Package Vignette for Genomic Interactions: ChIA-PET data

ChIA-PET

Chromatin interaction analysis with paired-end tag sequencing (ChIA-PET) is a recent method to study protein-mediated interactions at a genome-wide scale. Like most techniques for studying chromatin interaction it is based on chromosome conformation capture technology. Unlike 3C, 4C and 5C, however, it can detect interactions genome-wide, and includes a ChIP step to purify interactions involving a protein of interest.

The raw data from ChIA-PET is in the form of paired-end reads attached to one of two linker sequences. Reads with chimeric linkers are removed, and the data is aligned to the reference genome. The **ChIA-PET tool** can then be used to find pairs of regions ("anchors") which have a significant number of reads mapping between them and therefore represent biologically meaningful chromatin interactions in the sample.

Imports

First we need to load the GenomicInteractions package, and the mm9 reference genome:

```
library(GenomicInteractions)
library(GenomicRanges)
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     Filter, Find, Map, Position, Reduce, anyDuplicated, append,
##     as.data.frame, as.vector, cbind, colnames, do.call,
##     duplicated, eval, evalq, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, rank, rbind, rownames,
##     sapply, setdiff, sort, table, tapply, union, unique, unlist,
##     unsplit
```

```
## Loading required package: S4Vectors

## Loading required package: stats4

## Loading required package: IRanges

## Loading required package: GenomeInfoDb
```

Data

We can then read in our data directly from the output of the **ChIA-PET tool**. At this stage we can also provide information about the cell type and a description tag for the experiment. The data is taken from Li et al., 2012, published in **Cell**. They have used antibodies against the initiation form of Pol II, which you would expect to find at active promoters, and we are looking at data from the K562 myelogenous leukemia cell line. The data should therefore give us an insight into the processes which regulate genes that are being actively transcribed.

```
chiapet.data = system.file("extdata/k562.rep1.cluster.pet3+.txt",
                           package="GenomicInteractions")

k562.rep1 = makeGenomicInteractionsFromFile(chiapet.data,
                                type="chiapet.tool",
                                experiment_name="k562",
                                description="k562 pol2 8wg16")
```

This loads the data into a `GenomicInteractions` object, which consists of two linked `GenomicRanges` objects containing the anchors in each interaction, as well as the p-value, FDR and the number of reads supporting each interaction.

GenomicInteractions Objects

The metadata we have added can easily be accesed, and edited:

```
name(k562.rep1)
```

```
## [1] "k562"
```

```
description(k562.rep1) = "PolII-8wg16 Chia-PET for K562"
```

As can the data from the ChIA-PET experiment:

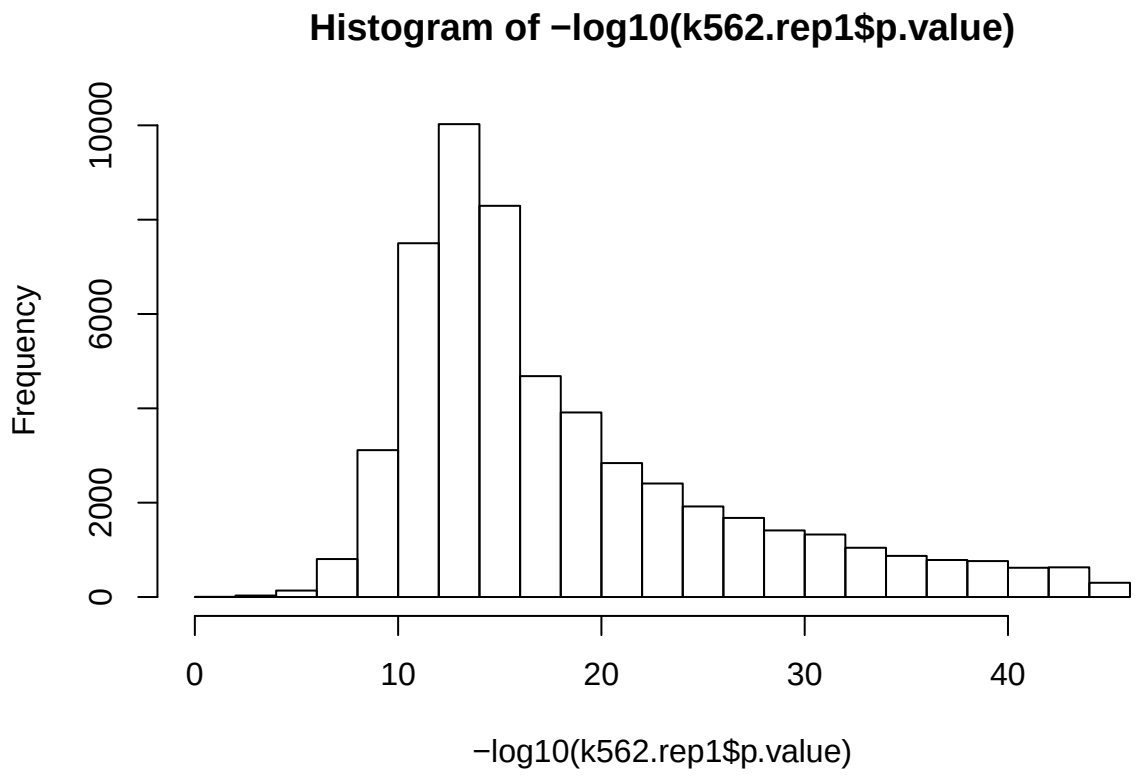
```
head(interactionCounts(k562.rep1))
```

```
## [1]   3 562   3   3   3   3
```

```
head((k562.rep1)$fdr)
```

```
## [1] 1.25703e-10 0.00000e+00 1.17148e-06 4.86859e-08 2.76777e-08 3.97019e-08
```

```
hist(-log10(k562.rep1$p.value))
```



The two linked `GRanges` objects can be returned, but not altered in-place:

```
anchorOne(k562.rep1)
```

```
## GRanges object with 64565 ranges and 0 metadata columns:
##           seqnames                ranges strand
##              <Rle>             <IRanges>  <Rle>
##       [1]     chr1       [569922, 571422]      *
##       [2]     chr1       [832761, 905482]      *
##       [3]     chr1       [839092, 842325]      *
##       [4]     chr1       [839393, 841792]      *
##       [5]     chr1       [852731, 855234]      *
##       ...      ...                    ...    ...
##   [64561]     chrX [154432946, 154435728]      *
##   [64562]     chrX [154436728, 154439876]      *
##   [64563]     chrX [154439789, 154442306]      *
##   [64564]     chrX [154459648, 154462031]      *
##   [64565]     chrX [154839050, 154843949]      *
##   -------
##   seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

```
anchorTwo(k562.rep1)
```

```
## GRanges object with 64565 ranges and 0 metadata columns:
##           seqnames                ranges strand
##              <Rle>             <IRanges>  <Rle>
##       [1]     chrM       [  8342,  10675]      *
##       [2]     chr1       [838470, 920603]      *
##       [3]     chr1       [935528, 939051]      *
##       [4]     chr1       [955081, 956755]      *
##       [5]     chr1       [933685, 937006]      *
##       ...      ...                    ...    ...
##   [64561]     chrX [154442294, 154446983]      *
##   [64562]     chrX [154442540, 154445105]      *
##   [64563]     chrX [154448371, 154451728]      *
##   [64564]     chrX [154469339, 154471852]      *
##   [64565]     chrX [154843728, 154848393]      *
##   -------
##   seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

`GenomicInteractions` objects can easily handle interactions detected between chromosomes, known as *trans*-chromosomal interactions, since the anchors can be at any point along the genome. `is.trans` returns a logical vector; likewise `is.cis` is the opposite of this function.

```
sprintf("Percentage of trans-chromosomal interactions %.2f",
        100*sum(is.trans(k562.rep1))/length(k562.rep1))
```

```
## [1] "Percentage of trans-chromosomal interactions 1.00"
```

