CSE 5559 – Translational Bioinformatics
Week 3
An Introduction to R & Bioconductor
Bioconductor

✓ Open source & open development software project

✓ Analysis of biomedical and genomic data.
The Open Source Kool-Aid
Salient Essentials
Code Unit - Packages

- General infrastructure
  Biobase, Biostrings, biocViews
- Annotation:
  annotate, annaffy, biomaRt, AnnotationDbi \(\rightarrow\) data packages.
- Graphics/GUIs:
  geneplotter, hexbin, limmaGUI, exploRase
- Pre-processing:
  affy, affycomp, oligo, makecdfenv, vsn, gcrm, limma
- Differential gene expression:
  genefilter, limma, ROC, siggenes, EBArrays, factDesign
- GSEA/Hypergeometric Testing
  GSEABase, Category, GOstats, topGO
- Graphs and networks:
  graph, RBGL, Rgraphviz
- Flow Cytometry:
  flowCore, flowViz, flowUtils
- Protein Interactions:
  ppiData, ppiStats, ScISI, Rintact
- Sequence Data:
  Biostrings, ShortRead, rtracklayer, IRanges, GenomicFeatures, VariantAnnotation
- Other data:
  xcms, DNAcopy, PROcess, aCGH, rsbml, SBMLR, Rdisop
Literate Programming - Vignette

• Not a Manual

• Use of software

The BioConductor Portal

bioconductor.org
Main Resource Categories

Bioconductor.org/packages/release/BiocViews.html

• Software
• Annotation Data
• Experiment Data
ExperimentData Containers

• Self documenting representations of complex experimental data
• single R object – provenance/details.

• Examples
  – Neve2006
  – ALL
  – CLL
  – golubEsets
  – Affydata
Annotation Data Methods

Metadata package hgu95av2 mappings between different gene IDs for this chip.

- Realational Database table.
- SQLite – sqlite.org
- Assemble and process genomic annotation data from public repositories.
- Build annotation data packages.
- Associate experimental data in real time to biological metadata from web databases such as GenBank, GO, KEGG, Entrez Gene, and PubMed.
- Process and store query results: e.g., search PubMed abstracts.
- Generate HTML reports of analyses.

**Gene Information**

- **AffyID**: 41046_s_at
- **GENENAME**: zinc finger protein 261
- **ENTREZID**: 9203
- **MAP**: Xq13.1
- **ACCNUM**: X95808
- **PMID**: 10486218, 9205841, 8817323
- **GO**: GO:0003677, GO:0007275, GO:0016021
## Software – Packages

- **General infrastructure**  
  Biobase, Biostrings, biocViews
- **Annotation:**  
  annotate, annaffy, biomaRt, AnnotationDbi → data packages.
- **Graphics/GUIs:**  
  geneplotter, hexbin, limmaGUI, exploRase
- **Pre-processing:**  
  affy, affycomp, oligo, makecdfenv, vsn, gcrm, limma
- **Differential gene expression:**  
  genefilter, limma, ROC, siggenes, EBArrays, factDesign
- **GSEA/Hypergeometric Testing**  
  GSEABase, Category, GOstats, topGO
- **Graphs and networks:**  
  graph, RBGL, Rgraphviz
- **Flow Cytometry:**  
  flowCore, flowViz, flowUtils
- **Protein Interactions:**  
  ppiData, ppiStats, ScISI, Rintact
- **Sequence Data:**  
  Biostrings, ShortRead, rtracklayer, IRanges, GenomicFeatures, VariantAnnotation
- **Other data:**  
  xcms, DNAcopy, PROcess, aCGH, rsbml, SBMLR, Rdisop
Those Packages

- Bioconductor software consists of R add-on packages
  - Structured collection of code (R, C, or other)
  - Documentation
  - Data
  - Perform specific types of statistical/graphical analyses

- Examples
  - affy, cluster, graph, hexbin
Not Just Code

• **Data packages:**
  – Metadata: mappings between different gene identifiers (e.g., AffyID, GO, LocusID, PMID), CDF (chip definition file) and probe sequence information (e.g., Affy arrays)
    • hgu95av2, GO, KEGG.
  – Experimental data: code, data, and documentation for specific experiments or projects.
    • yeastCC: Spellman et al. (1998) yeast cell cycle.
    • golubEsets: Golub et al. (2000) ALL/AML data.

