Microarray Data Analysis using R and Bioconductor

2007 R 統計軟體研習會
蔡政安

Associate Professor
Department of Public Health & Biostatistics Center
China Medical University

Bioconductor

- Bioconductor is an open source and open development software project for the analysis of bioinformatic and genomic data.
- The project was started in the Fall of 2001 and includes 24 core developers in the US, Europe, and Australia.
- **Bioconductor** www.bioconductor.org

  - software, data, and documentation (vignettes);
  - training materials from short courses;
  - mailing list.

Installation of Bioconductor

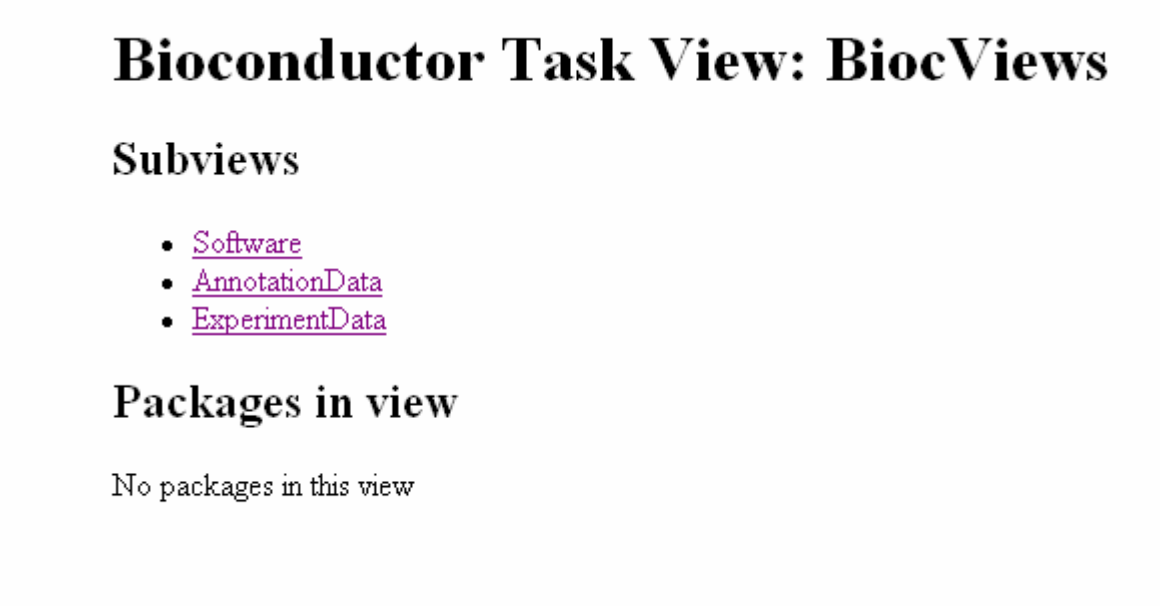
- The latest instructions for installing Bioconductor packages are available on the Download page.
- To install BioConductor packages, execute from the R console the following commands:

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

  biocLite() # Installs the default set of Bioconductor packages.
- **biocLite(c(“made4", “Heatplus"))** # Command to install additional packages from BioC.
- **source("http://www.bioconductor.org/getBioC.R")** # Sources the getBioC.R installation script, which works the same way as biocLite.R, but includes a larger list of default packages.
- **getBioC()** # Installs the getBioC.R default set of BioConductor packages.

Release Packages

- Click on each of these clicks to explore the packages available in Bioconductor.



Bioconductor Task View: BiocViews

Subviews

- Software
- AnnotationData
- ExperimentData

Packages in view

No packages in this view

Bioconductor Software Packages

- Software packages are sub divided into seven categories.
- Each contains a long list of contributed packages.

