scsR (seed correction for siRNA) Package Vignette

Andrea Franceschini

28 October 2014

1 INTRODUCTION

ScsR (Seed Correction for siRNA), is an R package that provides suitable methods to identify, visualize and correct for off-target effects in genome-wide siRNA screens. As it has been shown in many other works (1,2,3,5), siRNAs do have off-target effects (i.e. other genes are targeted apart those for which they have been designed for). Off-target effects are still very pronounced even in the newest siRNA libraries (3) and they are the cause of very high false positive rates in all the genome-wide siRNA screens (3,4,5). Off-target effects are mainly mediated by the seed sequence of the siRNA (2,3), that hence act in a miRNA-like fashion.
Both artificial siRNA and natural mature miRNAs are in fact RNA molecules of around 21 nucleotides and they act in a very similar way, being both incorporated in the RISC complex. To be more precise, the siRNA is double stranded but only one strand (the guide strand) can enter in the RISC and hence downregulate the transcripts (i.e. most of the companies, including QIAGEN, AMBION and DHARMACON introduced in 2005 a chemical modification that prevents the other strand to enter in the RISC. In our work we confirm that this chemical modification works very well (3) and that most of the off-targets are caused by the guide strand only).
ScsR tries to infer the seed-effect (and correct for it) using the siRNA genome-wide screen data. We will show in this tutorial real case examples of how scsR can get rid of the seed effect on real genome-wide screens.
Besides, scsR provides several innovative visualization and utility methods that can help the screener in visualizing and mining his data.
For example, only recently, certain groups have used information acquired from off-target effects to predict which human miRNAs can be effective on the phenotype under study (5). In this context, scsR offers several statistics that associate a probability value to each seed (so that the effect of the seed can be determined more reliably), and it maps the seeds automatically to the latest miRNA libraries.

ScsR works best with QIAGEN and AMBION single oligo (i.e. unpooled) genome-wide siRNAs screens and with continuos scores in z-scored form.

ScsR can also correct pooled siRNA libraries (e.g. DHARMACON), but with lower performances than correcting unpooled siRNAs screens.
We encourage our users to use our benchmark functions to test the results of the correction (and eventually detect possible errors or misusage of the package).

```
> library(scsR)
```

In order to use scsR you need a data frame with 3 columns:

1) GeneID (or Gene Name/identifier)

2) siRNA sequence

3) siRNA score

The name of these columns should be: GeneID, siRNA_seq and score. If you want to use different names then you need to set the appropriate variables when you call the various functions in this package.

In case you are using a pooled library, you have to add one line for every oligo in the pool (i.e. if geneX has 4 oligos pooled, you will put 4 rows in the data frame with the same gene identifier, but different oligo sequence).

If you come from Excel and/or Matlab you can generate a text file with the format specified above. Then you can use the following function in order to import the file in a data frame in R:
read.table(file_name, header=TRUE, fill=TRUE, quote="", stringsAsFactors=FALSE)

As an example, we use the analyzed data of two genome-wide siRNA screens on the UUKUNIEMI virus (5), performed with different siRNA libraries: QIAGEN unpooled and DHARMACON pooled. In both the screens Hela Cell Lines have been infected by the UUKUNIEMI virus and the data has been collected and analyzed using latest technologies. Two phenotypes can be inferred from this screen:
- infection inhibition (i.e. when the number of infected cells is reduced)
- infection enhancement (i.e. when the number infected cells increses)
- cell number reduction (i.e. when cell-death and/or block of cell proliferation occurs)
- cell number increase (i.e. when increase in cell proliferation occurs)
We added to this package the data for the infection phenotype (that we are going to use in the following examples). However we encrypted the siRNA sequences for copyright issues, in a way that preserves the relation seed vs sequences (i.e. the groups of sequences that share the same seed remain the same)

```
> data(uuk_screen)
> data(uuk_screen_dh)
> data(miRBase_20)

> head(uuk_screen)

  GeneID             siRNA_seq     score
1  59286 UCAGUUGAGGCACCUCCAAGG -1.962496
2  59286 UCUGCGGCGGGUACAAGUUCC -1.961901
3  59286 UCUUUACUAGGCAGUACCACC -1.838258
4   9793 UGUCCCGUCGGGACGGGUACU -2.050937
5   9793 UCGAACGCUUGUUGAGACUCU -1.995524
6   9793 UUGGCAGCCUGUCAUGAAACC -1.901085
```

We also provide a function that checks your input data frame for possible problems/inconsistencies.

```
> uuk_screen <- check_consistency(uuk_screen)
```

For example, it could also be that the library contains multiple instances of the same oligo sequences (i.e. replicates of the same oligo).We do suggest to condense this information, making the median of the score of the instances. For an unpooled library, you can use the following function:

```
> uuk_screen <- median_replicates(uuk_screen)
```

IMPORTANT NOTE:
If instead of the siRNA sequence, you have the target sequence of the siRNA (pay attention that many siRNA producers do provide target sequences instead of real siRNA sequences !!!), you must transcribe the target sequence in order to obtain the guide (antisense) strand of the siRNA sequence. In order to do that you can use the transcribe_seqs function in our package (look at the documentation for more details).

