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

2018-01-04

Contents

1 Loading data

```
library(GENOVA)
```

1.1 Index-based

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 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
```

1.3 Recommended resolutions

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

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	$\frac{width\ in\ bp\ of\ window}{500}$

1.4 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 and the data is normalised to the total sum of reads. You can also add centromere-information in the form of a three-column data.frame:

```
# 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
```

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(
                                signalPath = 'data/WT_10000_iced.matrix',
                                indicesPath = 'data/WT_10000_abs.bed',
                                name = "WT",
                                centromeres = centromeres,
                                color = "black",
                                comments = "This is an optional memo-field.")

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

Hap1_WT_40kb <- construct.experiment(
                                signalPath = 'data/WT_40000_iced.matrix',
                                indicesPath = 'data/WT_40000_abs.bed',
                                name = "WT",
                                centromeres = centromeres,
                                color = "black",
                                comments = "This is an optional memo-field.")

Hap1_WAPL_40kb <- construct.experiment(
                                signalPath = 'data/WAPL_40000_iced.matrix',
                                indicesPath = 'data/WAPL_40000_abs.bed',
```

```
                                    name = "WAPL",
                                    centromeres = centromeres,
                                    color = "red")

Hap1_WT_1MB <- construct.experiment(
                                    signalPath = 'data/WT_1000000_iced.matrix',
                                    indicesPath = 'data/WT_1000000_abs.bed',
                                    name = "WT",
                                    centromeres = centromeres,
                                    color = "black",
                                    comments = "This is an optional memo-field.")

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

In this object are several slots:

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

2 Genome-wide analyses

2.1 Cis-quantification

Another important quality-metric is the fraction of *cis*-interactions. Work by the group of Amons Tanay showed that the expected amount of intra-chromosomal contacts is 90-93% (Olivares-Chauvet et al. 2016). Assuming that any extra inter-chromosomal contacts are due to dubris/noise, the user might aspire to get the *cis*-percentages as close to 90% as possible.

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

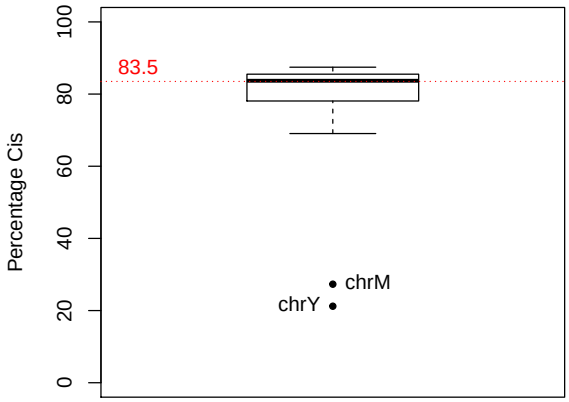

Figure 1: **Fraction of cis-contacts per chromosome**

Of course, one can also inspect the results per chromosome more closely:

```
plot(cisChrom_out$perChrom, las=2)
abline(h = cisChrom_out$genomeWide) # genome-wide percentage
```

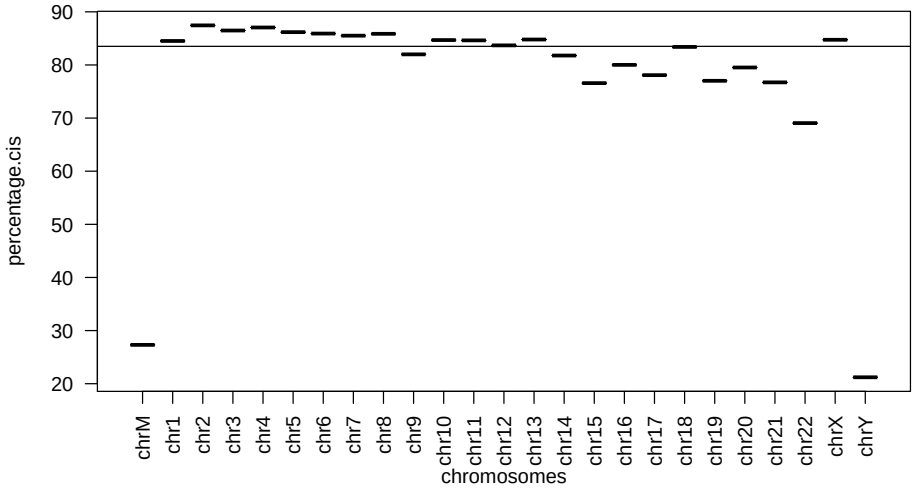

Figure 2: **Fraction of cis-contacts per chromosome**
Chromosomes 9, 15, 19 & 22 have translocations, which leads to the appearance of more trans-interactions than similar-sized chromosomes.

2.2 chromosome plots

To find possible translocations and/or flawed mapping, we can plot the genome-wide enrichment of interactions between all combinations of chromosomes. The values in the matrix are $log10(observed/expected)$.

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

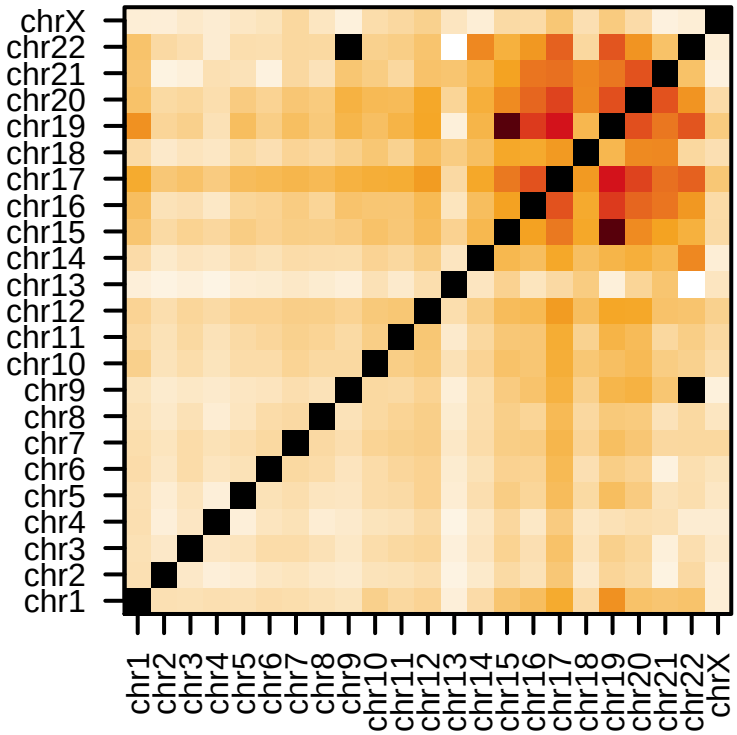

Figure 3: Chromosome matrix
The two known translocations of Hap1 cells are easily seen (15-19 & 9-22).

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.

