GENOVA: explore the Hi-Cs

***Robin H. van der Weide*[1], *Teun van den Brand*[1], *and Elzo de Wit*[*1]**

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

[*]e.d.wit@nki.nl

03 February 2021

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

Loading data

```
# devtools::install_github("robinweide/GENOVA", ref = 'dev')
library(GENOVA)
```

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.

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
cis_trans	500kb-1Mb
chromosome_matrix	500kb-1Mb
RCP	40kb-500kb
intra_inter_TAD	20kb - 40kb
PE-SCAn	20kb-40kb
hic_matrixplot	$\frac{width\ in\ bp\ of\ window}{500}$
centromere.telomere.analysis	40kb
Other matrixplots	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

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/symlinks/../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
```

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 <- load_contacts(
  signal_path = '../data/symlinks/WT_10000_iced.matrix',
  indices_path = '../data/symlinks/WT_10000_abs.bed',
  sample_name = "WT",
  colour = "black"
)
#> Reading data...

Hap1_WAPL_10kb <- load_contacts(
  signal_path = '../data/symlinks/WAPL_10000_iced.matrix',
  indices_path = '../data/symlinks/WAPL_10000_abs.bed',
  sample_name = "WAPL",
  colour = "red"
)
#> Reading data...

Hap1_SCC4_10kb <- load_contacts(
  signal_path = '../data/symlinks/SCC4_10000_iced.matrix',
  indices_path = '../data/symlinks/SCC4_10000_abs.bed',
  sample_name = "SCC4",
  colour = "green"
)
#> Reading data...

Hap1_WT_40kb <- load_contacts(
  signal_path = '../data/symlinks/WT_40000_iced.matrix',
```

```
  indices_path = '../data/symlinks/WT_40000_abs.bed',
  sample_name = "WT",
  colour = "black"
)
#> Reading data...

Hap1_WAPL_40kb <- load_contacts(
  signal_path = '../data/symlinks/WAPL_40000_iced.matrix',
  indices_path = '../data/symlinks/WAPL_40000_abs.bed',
  sample_name = "WAPL",
  colour = "red"
)
#> Reading data...

Hap1_WT_1MB <- load_contacts(
  signal_path = '../data/symlinks/WT_1000000_iced.matrix',
  indices_path = '../data/symlinks/WT_1000000_abs.bed',
  sample_name = "WT", centromeres = centromeres,
  colour = "black"
)
#> Reading data...

Hap1_WAPL_1MB <- load_contacts(
  signal_path = '../data/symlinks/WAPL_1000000_iced.matrix',
  indices_path = '../data/symlinks/WAPL_1000000_abs.bed',
  sample_name = "WAPL",
  centromeres = centromeres,
  colour = "red"
)
#> Reading data...
```

The resulting `contacts`-object has several slots. `MAT` and `IDX` are the signal- and index-data.tables. We also have slots for the included chromosomes (`CHRS`) and the given centromers (`CENTROMERES`).

```
#> List of 4
#> $ MAT :Classes 'data.table' and 'data.frame': 179689855
#>    obs. of 3 variables:
#> ..$ V1: int [1:179689855] NULL ...
#> ..$ V2: int [1:179689855] NULL ...
#> ..$ V3: num [1:179689855] NULL ...
#> ..- attr(*, ".internal.selfref")=<externalptr>
#> ..- attr(*, "sorted")= chr [1:2] ...
#> $ IDX :Classes 'data.table' and 'data.frame': 309581 obs.
#>    of 4 variables:
#> ..$ V1: chr [1:309581] ...
#> ..$ V2: int [1:309581] NULL ...
#> ..$ V3: int [1:309581] NULL ...
#> ..$ V4: int [1:309581] NULL ...
#> ..- attr(*, ".internal.selfref")=<externalptr>
#> ..- attr(*, "sorted")= chr [1:2] ...
```

```
#> $ CHRS : chr [1:24] ...
#> $ CENTROMERES:Classes 'data.table' and 'data.frame': 24
#>    obs. of 3 variables:
#> ..$ chrom: chr [1:24] ...
#> ..$ start: int [1:24] NULL ...
#> ..$ end : int [1:24] NULL ...
#> ..- attr(*, ".internal.selfref")=<externalptr>
#> ..- attr(*, "sorted")= chr ...
#> - attr(*, "class")= chr ...
#> - attr(*, "znorm")= logi NULL ...
#> - attr(*, "samplename")= chr ...
#> - attr(*, "colour")= chr ...
#> - attr(*, "resolution")= num NULL ...
#> - attr(*, "rmChrom")= logi NULL ...
#> - attr(*, "balanced")= logi NULL ...
#> - attr(*, "scale_cis")= logi NULL ...
#> - attr(*, "package")= chr ...
```

Finally, the object has a lot of specific attributes, like metadata and given parameters during loading. 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 `scale_bp`-option in `load_contacts()`).

```
#>      class znorm samplename colour resolution rmChrom balanced scale_cis
#> 1 contacts FALSE         WT  black      10000    TRUE     TRUE     FALSE
#>   package
#> 1  GENOVA
```

Juicebox & cooler

Both `.hic` and `.cooler` files can be loaded from version 1 onwards. The same `load_contacts()` function can be used, which will automatically determine the file-type based on the extention.

```
Hap1_WT_10kb_juicer <- load_contacts(signal_path = '../data/symlinks/WT.hic',
                                     sample_name = "WT",
                                     resolution = 10e3,
                                     balancing = 'KR', # this is the default
                                           colour = "black")

Hap1_WT_10kb_cooler <- load_contacts(signal_path = '../data/symlinks/WT.cooler',
                                     sample_name = "WT",
                                     balancing = T,
                                           colour = "black")
```

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).

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-sample *cis*-contacts, we simply provide `cis_trans()` with the contacts-object of interest. The output can be used to make a barplot of the percentages *cis* per sample (figure 1).

```
cisChrom_out <- cis_trans( list(Hap1_WT_1MB, Hap1_WAPL_1MB) )
barplot(cisChrom_out$cis, names.arg = cisChrom_out$sample, ylim = c(0, 100) )
abline(h = cisChrom_out$cis[1], col = 'red', lty = 3)
abline(h = cisChrom_out$cis[2], col = 'red', lty = 3)
```

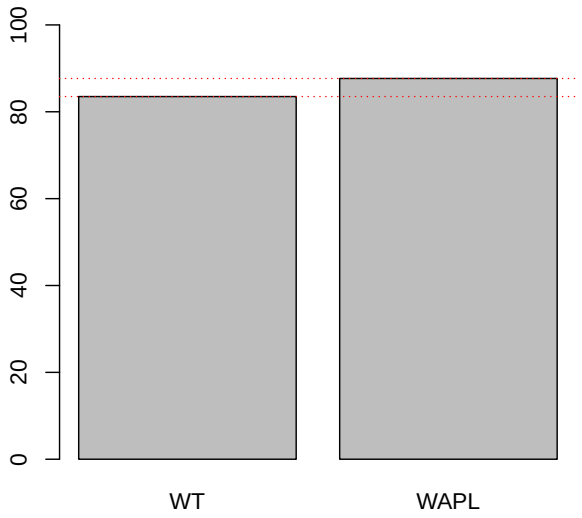


Figure 1: Fraction of cis-contacts per sample
Red dotted line denote the percentage-range from Olivares et al..

The same function can also be ran on specific regions. For this example, we will compute the intra-arm percentages. Plotting this shows us that there are interesting differences between the amounts of intra-arm contacts in *cis*, which is can be attributed to the loss of TADs (Haarhuis et al. 2017) (figure 2).

```
p_arms <- data.frame('chromosome' = centromeres[,1],
                     'start' = 0,
                     'end' = centromeres[,2])
cisChrom_out <- cis_trans( list(Hap1_WT_10kb, Hap1_SCC4_10kb) , bed = p_arms)
barplot(cisChrom_out$cis, names.arg = cisChrom_out$sample, ylim = c(0, 100))
```

```
        abline(h = 90, col = 'red', lty = 3)
        abline(h = 93, col = 'red', lty = 3)
```

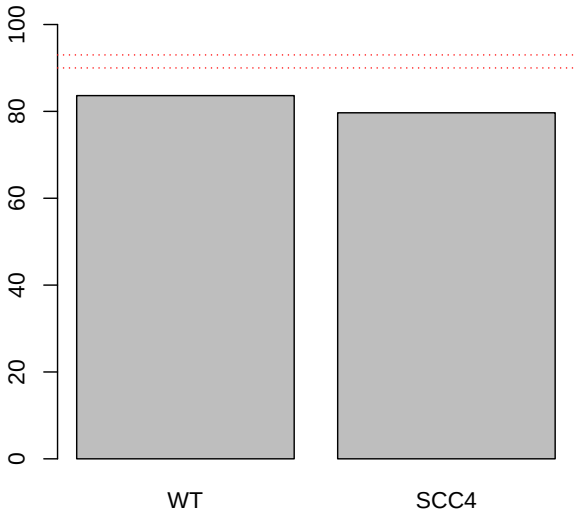


Figure 2: Fraction of cis-contacts per sample at the p-arms

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 $log_2(observed/expected)$. 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).

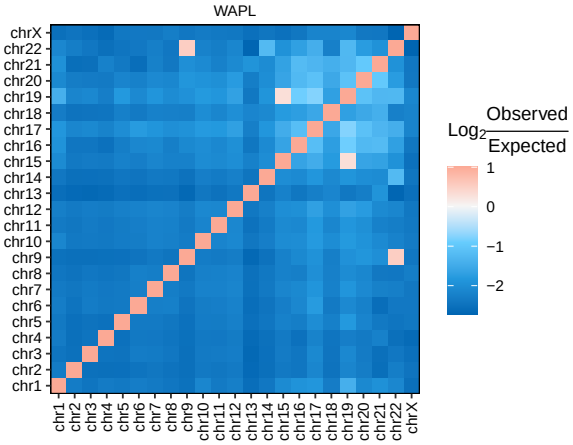


Figure 3: Chromosome matrix
The two known translocations of Hap1 cells are easily identified (15-19 & 9-22).

RCP

The Relative Contact Probability (RCP) 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 ignore centromeric contacts (which have a aberrant RCP), centromeric information is needed. This is taken from the experiment-object or found empirically by comparing trans-interactions.

```
RCP_out <- RCP(explist = list(Hap1_WT_40kb, Hap1_WAPL_40kb),
               chromsToUse = 'chr1')
```

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 ($[\pm$10Mb, $\pm$100Mb$]$): A- and B-compartments are often this size, so a next step could be to look more into compartmentalisation with `compartment_matrixplot` or `trans.compartment.plot`, for example.

```
visualise(RCP_out)
```

Differentials

We can directly compare samples to one another (for example WT versus WAPL). To plot this, the `metric` argument has to be set to `lfc` and `contrast` to 1, indicating the WT sample (figure 5). This plots the log-fold change of average probabilities.

```
# Plot RCP: combined
visualise(RCP_out, contrast = 1, metric = 'lfc')
```

Regions

But what if you want to compare the contact probabilities of specific regions, like Cohesin- or CTCF-bound regions? For this purpose, we added the possibility to add a `list` of BED-`data.frame`s to the `bedList`-argument. Under the hood, we perform a standard per-arm RCP (thus still enabling users to set the `chromsToUse`-parameter). Thereafter, we filter out Hi-C bins that do not have entries in the `data.frame`(s) of `bedList`. The same plot-function `visualise()` can be used.

```
CTCF = read.delim('../data/symlinks/CTCF_WT_motifs.bed', header = FALSE)
SMC1 = read.delim('../data/symlinks/SMC1_WT_peaks.narrowPeak', header = FALSE)

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

visualise(RCP_out)
```

