# GenometriCorr (Genometric Correlation): an R package for spatial correlation of genome-wide interval datasets

Alexander Favorov*, Loris Mularoni, Yulia Medvedeva,
Harris A. Jaffee, Ekaterina V. Zhuravleva, Leslie M. Cope,
Andrey A. Mironov, Vsevolod J. Makeev, Sarah J. Wheelan

December 21, 2016

# 1 Introduction

## 1.1 Genometric layouts; independence or correlation

### 1.1.1 Using distance as a proxy for functional correlation

High-throughput sequencing has become a popular and ever more affordable method in molecular biology and clinical experiments, and will soon become a routine laboratory technique. Sequencing reads are mapped to a reference genome and the results are often drawn as points along lines representing chromosomes, with the layout of the points reflecting the physical distance between mapped reads. While this depiction is a convention, it is based on a longstanding belief that proximity on a chromosome implies potential functional interaction.

If sequencing results are analyzed as points on a line, we can measure the physical distance between sequencing results and annotated genomic features. Deviations in these measurements from the expected distributions indicate associations (or anti-associations) that may be biologically interesting. While it is not difficult to judge these associations by eye, a genome-wide assessment of spatial correlations is impractical to do manually.

Many, though certainly not all, functional relationships in genetics are based on proximity. For example, a promoter will be near the 5' end of the gene that it controls, a splicing control signal will be near splice sites, transcription factor binding sites will cluster where the transcription factors bind to regulate gene activity, and more. Measuring the proximity of a set of points on the genome to various genomic features will direct future experiments.

Note that this type of correlation is not intended to give final results, but to generate testable hypotheses.

*favorov@sensi.org

Genomic intervals are anything that can be stored as a chromosome number, start, and end. Strands are not considered but could be managed by subsetting the data. Genomic intervals can be represented as blocks on a line:

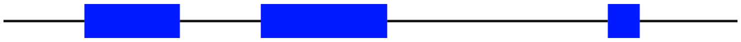


Figure 1: Genomic intervals

### 1.1.2 Intervals can be correlated or independent

If we have two types of features, they can be correlated, or they can be independent. If the features are independent, the locations of one type of feature are randomly positioned with respect to the other, while if they are correlated, the locations of the feature types will, on average, follow a recognizable pattern; the features can be a relatively constant distance apart, they can be consistently near or far away from each other in genomic coordinates, or they can preferentially overlap (or only rarely overlap). Thus, if features are correlated, the locations of one type of feature give information about the positions of the other type of feature.

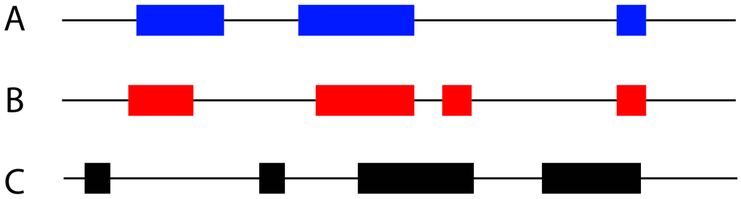


Figure 2: Three sets of genomic intervals. A and B are correlated, A and C are independent.

### 1.1.3 Correlation comes in many different flavors

We will introduce some terminology for simplicity. The set of intervals whose positions are considered fixed in the genome is the reference set. The intervals whose positions are being tested to see whether they are related to the reference set in any way is the query set. Note that the comparison is thus asymmetric.

Figure 3 shows the basic question we are asking.

Figure 4 illustrates some important complications that we address.

Comparing the intervals in query 1 to the reference intervals, we see that the two sets of intervals consistently do not overlap. They are not independent, and the statistics will show that they are anticorrelated. The query 2 intervals do overlap substantially with the reference intervals and are thus correlated; again the statistics will reflect this and will show a positive association. Query 3 has only one interval. That interval overlaps with a reference interval, so query 3 is correlated with the reference. However, if the query and reference identities

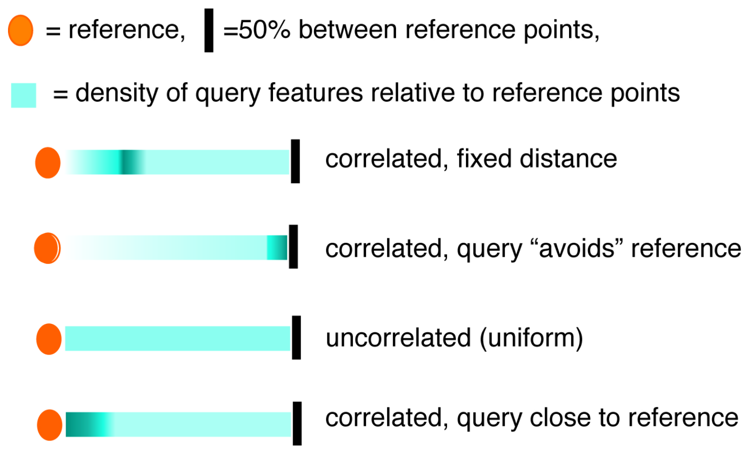

= reference,
=50% between reference points,
= density of query features relative to reference points
correlated, fixed distance
correlated, query “avoids” reference
uncorrelated (uniform)
correlated, query close to reference

Figure 3: Our goal is to determine whether the query points are correlated to the reference points. We do this, in essence, by assuming that if they are independent, they will be uniformly distributed with respect to the reference points, and if not, the density of distance between query and reference points will be nonuniform.

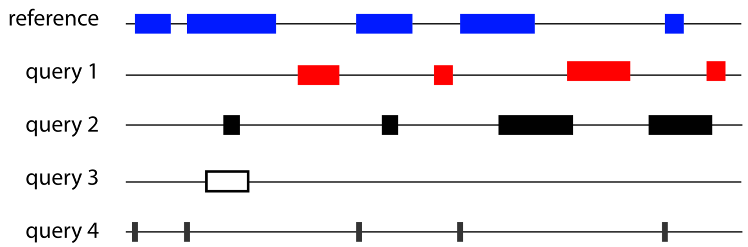


Figure 4: Four scenarios.

are reversed, most of the new query intervals do not overlap with and are not near the single new reference interval, so these two datasets have an asymmetric relationship. Only by testing every query-reference set in both directions can we uncover such an asymmetry. The last set of intervals, query 4, brings up different issues. We can measure the relationship between the query and reference in two ways. First, we can look at the distribution of the midpoints of the query intervals with respect to the distribution of the reference interval midpoints and record the distances as ratios (for example, if the query is 10 units from one reference point and 90 units from the nearest on the other side, its ratio will be 0.1). In a large genome this works well, because average distances are big, so distinguishing a position 10% into an inverval from a position 30% into an interval is easy. Second, we can look at the raw distance between the midpoints of the query and the midpoints of the reference. This works well for small genomes because here the midpoints of the reference can be close enough that if a query midpoint is, for example, always 100 bp from a reference midpoint, the ratio test will show a much wider distribution, as when the query is between two reference midpoints that are only 300 bp away the ratio test will read 0.33, but when the reference midpoints are 1000 bp away the ratio test will read 0.1, and the query will appear to be uncorrelated. For this reason we find it useful to do both tests, in both directions. These concepts will be elaborated in the next section.

