Analysis of Pairwise Interaction Screens

Bernd Fischer

October 27, 2020

Contents

1 Introduction

The package contains an analysis pipeline for data from quantitative interaction screens. The package is the basis for the analysis of an RNAi interaction screen reported in

> Thomas Horn, Thomas Sandmann, Bernd Fischer, Elin Axelsson, Wolfgang Huber, and Michael Boutros (2011):*Mapping of Signalling Networks through Synthetic Genetic Interaction Analysis by RNAi*, Nature Methods 8(4): 341-346.

This package provides the software that implements the methods used in this paper; these methods are further described in the Supplemental Methods section of the paper.

2 Installation of the RNAinteract package

To install the package RNAinteract, you need a running version of R (www.r-project.org, version $\geq$ 2.13.0). After installing R you can run the following commands from the R command shell to install RNAinteract and all required packages.

```
> if (!requireNamespace("BiocManager", quietly=TRUE))
+     install.packages("BiocManager")
> BiocManager::install("RNAinteract")
```

The R code of this vignette can be accessed in the Sweave file `RNAinteract.Rnw`. The R code is extracted from the Sweave file and written to a file `RNAinteract.R` by

```
> Stangle(system.file("doc", "RNAinteract.Rnw", package="RNAinteract"))
```

3 Creating an RNAinteract object

The package RNAinteract is loaded by the following command.

```
> library("RNAinteract")
```

In the package included is an example for a small genetic interaction screen. Text files containing a description of the design of the screen and the screen data are located in the following directory.

```
> inputpath = system.file("RNAinteractExample",package="RNAinteract")
> inputpath
```

The directory `inputpath` contains five text files:
`Targets.txt`, `Reagents.txt`, `TemplateDesign.txt`, `QueryDesign.txt`, `Platelist.txt`.
Open these files in a text editor to inspect the file format.
The first file (`Targets.txt`) contains information about the targeted genes. The three columns `TID`, `Symbol`, and `group` are required. Optionally other columns can be added. `TID` is a unique identifier for the target gene. Preferably, this is the ENSEMBL gene identifier or the identifier of another reference database. A short, human-readable gene name is provided in the column `Symbol`. The column `group` should contain a grouping of the genes into *sample* genes, negative (*neg*), or positive (*pos*) controls. The grouping is used later on, in quality control plots, or in displays such as the heatmap where the control data need to be omitted.

```
> inputfile <- system.file("RNAinteractExample/Targets.txt",package="RNAinteract")
> T <- read.table(inputfile, sep="\t", stringsAsFactors=FALSE, header=TRUE)
> head(T)
```

```
          TID Symbol  group         GID AnnotationSymbol                Name
1 FBgn0000229    bsk sample FBgn0000229           CG5680              basket
2 FBgn0001291    Jra sample FBgn0001291           CG2275 Jun-related antigen
3 FBgn0001297    kay sample FBgn0001297          CG33956               kayak
4 FBgn0001965    Sos sample FBgn0001965           CG7793    Son of sevenless
5 FBgn0003177    pyd sample FBgn0003177          CG31349        polychaetoid
6 FBgn0003205 Ras85D sample FBgn0003205           CG9375 Ras oncogene at 85D
```

The file `Reagents.txt` contains the $n : 1$ mapping of reagents to target genes. In the example screen each gene is targeted by two independet dsRNA designs. The mandatory columns are `RID` (a unique identifier of the ragent) and `TID` (the target gene identifier as defined in the file `Targets`). Optionally, additional columns such as RNA sequences can be added.

```
> inputfile <- system.file("RNAinteractExample/Reagents.txt",package="RNAinteract")
> T <- read.table(inputfile, sep="\t", stringsAsFactors=FALSE, header=TRUE)
> head(T[,c("RID", "TID", "PrimerSeqFor","PrimerSeqRev","Length")])
```

```
       RID         TID         PrimerSeqFor         PrimerSeqRev Length
1 SGI00002 FBgn0000229 ATAGACTTTTCCCCGATGGC CCTCAGCATCATACCACACG    174
2 SGI00005 FBgn0001291 TCAACTCACCGGATCTGTCA TTGTGTAAGGCCTCCTCGAA    175
3 SGI00006 FBgn0001297  AAAGGTGCTACCCAATGCC AGTATCGGTCGTGTCCTGCT    175
4 SGI00007 FBgn0001965 CAGGACGACATCGAGAGCTT AGTGCTCCACCTTCATTTGG    175
5 SGI00010 FBgn0003177 CTGGGACGATGTGGTCTTCT ATCCTGCAGTGGAGTCGAAA    175
6 SGI00011 FBgn0003205  ATGGAGAGACCTGCCTGCT CTTTACGCGCTTGATCTGCT    173
```

