Uncovering a role for transcribed ultra-conserved regions as long non-coding RNAs in gliomas

Myron Gibert Jr

January 27, 2020

Contents

Abstract **3**

Introduction **3**

Setup . . . 4

Set parameters . . . 4

Install required programs . . . 4

Create output directory . . . 7

Section 1: Transcribed Ultra-conserved Regions (TUCRs) are a broad class of molecules **7**

Methodology . . . 7

Generate Figure 1E for TUCR annotation . . . 7

Acquiring SNPs . . . 8

Getting SNP intersects for genes . . . 10

Count intersecting SNPs . . . 11

Generate SNP proportion graph . . . 12

Aquire the data using the SRA toolkit . . . 12

Generate fastq files . . . 13

Build hg38 genome . . . 13

Align fastq files to reference genome . . . 13

Convert sam to bam . . . 13

Index bam files . . . 13

Download mouse genes and convert to bed . . . 14

Download Genome Annotation file . . . 14

Generate Random Intervals and sort all files . . . 14

Find closest Pol.II binding site using bedtools closest . . . 15

Find closest H3K4me3 binding site using bedtools closest . . . 15

Graphing closest H3K4me3 marks . . . 15
Graph closest RNA Pol.II Binding Sites . . . 17
Generate TUCR "Windows" . . . 19
Fisher . . . 19
Show that TUCRs are enriched for hallmarks of lncRNAs . . . 20

Section 2: TUCRs are deregulated in GBM and LGG tumors **21**
Results . . . 21
Setup for TCGA RNA-Seq analysis . . . 21
Extract headers from BAM files . . . 21
Extract TCGA patient barcodes from BAM headers . . . 22
Reindex TCGA BAM files under new names . . . 23
Generating count tables for survival and differential expression. . . . 24
Acquiring total reads from RNA-Seq BAM files . . . 24
Identifying differentially expressed TUCRs with DESeq2 . . . 25
Writing a script to generate a volcano plot . . . 26
Generating volcano plot to visualize DE genes . . . 27
Volcano plot for intergenic TUCRs only . . . 27
Determining sequencing depth for TCGA files . . . 28
Identifying and comparing TUCR absolute expression in GBM and LGG . . . 28
Generate RPKM Box Plots for each TUCR . . . 31

Section 3: TUCR expression correlates with survival in LGG and GBM **31**
Results . . . 31
Methodology . . . 32
Acquiring and parsing clinical survival data . . . 32
Completing survival analysis for TUCRs . . . 32
Summarize Survival Data . . . 34
Writing a script to generate a volcano plot for survival . . . 35
Draw Volcano plot for TUCRs . . . 36

Section 4: TUCRs are coregulated with genes that have known functions **37**
Results . . . 37
Methodology . . . 37
Identify DE TUCRs that are correlated with each other . . . 37
Generate GO Terms for coregulated genes . . . 39
Summarize GO term data for oncogenes and tumor suppressors . . . 41
Summarize GO Term Data (new) . . . 42

