Check for updates

RESEARCH ARTICLE

An end to end workflow for differential gene expression using Affymetrix microarrays [version 1; peer review: 1 approved, 1 approved with reservations]

Bernd Klaus

EMBL Heidelberg, Heidelberg, 69117, Germany

V1 First published: 15 Jun 2016, 5:1384
<https://doi.org/10.12688/f1000research.8967.1>
 Latest published: 03 Jul 2018, 5:1384
<https://doi.org/10.12688/f1000research.8967.2>

Abstract

In this article, we walk through an end-to-end Affymetrix microarray differential expression workflow using Bioconductor packages. This workflow is directly applicable to current “Gene” type arrays, e.g. the HuGene or MoGene arrays but can easily adapted to similar platforms. The data re-analyzed is a typical clinical microarray data set that compares inflamed and non-inflamed colon tissue in two disease subtypes. We will start from the raw data CEL files, show how to import them into a Bioconductor ExpressionSet, perform quality control and normalization and finally differential gene expression (DE) analysis, followed by some enrichment analysis. As experimental designs can be complex, a self contained introduction to linear models is also part of the workflow.

Keywords

gene expression , Affimatrix , microarrays , workflow

This article is included in the **Bioconductor** gateway.

Open Peer Review

Approval Status

	1	2
version 2 (revision) 03 Jul 2018	 view	 view
version 1 15 Jun 2016	 view	 view

1. **James W. MacDonald** , University of Washington, Seattle, WA, USA
2. **Andrea Rau** , French National Institute for Agricultural Research (INRA), Jouy-en-Josas, France

Any reports and responses or comments on the article can be found at the end of the article.

Corresponding author: Bernd Klaus (bernd.klaus@embl.de)

Competing interests: The author declares that there are no competing interests.

Grant information: The author(s) declared that no grants were involved in supporting this work.

Copyright: © 2016 Klaus B. This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Data associated with the article are available under the terms of the [Creative Commons Zero "No rights reserved" data waiver](#) (CC0 1.0 Public domain dedication).

How to cite this article: Klaus B. **An end to end workflow for differential gene expression using Affymetrix microarrays [version 1; peer review: 1 approved, 1 approved with reservations]** F1000Research 2016, 5:1384 <https://doi.org/10.12688/f1000research.8967.1>

First published: 15 Jun 2016, 5:1384 <https://doi.org/10.12688/f1000research.8967.1>

Introduction

In this article we introduce a complete workflow for a typical (Affymetrix) microarray analysis. Data import, preprocessing, differential expression and enrichment analysis are discussed. We also introduce some necessary mathematical background on linear models along the way.

The data set used¹ is from a paper studying the differences between patients suffering from Ulcerative colitis (UC) or Crohn's disease (CD). This is a typical clinical data set consisting of 14 UC and 15 CD patients from which inflamed and non-inflamed colonic mucosa tissue was obtained via a biopsy. Our aim is to analyze differential expression (DE) between the tissues in the two diseases.

Required packages and other preparations

```
library(Biobase)
library(oligoClasses)
library(knitr)
library(BiocStyle)
library(oligo)
library(genefilter)
library(ggplot2)
library(dplyr)
library(LSD)
library(gplots)
library(RColorBrewer)
library(ArrayExpress)
library(arrayQualityMetrics)
library(stringr)
library(matrixStats)
library(topGO)
library(genefilter)
library(pd.hugene.1.0.st.v1)
library(hugene10sttranscriptcluster.db)
library(pheatmap)
library(mvtnorm)
library(DAAG)
library(multcomp)
library(limma)
library(ReactomePA)
library(clusterProfiler)
library(openxlsx)
library(devtools)
library(biomaRt)
library(EnrichmentBrowser)
set.seed(777)
raw_data_dir <- file.path(getwd(), "rawDataMAWorkflow")
```

Download the raw data from from ArrayExpress

The first step of the analysis is to download the raw data CEL files. These files are produced by the array scanner software and contain the probe intensities measured. The data have been deposited at [ArrayExpress](#) and have the accession code **E-MTAB-2967**.

Each ArrayExpress data set has a landing page summarizing the data set, and we use the [ArrayExpress](#) Bioconductor package to obtain the ftp links to the raw data files ([Data from Palmieri *et al.* on ArrayExpress](#)).

Information stored in ArrayExpress

Each dataset at ArrayExpress is stored according to the MAGE-TAB (MicroArray Gene Expression Tabular) specifications as a collection of tables bundled with the raw data. The MAGE-TAB format specifies up to five different types of files, namely the Investigation Description Format (IDF), the Array Design Format (ADF), the Sample and Data Relationship Format (SDRF), the raw data files and the processed data files.

For use, the IDF and the SDRF file are important. The IDF file contains top level information about the experiment including title, description, submitter contact details and protocols. The SDRF file contains essential information on the experimental samples, e.g. the experimental group(s) they belong to.

Download the raw data and the annotation data

With the code below, we download the raw data from [ArrayExpress](#)². It is saved in the directory `raw_data_dir` which defaults to the subdirectory `rawDataMAWorkflow` of the current working directory. The names of the downloaded files are returned as a list.

```
if(!dir.exists(raw_data_dir)){
  dir.create(raw_data_dir)
}

anno_AE <- getAE("E-MTAB-2967", path=raw_data_dir, type="raw")
```

We now import the SDRF file directly from ArrayExpress in order to obtain the sample annotation.

The raw data consists of one CEL file per sample (see below) and we use the CEL file names as row names for the imported data. These names are given in a column named `Array.Data.File` in the SDRF table. We turn the SDRF table into an `AnnotatedDataFrame` from the *Biobase* package that we will need later to create an `ExpressionSet` for our data³.

```
SDRF <- read.delim(
url("http://www.ebi.ac.uk/arrayexpress/files/E-MTAB-2967/E-MTAB-2967.sdrf.txt"))

rownames(SDRF) <- SDRF$Array.Data.File
SDRF <- AnnotatedDataFrame(SDRF)
```

Before we move on to the actual raw data import, we will briefly introduce the `ExpressionSet` class contained in the *Biobase* package. It is commonly used to store microarray data in Bioconductor.

Figure 1. Structure of Bioconductor's `ExpressionSet` class.

Bioconductor ExpressionSets

Genomic data can be very complex, usually consisting of a number of different bits and pieces, e.g. information on the experimental samples, annotation of genomic features measured as well as the experimental data itself. In Bioconductor the approach is taken that these pieces should be stored in a single structure to easily manage the data.

The package *Biobase* contains standardized data structures to represent genomic data. The `ExpressionSet` class is designed to combine several different sources of information (i.e. as contained in the various MAGE-TAB files) into a single convenient structure. An `ExpressionSet` can be manipulated (e.g., subsetted, copied), and is the input to or output of many Bioconductor functions.

The data in an `ExpressionSet` consist of

- **assayData:** Expression data from microarray experiments.
- **metaData:** A description of the samples in the experiment (`phenoData`), metadata about the features on the chip or technology used for the experiment (`featureData`), and further annotations for the features, for example gene annotations from biomedical databases (`annotation`).
- **experimentData:** A flexible structure to describe the experiment.

The `ExpressionSet` class coordinates all of these data, so that one does not have to worry about the details. However, some constraints have to be met. In particular, the rownames of the `phenoData` (which holds the content of the SDRF file) have to match the column names of the assay data (as they represent the sample identifiers), while the row names of the expression data have to match the row names of the `featureData` (as they represent the feature identifiers). This is illustrated in the figure.

You can use the functions `pData` and `fData` to extract the sample and feature annotation respectively from an `ExpressionSet`. The function `exprs` will return the expression data itself as a matrix.

Import of the raw microarray data

The analysis of Affymetrix arrays starts with CEL files. These are the result of the processing of the raw image files using the Affymetrix software and contain estimated probe intensity values. Each CEL file additionally contains some metadata, such as a chip identifier.

The function `read.celfiles` from the *oligo*⁴ can be used to import the files. The package automatically uses *pd.hugene.1.0.st.v1* as the chip annotation package as the chip-type is also stored in the .CEL files.

We specify our `AnnotatedDataFrame` created earlier as `phenoData`. Thus, We have to be sure that we import the CEL files in the order that corresponds to the SDRF table — to enforce this, we use the column `Array.Data.File` of the SDRF table as the `filenames` argument.

Finally, we check whether the object created is valid. (e.g. sample names match between the different tables).

We collect the information about the CEL files and import and them into the variable `raw_data`:

```
raw_data <- read.celfiles(filenames = file.path(raw_data_dir,
                                              SDRF$Array.Data.File),
                        verbose = FALSE, phenoData = SDRF)
validObject(raw_data)
```

We now inspect the raw data a bit and retain only those columns that are related to the experimental factors of interest (identifiers of the individuals, disease of the individual and the mucosa type).

```
head(pData(raw_data))
```

	Source.Name	Characteristics.individual.	Characteristics.organism.
164_I.CEL	164_I	164	Homo sapiens
164_II.CEL	164_II	164	Homo sapiens
183_I.CEL	183_I	183	Homo sapiens

183_II.CEL	183_II	183	Homo sapiens
2114_I.CEL	2114_I	2114	Homo sapiens
2114_II.CEL	2114_II	2114	Homo sapiens

Characteristics.disease. Characteristics.organism.part.

164_I_.CEL	Crohn's disease	colon
164_II.CEL	Crohn's disease	colon
183_I.CEL	Crohn's disease	colon
183_II.CEL	Crohn's disease	colon
2114_I.CEL	Crohn's disease	colon
2114_II.CEL	Crohn's disease	colon

Characteristics.phenotype. Material.Type Protocol.REF

164_I_.CEL	non-inflamed colonic mucosa	organism part	P-MTAB-41361
164_II.CEL	inflamed colonic mucosa	organism part	P-MTAB-41361
183_I.CEL	non-inflamed colonic mucosa	organism part	P-MTAB-41361
183_II.CEL	inflamed colonic mucosa	organism part	P-MTAB-41361
2114_I.CEL	non-inflamed colonic mucosa	organism part	P-MTAB-41361
2114_II.CEL	inflamed colonic mucosa	organism part	P-MTAB-41361

Protocol.REF.1 Extract.Name Protocol.REF.2 Labeled.Extract.Name

164_I_.CEL	P-MTAB-41363	164_I	P-MTAB-41364	164_I:Biotin
164_II.CEL	P-MTAB-41363	164_II	P-MTAB-41364	164_II:Biotin
183_I.CEL	P-MTAB-41363	183_I	P-MTAB-41364	183_I:Biotin
183_II.CEL	P-MTAB-41363	183_II	P-MTAB-41364	183_II:Biotin
2114_I.CEL	P-MTAB-41363	2114_I	P-MTAB-41364	2114_I:Biotin
2114_II.CEL	P-MTAB-41363	2114_II	P-MTAB-41364	2114_II:Biotin

Label Protocol.REF.3 Assay.Name Technology.Type Array.Design.REF

164_I_.CEL	biotin	P-MTAB-41366	164_I_	array assay	A-AFFY-141
164_II.CEL	biotin	P-MTAB-41366	164_II	array assay	A-AFFY-141
183_I.CEL	biotin	P-MTAB-41366	183_I	array assay	A-AFFY-141
183_II.CEL	biotin	P-MTAB-41366	183_II	array assay	A-AFFY-141
2114_I.CEL	biotin	P-MTAB-41366	2114_I	array assay	A-AFFY-141
2114_II.CEL	biotin	P-MTAB-41366	2114_II	array assay	A-AFFY-141

Term.Source.REF Protocol.REF.4 Array.Data.File

164_I_.CEL	ArrayExpress	P-MTAB-41367	164_I_.CEL
164_II.CEL	ArrayExpress	P-MTAB-41367	164_II.CEL
183_I.CEL	ArrayExpress	P-MTAB-41367	183_I.CEL
183_II.CEL	ArrayExpress	P-MTAB-41367	183_II.CEL
2114_I.CEL	ArrayExpress	P-MTAB-41367	2114_I.CEL
2114_II.CEL	ArrayExpress	P-MTAB-41367	2114_II.CEL

Comment..ArrayExpress.

164_I_.CEL ftp://ftp.ebi.ac.uk/pub/databases/microarray/data/experiment/MTAB/E-MTAB-2967/E-MTAB-2967.

164_II.CEL ftp://ftp.ebi.ac.uk/pub/databases/microarray/data/experiment/MTAB/E-MTAB-2967/E-MTAB-2967.

183_I.CEL ftp://ftp.ebi.ac.uk/pub/databases/microarray/data/experiment/MTAB/E-MTAB-2967/E-MTAB-2967.

183_II.CEL ftp://ftp.ebi.ac.uk/pub/databases/microarray/data/experiment/MTAB/E-MTAB-2967/E-MTAB-2967.

2114_I.CEL ftp://ftp.ebi.ac.uk/pub/databases/microarray/data/experiment/MTAB/E-MTAB-2967/E-MTAB-2967.