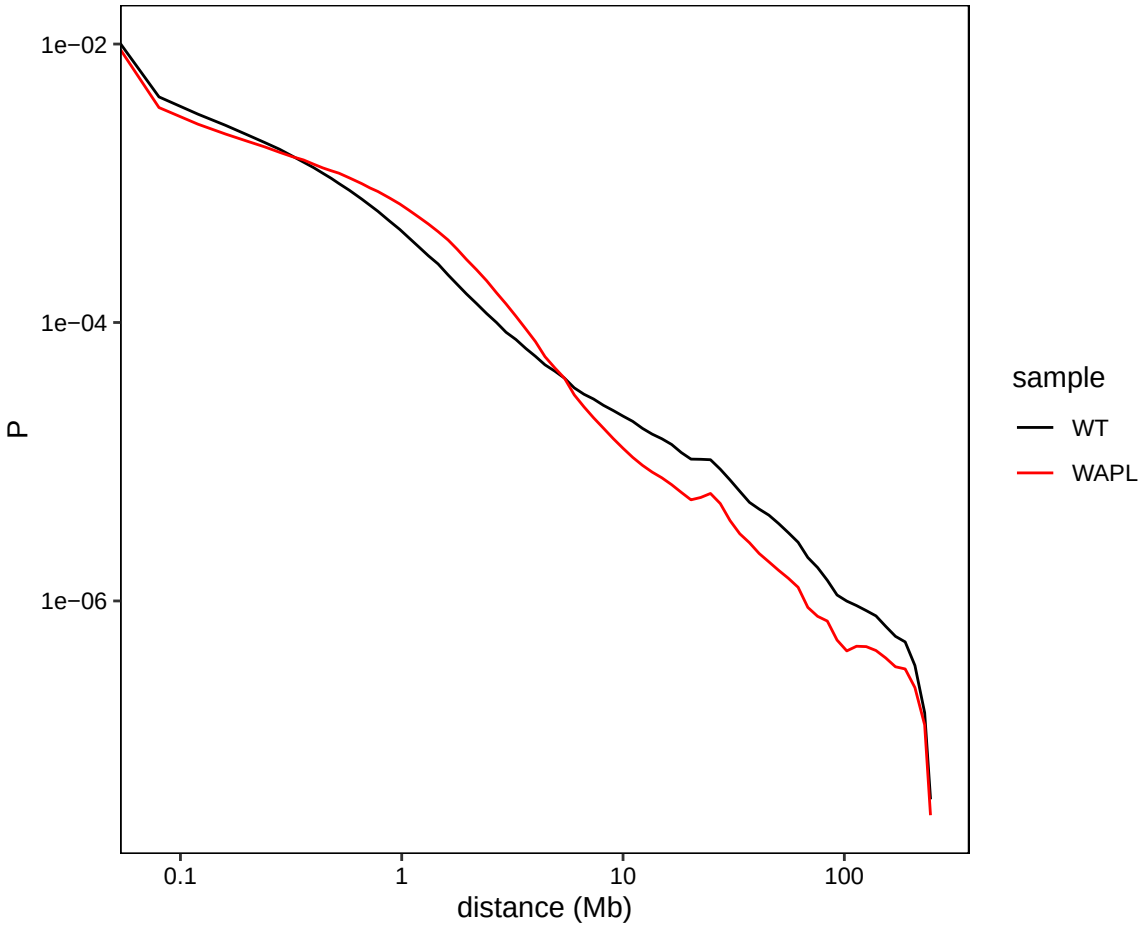


Figure 4: RCP
Every facet shows the RCP of one chromosome.

```
visualise(RCP_out, contrast = 1, metric = 'lfc')
```

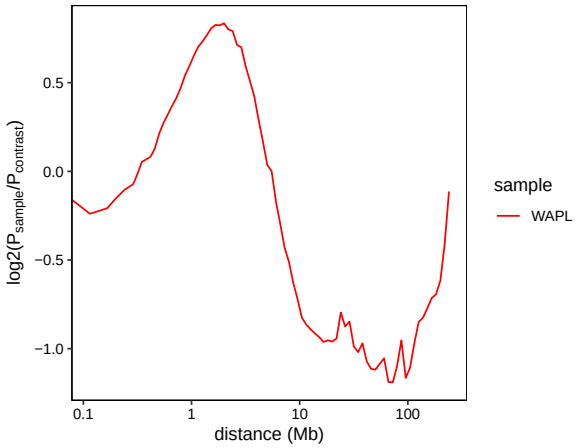


Figure 5: RCP in lfc-mode

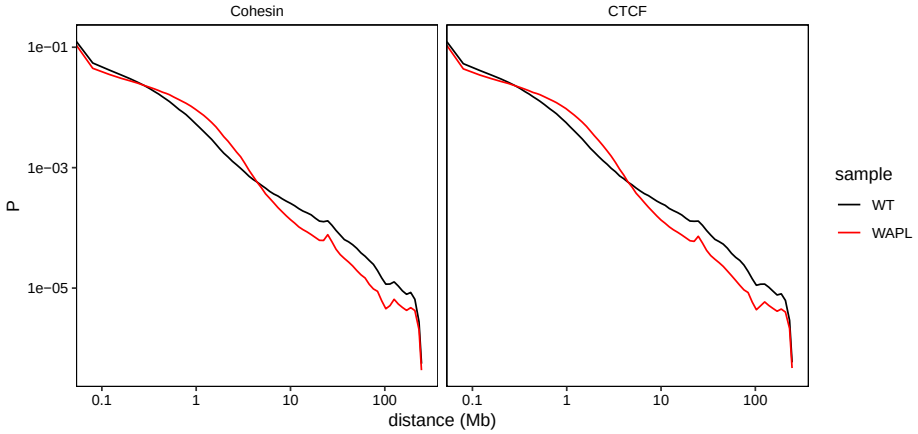


Figure 6: RCP with BEDs
We can also add BEDs as sites to compute the RCP.

A- and B-compartments

Dividing chromosomes into A- and B-compartments requires the calculation of a compartment score along with information about what parts of the genome are active. 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). Below, we can use the `compartment_score()` function with H3K27ac peaks to distinguish the compartments. The compartment score uses the first eigenvector of an eigendecomposition on the distance-dependant observed/expected matrix to get an unsigned compartment score. This score is then correlated to H3K27ac presence to yield signed compartment scores that can be interpreted as A compartments when positive and B compartments when negative.

```
H3K27acPeaks = read.delim('../data/symlinks/H3K27ac_WT.narrowPeak',
                          header = FALSE)

CS_out = compartment_score(list(Hap1_WT_40kb, Hap1_WAPL_40kb),
                           bed = H3K27acPeaks)
```

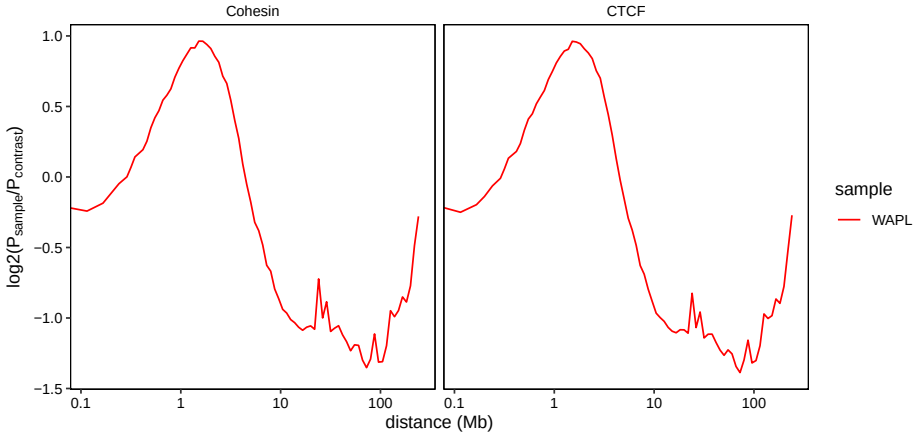


Figure 7: RCP with BEDs
We can also add BEDs as sites to compute the RCP.

```
visualise(CS_out, chr = "chr17")
```

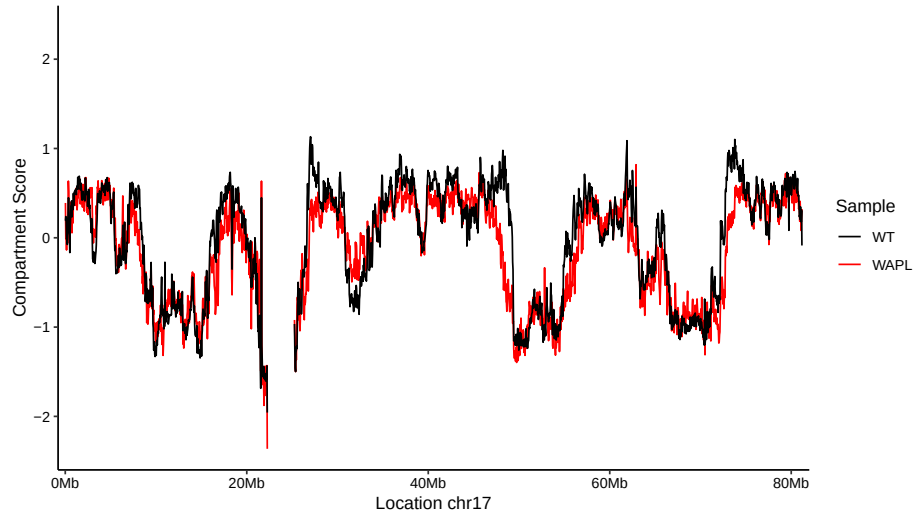


Figure 8: The compartment-scores on chromosome 17

Saddle-analyses

To illustrate the self-preference of compartments, one can use the `saddle()` function to perform a compartment score 'quantile versus quantile' analysis. Every chromosome arm is divided into quantiles based on the compartment score, and distance dependent observed over expected is computed. Every combination of quantile bins is then averaged to produce the saddle analysis.

```
saddle_out = saddle(list(Hap1_WT_40kb, Hap1_WAPL_40kb),
                    CS_discovery = CS_out,
                    bins = 50)
```

```
visualise(saddle_out)
```

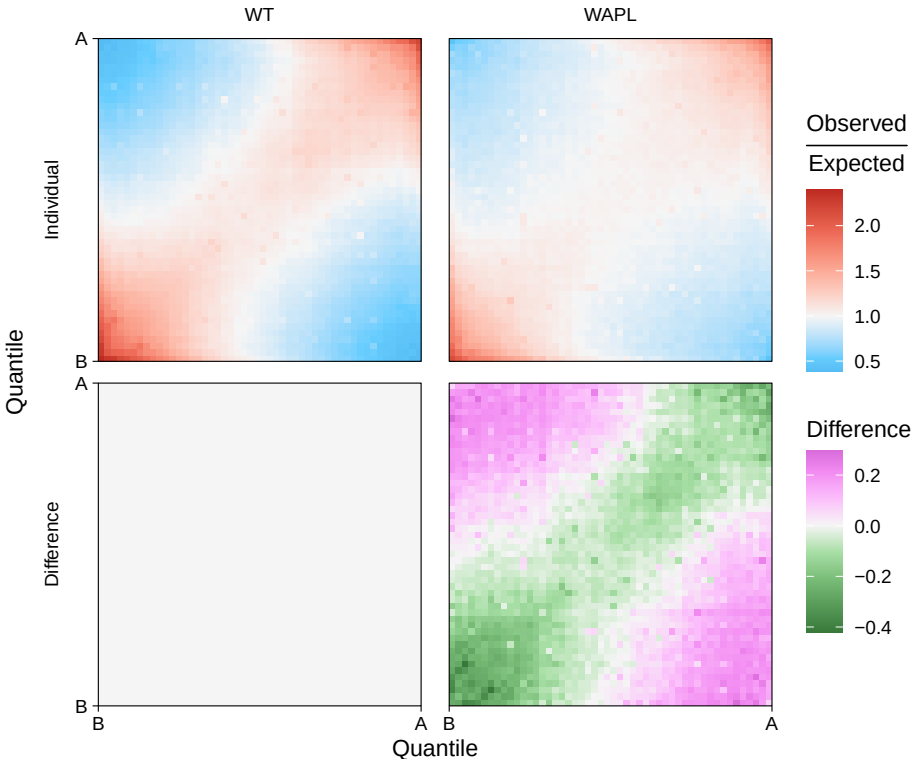


Figure 9: **A saddle-plot: WAPL-knockout cells have less intra-compartment enrichment**

