**Supplemental methods: Identification of circRNA reads**

Existing methods to identify circRNAs from sequence data (Salzman et al. 2012; Memczak et al. 2013; Guo et al. 2014; Nair et al. 2016; Szabo and Salzman 2016) rely on using unmapped reads in a transcriptome BAM file, either by fragmenting the reads and realigning these shorter fragments to the reference genome (Memczak et al. 2013; Nair et al. 2016), or by aligning the unmapped reads to a custom database of exon-exon junctions (Salzman et al. 2012). In contrast, the method developed here takes a new position by taking advantage of the information provided by the STAR aligner for the mapped reads. When one part of a sequence-read maps to a location and the remaining part to another location in a manner not explained by canonical exon-intron-exon structure, the STAR mapper can assign a "Secondary Alignment" tag (SA) to such a read.

When using paired-end sequence data, and assuming a circular RNA molecule is present (top part Figure 1) the sequence read that aligns over the crossing of the non-canonical junction (green arrows) would 'point toward' its read-mate somewhere in the circle. Aligning these reads to the linear reference (middle part Figure 1), the junction read will get an SA tag if and only if this is the one and unique alignment the STAR software can find. Such a sequence read with a secondary alignment aligns to the reference on one location (here the 5' end of exon n) with e.g. 50 nucleotides while the remaining nucleotides of the read (in our cohort the read-length is 75 bases) align with the second location (3' end of exon n+1). The read-mate (orange arrow) aligns somewhere in between these two locations with the head of the read pointing toward the head of the junction read, as would be expected in a properly paired read. Finding additional read-pairs showing this configuration, but always with a breakpoint at the exact same location, strengthens the evidence for circular transcripts. A custom Perl script (next section in this document and https://bitbucket.org/snippets/MSmid/Le949d/identify-circularrna-reads) was written to obtain such a configuration of reads from the BAM files. It is crucial to only use uniquely mapped reads (MAPQ=255 in STAR notation), to circumvent the situation that the read that spans the circular junction has homology to other parts of the

genome as well. Of the sequence reads satisfying that criterion, the proper reads indicative for a circular region were teased out using the alignment locations, flags, CIGAR score (Concise Idiosyncratic Gapped Alignment Report) and tags available for each read in the BAM file. In full, 'normal' properly paired reads have flags 163 and 83, or flag 147 and 99, depending on orientation and strand (flags derived from the Sequence Alignment Map specification (Li et al. 2009)). When one of these reads contains a SA tag (Secondary Alignment), part of the read uniquely maps to one location while the remaining part of the read maps to another location, with a controversy between the orientation and location of the two parts of a read. If this is the case, the read gets an additional entry in the BAM file with the information for the secondary alignment for that part of the read, with the flag changed to 2129 (if the flag was 83), 2209 (for 163), 2193 (for 147) and 2145 (for 99). To identify a circular region, a trio of read alignments then must be present with flags in the following possible orders: 2129_163_83, 83_163_2129, 2209_83_163, 163_83_2209, 2193_99_147, 147_99_2193, 2145_147_99 and 99_147_2145. This does depend on the BAM file being ordered by chromosomal start location, but it is not required that these 3 read alignments are consecutively listed in the BAM file. The one additional constraint was that the trio of reads must all be mapped to the same chromosome. The start location of a region is defined as the alignment location of the first read in the trio and, since the alignment location in the BAM file is defined by the 5' end of the read, the end location of the circular region is obtained by using the alignment location of the last read of the trio and add the number of matches from the CIGAR score of that read. Only regions with at least 5 reads crossing the circular junction were included. Then, we verified if a read pair exists with one read crossing the junction (green arrow figure 1) with the read-mate positioned outside the candidate circular region. If the molecule is circular such read-pairs should not be present; thus, when such read-pairs are found, the region is removed for that sample. Regions were furthermore removed if the start and end coordinates were less than 175 bases apart, to avoid overlapping read-mates. Regions larger than 2.3 Mb were also excluded (size of biggest known gene *CNTNAP2*) to avoid identification of regions that are artefacts of the RNA-seq protocol (inaccurate ligation of distinct cDNAs or template switching