Section 5: TUCRs are contained within mRNA or lncRNA transcripts **45**

Results 45

Methodology 45

Stringtie GBM 45

Stringtie LGG 46

Show that TUCRs are contained within novel genes 47

Abstract

Gliomas represent the most common brain tumors. Particularly, glioblastoma (GBM) is the most common and most deadly malignant brain tumor. Most glioma research has focused on protein-coding genes and much less on the non-coding transcripts that make up 98% of cellular RNA. Transcribed Ultra-Conserved Regions (TUCRs) represent an understudied class of molecules that are found conserved across multiple species. These transcripts are highly resistant to variation and are commonly deregulated in cancer, suggesting regulatory and functional importance. Intergenic TUCRs specifically represent potential novel protein coding and non-coding transcripts, as they do not overlap with known genes. We performed the first analysis of TUCRs in glioblastoma and low grade glioma (LGG). Previously established methodologies were used to annotate TUCRs and confirm that they are transcribed. Then, differentially expressed (DE) TUCRs were identified in GBM (n = 197) and LGG (n = 149), along with those who correlate with survival in GBM (n = 36) and LGG (n = 167). TUCRs were then scored by their absolute and relative expression in GBM and LGG, frequency of deregulation, and correlation with survival. These scores provide a means for prioritizing TUCRs for future experiments. Using a GBM and LGG RNA-Seq data to generate a list of co-regulated genes, a guilt-by-association analysis was performed to determine what biological processes and molecular functions may be shared by TUCRs and their co-regulated genes. Lastly, as TUCRs represent fractions of larger host transcripts, we sought to identify and characterize the parent transcripts of intergenic TUCRs. We propose that these host transcripts represent novel genes, with the TUCR itself serving as a component of these transcripts. The results of this study provide an insight into several exciting future research directions.

Introduction

Gliomas account for approximately 80% of all brain tumors. In particular, glioblastoma (GBM) accounts for about half of all primary brain and central nervous system cancers. It is also one of the most-deadly cancers. One-, five-, and ten-year survival rates for patients with GBM are 37 percent, 5%, and 3%, respectively. Although there has been extensive research on protein coding genes and pathways in GBM and low grade gliomas (LGG), the current standard of care involves surgical resection and radiation, with limited targeted therapies.

Many targeted therapies in the context of GBM, LGG, and other malignancies focus on protein-coding genes. And yet, these genes make up only a small portion of the transcriptome; ~90% of the genome is transcribed, but only ~2 percent of the transcriptome is then translated. The remainder of the transcriptome is represented by non-coding elements that can serve key regulatory roles. Of these elements, long non-coding RNAs (lncRNAs) have recently been determined to serve as important regulators of malignancy and potential therapeutic targets in cancer. These lncRNAs are transcribed RNA molecules that contain greater than 200 nucleotides but typically lack significant protein coding capabilities.

There are several genetic regions that are "ultra-conserved" across species (UCR). From these regions, Transcribed ultra-conserved regions (TUCRs) represent 481 unique transcribed regions of the genome that are greater than 200 base pairs in length and are highly conserved across multiple species, including human, mouse (100%), rat (100%), dog (98%), and chicken (95%) genomes. Due to their size (>200 nt) and lack of known associated protein products, TUCRs may function as lncRNAs. The existence of highly conserved

lncRNAs is significant, as lncRNAs are typically poorly conserved as a class of molecules. TUCRs are highly resistant to variation, with the 106,767 conserved bases overlapping with as few as six canonical single nucleotide polymorphisms (SNPs). Their expression is commonly deregulated in cancer.

Because of this, it is believed that these TUCRs may serve crucial regulatory roles in cancer. The non-coding RNA field is a growing one. While only a single lncRNA paper was published in 2007, that number has exploded to 1342 publications by the year 2016. Despite this explosion of interest in lncRNAs, very little is known of TUCRs. In particular, the literature elucidating the expression, functions and mechanisms of action of TUCRs in GBM and/or LGG is nonexistent. Studying them therefore represents an untouched avenue for understanding novel oncogenic mechanisms and discovering new biomarkers and therapeutic targets for these diseases.

Here, we utilize contemporary and novel bioinformatics techniques to perform the first characterization of TUCR expression, function, and mechanism of action in GBM and LGG. First, TUCRs were annotated and validated to confirm that annotations from 2010 are consistent with more recent databases. Then, programming tools were used to identify TUCRs that are differentially expressed in GBM and LGG tissues. TUCRs that are correlated with GBM and LGG patient survival were also identified. Then, a guilt-by-association analysis was performed to predict TUCR functional roles using coregulated genes. Lastly, we predict that intergenic TUCRs may represent markers for novel genes. To test this hypothesis, de novo transcript reassembly was used to identify transcripts present in RNA-Seq datasets in a genome agnostic manner. Selection criteria were then used to identify potentially novel genes from the resulting dataset.