Then, you need to extract the seed from the sequence:

```
> uuk_screen <- add_seed(uuk_screen)

> uuk_screen_dh <- add_seed(uuk_screen_dh)
```

Finally you can analyze the screen and get information about the effective seeds.

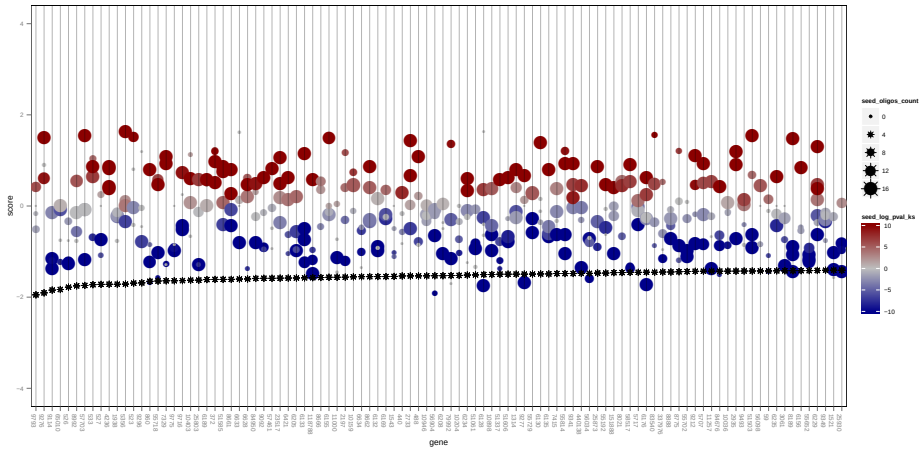
```
> seeds <- seeds_analysis(uuk_screen, miRBase = miRBase_20 )

> head(seeds)

    seed7   score count countHits ratiosHitsVsCount       pvalue     sd
1 CUGGUGC -1.3011    64        50            0.7812 9.820200e-38 0.6470
2 CUGGUGU -1.5642    40        34            0.8500 1.949432e-28 0.2948
3 CUGUUUA -1.7256    27        27            1.0000 9.593171e-28 0.2076
4 CGGUUCA -1.7386    29        28            0.9655 2.506075e-27 0.2546
5 CGUGUGC -1.6916    26        26            1.0000 9.623549e-27 0.1967
6 CAGCAGA -1.5824    31        28            0.9032 3.172958e-25 0.2489
         miRNA
1 hsa-miR-548q
2 hsa-miR-7978
3         <NA>
4         <NA>
5         <NA>
6  hsa-miR-922
```

As result, you get back a table containing one line for every seed that is found in your siRNA sequences. The number of oligos that share the seed sequence is reported, together with their average score and their standard deviation.
Besides we do provide two different probability values. These probability values reflect the probability that given a set of scores that refer to oligos that share the same seed, you can obtain such scores by chance.
The "pvalue-ks" uses a Kolmogorov Smirnov statistical test to compute the probability that the vector of oligo's score relative to a given seed can be drawned by chance from the big vector of all the possible scores of the genome-wide screen (tipically 80,000 oligos).
Differently, the "pvalue" column do report the p-value that is obtained using an hypergeometric test that computes the probability that you can obtain such a number of hits by chance (an "hit" is an oligo

having a score that is below a defined threshold, by default the score that limit the top 10% of the screen).
In case you are looking at the seed effect on the opposite direction (i.e. that causes the oligos to have positive z-scores instead of negative zscores) you should remember to set to TRUE the "enhancer" parameter, in order to get a correct p-value.
Finally, we do report in a dedicated column the miRNAs that contain the given seed, if any of those can be found in the provided miRBase version.

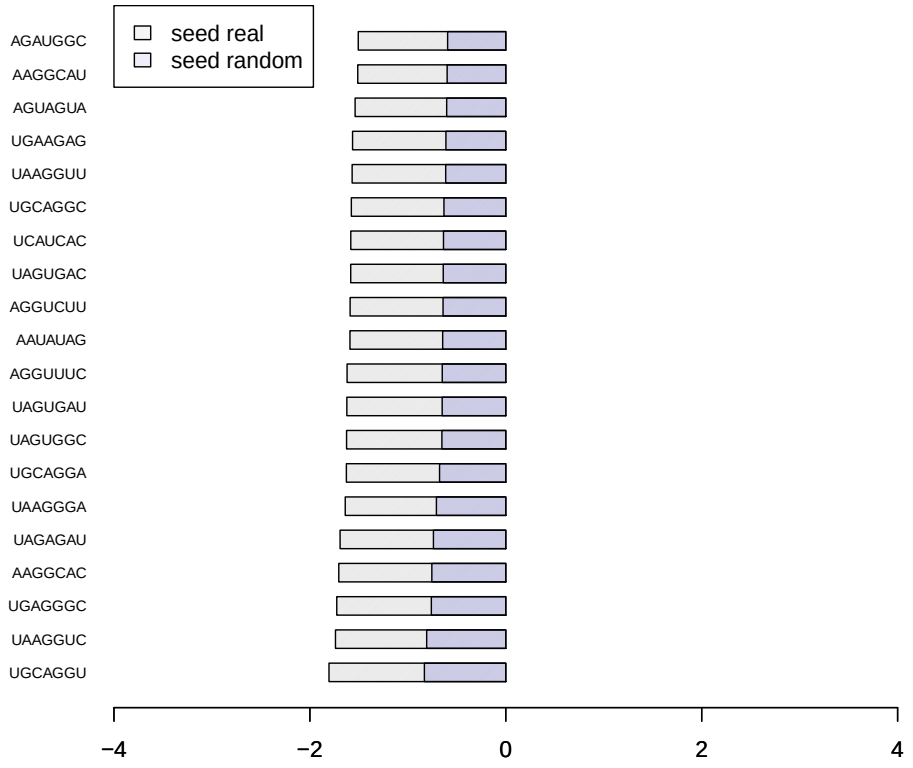
At this point, we can make use of suitable functions to inspect interesting properties of the screen and of its sequences.
We can easily plot the top 100 genes of the screen (sorted via median) and display the seed effect that shows up in the oligos relative to these genes.
In this plot the median score of each gene is represented as a black rhombus, while the seed effect of each oligo is represented as a colored circle (of size as big as the number of oligos that share the same seed).
This plot can be easily examined manually and can help the screener to distinguish (by eye) the false positive genes with respect to the true positives. Ideally a true positive gene should have all its seed to be NOT effective (i.e. around zero).

```
> plot_screen_hits(arrange(add_rank_col(uuk_screen), median))
```