during cDNA synthesis may yield false-positive isoforms). Finally, regions annotated to pseudogenes were removed, as were those annotated to the immunoglobin variable chain regions. Next, the GENCODE annotation was used to obtain the exon locations of genes that exactly matched with the circular region. To quantify the abundance, it is important to note that in the RNA-seq protocol fragmentation of RNA is performed before the cDNA synthesis step. This removes the question of unwanted multiple copies of a circular RNA during cDNA synthesis, because through strand displacement the reverse transcriptase (RT) could generate a linear cDNA molecule containing multiple copies of the circle (Szabo and Salzman 2016) (similar to rolling circle amplification). For each sample, STAR also outputs the raw read counts for all genes. These were used to correlate with the number of junction reads of the circular transcripts found in a gene. To normalize between samples, the Trimmed Mean of M-values (TMM) implemented in edgeR (Robinson and Oshlack 2010) was used.

## References


Guo JU, Agarwal V, Guo H, Bartel DP. 2014. Expanded identification and characterization of mammalian circular RNAs. *Genome Biol* **15**: 409.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, Genome Project Data Processing S. 2009. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**: 2078-2079.

Memczak S, Jens M, Elefsinioti A, Torti F, Krueger J, Rybak A, Maier L, Mackowiak SD, Gregersen LH, Munschauer M et al. 2013. Circular RNAs are a large class of animal RNAs with regulatory potency. *Nature* **495**: 333-338.

Nair AA, Niu N, Tang X, Thompson KJ, Wang L, Kocher JP, Subramanian S, Kalari KR. 2016. Circular RNAs and their associations with breast cancer subtypes. *Oncotarget* **7**: 80967-80979.

Robinson MD, Oshlack A. 2010. A scaling normalization method for differential expression analysis of RNA-seq data. *Genome Biol* **11**: R25.

Salzman J, Gawad C, Wang PL, Lacayo N, Brown PO. 2012. Circular RNAs are the predominant transcript isoform from hundreds of human genes in diverse cell types. *PLoS One* **7**: e30733.

Szabo L, Salzman J. 2016. Detecting circular RNAs: bioinformatic and experimental challenges. *Nat Rev Genet* **17**: 679-692.

**Perl script to identify circularRNAs.**

```
# Author: Marcel Smid
# License: CC:BY:SA:NC (Creative Commons Attribution-ShareAlike-NonCommercial 4.0 International)
# Disclaimer: author is not professionally trained as programmer, and realizes TIMTOWTDI.
#
# find circular RNAs, using mapped reads with mate spanning the junction
# prerequisites:
#  DATA
#           -RNA processed without a poly(A) selection step
#           -paired-end sequence data
#           -mapped by the STAR aligning software (https://github.com/alexdobin/STAR)
#           -bam file sorted and indexed
#           -expects prefix 'chr' for chromosomes
#
#  Perl and other necessary files / tools
#           -package File::Find
#           -samtools (https://github.com/samtools/)
#           -Gencode gene annotation, gtf format (ftp://ftp.ebi.ac.uk/pub/databases/gencode/)
#
#
############       VERSION DEVELOPED FOR WINDOWS BOX, HARDCODED PATHS
############             - set lines 27-33


use File::Find;

#location of bam files (.bai needed!)
$dir = "C:/temp/";
#location of output files
$outdir = "C:/temp/";
#location of samtools
$samdir = "C:/temp/";
#location of Gencode GTF
$gtf = "C:/temp/gencode.v23.annotation.gtf";

# patterns of read-flags, as provided by SAM specifications
$pattern  = " 2129 2209 2193 2145";
$pattern2 = "83 163 147 99";

# needed for windows, otherwise samtools will not auto-close after each bam file
# optional change to wanted chromosomes only
$chr_string = "chr1 chr2 chr3 chr4 chr5 chr6 chr7 chr8 chr9 chr10 chr11 chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19 chr20 chr21 chr22 chrX";

#show print statements immediately
$|=1;