Setup

Set parameters

This initial section is used to set the dependent variables that are used for this analysis.

The following variables can be set: * countfilename is the name of the RNA-Seq feature count file that is generated in the "Generating count tables for survival and differential expression" section. This can be any tab-delimited text file where the row names are the genes, the column names are the samples, and the cells are the generated counts. This table is used for both survival and differential expression analyses. * makekpmplots is a logical vector. When set to TRUE, the script will generate Kaplan-Meier survival plots for each gene in the counts file. * makerpkmoxplots is also a logical vector. When set to TRUE, the script will generate reads per kilobase million boxplots for each gene, showing median RPKM, mean RPKM (red dot), and additional summary information. This is used to determine to what extent a gene is expressed across all samples * repel is also a logical vector. When set to true, the volcano summary plot for RNA-Seq expression data will contain labels for a given set of genes specified in the "Generating volcano plot to visualize DE genes" * outputdir is a character variable containing the "output directory", or the location where all output files will go. This can be any characters enclosed in quotations marks (ex. "Outputs"). Set to "Outputs" to use the default outputs folder * useintermediate is a logical vector. When set to TRUE, the analysis will use a pregenerated gene correlation matrix for the guilt by association analysis. Unless there is a specific reason to generate a new matrix (VERY memory intensive), this variable should be set to TRUE. * deleteoutputs is a logical vector that should only be set to TURE or FALSE. When set to true, the output directory will be repopulated after every program run. If set to false, the program will only run if the output directory does not already exist in the working directory, to prevent the data from being overwritten.

Install required programs

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, message=FALSE,eval = FALSE)

if (!require("tidyverse")) install.packages("tidyverse")
library("tidyverse")
```

```r
if (!require("ggplot2")) install.packages("ggplot2")
library("ggplot2")

if (!require("ggforce")) install.packages("ggforce")
library("ggforce")

if (!require("ggrepel")) install.packages("ggrepel")
library("ggrepel")

if (!require("ggfortify")) install.packages("ggfortify")
library("ggfortify")

if (!require("ggdendro")) install.packages("ggdendro")
library("ggdendro")

if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")
if (!requireNamespace("DESeq2", quietly = TRUE))BiocManager::install("DESeq2")
library("DESeq2")

if (!require("survival")) install.packages("survival")
library("survival")

if (!require("broom")) install.packages("broom")
library("broom")

if (!requireNamespace("limma", quietly = TRUE))BiocManager::install("limma")
library("limma")

if (!require("Tmisc")) install.packages("Tmisc")
library("Tmisc")

if (!require("survminer")) install.packages("survminer")
library("survminer")

if (!require("corrr")) install.packages("corrr")
library("corrr")

if (!require("here")) install.packages("here")
library("here")

if (!require("lessR")) install.packages("lessR")
library("lessR")

if (!require("splitstackshape")) install.packages("splitstackshape")
library(splitstackshape)

if (!requireNamespace("BiocManager", quietly = TRUE)) install.packages("BiocManager")

if (!require('org.Hs.eg.db')) BiocManager::install('org.Hs.eg.db')
library(org.Hs.eg.db)

if (!require('GO.db')) BiocManager::install('GO.db')
library(GO.db)
```

```
if (!require('limma')) BiocManager::install('limma')
library(limma)

if (!require("ggthemes")) install.packages('ggthemes', dependencies = TRUE)
library(ggthemes)

if (!require("gplots"))
install.packages("gplots")
library(gplots)

if (!require("heatmap.plus"))
install.packages("heatmap.plus")
library(heatmap.plus)

if (!require("RColorBrewer"))
install.packages("RColorBrewer")
library(RColorBrewer)

