Mapping genetic interactions in human cancer cells with RNAi and multiparametric phenotyping

Christina Laufer, Bernd Fischer, Maximilian Billman, Wolfgang Huber, and Michael Boutros. Nature Methods (2013) 10(5):427-31 doi: 10.1038/nmeth.2436

Bernd Fischer

European Molecular Biology Laboratory (EMBL),
Heidelberg, Germany
bernd.fischer@embl.de

October 18, 2014

Contents

1 Introduction

This document is associated with the *Bioconductor* package *HD2013SGI*, which contains the data and the *R* code for the statistical analysis presented in the paper

> *Mapping genetic interactions in human cancer cells with RNAi and multiparametric phenotyping*
> Christina Laufer, Bernd Fischer, Maximilian Billman, Wolfgang Huber, and Michael Boutros
> Nature Methods (2013) 10(5):427-31
> doi: 10.1038/nmeth.2436.

In Section 2, the access to the data is described. Section 3 shows the figures from the original publication and links each figure to one of the subsequent sections, which describe the statistical analyses in the order in which they were executed. The *R* code within each section can be executed independently of the other sections. Intermediate results from each section are available in the form of *R* data objects.

To install the *HD2013SGI*, please start a current version of *R* and type

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("HD2013SGI")
```

2 Data access

The interaction matrix can be loaded by

```
> library("HD2013SGI")
> data("Interactions", package="HD2013SGI")
```

The element `Interactions$piscore` of this data object is a 6-dimensional array of π-scores, `Interactions$padj` is a 5-dimensional array of BH-adjusted p-values, and `Interactions$Anno` contains annotation for target genes, query genes, and phenotypes. You can type

```
> ? Interactions
```

to see the documentation of this data object. Similarly, you can use the ? operator to obtain help on any of the below data objects; The following datasets are available:

- **Interactions** The genetic interaction data (π-scores, p-values, and annotation). **This is the main data source of this package.**
- **featuresPerWell** The screen data in screen order.
- **datamatrixfull** The phenotype data of all pairwise genetic perturbation experiments before quality control and feature selection.
- **QueryAnnotation** Annotation of all the query genes in the screen.
- **TargetAnnotation** Annotation of all target genes in the screen.
- **stabilitySelection** Results from the feature selection step.
- **datamatrix** The phenotype data of all pairwise genetic perturbation experiments after quality control and feature selection.
- **mainEffects** Estimated main effects (single knock down effect).
- **nrOfInteractionsPerTarget** Number of interactions per target gene.

They are intermediate results of the analysis steps described in the following, precomputed and loadable for convenience.

3 Figures from the paper

In this section all figures from the original publication are shown. In the next sections the *R* code for the analysis is documented. A link to the respective section in which the figure is generated is given for each figure.

Figure 1cd

R code for this figure is documented in Section 7.2.

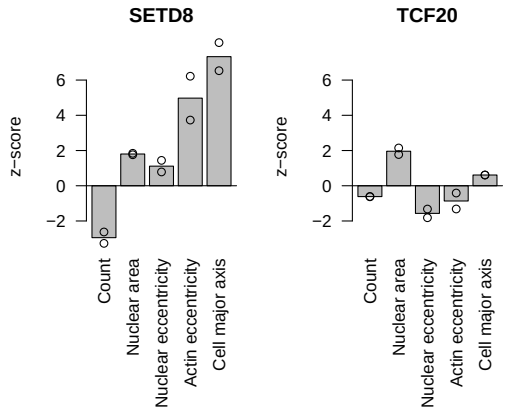


Figure 2a

R code for this figure is documented in Section 6.3.

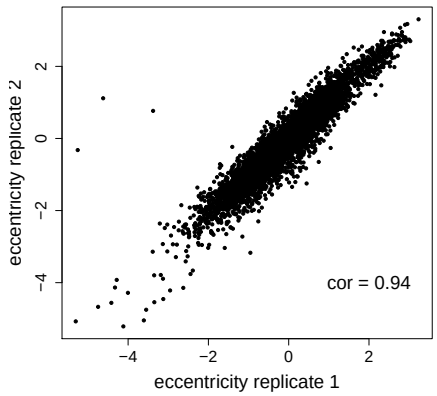


Figure 2bd

R code in Section 6.3 and 6.4.

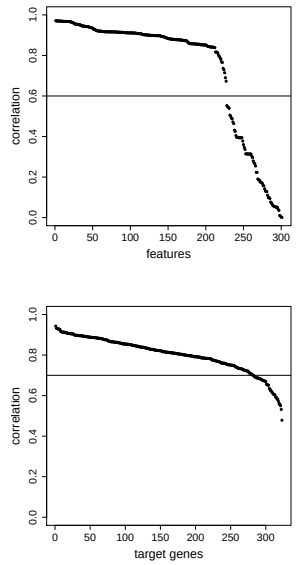


Figure 2c

R code in Section 6.4.

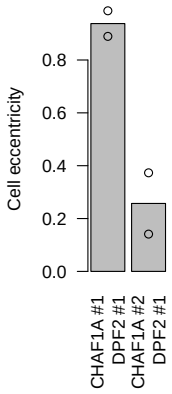


Figure 2e

R code in Section 10.

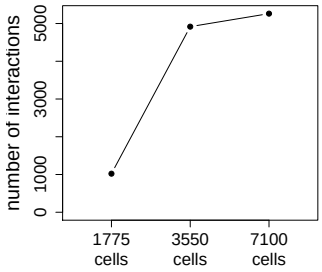


Figure 3ab

R code for this figure is documented in Section 7.3.

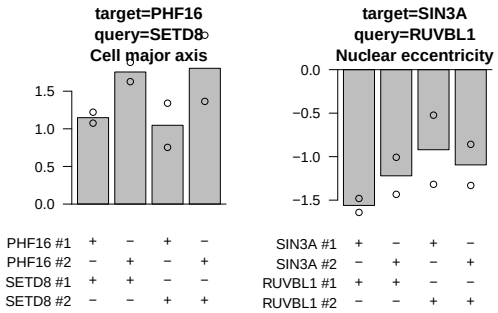


Figure 3c

R code for this figure is documented in Section 7.4.

```
> # load(file.path("result","Figures","Overlap.rda"))
> # print(Overlap)
```

Figure 4ac

R code for this figure is documented in Section 9.

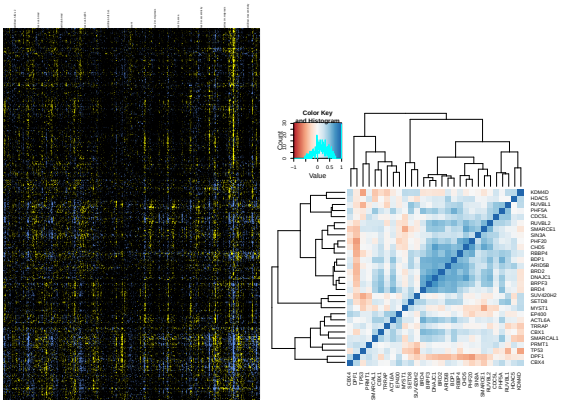


Figure 4b

R code for this figure is documented in Section 8.

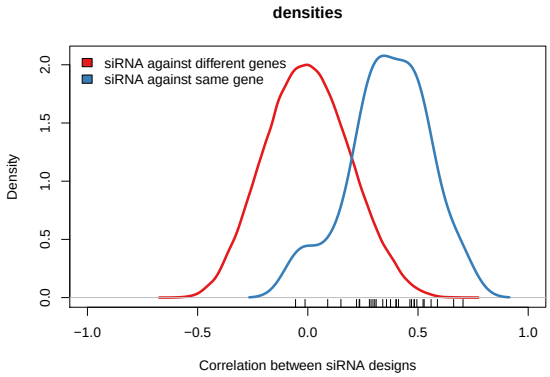


Figure S3

R code for this figure is documented in Section 6.5.

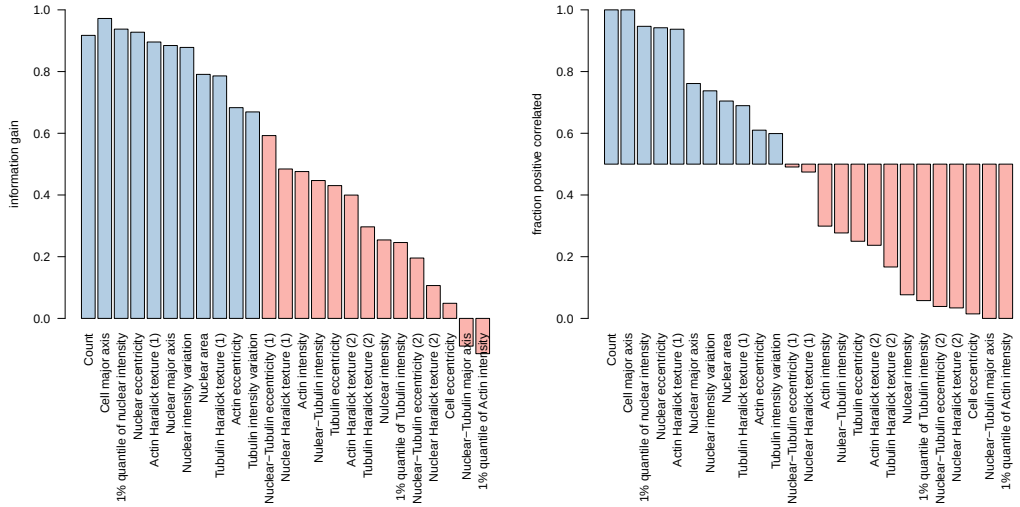


Figure S4

R code for this figure is documented in Section 12.

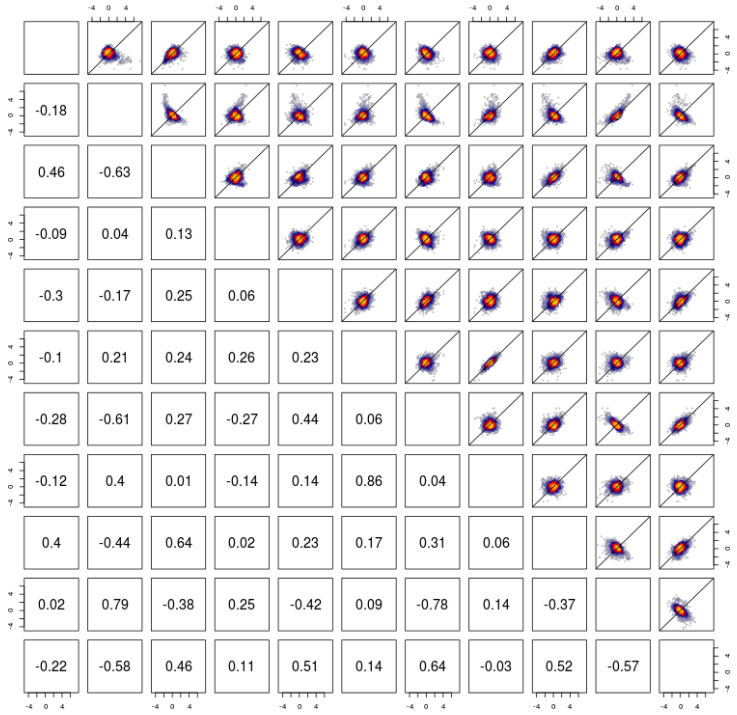


Figure S5

R code for this figure is documented in Section 13.

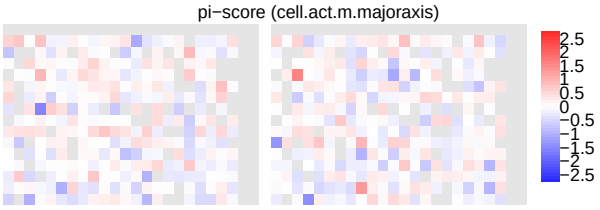
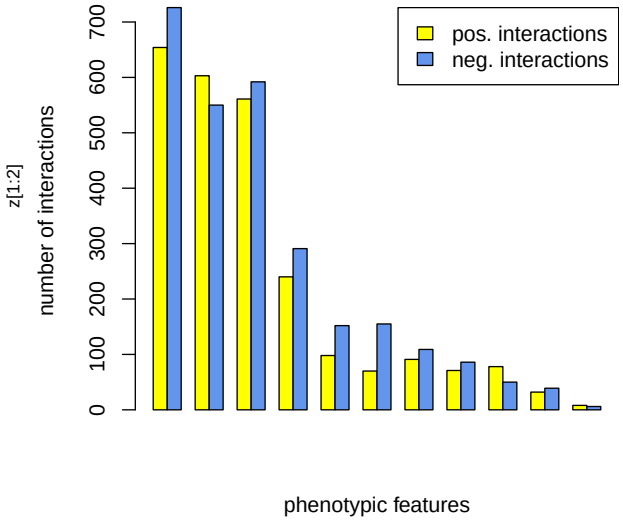


Figure S6

R code for this figure is documented in Section 11.



4 Image segmentation and feature extraction

4.1 Preliminaries

Load the *Bioconductor* package *HD2013SGI*.

```
> library("HD2013SGI")
> dir.create(file.path("result","Figures"),recursive=TRUE,showWarnings=FALSE)
```

We adapted image segmentation and feature extraction methods from previous work[1, 2, 3], using the *Bioconductor* package *EBImage*[4]. The data set comprised 5.6 terabytes. Here, exemplarily we show the image segmentation and feature extraction on a clipped imaged of size 340 pixels $\times$ 490 pixels.

4.2 Image segmentation

Images were obtained from the InCell Analyzer 2000 as 12-bit TIFF images of size 2,048 pixels $\times$ 2,048 pixels at three colors. Techically the images were stored as 16bit-images, but their dynamic range only extended to 12bit $(0, \ldots, 4095)$. The package contains a clipped image of size 340 pixels $\times$ 490 pixels. The three channels were saved in separate TIFF files, and we load them into the workspace as follows.

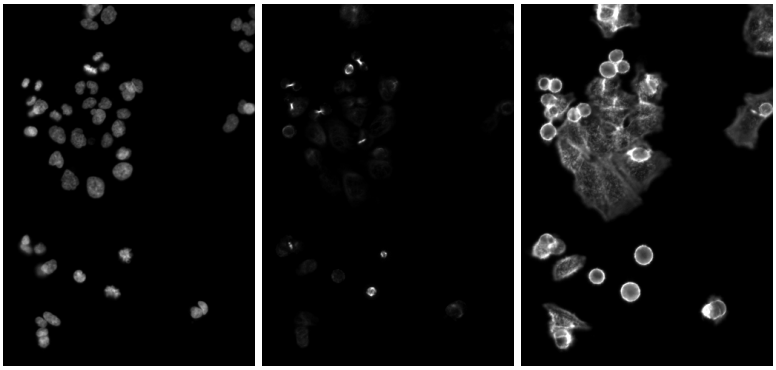
```
> fnch1 = system.file(file.path("images","image-DAPI.tif"), package="HD2013SGI")
> fnch2 = system.file(file.path("images","image-FITC.tif"), package="HD2013SGI")
> fnch3 = system.file(file.path("images","image-Cy3.tif"),  package="HD2013SGI")
> Img1 = readImage(fnch1)
> Img2 = readImage(fnch2)
> Img3 = readImage(fnch3)
```

The images `Img1`, `Img2` and `Img3` are represented as arrays of floating point numbers. Their dynamic range depends in each case on the staining reagent, the power of the light source and the excitation time. For display purposes, the following commands obtain linearly transformed versions of the images; note however that the subsequent analysis will be performed on the original images.