## 1.2 Statistical approach

### 1.2.1 Working with intervals

**Tests on relative distances**

Many of the tests we use work only with pointwise data, not with intervals. Very large intervals may relate to genomic features in different ways, depending on whether we examine their start points, end points, both boundaries, or just a point in the middle. Rather than trying to address this ambiguity or to randomly guess at what the user hopes to do, we expect the user to specify the points when the exact point is important, and we use the midpoint when the user inputs an interval. Also, the user can provide a custom calculation to define the representative point to use for each interval.

Now, we can characterize each query point by its relative distance, as illustrated in figure 5.

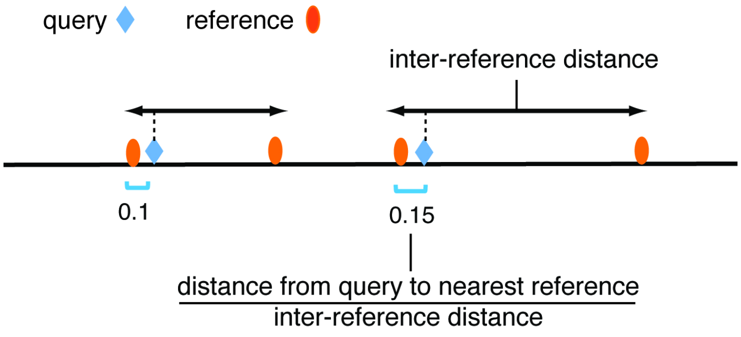


Figure 5: Relative distance

Formally, the relative distance $d_i$ for a query point $i$ is:

$$d_i = \frac{\min\left(|q_i - r_k|, |r_{k+1} - q_i|\right)}{|r_{k+1} - r_k|}, k = \arg\min_{q_i \geq r_k}(q_i - r_k).$$

If the reference and query intervals are independent, the query points are positioned randomly with respect to the reference, and the $d_i$'s will be distributed uniformly in $[0, 0.5]$. The corresponding $p - value$ is obtained by using the Kolmogorov-Smirnov test.

The Kolmogorov-Smirnov test (K-S test) is accompanied by permutation tests to determine the level and direction of deviation from the null expectation. The ECDF (Empirical Distribution Cumulative Function) of the relative

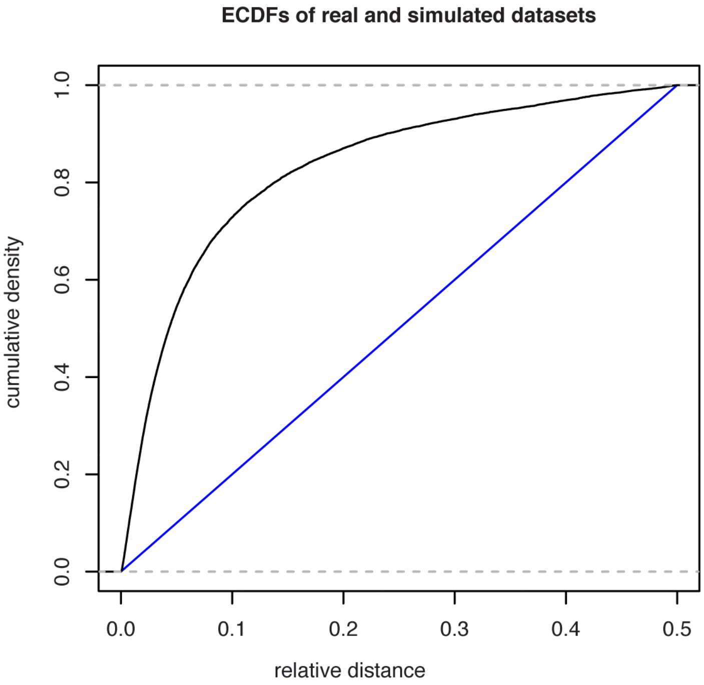


Figure 6: Area between uniform ECDF for unrelated feature sets (blue) and experimental ECDF for related feature sets (black) is a measure of correlation of the query and reference feature sets.

distances $d_i$ is a straight line between $(0, 0)$ and $(0.5, 1)$ if the query and reference points are perfectly independent, so we compare our data to this line.

The area between the ECDF for the reference and query points and the ideal straight line

$$S = \int_0^{0.5} |ECDF(d) - ECDF_{ideal}(d)|\ dd$$

is a measure of the correlation between the query and reference. So, by drawing $N$ sets of values that model uniform distribution of $d_i$ we get $N$ outcomes of a null distribution for $S$ and thus we can evaluate the $p - value$ for S.

Both the area permutation test and the Kolmogorov-Smirnov test show only how much the data deviate from independence, not whether they are positively or negatively correlated.

The sign of difference between the areas under the real ECDF curve and the ideal ECDF curve indicates the direction of the correlation. We define a correlation-like measure

$$Corr_{ECDF} = \frac{\displaystyle\int_0^{0.5} (ECDF(d) - ECDF_{ideal}(d))\, dd}{\displaystyle\int_0^{0.5} ECDF_{ideal}(d)\, dd}.$$

Positive $Corr_{ECDF}$ indicates positive correlation (query points tend to be close to reference points) and vice versa.

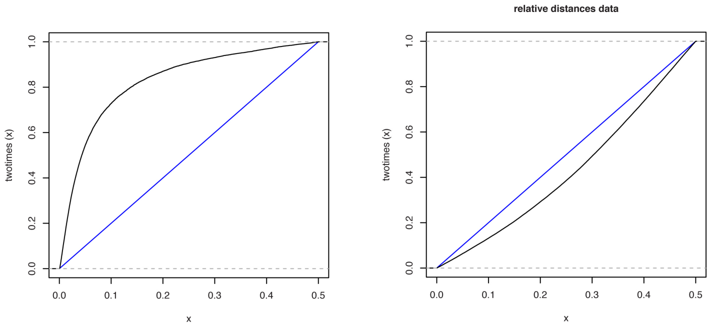


Figure 7: $Corr_{ECDF}$ is positive for the data represented by the upper black line (left pane); the area under it is more than the area under the blue line that marks the distribution of independent data, so the correlation is positive. For the data represented by the lower black line (right pane), the correlation is negative.