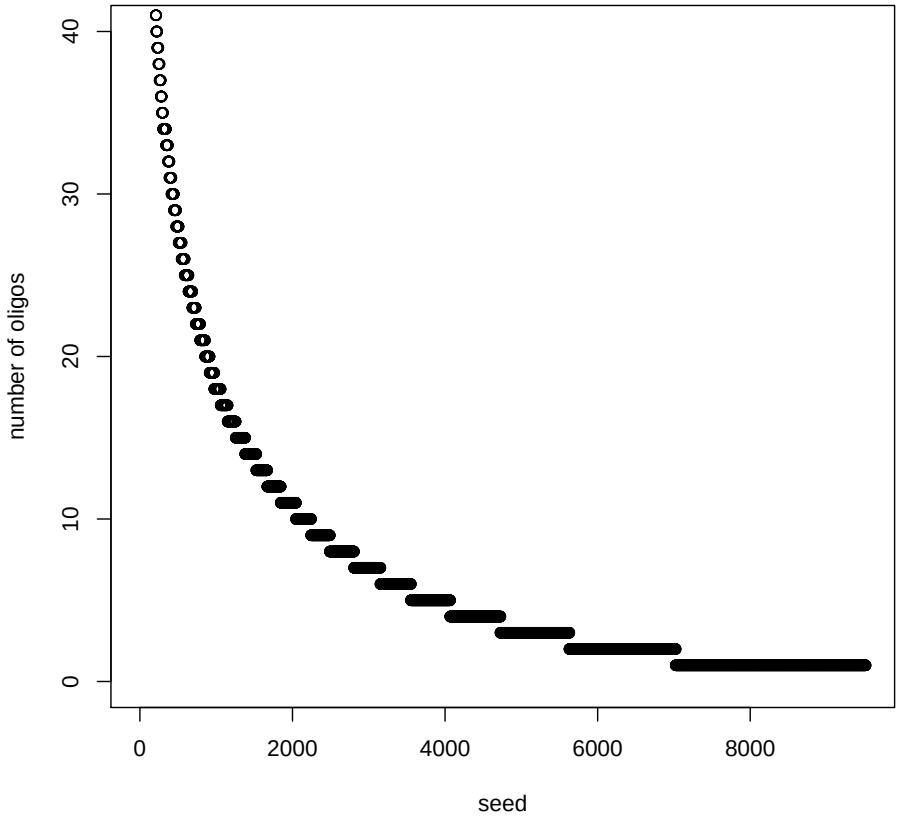
Then we can plot the most effective seeds of the screen, using the plot_effective_seeds_head function. In the histograms you can see two bars, that correspond to the average score of the oligos that share the same seed. The bars are sorted by this score (the white bar corresponds to a seed that is obtained when the score column of the screen is randomized, and it corresponds to the intensity of the signal that you would expect by chance).

```
> plot_effective_seeds_head(uuk_screen)
```



