

# GENOVA: explore the Hi-C's

***Robin H. van der Weide<sup>1</sup> and Elzo de Wit<sup>\*1</sup>***

<sup>1</sup>Division of Gene Regulation, the Netherlands Cancer Institute

<sup>\*</sup>e.d.wit@nki.nl

**2018-01-18**

## **Abstract**

The increase in interest for Hi-C methods in the chromatin community has led to a need for more user-friendly and powerful analysis methods. The few currently available software packages for Hi-C do not allow a researcher to quickly summarize and visualize their data. An easy to use software package, which can generate a comprehensive set of publication-quality plots, would allow researchers to swiftly go from raw Hi-C data to interpretable results. Here, we present **GENome Organisation Visual Analytics** (GENOVA): a software suite to perform in-depth analyses on various levels of genome organisation, using Hi-C data. GENOVA facilitates the comparison between multiple datasets and supports the majority of mapping-pipelines.

## Contents

|     |  |    |
|-----|--|----|
| 1   | Loading data . . . . .                 | 3  |
| 1.1 | Data structures of input . . . . .     | 3  |
| 1.2 | Recommended resolutions . . . . .      | 3  |
| 1.3 | construct.experiment . . . . .         | 4  |
| 1.4 | Juicebox . . . . .                     | 6  |
| 2   | Genome-wide analyses . . . . .         | 6  |
| 2.1 | <i>Cis</i> -quantification . . . . .   | 6  |
| 2.2 | chromosome plots . . . . .             | 7  |
| 2.3 | RCP . . . . .                          | 7  |
| 3   | Interaction plots . . . . .            | 11 |
| 3.1 | <i>cis</i> -interactions . . . . .     | 11 |
| 3.2 | <i>trans</i> -interactions . . . . .   | 11 |
| 3.3 | matrix plots . . . . .                 | 12 |
| 4   | TADs . . . . .                         | 20 |
| 4.1 | Insulation . . . . .                   | 20 |
| 4.2 | Call TADs . . . . .                    | 23 |
| 4.3 | ATA . . . . .                          | 23 |
| 4.4 | TAD+N . . . . .                        | 25 |
| 5   | Loops . . . . .                        | 27 |
| 5.1 | APA . . . . .                          | 27 |
| 6   | Far- <i>cis</i> interactions . . . . . | 28 |
| 6.1 | PE-SCAn . . . . .                      | 28 |
| 6.2 | centromere.telomere.analysis . . . . . | 30 |
| 7   | To-do . . . . .                        | 31 |
| 8   | Session info . . . . .                 | 31 |
|     | References . . . . .                   | 33 |



## 1 Loading data

```
library(GENOVA)
```

### 1.1 Data structures of input

GENOVA expects two input files: the signal- and the index-file. Signal-files have three columns (bin1, bin2, contactCount) and index-files have four (chromosome, start, end, bin). These are the default output of the Hi-C mapping pipeline HiC-Pro (Servant et al. 2015), where they are called \*.matrix and \*.bed. The files are expected to be genome-wide and may be corrected with ICE-normalisation.

### 1.2 Recommended resolutions

To ensure computational strain and time is kept to a minimum, we recommend different resolutions for different functions (table 1). More experienced users are free to deviate, while keeping in mind that these datasets are quite memory-heavy (table 2).

**Table 1: Recommended resolutions**

These will provide optimal resource/result tradeoffs.

| Function                 | Resolution                          |
|--------------------------|-------------------------------------|
| APA                      | 10kb-20kb                           |
| ATA                      | 10kb-40kb                           |
| cisTotal.perChrom        | 500kb-1Mb                           |
| chromosomeMatrix         | 500kb-1Mb                           |
| RCP                      | 40kb-500kb                          |
| intra.inter.TAD.contacts | 20kb - 40kb                         |
| PE-SCAn                  | 20kb-40kb                           |
| hic.matrixplot           | <i>width in bp of window</i><br>500 |

**Table 2: Memory footprints of objects loaded into R**

| Experiment | Resolution | Contacts (millions) | Memory (GB) |
|------------|------------|---------------------|-------------|
| Hap1 WT    | 10kb       | 433.5               | 2.9         |
| Hap1 WT    | 40kb       | 433.5               | 1.7         |
| Hap1 WT    | 100kb      | 433.5               | 1.1         |
| Hap1 WT    | 1Mb        | 433.5               | 0.1         |

## 1.3 construct.experiment

Every Hi-C experiment will be stored in an experiment-object. This is done by invoking the `construct.experiment` function. Inside, several sanity checks will be performed, data is normalised to the total number of reads and scaled to a billion reads (the default value of the `BPscaling`-option).

For this example, we are going to use the Hi-C maps of WT and  $\Delta$ WAPL Hap1 cells from Haarhuis et al. (2017). Since the genome-wide analyses do not need very high-resolution data, we will construct both 10kb, 40kb and 1Mb resolution experiment-objects.

```
Hap1_WT_10kb <- construct.experiment(
  signalPath = 'data/WT_10000_iced.matrix',
  indicesPath = 'data/WT_10000_abs.bed',
  name = "WT",
  centromeres = centromeres,
  color = "black")

Hap1_WAPL_10kb <- construct.experiment(
  signalPath = 'data/WAPL_10000_iced.matrix',
  indicesPath = 'data/WAPL_10000_abs.bed',
  name = "WAPL",
  centromeres = centromeres,
  color = "red")

Hap1_WT_40kb <- construct.experiment(
  signalPath = 'data/WT_40000_iced.matrix',
  indicesPath = 'data/WT_40000_abs.bed',
  name = "WT",
  centromeres = centromeres,
  color = "black")

Hap1_WAPL_40kb <- construct.experiment(
  signalPath = 'data/WAPL_40000_iced.matrix',
  indicesPath = 'data/WAPL_40000_abs.bed',
  name = "WAPL",
  centromeres = centromeres,
  color = "red")

Hap1_WT_1MB <- construct.experiment(
  signalPath = 'data/WT_1000000_iced.matrix',
  indicesPath = 'data/WT_1000000_abs.bed',
  name = "WT",
  centromeres = centromeres,
  color = "black")

Hap1_WAPL_1MB <- construct.experiment(
  signalPath = 'data/WAPL_1000000_iced.matrix',
  indicesPath = 'data/WAPL_1000000_abs.bed',
  name = "WAPL",
  centromeres = centromeres,
  color = "red")
```

## GENOVA: explore the Hi-C's

Several functions rely on centromere-information. You can add this in the form of a BED-like three-column data.frame when constructing the experiment-object.<sup>1</sup> If not present, the centromeres will be empirically identified by searching for the largest stretch of no coverage on a chromosome.

```
centromeres = read.delim('data/hg19_cytobandAcen.bed',
                         sep = '\t',
                         h = F,
                         stringsAsFactors = F)

head(centromeres)
##      V1      V2      V3
## 1 chr1 121500000 128900000
## 2 chr10 38000000 42300000
## 3 chr11 51600000 55700000
## 4 chr12 33300000 38200000
## 5 chr13 16300000 19500000
## 6 chr14 16100000 19100000
```

<sup>1</sup>Please make sure that the chromosome-names match.

The resulting object has several slots. *ICE* and *ABS* are the signal- and index-data.tables, resp., and *RES* is the automatically determined resolution of the Hi-C data. The *NAME*, *COL* and *COMM* are user-provided metadata vectors, where the latter is a free-from slot to store comments and/or output of different functions. The amount of contacts in the *ICE* data.table is likely different from the input-data, because it is scaled to a fixed number of reads (which can be set with the `BPscaling`-option in `construct.experiment`).

```
## List of 9
## $ ICE :Classes 'data.table' and 'data.frame': 105110621
##   obs. of 3 variables:
##   ..$ V1: int [1:105110621] 1 1 ...
##   ..$ V2: int [1:105110621] 1 16 ...
##   ..$ V3: num [1:105110621] 275 ...
##   ...- attr(*, ".internal.selfref")=<externalptr>
##   ...- attr(*, "sorted")= chr [1:2] "V1" ...
## $ ABS :'data.frame': 77404 obs. of 4 variables:
##   ..$ V1: chr [1:77404] "chrM" ...
##   ..$ V2: int [1:77404] 0 0 ...
##   ..$ V3: int [1:77404] 16571 40000 ...
##   ..$ V4: int [1:77404] 1 2 ...
## $ NAME : chr "WT"
## $ RES : num 40000
## $ CHRS : chr [1:25] "chrM" ...
## $ COL : chr "black"
## $ COMM : NULL
## $ MASK : logi(0)
## $ CENTROMERES:'data.frame': 24 obs. of 3 variables:
##   ..$ V1: chr [1:24] "chr1" ...
##   ..$ V2: int [1:24] 121500000 38000000 ...
##   ..$ V3: int [1:24] 128900000 42300000 ...
```

## 1.4 Juicebox

We added a convenience script **juicerToGENOVA.py**, to load files from Juicerbox (.hic files). This allows for a fast conversion to signal- and index-files from, for example, data from Sanborn et al.(2015):

```
# Convert data from Sanborn et al. normalised at 10kb resolution:
juicerToGenova.py -C ucsc.hg19_onlyRealChromosomes.noChr.chromSizes \
-JT ~/bin/juicer/AWS/scripts/juicebox_tools.7.0.jar \
-H ~/Downloads/Sanborn_Hap1_combined_30.hic \
-R 10000 \
-force TRUE \
-norm KR \
-o Sanborn_Hap1_combined_30.hic_10kb_KR
```

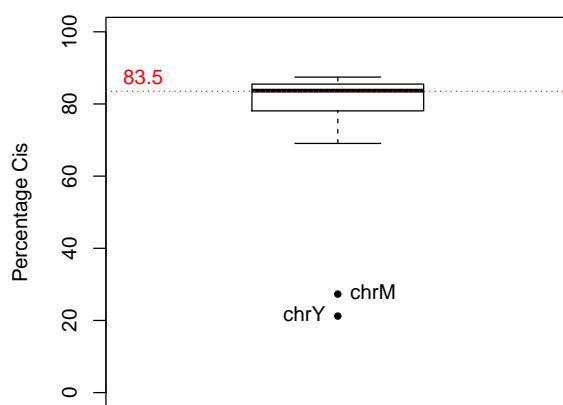
## 2 Genome-wide analyses

A good place to start your analyses are some functions on a genome-wide level. We can assess the quality of the library, identify translocations and generate contact probability (aka scaling or interaction decay plots).