### 1.2.2 Absolute distance test

We can also determine whether the query intervals are spaced more often than expected at a specific distance from the reference intervals; for example a polymerase binding site and a transcription factor binding site. Figure 8 illustrates the design.

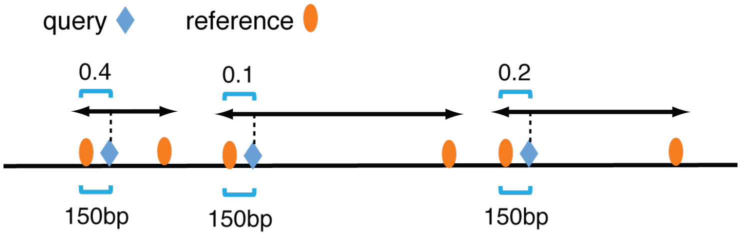


Figure 8: Query intervals are found at a fixed distance from reference intervals. Relative distances (0.4, 0.1, 0.2) are not consistent.

For each query point (contracted interval) we define the minimal distance to a reference point, $l_i = \min_k(q_i - r_k)$. Its mean value,

$$L = \frac{\sum_i l_i}{\#q}$$

characterizes the correlation between query and reference points. We perform a permutation test for significance: keeping the reference points fixed, we draw $N$ simulated query positions that are uniformly distributed along the chromosome and calculate $L$ for each, to obtain the null distribution of $L$. The test is two-sided and gives both the $p-value$ for the real $L$, and the sign of any correlation.

### 1.2.3 Projection test

Another test we find useful is the projection test, which is robust when the intervals being tested cover a fairly large percentage of the length of the reference sequence. Here, we determine the number of midpoints of query intervals overlapping the reference intervals and test whether it is outside of the null expectation. The probability of a query midpoint falling into a reference interval is:

$$p = \frac{\text{coverage of the reference}}{\text{chromosome length}}$$

Therefore, the distribution of the number of query points that overlap reference interval can be approximated by the binomial distribution for $\#q$ trial with success probability $p$. The null hypothesis is that query points hit the reference intervals randomly. The test is also two-sided; it provides both p-value and the direction of correlation. Additionly, we estimate the size of the effect with the observed/expected reatio for the number of query points that hit a reference interval.

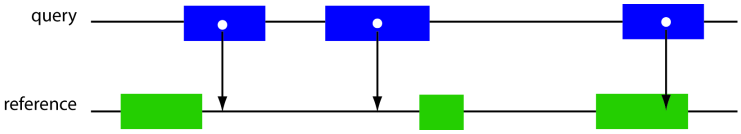


Figure 9: Projection test.

### 1.2.4 Naïve Jaccard approach

For this we do not test the relationships between points, but between intervals, so this test is quite complementary to the pointwise tests.

$$\text{Jaccard measure (index):} \quad J(A,B) = \frac{A \cap B}{A \cup B}$$

To create a null distribution for comparison, we permute the order of the query intervals across the genome, not retaining the lengths of the gaps between the query intervals.

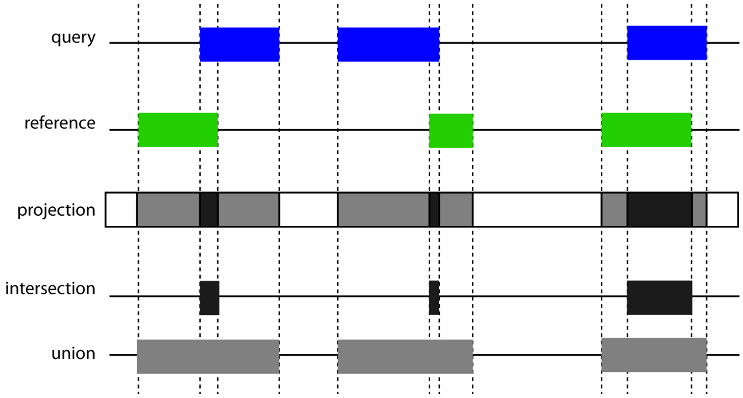


Figure 10: The Jaccard measure of the correlation of two interval sets is the ratio of the lengths (in bases) of their intersection and their union.

### 1.3 Tests of correlation over an entire genome

All the tests described above are applicable to a single chromosomes or a set of chromosomes (for example, a whole genome). The data for each test are simply summarized over all chromosomes before checking for significance. Another option that is available is restricting the analysis to a set of genomic intervals; for example, asking questions about whether features are correlated when the features always lie within genes is impossible when using the entire genome as the potential space for feature positions, as the features will always look tightly correlated since they co-occur within genes. Using the mapping functions provided, each sub-interval (here, a gene) can be considered as a separate ”chromosome,” to enable detection of correlations within smaller intervals.

### 1.4 GenometriCorr (Genometric Correlation) package

The package `GenometriCorr` provides functions to load interval data from a plain text file (any accepted format), as well as the main procedure that tests whether the interval sets are spatially independent, and plotting functions to generate graphical representations of the relationships between the features.

## 2 Using GenometriCorr: examples

### 2.1 R objects that `GenometriCorr` understands

The interval sets are represented by the `IRanges` or `RangedData` or `GRanges` objects, which are interval data representation classes defined by `IRanges` package. `IRanges` is a set of intervals in one space (chromosome). `RangedData` is a set of intervals defined over different spaces (here, chromosomes), so it is suitable for encoding a full-genome annotation. `GRanges` is very simlar to `RangedData` but it contains the chromosome length information and ensures that operations do not return intervals beyond the length of the chromosomes.

