**MIPs with SUNKs: a novel method for genotyping copy number of highly identical paralogs**

-concept

***Implementation:***

**1. Align paralogous sequences with Clustal 2.1 using the commands below (the input you provide is highlighted).** You should have one fasta file containing all paralogous sequences of interest. These sequences should only be sequences from regions shared between paralogs. That is, if you have multiple contig sequences each containing a shared paralogous region and unique flanking regions, only include sequences from the shared paralogous region in this fasta file. This file will serve as input to the multiple sequence alignment program Clustal 2.1.

clustalw2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\* CLUSTAL 2.1 Multiple Sequence Alignments \*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Sequence Input From Disc

2. Multiple Alignments

3. Profile / Structure Alignments

4. Phylogenetic trees

S. Execute a system command

H. HELP

X. EXIT (leave program)

Your choice: 1

Sequences should all be in 1 file.

7 formats accepted:

NBRF/PIR, EMBL/SwissProt, Pearson (Fasta), GDE, Clustal, GCG/MSF, RSF.

Enter the name of the sequence file : /net/gs/vol2/home/xnuttle/MIPs/RH/RHD\_flank\_shared\_seqs.fasta

Sequence format is Pearson

Sequences assumed to be DNA

Sequence 1: small\_dup\_5p 9143 bp

Sequence 2: small\_dup\_3p 9329 bp

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\* CLUSTAL 2.1 Multiple Sequence Alignments \*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Sequence Input From Disc

2. Multiple Alignments

3. Profile / Structure Alignments

4. Phylogenetic trees

S. Execute a system command

H. HELP

X. EXIT (leave program)

Your choice: 2

\*\*\*\*\*\* MULTIPLE ALIGNMENT MENU \*\*\*\*\*\*

1. Do complete multiple alignment now Slow/Accurate

2. Produce guide tree file only

3. Do alignment using old guide tree file

4. Toggle Slow/Fast pairwise alignments = SLOW

5. Pairwise alignment parameters

6. Multiple alignment parameters

7. Reset gaps before alignment? = OFF

8. Toggle screen display = ON

9. Output format options

I. Iteration = NONE

S. Execute a system command

H. HELP

or press [RETURN] to go back to main menu

Your choice: 1

Enter a name for the CLUSTAL output file [/net/gs/vol2/home/xnuttle/MIPs/RH/RHD\_flank\_shared\_seqs.aln]: /net/gs/vol2/home/xnuttle/MIPs/RH/out.aln

Start of Pairwise alignments

Aligning...

Sequences (1:2) Aligned. Score: 98

Enter name for new GUIDE TREE file [/net/gs/vol2/home/xnuttle/MIPs/RH/RHD\_flank\_shared\_seqs.dnd]: /net/gs/vol2/home/xnuttle/MIPs/RH/out.dnd

Guide tree file created: [/net/gs/vol2/home/xnuttle/MIPs/RH/out.dnd]

There are 1 groups

Start of Multiple Alignment

Aligning...

Group 1: Sequences: 2 Score:169972

Alignment Score 64406

Consensus length = 9345

CLUSTAL-Alignment file created [/net/gs/vol2/home/xnuttle/MIPs/RH/out.aln]

CLUSTAL 2.1 multiple sequence alignment

small\_dup\_5p TCCAATGTTCGCGCAGGCACTGGAGTCAGAGAAAATGGAGTTGAATCCTTTCTCTGCCAC

small\_dup\_3p TCCAATGTTCGCGCAGGCACTGGAGTCAGAGAAAATGGAGTTGAATCCTTTCTCTGCCAC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

small\_dup\_5p TCTTTGAGGAGAATCTCACCATTTATTATGCACTGTAGAATACAACAATAAAATACAGCC

small\_dup\_3p TCTTTGAGGAGAATCTCACCATTTATTATGCACTGTAGAATACAACAATAAAATACAGCC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

small\_dup\_5p ATGTACCACATAACAACATCTTGGTAAACAACAGACTGCATATATGATGGTGGTCATCCA

small\_dup\_3p ATGTACCACATAACAACATCTTGGTAAACAACAGACTGCATATATGATGGTGGTCATCCA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

small\_dup\_5p GTAAGCTAAGGTTAATTTATTATTATTCCCT--------TTTTTTTTTCTTTTTTTTGAG

small\_dup\_3p GTAAGCTAAGGTTAATTTATTATTATTCCTTGTTTTTTTTTTTTTTTTTTTTTTTTTGAG

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \* \*\*\*\*\*\*\*\*\* \*\*\*\*\*\*\*\*\*\*\*

small\_dup\_5p ATGTAGTCTTACTCTGTCACCCAGGCTAGAGTGCAATGGCACCATCTTGGCTCACTGCAA

small\_dup\_3p ATGTAGTCTTACTCTGTCACCCAGGCTAGAGTGCAATGGCACCATCTTGGCTCACTGCAA

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Press [RETURN] to continue or X to stop: x

