Description of Envisage

Sam Robson

June 26, 2008

Contents

1 Introduction

1.1 Motivation

The development of high-density microarrays for gene expression analysis has been a breakthrough in functional genomic studies, allowing parallel analyses of the expression of many thousands of genes simultaneously at the transcriptome level.

Such high-throughput data require suitably advanced methods of statistical hypothesis testing, such as analysis of variance, to ensure that measured changes in gene expression represent true biological events. Noise, due to a combination of technical, environmental, biological and random variation, can become a major problem in any microarray analysis and, whilst this can be reduced somewhat with careful experiment planning and implementation, statistical methods must be used to isolate real changes in gene expression from those occurring by random chance. To increase the power of statistical hypothesis testing techniques, replicate data must be collected (7).

It is all but impossible to design an experiment such that the variable(s) under the direct control of the experimenter ('parameter(s)') are the sole sources of variation within the experiment. Unavoidable environmental effects, nuisance variables such as batching effects, and phenotypic differences in *in vivo* experimental samples will effect the response of the gene expression to the parameters of interest. These 'covariates' must be considered when testing for significant changes in gene expression for the experiment parameters to ensure the biological relevence of the results.

1.2 Envisage

Envisage (Enable Numerous Variables In Significance Analysis of Gene Expression) is a package developed for the Bioconductor project in R (4) that utilises an automated linear model (LM) selection process to extend ANOVA to a wide range of predictor variables, including experimental parameters (drug treatment, disease state, time point, etc.), phenotypic covariates (gender, age, height, weight, etc.) and nuisance covariates (such as batch effects). This allows the inclusion of a wider range of variables, be they interesting experiment parameters or unavoidable covariates, when testing for significance.

The inclusion of such confounding variables in the significance tests provides a more accurate representation of the true biology of the system under study. This process would be particularly useful for clinical studies, where the between-sample variation can be very high within each condition due to the difficulties in complete patient matching in a clinical setting.

The **Envisage** package has been designed to be user friendly, with a simple to use graphical user interface (GUI), described in section 3.2, for those that don't want to use the command line interface of R. Also, **Envisage** can be run on an object of class *ExpressionSet* within the R environment, or alternatively can be run directly from the gene expression analysis suite GeneSpring GX[1]. Use of **Envisage** within Bioconductor is described in 3.1.1, and through the GeneSpring GX plug-in external program interface in section 3.1.2.

1.3 The Envisage Modelling Procedure

This section gives a brief overview of the steps involved in the **Envisage** model selection procedure. The input file for **Envisage** is an object of class *ExpressionSet*, created within Bioconductor (section 3.1.1) or created automatically using the GeneSpring EPI plugin (section 3.1.2). This object contains information about probe level signals for the array, details on experimental variables for each sample, and annotation lists that can be used to link probes to their respective gene targets.

The model fitting procedure is a two step operation. First, a model is fit containing only main effect variables. Next, a more complex model is fit based on the first order interactions of variables found to be significant in the first step. This model is selected by first fitting a saturated model containing all available terms, and then removing terms based on the Akaike information criterion. This is essentially a penalised log-likelihood

---

[1]Agilent Technologies, Santa Clara, CA

function that determines the benefit of keeping terms in the model with respect to its explanatory power; if the addition of the term has no clear benefit to the fit of the model, it is deemed unnecessary and removed. This two-step method was chosen to reduce run-time at the risk of missing important interaction information.

A type II ANOVA test is used to test the significance of each of the model terms for each gene. The model for each gene is compared with and without each of the terms, and a p-value is calculated using an F-test statistic. A table of unadjusted p-values is created with genes as rows and all model terms as columns.

It is recommended that a multiple testing correction procedure such as the Benjamini & Hochberg false discovery rate (1) adjustment be applied to the p-values to correct the matrix of p-values to take the large number of individual statistical test performed into account. Available multiple testing corrections can be found in section 4.3). Corrected p-values are compared to the specified p-value cutoff to test for significant differential expression. Genes showing a significant ANOVA F-test p-value for a particular term are considered to be changing significantly based on that variable. A list of genes with significant p-values is output to the user for each term as a tab-delimited text files.

For more details on the model selection procedure, please see Robson *et al.*(manuscript in preparation).

2 Getting Started

The following description of the **Envisage** package is focused on the analysis of single-channel microarray data such as Affymetrix GeneChips, but the process can easily be extended to two-colour microarray and even other high-throughput data such as mass spectrometry data. The primary input file for the package is an object of class *ExpressionSet*, a class of objects in R for representing microarray data. The object must contain all necessary gene expression information and phenodata (experiment variable information) for each sample, and also annotation information for the genome of interest. Relevant information for MIAME (Minimum Information About a Microarray Experiment) compliance can also be entered (3).

Envisage can be run within the R environment using an *ExpressionSet* object created using the appropriate package in Bioconductor (**affy** for 1-colour data), or the package can be run directly through the Genespring GX analysis suite, which will automatically create the required object from the current Genespring GX experiment interpretation.

This chapter gives information on setting up the R environment, installing the Bioconductor packages, installing the **Envisage** package, and creating the required Bioconductor object (both in Bioconductor itself and Genespring GX). Information for running the program itself can be found in section 3.

2.1 Getting Envisage Up and Running

2.1.1 Installing R and Bioconductor

Envisage is a package available from Bioconductor, a publicly available collection of packages for the statistical programming language R, allowing statistical and graphical methods for analysis of gene expression data.