```
> ImgDAPI = normalize(Img1, inputRange=c(0.002, 0.03))
> ImgFITC = normalize(Img2, inputRange=c(0.004, 0.04))
> ImgCy3 = normalize(Img3,  inputRange=c(0.002, 0.03))
```

```
> display(combine(ImgDAPI, ImgFITC, ImgCy3))
```

The `display` function will open the images in a browser, and you can navigate through the three channels with the arrows on the top of the page.

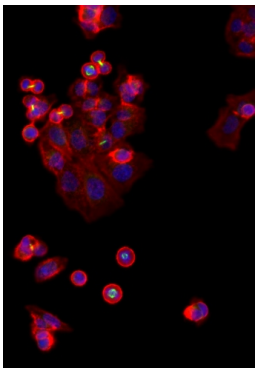


ImgDAPI ImgFITC ImgCy3

We can combine the three channels to a single false color image using red for Actin, yellow for α-Tubulin, and blue for the DAPI staining.

```
> ImgColor = rgbImage(ImgCy3,ImgFITC,ImgDAPI)
```

```
> display(ImgColor)
```



ImgFITC

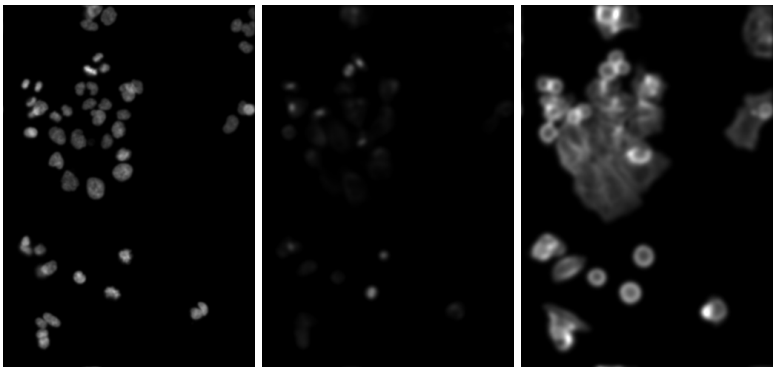
In a first processing step we smoothed the images by linear filtering with a Gaussian kernel with a small bandwidth of 1 pixel, respectively 3 pixels.

```
> Filter1 = makeBrush(size=51,shape = "gaussian",sigma=1)/12.91571
> Filter3 = makeBrush(size=51,shape = "gaussian",sigma=3)/12.91571
> Img1smooth = filter2(Img1, filter=Filter1)
> Img2smooth = filter2(Img2, filter=Filter3)
> Img3smooth = filter2(Img3, filter=Filter3)
```

The smoothed images can be displayed as above after normalization. The difference to the original images is almost not visible, but it helped to smoothen the segmentation boundaries that were generated by the next steps.

```
> smoothDAPI = normalize(Img1smooth*12.91571, inputRange=c(0.002, 0.03))
> smoothFITC = normalize(Img2smooth*12.91571, inputRange=c(0.004, 0.04))
> smoothCy3 = normalize(Img3smooth*12.91571,  inputRange=c(0.002, 0.03))
```

```
> display(combine(smoothDAPI,smoothFITC,smoothCy3))
```



smoothDAPI smoothFITC smoothCy3

Nuclei were segmented by adaptive thresholding with a window size of 10 pixels (corresponding to 7.4 μm). See the next row of images for the outcomes of each of the following steps.

```
> nucleusThresh = thresh(Img1smooth, w = 10, h = 10, offset = 0.0001)
```

A morphological operation (opening) removes tiny segments and smoothens their shapes.

```
> nucleusOpening = opening(nucleusThresh, kern=makeBrush(3, shape="disc"))
```

An integer value is assigned to individually label the segmented regions.

```
> nucleusSeed = bwlabel(nucleusOpening)
```

The resulting segmentation (nucleusSeed), shown below, nicely separates the different nuclei from each other, but unfortunately in some cases shows holes and does not cover all of the area with nuclear staining in the image. Therefore, we generated a second segmentation, to be used as a mask, that covered the whole nuclear stained region, but which was in many places connected across neighbouring nuclei. To do so, we started with a less stringent adaptive thresholding and applied to it another morphological operation, which fills holes that are surrounded by foreground pixels.

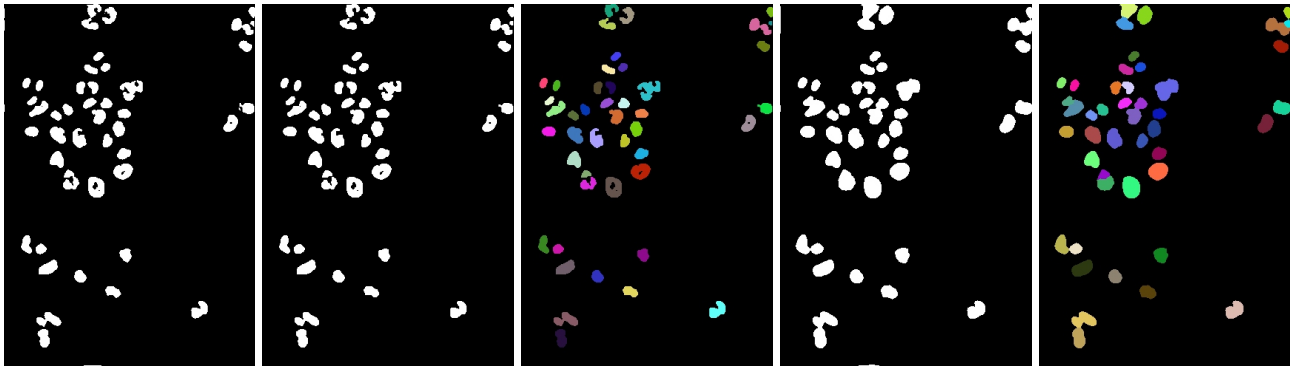
```
> nucleusFill = fillHull(thresh(Img1smooth, w = 20, h = 20, offset = 0.00005))
```

To improve on what we had in (nucleusSeed), we propagated the segmented objects until the mask defined nucleusFill was filled. Boundaries between nuclei, in those places where the mask was connected, were found by Voronoi tessellation on a metric space that depends on the gradient field of smoothDAPI using the algorithm by Jones et al. [5].

```
> nucleusRegions = propagate(Img1smooth, nucleusSeed, mask=nucleusFill)
```

We can now display the five above images from the nucleus segmentation.

```
> display(combine(nucleusThresh,
+                 nucleusOpening,
+                 colorLabels(nucleusSeed),
+                 nucleusFill,
+                 colorLabels(nucleusRegions)))
```



nucleusThresh nucleusOpening nucleusSeed nucleusFill nucleusRegions

To segment the cell bodies, we applied adaptive thresholding followed by morphological opening on the actin channel `Img3smooth`, similar to what we did for the nuclei on the DAPI channel (`Img1smooth`).

```
> cytoplasmThresh = thresh(Img3smooth, w = 20, h = 20, offset = 0.000001)
> cytoplasmOpening = opening(cytoplasmThresh,kern=makeBrush(3,shape="disc"))
```

Because the cell bodies often covered large sections of whole image, adaptive thresholding did not always detect all cellular areas, and we employed a second segmentation by a global, large threshold.

```
> globalThreshold = 0.0003
> cytoplasmOpening2 = opening(Img3smooth > globalThreshold)
```

To define the mask of image area covered by cell bodies, we combined the two cytoplasmic masks and the nuclear mask by taking their union.

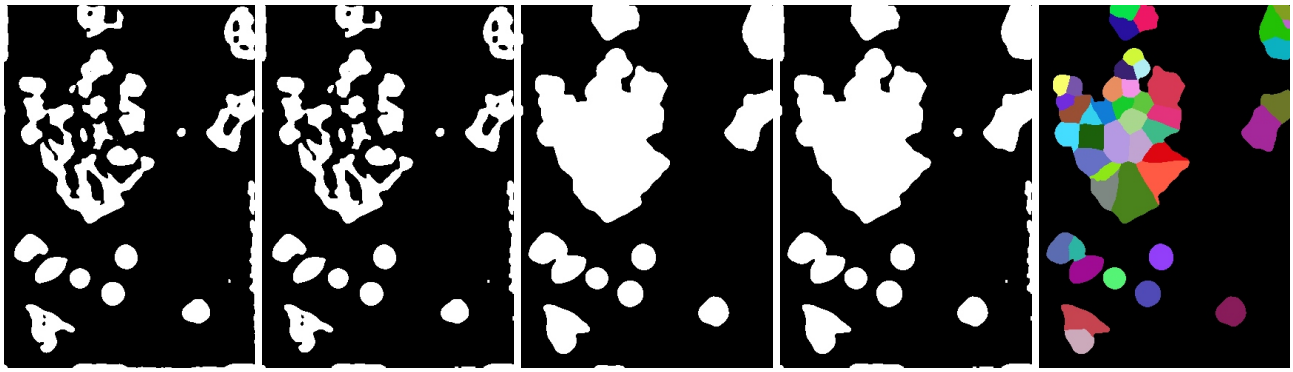
```
> cytoplasmCombined = cytoplasmOpening
> cytoplasmCombined[cytoplasmOpening2 > cytoplasmCombined] =
+               cytoplasmOpening2[cytoplasmOpening2 > cytoplasmCombined]
> cytoplasmCombined[nucleusFill > cytoplasmCombined] =
+               nucleusFill[nucleusFill > cytoplasmCombined]
```

To define the cellular bodies, the nucleus segmentation was extended by a Voronoi tessellation-based propagation algorithm[5].

```
> cytoplasmRegions = propagate(Img3smooth, nucleusRegions, lambda=1.0e-4, mask=cytoplasmCombined)
```

The three image masks for the segmentation of the cytoplam are displayed below.

```
> display(combine(cytoplasmThresh,
+                 cytoplasmOpening,
+                 cytoplasmOpening2,
+                 cytoplasmCombined,
+                 colorLabels(cytoplasmRegions)))
```

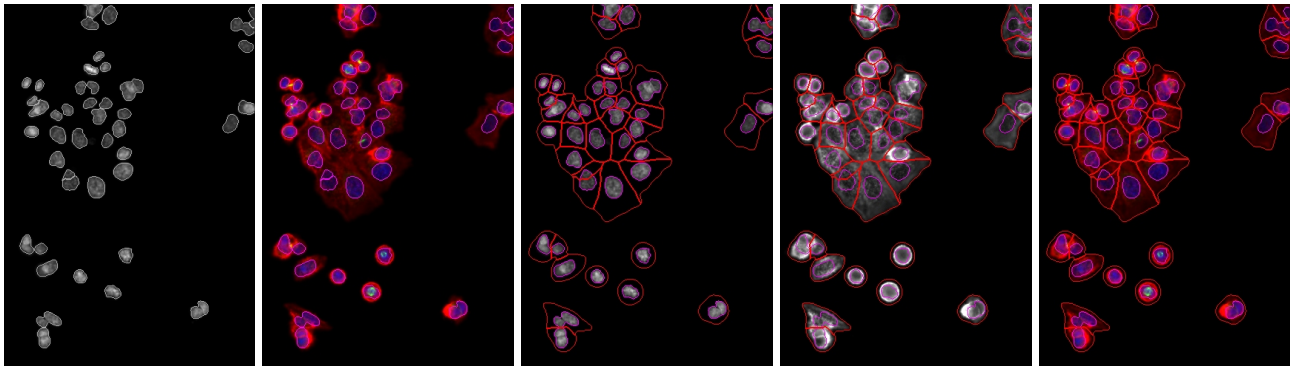


cytoplasmThresh cytoplasmOpening cytoplasmOpening2 cytoplasmCombined cytoplasmRegions

As an alternative representation, below we display the nucleus segmentation and cell body segmentation on top of the original images.

```
>   Imgout1 = paintObjects(nucleusRegions,ImgDAPI, col='#ff00ff')
>   Imgout2 = paintObjects(nucleusRegions,ImgColor, col='#ff00ff')
>   Imgout3 = paintObjects(cytoplasmRegions,
+                          paintObjects(nucleusRegions,
+                                       toRGB(ImgDAPI),
+                                       col='#ff00ff'),
+                          col='#ff0000')
>   Imgout4 = paintObjects(cytoplasmRegions,
+                          paintObjects(nucleusRegions,
+                                       toRGB(ImgCy3),
+                                       col='#ff00ff'),
+                          col='#ff0000')
>   Imgout5 = paintObjects(cytoplasmRegions,
```

```
+                          paintObjects(nucleusRegions,ImgColor, col='#ff00ff'),
+                          col='#ff0000')
```



nucleusRegions DAPI | nucleusRegions Actin/α-Tubulin/DAPI | cytoplasmRegions DAPI | cytoplasmRegions Actin | cytoplasmRegions Actin/α-Tubulin/DAPI

4.3 Feature extraction

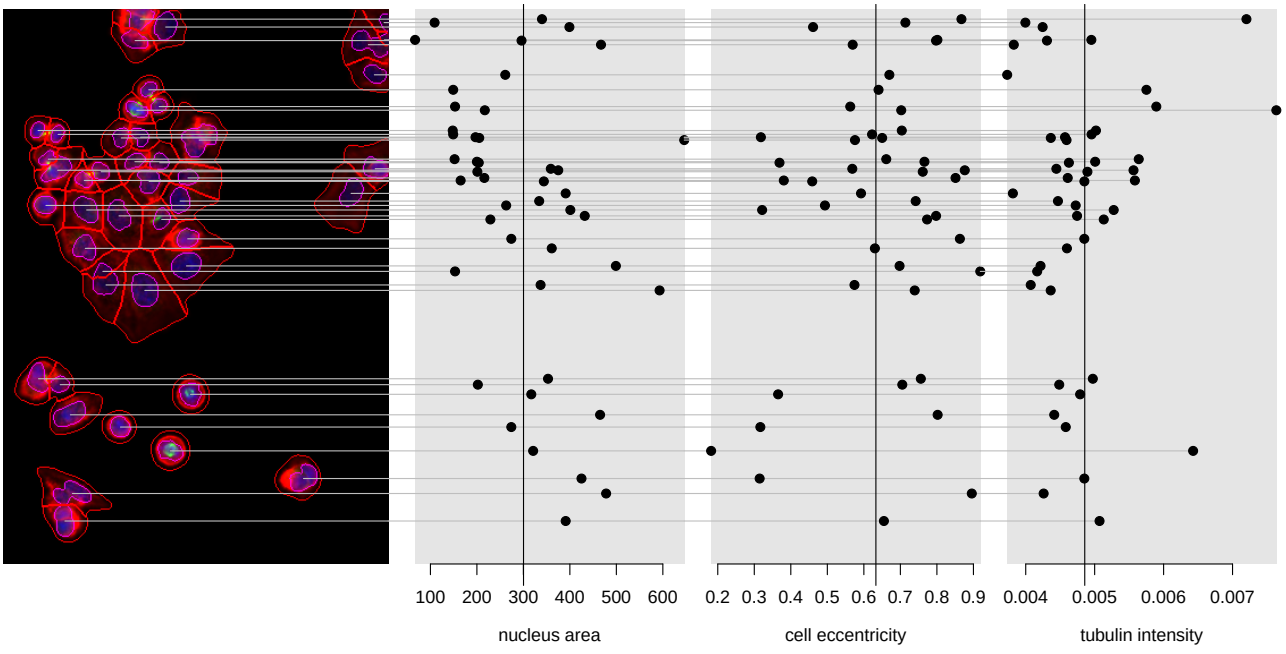
Features for intensity, shape and texture were extracted for each cell from the DAPI channel using the nucleus segmentation (`nucleusRegions`) and from the actin and tubulin channels using the cell body segmentation (`cytoplasmRegions`).