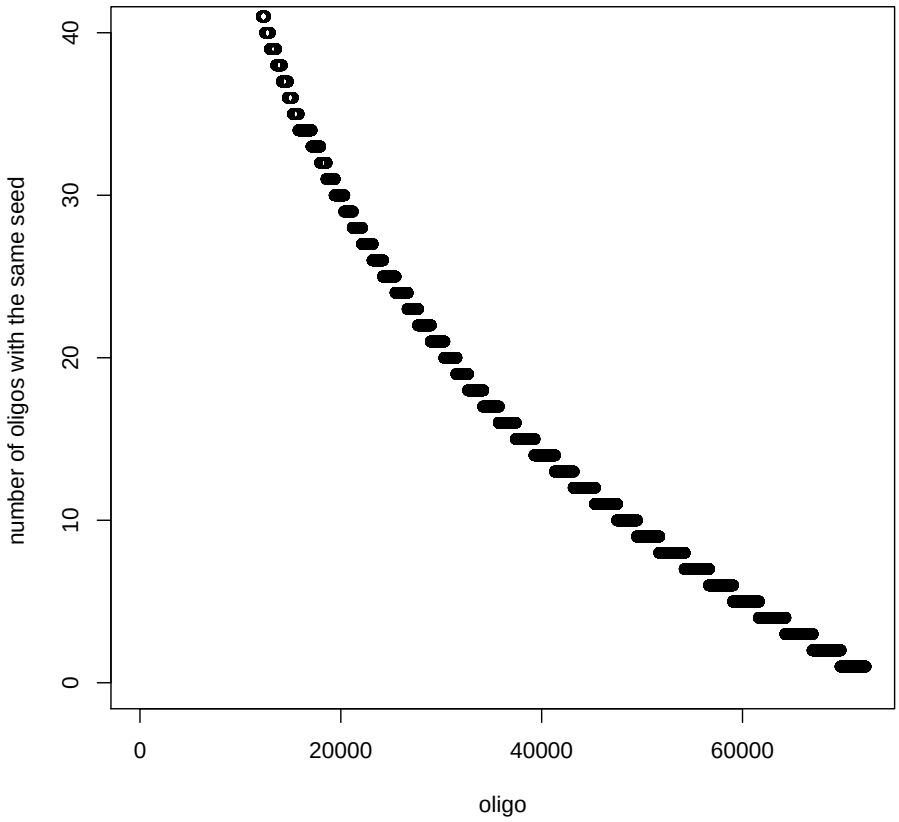
Another important information that we may want to look at, is the representation of the seed in the library (i.e. in how many oligos each seed is found).
In the following plot, for example, we can see that in this QIAGEN library we have about 2000 seeds (over the 16,384 possible seeds) that are contained in at least 10 different oligos in the screen. Hence we cannot be sure about the effect of most of the possible seeds, because they are not sufficiently represented in the screen.

```
> plot_seeds_oligo_count(uuk_screen)
```



If we draw the same plot at the level of the oligos (i.e. for each oligo, how many other oligos share the same seed), we can see that around 50,000 oligos (over 80,000) have their seed contained in at least 10 other oligos. Hence, most of the QIAGEN library is covered by a minor number of seeds, and this fact is good for the establishment of a our correction method.

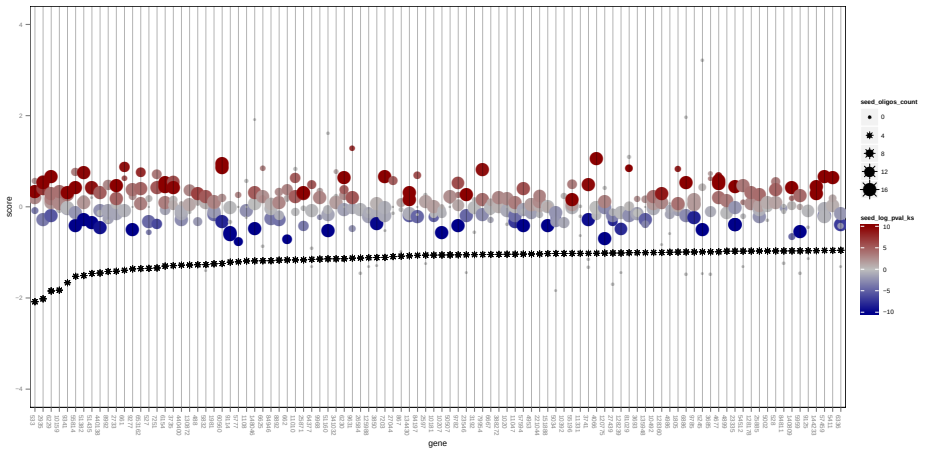
```
> plot_screen_seeds_count(uuk_screen)
```