### 2.1 Cis-quantification

Work by the group of Amos Tanay showed that the expected amount of intra-chromosomal contacts is the range of 90 to 93 percent (Olivares-Chauvet et al. 2016). Assuming that any extra inter-chromosomal contacts are due to debris/noise, the user might aspire to get the *cis*-percentages as close to 90% as possible. To compute the percentage of per-chromosome *cis*-contacts, we simply provide `cisTotal.perChrom` with the `exp`-object of interest. It will produce a boxplot of the percentages *cis* per chromosome and draw a red line with the genome-wide percentage. If you assign a variable to the output of this function, you will also get a list with the underlying data.

```
cisChrom_out <- cisTotal.perChrom(Hap1_WT_1MB)
```

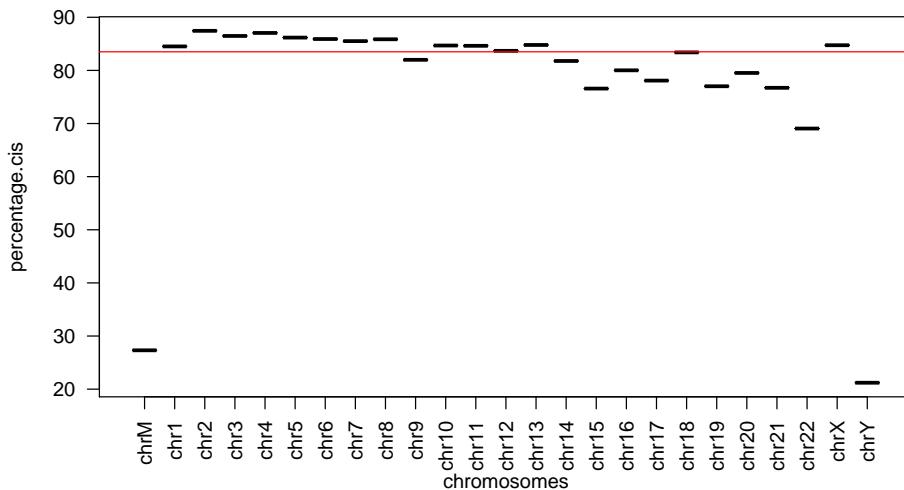


**Figure 1: Fraction of cis-contacts per chromosome**

## GENOVA: explore the Hi-C's

Using the underlying data stored in the variable `cisChrom_out`, we can also inspect the results per chromosome more closely. The list has two entries: a data.frame with the per-chromosome percentages (`perChrom`) and the genome-wide percentage (`genomeWide`).

```
plot(cisChrom_out$perChrom, las=2)
abline(h = cisChrom_out$genomeWide, col = 'red')
```



**Figure 2: Fraction of cis-contacts per chromosome**

Chromosomes 9, 15, 19 & 22 have translocations, which therefore appear to have more trans-contacts, but which in reality are cis-contacts.

## 2.2 chromosome plots

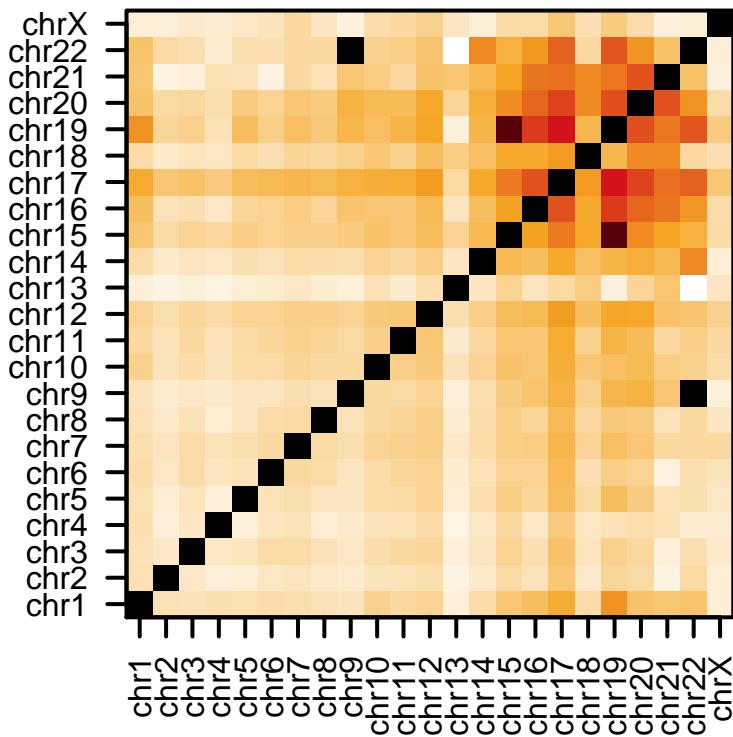
Hi-C has been shown to be a powerful data-source to detect chromosomal rearrangements (Harewood et al. 2017). To find possible translocations, we can plot the genome-wide enrichment of interactions between all combinations of chromosomes. The values in the matrix are  $\log_{10}(\text{observed}/\text{expected})$ . The Hap1 cell line has two known translocations, which we can easily see in the resulting plot. To narrow-in on this location, you could use the `trans.compartment.plot`-function (discussed below).

```
# Lets remove mitochondrial and Y-chromosomal contacts
chromosomeMatrix(Hap1_WT_1MB, remove = c("chrM", "chrY"))
```

## 2.3 RCP

The Relative Contact Probability computes the contact probability as a function of genomic distance, as described in (Lieberman-Aiden and Berkum 2009). This can be computed for a specific set of chromosomes or genome-wide. To be able to ignore centromeric contacts (which have a aberrant RCP), centromeric information is need. This is taken from the experiment-object or found emperically by comparing trans-interactions.

```
RCP_out135 <- RCP(experimentList = list(Hap1_WT_40kb, Hap1_WAPL_40kb),
                      chromsToUse = c('chr1', 'chr3', 'chr5'))
```

**Figure 3: Chromosome matrix**

The two known translocations of Hap1 cells are easily identified (15-19 & 9-22).

The user can decide to plot the RCP per chromosome. If the data is sparse, a LOESS-smoothing could be convenient. It takes the color and name from the experiment-objects. If we look at the resulting plot, we can see that the  $\Delta WAPL$  has more interactions in the  $[\pm 800kb, \pm 2Mb]$  range. The sizes of TADs are fall into this range, so a next step could be to dive into the TAD-specific analyses (discussed below). Moreover, the  $\Delta WAPL$  has less interactions in the far-*cis* range ( $[10Mb, 100Mb]$ ): A- and B-compartments are often of these sizes, so a next step could be to look more into compartmentalisation with `cis.compartment.plot` or `trans.compartment.plot`, for example.

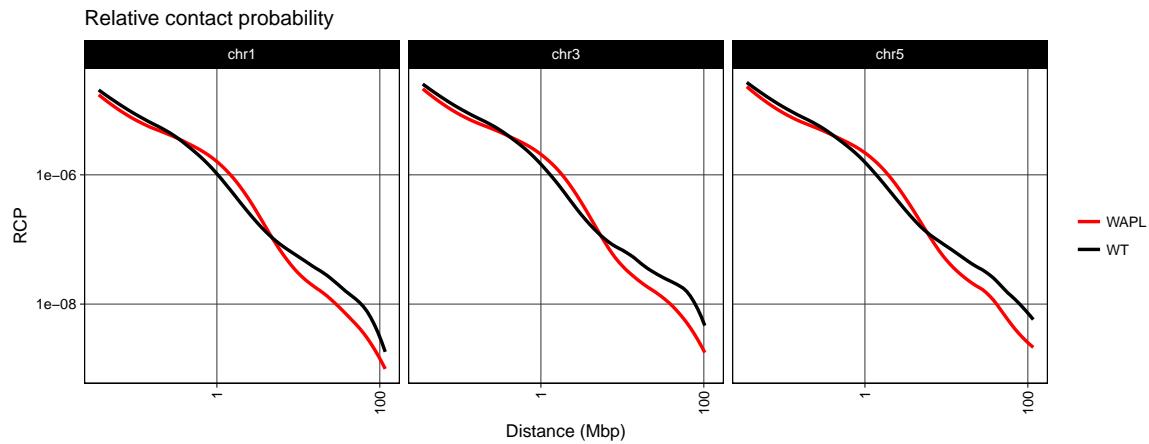
```
# Plot RCP: combined
visualise.RCP.ggplot(RCPdata = RCP_out135,
                      smooth = T, # use a LOESS smoothing
                      combine = F) # Don't merge data from all chromosomes
```

### 2.3.1 combined

It is also possible to combine all available data into a genome-wide RCP-plot.

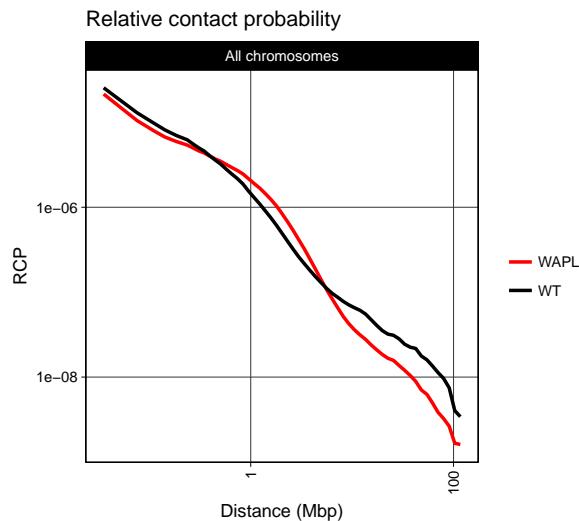
```
# Plot RCP: per-chromosome
visualise.RCP.ggplot(RCPdata = RCP_out135,
                      smooth = F, # do not use a LOESS smoothing
                      combine = T) # Merge data from all chromosomes
```

## GENOVA: explore the Hi-C's



**Figure 4: RCP**

Every facet shows the RCP of one chromosome.



**Figure 5: RCP**

All data combined in one plot.

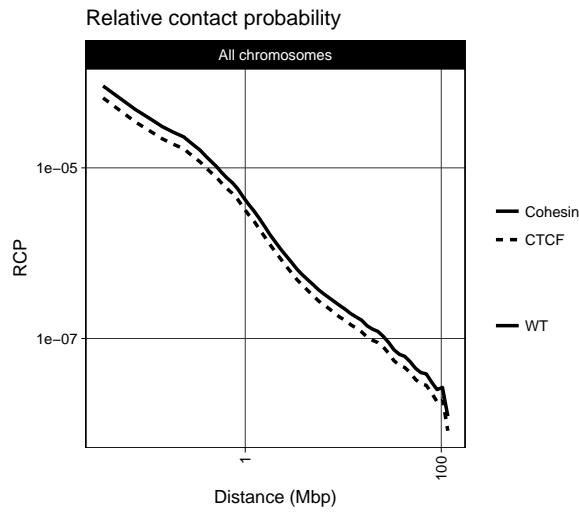
### 2.3.2 regions

```
CTCF = read.delim('data/CTCF_WT_motifs.bed', h = F)
SMC1 = read.delim('data/SMC1_WT_peaks.narrowPeak', h = F)

RCP_out = RCP(experimentList = list(Hap1_WT_40kb),
               bedList = list("CTCF" = CTCF,
                             'Cohesin' = SMC1),
               chrsToUse = c('chr1'))

visualise.RCP.ggplot(RCP_out)
```

## GENOVA: explore the Hi-C's



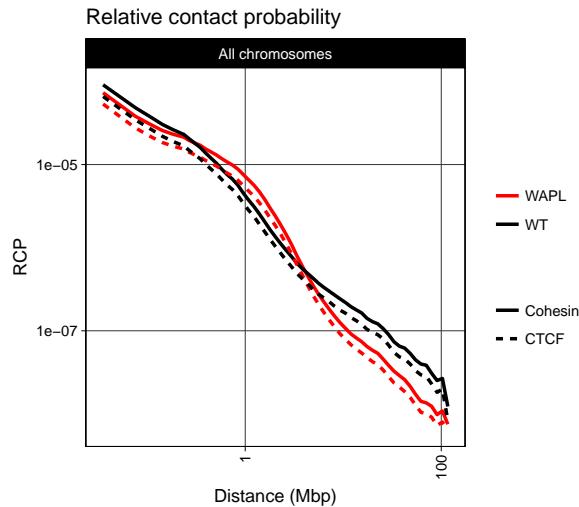
**Figure 6: RCP with BEDs**

We can also add BEDs as sites to compute the RCP.

```
CTCF = read.delim('data/CTCF_WT_motifs.bed', h = F)
SMC1 = read.delim('data/SMC1_WT_peaks.narrowPeak', h = F)

RCP_out = RCP(experimentList = list(Hap1_WT_40kb, Hap1_WAPL_40kb),
               bedList = list("CTCF" = CTCF,
                             'Cohesin' = SMC1),
               chromsToUse = c('chr1'))

visualise.RCP.ggplot(RCP_out)
```



**Figure 7: RCP with BEDs**

Multiple samples and BEDs can be combined.

