Input

files

```
REF_FASTA=/data/projects/misc/genomes/Mm/mm9/fasta/mm9.fa
VCF_SNP=/data/projects/misc/genomes/Mm/mm9/snps_indels/mgp.v2.snps.annot.reformat.vcf
VCF_INDELS=/data/projects/misc/genomes/Mm/mm9/snps_indels/mgp.v2.indels.annot.reformat.vcf
FASTQ_FOLDER=( /data/projects/akhtar/Tomek_ChIP-seq_2012-2013/fastq/fastqs_femNPC/ /data/projects/akhtar/Tomek_C
```

for RNA-seq: GTF file is needed, too

information

```
GENOME=mm9 # reference genome
MAT_STRAIN=129S1 # refers to vcf column
PAT_STRAIN=CASTEiJ # refers to vcf column
CHROMOSOMES=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,X
```

1. get_refmeta: Meta-data for reference genome (needed for vcf2mod)

```
/package/lapels-1.0.5/bin/get_refmeta -o ${GENOME}.meta ${GENOME} ${REF_FASTA}
```

2. vcf2mod: generating MOD files

a) Prepare VCF files (need to be indexed)

for vcf in ${VCF_INDELS} ${VCF_SNP}
do
/package/tabix-0.2.6/bgzip -c ${vcf} > ${vcf}.gz
/package/tabix-0.2.6/tabix -p vcf ${vcf}.gz
done

b) 1 MOD file per VCF file and genotype

```
echo "generating MOD files for ${MAT_STRAIN} and ${PAT_STRAIN}, SNPs with bad quality (FI tag = 0) will be disca
for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
/package/lapels-1.0.5/bin/vcf2mod -c ${CHROMOSOMES} -f -o ${GENOME}_SNPs_${genotype}.mod ${GENOME} ${GENOME}.met

/package/lapels-1.0.5/bin/vcf2mod -c ${CHROMOSOMES} -f -o ${GENOME}_indels_${genotype}.mod ${GENOME} ${GENOME}.m
done
wait
```

c) merge MOD files for SNPs and indels per genotype

```
for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
cat ${GENOME}_SNPs_${genotype}.mod ${GENOME}_indels_${genotype}.mod | sort -k2,2n -k3,3n | uniq | awk -f changeC
done
wait
```

3. insilico: generating pseudogenomes; CAVE: after this step, the MOD file will be gzipped (without any indication in the file name)

```
for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
/package/lapels-1.0.5/bin/insilico ${GENOME}_indels_SNPs_${genotype}_changedChr.mod ${REF_FASTA} -v -f -o pseudo
```

```
done
wait
```

4. Bowtie/Tophat: mapping to each pseudogenome

First, one needs to generate bowtie indeces for both pseudogenomes using **bowtie2-build** on the new FASTA files.
bowtie2-build outputs a set of 6 files with suffixes .1.bt2, .2.bt2, .3.bt2, .4.bt2, .rev.1.bt2, and .rev.2.bt2.
These files together constitute the index: they are all that is needed to align reads to that reference.
The original sequence FASTA files are no longer used by Bowtie 2 once the index is built.

```
/package/bowtie2-2.1.0/bowtie2-build pseudogenome_mm9_129S1.fa pseudogenome_mm9_129S1 > 129S1.bowtieIndex_`date
/package/bowtie2-2.1.0/bowtie2-build pseudogenome_mm9_CASTEiJ.fa pseudogenome_mm9_CASTEiJ > CASTEiJ.bowtieIndex_

for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do /package/bowtie2-2.1.0/bowtie2-build pseudogenome_${GENOME}_${genotype}.fa pseudogenome_${GENOME}_${genotype}
done
wait
```

ChIP-seq: bowtie2

Number of mismatches in seed: default is 0 (because the default alignment mode is --sensitive)
Seed length: default is 20 (because the default alignment mode is --sensitive)
Number of ambiguous character per read: determined by the function f(read_length) = a + b*read_length; default is a=0, b=0.15
Minimum alignment score needed for valid alignment: f(read_length) = a + b*read_length; default is a=0, b = -0.66
Increasing -M makes bowtie2 slower, but increases the likelihood that it will pick the correct alignment for a read that aligns many places.For reads that have more than +1 distinct, valid alignments, Bowtie 2 does not gaurantee that the alignment reported is the best possible in terms of alignment score.

