Analysis of complex microarray experimental designs

Martin Morgan

10 January, 2007

1 Introduction

Motivation: two-color microarray experiment.

- 5 cell lines PEC32, PEC34, PEC36, PEC39, PEC40.
- 2 drugs (`SFN` and `HGF`) each present at two levels (`Low` and `High`).
- Each chip has 15488 spots representing 2 technical replicates of each of 7744 genes.

1.1 Terminology

Response A **dependent** variable, e.g., gene expression level.

Factor A controlled **independent** variable, e.g., tissue source, drug treatment, wild-type versus mutant, gender. A factor represents a general category of **treatment**, and is likely to be administered at different **levels**, e.g., presence versus absence of tumor, low medium or high concentration of a drug.

Covariate Additional independent variables, measured not explicitly controlled by the experimenter.

Biological replicate Statistically independent measurement on different objects, e.g., tissues from different individuals, independent samples of a cell line.

Technical replicate Repeated measurements on the same biological material, e.g., replicate spots on an array; classical.

Blocking Grouping samples into homogenous clusters. Blocking is often an effective way to accommodate known differences in samples.

Randomization Assigning samples randomly to treatments. Randomization also attempts to group samples into homogenous clusters, but does so without retaining information about why samples might differ.

Experimental design The sensible assembly of replicates, factors, and response variables in order to address specific statistical questions.

1.2 A simple experimental design

- One factor with two different treatments, samples assigned randomly to each treatment; measure a single response variable. Label the treatments `Low` and `High`.

- The classical way to analyze this type of experiment is with a t-test: compare variation within each treatment to variation between treatments. If the variation between groups is greater than variation within groups, then conclude that members of the two groups differ in some systematic way.

- A formula for the t-test...

$$t = \frac{\bar{x}_{\texttt{High}} - \bar{x}_{\texttt{Low}}}{\sqrt{s^2_{\texttt{High}}/n_{\texttt{High}} + s^2_{\texttt{Low}}/n_{\texttt{Low}}}} \tag{1}$$

  seems like it is comparing means (symbol $\bar{x}$) but is actually asking about variation between groups ($\bar{x}_{\texttt{High}} - \bar{x}_{\texttt{Low}}$) compared to average variation within groups ($\sqrt{s^2_{\texttt{High}}/n_{\texttt{High}} + s^2_{\texttt{Low}}/n_{\texttt{Low}}}$; $s^2_{\texttt{High}}$ is the sample variance in the `High` treatment).

- We can view the response measured on an individual j assigned to the ith treatment as due to an overall average effect μ, plus a deviation caused by the effect of the treatment i they were exposed to α_i, plus a residual deviation ϵ_{ij} unique to that individual:

$$y_{ij} = \mu + \alpha_i + \epsilon_{ij} \tag{2}$$

  The classical assumption is that the deviations ϵ_{ij} follow a normal distribution with mean 0 and variance σ^2.

- The **expected value** of an individual sampled from the `Low` treatment group is

$$E[y_{ij}|i = \texttt{Low}] = \mu + \alpha_{\texttt{Low}} \tag{3}$$

The expected value of an individual sampled from the `High` treatment group is

$$E[y_{ij}|i = \texttt{High}] = \mu + \alpha_{\texttt{High}} \tag{4}$$

The expected value of an individual sampled from the entire experiment is

$$E[y_{ij}] = \mu \tag{5}$$

- In a slightly different formulation, we could designate one of the treatments as a kind of standard against which others are measured. Suppose that we choose `Low` to be the standard. Then μ is the expected value of this standard and $\mu + \alpha_{\texttt{High}}$ the expected value of the `High` treatment (and $(\mu + \alpha_{\texttt{High}}) - (\mu) = \alpha_{\texttt{High}}$ the deviation of the `High` from the standard treatment). Moreover, we can write an expression describing *all* the data as

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \tag{6}$$

$\mathbf{y}$ is a vector of observed values. β is a vector with two elements, corresponding to the expected value in the `Low` treatment and the deviation between the `Low` and `High` treatments. $\mathbf{X}$ is a **design matrix** or **model matrix** of 0s and 1s that indicate how the components of β are to be combined to describe the observed values. Here is a quick example, supposing that the first 3 individuals are from the `Low` and the second 3 individuals from the `High` treatment, and that the means of the treatments are 1 and 2:

$$\begin{bmatrix} 0.9 \\ 1.1 \\ 1.0 \\ 2.0 \\ 2.1 \\ 1.9 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} + \begin{bmatrix} -0.1 \\ 0.1 \\ 0.0 \\ 0.0 \\ 0.1 \\ -0.1 \end{bmatrix} \tag{7}$$

Normally, the values of β, ϵ are not known.

