ChIA-PIPE output file explation

1. linker filter

need to know the linker
and get the fastq file of without linker sequence
linker A and linker B
remove (AA/ BB/AB) from sequence reads and get the "no linker" reads for downstream analysis
and during the process of linker filter, maybe need to keep the min length of reads fot the reference genome mapping steps
Linkers are first trimmed for each pair of reads in different ways according to different ChIA-PET protocols. for ChIA-PIPE, they would keep the min length of reads for mapping.

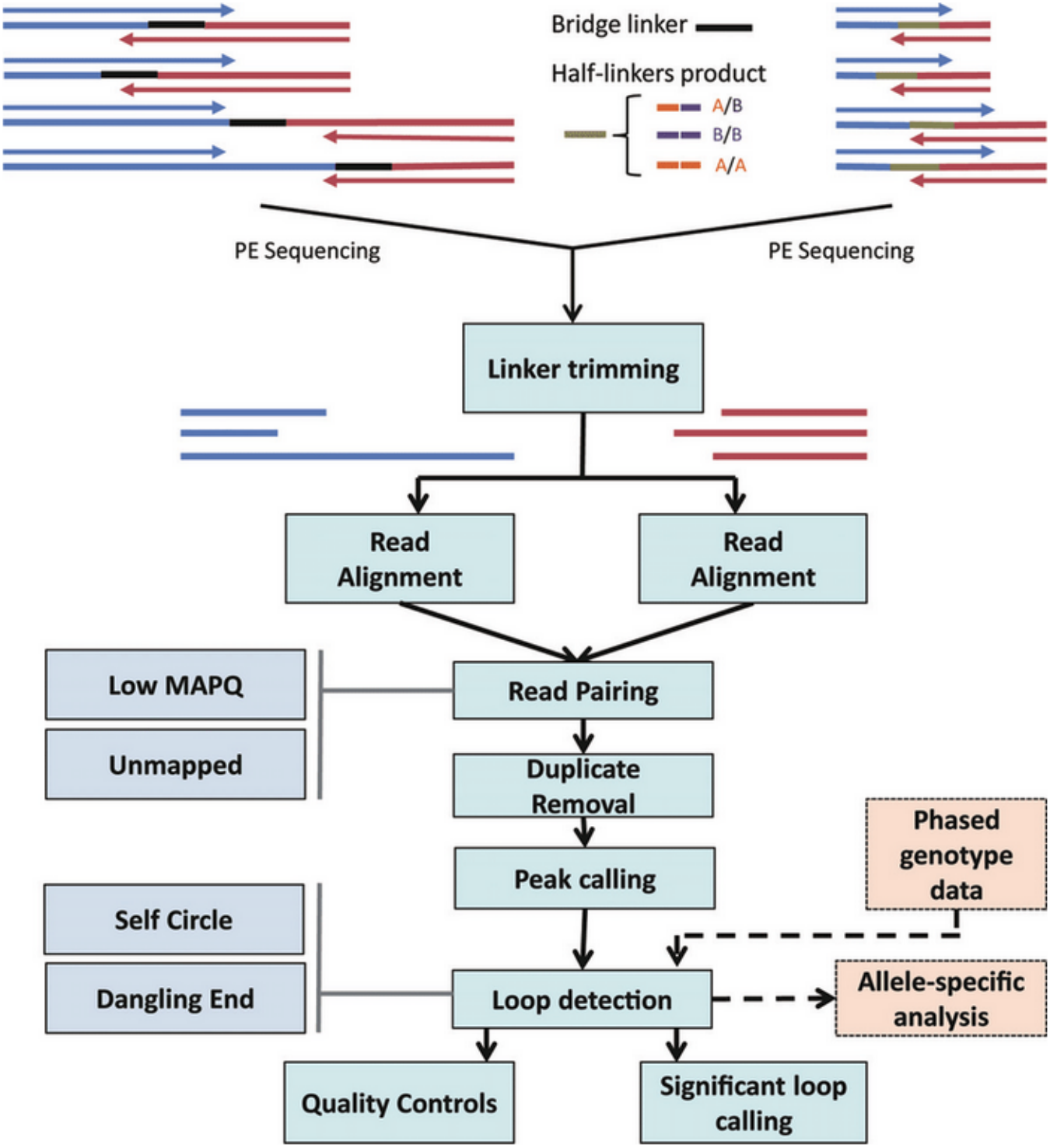

figure 1.
ChIA-PET2 workflow.Only read pairs that satisfy certain user-defined conditions are kept.