The R programming environment can be downloaded from the comprehensive R network archive (*CRAN*) at www.cran.r-project.org. The current release, R-2.6-0, was released on 2007-10-03. The latest release of Bioconductor (Bioconductor 2.1 – 2007-10-08) can be found at www.bioconductor.org.

The simplest way to set up Bioconductor is to run it from the R environment, using the *biocLite.R* installation script. This will install the basic packages required for gene expression analysis; **affy, affydata, affyPLM, annaffy, annotate, Biobase, Biostrings, DynDoc, gcrma, genefilter, geneplotter, hgu95av2, limma, marray, matchprobes, multtest, reposTools, ROC, vsn** and **xtable**.

To do this, open the R environment, and type the following in the command line:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

This will establish a connection to the Bioconductor website and initialise installation. To install additional packages, use the package name as an argument in the biocLite.R method. e.g. To install packages **affy** and **marray**, type the following in the command line:

```
> biocLite(c("affy", "marray"))
```

The *getBioC.R* installation script can also be used, which contains a larger number of default packages than the *biocLite.R* script. To install this, follow the above steps, replacing *'biocLite'* with *'getBioC'*:

```
> source("http://bioconductor.org/getBioC.R")
> getBioC()
```

Once packages have been installed, they must be loaded into the R environment to allow use of the methods contained within the package. This must be done at the start of every session. To do this, use the `library()` method. e.g. To load package **affy**, type the following into the command line:

```
> library("affy")
```

Packages can also be downloaded, installed and loaded from the *'Packages'* tab at the top of the R shell. Selecting *'Install package(s)'* brings up a list of available mirrors for the CRAN depository. Select the most suitable mirror, and a list of available packages is displayed. Simply select the package of interest and choose *'OK'*. The package will be downloaded and installed automatically. Selecting *'Load package...'* brings up a list of packages available to be loaded. Selecting the required package and selecting *'OK'* has the same effect as using the `library()` function.

Bioconductor packages are well documented, with help files available for all user-level functions. Every package contains at least one *vignette*, an executable document consisting of a collection of code and documentation text, providing information regarding usage of the package. These vignettes can be found in the *doc* subdirectory of an installed package, or can be called from the R command line using the `openVignette()` method from the Biobase package. To view documentation for package **affy**, type the following into the command line:

```
> library("Biobase", "affy")
> openVignette(package = "affy")
```

Help files for particular methods can be called internally within R by using the following methods, replacing 'affy' with the name of the required method:

```
> help.start()
> help.search("affy")
> help(affy)
> `?`(affy)
> apropos(affy)
> example(affy)
> demo()
> demo(affy)
```

2.1.2 Installing Additional Required Packages in R

Envisage requires the following additional packages which can be obtained from Bioconductor as previously described:

multtest Multiple testing procedures for controlling the family-wise error rate (FWER) and false discovery rate (FDR)

GeneSpring (OPTIONAL) Package for creating *ExpressionSet* object from GeneSpring GX experiment interpretations

tcltk Package for integration of Tcl/Tk libraries and widgets within the R environment

tkWidgets Package containing R based widgets from the Tcl/Tk libraries

Envisage also requires the following additional packages which can be obtained from the comprehensive R archive network (CRAN) by going to the CRAN website at http://www.r-project.org/, selecting "CRAN" under 'downloads', and selecting a CRAN mirror site from the ones available. Available packages can be obtained by clicking on "Contributed extension packages":

car Package accompanying J. Fox, *An R and S-PLUS Companion to Applied Regression* (2002) and contains functions for applied regression, linear models, and generalized linear models, including the method `Anova()` which allows Type II ANOVA testing.

2.1.3 Installing Envisage in R

Envisage is available as a package within Bioconductor and can be installed directly through the R shell from the '*Packages*' tab as described previously in section 2.1.1. Alternatively, the package source files can be obtained from the author's website at www2.warwick.ac.uk/fac/sci/moac/currentstudents/2003/sam_robson/linear_models/downloads/. To install the package in R, please do one of the following:

UNIX/LINUX To install the **Envisage** source file (`.tar.gz`), first ensure that your `.../R/bin/` path is included in your PATH environment variable, and type the following into the terminal: "R CMD INSTALL [full path to source tar.gz file]".

WINDOWS To install the **Envisage** binary file on a machine running windows (`.zip`)go to the '*Packages*' tab and select '*Install package(s) from local zip files...*'. Navigate to the destination folder to which the **Envisage** .zip file has been saved and select it. R will install the package automatically.

To load the package in R, simply use the `library()` method:

```
> library(Envisage)
```

2.1.4 Installing Tcl/Tk

The graphical interface for **Envisage** has been developed using the R based integration of Tcl/Tk developed by Peter Dalgaard in the **tcltk** package. This package can be installed directly through the R shell from the '*Packages*' tab as described previously in section 2.1.1. The **Envisage** GUI also uses widgets that are currently only implemented through the **BWidgets** and **Iwidgets** Tcl/Tk packages that are currently not supported by **tcltk**. To allow integration of R with these packages, users must install a current version of Tcl/Tk, such as the ActiveTcl distribution from ActiveState[2]. The following is a step-by-step guide to installing ActiveTcl and loading the **BWidgets** and **Iwidgets**package for use in R:

1. Download the current distribution of ActiveTcl (release 8.4.16) free from the developer's website at www.activestate.com/Products/activetcl/.

2. Install ActiveTcl as instructed.

---

[2]www.activestate.com

3. **For Windows**

   Go to your computers advanced system settings and select '*Environmental Variables*'. Under `System Variables`, select '*New...*' to create a new environment variable called `MY_TCLTK` with value '`yes`'.