\*\*\*\*\*\* MULTIPLE ALIGNMENT MENU \*\*\*\*\*\*

1. Do complete multiple alignment now Slow/Accurate

2. Produce guide tree file only

3. Do alignment using old guide tree file

4. Toggle Slow/Fast pairwise alignments = SLOW

5. Pairwise alignment parameters

6. Multiple alignment parameters

7. Reset gaps before alignment? = OFF

8. Toggle screen display = ON

9. Output format options

I. Iteration = NONE

S. Execute a system command

H. HELP

or press [RETURN] to go back to main menu

Your choice: <ENTER>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\* CLUSTAL 2.1 Multiple Sequence Alignments \*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. Sequence Input From Disc

2. Multiple Alignments

3. Profile / Structure Alignments

4. Phylogenetic trees

S. Execute a system command

H. HELP

X. EXIT (leave program)

Your choice: x

**2. Convert Clustal 2.1 alignment output into multiple fasta files containing aligned sequence for each paralog.** To do this, follow the steps below:

-open Jalview

-File 🡪 Input Alignment 🡪 from File

-Browse and select the Clustal 2.1 output alignment file (out.aln) from the previous step.

-In the small window that opens, File 🡪 Output to Textbox 🡪 FASTA

-For each sequence, copy it (including the line with the sequence name having the character “>”) from the “Alignment output – FASTA” window and paste it into a new text file

-Save these new text files as fasta files

-Designate one of these sequences as the master sequence, from which all MIPs will be designed (which sequence you choose to be the master sequence may be arbitrary, or it may not be; a clear case of the latter scenario is *SRGAP2*, in which *SRGAP2A* has additional sequence of interest to target, 3’ unique exons not shared with other family members; thus, it makes sense that *SRGAP2A* is master in this case)

**3. Identify SUNs.** To do this, you will need to use a SUN finding program. For generic cases, the program get\_SUN\_coords.c will work well. Generic cases are those in which:

-there is only one region of interest that aligns between contigs; that is, contigs contain only a single segmental duplication (for *RH*, this is not true; there are two distinct segmental duplications of interest)

-there is no large-scale structural variation between contigs, such as a large deletion within one of the paralogs in the shared region (for *SRGAP2*, this is not true; *SRGAP2D* contains a large internal deletion)

Here is an example use of this program (all files can be found in the directory /net/gs/vol2/home/xnuttle/MIPs/SRGAP2):

./get\_SUN\_coords HYDIN16\_RP11\_shared.fasta HYDIN1\_CH17\_shared.fasta HYDIN\_coord\_convert.bed HYDIN.suns

This program takes fasta files containing aligned sequences (generated in step 2 above) as its first arguments (at least two such files, but possibly more), a file having contig names and start positions of aligned sequence within each contig as its penultimate argument, and the name of an output file (I recommend ending this name with “.suns”) as its final argument. The output generated is as follows:

1st column: name of contig containing a SUN

2nd column: the position (in base 1) of that SUN in the contig sequence

3rd column: the position (in base 1) of that SUN in the master sequence

**4. Identify suitable MIP target regions.** To do this, you will need to modify and use a form of mipmaker\_score.c (/net/gs/vol2/home/xnuttle/MIPs/RH/RH\_mipmaker\_score.c or /net/gs/vol2/home/xnuttle/MIPs/SRGAP2/mipmaker\_score.c). Unfortunately, this code is not very generalizable in its current form!!!

./mipmaker\_score SRGAP2A\_shared.fasta SRGAP2B\_shared.fasta SRGAP2C\_shared.fasta SRGAP2D\_shared.fasta regions\_to\_target.bed

This program takes fasta files containing aligned sequences (generated in step 2 above) as its first arguments and the name of an output file as its final argument. The output generated is as follows:

1st column: master sequence name

2nd column: start position (in base 0) in master sequence of region to target

3rd column: end position (in base 1) in master sequence of region to target

4th column: score for targeted region

This program outputs all possible regions for MIP targeting, scoring them according to how much paralog-distinguishing variation they contain (more such variation leads to higher scores). The output file will be used for the MIP design process.