```
RCP_out <- RCP(experimentList = list(Hap1_WT_40kb, Hap1_WAPL_40kb),
               chromsToUse = c('chr1','chr2','chr3'))
```

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.

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

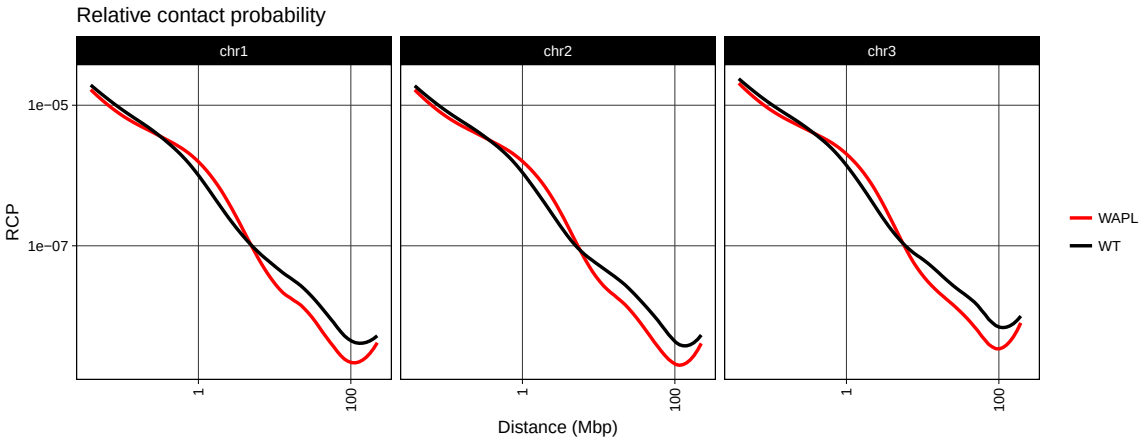

Figure 4: RCP
Every facet shows the RCP of one chromosome.

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

```
# Plot RCP: per-chromosome
visualise.RCP.ggplot(RCPdata = RCP_out,
                     smooth = T, # use a LOESS smoothing
                     combine = T) # Merge data from all chromosomes
```

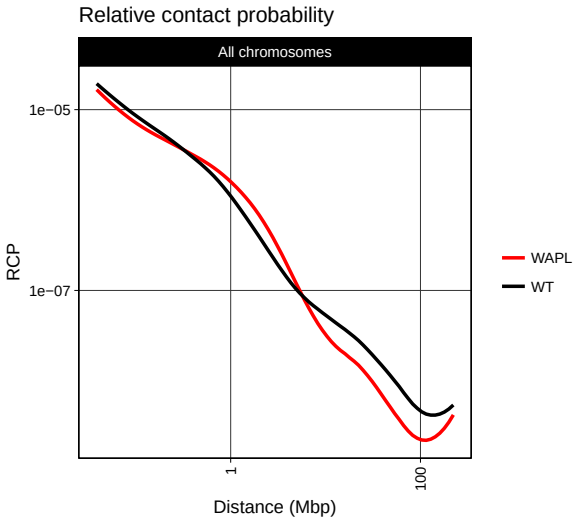

Figure 5: RCP
All data combined in one plot.

2.4 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.