   For LINUX

   Create a new global environment variable called `MY_TCLTK` with value '`yes`' by updating your profile file (eg .bash_profile) by typing the following into the shell:

   export MY_TCLTK = "yes"

4. Repeat the above to create a new environment variable called `TCL_LIBRARY` and set the value to the library path of your installation of ActiveTcl (e.g. "`C:/Tcl/lib`" if you keep the default settings).

5. In R, use the `addTclPath(path)` method to tell R where to find the ActiveTcl library files, where `path` is the library path of your installation of ActiveTcl (e.g. "`addTclPath('C:/Tcl/lib')`" if you keep the default settings).

You will now be able to use widgets from all packages contained in the full version of Tcl/Tk.

2.1.5 (OPTIONAL) Installing GeneSpring-R Integration External Program Interface

A script is available for those using Genespring GX for analysis of their microarray data, which allows use of the **Envisage** package from within the Genespring GX environment. This script will export the selected experiment interpretation and gene list into an R environment, and use these to create a Bioconductor *ExpressionSet* object.

Genespring GX uses an External Program Interface (EPI) to allow interaction between the Genespring GX environment and the R environment. A file, `GS_R_in.txt`, is created in the Genespring GX *'Data'* folder, and the batch file `GS_exec_R_GUI.bat` opens the R environment directly. The function `GSload.int` in package **GeneSpring** can be called on the input file `GS_R_in.txt` to load the Genespring GX experiment interpretation as an object of class `GSint`. This can be converted directly to an object of class *ExpressionSet* using the **GeneSpring** function `GSint2BC()`.

Note: The current `GSint2BC()` is defunct since it still uses the old `exprSet` class for Bioconductor objects. The **Envisage** package contains a modified function, `GS2BC`, which has been updated to create a Bioconductor object of class *ExpressionSet*.

The R script `Envisage.R` runs the function `Envisage()` on the GeneSpring `ExpressionSet` object. Output genelists are returned in a tab-delimited text format as described in section 3.3 that can be read into the GeneSpring environment using the '*Load Gene List From File*' and '*Load Gene List With Numbers From File*' JAVA class external plugins. These are included in the Genespring-R integration package, and can be found in the '*External Programs*' folder to the left of the main Genespring GX window).

The following is a step-by-step guide to installing the GeneSpring-R Integration EPI to allow analysis of microarray data using **Envisage** directly through GeneSpring GX:

1. Download the **Genespring-R** integration package *.zip* file from the Agilent website using one of the following links:

   - For WINDOWS including R program (2.1.0) and BioConductor & GeneSpring library – www.chem.agilent.com/cag/bsp/SiG/Downloads/zip/gs_r_winall.zip

   - For WINDOWS NOT including R program or any library – www.chem.agilent.com/cag/bsp/SiG/Downloads/zip/gs_r_win.zip

   - For UNIX NOT including R program or any library – www.chem.agilent.com/cag/bsp/SiG/Downloads/zip/gs_r_unix.zip

2. Unzip the **Genespring-R** integration package directly in Genespring GX by dragging the *.zip* file into any active Genespring GX window. Follow the setup procedure as instructed.

3. Download and install the **GeneSpring** Bioconductor package from the Bioconductor website as described in section 2.1.1.

4. Download the files `Envisage.programdef` and `Envisage.R` from the author's website at www2.warwick.ac.uk/fac/sci/moac/currentstudents/2003/sam_robson/linear_models/downloads/. The `Envisage.programdef` file should be added to the "*. . . /Agilent/Genespring GX/data/Programs*" folder, and the `Envisage.R` file should be added to the "*. . . /Agilent/Genespring GX/data*" folder. Upon loading Genespring GX, the external program **Envisage** will be available from the "*External Programs*" folder (see figure 1).

2.2 Requirements

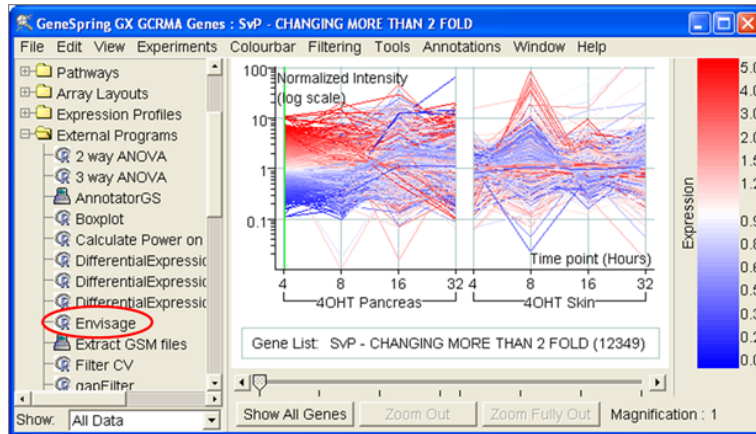The following are required for package **Envisage**:

Figure 1: Selecting Envisage from Genespring GX.

- R version 2.6.1 or higher

- Bioconductor version 1.9 or higher

- tkWidgets version 1.16.0 or higher

- R packages **tcltk, car**

- Bioconductor packages **affy, multtest**

- Bioconductor package **GeneSpring** (if pre-processing data using Genespring GX)

- tcl packages **BWidgets** and **Iwidgets** (available as part of ActiveTcl.)