Compartment-strength

The output of the saddle analysis gives a starting point for calculating the compartment strength, which gives a score for how much A-A and B-B interactions occur versus A-B interactions. The comparment strength can be calculated using the `quantify()` function on the output of the `saddle()` function.

```
CSS <- quantify(saddle_out)

compared <- tidyr::spread(unique(CSS[,-c(3,4)]), key = 'exp', value = 'strength')

with(compared, plot(WT, WAPL, xlim = c(0,4), ylim = c(0,4), pch = 20))
abline(a = 0, b = 1, lty = 1)
```

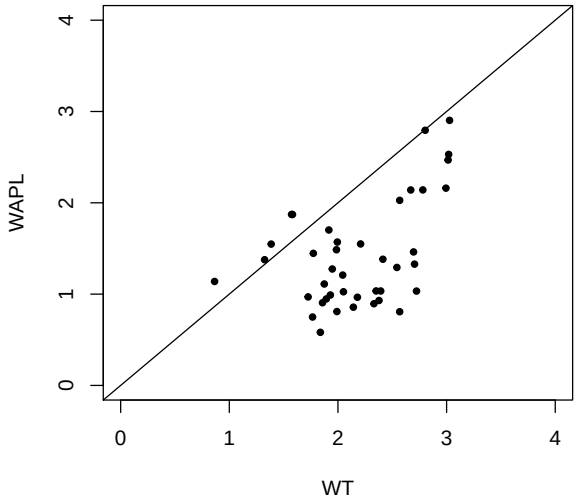


Figure 10: The per-arm compartment strength is at almost every arm lower in the WAPL knockout

Interaction plots

GENOVA has several plotting-functions for genomic loci. `compartment_matrixplot()` and *trans.compartment.plot* provide a easy way to plot whole chromosome arms, including compartmentalisation-score tracks. For more zoomed-in plots, `hic_matrixplot()` can be used. This function also allows rich annotation and between-experiment comparison possibilities. All functions try to guess the most appropriate colour-scale limits, but finer control of this can be gotten by setting the `colour_lim`-argument.

Cis-interactions

The compartmentalisation of the chromatin into A and B van already described in the original Hi-C paper (Lieberman-Aiden and Berkum 2009). Serval papers have described the loss of compartmentalisation when the Cohesin complex is stabalised Gassler et al. (2017). To view this interesting level of chromatin organisation, we can use `compartment_matrixplot()`. This function can plot one arm of a chromosome with the compartment-score plotted above. In figure 11 you can see the resulting plots, where you can see that the chequerboard-pattern in the matrix and the amplitude of the compartment-score are diminished in the WAPL-knockout.

```
compartment_matrixplot(
  exp1 = Hap1_WT_40kb,
  exp2 = Hap1_WAPL_40kb,
  CS_discovery = CS_out,
  chrom = "chr14", arm = "q",
  colour_lim = c(0, 15)
)
```

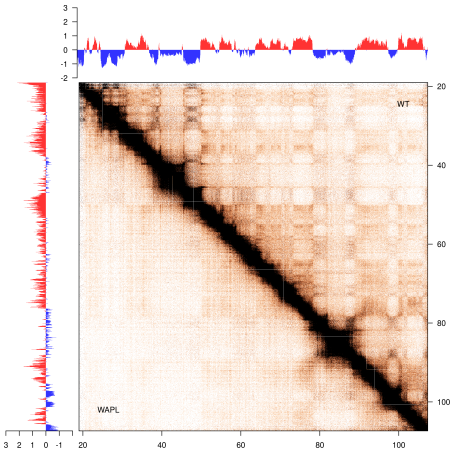


Figure 11: **Cis compartment plot: WT vs WAPL**
Stabilised Cohesin-mediated loops by WAPL-knockout leads to loss of compartments.

The compartment-score is calculated by performing an eigenvector decomposition on the correlation-matrix of the expected over expected matrix. To view this observed over expected matrix, we can set the `metric`-option to `"obsexp"`. Alternatively, we can set `metric = "correlation"` to view the Pearson correlation of the observed over expected matrix. These metrics gives a visually better indication of the A- and B-compartments (figure 12).

```
compartment_matrixplot(
  exp1 = Hap1_WT_40kb,
  exp2 = Hap1_WAPL_40kb,
  CS_discovery = CS_out,
  chrom = "chr20", arm = "q",
  metric = "log2obsexp"
)
```

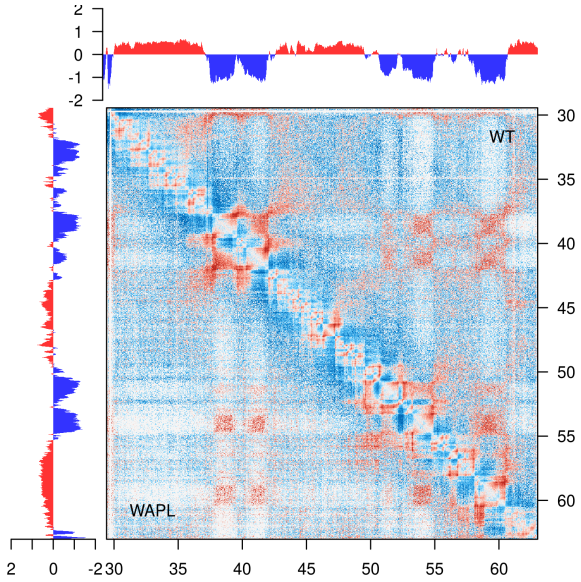


Figure 12: Cis compartment plot
Observed over expected.

Trans-interactions

The function `trans_matrixplot()` will allow the user to plot a trans-matrix (i.e. a matrix of the arms of two different chromosomes). This function could also be used to investigate chromosomal translocations: the 9q;22q translocation can be clearly seen if we use this function, as in figure 13.

```
trans_matrixplot(
  Hap1_WT_40kb,
  chrom_up = "chr9", start_up = 100e6, end_up = 140e6,
  chrom_down = "chr22", start_down = 16e6, end_down = 40e6,
  colour_lim = c(0, 20)
)
```

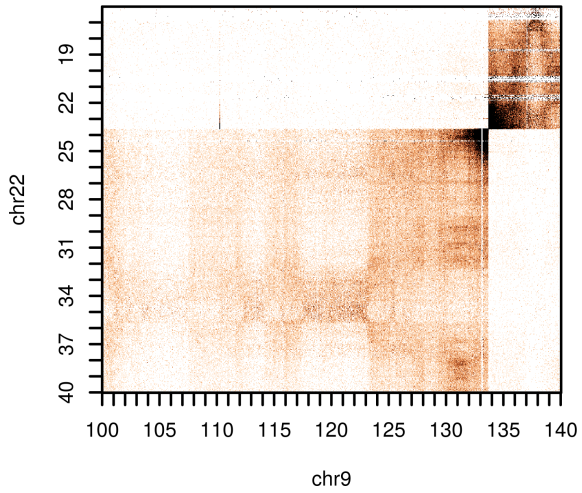


Figure 13: Trans compartment plot
The t(9q;22q) translocation is easily identified.

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 14).

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

Two experiments

Adding a second experiment will give us the option of `coplot`, which can be `dual` (default) or `diff`. The left plot shows the WT sample in the upper triangle and KO sample in the lower. In the right plot, the KO sample is subtracted from the WT in `diff`-mode: red is therefore more contacts in the KO and blue denotes more contacts in the WT (figure 15).

```
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,
```

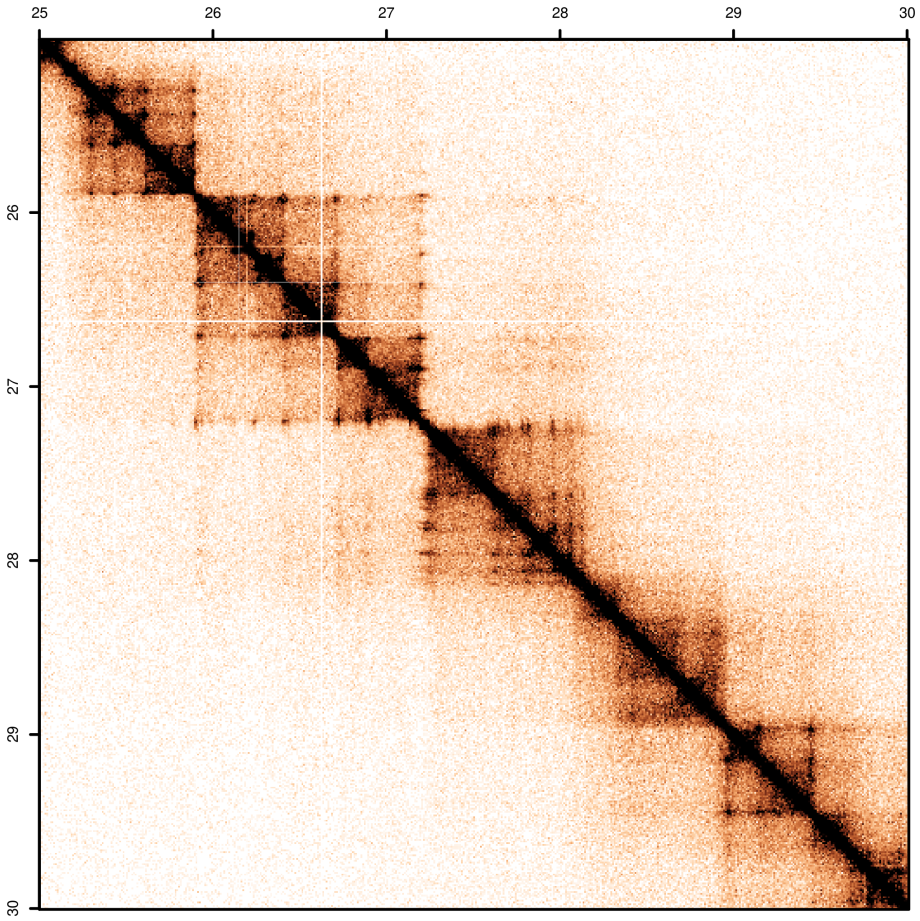


Figure 14: Hi-C matrixplot
Simplest example: one experiment, no annotation

```
                    coplot = 'diff',
                    chrom = 'chr7',
                    start = 25e6,
                    end=30e6, # upper limit of contacts
                    cut.off = 25)
```

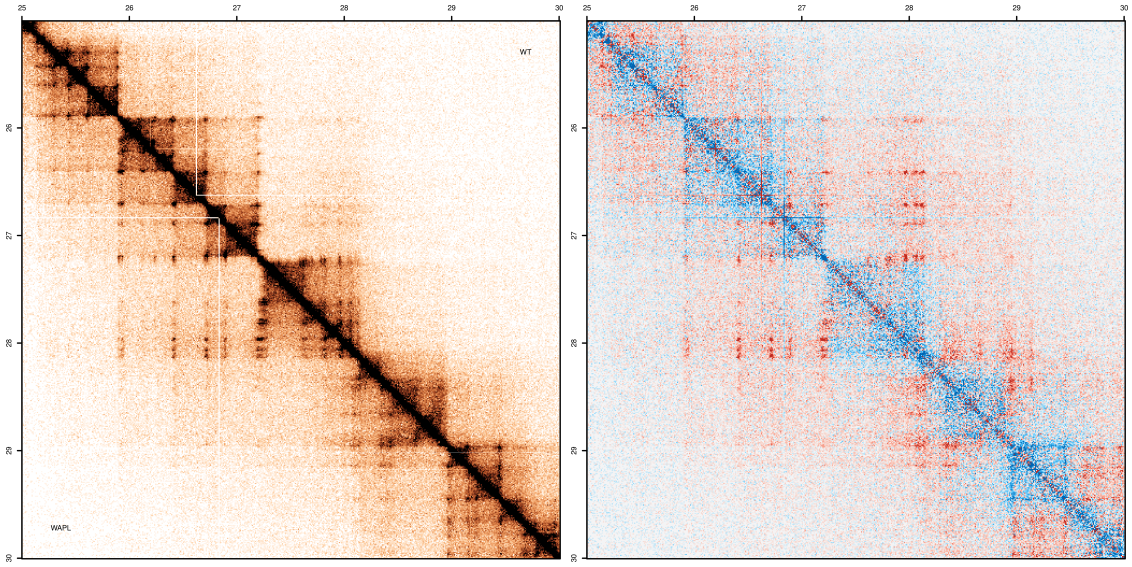


Figure 15: Hi-C matrixplot with two experiments: dual vs diff mode
The extended loops in the WAPL knockout are easily seen at around 28Mb in the lower triangle in dual-mode (left panel) and as red points in diff-mode (right panel).

