Short version of the vignette *End-to-end analysis of cell-based screens: from raw intensity readings to the annotated hit list*

Michael Boutros, Lígia Brás and Wolfgang Huber

November 28, 2007

Contents

1 Introduction

This is a short version of the technical report *End-to-end analysis of cell-based screens: from raw intensity readings to the annotated hit list*, focusing on the essential steps necessary to run an analysis of a cell-based high-throughput screen (HTS), from raw intensity readings to an annotated hit list, using the *cellHTS2* package. This package is a revised version of the *cellHTS* package [1], which accompanied the paper *Analysis of cell-based RNAi screens* by Michael Boutros, Lígia Brás and Wolfgang Huber [1].

This report has been produced as a reproducible document [4]. It contains the actual computer instructions for the method it describes, and these in turn produce all results, including the figures and tables that are shown here. The computer instructions are given in the language R, thus, in order to reproduce the computations shown here, you will need an installation of R (version 2.3 or greater) together with a recent version of the package *cellHTS2* and of some other add-on packages.

To reproduce the computations shown here, you do not need to type them or copy-paste them from the PDF file; rather, you can take the file *cellhts2.Rnw* in the *doc* directory of the package, open it in a text editor, run it using the R command *Sweave*, and modify it to your needs.

For a more complete analysis, please refer to the complete vignette, which should be manually produced from the file *cellhts2Complete.Rnw* that can be found in the *scripts* directory of the package. The complete vignette includes sections exemplifying how to add more annotation from public databases, how to perform an analysis for Gene Ontology categories, and compares the obtained results with previously reported ones.

First, we load the package.

```
> library("cellHTS2")
```

---

[1]To convert a S3 class *cellHTS* object obtained using *cellHTS* package to an S4 class *cellHTS* object suitable for *cellHTS2*, please see the function *convertOldCellHTS*.

Filename	Plate	Replicate
FT01-G01.txt	1	1
FT01-G02.txt	1	2
FT02-G01.txt	2	1
FT02-G02.txt	2	2
FT03-G01.txt	3	1
...	...	...

Table 1: Selected lines from the example plate list file `Platelist.txt`.

2 Reading the intensity data

We consider a cell-based screen that was conducted in microtiter plate format, where a library of double-stranded RNAs was used to target the corresponding genes in cultured *Drosophila* Kc_{167} cells [2]. Each of the wells in the plates contains either a gene-specific probe, a control, or it can be empty. The experiments were done in duplicate, and the viability of the cells after treatment was recorded by a plate reader measuring luciferase activity, which is indicative of ATP levels. Although this set of example data corresponds to a single-channel screening assay, the *cellHTS2* package can also deal with cases where there are readings from more channels, corresponding to different reporters.

Usually, the measurements from each replicate and each channel come in individual result files. The set of available result files and the information about them (which plate, which replicate, which channel) is contained in a spreadsheet, which we call the *plate list file*. This file should contain the following columns: *Filename*, *Plate*, and *Replicate*. The last two columns should be numeric, with values ranging from 1 to the maximum number of plates or replicates, respectively. The first few lines of an example plate list file are shown in Table 1, while Table 2 shows the first few lines from one of the plate result files listed in the plate list file.

The first step of the analysis is to read the plate list file, to read all the intensity files, and to assemble the data into a single R object that is suitable for subsequent analyses. The main component of that object is one big table with the intensity readings of all plates, channels, and replicates. We demonstrate the R instructions for this step. First we define the path where the input files can be found.

```
> experimentName <- "KcViab"
> dataPath <- system.file(experimentName, package = "cellHTS2")
```

FT01-G01	A01	887763
FT01-G01	A02	958308
FT01-G01	A03	1012685
FT01-G01	A04	872603
FT01-G01	A05	1179875
...	...	...

Table 2: Selected lines from the example signal intensity file `FT01-G01.txt`.

In this example, the input files are in the `KcViab` directory of the *cellHTS2* package. To read your own data, modify `dataPath` to point to the directory where they reside. We show the names of 12 files from our example directory:

```
> dataPath

[1] "/tmp/Rinst788361513/cellHTS2/KcViab"

> rev(dir(dataPath))[1:12]

 [1] "Screenlog.txt"          "Platelist.txt"      "Plateconf.txt"
 [4] "GeneIDs_Dm_HFA_1.1.txt" "FT57-G02.txt"       "FT57-G01.txt"
 [7] "FT56-G02.txt"           "FT56-G01.txt"       "FT55-G02.txt"
[10] "FT55-G01.txt"           "FT54-G02.txt"       "FT54-G01.txt"
```

and read the data into the object x

```
> x <- readPlateList("Platelist.txt", name = experimentName, path = dataPath)

> x

cellHTS (storageMode: lockedEnvironment)
assayData: 21888 features, 2 samples
  element names: ch1
phenoData
  sampleNames: 1, 2
  varLabels and varMetadata description:
    replicate: Replicate number
    assay: Biological assay
  additional varMetadata: channel
featureData
  featureNames: 1, 2, ..., 21888  (21888 total)
```

```
  fvarLabels and fvarMetadata description:
    plate: Plate number
    well: Well ID
    controlStatus: Well annotation
experimentData: use 'experimentData(object)'
state:  configured = FALSE
        normalized = FALSE
        scored = FALSE
        annotated = FALSE
Number of plates: 57
Plate dimension: nrow = 16, ncol = 24
Number of batches:
Well annotation:
Annotation:
```

The plate format used in the screen (96-well or 384-well plate design) is automatically determined from the raw intensity files, when calling the *read-PlateList* function.

2.1 Importing intensity data files with other formats

The function *readPlateList* has the argument `importFun` that can be used to provide a different import function to read plate result files with a format different from that shown in Table 2. For example, to import plate data files from EnVision plate reader, set `importFun=getEnVisionRawData` or `importFun=getEnvisionCrosstalkCorrectedData` when calling *readPlateList*. Please see the help page of function *getEnVisionRawData* for an example. Another import function ("importData.R") is given together with the example data set for a two-way assay in the directory called *TwoWayAssay* of this package.