2.3   Data Pre-Processing

Pre-processing and sample quality control must be carried out before using **Envisage** to avoid biasing the modelling procedures. This can be performed in Bioconductor, or within the gene expression analysis suite GeneSpring GX. The modelling procedure uses log-transformed gene expression data. In this way, the additive linear model corresponds to a multiplicative model on the original non-logged expression values. **Envisage** will automatically adjust the gene expression data if it is not log-transformed, but users are encouraged to perform this step before beginning the procedure.

It is also recommended that in the quality control step, appropriate normalisation steps are used to limit the effects of outlying samples. It is left to the user to decide the level to which this quality control is performed. Quality control of genes should also be carried out across the samples, firstly to limit the number of genes used in the calculation (thus saving on run-time), and secondly to ensure that erroneous genes do

11

not elicit a significant effect. For instance, non-changing genes should be removed from the experiment interpretation to avoid biasing in the multiple testing correction stage.

The *ExpressionSet* class is the basic container of microarray data within Bioconductor and contains the following slots:

`phenoData` Variable information describing the phenotype for a microarray sample (such as age, weight, treatment, etc.). This is stored as an object of class `Annotated-DataFrame`, which itself contains two slots; the `phenoData` slot containing the phenotype data, and the `varMetaData` slot containing information about the phenotype variables. See `help(class:AnnotatedDataFrame)` for more details.

`experimentData` Experimental information required for MIAME compliance (see 3). See `help(class:MIAME)` for more details.

`annotation` Label associated with experiment annotation.

`exprs` A matrix of gene expression values for each sample.

These slots can be filled either directly in Bioconductor, using the **affy** package for 1-colour data, or the **marray** package for 2-colour data (for further assistance on data pre-processing in Bioconductor, please see the appropriate package vignette), or by using the GeneSpring-R integration EPI as described previously in section 2.1.5.

3 Running Envisage

3.1 Initialisation of Envisage

3.1.1 Initialising Envisage from Bioconductor

To run **Envisage** from Bioconductor, follow these steps:

1. Load the R environment as normal. Create an object of class *ExpressionSet* containing all of your experiment data (for instance, for Affymetrix data use the `ReadAffy()` function from package **affy** to load in information from the experiment `.CEL` files).

2. Ensure that package **Envisage** has been correctly installed and load it as described in section 2.1.1.

3. Either load the **Envisage** GUI by using the function call `envisage()` with argument `widget` set to TRUE:

   ```
   > envisage(expData, widget = TRUE)
   ```

   Or run `envisage()` through the command line by setting the argument `widget` to FALSE and providing further arguments as described in section 4.3. *'expData'* is the name of the experiment *ExpressionSet* object for analysis. Use of the **Envisage** GUI is described in section 3.2.

   If the GUI is used, the results will be assigned to a global variable '`results`' so please be sure that this will not overwrite any other objects within the R environment. If using `envisage()` through the command line, users can assign the results to an object of their choice.

3.1.2 Initialising Envisage from Genespring

To run **Envisage** from GeneSpring GX, follow these steps:

1. Ensure that the **GeneSpring** Bioconductor package has been downloaded and installed as described in section 2.1.1.

2. Ensure that the `Envisage.programdef` and `Envisage.R` files have been downloaded and installed as described in section 2.1.5.
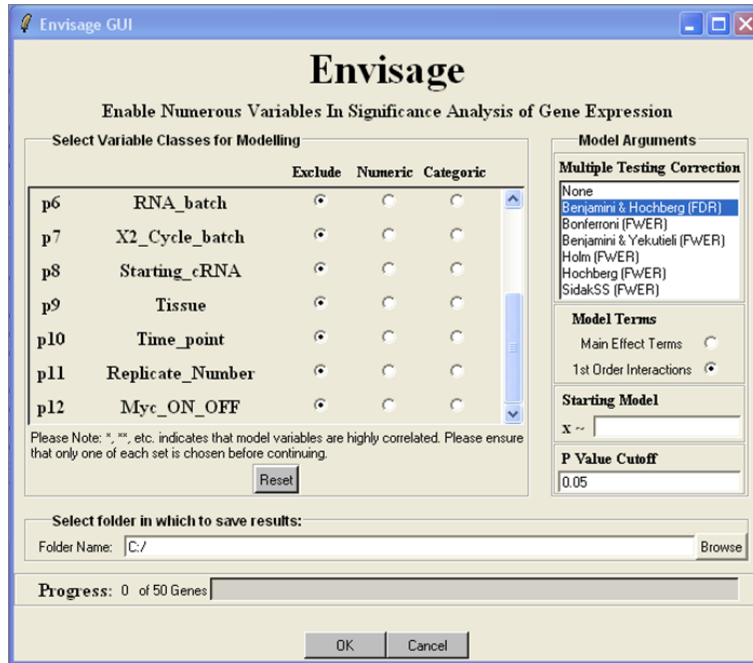
Figure 2: The Envisage GUI.

3. Ensure that the Genespring-R integration package has been downloaded and installed as described in section 2.1.5.

4. Load the Genespring GX environment as normal. Ensure that *'Envisage'* is available from the *'External Programs'* folder (figure 1).

5. Select your normalised experiment interpretation and curated gene list and ensure that the *'All Samples'* experiment interpretation is selected. Selecting the *'All Samples'* interpretation ensures that samples are not grouped.

6. Finally, select the *Envisage* icon in the *'External Programs'* drop-down menu to the left of the interpretation view window. This will load the **Envisage** GUI, as described in section 3.2.