TADs and loops

It can be very useful to annotate the matrix with the positions 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/symlinks/WT_hicseg_TADs.bed', h = F)
WT_Loops = read.delim('../data/symlinks/WT_HICCUPS.bedpe', h = F, skip = 1)
sanborn2015_Loops = read.delim('../data/symlinks/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 triangles and coloured as deemed appropriate (figure 16). 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 distance in basepairs using `loops.resize`. This will draw a circle around the loop location.

```
hic_matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               loops = WT_Loops, # see APA
               loops.colour = 'blue', # blue loops
               loops.type = 'upper', # only plot in upper triangle
               loops.radius = 20e3, # expand for visibility
               tads = WT_TADs, # see ATA
               tads.type = 'lower', # only plot in lower triangle
               tads.colour = 'limegreen', # green TAD-borders
```

```
                    cut.off = 25) # upper limit of contacts
```

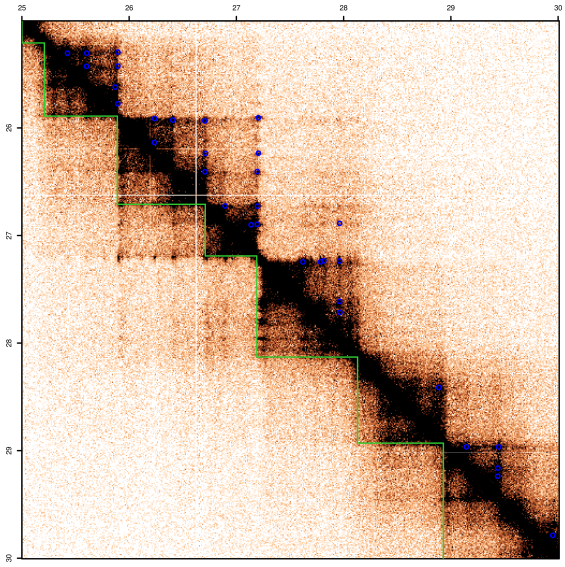


Figure 16: Hi-C matrixplot: TAD- and loop-annotations from Haarhuis et al (2017).

BigWigs and BEDs

Manipulation of CTCF-binding sites can result in loss or gain of loops and/or TADs (Wit et al. 2015). If you are 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 helpful. You can add up to two tracks above and two tracks to the left. These can be either BED-like `data.frame`s or the paths to the `.bw` files. For example, let's load a BED6-file (chrom, start, end, name, score, and strand)[2] of CTCF-motifs under CTCF-ChIP peaks. The argument `type` can be set to either *triangle* or *rectangle*. The triangle is nice to use if you want to look at the orientation of the BED-entries (figure 17). If you only have a three column BED, then the output will always be *rectangle*.

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

```
CTCF = read.delim('../data/symlinks/CTCF_WT_motifs.bed', h = F)
SMC1 = read.delim('../data/symlinks/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 load the bigwig data with the `rtracklayer` package that is available from Bioconductor. The `yMax` argument is handy if you want to compare bigwig-tracks: it lets you set the y-axis maximum.

```
requireNamespace("rtracklayer", quietly = TRUE)

hic_matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',  start = 26.75e6,  end=28.5e6,
               loops = WT_Loops, # see APA
               loops.colour = 'purple', # purple loops
               loops.type = 'upper', # only plot in upper triangle
               loops.radius = 20e3, # expand for visibility
               type = 'triangle',
               chip = list('../data/symlinks/SMC1_WT.bw', # inner top
                           CTCF),# outer-top
               symmAnn = F, # place annotations also on left side
               cut.off = 65) # upper limit of contacts
#> chip.yMax not given for a .bw-track: yMax is 2.32372999191284
#> chip.yMax not given for a .bw-track: yMax is 2.32372999191284
```

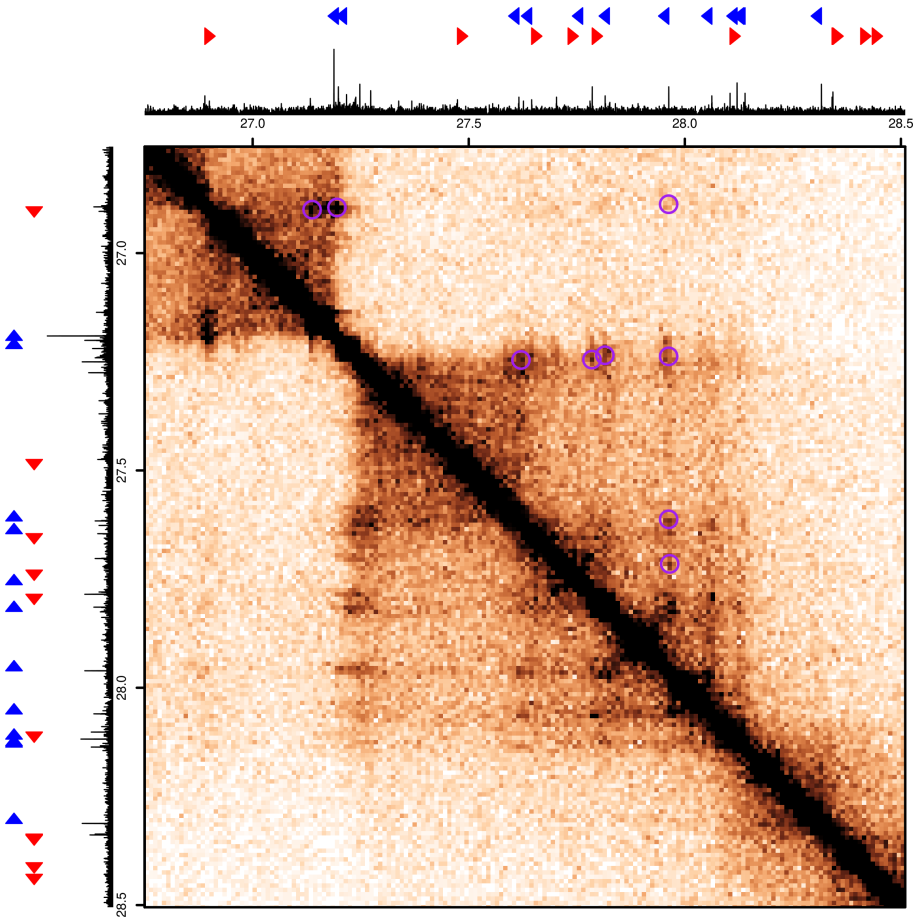


Figure 17: Hi-C matrixplot: ChIPseq
A BED-file of CTCF-sites is plotted at the top and a coverage-track of SMC1 ChIP-seq is plotted beneath this. The symmAnn-option leads to the same tracks being plotted on the left.

Genes

(Dixon et al. 2012) showed that housekeeping-genes are enriched in the vicinity of TAD-borders. Another interesting question could be whether 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 appropriate gene-models to `hic_matrixplot()`. To do this, we use a `data.fame`, where each row is an exon of a gene. There are several ways to get this, and one of the easier ways is to use biomart to get exon-coordinates. You could use the biomaRt-package or 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 18).

```
# 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 +/-
load('../data/symlinks/../martExport.Rdata')
```

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

	chrom	txStart	txEnd	exonStart	exonEnd	strand
38182	chr7	1022835	1029276	1022835	1023021	+
38183	chr7	1022835	1029276	1024048	1024210	+
38184	chr7	1022835	1029276	1024586	1024735	+
38185	chr7	1022835	1029276	1024802	1024959	+
38186	chr7	1022835	1029276	1026260	1026433	+

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

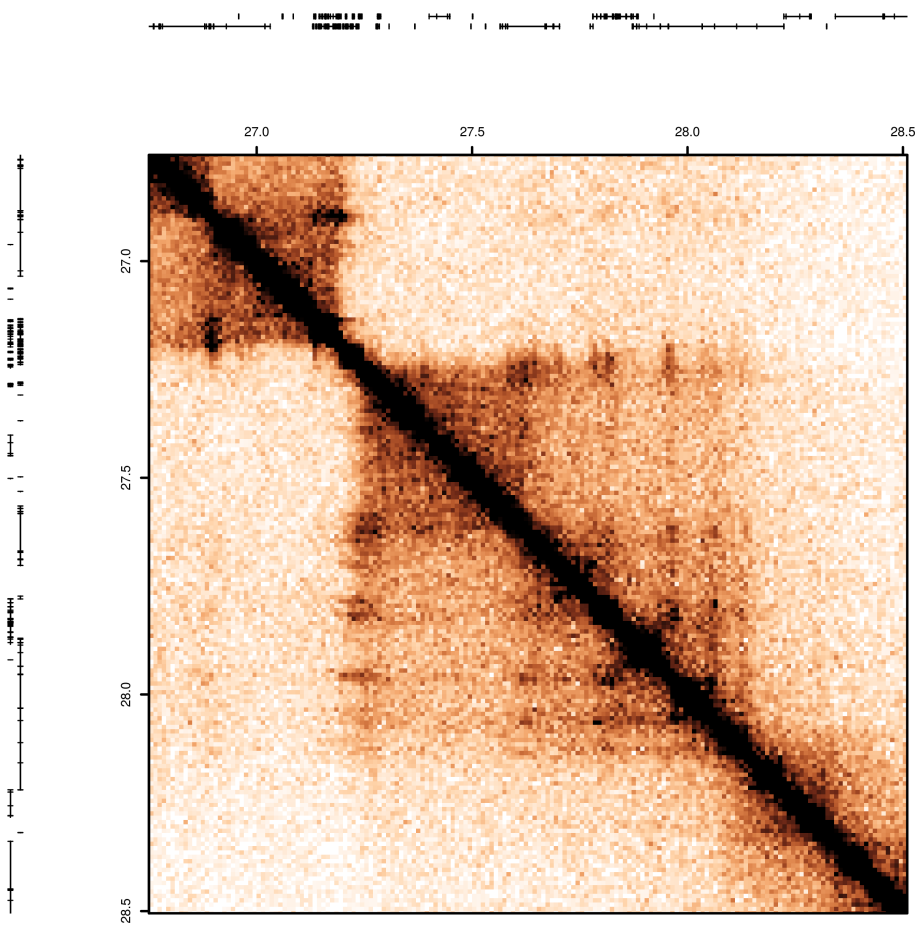


Figure 18: Hi-C matrixplot: genes

Everthing together

Finally, we can combine all these options in one. This might put too much information in a single plot, but it can be quite handy. In this example, we can see that most TAD-borders and loop-anchors have clear SMC1- and CTCF-signal (figure 19). 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, respectively. 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.colour = '#998ec3', # purple loops
               loops.type = 'upper', # only plot in upper triangle
               loops.radius = 20e3, # expand for visibility
               genes = martExport,
               chip.colour = 'black',
               chip = list('../data/symlinks/SMC1_WT.bw', # inner-top
```

```
                    SMC1, # outer-top
                    '../data/symlinks/SMC1_WT.bw', # inner-left
                    CTCF), # outer-left
              tads = WT_TADs, # see ATA
              tads.type = 'lower', # only plot in lower triangle
              tads.colour = '#91cf60', # green TAD-borders
              cut.off = 50) # upper limit of contacts
```