2114_II.CEL ftp://ftp.ebi.ac.uk/pub/databases/microarray/data/experiment/MTAB/E-MTAB-2967/E-MTAB-2967.

Factor.Value.disease. Factor.Value.phenotype.

164_I_.CEL	Crohn's disease	non-inflamed colonic mucosa
164_II.CEL	Crohn's disease	inflamed colonic mucosa
183_I.CEL	Crohn's disease	non-inflamed colonic mucosa
183_II.CEL	Crohn's disease	inflamed colonic mucosa
2114_I.CEL	Crohn's disease	non-inflamed colonic mucosa
2114_II.CEL	Crohn's disease	inflamed colonic mucosa

```
head(exprs(raw_data))
```

	164_I_.CEL	164_II.CEL	183_I.CEL	183_II.CEL	2114_I.CEL	2114_II.CEL	2209_A.CEL
1	4496	5310	4492	4511	2872	5081	3357
2	181	280	137	101	91	114	175
3	4556	5104	4379	4608	2972	4763	3519
4	167	217	99	79	82	84	157
5	89	110	69	58	47	46	97
6	80	148	96	66	60	54	114
	2209_B.CEL	2255_I.CEL	2255_II.CEL	2400_I.CEL	2400_II.CEL	2424_A.CEL	
1	4579	5878	3197	3111	3018	2816	
2	209	132	87	109	131	96	
3	4751	5913	3517	3244	3078	2637	
4	206	142	79	65	129	97	
5	106	59	54	57	104	47	
6	102	106	58	73	66	61	
	2424_B.CEL	255_I.CEL	255_II.CEL	2826_I.CEL	2826_II.CEL	2853_I.CEL	2853_II.CEL
1	3996	2097	5305	2028	7146	4307	5014
2	159	91	142	69	163	245	178
3	4268	2923	5513	2259	6811	4571	4782
4	96	94	87	61	88	182	168
5	50	61	58	39	51	92	81
6	77	64	72	42	105	166	139
	2978_I.CEL	2978_II.CEL	2987_I.CEL	2987_II.CEL	2992_I.CEL	2992_II.CEL	
1	5525	6085	2856	3056	4685	3403	
2	125	158	113	128	172	150	
3	5576	6291	3469	3430	5218	3430	
4	121	115	71	91	188	152	
5	51	54	62	80	81	102	
6	90	100	87	89	118	91	
	2995_I.CEL	2995_II.CEL	321_I.CEL	321_II.CEL	3222_I.CEL	3222_II.CEL	3223_I.CEL
1	5458	2698	4607	3338	4396	3876	4300
2	191	97	126	106	143	142	179
3	5043	3075	5034	3496	4833	4104	4845
4	161	101	106	81	126	120	143
5	59	86	45	37	61	76	154
6	110	72	79	67	98	78	138
	3223_II.CEL	3226_I.CEL	3226_II.CEL	3233_I.CEL	3233_II.CEL	3258_I.CEL	
1	4669	3281	2263	5800	4418	6408	
2	197	136	84	182	133	263	
3	5256	3610	2461	5440	4835	5806	
4	227	137	110	179	125	236	
5	67	65	70	64	62	104	
6	126	88	53	91	69	157	
	3258_II.CEL	3259_I.CEL	3259_II.CEL	3262_I.CEL	3262_II.CEL	3266_I.CEL	
1	4613	4743	4137	4745	2293	4853	
2	165	197	191	109	99	130	
3	4754	4941	4166	5158	2267	4912	
4	192	120	115	117	75	85	
5	103	54	111	43	37	52	
6	133	108	125	74	63	73	
	3266_II.CEL	3269_I.CEL	3269_II.CEL	3271_I.CEL	3271_II.CEL	3302_I.CEL	
1	4192	4069	3295	5790	2361	4606	
2	143	130	101	143	97	137	
3	4099	4239	3686	5322	3011	5195	
4	103	130	88	109	85	102	
5	49	65	59	61	52	65	
6	59	80	89	80	55	98	

	3302_II.CEL	3332_I.CEL	3332_II.CEL	848_A.CEL	848_B.CEL	888_I.CEL	888_II.CEL
1	3007	3839	2667	5211	4775	5886	4383
2	84	102	75	216	204	206	140
3	3389	3878	3224	4949	5110	6004	4026
4	82	61	65	178	183	193	165
5	42	48	59	76	91	56	75
6	63	81	80	98	140	104	95

```
stopifnot(validObject(raw_data))
```

```
pData(raw_data) <- pData(raw_data)[, c("Source.Name",
    "Characteristics.individual.",
    "Factor.Value.disease.",
    "Factor.Value.phenotype.")]
```

Quality control of the raw data

The first step after the initial data import is the quality control of the data. Here we check for outliers and try to see whether the data clusters as expected, e.g. by the experimental conditions. We use the identifiers of the individuals as plotting symbols.

```
exp_raw <- log2(exprs(raw_data))
PCA_raw <- prcomp(t(exp_raw), scale = FALSE)

dataGG <- data.frame(PC1 = PCA_raw$x[,1], PC2 = PCA_raw$x[,2],
    Disease = pData(raw_data)$Factor.Value.disease.,
    Phenotype = pData(raw_data)$Factor.Value.phenotype.,
    Individual = pData(raw_data)$Characteristics.individual.)

(qplot(PC1, PC2, data = dataGG, color = Disease,
    main = "PCA plot of the raw data (log-transformed)", size = I(2),
    asp = 1.0, geom = "text",
    label = Individual)
+ scale_colour_brewer(palette = "Set2"))

boxplot(raw_data, target = "core",
    main = "Boxplots of log2-intensities for the raw data")
```

The PCA (performed on the log-intensity scale) plot of the raw data shows that the first principal component differentiates between the diseases. However, the intensity boxplots show that the intensity distributions of the individual arrays are quite different, indicating the need of an appropriate normalization, which we will discuss next.

A wide range of quality control plots can be created using the package *arrayQualityMetrics*⁵. The package produces an html report, containing the quality control plots together with a description of their aims and an identification of possible outliers. We don't discuss this tool in detail here, but the code below can be used to create a report for our raw data.

Figure 2. PCA plot of the log-transformed raw data.

Figure 3. Intensity boxplots of the log2-transformed raw data.

```
arrayQualityMetrics(expressionset = raw_data,
  outdir = "Report_for_Palmieri_raw",
  force = TRUE, do.logtransform = TRUE,
  intgroup = c("Factor.Value.disease.", "Factor.Value.phenotype."))
```

Background adjustment, calibration, summarization and annotation

Background adjustment

After the initial import and quality assessment, the next step in processing of microarray data is background adjustment. This is essential because a part of the measured probe intensities are due to non-specific hybridization and the noise in the optical detection system. Therefore, observed intensities need to be adjusted to give accurate measurements of specific hybridization.

Across-array normalization (calibration)

Without proper normalization across arrays, it is impossible to compare measurements from different array hybridizations due to many obscuring sources of variation. These include different efficiencies of reverse transcription, labeling or hybridization reactions, physical problems with the arrays, reagent batch effects, and laboratory conditions.

Summarization

After normalization, summarization is needed because on the Affymetrix platform transcripts are represented by multiple probes. For each gene, the background adjusted and normalized intensities need to be summarized into one quantity that estimates an amount proportional to the amount of RNA transcript.

After the summarization step, the summarized data can be annotated with various information, e.g. gene symbols and EMSEMBL gene identifiers. There is an annotation database available from Bioconductor for our platform, namely the package *hugene10sttranscriptcluster.db*.

You can view its content like this

```
head(ls("package:hugene10sttranscriptcluster.db"))

[1] "hugene10sttranscriptcluster"
[2] "hugene10sttranscriptclusterACCNUM"
[3] "hugene10sttranscriptclusterALIAS2PROBE"
[4] "hugene10sttranscriptclusterCHR"
[5] "hugene10sttranscriptclusterCHRLNGTHS"
[6] "hugene10sttranscriptclusterCHRLOC"
```

Additional information is available from the reference manual of the package. Essentially, the package provides a mapping from the transcript cluster identifiers to the various annotation data.

Old and new “probesets” of Affymetrix microarrays

Traditionally, Affymetrix arrays (the so-called 3’ IVT arrays) were probeset based: a certain fixed group of probes were part of a probeset which represented a certain gene or transcript (note however, that a gene can be represented by multiple probesets).

The more recent “Gene” and “Exon” Affymetrix arrays are exon based and hence there are two levels of summarization. The exon level summarization leads to “probeset” summary. However, these probesets are not the same as the probesets of the previous chips, which usually represented a gene/transcript. Furthermore, there are also no longer designated match/mismatch probes present on “Gene” type chips.

For the newer Affymetrix chips a gene/transcript level summary is given by “transcript clusters”. Hence the appropriate annotation package is called *hugene10sttranscriptcluster.db*.

To complicate things even a bit more, note that the “Gene” arrays were created as affordable versions of the “Exon” arrays by taking the “good” probes from the Exon array. So the notion of a probeset is based on the original construction of the probesets on the Exon array, which contains usually at least four probes.

But since Affymetrix selected only a the subset of “good” probes for the Gene arrays, a lot of the probesets on the “Gene” arrays are made up of three or fewer probes. Thus, a summarization on the probeset/exon level is not recommended for “Gene” arrays but nonetheless possible by using the *hugene10stprobeset.db* annotation package.

One-go preprocessing in oligo

The package *oligo* allows us to perform background correction, normalization and summarization in one single step using a deconvolution method for background correction, quantile normalization and the RMA (robust multichip average) algorithm for summarization.

This series of steps as a whole is commonly referred to as RMA algorithm, although strictly speaking RMA is merely a summarization method⁶⁻⁸.

```
palmieri_eset <- oligo::rma(raw_data, target="core")

Background correcting
Normalizing
Calculating Expression
```

The parameter `target` defines the degree of summarization, the default option of which is “core”, using transcript clusters containing “safely” annotated genes. Other options for `target` include “extended” and “full”. For summarization on the exon level (not recommended for Gene arrays), one can use “probeset” as the target option.

Although other methods for background correction and normalization exist, RMA is usually a good default choice. RMA shares information across arrays and uses the versatile quantile normalization method that will make the array intensity distributions match. However, it is preferable to apply it only after outliers have been removed. The quantile normalization algorithm used by RMA works by replacing values by the average of identically ranked (with a single chip) values across arrays. A more detailed description can be found on the [Wikipedia page](#) about it.

An alternative to quantile normalization is the *vsn* algorithm, that performs background correction and normalization by robustly shifting and scaling log-scale intensity values within arrays⁹. This is less “severe” than quantile normalization.

Some mathematical background on normalization (calibration) and background correction

A generic model for the value of the intensity Y of a single probe on a microarray is given by

$$Y = B + \alpha \cdot S$$

where B is a random quantity due to background noise, usually composed of optical effects and non-specific binding, α is a gain factor, and S is the amount of measured specific binding. The signal S is considered a random variable as well and accounts for measurement error and probe effects. The measurement error is typically assumed to be multiplicative so we can write:

$$\log(S) = \theta + \varphi + \varepsilon$$

Here θ represents the logarithm of the true abundance, φ is a probe-specific effect, and ε accounts for the nonspecific error. This is the additive-multiplicative-error model for microarray data used by RMA and also the *vsn* algorithm⁹. The algorithms differ in the way B is removed and an estimate of θ is obtained.

Figure 4. PCA plot of the calibrated data.

Quality assessment of the calibrated data

We now produce a clustering and another PCA plot using the calibrated data. In order to display a heatmap of the sample-to-sample distances, we first compute the distances using the `dist` function. We need to transpose the expression values since the function computes the distances between the rows (i.e. genes in our case) by default. The default distance is the Euclidean one. However this can be changed and we choose the manhattan distance here (it uses

Figure 5. Heatmap of the sample to sample distances for the calibrated data.

absolute instead of squared distances). We set the diagonal of the distance matrix to NA in order to increase the contrast of the color coding. Those diagonal entries do not contain information since the distance of a sample to itself is always equal to zero.

```
exp_palmieri <- exprs(palmieri_eset)
PCA <- prcomp(t(exp_palmieri), scale = FALSE)

dataGG <- data.frame(PC1 = PCA$x[,1], PC2 = PCA$x[,2],
                     Disease = pData(palmieri_eset)$Factor.Value.disease.,
                     Phenotype = pData(palmieri_eset)$Factor.Value.phenotype.)

(qplot(PC1, PC2, data = dataGG, color = Disease, shape = Phenotype,
       main = "PCA plot of the calibrated data", size = I(2), asp = 1.0)
+ scale_colour_brewer(palette = "Set2"))

dists <- as.matrix(dist(t(exp_palmieri), method = "manhattan"))
colnames(dists) <- NULL
diag(dists) <- NA
rownames(dists) <- pData(palmieri_eset)$Factor.Value.phenotype.
hmcol <- colorRampPalette(rev(brewer.pal(9, "PuOr")))(255)

pheatmap(dists, col = rev(hmcol), clustering_distance_rows = "manhattan",
          clustering_distance_cols = "manhattan")
```

The second PC roughly separates Crohn's disease from ulcerative colitis, while the first separates the tissues. This is what we expect: the samples cluster by their experimental conditions. On the heatmap plot we also see that the samples do not cluster strongly by tissue, confirming the impression from the PCA plot that the separation between the tissues is not perfect. The stripe in the heatmap might correspond to outlier that could potentially remove. The *arrayQualityMetrics* package produces reports that compute several metrics that can be used for outlier removal.