Bioconductor Task View: Software

Subview of

- BiocViews

Subviews

- Microarray
- Annotation
- Visualization
- Statistics
- GraphsAndNetworks
- Technology
- Infrastructure

Packages in view

Package	Maintainer	Title
EBarrays	Ming Yuan	Unified Approach for Simultaneous Gene Clustering and Differential Expression Identification
Harshlight	Maurizio Pellegrino	A "corrective make-up" program for microarray chips
keggorth	VJ Carey	graph support for KO, KEGG Orthology
panp	Peter Warren	Presence-Absence Calls from Negative Strand Matching Probesets
plier	Crispin Miller	Implements the Affymetrix PLIER algorithm
RbcBook1	Vince Carey	Support for Springer monograph on Bioconductor
splots	Oleg Sklyar	Visualization routines for high throughput screens

Bioconductor Annotation Data Packages

There are over 1,800 bioconductor annotation packages.

These packages provide annotation on the genes on microarrays.

Bioconductor Task View: AnnotationData

Subview of

- BiocViews

Subviews

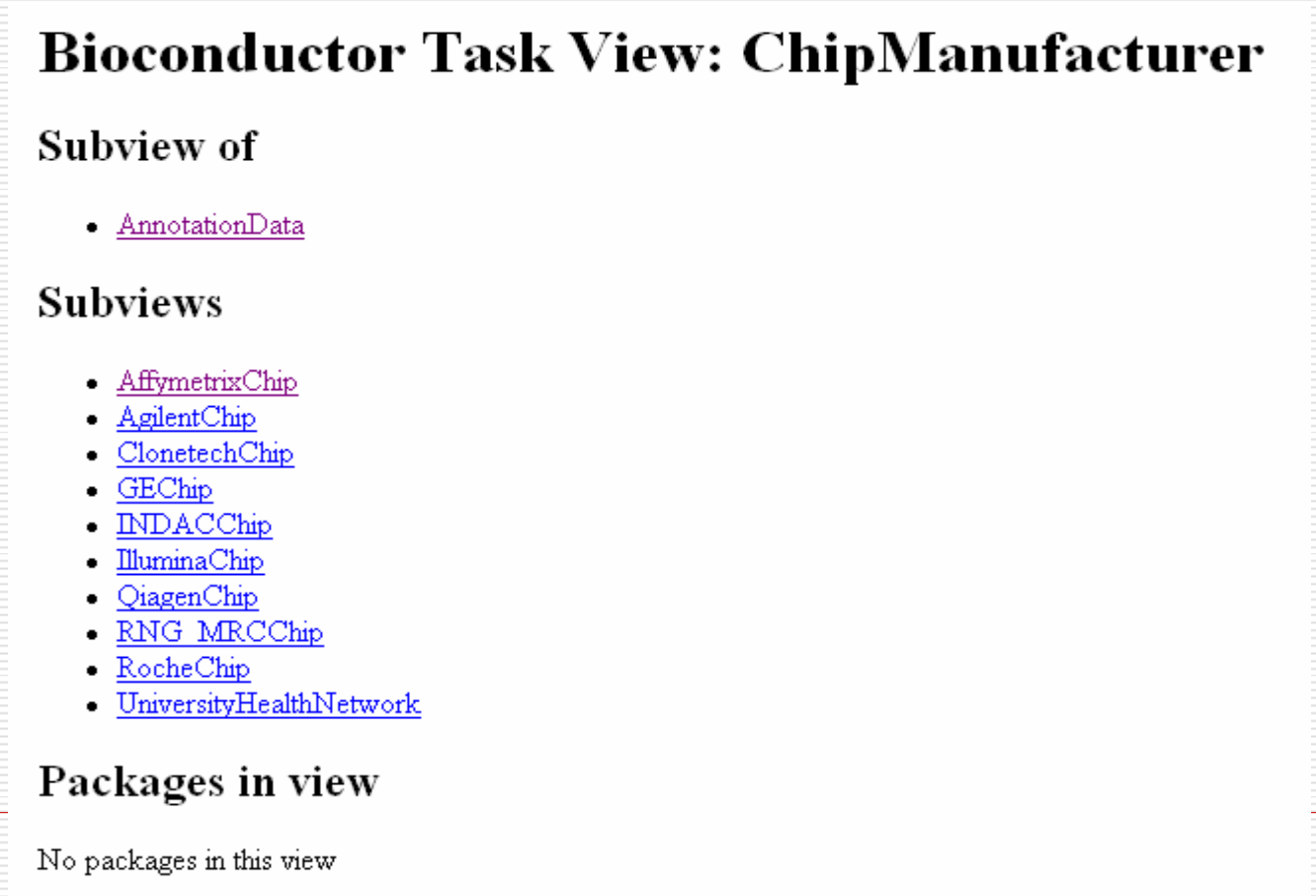
- Organism
- SequenceAnnotation
- FunctionalAnnotation
- ChipManufacturer
- CustomCDF
- CustomArray
- ChipName