The screen design is assumed to be a template-query design. A template plate contains in each well a different reagent. There may be multiple template plates to cover the whole set of reagents. Afterwards one query reagent will be added to each well on the template plates. By this strategy a matrix of double RNAi treatments is obtained. One can either choose to screen all selected genes against all selected genes, or some number of template genes against another number of query genes. The file `TemplateDesign.txt` contains information about the template plate design. `TemplatePlate` is the number of the template plate. Numbering starts with 1. In the example there is only one template plate; in other applications, there may be multiple template plates. `Well` is a single letter followed by a number and identifies the well coordinates within the plates. `RID` defines the reagent in the respective well with the same identifier as in the file `Reagents.txt`. In the example screen

there are 48 different template reagents. These span the left hand side of the multiwell plate. The right hand side is filled with the same reagents and will be covered with a different query reagent from the left hand side. To distinguish which query is spotted on each well the last column `QueryNr` denotes the number of the query. In the example the left hand side gets the query number 1, the right hand side gets the query number 2.

```
> inputfile <- system.file("RNAinteractExample/TemplateDesign.txt",package="RNAinteract")
> T <- read.table(inputfile, sep="\t", stringsAsFactors=FALSE, header=TRUE)
> head(T)
```

```
  TemplatePlate Well      RID QueryNr
1             1   A1 SGI00013       1
2             1   A2 SGI00017       1
3             1   A3 SGI00021       1
4             1   A4 SGI00109       1
5             1   A5 SGI00113       1
6             1   A6 SGI00117       1
```

The file `QueryDesign.txt` specifies the query genes on each physical plate in the screen design. The plates are numbered starting from 1. In the example there are two different query reagents on the same physical plate: One on the left half of the plate and one on the right half of the plate.

```
> inputfile <- system.file("RNAinteractExample/QueryDesign.txt",package="RNAinteract")
> T <- read.table(inputfile, sep="\t", stringsAsFactors=FALSE, header=TRUE)
> head(T)
```

```
  Plate TemplatePlate QueryNr       RID
1     1             1       1  SGI00005
2     1             1       2  SGI00007
3     2             1       1  SGI00011
4     2             1       2 Ctrl_Fluc
5     3             1       1  SGI00089
6     3             1       2  SGI00101
```

In many cases a screen is repeated (as technical or biological replicates) or it is conducted under multiple conditions (e.g. with an additional drug). In the following, we will refer to either replicates or different conditions as screens. Furthermore the plates are usually numbered with a platebarcode. The file `Platelist.txt` shows in which file the readout data is stored (column `Filename`), and which `Platebarcode` is associated with which `Plate` as defined in the query design and which replicate it represents.

```
> inputfile <- system.file("RNAinteractExample/Platelist.txt",package="RNAinteract")
> T <- read.table(inputfile, sep="\t", stringsAsFactors=FALSE, header=TRUE)
> head(T)
```

```
                     Filename Platebarcode Plate Replicate
1 DataRNAinteractExample_1.txt   PLATE11001     1         1
2 DataRNAinteractExample_1.txt   PLATE11002     2         1
3 DataRNAinteractExample_1.txt   PLATE11003     3         1
4 DataRNAinteractExample_1.txt   PLATE11004     4         1
5 DataRNAinteractExample_1.txt   PLATE11005     5         1
6 DataRNAinteractExample_1.txt   PLATE11006     6         1
```

The platelist in the example says that the data is distributed in two files. The first file is shown below. Beside the mandatory columns `Platebarcode` and `Well` there is a column for each quantitative value (readout channel) that is measured within each well.

```
> inputfile <- system.file("RNAinteractExample/DataRNAinteractExample_1.txt",package="RNAinteract")
> T <- read.table(inputfile, sep="\t", stringsAsFactors=FALSE, header=TRUE)
```

