GENOVA: explore the Hi-C's

Robin H. van der Weide [1] ***and Elzo de Wit*** [*1]

[1]Division of Gene Regulation, the Netherlands Cancer Institute

*e.d.wit@nki.nl

23 May 2018

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 **GEN**ome **O**rganisation **V**isual **A**nalytics (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

```
# devtools::install_github("robinweide/GENOVA", ref = 'dev')
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	$\frac{width\ in\ bp\ of\ window}{500}$
centromere.telomere.analysis	40kb
.compartment.plot	100kb

Table 2: Memory footprints of objects loaded into R

Experiment	Contacts	10kb	40kb	100kb	1Mb
Hap1 (Haarhuis et al. 2017)	433.5M	2.9Gb	1.7Gb	1.1Gb	0.1Gb
iPSC (Krijger et al. 2016)	427.9M	3.1Gb	1.9Gb	1.0Gb	53.1MB

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 ΔWAPL 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(ignore.checks = T, # time-saver for vignette.
                                     signalPath = 'data/WT_10000_iced.matrix',
                                     indicesPath = 'data/WT_10000_abs.bed',
                                     name = "WT",
                         color = "black")

Hap1_WAPL_10kb <- construct.experiment(ignore.checks = T,
                                     signalPath = 'data/WAPL_10000_iced.matrix',
                                     indicesPath = 'data/WAPL_10000_abs.bed',
                                     name = "WAPL",
                                     color = "red")

Hap1_SCC4_10kb <- construct.experiment(ignore.checks = T,
                                     signalPath = 'data/SCC4_10kb_iced.matrix',
                                     indicesPath = 'data/SCC4_10kb_abs.bed',
                                     name = "SCC4",
                                     color = "green")

Hap1_WT_40kb <- construct.experiment(ignore.checks = T,
                                     signalPath = 'data/WT_40000_iced.matrix',
                                     indicesPath = 'data/WT_40000_abs.bed',
                                     name = "WT",
                                     color = "black")

Hap1_WAPL_40kb <- construct.experiment(ignore.checks = T,
                                     signalPath = 'data/WAPL_40000_iced.matrix',
                                     indicesPath = 'data/WAPL_40000_abs.bed',
                                     name = "WAPL",
                                     color = "red")

Hap1_WT_1MB <- construct.experiment(ignore.checks = T,
                                     signalPath = 'data/WT_1000000_iced.matrix',
                                     indicesPath = 'data/WT_1000000_abs.bed',
                                     name = "WT", centromeres = centromeres,
                                     color = "black")

Hap1_WAPL_1MB <- construct.experiment(ignore.checks = T,
                                     signalPath = 'data/WAPL_1000000_iced.matrix',
                                     indicesPath = 'data/WAPL_1000000_abs.bed',
                                     name = "WAPL",
                                     centromeres = centromeres,
```

```
                                    color = "red")
```

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.[1] If not present, the centromeres will be emperically identified by searching for the largest stretch of no coverage on a chromosome.

[1]Please make sure that the chromosome-names match.

```
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
```

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 10
## $ 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: NULL
## $ RMCHROM : logi FALSE
```

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 resoltion:
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 (figure 1). 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 )
```

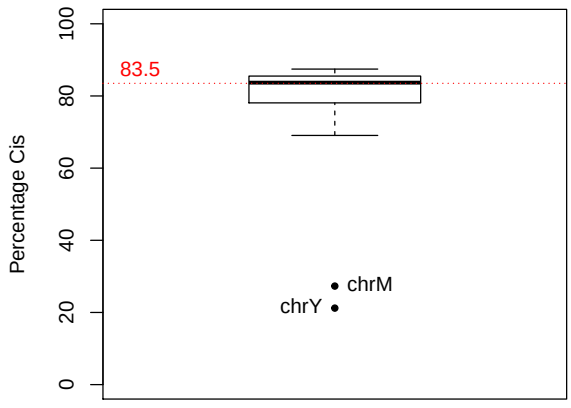

Figure 1: Fraction of cis-contacts per chromosome

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*). Invoking `plot` will provide a nice overview of the percentages *cis* (figure 2).

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

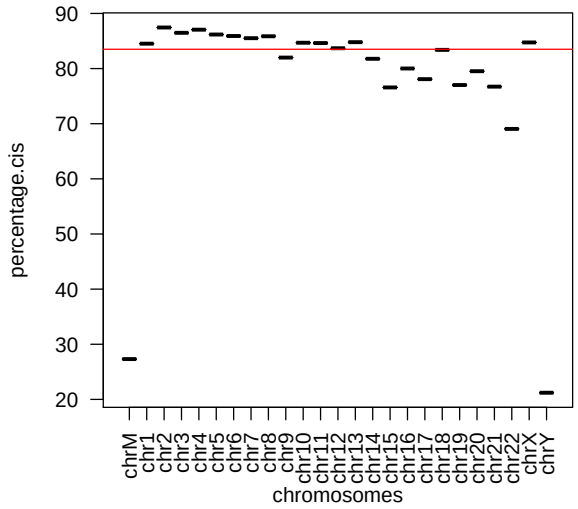

Figure 2: Fraction of cis-contacts per chromosome
Chromosomes 9, 15, 19 & 22 have translocations, which therefore appear to have more trans-contacts, but which in reality are cis-contacts.

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 $log10(observed/expected)$. The Hap1 cell line has two known translocations, which we can easily see in the resulting plot (figure 3). 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 abberant RCP), centromeric information is need. This is taken from the experiment-object or found emperically by comparing trans-interactions.

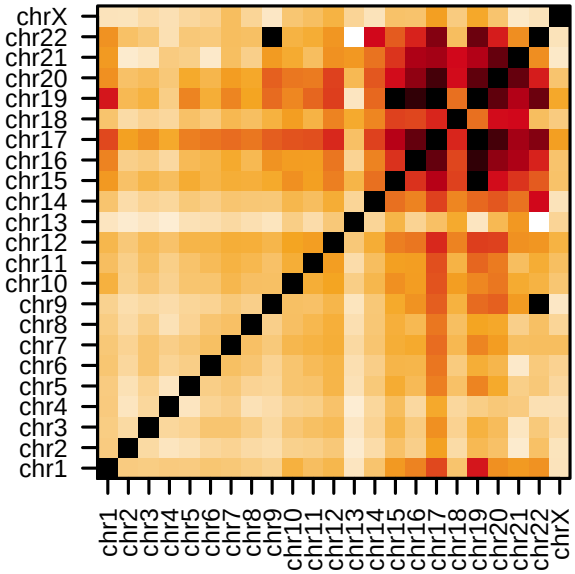

Figure 3: Chromosome matrix
The two known translocations of Hap1 cells are easily identified (15-19 & 9-22).

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

The user can decide to plot the RCP per chromosome. If the data is sparse, a LOESS-smooting 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 (figure 4). 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 comparmentalisation with `cis.compartment.plot` or `trans.compartment.plot`, for example.

```
# Plot RCP: per-chromosome
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 (figure 5). The average probabilities are then plotted, with optional errorbars for the standard error of the mean.

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

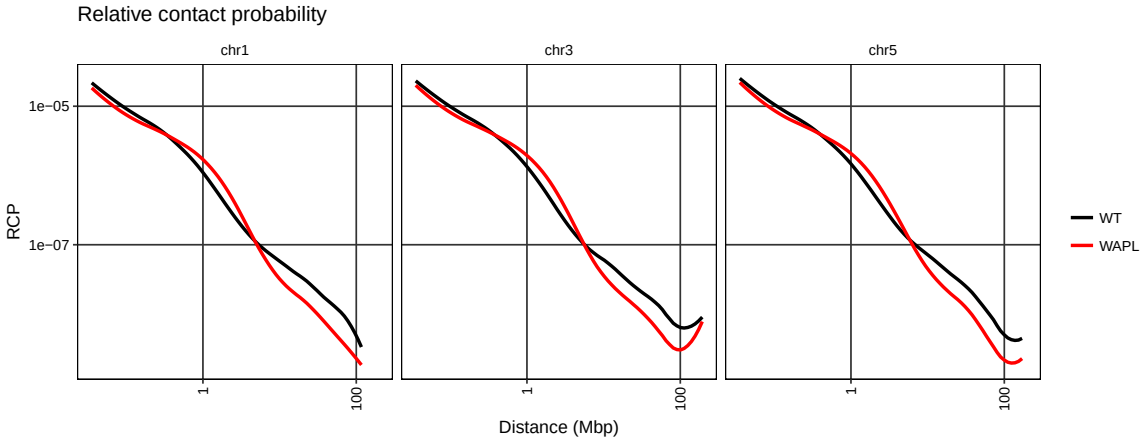

Figure 4: RCP
Every facet shows the RCP of one chromosome.

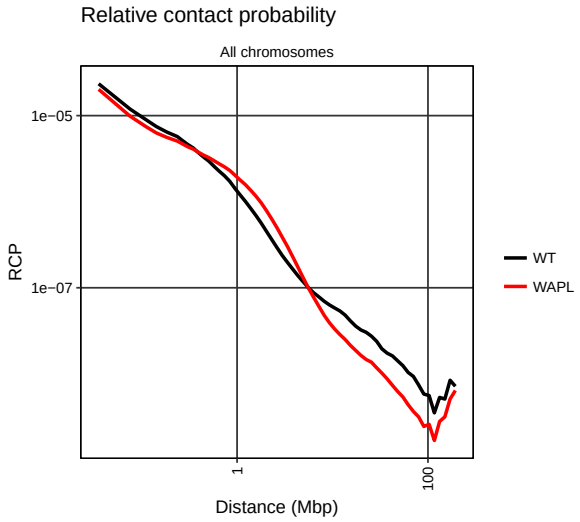

Figure 5: RCP
All data combined in one plot.

2.3.2 regions

But what if you want to compare the contact probabilities of specific regions, like Cohesin- or CTCF-bound regions? For this, we added the possibility to add a list of BED-data_frames to the `bedList`-argument. Under the hood, we perform a standard per-arm RCP (thus still enabling users to alse set the `chromsToUse`-parameter), whereafter we filter out Hi-C bins that do not have entries in the dataframe(s) of `bedList`. The same plot-function can be used: different BED-files will have different line-types. The fact that we use linetype for the `bedList` entries, allows us to still use multiple samples in `experimentList`, as shown in figure 6. But if you only provide one experiment-object, we will use different line-colours of the different BEDs.