3.2 Envisage Graphical User Interface

Envisage utilises a graphical user interface (GUI) for input of arguments to the `envisage()` method. This prevents errors in argument input.

3.2.1 Selection of Variables Classes

The main GUI window (figure 2) lists the experiment interpretation variables, and allows the user to specify which of these to use in the model calculations, and what class of variable they are. The experiment interpretation may contain several variables that contain identical information (for instance, one variable may contain a range of values, and another may contain the averages of those ranges). Any such correlated variables are flagged with star symbols ('*', '**', '***', etc) which should not be confused with similar flags used for significance in ANOVA tables. Only one of these highly correlated variables should be used in the modelling procedure to avoid overfitting of the data, so users should be sure to select 'None' for all but at most one variable flagged with the same number of stars. Selecting more than one may result in overfitting of the data and errors in the hypothesis tests.

Users must specify the class of each of the variables to be used in the model by using the variable class radio buttons. By default, None is selected for all variables. Selecting None prevents a variable from being included in the modelling procedure. The two available classes are numeric, for variables where each value has a numerical relevance (e.g. weight, height, etc.), and categoric, for variables that have a small number of specific factor levels (e.g. gender, disease state, etc.). Choosing the correct class for each of the variables is essential for the modelling procedure. Selecting an incorrect class can result in the modelling procedure failing.

The automated model selection process occurs in two steps. The first step optimises a model with only main effect terms to find those variables that have the greatest effect on the data. The second step calculates a model including first order interactions for the variables from the main effect model. However, the user can choose to consider only main effect terms if necessary. This may be done as a preliminary observation to see which variables are of interest, or possibly because the number of samples is low compared to the number of variables of interest, providing few degrees of freedom for the model.

If the user chooses to look at interaction terms, but there are too many variable terms compared to the number of samples, the user will be prompted to remove some variables from the first order interaction calculations (figure 3). The number of degrees of freedom for the fitted model is dependent on the number of samples, so the size of the experiment will determine the variables that can be included. If some categoric variables have a particularly large number of levels, it is recommended to remove these from the interaction calculations to avoid over-fitting of the data.

To ensure that variable names do not cause an internal error within R, they may be
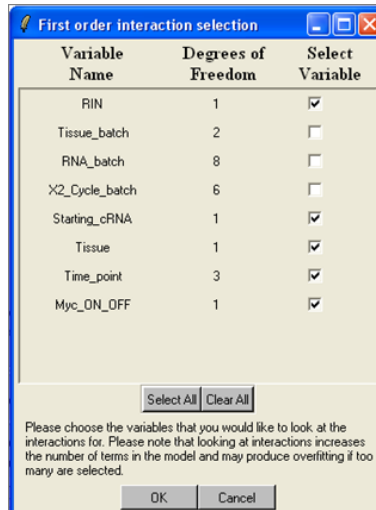
Figure 3: Selection of variables for which to calculate interactions.

renamed by the program to ensure compatability. For instance, R does not like objects to have names beginning with a numeric character (e.g. '`2-Cycle batch`'). If such a name is discovered, it will be prefixed with the letter '`X`' to prevent errors. Also, any hyphens, slash characters or spaces will be removed and replaced with the character '`_`' for the same reason.

3.2.2 Envisage GUI Arguments

Multiple Testing Correction A number of multiple testing correction algorithms are available for use with the **Envisage** package. They are called through the Bioconductor package **multtest**. More information can be obtained by reading the **multtest** package vignette. It is suggested that a multiple testing correction procedure is used due to the large number of genes being analysed. A list of available multiple testing corrections can be found in section 4.3). A false discovery rate correction is recommended, and the method of Benjamini and Hochberg is selected by default (1).

Model Terms The user can choose to model only main effect terms with no interactions. If there are a large number of variables to model compared to the number of samples, this may be a useful preliminary analysis to find the most significant variables which may then be used with interactions.

Starting Model If users have some idea of the variables that they would like to see in the per-gene model, they can specify the minimum model to force these terms to

16

be present for all genes. In this way, the effect of variables of interest to the user can be studied whilst allowing for leeway in the selection of further terms for each gene. Variables are identified by a number with the prefix 'p'. These identifiers are used to specify the model. For instance, if you want to force variables 'p1' and 'p2' into the per-gene model, type 'p1 + p2' into the appropriate field. The syntax for this field follow the usual standards for model entry, with '+' indicating inclusion of a variable, ':' indicating an interation between two or more variables, and '*' indicating the inclusion of two or more variables and all possible interactions (e.g. "A*B" = "A + B + A:B").

P-Value Cutoff The *p-Value Cutoff* defines the significance cutoff for the type II ANOVA tests. A default of 0.05 is used by convention.

Results folder Finally, the folder in which to save the output files (described in section 3.3) can be specified. A new folder named *EnvisageResults* is created within the specified directory, containing all of the **Envisage** output files.

3.3 Envisage Output

Once the modelling procedure is completed, results are output for significantly changing genes for each term within the analysis. If the GUI is used, a new window is created (figure 4) allowing the user to specify for which of the variables (and subsequent first-order interactions) to output the results. If the output directory *EnvisageResults* contains files from a previous analysis, these will be suffixed with the letters `_OLD` to avoid overwriting data.

For each model term, the column *'All Changing Genes'* gives the percentage of the genes that show the term as significant in the model. The column *'Significantly Changing Genes'* gives the percentage of all genes that change significantly and also have a corrected p-value less than the specified cutoff range. These values give the user a brief idea as to which of the model terms have a significant effect on the gene expression.

