Code explanation for Chahar_etal_analysis

This document gives an overview and explanation of the pipelines used in this study, and points to the custom scripts that were used. Some of these scripts used parallel processing with SLURM, and the loader scripts will not be immediately compatible with most systems. These scripts are therefore intended as a foundation for you to set up your own analyses, or to (hopefully) comprehend how we generated our results, rather than something you can "plug and play" into your own data. Commands directly written in the command line are given after the ">" prompt and in `Courrier New` font; variable arguments are enclosed in [square brackets].

1. Hi-C processing

Dependencies:

- bowtie2, with index files for the required genome assembly downloaded
- samtools
- FAN-C (https://github.com/vaquerizaslab/fanc)
- R (>=3.2)

The paired fastq files are first mapped to the reference genome with bowtie2, separately, then sorted by name with samtools.

```
> bowtie2 -x [BOWTIE2_INDEX_PATH] -U [READ1.fastq.gz] -t -p 24 -S [READ1.sam]
> samtools view -b [READ1.sam] > [READ1.bam]
> rm [READ1.sam]
> samtools sort -n -o [READ1_sorted.bam] [READ1.bam]
> rm [READ1.bam]
> bowtie2 -x [BOWTIE2_INDEX_PATH] -U [READ2.fastq.gz] -t -p 24 -S [READ2.sam]
> samtools view -b [READ2.sam] > [READ2.bam]
> rm [READ2.sam]
> samtools sort -n -o [READ2_sorted.bam] [READ2.bam]
> rm [READ2.bam]
```

These reads are then mated to pairs with FAN-C, keeping only pairs with both ends uniquely mapping, and attributes them to restriction fragment space. These pairs are further filtered to remove self-ligated fragments, reads considered too far (>= 10 kb) from a restriction site, or those likely to be PCR duplicates (within 2 nt). Finally, the pairs are converted to the starting Hi-C matrix.

```
> fanc pairs -g [FRAGMENT_BED] -t 32 -m -us -S [READ1_sorted.bam] [READ2_sorted.bam] [PAIRS.pairs]
> fanc pairs -l -d 10000 -p 2 [PAIRS.pairs]
> fanc hic [PAIRS.pairs] [HIC.hic]
```

The restriction fragment space is determined by *in silico* digestion of the reference genome with the appropriate restriction fragment, generating a bed file of the fragment coordinates. This can be made using FAN-C, starting from the fasta file of the genome sequence:

```
> fanc fragments [FASTA.fa] [ENZYME] [FRAGMENT_BED]
```

To pool data, the "raw" Hi-C maps for each replicate are first merged, and matrix normalization is only applied to the final dataset, at the set bin size (10 kb in our case).

```
> fanc hic [REP1.hic] [REP2.hic] … [REPn.hic] [COMBINED.hic]
> fanc hic -n -b 10000 [COMBINED.hic] [COMBINED_10kb.hic]
```

In this study, we pooled Hi-C datasets using different restriction enzymes, HindIII and DpnII. Because they are mapped to different restriction fragment spaces, the "raw" matrices cannot be merged. Instead, the 10 kb matrices need to be created, merged and re-normalized:

```
> fanc hic [REP1_DpnII.hic] [REP2_DpnII.hic] [DpnII.hic]
> fanc hic [REP1_HindIII.hic] [REP2_HindIII.hic] [HindIII.hic]
> fanc hic -b 10000 -n [DpnII.hic] [DpnII_10kb.hic]
> fanc hic -b 10000 -n [HindIII.hic] [HindIII_10kb.hic]
> fanc hic [DpnII_10kb.hic] [HindIII_10kb.hic] [Combined.hic]
> fanc hic -b 10000 -n [Combined.hic] [Combined_10kb.hic]
```

To visualize specific Hi-C regions in the same manner as was done for the Capture Hi-C regions, the relevant sub-matrix is first extracted with FAN-C:

```
> fanc dump -s [CHR]:[START]-[END]--[CHR]:[START]-[END] -S [HIC]
[SUBMATRIX.mat] [REGIONS.txt]
```

This generates an *n* x *n* matrix, SUBMATRIX.mat, of the Hi-C interaction scores, and a regions file, REGIONS.txt, of *n* rows, giving the chromosome and coordinates for each row/column of the submatrix.

For visualization using the same tools as for the Capture Hi-C data, these data are reformatted using the script fanc2cap.r:

```
> Rscript fanc2cap.r [CHR] [ID] [SUBMATRIX.mat] [REGIONS.txt]
[OUTFILE]
```

2. Capture Hi-C processing

Dependencies:

- perl
- R (>=3.2)
- bowtie, with index files for the required genome assembly downloaded

The first steps of the pipeline are operationally very similar for FAN-C analysis, but have been maintained with these custom scripts to allow downstream filtering for captured sequences. The

scripts for parallel processing of the files have been included, but the basic scripts which these call are emphasized for development within your own infrastructures.

Fastq files are first split to make parallel jobs, and then truncated to the first instance of a Hi-C-ligated DpnII site (GATC_GATC):

```
> split -a 4 -d -l 2000000 [R1.fastq] [SPLIT_DIR]/R1.fastq
> split -a 4 -d -l 2000000 [R2.fastq] [SPLIT_DIR]/R2.fastq
> submit_trunc.pl [SPLIT_DIR] [TRUNC_DIR] [QSUB_DIR] GATC_GATC
[DATASET] [MAXJOBS.FN] [WORKING_DIR] 400 [STATS_DIR]
```

This last command is a wrapper script for the implementation (using attached submit_batch.r utility script) of the script, trunc.pl, which performs the truncation. Another script, stats_truncation.pl, is finally called to determine how many reads were truncated, and the average truncation size.