```
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),
              chromsToUse = c('chr1','chr3', 'chr5'))

visualise.RCP.ggplot(RCPdata = RCP_out)
```

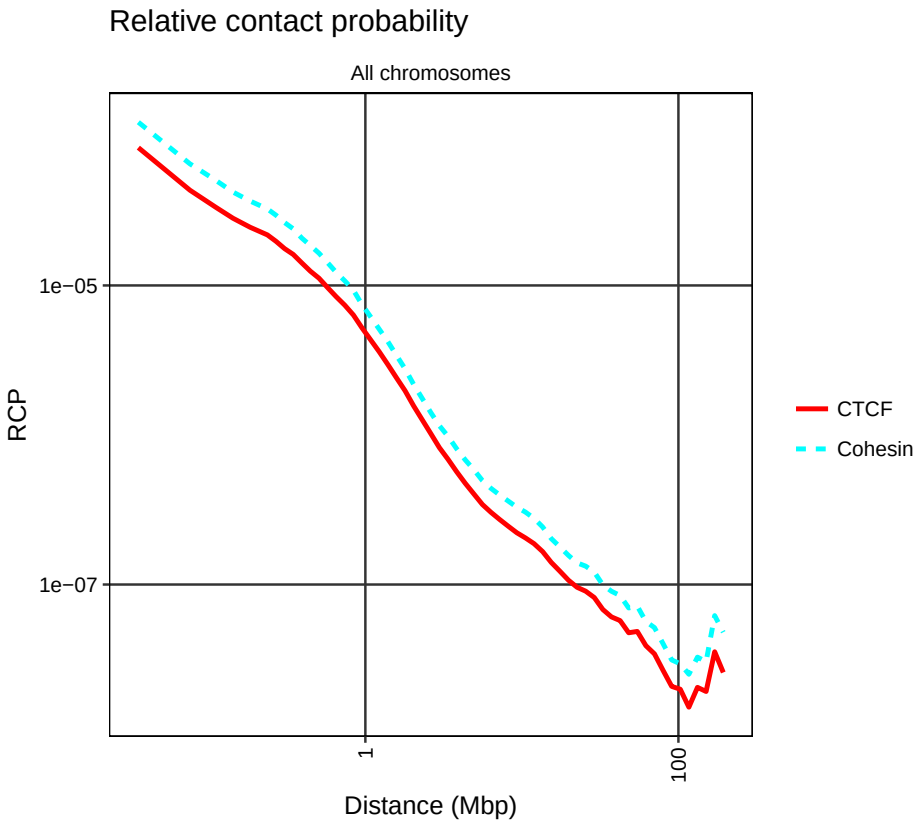

Figure 6: RCP with BEDs
We can also add BEDs as sites to compute the RCP.

2.4 A- and B-compartments

```
H3K27acPeaks = read.delim('data/H3K27ac_WT.narrowPeak', h = F)
CS_WT = compartment.score(Hap1_WT_40kb, chromsToUse = paste0('chr', 1:15))
saddle_WT = saddle(exp = Hap1_WT_40kb,
                   chip = H3K27acPeaks,CS = CS_WT,
                   chromsToUse = paste0('chr', 1:15),
                   nBins = 50)

CS_WAPL = compartment.score(Hap1_WAPL_40kb, chromsToUse = paste0('chr', 1:15))
saddle_WAPL = saddle(exp = Hap1_WAPL_40kb,
                   chip = H3K27acPeaks, CS = CS_WAPL,
```

```
                              chromsToUse = paste0('chr', 1:15),
                              nBins = 50)
```

2.4.1 Saddle-plot

```
visualise.saddle(SBoutList = list(saddle_WT, saddle_WAPL),
                 crossLines = T,
                 addText = T)
```

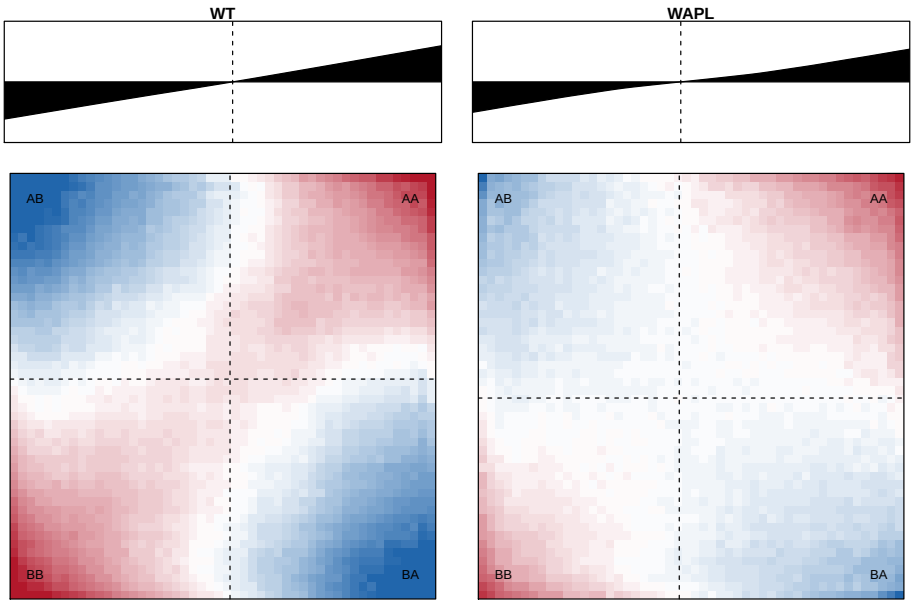

Figure 7: A saddle-plot

2.4.2 Compartment-strength

```
visualise.compartmentStrength(list(saddle_WT,
                                   saddle_WAPL))
```

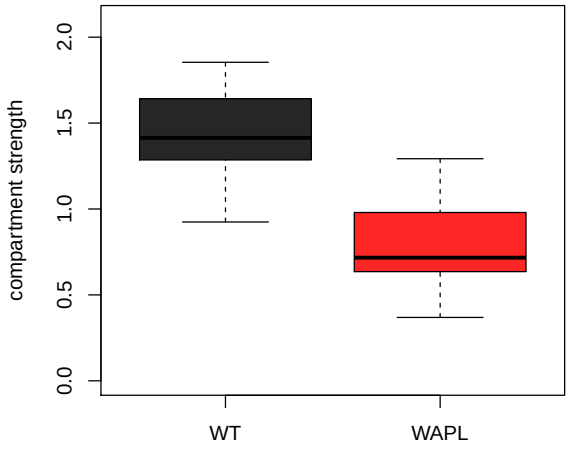

Figure 8: The per-arm compartment strength

3 Interaction plots

GENOVA has serveral plotting-functions for genomic loci. *cis.compartment.plot* and *trans.compartment.plot* provide a easy way to plot whole chromosome arms, including compartmentalisation-score tracks. For more zomomed-in plots *hic.matrixplot* can be used. This function also allows rich annotation and between-experiment comparision possibilities. All functions try to guess the most appropriate color-scale limits, but finer control of this can be gotten by setting the `cut.off`-argument.

3.1 *cis*-interactions

The compartmentalisation of the chromatin into A and B van already described in the orignial Hi-C paper (Lieberman-Aiden and Berkum 2009). Serval papers have discribed the loss of compartmentalisation when the Cohesin complex is stabalised (Haarhuis et al. 2017,@Wutz2017,@Gassler2017). To view this interesting level of chromatin organisation, we can use *cis.compartment.plot*. With this, we can plot one arm of a chromosome with the compartment-score plotted above. To infer which compartment is A (viewed as the active state) and which is B, we can add a BED-data.frame of ChIP-seq peaks from active histone marks (e.g. H3K27ac, H3K4me1). In figure 9 you can see the resulting plots, where you can see that the checkerboard-pattern in the matrix and the amplitude of the compartment-score are deminished in the WAPL-knockout.