• **Course packages:** code, data, documentation, and labs for the instruction of a particular course.
  – EMBL course packages
Code - R Script(s)

stuff<-read.csv(file.choose(),header=TRUE)
# or full path to filename in place of file.choose() function
attach(stuff)
stuff$Year <- stuff$Year * rep(-1, 31)
# years are BC, so make them negative

library(car)
svg("r-transparent.svg",bg="transparent",width=5,height=5)
scatterplot(Word.Count~Year)
dev.off()
# file will be saved in working directory (no screen display)
One Case Study
Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring
T. R. Golub et al.
Science 286, 531 (1999);
DOI: 10.1126/science.286.5439.531

Clinical Biopsy

Integrated Biomarkers

Patient Stratification
Goal - Stratification
Acute Leukemias - Histology

• Non-Hodgkin’s lymphomas,
• Childhood “small round blue cell tumors”
  – neuroblastomas,
  – rhabdomyosarcoma,
  – Ewing’s sarcoma
  – Etc.
Other Subtypes?

- Class discovery - unrecognized tumor subtypes.
- Class prediction - assignment of particular tumor samples to already-defined classes.
- Class - Reflect current states or future outcomes.
- Difficult Task – Given heterogeneity of outcomes.
Class Prediction by Gene Expression Monitoring

T. R. Golub,1,2*† D. K. Slonim,1† P. Tamayo,1 C. Huard,1
M. Gaasenbeek,1 J. P. Mesirov,1 H. Coller,1 M. L. Loh,2
J. R. Downing,3 M. A. Caligiuri,4 C. D. Bloomfield,4
E. S. Lander1,5*

Although cancer classification has improved over the past 30 years, there has been no general approach for identifying new cancer classes (class discovery) or for assigning tumors to known classes (class prediction). Here, a generic approach to cancer classification based on gene expression monitoring by DNA microarrays is described and applied to human acute leukemias as a test case. A class discovery procedure automatically discovered the distinction between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) without previous knowledge of these classes. An automatically derived class predictor was able to determine the class of new leukemia cases. The results demonstrate the feasibility of cancer classification based solely on gene expression monitoring and suggest a general strategy for discovering and predicting cancer classes for other types of cancer, independent of previous biological knowledge.
Also Available @

http://www.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43

http://www.broadinstitute.org/cancer/software/genepattern/datasets/
The Premise

• Class discovery to automatically distinguish between
  – acute myeloid leukemia: AML
  – acute lymphoblastic leukemia: ALL

• No knowledge of these classes

• Genetic causes known - specific chromosomal translocations

• Use microarrays
The Data

- 38 bone marrow samples (27 ALL, 11 AML)
- Obtained from patients at time of diagnosis
- RNA from bone marrow mononuclear cells
- Affymetrix microarrays
- Probes for 6817 human genes
The Methods