Packages in view

Package	Maintainer	Title
adme16cod	Diego Diez	GE CodeLink ADME Rat 16-Assay Bioarray Annotation Data (adme16cod)
ag	Biocore Data Team	Affymetrix Arabidopsis Genome Array Annotation Data (ag)
agahomology	Biocore Data Team	A data package containing annotation data for agahomology
agcdf	Biocore Data Team	agcdf
ag.db	Biocore Data Team	Affymetrix Arabidopsis Genome Array annotation data (chip ag)
agprobe	Biocore Data Team	Probe sequence data for microarrays of type ag

Annotations of Affymetrix GeneChip

- click on chip manufacturer -> Affymetrix



Bioconductor Task View: ChipManufacturer

Subview of

- AnnotationData

Subviews

- AffymetrixChip
- AgilentChip
- ClonetechChip
- GEChip
- INDACChip
- IlluminaChip
- QiagenChip
- RNG_MRCChip
- RocheChip
- UniversityHealthNetwork

Packages in view

No packages in this view

AffymetrixChip Annotations

- Each Affymetrix array is represented by 3 annotation packages
- These contain probe annotation, chip description file (cdf) and probe sequence data (probe). The latter two are compiled from Affymetrix or manufacturer information. The probe annotation (hs133xptense) is compiled from data from public data

hs133xptense	Manhong Dai	Array annotation data of custom CDF hs133xptense for original CHIP u133x3p
hs133xptensecdf	Manhong Dai	hs133xptensecdf
hs133xptenseprobe	Manhong Dai	Probe sequence data for microarrays of type hs133xptense
hs133xptensg	Manhong Dai	Array annotation data of custom CDF hs133xptensg for original CHIP u133x3p
hs133xptensgcdf	Manhong Dai	hs133xptensgcdf
hs133xptensgprobe	Manhong Dai	Probe sequence data for microarrays of type hs133xptensg
hs133xptenst	Manhong Dai	Array annotation data of custom CDF hs133xptenst for original CHIP u133x3p
hs133xptenstcdf	Manhong Dai	hs133xptenstcdf
hs133xptenstprobe	Manhong Dai	Probe sequence data for microarrays of type hs133xptenst
hs133xptentrezg	Manhong Dai	Array annotation data of custom CDF hs133xptentrezg for original CHIP u133x3p
hs133xptentrezgcdf	Manhong Dai	hs133xptentrezgcdf
hs133xptentrezgprobe	Manhong Dai	Probe sequence data for microarrays of type hs133xptentrezg
hs133xptrefseq	Manhong Dai	Array annotation data of custom CDF hs133xptrefseq for original CHIP u133x3p
hs133xptrefseqcdf	Manhong Dai	hs133xptrefseqcdf
hs133xptrefseqprobe	Manhong Dai	Probe sequence data for microarrays of type hs133xptrefseq

Affymetrics Chip System

- Probes
  - Oligionucleotides 25 bp in length (~10^{30} possible)
  - Attach to different segments of each RNA target
  - Probe Pair
    - Perfect Match (PM) probe: exact match to target
    - Mismatch (MM) probe: middle base pair mismatch
  - Probe Set
    - 16~20 probe pairs for a specific RNA
    - Ordered from 5' to 3' end of the RNA
- Files
  - CEL: probe level PMs and MMs related to a common affyID (one CEL file for Chip Expression Level)
  - CDF: relates probe pair sets to location on the array (same map for all chips used in an experiment)