... noting to see here, move along...

## 3 Interaction plots

... noting to see here, move along...

### 3.1 *cis*-interactions

```
cis.compartment.plot(exp = Hap1_WT_40kb,
                      chrom = 'chr20',
                      arm = 'q',
                      zlim = 20)
```

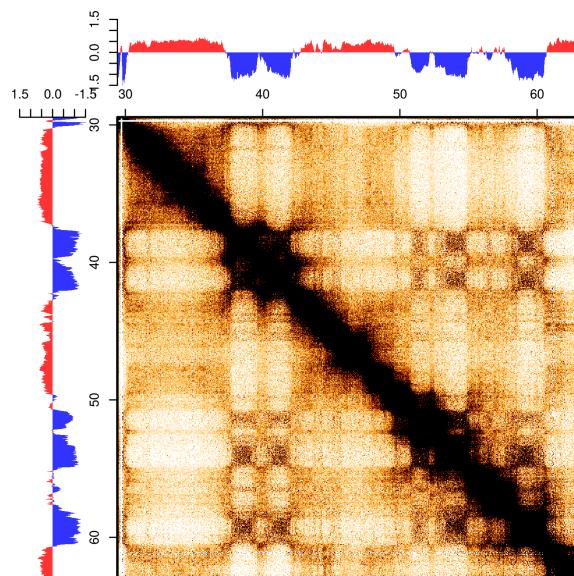
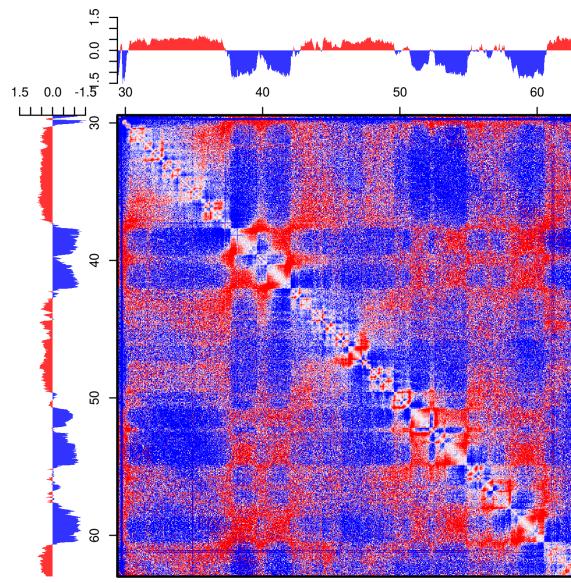


Figure 8: Cis compartment plot

```
cis.compartment.plot(exp = Hap1_WT_40kb,
                      chrom = 'chr20',
                      arm = 'q',
                      zlim = 1,
                      obs.exp = T)
```

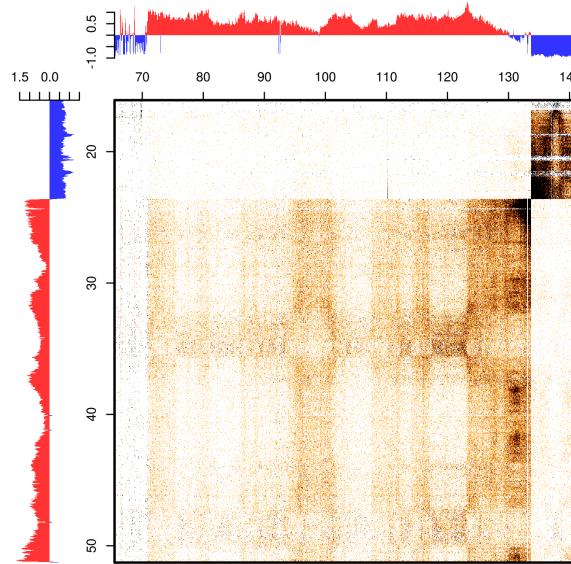
### 3.2 *trans*-interactions

```
trans.compartment.plot(exp = Hap1_WT_40kb,
                        chrom1 = 'chr9',
                        arm1 = 'q',
                        chrom2 = 'chr22',
                        arm2 = 'q',
                        zlim = 10)
```



**Figure 9: Cis compartment plot**

Observed over expected.



**Figure 10: Trans compartment plot**

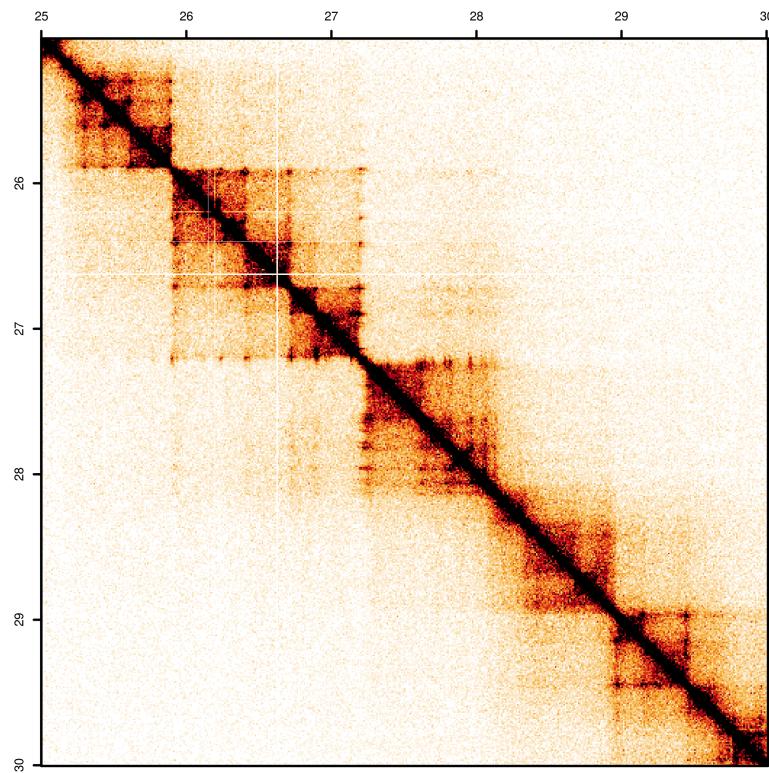
The t(9q;22q) translocation is easily identified.

### 3.3 matrix plots

To produce richly annotated zoomed-in (i.e. max 10Mb) plots of specific regions, we use the `hic.matrixplot` function. In this, we can use one or two experiment objects: two can be shown either in diff-mode (the difference between the two) or upper/lower triangle mode. TAD- and loop-annotations can be added, as well as bigwig- and bed-tracks. Moreover, genemodel-files can be added.

## GENOVA: explore the Hi-C's

```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               cut.off = 50) # upper limit of contacts
```



**Figure 11: Hi-C matrixplot**

Simplest example: one experiment, no annotation

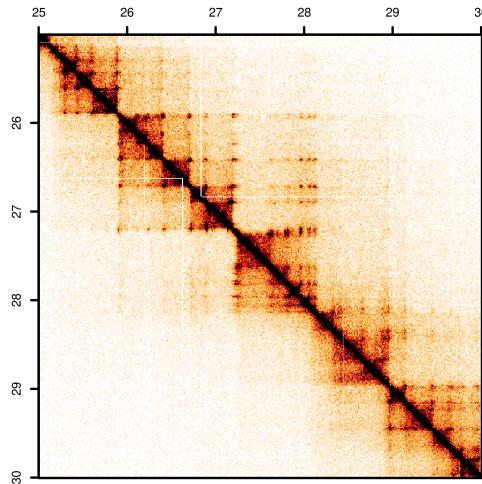
### 3.3.1 two experiments

Adding a second experiment will give us the option of `coplot`, which can be `dual` (default) or `diff`. The first shows `exp1` in the lower triangle and `exp2` in the upper. `Exp1` is subtracted from `exp2` in `diff`-mode: red is therefore more contacts in `exp2` and blue denotes more contacts in `exp1`.

```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               exp2 = Hap1_WAPL_10kb,
               chrom = 'chr7',
               start = 25e6,
```

## GENOVA: explore the Hi-C's

```
end=30e6,  
cut.off = 50) # upper limit of contacts
```



**Figure 12: Hi-C matrixplot: dual**

The extended loops in the WAPL knockout are easily seen at around 28Mb in the upper triangle.

```
hic.matrixplot(exp1 = Hap1_WT_10kb,  
              exp2 = Hap1_WAPL_10kb,  
              coplot = 'diff',  
              chrom = 'chr7',  
              start = 25e6,  
              end=30e6,  
              cut.off = 25) # upper limit of contacts
```

### 3.3.2 TADs and loops

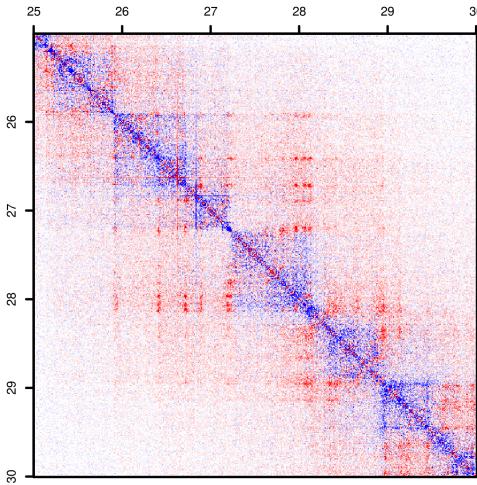
We use the TAD-calls of WT Hap1 20-kb matrices from Haarhuis et al. (2017), generated with HiCseg (Lévy-Leduc et al. 2014).