- The primary goal of an analysis is to estimate the values of β. With an estimate of β in hand, it is possible to estimate the residuals ϵ_{ij} and to use these to assess whether our description of the data is somehow adequate.

- Estimates of β are usually accompanied by some statement of statistical support for the significance of each component. One way of thinking about this is contrasting a model where a component of β (e.g., corresponding to α_i) is zero (e.g., $y_{ij} = \mu + \epsilon_{ij}$), versus a model where the component is different from zero ($y_{ij} = \mu + \alpha_i + \epsilon_{ij}$). We then compare the residuals in the first model with the residuals in the second model. If the residuals are significantly smaller in the second model, we accept it as a better description of the data. Having accepted the second model, we also accept the estimate of the coefficient α_i as statistically signficant.

- The approach extends to many other more complicated designs.

- A better estimate of a treatment mean is possible when there are more samples assigned to each treatment, and in general statistical **power** (the ability to detect differences between treatments, when in fact the differences exist) increases with replication within treatments.

Factorial designs combine two or more factors simultaneously.

- Each factor can have two or more different levels.

- **Main effects** describe the average effect of a factor on the response, *averaging over all other factors*.

- **Interaction effects** describe how different levels of one factor differ over levels of another factor.

- The classical approach is to describe variation first in terms of main effects, and then to attempt to explain remaining variation in terms of interactions. Note that this is *reductionist* and *parsimonious*.

- For example, a linear model for an experiment with two factors (i, j) each with two levels (`Low`, `High`) might be written as:

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk} \tag{8}$$

  We investigate the corresponding model matrix in the lab.

What makes for a good factorial design?

- Simplicity – There is no theoretical limit to the complexity of a factorial ANOVA, but even three-way interactions (i.e., three factors) can be difficult to interpret biologically, even at the best of times.

- Replication – In a one-way ANOVA, a treatment represented by a single sample has a mean but no variance, and hence the effect of the treatment cannot be compared to other treatments in the experiment. In a factorial experiment with two factors each with two treatment, and with each treatment represented by a single individual, it is mathematically impossible to estimate an interaction effect. More generally, replication of each treatment combination increases the ability to assess statistical significance.

2 Using `limma` to analyze microarray experiments

2.1 An example

The following data frame, constructed from the description of the experiment (see the lab), summarizes important aspects:

```
> head(exptlDesign)
```

```
  cellLine  SFN  HGF
1       34  Low  Low
2       34  Low High
3       34 High High
4       34 High  Low
5       32  Low  Low
6       32  Low High
```

```
> table(exptlDesign[, -1])
```

```
      HGF
SFN    Low High
  Low    5    5
  High   5    5
```

We initially focus on the `SFN` and `HGF` treatments, glossing over `cellLine` and spot replication on each chip.

2.2 The model

We are interested in a model with two main effects (`SFN, HGF`) and their interaction. The linear model is

$$y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \epsilon_{ijk} \tag{9}$$

We can think of α as a way of encoding main effect of the `SFN` factor, β the main effect for factor `HGF`, and γ the interaction. There are two possible values of α, corresponding to `Low` and `High` levels of `SFN`. We encode these as $\alpha_{\texttt{Low}} = 0, \alpha_{\texttt{High}} = 1$. Similar considerations apply to β and γ.

We can specify this model with an R formula. Formulas are written as

```
> Response ~ Dependent
```

Where `Response` enumerates (one or more) response variables, and `Dependent` specifies the model. Here are two equivalent descriptions of our experiment:

```
> ~1 + SFN + HGF + SFN:HGF
> ~SFN * HGF
```

The response is not (yet) specified in these formulas. In the first formula, the 1 is a place-holder for the overall mean μ, each of the main effects are listed, and then their interaction (with the `:` symbol). The second formula relies on automatic expansion of `*`, which indicates that all main effects and their interactions are to be included. Details about formula specification, including how to remove the overall mean, are available with `?formula`.