###

disease <- "LGG"

outputdir <- "Outputs_LGG"

makeSNP <- TRUE

makeannotate <- TRUE

makelncRNA <- TRUE

makeDE <- TRUE

countfilename <- "lgg_TUCR_counts.txt"

metadatafile <- "lgg_tcga_metadata.csv"

repel <- TRUE

makesurvival <- TRUE

mergedcountfile <- "lgg_mergedcounts.txt"

seqdepthfile <- "lgg_seqdepth.csv"

makeSuvivalplots <- TRUE

makeRPKMboxplots <- TRUE

makecorrelations <- TRUE

useintermediate <- TRUE

intermediatefile <- "lgg_matrixintermediate.Rdata"
```

```
makeGOterms <- TRUE
```

Create output directory

This script checks to see if the defined output directory already exists. If it doesn't exist, it will create the directory.

```
if(!dir.exists(outputdir)){
  dir.create(outputdir)
}
```

Section 1: Transcribed Ultra-conserved Regions (TUCRs) are a broad class of molecules

##Results

TUCRs are a broad class of molecules spanning the entire human genome. Many TUCRs are contained within coding and non-coding transcripts. Some TUCRs are exonic and are contained within an exon of the "host" gene (Figure 1A). Others are contained within an intron instead (Figure 1B). Some TUCRs straddle a region that spans exonic and intronic regions of the host gene (intronic/exonic) (Figure 1C), and others are not contained within a known genetic element at all (intergenic) (Figure 1D). Each of the 481 TUCRs were previously annotated in 2010. Here, we provide an updated annotation that includes scientific discoveries made in the intervening decade (Figure 1E). Through manual annotation, we have identified 46 exonic, 150 intronic, 67 intronic/exonic, and 218 intergenic TUCRs. The full list of these annotations is available as a supplement (Supplementary Table 1)

It is critical that we confirm that TUCRs are transcribed to an extent that is comparable to known protein and non-coding genes. Much of the genome is transcribed, but a relatively small fraction represents functional transcripts and genes. The rest is considered transcriptional "noise". We sought to confirm that TUCRs do not represent transcriptional noise, but functional genomic units.

To analyze the transcription of TUCRs, we evaluated the local chromatin landscape and compared it to that of protein coding genes, non-coding RNAs, and randomized control genomic intervals. We specifically wanted to focus on H3K4me3, a marker for open chromatin, and RNA Pol.II, which is involved in the transcription of many non-coding RNAs. Using GBM U87 CHIP-Seq data, we performed two separate analyses of epigenetic density.

First, publicly available U87 H3K4me3 and RNA Pol.II CHIP-Seq data were downloaded and processed. Then, we identified the nearest epigenetic marker to each TUCR and compared these results to randomized control intervals of a similar length, protein coding genes, and various non-coding elements. These analyses were performed using H3K4me3 (Figure 2A) and RNA Pol.II (Figure 2B) binding sites. Then, we considered the density of H3K4me3 and RNA Pol.II binding within a 10 kilobase (kb) window up- and downstream of each class of gene (Control, TUCR, lncRNA, and mRNA). Fisher's exact test was used to confirm that the chromatin landscape is more significantly enriched for H3K4me3 and RNA Pol.II than a random control interval and is more consistent with coding and non-coding genes. (Figure 2C)

Methodology

Generate Figure 1E for TUCR annotation

```
if(makeannotate == TRUE){
tucr_annot <- read.csv("tucr_annot.csv")

plot_annotation <- tucr_annot %>%
  group_by(annot) %>%
  summarise(count = n()) %>%
  mutate(percent = count/sum(count)) %>%
  arrange(desc(count)) %>%
  mutate(lab.ypos = cumsum(count) - 0.5*count)

names(plot_annotation)[names(plot_annotation) == 'annot'] <-
  'Annotation'