```
> F1 = computeFeatures(nucleusRegions,  Img1, xname="nuc",  refnames="nuc")
> F2 = computeFeatures(cytoplasmRegions,Img2, xname="cell", refnames="tub")
> F3 = computeFeatures(cytoplasmRegions,Img3, xname="cell", refnames="act")
```

Additional features were extracted from the joint distribution of the DAPI and the tubulin signals using the cell body mask.

```
> F4 = computeFeatures(cytoplasmRegions, (Img1 - mean(Img1)) * (Img2 - mean(Img2)),
+                      xname="cell", refnames="nuctub")
```

The below figure shows three phenotypic features for the cells segmented in the example image.



Features were summarized per experiment by the arithmetic mean over all cells, which in the figure above is indicated by the black vertical lines. The number of segmented nuclei was used as a proxy for cell count.

```
> F = cbind(F1, F2, F3, F4)
> Fwell = c(count=nrow(F), apply(F, 2, mean))
```

The equivalent of `Fwell`, computed for all images in the screen, together with associated metadata, is available as R-object `featuresPerWell` from the package and can be loaded by `data("featuresPerWell", package="HD2013SGI")`; see the next section.

5 Convert data from plate order to multi-dimensional SGI-arrary

5.1 Preliminaries

Load the package *HD2013SGI*.

```
> library("HD2013SGI")
```

Load screening data in workspace.

```
> data("featuresPerWell", package="HD2013SGI")
```

After image segmentation and feature extraction, a feature vector is available for each image field. The feature data are represented as a 2 dimensional table with extension 231840 image fields $\times$ 353 phenotypic features. There are 168 plates in the screen. On those plates, `NROW` rows and `NCOL` columns were used, and `NFIELD` fields were imaged per well:

```
> NROW = 15
> NCOL = 23
> NFIELD = 4
```

In this section, these data are rearranged into a 6-dimensional array with one dimension each for target designs, features, and replicatess.

An output directory is created where the data array will be placed in.

```
> dir.create(file.path("result","data"), recursive=TRUE, showWarnings=FALSE)
```

5.2 Parse plate barcodes to annotate the plates

Plate barcodes and plate numbers are extracted from the screen annotation. The plate annotation is summarized in a data frame.

```
> plates = featuresPerWell$Anno[seq(1, nrow(featuresPerWell$Anno), by=NFIELD*NCOL*NROW),"plate"]
> PlateAnnotation = HD2013SGI:::parsePlateBarcodes(plates)
> head(PlateAnnotation)
```

```
          plate targetDesign queryGroup queryGene queryDesign replicate
1   001CIQ01IRI            1     sample        01           1         1
2  002CIQ01IIRI            1     sample        01           2         1
3  003CIIQ01IRI            2     sample        01           1         1
4 004CIIQ01IIRI            2     sample        01           2         1
5   005CIQ02IRI            1     sample        02           1         1
6  006CIQ02IIRI            1     sample        02           2         1
```

The names of all query genes, siRNA designs, and replicates are extracted for all sample experiments.

```
> S = which(PlateAnnotation$queryGroup=="sample")
> tdnames = unique(PlateAnnotation$targetDesign[S])
> qnames = unique(PlateAnnotation$queryGene[S])
> qdnames = unique(PlateAnnotation$queryDesign[S])
> repnames = unique(PlateAnnotation$replicate[S])
```

5.3 Reorder data

In this step, we create the array `D` and fill it with the data. The target genes are spread over the three dimensions `field`, `col` and `row`. The mean of the measurements in the 4 fields per well is taken.

```
> D = array(0, dim=c(field=NFIELD, col=NCOL, row=NROW,
+                      features=dim(featuresPerWell$data)[2],
+                      targetDesign=length(tdnames),
+                      query=length(qnames),
+                      queryDesign=length(qdnames),
+                      replicate=length(repnames)))
> dimnames(D) = list(field=seq_len(NFIELD),
+                    col=seq_len(NCOL), row=LETTERS[seq_len(NROW)+1],
+                    features=dimnames(featuresPerWell$data)[[2]],
+                    targetDesign=tdnames,
+                    queryGene=qnames,
+                    queryDesign=qdnames,
+                    replicate=repnames)
> for (td in tdnames) {
+   for (q in qnames) {
+     for (qd in qdnames) {
+       for (r in repnames) {
+         plate = PlateAnnotation$plate[
+                     which((PlateAnnotation$targetDesign == td) &
+                           (PlateAnnotation$queryGene    == q ) &
+                           (PlateAnnotation$queryDesign  == qd) &
+                           (PlateAnnotation$replicate    == r )) ]
+         I = which(featuresPerWell$Anno$plate == plate)
+         D[,,,,td,q,qd,r] = as.vector(featuresPerWell$data[I,])
+       }
+     }
+   }
+ }
> D[is.na(D)] = 0
> D = (D[1,,,,,,,] + D[2,,,,,,,] + D[3,,,,,,,] + D[4,,,,,,,])/4
> # faster than D = apply(D,2:8,mean,na.rm=TRUE)
> dim(D)

[1]  23  15 353   2  20   2   2
```

The dimensions are reordered, and row and column dimensions are merged into a single dimension for target genes. The data is saved to disk.

```
> D = aperm(D,c(1,2,4,5,6,3,7))
> dn = dimnames(D)
> dim(D) = c(prod(dim(D)[1:2]),dim(D)[3:7])
> dimnames(D) = c(list(targetGene =
+                        sprintf("%s%d",rep(LETTERS[seq_len(NROW)+1],each=NCOL),
+                                       rep(seq_len(NCOL),times=NROW))),
+                     dn[3:7])
> datamatrixfull = list(D = D)
> save(datamatrixfull, file=file.path("result","data","datamatrixfull.rda"))
```

The raw data are now represented in the 6-dimensional array `D` with dimensions

	345	target genes
×	2	siRNA target designs
×	20	query genes
×	2	siRNA query designs
×	353	phenotypic features
×	2	biological replicates

A precomputed version of the `datamatrix` is available from the package and can be loaded by `data(datamatrix, package="HD2013SGI")`.

6 Processing of the main data set

6.1 Preliminaries

Load the *Bioconductor*-package *HD2013SGI*.

```
> library("HD2013SGI")
```

Load data and screen annotation in workspace.

```
> data("datamatrixfull",package="HD2013SGI")
> D = datamatrixfull$D
> data("TargetAnnotation",package="HD2013SGI")
> data("QueryAnnotation",package="HD2013SGI")
```

Output directories are created where the data, figures and tables will be placed in.

```
> dir.create(file.path("result","data"),recursive=TRUE,showWarnings=FALSE)
> dir.create(file.path("result","Figures"),recursive=TRUE,showWarnings=FALSE)
> dir.create(file.path("result","Tables"),recursive=TRUE,showWarnings=FALSE)
```

Get indices for different sets of experiments (samples, controls, ...).

```
> IndexSAMPLE = which(TargetAnnotation$group == "sample")
> IndexCTRL = which(TargetAnnotation$group == "SingleKDctrl")
> IndexNEG = which(TargetAnnotation$group == "negctrl")
> IndexSAMPLENEG = c(IndexSAMPLE, IndexNEG)
```

Set the color value for controls.

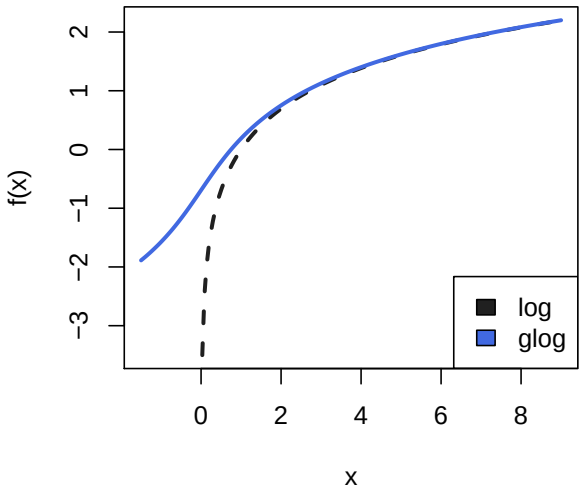
```
> colCTRL = rep("gray", nrow(TargetAnnotation))
> colCTRL[TargetAnnotation$group == "SingleKDctrl"] = "royalblue"
> colCTRL[TargetAnnotation$group == "negctrl"] = "red"
```

6.2 Transform features and screen normalization

Previous genetic interaction screens that were based on a quantitative cell viability phenotype were analysed using a multiplicative interaction model [6, 7, 8, 9, 10]. For model fitting, it is then convenient to transform the data to a logarithmic scale. Motivated by [11, 12], we adapted this approach to all features considered here. Since some features had a non-positive range of values, instead of the logarithm we applied a generalized logarithm transform [13]

$$f(x;c) = \log\left(\frac{x+\sqrt{x^2+c^2}}{2}\right). \tag{1}$$

This family of functions has one parameter c. For $c = 0$, the function is equivalent to an ordinary logarithm transformation. For $c > 0$, the function is smooth for all values of x (including 0 and negative values), avoiding the singularity of the ordinary logarithm at $x = 0$, but still approximately equivalent to the ordinary logarithm for $x \gg c$ as shown in the following plot.



For each feature, we chose c to be the 3%-quantile of the feature's empirical distribution.

```
> for (i in seq_len(dim(D)[5])) {
+   m = quantile(D[,,,,i,],probs=0.03,na.rm=TRUE)
+   D[,,,,i,] = glog(D[,,,,i,],m)
+ }
```

After transformation, to take account of the plate and batch effects and differences in efficiency of the siRNA transfection, an additive normalization is performed per plate. The plate median is set to be equal for all siRNA designs and replicates for each query gene and phenotypic feature. This compensates for global differences between replicates and siRNA designs.

```
> M = apply(D,c(2:6),median,na.rm=TRUE)
> M2 = apply(M, c(2,4), mean)
> M2 = rep(M2, times=8)
> dim(M2) = dim(M)[c(2,4,1,3,5)]
> M2 = aperm(M2,c(3,1,4,2,5))
> M = M - M2
> M = rep(M[], each=dim(D)[1])
> dim(M) = dim(D)
> D = D - M
```

After transformation, to take account of the fact that the data range of the different features was different, data were centered and scaled separately for each feature. Center and scale were computed as the median and median absolute deviation, respectively.

```
> for (i in seq_len(dim(D)[5])) {
+   D[,,,,i,] = (D[,,,,i,] - median(D[,,,,i,],na.rm=TRUE)) /
+                 mad(D[,,,,i,],na.rm=TRUE)
+ }
```

In the following, we will refer to the array of transformed, centered and scaled values as D_{ijklmr} with the indices i, j, k, l, m,r counting over the 6 dimensions of the data cube with extensions

	345	target genes (i)
$\times$	2	siRNA target design (j)
$\times$	20	query genes (k)
$\times$	2	siRNA query designs (l)
$\times$	353	phenotypic features (m)
$\times$	2	biological replicates (r)

6.3 Quality control of features

To control for the quality of each feature, its reproducibility over replicate measurements was assessed. For each feature f, we computed the two vectors $\mathbf{v}_f^1$ and $\mathbf{v}_f^2$,

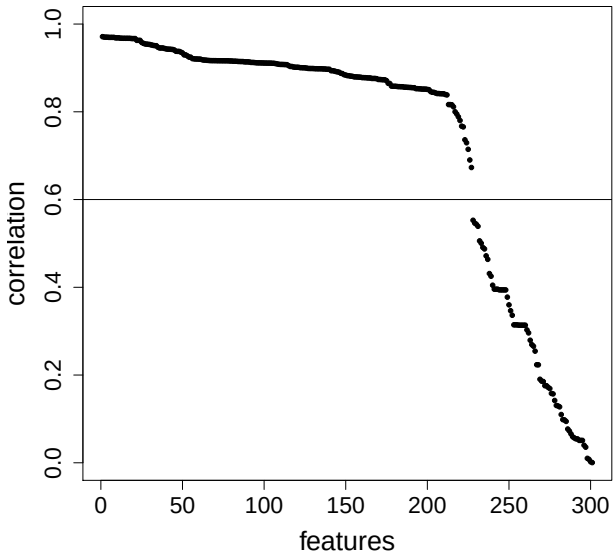
$$\mathbf{v}_f^l = D._-._-f\,l \qquad \text{for } l = 1, 2. \tag{2}$$

Here, the notation $-$ indicates averaging over an index, and the notation $\cdot$ indicates extraction of the whole subspace spanned by this index. Here, we used only the 323 sample genes, since the set of target genes contained a number of negative and positive controls. Thus, $\mathbf{v}_m^l$ is a vector with $323 \times 20 = 6460$ elements. We then computed the correlation coefficient ρ_f between $\mathbf{v}_f^1$ and $\mathbf{v}_f^2$.

```
> C = rep(NA_real_,dim(D)[5])
> D2 = (D[,1,,1,,] + D[,2,,1,,] + D[,1,,2,,] + D[,2,,2,,]) / 4
> for (i in seq_len(dim(D)[5])) {
+   C[i] = cor(as.vector(D2[IndexSAMPLE,,i,1]),as.vector(D2[IndexSAMPLE,,i,2]))
+ }
```

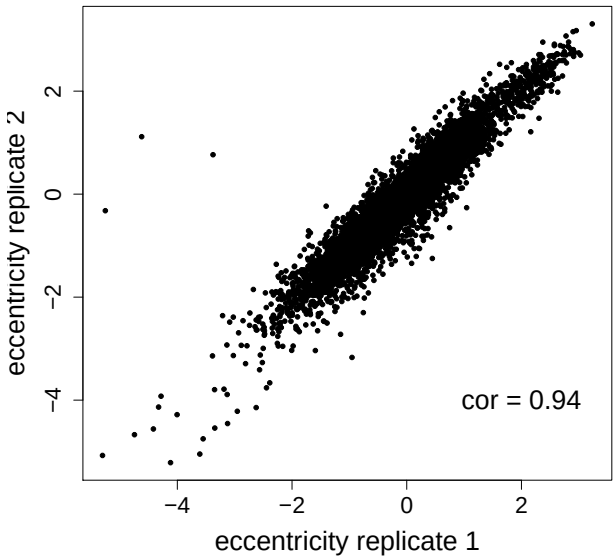
We plot the correlation coefficients for all 353 features.