```
par(pty="s")
# lets use a 5Mb-region on chromosome seven.
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               cut.off = 50) # upper limit of contacts
```

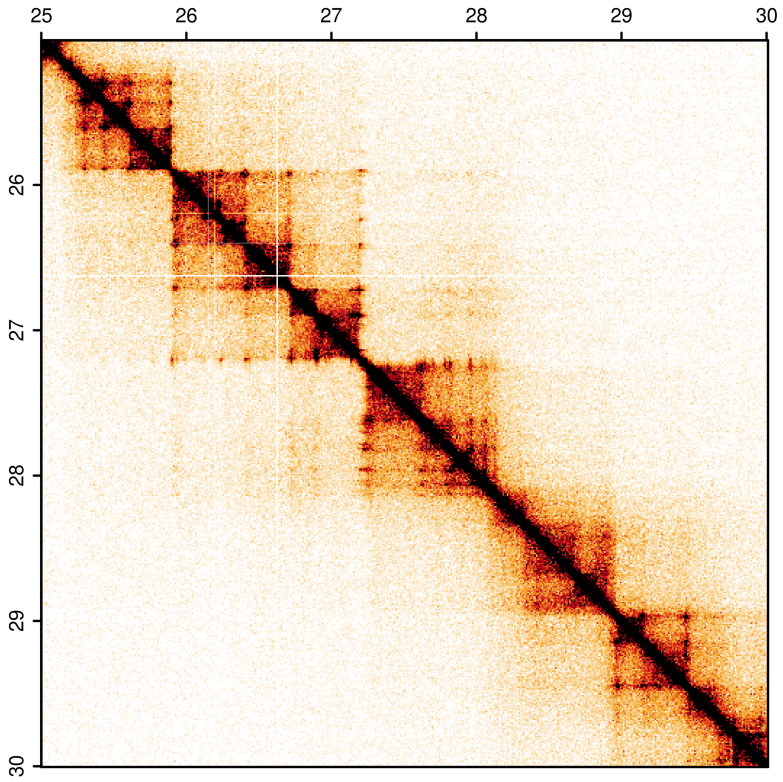

Figure 6: Hi-C matrixplot
Simplest example: one experiment, no annotation

2.4.1 two experiments

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

```
par(pty="s")
# lets use a 5Mb-region on chromosome seven.
hic.matrixplot(exp1 = Hap1_WT_10kb,
               exp2 = Hap1_WAPL_10kb,
               coplot = 'dual',
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               cut.off = 50) # upper limit of contacts
```

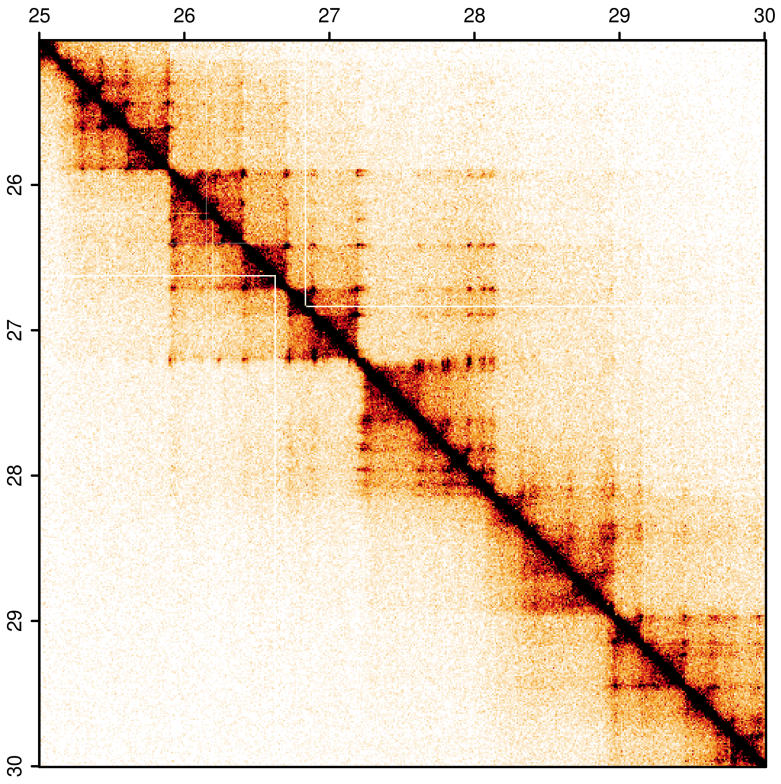

Figure 7: Hi-C matrixplot: dual
The extended loops in the WAPL knockout are easily seen at around 28Mb in the upper triangle.

```
par(pty="s")
# lets use a 5Mb-region on chromosome seven.
```

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

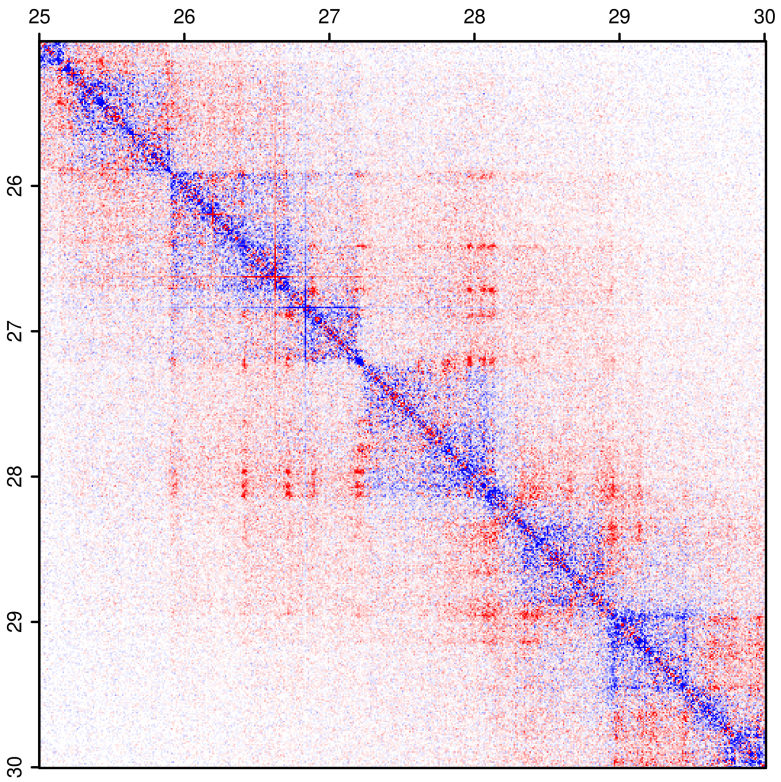

Figure 8: Hi-C matrixplot: diff
Note the ease of identifying the extended loops.

2.4.2 TADs and loops

Lets load some TAD- and loop-annotations:

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

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

