Analysis of screens with enhancer and suppressor controls

Lígia Brás, Michael Boutros and Wolfgang Huber

October 30, 2018

Contents

1 Introduction **1**

2 Assembling the data **2**
2.1 Reading the raw intensity files 2
2.2 Configuring and annotating the *cellHTS* object 3

3 Data preprocessing and summarization of replicates **5**

4 Session info **8**

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 *cellHTS2* package.

The report explains how the *cellHTS2* package can be used to document and analyse enhancer-suppressor screens (or *two-way assays*), where we can have two types of effectors: *activators* (or enhancers) and *inhibitors* (or repressors). In such cases, both type of effectors are used as positive controls, and they need to be considered indivually by the *cellHTS2* package.

This text has been produced as a reproducible document [1], 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 *cellHTS2* and of some other add-on packages. Then, you can simply take the file *twoWay.Rnw*

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

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

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("cellHTS2")
```

2 Assembling the data

The example data set corresponds to one 96-well plate of an enhancer-suppressor screen 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="cellHTS2")
```

The input files are in the `TwoWayAssay` directory of the *cellHTS2* package.

The function *readPlateList* of *cellHTS2* 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 *readPlateList*, 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"))
```

Finally, we call *readPlateList*:

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

> x

cellHTS (storageMode: lockedEnvironment)
assayData: 96 features, 2 samples
  element names: Channel 1
phenoData
  sampleNames: 1 2
  varLabels: replicate assay
  varMetadata: labelDescription channel
featureData
  featureNames: 1 2 ... 96 (96 total)
  fvarLabels: plate well controlStatus
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
state:  configured = FALSE
        normalized = FALSE
        scored = FALSE
        annotated = FALSE
Number of plates: 1
Plate dimension: nrow = 8, ncol = 12
Number of batches: 1
```

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 content of this file is 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, descripFile = "Description.txt",
+                confFile="Plateconf.txt", path=dataPath)
```

Wells:	96		
Plates:	1		
Plate	Well	Content	Comment
*	*	sample	NA
*	[A,H]0[1-2]|D02|E01	mock	mock
*	B01|G02	GFP	GFP
*	B02|G01	empty	DDIT3
*	C01|F02	AATK	up
*	C02|F01	MAP2K6	down
*	D01|E02	empty	CD14

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`.

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

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

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

Here, *AATK*, *ATTK* and *MAP2K6* are positive controls, whereas *GFP* and *mock* (mock transfection) correspond to negative controls. We shall return to this issue in Section 3, but now, we proceed to the next step of data assemble, 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 is 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 an enhancer-suppressor screen, the argument `posControls` of *writeReport* should be defined as a list with two components: *act* (for activator or enhancers) and *inh* (for inhibitors or suppressors), 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 are passed to the *regexpr* function for pattern matching within the well annotation (`wellAnno(x)`).

In the example presented here, in the positive controls, we have *AATK* and *ATTK* as enhancers, and *MAP2K6* as an inhibitor effector. The negative controls are *GFP* and the mock transfection (indicated as *mock* in the

[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 plate configuration file) is stored in slot `plateConf` of the `cellHTS` object - see `plateConf(x)$Content`.

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:

```
> setSettings(list(platelist=list(intensities=list(range=c(300, 4000),
+                                                  include=TRUE))))
> out <- writeReport(raw=x, outdir="2Wraw",
+                    posControls=posCtr, negControls=negCtr)
```

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. First we $\log_2$ transformed to make the data scale additive. Then, for each replicate, the plate intensity values at the negative control wells are taken as a correction factor, so that each plate measurement is subtracted by it.

```
> xn <- normalizePlates(x, scale="multiplicative", log=TRUE, method ="negatives",
+                       varianceAdjust="none", negControls = negCtr)
```

The normalized intensities are stored in the slot `assayData` of `xn`. This slot can be assessed as an array with dimensions equal to the total number of features (product between the plate dimension and the total number of plates) x number of samples (or replicates) x number of channels (in this case, 1).

```
> xnorm <- Data(xn)
> dim(xnorm)

Features  Samples Channels
      96        2        1
```

Below, we call the function *scoreReplicates* to determine the z-score values for each replicate, and the function *summarizeReplicates* to summarize the replicate 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').

```
> xsc <- scoreReplicates(xn, sign="+", method="zscore")
> xsc <- summarizeReplicates(xsc, summary="mean")
```

The resulting single z-score value per probe were stored again in the slot `assayData` of `xsc`. Figure 1 shows the boxplots of the z-scores for the different types of probes (the "empty" wells were excluded from the plot).