for fastq_folder in ${FASTQ_FOLDER[@]}
do
for fastq in ${fastq_folder}*R1.fastq.gz
do
sample=$(basename "$fastq" .fastq.gz | sed 's/_R[1-2]//') # extracting sample name
for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
echo ${sample} "mapped to" ${genotype}
(/package/bowtie2-2.1.0/bowtie2 -x pseudogenome${GENOME}${genotype} -1 ${fastq_folder}${sample}_R1.fastq.gz -2 ${fastq_folder}${sample}_R2.fastq.gz -X 1000 -p 40 --rg-id mpi-ie --rg CN:deep_sequencing_unit --rg PL:illumina | /package/samtools/samtools view -Sb - | /package/samtools/samtools sort - ${genotype}${sample}.sorted) 2>&1 > MappingTo${genotype}${sample}.log
/package/samtools/samtools index ${genotype}${sample}.sorted.bam &
done
done
done
wait

RNA-seq: tophat

Given RNA-seq reads as input, TopHat2 begins by mapping reads against the known transcriptome, if an annotation file is provided.
This transcriptome mapping improves the overall sensitivity and accuracy of the mapping. It also gives the whole pipeline a significant
speed increase, owing to the much smaller size of the transcriptome compared with that of the genome.
For this case, the trancriptome files must be turned into strain-specific versions as well.

generate GTFs for S129 and CASTEiJ: **modmap**
transcriptome build for S129I and CASTEiJ genotypes using 1 set of FASTQs with **TopHat**
run TopHat on remaining samples

Lifting annotation: modmap

modmap expects 0-based positions, since gtf files are 1-based, I first convert them into 0-based, use modmap to change the annotation to match the individual genomes, and turn the resulting 0-based gtf file back into 1-based (additionally, I remove the negative numbers introduced by modmap for regions that fall into deletions)

```
awk -F "\t" '{OFS="\t"; print $1,$2,$3,$4-1, $5-1, $6, $7, $8,$9}' /data/projects/muehlpfordt/2013_Tomek/data_T1

for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
    /package/lapels-1.0.5/bin/modmap -d '\t' -f ${GENOME}_indels_SNPs_${genotype}_changedChr.mod Mus_musculus.NC

    awk -F "\t" '{OFS="\t";print $1,$2,$3, $4<0?$4*-1:$4+1,$5<0?$5*-1:$5+1,$6,$7,$8,$9}' Mus_musculus_${genotype

done
```

for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
for BAM in *${genotype}*bam
do
bam=$(basename "$BAM" .bam)
echo ${BAM} ${bam}*mappedBackTo_mm9.bam ${GENOME}_indels_SNPs*${genotype}*changedChr.mod*
/package/lapels-1.0.5/bin/pylapels -p 50 -o ${bam}_mappedBackTo_mm9.bam
*${GENOME}_indels_SNPs*${genotype}_changedChr.mod > ${bam}_stdout.txt 2> ${bam}_stderr.txt
done
done

TopHat

from the TopHat manual:

> For subsequent TopHat runs with the same genome and known transcripts but different reads, TopHat will no longer spend time building the transcriptome index because it can use the previously built transcriptome index files, so the -G option is no longer needed (however using it again will not force TopHat to rebuild the transcriptome index files if they are already present and with the matching version).
>
> *Note*: Only after the transcriptome files are built with one of the methods above, by a single TopHat process, it is safe to run multiple TopHat processes simultaneously making use of the same pre-built transcriptome index data.

** building the transcriptome index for 129S1 and CASTEiJ**

```
STD_DEV=`sed '1d' /data/projects/akhtar/Tasneem_Matt_RNA-seq_2013/fastq/insert_stats/insert_stats_RNA_WT_femES_1
INNER_DIST=`sed '1d' /data/projects/akhtar/Tasneem_Matt_RNA-seq_2013/fastq/insert_stats/insert_stats_RNA_WT_femE

for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
/package/tophat-2.0.8b.Linux_x86_64/tophat2 -G Mus_musculus_${genotype}_NCBIM37.67_1based.gtf \
--transcriptome-index transcriptome_data/Mus_musculus_${genotype}_transcriptomeIndex \
--no-coverage-search --library-type fr-firststrand --mate-inner-dist ${INNER_DIST} \
--mate-std-dev ${STD_DEV} \
pseudogenome_${GENOME}_${genotype} \
-o khanam_Akhtar_RNA_WT_femES_1_${genotype}
/data/projects/akhtar/Tasneem_Matt_RNA-seq_2013/fastq/khanam_Akhtar_RNA_WT_femES_1_R1.fastq.gz \
/data/projects/akhtar/Tasneem_Matt_RNA-seq_2013/fastq/khanam_Akhtar_RNA_WT_femES_1_R2.fastq.gz
done
```

