames(description)])
```

2. Costruiamo quindi un dataframe che contiene solo i dati necessari alla visualizzazione del boxplot

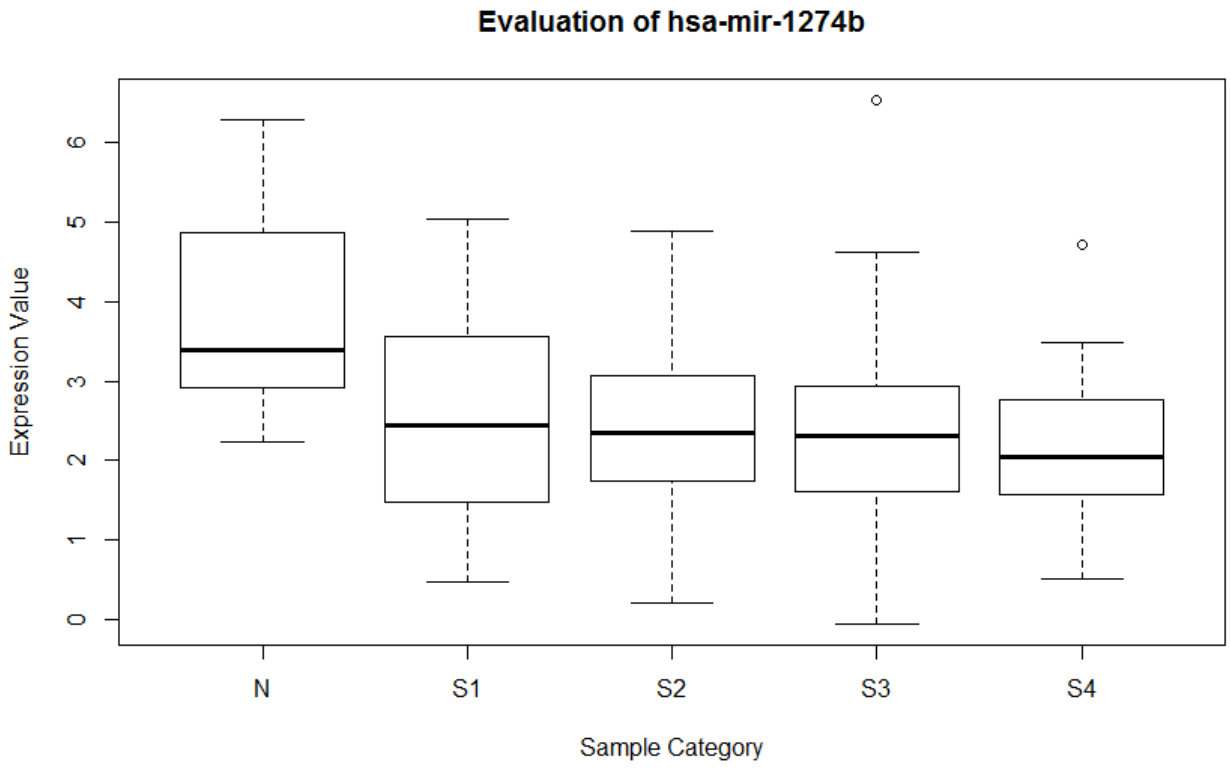
```
> boxplot.data <- data.frame(Sample.Category=description$Category,
Sample.Value=exp.values, row.names=rownames(description))
```

3. Ora possiamo visualizzare il boxplot

```
> boxplot(Sample.Value~Sample.Category, data=boxplot.data,
main=paste0("Evaluation of ", gene), xlab="Sample Category",
ylab="Expression Value")
```



Il risultato in figura mostra un buon biomarcatore che distingue tra le classi tumorali e il normale ma non è sufficiente a distinguere all'interno delle varie categorie tumorali. La figura sottostante mostra invece un gene differenzialmente espresso con p-value basso (quindi statisticamente significativo) ma che non è un ottimo biomarcatore in quanto i boxplot si sovrappongono.



ANALISI DI POSSIBILI BIOMARCATORI CON PAMR

A volte il numero di geni differenzialmente espressi ottenuti tramite l'analisi limma è molto elevato e una analisi dei biomarcatori basata sull'uso dei boxplot non è sufficiente, infatti, è necessario analizzare anche le combinazioni dei possibili sottogruppi di geni per ottenere una corretta classificazione dei campioni. In questo senso, un tool come PAMR è estremamente utile per eseguire una serie di test statistici al fine di selezionare il gruppo di geni differenzialmente espressi che più probabilmente è da considerarsi come insieme di biomarcatori.

1. Come prima cosa occorre eseguire l'analisi dei differenzialmente espressi con limma
2. È quindi necessario caricare il pacchetto PAMR:

```
> library("pamr")
```

3. Prepariamo l'elenco dei geni necessari a PAMR per eseguire i calcoli:

```
> de.genes  <- rownames(results)
>
> all.genes <- rownames(normalized.expressions$E)
> names(all.genes) <- rownames(normalized.expressions$E)
```

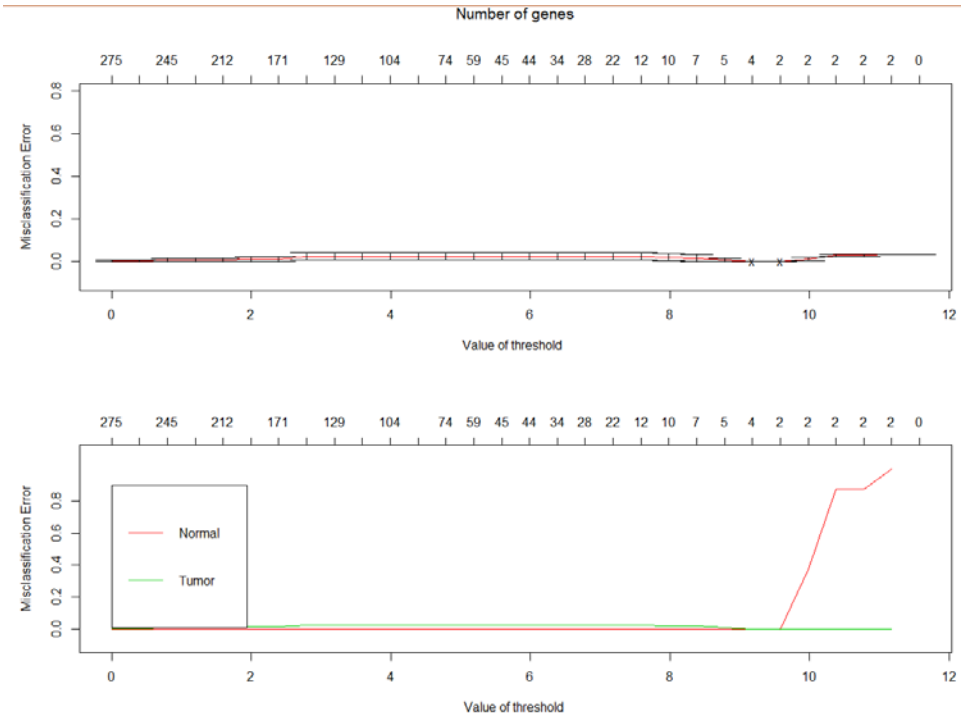
   Con la prima istruzione si mette nella variabile "**de.genes**" l'elenco dei geni differenzialmente espressi ricavato da limma. La seconda e terza istruzione preparano invece un elenco di tutti i geni dell'espresimento nel formato voluto da PAMR.
4. Dobbiamo ora preparare i dati di espressione affinché PAMR li possa caricare:

```
> exp.data <- list(
    "x"=normalized.expressions$E,
    "y"=description$Sample.Type,
    "genenames"=all.genes,
    "geneid"=all.genes
  )
```

   In questo modo prepariamo una lista chiamata "**exp.data**" con gli elementi:
   - "**x**": tutti i valori di espressione normalizzati da limma
   - "**y**": l'elenco delle classi utilizzate per calcolare i geni differenzialmente espressi (nell'esempio Tumorale/Normale)
   - "**genenames**" e "**geneid**": contiene l'elenco di tutti i geni calcolato al passo precedente
5. Fatto questo addestriamo PAMR sui dati in nostro possesso:

```
> pamr.trained <- pamr.train(exp.data, de.genes)
> pamr.cv <- pamr.cv(pamr.trained, exp.data)
> pamr.plotcv(pamr.cv)
```

   Con le prime due istruzioni si addestra PAMR sui dati e si esegue un test di cross-validazione per capire come funziona il modello addestrato. L'ultima istruzione mostra un grafico che ci permette di capire la qualità dell'addestramento:



Il grafico (per ognuna delle soglie scelte da PAMR) ci mostra la percentuale globale degli errori e la percentuale di errore per ogni classe. Al passo successivo sfrutteremo queste informazioni per selezionare la soglia migliore.

6. Estraiamo quindi la soglia calcolata da PAMR che classifica meglio i nostri dati (ovvero minimizza l'errore):