p <- ggplot(plot_annotation, aes(x=2,y=count,fill=Annotation))          +
  geom_bar(
    stat="identity",colour="black",
    width = 1)                                                          +
  geom_text(
    aes(y = lab.ypos, label = count),
    color = "black")                                                    +
  coord_polar("y", start=0)                                             +
  scale_fill_manual(values=c("#F8766D","#00BFC4","#7CAE00","#C77CFF"))  +
  ggtitle(paste("TUCR Genomic Location"))                               +
  theme(
    plot.title = element_text(size=rel(1.0),
    face="bold",hjust = 0.5,vjust=-10),
    legend.title.align=0.5,
    axis.title = element_blank(),
    axis.text.x=element_blank(),
    axis.text.y=element_blank(),
    axis.line=element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.ticks=element_blank(),
    panel.background = element_blank())                                 +
  xlim(0.5, 2.5)

p

ggsave(file=paste(outputdir,"/tucr_annotation.png",sep=""),
       plot = print(p),
       width = 5,
       height = 5,
       dpi = 300)
}
```

Acquiring SNPs

```
mkdir SNPs

cd SNPs
```

```
wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_1.bed.gz
gunzip bed_chr_1.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_10.bed.gz
gunzip bed_chr_10.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_11.bed.gz
gunzip bed_chr_11.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_13.bed.gz
gunzip bed_chr_13.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_12.bed.gz
gunzip bed_chr_12.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_14.bed.gz
gunzip bed_chr_14.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_15.bed.gz
gunzip bed_chr_15.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_16.bed.gz
gunzip bed_chr_16.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_17.bed.gz
gunzip bed_chr_17.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_18.bed.gz
gunzip bed_chr_18.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_19.bed.gz
gunzip bed_chr_19.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_2.bed.gz
gunzip bed_chr_2.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_20.bed.gz
gunzip bed_chr_20.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_21.bed.gz
gunzip bed_chr_21.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_22.bed.gz
gunzip bed_chr_22.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_3.bed.gz
gunzip bed_chr_3.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_4.bed.gz
gunzip bed_chr_4.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_5.bed.gz
gunzip bed_chr_5.bed.gz
```

```
wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_6.bed.gz
gunzip bed_chr_6.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_7.bed.gz
gunzip bed_chr_7.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_8.bed.gz
gunzip bed_chr_8.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_9.bed.gz
gunzip bed_chr_9.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_X.bed.gz
gunzip bed_chr_X.bed.gz

wget ftp://ftp.ncbi.nih.gov/snp/organisms/human_9606_b151_GRCh38p7/BED/bed_chr_Y.bed.gz
gunzip bed_chr_Y.bed.gz
```

Getting SNP intersects for genes