```
H3K27ac_peaks = read.delim('data/H3K27ac_WT.narrowPeak', h = F)

cis.compartment.plot(exp = Hap1_WT_40kb,
                     chrom = 'chr14',
                     arm = 'q',
                     cs.lim = 1.75, # max compartment-score
                     cut.off = 15,
                     chip = H3K27ac_peaks)

cis.compartment.plot(exp = Hap1_WAPL_40kb,
                     chrom = 'chr14',
```

```
                    arm = 'q',
                    cs.lim = 1.75,
                    cut.off = 15,
                    chip = H3K27ac_peaks)
```

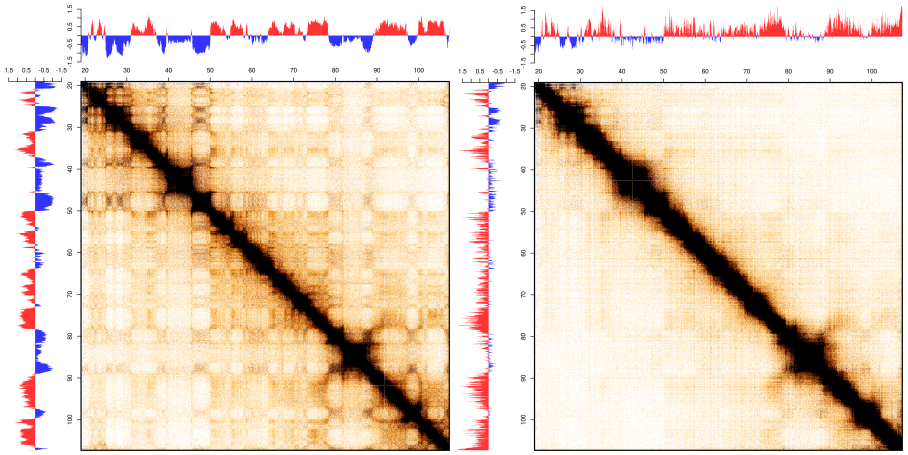

Figure 9: Cis compartment plot: WT vs WAPL
Stabalised Cohesin-mediated loops by WAPL-knockout leads to loss of compartments.

The compartment-score is calculated by performing an eigenvector decomposition on the correlation-matrix of the expected over expected matrix. To view this O/E matrix, we can set the `obs.exp`-option to TRUE. This view gives a visually better view of the A- and B-compartments (figure 10).

```
cis.compartment.plot(exp = Hap1_WT_40kb,
                     chrom = 'chr20',
                     arm = 'q',
                     cut.off = 1,
                     obs.exp = T,
                     chip = H3K27ac_peaks)
```

3.2 *trans*-interactions

As could be seen above, A-compartments interact more with other A-compartments and the same is true for B-compartments. However, is the same true for *trans*? The function `trans.compartment.plot` will allow the user to plot a trans-matrix (i.e. a matrix of the arms of two different chromosomes) along with the respective *cis* compartment-scores. This function could also be used to investigate chromosomsal translocations: the $9q;22q$ translocation can be clearly seen if we use this function, as in figure 11.

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

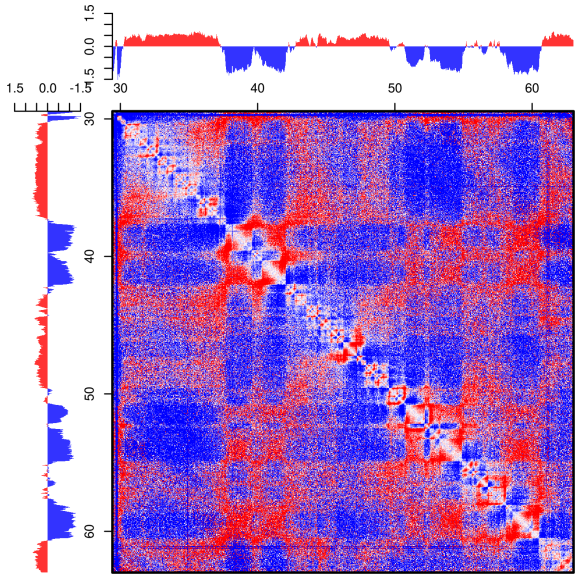

Figure 10: Cis compartment plot
Observed over expected.

```
                cut.off = 10,
                chip = H3K27ac_peaks)
```

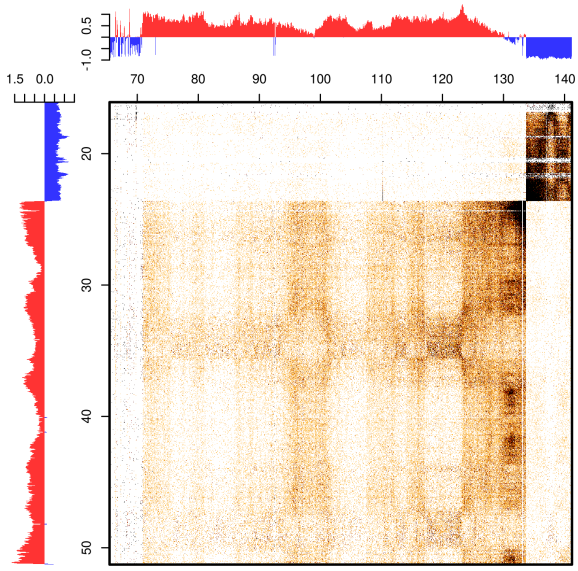

Figure 11: Trans compartment plot
The t(9q;22q) translocation is easily identified.

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. In this section, we will build up to a final, fully annotated, matrix from a humble one-experiment plot (figure 12).

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

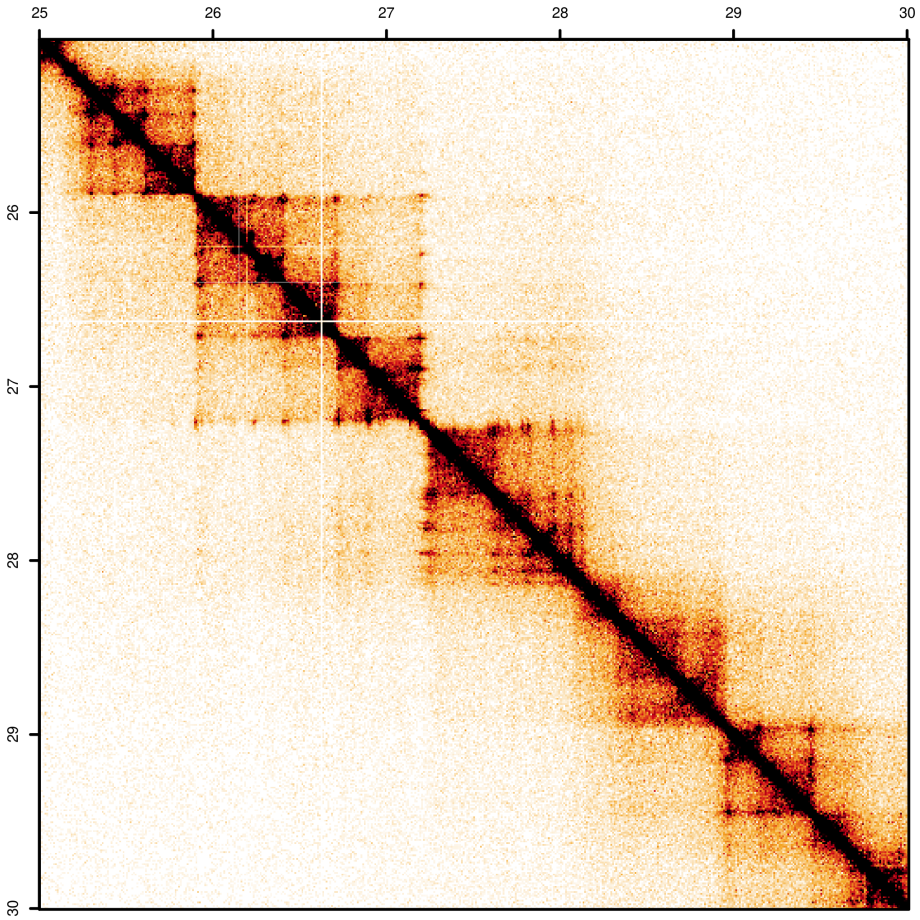

Figure 12: Hi-C matrixplot
Simplest example: one experiment, no annotation

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 upper triangle and exp2 in the lower Exp1 is subtracked from exp2 in `diff`-mode: red is therefore more contacts in exp2 and blue denotes more contacts in exp1 (figure 13).

```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               exp2 = Hap1_WAPL_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               cut.off = 50) # upper limit of contacts

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

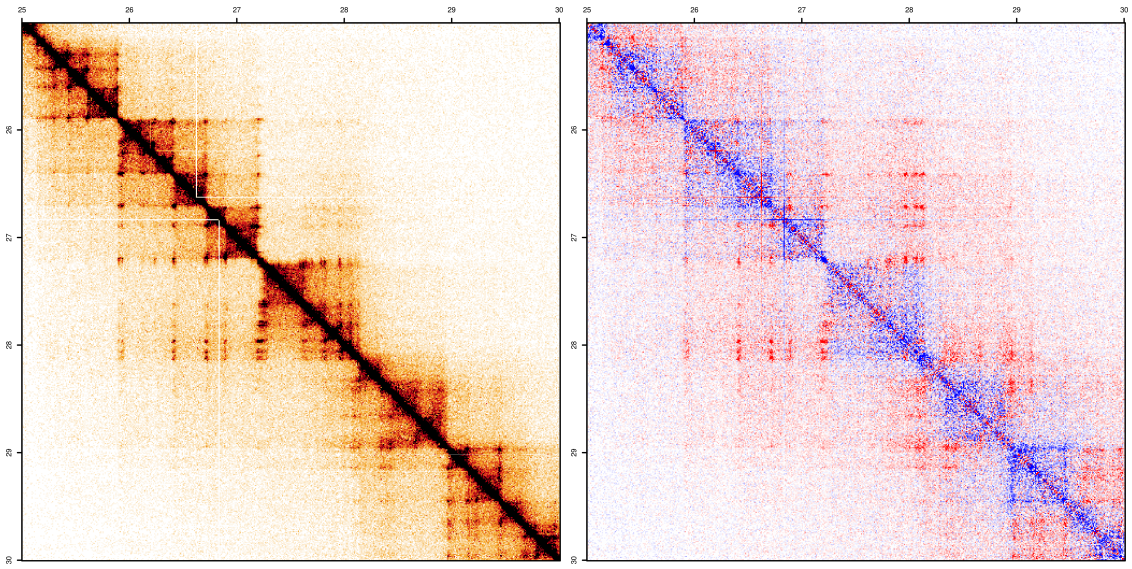

Figure 13: Hi-C matrixplot with two experiments: dual vs diff mode
The extended loops in the WAPL knockout are easily seen at around 28Mb in the lower triangle in dual-mode (left panel) and as red points in diff-mode (right panel).