2 CORRECTION

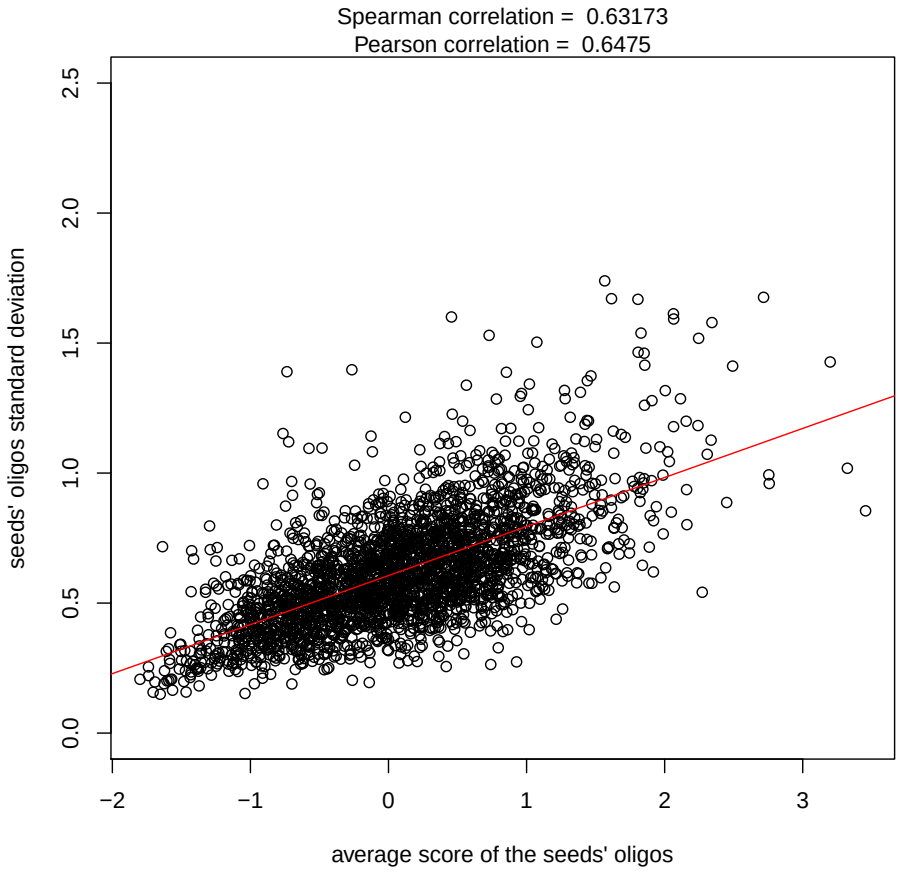
In order to correct for the off-target effect that is caused by the seed we do provide two alternative methods.
The first, and probably the most powerful, is the "seed_correction".
This method assumes that the seed effect acts in an "additive" way to the on-target signal. For example if we have an oligo X that has score -2, the method computes the average score of the other oligos in the libraries that contain the same seed as the oligo X. If for example this average score turns out to be -1.5, we can just subtract this score to the original oligo score to obtain the new "corrected" score ((-2) - (-1.5) = -0.5).

```
> screen_corrected <- seed_correction(uuk_screen)
> screen_corrected_dh <- seed_correction_pooled(uuk_screen_dh)

> plot_screen_hits(arrange(add_rank_col(screen_corrected), median))
```



However, the method assumes also that the correction factor (-1.5 in the previous example) should be multiplied by a coefficient c that reflects "how confident" we are about the correction ((-2) - c*(-1.5)).
The coefficient c can be a constant (e.g. 0.5) or it can vary depending on the behavior of the oligos that share the seed. In particular, we observed the the last approach to be the most successful. Hence for our algorithm we set c = 0.4 + s (0 <= c <= 1).
S is a factor proportional to the score standard deviation of the group of oligos that share the same seed. The idea is to correct more the groups of oligos that have a lower standard deviation, because we are more confident on the value we are correcting for.
However, we observed that the expected standard deviation of the oligos that share a seed strictly depends on the average score as it can be seen using our plot_seed_score_sd function.
Hence we calibrate the variable S accordingly (i.e. we set S to be proportional to the quantile in which the standard deviation is found, given the score range).

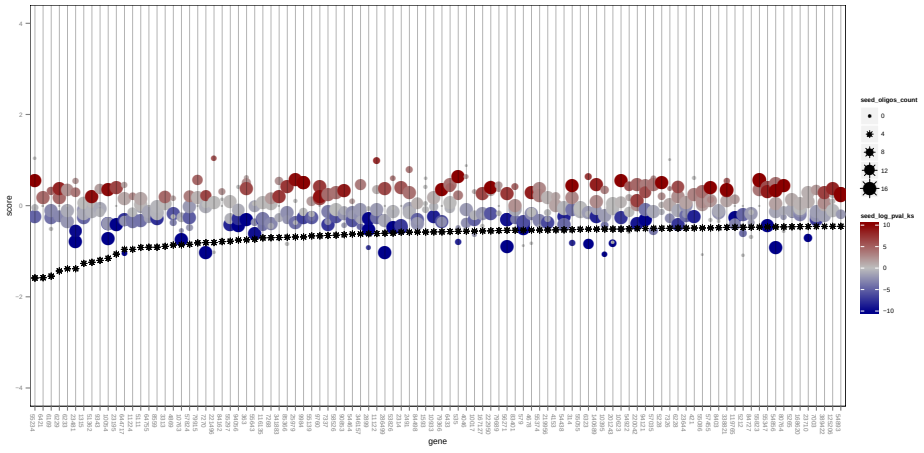
```
> plot_seed_score_sd(uuk_screen)
```



In order to correct pooled libraries, the user should make use of our seed_correction_pooled function. This function performs a weightened mean of all the seeds' scores in the pool, as described in (8). The user can also decide to try to correct using only the seed having the highest score in the pool (setting use_all_seeds = FALSE).

A second strategy of correction consists in simply discarding all the oligos that give a strong seed effect on the phenotype (i.e. having a seed average score over a certain threshold).
This strategy has the advantage to be more reliable (because we don't make any assumption on the additivity of the effect) but less comprehensive (because we often end up discarding many genes because they remain with no, or few, oligos).
Hence, after correcting with this method, we do suggest the user to look mainly at the genes that are on the very top of the resulting gene list that is obtained.
For this function, we offer a large set of parameters (e.g. thresholds) that the user can customize. Please look at the documentation of the function in order to get more insights on this.