## 2.2 Code loading

Load the package:

```
> library("GenometriCorr")
```

## 2.3 Let's load files

We will use the `import` function from `rtracklayer` package to read data for the query and reference intervals (the result will be used in later narration). The two files describe the coordinates of CpG islands and of RefSeq genes in human genome (v19).

```
> library("rtracklayer")
> library("TxDb.Hsapiens.UCSC.hg19.knownGene")
> refseq <- transcripts(TxDb.Hsapiens.UCSC.hg19.knownGene)
> cpgis <- import(system.file("extdata",
+     "UCSCcpgis_hg19.bed", package = "GenometriCorr"))
> seqinfo(cpgis) <- seqinfo(TxDb.Hsapiens.UCSC.hg19.knownGene)[seqnames(seqinfo(cpgis))]
```

The information about lengths and names of chromosomes is coded in `seqinfo` provided by `TxDb.Hsapiens.UCSC.hg19.knownGene`. The `refseq` object contains it; the `cpgis` is read from a `bed` file and thus we set the `seqinfo`. If both the query and reference objects do not carry `seqinfo`, the information can be passed via `chromosomes.length` parameter that is organized exactly as `seqlenghts(GenomicRanges)` result. It is, a numeric vector with names, which are chromosome names.

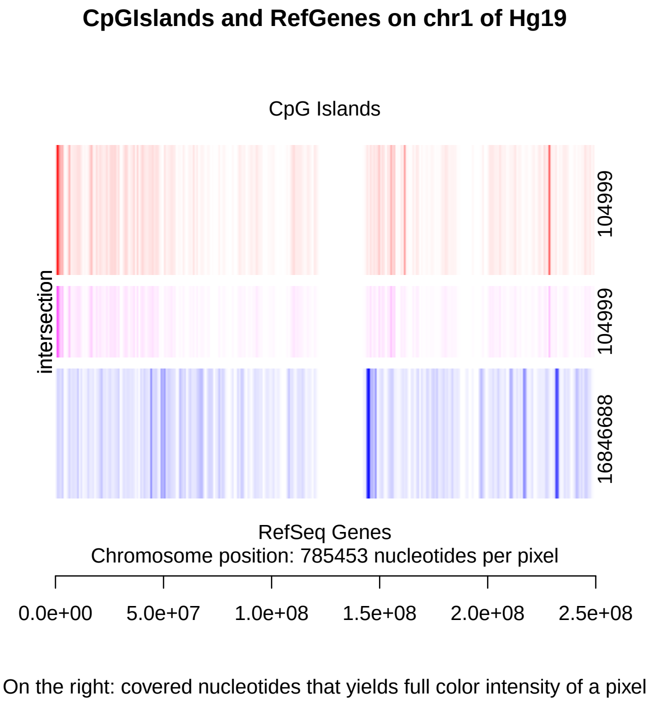
If the chromosome lengths are not either given explicitly or coded in reference or in query, they are taken to be the maximum of the reference and query coordinates provided, per chromosome, and a warning is generated. You can suppress the warning with `suppress.evaluated.length.warning=TRUE`.

## 2.4 Visualize the query and reference intervals

Sometimes, it is useful to examine a pair of interval sets by eye, to add information to the statistical results. Three visualization procedures are provided.

Let's look at the CpG islands and KnownGenes genes on chromosomes 1 (hg19).

```
> VisualiseTwoIRanges(ranges(cpgis[seqnames(cpgis) ==
+     "chr1"]), ranges(refseq[seqnames(refseq) ==
+     "chr1"]), nameA = "CpG Islands", nameB = "RefSeq Genes",
+     chrom_length = seqlengths(TxDb.Hsapiens.UCSC.hg19.knownGene)["chr1"],
+     title = "CpGIslands and RefGenes on chr1 of Hg19")
```



Actually, we can look at all the choromomes, one per page, in a milti-page pdf the is created by `VisualiseTwoGRanges` function.

## 2.5 The main procedure

`GenometriCorrelation` tests the null hypothesis that the query and reference intervals are spatially independent; that is, that they have no significant relationship in terms of genomic coordinates.

```
> pn.area <- 100
> pn.dist <- 100
> pn.jacc <- 100
> cpgi_to_genes <- GenometriCorrelation(cpgis,
+     refseq, chromosomes.to.proceed = c("chr1",
+         "chr2", "chr3"), ecdf.area.permut.number = pn.area,
+     mean.distance.permut.number = pn.dist,
+     jaccard.measure.permut.number = pn.jacc,
+     keep.distributions = TRUE, showProgressBar = FALSE)
```

The result is an object of the (S4) `GenometriCorrResult` class. The class is actually (inferred from) a list of results with some useful functionality. The simplest is the `show()` method.

```
> print(cpgi_to_genes)
```

```
                                               chr1
query.population                               2462
reference.population                           7967
query.coverage                                 1881629
reference.coverage                             116939971
relative.distances.ks.p.value                  2.199463e-11
relative.distances.ecdf.deviation.area         0.02188097
relative.distances.ecdf.area.correlation       0.08747395
query.reference.intersection                   1305420
query.reference.union                          117516180
jaccard.measure                                0.01110843
projection.test.p.value                        0
projection.test.lower.tail                     FALSE
projection.test.obs.to.exp                     1.597282
scaled.absolute.min.distance.sum               1536.773
reference.middles                              Numeric,7967
relative.distances.ecdf.deviation.area.p.value "<0.01"
scaled.absolute.min.distance.sum.p.value       "<0.01"
scaled.absolute.min.distance.sum.lower.tail    TRUE
jaccard.measure.p.value                        "<0.01"
jaccard.measure.lower.tail                     FALSE
                                               chr2
query.population                               1688
reference.population                           5092
query.coverage                                 1379397
reference.coverage                             109822784
relative.distances.ks.p.value                  0
relative.distances.ecdf.deviation.area         0.03430315
relative.distances.ecdf.area.correlation       0.1370043
query.reference.intersection                   889234
query.reference.union                          110312947
jaccard.measure                                0.008061012
projection.test.p.value                        0
projection.test.lower.tail                     FALSE
projection.test.obs.to.exp                     1.474565
scaled.absolute.min.distance.sum               959.9135
reference.middles                              Numeric,5092
relative.distances.ecdf.deviation.area.p.value "<0.01"
scaled.absolute.min.distance.sum.p.value       "<0.01"
scaled.absolute.min.distance.sum.lower.tail    TRUE
jaccard.measure.p.value                        "<0.01"
jaccard.measure.lower.tail                     FALSE
                                               chr3
query.population                               1163
reference.population                           4328
query.coverage                                 882783
```

```
reference.coverage                             100086725
relative.distances.ks.p.value                  2.589502e-10
relative.distances.ecdf.deviation.area         0.02144485
relative.distances.ecdf.area.correlation       0.08809636
query.reference.intersection                   631106
query.reference.union                          100338402
jaccard.measure                                0.006289775
projection.test.p.value                        0
projection.test.lower.tail                     FALSE
projection.test.obs.to.exp                     1.478352
scaled.absolute.min.distance.sum               681.6992
reference.middles                              Numeric,4328
relative.distances.ecdf.deviation.area.p.value "<0.01"
scaled.absolute.min.distance.sum.p.value       "<0.01"
scaled.absolute.min.distance.sum.lower.tail    TRUE
jaccard.measure.p.value                        "<0.01"
jaccard.measure.lower.tail                     FALSE
                                               awhole
query.population                               5313
reference.population                           17387
query.coverage                                 4143809
reference.coverage                             326849480
relative.distances.ks.p.value                  0
relative.distances.ecdf.deviation.area         0.02563675
relative.distances.ecdf.area.correlation       0.1033465
query.reference.intersection                   2825760
query.reference.union                          328167529
jaccard.measure                                0.008610724
projection.test.p.value                        0
projection.test.lower.tail                     FALSE
projection.test.obs.to.exp                     1.526032
scaled.absolute.min.distance.sum               3178.385
reference.middles                              NULL
relative.distances.ecdf.deviation.area.p.value "<0.01"
scaled.absolute.min.distance.sum.p.value       "<0.01"
scaled.absolute.min.distance.sum.lower.tail    TRUE
jaccard.measure.p.value                        "<0.01"
jaccard.measure.lower.tail                     FALSE
```