3.3.2 TADs and loops

It can be very usefull to annotate the matrix with the postions of TADs and loops: take, for example, the situation where these structures are altered in a knockout for example. We are going to use the TAD- and loop-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)
sanborn2015_Loops = read.delim('data//GSE74072_Hap1_HiCCUPS_looplist.txt.gz')
```

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 (figure 14). 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. If you have mulitple loop- or tad-dataframes, you can provide a list of them: `tads = WT_TADs` for one dataframe or `tads = list(WT_TADs, KO_TADs)` for multiples, for example. All other arguments (e.g. type, color) will be recycled if only one is given.

```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               loops = WT_Loops, # see APA
               loops.color = 'blue', # purple loops
               loops.type = 'upper', # only plot in upper triangle
               loops.resize = 20e3, # expand for visibility
               tads = WT_TADs, # see ATA
               tads.type = 'lower', # only plot in lower triangle
               tads.color = '#ffd92f', # green TAD-borders
               cut.off = 25) # upper limit of contacts

hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               loops = list(WT_Loops, sanborn2015_Loops), # see APA
               loops.color = c('blue','green'), # purple loops
               loops.type = c('upper','lower'), # only plot in upper triangle
               loops.resize = c(20e3,20e3), # expand for visibility
               tads = WT_TADs, # see ATA
               tads.type = 'lower', # only plot in lower triangle
               tads.color = '#ffd92f', # green TAD-borders
               cut.off = 25) # upper limit of contacts
```

3.3.3 BigWigs and BEDs

Manipulation of CTCF-binding sites can result in loss or gain of loops and/or TADs (Wit et al. 2015). If one is interested in the effects of a knockout on the binding of a protein in combination with changes in interaction frequencies, adding ChIP-seq signal or -peaks to the matrix can be very helpfull. 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[2]) of CTCF-motifs under CTCF-ChIP

[2] https://genome.ucsc.edu/FAQ/FAQformat.html

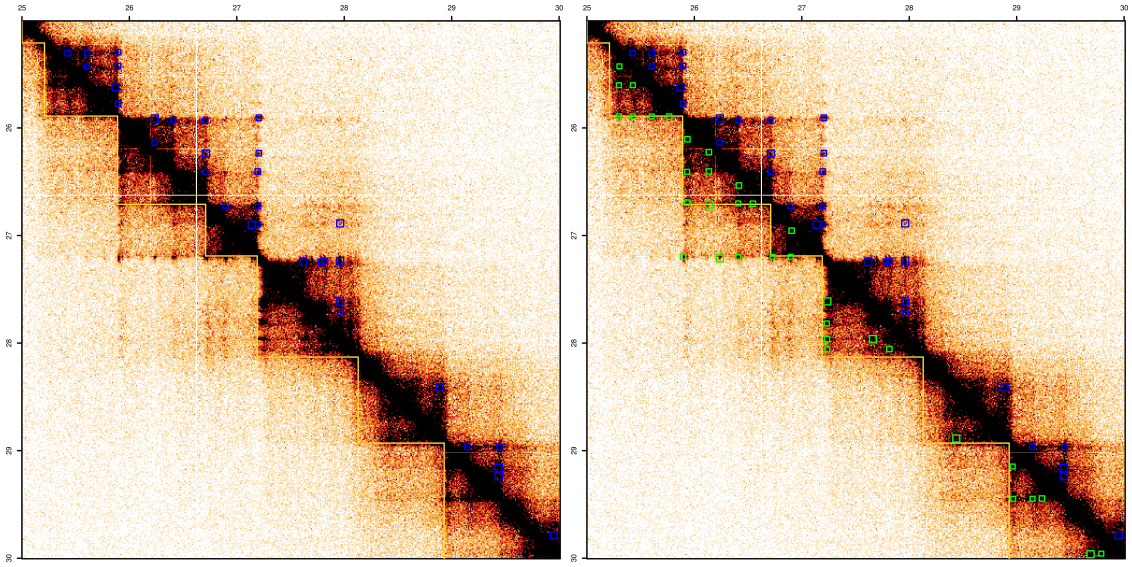

Figure 14: Hi-C matrixplot: TAD- and loop-annotations
Left: one loops-dataframe, right: a list of two dataframes from Haarhuis et al. (2017) and Sanborn et al. (2015).

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 (figure 15). If you only have a three column BED, then the autput will allways be *rectangle*.

```
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	GCAGCACCAGGTGGCAGCA	1412	+
chr1	714180	714199	CGGCCACCAGTAGGCAGCG	1428	-
chr1	793458	793477	CCACCAGCAGGTGGCCTCC	1160	-

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 `bigwrig` package, which is easily installed from github using `devtools::install_github()`. The `yMax` argument is handy if you want to compare bigwig-tracks: it lets you set the y-axis maximum.

```
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
               type = 'triangle',
               chip = list('data/SMC1_WT.bw', # inner top
                           CTCF),# outer-top
               symmAnn = F, # place annotations also on left side
               cut.off = 65) # upper limit of contacts
```

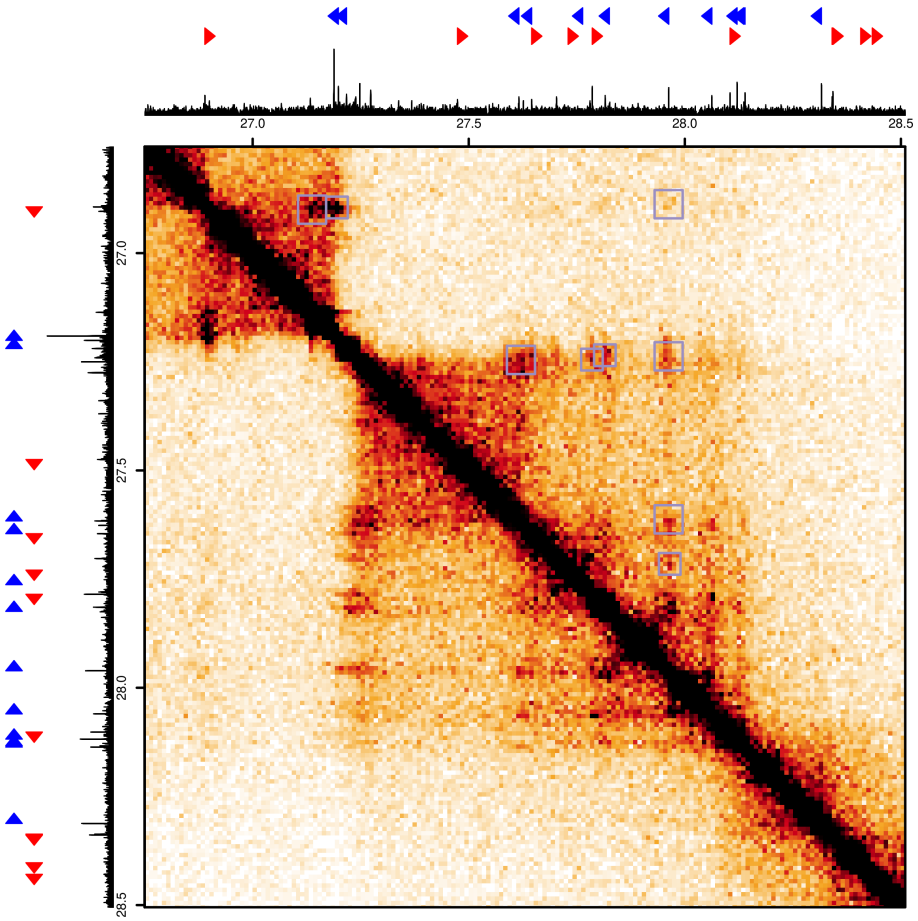

Figure 15: Hi-C matrixplot: ChIPseq
A BED-file of CTCF-sites is plotted at the top and a coverage-track of SMC1 ChIP-seq is plotted beneath this. The symmAnn-option leads to the same tracks being plotted on the left.