looping over remaining files (for simplicity, I moved the fastq files)

```
for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
    for fastq in /data/projects/muehlpfordt/2014_AlleleSpecificMapping/02b_MappingToPseudogenomes_RNA-seq/remair
    do
    forStats=$(basename "$fastq" .fastq.gz | sed 's/_R[1-2]//' |  awk '{print substr($0,index($0,"_RNA")+1)}')
    STD_DEV=`sed '1d' /data/projects/akhtar/Tasneem_Matt_RNA-seq_2013/fastq/insert_stats/insert_stats_${forStats
    INNER_DIST=`sed '1d' /data/projects/akhtar/Tasneem_Matt_RNA-seq_2013/fastq/insert_stats/insert_stats_${forSt

    sample=$(basename "$fastq" .fastq.gz | sed 's/_R[1-2]//') # extracting sample name

    /package/tophat-2.0.8b.Linux_x86_64/tophat2 -o ${sample}_${genotype} -p 20 -G Mus_musculus_${genotype}_NCBIM
    done
done
```

indexing

for FOLDER in /data/projects/muehlpfordt/2014_AlleleSpecificMapping/02b_MappingToPseudogenomes_RNA-seq/*Akhtar*
do
/package/samtools/samtools index ${FOLDER}/accepted_hits.bam &
done

5. Lapels: translate different pseudogenome mappings back to reference genome

from the README:

> Lapels will generate a new BAM file with **corrected read positions**, **adjusted cigar strings**, and **annotated tags of variants** (eg. SNPs, Insertions, and Deletions). (...)
> The length of each read(template) and the mate position (if any) will be updated.
>
> input:
> UNC9-SN296_0254:3:1305:8580:174183#TGACCA 163 chr2 6361064 255 100M = 6361092 128 NM:i:1 NH:i:1
> UNC9-SN296_0254:3:1305:8580:174183#TGACCA 83 chr2 6361092 255 100M = 6361064 -128 NM:i:1 NH:i:1
>
> output:
> UNC9-SN296_0254:3:1305:8580:174183#TGACCA 163 chr2 6363700 255 35M7D63M1I1M = 6363728 134 NH:i:1 OC:Z:100M OM:i:1 d0:i:7 i0:i:1 s0:i:1
> UNC9-SN296_0254:3:1305:8580:174183#TGACCA 83 chr2 6363728 255 7M7D63M1I29M = 6363700 -134 NH:i:1 OC:Z:100M OM:i:1 d0:i:7 i0:i:1 s0:i:1
>
> New tags have been added for each read. The default tags and their meaning are shown below:

tag	meaning
OC	the old cigar in the alignment of the in silico genome
OM	the old NM (edit distance to the in silico sequence)
s0	the number of observed SNP positions having the in silico alleles
i0	the number of bases in the observed insertions having in silico alleles
d0	the number of bases in the observed deletions having in silico alleles

NOTE:

new CIGAR string! (cigar strings are in the reference coordinate)

NM:i tag will **disappear**! any filtering on mismatches to the pseudogenome must be done using the **OM:i** tag!

paired end information is adjusted

ChIP-seq

```
for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
```

```
do
for BAM in ${genotype}*bam
do

bam=$(basename "$BAM" .sorted.bam)

/package/lapels-1.0.5/bin/pylapels -p 50 -f -o ${bam}_mappedBackTo_mm9.bam ${GENOME}_indels_SNPs_${genotype}_cha

done
done
```

RNA-seq

```
for FOLDER in /data/projects/muehlpfordt/2014_AlleleSpecificMapping/02b_MappingToPseudogenomes_RNA-seq/*Akhtar*
do

OUT=$(basename "$FOLDER" | sed 's/[a-zA-Z]*\_Akhtar\_//'| sed 's/\_378//' | sed 's/femES\_//' | sed 's/RNA/femES
echo ${OUT}

ln -s ${FOLDER}/accepted_hits.bam ${OUT}_indels_SNPs.bam
ln -s ${FOLDER}/accepted_hits.bam.bai ${OUT}_indels_SNPs.bam.bai

done

for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
for BAM in *${genotype}*bam
do

bam=$(basename "$BAM" .bam)
echo ${BAM} ${bam}_mappedBackTo_mm9.bam ${GENOME}_indels_SNPs_${genotype}_changedChr.mod

/package/lapels-1.0.5/bin/pylapels -p 50 -o ${bam}_mappedBackTo_mm9.bam ${GENOME}_indels_SNPs_${genotype}_change

done
done
```