While for the above data sets, the measurements from each replicate and channel come in separate result files, this is not the case when measurement files are provided in the HTanalyser format. In this case, each output file contains meta-experimental data together with intensity readings (in a matrix-like layout) of a set of plates made for the same replicate or screen. Thus, there is no need to have a plate list file, and instead of using *readPlateList* we should call the function *readHTAnalystData*. Please see the help page for this function (`?  readPlateHTAnalystData`), where we illustrate how it can be applied to import HTAnalyst data files.

3   The *cellHTS* class and reports

The basic data structure of the package is the class *cellHTS* which is a container for cell-based high-throughput RNA interference assays (data and experimental meta-data) performed in multi-plate format. This class extends the class *NChannelSet* of *Biobase* package [5].

Using the notation of the class *NChannelSet*, each group of plate results (replicates, conditions) for a given channel is called a *sample*, and data contained in such an object are from experiments where the same set of features (probes) where used. Moreover, a *cellHTS* object can contain data from multi-channel assays (see the supplement vignette accompanying this package: *"Analysis of multi-channel cell-based screens"*).

The *cellHTS* structure includes:

- *assayData*: this is an object of class *AssayData*, usually an environment containing a set of matrices of identical size. Each matrix represents a single channel. In each matrix, the rows correspond to features or reporters (e.g., siRNAs, dsRNAs) and the colums to samples (or replicates).

- *phenoData*: a dataframe (more precisely, an object of class *AnnotatedDataFrame*) containing information about the screens, namely the replicate number and the name of the biological assay. It must have the following columns in its data component: *replicate* and *assay*, where *replicate* is expected to be a vector of integers giving the replicate number, while *assay* is expected to be a character vector giving the name of the biological assay. Both of these vectors should have the same length as the number of samples in the `assayData` slot.

- *featureData*: a dataframe (more precisely, an object of class *AnnotatedDataFrame*) that contains information about the experiment, such as information about the reagents, namely: *plate*, *well*, and *controlStatus*. The latter column gives the annotation for each feature, indicating whether it is a sample or a control or is empty, etc.

  For a *cellHTS* object, the `featureData` slot must contain in its data component at least 3 mandatory columns named *plate*, *well* and *controlStatus*, which are vectors with the same length as the number of features in the experiment. Column *plate* is expected to be a numeric vector giving the plate number (e.g. 1, 2, ...), *well* should be a character vector (alphanumeric characters) giving the well ID within the

plate (e.g. *A01*, *B01*, ..., *P24*). Column *controlStatus* should be a factor specifying the annotation for each well with possible levels: *empty, other, neg, sample, pos*. Other levels may be employed for the positive and negative controls, besides *pos* and *neg*.

- *experimentData*: is an object of class *MIAME* containing descriptions of the experiment.

The *cellHTS* class includes additional slots that are used to store the input files used to assemble the *cellHTS* object. These are:

- *plateList*: a dataframe containing what was read from input measurement data files plus a column status of type character containing the string "OK" if the data import appeared to have gone well, and the respective error or warning message otherwise.

- *intensityFiles*: a list, where each component contains a copy of the imported input data files. Its length corresponds to the number of rows of *plateList*.

- *plateConf*: a dataframe containing what was read from the configuration file for the experiment (except the first two header rows).

- *screenLog*: a dataframe containing what was read from the screen log file for the experiment, in case it exists.

- *screenDesc*: a character containing what was read from the description file of the experiment.

Other slots are *batch*, *rowcol.effects*, *overall.effects* and *annotation*. For a detailed description of this class, please type ? `cellHTS`.

In Section 2, we created the object `x`, which is an instance of *cellHTS* class. The measurements intensities were stored in slot `assayData` of `x`, and this slot is overridden each time we preprocess the experimental data. The slot called `state`, which is specific for the *cellHTS* class, helps to keep track of the preprocessing state of our *cellHTS* object. This slot can be accessed via *state*, as shown below:

```
> state(x)
```

```
configured normalized     scored  annotated
    FALSE      FALSE      FALSE      FALSE
```

It contains a logical vector of length 4 representing the processing status of the object. It should have the names "configured", "normalized", "scored" and "annotated". We can thus see that x has not been configured, annotated, normalized or scored yet.

These 4 main stages are explained along this vignette and can be briefly described as follows:

Configuration involves annotating the experiment with information about the screen (e. g. title, when and how it was performed, which organism, which library, type of assay, etc.), annotating the measured values with information on controls and flagging invalid measurements. This step is covered in Section 4 and is prerequisite for preprocessing the experimental screening data (i. e. , for normalization and scoring), since for that we need to know the content of the plate wells.

Annotation this involves annotating the features with information about the assayed genes. This step is not essential for preprocessing the screening data, since assayed genes are identified primarily by the identifiers of the well and plate where the probes targeting them were placed. Neverthless, it is useful to complement (and complete it) the *cellHTS* object with annotation about the probes as detailed in Section 6, since this information can be displayed in the HTML quality reports generated by *cellHTS2* package, and can be used in further analysis (see the complete vignette, which is given as a zipped PDF file in the *doc* directory of the package, or can otherwise be manually produced from the file *cellhts2Complete.Rnw* that resides in the *scripts* directory of the package).

Normalization involves removing systematic variations, making measurements comparable across plates and within plates, enhancing the biological signal and eventually transforming the data to a scale more suitable for subsequent analyses.

Replicates scoring and summarization involves standardizing the values for each replicate and then summarizing replicate values in order to obtain a single-value per probe (for example, a robust z-score).

The steps of data normalization, replicate scoring and summarization constitute the preprocessing work-flow of the screening data, and as mentioned above, alter the contents of assayData slot and even the number of

channels of the *cellHTS* object. These steps are covered in Section 5.

The complete analysis project is contained in a set of *cellHTS* objects that reflect different preprocessing stages and that can be shared with others and stored for subsequent computational analyses.

