Day 2: Basic statistics

Jerry Davison and Martin Morgan

Contents

1 Data

Today we'll cover statistical concepts and tests commonly used in cancer research. The dataset we'll access is a subset of the ALL expression data whose patient information we worked with in the first day's material. In addition to that information we'll access 1000 associated expression microarray features that present the highest variance across the patient samples. The data have been saved in a `binary` format to reduce file sizes.

```
> ### Go to your data directory to get these files:
> dataFile = file.choose() # ALL1k.rda
> dataName = load(file=dataFile) # "ALL1k" is the data frame's name
> infoFile = file.choose() # ALL1k_pData.rda
> infoName = load(file=infoFile) # "info" is the data frame's name
```

2 Descriptive statistics

Let's look a little more closely at patient information in the `info` file:

```
> median(info$age)
```

```
[1] NA
```

```
> ?median

> median(info$age, na.rm=TRUE)

[1] 29

> range(info$age, na.rm=TRUE)

[1]  5 58

> quantile(info$age, na.rm=TRUE)

  0%  25%  50%  75% 100%
 5.0 19.0 29.0 45.5 58.0
```

- Some simple plots of patient ages – note the nested functions!

```
> plot(info$age)
> plot(sort(info$age))
> sortedAge = sort(info$age)
> ?plot
```

- **Exercise:** Plot the `sortedAge` with markers at each data point and connect the points with red lines.

- **Exercise:** Plot one variable (e.g., age) as a function of another (e.g., sex). Since sex is a factor, *R* chooses to create a box plot; does this make sense?

```
> plot(age ~ sex, info)
```

- Histograms, and their display options:

```
> hist(info$age)
> ?hist
> hist(info$age, br=25)
```

- Cross tables use `formulas` to describe the relationship between the data they present:

```
> xtabs(~sex, data=info, exclude=NA)

sex
 F  M
42 83

> xtabs(~sex + remission, data=info, exclude=NA)

   remission
sex CR REF
  F 27   7
  M 71   8
```

- **Exercise:** How many hyperdiploid males are refractory? Hint: review the 'doc' data frame from last week's lesson for column descriptions.

3 t-tests, linear models, and chi-square tests

3.1 t-tests

Use `plot` to visualize the distribution of female and male ages in the `info` data set.

```
> plot(age ~ sex, info)
```

It looks like females are on average older than males. Use `t.test` to find out.

```
> t.test(age ~ sex, info)
```

```
        Welch Two Sample t-test

data:  age by sex
t = 1.6034, df = 79.88, p-value = 0.1128
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.022660  9.504142
sample estimates:
mean in group F mean in group M
       35.16667        30.92593
```

Check out the help page for `t.test`

```
> ?t.test
```

What are all those additional arguments to `t.test`? For example, what is the meaning of the `var.equal` argument?Why are there 79.88 degrees of freedom?

```
> t.test(age ~ sex, info, var.equal=TRUE)
```

```
        Two Sample t-test

data:  age by sex
t = 1.6266, df = 121, p-value = 0.1064
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.9207109  9.4021924
sample estimates:
mean in group F mean in group M
       35.16667        30.92593
```

3.2 Linear models

A t-test can also be viewed as an analysis of variance (ANOVA); analysis of variance is a form of linear model. Use `lm` to fit a linear model that describes how age changes with sex; the `anova` function summarizes the linear model in a perhaps more familiar ANOVA table.

```
> (fit <- lm(age ~ sex, info))
```

```
Call:
lm(formula = age ~ sex, data = info)

Coefficients:
(Intercept)         sexM
     35.167       -4.241
```

```
> anova(fit)
```

```
Analysis of Variance Table

Response: age
           Df  Sum Sq Mean Sq F value Pr(>F)
sex         1   497.4  497.41  2.6459 0.1064
Residuals 121 22747.4  187.99
```

What kinds of assumptions are being made in the linear model, e.g., about equality of variances? Try plotting `fit`; what are the figures trying to tell you?

```
> plot(fit)
```

`fit` is an example of an *R object*. Find out it's class

```
> class(fit)
```

```
[1] "lm"
```

