GEO: Breast Cancer

cancer-informatics.org

2023-11-20

Packages

We are using a couple of external packages for the following exampels. Please note that some are from CRAN (the default R repository), while others are from BioConductor (a repository for bioinformatics packages). After installing the packages you need to load them into R:

```
#Load Packages
library(GEOquery)
library(limma)
library(preprocessCore)
library(ggplot2)
library(class)
library(ggfortify)
library(Rtsne)
```

GEO

The Gene Expression Omnibus (GEO) is a public repository of gene expression data. With the `GeoQuery` package from BioConductor we can easily access the data from the GEO database.

```
# Download the dataset
gset <- getGEO("GSE2034", GSEMatrix = TRUE, AnnotGPL = TRUE)
```

```
## Found 1 file(s)
```

```
## GSE2034_series_matrix.txt.gz
```

```
gpl <- getGEO("GPL96")
pData <- pData(gset[[1]])

# Extract the expression data
exprs <- exprs(gset[[1]])
# Extract the phenotype data
pData <- pData(gset[[1]])
```

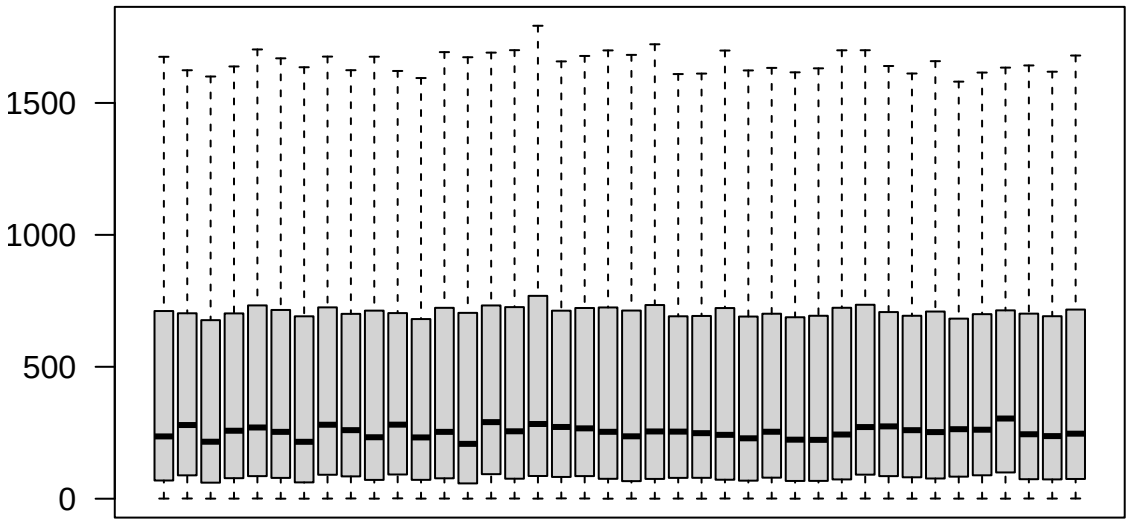
The `exprs()` function extracts the expression data from a series dataset, and the `pData()` function extracts the phenotype data from a series dataset.

```
# We define a index (idx) to align the order from the GPL and GSE probe names
idx <- which(Table(gpl)$ID %in% rownames(exprs))
# Get gene names from gpl and append it to the expression table
geneMatrix <- cbind(exprs, Table(gpl)[idx, "Gene Symbol", drop=FALSE])
# Some genes have multiple probes, so we need to make the probe names
# unique to utilize them as rownames
rownames(geneMatrix) <- make.unique(geneMatrix[, "Gene Symbol"])
# Remove the last column, which is the gene symbol column
```

```
# (as we already have the gene names as rownames)
geneMatrix <- geneMatrix[,1:length(geneMatrix)-1]
# Convert the data.frame to a numeric matrix
geneMatrix <- as.matrix(geneMatrix)
```