The *cellHTS2* package offers export functions for generating human-readable reports, which consist of linked HTML pages with tables and plots. The final scored hit list is written as a tab-delimited format suitable for reading by spreadsheet programs.

Returning to our example data set, we create a report using the function *writeReport*. Since until now, we only have unnormalized experimental data, the *cellHTS* object should be given to the function as a list component named "raw":

```
> out <- writeReport(list(raw = x))
```

It will create a directory of the name given by `name(x)` in the working directory. Alternatively, the argument `outdir` can be specified to direct the output to another directory [2]

```
> out = writeReport(list(raw = x))
```

It can take a while to run this function, since it writes a large number of graphics files. After this function has finished, the index page of the report will be in the file indicated by the variable `out`,

```
> out
```

and you can view it by directing a web browser to that file.

```
> browseURL(out)
```

4 Screen configuration: annotating the plate results

The next step of the analysis involves reading and processing three input files specific of the screening experiment:

---

[2]To reduce the amount of time this vignette takes to be created, the given code for generating the quality reports will not be evaluated when calling *Sweave*. You can easily change this by editing the *cellhts2.Rnw* file.

Wells:	384	
Plates:	57	
Plate	Well	Content
*	*	sample
*	A0[1-2]	other
*	B01	neg
*	B02	pos

Table 3: Selected lines from the example plate configuration file `Plate-conf.txt`.

Plate	Sample	Well	Flag	Comment
6	1	A01	NA	Contamination
6	2	A01	NA	Contamination
6	1	A02	NA	Contamination
...	...	...	...	...

Table 4: Selected lines from the example screen log file `Screenlog.txt`.

- *Screen description file* contains a general description of the screen, its goal, the conditions under which it was performed, references, and any other information that is pertinent to the biological interpretation of the experiments.

- *Plate configuration file* is used to annotate the measured data with information on controls. The content of this file for the example data set analysed here is shown in Table 3 and the expected format for this file is explained in Section 4.1.

- *Screen log file* (optional) is used to flag individual measurements as invalid. The first 5 lines of this file are shown in Table 4, and the layout for the *screen log file* is detailed in Section 4.2.

To apply the information contained in these three files in our *cellHTS* object, we call:

```
> x <- configure(x, "Description.txt", "Plateconf.txt", "Screenlog.txt",
+     path = dataPath)
```

Note that the function *configure*[3] takes x, the result from Section 2, as an argument, and we then overwrite x with the result of this function. If no screen log file is available for the experiment, the argument `logFile` of the function *configure* should be omitted.

4.1   Format of the plate configuration file

The software expects this to be a rectangular table in a tabulator delimited text file, with mandatory columns *Plate*, *Well*, *Content*, plus two additional header lines that give the total number of wells and plates (see Table 3 for an example). The content of this file (except the two header lines) are stored in slot `plateConf` of x.

As the name suggests, the *Content* column provides the content of each well in the plate (here referred to as the *well annotation*). Mainly, this annotation falls into four categories: empty wells, wells containing genes of interest, control wells, and wells containing other things that do not fit in the previous categories. The first two types of wells should be indicated in the *Content* column of the plate configuration file by *empty* and *sample*, respectively, while the last type of wells should be indicated by *other*. The designation for the control wells in the *Content* column is more flexible. By default, the software expects them to be indicated by *pos* (for positive controls), or *neg* (for negative controls). However, other names are allowed, given that they are specified by the user whenever necessary (for example, when calling the *writeReport* function). This versatility for the control wells' annotation is justified by the fact that, sometimes, multiple positive and/or negative controls can be employed in a given screen, making it useful to give different names to the distinct controls in the *Content* column. Moreover, this versatility might be required in multi-channel screens for which we frequently have reporter-specific controls.

The *Well* column contains the name of each well of the plate in alphanumeric format (in this case, A01 to P24), while column *Plate* gives the plate number (1, 2, ...). These two columns are also allowed to contain regular expressions. In the plate configuration file, each well and plate should be covered by a rule, and in case of multiple definitions only the last one is considered. For example, in the file shown in Table 3, the rule specified by the first line after the column header indicates that all of the wells in each of the 57 assay plate contain "sample". However, a following rule indicate

---

[3]More precisely, *configure* is a method for the S4 class *cellHTS*.

that the content of wells `A01`, `A02` and `B01` and `B02` differ from "sample", containing other material (in this case, "other" and controls).

Note that the well annotations mentioned above are used by the software in the normalization, quality control, and gene selection calculations. Data from wells that are annotated as *empty* are ignored, i. e. they are set to `NA`.

Here we look at the frequency of each well annotation in the example data:

```
> table(wellAnno(x))
```

```
sample  other    neg    pos
 21660    114     57     57
```

We can also use the function *configurationAsScreenPlot* to see the original plate configuration of all of the plates as a screen plot:

```
> configurationAsScreenPlot(x)
```

A special case of well annotation is when different types of positive controls are used for the screening, that is *activator* and *inhibitor* compounds. The vignette *Analysis of two-way cell-based assays* accompanying this package explains how such screens can be handled using *cellHTS2* package.

4.1.1 Multiple plate configurations

Although it is good practice to use the same plate configuration for the whole experiment, sometimes this does not work out, and there are different parts of the experiment with different plate configurations. The use of regular expressions in columns *Plate* and *Well* of the plate configuration file allow therefore to specify different configurations within and between assay plates. The two header rows of this file ascertain that all of the plates and wells are covered in the plate configuration file.

Note that, contrarily to *cellHTS* package, in *cellHTS2* package the concept of *batch* is separated from the concept of having multiple plate configurations. So, for example, different replicate of the same plate can be set as to belong to different batches (even though they are required to have the same plate configuration!) - since readouts were performed on different days, or due to the use of different lots of reagents, etc.