`plot` is an example of an *R generic*; it has different *methods* implemented for different classes of objects. Use `methods` to see available methods

```
> methods(plot)
```

```
 [1] plot.HoltWinters*   plot.TukeyHSD*      plot.acf*           plot.data.frame*
 [5] plot.decomposed.ts* plot.default        plot.dendrogram*    plot.density*
 [9] plot.ecdf           plot.factor*        plot.formula*       plot.function
[13] plot.hclust*        plot.histogram*     plot.isoreg*        plot.lm*
[17] plot.medpolish*     plot.mlm*           plot.ppr*           plot.prcomp*
[21] plot.princomp*      plot.profile.nls*   plot.spec*          plot.stepfun
[25] plot.stl*           plot.table*         plot.ts             plot.tskernel*
```

```
   Non-visible functions are asterisked
```

Look up the help page for the `plot` generic, `lm` method with

```
> ?plot.lm
```

Fitted models can be used in other functions, for instance to predict values for new data. Construct a `data.frame` with a single column `sex` with values `"M"` and `"F"`. Consult the help page for the `predict.lm` method, and calculate the expected value of the fitted model for males and for females.

```
> df = data.frame(sex=c("F", "M"))
> predict(fit, df)
```

```
       1        2
35.16667 30.92593
```

What do the predicted values correspond to in the `t.test`? Use `coefficients` to extract the coefficients of the fitted model.

```
> coefficients(fit)
```

```
(Intercept)        sexM
  35.166667   -4.240741
```

Interpret the `(Intercept)` and `sexM` coefficients in terms of female and male ages.

3.3 Chi-squared tests

The article from which the `info` object is derived states that "Although chromosome translocations and molecular rearrangements are relatively infrequent in T-lineage ALL, these events occur commonly in B-lineage ALL and reflect distinct mechanisms of transformation". Let's investigate this statement.

The relevant columns of data are summarized as

```
> summary(info[,c("BT", "cyto.normal")])
```

```
       BT     cyto.normal
 B2     :36   Mode :logical
 B3     :23   FALSE:69
 B1     :19   TRUE :24
 T2     :15   NA's :35
 B4     :12
 T3     :10
 (Other):13
```

Simplify the number of BT levels by creating a map between subtypes and types

```
> old <- levels(info$BT)
> new <- substr(old, 1, 1)
> (map <- setNames(new, old))
```

```
  B  B1  B2  B3  B4   T  T1  T2  T3  T4
"B" "B" "B" "B" "B" "T" "T" "T" "T" "T"
```

The names of the map are the subtypes, the elements are the types. Map the levels of the BT variable from sub-type to type with the following command, and cross-tabulate the data

```
> levels(info$BT) <- map
> xtabs(~ BT + cyto.normal, info)
```

```
   cyto.normal
BT  FALSE TRUE
  B    56   17
  T    13    7
```

The data are qualitatively consistent with the statement that molecular rearrangements are more common in B-lineage ALL. Let's test this with a chi-squared test

```
> chisq.test(info$BT, info$cyto.normal)
```

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  info$BT and info$cyto.normal
X-squared = 0.5962, df = 1, p-value = 0.44
```

Interpret the results. What about additional parameters documented on `?chisq.test`?

4 Higher dimensional data

Earlier we read in the ALL1k object. This is a data frame. The rows represent 1000 microarray probe sets. The first two columns contain information about the probe sets. The remaining 128 columns are (normalized) expression values of 128 samples. The 128 samples correspond to the phenotypic data in the info object we've been working with. The following lines take the expression values and transform them from a *data.frame* to a *matrix*, adjusting the row and column names to reflect the probe and sample ids.

```
> m <- as.matrix(ALL1k[, -c(1, 2)])
```

Some of the results below involve plots, and it's convenient to choose pretty and functional colors. We use the *RColorBrewer* package; see colorbrewer.com.