The length of each interaction is not stored as metadata, but we can calculate the distance of each interaction using either the inner edge, outer edge or midpoints of the anchors. This is undefined for inter-chromosomal interactions, so NA is returned, so it is important to exclude these interactions from some analyses.

```
head(calculateDistances(k562.rep1, method="midpoint"))
```

```
## [1]     NA  10414  96580 115324  81362  79097
```

`GenomicRanges` objects can be subsetted by either integer or logical vectors like most R objects, and also BioConductor `Rle` objects.

```
k562.rep1[1:10] # first interactions in the dataset
```

```
## GenomicInteractions object with 10 interactions and 2 metadata columns:
##  Name: k562
##  Description: PolII-8wg16 Chia-PET for K562
##  Sum of interactions: 624
##  Annotated:  no
##  Interactions:
##                 Anchor One              Anchor Two Counts |     p.value
##    [1] chr1:569922..571422 ---    chrM:8342..10675      3 |  1.6214e-12
##    [2] chr1:832761..905482 --- chr1:838470..920603    562 |           0
```

```
##    [3] chr1:839092..842325 --- chr1:935528..939051      3 | 4.21364e-08
##    [4] chr1:839393..841792 --- chr1:955081..956755      3 | 1.45938e-09
##    [5] chr1:852731..855234 --- chr1:933685..937006      3 | 7.85539e-10
##    [6] chr1:855856..858861 --- chr1:935669..937245      3 | 1.16802e-09
##    [7] chr1:874165..879175 --- chr1:933340..938306     10 | 1.23139e-25
##    [8] chr1:874190..877867 --- chr1:955674..959630      5 | 6.63691e-15
##    [9] chr1:889676..896594 --- chr1:933897..938982     13 | 4.91311e-36
##   [10] chr1:898753..907581 --- chr1:931133..939571     19 |           0
##                fdr
##    [1] 1.25703e-10
##    [2]           0
##    [3] 1.17148e-06
##    [4] 4.86859e-08
##    [5] 2.76777e-08
##    [6] 3.97019e-08
##    [7] 3.58932e-23
##    [8] 6.98795e-13
##    [9] 2.33753e-33
##   [10]           0
##
##    -------
##   seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

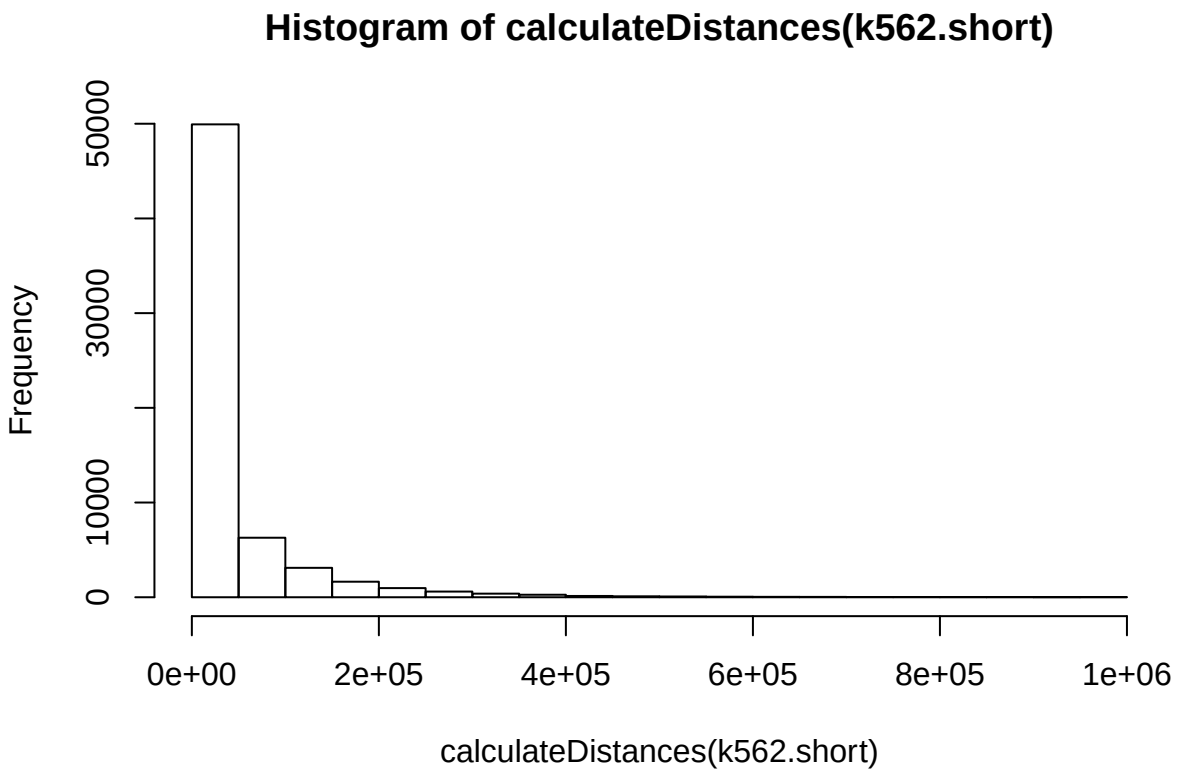
```
# k562.rep1[sample(length(k562.rep1), 100)] # 100 interactions subsample
k562.cis = k562.rep1[is.cis(k562.rep1)]
```

The length of each interaction is not stored as metadata, but we can calculate the distance of each interaction using either the inner edge, outer edge or midpoints of the anchors. Since this is undefinable for *trans*-chromosomal interactions it is best to first subset only *cis* interactions before calling `calculateDistances`, otherwise `NA`s will be present in the returned vector.