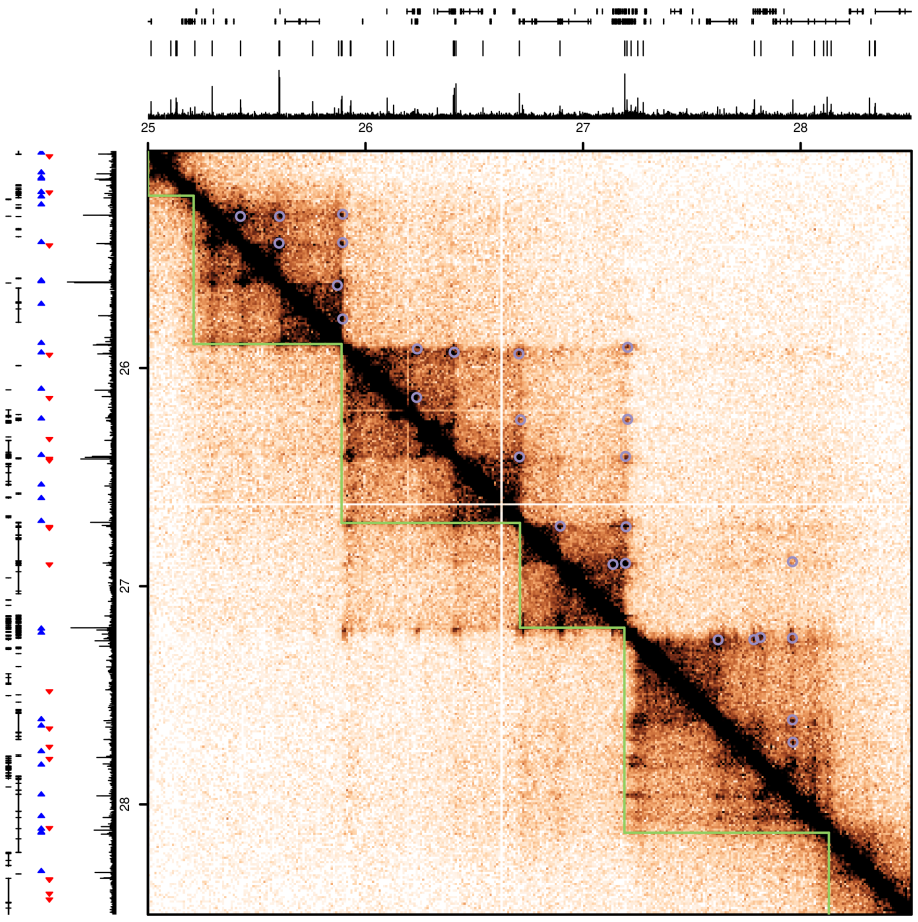


Figure 19: Hi-C matrixplot: a complex case
Loops and TADs are annotated within the Hi-C matrix. On the top annotation-bar, we have plotted the ChIP-seq signal and peaks of SMC1. On the left annotation-bar are the ChIP-seq signal and peaks (with orientiation) of CTCF. Genes are plotted on both annotation-bars.

Pyramid plots

In GENOVA, 'pyramid plots' are plots of a region in the Hi-C interaction map rotated at an 45 degree angle (figure 20). This rotation coincides the diagonal with the x-axis, and the y-direction indicates distance from the diagonal.

```
pyramid(exp = Hap1_WT_10kb,
        chrom = 'chr7',
        start = 25e6,
        end=28.5e6,
        colour = c(0, 50))
```

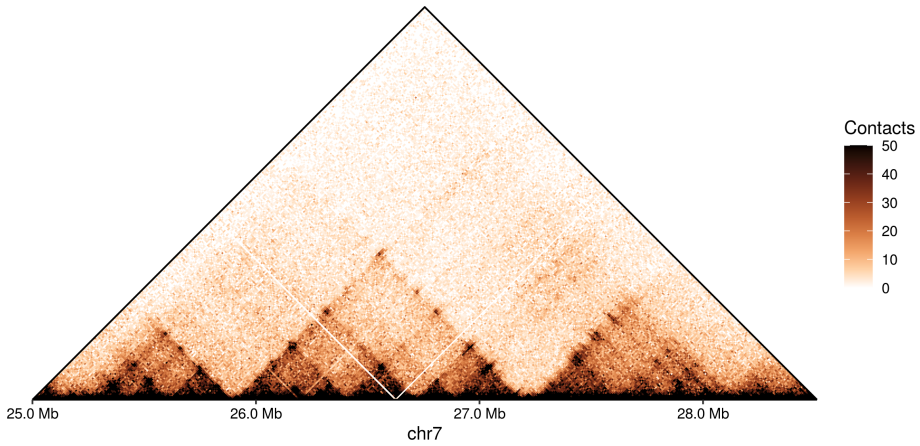


Figure 20: A pyramid plot, showing a rotated Hi-C interaction map

Pyramid plots can be cropped in the x-and y-direction to make them more space efficient (figure 21).

```
pyramid(exp = Hap1_WT_10kb,
        chrom = 'chr7',
        start = 25e6,
        end=28.5e6,
        crop_x = c(26e6, 27.5e6), # cropping in x direction uses locations
        crop_y = c(0, 2e6), # cropping in y direction uses distance
        colour = c(0, 50))
```

For comparing two different samples with pyramid plots, there are two options. The first option is to use `pyramid_difference()`, which works very similar to `pyramid()`, but takes two samples instead of one 22).

```
pyramid_difference(
  Hap1_WT_10kb,
  Hap1_WAPL_10kb,
  chrom = "chr7", start = 25e6, end = 28.5e6
)
```

The alternative option is to stack the plots together using plot composition. Because `pyramid()` is based on the ggplot2 package, we can use the patchwork plot composition package to stack two (or more) plots 23).

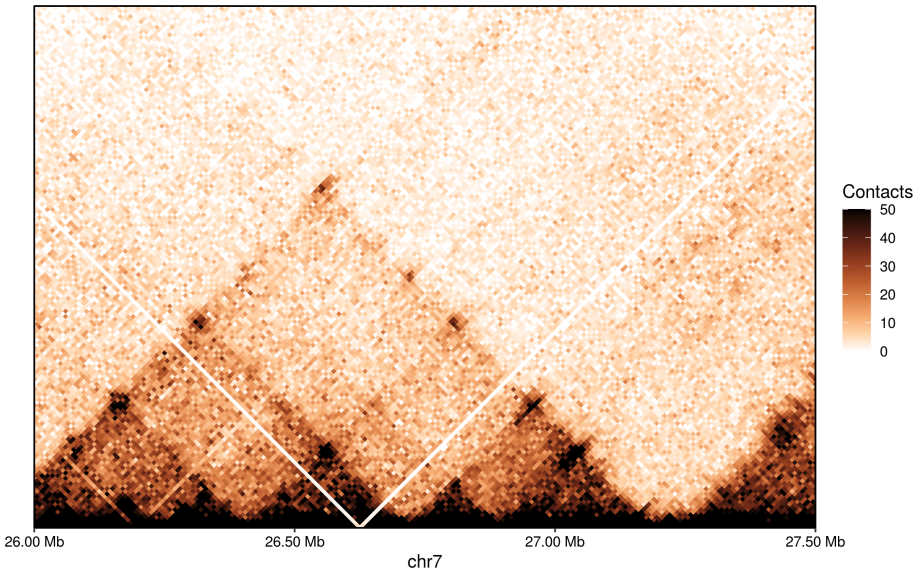


Figure 21: A cropped pyramid no longer looks like a pyramid

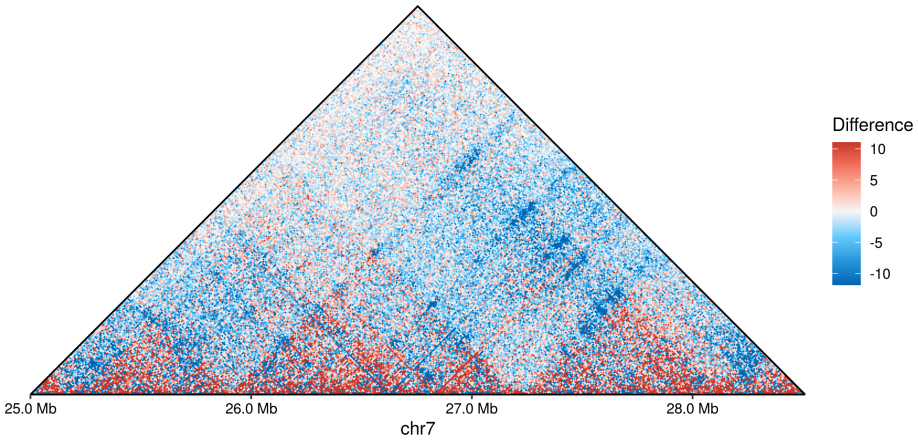


Figure 22: A pyramid plot showing the difference between two samples

```
library(patchwork)

wt_pyramid <- pyramid(
  Hap1_WT_10kb,
  chr = "chr7", start = 25e6, end = 28.5e6,
  crop_y = c(0, 1.5e6),
  colour = c(0, 50)
) + ggplot2::ggtitle("Wildtype")

ko_pyramid <- pyramid(
  Hap1_WAPL_10kb,
  chr = "chr7", start = 25e6, end = 28.5e6,
  crop_y = c(0, 1.5e6),
  colour = c(0, 50)
```

```
        ) + ggplot2::ggtitle("WAPL KO")

        ko_pyramid / wt_pyramid + plot_layout(guides = "collect")
```

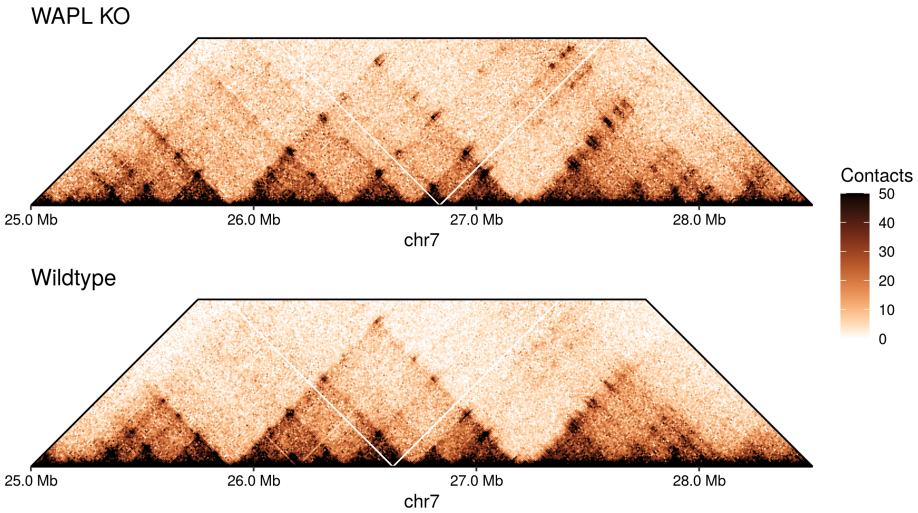


Figure 23: Using the patchwork package to stack pyramid plots

TADs

Topologically Associated Domains (TADs) are $\pm 0.8 - 2$Mb regions, which are seen as triangles in the matrix: regions that have more interactions within than outside. GENOVA has a repertoire 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 aggregates of TADs. Finally, we'll investigate whether TADs interact with their neighbouring TADs.