3.3.4 Genes

(Dixon et al. 2012) showed that housekeeping-genes are enriched in the vincinity of TAD-borders. Another interesting question could be wheter differentially expressed genes are also found near TAD-borders or binding sites of specific proteins when studying a knockout. These type of questions can be tackled by adding the appropiate gene-models to `hic.matrixplot`. To do this, 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 data of all exons from the Ensembl biomart and plotted both the BED-file and the genes (figure 16).

```
# features downloaded:
## 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 +/-
```

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	+

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

3.3.5 Everthing together

Finally, we can combine all these options in one. This may be complete overkill, but it could be quite handy. In this example, we can see that most TAD-borders and loop-anchors have clear SMC1- and CTCF-signal (figure 17). Both these are expected to be found at these locations according to the *chromatin extrusion model*. Moreover, we can also see that the CTCF-orientation of the upstream and downstream loop-anchor are forward and reverse, resp. This *convergent rule* is a known feature of loops (de Wit et al. 2015).

```
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
               tads.type = 'lower', # only plot in lower triangle
               tads.color = '#91cf60', # green TAD-borders
               cut.off = 50) # upper limit of contacts
```

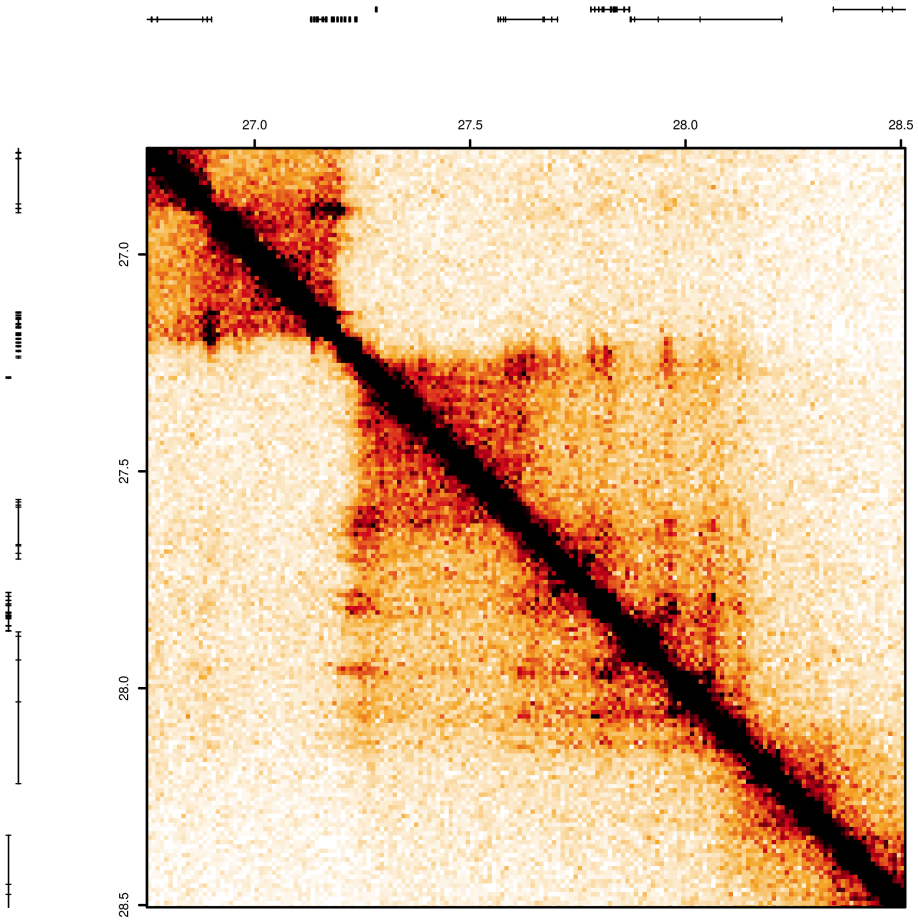

Figure 16: Hi-C matrixplot: genes

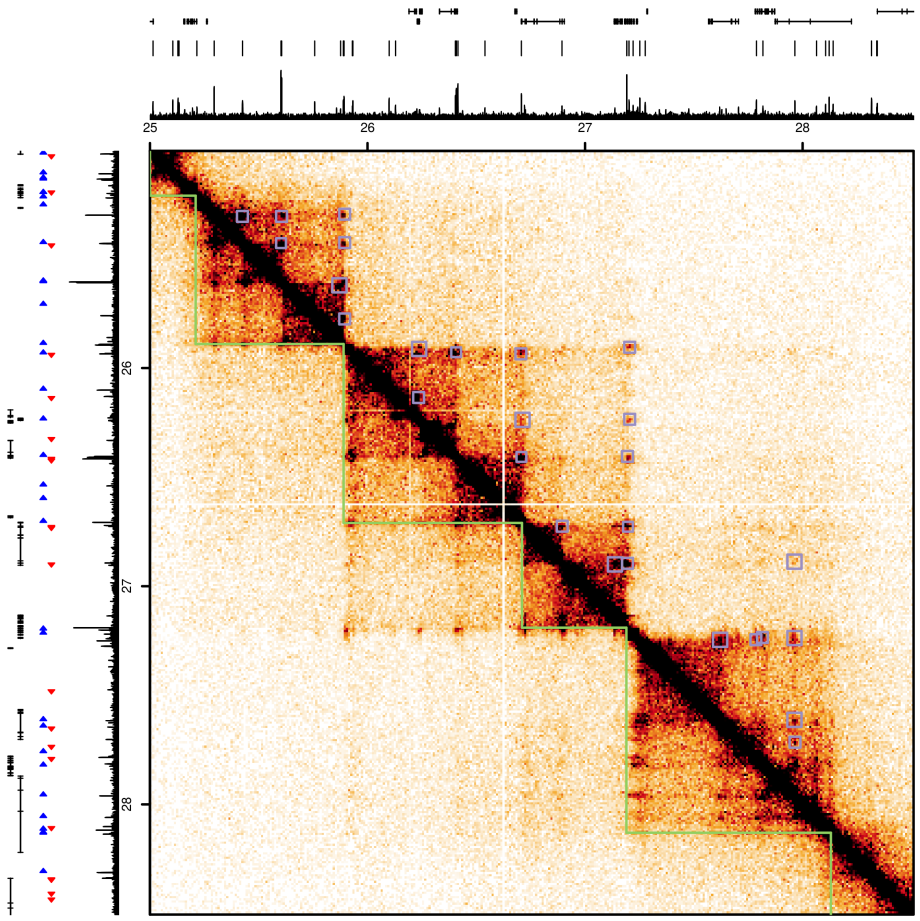

Figure 17: **Hi-C matrixplot: a complex case**
Loops and TADs are annotated within the Hi-C matrix. On the top annotation-bar, we have plotted the ChIP-seq signal and peaks of SMC1. On the left annotation-bar are the ChIP-seq signal and peaks (with orientiation) of CTCF. Genes are plotted on both annotation-bars.

4 TADs

Topologically Associated Domains (TADs) are $\pm .8 - 2Mb$ regions, which are seen as triangles in the matrix: regions that have more interactions within than outside. GENOVA has a repetoire of functions to generate and analyse TADs. Fist, we will use the insulation score to call TADs and compare the strength of TAD-borders between samples. Next, we will explore `ATA` to analyse aggegrates of TADs. Finally, wil investigate wheter TADs interact with their neighbouring TADs.

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`. The choice of window-size is quite tricky, since smaller will be sensitive to very local effect (i.e. mapping-errors, loops), while too big windows will lead a an underrepresentation. Luckily, we can generate a domainogram of a range of window-sizes for a specific genomic region with `insulation.domainogram`.

4.1.1 Domainogram

To make a domainogram, we simply choose our experiment and our region of interest[3]. The window-size is directly proportional to the amount of Hi-C bins.

[3] The colorbar is there to get you acquainted with this type of plot.

```
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)
```

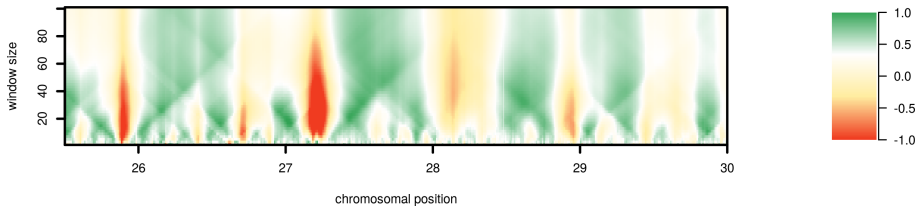

Figure 18: Insulation domainogram
Insulation-hotspots can be identified in red, which are regions with a very negative score.

A nice feature of hic.matrixplot is that if you use it without plotting anything on the sides (i.e. genes and/or ChIP-tracks), you can insert other plots. This allows us to plot the domainogram directly under the matrix, making it very easy to compare the insulation with the actual data (figure 19).

```
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=29e6,
               tads = WT_TADs, # see ATA
               tads.type = 'upper', # only plot in lower triangle
               tads.color = '#91cf60', # green TAD-borders
               cut.off = 25, # upper limit of contacts
               skipAnn = T) # skip the outside annotation
insulation.domainogram(Hap1_WT_10kb,
                       'chr7',
                       25e6,
                       29e6,
                       window.size1 = 1,
                       window.size2 = 111,
                       step = 2,
                       axes = F)
```

4.1.2 Computing the insulation score

To get the genome-wide insulation score in .bedgraph-format,[4] we provide the `genome.wide.insulation` with an experiment-object and the window-size of choice. As can be seen in the domainogram above, at $W = 25$ we will catch the mojority of the hotspots, while limiting the amount of noise.

