Analysis of Microarray Data

Keith Baggerly and Bradley Broom
Department of Bioinformatics and Computational Biology
UT M. D. Anderson Cancer Center
kabagg@mdanderson.org
bmbroom@mdanderson.org

17 September 2009
Lecture 6: Sweave, More on R, and Affy. Arrays

- The Reproducibility Problem
- Installing \TeX
- Introductory \LaTeX
- Writing Documented R Analyses
- R Revisited: Beyond the Matrix
- Reading Data Into R
- Obtaining extra R packages
- Bioconductor Packages
The Reproducibility Problem

1. Researcher contacts analyst: “I just read this interesting paper. Can you perform the same analysis on my data?”

2. Analyst reads paper. Finds algorithms described by biologists in English sentences that occupy minimal amount of space in the methods section.

3. Analyst gets public data from the paper. Takes wild guesses at actual algorithms and parameters. Is unable to reproduce reported results.

4. Analyst considers switching to career like bicycle repair, where reproducibility is less of an issue.
Alternate Forms of the Same Problem

1. Remember that microarray analysis you did six months ago? We ran a few more arrays. Can you add them to the project and repeat the same analysis?

2. The statistical analyst who looked at the data I generated previously is no longer available. Can you get someone else to analyze my new data set using the same methods (and thus producing a report I can expect to understand)?

3. Please write/edit the methods sections for the abstract/paper/grant proposal I am submitting based on the analysis you did several months ago.
The Code/Documentation Mismatch

Most of our analyses are performed using R. We can usually find an R workspace in a directory containing the raw data, the report, and one or more R scripts.

There is no guarantee that the objects in the R workspace were actually produced by those R scripts. Nor that the report matches the code. Nor the R objects.

Because R is interactive, unknown commands could have been typed at the command line, or the commands in the script could have been cut-n-pasted in a different order.

This problem is even worse if the software used for the analysis has a fancy modern GUI. It is impossible to document how you used the GUI in such a way that someone else could produce the exact same results—on the same data—six months later.
The Solution: Sweave

Literate programming is an approach that embeds small program fragments within an otherwise high-quality document.

Sweave is a literate programming framework for R.

This talk was prepared using Sweave. So was this standard report.

\[ \text{Sweave} = \text{R} + \LaTeX. \]

Once you know both R and \LaTeX, then the thirty-second version of this talk takes only two slides.

First, we take a few moments to learn \LaTeX. (You already know R.)
LaTeX Document Preparation System

LaTeX is a document preparation system for high-quality typesetting.

LaTeX is not a word processor.

LaTeX separates document content (written by author) from layout (written by document designers).

You can read more about LaTeX at the website for the Comprehensive TeX Archive Network (CTAN): http://www.ctan.org.
CTAN Website

Welcome to CTAN!

The Comprehensive TeX Archive Network is the authoritative collection of materials related to the TeX typesetting system.

Are you new here?

You can learn about TeX, LaTeX, and friends. Then our introduction page will get you started.

Can we be of service?

If you’re looking for something TeX related then you’ve come to the right place.

- Know exactly what you want? You can search for a file by name.
- Know sort-of what you want? You can search package description or documentation.
- Just looking? You can browse through our directories.

Help yourself

http://www.ctan.org/starter.html
Starting out with TeX, LaTeX, and friends

Do you want to begin working with the TeX typesetting system? Most people start out by downloading free versions of the needed software, and a tutorial. This page gets you to the most popular choices.

Step one: Get a distribution

You first need a collection of the software. Such a collection is called a distribution, and comes with TeX, LaTeX, pdfTeX, ConTeXt, BibTeX, and everything else that will help you to perform TeX’s magic on your computer. Each distribution also comes with programs that run only on one type of computer platform, so make your choice from the list below.

- Windows The most popular choice here is the MikTeX distribution, which lets you easily manage TeX packages. (Many people advise beginners to get the proTeXt bundling of MikTeX, which lets you install by using a .pdf file with links so you can read about your options and then click on the right one. And it includes other components that help you work with your TeX system.)
- Unix-type systems, including GNU/Linux The best choice here is TeX Live, which contains many packages and programs. It is freely available over the Internet or on CD/DVD; see the web page for details. Note that most systems have TeX as an installation option, so
Installing \TeX

The standard version of \TeX or \LaTeX for Windows is MiKTeX, which is available at http://www.miktex.org. The current version is 2.8.