```
> plot(sort(C, decreasing=TRUE), pch=20, xlab="features", ylab="correlation", ylim=c(0,1),
+      cex.lab=1.75, cex.axis=1.5)
> abline(h=0.6)
```



As an example, the scatter plot of eccentricity measurements is plotted for two replicates.

```
> i = "cell.Bact.m.eccentricity"
> plot(as.vector(D2[IndexSAMPLE,,i,1]),
+      as.vector(D2[IndexSAMPLE,,i,2]),
+      pch=20, cex.lab=1.75, cex.axis=1.5,
+      xlab="eccentricity replicate 1",
+      ylab="eccentricity replicate 2")
> cc = cor(as.vector(D2[IndexSAMPLE,,i,1]),
+          as.vector(D2[IndexSAMPLE,,i,2]))
> text(x=2, y=-4, sprintf("cor = %0.2f",cc), cex=1.75)
```



We selected features with a correlation of at least 0.6 for subsequent analysis. 227 out of 353 features passed quality control.

```
> I = which(C >= 0.6)
> D = D[,,,,I,,drop=FALSE]
> dim(D)
```

```
[1] 345   2  20   2 227   2
```

The dimension of the data cube after quality control of the phenotypic features is

	345	target genes
×	2	siRNA target designs
×	20	query genes
×	2	siRNA query designs
×	227	phenotypic features
×	2	biological replicates

6.4 Quality control of siRNA designs

To detect cases where the siRNA reagents for our target genes had off-target effects, we compared the phenotypic profiles of the two siRNA designs for each target gene. To this end, we computed the vectors

$$\mathbf{w}_{if}^{j} = D_{ij\cdot-f-} \tag{3}$$

using only those features that passed the quality filter described in Section 6.3. We then computed the correlation $\tilde{\rho'}_{if}$ between $\mathbf{w}_{if}^{1}$ and $\mathbf{w}_{if}^{2}$. The congruence score

$$\tilde{\rho}_i = \tilde{\rho'}_{i-} \tag{4}$$

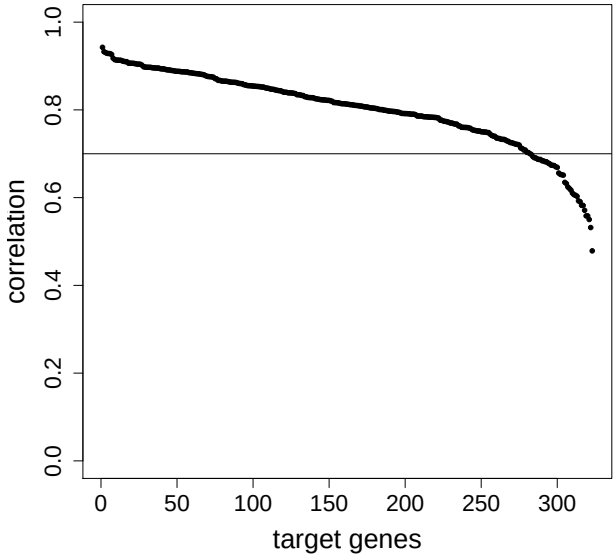
is the mean over the correlation coefficients of each feature.

```
> D1 = (D[,,,1,,1] + D[,,,1,,2] + D[,,,2,,1] + D[,,,2,,2])/4
> Cdesign1 = rep(NA_real_,dim(D)[1])
> for (k in seq_len(dim(D)[5])) {
+   for (i in seq_len(dim(D)[1])) {
+     Cdesign1[i] = cor(as.vector(D1[i,1,,k]),as.vector(D1[i,2,,k]))
+   }
+   if ( k == 1) {
```

```
+     Cdesign1all = Cdesign1
+   } else {
+     Cdesign1all = Cdesign1all + Cdesign1
+   }
+ }
> Cdesign1all = Cdesign1all / dim(D)[5]
```

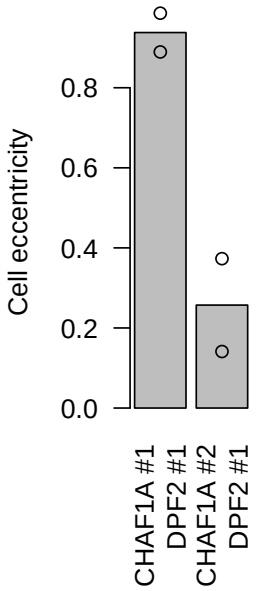
The congruence scores for all target genes are plotted in decreasing order.

```
> plot(sort(Cdesign1all[IndexSAMPLE], decreasing=TRUE),
+      pch=20, cex.lab=1.75, cex.axis=1.5, ylim=c(0,1),
+      xlab="target genes",ylab="correlation")
> abline(h=0.7)
```



CHAF1A is an example gene with a low congruence score. Exemplarily a barchart for the cell eccentricity for the double knock-downs of both siRNA designs of CHAF1A together with siRNA design 1 of DPF2 is plotted.

```
> f = "cell.Bact.m.eccentricity"
> D1 = D[ TargetAnnotation$Symbol == "CHAF1A",,QueryAnnotation$Symbol == "DPF2",1,f,]
> bp = barplot(apply(D1,1,mean),ylab="Cell eccentricity",
+         names.arg=c("CHAF1A #1\nDPF2 #1","CHAF1A #2\nDPF2 #1"),las=2)
> points(bp,D1[,1])
> points(bp,D1[,2])
```



Target gene i was selected for subsequent analysis if $\tilde{\rho}_i \geq 0.7$. The number of target genes passing quality control was 282 out of 323. Additionally, 7 negative control target siRNAs were selected.

```
> I = IndexSAMPLE[Cdesign1all[IndexSAMPLE] >= 0.7]
> I = c(I, IndexNEG)
> D = D[I,,,,,,drop=FALSE]
> TargetAnnotation = TargetAnnotation[I,]
> dim(D)
```

```
[1] 289   2  20   2 227   2
```

The dimension of the data cube `D` after quality control of the siRNA designs is

	289	target genes
×	2	siRNA target designs
×	20	query genes
×	2	siRNA query designs
×	227	phenotypic features
×	2	biological replicates

6.5 Selection of non-redundant features

The 227 features are highly redundant. We selected a set of non-redundant features. To do so, in the following the phenotypic profiles are summarized over all siRNA designs, and control measurements are excluded.

```
> D1 = (D[,1,,1,,] + D[,1,,2,,] + D[,2,,1,,] + D[,2,,2,,]) / 4
> D1 = D1[TargetAnnotation$group == "sample",,,]
```

3000 perturbations are randomly selected to speed up the selection process.

```
> dim(D1) = c(prod(dim(D1)[1:2]),dim(D1)[3:4])
> D1 = aperm(D1,c(1,3,2))
> set.seed(5830458)
> Sample = sample(seq_len(dim(D1)[1]), 3000)
> subSampleForStabilitySelection = list(D = D1[Sample,,],
+                                       Sample = Sample,
+                                       phenotype = dimnames(D)[[5]])
```

Next, the feature selection was performed. The feature *number of cells* was manually preselected because of its biological interest. All other features were selected automatically in a sequential way. In each step, all remaining

candidate features were evaluated separately. Let $F(i)$ be the features that were selected after step i. In step i, for all candidate features f a linear regression was fit on the previously selected features.

$$D._-._f. \sim D._-._{F(i-1)}. \tag{5}$$

Let $r._.f.$ be the residuals of this fit, i. e., the component of $D._-._f.$ that is orthogonal to the linear subspace spanned by $D._-._{F(i-1)}.$. The residuals are a composition of random noise and biological information that is non-redundant to the previously selected features. To estimate if there still exists unexplained biological signal, the Pearson correlation coefficient $\bar{\rho}_f$ of the residual vectors $r._.f1$ and $r._.f2$ between the two biological replicates was computed. It can be considered a proxy for the signal-to-noise ratio of these residuals. The feature with the largest correlation coefficient $\bar{\rho}$ was selected.

```
> stabilitySelection = HD2013SGIselectByStability(subSampleForStabilitySelection,
+                                                 preselect = c("count"),
+                                                 Rdim = 25, verbose = TRUE)
```

The step-wise approach was stopped when the number of positive correlation coefficients was smaller than or equal to the number of negative correlation coefficients,

$$\sum_f \left(\Theta\left(\bar{\rho}_f\right)\right) \leq \sum_f \left(\Theta\left(-\bar{\rho}_f\right)\right) , \tag{6}$$

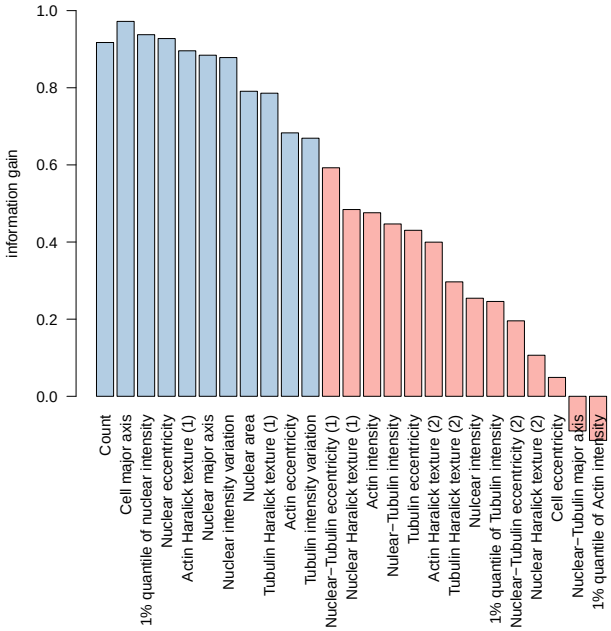
where $\Theta(x)$ is the Heaviside function. This criterion is motivated by the observation that for random data, half of the correlation coefficients is expected to be positive and the other half is expected to be negative.

```
> Sel = (stabilitySelection$ratioPositive >= 0.5)
> sum(Sel)
```

```
[1] 11
```

Thus, a set of 11 features was considered to contain non-redundant information. The barplot below shows the correlation coefficients of the residual features, which was considered the proxy for information content.

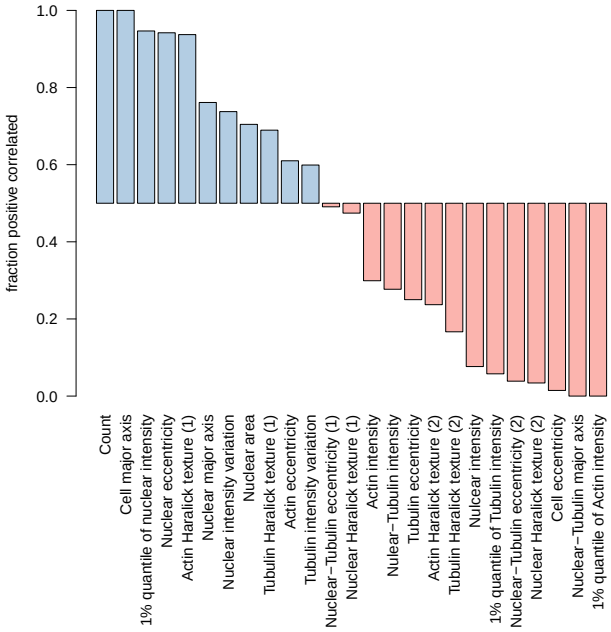
```
> barplot(stabilitySelection$correlation,
+         names.arg=HD2013SGI:::humanReadableNames[stabilitySelection$selected],
+         col=ifelse(Sel, col[2], col[1]),
+         ylim=c(0,1), las=2, ylab="information gain")
```



Now we plot the fraction of positive correlation coefficients, which served as a stop criterion.

```
> barplot(stabilitySelection$ratioPositive-0.5,
+         names.arg=HD2013SGI:::humanReadableNames[stabilitySelection$selected],
```

```
+         offset=0.5,col=ifelse(Sel, col[2], col[1]),ylim=c(0,1),
+         las=2,ylab="fraction positive correlated")
```



The criteria for feature selection are saved and the features are selected.

```
> save(stabilitySelection, file=file.path("result","data",
+                                         "stabilitySelection.rda"))
> D = D[,,,,stabilitySelection$selected[Sel],,drop=FALSE]
> dimnames(D)[[1]] = TargetAnnotation$Symbol
> dimnames(D)[[3]] = QueryAnnotation$Symbol
> dim(D)
```

```
[1] 289   2  20   2  11   2
```

After selecting 11 non-redundant features, the dimension of the data cube is

	289	target genes
×	2	siRNA target designs
×	20	query genes
×	2	siRNA query designs
×	11	phenotypic features
×	2	biological replicates

The datamatrix is now completely pre-processed and can be saved.

```
> datamatrix = list(D=D, Anno = list(target = TargetAnnotation,
+                                    query = QueryAnnotation,
+                                    phenotype=dimnames(D)[[5]]))
> save(datamatrix, file=file.path("result","data","datamatrix.rda"))
```

6.6 Pairwise interaction scores

Pairwise interaction scores (π-scores) were estimated using a robust linear fit. The query main effects are lifted such that they equal to the mean of the single knock down measurements of the query genes (target siRNA is a scrambled sequence serving as negative control).

```
> D = datamatrix$D
> pimatrix = datamatrix
> pimatrix$D[] = NA_real_
> mainEffects = list(target = D[,,1,,,],
```

```
+                     query = D[1,,,,,],
+                     overall = D[1,,1,,,],
+                     Anno = datamatrix$Anno)
> for (i in seq_len(dim(D)[2])) {
+   for (j in seq_len(dim(D)[4])) {
+     for (k in seq_len(dim(D)[5])) {
+       for (l in seq_len(dim(D)[6])) {
+         MP = HD2013SGImaineffects(D[,i,,j,k,l],
+             TargetNeg=which(TargetAnnotation$group == "negctrl"))
+         pimatrix$D[,i,,j,k,l] = MP$pi
+         mainEffects$target[,i,j,k,l] = MP$targetMainEffect
+         mainEffects$query[i,,j,k,l] = MP$queryMainEffect
+         mainEffects$overall[i,j,k,l] = MP$neg
+       }
+     }
+   }
+ }
> save(mainEffects, file=file.path("result","data","mainEffects.rda"))
```

6.7 Statistical testing of interaction terms

For statistical testing, the interaction terms that differed by ≥ 4 times the median standard deviation of the interaction scores were flagged as outliers. Afterwards the interaction terms were tested for significance by a moderated t-test implemented in the R package *limma*.