The data and annotation are loaded and an *RNAinteract* object is created with the following command.

```
> sgi = createRNAinteractFromFiles(name="RNAi interaction screen", path = inputpath)
> sgi
```

Multiple pairwise interaction screens with the same annotation can be stored in one *RNAinteract* object, e.g. if the screen is replicated or if the screen is repeated under multiple conditions. Here, the object `sgi` contains two replicate screens. Multiple readout channels can be captured in the same object as well. In this case we have three channels.

```
> getChannelNames(sgi)
```

```
[1] "nrCells"   "area"      "intensity"
```

4 Single Perturbation effects and pairwise interactions

The functions for manipulating the *RNAinteract* object are working as follows: the *RNAinteract*-object is given as the first argument of each function. The function performs some calculations and stores the result again in the object that is returned by the functions.
First, the single perturbation effects (called main effects) are estimated from the data. For each template position and for each query reagent a main effect is estimated.

```
> sgi <- estimateMainEffect(sgi, use.query="Ctrl_Fluc")
```

If the main effects contain time or plate dependent trends, these can be adjusted and removed ("normalized"). The normalization of the main effects does not influence the subsequent estimation of the pairwise interactions, but it makes the main effects better comparable between replicates and different screens.
When the main effects are available, the pairwise interaction term can be estimated.

```
> sgi <- computePI(sgi)
```

The object `sgi` contains two replicate screens. We summarize these two screens by taking the mean value for each measurement and add the mean screen as a new screen to the original screen.

```
> sgim <- summarizeScreens(sgi, screens=c("1","2"))
> sgi3 <- bindscreens(sgi, sgim)
```

The p-values are computed by

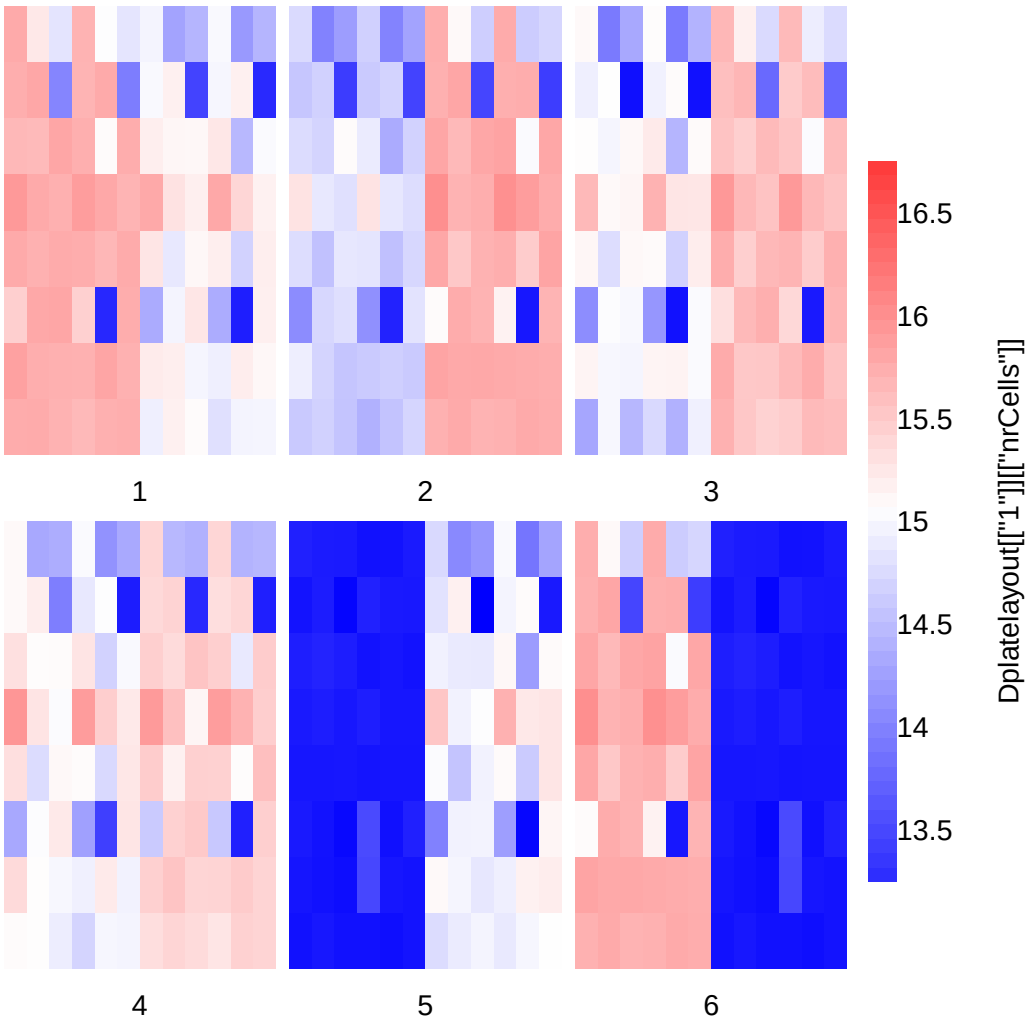
```
> sgi3 <- computePValues(sgi3)
> sgi3limma <- computePValues(sgi3, method="limma")
> sgi3T2 <- computePValues(sgi3, method="HotellingT2")
```