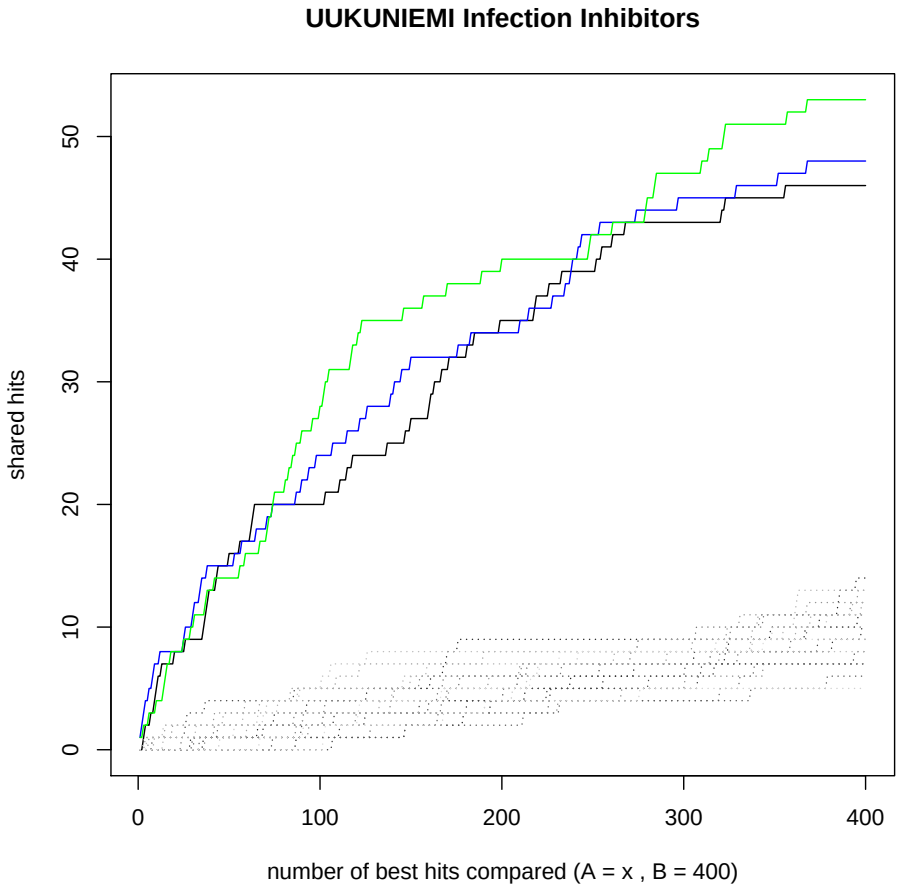
```
> screen_corrected2 <- seed_removal(uuk_screen)

> plot_screen_hits(arrange(add_rank_col(screen_corrected2), median))
```



3 BENCHMARKING

After having used a correction method, we may want to benchmark its performances.
The most reliable benchmark system consists in comparing the top n genes of the screen (before and after correction) with a second independent genome wide screen (if available).
The higher the number of shared hits, the better is the correction method.
Indeed, in our example data set, we compare the hits of our UUKUNIEMI QIAGEN screen with he hits of a DHARMACON genome-wide screen that has been performed on the same pathogen (we repeat the comparison for different n and draw a graph).

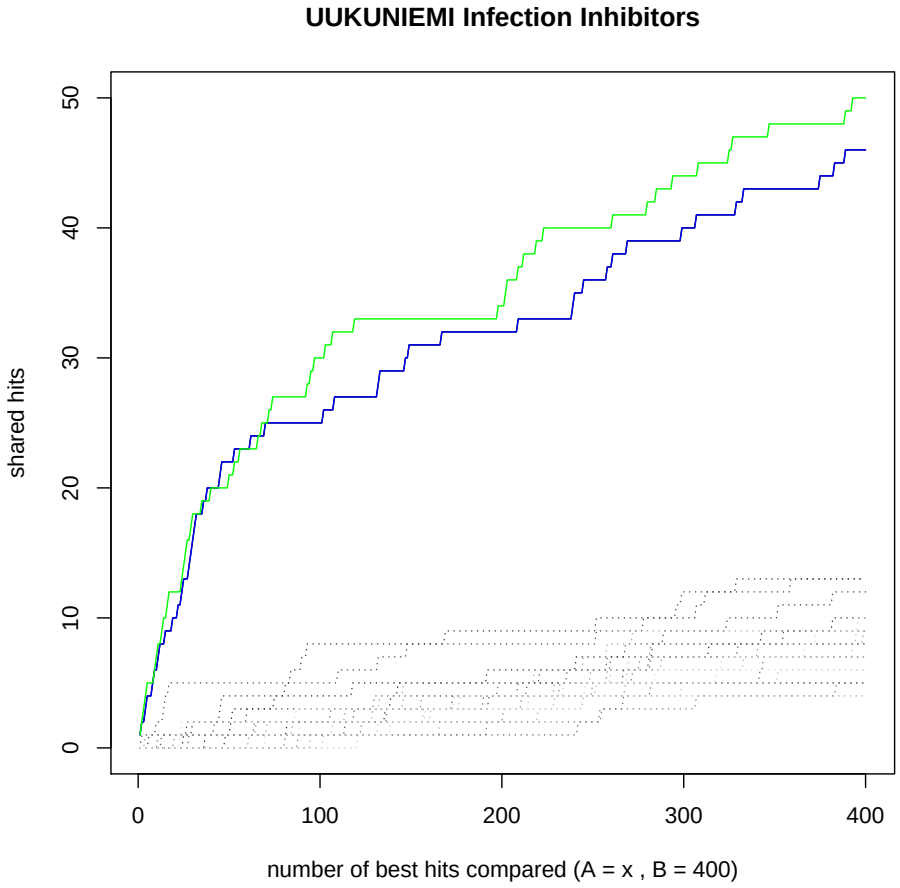
```
> benchmark_shared_hits(
+   glA=list(
+           arrange(add_rank_col(uuk_screen), median)$GeneID,
+           arrange(add_rank_col(uuk_screen), log_pval_rsa)$GeneID,
+           arrange(add_rank_col(screen_corrected), median)$GeneID
+   ),
+   glB=list(arrange(add_rank_col(uuk_screen_dh), median)$GeneID),
+   col=c("black", "blue", "green"),
+   title="UUKUNIEMI Infection Inhibitors"
+ )
```