Calibration Issues

- Global Background Correction
  - Spatial flaws in chip (dust, scratches, etc.)
  - Variations in binding
    - due to sample preparation
    - due to variations in number of G-C bonds
    - due to positioning of probe along length of target
- Probe Specific Background Correction
  - A MM probe may be similar to other PM probes, some corresponding to highly expressed genes, making MM > PM
  - Sample may have contained DNA fragments
- Probe set Summarization
- Normalization to compensate for chip to chip variations

Reading in CEL files

- The Affy package provides basic methods for accessing and analyzing affymetrix oligonucleotide arrays.
- Move your CEL files into your working directory and make sure that the corresponding CDF library for your chips is available on your system.
- **library(affy)** # Loads the affy package. **library(estrogen)** # An example data set
- **mydata <- ReadAffy()** # Reads all *.CEL (*.cel) files in your current working directory and stores them into the AffyBatch object 'mydata'.
- **mydata <- ReadAffy(widget=TRUE)** # Opens file browser to select specific CEL files.

Analysis of Affymetrix Arrays

- There are five normalization and probe summarization methods, MAS5, RMA, GCRMA, Plier and dChip.
- **eset <- rma(mydata)** # Creates normalized and background corrected expression values using the RMA method. The generated data are stored as ExpressionSet class in the 'eset' object. For large data sets use the more memory efficient **justRMA()** function.
- **eset <- mas5(mydata)** # Uses expresso (MAS 5.0 method) module instead of RMA, which is much slower than RMA.
- **eset_gcrma <- gcrma(mydata)** # Use this command to employ gcrma method. The library(gcrma) needs to be loaded first.
- **eset_plier <- justPlier(mydata)** # Use this command to employ plier method. The library(plier) needs to be loaded first.
- **eset <- expresso(mydata, normalize.method="invariantset", bg.correct =FALSE, pmcorrect.method="pmonly", summary.method="liwong")** # Generates expression calls similar to dChip (MBEI) method from Li and Wong.

Expresso

- Integrates 4 preprocessing steps into one command.
- bgcorrect.methods
  - mas
  - none
  - rma
  - rma2
- normalize.methods
  - constant
  - contrasts
  - invariantset
  - loess
  - qspline
  - quantiles
  - quantiles.robust

Expresso (Cont'd)

- pmcorrect.methods
  - mas
  - pmonly
  - subtractmm
- express.summary.stat.methods
  - avgdiff
  - liwong
  - mas
  - medianpolish
  - playerout

Data from 'ExpressionSet' objects

- **write.exprs(eset, file="mydata.txt")** # Writes expression values to text file in working directory.

  exprs2excel(eset, file="mydata.csv") # Writes expression values to csv excel file in working directory.

- Retrieving single probe-level data from 'affybatch' objects,

  mypm <- pm(mydata) # retrieves PM intensity values for single probes

  mymm <- mm(mydata) # retrieves MM intensity values for single probes

  myaffyids <- probeNames(mydata) # retrieves Affy IDs

  result <- data.frame(myaffyids, mypm, mymm) # combines the above information in data frame

Working with 'ExpressionSet' objects

- **eset; pData(eset)** # Provides summary information of ExpressionSet object 'eset' and lists the analyzed file names.

- **exprs(eset)[10:20,1:4]; exprs(eset)[c("1910_s_at","1933_g_at"),1:4]** # Retrieves specific rows and fields of ExpressionSet object.

- **test <- as.data.frame(exprs(eset)); eset2 <-new("ExpressionSet", exprs = as.matrix(test), annotation="hgu95av2"); eset2** # Example for creating an ExpressionSet object from a data frame. To create the object from an external file, use the read.delim() function first and then convert it accordingly.

- **data.frame(eset)** # Prints content of 'eset' as data frame to STDOUT.