```
head(calculateDistances(k562.cis, method="midpoint"))
```

```
## [1]  10414  96580 115324  81362  79097  59152
```

```
k562.short = k562.cis[calculateDistances(k562.cis) < 1e6] # subset shorter interactions
hist(calculateDistances(k562.short))
```



We can also subset based on the properties of the linked `GRanges` objects.

```
chrom = c("chr17", "chr18")
sub = as.vector(seqnames(anchorOne(k562.rep1)) %in% chrom & seqnames(anchorTwo(k562.rep1)) %in% chrom)
k562.rep1 = k562.rep1[sub]
```

Annotation

Genomic Interaction data is often used to look at the interactions between different elements in the genome, which are believed to have different functional roles. Interactions between promoters and their transcription termination sites, for example, are thought to be a by-product of the transcription process, whereas long-range interactions with enhancers play a role in gene regulation.

Since `GenomicInteractions` is based on `GenomicRanges`, it is very easy to interrogate `GenomicInteractions` objects using `GenomicRanges` data. In the example, we want to annotate interactions that overlap the promoters, transcription termination sites or the body of any gene. Since this can be a time-consuming and data-heavy process, this example runs the analysis for only chromosomes 17 & 18.

First we need the list of RefSeq transcripts:

```
library(GenomicFeatures)

hg19.refseq.db <- makeTxDbFromUCSC(genome="hg19", table="refGene")
refseq.genes = genes(hg19.refseq.db)
refseq.transcripts = transcriptsBy(hg19.refseq.db, by="gene")
```

```
non_pseudogene = names(refseq.transcripts) %in% unlist(refseq.genes$gene_id)
refseq.transcripts = refseq.transcripts[non_pseudogene]
```

Rather than downloading the whole Refseq database, these are provided for chromosomes 17 & 18:

```
data("hg19.refseq.transcripts")
refseq.transcripts = hg19.refseq.transcripts
```

We can then use functions from `GenomicRanges` to call promoters and terminators for these transcripts. We have taken promoter regions to be within 2.5kb of an annotated TSS and terminators to be within 1kb of the end of an annotated transcript. Since genes can have multiple transcripts, they can also have multiple promoters/terminators, so these are `GRangesList` objects, which makes handling these objects slightly more complicated.