Follow the MiKTeX link, and then choose Download MiKTeX 2.8 from the panel on the left.

I used the “MikTeX 2.8 Net Installer” because it’s relatively small (3 MB), but ran into some problems adding additional packages.

It might be better to download the “Basic MiKTeX 2.8” installer (92 MB).

Keep track of where you save this file (your desktop will work just fine) and then double-click on the resulting icon to start the installation.

CTAN states that the standard version of \LaTeX for Macintosh computers is MacTeX. (Since I have never installed this version, you will have to figure out how to install it yourself . . . .)
The MiKTeX Installer: License

COPYING CONDITIONS FOR MiKTeX

To the best of our knowledge, all software in this distribution is freely redistributable (libre, that is, not necessarily gratis) within the Free Software Foundation's definition and Debian Software Guidelines. If you find any non-free files included, contact us (references given below).

That said, MiKTeX has neither a single copyright holder nor a single distribution organization.

I accept the MiKTeX copying conditions.
The MiKTeX Installer: Download

![MiKTeX Installer](image)

**Task**
Choose the task you want carried out.

- **Download MiKTeX**
  MiKTeX will be downloaded into a directory on your computer.
  - [Connection Settings...]

- **Install MiKTeX**
  MiKTeX will be installed from a directory on your computer.
The MiKTeX Installer: Package Set

**Basic MiKTeX**
A directory will be created which contains the basic MiKTeX distribution.
This is the recommended option. MiKTeX can be configured to install missing packages automatically (in the course of use).

**Complete MiKTeX**
A directory will be created which contains the complete MiKTeX distribution.
Don’t use this option unless you have a fast and reliable Internet connection.
The MiKTeX Installer: Download Source

Choose an up-to-date US mirror.
The MiKTeX Installer: Distribution Directory

Remember this directory for later.
The MiKTeX Installer: Review Settings

MiKTeX 2.8 Net Installer

Information
Review the settings.

MiKTeX Setup Wizard has enough information to start the task. If you want to review or change any of the settings, click Back. If you are satisfied with the settings, click Start.

Download basic packages from
ftp://gentoo.chem.wisc.edu/tex-archive/systems/win32/miktex/tm/packages

to
C:\Documents and Settings\Administrator\Desktop\MiKTeX 2.8 Setup

< Back  Start  Cancel
The MiKTeX Installer: Downloading
The MiKTeX Installer: Install

Rerun the installer, but this time choose install.
The MiKTeX Installer: Package Set

- Basic MiKTeX
  A directory will be created which contains the basic MiKTeX distribution.
  This is the recommended option. MiKTeX can be configured to install missing packages automatically (in the course of use).

- Complete MiKTeX
  A directory will be created which contains the complete MiKTeX distribution.
  Don't use this option unless you have a fast and reliable Internet connection.
The MiKTeX Installer: Shared Install

Shared Installation
You have the option to share the MiKTeX installation with other users.

Install MiKTeX for:
- Anyone who uses this computer (all users)
- Only for: Administrator
The MiKTeX Installer: Distribution Directory

Use the directory you downloaded the distribution to earlier.
The MiKTeX Installer: Installation Directory

Best if you can avoid spaces in this path. Remember this path.
The MiKTeX Installer: Default Settings

Make sure you change these settings from the defaults.
The MiKTeX Installer: Install Review
The MiKTeX Installer: Installing at last

Executing
The main task is being executed.

Installing: ec

Overall progress

Loading package database...
starting package maintenance...
installation directory: "C:\Program Files\MiKTeX 2.8"
package repository: "C:\Documents and Settings\Administrator\Desktop\MiKTeX 2.8 Setup visiting repository "C:\Documents and Settings\Administrator\Desktop\MiKTeX 2.8 Setup"
repository type: local package repository

< Back  Next >  Cancel
Ghostscript and GSview

Goto http://www.ghostscript.com and download gs870w32.exe from either cs.wisc.edu or sourceforge.net.

