```
#!/bin/bash
# Super script to run the pRESTO pipeline on AbVitro library v3.0 data
#
# Author:  Jason Anthony Vander Heiden, Gur Yaari, Namita Gupta, Ang Cui
# Date:    2015.10.20
#
# Required Arguments:
#   $1 = read 1 file (C-region start sequence)
#   $2 = read 2 file (V-region start sequence)
#   $3 = output directory (absolute path)
#   $4 = number of subprocesses for multiprocessing tools

# Define run parameters
LOG_RUNTIMES=true
ZIP_FILES=false
ALIGN_STEP=false
CALC_DIV=true
MAXDIV=0.1
PRCONS=0.6
FSQUAL=20
BCQUAL=0
R1_MAXERR=0.2
R2_MAXERR=1.0 # VPRIMERs start at variable positions but are not very relevant;
make the maxerr cutoff high
AP_MAXERR=0.2
ALPHA=0.01
FS_MISS=20
CS_MISS=20
MUSCLE_EXEC=$HOME/bin/muscle3.8.31_i86linux64
NPROC=$4

# Define input files
R1_FILE=$1
R2_FILE=$2
OUTDIR=$3
R1_PRIMER_FILE='<path>/CPrimers_MouseV3.fasta'
R2_PRIMER_FILE='<path>/VPrimers_MouseV3.fasta'

# Define script execution command and log files
mkdir -p $OUTDIR; cd $OUTDIR
RUNLOG="${OUTDIR}/Pipeline.log"
echo '' > $RUNLOG
if $LOG_RUNTIMES; then
        TIMELOG="${OUTDIR}/Time.log"
        echo '' > $TIMELOG
        RUN="nice -19 /usr/bin/time -o ${TIMELOG} -a -f %C\t%E\t%P\t%Mkb"
```

```
else
	RUN="nice -19"
fi

# Start
echo "DIRECTORY:" $OUTDIR
echo "VERSIONS:" >> $RUNLOG
AlignSets.py -v >> $RUNLOG
AssemblePairs.py -v >> $RUNLOG
BuildConsensus.py -v >> $RUNLOG
CollapseSeq.py -v >> $RUNLOG
FilterSeq.py -v >> $RUNLOG
MaskPrimers.py -v >> $RUNLOG
PairSeq.py -v >> $RUNLOG
ParseHeaders.py -v >> $RUNLOG
ParseLog.py -v >> $RUNLOG
SplitSeq.py -v >> $RUNLOG

# Filter low quality reads
echo "   1: FilterSeq quality     $(date +'%H:%M %D')"
$RUN FilterSeq.py quality -s $R1_FILE -q $FSQUAL --nproc $NPROC --outname R1 \
   --outdir . --log QualityLogR1.log --clean >> $RUNLOG
$RUN FilterSeq.py quality -s $R2_FILE -q $FSQUAL --nproc $NPROC --outname R2 \
   --outdir . --log QualityLogR2.log --clean >> $RUNLOG

# Identify primers and UID
echo "   2: MaskPrimers score     $(date +'%H:%M %D')"
$RUN MaskPrimers.py score -s R1_quality-pass.fastq -p $R1_PRIMER_FILE --mode
cut --start 0 \
   --maxerror $R1_MAXERR --nproc $NPROC --log PrimerLogR1.log --clean >>
$RUNLOG
$RUN MaskPrimers.py score -s R2_quality-pass.fastq -p $R2_PRIMER_FILE --mode
cut --barcode --start 17 \
   --maxerror $R2_MAXERR --nproc $NPROC --log PrimerLogR2.log --clean >>
$RUNLOG

# Assign UIDs to read 1 sequences
echo "   3: PairSeq               $(date +'%H:%M %D')"
$RUN PairSeq.py -1 R2*primers-pass.fastq -2 R1*primers-pass.fastq -f BARCODE --
coord illumina --clean >> $RUNLOG

if $ALIGN_STEP; then
	# Multiple align UID read groups
	echo "   4: AlignSets muscle      $(date +'%H:%M %D')"
	$RUN AlignSets.py muscle -s R1*pair-pass.fastq --exec $MUSCLE_EXEC --
nproc $NPROC \
```

```
        --log AlignLogR1.log --clean >> $RUNLOG
    $RUN AlignSets.py muscle -s R2*pair-pass.fastq --exec $MUSCLE_EXEC --
nproc $NPROC \
        --log AlignLogR2.log --clean >> $RUNLOG
    BCR1_FILE=R1*align-pass.fastq
    BCR2_FILE=R2*align-pass.fastq
else
    BCR1_FILE=R1*pair-pass.fastq