Retrieving annotation data for Affy IDs

- **library(hgu95av2)** # Opens library with annotation data.
- **library(help=hgu95av2)** # Shows availability and syntax for annotation data.
- **hgu95av2()** # Provides a summary about the available annotation data sets of an anntation library.
- **library(hgu95av2cdf); ls(hgu95av2cdf)** # Retrieves all Affy IDs for a chip in vector format.
- **x <-c("191_at","1910_s_at","1911_s_at","1912_s_at","1913_at","1914_at")**
  # Generates sample data set of Affy ID numbers.
- **mget(x, hgu95av2ACCNUM, ifnotfound=NA)** # Retrieves locus ID numbers for Affy IDs.
- **mget(x, hgu95av2CHR); mget(x, hgu95av2CHRLOC)** # Retrieves chromosome numbers and locations of Affy IDs.
- **mget(x, hgu95av2GO)** # Retrieves GO information for Affy IDs

Visualization and quality controls

- **library(affyQCReport); QCReport(mydata, file="ExampleQC.pdf")** # Generates a comprehensive QC report for the AffyBatch object 'mydata' in PDF format.
- **deg <- AffyRNAdeg(mydata, log.it=F); summaryAffyRNAdeg(deg); plotAffyRNAdeg(deg)** # Performs RNA degradation analysis. It averages on each chip the probes relative to the 5'/3' position on the target genes.
- **image(mydata[,1])** # Reconstructs image with log intensities of first chip.
- **hist(mydata[,1:2])** # Plots histogram of PM intensities for 1st and 2nd array.
- **hist(log2(pm(mydata[,1])), breaks=100, col="blue")** # Plos bar histogram of the PM ('pm') or MM ('mm') log intensities of 1st array.
- **boxplot(mydata,col="red")** # Generates a box plot of un-normalized log intensity values.
- **boxplot(data.frame(exprs(eset)),col="blue", main="Normalized Data")** # Generates a box plot of normalized log intensity values.
- **mva.pairs(pm(mydata)[,c(1,4)])** # Creates MA-plot for un-normalized data. A MA-plot is a plot of log-intensity ratios (M-values) versus log-intensity averages (A-values) between selected chips (here '[1,4]').
- **mva.pairs(exprs(eset)[,c(1,4)])** # Creates MA-plot for normalized data.

Analysis of Differentially Expressed Genes

- Limma is a package for the analysis of gene expression microarray data, especially the use of linear models for analysing designed experiments and the assessment of differential expression. The package includes pre-processing capabilities for two-color spotted arrays. The differential expression methods apply to all array platforms and treat Affymetrix, single channel and two channel experiments in a unified way.
- Basic usage: library(limma);

  library(affylmGUI); affylmGUI() # Requires Tcl/Tk
- Data objects in limma:

  RGList: for cDNA data created by function 'read.maimages()'

  MAList: for cDNA data created by functions MA.RG() or 'normalizeWithinArrays()'

  MArrayLM: created by function 'lmFit()'

  TestResults: created by function 'decideTests()'

Example

- These sample files generated by the Cold Stress Time Course of the AtGenExpress site (ftp batch download) are used to demonstrate the analysis.
- **library(affy); library(limma)**
- **targets <- readTargets("g:\\workshop\\affy_targets.txt")**
- **setwd("g:\\workshop\\")**

  data <- ReadAffy(filenames=targets$FileName)
- **eset <- rma(data)** # Normalization

  # exprs(eset) <- log2(exprs(eset)) for MAS5 results. The limma model requires log2 transformed values.

Example (Cont'd)

- **pData(eset); write.exprs(eset, file="affy_all.txt")** #Lists the file name and exports all expression values to text file.
- **design <- model.matrix(~ -1+factor(c(1,1,2,2,3,3)))** # Creates appropriate design matrix.
- **colnames(design) <- c("group1", "group2", "group3")** # Assigns column names.
- **fit <- lmFit(eset, design)** # Fits a linear model for each gene based on the given series of arrays.
- **contrast.matrix <- makeContrasts(group2-group1, group3-group2, group3-group1, levels=design)** # Creates appropriate contrast matrix to perform all pairwise comparisons.
- **fit2 <- contrasts.fit(fit, contrast.matrix)** # Computes estimated coefficients and standard errors for a given set of contrasts.