```
par(pty="s")
# lets use a 5Mb-region on chromosome seven.
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6,
               loops = WT_Loops, # see APA
               loops.color = '#998ec3', # purple loops
               loops.type = 'upper', # only plot in upper triangle
               loops.resize = 20e3, # expand for visibility
               tads = WT_TADs, # see ATA
               tad.type = 'lower', # only plot in lower triangle
               tads.color = '#91cf60', # green TAD-borders
               cut.off = 25) # upper limit of contacts
```

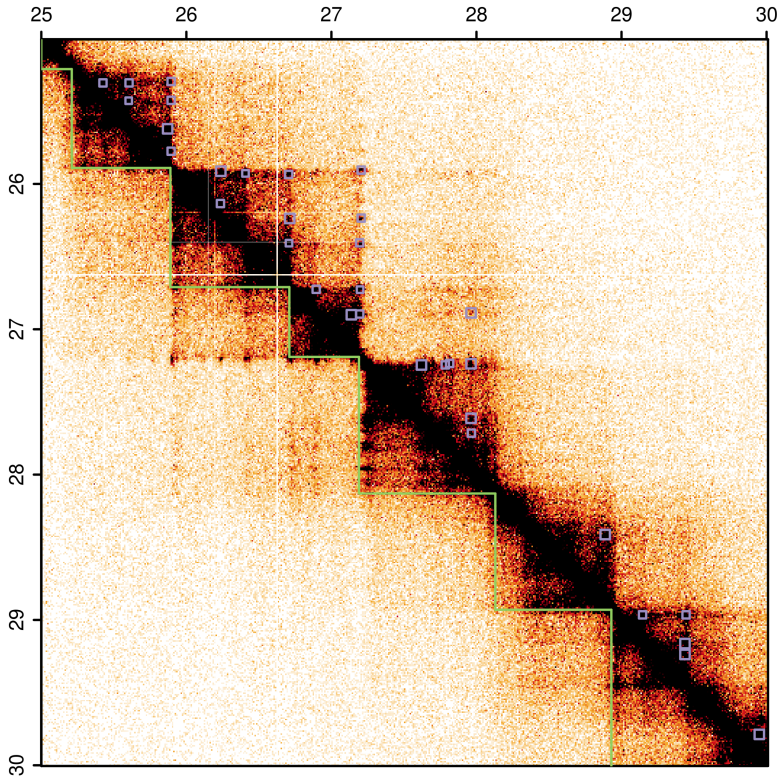

Figure 9: Hi-C matrixplot: TAD- and loop-annotations

2.4.3 BigWigs and BEDs

Two tracks above and two tracks to the left can be added. These can be either BED-like data.frames or the paths the .bw files. For example, lets load a BED6-file (chrom, start, end, name, score, and strand) of CTCF-motifs under CTCF-ChIP peaks.

```
CTCF = read.delim('/DATA/oidBackup/WAPL_Project/Hi-C/analysis/selected_regions/CTCF_WT_motifs.bed', h = F)
```

Table 2: 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	-
chr1	793463	793482	CCACCTGCTGGTGGCAGTG	1177	+
chr1	805297	805316	CTGCCACCAGGGGGCGCGC	2073	+

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:

```
library(devtools)
install_github(repo ='bigwrig', username =  'jayhesselberth')
```

```
# lets use a 5Mb-region on chromosome seven.
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
               start = 25e6,
               end=30e6, chip = list(CTCF,
                                     NULL,
                                     NULL, # outer-left
                                     'data/SMC1_WT.bw' ), # inner-left
               cut.off = 25) # upper limit of contacts
```

2.4.4 Genes

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