Lets load some TAD- and loop-annotations:

```
WT_TADs = read.delim('data/WT_hicseg_TADs.bed', h = F)  
WT_Loops = read.delim('data/WT_HICCUPS.bedpe', h = F, skip = 1)
```

Add them to the plot by using the `tad`- and `loops`-arguments. Both can be plotted in one or both of the traingles and colored as whished. Since loops are very small in a hic-matrixplot, they will be fully overlapped by the loop-annotations. To overcome this, we expand the annotations with a fixed bp using `loops.resize`. This will lead to a more box-like annotation surrounding the loop.

```
hic.matrixplot(exp1 = Hap1_WT_10kb,  
              chrom = 'chr7',
```

**Figure 13: Hi-C matrixplot: diff**

Note the ease of identifying the extended loops.

```

start = 25e6,
end=30e6,
loops = WT_Loops, # see APA
loops.color = '#998ec3', # purple loops
loops.type = 'upper', # only plot in upper triangle
loops.resize = 20e3, # expand for visibility
tads = WT_TADs, # see ATA
tad.type = 'lower', # only plot in lower triangle
tads.color = '#91cf60', # green TAD-borders
cut.off = 25) # upper limit of contacts

```

### 3.3.3 BigWigs and BEDs

Two tracks above and two tracks to the left can be added. These can be either BED-like data.frames or the paths the .bw files. For example, lets load a BED6-file (chrom, start, end, name, score, and strand<sup>2</sup>) of CTCF-motifs under CTCF-ChIP peaks. The argument `type` can be set to either `triangle` or `rectangle`: triangle is nice to use if you want to look at the orientation of the BED-entries.

```

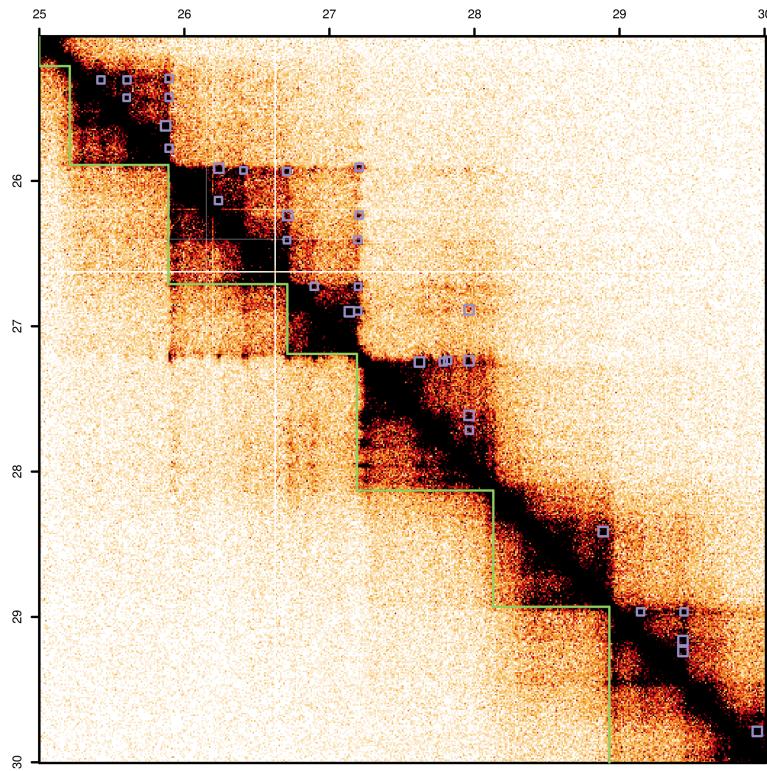
CTCF = read.delim('data/CTCF_WT_motifs.bed', h = F)
SMC1 = read.delim('data/SMC1_WT_peaks.narrowPeak', h = F)

```

**Table 3: A data.frame holding a standard BED6 format**

| V1   | V2     | V3     | V4                   | V5   | V6 |
|------|--------|--------|----------------------|------|----|
| chr1 | 237749 | 237768 | GCAGCACCAAGGTGGCAGCA | 1412 | +  |
| chr1 | 714180 | 714199 | CGGCCACCAGTAGGCAGCG  | 1428 | -  |
| chr1 | 793458 | 793477 | CCACCAGCAGGTGGCCTCC  | 1160 | -  |

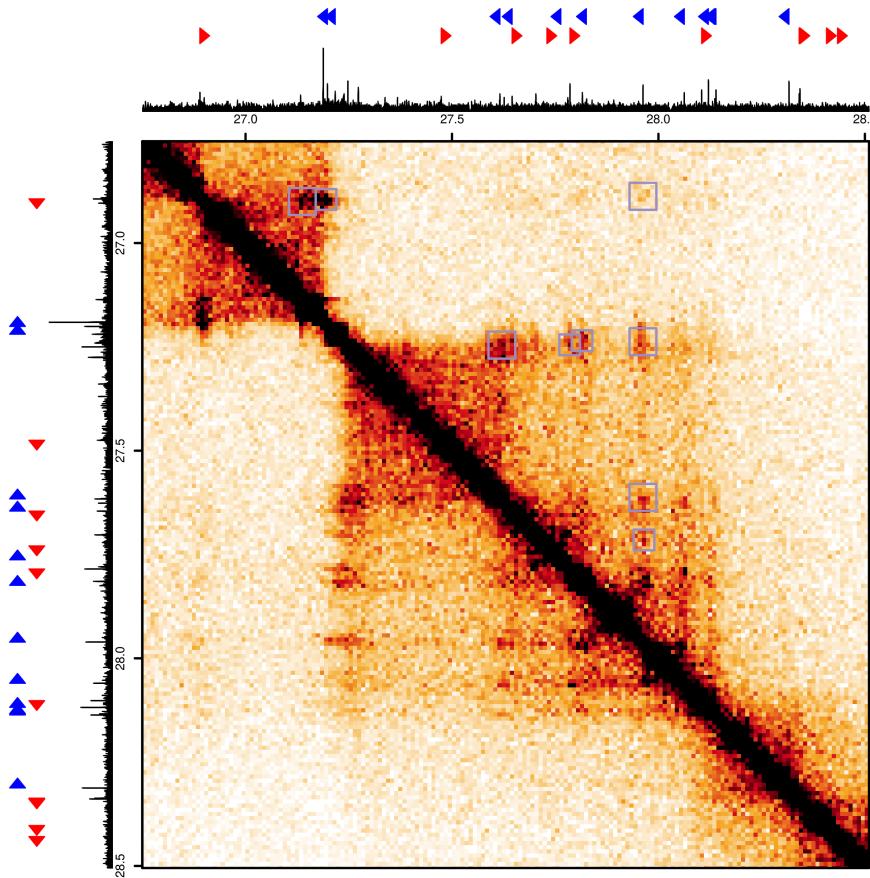
<sup>2</sup><https://genome.ucsc.edu/FAQ/FAQformat.html>



**Figure 14:** Hi-C matrixplot: TAD- and loop-annotations

Moreover, we can use a bigwig (.bw) file to plot a track. For this example, we are using a SMC1 ChIP-seq track from (Haarhuis et al. 2017). We need the `bigrIGR` package, which is easily installed from github using `devtools::install_github()`.

```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7', start = 26.75e6, end=28.5e6,
               loops = WT_Loops, # see APA
               loops.color = '#998ec3', # purple loops
               loops.type = 'upper', # only plot in upper triangle
               loops.resize = 20e3, # expand for visibility
               chip = list('data/SMC1_WT.bw', # inner top
                           CTCF),# outer-top
               symmAnn = T, # place annotations also on left side
               cut.off = 65) # upper limit of contacts
```

**Figure 15: Hi-C matrixplot: ChIPseq**

A BED-file of CTCF-sites is plotted at the top and a coverage-track of SMC1 ChIP-seq is plotted beneath this. The symmAnn-option leads to the same tracks being plotted on the left.

### 3.3.4 Genes

We make use of the data.fame, where each row is an exon from a gene. There are several ways to get this. One of the easiest is to use biomart to get exon-coordinates. This can be done with the biomaRt-package or via the web-based service. For this example, we downloaded some data of all exons from the Ensembl biomart:

```
## Gene stable ID & Transcript stable ID & Chromosome/scaffold name &
## Transcript start (bp) & Transcript end (bp) & Exon region start (bp) &
## Exon region end (bp) & Strand
martExport = read.delim('data/mart_export.txt.gz', stringsAsFactors = F)
colnames(martExport) = c('ENSG','ENST','chrom' , # change column names
                        'txStart' , 'txEnd' ,
                        'exonStart' , 'exonEnd' , 'strand')
martExport$chrom = gsub(martExport$chrom, # add chr-prefix
                        pattern = '^',
                        replacement = 'chr')
martExport$strand = ifelse(martExport$strand == 1, '+', "-") # 1/-1 to +/-
```

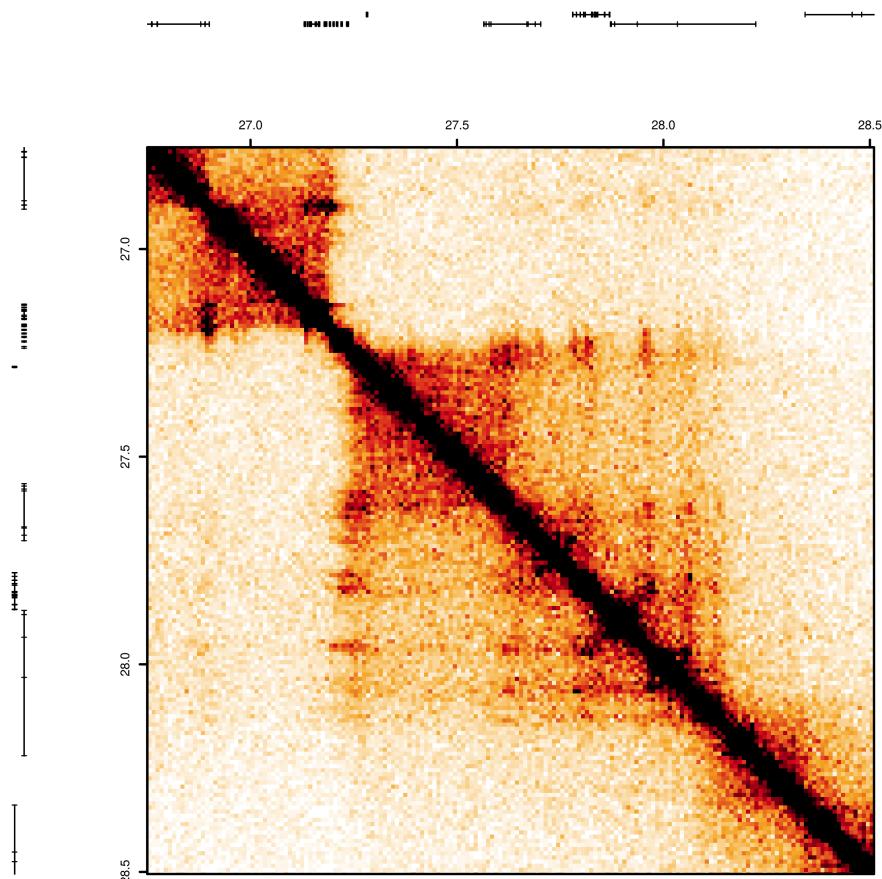
## GENOVA: explore the Hi-C's

**Table 4:** A data.frame holding the needed columns for plotting genes

| chrom | txStart  | txEnd    | exonStart | exonEnd  | strand |
|-------|----------|----------|-----------|----------|--------|
| chr1  | 44457280 | 44462200 | 44457519  | 44457676 | +      |
| chr1  | 44457280 | 44462200 | 44457280  | 44457418 | +      |
| chr1  | 44457280 | 44462200 | 44457884  | 44458059 | +      |
| chr1  | 44457280 | 44462200 | 44458195  | 44458311 | +      |
| chr1  | 44457280 | 44462200 | 44459559  | 44459636 | +      |

Now we can plot both the BED-file and the genes.