Example (Cont'd)

- **fit2 <- eBayes(fit2)** # Computes moderated t-statistics and log-odds of differential expression by empirical Bayes
- **topTable(fit2, coef=1, adjust="fdr", sort.by="B", number=10)** # Generates list of top 10 ('number=10') differentially expressed genes sorted by B-values ('sort.by=B') for first comparison group.
- **write.table(topTable(fit2, coef=1, adjust="fdr", sort.by="B", number=500), file="limma_complete.xls", row.names=F, sep="\t")** # Exports complete limma statistics table for first comparison group.
- **results <- decideTests(fit2, p.value=0.05); vennDiagram(results)**

  # Creates venn diagram of all D.E. genes at the level of 0.05.
- x <- topTable(fit2, coef=1, adjust="fdr", sort.by="P", number=50000); y <- x[x$P.Value < 0.05,];

  y <- x[x$P.Value < 0.01 & (x$logFC > 1 | x$logFC < -1) & x$AveExpr > 10,]
- print("Number of genes in this list:"); length(y$ID)

Packages for Analysis of Differentially Expressed Genes

- "siggenes" package for the SAM procedure.
- "R/maaova" package

  MicroArray ANalysis Of VAriance (maanova) from Gary Churchill's group.
- "Marray" package

  Exploratory analysis for two-color spotted cDNA arrays.
- "RankProd" package

  The RankProd package contains functions for the identification of differentially expressed genes using the rank product non-parametric method from Breitling et al., 2004
- "fdrtool", "qvalue", "localFDR", and "SPLOSH" packages

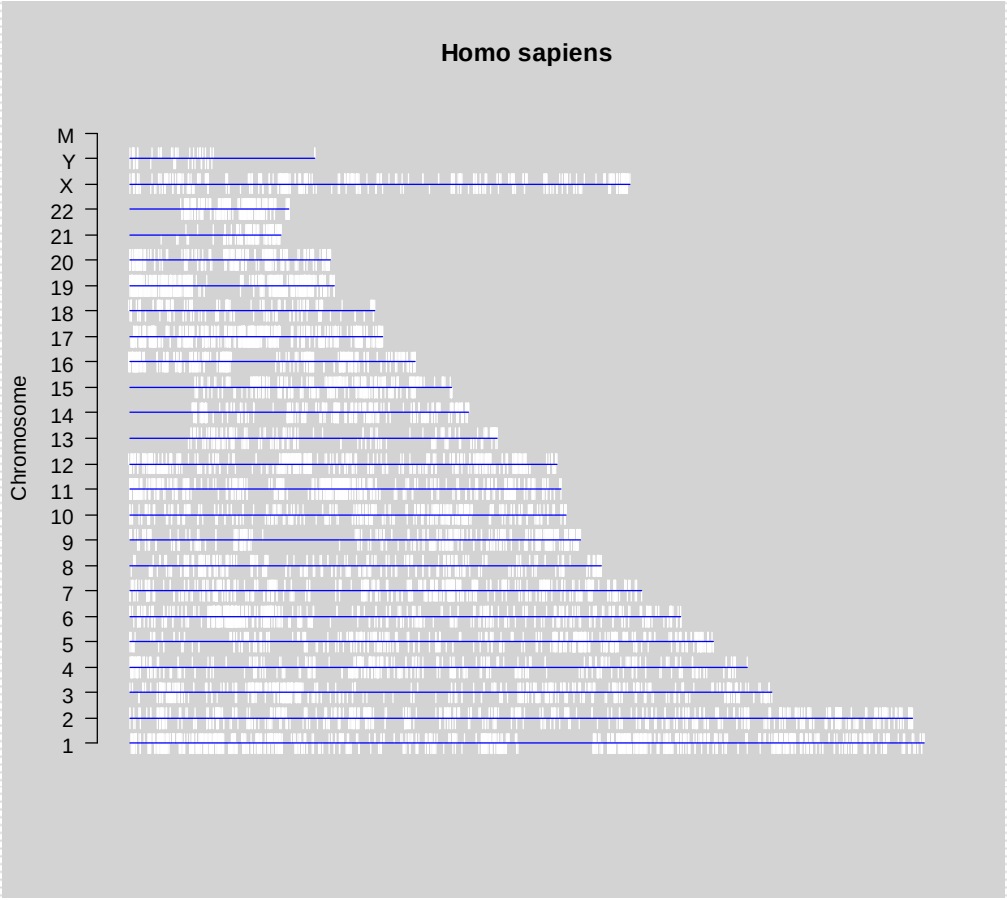
  Estimation and control of False Discovery Rate
- "Simpleaffy" package

  The simpleaffy package automates many repetitive high-level analysis steps.