The model matrix is obtained by applying the general formula describing the treatments in the experiment to the specific data collected:

```
> X <- model.matrix(~SFN * HGF, exptlDesign)
> head(X)

  (Intercept) SFNHigh HGFHigh SFNHigh:HGFHigh
1           1       0       0               0
2           1       0       1               0
3           1       1       1               1
4           1       1       0               0
5           1       0       0               0
6           1       0       1               0
```

The model matrix has 20 rows (corresponding to each sample in the experiment) and 4 columns. The column `(Intercept)` consists only of 1s, indicating that all samples have a term corresponding to the standard μ. The column `SFNHigh` summarizes whether deviations associated with the `High` level of `SFN` are included. Can you interpret the rest of the terms in the matrix?

2.3 The data

The following `ExpressionSet` summarizes the data; it is created with commands in the lab.

```
> M
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 14772 features, 20 samples
  element names: exprs
phenoData
  rowNames: PEC34_cntrl, PEC34_HGF, ..., PEC39_cntrl1 (20 total)
  varLabels and varMetadata:
    FileName: NA
featureData
  rowNames: F:190753, F:190754, ..., F:206217 (14772 total)
  varLabels and varMetadata: none
experimentData: use 'experimentData(object)'
Annotation character(0)
```

For reasons that will become clear below, initially we only want to analyse the first half of the columns in `M`, i.e.,

```
> Mh <- M[1:(nrow(M)/2), ]
```

2.4 Model fit

There are two steps to model fit in *limma*: the actual fit of the model, and an **empirical Bayes** step that assessess statistical significance of the fit while attempting to correct for the large number of comparisons that are being performed:

```
> fit <- lmFit(Mh, X)
> efit <- eBayes(fit)
```

The results in `efit` contain estimates of the coefficients (our β), intermediate calculations relevant in other statistical analyses (e.g., `qr`),and measures relevant to assessing statistical significance (e.g., `t, p.value, lods, F, F.p.value`);

```
> names(efit)
```

```
 [1] "coefficients"     "rank"
 [3] "assign"           "qr"
 [5] "df.residual"      "sigma"
 [7] "cov.coefficients" "stdev.unscaled"
 [9] "pivot"            "genes"
[11] "Amean"            "method"
[13] "design"           "df.prior"
[15] "s2.prior"         "var.prior"
[17] "proportion"       "s2.post"
[19] "t"                "p.value"
[21] "lods"             "F"
[23] "F.p.value"
```

```
> head(efit[["coefficients"]])
```

```
         (Intercept)     SFNHigh    HGFHigh SFNHigh:HGFHigh
F:190753  0.27810206 -0.13508752 0.15645506     -0.12258728
F:190754  1.01380493 -0.08379969 0.17706461      0.01436019
F:190755  0.39159498 -0.07912876 0.04372644      0.11018118
F:190756  0.02626411 -0.04883235 0.06043701     -0.04170700
F:190757  0.04954992 -0.02475607 0.06383627     -0.09174890
F:190758  0.26172803 -0.08856251 0.11831189     -0.06009859
```

As noted, the `coefficients` represent estimates of β, with each row corresponding to a different unique identifier. We can combine these coefficients with the experimental status of each individual to calculate the **expected (fitted) value** of the expression expected for the unique ID:

```
> head(cbind(X, X %*% efit[["coefficients"]][1,
+     ]))
```

```
  (Intercept) SFNHigh HGFHigh SFNHigh:HGFHigh
1           1       0       0               0 0.2781021
2           1       0       1               0 0.4345571
3           1       1       1               1 0.1768823
4           1       1       0               0 0.1430145
5           1       0       0               0 0.2781021
6           1       0       1               0 0.4345571
```

limma provides methods for calculating fitted values `?fitted.MArrayLM` and residuals `?residuals.MArrayLM`.