Follow the links at the bottom of the ghostscript page to download GSview 4.9 (gsv49w32.exe).

These are both straight-forward installs ....
TeXnicCenter

Optionally, you can download and install TeXnicCenter.

TeXnicCenter is an integrated environment for creating LaTeX documents using Microsoft Windows.

- LaTeX specific editor with syntax highlighting, bracket matching, etc.
- Buttons for inserting predefined LaTeX snippets.
- Buttons for building and viewing document.

Unfortunately, TeXnicCenter does not know about Sweave, so we will still need to do some stuff manually.

Goto http://www.texniccenter.org/ and follow the links to download the TeXnicCenter installer.
The TeXnicCenter Installer: Welcome

Welcome to the TeXnicCenter Setup Wizard

This will install TeXnicCenter Version 1.0 Stable RC1 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.
The TeXnicCenter Installer: License

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA
Everyone is permitted to copy and distribute

- I accept the agreement
- I do not accept the agreement

< Back  Next >  Cancel
The TeXnicCenter Installer: Location

Best to avoid spaces in this path.
The TeXnicCenter Installer: Components

Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.

Typical (Recommended)

- Application Files (TeXnicCenter core components) 8.5 MB
- Help Files (Help for TeXnicCenter and LaTeX reference) 2.5 MB
- LaTeX Templates (Templates for creating new projects and documents) 0.2 MB

Current selection requires at least 13.0 MB of disk space.
The TeXnicCenter Installer: Start Menu

The TeXnicCenter Installer will prompt you to select a Start Menu folder where the program's shortcuts will be placed. You can either select a default folder by clicking Next, or you can click Browse to choose a different folder. You also have the option to not create a Start Menu folder by checking the box labeled "Don't create a Start Menu folder."
The TeXnicCenter Installer: Create Icon

Select Additional Tasks

Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing TeXnicCenter, then click Next.

- [ ] Create a desktop icon
- [ ] Add TeXnicCenter to the 'Send To' menu
The TeXnicCenter Installer: Ready

Setup is now ready to begin installing TeXnicCenter on your computer.

Destination location: C:\Program Files\TeXnicCenter

Setup type: Typical (Recommended)

Selected components:
- Application Files (TeXnicCenter core components)
- Help Files (Help for TeXnicCenter and LaTeX reference)
- LaTeX Templates (Templates for creating new projects and documents)

Start Menu folder: TeXnicCenter
TeXnicCenter: Config Wizard

Welcome to the Configuration Wizard.

The Configuration Wizard will help you creating the output profiles. In TeXnicCenter output profiles are necessary to generate printable files.

Be sure, that you have already installed the TeX distribution. If not, you should quit the wizard and install the TeX distribution and then restart this wizard.

Tips: If you are using the MiKTeX Distribution (www.miktex.org), TeXnicCenter will do the most part of the configuration for you.
TeXnicCenter: Config Wizard

Append `\miktex\bin` to the MiKTeX install path.
TeXnicCenter: Config Wizard
TeXnicCenter: Config Wizard

The following output profiles are about to be created:
- Latex -> DVI
- Latex -> PS
- Latex -> PDF
- Latex -> PS -> PDF

Click on 'Finish' to generate the listed profiles.
TeXnicCenter: Running

Did you know...

Welcome to TeXnicCenter!

We have some tips for you, about how to use LaTeX and especially TeXnicCenter.

Leave this dialog enabled for some time, until you think you know all features of TeXnicCenter. If you don't want to see the dialog anymore, you can uncheck the box in the lower left corner. You can popup this dialog every time by choosing the menu item Help | Tip of the day...
Introduction to \LaTeX

A \LaTeX source file consists of free format text interspersed with commands to the \LaTeX formatting engine. (It's important to use a text editor — not Word — to edit these files.)

Except for approximately 10 characters with special meaning to \LaTeX, the printable characters in the source file are copied to the output document.