```
# Human genes (GRCh37.p13)
# Filters:
## With RefSeq mRNA ID(s): Only
# Attributes:
## Gene stable ID
## Transcript stable ID
## Chromosome/scaffold name
## Transcript start (bp)
## Transcript end (bp)
```

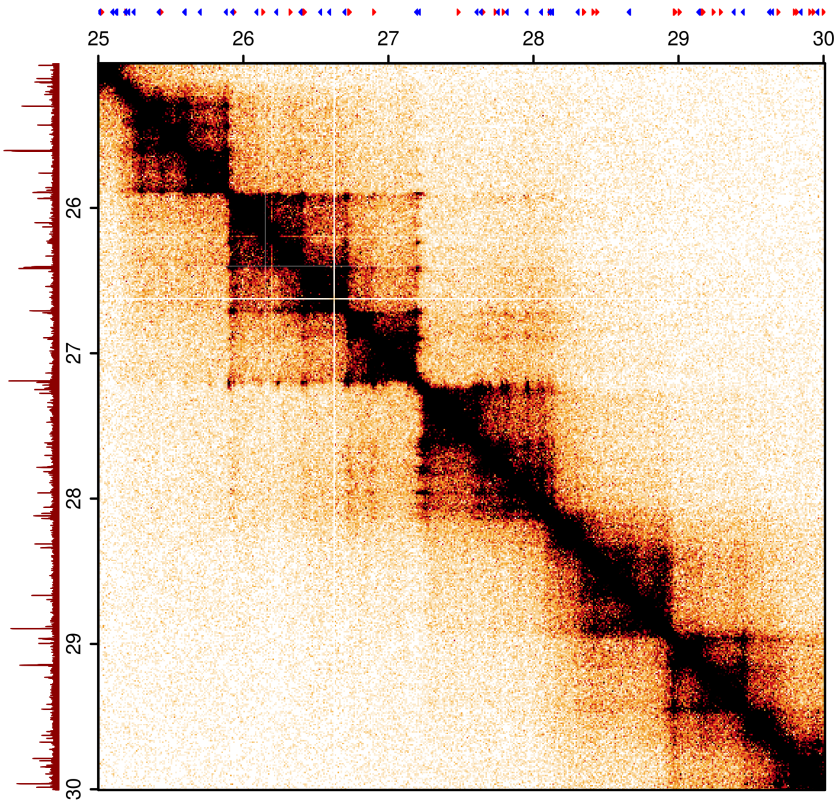

Figure 10: Hi-C matrixplot: ChIPseq
A BED-file of CTCF-sites is plotted at the top and a coverage-track of SMC1 ChIP-seq is plotted to the left.

```
## 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 +/-
```

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

```
# lets use a 5Mb-region on chromosome seven.
hic.matrixplot(exp1 = Hap1_WT_10kb,
               chrom = 'chr7',
```

Table 3: A data.frame holding the needed columns

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	+

```
                    start = 25e6,
                    end=30e6,
                    genes = martExport,
                    chip = list(CTCF,
                                NULL,
                                NULL, # outer-left
                                NULL ), # inner-left
                    cut.off = 25) # upper limit of contacts
## NULL
```

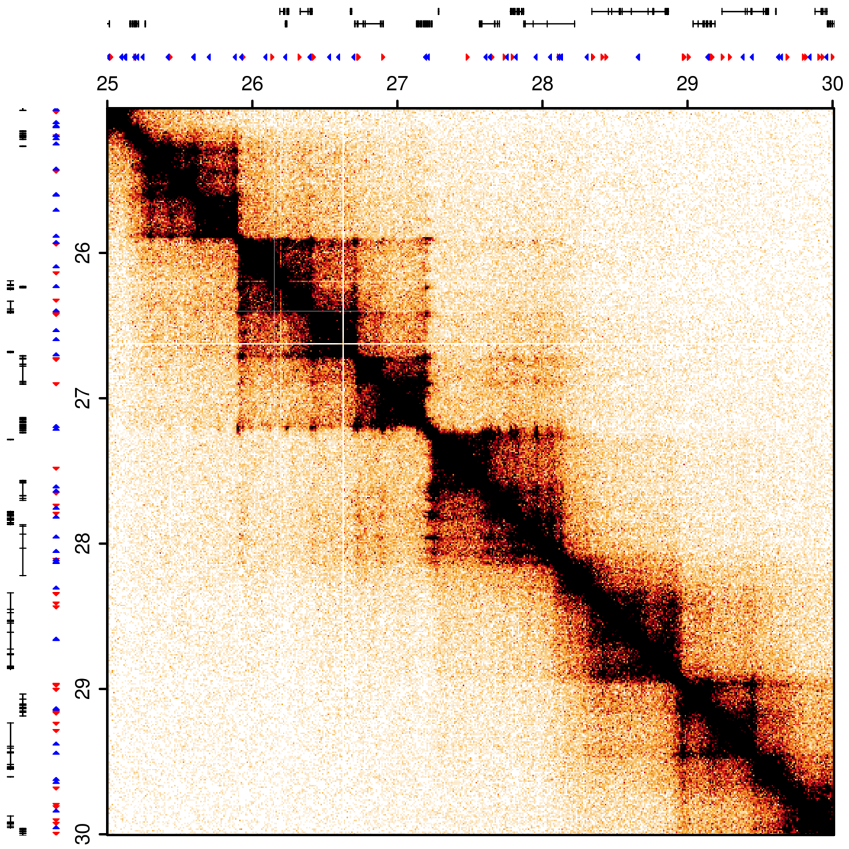

Figure 11: Hi-C matrixplot: ChIPseq and genes
A BED-file of CTCF-sites is plotted below the genes.

3 TADs

GENOVA has a large repetoire of functions to analyse TADs. We use the TAD-calls of WT Hap1 20-kb matrices from Haarhuis et al. (2017), generated with HiCseg (Lévy-Leduc et al. 2014).