Filtering based on intensity

We now filter out lowly expressed genes. Microarray data commonly show a large number of probes in the background intensity range. They also do not change much across arrays. Hence they combine a low variance with a low intensity. Thus, they could end up being detected as differentially expressed although they are barely above the "detection" limit and are not very informative in general. We will perform a "soft" intensity based filtering here, since this is recommended by *limma*'s^{10,11} user guide (a package we will use below for the differential expression analysis). However, note that a variance based filter might exclude a similar set of probes in practice. In the histogram of the gene-wise medians, we can clearly see an enrichment of low medians on the left hand side. These represent the genes we want to filter.

In order to infer a cutoff from the data, we inspect the histogram of the median-intensities. We visually fit a central normal distribution given by $0.5 \cdot N(5.1, 1.18)$ to the probe-wise medians, which represents their typical behavior in the data set at hand.

Then we use the 5% quantile of this distribution as a threshold. We keep only those genes that show an expression higher than the threshold in at least as many arrays as in the smallest experimental group.

```
no_of_samples <- table(paste0(pData(palmieri_eset)$Factor.Value.disease., "_",
                               pData(palmieri_eset)$Factor.Value.phenotype.))
no_of_samples

      Crohn's disease_inflamed colonic mucosa
                                15
Crohn's disease_non-inflamed colonic mucosa
                                15
      ulcerative colitis_inflamed colonic mucosa
                                14
      ulcerative colitis_non-inflamed colonic mucosa
                                14
```

In our case this would be 14.

```
palmieri_medians <- rowMedians(exprs(palmieri_eset))

hist_res <- hist(palmieri_medians, 100, col="#e7efd8", freq = FALSE,
  main = "Histogram of the median intensities",
  xlab = "Median intensities")

emp_mu <- hist_res$breaks[which.max(hist_res$density)]
emp_sd <- mad(palmieri_medians)/2
prop_cental <- 0.50

lines(sort(palmieri_medians), prop_cental*dnorm(sort(palmieri_medians),
  mean = emp_mu, sd = emp_sd),
  col = "grey10", lwd = 4)

cut_val <- 0.05 / prop_cental
thresh_median <- qnorm(0.05 / prop_cental, emp_mu, emp_sd)

samples_cutoff <- min(no_of_samples)

idx_thresh_median <- apply(exprs(palmieri_eset), 1, function(x){
  sum(x > thresh_median) >= samples_cutoff})
table(idx_thresh_median)

      idx_thresh_median
FALSE  TRUE
 9371 23926
palmieri_filtered <- subset(palmieri_eset, idx_thresh_median)
```

Figure 6. Histogram of the median intensities per gene.

Annotation of the transcript clusters

Before we continue with the linear models for microarrays and differential expression we describe how to add “feature Data”, i.e. annotation information to the transcript cluster identifiers stored in the featureData of our ExpressionSet. We use the function `select` from *AnnotationDbi* to query the gene symbols and associated short descriptions for the transcript clusters. For each cluster, we add the gene symbol and a short description of the gene the cluster represents.

```
anno_palmieri <- AnnotationDbi::select(hugene10sttranscriptcluster.db,
                                     keys=(featureNames(palmieri_filtered)),
                                     columns = c("SYMBOL", "GENENAME"),
                                     keytype="PROBEID")
```

Removing multiple mappings and building custom annotations

Many transcript–cluster identifiers will map to multiple gene symbols. We compute a summary table in the code below to see how many there are.

```
probe_stats <- anno_palmieri %>%
  group_by(PROBEID) %>%
  summarize(no_of_matches = n_distinct(SYMBOL)) %>%
  filter(no_of_matches > 1)
```

```
probe_stats
```

```
Source: local data frame [2,142 × 2]
```

	PROBEID (chr)	no_of_matches (int)
1	7896742	8
2	7896754	2
3	7896937	3
4	7896952	2
5	7896961	3
6	7897006	2
7	7897632	3
8	7897745	2
9	7897953	2
10	7897991	2
..	...	...

```
dim(probe_stats)
```

```
[1] 2142    2
```

We have over 2000 transcript–clusters that map to multiple gene symbols. It is difficult to decide which mapping is “correct”. Therefore, we exclude these transcript–clusters. Additionally, we also exclude transcript–clusters that do not map to gene symbols.

```
ids_to_exclude <- ((featureNames(palmieri_filtered) %in% probe_stats$PROBEID) |
                  featureNames(palmieri_filtered) %in% subset(anno_palmieri ,
                                                              is.na(SYMBOL))$PROBEID)

table(ids_to_exclude)

ids_to_exclude
FALSE TRUE
16606 7320

palmieri_final <- subset(palmieri_filtered, !ids_to_exclude)
```

```
validObject(palmieri_final)

[1] TRUE

fData(palmieri_final)$PROBEID <- rownames(fData(palmieri_final))
fData(palmieri_final) <- left_join(fData(palmieri_final), anno_palmieri)

Joining by: "PROBEID"

# restore rownames after left_join
rownames(fData(palmieri_final)) <- fData(palmieri_final)$PROBEID

validObject(palmieri_final)

[1] TRUE
```

Alternatively, one can re-map the probes of the array to a current annotation, a workflow to do this for Illumina arrays is given in 12. Essentially, the individual probe sequences are re-aligned to an in-silico “exome” that consists of all annotated transcript exons.

In any case, the package *pdInfoBuilder* can be used to build custom annotation packages for use with *oligo*. In order to do this, PGF/CLF files (called “library files” on the Affymetrix website) as well as the probeset annotations are required. The probesets typically represent a small stretches of the genome (such as a single exon) and multiple probesets are then used to form a transcript cluster.

The CLF file contains information about the location of individual probes on the array. The PGF file then contains the individual probe sequences and shows the probeset they belong to. Finally, The probeset annotation .csv then contains information about which probesets are used in which transcript cluster. Commonly, multiple probesets are used in one transcript cluster and some probesets are contained in multiple transcript clusters.

A short overview of linear models

I am afraid this section is rather technical. However general experience shows that most questions on the Bioconductor support site about packages using linear models like *limma*¹⁰, *DESeq2*¹³ and *edgeR*¹⁴ are actually not so much about the packages themselves but rather about the underlying linear models. It might also be helpful to learn a bit of linear algebra to understand the concepts better. The Khan Academy offers nice (and free) online courses. Mike Love’s and Michael Irizarry’s *genomics class* is also a very good resource, especially its section on *interactions and contrasts*.

Regression models

In regression models we use one variable to explain or predict the other. It is customary to plot the predictor variable on the x-axis and the predicted variable on the y-axis. The predictor is also called the independent variable, the explanatory variable, the covariate, or simply x . The predicted variable is called the dependent variable, or simply y .

In a regression problem the data are pairs (x_i, y_i) for $i = 1, \dots, n$. For each i , y_i is a random variable whose distribution depends on x_i . We write

$$y_i = g(x_i) + \varepsilon_i.$$

The above expresses y_i as a systematic or explainable part $g(x_i)$ and an unexplained part ε_i . Or more informally: response = signal + noise. g is called the regression function. Once we have an estimate $\hat{g}$ of g , we can compute $r_i := y_i - g(x_i)$. The r_i ’s are called residuals. The ε_i ’s themselves are called errors.

Residuals are used to evaluate and assess the fit of models for g . Usually one makes distributional assumption about them, e.g. that they are independent and identically normally distributed with identical variance σ^2 and mean zero:

$$r_i \sim N(0, \sigma^2)$$

This allows to derive statistical tests for model coefficients.

Linear regression models

Linear regression is a special case of the general regression model. Here, we combine the predictors linearly to produce a prediction. If we have only single predictor x , the simple linear regression model is:

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i; \varepsilon_i \sim N(0, \sigma);$$

We can of course always add more predictors, let their total number be denoted by p . Then we get a multiple linear regression:

$$y_i = \beta_0 + \sum_{j=1}^p \beta_j x_{ij} + \varepsilon_i; j = 1, \dots, p$$

The equation for a multiple linear regression model can also be written in matrix form (we will denote matrices and vectors in bold font).

$$\mathbf{Y}_{n \times 1} = \mathbf{X}_{n \times (p+1)} \boldsymbol{\beta}_{(p+1) \times 1} + \boldsymbol{\varepsilon}_{n \times 1}$$

With $\mathbf{X}$ being the so called **design matrix**:

$$\mathbf{X} := (\mathbf{1}_n, X_1, \dots, X_p)$$

$\mathbf{1}_n$ is a column vector of ones called the **intercept** and $\mathbf{X}_p = (x_{p1}, \dots, x_{pn})^T$ is a column vector of measurements for covariate p . The regression coefficients are commonly estimated by the method of “ordinary least squares” (OLS):

$$\text{OLS: } \|Y - \beta_0 - \sum_{j=1}^p \beta_j X_j\|_E \rightarrow \min!$$

leading to the estimate of the coefficient vector

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

Creating design matrices in R

To get an idea of what design matrices look like, we consider several examples. It is important to know some fundamentals about design matrices in order to be able to correctly transfer a design of a particular study to an appropriate linear model.

We will use the base R functions:

- `formula`
- `model.matrix`

... in order to produce design matrices for a variety of linear models. R uses the formula interface to create design matrices automatically. In the first example, we have two groups of two samples each. Using `formula` and `model.matrix` we will create a model matrix with so called treatment contrast parameterization (the default setting in R). This means that an intercept is included in the model, i.e. $X_0 = \mathbf{1}_n$ and X_1 is equal to 1 if the samples belongs to group two and zero otherwise as we can see in the following code chunk.

```
two_groups <- factor(c(1, 1, 2, 2))
f <- formula(~ two_groups)
model.matrix(f)
```

```
(Intercept) two_groups2
1          1          0
2          1          0
3          1          1
4          1          1
```

```

attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$two_groups
[1] "contr.treatment"

```

This design is called treatment contrast parameterization for an obvious reason: the first column of the design matrix represents a “base level”, i.e. the mean β_0 for group one and the second column, corresponding to β_1 , represents the difference between the group means since all group two samples have means represented by $\beta_0 + \beta_1$. As β_0 is the mean of group 1, β_1 corresponds to the difference of the means of group two and group one and thus shows the effect of a “treatment”.

However, this design is not orthogonal, i.e. the columns of the design matrix are not independent. We can construct an equivalent orthogonal design as follows:

```

gr1 <- ifelse(two_groups == 1, 1, 0)
gr2 <- ifelse(two_groups == 2, 1, 0)
orth_model <- model.matrix(~ 0 + gr1 + gr2)

```

Here, we loose the nice direct interpretability of the coefficients. Now β_1 is simply the mean of the second group. We will discuss the extraction of interesting contrasts (i.e. linear combinations of coefficients) from a model like this below.

We explicitly excluded the intercept by specifying it as zero. Commonly it makes sense to include an intercept in the model, especially in more complex models. We can specify a more complex design pretty easily: if we have two independent factors, the base mean now corresponds to the first levels of the two factors.

```

x <- factor(c(1, 1, 1, 1, 2, 2, 2, 2))
y <- factor(c("a", "a", "b", "b", "a", "a", "b", "b"))
two_factors <- model.matrix(~ x + y)
two_factors

      (Intercept) x2 yb
1             1  0  0
2             1  0  0
3             1  0  1
4             1  0  1
5             1  1  0
6             1  1  0
7             1  1  1
8             1  1  1
attr(,"assign")
[1] 0 1 2
attr(,"contrasts")
attr(,"contrasts")$x
[1] "contr.treatment"

attr(,"contrasts")$y
[1] "contr.treatment"

```

The “drop one” strategy is the default method for creating regression coefficients from factors in R. If a factor has d levels, adding it to the model will give you $d - 1$ regression coefficients corresponding to $d - 1$ columns in a design matrix. Apart from excluding the intercept, you can also use the I function to treat a covariate as it is without using the formula syntax. The code below includes z^2 as a covariate.

```

z <- 1:4
model.matrix(~ z)

      (Intercept) z

```

```

1      1 1
2      1 2
3      1 3
4      1 4
attr(,"assign")
[1] 0 1

model.matrix(~ 0 + z)

      z
1 1
2 2
3 3
4 4
attr(,"assign")
[1] 1

model.matrix(~ z + I(z^2))

      (Intercept)  z  I(z^2)
1             1  1      1
2             1  2      4
3             1  3      9
4             1  4     16
attr(,"assign")
[1] 0 1 2

```

Singular model matrices

Whatever your model matrix looks like, you should make sure that it is non-singular. Singularity means that the measured variables are linearly dependent and leads to regression coefficients that are not uniquely defined. In linear algebra terms, we say that the matrix does not have full rank, which for design matrices means that the actual dimension of the space spanned by the column vectors is in fact lower than the apparent one. This leads to a redundancy in the model matrix, since some columns can be represent by linear combinations of other columns.