independently for each screen and each channel. The genetic interaction scores for each gene pair are tested against the null-hypothesis that the interaction term is zero. It is a two sided test. For `sgi3`, a conservatie test is used, that regularizes the variance term in the t-statistic by taking the maximum of the empirical variance per gene pair, and a global value obtained from a pooled within-group variance across all data. p-values are derived from a t-statistics with n-1 degrees of freedom. `sgi3limma` contains p-values derived by limma (See R-package limma). limma uses a more gradual moderation between the local and the global variance estimate. The object `sgi3T2` will contain p-values derived from a Hotelling T^2 test, where the deviation from the non-interactiong model is tested in a multivariate manner in all channel dimensions.

5 Data Access

The main function for data access is `getData`. The raw data can be accessed in different formats. The code below shows the access of the raw data in plain format, in plate layout, and as a matrix of genes. Usually the data stored in the RNAinteract object is log-transformed. Therefore the inverse transformation has to be applied to obtain the raw input data.

```
> data("sgi")
> D <- getData(sgi, type="data", do.inv.trafo = TRUE)
> Dplatelayout <- getData(sgi, type="data",
+              format="platelist", do.inv.trafo = TRUE)
> splots::plotScreen(Dplatelayout[["1"]][["nrCells"]],
+              nx=sgi@pdim[2], ny=sgi@pdim[1], ncol=3)
> Dmatrix <- getData(sgi, type="data",
+              format="targetMatrix", do.inv.trafo = TRUE)
```