[4] BED3 + signal column

```
Hap1_WT_10kb_insulation = genome.wide.insulation(hic = Hap1_WT_10kb,
                                                 window.size = 25)
Hap1_SCC4_10kb_insulation = genome.wide.insulation(hic = Hap1_SCC4_10kb,
                                                   window.size = 25)
```

4.1.3 Insulation-heatmap

We can align the border-strength of TADs in multiple samples to a specific BED-file, to compare *"borderness"* of feature. For example, let's use the TAD-borders from (Haarhuis et al. 2017). In figure 20 we can see that the average signal drops at the border (which is to be expected) and that this is a genome-wide feaute, as we see in the heatmap.

```
insulation.heatmap_out = insulation.heatmap(
  insulationList = list(WT = Hap1_WT_10kb_insulation,
                        SCC4 = Hap1_SCC4_10kb_insulation ),
  bed = WT_TADs,
  zlim = c(-.5,0.25), # zlim.
  profileZlim = c(-.75,-.1) # zlim for the profile
  )
```

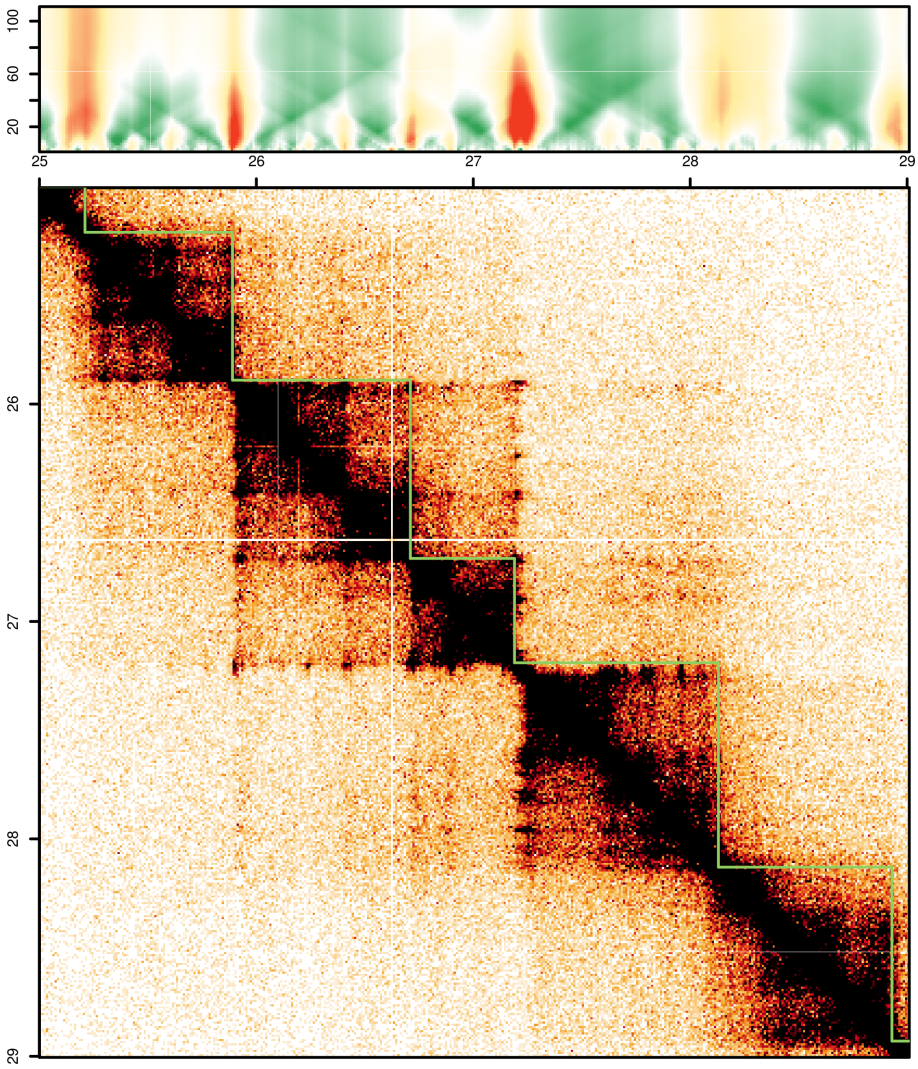

Figure 19: Insulation domainogram with Hi-C matrix
The insulation-hotspots are the sites where HiC-seg has called a TAD-border.

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
               tads.type = 'lower', # only plot in lower triangle
```

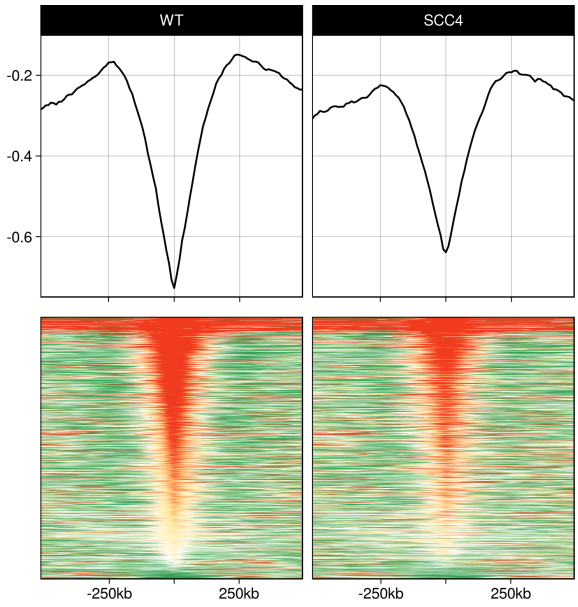

Figure 20: Insulation heatmap
The upper panel shows the average score. Each row is a TAD-border in the lower panel.

```
                    tads.color = '#91cf60', # green TAD-borders
                    cut.off = 25) # upper limit of contacts
```

4.3 ATA

```
TADcalls = WT_TADs
ATA_Hap1_WT   <- ATA(experiment = Hap1_WT_10kb, verbose = F,
                     tad.bed = TADcalls)

ATA_Hap1_WAPL <- ATA(experiment = Hap1_WAPL_10kb,verbose = F,
                     tad.bed = TADcalls)
```

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
```

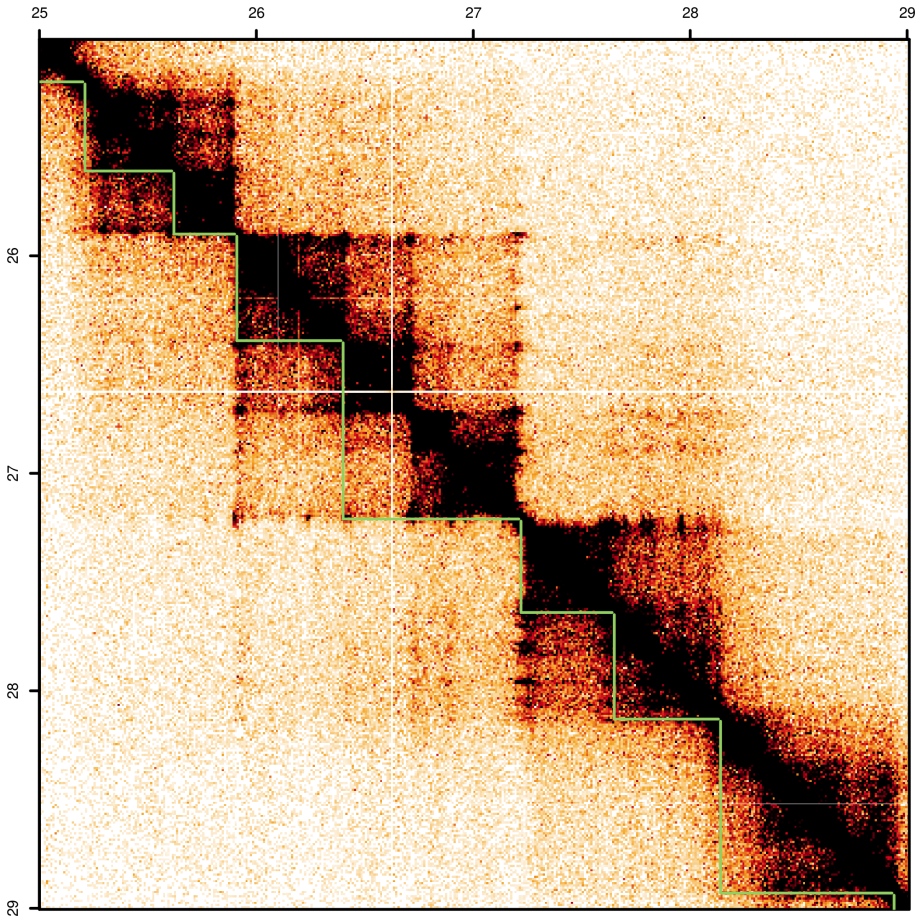

Figure 21: TADs called within GENOVA

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 shoose to add a diagonal line with `line = T`. Furthermore, you can shoose to zoom in by `allData == F`.