For design matrices, which contain factors, this happens if two conditions are confounded, e.g. in one experimental group there are only females and in the other group there are only males. Then the effect of sex and experimental group cannot be disentangled.

Let's look at an example. We set three factors, of which the third one is nested with the first two. We can check the singularity of the model matrix by computing its so called singular value decomposition and check it's minimal singular value. If this is zero, the matrix is singular. As we can see, this is indeed the case here.

```

x <- factor(c(1, 1, 1, 1, 2, 2, 2, 2))
y <- factor(c("a", "a", "b", "b", "a", "a", "b", "b"))
z <- factor(c("m", "m", "m", "m", "k", "k", "l", "l"))
sing_model <- model.matrix(~ x+y+z)
sing_model

      (Intercept) x2 yb z1 zm
1             1  0  0  0  1
2             1  0  0  0  1
3             1  0  1  0  1
4             1  0  1  0  1
5             1  1  0  0  0
6             1  1  0  0  0
7             1  1  1  1  0
8             1  1  1  1  0
attr(,"assign")
[1] 0 1 2 3 3
attr(,"contrasts")
attr(,"contrasts")$x

```

```

[1] "contr.treatment"
attr(,"contrasts")$y
[1] "contr.treatment"
attr(,"contrasts")$z
[1] "contr.treatment"

round(min(svd(sing_model)$d))

[1] 0

```

We have one column in the design matrix that can be represented by a linear combination of the other columns, thus the column space has actually a lower dimension than the apparent one. For example, we can represent column 5 (“zm”) by a linear combination of the first two columns “intercept” and “x2”:

```

comb_coefs <- coef(lm(sing_model[, 5] ~ 0 + sing_model[, -5]))
comb_coefs <- ifelse(is.na(comb_coefs), 0L, comb_coefs)
comb_coefs

      sing_model[, -5](Intercept)      sing_model[, -5]x2
                    1.00e+00                    -1.00e+00
      sing_model[, -5]yb      sing_model[, -5]z1
                    2.36e-16                    -2.36e-16

round(sum(sing_model[, -5] %*% comb_coefs - sing_model[, 5]))

[1] 0

```

I.e., in mathematical notation this means

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = 1 \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} - 1 \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Thus, the corresponding regression coefficients are not uniquely determined and the model does not make much sense. Therefore, the non-singularity of the model matrix should always be checked beforehand.

Using contrasts to test hypotheses in linear regression

In differential expression analysis, our most important covariates will be factors that differentiate between two or more experimental groups, e.g. the covariate $X_p = (x_{p1}, \dots, x_{pn})$ is either zero or one depending on which group the sample belongs to.

We will illustrate this concept using a small data set called `toycars` from the **DAAG** package. The data set `toycars` gives the distance traveled by one of three different toy cars on a smooth surface, starting from rest at the top of a 16 inch long ramp tilted at varying angles. We have the variables:

- `angle`: Angle
- `distance`: Traveled distance
- `car`: Car number (1, 2 or 3)

We transform `car` into a factor so that R performs the necessary parameterization of the contrasts automatically.

```

toycars$car <- as.factor(toycars$car)
qplot(car, distance, data = toycars, geom="boxplot") + geom_point()

```

Figure 7. Boxplots showing the distance traveled by each car in the toycars data.

By looking at the box plots of distance by car, we can clearly see differences between the three types of cars. We can now fit a linear model with distance as the dependent variable and car and angle as the predictors. As we can see from the linear model output, the treatment contrast parameterization was used, with car 1 being the base level.

```
lm_car <- lm(distance ~ angle + car, data = toycars)
summary(lm_car)
```

Call:

```
lm(formula = distance ~ angle + car, data = toycars)
```

Residuals:

	Min	1Q	Median	3Q	Max
Residuals:	-0.09811	-0.04240	-0.00669	0.01741	0.17251

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.09252	0.03467	2.67	0.014	*
angle	0.18854	0.00995	18.96	1.5e-15	***
car2	0.11111	0.03195	3.48	0.002	**
car3	-0.08222	0.03195	-2.57	0.017	*

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0678 on 23 degrees of freedom

Multiple R-squared: 0.945, Adjusted R-squared: 0.938

F-statistic: 132 on 3 and 23 DF, p-value: 1.22e-14

Testing general linear hypotheses

The estimated coefficients now give us the difference between the distance traveled between car 1 and car 2 (0.11) and car 1 and car 3 (-0.08), and the associated t-tests of these coefficients. However we cannot see a test of car 2 vs. car 3. This contrast test would correspond to testing the difference between the car 2 and car 3 regression coefficients.

Thus, contrasts of interest to us may not readily correspond to coefficients in a fitted linear model. However, one can easily test general linear hypotheses of the coefficients of the form:

$$H_0 : C_{q \times (p+1)} \beta_{(p+1) \times 1} = \alpha_{q \times 1}$$

Where C is the contrast matrix containing the between group differences of interest, q is the total number of comparisons to be performed and α contains the difference to be tested, this is usually a vector of zeros. If one tests multiple coefficients at once (e.g. $\beta_1 = 0$ and $\beta_2 = 0$) the corresponding test statistic is F -distributed. If one just tests linear combinations of coefficients, e.g. $\beta_1 - \beta_2 = 0$, $\beta_1 - \beta_2 - 2\beta_3 = 0$ or something similar the test statistic has t -distribution. The function `summary` for `lm` reports the results of $\beta_1 = \beta_2 = \dots = \beta_p = 0$ (F -test) and the t -tests of $\beta_j = 0$ each of the coefficients.

Note that the model does not actually have to be refitted in order to test the contrasts. This makes contrast matrix based testing more efficient and convenient than reformulating the model using a new parameterization of the factors to obtain the desired tests. We can use the function `glht` from the *multcomp* package to test these general linear hypotheses¹⁵. Let us assume we want all pairwise comparisons between the cars, this can be achieved by defining a specific contrast matrix for our current model as given below.

There are three such comparisons and we can print the results by using the `summary` function. As we can see the estimates for the contrasts already contained in the original model agree with the obtained results from contrast fit.

```
C <- rbind(c(0, 0, 1, 0), c(0, 0, 0, 1), c(0, 0, 1, -1))
colnames(C) <- rownames(summary(lm_car)$coefficients)
rownames(C) <- c("car2 - car1", "car3 - car1", "car2 - car3")
C
```

	(Intercept)	angle	car2	car3
car2 - car1	0	0	1	0
car3 - car1	0	0	0	1
car2 - car3	0	0	1	-1

```
summary(glht(lm_car, C), test = adjusted(type = "none"))
```

Simultaneous Tests for General Linear Hypotheses

```
Fit: lm(formula = distance ~ angle + car, data = toycars)
```

Linear Hypotheses:

	Estimate	Std. Error	t value	Pr(> t)
car2 - car1 == 0	0.1111	0.0320	3.48	0.002 **
car3 - car1 == 0	-0.0822	0.0320	-2.57	0.017 *
car2 - car3 == 0	0.1933	0.0320	6.05	3.6e-06 ***

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- none method)
```

Note that the term “contrast” is used in the context of (re)parameterization of the original model (as in “treatment contrasts”) and in the testing of linear hypotheses about the model coefficients. This can lead to some confusion, however usually it should be clear from the context whether a reparameterization or test of linear hypotheses is intended.

We can also fit a linear model without an intercept to the toycars data set. Now, the coefficients derived from the “car” factor represent car-wise means. Thus, the contrasts we have to form change, however, the results for the group comparisons do not.

```
toycars$car <- as.factor(toycars$car)
lm_car_no_I <- lm(distance~0 + angle + car, data = toycars)
summary(lm_car_no_I)
```

```

Call:
lm(formula = distance ~ 0 + angle + car, data = toycars)

Residuals:
    Min       1Q   Median       3Q      Max
-0.09811 -0.04240 -0.00669  0.01741  0.17251

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
angle    0.18854      0.00995   18.96 1.5e-15 ***
car1     0.09252      0.03467    2.67  0.014 *
car2     0.20364      0.03467    5.87 5.5e-06 ***
car3     0.01030      0.03467    0.30  0.769
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0678 on 23 degrees of freedom
Multiple R-squared:  0.991, Adjusted R-squared:  0.989
F-statistic: 629 on 4 and 23 DF, p-value: <2e-16

C_no_I <- rbind(c(0, -1, 1, 0), c(0, -1, 0, 1), c(0, 0, 1, -1) )
colnames(C_no_I) <- rownames(summary(lm_car_no_I)$coefficients)
rownames(C_no_I) <- c("car2 - car1", "car3 - car1", "car2 - car3")
C_no_I

           angle car1 car2 car3
car2 - car1      0   -1    1    0
car3 - car1      0   -1    0    1
car2 - car3      0    0    1   -1

summary(glht(lm_car_no_I, C_no_I), test = adjusted(type = "none"))

Simultaneous Tests for General Linear Hypotheses

Fit: lm(formula = distance ~ 0 + angle + car, data = toycars)

Linear Hypotheses:

            Estimate Std. Error t value Pr(>|t|)
car2 - car1 == 0    0.1111      0.0320    3.48  0.002 **
car3 - car1 == 0   -0.0822      0.0320   -2.57  0.017 *
car2 - car3 == 0    0.1933      0.0320    6.05 3.6e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- none method)

```

Linear models for microarrays

We now apply linear models to microarrays. Specifically, we discuss how to use the *limma* for differential expression analysis. The package is designed to analyze complex experiments involving comparisons between many experimental groups simultaneously while remaining reasonably easy to use for simple experiments. The main idea is to fit a linear model to the expression data for each gene. Empirical Bayes and other shrinkage methods are used to borrow information across genes for the residual variance estimation leading to a “moderated” *t*-statistics, and stabilizing the analysis for experiments with just a small number of arrays¹¹.

In the following, we use appropriate design and contrast matrices for our linear models and fit a linear model to each gene separately.

A linear model for the data

The original paper is interested in changes in transcription that occur between inflamed and adjacent non-inflamed mucosal areas of the colon. This is studied in both inflammatory bowel disease types.

Since we have two arrays per individual, the first factor we need is a blocking factor for the individuals that will absorb differences between them. Then we create a factors that give us the grouping for the diseases and the tissue types. We furthermore simplify the names of the diseases to UC and DC, respectively. Then, we create two design matrices, one for each of the two diseases as we will analyze them separately in order to follow the analysis strategy of the original paper closely (one could also fit a joint model to the complete data set, however, the two diseases might behave very differently so that a joint fit might not be appropriate).

```
individual <- as.character(pData(palmieri_final)$Characteristics.individual.)

tissue <- str_replace_all(pData(palmieri_final)$Factor.Value.phenotype., " ", "_")
tissue <- ifelse(tissue == "non-inflamed_colonic_mucosa", "nI", "I")

disease <- str_replace_all(pData(palmieri_final)$Factor.Value.disease., " ", "_")
disease <- ifelse(disease == "Crohn's_disease", "CD", "UC")

i <- individual[disease == "CD"]
design_palmieri_CD <- model.matrix(~ 0 + tissue[disease == "CD"] + i)
colnames(design_palmieri_CD)[1:2] <- c("I", "nI")

i <- individual[disease == "UC"]
design_palmieri_UC <- model.matrix(~ 0 + tissue[disease == "UC"] + i)
colnames(design_palmieri_UC)[1:2] <- c("I", "nI")
```

We can inspect the design matrices and test their rank.

```
head(design_palmieri_CD[, 1:6])

  I nI i183 i2114 i2209 i2255
1 0  1    0     0     0     0
2 1  0    0     0     0     0
3 0  1    1     0     0     0
4 1  0    1     0     0     0
5 0  1    0     1     0     0
6 1  0    0     1     0     0

dim(design_palmieri_CD)

[1] 30 16

min(svd(design_palmieri_CD)$d)

[1] 0.344

head(design_palmieri_UC[, 1:6])

  I nI i2424 i2987 i2992 i2995
1 0  1    0     0     0     0
2 1  0    0     0     0     0
3 0  1    1     0     0     0
4 1  0    1     0     0     0
```



```

5 0 1 0 1 0 0
6 1 0 0 1 0 0

dim(design_palmieri_UC)

[1] 28 15

min(svd(design_palmieri_UC)$d)

[1] 0.355

```

Contrasts and hypotheses tests

We now fit the linear models and define appropriate contrasts to test hypotheses of interest. We want to compare the inflamed to the non-inflamed tissue. Thus, we create a contrast matrix consisting of one row. *limma*'s function `makeContrasts` creates this matrix from a symbolic description of the contrast of interest. We can fit the linear model, compute the moderated *t*-statistics by calling the `eBayes` function and finally extract the number of differentially expressed genes while controlling the FDR by requiring BH-corrected p-value below a certain threshold.

Figure 8. Histogram of the p-values for Crohn's disease.

Figure 9. Histogram of the p-values for ulcerative colitis.