2a. Mapping "No linker" to Reference genome(ChIA-pipe , the minium MQ=30)

tag="none"
mapping results is sam file and compressed sam file

```
${main_prog} memaln -T ${map_qual} -t ${n_thread} ${fasta} \
    ${run}.${tag_name}.fastq.gz 1> ${run}.${tag_name}.sam 2>> ${log_file}
pigz -p ${n_thread} ${run}.${tag_name}.sam
```

pairing the tags

```
${main_prog} pair -S -q ${map_qual} -t ${n_thread} -s ${self_bp} \
   ${run}.${tag_name}.sam.gz \
   1>${run}.${tag_name}.stat.xls 2>> ${log_file}
```

Span computation (this part there are more one file with UU)

```
${main_prog} span -g -t ${n_thread} -s ${self_bp} \
   ${run}.${tag_name}.${suffix}.bam 2>> ${log_file} \
   1>${run}.${tag_name}.${suffix}.span.xls
```

Duplicate the paired tags

```
# Tag deduplication
${main_prog} dedup -g -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.bam \
    1> ${run}.${tag_name}.${suffix}.dedup.lc 2>> ${log_file}
```

Non-redundant span computation

```
${main_prog} span -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.nr.bam \
    2>> ${log_file} 1>${run}.${tag_name}.${suffix}.nr.span.xls
```

note: for the paring and span are not required by non-loop reads.

2b. Map "linker, one tag" reads "this part are confict Reads"

mapping process and compress sam results

```
${main_prog} memaln -T ${map_qual} -t ${n_thread} ${fasta} \
    ${run}.${tag_name}.fastq.gz 1> ${run}.${tag_name}.sam 2>> ${log_file}
pigz -p ${n_thread} ${run}.${tag_name}.sam
```

Pair the tags

```
${main_prog} pair -S -q ${map_qual} -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.sam.gz \
```

```
    1>${run}.${tag_name}.stat.xls 2>> ${log_file}
```

Span computation

```
${main_prog} span -g -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.bam 2>> ${log_file} \
    1>${run}.${tag_name}.${suffix}.span.xls
```

Tag deduplication

```
${main_prog} dedup -g -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.bam \
    1> ${run}.${tag_name}.${suffix}.dedup.lc 2>> ${log_file}
```

Non-redundant span computation

```
${main_prog} span -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.nr.bam \
    2>> ${log_file} 1>${run}.${tag_name}.${suffix}.nr.span.xls
```

2c. Map "Linker, two tags" reads ## are really reads for loop analysis with cluster analysis

#-- perform hybrid bwa-mem and bwa-aln mapping,
#de-duplication, span computation, and tag clustering --#

Perform mapping using memaln (hybrid of bwa-mem and bwa-aln)

```
${main_prog} memaln -T ${map_qual} -t ${n_thread} ${fasta} \
    ${run}.${tag_name}.fastq.gz 1> ${run}.${tag_name}.sam 2>> ${log_file}
pigz -p ${n_thread} ${run}.${tag_name}.sam
```

Pairing

```
${main_prog} pair -S -q ${map_qual} -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.sam.gz \
    1>${run}.${tag_name}.stat.xls 2>> ${log_file}
```

Span computation

```
${main_prog} span -g -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.bam 2>> ${log_file} \
    1>${run}.${tag_name}.${suffix}.span.xls
```

tag deduplication

```
${main_prog} dedup -g -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.bam \
    1> ${run}.${tag_name}.${suffix}.dedup.lc 2>> ${log_file}
```