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

Domainogram

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

```
ID <- insulation_domainogram(
  Hap1_WT_10kb,
  chrom = 'chr7',
  start = 25e6,
  end   = 29e6,
  window_range = c(1, 101),
  step = 2
)
visualise(ID)
```

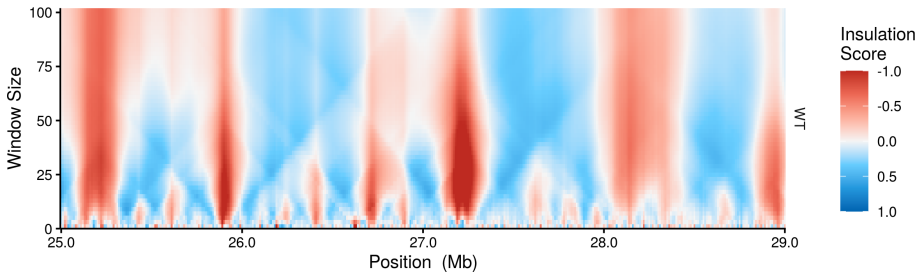


Figure 24: Insulation domainogram
Insulation-hotspots can be identified in red, which are regions with a very negative score.

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 25).

```
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.colour = '#91cf60', # green TAD-borders
               cut.off = 25, # upper limit of contacts
               skipAnn = T) # skip the outside annotation
plot(ID, minimalist = TRUE)
```

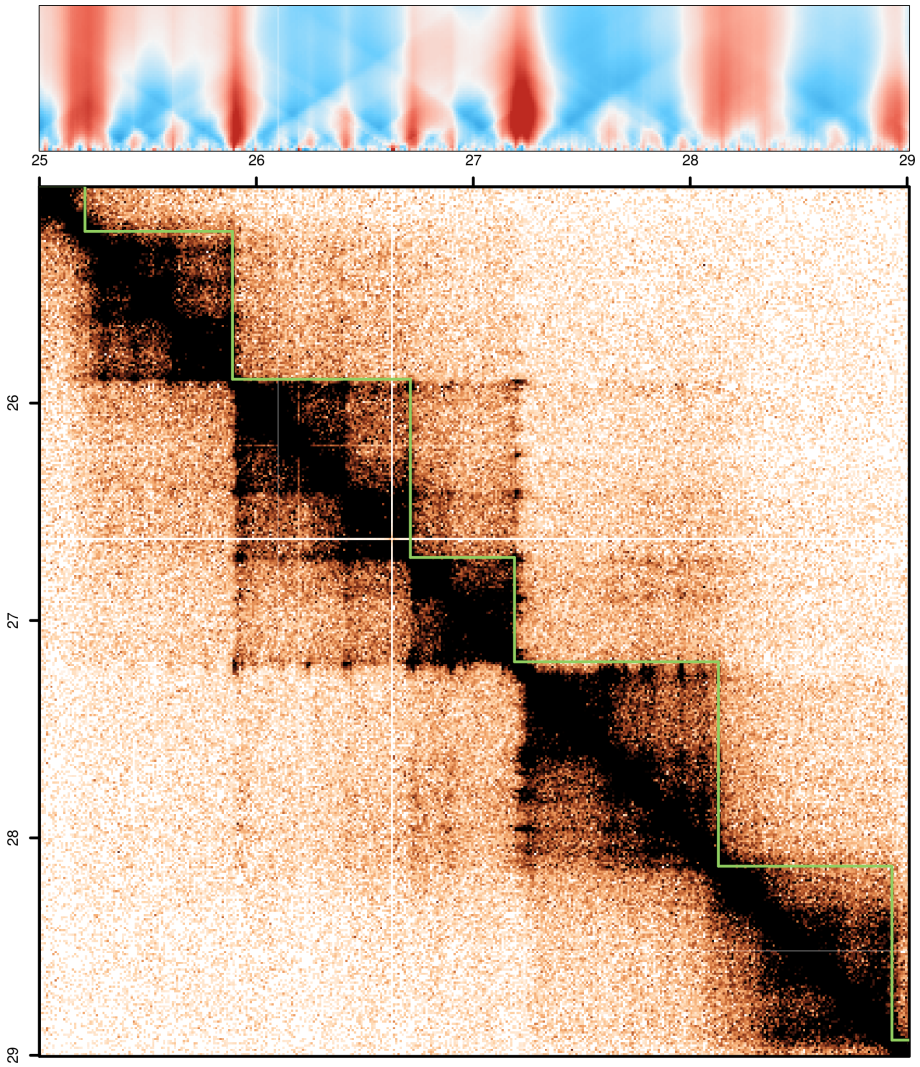


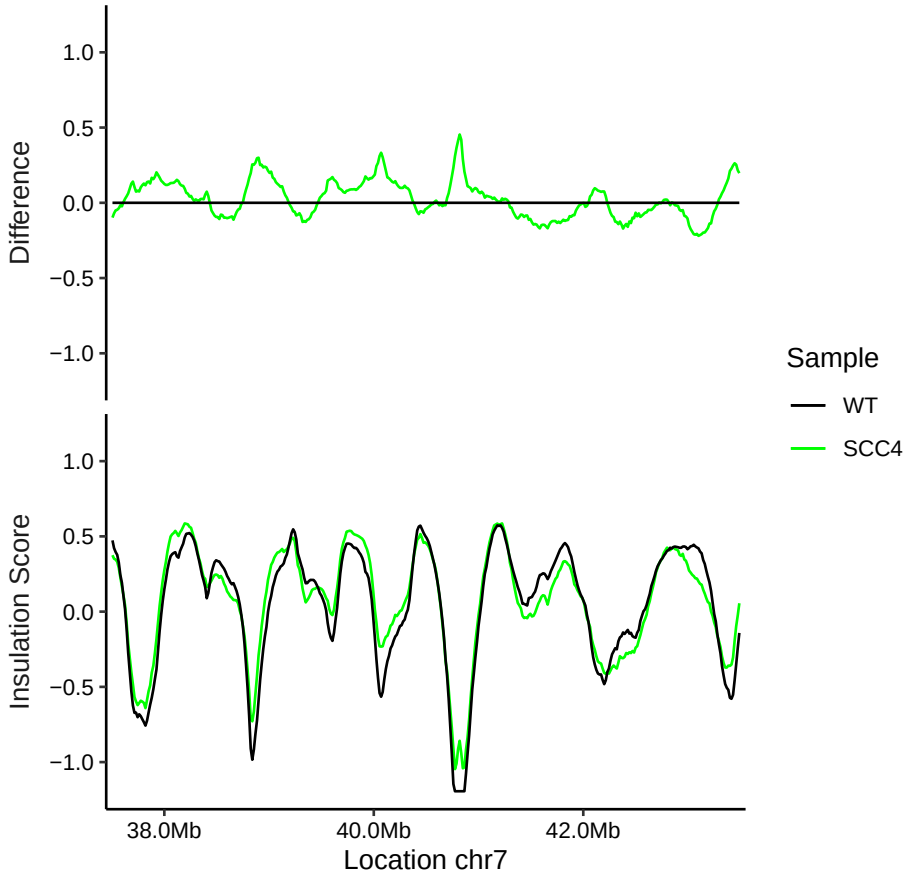
Figure 25: Insulation domainogram with Hi-C matrix
The insulation-hotspots are the sites where HiC-seg has called a TAD-border.

Computing the insulation score

To get the genome-wide insulation score in .bedgraph-format,[3] we provide the `insulation_score()` function, that takes a contacts-object and the window-size of choice. As can be seen in the domainogram above, at $W = 25$ we will catch the majority of the hotspots, while limiting the amount of noise. The `visualise()`-function can show both the insulation-scores and the difference between them.

[3] BED3 + signal column

```
Hap1_10kb_insulation <- insulation_score(
  list(Hap1_WT_10kb, Hap1_SCC4_10kb),
  window = 25
)
visualise(Hap1_10kb_insulation,
          chr = 'chr7', start = 25e6, end=29e6,
          contrast = 1)
```



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 26 we can see that the average signal drops at the border (which is to be expected) and that this is a genome-wide feature, as we see in the heatmap.

```
tornado_insulation(Hap1_10kb_insulation, bed = CTCF, bed_pos = 'center')
```

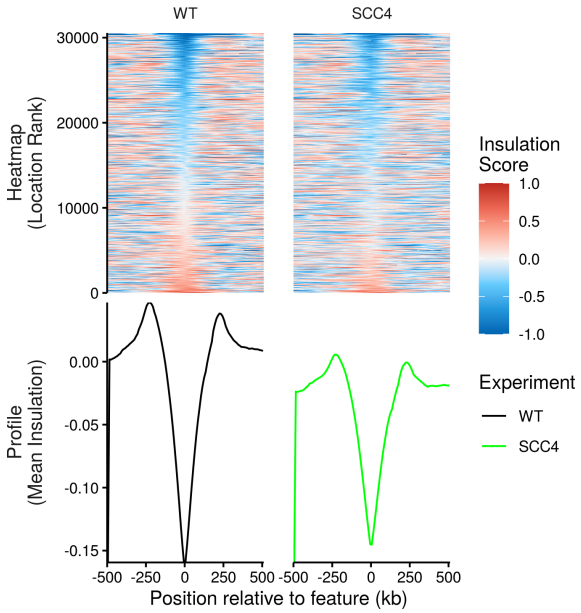


Figure 26: Insulation heatmap
The upper panel shows the average score. Each row is a TAD-border in the lower panel.

Call TADs

The `insulation`-discovery object can also be used to call TADs.

```
TADcalls <- call_TAD_insulation(Hap1_10kb_insulation)
```

```
hic_matrixplot(exp1 = Hap1_WT_10kb,
               exp2 = Hap1_SCC4_10kb,
               chrom = 'chr7',
               start = 24e6,
               end=27e6,
               tads = list(TADcalls$SCC4, TADcalls$WT), # see ATA
               tads.type = list('lower', 'upper'), # only plot in lower triangle
               tads.colour = c('green', 'grey'), # green TAD-borders
cut.off = 25) # upper limit of contacts
```