```
> library(RColorBrewer)
> divergent <- brewer.pal(11, "RdBu")
> highlight <- brewer.pal(3, "Set2")[1:2]
```

'divergent' is a vector of colors that go from red (negative) to blue (positive). 'higlight' is a vector of length 2, light and dark green.

4.1 Principle components

It is hard to visualize data from 1000 probesets and 128 samples. In addition, the data suffers from the problem that there are many more measurements per sample (1000) than there are samples (128). For these reasons, we might wish to reduce the number of dimensions in which the data is represented. One way of reducing the dimensionality is by calculating *principle components*. Here we calculate principle components:

```
> tm = t(m)
> pc <- prcomp(tm)
```

The principle components are, roughly, vectors that pass through the data in such a way as to explain as much variability as possible. We can visualize the data in two-dimensional space by plotting the 'rotation', coloring points by whether they belong to the B or T ALL lineage (we will discuss other arguments to `plot` in a subsequent class).

```
> color <- highlight[info$BT]
> plot(pc$x, col=color, pch=20, cex=2)
```

Notice that the B and T lineages separate quite well.

There are a number of variants to principle components, and some tricky statistical issues involved even in the simple example above, so proceed with caution!

4.2 Clustering: `hclust`

Clustering, or 'unsupervised machine learning', tries to group samples based on similarity. We start by calculating the correlation between each of our samples.

```
> cm <- cor(m)
```

`cm` is a 128 x 128 matrix, and it measures the correlation between each pair of samples – samples with similar patterns of gene expression will be correlated.

We can use the strength of correlation as a basis for describing the distance between samples. Two samples will be 'near' each other if they have similar patterns of correlation with other samples.

```
> d <- dist(cm)
```

There are several ways of measuring distance; the default is a euclidean measure.

The next step is to group (cluster) samples that are similar to one another into a dendrogram summarizing similarity.

```
> cl <- hclust(d)
> plot(cl)
```

Somewhat more revealing of structure is a *heatmap*, with rows and columns clustered as we've just described, and with a column color bar to indicate whether the sample came from a B or T lineage sample.

```
> color <- highlight[info$BT]
> heatmap(cm, ColSideColor=color, col=divergent)
```

The plot shows that, mostly, the B lineage samples cluster together, and the T lineages cluster together. If this were an exploratory analysis, one would look carefully to ensure that the mis-clustered samples were not mis-labeled.

4.3 Classification: `kmeans`

Classification ('supervised machine learning') groups samples into a pre-specified number of groups. There are many varieties of classification. *k-means* classification aims to place k 'centroids' in such a way that the sum of squares of each point to the nearest centroid is minimized. Here we perform k-means classification with 2 groups, anticipating that we will recover the B and T ALL lineages; we use `set.seed` to set the random number seed to a particular (arbitrary) value to ensure reproducibility.

```
> set.seed(123)
> cl <- kmeans(t(m), 2)
```

The 'confusion' matrix is one measure of how effective the classification is; here we see that all but one of the samples were assigned to the correct group.

```
> xtabs(~ cl$cluster + info$BT)
```

```
           info$BT
cl$cluster  B  T
         1 94  0
         2  1 33
```

There are objective criteria for parameter choice and validation of supervised machine learning algorithms.

5 Lessons learned

There are several important lessons from this brief tour of statistical functions in *R*:

1. The 'formula' notation is very powerful, and worth learning.
2. *R* can perform diverse statistical analyses.
3. Help pages are your friends, even if they initially seem challenging.
4. Many analyses, even straight-forward t-tests or ANOVA, require a level of statistical sophistication.

6 Homework

In the next class we'll use data from the Center for Disease Control's Behavioral Risk Factor Surveillance System (BRFSS) annual survey. Check out the web page for a little more information. We are using a small subset of this data, including a random sample of 10000 observations from each of 1990 and 2010. As preparation for the class:

1. Input the data using `read.csv`, creating a variable `brfss` to hold it. The data file is `BRFSS-subset.csv` in your home directory.
2. Explore the data using `class`, `dim`, `head`, `summary`, etc. Use `xtabs` to summarize the number of males and females in the study, in each of the two years.
3. An expression like `xtabs(Weight ~ Sex, brfss)` calculates the sum of the `Weight` column, separately for each `Sex`. Can you divide this sort of cross tabulation by the result of another `xtabs` function to calculate average weight per sex in each year of the study?

7 Resources

- Dalgaard, P. 2002. *Introductory Statistics with R*. Springer.