Batch information (expressed in terms of integer values giving the batch number: 1, 2, ...) can go into a particular slot called `batch`. This is expected

to be a 3D-array (`Features` × `Samples` × `Channels`) of integer values giving the batch number (1, 2, ...) for each plate, sample and channel. Its first two dimensions correspond to the dimension of each matrix stored in `assayData` slot and the third dimension corresponds to the number of channels in `assayData` slot. This slot should be filled in by the user in case s/he wants to take into account this information in the analysis (for example, see *normalizePlates* function, which allows to adjust the data variance in a per-batch basis).

4.2   Format of the screen log file

The screen log file is a tabulator delimited file with mandatory columns *Plate*, *Well*, *Flag*. If there are multiple samples (replicates or conditions), a column called *Sample* should also be present; a column named *Channel* must also be provided when there are multiple channels. In addition, it can contain arbitrary optional columns. Each row corresponds to one flagged measurement, identified by the plate number (and possible sample number and channel number) and the well identifier (alphanumeric identifier). The type of flag is specified in the column *Flag*. Most commonly, this will have the value "NA", indicating that the measurement should be discarded and regarded as missing.

5   Normalization, scoring and summarization of replicates

The data normalization, replicates scoring and summarization functions available in *cellHTS2* package use the data stored in the `assayData` slot of the *cellHTS* object, saving their output to this slot, overridding its previous content. For a list of the available functions, type `?  cellHTS2`.

The preprocessing work-flow of a typical RNAi screen using *cellHTS2* package is the following:

(a) Per-plate normalization to remove plate and/or edge effects.

(b) Scoring of replicate measurements (for example, compute z-score values).

(c) Summarization of replicates (for example, take the median value).

In *cellHTS2* package, step **(a)** is performed using the function *normalizePlates*, step **(b)** is done using the function *scoreReplicates* and step **(c)** is done using *summarizeReplicates* function. In the above preprocessing workflow, we apply step **(b)** before step **(c)** so that the summary selected for replicate summarization has the same meaning independently of the type of the assay.

Note that this work-flow is suitable for single-channel data. For dual-channel data, further steps are required, as explained in the vignette *Analysis of multi-channel cell-based screens*, accompanying this package.

Below, we illustrate how these functions can be called using our example data set.

The function *normalizePlates* can be called to perform per-plate data transformation, normalization and variance adjustment. The normalization is performed in a plate-by-plate fashion, following this work-flow:

1. *Data transformation*: log transformation of the data (optional, if data are in multiplicative scale);

2. *Per-plate normalization*: per-plate normalization using the chosen method;

3. *Variance adjustment*: (optional) variance adjustment of the plate intensity corrected data.

For more details about this function and available normalization methods, please type `? normalizePlates`. To provide the means to perform the above steps, *normalizePlates* has the arguments `scale`, `log`, `method` and `varianceAdjust`.

The argument `scale` allows to define the scale of the data ("additive" or "multiplicative"), which will then control the subsequent data transformation and normalization steps. Namely, when data are in multiplicative scale, the function allows to perform $\log_2$ data transformation. For that we need to set the function's argument `log` to `TRUE`. Log transformation will then change the scale of the data to be "additive".

In the next step of preprocessing, intensities are corrected in a plate-by-plate basis using the chosen normalization method. Per-plate median normalization is one of the methods available in *normalizePlates*, and can be chosen by setting the argument `method="median"`. In this case, plate effects are corrected by dividing (if the current scale of the data is multiplicative) each measurement by the median value across wells annotated as sample, for each plate and replicate. If data are in additive scale, the per-plate

14

median values are subtracted from each plate measurement instead. All of the available normalization methods are described in Appendix 8.

In the final preprocessing step, variance of plate-corrected intensities can be adjusted according to argument `varianceAdjust`, as follows:

- `varianceAdjust="byPlate"`: per plate normalized intensities are divided by the per-plate median absolute deviations (MAD) in "sample" wells. This is done separately for each replicate and channel;

- `varianceAdjust="byBatch"`: using the content of slot `batch`, plates are split according to assay batches and the individual normalized intensities in each group of plates (batch) are divided by the per-"batch of plates" MAD values (calculated based on "sample" wells). This is done separately for each replicate and channel;

- `varianceAdjust="byExperiment"`: each normalized measurement is divided by the overall MAD of normalized values in wells containing "sample". This is done separately for each replicate and channel.

If `varianceAdjust="none"`, no variance adjustment is performed (default).

As explained above, the parameter `method` of *normalizePlates* allows to choose between different types of per-plate normalization methods. Returning to our example data set, we choose to apply *plate median scaling*:

$$
\begin{aligned}
x'_{ki} &= \frac{x_{ki}}{M_i} \qquad \forall k, i & (1) \\
M_i &= \underset{m \in \text{samples}}{\text{median}} \ x_{mi} & (2)
\end{aligned}
$$

where x_{ki} is the raw intensity for the k-th well in the i-th replicate file, and x'_{ki} is the corresponding normalized intensity. The median is calculated across the wells annotated as *sample* in the i-th result file. This is achieved by calling:

```
> xn <- normalizePlates(x, scale = "multiplicative", log = FALSE,
+     method = "median", varianceAdjust = "none")
```

after which we obtain a *cellHTS* object with the normalized intensities stored in the slot `assayData`. In the previous call to *normalizePlates* function, we have chosen not to adjust the data variance (default behaviour of *normalizePlates*). For example, we can use function *compare2cellHTS* provided with the package to check whether these two *cellHTS* objects, x and xn belong to the same experiment:

```
> compare2cellHTS(x, xn)
```

```
[1] TRUE
```

After normalizing the data, we standardize the values for each replicate experiment using Equation (3). In this equation, $\hat{\mu}$ and $\hat{\sigma}$ are estimators of location and scale of the distribution of x'_{ki} taken across all plates and wells of a given replicate experiment. This function uses robust estimators, namely, median and median absolute deviation (MAD). Moreover, it only considers the wells containing "sample" for estimating $\hat{\mu}$ and $\hat{\sigma}$. The symbol $\pm$ indicates that we allow for either plus or minus sign in Equation (3); the minus sign can be useful in the application to an inhibitor assay, where an effect results in a decrease of the signal and we may want to see this represented by a large score. This is done by calling the *scoreReplicates* function, where arguments `sign` and `method` define the sign and the scoring method to apply (robust z-scores, in this case), respectively:

```
> xsc <- scoreReplicates(xn, sign = "-", method = "zscore")
```