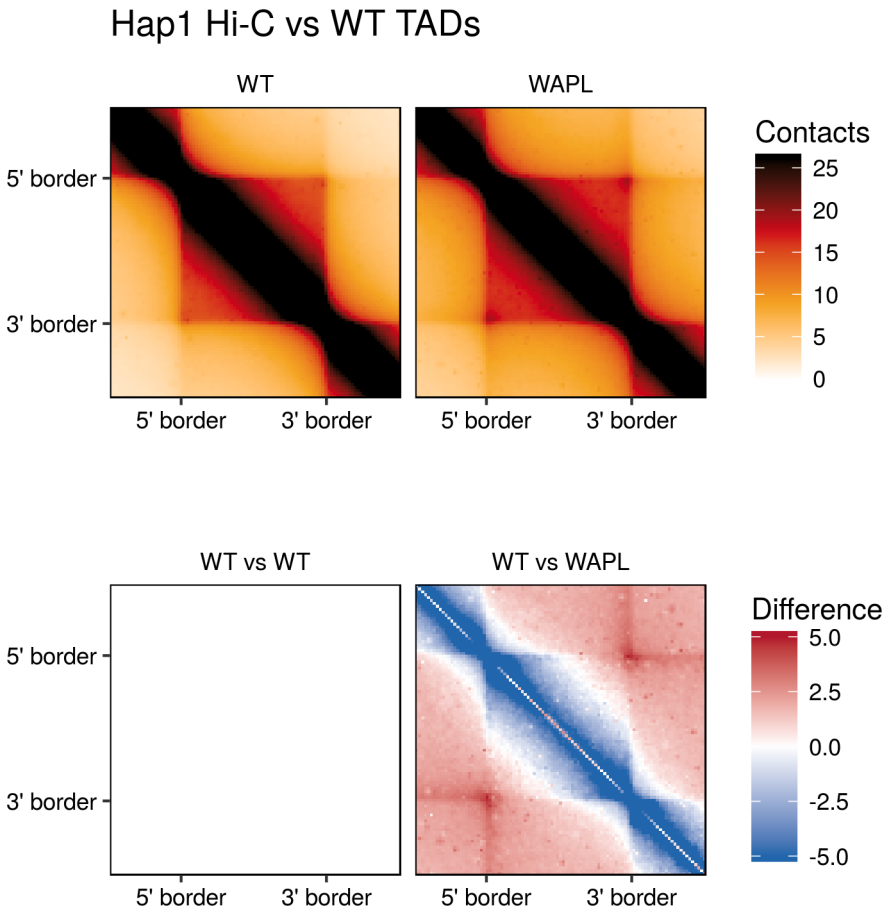

Figure 22: ATA
In the WAPL-knockout, we see a decrease of contacts within the TAD, but an increase at the corner.

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

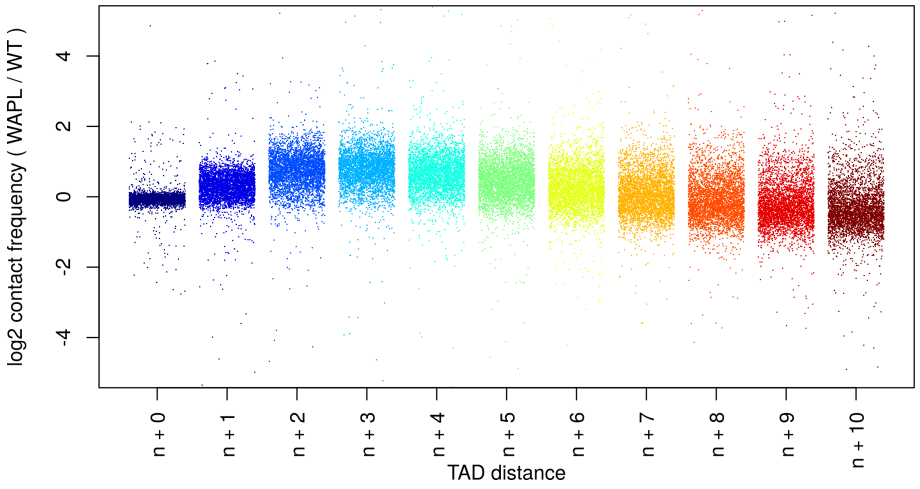

Figure 23: Differential TAD-analysis
Experiment 2 (WAPL) has more interactions between neighbouring TADs compared to wild type.

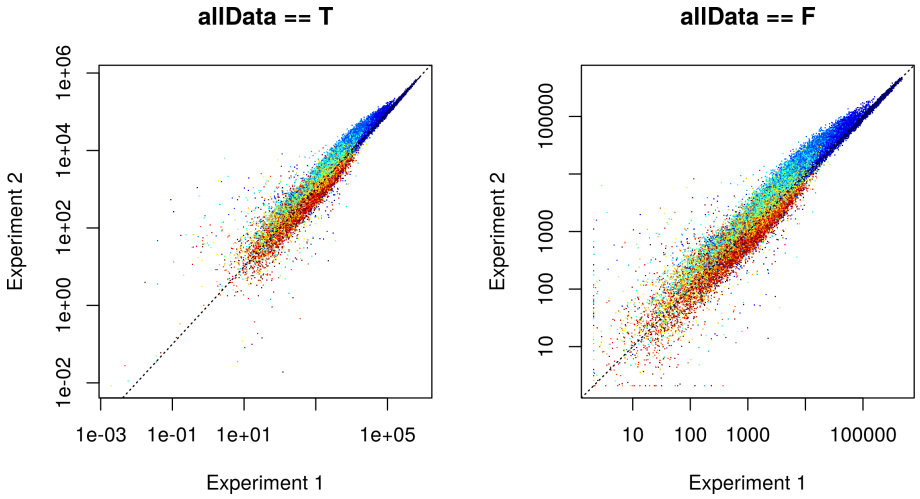

Figure 24: Differential TAD-analysis: scatterplot
Experiment 2 (WAPL) has more interactions between neighbouring TADs compared to wild type.

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)
```

5.1 APA

Explain smalltreshold

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

```
APA_Hap1_WT_extended   <- APA(experiment = Hap1_WT_10kb,
                             loop.bed = WT_Loops_extended)

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(0,9.5), # set the zlims of the upper row
                     zBottom = c(-5,5),
                     focus = 1) # which item in APAlist to use as comparison
```

To get some basic statistics on the output(s) of APA-run(s), we use `quantifyAPA()`. This function averages the region surrounding the center of each loop, where a region is defined as a square of $pixWidth \times pixWith$.

```
quantifyAPA_out <- quantifyAPA(APAlist = list('WT' = APA_Hap1_WT_extended,
                                              'WAPL' = APA_Hap1_WAPL_extended),
                               pixWidth = 3)
print(quantifyAPA_out$stats)
##   sampleA sampleB Pwilcox log2_BoverA
## 1      WT    WAPL       0   0.9995039

# pot boxplot with base-R (ggplot2 would be also easy)
boxplot(split(quantifyAPA_out$data$value, f = quantifyAPA_out$data$sample),
        col = c('darkgrey', 'red'), outline = F)
```

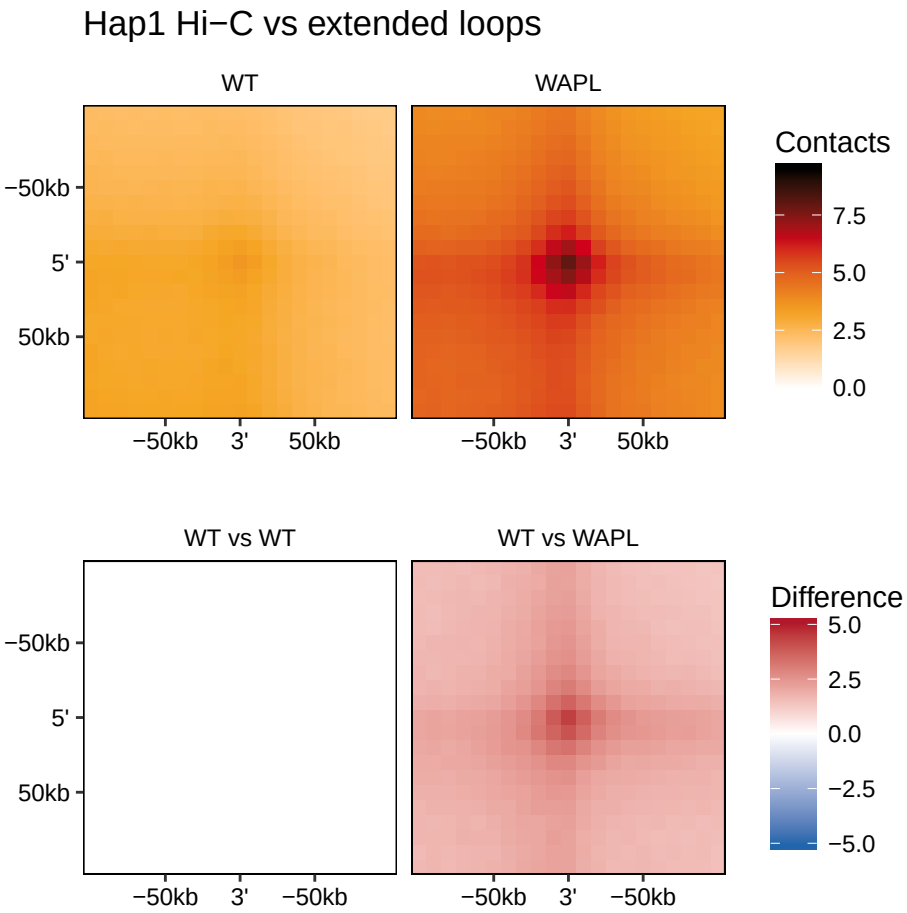

Figure 25: APA
In the WAPL-knockout, we see an increase of contacts at the loop.

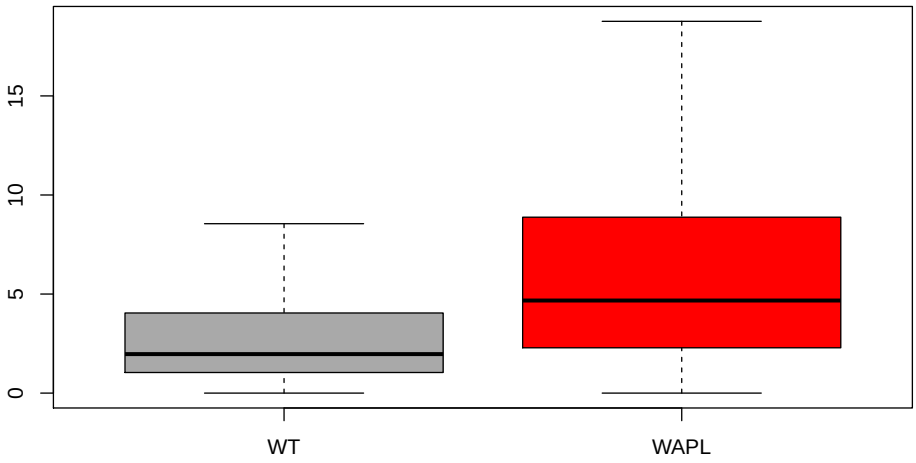

Figure 26: With quantifyAPA In the WAPL-knockout, we see an increase of contacts at the loop

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 fxed distance (1Mb: can be changed with the `shift`-argument).

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

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

The baisc 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),
                        resolution = 40e3,
                        smooth = F)
```

