Analysis of two-way cell-based assays

Lígia Brás, Michael Boutros and Wolfgang Huber

May 3, 2016

Contents

1 Introduction

This technical report is a supplement of the main vignette *End-to-end analysis of cell-based screens: from raw intensity readings to the annotated hit list* that is given as part of the *cellHTS* package, and that accompanies the paper *Analysis of cell-based RNAi screens* by Michael Boutros, Lígia Brás and Wolfgang Huber [1].

The report explains how the *cellHTS* package can be used to document and analyse two-way assays. We call *two-way assays* to the assays where we can have two types of effectors: *activators* and *inhibitors*. In such cases, both type of effectors are used as positive controls, and they need to be considering indivually by the *cellHTS* package.

This text has been produced as a reproducible document [2], containing the actual computer instructions, given in the language R, to produce all results, including the figures and tables that are depicted here. 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 *cellHTS* and of some

Filename	Plate	Replicate	Channel
DHA001LU1.CSV	1	1	1
...	...	...	...

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

other add-on packages. Then, you can simply take the file *twoWay.Rnw* in the *doc* directory of the package, open it in a text editor, run it using the R command *Sweave*, and modify it according to your needs.

We start by loading the package.

```
> library("cellHTS")
```

2   Assembling the data

The example data set corresponds to one 96-well plate of a two-way assay performed with human cells. The screen was performed in duplicate, and a luminescence-reporter was used in the assay.

2.1   Reading the raw intensity files

The set of available result files and the information about them (which plate, which replicate) is given in the *plate list file*. The first line of the plate list file for this data set is shown in Table 1.

The path for the raw data is defined below:

```
> experimentName = "TwoWayAssay"
> dataPath=system.file(experimentName, package="cellHTS")
```

The input files are in the `TwoWayAssay` directory of the *cellHTS* package.

The function *readPlateData* of *cellHTS* package reads the plate list file, and the intensity files, assembling the data into a single R object. In this example, the raw intensity files correspond to csv files, in which the first part consists of data from the acquisition instrument, while the relevant data values are in the 12×8 matrix at the end of the file. So, we need to give as argument to *readPlateData*, a function suitable to import such data file format. This function, which we named *importData*, is given in the same directory as the data files, in a R file with the same name, and needs to be sourced:

```
> source(file.path(dataPath, "importData.R"))
```

Batch	Well	Content	Comment
1	A01	mock	mock
1	A02	mock	mock
1	A03	sample	NA
1	A04	sample	NA
...	...	...	...

Table 2: Selected lines from the example plate configuration file `Plate-conf.txt`.

Plate	Well	GeneID	GeneSymbol
1	A03	22848	AAK1
1	A04	9625	AATK
1	A05	64781	CERK
1	A06	11069	RAPGEF4
...	...	...	...

Table 3: Selected lines from the example gene ID file `GeneIDs.txt`.

Finally, we call *readPlateData*:

```
> x <- readPlateData("Platelist.txt",
+   importFun=importData, name=experimentName, path=dataPath)

> x

cellHTS object of name 'TwoWayAssay'
1 plates with 96 wells, 2 replicates, 1 channel. State:
configured normalized     scored  annotated
     FALSE      FALSE      FALSE     FALSE
```

2.2 Configuring and annotating the *cellHTS* object

The *plate configuration file* gives the information about the content of the plate wells, which is used by the software to annotate the measured data. The first lines of this file are shown in Table 2. For the sample data, there is no *screen log file*, meaning that no measurements were marked to be flagged during the assay. Another file necessary for the configuration step of the `cellHTS` object is the *screen description file*, which gives a general description of the screen.