```
> D = pimatrix$D
> PADJ = D[pimatrix$Anno$target$group == "sample",,,,,1]
> s = rep(NA_real_, dim(D)[5])
> for (i in seq_len(dim(D)[5])) {
+   Data = D[,,,,i,]
+   Data = Data[pimatrix$Anno$target$group == "sample",,,,]
+   d = dim(Data)
+   dim(Data) = c(prod(d[1:4]),prod(d[5]))
+   Data[abs(Data[,1]-Data[,2]) > 4*mad(Data[,1]-Data[,2],center=0.0),]=NA_real_
+
+   s[i] = median(apply(Data,1,sd), na.rm=TRUE)
+   padj = rep(NA_real_, nrow(Data))
+   K = which(apply(!is.na(Data),1,all))
+   fit = eBayes(lmFit(Data[K,]))
+   padj[K] = p.adjust(fit$p.value, method="BH")
+   PADJ[,,,,i] = padj
+   cat(sprintf("i=%2d",i),
+       " nr int (1%) = ", sum(padj <= 0.01, na.rm=TRUE)/nrow(Data),
+       " nr int (3%) = ", sum(padj <= 0.03, na.rm=TRUE)/nrow(Data), "\n")
+ }
```

```
i= 1  nr int (1%) =  0.0006205674  nr int (3%) =  0.006826241
i= 2  nr int (1%) =  0.02353723  nr int (3%) =  0.07539894
i= 3  nr int (1%) =  0.05110816  nr int (3%) =  0.1007535
i= 4  nr int (1%) =  0.06117021  nr int (3%) =  0.1077128
i= 5  nr int (1%) =  0.009973404  nr int (3%) =  0.03922872
i= 6  nr int (1%) =  0.008865248  nr int (3%) =  0.02451241
i= 7  nr int (1%) =  0.05110816  nr int (3%) =  0.1058067
i= 8  nr int (1%) =  0.00695922  nr int (3%) =  0.02234043
i= 9  nr int (1%) =  0.005673759  nr int (3%) =  0.01710993
i=10  nr int (1%) =  0.01108156  nr int (3%) =  0.0454344
i=11  nr int (1%) =  0.003147163  nr int (3%) =  0.03843085
```

π-scores for each siRNA pair were summarized over both replicates. Furthermore, interaction scores were divided by their median deviation over all siRNA pairs, to scale them to comparable dynamic range.

```
> PI = pimatrix$D
> PI = PI[pimatrix$Anno$target$group == "sample",,,,,]
> Interactions = list(newpiscore = PI,
+                     scale = s,
+                     padj = PADJ,
+                     Anno = pimatrix$Anno)
> Interactions$Anno$target = Interactions$Anno$target[
+            pimatrix$Anno$target$group == "sample",]
> save(Interactions, file=file.path("result","data","Interactions.rda"))
```

7 Example phenotypes and interactions

7.1 Preliminaries

```
> library("HD2013SGI")
> data("Interactions",package="HD2013SGI")
> data("mainEffects",package="HD2013SGI")
> dir.create(file.path("result","Figures"),
+            recursive=TRUE,showWarnings=FALSE)
```

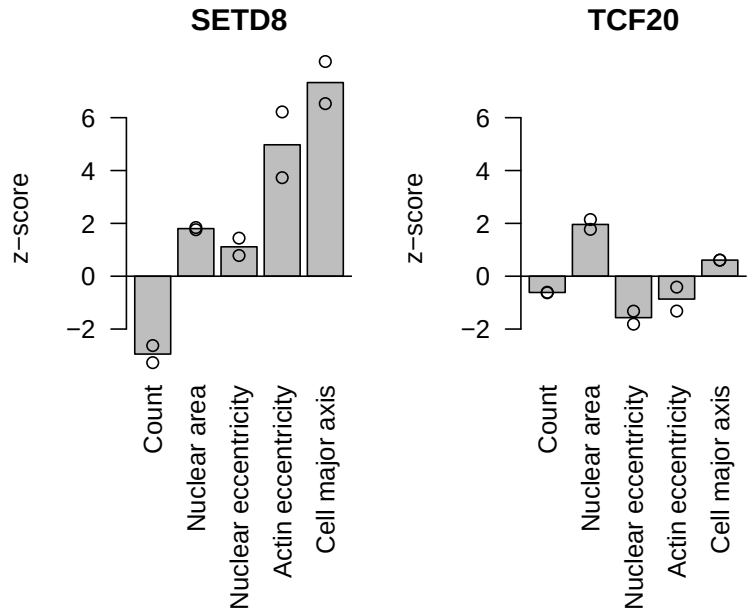
7.2 Examples of single knock down phenotypes

Main effects were first summarized over all four siRNA design pairs and divided by the median deviation to compute a z-score. Bars in the following barplots show the mean of main effects calculated over the two replicates of the screen. Circle symbols represent individual replicate measurements. Main effects for five different phenotypes are plotted for the two genes SETD8 and TCF20.

```
> features = c("count", "nuc.0.s.area","nuc.0.m.eccentricity",
+              "cell.Bact.m.eccentricity","cell.act.m.majoraxis")
> ylim = c(-2.947124, 7.329518)
```

```
> HD2013SGI:::plotExampleSingleGeneEffects(gene="SETD8",features,mainEffects,ylim)
```

```
> HD2013SGI:::plotExampleSingleGeneEffects(gene="TCF20",features,mainEffects,ylim)
```

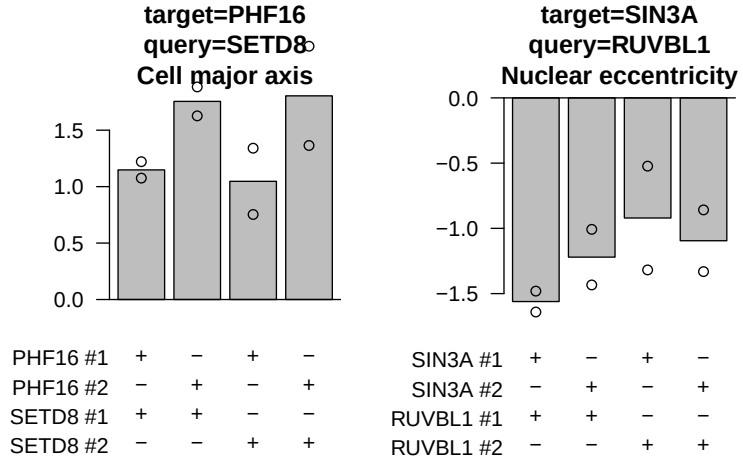


7.3 Example of genetic interactions

Barcharts for example interactions are plotted.

```
> HD2013SGI:::plotExampleInteractions(feature="cell.act.m.majoraxis",
+                                     target="PHF16",query="SETD8",
+                                     Interactions, mainEffects)
```

```
> HD2013SGI:::plotExampleInteractions(feature="nuc.0.m.eccentricity",
+                                     target="SIN3A",query="RUVBL1",
+                                     Interactions, mainEffects)
```



7.4 Overlap of interactions

To show the overlap of interactions called in different phenotypes, interactions with an adjusted p-value of ≤ 0.01 were considered.

```
> features = c("nuc.0.s.area","nuc.0.m.majoraxis",
+              "cell.Bact.m.eccentricity","cell.act.m.majoraxis")
> SIG = Interactions$padj[,,,,features] <= 0.01
> SIG[is.na(SIG)] = FALSE
```

The 5-dimensional array SIG of Boolean values indicating the significant interactions was reshaped to a 2-dimensional table with one column for each feature.

```
> dim(SIG) = c(prod(dim(SIG)[1:4]), dim(SIG)[5])
> row.names(SIG) = sprintf("A%d",seq_len(nrow(SIG)))
> colnames(SIG) = HD2013SGI:::humanReadableNames[features]
```

A contingency table was computed considering all gene pairs that interacted in at leat one phenotype.

```
> I = which(apply(SIG, 1, any))
> Overlap = table(as.data.frame(SIG[I,]))
> Overlap
```

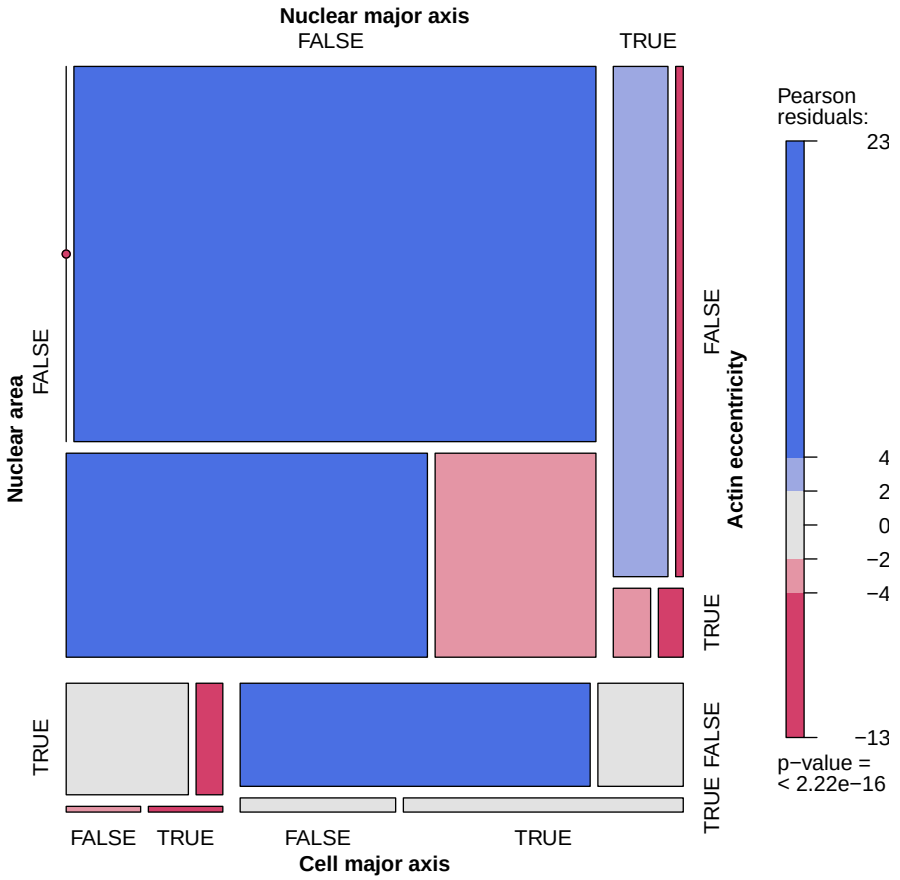
```
, , Actin eccentricity = FALSE, Cell major axis = FALSE

            Nuclear major axis
Nuclear area FALSE TRUE
       FALSE     0   65
       TRUE     32   82

, , Actin eccentricity = TRUE, Cell major axis = FALSE

            Nuclear major axis
Nuclear area FALSE TRUE
       FALSE   155    6
       TRUE      1    5

, , Actin eccentricity = FALSE, Cell major axis = TRUE

            Nuclear major axis
Nuclear area FALSE TRUE
       FALSE   412    9
       TRUE      7   20

, , Actin eccentricity = TRUE, Cell major axis = TRUE

            Nuclear major axis
Nuclear area FALSE TRUE
       FALSE    69    4
       TRUE      1    9
```

```
> save(Overlap, file=file.path("result","Figures","Overlap.rda"))
```

A Venn diagram was generated with a webtool (http://bioinfogp.cnb.csic.es/tools/venny/index.html). Lists of identifiers of the significant genes were written to be used with the webtool.

```
> L = apply(SIG, 2, function(x) names(which(x)))
> for (i in seq_along(L)) {
+   writeLines(L[[i]],
+     file.path("result", "Figures", sprintf("interactionlist%d.txt",i)))
+ }
```

In paper, a Venn diagram is shown. An alternative plot where the region areas are proportional to the numbers, is a mosaic plot as shown here.



8 Correlation of interaction profiles for different siRNA designs of the same gene

8.1 Preliminaries

```
> library("HD2013SGI")
> data("Interactions",package="HD2013SGI")
> data("mainEffects",package="HD2013SGI")
> dir.create(file.path("result","Figures"), recursive=TRUE)
> dir.create(file.path("result","data"), recursive=TRUE)
```

8.2 Correlation of interaction profiles for different siRNA designs of the same gene

π-scores were summarized over two replicates. Pearson correlation coefficients were computed for interaction profiles for each pair of siRNA and saved in the matric `C`.

```
> PI = Interactions$piscore
> for (k in seq_len(dim(PI)[5])) {
+   PI[,,,,k,] = PI[,,,,k,] / Interactions$scale[k]
+ }
> PI = (PI[,,,,,1]+PI[,,,,,2]) / 2
> PI[is.na(Interactions$padj)] = NA_real_
> PI2 = PI
```

```
> dim(PI2) = c(prod(dim(PI)[1:2]),prod(dim(PI)[3:5]))
> C = cor(t(PI2),use="pairwise.complete")
```

Correlation coefficients for siRNA pairs, where both siRNA targeted the same gene, were extracted.

```
> IDX = matrix(NA_integer_, nr=dim(Interactions$piscore)[1],nc=2)
> IDX[,1] = seq_len(dim(Interactions$piscore)[1])
> IDX[,2] = IDX[,1] + dim(Interactions$piscore)[1]
> cc = C[IDX]
```

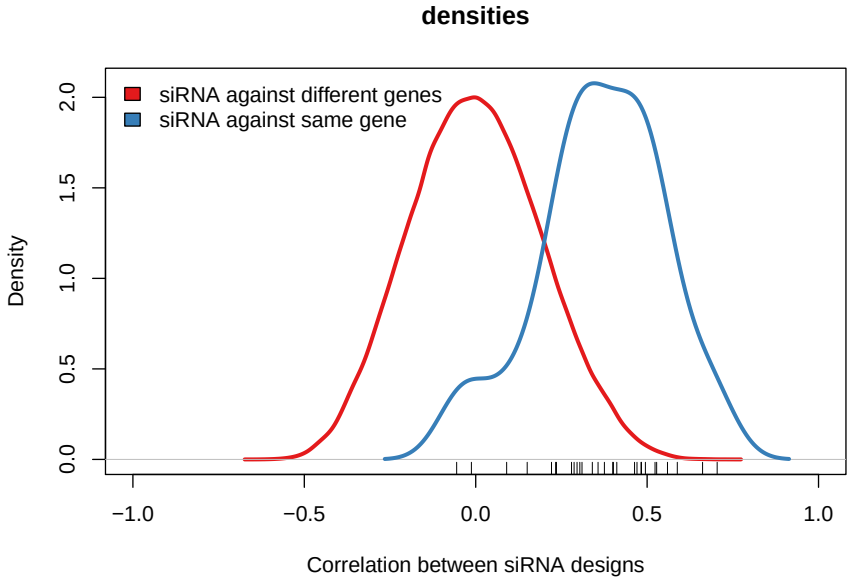
Next, we counted the number of significant interactions per target gene to select genes that showed a strong interaction profile.

```
> PI3 = PI
> dim(PI3) = c(dim(PI)[1],dim(PI)[2],prod(dim(PI)[3:5]))
> PI3[abs(PI3) < 5] = NA_real_
> S = sign(PI3)
> nrOfInteractionsPerTarget = apply(!is.na(S[,1,] == S[,2,]),1,sum,na.rm=TRUE)
```

We plot the density of correlation coefficients of interaction profiles between siRNAs against different genes and between siRNAs targeting the same gene.