```
# Lets load in some TAD-annotations from HiC-seg
WT_TADs = read.delim('data/WT_hicseg_TADs.bed', h = F)
```

Table 4: A data.frame holding a standard BED3 format

V1	V2	V3
chr1	50000	120000
chr1	120000	210000
chr1	210000	240000
chr1	240000	610000
chr1	610000	900000

3.1 ATA

```
ATA_Hap1_WT   <- ATA(experiment = Hap1_WT_10kb,
                     tad.bed = WT_TADs)
ATA_Hap1_WAPL <-  ATA(experiment = Hap1_WAPL_10kb,
                      tad.bed = WT_TADs)
```

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 from HiCseg",
                     zlim1 = c(0,75),
                     zlim2 = c(-5,5),
                     focus = 1) # which entry to use as comparison
```

3.2 TAD+N

```
TAD_N_WT   <- intra.inter.TAD.contacts(TAD = WT_TADs,
                                       max.neighbor = 10,
                                       exp = Hap1_WT_40kb)
TAD_N_WAPL <- intra.inter.TAD.contacts(TAD = WT_TADs,
                                       max.neighbor = 10,
                                       exp = Hap1_WAPL_40kb)
```

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

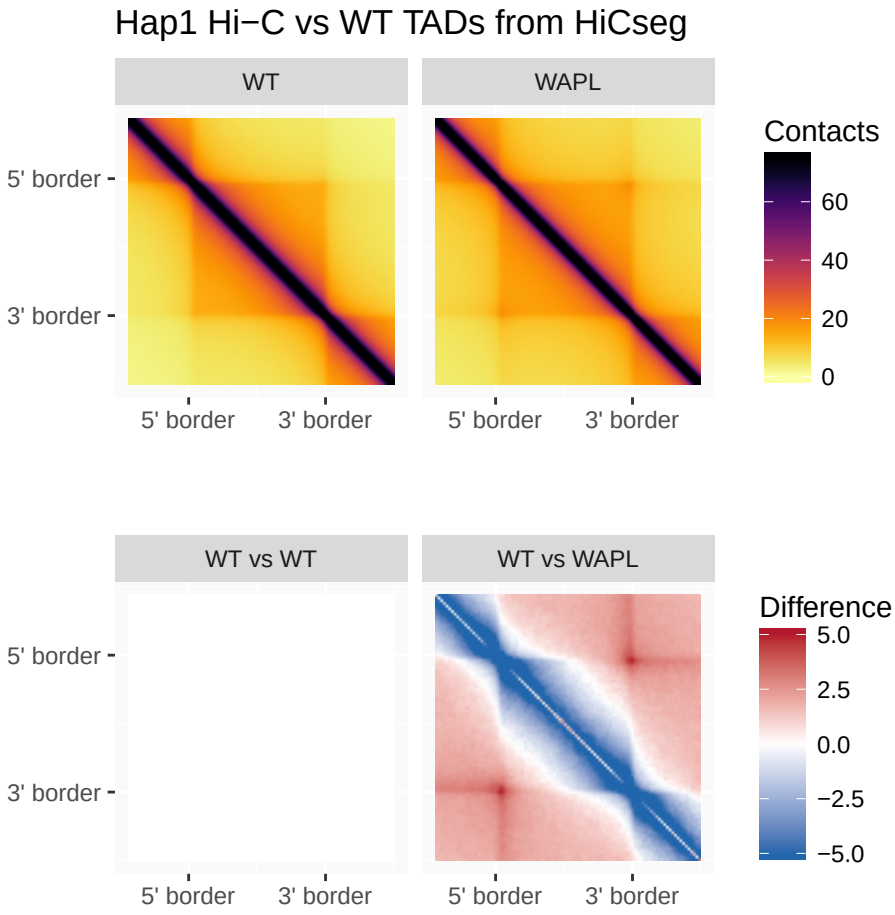

Figure 12: ATA
In the WAPL-knockout, we see a decrease of contacts within the TAD, but an increase at the corner.

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

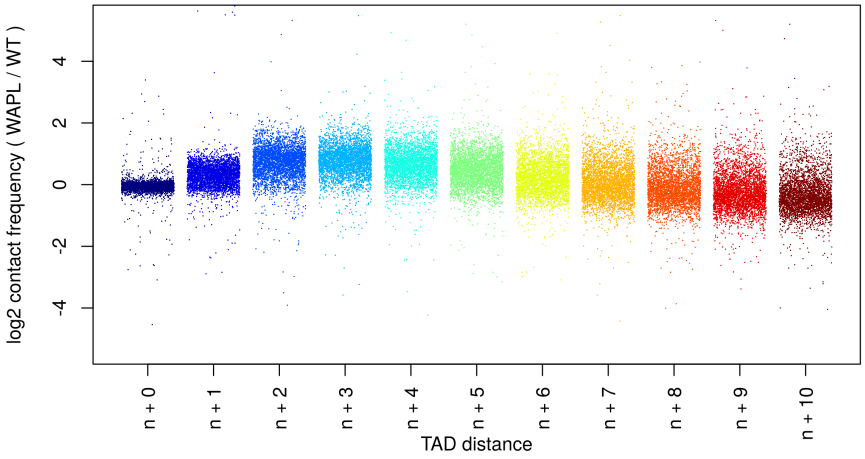

Figure 13: Differential TAD-analysis
Experiment 2 (WAPL) has more interactions between neighbouring TADs

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

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

V1	V2	V3	V4	V5	V6
chr11	875000	9e+05	chr11	2020000	2025000
chr11	875000	9e+05	chr11	2162500	2187500
chr11	875000	9e+05	chr11	2020000	2025000
chr11	875000	9e+05	chr11	2020000	2030000
chr11	875000	9e+05	chr11	1940000	1945000

4.1 APA

Explain smalltreshold

```
APA_Hap1_WT   <- APA(experiment = Hap1_WT_10kb,
                     smallTreshold = 300e3,
                     loop.bed = WT_Loops)
APA_Hap1_WAPL <- APA(experiment = Hap1_WAPL_10kb,
                     smallTreshold = 300e3,
                     loop.bed = WT_Loops)
```