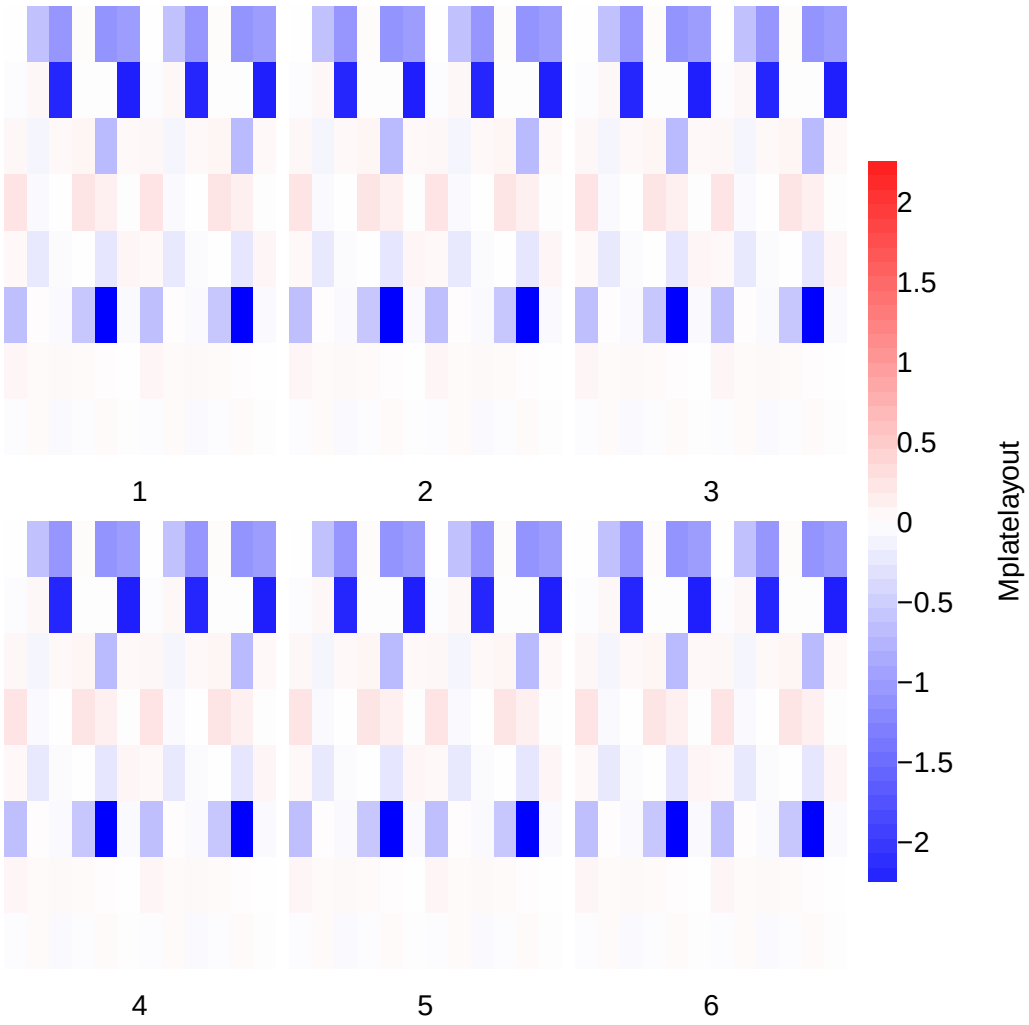


One can access the raw data of a single screen or single readout channel.

```
> data("sgi")
> D <- getData(sgi, screen="2", channel="nrCells",
+              type="data", do.inv.trafo = TRUE, format="targetMatrix")
```

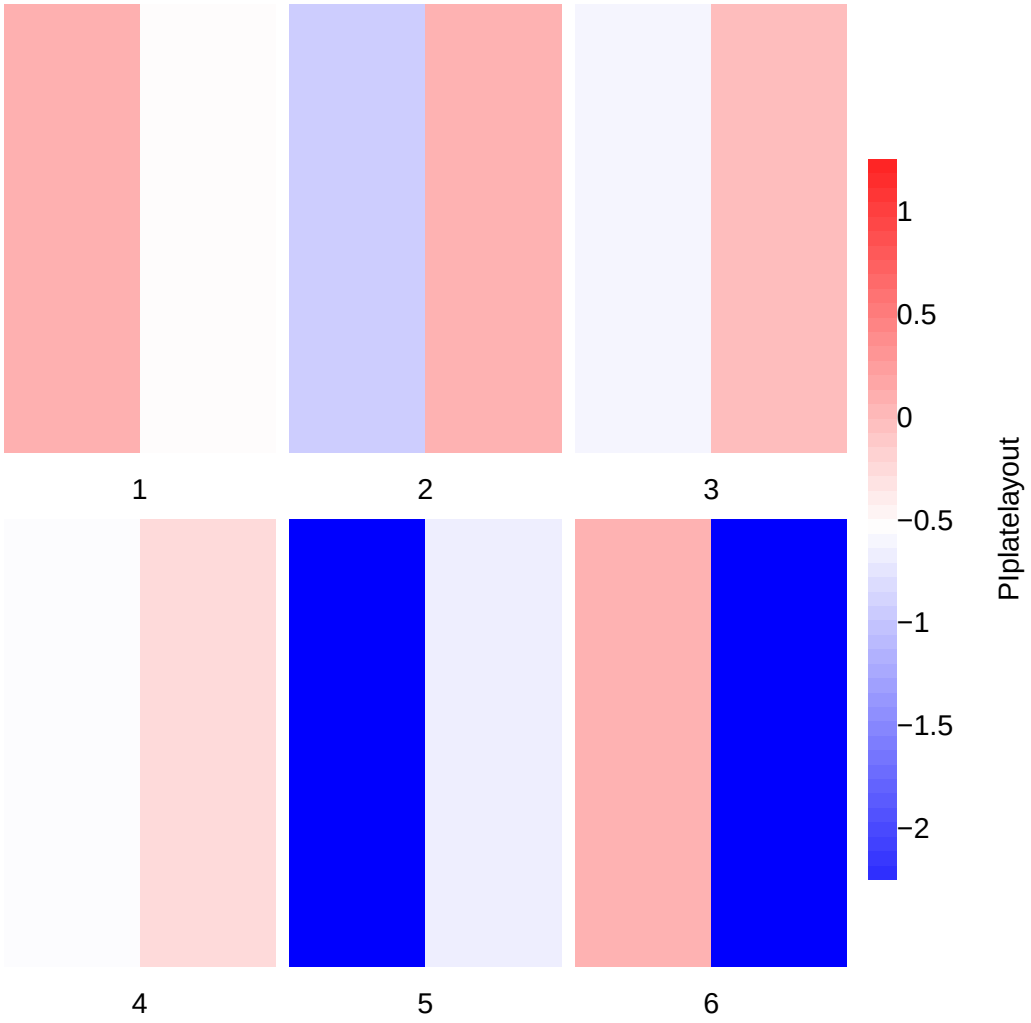
The main effects can be accessed in the same way and displayed in plate layout. In this case the main effects are returned in a log-transformed way.

```
> Mplatelayout <- getData(sgi, type="main", design="template",
+              screen="1", channel="nrCells", format="platelist")
> splots::plotScreen(Mplatelayout, nx=sgi@pdim[2], ny=sgi@pdim[1],
+              ncol=3)
```