```
> differentGenes = upper.tri(C)
> differentGenes[IDX] = FALSE
> differentGenes[IDX[,2:1]] = FALSE
> sel = which(nrOfInteractionsPerTarget >= 5)
```

```
> multidensity(list(againstDifferentGenes = C[differentGenes],
+                   againstSameGene = cc[sel]),
+              legend = list(
+                x = "topleft",
+                legend = c("siRNA against different genes",
+                           "siRNA against same gene"),
+                fill = brewer.pal(9,"Set1")[1:2],bty="n"),
+              xlim=c(-1,1),xlab="Correlation between siRNA designs",lwd=3)
> rug(cc[sel])
```



Save the number of interactions per target gene.

```
> save(nrOfInteractionsPerTarget,
+      file=file.path("result","data","nrOfInteractionsPerTarget.rda"))
```

9 Heatmaps of interaction profiles

9.1 Preliminaries

```
> library("HD2013SGI")
> data("Interactions",package="HD2013SGI")
> data("mainEffects",package="HD2013SGI")
> data("nrOfInteractionsPerTarget",package="HD2013SGI")
> dir.create(file.path("result","Figures"), recursive=TRUE)
```

9.2 Heatmap of all siRNA profiles

π-scores were normalized per feature by dividing with the median deviation between two replicates. For each siRNA-pair they were summarized over two replicates.

```
> PI = Interactions$piscore
> for (k in seq_len(dim(PI)[5])) {
+   PI[,,,,k,] = PI[,,,,k,] / Interactions$scale[k]
+ }
> PI = (PI[,,,,,1]+PI[,,,,,2]) / 2
```

The 5-dimensional array `PI` was reshaped to a 3-dimensional array by flattening the dimensions of target genes and siRNA designs as well as query genes and their respective siRNA designs.

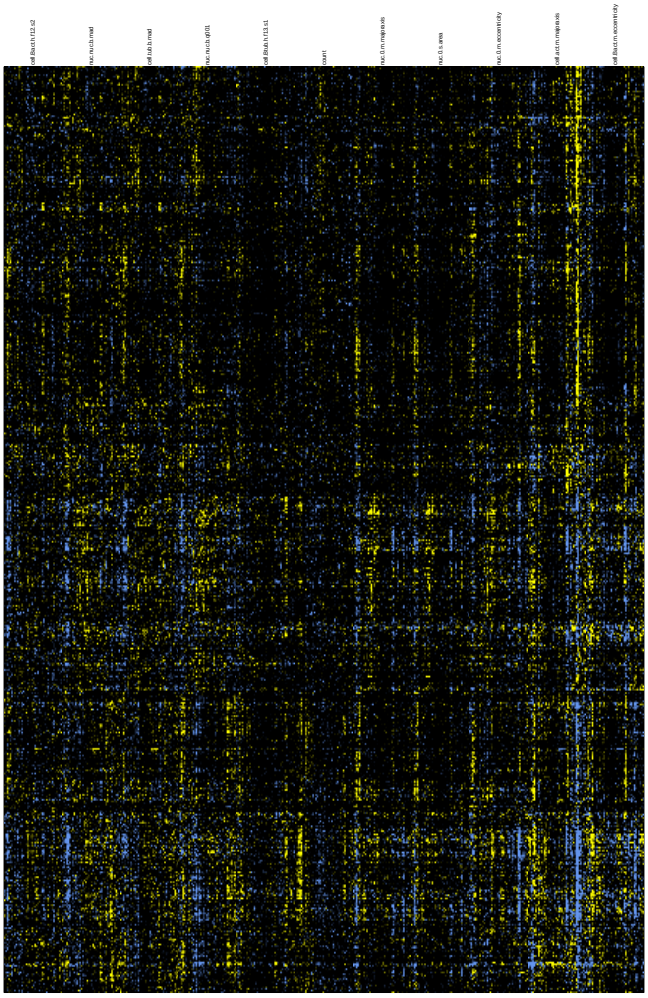
```
> dim(PI) = c(prod(dim(PI)[1:2]),prod(dim(PI)[3:4]),dim(PI)[5])
> dimnames(PI) = list(
+   sprintf("%s_%d",rep(Interactions$Anno$target$Symbol,times=2),
+           rep(seq_len(dim(Interactions$piscore)[2]),
+               each=dim(Interactions$piscore)[1])),
+   sprintf("%s_%d",rep(Interactions$Anno$query$Symbol,times=2),
+           rep(seq_len(dim(Interactions$piscore)[4]),
+               each=dim(Interactions$piscore)[3])),
+   rep(dimnames(Interactions$piscore)[[5]]))
```

After this step, the dimension of the array `PI` was 564 target siRNA $\times$ 40 query siRNA $\times$ 11 phenotypic features. The color scale for interaction scores was limited to the range $[-6, \ldots, 6]$. Interaction scores between -2 and 2 are colored black.

```
> cuts = c(-Inf,
+          seq(-6, -2, length.out=(length(HD2013SGI:::colBY)-3)/2),
+          0,
+          seq( 2,  6, length.out=(length(HD2013SGI:::colBY)-3)/2),
+          +Inf)
```

A heatmap of π-scores for all siRNA pairs is plotted.

```
> I = HD2013SGIorderDim(PI,1)$order
> J = HD2013SGIorderDim(PI,2)$order
> K = HD2013SGIorderDim(PI,3)$order
> HD2013SGIHeatmapHuman(x=PI[I,J,K],cuts=cuts,col=HD2013SGI:::colBY,
+                 colnames=TRUE,mcol=10,cexcol=0.5,mrow=0)
```



9.3 Heatmap of best correlating siRNA profiles

π-scores were normalized per feature by dividing by the median deviation between two replicates. For each gene pair π-scores were summarized over the two replicates and the 2×2 siRNA designs.

```
> PI = Interactions$piscore
> for (k in seq_len(dim(PI)[5])) {
+   PI[,,,,k,] = PI[,,,,k,] / Interactions$scale[k]
+ }
> PI = (PI[,,,,,1]+PI[,,,,,2]) / 2
> PI = (PI[,1,,1,] + PI[,2,,1,] + PI[,1,,2,] + PI[,2,,2,])/4
```

Target genes with a minimum of 5 π-scores with $|\pi_{ij}| \geq 5$ were selected.

```
> sel = which(nrOfInteractionsPerTarget >= 5)
> PI = PI[sel,,]
```

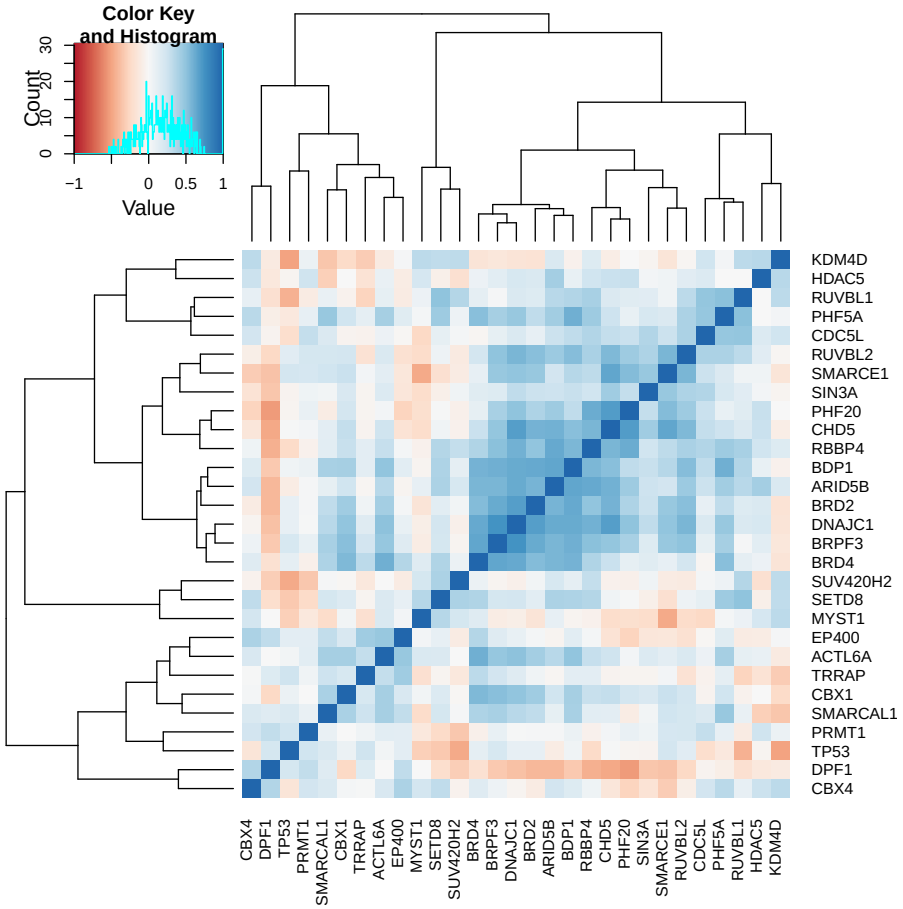
Pearson correlation coefficients were computed comparing interaction profiles between all gene pairs.

```
> hcAll = HD2013SGIorderDim(PI,1)
> dim(PI) = c(dim(PI)[1],prod(dim(PI)[2:3]))
> dimnames(PI)[[1]] = Interactions$Anno$target$Symbol[sel]
> C2 = cor(t(PI),use="pairwise.complete")
```

The heatmap of correlation coefficients is plotted.

```
> dd = as.dendrogram(hcAll)
> heatmap.2(C2, col=colorRampPalette(brewer.pal(9,"RdBu"))(255),
```

```
+             breaks=seq(-1,1,length.out=256),
+             trace="none",Rowv=dd,Colv=dd)
```



10 Simulation of small cell number data

10.1 Preliminaries

```
> library("HD2013SGI")
> dir.create(file.path("result","Figures"),recursive=TRUE,showWarnings=FALSE)
> data("featuresPerWell", package="HD2013SGI")
> data("TargetAnnotation",package="HD2013SGI")
> data("QueryAnnotation",package="HD2013SGI")
> data("datamatrix", package="HD2013SGI")
```

10.2 Convert data from plate order to SGI-array

The simulation of a genetic interaction experiment with smaller cell numbers followed the scripts in Sections 5 and 6.

First, the plate barcodes were parsed.

```
> plates = featuresPerWell$Anno[seq(1,nrow(featuresPerWell$Anno),
+                                   by=NFIELD*NCOL*NROW),"plate"]
> PlateAnnotation = HD2013SGI:::parsePlateBarcodes(plates)
```

The names of all query genes were extracted.

```
> S = which(PlateAnnotation$queryGroup=="sample")
> tdnames = unique(PlateAnnotation$targetDesign[S])
> qnames = unique(PlateAnnotation$queryGene[S])
> qdnames = unique(PlateAnnotation$queryDesign[S])
> repnames = unique(PlateAnnotation$replicate[S])
```

The data were converted from the screen plate layout to a multi-dimensional array.

```
> D = array(0.0, dim=c(field=NFIELD,col=NCOL,row=NROW,
+                      features=dim(featuresPerWell$data)[2],
+                      targetDesign=length(tdnames),
+                      query=length(qnames),queryDesign=length(qdnames),
+                      replicate=length(repnames)))
> dimnames(D) = list(field=seq_len(NFIELD),
+                    col=seq_len(NCOL),row=LETTERS[seq_len(NROW)+1],
+                    features=dimnames(featuresPerWell$data)[[2]],
+                    targetDesign=tdnames,
+                    queryGene=qnames,queryDesign=qdnames,replicate=repnames)
> z=0
> for (td in tdnames) {
+   for (q in qnames) {
+     for (qd in qdnames) {
+       for (r in repnames) {
+         plate = PlateAnnotation$plate[
+                       which((PlateAnnotation$targetDesign == td) &
+           (PlateAnnotation$queryGene == q) &
+           (PlateAnnotation$queryDesign == qd) &
+           (PlateAnnotation$replicate == r) ) ]
+         z=z+1
+         I = which(featuresPerWell$Anno$plate == plate)
+         D[,,,,td,q,qd,r] = as.vector(featuresPerWell$data[I,])
+       }
+     }
+   }
+ }
> D[is.na(D)] = 0.0
```

The data were summarized by their mean value

1. considering only one imaging field per well,
2. considering two imaging fields per well, and
3. considering all four imaging fields per well.

```
> Dsub1 = D[1,,,,,,,]
> Dsub2 = (D[1,,,,,,,] + D[2,,,,,,,])/2
> D = (D[1,,,,,,,] + D[2,,,,,,,] + D[3,,,,,,,] + D[4,,,,,,,])/4
> D = aperm(D,c(1,2,4,5,6,3,7))
> dn = dimnames(D)
> dim(D) = c(prod(dim(D)[1:2]),dim(D)[3:7])
> dimnames(D) = c(list(targetGene =
+                        sprintf("%s%d",rep(LETTERS[seq_len(NROW)+1],each=NCOL),
+                                       rep(seq_len(NCOL),times=NROW))),
+                     dn[3:7])
> Dsub1 = aperm(Dsub1,c(1,2,4,5,6,3,7))
> dim(Dsub1) = c(prod(dim(Dsub1)[1:2]),dim(Dsub1)[3:7])
> dimnames(Dsub1) = dimnames(D)
> Dsub2 = aperm(Dsub2,c(1,2,4,5,6,3,7))
> dim(Dsub2) = c(prod(dim(Dsub2)[1:2]),dim(Dsub2)[3:7])
> dimnames(Dsub2) = dimnames(D)
> datamatrixfullsub = list(Dsub1 = Dsub1, Dsub2 = Dsub2, Dsub4 = D)
```

Get indices for different sets of experiments (samples, controls, ...).

```
> IndexSAMPLE = which(TargetAnnotation$group == "sample")
> IndexCTRL = which(TargetAnnotation$group == "SingleKDctrl")
> IndexNEG = which(TargetAnnotation$group == "negctrl")
> IndexSAMPLENEG = c(IndexSAMPLE, IndexNEG)
```

Set the color value for controls.

```
> colCTRL = rep("gray", nrow(TargetAnnotation))
> colCTRL[TargetAnnotation$group == "SingleKDctrl"] = "royalblue"
> colCTRL[TargetAnnotation$group == "negctrl"] = "red"
```

10.3 Transform features and screen normalization

A function for a generalized log transformation.

```
> logtrafo <- function(x,c) {
+   log2((x+sqrt(x^2+c^2))/2)
+ }
```

Transform all features with a generalized log transformation. A 3 percent quantile was used as an additive constant in the glog transformation. The same constant was used for the glog-transformation of all three data sets.

```
> for (i in seq_len(dim(D)[5])) {
+   m = quantile(D[,,,,i,],probs=0.03,na.rm=TRUE)
+   D[,,,,i,] = logtrafo(D[,,,,i,],m)
+   Dsub1[,,,,i,] = logtrafo(Dsub1[,,,,i,],m)
+   Dsub2[,,,,i,] = logtrafo(Dsub2[,,,,i,],m)
+ }
```

Normalize median per plate and feature to compensate for global differences between replicates and siRNA designs.