```
refseq.promoters = promoters(refseq.transcripts, upstream=2500, downstream=2500)
# unlist object so "strand" is one vector
refseq.transcripts.ul = unlist(refseq.transcripts)
# terminators can be called as promoters with the strand reversed
strand(refseq.transcripts.ul) = ifelse(strand(refseq.transcripts.ul) == "+", "-", "+")
refseq.terminators.ul = promoters(refseq.transcripts.ul, upstream=1000, downstream=1000)
# change back to original strand
strand(refseq.terminators.ul) = ifelse(strand(refseq.terminators.ul) == "+", "-", "+")
# `relist' maintains the original names and structure of the list
refseq.terminators = relist(refseq.terminators.ul, refseq.transcripts)
```

These can be used to subset a `GenomicInteractions` object directly from `GRanges` using the `GenomicRanges` overlaps methods. `findOverlaps` called on a `GenomicInteractions` object will return a list containing `Hits` objects for both anchors.

We can finds any interactions involving a RefSeq promoter:

```
subsetByFeatures(k562.rep1, refseq.promoters)
```

```
## GenomicInteractions object with 2907 interactions and 2 metadata columns:
##  Name: k562
##  Description: PolII-8wg16 Chia-PET for K562
##  Sum of interactions: 58468
##  Annotated:  no
##  Interactions:
##                        Anchor One                    Anchor Two Counts   |
##      [1] chr18:32867581..32873274 --- chr18:32922822..32925514      7   |
##      [2] chr18:32868753..32872112 --- chr18:32951673..32954977      4   |
##      [3] chr18:32869486..32873870 --- chr18:32874778..32879603     13   |
##      [4] chr18:32869839..32873068 --- chr18:32879912..32884536      4   |
##      [5] chr18:47003048..47019610 --- chr18:47008665..47025005     85   |
##      ...                      ... ...                      ...    ... ...
##   [2903] chr17:65222692..65227179 --- chr17:65239229..65241334      4   |
##   [2904] chr17:65231015..65237268 --- chr17:65235900..65244024     12   |
##   [2905] chr17:45248915..45264031 --- chr17:45257291..45268317     43   |
##   [2906] chr17:60130322..60137699 --- chr17:60137079..60144580     21   |
##   [2907]     chr18:227929..232043 ---     chr18:265870..268355      3   |
##              p.value         fdr
```

```
##      [1] 2.51251e-23 6.28044e-21
##      [2] 4.69817e-16 5.97214e-14
##      [3]           0           0
##      [4] 3.61832e-16 4.67152e-14
##      [5]           0           0
##      ...         ...         ...
##   [2903] 3.21725e-18 5.20359e-16
##   [2904]           0           0
##   [2905]           0           0
##   [2906]           0           0
##   [2907] 1.57957e-11 9.55328e-10
##
##    -------
##   seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

However, one of the most powerful features in the `GenomicInteractions` package is the ability to annotate each anchor with a list of genomic regions and then summarise interactions according to these features. This annotation is implemented as metadata columns for the anchors in the `GenomicInteractions` object and so is fast, and facilitates more complex analyses.

The order in which we annotate the anchors is important, since each anchor can only have one `node.class`. The first listed take precedence. Any regions not overlapping ranges in `annotation.features` will be labelled as `distal`.

```
annotation.features = list(promoter=refseq.promoters,
                           terminator=refseq.terminators,
                           gene.body=refseq.transcripts)
annotateInteractions(k562.rep1, annotation.features)
```

```
## Annotating with promoter ...

## Annotating with terminator ...

## Annotating with gene.body ...
```

```
annotationFeatures(k562.rep1)
```

```
## [1] "gene.body"  "promoter"   "distal"     "terminator"
```