6. Suspenders

merging the parental and maternal mapping, determining best fit for each read --> 1 BAM file with reads where flags indicate maternal or paternal origin
in theory, suspenders can run on several processors, but I have the feeling that it doesn't thoroughly make use of it which is why I start the processes in parallel manually

> from suspenders README

```
    mother.bam:
    ...
    UNC9-SN296_0254:7:1101:1482:32883#CGATGT 153 chr2 22809107 50 25M3406N75M * 0 0 <SEQ> <MAPQ>
    NH:i:1 MD:Z:30C69 OM:i:1 XO:i:0 XM:i:1 i0:i:0 s0:i:0 OC:Z:25M3405N75M XG:i:0 AS:i:-1 XS:A:- d0:i:0
    ...
    UNC9-SN296_0254:7:1101:1483:94304#CGATGT 137 chr1 81336983 50 100M * 0 0 <SEQ> <MAPQ>
    NH:i:1 MD:Z:69A13C16 OM:i:2 XN:i:0 XO:i:0 XM:i:2 i0:i:0 s0:i:0 OC:Z:100M XG:i:0 AS:i:-7 YT:Z:UU d0:i:0
    ...

    father.bam:
    ...
    UNC9-SN296_0254:7:1101:1482:32883#CGATGT 153 chr2 22809107 50 25M3406N75M * 0 0 <SEQ> <MAPQ>
    NH:i:1 MD:Z:30C69 OM:i:1 XO:i:0 XM:i:1 i0:i:0 s0:i:0 OC:Z:25M3407N75M XG:i:0 AS:i:-1 XS:A:- d0:i:0
    ...
    UNC9-SN296_0254:7:1101:1483:94304#CGATGT 137 chr1 81336983 50 100M * 0 0 <SEQ> <MAPQ>
    NH:i:1 MD:Z:69A30 OM:i:1 XN:i:0 XO:i:0 XM:i:1 i0:i:0 s0:i:1 OC:Z:100M XG:i:0 AS:i:-1 YT:Z:UU d0:i:0
    ...
```

```
merged.bam:
...
UNC9-SN296_0254:7:1101:1482:32883#CGATGT 153 chr2 22809107 50 25M3406N75M * 0 0 <SEQ> <MAPQ>
NH:i:1 MD:Z:30C69 OM:i:1 XO:i:0 XM:i:1 i0:i:0 s0:i:0 OC:Z:25M3405N75M XG:i:0 AS:i:-1 XS:A:- d0:i:0
YA:A:3 ms:i:0 mi:i:0 md:i:0 ps:i:0 pi:i:0 pd:i:0 pc:Z:25M3407N75M pm:i:1 po:A:3 ct:A:U
...
UNC9-SN296_0254:7:1101:1483:94304#CGATGT 137 chr1 81336983 50 100M * 0 0 <SEQ> <MAPQ>
NH:i:1 MD:Z:69A30 OM:i:1 XN:i:0 XO:i:0 XM:i:1 i0:i:0 s0:i:1 OC:Z:100M XG:i:0 AS:i:-1 YT:Z:UU d0:i:0
YA:A:3 ms:i:0 mi:i:0 md:i:0 ps:i:1 pi:i:0 pd:i:0 mc:Z:100M mm:i:2 po:A:2 ct:A:Q
...
```

> In the output, reads are merged if they are determined to be 'equal' based on a variety of filters. Additionally, new tags have been added for each read to identify how the reads were merged:

Major tags:
po : an integer flag set representing the Parent of Origin of this particular read. If multiple inputs have identical positions, cigar strings, and number of mismatches, then we consider it to be an identical alignment. In this case, the bit for each input will be set. For example, given three inputs and an identical read in the second and third input, our flag would be 0b110 which would be stored as integer value 6.

ct : the Choice Type for this read, eg the filter that determined which read to save
* 'U' for unique filter: only one possible alignment available, so it was kept
* 'Q' for quality score filter: chose the possible alignment with the highest score (calculated using the Bowtie schema) (e.g. when paternal alignment resulted in one more mismatch than the maternal one, the maternal one will be used)
* 'P' for pileup height filter: chose the possible alignment with the highest pileup height
* 'R' for random filter: chose a random possible alignment from a set of 'equal' alignments based on the previous filters (Each mapping has a 33.3% chance of being chosen for the final output).
* 'K' for kept-all filter: keep all possible alignments from a set of 'equal' alignments based on the previous filters

---

> from Database(2014) doi: 10.1093/database/bau057

For **paired-end reads** of a fragment, we 'link' parental origins. The main idea is that if one read from a fragment can be assigned to a parent unambiguously, we infer that its mate also came from that same parent, even if the mate contains no informative variants. (i) the mappings of two properly paired reads will be marked with the same parental origin, (ii) the quality score will be calculated once based on the mismatches and indels from the paired mapping and (iii) the merge always prefers a paired-end mapping to one or two single-end mappings from the pair.