```

```
# annotation from Gencode, extract HAVANA exons information
# make sure the Gencode version matches the reference genome version used by STAR for mapping
open GTF, $gtf or die "Cannot find gtf file";
while ($line=<GTF>) {
	chomp $line;
	@fields = split "\t", $line;
	if ($fields[2] eq "exon" && $fields[1] eq "HAVANA") {
		#chr_exon start
		$uni = $fields[0]."_".$fields[3];

		#chr_exon end
		$uni2 = $fields[0]."_".$fields[4];

		#annot
		$temp = $fields[8];
		$temp =~ s/"//g;
		$temp =~ /gene_id (.+?);.+gene_name (.+?);.+exon_number (\d+?);/;
		# $ensembl = $1; $name = $2; $exon = $3;
		$annot = $1."\t".$2."\t".$3;

		unless (defined $gtf_annot{$uni}) {
			$gtf_annot{$uni} = $annot;
		}
		unless (defined $gtf_annot{$uni2}) {
			$gtf_annot{$uni2} = $annot;
		}
	}
}
print "GTF loaded\n";
find(\&process_file, $dir);


sub process_file {
	$open = $File::Find::name;

	#loop per bam-file
	if ($open =~ /\.bam$/) {
		$sample_name = $_;
		# optionally parse $_ to get a cleaner sample_name for output

		print "running ",$sample_name,"\n";

		# get reads using samtools, -F 4 for mapped reads only
		open IN, $samdir."samtools view -F 4 $open $chr_string |"  or die "cannot find input bam $open";
		open OUT, ">".$outdir.$sample_name."_mapped_circRNA.txt";

		#reset variables for new sample
		$start=time();$row=0;undef %wanted;undef %region; undef %region_flag;
		undef %flags; $correct_count=$regions_count=0;
```

```
LBL1:while ($line=<IN>) {
	$row++;
	if ($row%10000000==0) {print $sample_name,": ",$row,"\n";}

	@fields = split "\t", $line;
	# just to really make sure it doesn't parse MT if ppl remove $chr_string
	if ($fields[2] eq "chrM") {
		close IN;
		last LBL1;
		print "oops. parsing chrM is a bad idea\n";
	}

	#prevent match of flag 145 129
	$temp = " ".$fields[1];

	#check flag pattern for wanted reads, only use uniquely mapped (MAPQ==255)
	if ($pattern =~ /$temp/ && $fields[4]==255) {
		chomp $line;

		#fill hash by readname
		push @{$wanted{$fields[0]}}, $line;
		next LBL1;
	}
	if ($pattern2 =~ /$fields[1]/ && $fields[4]==255) {
		chomp $line;
		# contains proper pair flags; check for secondary alignment label and MAPQ of 255 (unique map)
		# in STAR mapped bam file, the secondary alignment is present in the last field
		# if data has been MarkDuplicate by Picard, then it is not the last field, but in 1 case field[11]
		# not sure if this field - after MarkDup by Picard - is universal

		# after STAR
		if ($fields[$#fields] =~ /^SA:Z/) {

		#after STAR & Markduplicate
		#if ($fields[11] =~ /^SA:Z/) {
			#$flag{$fields[1]}++;

			#fill hash by readname
			push @{$wanted{$fields[0]}}, $line;

			next LBL1;

		}
		# but also the third partner is needed; this read does not have secondary alignment
		# will only be needed/present if either the 21xx read has been seen, or the one with the SA label
		if (defined $wanted{$fields[0]}) {
				push @{$wanted{$fields[0]}}, $line;
		}
```

```
144                     }
145
146             }
147             $inter=time();print "all lines parsed in ",$inter-$start," seconds\n";
148
149             # reads indicative of circRNA have specific order of flags
150             # several possibilities depending on read orientations
151             $correct = " 2129_163_83 83_163_2129 2209_83_163 163_83_2209 2193_99_147 147_99_2193 2145_147_99 99_147_2145";
152
153             # now check for circular RNA
154             foreach $readname (keys %wanted) {
155                     # we need a trio
156                     if (scalar @{$wanted{$readname}} == 3) {
157                             # need certain combinations in the correct order, so get order of flags for this read
158                             $flag=" ";
159                             for $i (0..2) {
160                                     $temp=$wanted{$readname}[$i];
161                                     @fields=split "\t", $temp;
162                                     $flag .= $fields[1]."_";
163                                     # while we are here, save some data
164                                     if ($i==0) { $junction_left = $fields[3]; $chr_left=$fields[2];}
165                                     if ($i==1) { $chr_middle=$fields[2];}
166                                     if ($i==2) { $pos_right = $fields[3]; $cigar_right = $fields[5]; $SA = $fields[$#fields];}
167                             }
168                             #remove last _
169                             chop $flag;
170
171
172                             # check if this is a combi plus order indicative of circular read
