# Supplementary File 1

The following are the scripts that were used in this protocol

1. Prediction and differential expression analysis of circRNAs using CIRIquant

Install BWA, hisat2, stringTie and samtools:

**BWA**

$ wget https://jaist.dl.sourceforge.net/project/bio-bwa/bwa-0.7.17.tar.bz2

$ tar xvf bwa-0.7.17.tar.bz2

$ cd bwa-0.7.17.tar.bz2

$ make

$ export PATH=$PATH:/path/to/ bwa-0.7.17

$ source ~/.bashrc

**hisat2**

$ sudo apt-get -y install hisat2

**stringtie**

$ git clone https://github.com/gpertea/stringtie

$ cd stringtie

$ make release

**samtools**

$ wget https://codeload.github.com/samtools/samtools/zip/refs/heads/develop

$ Extract its contents and use the cd to enter the directory

$./configure **# Needed for choosing optional functionality**

$ make

$ make install

Index the host’s reference genome using BWA and HISAT2:

$ bwa index -p <name.given.to.output> </path/to/the/reference/genome>

$ hisat2-build </path/to/the/reference/genome> <name.given.to.output>

Execute CIRIquant:

$ sudo CIRIquant -l <library.type> -t <number.of.threads> -p <prefix.of.circRNA> -o <where.to.output.file> --config <path/to/yml/file> -1 </path/to/mate1/fastq> -2 </path/to/mate2/fastq>

Perform differential analysis on CIRIquant output result:

*$ prep\_CIRIquant -i <first.lst.file> --lib <output.library.info.csv.name> --circ <output.circRNA.info.csv.name> --bsj <output.circRNA.bsj.csv.name> --ratio <output.circRNA.ratio.csv.name>*

 $ *prepDE.py -i <second.lst.file>*

*$ sudo CIRI\_DE\_replicate --lib <library.info.csv> --bsj <circRNA.bsj.csv> --gene <gene.count.matrix.csv> --out <**output.circRNA.de.tsv.name>*

Filtering of DE circRNAs:

$R

**# Read the output file of CIRIquant**

>X <- read.table("/path/to/directory/of/output.circRNA.de.tsv.name ", sep=",", header = T)

>colnames(X)[colnames(X) == 'X'] <- 'CircRNA\_ID'

**#Change to numeric**

>X$logFC <- as.numeric(X$logFC)

>X$FDR <- as.numeric(X$FDR)

**# Identify DE**

>X\_DE <- X %>% filter(abs(logFC) > 2, FDR < 0.05)

>write.table(X\_DE, file = "DE\_circRNAs.txt", col.names = F, row.names = F ,quote = F, sep = "\t")

1. Characterisation and annotation of predicted DE circRNAs

Annotation status of DE circRNAs:

$R

**#Load the gtf files from CIRIquant into one dataframe**

>setwd(‘/path/to/directory/of/ CIRIquant/gtf/files’)

>file\_list <- list.files(pattern = ".gtf")

>ldf <- lapply(file\_list , rtracklayer::import)

>df <- ldply(ldf, data.frame)

>df <- distinct(df,circ\_id, .keep\_all= TRUE)

>df <- df[order(df$circ\_id),]

**#Merge the information from GTF files to the DE circRNAs**

>X\_DE <- read.table("/path/to/directory/of/DE\_circRNAs.txt”, sep=”,”, header = T)

>Full\_X <- df[which(df$circ\_id %in% X\_DE$CircRNA\_ID),]

>colnames(Full\_X)[colnames(Full\_X) == ‘circ\_id’] <- ‘CircRNA\_ID’

>Full\_merge <- merge(X\_DE,Full\_X,by=”CircRNA\_ID”) %>% select(CircRNA\_ID, logFC, Pvalue, DE, FDR, strand, circ\_type, everything())

>Full\_merge\_1 <- Full\_merge %>% tidyr::separate(CircRNA\_ID, into = c(“Chr”, “Start”, “End”), sep = “\t”)

>Full\_merge\_1$Start <- as.numeric(Full\_merge\_1$Start) – 1

>Full\_merge\_1$CircRNA\_ID <- paste0(Full\_merge\_1$Chr, “:”, Full\_merge\_1$Start, “|”, >Full\_merge\_1$End, Full\_merge\_1$strand, sep=””)

>write.table(Full\_merge, file = "/name/of/file.txt", col.names = F, row.names = F ,quote = F, sep = "\t")

**##Load circBase annotation (NOTE: In this case we are using the hg38 genome version)**

>Full\_merge <- read.delim2((‘/directory/path/to/circBase\_circRNA.txt', header = T, sep=”\t”)

>Y <- read.delim2(‘/directory/path/to/circBase\_circRNA.txt', header = F) %>% filter(V6 == "hg38")

>Y\_1 <- paste0(Y$V1, ":", Y$V2, "|", Y$V3) %>% as.data.frame()

>Y\_1 <- str\_trim(Y\_1$.) %>% as.data.frame()

>colnames(Y\_1) <- "CircRNA\_ID"

**##CircRNAs Annotation status labelling**

>Full\_merge\_1$Annotated <- NA

>for (i in 1:nrow(Full\_merge\_1)) {

 if (Full\_merge\_1$CircRNA\_ID[i] %in% Y\_1$CircRNA\_ID) {

 Full\_merge\_1$Annotated[i] <- "Annotated"

 } else {

 Full\_merge\_1$Annotated[i] <- "Non-Annotated"

 }

}

> write.table(Full\_merge\_1\_2,file= “name\_of\_file.txt”, sep = "\t", row.names = F, quote = F)

Characterisation of DE circRNAs:

Number of circRNAs according to circRNA types:

$R

>Circ\_type <- table(Full\_merge$circ\_type) %>% as.data.frame()

>write.table(Circ\_type,file= “name\_of\_file.txt”, sep = "\t", row.names = F, quote = F)

Number of genes circRNAs span (1 or >1):

$R

**#Calculate number of genes**

>PreNoG <- na.omit(Full\_merge$gene\_name) %>% as.data.frame()

>colnames(PreNoG) <- "gene\_name"

>NoG <- sapply(strsplit(PreNoG$gene\_name, ","), length) %>% as.data.frame()

>NoG\_1 <- NoG %>% mutate(category = cut(., breaks = c(0,1, Inf), labels=c("1", "> 1")))

>NoG\_2 = table(NoG\_1$category) %>% as.data.frame()

>write.table(NoG\_2,file= “name\_of\_file.txt”, sep = "\t", row.names = F, quote = F)

1. Predicting circRNA-miRNA interaction using Circr

Install miRanda, RNAhybrid, and Pybedtools:

*miRanda ($ conda install -c bioconda miranda / conda install -c bioconda/label/cf201901 miranda)*

*RNAhybrid ($ conda install -c genomedk rnahybrid)*

*Pybedtools ($ pip3 install pybedtools)*

Index the fasta file:

*$ samtools faidx <reference.genome.fa>*

Changes in circR.py file:

*In line 191, 238 and 245, replace “rm -rf” to “echo” to prevent the removal of miRNA prediction tool output files.*

Execute Circr:

*$ python3 </path/to/Circr.py> -i <`INPUT`.bed> --gtf </path/to/my.gene.anno.gtf> --genome </path/to/my.genome.ref. fa> --rRNA </path/to/my.rRNA.coord.bed> --miRNA </path/to/my.miRNA.seq.fa> --AGO </path/to/my.custom.AGO.peaks.bed> --validated\_interactions </path/to/my\_custom\_interactions.bed> -o example\_custom\_anno\_circr.csv*

Filtering step:

**# Load circR output file in Rstudio**

**# Filter according to circRNA name, number of software matched, seed category, AGO and validation**

$R

>seeds <- c("No Seed", "SM", "6mer", "Off-6mer")

>X <- read.table("/directory/path/to/circR/output/file", sep=",", header = T) %>% filter(Circ.Name == "circRNA\_of\_interest") %>% filter(Software.Matched == "3") %>% filter(!Seed.Category %in% seeds) %>% filter(AGO == "Yes" | Validated == "Yes")

**# Filter according to the number of binding sites from TargetScan and miRanda**

**##Load Targetscan output file and add the number of binding sites**

>X\_1 <- read.table("TS\_TMP\_(file).txt", sep = "\t", header = T)

>X\_1$miRNA\_family\_ID\_1 <- paste("hsa-", X\_1$miRNA\_family\_ID, sep = "")

>X\_1\_table <- table(X\_1$miRNA\_family\_ID\_1) %>% as.data.frame()

>Z <- X\_1\_table[which(X\_1\_table$Var1 %in% X$miRNA.Name),]

>Z <- Z[match(X$miRNA.Name, Z$Var1),]

>X$TS\_Freq <- Z$Freq

**## Parse and load miRanda output file to add the number of binding sites**

$ grep ">>" mir\_TMP\_\*.txt | sed 's/>>//g' > parsed\_miRanda\_filename

$ R

>X\_2 <- read.delim("parsed\_miRanda\_filename ", header = F)

>colnames(X\_2) <- c("Seq1","Seq2","Tot Score","Tot Energy","Max Score","Max Energy","Strand","Len1","Len2","Positions")

>X\_2$Number\_ofBindingSites <- sapply(strsplit(X\_2$Positions, " "), length)

>X\_2$Number\_ofBindingSites <- as.numeric(X\_2$Number\_ofBindingSites) - 1

>Z\_1 <- X\_2[which(X\_2$Seq1 %in% X$miRNA.Name),]

>Z\_1 <- Z\_1[match(X$miRNA.Name, Z\_1$Seq1),]

>X$mir\_Freq <- Z\_1$Number\_ofBindingSites

**## Filter the number of binding sites and write the circRNA-miRNA file**

>X <- X %>% filter(TS\_Freq >= 2) %>% filter(mir\_Freq >= 2)

>write.table(X, file = "name\_of\_file.txt", col.names = T, row.names = F ,quote = F, sep = "\t")

4. For the GO and KEGG enrichment of mRNA

Use Rstudio to convert the mRNAs Ensembl ID to EntrezID by:

$ R

>circ\_GO <- read.delim2((‘/directory/path/to/circBase\_circRNA.txt', header = T, sep=”\t”)

>gene.df\_mRNA <- bitr(circ\_GO$gene\_name, fromType = "ENTREZID", toType = c "ENSEMBL","ENTREZID"), OrgDb = org.Hs.eg.db)

Run GO and KEGG enrichment:

$ R

**#GO**

>GO\_res <-enrichGO(gene.df\_mRNA $ENTREZID, keyType = "ENTREZID", org.Hs.eg.db, ont = "All", pvalueCutoff = 0.01, pAdjustMethod = “BH”, qvalueCutoff = 0.05, readable=F)

**#KEGG**

>keg\_res <- enrichKEGG(gene.df\_mRNA $ENTREZID, organism="hsa", keyType="kegg", pvalueCutoff = 0.05, pAdjustMethod = "BH", qvalueCutoff = 0.05)

**##Convert to entrez ID to gene names**

>keg\_res <- setReadable(keg\_res, OrgDb = org.Hs.eg.db, keyType="ENTREZID")