Once we have fit the model to our data, we can get an overview of important genes for each coefficient in the model with `decideTests`. This performs statistical assessment of model fits, identifiying which coefficients are significant for each feature.

```
> decided <- decideTests(efit)
> summary(decided)
```

```
   (Intercept) SFNHigh HGFHigh SFNHigh:HGFHigh
-1        2004       1       0               0
0         2778    7385    7386            7386
1         2604       0       0               0
```

Entries indicate the number of statistical tests falling into different categories. The row labeled `0` are statistical tests that are not significant. `-1` indicates that individuals in groups with the low value of the corresponding coefficient had higher expression values than individuals with the high value of the coefficient, and vice versa for rows labeled `1`.

Specific features with important effects (e.g., with respect to `SFN`) can be identified using

```
> topTable(efit, coef = 2, n = sum(decided[, 2] !=
+     0))
```

```
           ID     logFC    AveExpr        t      P.Value
1430 F:192218 -1.193878 -0.2723395 -6.41404 3.064263e-06
      adj.P.Val        B
1430 0.02263264 3.458403
```

3 Technical replicates

Technical replication refers to a single biological sample measured in the same assay two or more times. For instance, a chip might have replicate spots for each gene, or a single biological sample might be applied to two identical chips. The major goal of technical replication is to understand variation due solely to imprecision in the technology. The idea is that this variation can then be statistically removed, allowing underlying variation in biologically interesting aspects to be more readily visible.

Technical replicates are much like an un-interesting factor:

- No *information* about the level of the factor, just that all members of the technical replicate share some causal component.

- No *interest* in the consequences of the technical replicate, only in using the information they provide to reduce unexplained variation and hence increase power to make inferences about biological factors.
- No *replication* of the factor across different treatments, since the circumstance of each technical replicate is unique.

A classical statistical approach would interpret techincal replicates as a **repeated measure**, and construct a linear model that includes the repeated measure as a factor. Subsequent statistical tests then incorporate this information, in the form of a covariance or correlation between repeated measures, as part of the hypothesis being evaluated.

The `duplicateCorrelation` function of *limma* takes a slightly different approach. It is likely that technical variation has a modicum of consistency across genes. It might therefore make sense to use the information from *all* the genes to estimate an overall correlation due to technical replication. This overall estimate can then be used to improve the estimate of the correlation for individual genes.

Each chip in our data has two spots per gene. The layout is such that all genes are represented once, and then all genes represented a second time. The following code estimates and then summarizes the correlation between techinical replicates:

```
> dupCor <- duplicateCorrelation(exprs(M), design = X,
+     ndups = 2, spacing = nrow(M)/2)

> summary(dupCor[["atanh.correlations"]])

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.8290  0.3245  0.5869  0.6384  0.9033  2.7360

> dupCor[["consensus"]]

[1] 0.5399939
```

All correlation coefficients are transoformed so that they lie on the `atanh` scale (this is a common transformation for correlations, making their expected distribution more normal and hence the use of `atanh`-transformed correlation coefficients less problematic in analyses that rely on parametric assumptions). The `concensus` element is the correlation averaged over all genes; `atanh.correlations` is a vector of individual unique identifier correlations.

Including technical replication in our design should decrease the unexplained variation, and hence increase our ability to detect biologically interesting differences:

```
> dupFit <- lmFit(M, design = X, ndups = 2, spacing = nrow(M)/2,
+     correlation = dupCor[["consensus.correlation"]])
> dupFitE <- eBayes(dupFit)
> dupDecided <- decideTests(dupFitE)

> summary(dupDecided)

   (Intercept) SFNHigh HGFHigh SFNHigh:HGFHigh
-1        2196      28       0               0
0         2612    7357    7386            7386
1         2578       1       0               0
```

4 More complex models

Statistical linear models allow for complicated experimental designs, some of which can be incorporated into analysis with *limma*.

4.1 Including cell lines

Our experimental design includes several different cell lines:

```
> levels(exptlDesign[["cellLine"]])

[1] "32" "34" "36" "39" "40"
```

The cell lines represent a kind of replication, but the replication is different from technical replication. Specifically, we expect cell lines to show more idiosyncratic differences than technical replicates (i.e., genes from different cell lines are unlikley to shared a common correlation across spots). At the same time, we might want to include cell lines in our model for the same type of reason as we included technical replicates: cell lines could represent a source of variation that we can account for, and hence improve power to detect important differences in our model.