The expected values from the non-interacting model, pairwise interaction scores, p-values and q-values can be accessed in the same way. p- and q-values can not be accessed in plain format or plate layout format, because they are not values of a single experiment.

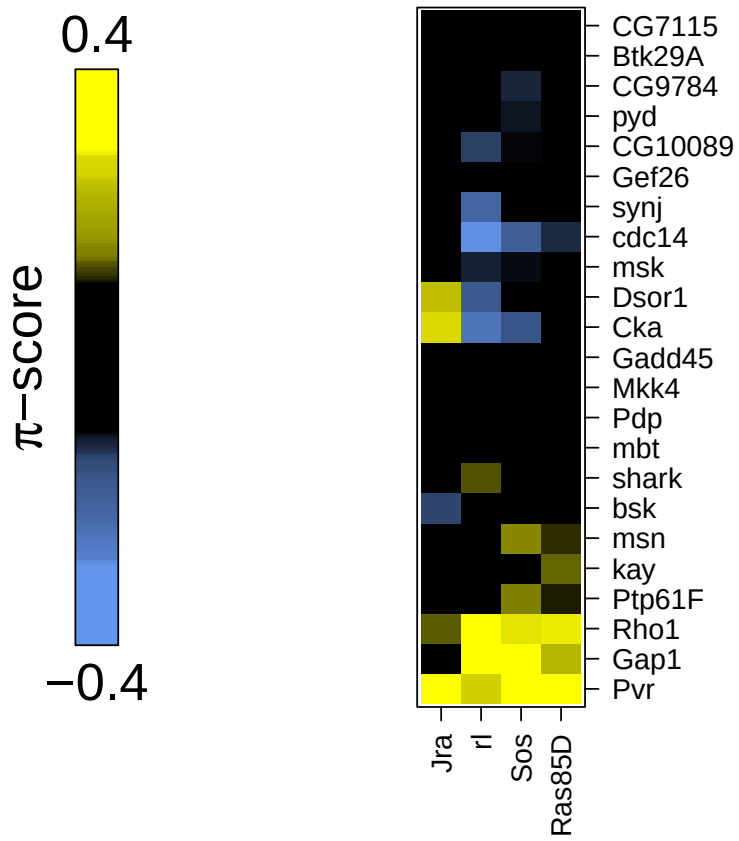
```
> NImatrix <- getData(sgi, type="ni.model", format="targetMatrix")
> PImatrix <- getData(sgi, type="pi", format="targetMatrix")
> PIplatelayout <- getData(sgi, type="main", design="query",
+               screen="1", channel="nrCells", format="platelist")
> splots::plotScreen(PIplatelayout, nx=sgi@pdim[2], ny=sgi@pdim[1],
+               ncol=3)
> p.value <- getData(sgi, type="p.value", format="targetMatrix")
> q.value <- getData(sgi, type="q.value", format="targetMatrix")
```