Next, the truncated fastq files are mapped to the appropriate genome assembly with bowtie:

```
> perl submit_bowtie.pl [TRUNC_DIR] [BOWTIE_INDEX_PATH] [MAPPED_DIR]
[QSUB_DIR] [STATS_DIR] [DATASET] [MAXJOBS.FN] 3000 [WORKING_DIR] L
400
```

The wrapper script similarly distributes bowtie mapping jobs, then derives the mapping statistics with stats_mapping.pl.

Mapped reads are mated to pairs:

```
> Rscript make_unit_file [MAPPED_DIR] NA 1 2
> perl submit_unite_paired_bowtie_mapped_reads.pl [MAPPED_DIR]
[NODEDUP_DIR] T [QSUB_DIR] NA [DATASET] [MAXJOBS.FN] [WORKING_DIR] L
```

The wrapper scripts determine the pairs of files which need to be mated each time, then distributes them to their mating with bowtie_prepare_pairs.pl.

The paired files are filtered for PCR duplicates:

```
> perl deduplicator.pl [NODEDUP_DIR] [STATS_DIR] [PAIRED_DIR] 19
```

The deduplication statistics are output. Note that the last argument (19) is for the name of the last somatic chromosome (so chr19 for mouse; chr22 for human).

The paired files are then mapped to restriction fragment space, filtered for Hi-C artefacts (as above), and then further split based on whether none, one or both of the paired reads contain captured sequence (the latter, which we term the “P2 table” is the required input for TAD Capture Hi-C experiments).

```
> Rscript distrib_coord2fend_CapC.r [DATASET] [PAIRED_DIR]
[PROBE_FENDTABLE] [READ_LENGTH] 2000 500 [SPLIT_DIR] [MAT_DIR]
[QSUB_DIR] [WORKING_DIR] 30000
```

This wrapper script first uses the utility script distrib.r to map each individual paired file to restriction fragment space and filter out potential artefacts, using the script coords2fends.pl, and outputting each file to the SPLIT_DIR. These are then merged into the final file with merge_mat_files2.pl. Finally, this table is split using merge_mat_probe_files.pl to the different captured subtypes. Appropriate mapping requires a table of all the restriction fragment ends, further annotated with whether each fragment contains a capture probe or not. For this study, we used DpnII fragments on the mm10 genome, with a specific capture library (see accompanying article for the capture design). This file is too large to upload to Github, but is available on request: sexton@igbmc.fr

The P2 table is converted to the 5 kb (or 2 kb) fixed bin table:

```
> perl observed_cbin_counts.pl [5kb_BINNED] [5kb_BINS] [P2_TABLE] 0 [5kb_TABLE]
```

The BINNED and BINS files are lookup tables for which 5 kb bin each restriction fragment corresponds to. The files for this study (mm10_DpnII_5kb and 2kb) are available on request: sexton@igbmc.fr

The relevant sub-matrices are extracted separately, and balanced with Knight-Ruiz normalization:

```
> Rscript normalise_submatrices.r [TADTABLE] [5kb_TABLE] [5kb_BINS] [NAME] [FILE_PREFIX]
```

The TADTABLE file (TAD_mm10.table in this study) gives the coordinates for each different captured region. The output file of most importance is [FILE_PREFIX]_all_normalised.txt, which is the input file for visualization and subsequent analyses.

3. Visualization of (Cap)-HiC matrices

Dependencies:

- R (>=3.2), with the following packages:
  - gplots
  - rtracklayer

Different functions are contained within the R script, TADcaptureplots.r. The major function of interest is *tri*, which plots the triangular interaction matrix, and is set up with the following arguments:

- tab; the data.frame of the input data (the outputs of fan2capc.r or normalise_submatrices.r).
- id; stipulates which submatrix to plot, matches one of the names in TADTABLE, and is one of the entries on the “ID” column of tab.
- vmax; the dynamic range of the interaction scores plotted goes from 0 to vmax (or when diff set to TRUE, from -vmax to vmax).
- main; the title of the plot.
- x; default to FALSE, argument for whether the x-axis (chromosome coordinates) is plotted.
- rev; default to FALSE. When set to TRUE, makes the plot upside-down (useful for plotting two interaction maps together).

- diff; default to FALSE. When set to TRUE, the colour scheme is altered for plotting differentials/log-ratios.

Other functions are present for plotting gene and epigenomic tracks (as imported bigWig files), just like in ChiCMaxima (https://github.com/yousra291987/ChiCMaxima) or 4See (https://github.com/TomSexton00/4See).

4. Computing insulation

Dependencies:

- FAN-C
- R (>=3.2), with the package psych.

For Hi-C plots, insulation scores and TAD boundaries are computed directly with the FAN-C functions *insulation* and *boundaries*, respectively. For the Capture Hi-C submatrices, the functions are contained within the R script, insulation.r. The major function of interest is *inspipe*, with the following arguments:

- ifn; the path to the "all_normalised.txt" file, resulting from normalise_submatrices.r.
- ofn; the output file for the insulation scores.
- w; the window size, in bins, for computation of insulation scores.