ChIP-seq

```
for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
do
    for BAM in /data/projects/muehlpfordt/2014_AlleleSpecificMapping/03_MappingBack_ToReferenceGenome/${geno
    do
        bam=$(basename "$BAM" .bam)
        /package/samtools/samtools sort -n ${BAM} ${bam}.ReadSorted &
    done
    wait
done
wait

for BAM in ${MAT_STRAIN}*mappedBackTo_mm9.ReadSorted.bam
do

    Sample=$(basename "$BAM" _mappedBackTo_mm9.ReadSorted.bam | sed 's/129S1\_//' )
    echo ${Sample} ${MAT_STRAIN}_${Sample}_mappedBackTo_mm9.ReadSorted.bam ${PAT_STRAIN}_${Sample}_mappedBac

    ~/bin/pysuspendersNew --pileup -c pileupImage_${Sample}.png ./${Sample}_${MAT_STRAIN}.${PAT_STRAIN}_susp

done
```

RNA-seq

```
    cd /data/projects_2/muehlpfordt/2014_AlleleSpecificMapping/05b_Suspenders_RNA-seq


    # read name sorting

    for genotype in ${MAT_STRAIN} ${PAT_STRAIN}
    do
        for BAM in /data/projects/muehlpfordt/2014_AlleleSpecificMapping/03b_MappingBack_ToReferenceGenome_RNA-s
        do
            bam=$(basename "$BAM" .bam)
            /package/samtools/samtools sort -n ${BAM} ${bam}.ReadSorted &
        done
        wait
    done
    wait
```

28.7. waiting for the sorting to finish

```
    for BAM in *${MAT_STRAIN}_*ReadSorted.bam
    do

        Sample=$(basename "$BAM" _mappedBackTo_mm9.ReadSorted.bam | sed 's/129S1\_//' )
        echo ${Sample} ${MAT_STRAIN}_${Sample}_mappedBackTo_mm9.ReadSorted.bam ${PAT_STRAIN}_${Sample}_mappedBac

        ~/bin/pysuspendersNew --pileup -c pileupImage_${Sample}.png ./${Sample}_${MAT_STRAIN}.${PAT_STRAIN}_susp

    done
    wait
```

7. BAM file filter, merge, sort, index

Suspender files should be

filtered
multiple alignments
randomly assigned reads
chrM, random chromosomes (/data/projects/scripts/samFilter_v2.py)
sort
merge replicates
index

> 24.7.2014: see Allele*nonoptimalFiltering.md for what I really did since I modified the samFilter_v2_suspenders.py script after doing the initial round of filtering

ChIP-seq

```
    wait
    for BAM in /data/projects/muehlpfordt/2014_AlleleSpecificMapping/05_Suspenders/*129S1.CASTEiJ_suspendersMerg
    do
        out=$(basename "$BAM" .bam | sed 's/\_129S1.CASTEiJ//')
        echo ${out}
        /package/samtools/samtools view -h -F 4 ${BAM}|\
         ./samFilter_v2_suspenders.py \
            --filter_out_from_BED /data/projects_2/muehlpfordt/2012/2012-08-16/suspiciousPeaks/conspicuousEnrich
             --random --chrM --multiple --mismatch --lowqual --suspendersRandom |\
             /package/samtools/samtools view -Sb - > ${out}_filtered.rnsort.bam &
     done
     wait

 # merging replicates

     for CHIP in Input MOF MSL1 MSL2 MCRS1 NSL3
     do
```

```
        CELL=ES
        /package/samtools/samtools merge - ${CHIP}A_fem${CELL}_suspendersMerged_filtered.bam ${CHIP}B_fem${CELL}
     done
    wait

    for CHIP in Input MOF MSL2 NSL3
    do
        CELL=NPC
        /package/samtools/samtools merge - ${CHIP}A_fem${CELL}_suspendersMerged_filtered.bam ${CHIP}B_fem${CELL}
    done
    wait

 # indexing (for this, I have to re-sort the files as the suspenders output is sorted by read name)

    for BAM in *rnsort*bam
    do
    out=`basename "$BAM" .bam`
    /package/samtools/samtools sort -m 4000000000 ${BAM} ${out}.sort
    done

    for BAM in *.sort.bam
    do
    /package/samtools/samtools index ${BAM} &
    done
    wait
```

8. bigWig file generation

--> see README_bigWigs.md