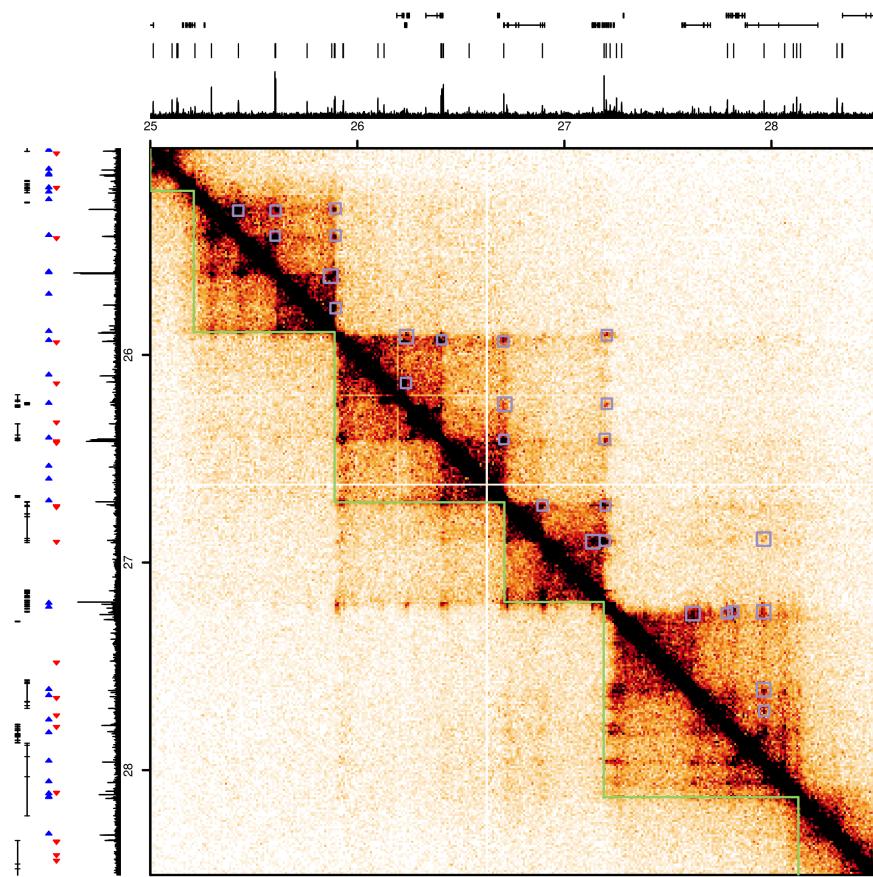
```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7', start = 26.75e6, end=28.5e6,
               genes = martExport,
               cut.off = 65) # upper limit of contacts
```



**Figure 16:** Hi-C matrixplot: genes

### 3.3.5 Everthing together

```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=28.5e6,
               loops = WT_Loops, # see APA
               loops.color = '#998ec3', # purple loops
               loops.type = 'upper', # only plot in upper triangle
               loops.resize = 20e3, # expand for visibility
               genes = martExport,
               bed.col = 'black',
               chip = list('data/SMC1_WT.bw', # inner-top
                           SMC1, # outer-top
                           'data/SMC1_WT.bw', # inner-left
                           CTCF), # outer-left
               tads = WT_TADs, # see ATA
               tad.type = 'lower', # only plot in lower triangle
               tads.color = '#91cf60', # green TAD-borders
               cut.off = 50) # upper limit of contacts
```



**Figure 17: Hi-C matrixplot: a complex case**

Loops and TADs are annotated within the Hi-C matrix. On the top annotation-bar, we have plotted the ChIP-seq signal and peaks of SMC1. On the left annotation-bar are the ChIP-seq signal and peaks (with orientation) of CTCF. Genes are plotted on both annotation-bars.

## 4 TADs

GENOVA has a repertoire of functions to analyse TADs. First, we will use the insulation score to BLABLA

### 4.1 Insulation

To estimate the strength of TAD-borders, we can look at the insulation-score (Crane et al. 2015). At a TAD-border, this score reaches a local minimum: the lower the score, the stronger the insulation. We can generate this for a specific sliding-window size with `genome.wide.insulation`. Moreover, we can generate a domainogram of a range of window-sizes for a specific genomic region with `insulation.domainogram`.

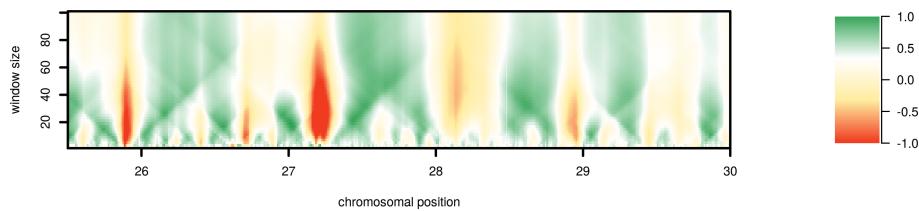
## GENOVA: explore the Hi-C's

### 4.1.1 Domainogram

To make a domainogram, we simply choose our experiment and our region of interest<sup>3</sup>.

```
layout(matrix(c(1,2,3), nrow = 1, ncol = 3), widths = c(5,1,0.1) )
insulation.domainogram(Hap1_WT_10kb,
                        'chr7',
                        25.5e6,
                        30e6,
                        window.size1 = 1,
                        window.size2 = 101,
                        step = 2)
cols = c("#f03b20", "#ffeda0", "white", "#31a354")
color.bar(colorRampPalette(cols)(100), -1, nticks = 5)
```

<sup>3</sup>The colorbar is there to get you acquainted with this type of plot.



**Figure 18: Insulation domainogram**

Insulation-hotspots can be identified in red, which are regions with a very negative score.

By running hic.matrixplot first without ChIP- and gene-annotation, we can plot the domainogram within the same figure.

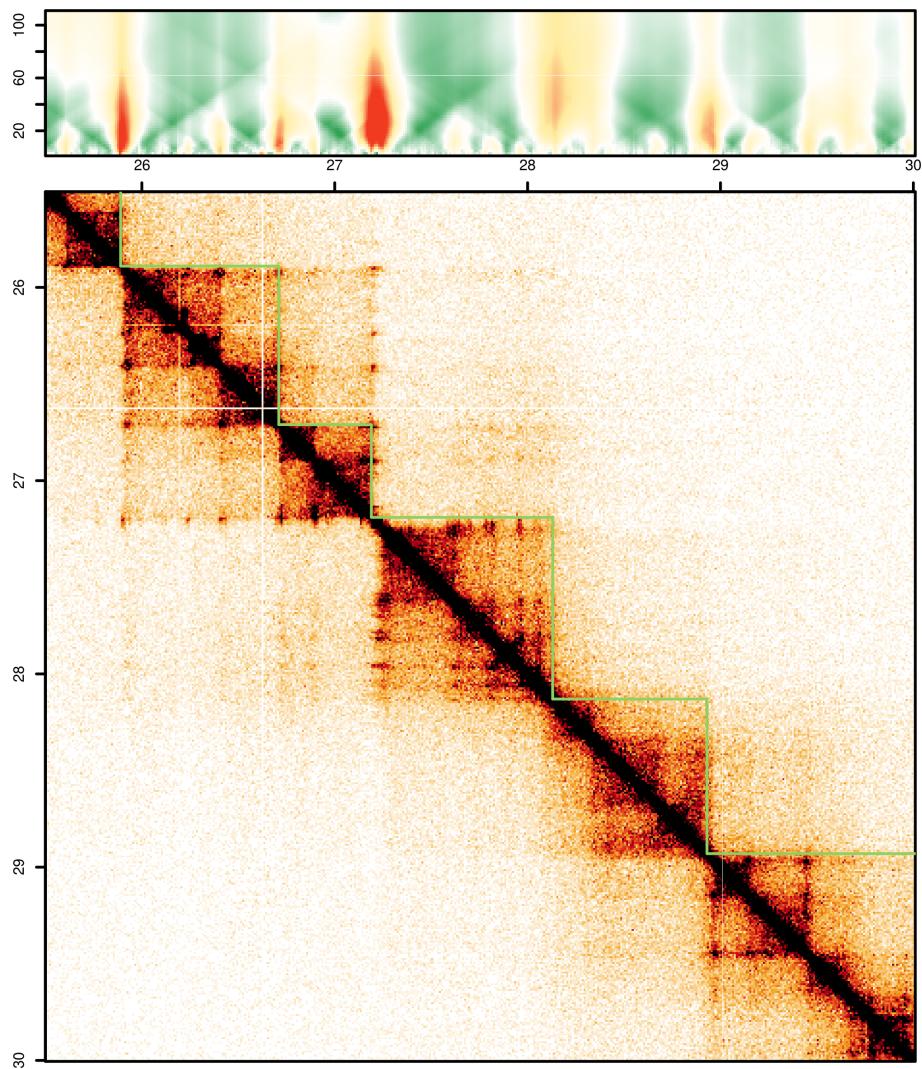
```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25.5e6,
               end=30e6,
               tads = WT_TADs, # see ATA
               tad.type = 'upper', # only plot in lower triangle
               tads.color = '#91cf60', # green TAD-borders
               cut.off = 50, # upper limit of contacts
               skipAnn = T) # skip the outside annotation

insulation.domainogram(Hap1_WT_10kb,
                        'chr7',
                        25.5e6,
                        30e6,
                        window.size1 = 1,
                        window.size2 = 111,
                        step = 2,
                        axes = F)
```

### 4.1.2 Computing the insulation score

Windows size is scary and tricky

## GENOVA: explore the Hi-C's



**Figure 19: Insulation domainogram with Hi-C matrix**

The insulation-hotspots are the sites where HiC-seg has called a TAD-border.

```
Hap1_WT_10kb_insulation = genome.wide.insulation(hic = Hap1_WT_10kb,
                                                    window.size = 40)
Hap1_WAPL_10kb_insulation = genome.wide.insulation(hic = Hap1_WAPL_10kb,
                                                    window.size = 40)
```