Schematic illustration of methodology. (a) Strategy for cancer classification. Tumor classes may be known a priori or discovered on the basis of the expression data by using Self-Organizing Maps (SOMs) as described in the text. Class Prediction involves assignment of an unknown tumor sample to the appropriate class on the basis of gene expression pattern. This consists of several steps: neighborhood analysis to assess whether there is a significant excess of genes correlated with the class distinction, selection of the informative genes and construction of a class predictor, initial evaluation of class prediction by cross-validation, and final evaluation by testing in an independent data set.
Neighborhood Analysis. The class distinction is represented by an 'idealized expression pattern c, in which the expression level is uniformly high in class 1 and uniformly low in class 2. Each gene is represented by an expression vector, consisting of its expression level in each of the tumor samples. In the figure, the dataset consists of 12 samples comprised of 6 AMLs and 6 ALLs. Gene g1 is well correlated with the class distinction, while g2 is poorly correlated. Neighborhood analysis involves counting the number of genes having various levels of correlation with c. The results are compared to the corresponding distribution obtained for random idealized expression patterns c*, obtained by randomly permuting the coordinates of c. An unusually high density of genes indicates that there are many more genes correlated with the pattern than expected by chance. The precise measure of distance and other methodological details are described in notes (16,17) and on our web site.
The prediction of a new sample is based on 'weighted votes' of a set of informative genes. Each such gene $g_i$ votes for either AML or ALL, depending on whether its expression level $x_i$ in the sample is closer to $\mu_{\text{AML}}$ or $\mu_{\text{ALL}}$ (which denote, respectively, the mean expression levels of AML and ALL in a set of reference samples). The magnitude of the vote is $w_i v_i$, where $w_i$ is a weighting factor that reflects how well the gene is correlated with the class distinction and $v_i = |x_i - (\mu_{\text{AML}} + \mu_{\text{ALL}})/2|$ reflects the deviation of the expression level in the sample from the average of $\mu_{\text{AML}}$ and $\mu_{\text{ALL}}$. The votes for each class are summed to obtain total votes $V_{\text{AML}}$ and $V_{\text{ALL}}$. The sample is assigned to the class with the higher vote total, provided that the prediction strength exceeds a predetermined threshold. The prediction strength reflects the margin of victory and is defined as $(V_{\text{win}}-V_{\text{lose}})/(V_{\text{win}}+V_{\text{lose}})$, where as $V_{\text{win}}$ and $V_{\text{lose}}$ are the respective vote totals for the winning and losing classes. Methodological details are described in the paper (notes 19,20).
Prediction strengths. The scatterplots show the prediction strengths (PS) for the samples in cross-validation (left) and on the independent sample (right). Median PS is denoted by a horizontal line. Predictions with PS below 0.3 are considered as uncertain.
Genes distinguishing ALL from AML. The 50 genes most highly correlated with the ALL/AML class distinction are shown. Each row corresponds to a gene, with the columns corresponding to expression levels in different samples. Expression levels for each gene are normalized across the samples such that the mean is 0 and the standard deviation is 1. Expression levels greater than the mean are shaded in red, and those below the mean are shaded in blue. The scale indicates standard deviations above or below the mean. The top panel shows genes highly expressed in ALL, the bottom panel shows genes more highly expressed in AML. Note that while these genes as a group appear correlated with class, no single gene is uniformly expressed across the class, illustrating the value of a multi-gene prediction method.
Code ...


http://web.cse.ohio-state.edu/~raghu/teaching/CSE5599-BMI7830/Scripts/Class-Chapter7-Script.R
GenePattern

http://www.broadinstitute.org/cancer/software/genepattern/tutorial/gp_tutorial
Construct One?

Golub Case Study – golubEsets
Class-chapter7-Script.R

(Text) Chapter 7
Leverage Biobase – The Base

• Used by many other packages

• Contains standardized data structures for genomic data


• `expressionSet` class combines disparate sources of information into single convenient structure.
For CS Types 😊
Goal – Extract & Analyze

golubEsets
esp. ExperimentData
Constructing a R script

The Opening Line
To get going

• `source(http://www.bioconductor.org/biocLite.R)`
• `biocLite("whatever")`
> source("http://www.bioconductor.org/biocLite.R")
> biocLite(golubEsets)
> require(golubEsets) # attach package
Or
> library(golubEsets)
> data(Golub_Merge) #object
> experimentData(Golub_Merge)
Objects & Classes

• Golub_Merge is an object

• A class provides a software abstraction of a real world object.

• Classes are defined in terms of slots which contain the relevant data.