Tick the boxes for all model terms that you require results for and press *OK* (selecting '*Check All*' will output all significant results). This will save the gene names and corresponding p-values into a tab-delimited text format in the folder *EnvisageResults* in the specified folder. These files are named based on the variable names. Any spaces in the variable names are replaced by '_' to ensure file name compatability. Due to the convention of specifying the interaction of two variables using a colon (e.g. `param1:param2`), which is not a valid character for specifying filenames, it is replaced by the characters

Figure 4: The Envisage output selection GUI.

'_INT_'.

The *EnvisageResults* folder also contains information regarding genes that show problems in the ANOVA tests. We can see aliasing errors in the ANOVA calculation for interactions due to one of two reasons:

1. Extrinsic aliasing is due to the structure of the data – There is no relevant data for a particular interaction group (e.g. there are no diseased patients in a particular age group, so the effects for the interaction *diseased:age* cannot be estimated).

2. Intrinsic aliasing is due to the structure of the data – Particular effects cannot be separated adequately (e.g. all diseased individuals in a particular age group also have the same blood group. In this case we cannot separate the effects in *diseased:age* and *diseased:blood group*).

Extrinsic aliasing will generally occur due to the use of an unbalanced data set, but will not cause a problem with the significance calculations. Intrinsic aliasing is indicative of correlation between explanatory variables. So the model will fit one variable and then try to fit the same information again with the second variable. This may then pose a problem since the significance results will depend on the order in which the terms are fitted.

The two files produced, *AliasGenes* and *AliasReport*, give information on the genes showing intrinsic aliasing. *AliasGenes* is a list of the aliased genes in a tab-delimited text file. *AliasReport* gives more detailed information showing exactly which of the variable levels are aliased. This may be useful for users to spot correlation between their variables

18

that may be unexpected. Users may wish to run the program a second time with one of the two aliased variables removed.

If for some reason a gene gives a model for which the ANOVA calculation fails, the name of the gene is output to a file called *FailedAnova*. This file should generally be absent.

4 Program Specifications

4.1 User Methods

The following methods are available for users:

envisage This is the main function for the **Envisage** package. It takes in microarray gene expression data and variable phenodata in the form of a Bioconductor object of class *ExpressionSet*, and returns a list of genes that vary significantly based on the experiment variables and their first-order interactions.

GS2BC Updated version of the `GSint2BC()` function from package **GeneSpring** which converts GeneSpring experiment interpretation data into an object of class *ExpressionSet* for use with **Envisage**.

4.2 envisage() Argument Definition

The following arguments are required for the user-level function `envisage()`:

expData The data construct, of class *ExpressionSet*, containing all gene expression and phenotypic data for the experiment.

widget A logical value specifying whether or not to use the widget based GUI for argument specification. This is set to `TRUE` by default.

MTC Multiple testing correction to use for correcting calculated p-values (see section 4.3).

pCutoff A number defining the p-value cutoff to be used in significance analysis. Must be between 0 and 1.

useParams A character vector that defines which of the experiment variables to use in the model calculation.

param4INT A character vector that defines the model variables for which interactions should be calculated.

paramType A character vector specifying the classes ("*numeric*" or "*categoric*") of the variables in the model.

fileResults A character string defining the folder in which to save the results of the modelling procedure.

MEorINT A character string defining whether to consider only main effect terms in the model("*ME*"), or to look at first order interactions("*INT*").

startModel A character string representing the starting model for analysis. Must be of the form "$x \sim term1 + term2 + term1 : term2 + \ldots$". Care must be taken to ensure that the model terms match the names of variables specified in `useParams`.

The following are the default argument terms:

```
> envisage(expData, widget = TRUE, MTC = NULL, pCutoff = NULL,
+     useParams = NULL, param4INT = NULL, paramType = NULL, fileResults = getwd(),
+     MEorINT = "INT", startModel = NULL)
```

The following function call is sufficient if you want to run **Envisage** using the GUI:

```
> envisage(expData)
```

4.3 envisage() Argument Information

The following information is necessary when defining the arguments for `envisage()`:

expData Try to ensure that explanatory variables are labelled sensibly. Avoid beginning variable names with a numeric character and using special characters in variable names. **Envisage** contains several basic internal functions for renaming variables to avoid these problems, but this should be considered by the user. Spaces are acceptable in variable names, but will be replaced by an underscore in output files.

widget Default is 'TRUE'. If set to FALSE, great care must be taken to ensure that other variables are correctly implemented.

MTC Multiple testing corresction procedures are implemented through the package **multtest**. Available multiple testing procedures are as follows:

1. "**NA**" – No MTC selected
2. "**BH**" – (Default) Benjamini & Hochberg step-up false discovery rate (1995) (1)
3. "**BY**" – Benjamini & Yekutieli step-up false discovery rate (2001) (2)
4. "**Bonferroni**" – Bonferroni single-step family wise error rate
5. "**Holm**" – Holm step-down family wise error rate (1979) (6)
6. "**Hochberg**" – Hochberg step-up family wise error rate (1988) (5)

7. "**SidakSS**" – Sidak single-step family wise error rate

8. "**SidakSD**" – Sidak step-down family wise error rate

pCutoff Must be between 0 and 1. Default is $p = 0.05$.

useParams Users must ensure that no labelling variables are included in the list of variables to use. Users must also ensure that only one of any group of highly correlated variables is used, so be careful to select variables for modelling that represent different aspects of the data. Also be careful to ensure constant naming for all variables. This problem can be avoided by using the **Envisage** GUI.

param4INT Be careful to ensure constant naming with variables specified in the `useParams` argument. This problem can be avoided by using the **Envisage** GUI.

paramType Variable class must be one of "numeric" or "categoric". Users must be sure that the order of the variable classes is correct to ensure that variables are assigned the correct class. Assigning the wrong class to a variable can have a dramatic impact on the modelling procedure. This problem can be avoided by using the **Envisage** GUI.

fileResults Default is the working directory for R. Be sure to choose a directory that can be accessed.