```

contrast_matrix_CD <- makeContrasts(I-nI, levels = design_palmieri_CD)

palmieri_fit_CD <- eBayes(contrasts.fit(lmFit(palmieri_final[,disease == "CD"],
      design = design_palmieri_CD),
      contrast_matrix_CD))

contrast_matrix_UC <- makeContrasts(I-nI, levels = design_palmieri_UC)

palmieri_fit_UC <- eBayes(contrasts.fit(lmFit(palmieri_final[,disease == "UC"],
      design = design_palmieri_UC),
      contrast_matrix_UC))

```

Extracting results

Results can be extracted by use of the `topTable` function. We extract the comparisons for both Crohn's disease as well as ulcerative colitis and sort the results by their absolute t -statistics. As a diagnostic check, we also plot the p-value histogram: We expect a uniform distribution for the p-values that correspond to true null hypotheses, while the a peak near zero shows a enrichment for low p-values corresponding to differentially expressed (DE) genes. A p-value less than 0.001 was used in the original paper as a significance cutoff leading to 298 (CD) and 520 (UC) DE-genes for the two diseases.

We call around 500/1000 genes in the two conditions at the same cutoff, this higher number of DE genes identified is probably due to the increased power from the blocking according to the individuals and the moderated variance estimation that *limma* performs.

```

table_CD <- topTable(palmieri_fit_CD, number = Inf)
head(table_CD)

```

	PROBEID	SYMBOL	GENENAME	logFC
7928695	7928695	FAM213A	family with sequence similarity 213 member A	-0.599
8123695	8123695	ECI2	enoyl-CoA delta isomerase 2	-0.486
8164535	8164535	CRAT	carnitine O-acetyltransferase	-0.425
8009746	8009746	SLC16A5	solute carrier family 16 member 5	-0.518
7952249	7952249	USP2	ubiquitin specific peptidase 2	-0.848
8105348	8105348	GPX8	glutathione peroxidase 8 (putative)	0.831

```

      AveExpr      t  P.Value adj.P.Val      B
7928695    7.74 -7.14 1.15e-06    0.0161 5.51
8123695    6.88 -6.40 4.88e-06    0.0161 4.22
8164535    6.73 -6.32 5.64e-06    0.0161 4.09
8009746    5.56 -6.28 6.23e-06    0.0161 4.00
7952249    5.61 -6.24 6.68e-06    0.0161 3.94
8105348    5.30  6.11 8.70e-06    0.0161 3.70

```

```
table(table_CD$adj.P.Val < 0.05)
```

```

FALSE  TRUE
15152  1454

```

```
table(table_CD$P.Value < 0.001)
```

```

FALSE  TRUE
16037  569

```

```

hist(table_CD$P.Value, col = brewer.pal(3, name = "Set2")[1],
      main = "inflamed vs non-inflamed - Crohn's disease", xlab = "p-values")

```

```

table_UC <- topTable(palmieri_fit_UC, number = Inf)
head(table_UC)

```

	PROBEID	SYMBOL	GENENAME	logFC	AveExpr
8003875	8003875	SPNS2	spinstar homolog 2 (Drosophila)	0.741	6.48
8082012	8082012	SLC15A2	solute carrier family 15 member 2	0.806	5.32
7952290	7952290	TRIM29	tripartite motif containing 29	1.014	5.86
8096070	8096070	BMP3	bone morphogenetic protein 3	-1.696	6.42
7961693	7961693	LDHB	lactate dehydrogenase B	0.397	9.53
8072015	8072015	ADRBK2	adrenergic, beta, receptor kinase 2	0.471	5.58
	t	P.Value	adj.P.Val	B	
8003875	7.96	2.34e-07	0.00172	7.08	
8082012	7.91	2.56e-07	0.00172	7.00	
7952290	7.66	4.04e-07	0.00172	6.59	
8096070	-7.48	5.67e-07	0.00172	6.28	
7961693	7.45	6.04e-07	0.00172	6.22	
8072015	7.40	6.60e-07	0.00172	6.14	

```
table(table_UC$adj.P.Val < 0.05)
```

```
FALSE TRUE
14346 2260
```

```
table(table_UC$P.Value < 0.001)
```

```
FALSE TRUE
15633 973
```

```
hist(table_UC$P.Value, col = brewer.pal(3, name = "Set2")[2],
      main = "inflamed vs non-inflamed - Ulcerative colitis", xlab = "p-values")
```

Comparison to the paper results

We now compare our list of differentially expressed genes to the results obtained in the paper. The paper results can be downloaded as excel files from <http://links.lww.com/IBD/A795>. We save it in an .xlsx file named palmieri_DE_res.xlsx. The paper results are given as identified differentially expressed genes with a p-value less than 0.001, which corresponds to an FDR of 0.05 in Crohn's disease and 0.02 in ulcerative colitis. There are four tables in total giving the list of up and downregulated genes in CD and UC respectively. In the code below, we extract the gene symbols from the excel table and then compare them to the differentially expressed genes we identify at a p-value of 0.001.

```
palmieri_DE_res <- sapply(1:4, function(i) read.xlsx(cols = 1,
                                                    "palmieri_DE_res.xlsx",
                                                    sheet = i, startRow = 4))
names(palmieri_DE_res) <- c("CD_UP", "CD_DOWN", "UC_UP", "UC_DOWN")
palmieri_DE_res <- lapply(palmieri_DE_res, as.character)
paper_DE_genes_CD <- Reduce("c", palmieri_DE_res[1:2])
paper_DE_genes_UC <- Reduce("c", palmieri_DE_res[3:4])

overlap_CD <- length(intersect(subset(table_CD, P.Value < 0.001)$SYMBOL,
                                     paper_DE_genes_CD)) / length(paper_DE_genes_CD)

overlap_UC <- length(intersect(subset(table_UC, P.Value < 0.001)$SYMBOL,
                                     paper_DE_genes_UC)) / length(paper_DE_genes_UC)

overlap_CD

[1] 0.678

overlap_UC

[1] 0.692
```

Figure 10. Selecting a background set of genes for the gene ontology analysis.

We see that we get a moderate overlap of 0.678 for CD and 0.692 for UC. Note that it is recommended to always choose an FDR cutoff instead of a p-value cutoff, since this way you control an explicitly defined error rate and the results are easier to interpret and to compare. In what follows, we choose an FDR cutoff of 10%.

Gene ontology (GO) based enrichment analysis

We can now try to characterize the identified differentially expressed genes a bit better by performing a GO enrichment analysis. Essentially the gene ontology (<http://www.geneontology.org/>) is a hierarchically organized collection of functional gene sets^{16–18}.

Matching the background set of genes

The function `genefinder` from the *genefilter*¹⁹ will be used to find a background set of genes that are similar in expression to the differentially expressed genes. We then check whether the background has roughly the same distribution of average expression strength as the foreground.

We do this in order not to select a biased background since the gene set testing is performed by a simple Fisher test on a 2x2 table. Note that this approach is very similar to commonly used web tools like GOrilla²⁰. Here we focus on the CD subset of the data.

For every differentially expressed gene, we try to find genes with similar expression.

```
DE_genes_CD <- subset(table_CD, adj.P.Val < 0.1)$PROBEID

back_genes_idx <- genefinder(palmieri_final, as.character(DE_genes_CD),
                             method="manhattan", scale="none")

back_genes_idx <- sapply(back_genes_idx, function(x)x$indices)

back_genes <- featureNames(palmieri_final)[back_genes_idx]
back_genes <- setdiff(back_genes, DE_genes_CD)

intersect(back_genes, DE_genes_CD)

character(0)

length(back_genes)

[1] 8441
```

```

multidensity(list(
  all= table_CD[, "AveExpr"] ,
  fore= table_CD[DE_genes_CD , "AveExpr"],
  back= table_CD[rownames(table_CD) %in% back_genes, "AveExpr"]),
  col = c("#e46981", "#ae7ee2", "#a7ad4a"),
  xlab="mean expression",
  main = "DE genes for CD - background - matching")

```

We can see that the matching returned a sensible result and can now perform the actual testing. For this purpose we use the *topGO* which implements a nice interface to Fisher testing and also has additional algorithms taking the GO structure into account, by e.g. only reporting the most specific gene set in the hierarchy²¹.

The GO has three top ontologies, cellular component (CC), biological processes (BP), and molecular function (MF). For illustrative purposes we limit ourselves to the BP category here.

Running topGO

We first create a factor `all_genes` which indicates for every gene in our background/universe, whether it is differentially expressed or not.

```

gene_IDs <- rownames(table_CD)
in_universe <- gene_IDs %in% c(DE_genes_CD , back_genes)
inSelection <- gene_IDs %in% DE_genes_CD
all_genes <- factor(as.integer(inSelection[in_universe]))
names(all_genes) <- gene_IDs[in_universe]

```

We now initialize the *topGO* data set, using the GO annotations contained in the annotation data base for the chip we are using. The `nodeSize` parameter specifies a minimum size of a GO category we want to use: i.e. here categories with less than 10 genes are not included in the testing.

```

ont <- "BP"

top_GO_data <- new("topGOdata", ontology = ont, allGenes = all_genes,
  nodeSize = 10, annot=annFUN.db, affyLib = "hugene10sttranscriptcluster.db")

```

Now the tests can be run. *topGO* offers a wide range of options, for details see the paper or the package vignette.

We run two common tests: an ordinary Fisher test for every GO category, and the “elim” algorithm, which tries to incorporate the hierarchical structure of the GO and tries “decorrelate” it in order to report the most specific significant term in the hierarchy.

The algorithm starts processing the nodes/GO categories from the highest (bottommost) level and then iteratively moves to nodes from a lower level. If a node is scored as significant, all of its genes are marked as removed in all ancestor nodes. This way, the “elim” algorithm aims at finding the most specific node for every gene.

The tests uses a 0.01 p-value cutoff by default.

```

result_top_GO_elim <- runTest(top_GO_data, algorithm = "elim", statistic = "Fisher")
result_top_GO_classic <- runTest(top_GO_data, algorithm = "classic", statistic = "Fisher")

```

We can now inspect the results. We look at the top 100 GO categories according to the “Fisher elim” algorithm. The function `GenTable` produces a table of significant GO categories, the function `printGenes` gives significant genes annotated to them.

```

res_top_GO <- GenTable(top_GO_data, Fisher.elim = result_top_GO_elim,
  Fisher.classic = result_top_GO_classic,
  orderBy = "Fisher.elim" , topNodes = 100)

```

```

genes_top_GO <- printGenes(top_GO_data, whichTerms = res_top_GO$GO.ID,
  chip = "hugene10sttranscriptcluster.db", geneCutOff = 1000)

res_top_GO$sig_genes <- sapply(genes_top_GO, function(x) {
  str_c(paste0(x[x$'raw p-value' == 2, "Symbol.id"], ";"), collapse = "")
})

head(res_top_GO[,1:8], 20)

```

	GO.ID	Term	Annotated	Significant
1	GO:0006954	inflammatory response	411	191
2	GO:0006955	immune response	1221	448
3	GO:0050853	B cell receptor signaling pathway	64	36
4	GO:0006805	xenobiotic metabolic process	91	46
5	GO:0006958	complement activation, classical pathway	38	25
6	GO:0006911	phagocytosis, engulfment	50	30
7	GO:0006910	phagocytosis, recognition	30	21
8	GO:0042742	defense response to bacterium	123	61
9	GO:0007186	G-protein coupled receptor signaling pat...	421	154
10	GO:0032496	response to lipopolysaccharide	233	106
11	GO:0050871	positive regulation of B cell activation	69	36
12	GO:0016525	negative regulation of angiogenesis	51	29
13	GO:0051897	positive regulation of protein kinase B...	58	31
14	GO:0006635	fatty acid beta-oxidation	54	32
15	GO:0070098	chemokine-mediated signaling pathway	48	27
16	GO:0014068	positive regulation of phosphatidylinosi...	43	25
17	GO:0006898	receptor-mediated endocytosis	204	80
18	GO:0048661	positive regulation of smooth muscle cel...	51	28
19	GO:0030199	collagen fibril organization	34	21
20	GO:0030335	positive regulation of cell migration	273	127
	Expected Rank in Fisher.classic Fisher.elim Fisher.classic			
1	102.61	6	1.3e-10	3.5e-22
2	304.84	4	7.7e-08	9.7e-23
3	15.98	162	8.6e-08	8.6e-08
4	22.72	171	1.2e-07	1.2e-07
5	9.49	173	1.2e-07	1.2e-07
6	12.48	176	1.5e-07	1.5e-07
7	7.49	185	2.6e-07	2.6e-07
8	30.71	114	2.7e-07	2.8e-09
9	105.11	147	4.0e-07	4.1e-08
10	58.17	66	9.1e-07	4.5e-12
11	17.23	216	1.0e-06	1.0e-06
12	12.73	219	1.1e-06	1.1e-06
13	14.48	242	3.0e-06	3.0e-06
14	13.48	161	3.3e-06	8.5e-08
15	11.98	251	3.6e-06	3.6e-06
16	10.74	252	3.6e-06	3.6e-06
17	50.93	255	4.0e-06	4.0e-06
18	12.73	263	4.5e-06	4.5e-06
19	8.49	279	5.8e-06	5.8e-06
20	68.16	35	6.2e-06	4.0e-15

Visualization of the GO-analysis results

A graph of the results can also be produced. Here we visualize the three most significant nodes according to the Fisher elim algorithm in the context of the GO hierarchy.