### 4.1.3 Insulation-heatmap

```
insulation.heatmap_out = insulation.heatmap(
    insulationList = list(WT = Hap1_WT_10kb_insulation,
                          WAPL = Hap1_WAPL_10kb_insulation ),
    bed = SMC1,
    zlim = c(-.5,0.25), # zlim.
```

## GENOVA: explore the Hi-C's

```
profileZlim = c(-0.25,0.10) # zlim for the profile  
)
```

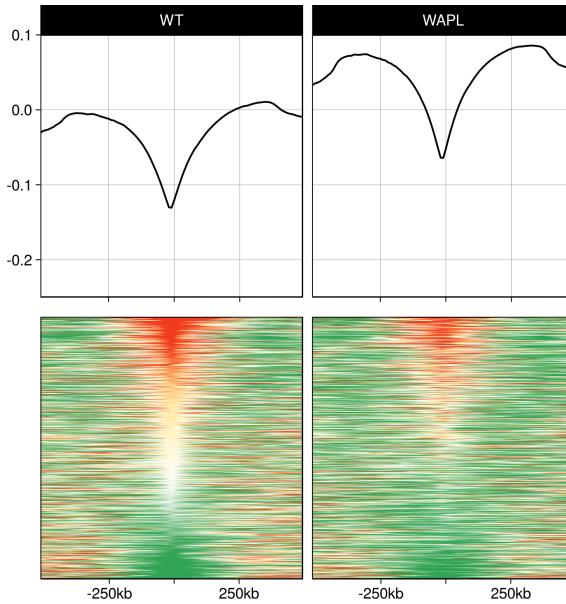


Figure 20: **Insulation heatmap**

## 4.2 Call TADs

Use the domanogram to choose a good window size:

```
Hap1_WT_10kb$INSULATION= Hap1_WT_10kb_insulation  
  
TADcalls = insulation.callTAD(Hap1_WT_10kb,BEDCOLOR = "#7ec0ee")  
  
hic.matrixplot(exp1 = Hap1_WT_10kb,  
               chrom = 'chr7',  
               start = 25e6,  
               end=29e6,  
               tads = TADcalls, # see ATA  
               tad.type = 'lower', # only plot in lower triangle  
               tads.color = '#91cf60', # green TAD-borders  
               cut.off = 25) # upper limit of contacts
```

## 4.3 ATA

```
ATA_Hap1_WT <- ATA(experiment = Hap1_WT_10kb, verbose = F,  
                     tad.bed = TADcalls)  
## Warning in ATA(experiment = Hap1_WT_10kb, verbose = F, tad.bed = TADcalls):
```

## GENOVA: explore the Hi-C's

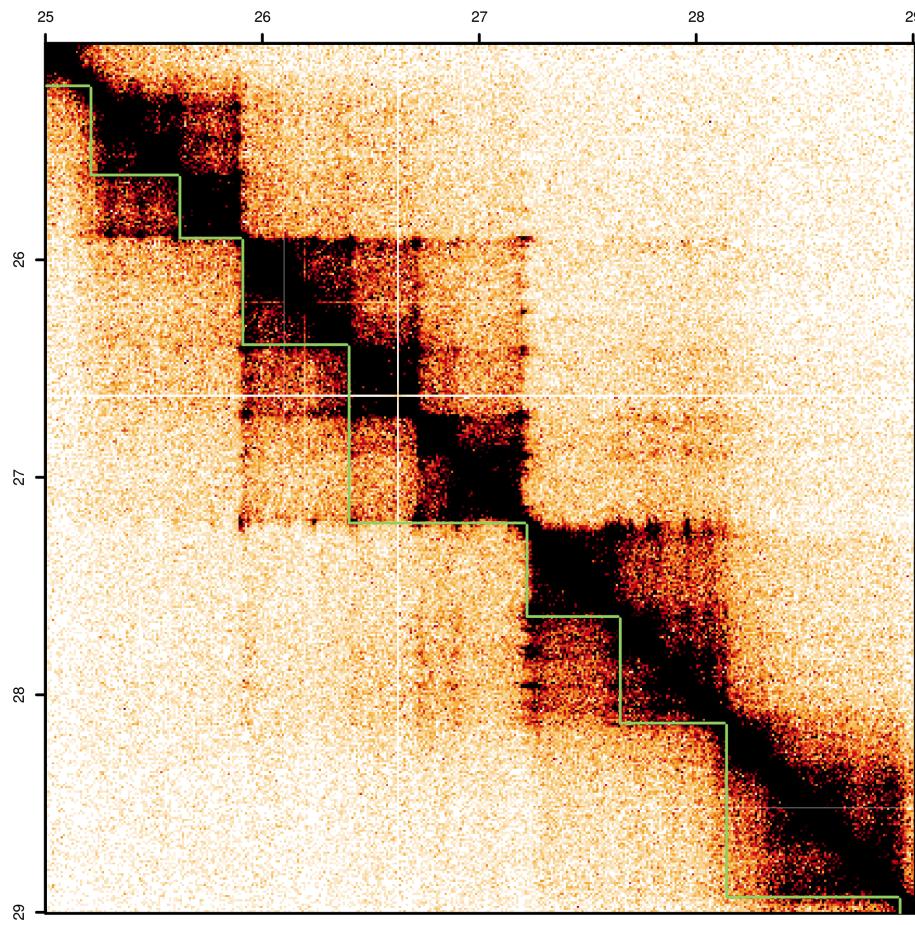


Figure 21: TADs called within GENOVA

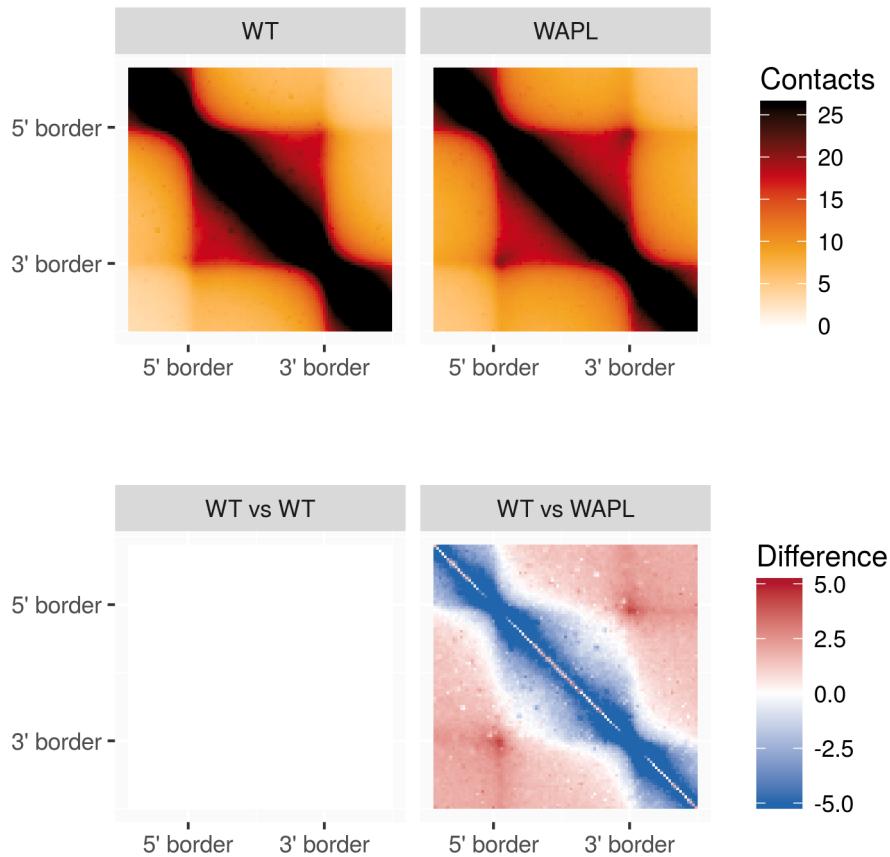
```
## The data is too sparse to do outlier-correction
## with current set of TADs.
## Output will be without outlier-correction

ATA_Hap1_WAPL <- ATA(experiment = Hap1_WAPL_10kb, verbose = F,
                      tad.bed = TADcalls)
## Warning in ATA(experiment = Hap1_WAPL_10kb, verbose = F, tad.bed = TADcalls):
## The data is too sparse to do outlier-correction
## with current set of TADs.
## Output will be without outlier-correction
```

We can use `visualise.ATA.ggplot` to combine the ATA-results.

```
visualise.ATA.ggplot(stackedlist = list('WT' = ATA_Hap1_WT,
                                         'WAPL' = ATA_Hap1_WAPL), # a named list
                      title = "Hap1 Hi-C vs WT TADs",
                      zlim1 = c(0,26),
                      zlim2 = c(-5,5),
                      focus = 1) # which entry to use as comparison
```

### Hap1 Hi-C vs WT TADs



**Figure 22: ATA**

In the WAPL-knockout, we see a decrease of contacts within the TAD, but an increase at the corner.

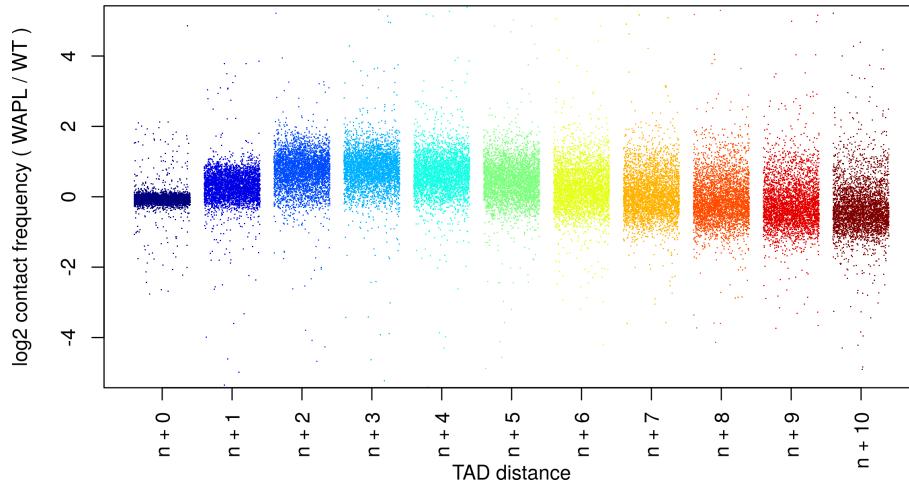
## 4.4 TAD+N

```
TAD_N_WT <- intra.inter.TAD.contacts(TAD = WT_TADs,
                                         max.neighbor = 10,
                                         exp = Hap1_WT_10kb)
TAD_N_WAPL <- intra.inter.TAD.contacts(TAD = WT_TADs,
                                         max.neighbor = 10,
                                         exp = Hap1_WAPL_10kb)
```