After data standardization, we summarize the replicates, calculating a single score for each gene. This is performed using the *summarizeReplicates* function, where we use its argument `summary` to select the summary to apply. One option is *rms*, which corresponds to take the root mean square of the replicates values, and is shown in Equation (4). The chosen summary is taken over all the n_{rep_k} replicates of probe k.

$$z_{ki} = \pm\frac{x'_{ki} - \hat{\mu}}{\hat{\sigma}} \tag{3}$$

$$z_k = \sqrt{\frac{1}{n_{\mathrm{rep}_k}}\sum_{r=1}^{n_{\mathrm{rep}_k}} z_{kr}^2}. \tag{4}$$

Depending on the intended stringency of the analysis, other plausible choices of summary function between replicates are the minimum, the maximum, the mean or the median. In the first case, the analysis would be particularly conservative: all replicate values have to be high in order for z_k to be high. For the cases where both sides of the distribution of z-score values are of interest, alternative summary options for the replicates are to select the value closest to zero (conservative approach) by setting `summary="closestToZero"` or the value furthest from zero (`summary="furthestFromZero"`). In order to compare our results with those obtained in the paper of Boutros *et al.* [2], we choose to consider the mean as a summary:
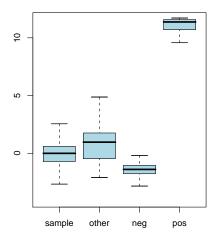
Figure 1: Boxplots of z-scores for the different types of probes.

```
> xsc <- summarizeReplicates(xsc, summary = "mean")
```

after which we obtain a *cellHTS* object with the resulting single z-score value per probe stored in `assayData` slot.

Boxplots of the z-scores for the different types of probes are shown in Figure 1.

```
> scores <- Data(xsc)
> ylim <- quantile(scores, c(0.001, 0.999), na.rm = TRUE)
> boxplot(scores ~ wellAnno(x), col = "lightblue", outline = FALSE,
+       ylim = ylim)
```

In the *cellHTS2* package, we provide a further transformation of the z-score values to obtain the so-called *calls*. This involves applying a sigmoidal transformation to the z-score values, with parameters z_0 and λ (> 0), given by:

$$y_k = \frac{1}{1 + e^{-\lambda(z_k - z_0)}} \tag{5}$$

This transformation maps the z-score values to the interval $[0, 1]$ and is intended to expand the scale of z-scores with intermediate values and shrink

Plate	Well	HFAID	GeneID
1	A03	HFA00274	CG11371
1	A04	HFA00646	CG31671
1	A05	HFA00307	CG11376
1	A06	HFA00324	CG11723
...	...	...	...

Table 5: Selected lines from the example gene ID file `GeneIDs_Dm_HFA_1.1.txt`.

the ones showing extreme values, therefore making the difference between intermediate phenotypes larger. The parameter z_0 defines the centre of the sigmoidal transformation, while λ controls the smoothness of the transformation.

This transformation can be done by calling the function *scores2calls*, as shown below:

```
> y <- scores2calls(xsc, z0 = 1.5, lambda = 2)
> plot(Data(xsc), Data(y), col = "blue", xlab = "z-scores", ylab = "calls",
+     main = expression(1/(1 + e^{
+         -lambda * (z - z[0])
+     })))
```

However, for the purpose of the present analysis, we will consider the z-score values instead of the call values.

6 Probe annotation

Up to now, the assayed genes have been identified solely by the identifiers of the plate and the well that contains the probe for them. The *annotation file* contains additional annotation, such as the probe sequence, references to the probe sequence in public databases, the gene name, gene ontology annotation, and so forth. Mandatory columns of the annotation file are *Plate*, *Well*, and *GeneID*, and it has one row for each well. The content of the *GeneID* column will be species- or project-specific. The first 5 lines of the example file are shown in Table 5, where we have associated each probe with CG-identifiers for the genes of *Drosophila melanogaster*.

```
> xsc <- annotate(xsc, geneIDFile = "GeneIDs_Dm_HFA_1.1.txt", path = dataPath)
```

An optional column named *GeneSymbol* can be included in the *annotation file*, and its content will be displayed by the tooltips added to the plate plots and screen-wide plot in the HTML quality report (see Section 7).

7   Report

We have now completed the analysis tasks: the dataset has been read, configured, normalized, scored, and annotated:

```
> xsc

cellHTS (storageMode: lockedEnvironment)
assayData: 21888 features, 1 samples
  element names: score
phenoData
  sampleNames: 1
  varLabels and varMetadata description:
    replicate: Replicate number
    assay: Biological assay
  additional varMetadata: channel
featureData
  featureNames: 1, 2, ..., 21888  (21888 total)
  fvarLabels and fvarMetadata description:
    plate: Plate number
    well: Well ID
    ...: ...
    GeneID: GeneID
    (5 total)
experimentData: use 'experimentData(object)'
state:  configured = TRUE
        normalized = TRUE
        scored = TRUE
        annotated = TRUE
Number of plates: 57
Plate dimension: nrow = 16, ncol = 24
Number of batches:
Well annotation: sample other neg pos
  pubMedIds: 14764878
Annotation:
```

We can now save the scored data set to a file.