```
showSigOfNodes(top_GO_data, score(result_top_GO_elim), firstSigNodes = 3,
               useInfo = 'def')
```

We can see that indeed GO categories related to inflammation, signalling and immune response show up as significant. Gene set enrichment analysis has been a field of very extensive research in bioinformatics. For additional approaches see the *topGO* vignette and the references therein and also in the [GeneSetEnrichment](#) view.

A pathway enrichment analysis using reactome

The package *ReactomePA* offers the possibility to test enrichment of specific pathways using the free, open-source, curated and peer reviewed pathway *Reactome* pathway database^{22,23}. The package requires entrez identifiers, so we convert our PROBEIDs (transcript cluster identifiers) to entrez identifiers using the function `mapIds` from the package *AnnotationDbi*. This will create a named vector that maps the PROBEIDs to the entrez ones.

```
entrez_ids <- mapIds(hugene10sttranscriptcluster.db,
                    keys = rownames(table_CD),
                    keytype="PROBEID",
                    column = "ENTREZID")
```

We can now run the enrichment analysis that performs a statistical test based on the hypergeometric distribution that is the same as a one sided Fisher-test, which *topGO* calls “Fisher-classic”. Details can be found in the vignette of the *DOSE* package²⁴.

```
reactome_enrich <- enrichPathway(gene = entrez_ids[DE_genes_CD],
                                universe = entrez_ids[c(DE_genes_CD,
                                                         back_genes)],
                                organism = "human",
                                pvalueCutoff = 0.05,
                                qvalueCutoff = 0.9,
                                readable = TRUE)

reactome_enrich@result$Description <- paste0(str_sub(
  reactome_enrich@result$Description, 1, 20),
  "...")

head(summary(reactome_enrich))[1:6]
```

	ID	Description	GeneRatio	BgRatio	pvalue	p.adjust
380108	380108	Chemokine receptors ...	23/1121	33/3919	1.07e-06	0.000816
71406	71406	Pyruvate metabolism ...	22/1121	34/3919	1.21e-05	0.003710
1474244	1474244	Extracellular matrix...	71/1121	162/3919	1.75e-05	0.003710
556833	556833	Metabolism of lipids...	126/1121	325/3919	2.37e-05	0.003710
216083	216083	Integrin cell surfac...	32/1121	59/3919	2.78e-05	0.003710
535734	535734	Fatty acid, triacylg...	41/1121	82/3919	2.92e-05	0.003710

Note that we trimmed pathway names to 20 characters.

Figure 11. A graphical representation of the topGO results.

Visualizing the reactome based analysis results

The *reactomePA* package offers nice visualization capabilities. The top pathways can be displayed as a bar chart that displays all categories with a p-value below the specified cutoff.

```
barplot(reactome_enrich)
```

The “enrichment map” displays the results of the enrichment analysis as a graph, where the color represents the p-value of the pathway and the edge-thickness is proportional to the number of overlapping genes between two pathways.

```
enrichMap(reactome_enrich, n = 10, vertex.label.font = 2)
```

Again, we see pathways related to signalling and immune response.

Figure 12. Enriched Reactome pathways and their p-values as a bar chart.

Figure 13. Enriched Reactome pathways enrichment results as a graph.

The package *clusterProfiler*²⁵ can also perform these analyses using downloaded KEGG data. Furthermore, the package *EnrichmentBrowser*²⁶ additionally offers network-based enrichment analysis of individual pathways. This allows the mapping of the expression data at hand to known regulatory interactions.

Session information

As the last part of this document, we call the function *sessionInfo*, which reports the version numbers of R and all the packages used in this session. It is good practice to always keep such a record of this as it will help to track down what has happened in case an R script ceases to work or gives different results because the functions have been changed in a newer version of one of your packages. By including it at the bottom of a script, your reports will become more reproducible.

The session information should also *always* be included in any emails to the [Bioconductor support site](#) along with all code used in the analysis.

```
sessionInfo()
```

```
R version 3.3.0 (2016-05-03)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 14.04.4 LTS
```

```
locale:
```

```
[1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
[3] LC_TIME=de_DE.UTF-8      LC_COLLATE=en_US.UTF-8
[5] LC_MONETARY=de_DE.UTF-8  LC_MESSAGES=en_US.UTF-8
[7] LC_PAPER=de_DE.UTF-8     LC_NAME=C
[9] LC_ADDRESS=C             LC_TELEPHONE=C
[11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
```

```
attached base packages:
```

```
[1] grid      stats4    parallel  stats      graphics  grDevices  utils
[8] datasets  methods  base
```

```
other attached packages:
```

```
[1] bibtex_0.4.0                knitcitations_1.0.7
[3] Rgraphviz_2.16.0            EnrichmentBrowser_2.2.2
[5] pathview_1.12.0            GSEABase_1.34.0
[7] devtools_1.11.1            openxlsx_3.0.0
[9] clusterProfiler_3.0.2      ReactomePA_1.16.2
[11] DOSE_2.10.2                 limma_3.28.5
[13] multcomp_1.4-5             TH.data_1.0-7
[15] MASS_7.3-45                 survival_2.39-4
[17] DAAG_1.22                   mvtnorm_1.0-5
[19] pheatmap_1.0.8             hugene10sttranscriptcluster.db_8.4.0
[21] org.Hs.eg.db_3.3.0         pd.hugene.1.0.st.v1_3.14.1
[23] RSQLite_1.0.0              DBI_0.4-1
[25] biomaRt_2.28.0             genefilter_1.54.2
[27] topGO_2.24.0               SparseM_1.7
[29] GO.db_3.3.0                 graph_1.50.0
[31] matrixStats_0.50.2         stringr_1.0.0
[33] arrayQualityMetrics_3.28.2 ArrayExpress_1.32.0
[35] RColorBrewer_1.1-2         ggplots_3.0.1
[37] LSD_3.0                     dplyr_0.4.3
[39] ggplot2_2.1.0              geneplotter_1.50.0
[41] annotate_1.50.0             XML_3.98-1.4
[43] AnnotationDbi_1.34.3       lattice_0.20-33
[45] oligo_1.36.1               Biostrings_2.40.1
[47] XVector_0.12.0             oligoClasses_1.34.0
[49] DESeq2_1.12.3              SummarizedExperiment_1.2.2
[51] GenomicRanges_1.24.0       GenomeInfoDb_1.8.1
[53] IRanges_2.6.0              S4Vectors_0.10.1
[55] Biobase_2.32.0             BiocGenerics_0.18.0
[57] knitr_1.13                  BiocStyle_2.0.2
```