Non-redundant span computation

```
${main_prog} span -t ${n_thread} -s ${self_bp} \
    ${run}.${tag_name}.${suffix}.nr.bam \
    2>> ${log_file} 1>${run}.${tag_name}.${suffix}.nr.span.xls
```

Cluster tags using 500bp extension

```
${main_prog} cluster -m -s ${self_bp} -B 1000 -5 5,0 -3 3,${exten_bp} \
    -t ${n_thread} -j -x -v 1 -g  -O ${run}.e500 \
    ${run}.${tag_name}.${suffix}.nr.bam 1>> ${log_file} 2>> ${log_file}
```

rename loop files appropriately (produced by cluster step)

loop file name

```
mv ${run}.e500.clusters.cis.chiasig.gz ${run}.e500.clusters.cis.gz
mv ${run}.e500.clusters.trans.chiasig.gz ${run}.e500.clusters.trans.gz
```

Make subset file with intrachrom loops with PET_count >= 3 (pet count >=3, the loop is more confident)

for browsing in Excel

```
cis_file="${run}.e500.clusters.cis.gz"
be3_file="${run}.e500.clusters.cis.BE3"
```

```
zcat ${cis_file} | awk '{ if ( $7 >= 3 ) print }' > ${be3_file}
```

BAM --->pairs --->hic

BAM to pairs

```
paired_tag_bam="${run}.singlelinker.paired.UU.nr.bam" ## produced from tag
deduplication
${bin_dir}/util/pairix_src/util/bam2pairs/bam2pairs -c ${chrom_sizes} \
    ${paired_tag_bam} ${run}
```

Pairs to .hic

```
juicer="${bin_dir}/util/juicer_tools.1.7.5_linux_x64_jcuda.0.8.jar"
hic_file="ChIA-
PET_${genome}_${cell_type}_${ip_factor}_${run}_${run_type}_pairs.hic"
java -Xmx2g -jar ${juicer} pre -r \
    2500000,1000000,500000,250000,100000,50000,25000,10000,5000,1000 \ ## the
resolutation of juicebox
    ${run}.bsorted.pairs.gz ${hic_file} ${chrom_sizes}
```

3. call peaks from all nr bam file

Perform ChIP-Seq peak calling and plot density from ChIA-PET data (did not plot density from ChIA-pet data)

Convert to bedgraph

Sort bam for samtools counting bases and for BASIC visualization

```
if [ ! -f ${run}.for.BROWSER.bam ]
then
    echo -e "`date` Converting file formats..\n" >> ${log_file}
    samtools sort -@ 16 ${run}.singlelinker.paired.UU.nr.bam \
        -o ${run}.singlelinker.paired.UU.nr.sorted.bam
    samtools sort -@ 16 ${run}.singlelinker.single.UxxU.nr.bam \
        -o ${run}.singlelinker.single.UxxU.nr.sorted.bam
    samtools sort -@ 16 ${run}.none.UU.nr.bam \
        -o ${run}.none.UU.nr.sorted.bam
  #### sort bam file for 1.none 2.confilct3. single.paied bam
    samtools merge ${run}.for.BROWSER.bam \
        ${run}.singlelinker.paired.UU.nr.sorted.bam \
        ${run}.singlelinker.single.UxxU.nr.sorted.bam \
```

```
        ${run}.none.UU.nr.sorted.bam
    ###merged bam file to a bam for peak calling
    # Create BAM index
    samtools index ${run}.for.BROWSER.bam ${run}.for.BROWSER.bam.bai

    # Make bedgraph
    bedtools genomecov -ibam ${run}.for.BROWSER.bam \
        -bg > ${run}.for.BROWSER.bedgraph

    # Sort bedgraph
    ${bin_dir}/util/scripts/bedSort \
        ${run}.for.BROWSER.bedgraph \
        ${run}.for.BROWSER.sorted.bedgraph

    # Make bigwig
    ${bin_dir}/util/scripts/bedGraphToBigWig \
        ${run}.for.BROWSER.sorted.bedgraph \
        ${chrom_sizes} \
        ${run}.for.BROWSER.bigwig
fi
```