• An object is an instance of a class.
An ExpressionSet can be manipulated

- subsetted,
- copied

Input or output from other Bioconductor functions
# Class Expression Set

Class "ExpressionSet" [package "Biobase"]

**Slots:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>experimentData</td>
<td>MIAME</td>
</tr>
<tr>
<td>assayData</td>
<td>AssayData</td>
</tr>
<tr>
<td>phenoData</td>
<td>AnnotatedDataFrame</td>
</tr>
<tr>
<td>featureData</td>
<td>AnnotatedDataFrame</td>
</tr>
<tr>
<td>annotation</td>
<td>character</td>
</tr>
<tr>
<td>protocolData</td>
<td>AnnotatedDataFrame</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>.<strong>classVersion</strong></td>
<td>Versions</td>
</tr>
</tbody>
</table>

**Extends:**

- Class "eSet", directly
- Class "VersionedBiobase", by class "eSet", distance 2
- Class "Versioned", by class "eSet", distance 3
### expressionSet class

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exprs</td>
<td>Matrix of expression measures, genes x samples</td>
</tr>
<tr>
<td>se.exprs</td>
<td>Matrix of SEs for expression measures, genes x samples</td>
</tr>
<tr>
<td>phenoData</td>
<td>Sample level covariates, instance of class phenoData</td>
</tr>
<tr>
<td>annotation</td>
<td>Name of annotation data</td>
</tr>
<tr>
<td>description</td>
<td>MIAME information</td>
</tr>
<tr>
<td>notes</td>
<td>Any notes</td>
</tr>
</tbody>
</table>

Processed Affymetrix or spotted array data

Processed Affymetrix or spotted array data

Processed Affymetrix or spotted array data

Processed Affymetrix or spotted array data
The Object Golub_Merge

ExpressionSet (storageMode: lockedEnvironment)

assayData: 7129 features, 72 samples

element names: exprs

protocolData: none

phenoData
  sampleNames: 39 40 ... 33 (72 total)
  varLabels: Samples ALL.AML ... Source (11 total)
  varMetadata: labelDescription

featureData: none

experimentData: use 'experimentData(object)'

pubMedIds: 10521349

Annotation: hu6800
Experimenter name: Golub TR et al.
   Laboratory: Whitehead
   Contact information:
   Title: ALL/AML discrimination
   URL: www-genome.wi.mit.edu/mpr/data_set_ALL_AML.html
   PMIDs: 10521349
   Abstract: A 133 word abstract is available. Use 'abstract' method.
Class-script1.R
Defining A ExpressionSet

Expression data from microarray experiments
- assayData: methods to access different data components,
- phenoData: samples in the experiment
- featureData, annotation: features on the chip or technology used for the experiment
- protocolData: protocol for processing each sample
- experimentData: flexible structure to describe the experiment.
Although cancer classification has improved over the past 30 years, there has been no general approach for identifying new cancer classes (class discovery) or for assigning tumors to known classes (class prediction). Here, a generic approach to cancer classification based on gene expression monitoring by DNA microarrays is described and applied to human acute leukemias as a test case. A class discovery procedure automatically discovered the distinction between acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL) without previous knowledge of these classes. An automatically derived class predictor was able to determine the class of new leukemia cases. The results demonstrate the feasibility of cancer classification based solely on gene expression monitoring and suggest a general strategy for discovering and predicting cancer classes for other types of cancer, independent of previous biological knowledge.
More

dim(exprs(Golub_Merge))
[1] 7129 72

dim(pData(Golub_Merge))
[1] 72 11

featureNames(Golub_Merge)[1001:1010]
[1] “HG_4390_at”

sampleNames(Golub_Merge)[1:10]
[1] “39” “40” …
## AML Data

```r
table(Golub_Merge$ALL.AML)
```

<table>
<thead>
<tr>
<th></th>
<th>ALL</th>
<th>AML</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>
Annotation

annotation(Golub_Merge)
[1] “hu6800”
biocLite(hu6800.db)
library(hu6800.db)
objects (package:hu6800.db)
What is in it?