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
      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))))
```

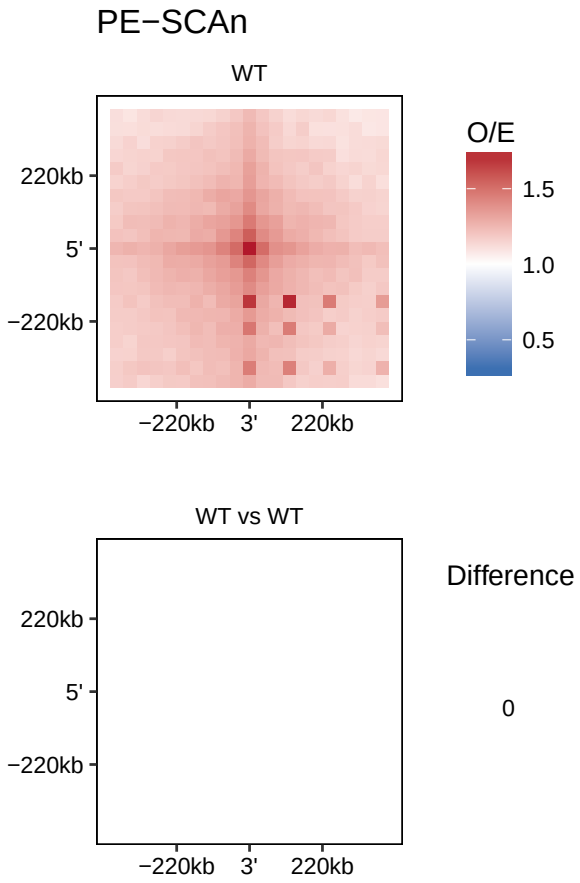

Figure 27: PE-SCAn
There is a pairwise enrichment of contacts between Superenhancers, compared to shifted regions in the WT.

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)
```

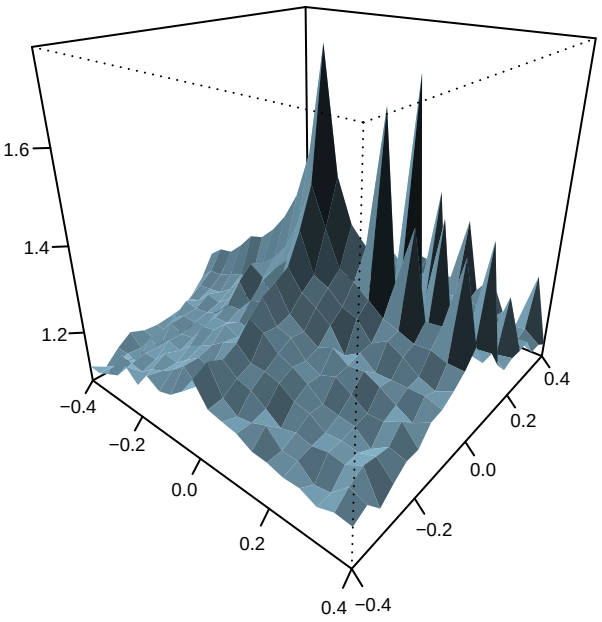

Figure 28: PE-SCAn perspective plot

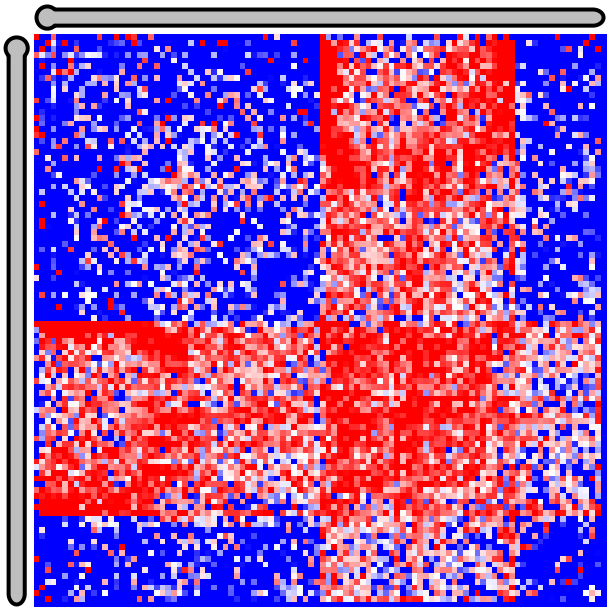

Figure 29: Centromere-telomere plot of chromosomes 15 and 19

7 To-do

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

- write `visualise.PESCAn.persp`
- write `findBadBin`
- optimise the time-v-nBin problem in `saddleBins`

Please post questions, comments and rants on our github issue tracker.

8 Session info

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.4 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-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:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] reshape2_1.4.3  bigwrig_0.1.0   ggplot2_2.2.1   bindrcpp_0.2
## [5] GENOVA_0.9.96   BiocStyle_2.6.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.16        knitr_1.20          bindr_0.1.1
##  [4] magrittr_1.5        munsell_0.4.3       colorspace_1.3-2
##  [7] R6_2.2.2            rlang_0.2.0.9001    dplyr_0.7.4
## [10] stringr_1.2.0       plyr_1.8.4          tools_3.4.4
## [13] grid_3.4.4          data.table_1.10.4-3 gtable_0.2.0
## [16] xfun_0.1            htmltools_0.3.6     assertthat_0.2.0
## [19] yaml_2.1.18         lazyeval_0.2.1      rprojroot_1.3-2
## [22] digest_0.6.15       tibble_1.4.2        bookdown_0.7
## [25] codetools_0.2-15    glue_1.2.0          evaluate_0.10.1
## [28] rmarkdown_1.9       stringi_1.1.5       pillar_1.2.2
## [31] compiler_3.4.4      scales_0.4.1        backports_1.1.2
## [34] pkgconfig_2.0.1
```

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. https://doi.org/10.1038/nature14450.

de Wit, Elzo, Erica S M Vos, Sjoerd J B Holwerda, Christian Valdes-Quezada, Marjon J A M Verstegen, Hans Teunissen, Erik Splinter, Patrick J. Wijchers, Peter H L Krijger, and Wouter de Laat. 2015. "CTCF Binding Polarity Determines Chromatin Looping." *Molecular Cell* 60 (4):676–84. https://doi.org/10.1016/j.molcel.2015.09.023.

Dixon, Jesse R., Siddarth Selvaraj, Feng Yue, Audrey Kim, Yan Li, Yin Shen, Ming Hu, Jun S. Liu, and Bing Ren. 2012. "Topological domains in mammalian genomes identified by analysis of chromatin interactions." *Nature* 485 (7398). Nature Publishing Group:376–80. https://doi.org/10.1038/nature11082.

Gassler, Johanna, Hugo B Brandão, Maxim Imakaev, Ilya M Flyamer, Sabrina Ladstätter, Wendy A Bickmore, Jan-Michael Peters, Leonid A Mirny, and Kikuë Tachibana. 2017. "A mechanism of cohesin-dependent loop extrusion organizes zygotic genome architecture." *The EMBO Journal*, e201798083. https://doi.org/10.15252/embj.201798083.

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. 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. https://doi.org/10.1186/s13059-017-1253-8.

Krijger, Peter H.L., Bruno Di Stefano, Elzo De Wit, Francesco Limone, Chris Van Oevelen, Wouter De Laat, and Thomas Graf. 2016. "Cell-of-origin-specific 3D genome structure acquired during somatic cell reprogramming." *Cell Stem Cell* 18 (5):597–610. https://doi.org/10.1016/j.stem.2016.01.007.

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. https://doi.org/10.1093/bioinformatics/btu443.

Lieberman-Aiden, E, and Nl Van Berkum. 2009. "Comprehensive mapping of long range interactions reveals folding principles of the human genome." *Science* 326 (5950):289–93. https://doi.org/10.1126/science.1181369.Comprehensive.

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. 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. 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*. 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. https://doi.org/10.1186/s13059-015-0831-x.

Wit, Elzo de, Erica S M Vos, Sjoerd J B Holwerda, Christian Valdes-Quezada, Marjon J A M Verstegen, Hans Teunissen, Erik Splinter, Patrick J. Wijchers, Peter H L Krijger, and Wouter de Laat. 2015. "CTCF Binding Polarity Determines Chromatin Looping." *Molecular Cell* 60 (4):676–84. https://doi.org/10.1016/j.molcel.2015.09.023.

Wutz, Gordana, Csilla Várnai, Kota Nagasaka, David A Cisneros, Roman R Stocsits, Wen Tang, Stefan Schoenfelder, et al. 2017. "Topologically associating domains and chromatin loops depend on cohesin and are regulated by CTCF, WAPL, and PDS5 proteins." *The EMBO Journal*, e201798004. https://doi.org/10.15252/embj.201798004.