The `GenometriCorrelation` function calculates the correlation statistics previously described, on each chromosome separately, and then on the genome as a whole (”awhole”). The visualization above suggested that the query and reference intervals are correlated. The resulting list is structured by the names of the chromosomes used and an additional `'awhole'` pseudochromosome. For each of them, it contains the following data:

- The very low p-value calculated in `relative.distances.ks.p.value` is in accordance with the observation. relative.distances.ecdf.area.correlation is positive, so the query and reference are in general closer to each other than we would expect if they are independent.

- The projection.test.p.value is zero, indicating either significant overlap or significant lack of overlap. projection.test.lower.tail is FALSE, meaning that we are in the upper tail of the distribution and there is significantly more overlap of the query and reference intervals than we would expect if they were independent. projection.test.obs.to.exp that is about 1.5 confirms it.
- All three permutation tests give <0.01 meaning that the observed spatial relationships (absolute or relative distance apart) are significantly different than what is seen in the permutation distribution.
- From the p-values of the permutation distributions we cannot tell whether the query and reference intervals are significantly close together or significantly far apart. As the value of the scaled.absolute.min.distance.sum.lower.tail is TRUE, we know that the absolute distances between query and reference are consistent and small, and, finally, the jaccard.measure.lower.tail is FALSE, indicating an unexpectedly high overlap, as defined by the Jaccard measure.

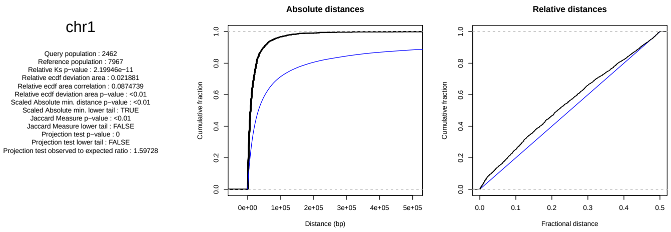
In this case, all tests indicate that the query and reference intervals are close together and/or overlapping. In other datasets, the values may be very different; for example, the query and reference intervals can be separated by a consistent and large distance, giving a significant absolute distance correlation with scaled.absolute.min.distance.sum.lower.tail FALSE, and the overlap measures may or may not indicate consistent non-overlapping intervals, depending on the size of the genome and the distances involved. Thus the various tests are extremely useful as a whole, to determine the relationship between the query and reference data in a much more precise way.

It is helpful to use visualization tools like those shown in Figures 6 and 7. Two types of graphical output are available for the GenometriCorrResult class. In both cases, use the keep.distributions=TRUE parameter in the GenometriCorrelation call so that the object returned by the call contains the distribution data needed to report anything more than the correlation statistics. Below is an example, generated by graphical.report function, for chromosome 1.

```
> graphical.report(cpgi_to_genes, pdffile = "CpGi_to_RefSeq_chr1_picture.pdf",
+     show.chromosomes = c("chr1"), show.all = FALSE)
```

Another graphical display is available. The visualize function has the same parameters as graphical.report and also requires that the initial function be run with keep.distributions=TRUE. An example for chromosome 1 follows.

```
> visualize(cpgi_to_genes, pdffile = "CpGi_to_RefSeq_chr1_picture_vis.pdf",
+     show.chromosomes = c("chr1"), show.all = FALSE)
```



```
Query population : 2462
Reference population : 7967
Relative Ks p−value : 2.19946e−11
Relative ecdf deviation area : 0.021881
Relative ecdf area correlation : 0.0874739
Relative ecdf deviation area p−value : <0.01
Scaled Absolute min. distance p−value : <0.01
Scaled Absolute min. lower tail : TRUE
Jaccard Measure p−value : <0.01
Jaccard Measure lower tail : FALSE
Projection test p−value : 0
Projection test lower tail : FALSE
Projection test observed to expected ratio : 1.59728
```

Figure 11: Simple graphical output for chromosome 1

The two panes represent absolute (left) and relative (right) distance distribution. The absolute distances aje not scaled by chromosome length, they are just distances, so the picture sometimes can be not in agreement with the scaled absolute distance statistics. Here, the observed distribitions are shown by black lines, while the expected are blue. The expected *cdf* for relative (right pane) distance is diagonal, and it correspond to uniform in $[0, 0.5]$. For the expected $\mathit{cdf}(a)$ for the absolute distance, we notice that

$$\mathit{cdf}(a) = p(x < a) = \sum_i p(x < a | x \in R_i) p(x \in R_i),$$

where $R_i$ is $i$-th reference interval. If $x$ is the closest distance between a query point and the $R_i$ border and the query point that is known to be in the $R_i$ interval, the distribition of $x$ is uniform in $[0, |R_i|/2]$.

$$p(x < a | x \in R_i) = \begin{cases} 1 & \text{for} \quad a \geq R_i/2 \\ 2a/R_i & \text{for} \quad a < R_i/2 \end{cases}$$