    BCR2_FILE=R2*pair-pass.fastq
fi

# Build UID consensus sequences
echo "   5: BuildConsensus       $(date +'%H:%M %D')"
if $CALC_DIV; then
    $RUN BuildConsensus.py -s $BCR1_FILE --bf BARCODE --pf PRIMER --prcons
$PRCONS \
        -q $BCQUAL --maxdiv $MAXDIV --nproc $NPROC --log ConsensusLogR1.log
--clean >> $RUNLOG
    $RUN BuildConsensus.py -s $BCR2_FILE --bf BARCODE --pf PRIMER \
        -q $BCQUAL --maxdiv $MAXDIV --nproc $NPROC --log ConsensusLogR2.log
--clean >> $RUNLOG
else
    $RUN BuildConsensus.py -s $BCR1_FILE --bf BARCODE --pf PRIMER --prcons
$PRCONS \
    -q $BCQUAL --nproc $NPROC --log ConsensusLogR1.log --clean >> $RUNLOG
    $RUN BuildConsensus.py -s $BCR2_FILE --bf BARCODE --pf PRIMER \
    -q $BCQUAL --nproc $NPROC --log ConsensusLogR2.log --clean >> $RUNLOG
fi


# Assemble paired ends
echo "   6: AssemblePairs        $(date +'%H:%M %D')"
$RUN AssemblePairs.py align -1 R2*consensus-pass.fastq -2 R1*consensus-
pass.fastq \
  --1f CONSCOUNT --2f PRCONS CONSCOUNT --coord presto --rc tail --maxerror
$AP_MAXERR --alpha $ALPHA \
  --nproc $NPROC --log AssembleLog.log --clean >> $RUNLOG

# Remove sequences with many Ns
echo "   7: FilterSeq missing     $(date +'%H:%M %D')"
$RUN FilterSeq.py missing -s *assemble-pass.fastq -n $FS_MISS --inner \
  --nproc $NPROC --log MissingLog.log >> $RUNLOG

# Rewrite header with minimum of CONSCOUNT
echo "   8: ParseHeaders collapse  $(date +'%H:%M %D')"
$RUN ParseHeaders.py collapse -s *missing-pass.fastq -f CONSCOUNT --act min \
```

```
   --outname Assembled --fasta > /dev/null

# Remove duplicate sequences
echo "   9: CollapseSeq          $(date +'%H:%M %D')"
$RUN CollapseSeq.py -s Assembled_reheader.fasta -n $CS_MISS --uf PRCONS \
   --cf CONSCOUNT --act sum --outname Assembled --inner >> $RUNLOG

# Filter to sequences with at least 2 supporting sources
echo "  10: SplitSeq group        $(date +'%H:%M %D')"
$RUN SplitSeq.py group -s Assembled_collapse-unique.fasta -f CONSCOUNT --num 2
>> $RUNLOG

# Create table of final repertoire
echo "  11: ParseHeaders table    $(date +'%H:%M %D')"
$RUN ParseHeaders.py table -s Assembled_collapse-unique_atleast-2.fasta -f ID
PRCONS CONSCOUNT DUPCOUNT >> $RUNLOG

# Process log files
echo "  12: ParseLog              $(date +'%H:%M %D')"
$RUN ParseLog.py -l QualityLogR[1-2].log -f ID QUALITY > /dev/null &
$RUN ParseLog.py -l PrimerLogR[1-2].log -f ID BARCODE PRIMER ERROR >
/dev/null &
$RUN ParseLog.py -l ConsensusLogR[1-2].log -f BARCODE SEQCOUNT CONSCOUNT
PRIMER PRCOUNT PRFREQ DIVERSITY > /dev/null &
$RUN ParseLog.py -l AssembleLog.log -f ID OVERLAP LENGTH PVAL ERROR
HEADFIELDS TAILFIELDS > /dev/null &
$RUN ParseLog.py -l MissingLog.log -f ID MISSING > /dev/null &
wait

if $ZIP_FILES; then
   tar -cf LogFiles.tar *LogR[1-2].log *Log.log
   gzip LogFiles.tar
   rm *LogR[1-2].log *Log.log

   tar -cf TempFiles.tar R[1-2]_*.fastq *under* *duplicate* *undetermined*
*reheader*
   gzip TempFiles.tar
   rm R[1-2]_*.fastq *under* *duplicate* *undetermined* *reheader*
fi

# End
echo -e "DONE\n"
cd ../
```