As you can see in the graph above, the green line (share best hits of CORRECTED QIAGEN with DHARMACON) is always above the black line (share hits of QIAGEN with DHARMACON) and blue line (share hits of QIAGEN with DHARMACON, but with the QIAGEN sorted via RSA method (9), instead of just using the median).

Similarly, we can benchmark the pooled library (i.e. share best hits of CORRECTED DHARMACON with QIAGEN):

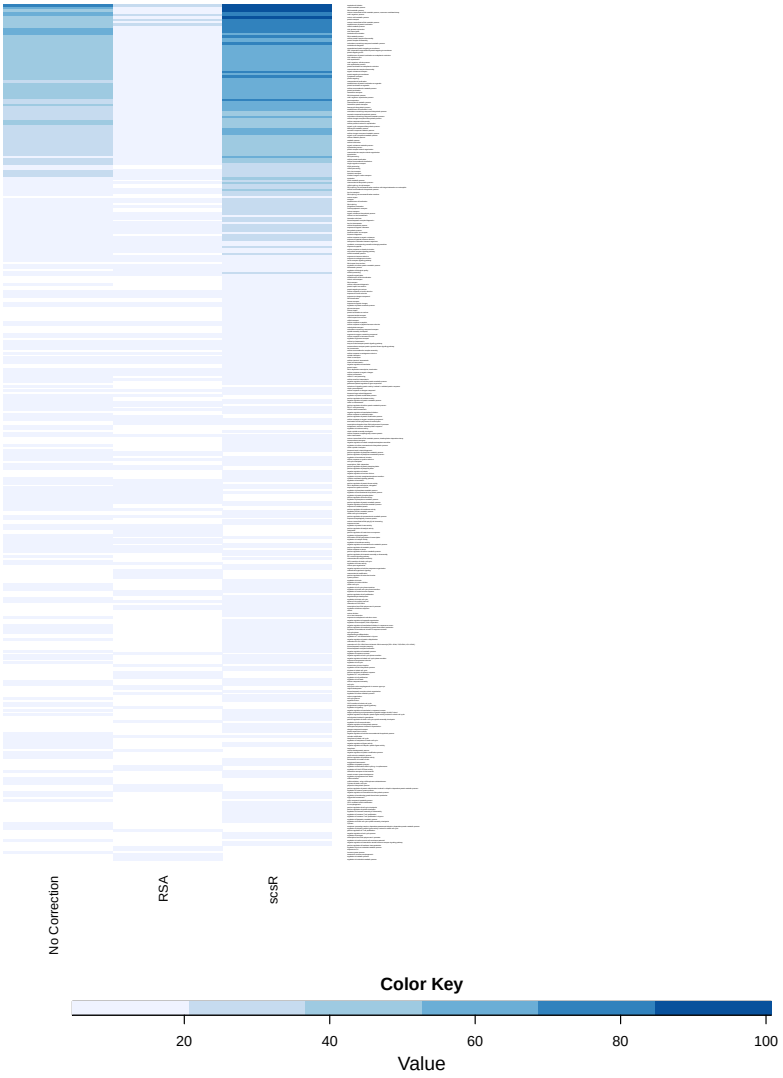
```
> benchmark_shared_hits(
+   glA=list(
+           arrange(add_rank_col(uuk_screen_dh), median)$GeneID,
+           arrange(add_rank_col(screen_corrected_dh), median)$GeneID
+   ),
+   glB=list(arrange(add_rank_col(uuk_screen), median)$GeneID),
+   col=c("black", "green"),
+   title="UUKUNIEMI Infection Inhibitors (pooled)"
+ )
```



An alternative benchmarking method, and the only choice in case you don't have a second genome-wide screen, is to look at the increase in the enrichment of gene annotations after correction, mainly the Gene Ontology Biological Process and the KEGG pathways.
We do provide a function that draws an heat map of the enriched annotations for either KEGG or GO. In the example below, we compare the enriched GO Biological Processes of the best 400 genes for three possible cases: QIAGEN sorted via median, QIAGEN sorted via RSA method, and QIAGEN corrected with scsR and then sorted via median.
As you can see, there is a strong enrichment in significant pathways after we do apply the seed correction using scsR.