```
loaded via a namespace (and not attached):
```

```
[1] R.utils_2.3.0                beadarray_2.22.2
[3] BiocParallel_1.6.2          munsell_0.4.3
[5] codetools_0.2-14           preprocessCore_1.34.0
[7] chron_2.3-47                withr_1.0.1
[9] colorspace_1.2-6            GOSemSim_1.30.2
```

[11] BiocInstaller_1.22.2	Category_2.38.0
[13] OrganismDbi_1.14.1	setRNG_2013.9-1
[15] labeling_0.3	KEGGgraph_1.30.0
[17] hwriter_1.3.2	biovizBase_1.20.0
[19] affxparser_1.44.0	R6_2.1.2
[21] illuminaio_0.14.0	gridSVG_1.5-0
[23] RJSONIO_1.3-0	locfit_1.5-9.1
[25] bitops_1.0-6	reshape_0.8.5
[27] assertthat_0.1	scales_0.4.0
[29] nnet_7.3-12	gtable_0.2.0
[31] Cairo_1.5-9	affy_1.50.0
[33] ggbio_1.20.1	sandwich_2.3-4
[35] ensemblDb_1.4.3	splines_3.3.0
[37] lazyeval_0.1.10	rtracklayer_1.32.0
[39] acepack_1.3-3.3	dichromat_2.0-0
[41] yaml_2.1.13	reshape2_1.4.1
[43] SVGAnnotation_0.93-1	GenomicFeatures_1.24.2
[45] httpuv_1.3.3	qvalue_2.4.2
[47] Hmisc_3.17-4	RBGL_1.48.1
[49] tools_3.3.0	affyio_1.42.0
[51] ff_2.2-13	Rcpp_0.12.5
[53] plyr_1.8.3	zlibbioc_1.18.0
[55] RCurl_1.95-4.8	rpart_4.1-10
[57] openssl_0.9.4	zoo_1.7-13
[59] cluster_2.0.4	magrittr_1.5
[61] data.table_1.9.6	DO.db_2.9
[63] reactome.db_1.55.0	mime_0.4
[65] evaluate_0.9	xtable_1.8-2
[67] gcrma_2.44.0	gridExtra_2.2.1
[69] KernSmooth_2.23-15	R.oo_1.20.0
[71] ReportingTools_2.12.2	htmltools_0.3.5
[73] GOstats_2.38.0	Formula_1.2-1
[75] tidyr_0.4.1	lubridate_1.5.6
[77] formatR_1.4	rappdirs_0.3.1
[79] Matrix_1.2-6	vsn_3.40.0
[81] R.methodsS3_1.7.1	gdata_2.17.0
[83] igraph_1.0.1	GenomicAlignments_1.8.0
[85] RefManager_0.10.13	foreign_0.8-66
[87] foreach_1.4.3	BeadDataPackR_1.24.2
[89] affyPLM_1.48.0	AnnotationForge_1.14.2
[91] VariantAnnotation_1.18.1	digest_0.6.9
[93] rmarkdown_0.9.6	base64_2.0
[95] edgeR_3.14.0	shiny_0.13.2
[97] Rsamtools_1.24.0	gtools_3.5.0
[99] graphite_1.18.0	PFAM.db_3.3.0
[101] BSgenome_1.40.0	GGally_1.0.1
[103] KEGGREST_1.12.2	httr_1.1.0
[105] interactiveDisplayBase_1.10.3	png_0.1-7
[107] iterators_1.0.8	bit_1.1-12
[109] stringi_1.1.1	AnnotationHub_2.4.2
[111] latticeExtra_0.6-28	caTools_1.17.1
[113] memoise_1.0.0	

Dataset 1. R markdown document to reproduce the results obtained in the article

<http://dx.doi.org/10.5256/f1000research.8967.d124759>

This file allows the reader to reproduce the analysis results obtained in the article.

Data and software availability

This article is based on an **R markdown** file (MA-Workflow.Rmd) which is available as Dataset 1 (Dataset 1. R markdown document to reproduce the results obtained in the article, [10.5256/f1000research.8967.d124759](http://dx.doi.org/10.5256/f1000research.8967.d124759))³¹ and will also become available as a **Bioconductor workflow**. This file allows the reader to reproduce the analysis results obtained in this article. All data analyzed are downloaded from ArrayExpress.

Competing interests

The author declares that there are no competing interests.

Grant information

The author(s) declared that no grants were involved in supporting this work.

Acknowledgements

The author would like to thank Vladislava Milchevskaya, Julian Gehring and Mike Smith for helpful comments on and small contributions to the workflow. This workflow draws a lot of inspiration from the Bioconductor books^{27,28} as well as Love *et al.*'s workflow for gene level analysis of RNA-Seq data²⁹. James W. MacDonald provided valuable information on the evolution of Affymetrix arrays in some of his posts of on the Bioconductor mailing list/support site. The author would also like to thank him for some friendly personal correspondence about the annotation resources available for microarrays in Bioconductor.

References

1. Palmieri O, Creanza TM, Bossa F, *et al.*: **Genome-wide Pathway Analysis Using Gene Expression Data of Colonic Mucosa in Patients with Inflammatory Bowel Disease**. *Inflamm Bowel Dis*. 2015; 21(6): 1260–8.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
2. Kolesnikov N, Hastings E, Keays M, *et al.*: **ArrayExpress update—simplifying data submissions**. *Nucleic Acids Res*. 2015; 43(Database issue): D1113–D1116.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
3. Huber W, Carey VJ, Gentleman R, *et al.*: **Orchestrating high-throughput genomic analysis with Bioconductor**. *Nat Methods*. 2015; 12(2): 115–121.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
4. Carvalho BS, Irizarry RA: **A framework for oligonucleotide microarray preprocessing**. *Bioinformatics*. 2010; 26(19): 2363–7.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
5. Kauffmann A, Gentleman R, Huber W: **arrayQualityMetrics—a bioconductor package for quality assessment of microarray data**. *Bioinformatics*. 2009; 25(3): 415–6.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
6. Irizarry RA, Bolstad BM, Collin F, *et al.*: **Summaries of Affymetrix GeneChip probe level data**. *Nucleic Acids Res*. 2003; 31(4): e15.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
7. Bolstad BM, Irizarry RA, Astrand M, *et al.*: **A comparison of normalization methods for high density oligonucleotide array data based on variance and bias**. *Bioinformatics*. 2003; 19(2): 185–193.
[PubMed Abstract](#) | [Publisher Full Text](#)
8. Irizarry RA, Hobbs B, Collin F, *et al.*: **Exploration, normalization, and summaries of high density oligonucleotide array probe level data**. *Biostatistics*. 2003; 4(2): 249–264.
[PubMed Abstract](#) | [Publisher Full Text](#)
9. Huber W, von Heydebreck A, Sultmann H, *et al.*: **Variance stabilization applied to microarray data calibration and to the quantification of differential expression**. *Bioinformatics*. 2002; 18(Suppl. 1): S96–S104.
[PubMed Abstract](#) | [Publisher Full Text](#)
10. Ritchie ME, Phipson B, Wu D, *et al.*: **limma powers differential expression analyses for RNA-sequencing and microarray studies**. *Nucleic Acids Res*. 2015; 43(7): e47.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
11. Smyth GK: **Linear models and empirical bayes methods for assessing differential expression in microarray experiments**. *Stat Appl Genet Mol Biol*. 2004; 3(1): 1–25, Article3.
[PubMed Abstract](#) | [Publisher Full Text](#)
12. Arloth J, Bader DM, Röh S, *et al.*: **Re-Annotator: Annotation Pipeline for Microarray Probe Sequences**. *PLoS One*. 2015; 10(10): e0139516.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
13. Love MI, Huber W, Anders S: **Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2**. *Genome Biol*. 2014; 15(12): 550.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
14. Robinson MD, McCarthy DJ, Smyth GK: **edgeR: a Bioconductor package for differential expression analysis of digital gene expression data**. *Bioinformatics*. 2010; 26(1): 139–140.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
15. Hothorn T, Bretz F, Westfall P: **Simultaneous inference in general**

- parametric models. *Biom J*. 2008; **50**(3): 346–363.
[PubMed Abstract](#) | [Publisher Full Text](#)
16. Gene Ontology Consortium: **Gene Ontology Consortium: going forward**. *Nucleic Acids Res*. 2015; **43**(Database issue): D1049–56.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 17. du Plessis L, Skunca N, Dessimoz C: **The what, where, how and why of gene ontology—a primer for bioinformaticians**. *Brief Bioinform*. 2011; **12**(6): 723–735.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 18. Ashburner M, Ball CA, Blake JA, *et al.*: **Gene ontology: tool for the unification of biology**. The Gene Ontology Consortium. *Nat Genet*. 2000; **25**(1): 25–29.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 19. Bourgon R, Gentleman R, Huber W: **Independent filtering increases detection power for high-throughput experiments**. *Proc Natl Acad Sci U S A*. 2010; **107**(21): 9546–9551.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 20. Eden E, Navon R, Steinfeld I, *et al.*: **GOrrilla: a tool for discovery and visualization of enriched GO terms in ranked gene lists**. *BMC Bioinformatics*. 2009; **10**(1): 48.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 21. Alexa A, Rahnenfuhrer J, Lengauer T: **Improved scoring of functional groups from gene expression data by decorrelating GO graph structure**. *Bioinformatics*. 2006; **22**(13): 1600–1607.
[PubMed Abstract](#) | [Publisher Full Text](#)
 22. Croft D, Mundo AF, Haw R, *et al.*: **The Reactome pathway knowledgebase**. *Nucleic Acids Res*. 2014; **42**(Database issue): D472–D477.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 23. Fabregat A, Sidiropoulos K, Garapati P, *et al.*: **The Reactome pathway knowledgebase**. *Nucleic Acids Res*. 2016; **44**(D1): D481–D487.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 24. Yu G, Wang LG, Yan GR, *et al.*: **DOSE: an R/Bioconductor package for disease ontology semantic and enrichment analysis**. *Bioinformatics*. 2015; **31**(4): 608–609.
[PubMed Abstract](#) | [Publisher Full Text](#)
 25. Yu G, Wang LG, Han Y, *et al.*: **clusterProfiler: an R package for comparing biological themes among gene clusters**. *OMICS*. 2012; **16**(5): 284–287.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 26. Geistlinger L, Csaba G, Zimmer R: **Bioconductor's EnrichmentBrowser: seamless navigation through combined results of set- & network-based enrichment analysis**. *BMC Bioinformatics*. 2016; **17**(1): 45.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 27. Hahne F, Huber W, Gentleman R, *et al.*: **Bioconductor Case Studies**. *Use R!*. Springer, 2008.
[Publisher Full Text](#)
 28. Gentleman R, Carey VJ, Huber W, *et al.*: **Bioinformatics and Computational Biology Solutions Using R and Bioconductor**. *Statistics for Biology and Health*. Springer, 2005.
[Publisher Full Text](#)
 29. Love MI, Anders S, Kim V, *et al.*: **RNA-Seq workflow: gene-level exploratory analysis and differential expression [version 1; referees: 2 approved]**. *F1000Res*. 2015; **4**: 1070.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 30. Yu G, He QY: **ReactomePA: an R/Bioconductor package for reactome pathway analysis and visualization**. *Mol Biosyst*. 2016; **12**(2): 477–479.
[PubMed Abstract](#) | [Publisher Full Text](#)
 31. Klaus B: **Dataset 1 in: An end to tend workflow for differential gene expression using Affymetrix microarrays**. *F1000Research*. 2016.
[Data Source](#)

Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 08 July 2016

<https://doi.org/10.5256/f1000research.9647.r14390>

© 2016 Rau A. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Andrea Rau

UMR 1313 GABI (Animal Genetics and Integrative Biology), French National Institute for Agricultural Research (INRA), Jouy-en-Josas, France

Klaus illustrates an analysis workflow of Affymetrix microarrays for an experimental design with paired samples from two tissues in two separate diseases. The workflow covers steps including downloading and loading raw CEL files into R/Bioconductor, pre-processing and normalization, differential analysis via limma, and functional/pathway enrichment analysis, with detailed R scripts throughout. The workflow is a nice complement to resources that are already available on similar topics (notably the Limma user's guide) as it unites into a single document topics that have been discussed elsewhere in diverse forums.

Major remarks:

- Although the normalization/linear model/contrast sections are technically correct, I find them to be overly disruptive to the presentation of the analysis. In particular, I would suggest moving the normalization section to an appendix. For the linear models section, presentation of ordinary least squares estimators, matrix notation, distributional assumptions of residuals, etc. seems outside of the scope of this work (or could potentially be presented in an appendix as well). I think that it would be more helpful if, rather than using a separate illustrative example based on the *toycars* data, the author described in detail: 1) the design matrix for the Palmieri example; 2) a linear model for a single gene from the Palmieri data; 3) a linear model fit independently for each gene, sharing information across all genes (limma), with a brief discussion about the advantage of such an approach; 4) a simplified discussion of various contrasts relevant for this particular study.
- I was surprised to see that differential analysis conclusions are based on the raw p-values rather than those adjusted for multiple testing.
- I have two additional suggestions that could be of great practical interest to many readers:
 1. I think it would be nice to have a discussion (or at least some references) discussing what can be done if hypotheses are NOT met (e.g., the raw p-values are not uniformly distributed

between 0 and 1 for genes under H_0). I have in mind the author's recent discussion about using `fdrtool` to estimate the variance of the null model in the context of differential analyses via `DESeq2` (<http://www-huber.embl.de/users/klaus/Teaching/DESeq2Predoc2014.html>).

2. In addition to the fixed patient effect model presented here, I think it would be interesting to present an alternative strategy possible with *limma* for such a design: estimating correlation within patient blocks using `duplicateCorrelation()`, see for example section 17.3.6 in the *limma* User's Guide. Note that I am not suggesting a detailed comparison or presentation of mixed models (that is of course well beyond the scope of this work!), but it may be useful to discuss the code/results for such an approach.

Minor remarks:

- I would suggest specifying in the abstract that the study "...compares paired inflamed and non-inflamed colon tissue...", as this was not immediately clear otherwise. In addition, on page 2/paragraph 1, the author writes that the original paper studied "differences between patients suffering UC or CD", which makes it sound like the disease comparison was of interest, when in fact it was the intra-patient tissue differences that were studied. It may in fact be helpful to present the experimental design in greater detail at the start of the paper; it was not clear to me until page 23 that the design actually involved paired samples from individuals.
- Page 12, the author says "In order to infer a cutoff from the data, we inspect the histogram of the median-intensities. We visually fit a central normal distribution given by $0.5 \cdot N(5.1, 1.18)$ to the probe-wise medians". From the R code, it appears that the distribution fit was $N(\text{emp_mu} = 5.3, \text{emp_sd} = 1.19)$, where `emp_mu` and `emp_sd` were estimated from the median expression values -- this is unclear from the expression "visually fit" in the text description. Also unclear to me is why `prop_cental` was set to be 0.5?
- The phrase "from the XXXX" is often used when referencing R packages, rather than "from XXXX" or "from the XXX package".
- The comparison to the original paper results (page 26) seems unnecessary.
- I found it a bit distracting that none of the figures are explicitly referred to by number in the text (e.g., "In Figure 1, ...").
- There are occasionally very verbose outputs included in the text (e.g. `head(pData(raw_data))`) that produce multiple pages of output that are not particularly useful.
- There are a few English typos throughout, e.g. "inflammed" should be "inflamed", "constrains" should be "constraints", "transcriptct clusters" should be "transcript clusters", etc.
- Since a very large number of R packages are loaded for the analysis on page 2, it would likely be helpful to either reduce the number of packages needed to be the strict minimum necessary, or to group packages by theme/functionality via comments, for example:

1. General Bioconductor packages
library(Biobase)
library(oligoClasses)
2. Annotation and data packages
library(pd.hugene.1.0.st.v1)
3. Quality control and pre-processing packages
library(oligo)
4. Analysis and statistics packages
library(limma)
library(mvtnorm)
5. Plotting and color options packageslibrary(gplots)
library(ggplot2)
6. Formatting/documentation packages
library(knitr)
library(BiocStyle)
library(dplyr)

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Author Response 14 Jun 2018

Bernd Klaus, EMBL Heidelberg, Heidelberg, Germany

Klaus illustrates an analysis workflow of Affymetrix microarrays for an experimental design with paired samples from two tissues in two separate diseases. The workflow covers steps including downloading and loading raw CEL files into R/Bioconductor, pre-processing and normalization, differential analysis via limma, and functional/pathway enrichment analysis, with detailed R scripts throughout. The workflow is a nice complement to resources that are already available on similar topics (notably the Limma user's guide) as it unites into a single document topics that have been discussed elsewhere in diverse forums.

#Major remarks:

remark 1

Although the normalization/linear model/contrast sections are technically correct, I find them to be overly disruptive to the presentation of the analysis. In particular, I would suggest moving the normalization section to an appendix. For the linear models section, presentation of ordinary least squares estimators, matrix notation, distributional assumptions of residuals, etc. seems outside of the scope of this work (or could potentially

be presented in an appendix as well). I think that it would be more helpful if, rather than using a separate illustrative example based on the toycars data, the author described in detail:

- 1) the design matrix for the Palmieri example;
- 2) a linear model for a single gene from the Palmieri data;
- 3) a linear model fit independently for each gene, sharing information across all genes (limma), with a brief discussion about the advantage of such an approach;
- 4) a simplified discussion of various contrasts relevant for this particular study.

Answer:

The linear model sections not directly related to the dataset at hand were removed from the workflow. In general, we have tried to tailor the revised workflow to beginners in R and Bioconductor. Therefore, rather than discussing elaborate tools / techniques in detail, we try to point to the relevant literature where appropriate.

This is helpful for the reader who needs to deviate from our workflow for his or her own data and maintains a straightforward workflow design.

Concerning the single points mentioned in the comment above:

- 1) The design matrix for the Palmieri example is now explained in detail.
- 2) In the subsequent part of the workflow, the advice of the reviewer to implement a linear model for a single gene from the Palmieri data was adopted by fitting a linear model to the "CRAT" gene, and testing it for differential expression.
- 3) The advantage of shared information across all genes is discussed briefly when introducing the "eBayes"-method in the "Contrasts and Hypotheses tests" part.
- 4) In this workflow, we restrict the contrast analysis to the contrast between non-inflamed and inflamed tissue, as it is the one that is also analyzed in the original paper, and do not include additional contrasts in order to keep the workflow concise.

remark 2

I was surprised to see that differential analysis conclusions are based on the raw p-values rather than those adjusted for multiple testing.

Answer: This is indeed a possible cause of confusion. In the workflow, parts of the analysis are based on raw p-values in order to make the results comparable to the original paper. However, we now mention the caveats of this approach more, introduce FDRs in a "hands-on" manner explicitly and caution against the use of raw p-values in practice.

For the subsequent enrichment analyses,

DE genes are identified using an FDR cutoff of 10%.

I have two additional suggestions that could be of great practical interest to many readers:

remark 3

I think it would be nice to have a discussion (or at least some references) discussing what can be done if hypotheses are NOT met (e.g., the raw p-values are not uniformly distributed between 0 and 1 for genes under H0). I have in mind the author's recent discussion about using `fdrtool` to estimate the variance of the null model in the context of differential analyses via `DESeq2` (<http://www-huber.embl.de/users/klaus/Teaching/DESeq2Predoc2014.html>).

Answer: As mentioned above, and in concordance with the requests of Jim, we tried to avoid introducing any additional (advanced) methods in order to not confuse the novice R/Bioconductor user. Nevertheless, for workflow users encountering the problem of unusual p-value distributions, we have included references to this phenomenon and its implications by referring to the article on false discovery rate estimation by Korbinnan Strimmer and chapter 1-6 of Efron's book on Large-Scale Inference.

remark 4

In addition to the fixed patient effect model presented here, I think it would be interesting to present an alternative strategy possible with `limma` for such a design: estimating correlation within patient blocks using `duplicateCorrelation()`, see for example section 17.3.6 in the `limma` User's Guide. Note that I am not suggesting a detailed comparison or presentation of mixed models (that is of course well beyond the scope of this work!), but it may be useful to discuss the code/results for such an approach.

Answer: We acknowledge the advantages in certain applications of a mixed model using `duplicateCorrelation()` compared to a fixed patient effect model used in this workflow.

Given that the workflow is tailored towards inexperienced users, we consider the fixed patient effect model more intuitively understandable. However, we now note explicitly that such an analysis could be performed.

Minor remarks:

remark 5

I would suggest specifying in the abstract that the study "...compares paired inflamed and non-inflamed colon tissue...", as this was not immediately clear otherwise. In addition, on

page 2/paragraph 1, the author writes that the original paper studied "differences between patients suffering UC or CD", which makes it sound like the disease comparison was of interest, when in fact it was the intra-patient tissue differences that were studied. It may in fact be helpful to present the experimental design in greater detail at the start of the paper; it was not clear to me until page 23 that the design actually involved paired samples from individuals.

Answer: The experimental design is now described in greater detail in abstract and introduction. Also, the sentence that the original paper studied "differences between patients suffering from UC or CD" was removed.

remark 6

Page 12, the author says "In order to infer a cutoff from the data, we inspect the histogram of the median-intensities. We visually fit a central normal distribution given by $0.5 \cdot N(5.1, 1.18)$ to the probe-wise medians". From the R code, it appears that the distribution fit was $N(\text{emp_mu} = 5.3, \text{emp_sd} = 1.19)$, where emp_mu and emp_sd were estimated from the median expression values -- this is unclear from the expression "visually fit" in the text description. Also unclear to me is why prop_central was set to be 0.5?

Answer: In an attempt to make low-intensity-filtering more intuitive, we removed the fitting a normal distribution to the histogram of the probe-wise medians. Instead, we now visually set a vertical cutoff line to the histogram and filter genes with a lower intensity than the cutoff in at least as many samples as the smallest experimental group has.

additional minor remarks

The phrase "from the XXXX" is often used when referencing R packages, rather than "from XXXX" or "from the XXX package".

Answer: Thank you, the respective sentences were corrected accordingly.

The comparison to the original paper results (page 26) seems unnecessary.

Answer: The comparison to the original paper serves as a "proof of principle" for the implemented workflow (to show that what we are doing makes sense). Additionally, we wanted to show that it is relatively straightforward to re-analyze publicly available microarray data using R and Bioconductor.

I found it a bit distracting that none of the figures are explicitly referred to by number in the text (e.g., "In Figure 1, ...").

Answer: Thank you, the respective parts were changed accordingly.

There are occasionally very verbose outputs included in the text (e.g. `head(pData(raw_data)))` that produce multiple pages of output that are not particularly useful.

Answer: The output was shortened where appropriate. The `pdata` call in particular was retained, as we feel that it is interesting to the readers to see what kind of information comes with a dataset from Array Express.

There are a few English typos throughout, e.g. "inflammed" should be "inflamed", "constrains" should be "constraints", "transcriptct clusters" should be "transcript clusters", etc.

Answer: Thanks, we tried to perform thorough spell-checking in the revised version.

Since a very large number of R packages are loaded for the analysis on page 2, it would likely be helpful to either reduce the number of packages needed to be the strict minimum necessary, or to group packages by theme/functionality via comments, for example:

General Bioconductor packages