MEorINT Default is "`INT`".

startModel This argument allows the user to specify a minimum model for use in the modelling procedure to force terms into the selected model. This model must be specified as a character string of the form "$x \sim term1 + term2 + term1 : term2 + \ldots$", which will be converted internally to an object of class formula. Care must be taken to ensure that the model terms match the names of variables specified in `useParams`.

4.4   Non-User Methods

The following methods are called by the user-level function `envisage()`, and should not be directly called by the user:

envisage.data This function takes the experiment data and extracts all relevant information (gene names, variable names, gene expression levels, etc) and assigns them to individual objects in R for future use. This function also ensures that the gene

expression data is logged, so `envisage()` can take both logged and non-logged data as its argument.

alias.glmNA This function directly extends the function `alias()` to objects of class `glmNA`. Class `glmNA` directly extend the classes `lm` and `glm` to objects that show aliasing in the modeling procedure. This allows the use of the type II ANOVA function.

coef.glmNA This function directly extends the function `coef()` to objects of class `glmNA`. Class `glmNA` directly extend the classes `lm` and `glm` to objects that show aliasing in the modeling procedure. This allows the use of the type II ANOVA function.

corParams This function flags highly correlated variables that may model the same information in the model. Parameters showing a correlation greater than 0.98 are flagged.

labelParams This function removes any labelling variables (variables whereby each sample has a unique value) from the experiment interpretation.

modelCalcME This function uses a stepwise method to fit a linear model for each gene based on main effect variables.

modelCalcINT This function takes the main effect model of `modelCalcME` and uses only the significant terms to fit a linear model including first-order interaction terms in a stepwise process for each gene. A type II ANOVA calculation is used to calculate significance p-values for each model term based on an F-statistic.

mulTestCor This function corrects the p-values for the type II ANOVA based on the specified multiple testing correction.

result.output This function compares the significance p-values for each gene and variable with the specified p-value cutoff. If a gene has a p-value within the limit for a particular variable or interaction, then that gene is added to a gene list for the variable. These gene lists are output to a folder specified by the user.

INTlevels This function calculates the number of levels for each main effect and first-order interaction term in the linear model. This is used to check model degrees of freedom against available degrees of freedom.

modelEntry This function allows users to specify a minimum model for the model selection process.

4.5 S3 Classes

The following classes are defined in the package **Envisage**:

class-glmNA Class `glmNA` directly extend the classes `lm` and `glm` to objects that show aliasing in the linear model.

5 An Example Envisage Session

We now demonstrate an example session using an example data set, `SkinvsPancreas`, available as part of the **Envisage** package. The example data set is an object of class *ExpressionSet* made up of gene expression levels for 50 probes over 48 individual Affymetrix MOE430 Plus 2 mouse GeneChips. This is a subset of a larger data set for which a large number of experiment variables were recorded (Robson et al., In Preparation). The experiment design is multi-factorial, with 3 main explanatory variables; tissue type (2 levels - skin and pancreas), time point after activation of the Myc protein (4 levels - 4, 8, 16 and 32 hours) and status of the Myc transgene (2 levels - Myc ON or Myc OFF). Three batching variables correspond to batching of samples at the tissue extraction stage, RNA extraction stage and the microarray hybridisation stage. Further experiment information, such as RNA quality and yield, was collected providing a total of 12 variables. This data set is thus well suited for use with **Envisage**.

The first step is to load the package into R, then load the example data set:

```
> library("Envisage")
> data(SkinvsPancreas)
```

This will load the example *ExpressionSet* object `SkinvsPancreas` into the R environment:

```
> SkinvsPancreas

ExpressionSet (storageMode: lockedEnvironment)
assayData: 50 features, 48 samples
  element names: exprs
phenoData
  rowNames: 1, 2, ..., 48  (48 total)
  varLabels and varMetadata description:
    Sample name: Sample name
    Treatment: Treatment
    ...: ...
    Myc ON/OFF: Myc ON/OFF
    (12 total)
featureData
  featureNames: 1416783_at, 1416780_at, ..., 1416685_s_at  (50 total)
  fvarLabels and fvarMetadata description: none
```

```
experimentData: use 'experimentData(object)'
Annotation:
```

Parameter names can be seen by looking at the `phenoData` slot of `SkinvsPancreas`:

```
> names(pData(SkinvsPancreas))

 [1] "Sample name"      "Treatment"      "RIN"              "Nanodrop Conc."
 [5] "Tissue batch"     "RNA batch"      "2-Cycle batch"    "Starting cRNA"
 [9] "Tissue"           "Time point"     "Replicate Number" "Myc ON/OFF"
```

5.1 Using the GUI

First, we will look at an example of how to run `envisage()` using the GUI:

```
> envisage(SkinvsPancreas)
```

The GUI should appear as in figure 2. It is worth noting that several of the variable names have been renamed from their original names:

1. **2-Cycle batch** -> X2_Cycle_batch

2. **Myc ON/OFF** -> Myc_ON_OFF

The variable `Sample name` has unique values for each sample as shown below, so is classed as a labelling variable. Any such variables should be removed from the modelling process to avoid overfitting of the data, or alternatively users must be sure that the variables are selected as numeric, and not categoric. Selecting any of these variables for use in the model as a categorical variable results in an error message to limit accidental assignment by the user.

```
> nlevels(pData(SkinvsPancreas)$"Sample name")

[1] 48

> length(sampleNames(SkinvsPancreas))

[1] 48
```

We make the following class selections for each variable:

Sample name `None`

Treatment `None`

RIN `numeric`

Nanodrop Conc. `None`

Tissue batch `categoric`

RNA batch `categoric`

2-Cycle batch `categoric`

Starting cRNA `numeric`

Tissue `categoric`

Time point `categoric`

Replicate Number `None`

Myc ON/OFF `categoric`

The default arguments are kept, which uses a Benjamini & Hochberg false discovery rate multiple testing correction with a P-value cutoff of 0.05. Results will be saved to a folder in the working directory unless otherwise stated. Pressing 'OK' brings up another window confirming the variable classes selected. Click 'Yes' to continue. The next window that appears allows users to specify which variables to calculate interaction terms for (figure 3). For now we leave all variables selected, meaning that all of these will be used to calculate interaction terms. Pressing 'Ok' brings up an error message that informs us that we have too many variable terms in our maximum model. We can choose to continue, and risk seeing errors in the analysis due to aliasing, or alternatively we can choose to exclude some variables from the interaction term calculations. The 3 batching variables each have a large number of levels which will have a large effect on the number of residual degrees of freedom. We can safely ignore the interaction terms for these variables, since randomisation of batches should remove significant effects. So press 'Cancel', deselect the batch variables and press 'Ok'. This will start the modelling process.

Once the program has finished running, the output window shown in figure 4 will appear. By default, all terms are selected which will output results for all variables and interactions to the file selected (the R working directory by default). Alternatively, users can select which terms are of interest to them and output only these gene lists.

5.2 Using the command line

The same output can be seen by running the `envisage()` method through the command line:

```
> useParams <- c("RIN", "Tissue batch", "RNA batch", "2-Cycle batch",
+     "Starting cRNA", "Tissue", "Time point", "Myc ON/OFF")
> paramType <- c("numeric", rep("categoric", 3), "numeric", rep("categoric",
+     3))
> param4INT <- c("RIN", "Starting cRNA", "Tissue", "Time point",
+     "Myc ON/OFF")
> results1 <- envisage(SkinvsPancreas, widget = FALSE, MTC = "BH",
+     pCutoff = 0.05, useParams, param4INT, paramType, fileResults = getwd(),
+     MEorINT = "INT")
```

5.3 Envisage output

The results folder `EnvisageResults` contains the resulting gene lists for the analysis, as well as the 2 alias outputs `AliasGenes.txt` and `AliasReport.txt`. These files contain information on genes that show aliasing in the modeling procedure. As we can see, the majority of the 50 genes (34 genes) have shown some problem due to aliasing of the variables used in the analysis, and this may mean that the results of the modeling procedure are affected. By looking at the alias report, we can see that aliasing is seen between the same two variables, `Tissue` and `Tissue batch`, for all genes:

```
> names(results1$errorGenes)[1]

[1] "1416780_at"

> results1$errorGenes[[1]]

[1] "TissueSkin & (Intercept)"   "TissueSkin & Tissue_batch2"
[3] "TissueSkin & Tissue_batch3"
```

Batching for tissue extraction was not randomised, and skin samples were all identified with the tissue batch identifier 1. Thus the variable `Tissue` is in fact a subset of the variable `Tissue_batch` which explains the intrinsic aliasing seen. We can therefore repeat the process with the `Tissue batch` variable removed. After repeating analysis with the `Tissue batch` variable removed, you will find that the result output file contains only

significantly changing lists of genes for several variables in the analysis with no problems due to aliasing.

The resulting gene lists can be used for further analysis.

References

[1] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society*, Ser. B 57:289–300, 1995.

[2] Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *Ann. Stat.*, 29(4):1165âĂŞ1188, 2001.

[3] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, T. Gaasterland, P. Glenisson, F. C. Holstege, I. F. Kim, V. Markowitz, J. C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron. Minimum information about a microarray experiment (miame)-toward standards for microarray data. *Nat Genet*, 29(4):365–71, 2001. 1061-4036 (Print) Journal Article Research Support, Non-U.S. Gov't Research Support, U.S. Gov't, P.H.S.

[4] R. C. Gentleman, V. J. Carey, D. M. Bates, B. Bolstad, M. Dettling, S. Dudoit, B. Ellis, L. Gautier, Y. Ge, J. Gentry, K. Hornik, T. Hothorn, W. Huber, S. Iacus, R. Irizarry, F. Leisch, C. Li, M. Maechler, A. J. Rossini, G. Sawitzki, C. Smith, G. Smyth, L. Tierney, J. Y. Yang, and J. Zhang. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, 5(10):R80, 2004. 1465-6914 Journal Article.

[5] Y. Hochberg. A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75:800–802, 1988.

[6] S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6:65âĂŞ70, 1979.

[7] M. L. Lee, F. C. Kuo, G. A. Whitmore, and J. Sklar. Importance of replication in microarray gene expression studies: statistical methods and evidence from repetitive cdna hybridizations. *Proc Natl Acad Sci U S A*, 97(18):9834–9, 2000. 0027-8424 (Print) Journal Article Research Support, Non-U.S. Gov't Research Support, U.S. Gov't, P.H.S.