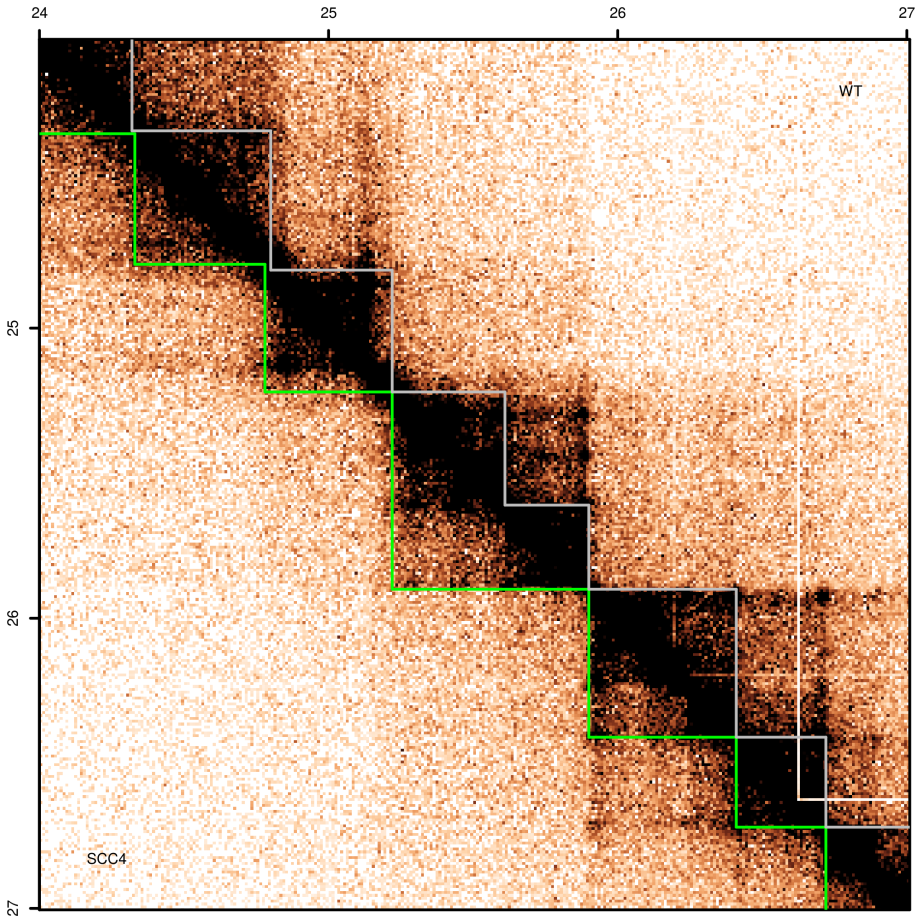


Figure 27: TADs called within GENOVA

ATA

TADs can be investigated globally by aggregating Hi-C matrix around TADs. In aggregate TAD analyses (ATA), because TADs have different sizes, they are rescaled to a uniform size and then the result is averaged across the genome.

```
ATA_Hap1_WTcalls <- ATA(list("WT" = Hap1_WT_10kb,
                             'WAPL' = Hap1_WAPL_10kb),
                        bed = TADcalls$WT)
```

We can use `visualise()` to plot the ATA-results.

```
visualise(ATA_Hap1_WTcalls,
          colour_lim = c(0,50),
          colour_lim_contrast = c(-5,5),
          metric = "diff",
          focus = 1) # which entry to use as comparison
```

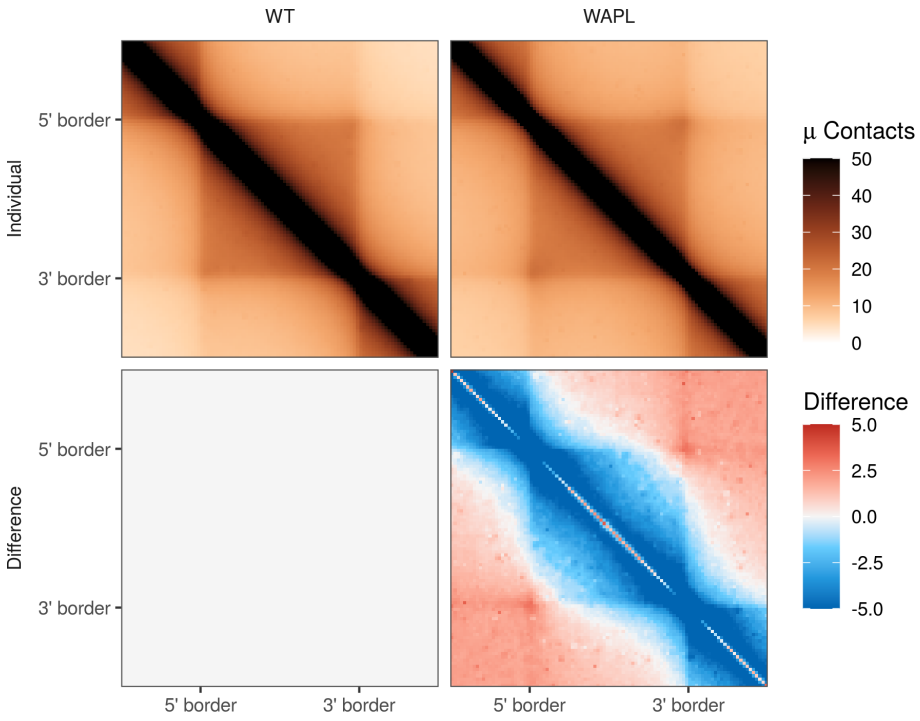


Figure 28: ATA
In the WAPL-knockout, we see a decrease of contacts within the TAD, but an increase at the corner.

TAD+N

A TAD+N analysis computes the interaction density within TADs and their 1,2,. . . ,N neighbours. This can be used to compare whether TADs in two samples interact differently with their neighbouring TADs.

```
TAD_N_WT   <- intra_inter_TAD(list("WT" = Hap1_WT_10kb,
                                   'WAPL' = Hap1_WAPL_10kb),
                              tad_bed = TADcalls$WT,
                              max_neighbour = 10)
```

We can compute the enrichment of contacts between TADs with the `visualise()`-function.

```
visualise(TAD_N_WT, geom = 'jitter')
```

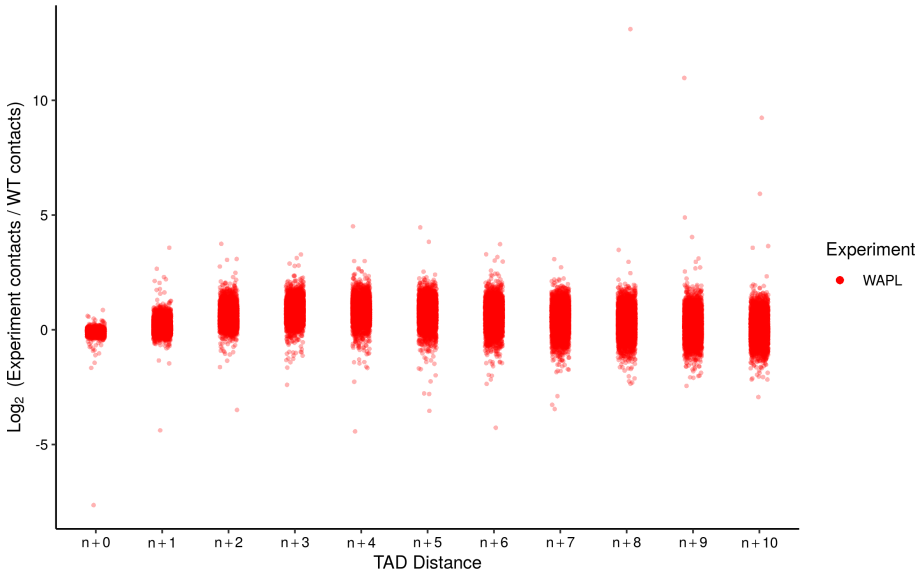


Figure 29: Differential TAD-analysis
Experiment 2 (WAPL) has more interactions between neighbouring TADs compared to wild type.

```
visualise(TAD_N_WT, geom = 'violin')
```

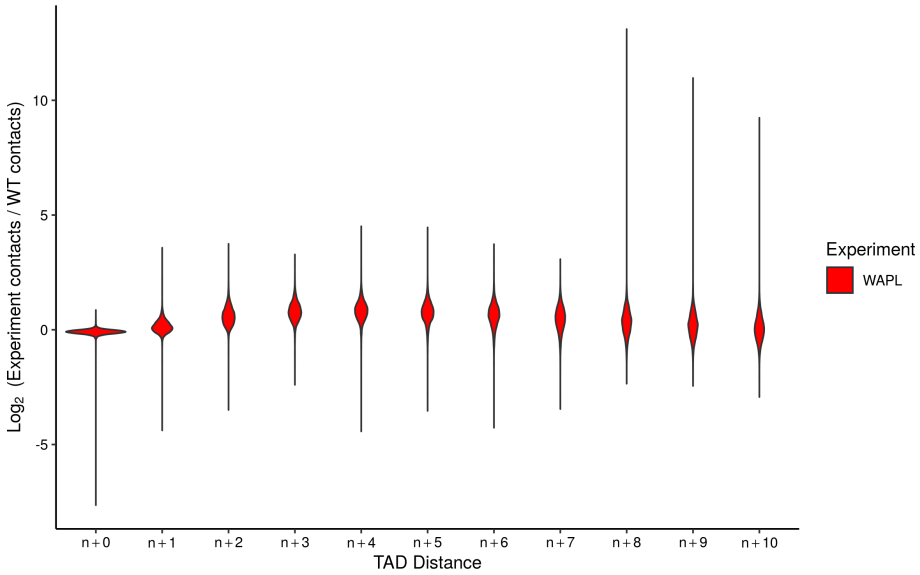


Figure 30: Differential TAD-analysis
Experiment 2 (WAPL) has more interactions between neighbouring TADs compared to wild type.

Loops

For this section, we are using both the called primary and the extended loops from Haarhuis et al. (2017). The primary loops 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). The extended loops are called by taking the primary loops and taking combinations of 5' and 3' anchor locations that are larger than the input loops, up to some distance.

```
# Anchors are defined as a matrix with two columns of bin IDs
WT_Loops_extended = anchors_extendedloops(Hap1_WT_10kb$IDX,
                                          res = resolution(Hap1_WT_10kb),
                                          bedpe = WT_Loops)
```

\begin{table}

\caption{Anchor-combinations of anchors_extendedLoops}

V1	V2
109	146
109	168
109	170
109	186
109	234

\end{table}

APA

Aggregate peak analysis (APA) looks up small portions of the Hi-C matrix from a twodimensional set of locations, such as loops. These are then aggregated to get a genome wide impression of the loops.

```
APA_Hap1_WT  <- APA(list("WT" = Hap1_WT_10kb,
                         'WAPL' = Hap1_WAPL_10kb),
                    dist_thres = c(200e3, Inf),
                    bedpe = WT_Loops)

APA_Hap1_WT_extended <- APA(list("WT" = Hap1_WT_10kb,
                                 'WAPL' = Hap1_WAPL_10kb),
                            bedpe = WT_Loops,
                            dist_thres = c(200e3, Inf),
                            anchors = WT_Loops_extended)
```

Once again, we can use `visualise()` to make plots of the APA results.

```
visualise(APA_Hap1_WT,
          title = "Hap1 Hi-C WT vs WAPL-KO loops",
          colour_lim = c(0, 40), # set the colour limits of the upper row
          colour_lim_contrast = c(-5, 5),
          metric = "diff",
          contrast = 1) # Compare against 1st sample in APA_Hap1_WT
```

```
visualise(APA_Hap1_WT_extended,
          title = "Hap1 Hi-C WT vs WAPL-KO extended loops",
          colour_lim = c(0, 8), # set the colour limits of the upper row
          colour_lim_contrast = c(-4, 4),
          metric = "diff",
          contrast = 1) # Compare against 1st sample in APA_Hap1_WT_extended
```

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

```
quantifyAPA_out <-  quantify(APA_Hap1_WT)
quantifyAPA_out_extended <-  quantify(APA_Hap1_WT_extended)
#> Warning: No IDX found. Returning location as bins.

#> Warning: No IDX found. Returning location as bins.

# plot boxplot with base-R (ggplot2 would be also easy)
boxplot(split(quantifyAPA_out_extended$per_loop$foldchange,
              f = quantifyAPA_out_extended$per_loop$sample),
        col = c('red', 'darkgrey'), outline = F,
        ylab = 'pixel enrichment extended loops')
```

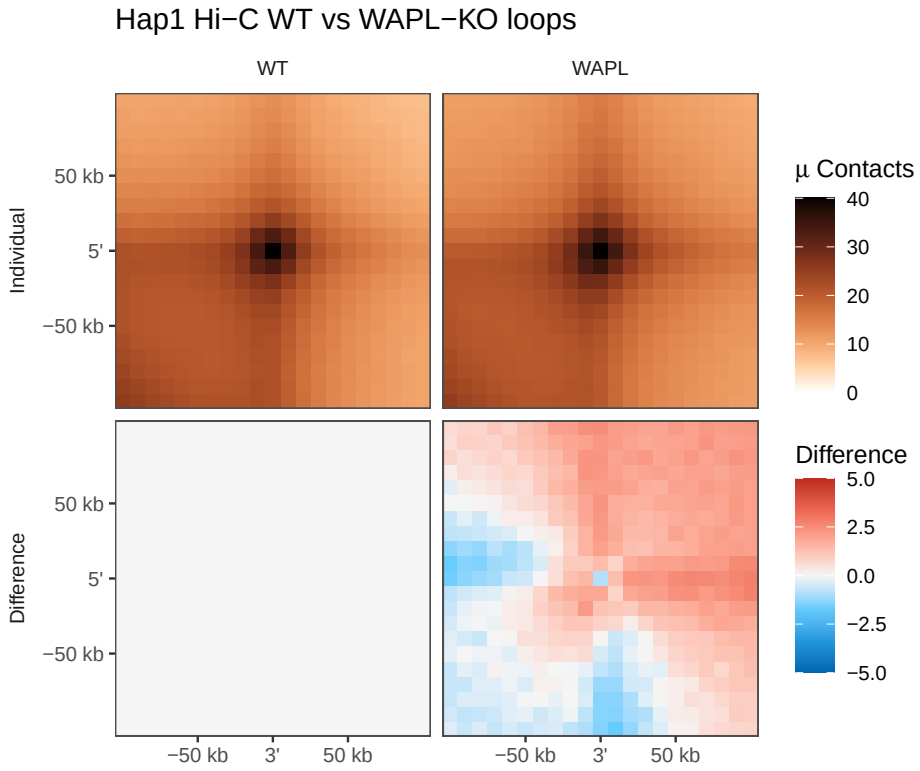


Figure 31: APA
In the WAPL-knockout, we see an increase of contacts at the loop.