```
> ylim <- quantile(Data(xsc), c(0.001, 0.999), na.rm=TRUE)
> wa <- factor(as.character(wellAnno(xsc)), exclude="empty")   # to exclude "empty" w
> boxplot(Data(xsc) ~ wa, col="lightblue", main="scores", outline=FALSE, ylim=ylim, x
> lab <- unique(plateConf(xsc)$Content)
> lab <- lab[match(levels(wa), tolower(lab))]
> axis(1, at=c(1:nlevels(wa)), labels=lab)
```

Now that the data have been normalized, scored and summarized between replicates, we call again *writeReport* and use a web browser to view the resulting report:

```
> setSettings(list(platelist=list(intensities=list(range=c(-1, 1), include=TRUE)),
+                  screenSummary=list(scores=list(range=c(-2,3)))))
> out <- writeReport(raw=x, normalized=xn, scored=xsc,
+                    outdir="2Wnormalized", posControls=posCtr, negControls=negCtr)

> 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 (enhancers and suppressors). Furthermore, the report shows the image plot with the final scored and summarized values for the whole data set. Finally, we save the scored and summarized *cellHTS* object to a file.

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

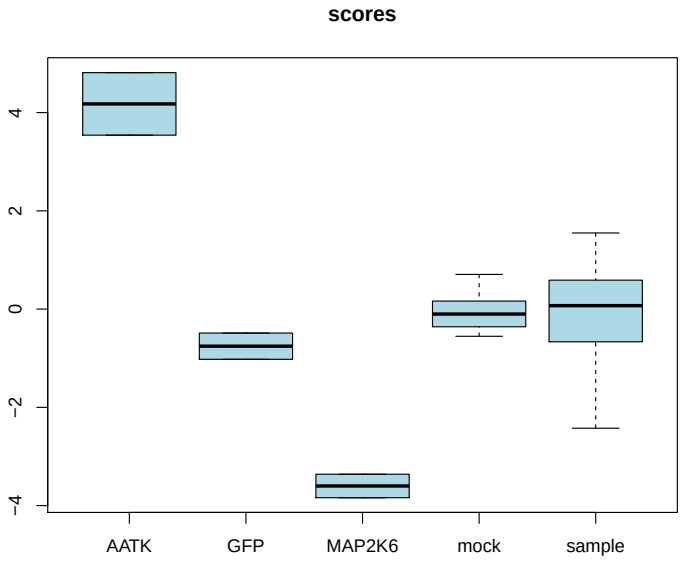

Figure 1: Boxplots of z-scores for the different types of probes.

4 Session info

This document was produced using:

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

- R Under development (unstable) (2018-10-16 r75448), `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`
- Running under: `Ubuntu 18.04.1 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.9-bioc/R/lib/libRblas.so`

- LAPACK: `/home/biocbuild/bbs-3.9-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.45.0, Biobase 2.43.0, BiocGenerics 0.29.0, Category 2.49.0, GO.db 3.7.0, GSEABase 1.45.0, IRanges 2.17.0, KEGG.db 3.2.3, Matrix 1.2-14, RColorBrewer 1.1-2, S4Vectors 0.21.0, XML 3.98-1.16, annotate 1.61.0, cellHTS2 2.47.0, genefilter 1.65.0, graph 1.61.0, hexbin 1.27.2, hwriter 1.3.2, locfit 1.5-9.1, splots 1.49.0, vsn 3.51.0
- Loaded via a namespace (and not attached): BiocManager 1.30.3, DBI 1.0.0, DEoptimR 1.0-8, KernSmooth 2.23-15, MASS 7.3-51, R6 2.3.0, RBGL 1.59.0, RCurl 1.95-4.11, RSQLite 2.1.1, Rcpp 0.12.19, affy 1.61.0, affyio 1.53.0, assertthat 0.2.0, bindr 0.1.1, bindrcpp 0.2.2, bit 1.1-14, bit64 0.9-7, bitops 1.0-6, blob 1.1.1, cluster 2.0.7-1, colorspace 1.3-2, compiler 3.6.0, crayon 1.3.4, digest 0.6.18, dplyr 0.7.7, ggplot2 3.1.0, glue 1.3.0, gtable 0.2.0, labeling 0.3, lattice 0.20-35, lazyeval 0.2.1, limma 3.39.0, magrittr 1.5, memoise 1.1.0, munsell 0.5.0, mvtnorm 1.0-8, pcaPP 1.9-73, pillar 1.3.0, pkgconfig 2.0.2, plyr 1.8.4, prada 1.59.0, preprocessCore 1.45.0, purrr 0.2.5, rlang 0.3.0.1, robustbase 0.93-3, rrcov 1.4-4, scales 1.0.0, splines 3.6.0, survival 2.43-1, tibble 1.4.2, tidyselect 0.2.5, tools 3.6.0, xtable 1.8-3, zlibbioc 1.29.0

References

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