$p(x \in R_i)$ in the expected case is proprtional to $|R_i|$,

$$p(x \in R_i) = |R_i| / \sum_i R_i = |R_i|/L.$$

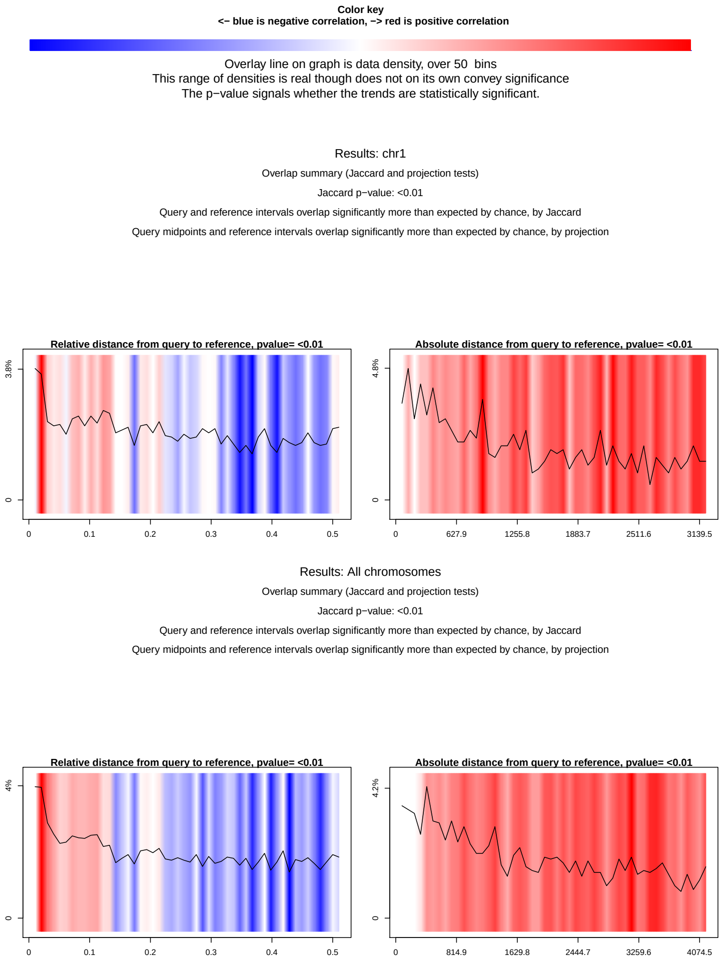
What we denote as $L$ here equals to the space length is we circled the reference. So,

$$\mathit{cdf}(a) = \sum_{2a \geq R_i} R_i/L + \sum_{2a < R_i} p(x \in R_i)(2a/R_i) =$$

$$\frac{\sum_{2a \geq R_i} |R_i|}{L} + \sum_{2a < R_i} 2a =$$

$$\frac{\sum_{2a \geq R_i} |R_i|}{L} + 2a \left| \{ i : 2a < R_i \} \right|,$$

where $|\{\ldots\}|$ denotes the power (number of elenemts) ot a set.



**Color key**

**<− blue is negative correlation, −> red is positive correlation**

Overlay line on graph is data density, over 50  bins
This range of densities is real though does not on its own convey significance
The p−value signals whether the trends are statistically significant.

Results: chr1

Overlap summary (Jaccard and projection tests)

Jaccard p−value: <0.01

Query and reference intervals overlap significantly more than expected by chance, by Jaccard

Query midpoints and reference intervals overlap significantly more than expected by chance, by projection

Results: All chromosomes

Overlap summary (Jaccard and projection tests)

Jaccard p−value: <0.01

Query and reference intervals overlap significantly more than expected by chance, by Jaccard

Query midpoints and reference intervals overlap significantly more than expected by chance, by projection

Relative distance from query to reference, pvalue= <0.01
3.8%
0
0
0.1
0.2
0.3
0.4
0.5
Absolute distance from query to reference, pvalue= <0.01
4.8%
0
0
627.9
1255.8
1883.7
2511.6
3139.5
Relative distance from query to reference, pvalue= <0.01
4%
0
0
0.1
0.2
0.3
0.4
0.5
Absolute distance from query to reference, pvalue= <0.01
4.2%
0
0
814.9
1629.8
2444.7
3259.6
4074.5

Figure 12: More colorful graphics with observed/expected trends for chromosome 1

### 2.6 Configuration file

The main function can be called in two ways. First, the arguments can be supplied directly to `GenometriCorrelation` function as before.

The second, S4-compliant way to run the main function is to create an S4 GenometriCorrConfig object from a configuration file, and then to pass the object to the `run.config` function with or without changes. This enables reproducibility and a simple interface. A file template is provided:

```
> config <- new("GenometriCorrConfig", system.file("extdata",
+     "template-config.ini", package = "GenometriCorr"))
```

Now, let's print the configuration object.

```
> print(config)
```

```
[data]
query=UCSCcpgis_hg19.bed
query.format=bed
reference=UCSCrefseqgenes_hg19.bed
reference.format=bed
do.mapping=FALSE

[chromosomes]
chr1
chr2
chr3

[chromosomes.length]
chr1=249250621
chr2=243199373
chr3=198022430
chr4=191154276
chr5=180915260
chr6=171115067
chr7=159138663
chrX=155270560
chr8=146364022
chr9=141213431
chr10=135534747
chr11=135006516
chr12=133851895
chr13=115169878
chr14=107349540
chr15=102531392
chr16=90354753
chr17=81195210
```

```
chr18=78077248
chr20=63025520
chrY=59373566
chr19=59128983
chr22=51304566
chr21=48129895
chrM=16571

[options]
add.chr.as.prefix=FALSE
awhole.only=FALSE
suppress.evaluated.length.warning=FALSE
cut.all.over.length=FALSE
keep.distributions=TRUE
showTkProgressBar=FALSE
showProgressBar=FALSE

[tests]
ecdf.area.permut.number=100
mean.distance.permut.number=100
jaccard.measure.permut.number=100
random.seed=1248312
```

We can change some fields:

```
> config$tests$ecdf.area.permut.number <- 10
> config$tests$mean.distance.permut.number <- 10
> config$tests$jaccard.measure.permut.number <- 10
> config$chromosomes <- "chr18"
> config$showTkProgressBar = FALSE
```

The `GenometriCorrConfig` object contains all parameters for the `GenometriCorrelation`. They are arranged in four groups: `[chromosomes]`, `[chromosomes.length]`, `[options]` and `[tests]` in the file and in corresponding lists in the `GenometriCorrConfig` .