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>&quot;hu6800&quot;</td>
<td>&quot;hu6800_dbconn&quot;</td>
</tr>
<tr>
<td>[4]</td>
<td>&quot;hu6800_dbInfo&quot;</td>
<td>&quot;hu6800_dbschema&quot;</td>
</tr>
<tr>
<td>[7]</td>
<td>&quot;hu6800ACCNUM&quot;</td>
<td>&quot;hu6800ALIAS2PROBE&quot;</td>
</tr>
<tr>
<td>[10]</td>
<td>&quot;hu6800CHRLENGTHS&quot;</td>
<td>&quot;hu6800CHRLOC&quot;</td>
</tr>
<tr>
<td>[13]</td>
<td>&quot;hu6800ENSEMBL&quot;</td>
<td>&quot;hu6800ENSEMBL2PROBE&quot;</td>
</tr>
<tr>
<td>[16]</td>
<td>&quot;hu6800ENZYME&quot;</td>
<td>&quot;hu6800ENZYME2PROBE&quot;</td>
</tr>
<tr>
<td>[19]</td>
<td>&quot;hu6800GO&quot;</td>
<td>&quot;hu6800GO2ALLPROBES&quot;</td>
</tr>
<tr>
<td>[22]</td>
<td>&quot;hu6800MAP&quot;</td>
<td>&quot;hu6800MAPCOUNTS&quot;</td>
</tr>
<tr>
<td>[25]</td>
<td>&quot;hu6800ORGANISM&quot;</td>
<td>&quot;hu6800ORGPKG&quot;</td>
</tr>
<tr>
<td>[28]</td>
<td>&quot;hu6800PATH2PROBE&quot;</td>
<td>&quot;hu6800PFAM&quot;</td>
</tr>
<tr>
<td>[31]</td>
<td>&quot;hu6800PMID2PROBE&quot;</td>
<td>&quot;hu6800PROSITE&quot;</td>
</tr>
<tr>
<td>[34]</td>
<td>&quot;hu6800SYMBOL&quot;</td>
<td>&quot;hu6800UNIGENE&quot;</td>
</tr>
</tbody>
</table>
Digging in

ls(hu6800SYMBOL)[1:1]
"A28102_at"

mget(ls(hu6800SYMBOL[1:1], hu6800SYMBOL)
$A28102_at
[1] "GABRA3"
SQL

hh=org.Hs.eg_dbconn()
dListTables(hh)
SQL = paste( "select * from gene_info inner join genes using (_id)", " where gene_name like '%sarcoma%oncogene%’")
sops = dbGetQuery(hh,SQL)
dim(sops)
sops$symbol
BoxPlot

sops[13,]
get("4067", revmap(hu6800ENTREZID))

boxplot(exprs(Golub_Merge)["M16038_at",] ~ Golub_Merge$ALL.AML, col = c("darkgreen", "yellow"))
gene LYN
gene SRC

sops[19,]

get("6194", revmap(hu6800ENTREZID))

boxplot(exprs(Golub_Merge)["M77232_rnal_at",]~Golub_Merge$ALL.AML, col=c("darkgreen", "yellow"))
The Prediction?

Class Discovery

Expression Data

Assess Gene-Class Correlation: Neighborhood Analysis

Build Predictor

Test Predictor by Cross-validation

Test Predictor on Independent Dataset

Known Classes
Learning

> biocLite("MLInterfaces")
> GMF=nsfilter(Golub_Merge, var.cutoff=0.9)[[1]]
> rf1=MLearn(ALL.AML ~ ., data=GMF,randomForestI,trainInd=xvalSpec("NOTEST"),importance=TRUE)
> Robject(rf1)

Special container for summarization

Error rate: 4.17%
Alas

I HAS A SAD.
But ..