We can now find interactions involving promoters using the annotated `node.class` for each anchor:

```
p.one = anchorOne(k562.rep1)$node.class == "promoter"
p.two = anchorTwo(k562.rep1)$node.class == "promoter"
k562.rep1[p.one|p.two]
```

```
## GenomicInteractions object with 2907 interactions and 2 metadata columns:
##  Name: k562
##  Description: PolII-8wg16 Chia-PET for K562
##  Sum of interactions: 58468
##  Annotated:  yes
##       Annotated with:  promoter, gene.body, distal, terminator
```

```
##  Interactions:
##                         Anchor One                    Anchor Two Counts   |
##      [1]     chr17:616579..621961 ---     chr17:620668..626263      7   |
##      [2]     chr17:632527..638035 ---     chr17:636589..641349      9   |
##      [3]     chr17:634119..651606 ---     chr17:642299..659172     55   |
##      [4]     chr17:654892..657597 ---     chr17:683191..687275      6   |
##      [5]     chr17:656002..658841 ---     chr17:679595..682692      3   |
##      ...                      ... ...                      ...    ... ...
##   [2903] chr18:77781151..77783476 --- chr18:77792968..77795855      3   |
##   [2904] chr18:77784590..77787797 --- chr18:77792148..77795822      6   |
##   [2905] chr18:77792093..77797983 --- chr18:77797455..77803127     13   |
##   [2906] chr18:77793365..77797939 --- chr18:77864889..77868321      8   |
##   [2907] chr18:77863992..77870413 --- chr18:77868294..77877151     18   |
##              p.value         fdr
##      [1] 2.06358e-32 8.50563e-30
##      [2]   2.395e-36  1.1518e-33
##      [3]           0           0
##      [4] 4.86283e-24 1.27856e-21
##      [5] 6.23098e-14 5.72161e-12
##      ...         ...         ...
##   [2903] 8.97548e-14 8.09366e-12
##   [2904] 4.16341e-26 1.24576e-23
##   [2905]           0           0
##   [2906] 6.61618e-30 2.41717e-27
##   [2907]           0           0
##
##    -------
##   seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

This information can be used to categorise interactions into promoter-distal, promoter-terminator etc. A table of interaction types can be generated with `categoriseInteractions`:

```
categoriseInteractions(k562.rep1)
```

```
##                 category count
## 1    gene.body-gene.body   795
## 2     gene.body-promoter   917
## 3       gene.body-distal    76
## 4   gene.body-terminator   164
## 5      promoter-promoter  1187
## 6        promoter-distal   519
## 7    promoter-terminator   284
## 8          distal-distal   396
## 9      distal-terminator   101
## 10 terminator-terminator    70
```

Alternatively, we can subset the object based on interaction type:

```
k562.rep1[isInteractionType(k562.rep1, "terminator", "gene.body")]
```

```
## GenomicInteractions object with 164 interactions and 2 metadata columns:
##  Name: k562
```

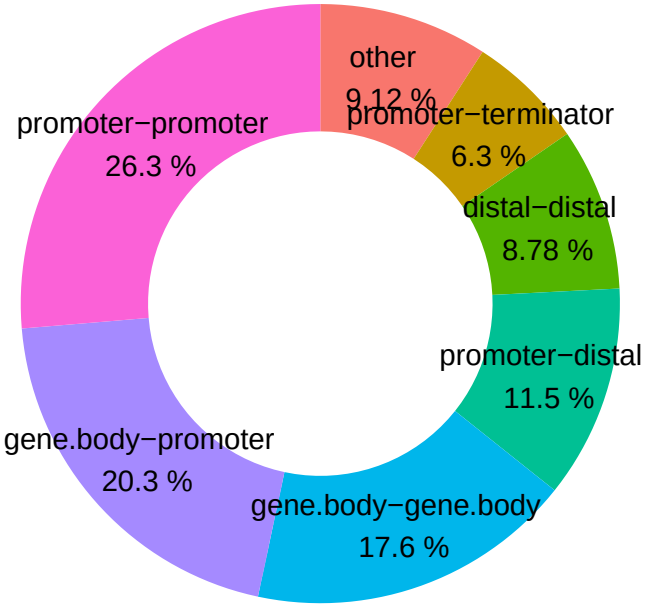
```
##  Description: PolII-8wg16 Chia-PET for K562
##  Sum of interactions: 930
##  Annotated:  yes
##       Annotated with:  terminator, gene.body
##  Interactions:
##                        Anchor One                    Anchor Two Counts   |
##     [1]   chr17:1471460..1474306 ---   chr17:1476212..1479585      4   |
##     [2]   chr17:1632603..1638741 ---   chr17:1636657..1642967     12   |
##     [3]   chr17:3845975..3849573 ---   chr17:3908008..3910817      5   |
##     [4]   chr17:4055645..4058706 ---   chr17:4063341..4068158      4   |
##     [5]   chr17:4443889..4451615 ---   chr17:4446814..4454998     11   |
##     ...                       ... ...                       ...    ... ...
##   [160] chr18:32873043..32876330 --- chr18:32911004..32914907      4   |
##   [161] chr18:33566139..33569380 --- chr18:33571111..33574360      3   |
##   [162] chr18:33689982..33692774 --- chr18:33695495..33699363      4   |
##   [163] chr18:42638745..42641928 --- chr18:42647110..42649707      3   |
##   [164] chr18:77914125..77916781 --- chr18:77919050..77922761      3   |
##              p.value         fdr
##     [1]  1.1438e-18 1.92712e-16
##     [2]           0 4.06377e-44
##     [3] 1.46527e-19 2.69952e-17
##     [4] 4.96495e-19 8.66918e-17
##     [5]  7.2293e-42 4.17953e-39
##     ...         ...         ...
##   [160] 1.37802e-15 1.62547e-13
##   [161] 5.37957e-15 5.74333e-13
##   [162] 1.35703e-19  2.5096e-17
##   [163] 1.03659e-15 1.24713e-13
##   [164]  2.9097e-15 3.24878e-13
##
##    -------
##   seqinfo: 25 sequences from an unspecified genome; no seqlengths
```

The 3 most common `node.class` values have short functions defined for convenience (see `?is.pp` for a complete list):

```
k562.rep1[is.pp(k562.rep1)] # promoter-promoter interactions
k562.rep1[is.dd(k562.rep1)] # distal-distal interactions
k562.rep1[is.pt(k562.rep1)] # promoter-terminator interactions
```

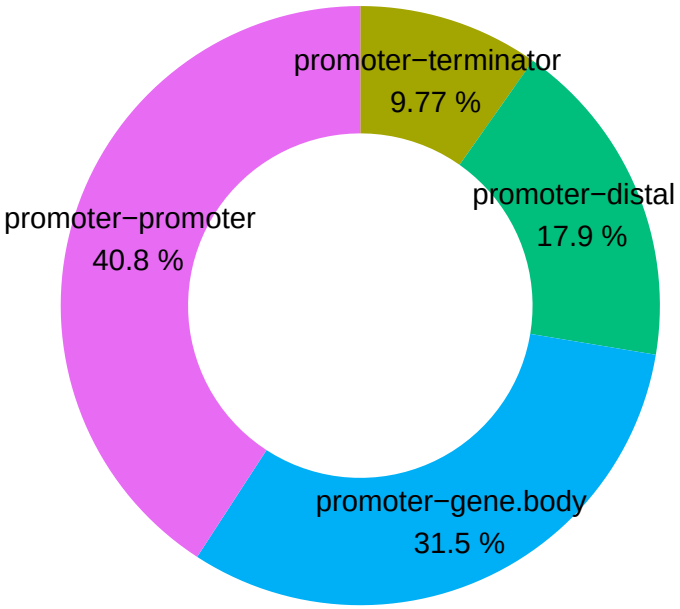
Summary plots of interactions classes can easily be produced to get an overall feel for the data:

```
plotInteractionAnnotations(k562.rep1, other=5)
```



`viewpoints` will only take those interactions with a certain `node.class`:

```
plotInteractionAnnotations(k562.rep1, other=5, viewpoints="promoter")
```



These are also combined in the function `plotSummaryStats`.

Feature Summaries

The `summariseByFeatures` allows us to look in more detail at interactions involving a specific set of loci. In this example we use all RefSeq promoters, which we already have loaded in a `GRangesList` object.

It is however possible to use any dataset which can be represented as a named `GRanges` object, for example transcription-factor ChIP data, predicted cis-regulatory sites or certain categories of genes.

The categories are generated automatically from the annotated `node.class` values in the object.