**5. Generate a SNP table file.** To do this, you will need to run two programs, one to flag regions of the master sequence having SUNs and indel differences between paralogs, and one to flag regions of the master sequence having common SNPs specified in the UCSC genome browser. (You will need to obtain sequence from the browser with such SNPs marked by lowercase characters.) Unfortunately, this code also is not very generalizable in its current form!!!

./make\_SNP\_table\_RH\_SNPs RH\_master\_with\_common\_SNPs\_lowercase.fasta RH\_common\_SNPs.bed

./make\_SNP\_table\_RH RH\_master\_aligned.fasta RH\_master\_other\_dups\_aligned.fasta RH\_shared\_SNP\_table.bed

cat RH\_common\_SNPs.bed >> RH\_shared\_SNP\_table.bed

sort –k3n,3 RH\_shared\_SNP\_table.bed|uniq > RH\_master\_SNP\_table.bed

Most fields within the resulting SNP table file contain dummy values. The only ones that matter are the 2nd and 3rd columns, which specify the location of any variant (SUN, common SNP, indel in any paralog) that may affect MIP arm hybridization. The MIP design pipeline uses this file to avoid designing MIP arms over these locations. This is important because the entire method assumes equal MIP hybridization efficiency to each paralog—this assumption would likely not hold if MIP arm hybridization regions contain sequence variants.

**6. Send the master sequence fasta file, the file containing all possible regions for MIP targeting (generated in step 4), and the SNP table file (generated in step 5) to Brian O’Roak, who will use this information to design MIPs.**

**7. Identify SUNKs.** To do this, you will need to use the SUNK finding program sunksearch.c. Currently, you will need to request 35 Gb of memory to run this program. Details of usage can be found in the file /net/gs/vol2/home/xnuttle/programs/sunksearch.r eadme.

**8. Map 12 high coverage human genomes to contigs.** Refer to /net/gs/vol2/home/xnuttle/pipeline\_directions/mapping2.readme for how to do this. The locations of sequence data for high coverage genomes are posted on the Eichler Lab Wiki.

**9. Identify likely fixed SUNs.** Guidelines for how to do this can be found at /net/gs/vol2/home/xnuttle/pipeline\_directions/SUN\_analysis. The C programs underlying this analysis are well commented and explain the process. The end result is a .suns.fixed file having the exact same information as the output file generated in step 3 above (the .suns file) as well as an additional column with SUN scores corresponding to the average number of high coverage genomes supporting a SUN's presence.

**10. Process the MIP design file (obtained from Brian) to get a set of good possible MIPs for paralog-specific copy number genotyping.** Name this design file with the ending .mipdesign. The easiest way to process this file is in Microsoft Excel. Here are criteria for processing:

-discard all MIPs having scores < 3.0

grep "3\.0" RH\_SUN\_target\_regions.mipdesign > RH\_SUN\_score3.mipdesign

grep "5\.0" RH\_SUN\_target\_regions.mipdesign > RH\_SUN\_score5.mipdesign

-discard all MIPs with arms overlapping SNPs

-discard all MIPs with ext\_copy\_count ot lig\_copy\_count > 8

-discard all MIPs with GC content < 35% or > 55% (this is a strict criterion but need not be, GC content between 30% and 60% is probably fine)

./calculate\_target\_GC mips.sequences > mips.gc

**11. Design MIPs.** I intentionally left this step to be done by hand to ensure quality and familiarity with the MIPs you are designing. This involves going through the list of good possible MIPs (sorted in Excel by target start position in the master sequence) and selecting those with target regions overlapping fixed SUNs (sorted by master sequence position). Ideally, you want a final set of MIPs distributed over the entire region of interest with good density and as even spacing as possible, having mostly high scores (mostly 5.0), with most target regions having distinguishing sequence for all paralogs and having multiple fixed SUNs, and with approximately equal numbers of MIPs targeting each strand (+ and -). You should record info on SUN content for each MIP target region in the final Excel design file.

