I. Create a Directory where you store your work

Make a directory for your data under it, call it Data for instance.

II. Starting R and manipulating data files.

Use whatever environment you like best, environments designed for optimizing your workflow include:

Emacs and ESS  [http://ess.r-project.org/](http://ess.r-project.org/)
Eclipse and StatET  [http://www.walware.de/goto/statet](http://www.walware.de/goto/statet)

You should now have a window with the prompt: “>”.

You should keep a diary of your work, the best protocol is to use Sweave or pgfSweave, some of the labs are associated to Rnw files that will be available at the end of the labs so you can run through them automatically, this is only recommended once you have gone through each command one at a time.

Set your working directory in R.

```r
> getwd()
[1] "/Users/susan/metag"
```

```r
> setwd("/Users/susan/metag/"

Under this directory, I created a subdirectory “Data” to put all my data, your configuration may differ.

Exercise 1  Creating and Manipulating data from inside R, load the package ade4 with the function library, then look at the data set from that package called lizards.

What type of data is the lizards dataset?
What are it’s components?

We will often start Labs with ‘internal data’ before moving to more realistic and tricky cases where the data are read from the inside. Some of the data we will work with come from R, they will be recognizable as such by their names: vsn28.RData, mice.RData, chipannotation.save with the .RData or .save suffixes. These only need to be loaded into R to make them available. However, depending on which packages were used to create them, a special package might have to be installed on your system for you to be able to work with them.
Exercise 2  Try manipulating the vsn28 data using:

```r
> load("./Data/vsn28.RData")
> require(Biobase)
> vsn28.exprs = exprs(vsn28)
> vsn28
```

**Manipulation of dimnames**

Exercise 3  What do these commands do?

```r
> substr(dimnames(vsn28.exprs)[[2]],1,3)
```

What does this command do?

```r
> dimnames(vsn28.exprs)[[2]]=sub(".CEL","",dimnames(vsn28.exprs)[[2]])
```

These are actually the data from 14 IBS and 14 Healthy Rats taken from different parts of the intestine. Look at the names of the samples (column names of the matrix vsn28.exprs).

Exercise 4  Can you guess how these might pair up?

Enter the following labels:

```r
> sampletypes <- c("IBS","CTL")
> status <- c(1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)
> sampleLabel <- factor(sampletypes[status])
> pairLabel <- c(9, 10, 11, 12, 1, 2, 3, 9, 10, 11, 12, 5, 6, 7, 8, 4,
13, 14, 13, 14, 5, 6, 7, 8, 1, 2, 3, 4)
```

**Making Paired Differences**

We are going to start by making a new matrix of paired differences between matched pairs.

Here is one way of doing this you can try and think of other ways:

```r
> paired.diffs <- matrix(0, nrow=nrow(vsn28.exprs), ncol=ncol(vsn28.exprs)/2)
> ctl.pool=rep("",14)
> ibs.pool=rep("",14)
> for (i in 1:14) {
    ctl.pool[i] = intersect(colnames(vsn28.exprs)[pairLabel == i],
                          colnames(vsn28.exprs)[sampleLabel == "CTL"])
    ibs.pool[i] = intersect(colnames(vsn28.exprs)[pairLabel == i],
                          colnames(vsn28.exprs)[sampleLabel == "IBS"])
    paired.diffs[,i] = vsn28.exprs[,ctl.pool[i]] - vsn28.exprs[,ibs.pool[i]]
}
> ######Make a Clearer Annotation###############################################
> names2= paste(ctl.pool,ibs.pool,sep="-")
> names2[13]= "Day3.CC-IC"
> names2[14]= "Day3.CP-IP"
> load("./Data/chipannotation.save")
```
Exercise 5  We want to have meaningful names, we use the file chip.annotation which contains a list chip.annotation of different names for the probes and the taxa. Explore the chip.annotation list and find what its components are, then use this to give better names to the probes than the Affymetrix identifiers. Here we call these names bacter.names.

> bacter.names = chip.annotation$previous_rep_org[match(rownames(vsn28.exprs), chip.annotation$chip.names)]
> dimnames(paired.diffs)=list(bacter.names,names2)

III. Reading in data from outside R

Documentation on import/export of R Data
http://cran.r-project.org/doc/manuals/R-data.html You could directly import data from the internet using its url using:

IBS28=read.delim("http://www-stat.stanford.edu/~susan/Summer11/Data/IBS28n.txt",as.is=TRUE)
IBS28=read.delim("./Data/IBS28n.txt",as.is=TRUE)

Factors, Matrices and Dataframes:  Exercise 6  Make a factor variable that contains the Day labels in a different way than below. Check it matches.

  day=rep(2,28)
  day[1:7:20]=3
  day[c(5:7,16,25:28)]=1
  t(cbind(colnames(vsn28.exprs),day))###Check days####

Make a factor variable for days, call it dayf.

Exercise 7  Suppose we want to use the data in the last 56 columns of the IBS28 matrix. The main entries in columns 14-41 are fractions, we want to make them into percentages, but we only want to retain the rows that have at least one of the samples present at at least a fraction of 0.95. Prepare a new matrix and call it conting28, remove the suffix from the column names.

This is one way of preparing such a set, note how we make a logical condition into a 0/1 variable.

> ibsn28=IBS28[-(1:13)]
> ibsn28.pres=matrix(0,ncol=28,nrow=7935)
> ibsn28.pres[which(ibs28[,1:28]>0.95,arr.ind=TRUE)]=1
> sums28=apply(ibs28.pres,1,sum)
> ###useful means present in at least one array
> useful=which(sums28>1)
> isbn28u=ibs28[useful,1:28]
> dimnames(isbn28u)[[2]]=sub(".CEL_pf","",dimnames(isbn28u)[[2]])
> conting28=round(isbn28u*100)

Exercise 8  Specific data class generation is often handled by specialized packages, for instance ape can handle reading in a tree with read.tree.
IV. Ranking and Thresholding Data

Because we want to replace the original abundances by more robust values which are not as sensitive to the experimental conditions.

- **rank function.** We make a matrix with the same dimensions and names but replace all the values by their ranks (order statistics).

  What does the rank function do? Try for example:
  ```r
  > vec8=c(3.2,2,2.5,5,7,5.6,2.2,4.2)
  > rank(vec8)
  [1] 4 1 3 6 8 7 2 5
  ```

  What is the rank of the smallest value?

  **Exercise 9** Use this function for replacing the values of `vsn28.exprs`, by their ranks.
  ```r
  > vsn28.ranks=vsn28.exprs
  > for (j in 1:28) vsn28.ranks[,j]=rank(vsn28.exprs[,j])
  ```

- **Thresholding**

  To protect against supposing that the values of ranks have jumped a lot just because the values are really at noise level, we need to threshold the ranks at a realistic level. In this case, we know there were not more than 2200 species present, so a reasonable threshold is around 6000.

  **Exercise 10** Replace all the values smaller than 6000 by the value 6000. Compute whichsmall containing the array indices of the ranks smaller than the threshold. What dimension is whichsmall?

  ```r
  > threshold=6001
  > truncrank=vsn28.ranks
  > whichsmall=which(vsn28.ranks<threshold,arr.ind=TRUE)
  > dim(whichsmall)
  > truncrank[whichsmall]=threshold
  > pairdifft = matrix(0, nrow=nrow(truncrank), ncol=ncol(truncrank)/2)
  > for (i in 1:14) {
      ctl.pool = intersect(colnames(truncrank)[pairLabel == i],
                           colnames(truncrank)[sampleLabel == "CTL"])
      ibs.pool = intersect(colnames(truncrank)[pairLabel == i],
                           colnames(truncrank)[sampleLabel == "IBS"])
      pairdifft[,i] = truncrank[,ctl.pool] - truncrank[,ibs.pool]
  }
  > dimnames(pairdifft)=list(bacter.names,names2)
  ```
V. Subsetting the data  Exercise 11  Create a new data set, from the vsn28.exprs data, only consider the species that are ranked strictly above the 6000 level in at least 14 of the 28 tables.

Create a new array whichpresent replacing the values by a binary coding of whether or not the rank is higher than 6000. Create a new vector called speciespresent that shows the number of species present at that level of rank across all samples.

You can recycle the whichsmall indices or work from scratch.

```r
> whichpresent = vsn28.ranks
> for (i in 1:nrow(whichpresent)) for (j in 1:ncol(whichpresent)) {whichpresent[i,j] = 1}
> whichpresent[whichsmall] = 0
> speciespresent = apply(whichpresent, 1, sum)
> arraypresent = vsn28.exprs[speciespresent > 13,]
```

speciespresent measures the number of present species among the 32 arrays, there is a good chance that if there are more than half the arrays that have a 1 in them then this species is a relevant one, we choose these. Make a table showing the frequencies of species present.