```
> normalize <- function(D) {
+   M = apply(D,c(2:6),median,na.rm=TRUE)
+   M2 = apply(M, c(2,4), mean)
+   M2 = rep(M2, times=8)
+   dim(M2) = dim(M)[c(2,4,1,3,5)]
+   M2 = aperm(M2,c(3,1,4,2,5))
+   M = M - M2
+   M = rep(M[], each=dim(D)[1])
+   dim(M) = dim(D)
+   D = D - M
+   D
+ }
> D <- normalize(D)
> Dsub1 <- normalize(Dsub1)
> Dsub2 <- normalize(Dsub2)
```

Subtract median and divide by median deviation per feature.

```
> for (i in 1:dim(D)[5]) {
+   D[,,,,i,] = (D[,,,,i,] - median(D[,,,,i,],na.rm=TRUE)) /
+     mad(D[,,,,i,],na.rm=TRUE)
+ }
> for (i in 1:dim(Dsub1)[5]) {
+   Dsub1[,,,,i,] = (Dsub1[,,,,i,] - median(Dsub1[,,,,i,],na.rm=TRUE)) /
+     mad(Dsub1[,,,,i,],na.rm=TRUE)
+ }
> for (i in 1:dim(Dsub2)[5]) {
+   Dsub2[,,,,i,] = (Dsub2[,,,,i,] - median(Dsub2[,,,,i,],na.rm=TRUE)) /
```

```
+     mad(Dsub2[,,,,i,],na.rm=TRUE)
+ }
```

10.4 Quality control of features

The dimension of the data cube before quality control is: 345 x 2 x 20 x 2 x 353 x 2 (targets x siRNA designs x queries x designs x features x replicates). Take the mean over all four siRNA design pairs and compute for each feature the Pearson correlation between first and second replicate.

```
> C = rep(NA_real_,dim(D)[5])
> D2 = (D[,1,,1,,] + D[,2,,1,,] + D[,1,,2,,] + D[,2,,2,,]) / 4
> D2sub1 = (Dsub1[,1,,1,,] + Dsub1[,2,,1,,] + Dsub1[,1,,2,,] + Dsub1[,2,,2,,]) / 4
> D2sub2 = (Dsub2[,1,,1,,] + Dsub2[,2,,1,,] + Dsub2[,1,,2,,] + Dsub2[,2,,2,,]) / 4
> for (i in 1:dim(D)[5]) {
+   C[i] = cor(as.vector(D2[IndexSAMPLE,,i,1]),as.vector(D2[IndexSAMPLE,,i,2]))
+ }
```

Select all features with a correlation of at least 0.6. The number of features passing quality control is 227 out of 353. The same features as in the main analysis were selected in all three simulated cases.

```
> I = which(C >= 0.6)
> D = D[,,,,I,,drop=FALSE]
> Dsub1 = Dsub1[,,,,I,,drop=FALSE]
> Dsub2 = Dsub2[,,,,I,,drop=FALSE]
> dim(D)
```

```
[1] 345   2  20   2 227   2
```

The dimension of the data cube is now: 345 x 2 x 20 x 2 x 227 x 2 (targets x siRNA designs x queries x designs x features x replicates).

10.5 Quality control of siRNA designs

For each target siRNA the phenotypic profiles were summarized over two query siRNA and two replicates.

```
> D1 = (D[,,,1,,1] + D[,,,1,,2] + D[,,,2,,1] + D[,,,2,,2])/4
```

For each target gene the Pearson correlation coefficient was computed for phenotypic profiles between the two siRNA designs separately for each feature. The mean of correlation coefficients over all features is reported for each target gene.

```
> Cdesign1 = rep(NA_real_,dim(D)[1])
> for (k in 1:dim(D)[5]) {
+   for (i in 1:dim(D)[1]) {
+     Cdesign1[i] = cor(as.vector(D1[i,1,,k]),as.vector(D1[i,2,,k]))
+   }
+   if ( k == 1) {
+     Cdesign1all = Cdesign1
+   } else {
+     Cdesign1all = Cdesign1all + Cdesign1
+   }
+ }
> Cdesign1all = Cdesign1all / dim(D)[5]
```

All gene with a correlation of the phenotypic profiles of at least 0.7 are selected for further analysis. The number of target genes passing quality control is 282 out of 323. Additionally, 7 negative control target siRNAs are selected. The same target genes as in the main analysis are selected for all three simulated cases.

```
> I = IndexSAMPLE[Cdesign1all[IndexSAMPLE] >= 0.7]
> I = c(I, IndexNEG)
> D = D[I,,,,,,drop=FALSE]
```

```
> Dsub1 = Dsub1[I,,,,,,drop=FALSE]
> Dsub2 = Dsub2[I,,,,,,drop=FALSE]
> TargetAnnotation = TargetAnnotation[I,]
```

The dimension of the data cube is now: 289 x 2 x 20 x 2 x 227 x 2 (targets x siRNA designs x queries x designs x features x replicates).

10.6 Selection of non-redundant features

```
> data(stabilitySelection, package="HD2013SGI")
> Sel = stabilitySelection$ratioPositive >= 0.5
```

A set of 11 features was considered to contain non-redundant information. The criteria for feature selection are saved and the features are selected. The same features as in the main analysis were selected for all three simulated datasets.

```
> D = D[,,,,stabilitySelection$selected[Sel],,drop=FALSE]
> dimnames(D)[[1]] = TargetAnnotation$Symbol
> dimnames(D)[[3]] = QueryAnnotation$Symbol
> Dsub1 = Dsub1[,,,,stabilitySelection$selected[Sel],,drop=FALSE]
> Dsub2 = Dsub2[,,,,stabilitySelection$selected[Sel],,drop=FALSE]
> dimnames(Dsub1) = dimnames(D)
> dimnames(Dsub2) = dimnames(D)
```

After selecting 11 features, the dimension of the data cube is now: 289 x 2 x 20 x 2 x 11 x 2 (targets x siRNA designs x queries x designs x features x replicates).

10.7 Pairwise interaction scores

Pairwise interaction scores (π-scores) are estimated using a robust linear fit. The query main effects are lifted such that they equal to the mean of the single knock down measurements of the query genes (target siRNA is a scrambled sequence serving as negative control).

```
> getInteractions <- function(D) {
+   pimatrix = datamatrix
+   pimatrix$D[] = NA_real_
+   mainEffects = list(target = D[,,1,,,],
+                      query = D[1,,,,,],
+                      overall = D[1,,1,,,],
+                      Anno = datamatrix$Anno)
+
+   for (i in 1:2) {
+     for (j in 1:2) {
+       for (k in 1:dim(D)[5]) {
+         for (l in 1:2) {
+           MP = HD2013SGImaineffects(D[,i,,j,k,l],
+               TargetNeg=which(TargetAnnotation$group == "negctrl"))
+           pimatrix$D[,i,,j,k,l] = MP$pi
+           mainEffects$target[,i,j,k,l] = MP$targetMainEffect
+           mainEffects$query[i,,j,k,l] = MP$queryMainEffect
+           mainEffects$overall[i,j,k,l] = MP$neg
+         }
+       }
+     }
+   }
+   D = pimatrix$D
+   PADJ = D[pimatrix$Anno$target$group == "sample",,,,,1]
+   s = rep(NA_real_, dim(D)[5])
+   for (i in 1:dim(D)[5]) {
```

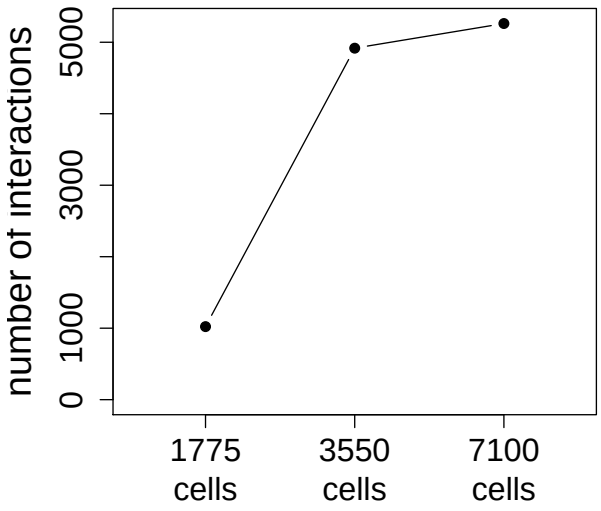
```
+     Data = D[,,,,i,]
+     Data = Data[pimatrix$Anno$target$group == "sample",,,,]
+     d = dim(Data)
+     dim(Data) = c(prod(d[1:4]),prod(d[5]))
+     Data[abs(Data[,1]-Data[,2]) >
+            4*mad(Data[,1]-Data[,2],center=0.0),] = NA_real_
+
+     s[i] = median(apply(Data,1,sd), na.rm=TRUE)
+     padj = rep(NA_real_, nrow(Data))
+     K = which(apply(!is.na(Data),1,all))
+     fit = eBayes(lmFit(Data[K,]))
+     padj[K] = p.adjust(fit$p.value, method="BH")
+     PADJ[,,,,i] = padj
+     cat("i=",i," nr int (1%) = ",sum(padj <= 0.01,na.rm=TRUE)/nrow(Data),
+         " nr int (3%) = ",sum(padj <= 0.03,na.rm=TRUE)/nrow(Data),"\n")
+   }
+   PI = pimatrix$D
+   PI = PI[pimatrix$Anno$target$group == "sample",,,,,]
+   Interactions = list(piscore = PI,
+                       scale = s,
+                       padj = PADJ,
+                       Anno = pimatrix$Anno)
+   Interactions$Anno$target = Interactions$Anno$target[
+               pimatrix$Anno$target$group == "sample",]
+   Interactions
+ }
> InteractionsSub4 = getInteractions(D)

i= 1  nr int (1%) =  0.0006205674  nr int (3%) =  0.006826241
i= 2  nr int (1%) =  0.02353723  nr int (3%) =  0.07539894
i= 3  nr int (1%) =  0.05110816  nr int (3%) =  0.1007535
i= 4  nr int (1%) =  0.06117021  nr int (3%) =  0.1077128
i= 5  nr int (1%) =  0.009973404  nr int (3%) =  0.03922872
i= 6  nr int (1%) =  0.008865248  nr int (3%) =  0.02451241
i= 7  nr int (1%) =  0.05110816  nr int (3%) =  0.1058067
i= 8  nr int (1%) =  0.00695922  nr int (3%) =  0.02234043
i= 9  nr int (1%) =  0.005673759  nr int (3%) =  0.01710993
i= 10  nr int (1%) =  0.01108156  nr int (3%) =  0.0454344
i= 11  nr int (1%) =  0.003147163  nr int (3%) =  0.03843085

> InteractionsSub1 = getInteractions(Dsub1)

i= 1  nr int (1%) =  0  nr int (3%) =  0.0005319149
i= 2  nr int (1%) =  0.008510638  nr int (3%) =  0.02832447
i= 3  nr int (1%) =  0.01023936  nr int (3%) =  0.02593085
i= 4  nr int (1%) =  0.008998227  nr int (3%) =  0.0206117
i= 5  nr int (1%) =  0.0006205674  nr int (3%) =  0.004166667
i= 6  nr int (1%) =  0.0006648936  nr int (3%) =  0.003679078
i= 7  nr int (1%) =  0.009219858  nr int (3%) =  0.0266844
i= 8  nr int (1%) =  0  nr int (3%) =  0.000177305
i= 9  nr int (1%) =  0  nr int (3%) =  0.0002216312
i= 10  nr int (1%) =  0  nr int (3%) =  0
i= 11  nr int (1%) =  0.007047872  nr int (3%) =  0.025

> InteractionsSub2 = getInteractions(Dsub2)

i= 1  nr int (1%) =  0.0009308511  nr int (3%) =  0.004964539
i= 2  nr int (1%) =  0.01830674  nr int (3%) =  0.05771277
i= 3  nr int (1%) =  0.06387411  nr int (3%) =  0.09968972
i= 4  nr int (1%) =  0.02814716  nr int (3%) =  0.06023936
i= 5  nr int (1%) =  0.00483156  nr int (3%) =  0.02132092
```

```
i= 6  nr int (1%) =  0.00625  nr int (3%) =  0.01586879
i= 7  nr int (1%) =  0.05820035  nr int (3%) =  0.09933511
i= 8  nr int (1%) =  0.008244681  nr int (3%) =  0.02109929
i= 9  nr int (1%) =  0.02318262  nr int (3%) =  0.03679078
i= 10  nr int (1%) =  0  nr int (3%) =  0.009042553
i= 11  nr int (1%) =  0.005984043  nr int (3%) =  0.0366578
```

The number of significantly interacting siRNA pairs is plotted.

```
> field1 = sum(InteractionsSub1$padj <= 0.01,na.rm=TRUE)
> field2 = sum(InteractionsSub2$padj <= 0.01,na.rm=TRUE)
> field4 = sum(InteractionsSub4$padj <= 0.01,na.rm=TRUE)
```

```
> plot(c(field1,field2,field4),type="b",pch=19,
+      ylab="number of interactions",xlab="",
+      xlim=c(0.5,3.5),ylim=c(0.0,field4),
+      main="",xaxt="n",cex.lab=1.75,cex.axis=1.5)
> axis(side=1,at=1:3,labels=c("","",""))
> axis(side=1,at=1:3,labels=c("1775\ncells","3550\ncells","7100\ncells"),
+      line=1.5,lwd=0,cex.axis=1.5)
```



11 Distribution of positive and negative interactions

11.1 Preliminaries

```
> library("HD2013SGI")
> data("Interactions",package="HD2013SGI")
> data("mainEffects",package="HD2013SGI")
> data("nrOfInteractionsPerTarget",package="HD2013SGI")
> dir.create(file.path("result","Figures"),showWarnings=FALSE,recursive=TRUE)
```

11.2 Distribution of interactions

The mean of the interaction scores was taken over the two replicates.