**12. Perform MIP experiments.** Refer to the document “MIP\_protocol\_july2012.docx” in ~/MIPs/.

**13. Gzip qseq sequencing output text files.** Run the following command from the directory containing qseq.txt files.

bash ~/xander/MIPs/analysis\_programs/mrzip.sh all.q

or

bash ~/xander/MIPs/analysis\_programs/mrzip.sh prod.q

OLD METHOD (serial gzipping)

gzip \*

or

for i in $(ls|grep 'qseq.txt'); do gzip $i; done

To clean up after zipping (remove unzipped files and zip\_commands.txt), run

bash ~/xander/MIPs/analysis\_programs/cleanup\_zipping.sh

**14. Make a file containing the last 2 digits of all qseq text file names, with one entry per line.** For example, if you have 6 gzipped qseq text files named “s\_1\_1\_0001\_qseq.txt.gz”, “s\_1\_2\_0001\_qseq.txt.gz”, “s\_1\_3\_0001\_qseq.txt.gz”, “s\_1\_1\_0002\_qseq.txt.gz”, “s\_1\_2\_0002\_qseq.txt.gz”, and “s\_1\_3\_0002\_qseq.txt.gz”, your file should have “01” on the first line and “02” on the second line. Name this file “qseq\_file\_nums.txt”. (If you’re doing a MiSeq run, the file has already been made, so you can skip this step.)

**15. Convert gzipped qseq text files to gzipped fastq files and move them to an input sequence directory for mapping.** Look at the source code of “qseq\_to\_fastq\_for\_mrfast.c” in ~/MIPs/analysis\_programs/ for more information. Run the following command from the directory containing gzipped qseq files.

To convert all qseq text file reads to fastq format, run:

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_mrfast.sh all.q

or

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_mrfast.sh prod.q

To extract only reads having a barcode sequence perfectly matching a known barcode sequence, run:

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_mrfast\_extract.sh all.q path\_to\_barcodekey\_file/barcodekeyfile

or

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_mrfast\_extract.sh prod.q path\_to\_barcodekey\_file/barcodekeyfile

Cleanup the conversion (sometimes a few fastq.gz files won’t get properly moved)

mv \*.fastq.gz ../mrfast\_mapping\_input/

OLD METHOD (serial qseq to fastq conversion; these .sh files now have \_serial added to their names):

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_mrfast.sh

or to extract only reads having a barcode sequence perfectly matching a known barcode sequence, run:

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_mrfast\_extract.sh path\_to\_barcodekey\_file/barcodekeyfile

**16. Map reads with mrFAST.**

Make a fasta file containing all contigs you want to map to.

Make an index file corresponding to your “genome” of contigs.

mrfast --index contig\_genome.fasta

Then run:

qsub –q prod.q ~/xander/MIPs/analysis\_programs/mrmap\_mrfast.sh \

path\_to\_contig\_genome\_fasta/contig\_genome.fasta \

/var/tmp/xnuttle/ \

path\_to\_mrfast\_mapping\_input\_directory/ \ path\_to\_mrfast\_mapping\_output\_directory/

or

qsub –q all.q ~/xander/MIPs/analysis\_programs/mrmap\_mrfast.sh \

path\_to\_contig\_genome\_fasta/contig\_genome.fasta \

/var/tmp/xnuttle/ \

path\_to\_mrfast\_mapping\_input\_directory/ \ path\_to\_mrfast\_mapping\_output\_directory/

Old Example (when MIPs folder was in home directory)

qsub -q prod.q ~/MIPs/analysis\_programs/mrmap\_mrfast.sh \

/net/gs/vol2/home/xnuttle/MIPs/experiments/pos\_ctrl\_expt/SRGAP2\_seqs.fasta \

/var/tmp/xnuttle/ \

/net/gs/vol2/home/xnuttle/MIPs/experiments/xtest/mrfast\_mapping\_input/ \

/net/gs/vol2/home/xnuttle/MIPs/experiments/xtest/mrfast\_mapping\_output/

OLD METHOD (only works for all.q):