```
> pamr.threshold <- pamr.cv$threshold[which.min(pamr.cv$error)]
```

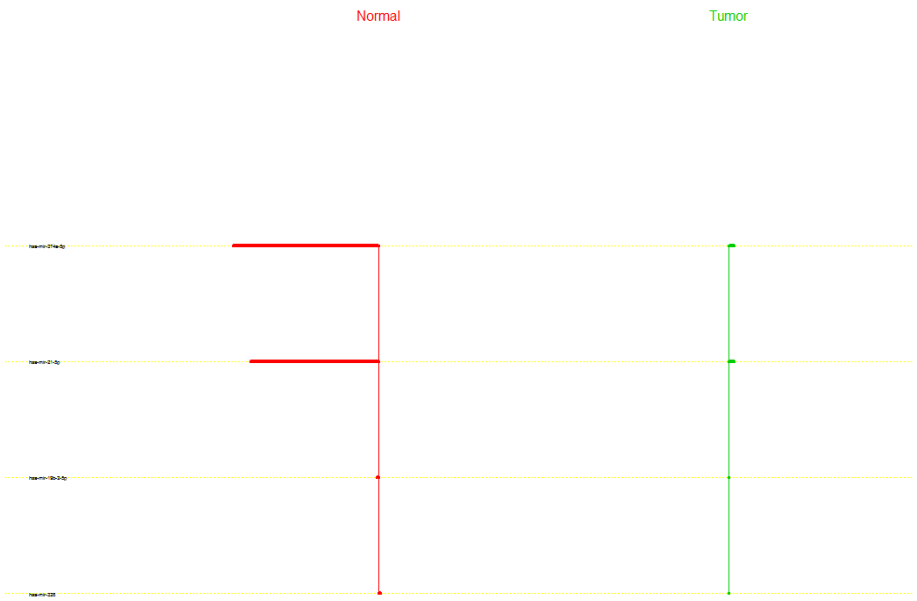
7. Preleviamo l'elenco dei geni scelti da PAMR come possibili biomarcatori

```
> pamr.biomarkers <- pamr.listgenes(pamr.trained, exp.data,
pamr.threshold)
> View(pamr.biomarkers)
```

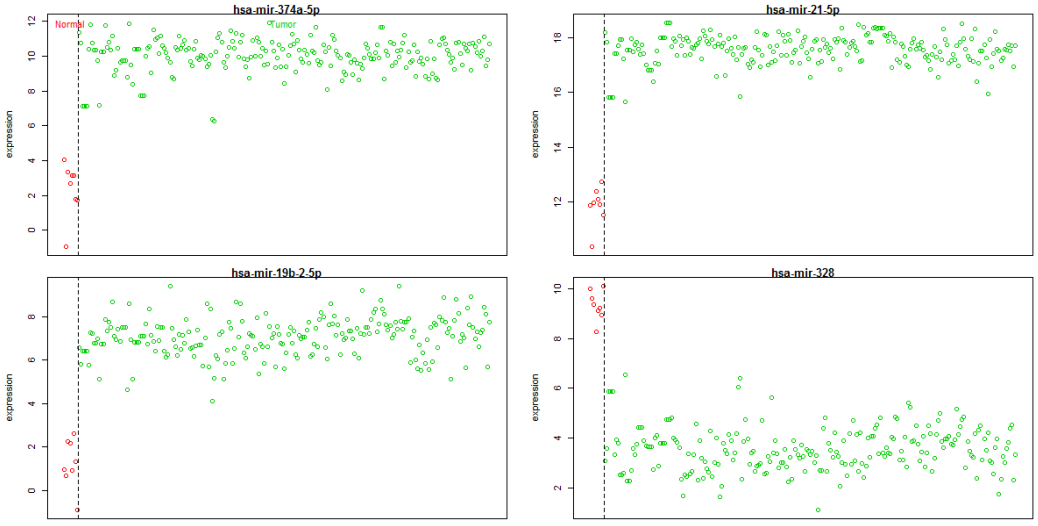
8. Possiamo quindi stampare una serie di grafici che ci aiutano a vedere graficamente i risultati, ovvero come si comportano i geni nelle classi del problema:

```
> pamr.plotcen(pamr.trained, exp.data, pamr.threshold)
> pamr.geneplot(pamr.trained, exp.data, pamr.threshold)
```

Il grafico generato dalla prima istruzione è simile al seguente:



Ci indica in base alla classe come si comporta mediamente ogni gene selezionato da PAMR.
Il secondo grafico invece ci mostra per ogni gene come si organizzano i valori di espressione dei campioni:



Costruito l'elenco dei biomarcatori si può quindi sfruttare R per generare delle heatmap come mostrato nel capitolo precedente. È importante tenere presente che l'oggetto "**pamr.biomarkers**" calcolato da PAMR è diverso da quello ottenuto tramite limma. Quindi per estrarre il nome dei geni differenzialmente espressi bisogna usare la seguente sintassi:

```
> de.genes.pamr <- as.vector(pamr.biomarkers[,"id"])
```

Si possono quindi seguire le istruzioni dei capitoli precedenti per generare la heatmap.