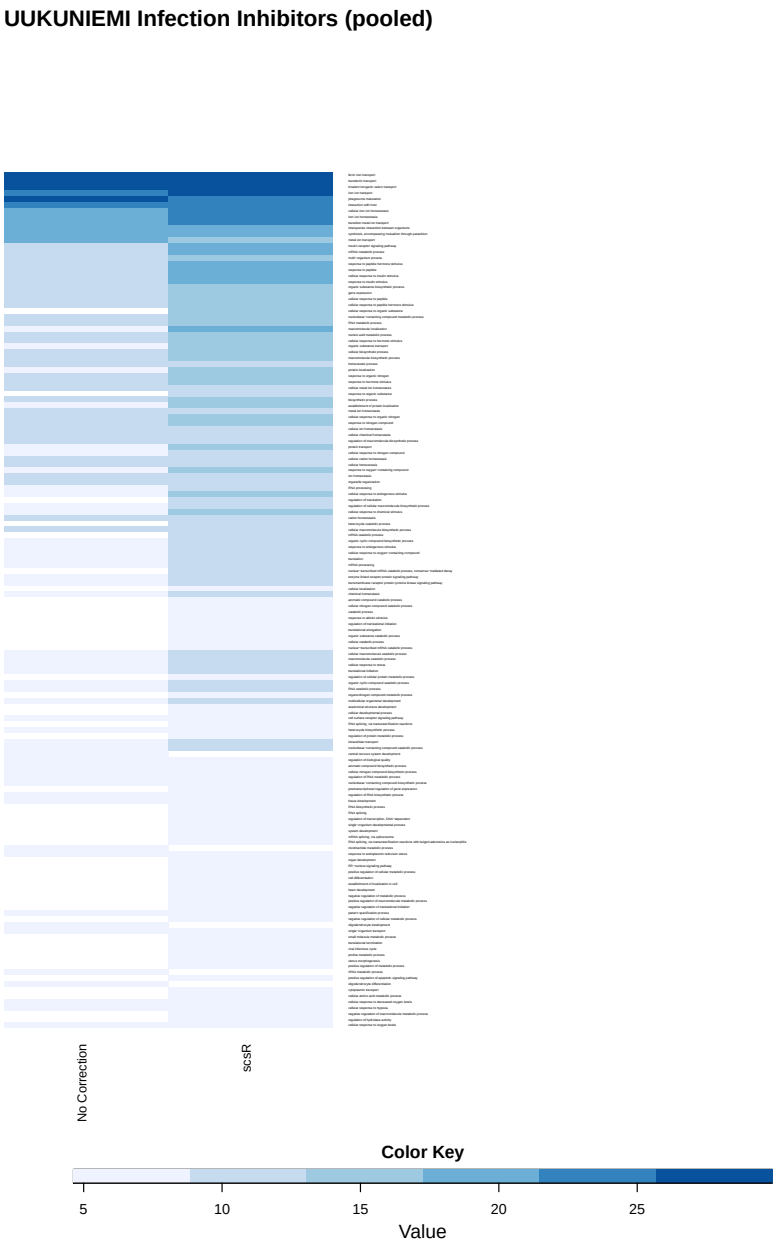
```
> em = enrichment_heatmap( list(arrange(add_rank_col(uuk_screen), median)$GeneID,
+                               arrange(add_rank_col(uuk_screen), log_pval_rsa)$GeneID,
+                               arrange(add_rank_col(screen_corrected), median)$GeneID
+ ),
+                          list("No Correction", "RSA", "scsR"),
+                          enrichmentType="Process", output_file=NULL,
+                          title="UUKUNIEMI Infection Inhibitors", fdr_threshold=0.01,
+                          limit=400
+ )
```

UUKUNIEMI Infection Inhibitors



We can use the same function to benchmark the correction of the DHARMACON pooled genome-wide screen:

```
> em_dh = enrichment_heatmap( list(arrange(add_rank_col(uuk_screen_dh), median)$GeneID,
+                                  arrange(add_rank_col(screen_corrected_dh), median)$GeneID
+ ),
+         list("No Correction", "scsR"), enrichmentType="Process", fdr_threshold=0.01,
+         title="UUKUNIEMI Infection Inhibitors (pooled)",
+ )
```



4 REFERENCES

1. Birmingham, A. et al. 3' UTR seed matches, but not overall identity, are associated with RNAi off-targets. Nat Methods 3, 199-204 (2006).

2. Jackson AL, et al. (2003) Expression profiling reveals off-target gene regulation by RNAi. Nature biotechnology 21(6):635-637.

3. Franceschini A. et al. Specific inhibition of diverse pathogens in human cells by synthetic miRNA-like oligonucleotides inferred from genome-wide RNAi screens. Submitted (2013).

4. Bushman FD, et al. (2009) Host cell factors in HIV replication: meta-analysis of genome-wide studies. PLoS pathogens 5(5):e1000437.

5. Meier et al. (2014) Genome-wide siRNA screens reveal VAMP3 as a novel host factor required for Uukuniemi virus late penetration. Journal of Virology.

6. Buehler E, et al. (2012) siRNA off-target effects in genome-wide screens identify signaling pathway members. Scientific reports 2:428.

7. Sigoillot FD, et al. (2012) A bioinformatics method identifies prominent off-targeted transcripts in RNAi screens. Nature methods 9(4):363-366.

8. Seed correction for siRNA (scsR) off-targets. Towards a better understanding of genome-wide siRNA screens reproducibility. In preparation.

9. Konig R, et al. A probability-based approach for the analysis of large-scale RNAi screens. Nat Methods. 2007;4:847-849.