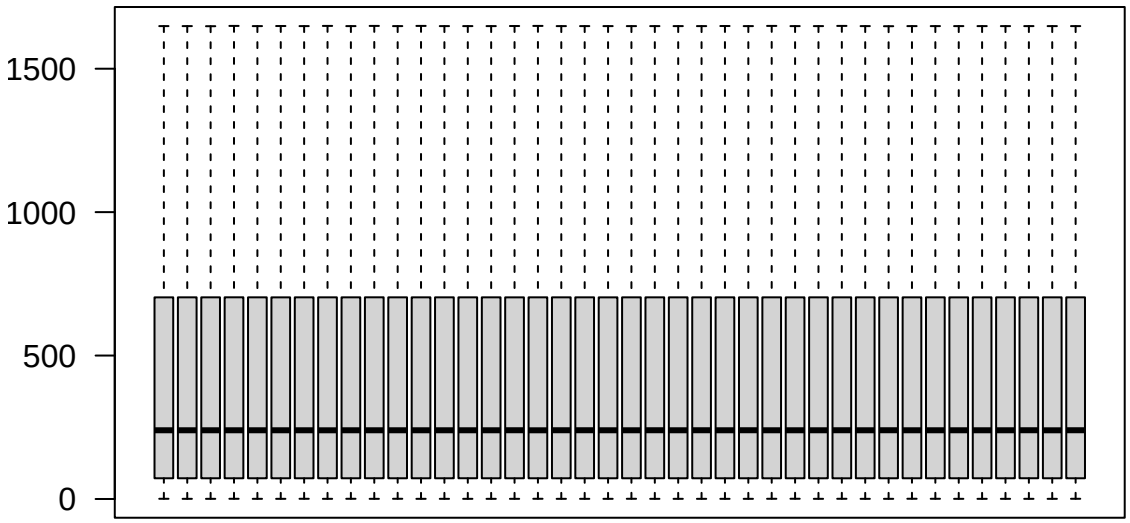
We need to verify if our data is adequately normalized. A solution is to use the `boxplot()` function to visualize the distribution of expression values for each sample.

```
boxplot(geneMatrix[,1:40], outline = F, las = 2, xaxt="n")
```



As we can see, the data is not well normalized. We can use the `normalize.quantiles()` function from the `preprocessCore` package to normalize the data.

```
geneMatrix.norm <- normalize.quantiles(geneMatrix, keep.names=T)
boxplot(geneMatrix.norm[,1:40], outline = F, las = 2, xaxt="n")
```



The data appears to be well normalized after applying `normalize.quantiles()`

Machine Learning

Our example dataset contains 22,283 probes and 286 samples/patients. One approach to explore this dataset is to utilize machine learning. First, we need to retrieve some clinical data from the dataset (e.g., bone

metastasis status). The matrix needs to be transposed from this point forward as currently each row represents one probe, but we need each row to represent one sample/patient.