Also, there is a group `[data]` that describes the input for query, reference and mapping (see below) if any. It can be empty, or it can specify the filenames and formats of the input files, or the names of R objects in the current workspace, to be used as query, reference and mapping data sources. If an R variable name is used, the corresponding format is ”

As a simple illustration of using the config file we will use sequencing data, instead of known annotations. The analyses performed by the GenometriCorr package are applicable to any type of whole-genome data as long as it can be specified as points or intervals in genomic coordinates. This means that the functions can analyze sequencing data or microarray data as well as perform comparison between annotations (as above); the sequencing or microarray data must be presented either as points or as intervals as the package contains no functions to do alignment. In this short example we compare ChIPseq data with expression data on a single human chromosome.

The ChIPseq data are in .bed format; the expression data are also in .bed format (we have assigned an expression cutoff to compare highly expressed genes with the ChIPseq data).

```
> library("rtracklayer")
> histones <- import(system.file("extdata",
+     "chr18.H3K4me3.bed", package = "GenometriCorr"))
> expr_genes <- import(system.file("extdata",
+     "chr18.mRNAseq.bed", package = "GenometriCorr"))
```

And, finally, let's start the GenometriCorrelation with this parameters:

```
> conf_res <- run.config(config, query = histones,
+     reference = expr_genes)
```

Here, we passed query and reference to the run.config call as RangedData R objects, wo they will be pased to the GenometriCorrelation as is. run.config also accepts filenames, file formats or the names or R variables to be used as query, reference and mapping. To show that a variable name is passed, the corresponding format identifier is 'R.variable.name'. All the methods of passing data to run.config except the direct pass of R object (as in the example) can be described in the configuration file.

The run.config returns a GenometriCorrResult object that was obtained from GenometriCorrResult. The @config slot of the returned object is a GenometriCorrConfig describing the run.

A config example with the permutation numbers already set to 10 is quick-config.ini:

```
> quickconfig <- new("GenometriCorrConfig",
+     system.file("extdata", "quick-config.ini",
+         package = "GenometriCorr"))
> print(quickconfig)
```

```
[data]
query=UCSCcpgis_hg19.bed
query.format=bed
reference=UCSCrefseqgenes_hg19.bed
reference.format=bed
do.mapping=FALSE

[chromosomes]
chr1
chr2
chr3
```

```
[chromosomes.length]
chr1=249250621
chr2=243199373
chr3=198022430
chr4=191154276
chr5=180915260
chr6=171115067
chr7=159138663
chrX=155270560
chr8=146364022
chr9=141213431
chr10=135534747
chr11=135006516
chr12=133851895
chr13=115169878
chr14=107349540
chr15=102531392
chr16=90354753
chr17=81195210
chr18=78077248
chr20=63025520
chrY=59373566
chr19=59128983
chr22=51304566
chr21=48129895
chrM=16571

[options]
add.chr.as.prefix=FALSE
awhole.only=FALSE
suppress.evaluated.length.warning=FALSE
cut.all.over.length=FALSE
keep.distributions=FALSE
showTkProgressBar=FALSE
showProgressBar=FALSE

[tests]
ecdf.area.permut.number=10
mean.distance.permut.number=10
jaccard.measure.permut.number=10
random.seed=1248312
```

## 2.7 Mapping

If we test the correlation of a reference and query that are consistently found in the same chromosomal regions (e.g. in genes), we will always achieve what looks like extremely high correlation on a genomic scale and we are unable to ask questions about finer-scale associations. To use the Genometric Correlation statistical tests on data such as these, we can convert the regions (that the query and reference occupy) to pseudochromosomes, and then run the `GenometriCorrelation` test. The mapping is available in the package as `MapRangesToGenomicIntervals`.

The MapRangesToGenomicIntervals accepts two RangedData or GRanges, and maps the intervals from the second interval set that are contained in the first interval set, recalculating the coordinates so that within each new interval the coordinates range from 1 to the length of the interval.

Let's see how it works. In this artificial example, we create two random feature sets that reside only in bases $[1000000, 2000000]$ in a chromosome of length 3000000. First, we correlate these two features as is, not accounting for their restricted range. Next we will run the correlation again, this time using the mapping functions to test their relationship only within the $[1000000, 2000000]$ chromosomal range.

In the first test, the query and reference appear to be correlated, and all of the correlations disappear when the test is confined to the correct genomic interval, except the `relative.distances` family of tests that identify the features as independent in both cases.

```
> population <- 1000
> chromo.length <- c(3e+06)
> names(chromo.length) <- c("the_chromosome")
> rquery <- RangedData(ranges = IRanges(start = runif(population,
+     1e+06, 2e+06 - 10), width = c(10)),
+     space = "the_chromosome")
> rref <- RangedData(ranges = IRanges(start = runif(population,
+     1e+06, 2e+06 - 10), width = c(10)),
+     space = "the_chromosome")
> unmapped_result <- GenometriCorrelation(rquery,
+     rref, chromosomes.length = chromo.length,
+     ecdf.area.permut.number = pn.area,
+     mean.distance.permut.number = pn.dist,
+     jaccard.measure.permut.number = pn.jacc,
+     keep.distributions = FALSE, showProgressBar = FALSE)
> map_space <- RangedData(ranges = IRanges(start = c(1e+06),
+     end = c(2e+06)), space = "the_chromosome")
> mapped_rquery <- MapRangesToGenomicIntervals(what.to.map = rquery,
+     where.to.map = map_space)
> mapped_rref <- MapRangesToGenomicIntervals(what.to.map = rref,
+     where.to.map = map_space)
> mapped_result <- GenometriCorrelation(mapped_rquery,
+     mapped_rref, ecdf.area.permut.number = pn.area,
+     mean.distance.permut.number = pn.dist,
+     jaccard.measure.permut.number = pn.jacc,
+     keep.distributions = FALSE, showProgressBar = FALSE)
> cat("Unmapped result:\n")

Unmapped result:

> print(unmapped_result)

                                               the_chromosome
query.population                               1000
reference.population                           1000
query.coverage                                 9957
reference.coverage                             9948
relative.distances.ks.p.value                  0.7824768
relative.distances.ecdf.deviation.area         0.004737047
relative.distances.ecdf.area.correlation       0.01510766
query.reference.intersection                   82
query.reference.union                          19823
jaccard.measure                                0.004136609
projection.test.p.value                        0.007018046
projection.test.lower.tail                     FALSE
projection.test.obs.to.exp                     2.412545
scaled.absolute.min.distance.sum               168.399
relative.distances.ecdf.deviation.area.p.value 0.48
scaled.absolute.min.distance.sum.p.value       "<0.01"
scaled.absolute.min.distance.sum.lower.tail    TRUE
jaccard.measure.p.value                        0.01
jaccard.measure.lower.tail                     FALSE

> cat("Mapped result:\n")

Mapped result:

> print(mapped_result)

                                               the_chromosome:1000000-2000000
query.population                               1000
reference.population                           1000
query.coverage                                 9957
reference.coverage                             9948
relative.distances.ks.p.value                  0.7824768
relative.distances.ecdf.deviation.area         0.004438979
relative.distances.ecdf.area.correlation       0.01304585
query.reference.intersection                   82
query.reference.union                          19823
jaccard.measure                                0.004136609
projection.test.p.value                        0.3375977
projection.test.lower.tail                     TRUE
projection.test.obs.to.exp                     0.8041825
scaled.absolute.min.distance.sum               505.1965
relative.distances.ecdf.deviation.area.p.value 0.47
scaled.absolute.min.distance.sum.p.value       0.48
scaled.absolute.min.distance.sum.lower.tail    FALSE
jaccard.measure.p.value                        0.35
jaccard.measure.lower.tail                     TRUE
```