peak calling with SPP or macs2

is use SPP, a control input bam file must be supplied

```
if [ ${peak_caller} == "spp" ] || [ ${peak_caller} == "SPP" ]
then

    # Report SPP to log file
    echo -e "`date` Peak calling using SPP..\n" >> ${log_file}

    # Call peaks using SPP
    R --vanilla < ${bin_dir}/util/scripts/spp.R --args ${run}.for.BROWSER.bam \
        ${input_control} ${bin_dir} ${z_thresh}
else
    # Report MACS2 to log file
    echo -e "`date` Peak calling using MACS2..\n" >> ${log_file}

    if [ ${input_control} == 'none' ] || [ ${input_control} == 'None' ]
    then
        # Call peaks using MACS2 without input control
        macs2 callpeak --keep-dup all --nomodel -t ${run}.for.BROWSER.bam \
            -f BAM -g hs -n ${run}.no_input_all 1>> ${log_file} 2>> ${log_file}
    else
        # Call peaks using MACS2 with input control
        macs2 callpeak --keep-dup all --nomodel -t ${run}.for.BROWSER.bam \
            -c ${input_control} \
            -f BAM -g hs -n ${run}.all 1>> ${log_file} 2>> ${log_file}
    fi
fi
```

Annotate loops with peak support and call CCDs

Annotate loops

```
be3_file="${run}.e500.clusters.cis.BE3"
```

##########################changed by panpan 06162023(for orginal scripts, it is wrong)################

```
if [ ${peak_caller} == "spp" ] || [ ${peak_caller} == "SPP" ]
   then
        peak_file=${run}.for.BROWSER.spp.z6.broadPeak
    else
        peak_file=${run}.no_input_all_peaks.narrowPeak
fi
```

annotate loop with peak support

```
${dep_dir}/python ${bin_dir}/util/scripts/annotate_loops_with_peak_support.py \
    -l ${be3_file} -p ${peak_file}
## this part will produce a file names ${be3_file}.peak_annot
```

Create subsetted files by peak support

2 anchors

```
cat ${be3_file}.peak_annot | awk '{ if ( $8 == 2 ) print }' \
    > ${be3_file}.peak_annot.E2
```

1 or more anchor

```
cat ${be3_file}.peak_annot | awk '{ if ( $8 >= 1 ) print }' \
    > ${be3_file}.peak_annot.BE1
```

Convert loops to WashU format

2 anchors

```
${dep_dir}/python ${bin_dir}/util/scripts/convert_loops_to_washu_format.py \
    -l ${be3_file}.peak_annot.E2
```

1 or more anchor

```
${dep_dir}/python ${bin_dir}/util/scripts/convert_loops_to_washu_format.py \
    -l ${be3_file}.peak_annot.BE1
```

Annotate enhancer-promoter loops

Split loops into anchors (left and right anchors)

```
${dep_dir}/python ${bin_dir}/util/scripts/split_loops_into_anchors.py \
    -l ${be3_file}.peak_annot.BE1
```

left_anch=${be3_file}.peak_annot.BE1.left_anchors
right_anch=${be3_file}.peak_annot.BE1.right_anchors

Intersect anchors with enhancers

bedtools intersect -u -a ${left_anch} -b ${enhancer_bed_file}
> ${left_anch}.enhancers

bedtools intersect -u -a ${right_anch} -b ${enhancer_bed_file}
> ${right_anch}.enhancers

Intersect anchors with promoters

bedtools intersect -u -a ${left_anch} -b ${promoter_bed_file}
> ${left_anch}.promoters

bedtools intersect -u -a ${right_anch} -b ${promoter_bed_file}
> ${right_anch}.promoters

Annotate enhancer promoter loops

${dep_dir}/python ${bin_dir}/util/scripts/annotate_enhancer_promoter_loops.py
--left_enhancers ${left_anch}.enhancers
--right_enhancers ${right_anch}.enhancers
--left_promoters ${left_anch}.promoters
--right_promoters ${right_anch}.promoters