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

In this data set, instead of using the default names *pos* and *neg* for positive and negative controls, respectively, we have used the names of the target genes. Thus, the well annotation in this sample plate looks like this [1]:

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

```
  mock sample    gfp  empty   aatk map2k6   attk
     6     80      2      4      1      2      1
```

Here, *AATK*, *ATTK* and *MAP2K6* are positive controls, whereas *GFP* and *mock* (mock transfection) correspond to negative controls. We will return to this issue in Section 3, but now, we will proceed to the next step in the data assemble stage, whereby we add the annotation information to x. This information is in the *screen annotation file*, and Table 3 shows the first 5 lines of this file. Besides the mandatory columns *Plate*, *Well* and *GeneID*, the file contains the optional column *GeneSymbol*, whose content will be used for the tooltips display in the HTML quality reports that will be produced later on.

```
> x <- annotate(x, geneIDFile="GeneIDs.txt", path=dataPath)
```

3 Data preprocessing and summarization of replicates

We can take a first look at the data by constructing the HTML quality reports using the *writeReport* function. For that, we must define the positive and negative controls needed when calling this function.

For a 2-way assay, the argument `posControls` of *writeReport* should be defined as a list with two components: *act* and *inh*, which should be defined as vectors of regular expressions with the same length as the number of channels (in this case, length one). These arguments will be passed to the *regexpr* function for pattern matching within the well annotation given

---

[1] Note that when reading the plate configuration file, the *configure* function puts the content of the column *Content* in lowercase. However, the orignal content of this column (and the complete plate configuration file) is stored in x$plateConf, so if you prefer, you can use the command `table(x$plateConf$Content)` instead of `table(x$wellAnno)`.

in `x$wellAnno`.

In the example presented here, in the positive controls, we have *AATK* and *ATTK* as activator effectors, and *MAP2K6* as an inhibitor effector. The negative controls are *GFP* and the mock transfection (indicated as *mock* in the plate configuration file). Thus, we define the negative and positive controls as follows:

```
> negCtr <- "(?i)^GFP$|^mock$"
> posCtr <- list(act = "(?i)^AATK$|^ATTK$", inh = "(?i)^MAP2K6$")
```

Finally, we construct the quality report pages for the raw data in a directory called `2Wraw`, in the working directory:

```
> out <- writeReport(x, outdir="2Wraw",
+     posControls=posCtr, negControls=negCtr,
+     plotPlateArgs=list(xrange=c(300, 4000)))
```

After this function has finished, we can view the index page of the report:

```
> browseURL(out)
```

As can be seen in the HTML reports, the dynamic range for each replicate (and the respective average) was calculated separately for the activators and inhibitors.

Next, we normalize the data using a simple approach. For each replicate, the plate intensity values at the negative control wells are taken as a correction factor, so that each plate measurement is divided by it. Then, the obtained values are $\log_2$ transformed.

```
> x <- normalizePlates(x, normalizationMethod ="negatives",
+     negControls = negCtr, transform=log2)
```

The normalized intensities are stored in the slot `x$xnorm`, in an array of the same size as `x$xraw`.

Below, we call the function *summarizeReplicates* to determine the *z*-score values for each replicate, and then summarize the replicated *z*-score values by taking the average. A more conservative approach would be to consider the *z*-score value closest to zero between replicates (**summary**='closestToZero').

```
> x <- summarizeReplicates(x, zscore="+", summary="mean")
```
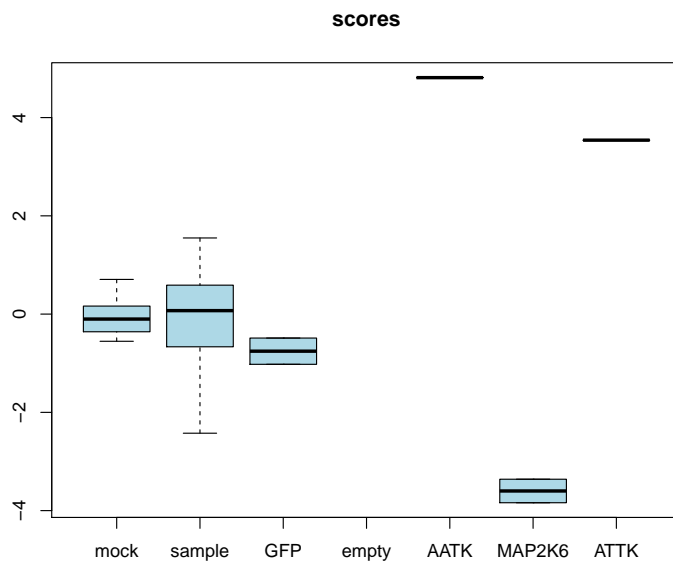
Figure 1: Boxplots of z-scores for the different types of probes.