```
> PI = Interactions$piscore
> PI = (PI[,,,,,1]+PI[,,,,,2]) / 2
```

The number of positive and negative interactions for each phenotypic feature was counted. Only significant interactions (FDR <= 0.01) were considered.

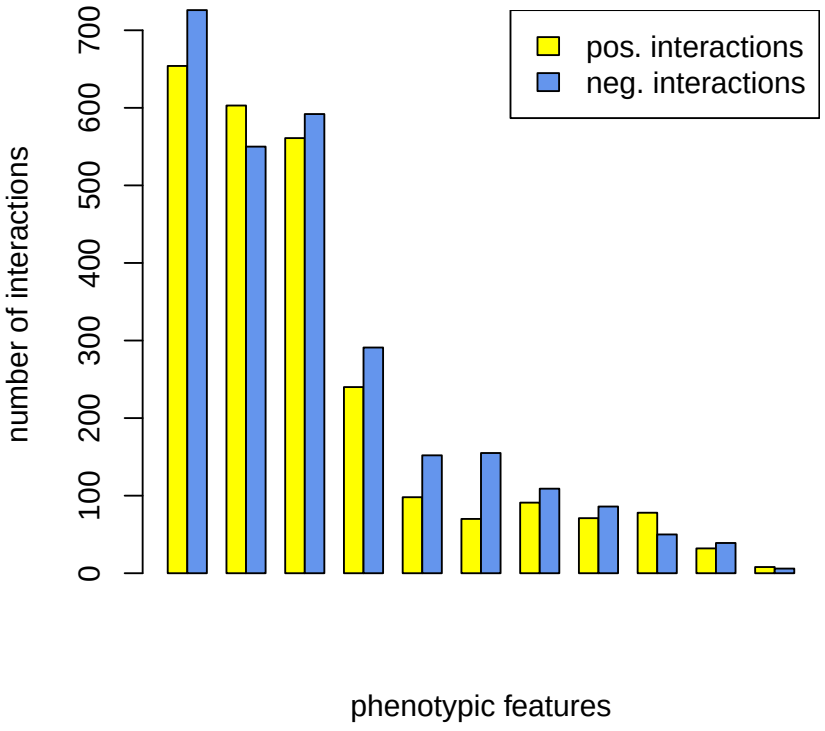
```
> npos = rep(NA_integer_, dim(PI)[5])
> nneg = rep(NA_integer_, dim(PI)[5])
> for (i in seq_len(dim(PI)[5])) {
+   I = which(Interactions$padj[,,,,i] <= 0.01)
+   pi = (PI[,,,,i])[I]
+   npos[i] = sum(pi >= 0)
+   nneg[i] = sum(pi < 0)
+ }
```

npos (nneg) contains the number of positive (negative) genetic interactions per phenotypic feature. The features were sorted by the total number (positive and negative) interactions starting with the largest number.

```
> I = order(-(npos+nneg))
> npos = npos[I]
> nneg = nneg[I]
```

A barchart was plotted showing the number of positive and negative interactions.

```
> barplot(rbind(npos,nneg),beside=TRUE,
+         col=HD2013SGI:::Colors[c(3,1)],
+         xlab="phenotypic features",ylab="number of interactions")
> legend("topright",legend=c("pos. interactions", "neg. interactions"),
+        fill=HD2013SGI:::Colors[c(3,1)])
```



12 Scatter plots between phenotypes

12.1 Preliminaries

```
> library("HD2013SGI")
> data("datamatrix",package="HD2013SGI")
> dir.create(file.path("result","Figures"),showWarnings=FALSE,recursive=TRUE)
```

12.2 Scatter plots of selected features

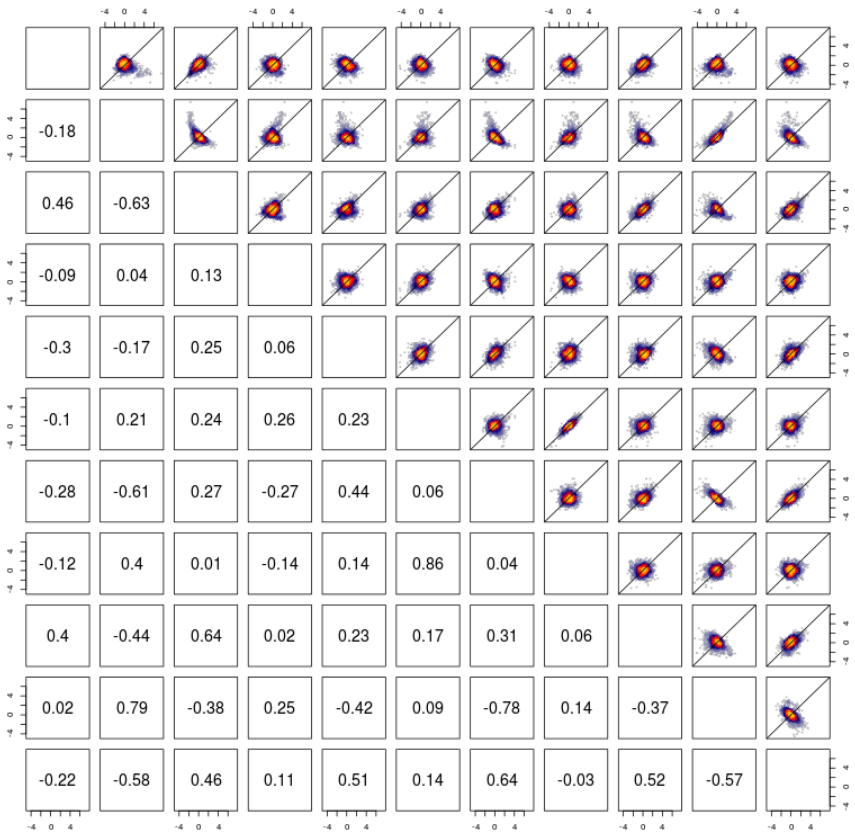
The 6-dimensional array of log-transformed features is reshaped in a 2-dimensional matrix with columns for each selected feature.

```
> D = datamatrix$D
> D = aperm(D, c(1,2,3,4,6,5))
> dim(D) = c(prod(dim(D)[1:5]),dim(D)[6])
```

Scatter plots of the selected features are plotted using 1000 randomly selected double knock-down experiments.

```
> set.seed(712608)
> S = sample(dim(D)[1],1000)
```

```
> heatpairs(D[S,], cor.cex = 1.8,pch=20,main="")
```



13 Screen plots

13.1 Preliminaries

```
> library("HD2013SGI")
> data("Interactions",package="HD2013SGI")
```

```
> dir.create(file.path("result","Figures"),
+            recursive=TRUE,showWarnings=FALSE)
```

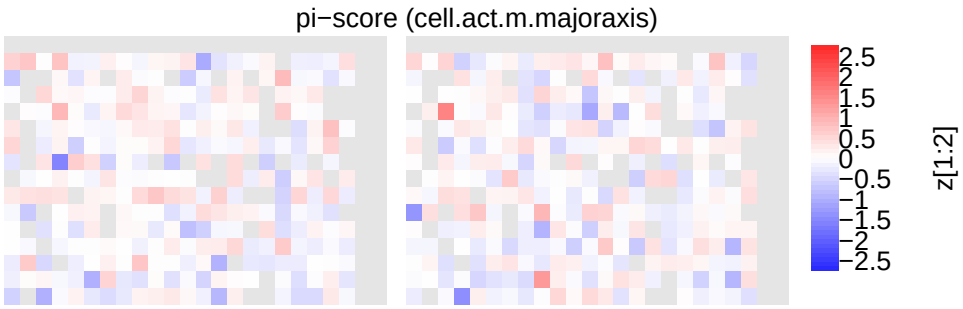
13.2 Screen plots

The π-scores for the feature `cell.act.m.majoraxis` were mapped back to the screen plate layout.

```
> F = "cell.act.m.majoraxis"
> EmptyPlate = rep(NA_real_, 24*16)
> names(EmptyPlate) = sprintf("%s%d",rep(LETTERS[1:16],each=24),
+                             rep(1:24,times=16))
> z = list()
> for (Rep in seq_len(dim(Interactions$piscore)[6])) {
+   for (Q in seq_len(dim(Interactions$piscore)[3])) {
+     for (QD in seq_len(dim(Interactions$piscore)[4])) {
+       for (TD in seq_len(dim(Interactions$piscore)[2])) {
+         Plate = EmptyPlate
+         Plate[Interactions$Anno$target$Well] =
+                 Interactions$piscore[,TD,Q,QD,F,Rep]
+         z[[sprintf("Q%d_td%d_qd%d_r%d",Q,TD,QD,Rep)]] = Plate
+       }
+     }
+   }
+ }
```

Screen plots of two plates are plotted.

```
> plotScreen(z[1:2], ncol=2L, na.fill="gray90",
+            main="pi-score (cell.act.m.majoraxis)",
+            zrange=c(-3,3),do.names=FALSE)
```



14 Main result table

14.1 Preliminaries

```
> library("HD2013SGI")
> data("Interactions",package="HD2013SGI")
> dir.create(file.path("result","Tables"),
+            recursive=TRUE,showWarnings=FALSE)
```

14.2 Main table of interaction scores

The mean of the interaction scores is taken over the two replicates.

```
> PI = Interactions$piscore
> PI = (PI[,,,,,1] + PI[,,,,,2])/2
> PADJ = Interactions$padj
> PI[is.na(PADJ)] = NA
```

The 5-dimensional arrays of π-scores and adjusted p-values are reshaped to a 2-dimensional matrix with one column per phenotypic feature.

```
> dim(PI) = c(prod(dim(PI)[1:4]),dim(PI)[5])
> dim(PADJ) = c(prod(dim(PADJ)[1:4]),dim(PADJ)[5])
```

The two matrices are merged in a way such that π-scores and adjusted p-values are interlaced and the two columns per features are next to each other.

```
> V = cbind(PI, PADJ)
> V = V[,rep(seq_len(dim(PI)[2]),each=2)+rep(c(0,dim(PI)[2]),times=dim(PI)[2])]
> colnames(V) = sprintf("%s.%s",rep(c("pi-score","padj"),times=dim(PI)[2]),
+         rep(HD2013SGI:::humanReadableNames[ Interactions$Anno$phenotype ],
+                          each=2))
```

Annotation of the target and query gene names and the index of siRNA design are added to the table. The table is written to a text file.

```
> target =       rep(Interactions$Anno$target$Symbol,
+                    times=prod(dim(Interactions$piscore)[c(2,3,4)]))
> targetDesign = rep(rep(c("#1","#2"),
+                    times=prod(dim(Interactions$piscore)[c(3,4)])),
+                    each=dim(Interactions$piscore)[1])
> query =        rep(rep(Interactions$Anno$query$Symbol,
+                    times=dim(Interactions$piscore)[4]),
+                    each=prod(dim(Interactions$piscore)[c(1,2)]))
> queryDesign =  rep(c("#1","#2"),
+                    each=prod(dim(Interactions$piscore)[c(1,2,3)]))
> df =  data.frame(targetGene=target, targetDesign = targetDesign,
+                  queryGene=query, queryDesign=queryDesign,
+                  V)
> write.table(df, file=file.path("result","Tables","interactions.txt"),sep="\t",
+             row.names=FALSE,quote=FALSE)
```

15 Session info

Here is the output of `sessionInfo` on the system on which this document was compiled:

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

- R version 3.1.1 Patched (2014-09-25 r66681), x86_64-unknown-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, stats4, utils
- Other packages: AnnotationDbi 1.28.0, Biobase 2.26.0, BiocGenerics 0.12.0, EBImage 4.8.0, GenomeInfoDb 1.2.0, HD2013SGI 1.5.1, IRanges 2.0.0, LSD 2.5, MASS 7.3-35, RColorBrewer 1.0-5, S4Vectors 0.4.0, XML 3.98-1.1, annotate 1.44.0, colorRamps 2.3, ellipse 0.3-8, geneplotter 1.44.0, gplots 2.14.2, gtools 3.4.1, lattice 0.20-29, limma 3.22.0, schoolmath 0.4, splots 1.32.0, vcd 1.3-2
- Loaded via a namespace (and not attached): BiocStyle 1.4.1, DBI 0.3.1, KernSmooth 2.23-13, RSQLite 0.11.4, abind 1.4-0, bitops 1.0-6, caTools 1.17.1, colorspace 1.2-4, gdata 2.13.3, jpeg 0.1-8, locfit 1.5-9.1, png 0.1-7, tiff 0.1-5, tools 3.1.1, xtable 1.7-4

References

[1] A.E. Carpenter, T.R. Jones, M.R. Lamprecht, C. Clarke, I.H. Kang, O. Friman, D.A. Guertin, J.H. Chang, R.A. Lindquist, J. Moffat, et al. CellProfiler: image analysis software for identifying and quantifying cell phenotypes. *Genome Biology*, 7(10):R100, 2006.

[2] F. Fuchs, G. Pau, D. Kranz, O. Sklyar, C. Budjan, S. Steinbrink, T. Horn, A. Pedal, W. Huber, and M. Boutros. Clustering phenotype populations by genome-wide RNAi and multiparametric imaging. *Molecular Systems Biology*, 6(1), 2010.

[3] M. Held, M.H.A. Schmitz, B. Fischer, T. Walter, B. Neumann, M.H. Olma, M. Peter, J. Ellenberg, and D.W. Gerlich. CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging. *Nature Methods*, 7(9):747–754, 2010.

[4] G. Pau, F. Fuchs, O. Sklyar, M. Boutros, and W. Huber. EBImage - an R package for image processing with applications to cellular phenotypes. *Bioinformatics*, 26(7):979–981, 2010.

[5] T. Jones, A. Carpenter, and P. Golland. Voronoi-based segmentation of cells on image manifolds. *Computer Vision for Biomedical Image Applications*, pages 535–543, 2005.

[6] R. Mani, R.P. St Onge, J.L. Hartman, G. Giaever, and F.P. Roth. Defining genetic interaction. *Proceedings of the National Academy of Sciences*, 105(9):3461, 2008.

[7] M. Schuldiner, S.R. Collins, N.J. Thompson, V. Denic, A. Bhamidipati, T. Punna, J. Ihmels, B. Andrews, C. Boone, J.F. Greenblatt, et al. Exploration of the function and organization of the yeast early secretory pathway through an epistatic miniarray profile. *Cell*, 123(3):507–519, 2005.

[8] M. Costanzo, A. Baryshnikova, J. Bellay, Y. Kim, E.D. Spear, C.S. Sevier, H. Ding, J.L.Y. Koh, K. Toufighi, S. Mostafavi, et al. The genetic landscape of a cell. *Science*, 327(5964):425, 2010.

[9] A. Baryshnikova, M. Costanzo, Y. Kim, H. Ding, J. Koh, K. Toufighi, J.Y. Youn, J. Ou, B.J. San Luis, S. Bandyopadhyay, et al. Quantitative analysis of fitness and genetic interactions in yeast on a genome scale. *Nature Methods*, 7(12):1017–1024, 2010.

[10] S. Bandyopadhyay, M. Mehta, D. Kuo, M.K. Sung, R. Chuang, E.J. Jaehnig, B. Bodenmiller, K. Licon, W. Copeland, M. Shales, et al. Rewiring of genetic networks in response to DNA damage. *Science*, 330(6009):1385, 2010.

[11] T. Horn, T. Sandmann, B. Fischer, E. Axelsson, W. Huber, and M. Boutros. Mapping of signaling networks through synthetic genetic interaction analysis by RNAi. *Nature Methods*, 8(4):341–346, 2011.

[12] E. Axelsson, T. Sandmann, T. Horn, M. Boutros, W. Huber, and B. Fischer. Extracting quantitative genetic interaction phenotypes from matrix combinatorial RNAi. *BMC Bioinformatics*, 12(1):342, 2011.

[13] W. Huber, A. Von Heydebreck, H. Sültmann, A. Poustka, and M. Vingron. Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(suppl 1):S96–S104, 2002.