We can compute the enrichment of contacts between TADs with the differential.TAD.dotplot-function.

```
differential.TAD.dotplot(exp1 = TAD_N_WT, # denominator
                           exp2 = TAD_N_WAPL) # numerator
```

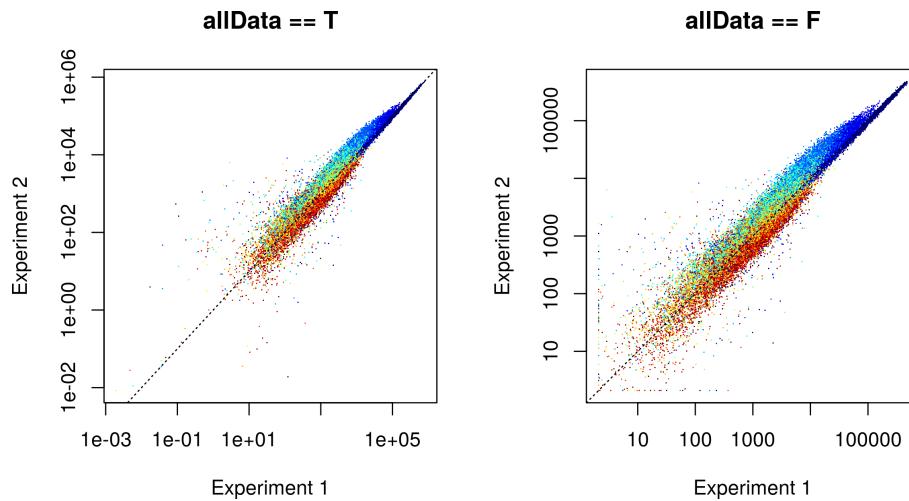
Or show it as a scatterplot. With differential.TAD.scatterplot, you can choose to add a diagonal line with line = T. Furthermore, you can choose to zoom in by allData == F.



**Figure 23: Differential TAD-analysis**

Experiment 2 (WAPL) has more interactions between neighbouring TADs compared to wild type.

```
par(mfrow = c(1,2), pty = 's')
differential.TAD.scatterplot(exp1 = TAD_N_WT, # denominator
                             exp2 = TAD_N_WAPL,
                             allData = T,
                             main = 'allData == T') # numerator
differential.TAD.scatterplot(exp1 = TAD_N_WT, # denominator
                             exp2 = TAD_N_WAPL,
                             allData = F,
                             main = 'allData == F') # numerator
```



**Figure 24: Differential TAD-analysis: scatterplot**

Experiment 2 (WAPL) has more interactions between neighbouring TADs compared to wild type.

## 5 Loops

For this section, we are using the extended loops from Haarhuis et al. (2017). These are the anchor-combinations of the merged loop-calls of WT Hap1 5-, 10- and 20-kb matrices, generated with HICCUPS (Rao et al. 2014).

```
WT_Loops_extended = read.delim('data/WT_3Mb_extended_loops.bed', h = F)
```

**Table 5: A data.frame holding a standard BEDPE format**

Columns 1-3 are describe the 5' anchor, columns 4-6 describe the 3' anchor.

| V1    | V2     | V3     | V4    | V5      | V6      |
|-------|--------|--------|-------|---------|---------|
| chr11 | 875000 | 900000 | chr11 | 2020000 | 2025000 |
| chr11 | 875000 | 900000 | chr11 | 2162500 | 2187500 |
| chr11 | 875000 | 900000 | chr11 | 2020000 | 2025000 |
| chr11 | 875000 | 900000 | chr11 | 2020000 | 2030000 |
| chr11 | 875000 | 900000 | chr11 | 1940000 | 1945000 |

### 5.1 APA

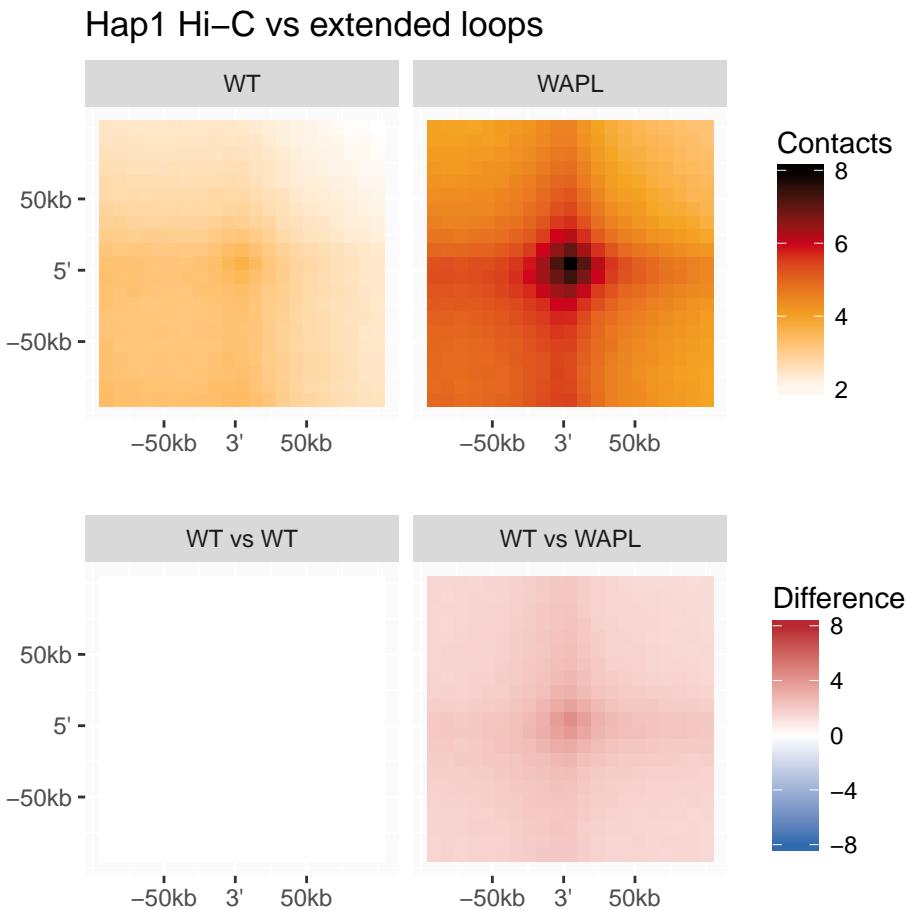
Explain smallthreshold

```
APA_Hap1_WT_extended <- APA(experiment = Hap1_WT_10kb,
                                loop.bed = WT_Loops_extended)
## Warning in APA(experiment = Hap1_WT_10kb, loop.bed = WT_Loops_extended):
## The data is too sparse to do outlier-correction
## with current set of loops.
## Output will be without outlier-correction

APA_Hap1_WAPL_extended <- APA(experiment = Hap1_WAPL_10kb,
                                 loop.bed = WT_Loops_extended)
```

We can use `visualise.APA.ggplot` to combine the APA-results.

```
visualise.APA.ggplot(APAlist = list('WT' = APA_Hap1_WT_extended,
                                      'WAPL' = APA_Hap1_WAPL_extended), # a named list
                      title = "Hap1 Hi-C vs extended loops",
                      zTop = c(1,45), # set the zlims of the upper row
                      zBottom = c(-8.33,8.33),
                      focus = 1) # which item in APAlist to use as comparison
```

**Figure 25: APA**

In the WAPL-knockout, we see an increase of contacts at the loop.

## 6 Far-cis interactions

### 6.1 PE-SCAn

Some regulatory features, like super-enhancers come together in 3D-space. To test this, we implemented PE-SCAn. Here, the enrichment of interaction-frequency of all pairwise combinations of given regions is computed. The background is generated by shifting all regions by a fixed distance (1Mb: can be changed with the `shift`-argument).

```
superEnhancers = read.delim('data/homerSuperEnhancers.txt',
                            h = F,
                            comment.char = "#")
```

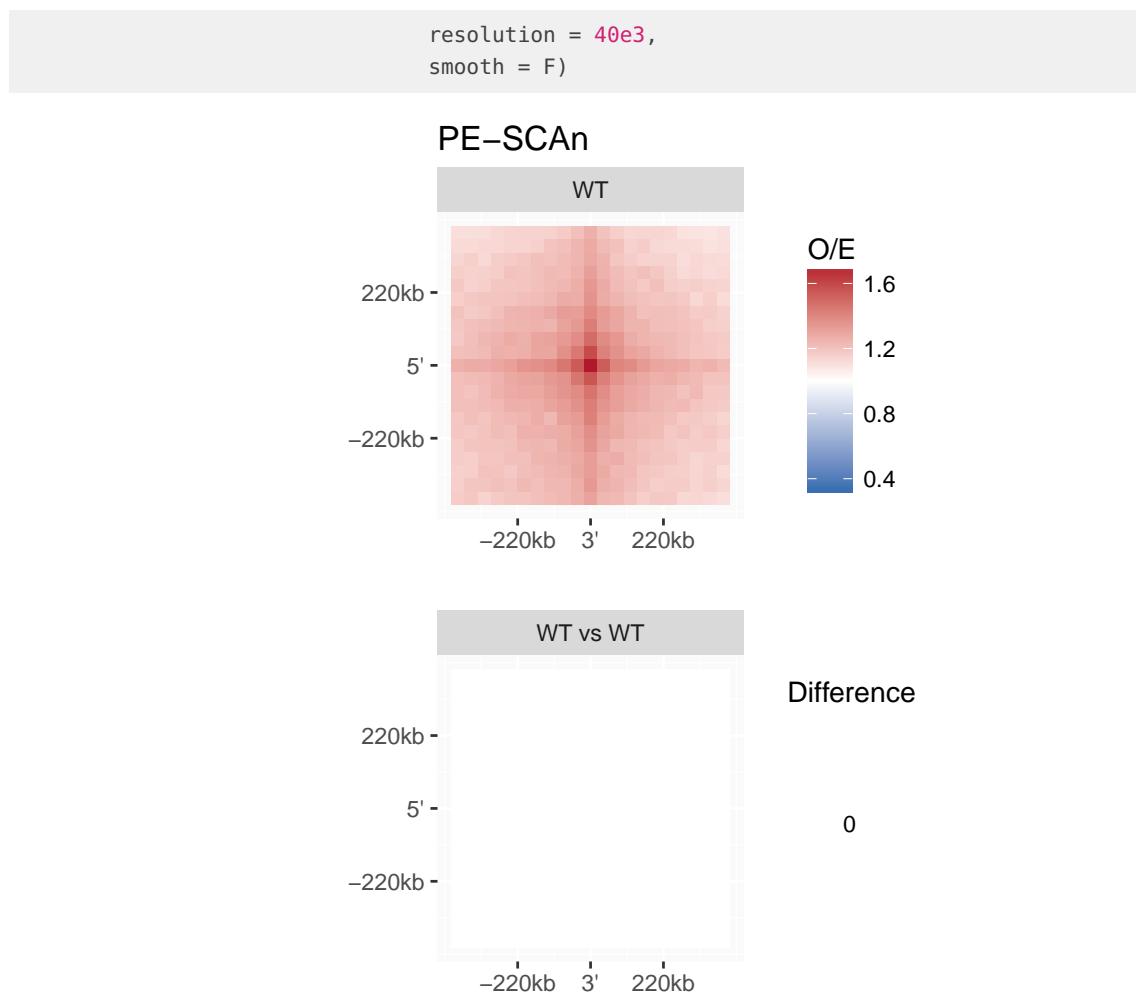
The basic visualisation is comparable to ATA and APA: the first row shows the enrichment of all included samples, while the bottom row shows the difference.

```
WT_PE_OUT = PESCAN(exp = Hap1_WT_40kb, bed = superEnhancers[,2:4])
visualise.PESCAN.ggplot(PESCANlist = list(WT = WT_PE_OUT),
```

## GENOVA: explore the Hi-C's

**Table 6:** A data.frame holding the output of homer's findPeaks -style super

| V1          | V2    | V3        | V4        | V5 | V6     |
|-------------|-------|-----------|-----------|----|--------|
| chr16-182   | chr16 | 73074453  | 73092750  | +  | 2572.8 |
| chr12-14931 | chr12 | 122219417 | 122249906 | +  | 2532.3 |
| chr2-1474   | chr2  | 133025386 | 133026123 | +  | 2523.7 |
| chr11-4061  | chr11 | 797422    | 842970    | +  | 2227.4 |
| chr15-2899  | chr15 | 89158155  | 89165379  | +  | 2087.3 |



**Figure 26: PE-SCAn**

There is a pairwise enrichment of contacts between Superenhancers, compared to shifted regions in the WT.