The most important document structural component is the paragraph. Paragraphs are specified by inserting a blank line in the source file.

The formatting engine takes each paragraph in the input file, formats it nicely (for instance, by tweaking the space between words), and outputs it.

Normally, multiple spaces in the source file are equivalent to a single space, and multiple blank lines are equivalent to a single blank line.
Basic \LaTeX{} Commands

\LaTeX{} commands start with a single backslash (\) followed either by one or more letters or by a single non-letter.

A \LaTeX{} document begins with the \texttt{\documentclass} command, which tells \LaTeX{} the base document layout to use:

\texttt{\documentclass\{article\}}

Following the \texttt{\documentclass} command itself is a parameter enclosed in braces.

- If required, multiple parameters are separated by commas.
- If there are no parameters, the braces are optional.
- If the braces are omitted, \LaTeX{} discards any spaces following the command. If you want the space preserved, the braces are required.
Basic $\LaTeX$ Commands

Following the $\texttt{\documentclass}$ command is the $\texttt{preamble}$, which basically contains additional instructions for the $\LaTeX$ system. The preamble cannot generate any output.

Following the preamble is the document body, which must be enclosed by the following commands:

$\texttt{\begin{document}}$
$\texttt{\end{document}}$

This is an example of an $\texttt{environment}$. All environments are strictly nested: the end environment must always match exactly to the corresponding begin environment.

Anything following the $\texttt{\end{document}}$ is ignored.
More Information

From the CTAN Starting out page, follow the links to:

1. The (Not So) Short Introduction to LaTeX2ε (lshort.pdf) and read chapters 1, 2, and 4 (except section 4.1).

2. The tutorials by Andrew Roberts and read tutorials 1 (ignoring the stuff about dvi output and converting to pdf — we will produce pdf directly) and 2.
TeXnicCenter: Simple LaTeX Document

\documentclass{article}
\author{Bradley Broom}
\title{Greetings}
\begin{document}
\maketitle
Hello world.
\end{document}
TeXnicCenter: Converting \LaTeX to PDF

The top-level auxiliary file: C:\Documents and Settings\Administrator\My Documents\Greetings.aux
I found no \citation command---while reading file C:\Documents and Settings\Administrator\My Documents\Greetings\mybibstyleclass87b.bib
I found no \bibstyle command---while reading file C:\Users\krcoombes\Documents and Settings\Administrator\My Documents\Greetings\mybibstyleclass87b.bib
There were 0 error messages.

LaTeX-Result: □ Error(s), □ Warning(s), □ Bad Box(es), 1 Page(s)
TeXnicCenter: Viewing PDF
Writing Documented R Analyses

1. Prepare a LaTeX document describing the analysis. Give it an “Rnw” extension instead of “.tex”. Say it is called “myfile.Rnw”
   - If you use TeXnicCenter, make sure it doesn’t silently append an invisible .tex extension.

2. Insert one or more R code chunks starting with \begin{verbatim}

3. Terminate each R code chunk with an “at” sign (@) followed by a space.
\documentclass{article}
\author{Bradley Broom}
\title{Simple Sweave Script}
\begin{document}

\maketitle

This is a simple script to demonstrate Sweave. Let's begin by generating some random numbers:

\begin{verbatim}
<<>>=
runif(10)
\end{verbatim}

disable text processing and follow that up with some random text.

\end{document}
TeXnicCenter: Simple Sweave Script

R version 2.9.2 (2009-08-14)
Copyright (C) 2009 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> Sweave("SimpleScript.Rnw")
Writing to file SimpleScript.tex
Processing code chunks ...
1 : echo term verbatim

You can now run LaTeX on 'SimpleScript.tex'

>
Simple Sweave Script

Bradley Broom

September 10, 2009

This is a simple script to demonstrate Sweave. Let's begin by generating some random numbers:

```r
> rnorm(10)
```

```
[1]  0.29824500  0.34793836 -0.18885553  2.24131255  1.31076623  1.78055102
[7] -0.00217223 -2.29464759  0.90080601  0.59015928
```

and follow that up with some random text.
TeXnicCenter: Simple Sweave Script

Make sure you never edit the .tex file: open it read-only.
Using Sweave

To produce the final document

1. In an R session, issue the command

   Sweave("myfile.Rnw")

   This executes the R code, inserts input commands and output computations and figures into a \LaTeX{} file called “myfile.tex”.