Some useful packages for Genomic Data

- Gene Ontology (GO) analysis: GOstats; goCluster
- Chromosome maps: geneplotter
- Phylogenetic Analysis: ape
- Protein Structure Analysis: Bio3D
- Motif identification in promoter regions: COSMO

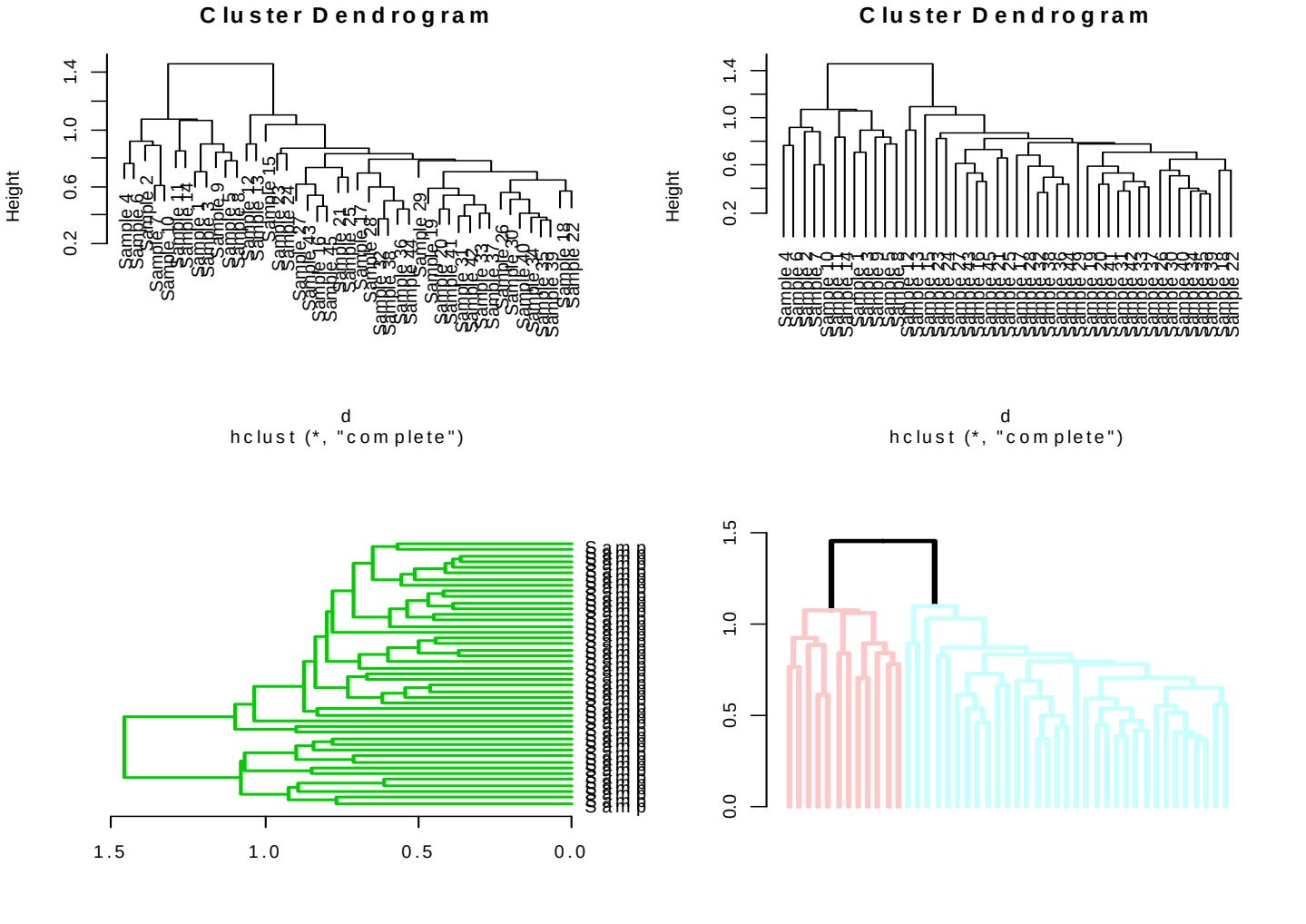
ChromLocation for Affymetrix hgu95av2 array