19

```
> save(xsc, file = paste(experimentName, ".rda", sep = ""), compress = TRUE)
```

The data set can be loaded again for subsequent analysis, or passed on to others. To produce a comprehensive report, we can call the function *writeReport* again, this time specifying in the functions argument `cellHTSlist` the three *cellHTS* objects: "raw", "normalized" and "scored" *cellHTS* objects.

```
> out <- writeReport(cellHTSlist = list(raw = x, normalized = xn,
+     scored = xsc), force = TRUE, plotPlateArgs = list(xrange = c(0.5,
+     3)), imageScreenArgs = list(zrange = c(-4, 8), ar = 1), map = TRUE)
```

and use a web browser to view the resulting report.

```
> browseURL(out)
```

The report contains a quality report for each plate, and also for the whole screening assays. The per-plate HTML reports display the scatterplot between duplicated plate measurements, the histogram of the normalized signal intensities for each replicate, and plate plots representing, in a false color scale, the normalized values of each replicate, and the standard deviation between replicate measurements at each plate position. It also reportes different measures of agreement between replicate measurements, such as the repeatability standard deviation between replicate plates and the Spearman correlation coefficient between duplicates. The per-plate reports also show the dynamic range, calculated as the ratio between the geometric means of the positive and negative controls. These measures can also be obtained independently from *writeReport* function, by using the functions *getMeasureRepAgreement* and *getDynamicRange* provided in *cellHTS2* package. If different positive controls were specified at the configuration step and when calling *writeReport*, the dynamic range is calculated separately for the distinct positive controls, since different positive controls might have different potencies.

The experiment-wide HTML report presents, for each replicate, the boxplots with raw and normalized intensities for the different plates, and two plots for the controls: one showing the signal from positive and negative controls at each plate, and another plot displaying the distribution of the signal from positive and negative controls, obtained from kernel density estimates. The latter plot further gives the Z'-factor determined for each experiment (replicate) using the negative controls and each different type of positive controls [9], as a measure to quantify the distance between their distributions. This measure can also be obtained by calling the function *getZfactor*.

The experiment-wide report also shows a screen-wide plot with the *z*-scores in every well position of each plate. If the argument map of *writeReport* function is set to TRUE, this plot and the plate plots of the per-plate reports contain tooltips (information pop-up boxes) dispaying the annotation information at each position within the plates. If the scored *cellHTS* object provided for *writeReport* is not annotated with gene identifiers, the annotation information shown by the tooltips is simply the well identifiers. For an annotated *cellHTS* object, if an optional column called *GeneSymbol* was included in the *annotation file* (see Section 6), and therefore is present in featureData slot of the annotated object, then its content is used for the tooltips. Otherwise, the content of column "GeneID" of the featureData slot (which can be accessed via geneAnno) is considered.

The screen-wide image plot can also be produced separately using the function *imageScreen* given in the *cellHTS2* package. This might be useful if we want to select the best display for our data, namely, the aspect ratio for the plot and/or the range of *z*-score values to be mapped into the color scale. These can be passed to the function's arguments ar and zrange, respectively. For example,

```
> imageScreen(xsc, ar = 1, zrange = c(-3, 4))
```

It should be noted that the per-plate and per-experiment quality reports are constructed based on the normalized *cellHTS* object, in case it is provided to *writeReport* function. Otherwise, it uses the data of the raw *cellHTS* object. The quality report produced by *writeReport* function has also a link to a file called *topTable.txt* that contains the list of scored probes ordered by decreasing *z*-score values, when the final scored *cellHTS* object is provided. This file has one row for each well and plate, and for the present example data set, it has the following columns:

- plate plate identifier for each feature;

- position gives the position of the well in the plate (runs from 1 to the total number of wells in the plate);

- well gives the alphanumeric well identifier for each feature;

- score corresponds to the summarized score calculated for each probe (data stored in the scored and summarized object xsc);

- wellAnno corresponds to the well annotation (as given by the plate configuration file);

- `finalWellAnno` gives the final well annotation for the scored values. It combines the information given in the plate configuration file with the values in `assayData` slot of the scored *cellHTS* object, in order to have into account the wells that have been flagged either by the screen log file, or manually by the user during the analysis. These flagged wells appear with the annotation *flagged*.

- `raw_r1_ch1` and `raw_r2_ch1` contain the raw intensities for replicate 1 and replicate 2, respectively (data stored in the unnormalized *cellHTS* object `x`). 'ch' refers to channel;

- `median_ch1` corresponds to the median of raw measurements across replicates;

- `diff_ch1` gives the difference between replicated raw measurements (only given if the number of replicates is equal to two);

- `average_ch1` corresponds to the average between replicate raw intensities (only given if the number of replicates is higher than two);

- `raw/PlateMedian_r1_ch1` and `raw/PlateMedian_r2_ch1` give the ratio between each raw measurement and the median intensity in each plate for replicate 1 and replicate 2, respectively. The plate median is determined for the raw intensities, using exclusively the wells annotated as "sample".

- `normalized_r1_ch1` and `normalized_r2_ch1` give the normalized intensities for replicate 1 and replicate 2, respectively. This corresponds to the data stored in the normalized *cellHTS* object `xn`.

Additionally, if any of the *cellHTS* objects provided in the argument `cellHTSlist` to *writeReport* has been annotated (as in the present case), it also contains the data given in the content of `featureData` slot of the annotated object. The above file with the list of scored probes can also be obtained without the need to run *writeReport* by using the function *getTopTable* provided in the package.