Wake up with DETERMINATION.
Go to bed with SATISFACTION!
Another Case Study

Summary – Container, ExpressionSet

• Data Management Idiom

• Data accessible through reference to variable X

• $X[R, S] = \text{selection of reporters on expression } R \text{ and samples } S$
Summary - Bioconductor

• Development of Repository

• Development of Data Resources to explore and create procedures to investigate genmoic data

• Establishment of standard practices
Other packages

• graph
• RGBL (network) structures
• GO.db – gene ontology,
• GO.KEGG.db – pathway
• Biostrings
• Ggtools – genotype and expression data
Canadian Bioinformatics Workshops

boris.steipe@utoronto.ca
R

Wexner Medical Center
Latest version: 3.1.1

support of very large datasets.
R Studio

```r
# Example R code

# Define a function
MySqrt <- function(y) {
  x <- y/2
  while (abs(x^2 - y) > 1e-10) {
    x <- (x + y/x)/2
  }
  x
}

# Use the function

# Plot the function
plot(1:7, MySqrt(1:7))
```
Baby Steps

1: Start R
2: Set the working directory.
   getwd()
   setwd("/path/and/name/of/your/directory")
3: Check that everything is correct.
   list.files()
> help

> help (pi)
> ?pi
> ?sqrt
> ?Special
Help searches

> help.search("trigonometry")
> ??input
More help

> apropos(plot)
Even More

+ A table of all available packages: [http://cran.r-project.org/](http://cran.r-project.org/)

+ R Manuals are at: [http://cran.r-project.org/manuals.html](http://cran.r-project.org/manuals.html)
Expressions

R evaluates expressions.

```r
> 2+2
[1] 4
> exp(-2)
[1] 0.1353353
> pi
[1] 3.141593
> sin (2*pi)
[1] -2.449294e-16
> 0/0
[1] NaN
```
Strings

R has a string datatype

```
> "Hello"
[1] "Hello"
> x <- paste("Hello", "World")
x
[1] "Hello World"
> m <- gregexpr("(\b\w{2})", x, perl=T)
> y<-regmatches(x,m)
y
[[1]]
[1] "He" "Wo"
> paste(y[[1]], collapse="")
y
[1] "HeWo"
```
Assignments

> x <- 1/sqrt(4)
> y <- sin(pi/6)
> x+y

[1] 1

R community prefers "<-" to "="

"<-" is more general

"=" with "==" are not same!
Vectors

Lists (vectors) can be created with the "c" operator (concatenate).

```r
> Weight <- c(60,72,75,90,95,72)
> Weight[1]
[1] 60
> Weight[2]
[1] 72
> Weight
[1] 60 72 75 90 95 72
> Height <- c(1.75,1.80,1.65,1.90,1.74,1.91)
> BMI <- Weight/Height^2 # vector based operation
> BMI
```
Vector Operations

Element-wise operation

```r
> x <- 1:5
> x+2
[1] 3 4 5 6 7

> y <- 6:2
> x+y
[1] 7 7 7 7 7
```
Vector Types

Can be of type:

- numeric
- character
- logical.

```r
> x <- c(1, 5, 8)  # Numeric
> x
[1] 1 5 8
> x <- c(TRUE, TRUE, FALSE, TRUE)  # Logical
> x
[1] TRUE TRUE FALSE TRUE
> x <- c("Hello","world")  # Character
> x
[1] "Hello" "world"
> x <- c(1,TRUE,"Thursday")  # Mixed
> x
[1] "1" "TRUE" "Thursday"
```
Special Dudes