Supplementary File 1: Script for Ig sequence pre-processing.

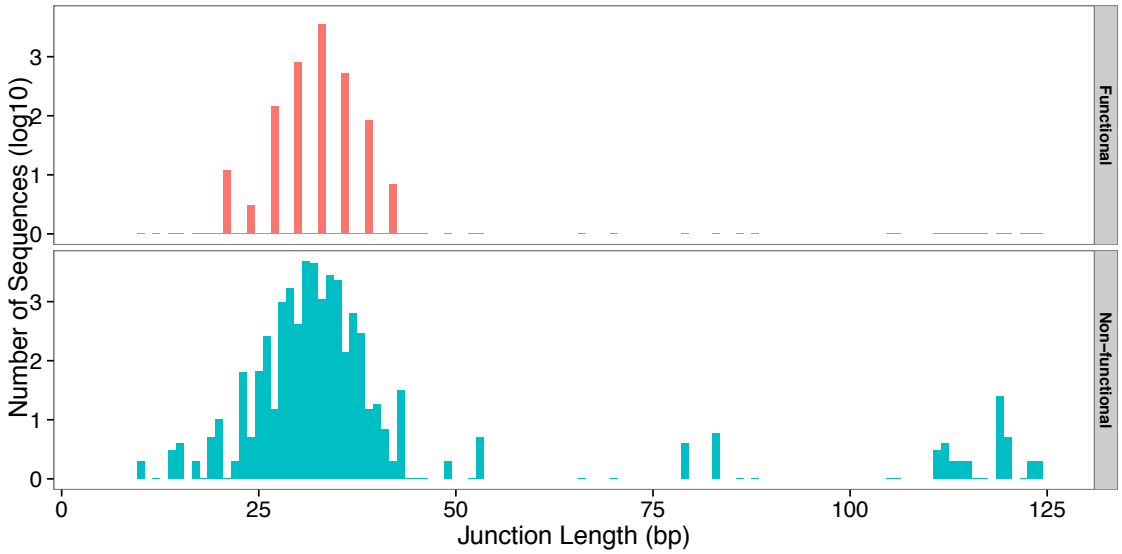

Figure S1: Many non-functional κ sequences contain out-of-frame junctions. Junction length distribution for functional (top panel) and non-functional (bottom panel) κ light chain sequences from NP-immunized mice (NP1-4 in Table 1). Functionality was determined by IMGT/HighV-Quest.