2. In the UNIX or DOS window (or using your favorite graphical interface), issue the command

   pdflatex myfile

   This produces a PDF file that you can use as you wish.
Viewing The Results

Here is a simple example, showing how the R input commands can generate output that is automatically included in the *\LaTeX* output of Sweave.

\begin{verbatim}
> x <- rnorm(30)
> y <- rnorm(30)
> mean(x)

[1] 0.2279967

> cor(x, y)

[1] 0.3408799
\end{verbatim}
A Figure

Next, we are going to insert a figure. First, we can look at the R commands that are used to produce the figure.

```r
> x <- seq(0, 6 * pi, length = 450)
> par(bg = "white", lwd = 2, cex = 1.3, mai = c(1.2,
+ 1.2, 0.2, 0.2))
> plot(x, sin(x), type = "l")
> abline(h = 0, col = "blue")
```

On the next slide, we can look at the actual figure. (Part of the point of this example is to illustrate that you can separate the input from the output. You can even completely hide the input in the source file and just include the output in the report.)
Sine Curve

\[
\begin{align*}
\text{x} & \quad \sin(x) \\
0 & \quad 0.0 \\
5 & \quad 0.5 \\
10 & \quad 0.0 \\
15 & \quad -0.5 \\
20 & \quad -1.0 \\
25 & \quad 0.0 \\
30 & \quad 0.5 \\
35 & \quad 0.0 \\
40 & \quad -0.5 \\
45 & \quad -1.0 \\
50 & \quad 0.0 \\
55 & \quad 0.5 \\
60 & \quad 0.0 \\
65 & \quad -0.5 \\
70 & \quad -1.0 \\
75 & \quad 0.0 \\
80 & \quad 0.5 \\
85 & \quad 0.0 \\
90 & \quad -0.5 \\
95 & \quad -1.0 \\
100 & \quad 0.0
\end{align*}
\]
A Table

```r
> library(xtable)
> x <- data.frame(matrix(rnorm(12), nrow = 3,
+   ncol = 4))
> dimnames(x) <- list(c("A", "B", "C"), c("C1",
+   "C2", "C3", "C4"))
> tab <- xtable(x, digits = c(0, 3, 3, 3, 3))
> tab

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.052</td>
<td>-0.720</td>
<td>0.015</td>
<td>-0.595</td>
</tr>
<tr>
<td>B</td>
<td>0.159</td>
<td>-1.059</td>
<td>0.321</td>
<td>1.753</td>
</tr>
<tr>
<td>C</td>
<td>-0.539</td>
<td>0.530</td>
<td>-0.734</td>
<td>0.119</td>
</tr>
</tbody>
</table>
```
A Table, Repeated

Again, we want to point out that you can show the results—including tables—without showing the commands that generate them.

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.052</td>
<td>−0.720</td>
<td>0.015</td>
<td>−0.595</td>
</tr>
<tr>
<td>B</td>
<td>0.159</td>
<td>−1.059</td>
<td>0.321</td>
<td>1.753</td>
</tr>
<tr>
<td>C</td>
<td>−0.539</td>
<td>0.530</td>
<td>−0.734</td>
<td>0.119</td>
</tr>
</tbody>
</table>
R Revisited: Beyond the Matrix

We have gone from scalar to vector to matrix, attaching names as we go, with the goal of keeping associated information together. So far, we've done this with numbers, but we could use character strings instead:

```r
> letters[1:3]

[1] "a"  "b"  "c"

> x <- letters[1]
> x <- letters[1:3]
> x <- matrix(letters[1:12], 3, 4)
```
No Mode Mixing in Vectors or Matrices