```
library(Biobase)
library(oligoClasses)
```

Annotation and data packages

```
library(pd.hugene.1.0.st.v1)
```

Quality control and pre-processing packages

```
library(oligo)
```

Analysis and statistics packages

```
library(limma)
library(mvtnorm)
```

Plotting and color options packages

```
library(gplots)
library(ggplot2)
```

Formatting/documentation packages

```
library(knitr)
library(BiocStyle)
library(dplyr)
```

Answer: Thank you for this very useful suggestion, we implemented it in the revised version; the package-import statements at the beginning of the workflow is now grouped by topic.

Competing Interests: No competing interests were disclosed.

Reviewer Report 05 July 2016

<https://doi.org/10.5256/f1000research.9647.r14392>

© 2016 MacDonald J. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

James W. MacDonald

Department of Environmental and Occupational Health Sciences, University of Washington, Seattle, WA, USA

This manuscript is intended to take the reader through a complete analysis of Affymetrix Gene ST arrays, based on a set of arrays downloaded from ArrayExpress. The author covers each step from quality control of the raw data all the way to making comparisons using linear models and testing and visualizing pathways or gene sets.

Major comments

While this manuscript is technically correct (e.g., the code does what the author claims, the explanatory text is valid), I am not sure it is as useful as it could be for its intended audience. In other words, this manuscript is intended to provide an inexperienced reader (inexperienced in either R/Bioconductor or statistics or both) with a road map they can follow to learn how to analyze microarray data. However, both the code and the statistical explanations are far too complex to be useful for such a person.

As an example, the author explains in mathematical terms what the background correction and summarization steps are intended to accomplish. While this is an important step in the analysis, this could instead be explained in heuristic terms that would be far more approachable for a less statistically savvy audience, while still conveying the general idea.

The section on linear modeling and design matrices are similarly impenetrable for non-statisticians. The *limma User's Guide* has dozens of examples of model matrices, with clear interpretations of the model coefficients. Yet as the author notes, this is probably the number one question on the Bioconductor support site. Rather than re-explaining something that most people clearly don't understand, it would be much more helpful to focus on a single model matrix, and provide a clear, heuristic explanation. The easiest to understand is the most basic orthogonal model that computes the mean of each group, followed by a contrast matrix to make comparisons of interest.

The overview on linear models is well beyond the scope of this manuscript, and should be excised.

The same is true for the section on testing general linear hypotheses. Ideally, an analyst using these tools would understand what they are doing from a statistical perspective, but it's difficult enough for a novice user to comprehend what the code is doing without trying to also understand the statistics.

Similarly, the code is more complex than necessary. If the goal is to teach novice users about Bioconductor packages, then the code should be restricted as much as possible to those packages and base R. While ggplot2 style graphics, magrittr style function piping and dplyr two-table verbs may be useful for more advanced R users, in this context they are an added distraction.

An example of overly-complex code is the filtering step. There are any number of ways to filter out genes that are arguably not expressed. The example is a very sophisticated way to perform this task, but a novice who is just learning doesn't require sophistication, they require something they can understand. Choosing a cutoff based on the distribution of probesets across each array, and then filtering all genes where fewer than 14 samples exceed this cutoff is not very sophisticated, but it is easy to understand, and would only require a few lines of code.

The learning curve for R and Bioconductor is steep, and making this manuscript both simpler in terms of the code, and more focused by excluding most of the statistics and any example analyses that do not involve the microarray data would make it more approachable for inexperienced users.

Minor comments

In the download step, `getAE()` is used to download the data, but then the SDRF file is downloaded directly. This is confusing, as `getAE()` has already downloaded that file. Is there a particular reason for the extra step?

To test for model matrices that are not full rank, it's easier to use either `nonEstimable()` or `is.fullrank()` from the `limma` package.

Conclusion

As noted above, this manuscript is technically accurate, and the code does exactly what the author claims. But it is too ambitious for the intended audience. Most people who could benefit from this manuscript are not statistically savvy, and the statistical sections will simply confuse them. In addition, it is not likely that many potential readers will be familiar with packages from the 'Hadleyverse', and incorporating those packages in the manuscript makes the code more difficult to understand.

Paring the manuscript down to very simple heuristic statistical explanations, and limiting the code to functions from base R and the various Bioconductor packages being illustrated would make the manuscript more useful for its intended audience.

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard, however I have significant reservations, as outlined above.

Author Response 14 Jun 2018

Bernd Klaus, EMBL Heidelberg, Heidelberg, Germany

Major comments

comment 1

While this manuscript is technically correct (e.g., the code does what the author claims, the explanatory text is valid), I am not sure it is as useful as it could be for its intended audience. In other words, this manuscript is intended to provide an inexperienced reader (inexperienced in either R/Bioconductor or statistics or both) with a road map they can follow to learn how to analyze microarray data. However, both the code and the statistical explanations are far too complex to be useful for such a person.

As an example, the author explains in mathematical terms what the background correction and summarization steps are intended to accomplish. While this is an important step in the analysis, this could instead be explained in heuristic terms that would be far more approachable for a less statistically savvy audience, while still conveying the general idea.

Answer: As suggested, the general style of the workflow was changed a lot in order to guide inexperienced readers through the analysis of microarray data. Code is explained in greater detail, and all technical parts not directly related to the data at hand have either been removed or improved.

However, we have decided to keep the the mathematical explanation of background correction and summarization, as we feel that the (rather simple) formulas are easier to understand than text.

comment 2

The section on linear modeling and design matrices are similarly impenetrable for non-statisticians. The limma User's Guide has dozens of examples of model matrices, with clear interpretations of the model coefficients. Yet as the author notes, this is probably the number one question on the Bioconductor support site. Rather than re-explaining something that most people clearly don't understand, it would be much more helpful to focus on a single model matrix, and provide a clear, heuristic explanation. The easiest to understand is the most basic orthogonal model that computes the mean of each group, followed by a contrast matrix to make comparisons of interest.

The overview on linear models is well beyond the scope of this manuscript, and should be excised. The same is true for the section on testing general linear hypotheses. Ideally, an analyst using these tools would understand what they are doing from a statistical perspective, but it's

difficult enough for a novice user to comprehend what the code is doing without trying to also understand the statistics.

Answer: We gladly adopted this point of criticism and eliminated the rather technical part in which the theory behind design matrices and linear modelling is explained. Instead, we focused on explaining the design matrix we have used in the dataset at hand in easy-to-follow terms and present its use in the context of single linear model for a single gene.

comment 3

Similarly, the code is more complex than necessary. If the goal is to teach novice users about Bioconductor packages, then the code should be restricted as much as possible to those packages and base R. While ggplot2 style graphics, magrittr style function piping and dplyr two-table verbs may be useful for more advanced R users, in this context they are an added distraction.

An example of overly-complex code is the filtering step. There are any number of ways to filter out genes that are arguably not expressed. The example is a very sophisticated way to perform this task, but a novice who is just learning doesn't require sophistication, they require something they can understand. Choosing a cutoff based on the distribution of probesets across each array, and then filtering all genes where fewer than 14 samples exceed this cutoff is not very sophisticated, but it is easy to understand, and would only require a few lines of code.

Answer: While not all of the code from the packages like ggplot2, magrittr and dplyr was removed, an effort was made to explain the code in more detail in order to give the inexperienced user the possibility to understand what it does. In particular, we think that ggplot2 is by now very commonly used and probably becoming a defacto standard plotting package.

In the low-intensity filtering step, the fitting of a normal distribution to the histogram was eliminated for the sake of clarity. Instead, we now visually set a vertical cutoff line to the histogram and filter genes with a lower intensity than the cutoff in at least as many samples as the smallest experimental group has.

comment 4

The learning curve for R and Bioconductor is steep, and making this manuscript both simpler in terms of the code, and more focused by excluding most of the statistics and any example analyses that do not involve the microarray data would make it more approachable for inexperienced users.

Answer: Thank you for this useful comment. We have made some effort to tailor the revised version to the beginner in R / Bioconductor and / or statistics. The revised article contains a more focused workflow with more in depth explanations of the code.

Minor comments

In the download step, getAE() is used to download the data, but then the SDRF file is downloaded directly. This is confusing, as getAE() has already downloaded that file. Is there a particular reason for the extra step?

Answer: There is no particular reason for this step, the code has been changed in order to import the SDRF file from the already downloaded data.

To test for model matrices that are not full rank, it's easier to use either nonEstimable() or is.fullrank() from the limma package.

Answer: This section was removed, as it is quite technical and is really more of a "sanity check" than a necessity.

Competing Interests: No competing interests were disclosed.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research