```
k562.rep1.promoter.annotation = summariseByFeatures(k562.rep1, refseq.promoters,
                                                    "promoter", distance.method="midpoint",
                                                    annotate.self=TRUE)
colnames(k562.rep1.promoter.annotation)
```

```
##  [1] "Promoter.id"
##  [2] "numberOfPromoterInteractions"
##  [3] "numberOfPromoterUniqueInteractions"
##  [4] "numberOfPromoterInterChromosomalInteractions"
##  [5] "numberOfPromoterUniqueInterChromosomalInteractions"
##  [6] "numberOfPromoterGene.bodyInteractions"
##  [7] "numberOfPromoterPromoterInteractions"
##  [8] "numberOfPromoterDistalInteractions"
##  [9] "numberOfPromoterTerminatorInteractions"
```

```
## [10] "numberOfUniquePromoterGene.bodyInteractions"
## [11] "numberOfUniquePromoterPromoterInteractions"
## [12] "numberOfUniquePromoterDistalInteractions"
## [13] "numberOfUniquePromoterTerminatorInteractions"
## [14] "PromoterDistanceMedian"
## [15] "PromoterDistanceMean"
## [16] "PromoterDistanceMinimum"
## [17] "PromoterDistanceMaximum"
## [18] "PromoterDistanceWeightedMedian"
## [19] "numberOfSelfPromoterGene.bodyInteractions"
## [20] "numberOfSelfPromoterPromoterInteractions"
## [21] "numberOfSelfPromoterTerminatorInteractions"
## [22] "numberOfSelfUniquePromoterGene.bodyInteractions"
## [23] "numberOfSelfUniquePromoterPromoterInteractions"
## [24] "numberOfSelfUniquePromoterTerminatorInteractions"
```

This allows us to very quickly generate summaries of the data and provides a quick method to isolate genes of interest. In this case we produce lists of RefSeq IDs, which can easily be converted to EntrezIDs or gene symbols through existing BioConductor packages (in this case `org.Hs.eg.db` provides bimaps between common human genome annotations).

Which promoters have the strongest Promoter-Promoter interactions based on PET-counts?

```
i = order(k562.rep1.promoter.annotation$numberOfPromoterPromoterInteractions,
          decreasing=TRUE)[1:10]
k562.rep1.promoter.annotation[i,"Promoter.id"]
```

```
##  [1] "100506779" "9256"      "406934"    "54894"     "100616220"
##  [6] "6827"      "56155"     "5889"      "5034"      "396"
```

Which promoters are contacting the largest number of distal elements?

```
i = order(k562.rep1.promoter.annotation$numberOfUniquePromoterDistalInteractions,
          decreasing=TRUE)[1:10]
k562.rep1.promoter.annotation[i,"Promoter.id"]
```

```
##  [1] "10140"     "400604"    "7050"      "100130581" "100616277"
##  [6] "26118"     "100874261" "101927666" "140735"    "5366"
```

What percentage of promoters are in contact with transcription termination sites?

```
total = sum(k562.rep1.promoter.annotation$numberOfPromoterTerminatorInteractions > 0)
sprintf("%.2f%% of promoters have P-T interactions", 100*total/nrow(k562.rep1.promoter.annotation))
```

```
## [1] "16.43% of promoters have P-T interactions"
```

References

1. Li, Guoliang, et al. "Software ChIA-PET tool for comprehensive chromatin interaction analysis with paired-end tag sequencing." Genome Biol 11 (2010): R22.
2. Li, Guoliang, et al. "Extensive promoter-centered chromatin interactions provide a topological basis for transcription regulation." Cell 148.1 (2012): 84-98