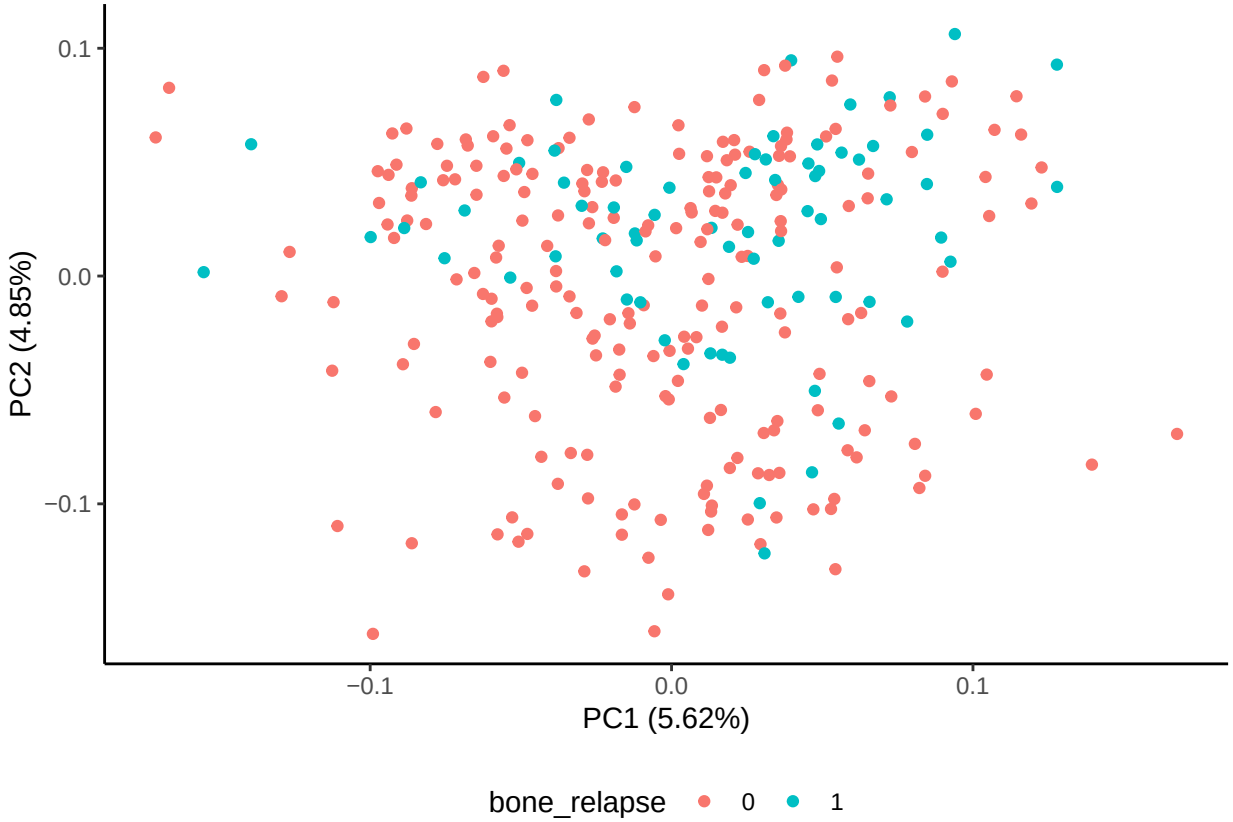
```
# use t() to transform i.e. flip a matrix (turn columns into rows and vice-versa)
geneMatrix.norm.t <- t(geneMatrix.norm)
```

Next, we need to extract the clinical data from the dataset.

```
# extract clinical data, the rownames(...) part is required to keep
# the clinical data in the same order as the expression data
bone_relapse <- pData[rownames(geneMatrix.norm.t),"bone relapses (1=yes, 0=no):ch1"]
# create a dataframe with clinical data and gene expression data
df <- cbind(geneMatrix.norm.t, bone_relapse)
```

It is not trivial to visualize 22,283 probes in a plot. We need to reduce the dimensionality e.g., via principal component analysis (PCA).

```
# perform PCA
pca_res <- prcomp(geneMatrix.norm.t, scale. = TRUE, center=T)
autoplot(pca_res, colour ="bone_relapse", data=df) +
  theme_classic()+ theme(legend.position="bottom")
```



Another popular example is the t-distributed stochastic neighbour embedding (t-SNE).

```
# Perform tSNE and plot via ggplot. Code adapted from:
# https://stackoverflow.com/questions/44837536/how-to-use-ggplot-to-plot-t-sne-clustering
tsne <- Rtsne(geneMatrix.norm.t)
tsne_plot <- data.frame(x = tsne$Y[,1], y = tsne$Y[,2],
                        col = bone_relapse)
ggplot(tsne_plot) + geom_point(aes(x=x, y=y, color=col)) +
```

```
theme_classic() + theme(legend.position="bottom")
```