7.1 Exporting data to a tab-delimited file

The *cellHTS2* package contains a function called *writeTab* to save the data stored in `assayData` slot of a *cellHTS* object to a tab-delimited file. The rows of the file are sorted by plate and well, and there is one row for each

plate and well. When the *cellHTS* object is annotated, the probe information (i.e. the probe identifiers stored in column "GeneID" of the `featureData` slot) is also added.

```
> writeTab(xsc, file = "Scores.txt")
```

Since you might be interestered in saving other values to a tab delimited file, below we demonstrate how you can create a matrix with the ratio between each raw measurement and the plate median, together with the gene and well annotation, and export it to a tab-delimited file using the function *write.tabdel* [4] also provided in the *cellHTS2* package.

```
> y <- array(as.numeric(NA), dim = dim(Data(x)))
> nrWell = prod(pdim(x))
> nrPlate = max(plate(x))
> for (p in 1:nrPlate) {
+     j <- (1:nrWell) + nrWell * (p - 1)
+     samples <- wellAnno(x)[j] == "sample"
+     y[j, , ] <- apply(Data(x)[j, , , drop = FALSE], 2:3, function(w) w/median(w[sam
+         na.rm = TRUE))
+ }
> y = signif(y, 3)
> out <- y[, , 1]
> out <- cbind(fData(xsc), out)
> names(out) = c(names(fData(xsc)), sprintf("Well/Median_r%d_ch%d",
+     rep(1:dim(y)[2], dim(y)[3]), rep(1:dim(y)[3], each = dim(y)[2])))
> write.tabdel(out, file = "WellMedianRatio.txt")
```

At this point we are finished with the basic analysis of the screen. As one example for how one could continue to further mine the screen results for biologically relevant patterns, we demonstrate an application of category analysis in the complete vignette, which is given as a zipped PDF file in the *doc* directory of the package, or can otherwise be manually produced from the file *cellhts2Complete.Rnw* that resides in the *scripts* directory of the package.

---

[4]This function is a wrapper of the function *write.table*, whereby you just need to specify the name of the data object and the file

8 Appendix: Normalization methods implemented in *cellHTS2* package

There are two main normalization methods available with *cellHTS2* package: methods based on the use of reference controls, and distribution-based methods. These methods can be applied using the function *normalizePlates*.

8.1 Controls-based normalization

8.1.1 Percent of control

Percent of control (POC) is a preprocessing method that tries to correct for plate-to-plate variability by normalizing each kth compound raw measurements in the ith result file, x_{ki}, relative to the average of within-plate controls. In an antagonist (or inhibition) type assay, it is defined as:

$$x_{ki}^{\text{POC}} = \frac{x_{ki}}{\mu_i^{\text{pos}}} \times 100 \tag{6}$$

where μ_i^{pos} is the average of the measurements on the positive controls in the ith result file (i. e., for a given plate and replicate).

In *cellHTS2* package, this method can be applied by setting the argument `method="POC"` when calling *normalizePlates* function.

We also provide in the package a normalization method for *normalizePlates* (`method="negatives"`) that consists of scaling the plate measurements by the per-plate median of the intensities on the negative controls [5].

8.1.2 Normalized percent inhibition

If *normalizePlates* is called with `method="NPI"`, the method known as normalized percent inhibition (NPI) is applied in a per-plate basis to correct for plate effects. For an antagonist assay, this method divides the difference between each measurement in a given result file i (x_{ki}) and the average of the positive controls on that plate (μ_i^{pos}) by the difference between the averages of the measurements on the positive (μ_i^{pos}) and the negative controls (μ_i^{neg}):

$$x_{ki}^{\text{NPI}} = \frac{\mu_i^{\text{pos}} - x_{ki}}{\mu_i^{\text{pos}} - \mu_i^{\text{neg}}} \tag{7}$$

---

[5]If the scale of the data is defined as being additive (i. e., argument `scale="additive"`, or arguments `scale="multiplicate"` and `log=TRUE`), measurements are subtracted by the median of per-plate negative controls instead.

8.2 Non-controls-based normalization

There are several normalization method implemented in *cellHTS2* package that make use of the overall distribution of values, instead of relying exclusively on controls. These are described in the following sections.

8.2.1 Z score method

Z score is a simple and widely known normalizing method that is performed in a per-plate basis as follows:

$$x_{ki}^{\mathrm{Z}} = \frac{x_{ki} - \mu_i}{\sigma_i}, \tag{8}$$

where μ_i and σ_i are the mean and standard deviation, respectively, of all measurements within the ith result file (replicated plate). In the Z score method, measurements are re-scaled relative to within-plate variation by subtracting the average of the plate values and dividing this difference by the standard deviation estimated from all measurements of the plate.

In *cellHTS2*, we consider a robust version of this method, where the mean and standard deviation are replaced by the median and the MAD, respectively, calculated at the sample wells. This robust Z score method is performed by calling *normalizePlates* as follows:

```
> xZ <- normalizePlates(x, scale = "additive", log = FALSE, method = "median",
+      varianceAdjust = "byPlate")
```

8.2.2 Plate median normalization

Plate median normalization involves calculating the relative signal of each well compared to the median of the sample wells in the plate, as shown in Equation (1) and Equation (2). The median is calculated among the m wells containing *sample* (i. e. , for wells that contain genes of interest) in result file i. Plate median normalization can be chosen by setting `method="median"` in *normalizePlates*. When applied to data on additive scale, the plate median normalization involves subtracting the plate measuments by the per-plate median instead.

We also have two variants of the plate median scaling which consist of using as the per-plate scaling factor M_i the per-plate average intensity on sample wells (`method="mean"`) or the midpoint of the shorth of the per-plate distribution of values on sample wells (`method="shorth"`).

The next methods are intended to explicitly correct for *spatial effects* within plates, i. e. , the presence of intensity gradients within the plates. Such signal gradients can be caused by differences in temperature, incubation time or concentration, etc., in different wells across a plate. Typically, these gradients produce repeatitive patterns, which make it possible to distinguish them from real actives that should be more or less randomly dispersed for quasi-randomized collections of compounds.

8.2.3 B score method

In the B score method, row and column biases within each plate are explicitly corrected for by fitting a two-way median polish to the raw data in a per-plate fashion [3]:

$$e_{rci} = x_{rci} - \hat{x}_{rci} = x_{rci} - \left(\hat{\mu}_i + \hat{R}_{ri} + \hat{C}_{ci} \right) \tag{9}$$

$$x^{\mathrm{B}}_{rci} = \frac{e_{rci}}{MAD_i} \tag{10}$$

Here, x_{rci} is the measurement value in the rth row and cth column of the plate corresponding to the ith result file, $\hat{x}_{rci}$ is the corresponding fitted value defined as the sum between the estimated average of the replicate plate ($\hat{\mu}_i$), the estimated systematic offset for row r ($\hat{R}_{ri}$) and the systematic offset for column c ($\hat{C}_{ci}$) in that replicated plate. In a second step, each of the obtained residual values e_{rci}'s of the ith result file are divided by their median absolute deviation (MAD_i) giving the final B score value – Equation (10).