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

```
visualise.APA.ggplot(APAlist = list('WT' = APA_Hap1_WT,
                                    'WAPL' = APA_Hap1_WAPL), # a named list
                     title = "Hap1 Hi-C vs extended loops",
                     zTop = c(1,12), # set the zlims of the upper row
                     zBottom = c(-8.33,8.33),
                     focus = 1) # which item in APAlist to use as comparison
```

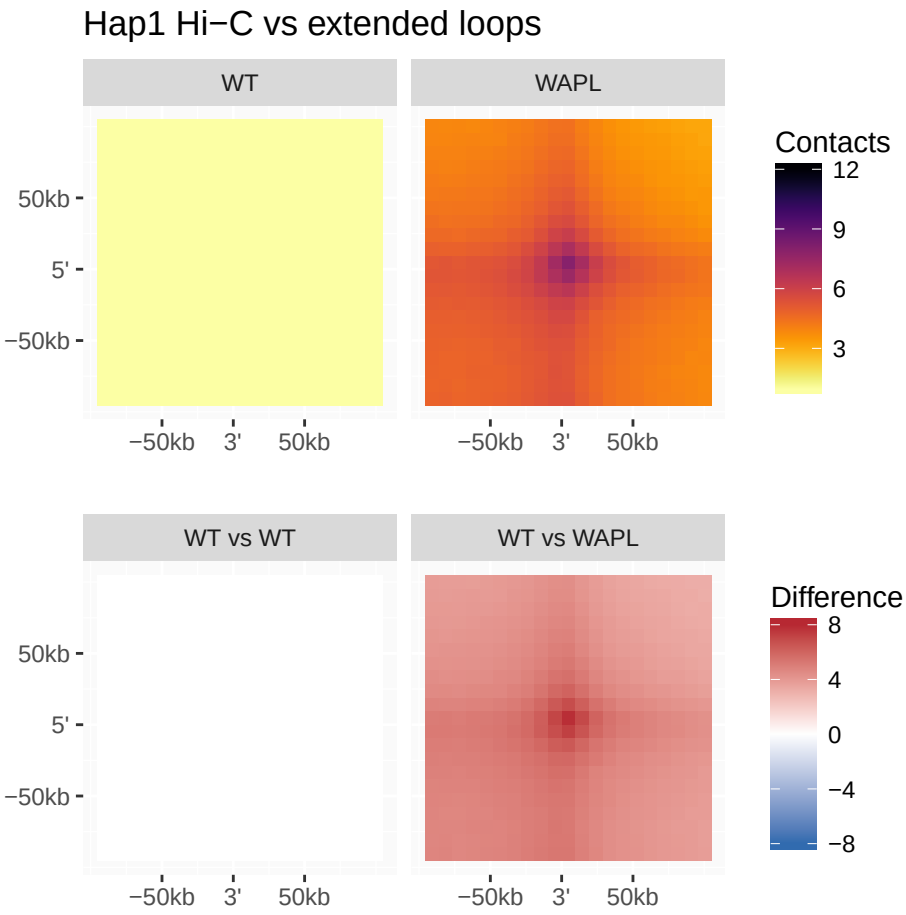

Figure 14: APA
In the WAPL-knockout, we see an increase of contacts at the loop.

4.2 PE-SCAn

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

```
WT_PE_OUT = PESCAn(exp = Hap1_WT_40kb, bed = superEnhancers[,2:4])
WAPL_PE_OUT = PESCAn(exp = Hap1_WAPL_40kb, bed = superEnhancers[,2:4])
visualise.PESCAn.ggplot(list(WT = WT_PE_OUT, WaplKO = WAPL_PE_OUT), resolution = 40e3, smooth = F)
```

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

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

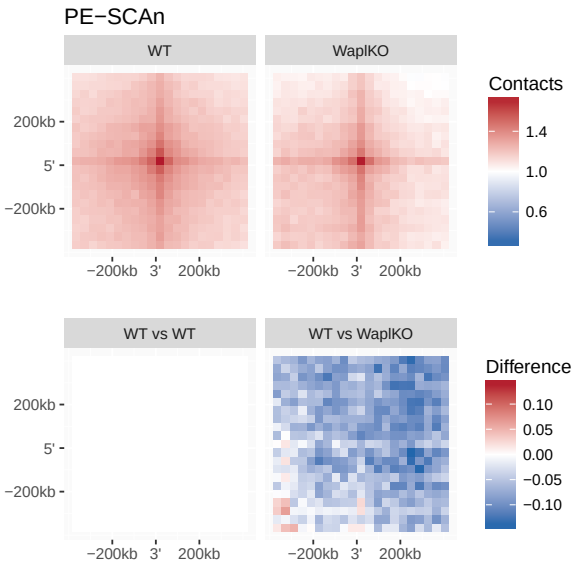

Figure 15: PE-SCAn
There is a pairwise enrichment of contacts between Superenhancers, compared to shifted regions in both the WT and Wapl samples.