As a first approximation, we treat cell lines as a factor much like `SFN` or `HGF`; they can be thought of as a series of distinct levels. There are two important differences. The first is that there are not two but several levels of cell line. Linear models have been designed to handle this possibility, by constructing a model matrix that includes coefficients describing how each cell line deviates from the expected value of the first cell line:

```
> head(model.matrix(~cellLine, exptlDesign))

  (Intercept) cellLine34 cellLine36 cellLine39 cellLine40
1           1          1          0          0          0
2           1          1          0          0          0
3           1          1          0          0          0
4           1          1          0          0          0
5           1          0          0          0          0
6           1          0          0          0          0
```

The second difference is that there is no sense in which one particular line is 'larger' than another, in contrast to levels of `SFN` or `HGF`, which have a natural ordering. The real impact of this is on the coding of `SFN` and `HGF`, which if they had more than two levels (e.g., 'High', 'Medium', 'Low') would have had to be coded as **ordered factors** using the `ordered` function.

NOTE that cell line is really a **random effect** representing samples drawn from a population. We are not inherently interested in a particular cell line, but would instead like to draw inferences about the population from which the cell lines are drawn. For this reason, a more correct analysis treats cell lines as **random effects** and performs an analysis using a package like *nlme* to formulate a **mixed effects** model including both fixed (`SFN, HGF`) and random (`cellLine`, technical replicates) effects. It is likely that, in this particular case, treating cell line as fixed versus random effect likely has limited consequence for the analysis.

Returning to our analysis in `limma`, the process for incorporating more complicated designs into an analysis is to modify the model matrix. Again, the expectation is that by accounting for relatively uninteresting (in the present context) variation between cell lines, we get a more accurate portrayal of biologically interesting sources of variation

```
> X <- model.matrix(~cellLine + SFN * HGF, exptlDesign)
> dupCor <- duplicateCorrelation(exprs(M), design = X,
+     ndups = 2, spacing = nrow(M)/2)
> dupFit <- lmFit(M, design = X, ndups = 2, spacing = nrow(M)/2,
+     correlation = dupCor[["consensus.correlation"]])
> dupFitE <- eBayes(dupFit)
> dupDecided <- decideTests(dupFitE)
```

Highlights of this computation are:

```
> dupCor[["consensus.correlation"]]
```

```
[1] 0.3987836
```

```
> summary(dupDecided)
```

```
   (Intercept) cellLine34 cellLine36 cellLine39 cellLine40
-1        1829         52        548        306        705
0         3080       7256       6647       6921       6077
1         2477         78        191        159        604
   SFNHigh HGFHigh SFNHigh:HGFHigh
-1      57       0               0
0     7301    7386            7386
1       28       0               0
```

As before, we can identify spots with important effect:

```
> topTable(dupFitE, coef = 6)
```

```
           ID      logFC    AveExpr         t      P.Value
7144 F:198212 -1.1272666 -1.0048335 -7.673119 3.474078e-09
4719 F:195687 -1.2164112 -1.3040290 -7.594594 4.406089e-09
346  F:191098 -1.6656265  1.4291720 -7.591244 4.451050e-09
2259 F:193075 -2.1767644 -4.0267582 -7.410235 7.716038e-09
2307 F:193123 -2.2107254 -4.1050810 -7.140581 1.760874e-08
5939 F:196947 -1.9716221 -4.6211507 -7.017006 2.575268e-08
303  F:191055 -1.1387461  0.9491611 -6.859584 4.186602e-08
5987 F:196995 -2.0323852 -4.5237169 -6.687570 7.133514e-08
699  F:191455 -1.0130084 -0.3076729 -6.675975 7.394917e-08
4406 F:195338 -0.5501169 -0.4901482 -6.610026 9.076062e-08
        adj.P.Val         B
7144 1.095848e-05 10.798928
4719 1.095848e-05 10.580588
346  1.095848e-05 10.571257
2259 1.424766e-05 10.065120
2307 2.601163e-05  9.304381
5939 3.170154e-05  8.953263
303  4.417463e-05  8.503900
5987 6.068762e-05  8.010459
699  6.068762e-05  7.977113
4406 6.185010e-05  7.787260
```

4.2 Assessing model performance

The results so far emphasize fitting a model and identifying features with significant effect. How do we know we are on the right track?