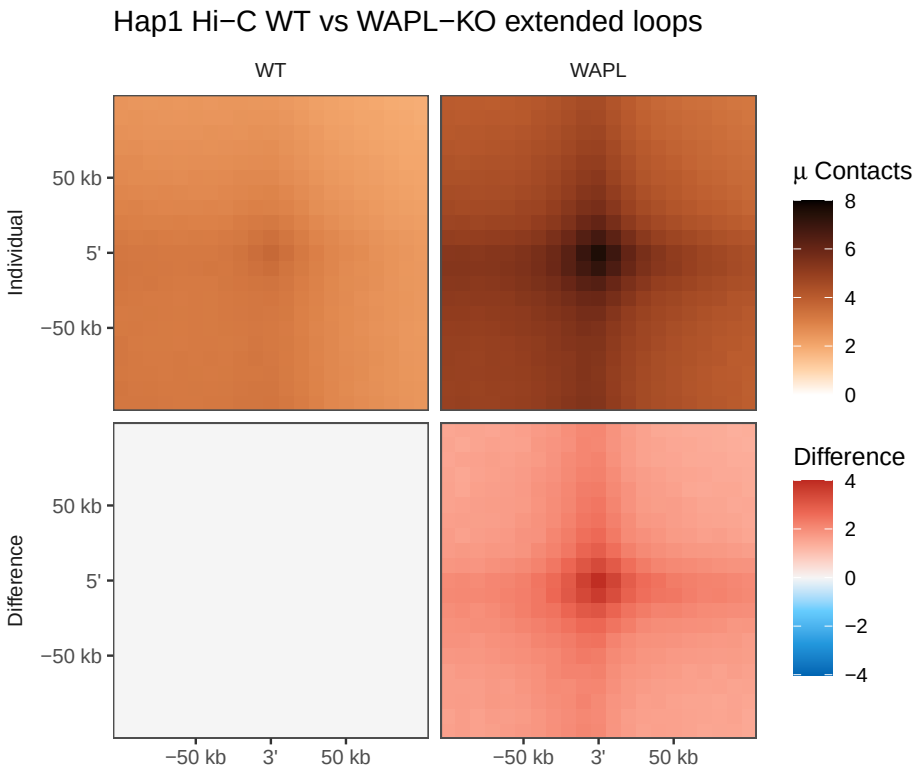


Figure 32: APA
In the WAPL-knockout, we see an increase of contacts at the loop.

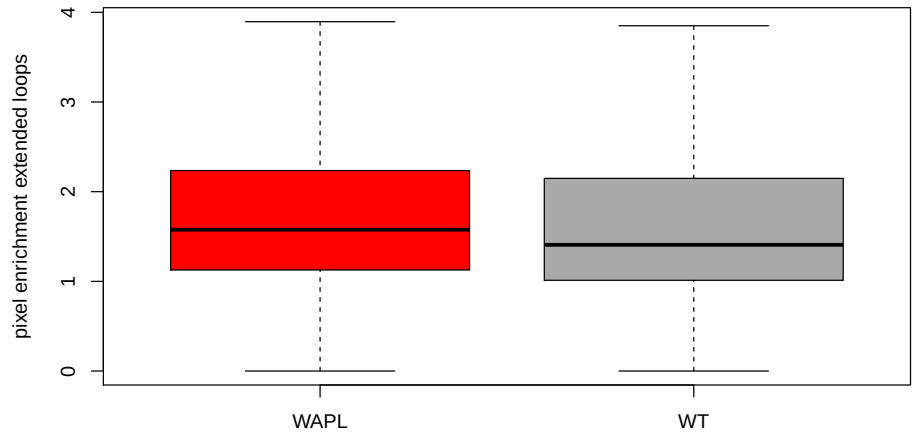


Figure 33: With quantifyAPA In the WAPL-knockout, we see an increase of contacts at the loop

Far-cis interactions

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/symlinks/superEnhancers.txt',
                            header = FALSE,
                            comment.char = "#")
```

Table 5: 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 standard 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(list(Hap1_WT_40kb, Hap1_WAPL_40kb), bed = superEnhancers[,2:4])
visualise(WT_PE_OUT)
```

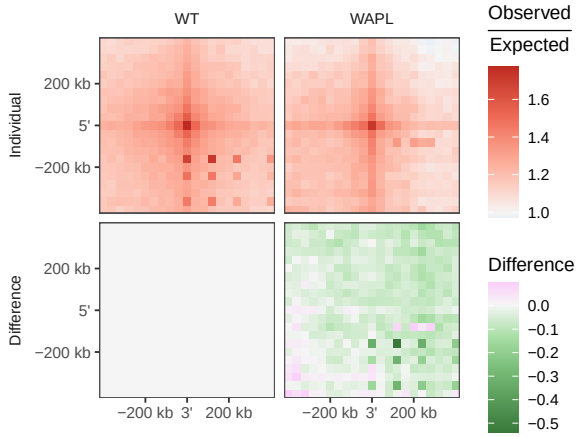


Figure 34: PE-SCAn
There is a pairwise enrichment of contacts between Superenhancers, compared to shifted regions in the WT.

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

```
persp(WT_PE_OUT, border = NA,
      cex.axis = 0.6, cex.lab = 0.6)
```

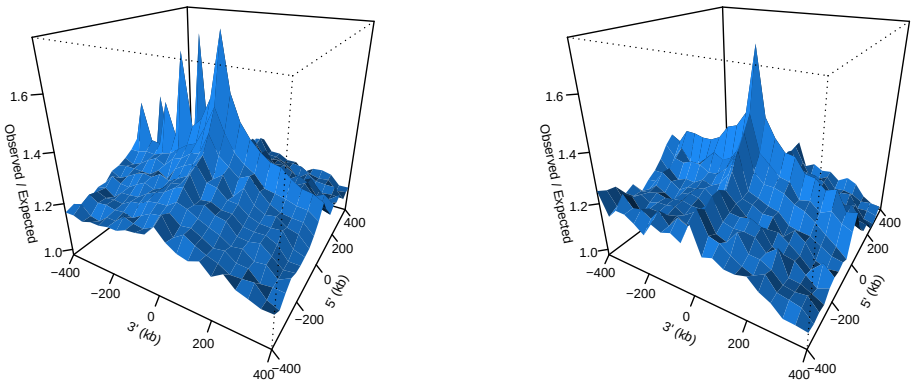


Figure 35: PE-SCAn perspective plot

centromere.telomere.analysis

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)
```

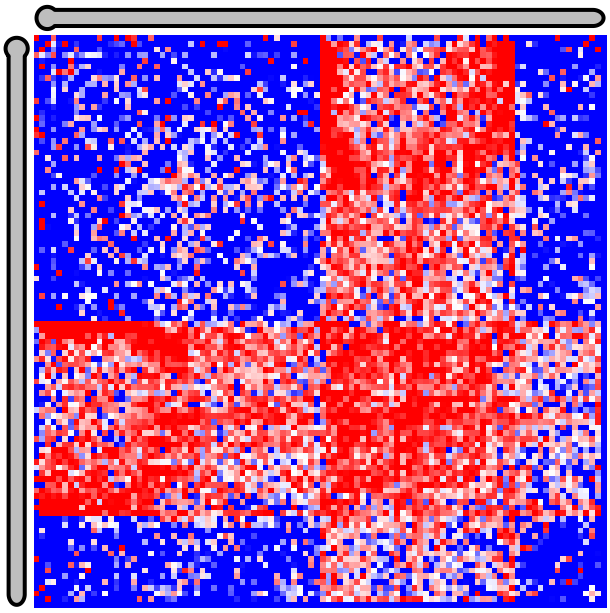


Figure 36: Centromere-telomere plot of chromosomes 15 and 19

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

Session info

```
#> R version 3.6.3 (2020-02-29)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 16.04.7 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] patchwork_1.1.0  GENOVA_1.0.0     BiocStyle_2.19.1
#>
#> loaded via a namespace (and not attached):
#>  [1] SummarizedExperiment_1.16.1 tidyselect_1.1.0
#>  [3] xfun_0.19                   purrr_0.3.4
#>  [5] lattice_0.20-41             colorspace_2.0-0
#>  [7] vctrs_0.3.5                 generics_0.1.0
#>  [9] htmltools_0.5.0             stats4_3.6.3
#> [11] rtracklayer_1.46.0          yaml_2.2.1
#> [13] XML_3.99-0.3                rlang_0.4.8
#> [15] pillar_1.4.7                withr_2.3.0
#> [17] glue_1.4.2                  BiocParallel_1.20.1
#> [19] BiocGenerics_0.32.0         matrixStats_0.57.0
#> [21] GenomeInfoDbData_1.2.2      lifecycle_0.2.0
#> [23] stringr_1.4.0               zlibbioc_1.32.0
#> [25] Biostrings_2.54.0           munsell_0.5.0
#> [27] gtable_0.3.0                evaluate_0.14
#> [29] Biobase_2.46.0              labeling_0.3
#> [31] knitr_1.30                  IRanges_2.20.2
#> [33] GenomeInfoDb_1.22.1         parallel_3.6.3
#> [35] Rcpp_1.0.5                  scales_1.1.1
#> [37] BiocManager_1.30.10         DelayedArray_0.12.3
#> [39] S4Vectors_0.24.4            magick_2.5.2
#> [41] XVector_0.26.0              farver_2.0.3.9000
#> [43] Rsamtools_2.2.3             RhpcBLASctl_0.20-137
#> [45] ggplot2_3.3.2.9000          digest_0.6.27
#> [47] stringi_1.5.3               bookdown_0.21
#> [49] dplyr_1.0.2                 GenomicRanges_1.38.0
#> [51] grid_3.6.3                  tools_3.6.3
#> [53] bitops_1.0-6                magrittr_1.5
#> [55] RCurl_1.98-1.2              tibble_3.0.4
```

```
#> [57] crayon_1.3.4               tidyr_1.1.2
#> [59] pkgconfig_2.0.3            Matrix_1.3-2
#> [61] ellipsis_0.3.1             data.table_1.13.4
#> [63] rmarkdown_2.6              R6_2.5.0
#> [65] GenomicAlignments_1.22.1   compiler_3.6.3
```

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. 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): 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): 1665–80. https://doi.org/10.1016/j.cell.2014.11.021.

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.