6 Graphical Output

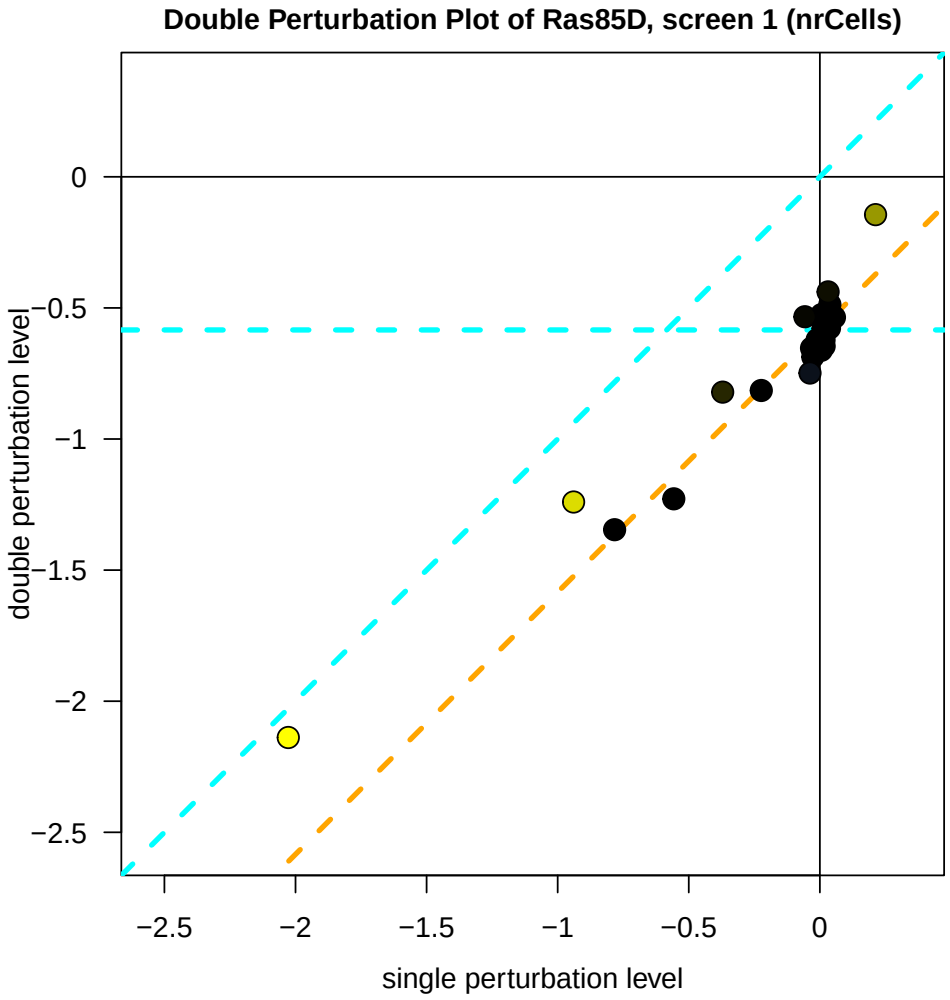
A heatmap is plotted with negative interactions colored blue and positive interactions colored yellow. To display the heatmap, one has to select the screen and the channel to be displayed.

```
> plotHeatmap(sgi, screen="1", channel="nrCells")
```



A double RNAi plot shows the interaction profile for one gene. Each dot corresponds to one gene pair containing the selected query gene (Ras85D) and one other gene. The x-axis depicts the single RNAi effect of the other gene. The y-axis shows the double RNAi effect of the gene pair. If the gene pair is not interacting, the dot lies on the orange, diagonal line. If the double RNAi effect is equal to the single RNAi effect of one of the genes (Epistasis), the dot lies on one of the blue lines.

```
> plotDoublePerturbation(sgi, screen="1", channel="nrCells", target="Ras85D")
```