The resulting single z-score value per probe is stored in the slot `x$score`.
Figure 1 shows the boxplots of the z-scores for the different types of probes.

```
> ylim <- quantile(x$score, c(0.001, 0.999), na.rm=TRUE)
> boxplot(x$score ~ x$wellAnno, col="lightblue", main="scores", outline=FALSE, ylim=y
> lab <- unique(x$plateConf$Content)
> lab <- lab[match(levels(x$wellAnno), tolower(lab))]
> axis(1, at=c(1:nlevels(x$wellAnno)), labels=lab)
```

Now that the data have been preprocessed and scored, we call again *writeRe-port* and use a web browser to view the resulting report:

```
> out <- writeReport(x, outdir="2Wnormalized", posControls=posCtr, negControls=negCtr
+    plotPlateArgs=list(xrange=c(-1,1)), imageScreenArgs=list(zrange=c(-2,3)))

> browseURL(out)
```

The quality reports have been created in the folder `2Wnormalized` in the working directory. Now the report shows also controls-related plots, distinguishing the two types of positive controls (activators and inhibitors).

Furthermore, the report shows the image plot with the final scored values for the whole data set. Finally, we save the data set to a file.

```
> save(x, file=paste(experimentName, ".rda", sep=""))
```

4 Session info

This document was produced using:

```
> toLatex(sessionInfo())
```

- R version 3.3.0 (2016-05-03), `x86_64-pc-linux-gnu`

- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`,
  `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`,
  `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`,
  `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`,
  `LC_IDENTIFICATION=C`

- Base packages: base, datasets, grDevices, graphics, grid, methods,
  parallel, stats, utils

- Other packages: Biobase 2.32.0, BiocGenerics 0.18.0, cellHTS 1.42.0,
  hexbin 1.27.1, vsn 3.40.0

- Loaded via a namespace (and not attached): AnnotationDbi 1.34.0,
  BiocInstaller 1.22.0, DBI 0.4, DEoptimR 1.0-4, IRanges 2.6.0,
  KernSmooth 2.23-15, MASS 7.3-45, Matrix 1.2-6,
  RColorBrewer 1.1-2, RSQLite 1.0.0, Rcpp 0.12.4.5, S4Vectors 0.10.0,
  XML 3.98-1.4, affy 1.50.0, affyio 1.42.0, annotate 1.50.0, cluster 2.0.4,
  colorspace 1.2-6, digest 0.6.9, genefilter 1.54.0, ggplot2 2.1.0,
  gridBase 0.4-7, gtable 0.2.0, labeling 0.3, lattice 0.20-33,
  limma 3.28.0, munsell 0.4.3, mvtnorm 1.0-5, pcaPP 1.9-60, plyr 1.8.3,
  prada 1.48.0, preprocessCore 1.34.0, robustbase 0.92-5, rrcov 1.3-11,
  scales 0.4.0, splines 3.3.0, stats4 3.3.0, survival 2.39-2, tools 3.3.0,
  xtable 1.8-2, zlibbioc 1.18.0

References

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

[2] Robert Gentleman. Reproducible research: A bioinformatics case study. *Statistical Applications in Genetics and Molecular Biology*, 3, 2004. 1