If you have mapped to this “genome” before, check to see whether your genome fasta file and index file are on all nodes (run from eeek):

cexec eeek: ls /var/tmp/xnuttle|less

If you have never mapped to this “genome” before or your genome fasta file and index file are no longer on all nodes, copy these files onto all nodes:

cexec eeek: mkdir /var/tmp/xnuttle

cexec eeek: cp path\_to\_fasta/genome.fasta /var/tmp/xnuttle

cexec eeek: cp path\_to\_index/genome.fasta.index /var/tmp/xnuttle

cexec eeek: chgrp -R eichlerlab /var/tmp/xnuttle

Set up a shell script to run to do the mapping, setting min = 150, max = 154, e=4, mincpu = number of gzipped fastq files in mapping input directory +1, maxcpu = 101:

mrmap -i fastq\_mapping\_input\_directory -o mapping\_output\_directory -g fasta\_file\_on\_nodes\_containing\_genome\_to\_map\_to -pe -min min\_insert\_size\_for\_mapping -max max\_insert\_size\_for\_mapping -e edit\_distance -mincpu minimum\_number\_of\_CPUs\_to\_use -maxcpu maximum\_number\_of\_CPUs\_to\_use -email email\_address -options "--maxoea 500 --seqcomp --outcomp"

Submit the generated shell script as a mapping job to the cluster:

qsub -q all.q mrmap.sh

Clean up the mapping and remove input files (run from the “mrfast\_mapping\_output” directory):

bash ~/xander/MIPs/analysis\_programs/cleanup\_mrfast\_mapping.sh

Make a file listing all mrfast mapping output files (run from the “mrfast\_mapping\_output” directory):

bash ~/xander/MIPs/analysis\_programs/makelist\_mrfast.sh

**17. Create a file containing computer-readable important information about each MIP.** Read the source code of “detail\_mip\_targets.c” in ~/MIPs/pos\_ctrl\_expt/ to understand how this is done and see an example. If the resulting output file already exists because you have analyzed data from an experiment using this MIP set before, you can skip this step.

**18. Parse the mapping output gzipped sam files to generate a file containing paralog-specific read counts for each individual-MIP combination.** Read the source code of “mrfast\_output\_to\_mipcounts.c” in ~/MIPs/analysis\_programs/ to understand how this is done and to see an example.

**19. Analyze and visualize the data in R.** See the code in “mip\_count\_analysis\_R\_code.txt” in my folder U:/pipeline\_directions/

**20. Map reads with bwa (for indel calling)**

Run from the directory containing gzipped qseq text files:

To convert all qseq text file reads to fastq format, run:

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_bwa.sh all.q

or

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_bwa.sh prod.q

To extract only reads having a barcode sequence perfectly matching a known barcode sequence, run:

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_bwa\_extract.sh all.q path\_to\_barcodekey\_file/barcodekeyfile

or

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_bwa\_extract.sh prod.q path\_to\_barcodekey\_file/barcodekeyfile

Cleanup the conversion (sometimes a few fastq.gz files won’t get properly moved)

mv \*.fastq.gz ../bwa\_mapping\_input/

OLD METHOD (serial qseq to fastq conversion; these .sh files now have \_serial added to their names):

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_bwa.sh

or to extract only reads having a barcode sequence perfectly matching a known barcode sequence, run:

bash ~/xander/MIPs/analysis\_programs/convert\_qseq\_to\_fastq\_for\_mrfast\_extract.sh path\_to\_barcodekey\_file/barcodekeyfile

Make an index file corresponding to your “genome” of contigs (run this from the directory containing your contig\_genome fasta file).

bwa index contig\_genome.fasta

Do the mapping (run from the “bwa\_mapping\_input” directory):

bash ~/xander/MIPs/analysis\_programs/mrmap\_bwa.sh all.q\ absolute\_path\_to\_contig\_genome.fasta/contig\_genome.fasta

or

bash ~/xander/MIPs/analysis\_programs/mrmap\_bwa.sh prod.q\ absolute\_path\_to\_contig\_genome.fasta/contig\_genome.fasta

Clean up the mapping and remove input files (run from the “bwa\_mapping\_output” directory):

bash ~/xander/MIPs/analysis\_programs/cleanup\_bwa\_mapping.sh

Make a file listing all bwa mapping output files (run from the “mrfast\_mapping\_output” directory):

bash ~/xander/MIPs/analysis\_programs/makelist\_bwa.sh

To run my C programs, just look at the first few lines to see how they should be called. For the visualization using my R script, first create a variable called base\_name that gets the name of your experiment (it must match up with analysis output file names prior to their extensions). Then call the R code (make sure all relevant input files (.mipcounts file, .simplecalls file, .compevnts file, .cncalls file) are in the R working directory):

base\_name<-“pos\_ctrl\_expt”

source(“pdf\_SRGAP2\_final.r”)

I have different visualization scripts for different figures and nicer plotting of paralog-specific absolute copy numbers rather than paralog-specific count frequencies, but most of these are conceptually similar.