In R, we cannot easily mix data of different modes in a vector or matrix:

```r
> x <- c(1, "a")
> x
[1] "1" "a"
```
However, a list can have (named) components that are of different modes and even different sizes:

```r
> x <- list(teacher = "Keith", n.students = 14,
+ grades = letters[c(1:4, 6)])
> x

$teacher
[1] "Keith"

$n.students
[1] 14

$grades
[1] "a" "b" "c" "d" "f"
```
Note that we named the components of the list at the same time that we created it. Many functions in R return answers as lists.
Extracting Items From Lists

If we want to access the first element of x, we might try using the index or the name in single brackets:

```r
> x[1]

$teacher
[1] "Keith"
```

```r
> x["teacher"]

$teacher
[1] "Keith"
```

These don’t quite work. The single bracket extracts a component, but
keeps the same mode; what we have here is a list of length 1 as opposed to a character string. Two brackets, on the other hand . . .

> x[[1]]

[1] "Keith"

> x["teacher"]

[1] "Keith"

The double bracket notation can be cumbersome, so there is a shorthand notation with the dollar sign. Using names keeps the goals clear.

> x$teacher

[1] "Keith"
Lists with Structure

The most common type of structured array is simply a table, where

- the rows correspond to individuals and
- the columns correspond to various types of information (potentially of multiple modes).

Because we want to allow for multiple modes, we can construct a table as a list, but this list has a constraint imposed on it – all of its components must be of the same length. This is similar in structure to the idea of a matrix that allows for multiple modes.

This structure is built into R as a data frame.

This structure is important for data import.
Reading Data Into R

Although we can simply type stuff in, or use `source()` to pull in small amounts of data we've typed into a file, what we often want to do is to read a big table of data. R has several functions that allow us to do this, including `read.table()`, `read.delim()`, and `scan()`.

We can experiment by using some of the files that we generated in dChip for the first HWK.

We could load the sample info file, and the list of filtered genes. Then we could use the sample info values to suggest how to contrast the expression values in the filtered gene table.
Importing our dChip Data

I exported all of the dChip quantifications to a single file. The file has a header row, with columns labeled “probe set”, “gene”, “Accession”, “LocusLink”, “Description” and then “N01” and so on, 1 column per sample. We can read this into R as follows:

```r
> singh.dchip.data <- 
  read.delim(c("../SinghProstate/Singh_",
  "Prostate_dchip_expression.xls"));
> class(singh.dchip.data)
[1] "data.frame"
> dim(singh.dchip.data)
[1] 12625 108
```

The number of columns is a bit odd...
More on Importing

If we invoke `help(read.delim)`, help pops up for `read.table`. The former is a special case of the latter. Let’s take a look at bits of the usage lines for each:

```r
read.table(file, header = FALSE, sep = "", quote = "\"", dec = ".", row.names, col.names, as.is = FALSE, na.strings = "NA", colClasses = NA, nrows = -1, skip = 0, check.names = TRUE, fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE, comment.char = "#")
```

```r
read.delim(file, header = TRUE, sep = "\t", quote="\"", dec="\"", fill = TRUE, ...)```

Note the default function arguments!
Speeding Up Import

Reading the documentation suggests a few speedups:

- we can use `comment.char = ""`, speeding things up
- we can use `nrows = 12626`, for better memory usage
- we could shift to using `scan` (use help!).

```r
singh.dchip.data <-
    read.delim(c(" ../SinghProstate/Singh_Prostate" ,"_dchip_expression.xls"),
               comment.char = "",
               nrows = 12626
    );
```

is indeed faster!
Is This What We Want?

All of the expression data is now nicely loaded in a data frame. But this data frame really breaks into two parts quite nicely – gene information, and expression values. If we split these apart, then the expression value matrix has 102 columns, corresponding to the sample info entries quite nicely.