Another way of looking at the PE-SCAn results, is to make a perspective plot. Here, the enrichment is encoded as the z-axis.

```
RES = 40e3 # resolution of the Hi-C
persp(list(x = seq(-1*(RES*10),(RES*10), length.out = 21)/1e6, # x-ticks (MB)
           y = seq(-1*(RES*10),(RES*10), length.out = 21)/1e6, # y-ticks (MB)
           z = WT_PE_OUT), # PE-SCAn out
```

## GENOVA: explore the Hi-C's

```
phi = 25, # colatitude
theta = 40, # rotation
col="#92c5de", # color of the surface
shade=0.4, # how much shading
xlab="",
ylab="",
zlab="",
cex.axis = .6,
ticktype="detailed",
border=NA,
zlim = c(min(c(WT_PE_OUT)),
          max(c(WT_PE_OUT))))
```

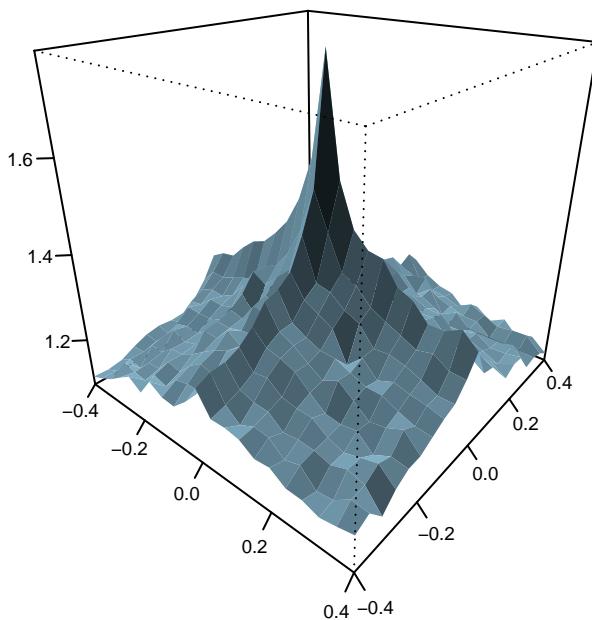


Figure 27: PE-SCAn perspective plot

## 6.2 centromere.telomere.analysis

```
centromere.telomere.analysis  
draw.centromere.telomere We saw a enriched signal between chromosomes 15 and 19. We  
can wh  
out1519 = centromere.telomere.analysis(Hap1_WT_40kb, chrom.vec = c('chr15', 'chr19'))  
draw.centromere.telomere(out1519)
```

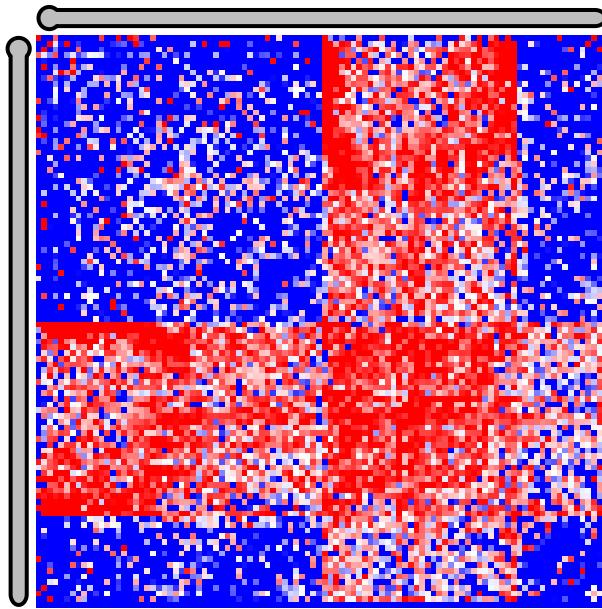


Figure 28: Centromere-telomere plot of chromosomes 15 and 19

## 7 To-do

For the next version, the following will be added/fixed:

- write `visualise.PESCA.n.persp`

Please post questions, comments and rants on [our github issue tracker](#).

## 8 Session info

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblas-p-r0.2.18.so
##
## locale:
## [1] LC_CTYPE=en_US.UTF-8      LC_NUMERIC=C
## [3] LC_TIME=en_US.UTF-8       LC_COLLATE=en_US.UTF-8
## [5] LC_MONETARY=en_US.UTF-8   LC_MESSAGES=en_US.UTF-8
## [7] LC_PAPER=en_US.UTF-8      LC_NAME=C
## [9] LC_ADDRESS=C              LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
```

## GENOVA: explore the Hi-C's

```
## [1] stats      graphics   grDevices utils      datasets   methods    base
##
## other attached packages:
## [1] viridis_0.4.0     viridisLite_0.2.0  reshape2_1.4.3    ggplot2_2.2.1
## [5] bigwrig_0.1.0     bindrcpp_0.2      GENOVA_0.9.9    BiocStyle_2.6.1
##
## loaded via a namespace (and not attached):
## [1] Rcpp_0.12.14      compiler_3.4.3    pillar_1.0.1
## [4] plyr_1.8.4        bindr_0.1         tools_3.4.3
## [7] digest_0.6.12     evaluate_0.10.1   tibble_1.4.1
## [10] gtable_0.2.0      pkgconfig_2.0.1   rlang_0.1.6
## [13] yaml_2.1.16       gridExtra_2.3    dplyr_0.7.4
## [16] stringr_1.2.0     knitr_1.18       rprojroot_1.3-1
## [19] grid_3.4.3        glue_1.2.0       data.table_1.10.4-3
## [22] R6_2.2.2          rmarkdown_1.8.5   bookdown_0.5
## [25] magrittr_1.5       codetools_0.2-15  backports_1.1.2
## [28] scales_0.4.1      htmltools_0.3.6   assertthat_0.2.0
## [31] colorspace_1.3-2   labeling_0.3     stringi_1.1.5
## [34] lazyeval_0.2.0     munsell_0.4.3
```

## References

- Crane, Emily, Qian Bian, Rachel Patton McCord, Bryan R. Lajoie, Bayly S. Wheeler, Edward J. Ralston, Satoru Uzawa, Job Dekker, and Barbara J. Meyer. 2015. "Condensin-driven remodelling of X chromosome topology during dosage compensation." *Nature* 523 (7559): 240–44. doi:[10.1038/nature14450](https://doi.org/10.1038/nature14450).
- Haarhuis, Judith H.I., Robin H. van der Weide, Vincent A Blomen, J Omar Yáñez-Cuna, Mario Amendola, Marjon S. van Ruiten, Peter H.L. Krijger, et al. 2017. "The Cohesin Release Factor WAPL Restricts Chromatin Loop Extension." *Cell* 169 (4): 693–707.e14. doi:[10.1016/j.cell.2017.04.013](https://doi.org/10.1016/j.cell.2017.04.013).
- Harewood, Louise, Kamal Kishore, Matthew D. Eldridge, Steven Wingett, Danita Pearson, Stefan Schoenfelder, V. Peter Collins, and Peter Fraser. 2017. "Hi-C as a tool for precise detection and characterisation of chromosomal rearrangements and copy number variation in human tumours." *Genome Biology* 18 (1): 125. doi:[10.1186/s13059-017-1253-8](https://doi.org/10.1186/s13059-017-1253-8).
- Lévy-Leduc, Celine, M. Delattre, T. Mary-Huard, and S. Robin. 2014. "Two-dimensional segmentation for analyzing Hi-C data." In *Bioinformatics*. Vol. 30. 17. doi:[10.1093/bioinformatics/btu443](https://doi.org/10.1093/bioinformatics/btu443).
- Lieberman-Aiden, E, and NI Van Berkum. 2009. "Comprehensive mapping of long range interactions reveals folding principles of the human genome." *Science* 326 (5950): 289–93. doi:[10.1126/science.1181369.Comprehensive](https://doi.org/10.1126/science.1181369).
- Olivares-Chauvet, Pedro, Zohar Mukamel, Aviezer Lifshitz, Omer Schwartzman, Noa Oded Elkayam, Yaniv Lubling, Gintaras Deikus, Robert P. Sebra, and Amos Tanay. 2016. "Capturing pairwise and multi-way chromosomal conformations using chromosomal walks." *Nature* 540 (7632): 296–300. doi:[10.1038/nature20158](https://doi.org/10.1038/nature20158).
- Rao, Suhas S P, Miriam H Huntley, Neva C Durand, and Elena K Stamenova. 2014. "A 3D Map of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping." *Cell* 159 (7). Elsevier Inc.: 1665–80. doi:[10.1016/j.cell.2014.11.021](https://doi.org/10.1016/j.cell.2014.11.021).
- Sanborn, Adrian L, Suhas S P Rao, Su-Chen Huang, Neva C Durand, Miriam H Huntley, Andrew I Jewett, Ivan D Bochkov, et al. 2015. "Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes." *Proceedings of the National Academy of Sciences*. doi:[10.1073/pnas.1518552112](https://doi.org/10.1073/pnas.1518552112).
- Servant, Nicolas, Nelle Varoquaux, Bryan R. Lajoie, Eric Viara, Chong-Jian Chen, Jean-Philippe Vert, Edith Heard, Job Dekker, and Emmanuel Barillot. 2015. "HiC-Pro: an optimized and flexible pipeline for Hi-C data processing." *Genome Biology* 16 (1): 259. doi:[10.1186/s13059-015-0831-x](https://doi.org/10.1186/s13059-015-0831-x).