NaN == Not-a-Number
Inf, -Inf == Google
NA == Not Available

```r
> Weight[5] <- NA
> mean(Weight)
[1] NA
> mean(Weight, na.rm=TRUE)
[1] 73.8
```
Matrices & Arrays

```r
> x<-1:12
> x
[1]  1  2  3  4  5  6  7  8  9 10 11 12
> length(x)
[1] 12
> dim(x)
NULL
> dim(x)<-c(3,4)
> x
[1,]  1  4  7 10
[2,]  2  5  8 11
[3,]  3  6  9 12
```

```r
> a<-matrix(1:12,nrow=3,byrow=TRUE)
> a
[1,]  1  2  3  4
[2,]  5  6  7  8
[3,]  9 10 11 12
> a<-matrix(1:12,nrow=3,byrow=FALSE)
> a
[1,]  1  4  7 10
[2,]  2  5  8 11
[3,]  3  6  9 12
> rownames(a)<-c("A","B","C")
> a
A  1  4  7 10
B  2  5  8 11
C  3  6  9 12
> colnames(a)<-c("1","2","x","y")
> a
 1 2 x y
A 1 4 7 10
B 2 5 8 11
C 3 6 9 12
```
Matrices & Arrays

Matrices can also be formed by "glueing" rows and columns using `cbind` and `rbind`.

```r
> x1 <- 1:4  # Define three vectors
> x2 <- 5:8
> y1 <- c(3,9)
> MyMatrix <- rbind(x1,x2)
> MyMatrix
x1   1   2   3   4
x2   5   6   7   8
> MyNewMatrix <- cbind(MyMatrix,y1)
> MyNewMatrix
y1
x1 1 2 3 4  3
x2 5 6 7 8  9
```
Factors

In R categorical variables are referred to as factors

A factor has a set of levels – not a partial or total order

```r
> Pain <- c(0,3,2,2,1)
> SevPain <- as.factor(c(0,3,2,2,1))
> levels(SevPain) <- c("none","mild","medium","severe")
> is.factor(SevPain)
[1] TRUE
> is.vector(SevPain)
[1] FALSE
```
Lists

Combine objects of different kinds/sizes

Components are named with arguments used

Components extracted with double bracket operator \[[\]\

Named components accessed with the "$" separator.

```r
> A<-c(31,32,40)
> S<-as.factor(c("F","M","M","F"))
> L<-c("London","School")
> MyFriends<-list(age=A,sex=S,meta=L)
> MyFriends
$age
[1] 31 32 40
$sex
[1] F M M F
Levels: F M
$meta
[1] "London" "School"
> MyFriends[[2]]
[1] 31 32 40
> MyFriends$age
[1] 31 32 40
```
Data frames

A matrix or a "set" of data

List of related vectors and/or factors of the same length
data in the same position come from the same experiment

```
> Probands <- data.frame(age=c(31,32,40,50),sex=S)
> Probands
  age sex
1 31   F
2 32   M
3 40   M
4 50   F
> Probands$age
[1] 31 32 40 50
```

Why do we need data frames if they do the same as a list?

More efficient storage, and indexing! R's `read...()` functions return data frames.
Names

Use: Give explicit names to variables. Names can be used for indexing.

```r
> x <- 1:3
> names(x)
NULL
> names(x) <- c("a", "b", "c")
> x
 a b c
1 2 3
> names(Probands)
[1] "age" "sex"
> names(Probands) <- c("age", "gender")
> names(Probands)[1] <- c("Age")
```
Indexing (Extracting) is a great way to directly assess elements of interest.

> # Indexing a vector
> Pain <- c(0,3,2,2,1)
> Pain[1]
[1] 0
> Pain[2]
[1] 3
> Pain[1:2]
[1] 0 3
> Pain[c(1,3)]
[1] 0 2
> Pain[-5]
[1] 0 3 2 2

> # Indexing a matrix
> MyNewMatrix
[,1] 1
[1] 1
> MyNewMatrix[1,]
y1
1 2 3 4 3
> MyNewMatrix[,-2]
y1
1 3 4 3
x2 5 7 8 9

> # Indexing a list
> MyFriends
$meta
[1] "London" "School"
> MyFriends[[3]]
[1] "London" "School"
> MyFriends[[3]][1]
[1] "London"

> # Indexing a data frame
> Probands
Age gender
1 31 F
> Probands[2,]
Age gender
2 32 M
Indexing by name