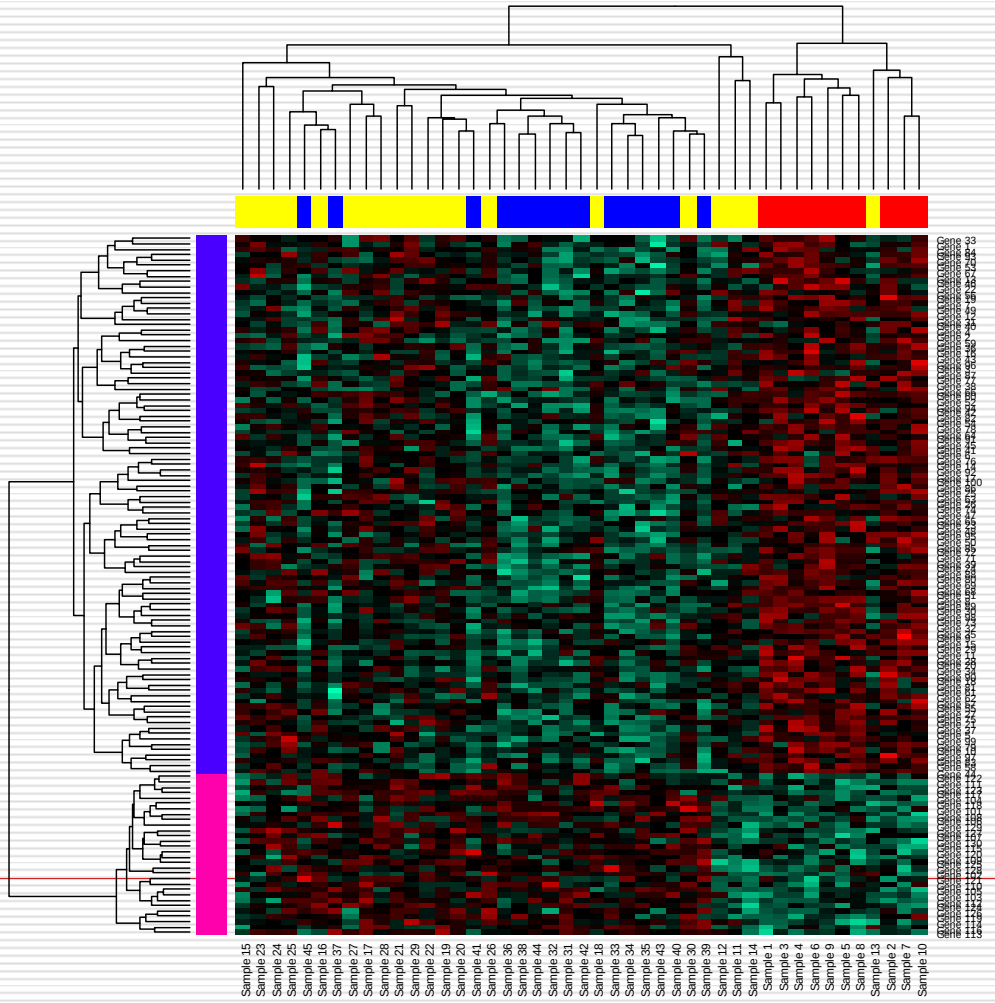
Clustering and Information Mining in R

- The basic hierarchical clustering functions in R are hclust() from the stats package, and agnes() and diana() from the cluster package.
- Dist(x, method = “correlation“) (amap package): Computes distances between the rows of x.
- d1 <- Dist(t(mm),method="correlation"); hr <- hclust(d, method = "complete", members=NULL)
- The generated tree can be plotted with the plot() function or cutplot.dendrogram {Heatplus package}
- **str(as.dendrogram(hr))** # Prints dendrogram structure as text.

Plot a Dendrogram



Heatmap Plots



Heatplus