7 A HTML-report

To generate a HTML report, first, the `outputpath` has to be defined. With `startReport` a HTML-page is opened for writing. The function returns a report object `report`. This object is handed over to the report functions. In the following example, the annotation (`reportAnnotation`) is written to the HTML-file, then a hit list for the pooled variance t-test (`reportGeneLists(sgi3, ...)`) and the limma t-test (`reportGeneLists(sgi3limma, ...)`) is added.

```
> outputpath = "RNAinteractHTML"
> report = startReport(outputpath)
> reportAnnotation(sgi3, path = outputpath, report = report)
> reportStatistics(sgi3, path = outputpath, report = report)
> reportGeneLists(sgi3, path = outputpath, report = report)
> reportGeneLists(sgi3limma, path = outputpath, dir="hitlistlimma",
+                 prefix = "hitlistlimma", report = report)
```

For further quality control, we compare the estimated main effects by scatter plots. To check for plate and edge effects in the screen we generate screen plots for the input data as well as for the estimated pairwise interactions.

```
> reportMainEffects(sgi3, path = outputpath, report = report)
> reportScreenData(sgi3, plotScreen.args=list(ncol=3L, do.legend=TRUE,
+                                             fill = c("red","white","blue")),
+                  path = outputpath, report = report)
```

Double perturbation plots are generated for each gene, each screen, and each channel to observe the genetic interaction profile of a single gene.

```
> reportDoublePerturbation(sgi3, path = outputpath, report = report,show.labels="p.value")
```

For each screen and each channel a heatmap is added to the report by

```
> reportHeatmap(sgi, path=outputpath, report=report)
```

The report is closed by a call to `endReport`.

```
> save(sgi, file=file.path(outputpath, "RNAinteractExample.rda"))
> endReport(report)
```

Finally the report can be opened in a browser.

```
> browseURL(file.path(outputpath, "index.html"))
```

8 Session Info

```
> sessionInfo()

R version 4.0.3 (2020-10-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 18.04.5 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.12-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.12-bioc/R/lib/libRlapack.so

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

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] RNAinteract_1.38.0  Biobase_2.50.0      BiocGenerics_0.36.0
[4] locfit_1.5-9.4      abind_1.4-5

loaded via a namespace (and not attached):
 [1] httr_1.4.2            vsn_3.58.0            bit64_4.0.5
 [4] splines_4.0.3         gtools_3.8.2          ICSNP_1.1-1
 [7] BiocManager_1.30.10   affy_1.68.0           stats4_4.0.3
[10] latticeExtra_0.6-29   RBGL_1.66.0           blob_1.2.1
[13] Category_2.56.0       robustbase_0.93-6     splots_1.56.0
[16] pillar_1.4.6          RSQLite_2.2.1         lattice_0.20-41
[19] glue_1.4.2            limma_3.46.0          digest_0.6.27
[22] RColorBrewer_1.1-2    colorspace_1.4-1      preprocessCore_1.52.0
[25] Matrix_1.2-18         survey_4.0            GSEABase_1.52.0
[28] pcaPP_1.9-73          XML_3.99-0.5          pkgconfig_2.0.3
[31] genefilter_1.72.0     zlibbioc_1.36.0       purrr_0.3.4
[34] xtable_1.8-4          mvtnorm_1.1-1         scales_1.1.1
[37] jpeg_0.1-8.1          affyio_1.60.0         cellHTS2_2.54.0
[40] tibble_3.0.4          annotate_1.68.0       generics_0.0.2
[43] IRanges_2.24.0        ggplot2_3.3.2         ellipsis_0.3.1
```

```
[46] survival_3.2-7        magrittr_1.5          crayon_1.3.4
[49] memoise_1.1.0         MASS_7.3-53           gplots_3.1.0
[52] hwriter_1.3.2         graph_1.68.0          tools_4.0.3
[55] mitools_2.4           lifecycle_0.2.0       S4Vectors_0.28.0
[58] munsell_0.5.0         AnnotationDbi_1.52.0  compiler_4.0.3
[61] caTools_1.18.0        rlang_0.4.8           grid_4.0.3
[64] bitops_1.0-6          gtable_0.3.0          DBI_1.1.0
[67] rrcov_1.5-5           R6_2.4.1              dplyr_1.0.2
[70] bit_4.0.4             KernSmooth_2.23-17    prada_1.66.0
[73] Rcpp_1.0.5            ICS_1.3-1             vctrs_0.3.4
[76] geneplotter_1.68.0    png_0.1-7             DEoptimR_1.0-8
[79] tidyselect_1.1.0
```