To start addressing this question, look at the residuals of our fitted models. *limma* allows us to calculate a matrix of fitted values from our model

```
> fittedVals <- fitted(dupFitE)
> dim(fittedVals)

[1] 7386   20
```

There is one fitted value for each (duplicated) feature and sample. We can ask how these fitted values compare to the actual values, and for convenience we will calculate a single observed value by averaging the two duplicate spots:

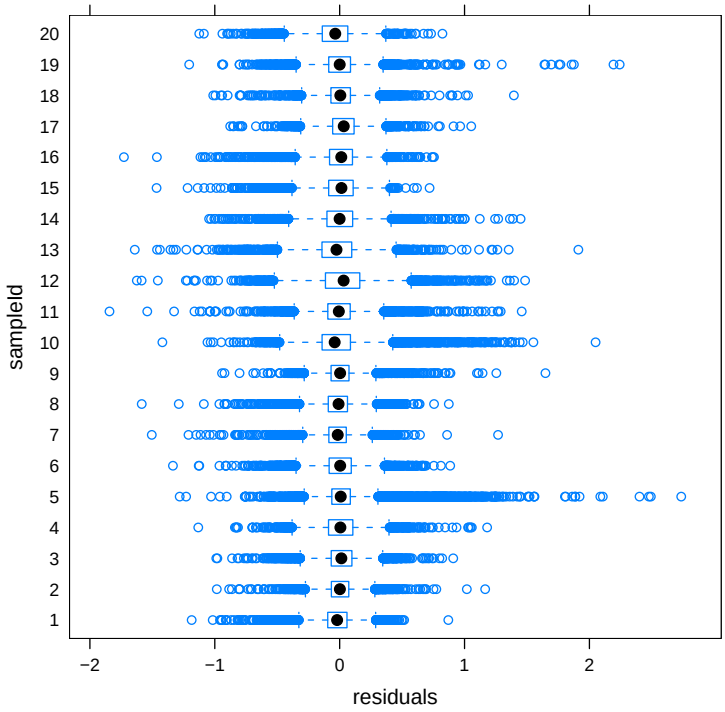
```
> observedVals <- (exprs(M)[1:(nrow(M)/2), ] + exprs(M)[nrow(M)/2 +
+     1:(nrow(M)/2), ])/2
> residualVals <- residuals(dupFitE, observedVals)
```

Each of the expression, fitted and residual values are formatted as a matrix, but it is convenient to flatten them each into data frame columns, and to add appropriate indicators of the experimental design

```
> df <- data.frame(sampleId = rep(1:20, each = nrow(observedVals)),
+     SFN = rep(exptlDesign[["SFN"]], each = nrow(observedVals)),
+     HGF = rep(exptlDesign[["HGF"]], each = nrow(observedVals)),
+     observed = as.vector(observedVals), fitted = as.vector(fittedVals),
+     residuals = as.vector(residualVals))
```

The residuals of each sample and feature should be independent of one another, so that no sample should have only 'large' residuals. Here is a quick check:

```
> library(lattice)
> print(bwplot(sampleId ~ residuals, df))
```



Additional plots for exploration are hinted at in the lab.

5 Further directions

The following topics are covered by the *limma* manual, and provide a taste of the advanced directions for analysis of complex experimental designs and the tools available for their analysis.

Dye swap Section 8.1.2, p. 35.

Common reference Section 8.4, p. 40.

Several groups Section 8.6, p. 44.

Time series Section 8.8, p. 48.

The *limma* package provides great value-added benefits (e.g., constructing contrasts, incorporating technical replicates, Bayesian assessment of significance). All of this functionality is, however, implemented in R, so all these techniques (and more) are directly accessible to the user. For instance, the R

function `lm` fits linear models, and can be used to perform identical analyses to those of `lmFit` (though in a less computationally efficient way).

Many alternatives to the analysis outlined here exist. For instance, *a priori* knowledge might suggest a subset of genes for an analysis, a subset of genes might be expected to show a correlated response, the linear model might have random as well as fixed effects, etc. Package exist for some of these problems (see the BiocViews web page), others represent formulations of classical statistical problems that R is able to accomodate. Still others remain areas for active research.