```r
singh.annotation <- singh.dchip.data[,1:5];
singh.dchip.expression <- as.matrix(singh.dchip.data[,6:107]);
rownames(singh.dchip.expression) <- singh.annotation$probe.set;
```
Grab the Sample Info Too

What are the columns in my sample info file?

scan name    sample name    type
run.date.block  cluster.block
N01__normal    N01    N    2    2

(the last two you might not have).

singh.sample.info <-
    read.delim("../SinghProstate/sample_info_2.txt",
                comment.char = "",
                nrows = 103
    );
Test Something Interesting

In the first homework, we saw that the data split into two clusters that didn’t agree well with the tumor/normal split. It might very well be that there was some type of batch effect in addition to the biological split of interest.

Can we factor the batch effect out? If we know what the batch split is, we can first fit a model using just the batches, subtract the fit off, and then fit a model using the tumor/normal split on what remains.
Tumor vs Normal

singh.probeset.lm <-
 lm(unlist(singh.dchip.expression[
    singh.annotation$probe.set
    == "31539_r_at",])
   ~ singh.sample.info$type
);

singh.probeset.anova <-
 anova(singh.probeset.lm);
Tumor vs Normal (cont)

```r
> singh.probeset.anova
Analysis of Variance Table

Response: unlist(singh.dchip.expression[
    singh.annotation$probe.set == "31539_r_at",])

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$type</td>
<td>1</td>
<td>71.42</td>
<td>71.42</td>
<td>5.3748</td>
<td>0.02247*</td>
</tr>
<tr>
<td>Residuals</td>
<td>100</td>
<td>1328.81</td>
<td>13.29</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```
T vs N, After Blocking

```r
singh.probeset.lm.full <-
  lm(unlist(singh.dchip.expression[
    singh.annotation$probe.set
    == "31539_r_at",])
  ~ singh.sample.info$cluster.block
  + singh.sample.info$type
);

singh.probeset.anova.full <-
  anova(singh.probeset.lm.full);
```
### T vs N, After Blocking (cont)

```r
> singh.probeset.anova.full
```

Analysis of Variance Table

Response: `unlist(singh.dchip.expression[singh.annotation$probe.set == "31539_r_at",])`

<table>
<thead>
<tr>
<th></th>
<th>Df</th>
<th>Sum Sq</th>
<th>Mean Sq</th>
<th>F value</th>
<th>Pr(&gt;F)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$block</td>
<td>1</td>
<td>404.97</td>
<td>404.97</td>
<td>40.6399</td>
<td>5.85e-09 ***</td>
</tr>
<tr>
<td>$type</td>
<td>1</td>
<td>8.75</td>
<td>8.75</td>
<td>0.8779</td>
<td>0.3511</td>
</tr>
<tr>
<td>Residuals</td>
<td>99</td>
<td>986.51</td>
<td>9.96</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
Hasn’t Someone Done This?

Other people have thought about the data structures that might be natural for microarray data. In particular, a lot of these functions are collected at Bioconductor.

Let’s try to grab some of the packages and functions that will help with this type of analysis.
Obtaining extra R packages

The R GUI makes it easy to get additional packages via the internet. From the “Packages” menu, you simply select “Install package(s)...”. (In order to install packages from Bioconductor, you must first use the “Select repositories...” menu item to tell R to look there.) The menu item presents a dialog box containing a list of the available packages. You then select one or more (by holding the control key while clicking with the mouse) and press the “OK” button. R then downloads the package, installs it, and updates the help files. It finishes by asking if you want to delete the downloaded files; unless you want to save them to install them on another computer without an internet connection, the usual answer is “yes”.

© Copyright 2004–2009 KR Coombes, KA Baggerly, and BM Broom

GS01 0163: Analysis of Microarray Data
Bioconductor Packages

You will need to install the following Bioconductor packages.

- Use the items “Select repositories...” and “Install package(s)...” on the “Packages” menu to get them.

From the BioC software repository:

**Biobase** : Base functions for BioConductor

**affy** : Methods for Affymetrix oligonucleotide arrays

**affypdnn** : Probe dependent nearest neighbor (PDNN) for affymetrix

From the BioC experiment repository:

**affydata** : Affymetrix data for demonstration purposes
Bioconductor Widget Packages

Installing some additional packages will provide graphical tools that make it easier to read Affymetrix microarray data and construct sensible objects describing the experiments.

From the BioC software repository:

**tkWidgets** : R based Tk widgets

**widgetTools** : Creates an interactive tcltk widget

**DynDoc** : Dynamic document tools