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

```
par(mfrow = c(1,2))
# get shared z-min and -max values
SHARED_Z = c(min(c(WT_PE_OUT, WAPL_PE_OUT)),
             max(c(WT_PE_OUT, WAPL_PE_OUT)))
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 = 60, # rotation
      col="#92c5de", # color of the surface
      shade=0.4, # how much shading
      xlab="",
      ylab="",
      zlab="",
      ticktype="detailed",
      main="WT",
```

```
        border=NA,
        zlim = SHARED_Z)

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 = WAPL_PE_OUT), # PE-SCAn out
      phi = 25, # colatitude
      theta = 60, # rotation
      col="#92c5de", # color of the surface
      shade=0.4, # how much shading
      xlab="",
      ylab="",
      zlab="",
      ticktype="detailed",
      main="WAPL",
      border=NA,
      zlim = SHARED_Z)
```

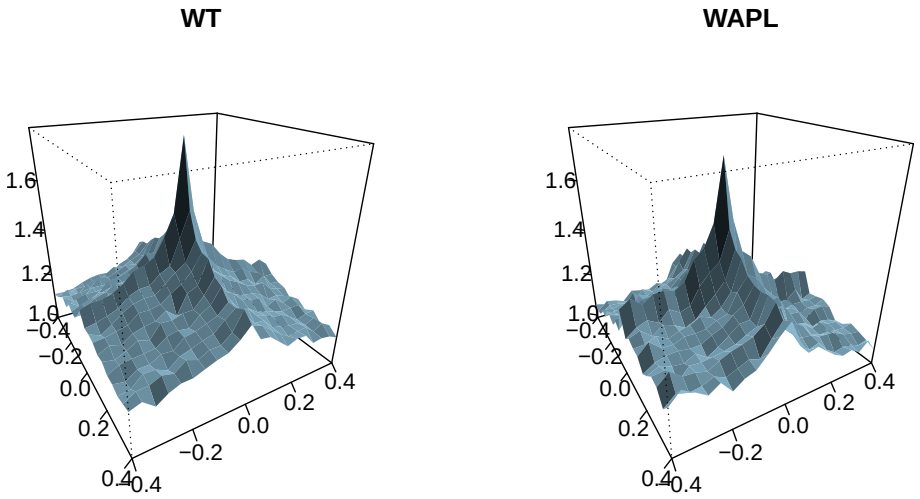

Figure 16: PE-SCAn perspective plots

5 To-do

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

- write `visualise.PESCAn.persp`

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

6 Session info

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/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] bigwrig_0.1.0     ggplot2_2.2.1     viridis_0.4.0     viridisLite_0.2.0
## [5] bindrcpp_0.2      GENOVA_0.9.5      BiocStyle_2.6.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.14           pillar_1.0.1           compiler_3.4.3
##  [4] GenomeInfoDb_1.14.0    plyr_1.8.4             XVector_0.18.0
##  [7] bindr_0.1              bitops_1.0-6           tools_3.4.3
## [10] zlibbioc_1.24.0        digest_0.6.12          evaluate_0.10.1
## [13] tibble_1.4.1           gtable_0.2.0           pkgconfig_2.0.1
## [16] rlang_0.1.6            yaml_2.1.16            parallel_3.4.3
## [19] gridExtra_2.3          GenomeInfoDbData_1.0.0 dplyr_0.7.4
## [22] stringr_1.2.0          knitr_1.18             S4Vectors_0.16.0
## [25] IRanges_2.12.0         stats4_3.4.3           rprojroot_1.3-1
## [28] grid_3.4.3             glue_1.2.0             data.table_1.10.4-3
## [31] R6_2.2.2               rmarkdown_1.8.5        bookdown_0.5
## [34] reshape2_1.4.2         magrittr_1.5           codetools_0.2-15
## [37] backports_1.1.2        scales_0.4.1           htmltools_0.3.6
## [40] BiocGenerics_0.24.0    GenomicRanges_1.30.1   assertthat_0.2.0
## [43] colorspace_1.3-2       labeling_0.3           stringi_1.1.5
## [46] RCurl_1.95-4.9         lazyeval_0.2.0         munsell_0.4.3
```

References

Haarhuis, Judith H.I., Robin H. van der Weide, Vincent A Blomen, J Omar Yáñez-Cuna, Mario Amendola, Marjon S. van Ruiten, Peter H.L. Krijger, et al. 2017. "The Cohesin Release Factor WAPL Restricts Chromatin Loop Extension." *Cell* 169 (4): 693–707.e14. doi:10.1016/j.cell.2017.04.013.

Lévy-Leduc, Celine, M. Delattre, T. Mary-Huard, and S. Robin. 2014. "Two-dimensional segmentation for analyzing Hi-C data." In *Bioinformatics*. Vol. 30. 17. doi:10.1093/bioinformatics/btu443.

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. doi: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. doi:10.1038/nature20158.

Rao, Suhas S P, Miriam H Huntley, Neva C Durand, and Elena K Stamenova. 2014. "A 3D Map of the Human Genome at Kilobase Resolution Reveals Principles of Chromatin Looping." *Cell* 159 (7). Elsevier Inc.: 1665–80. doi:10.1016/j.cell.2014.11.021.

Sanborn, Adrian L, Suhas S P Rao, Su-Chen Huang, Neva C Durand, Miriam H Huntley, Andrew I Jewett, Ivan D Bochkov, et al. 2015. "Chromatin extrusion explains key features of loop and domain formation in wild-type and engineered genomes." *Proceedings of the National Academy of Sciences*. doi:10.1073/pnas.1518552112.

Servant, Nicolas, Nelle Varoquaux, Bryan R. Lajoie, Eric Viara, Chong-Jian Chen, Jean-Philippe Vert, Edith Heard, Job Dekker, and Emmanuel Barillot. 2015. "HiC-Pro: an optimized and flexible pipeline for Hi-C data processing." *Genome Biology* 16 (1): 259. doi:10.1186/s13059-015-0831-x.