We implemented a method similar to the B score method described in Malo *et al.* [7] and Brideau *et al.* [3] using the Tukey's median polish procedure [8] (function *medpolish* of the package *stats*) which fits an additive model to the data according to Equation (10). The Tukey's median polish algorithm works by alternately removing the row and column medians and continues until the proportional reduction in the sum of absolute residuals is less than ϵ or until the maximum number of iterations has been reached. The B score method can be applied by defining argument `method="Bscore"` in *normalizePlates*. Alternatively, the method can be applied by calling a separate function called *Bscore* provided in the *cellHTS2* package.

In *cellHTS2* package, we provide two additional spatial normalization methods that fit a polynomial surface to the intensities within each assay plate using local regression and that can be performed via *normalizePlates* or *spatialNormalization* functions, although we advise to apply these methods

using the former function. The fit can be performed either using the *loess* procedure or the *locfit.robust* function of package *locfit*. In *normalizePlates*, if `method="locfit"`, spatial effects are removed by fitting a bivariate local regression to each plate and replicate, while if `method="loess"`, a loess curve is fitted instead.

9   Appendix: Data transformation

An obvious question is whether to do the statistical analyses on the original intensity scale or on a transformed scale such as the logarithmic one. Many statistical analysis methods, as well as visualizations work better if (to sufficient approximation)

- replicate values are normally distributed,

- the data are evenly distributed along their dynamic range,

- the variance is homogeneous along the dynamic range [6].

Figure 2 compares these properties for untransformed and log-transformed normalized data, showing that the difference is small. Intuitively, this can be explained by the fact that for small x,

$$\log(1 + x) \approx x$$

and that indeed the range of the untransformed data is mostly not far from 1. Hence, for the data examined here, the choice between original scale and logarithmic scale is one of taste, rather than necessity.

```
> library("vsn")
> par(mfcol = c(3, 2))
> myPlots = function(z, ...) {
+     hist(z[, 1], 100, col = "lightblue", xlab = "", ...)
+     meanSdPlot(z, ylim = c(0, quantile(abs(z[, 2] - z[, 1]),
+         0.95, na.rm = TRUE)), ...)
+     qqnorm(z[, 1], pch = ".", ...)
+     qqline(z[, 1], col = "blue")
+ }
> dv = Data(xn)[, , 1]
> myPlots(dv, main = "untransformed")
> xlog = normalizePlates(x, scale = "multiplicative", log = TRUE,
+     method = "median", varianceAdjust = "byExperiment")
```
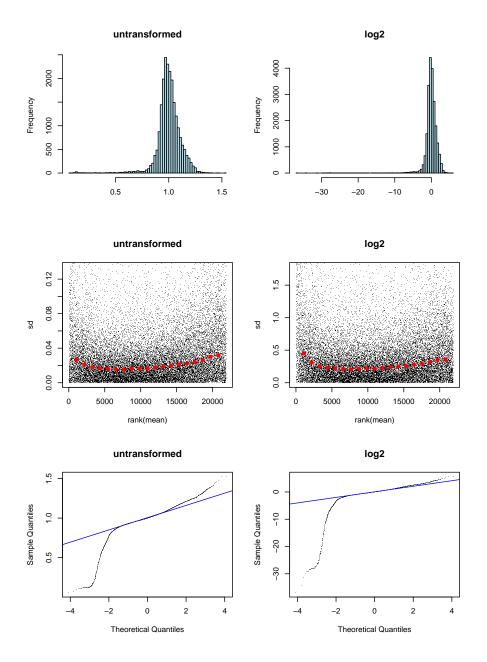
Figure 2: Comparison between untransformed (left) and logarithmically (base 2) transformed (right), normalized data. Upper: histogram of intensity values of replicate 1. Middle: scatterplots of standard deviation versus mean of the two replicates. Bottom: Normal quantile-quantile plots.

```
> dvlog = Data(xlog)[, , 1]
> myPlots(dvlog, main = "log2")
```

References

[1] M Boutros, LP Brás, and W Huber. Analysis of cell-based RNAi screens. *Genome Biology*, 7:R66, 2006. 2

[2] Michael Boutros, Amy A Kiger, Susan Armknecht, Kim Kerr, Marc Hild, Britta Koch, Stefan A Haas, Heidelberg Fly Array Consortium, Renato Paro, and Norbert Perrimon. Genome-wide RNAi analysis of growth and viability in Drosophila cells. *Science*, 303(5659):832–835, 2004. 3, 16

[3] C Brideau, B Gunter, B Pikounis, and A Liaw. Improved statistical methods for hit selection in high-throughput screening. *J. Biomol. Screen*, 8:634–647, 2003. 26

[4] Robert Gentleman. Reproducible research: A bioinformatics case study. *Statistical Applications in Genetics and Molecular Biology*, 3, 2004. 2

[5] Robert C Gentleman, Vincent J. Carey, Douglas M. Bates, et al. Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biology*, 5:R80, 2004. 6

[6] Wolfgang Huber, Anja von Heydebreck, Holger Sültmann, Annemarie Poustka, and Martin Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18 Suppl. 1:S96–S104, 2002. 27

[7] Nathalie Malo, James Hanley, Sonia Cerquozzi, Jerry Pelletier, and Robert Nadon. Statistical practice in high-throughput screening data analysis. *Nature Biotechnology*, 24(2):167–175, 2006. 26

[8] J.W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, Cambridge, MA, 1977. 26

[9] JH Zhang, TD Chung, and KR Oldenburg. A Simple Statistical Parameter for Use in Evaluation and Validation of High Throughput Screening Assays. *J Biomol Screen*, 4(2):67–73, 1999. 20