Mapping config example is `mapping-config.ini`:

```
> mapconfig <- new("GenometriCorrConfig",
+     system.file("extdata", "mapping-config.ini",
+         package = "GenometriCorr"))
> print(mapconfig)
```

```
[data]
query=UCSCcpgis_hg19.bed
query.format=bed
reference=UCSCrefseqgenes_hg19.bed
reference.format=bed
mapping=first_10000000.bed
mapping.format=bed
do.mapping=TRUE

[chromosomes]
chr1
chr2
chr3

[chromosomes.length]
chr1=249250621
chr2=243199373
chr3=198022430
chr4=191154276
chr5=180915260
chr6=171115067
chr7=159138663
chrX=155270560
chr8=146364022
chr9=141213431
chr10=135534747
chr11=135006516
chr12=133851895
chr13=115169878
chr14=107349540
chr15=102531392
chr16=90354753
```

```
chr17=81195210
chr18=78077248
chr20=63025520
chrY=59373566
chr19=59128983
chr22=51304566
chr21=48129895
chrM=16571

[options]
add.chr.as.prefix=FALSE
awhole.only=FALSE
suppress.evaluated.length.warning=FALSE
cut.all.over.length=FALSE
keep.distributions=FALSE
showTkProgressBar=FALSE
showProgressBar=FALSE

[tests]
ecdf.area.permut.number=100
mean.distance.permut.number=100
jaccard.measure.permut.number=100
random.seed=1248312
```

## 2.8 One more example: aligned reads in query

Let's try to work with aligned reads. The example file we contains reads from the ChIP-Seq Analysis of H3K4me3 in hESC H1 Cells, sample GSM433170, from GEO. The data is restricted to chr15:73842345-83842344 interval. We test the hypothesis of statistical collocation of the reads with RefSeq genes. As far as we are intersted only in the interval we have data for, we map both reads and genes to the interval.

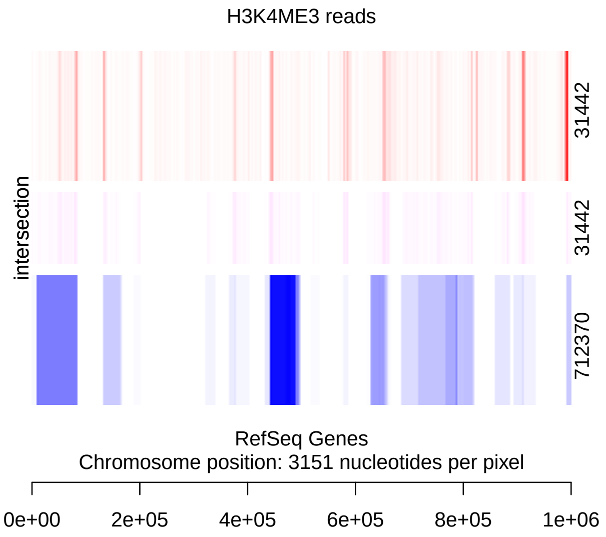
```
> reads <- import(con = system.file("extdata",
+     "GSM433170_BI.H1.H3K4me3.Solexa-8038_chr15.bed",
+     package = "GenometriCorr"), format = "bed")
> interval <- GRanges(seqnames = c("chr15"),
+     ranges = IRanges(start = c(73842345),
+         width = c(1e+07)))
> reads.in.interval <- MapRangesToGenomicIntervals(interval,
+     reads)
> genes.in.interval <- MapRangesToGenomicIntervals(interval,
+     refseq, unmapped.chromosome.warning = FALSE)
> H3K4Me3.vs.genes <- GenometriCorrelation(reads.in.interval,
+     genes.in.interval, showProgressBar = FALSE)
> cat("H3K4Me3 vs genes in chr15:73842345-83842344:\n")
```

```
H3K4Me3 vs genes in chr15:73842345-83842344:

> print(H3K4Me3.vs.genes)

                                             chr15:73842345-83842344
query.population                             15741
reference.population                         568
query.coverage                               1791354
reference.coverage                           5918085
relative.distances.ks.p.value                5.17808e-13
relative.distances.ecdf.deviation.area       0.007391792
relative.distances.ecdf.area.correlation     0.02950853
query.reference.intersection                 1049883
query.reference.union                        6659556
jaccard.measure                              0.1576506
projection.test.p.value                      0
projection.test.lower.tail                   FALSE
projection.test.obs.to.exp                   1.083659
scaled.absolute.min.distance.sum             21088.91
relative.distances.ecdf.deviation.area.p.value "<0.01"
scaled.absolute.min.distance.sum.p.value     "<0.01"
scaled.absolute.min.distance.sum.lower.tail  TRUE
jaccard.measure.p.value                      "<0.01"
jaccard.measure.lower.tail                   TRUE
```

Let's visualise first Mbp of the interval.

```
> VisualiseTwoIRanges(ranges(reads.in.interval),
+     ranges(genes.in.interval), end = 1e+06,
+     , nameA = "H3K4ME3 reads", nameB = "RefSeq Genes",
+     title = "H3K4Me3 vs RefGenes in chr15:73842345-83842344@Hg19")
```

**H3K4Me3 vs RefGenes in chr15:73842345−83842344@Hg19**



On the right: covered nucleotides that yields full color intensity of a pixel