173                             if ($correct =~ /$flag/) {
174                                     #check if all reads are on same chromosome
175                                     if ($SA =~ /SA:Z:(chr.+?),/) {
176                                             if ($1 eq $chr_left) {
177                                                     if ($chr_left eq $chr_middle) {
178                                                             $correct_count++;
179
180                                                             # gather region info for output
181                                                             # the left junction is the mapped position (5' start pos of read)
182                                                             # the right junction is the mapped position plus the number of matches
183                                                             # as listed by CIGAR. The CIGAR score always must be xxMyyH, or at least 1 or more digits followed by M
184                                                             $cigar_right =~ /(\d+)M/;
185                                                             $junction_right = $pos_right + $1 - 1;
186                                                             $uni = $chr_left.":".$junction_left."-".$junction_right;
187                                                             $region{$uni}++;
188                                                             $region_flag{$uni}{$flag}++;
189                                                             $flags{$flag}=1;
190                                                     }
191                                             }
192                                     }
```

```
193
194                                             }
195                                     }
196                             }
197                             print $correct_count," circular trio's found.\nFinding regions in gtf and output\n";
198
199                             # match with gencode
200                             # if no match, then still output
201                             # OUTPUT has region where circRNA has start and end-point (Chr-start-end),
202                             # the total number of reads supporting the junction, also specified by flag type
203                             # size of region (end-start), followed by annotation, if matching with HAVANA exons
204
205                             print OUT "Region\tTotal";
206                             foreach $flag (keys %flags) {
207                                     print OUT "\t",$flag;
208                             }
209                             print OUT "\tSize\tEnsemble_start\tGene_start\texon_number_start\tEnsemble_end\tGene_end\texon_number_end\n";
210
211                             foreach $region (keys %region) {
212                                     $regions_count++;
213                                     print OUT $region,"\t",$region{$region};
214                                     foreach $flag (keys %flags) {
215                                             print OUT "\t",$region_flag{$region}{$flag};
216                                     }
217                                     #split region for matchin with Encode
218                                     $region =~ /(chr.+):(\d+)-(\d+)/;
219                                     print OUT "\t",$3-$2,"\t";
220                                     $uni = $1."_".$2;
221
222                                     if (defined $gtf_annot{$uni}) {
223                                             print OUT $gtf_annot{$uni},"\t";
224                                     } else {
225                                             print OUT "\t\t\t";
226                                     }
227                                     $temp=$3;
228                                     $uni = $1."_".$temp;
229                                     if (defined $gtf_annot{$uni}) {
230                                             print OUT $gtf_annot{$uni},"\n";
231                                     } else {
232                                             print OUT "\n";
233                                     }
234                             }
235                             $end = time();
236                             print $regions_count," regions found.\nTotal runtime: ",$end-$start," seconds.\n\n";
237                             close;
238
239             } # end of bam loop
240
241     } # end of process dir loop
```