Names can also be used to index an R object.

```r
> MyFriends$age
[1] 31 32 40
> MyFriends["age"]
$age
[1] 31 32 40
> MyFriends[["age"]]
[1] 31 32 40
> Probands["Age"]
  Age
1  31
2  32
3  40
4  50
> Probands[[1]]
  Age
1  31
2  32
3  40
4  50
> Probands[[1]]
[1] 31 32 40 50
```
Conditional Indexing

Indexing can be conditional on another variable.

```r
> Pain; Fpain
[1] 0 3 2 2 1
[1] none  severe medium medium mild
Levels: none mild medium severe
> Age <- c(45,51,45,32,90)
> Pain[Fpain=="medium" | Fpain=="severe"]
[1] 3 2 2
> Pain[Age>32]
[1] 0 3 2 1
```
Functions

\[ \log(x), \text{plot}(\text{Weight,Height}). \]

\[ \text{plot}(\text{Weight, Height}) \text{ at } \]

\[ ?\text{plot}. \]

\[ \text{plot}(\text{Weight, Height, col=1}). \]
Creating Functions

Oracle <- function() {
  WiseWords <- c(
    "Joy",
    "Plan",
    "Disappear",
    "Perhaps",
    "Sorrow",
    "Hope",
    "Change"
  )
  n <- sample(WiseWords, 1)
  return(n)
}

> Oracle()
[1] "Disappear"
Creating Functions

```
MySqrt<-function(y) {
  x<-y/2
  while (abs(x*x - y) > 1e-10) {
    x <- (x + y/x)/2
  }
  x
}
```
Data Input

`read.table()`

Download the sample data to your local directory...

```r
> GvHD <- read.table("GvHD.txt", header=TRUE)
> GvHD[1:10,]

          FSC.Height SSC.Height   CD4.FITC  CD8.B.PE  CD3.PerCP  CD8.APC
     1       321         199       308       220        157       339
     2       303         210       319       271        223       350
     3       318         170       215       148        119       221
     4       202          49       104        49        284       178
     5       353         248       262       167        144       156
     6       192          68       423        97        344       113
     7       322         225       236       214        141       209
     8       350         152       258        82        253       205
     9       351         223       286       128        172       220
    10       269          78       169       289        224       537
```
Writing?

write.table(ge,"mygenes.txt)
Conditional Statements

# if statement
> x <- -2
> if(x>0) {
+   print(x)
+ } else {
+   print(-x)
+ }
[1] 2
>
> if(x>0) {
+   print(x)
+ } else if(x==0) {
+   print(0)
+ } else {
+   print(-x)
+ }
[1] 2
Loops

# for loop
n <- 1000000
x <- rnorm(n,10,1)
y <- x^2
y <- rep(0,n)
for (i in 1:n) {
y[i] <- sqrt (x[i])
}

# while loop
Counter <- 1
while (Counter <= n) {
y[Counter] <- sqrt(x[Counter])
Counter <- Counter+1
}
Graphics

plot(x,y)
> library(survival)
Loading required package: splines
> library(samr)
Error in library(samr) : there is no package called 'samr'
  ➢ biocLite("samr")

trying URL 'http://probability.ca/cran/bin/macosx/leopard/contrib/2.13/R.methodsS3_1.2.1.tgz'
Content type 'application/x-gzip' length 47709 bytes (46 Kb)
opened URL
=================================================================
downloaded 46 Kb

[...]
The downloaded packages are in
  /var/folders/dq/dqPEEPbFGFWs6MKN40ApRU+++Tl/-Tmp--/RtmpNDvKDP/downloaded_packages
> library(samr)
Loading required package: impute
Loading required package: matrixStats
Loading required package: R.methodsS3
matrixStats v0.2.2 (2010-10-06) successfully loaded. See ?matrixStats for help.
Other packages

- graph
- RGBL (network) structures
- GO.db – gene ontology,
- GO.KEGG.db – pathway
- Biostrings
- Ggtools – genotype and expression data
THE END

HD HAPPY

Wexner Medical Center