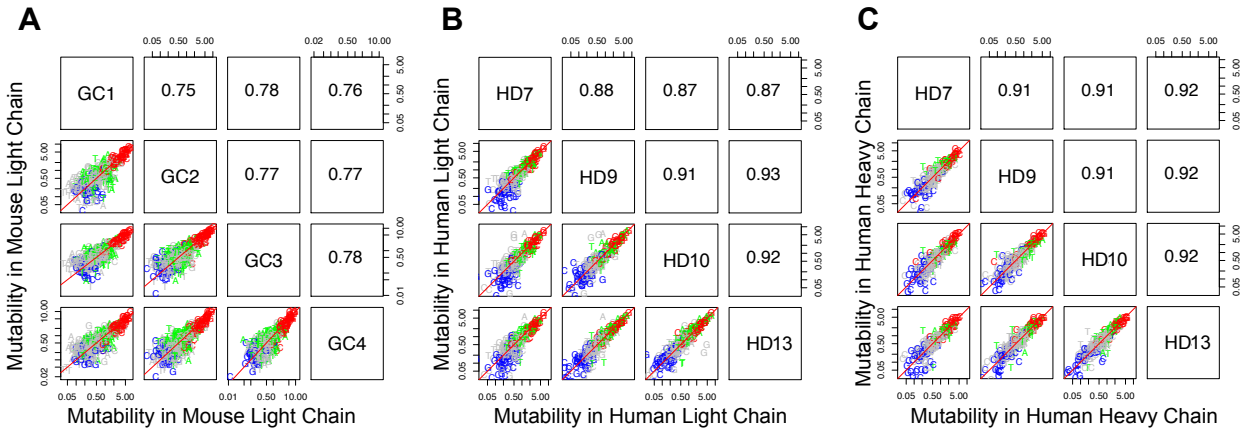

Figure S2: SHM targeting is highly consistent across individual samples. (A) Separate RS5NF models were constructed for each of the four individual mice (NP1-4 in Table 1) to estimate the mutability of 5-mer motifs in non-functional κ sequences from NP-binding B-cells. Separate S5F models were constructed for each of the four individual human subjects (HD samples in Table 2) to estimate the mutability of 5-mer motifs in (B) light chain sequences and (C) heavy chain sequences. The mutabilities were normalized such that the average mutability of the motifs observed in each pair of samples was 1. The lower triangle shows the pair-wise comparison of mutabilities for each 5-mer (points). The upper triangle shows the Spearman correlations between the estimated mutabilities in each motif.

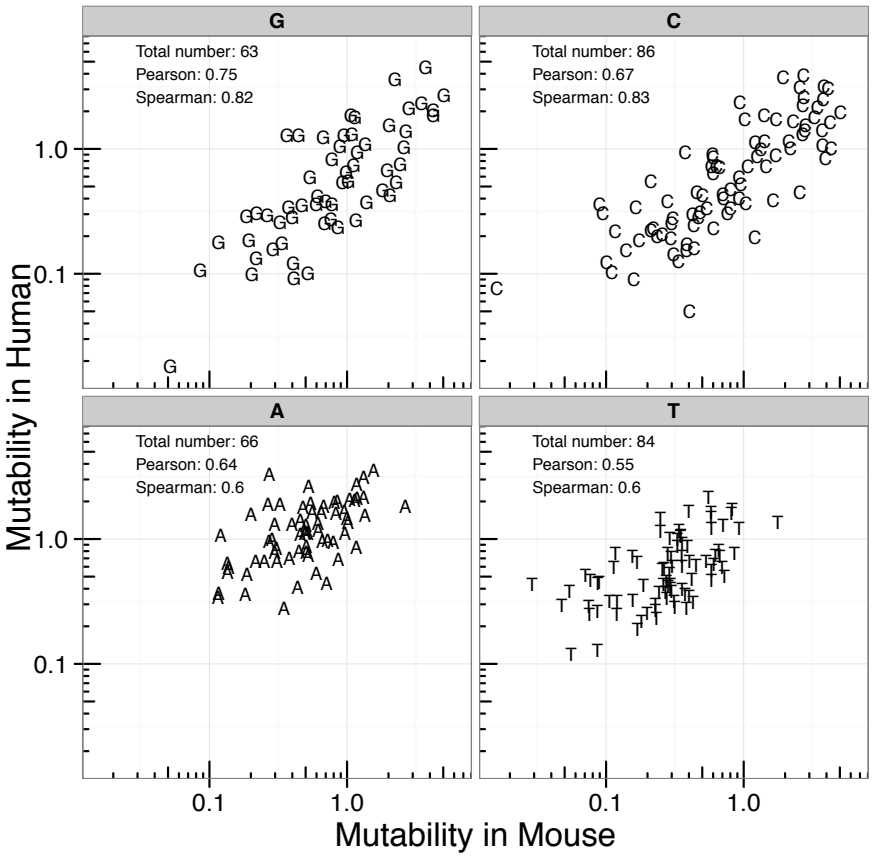

Figure S3: SHM targeting in mouse and human is highly correlated when conditioned on the base being targeted. The mutability values from Figure 4A were plotted separately for 5-mer motifs where the central base was A, T, G or C.