```
### TUCRs ###
for bed in bed_chr_*
do

name=$(echo $bed | awk -F ".bed" '{print $1}')
echo $name

intersectBed -a hg38.ultraConserved.bed -b $bed -wa -wb > $name.TUCR.SNPs.bed

done

cat *.TUCR.SNPs.bed > merged_TUCR_SNPs.bed

rm *TUCR.SNPs.bed

### coding genes ###
for bed in bed_chr_*
do

name=$(echo $bed | awk -F ".bed" '{print $1}')
echo $name

intersectBed -a CHESScoding.bed -b $bed -wa -wb > $name.coding.SNPs.bed

done

cat *coding.SNPs.bed > merged_coding_SNPs.bed

rm *coding.SNPs.bed

### lncRNAs ###
```

```
for bed in bed_chr_*
do

name=$(echo $bed | awk -F ".bed" '{print $1}')
echo $name

intersectBed -a CHESSlncRNA.bed -b $bed -wa -wb > $name.lncRNA.SNPs.bed

done

cat *lncRNA.SNPs.bed > merged_lncRNA_SNPs.bed

rm *lncRNA.SNPs.bed

### antisense RNAs ###

for bed in bed_chr_*
do

name=$(echo $bed | awk -F ".bed" '{print $1}')
echo $name

intersectBed -a CHESSantisense.bed -b $bed -wa -wb > $name.antisense.SNPs.bed

done

cat *antisense.SNPs.bed > merged_antisense_SNPs.bed

rm *antisense.SNPs.bed

### miscRNAs ###
for bed in bed_chr_*
do

name=$(echo $bed | awk -F ".bed" '{print $1}')
echo $name

intersectBed -a CHESSmisc.bed -b $bed -wa -wb > $name.misc.SNPs.bed

done

cat *misc.SNPs.bed > merged_misc_SNPs.bed

rm *misc.SNPs.bed
```

Count intersecting SNPs

```
wc -l merged_TUCR_SNPs.bed

wc -l merged_coding_SNPs.bed
```

```
wc -l merged_lncRNA_SNPs.bed

wc -l merged_antisense_SNPs.bed

wc -l merged_misc_SNPs.bed
```

Generate SNP proportion graph

```
if(makeSNP == TRUE){
tucr_SNP <- read.csv("tucr_SNPs.csv")

p <- ggplot(tucr_SNP, aes(x=reorder(annot,order),y=proportion*100,fill=reorder(annot,order))) +
  geom_bar(stat="identity", width = 0.7,
    color = "black") +
  geom_text(aes(label=round(proportion*100,3)), vjust=1.6, color="black", size=7) +
  scale_fill_manual(values=c("red3","green4","blue3","deeppink","yellow2","orange2")) +
  ggtitle(paste("TUCRs are less susceptible to variation than other genes")) +
  xlab("gene annotation") +
  ylab("Percent variant nucleotides (SNPs)") +
  theme(
    plot.title = element_text(size=rel(1.0), face="bold",hjust = 0.5),
    axis.title = element_text(size = rel(1.0), face="bold"),
    legend.title = element_blank(),
    legend.position = "none",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_line(colour = "black"))

ggsave(file=paste(outputdir,"/",disease,"_SNPGraph.png",sep=""),
       plot = print(p),
       width = 5,
       height = 6,
       dpi = 300)
}
```

Aquire the data using the SRA toolkit

```
#using SRA Toolkit version 2.10.5

#Transcriptional Amplification in Tumor Cells with Elevated c-Myc
#https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE36354


#CHIP-Seq U87-H3K4me3

prefetch SRR444442

#CHIP-Seq U87-RNAPolII
```

```
prefetch SRR444478
```

Generate fastq files

```
for sra in SRR*
do
echo $sra
fastq-dump $sra
done
```

Build hg38 genome

```
wget https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz
gunzip hg38.fa.gz

#Rivanna only.  Skip otherwise.
module load gcc/7.1.0
module load bowtie2/2.2.9

bowtie2-build hg38.fa hg38
```

Align fastq files to reference genome

```
for fq in *.fastq
do
name=$(echo $fq | awk -F".fastq" '{print $1}')
echo $name
bowtie2 -x hg38 -U $fq -S $name.sam
done
```

Convert sam to bam

```
for sam in *.sam
do
name=$(echo $sam | awk -F".sam" '{print $1}')
echo $name
samtools view -b $sam | samtools sort -o $name.bam
done
rm *sam
```

Index bam files

```
for bam in *bam
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
samtools index $bam
done
```

Download mouse genes and convert to bed

I got this methodology from here since my other methods were not working.

```
macs2 callpeak -t SRR444442.bam -n SRR444442 -g hs
macs2 callpeak -t SRR444478.bam -n SRR444478 -g hs

wc -l *Peak

cat SRR444442_peaks.narrowPeak | cut -f 1-4 | grep -v KI | grep -v MT | grep -v GL | grep -v JH | grep
cat SRR444478_peaks.narrowPeak | cut -f 1-4 | grep -v KI | grep -v MT | grep -v GL | grep -v JH | grep
```

Download Genome Annotation file

```
wget ftp://ftp.ensembl.org/pub/release-90/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembl

gunzip Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz

samtools faidx Homo_sapiens.GRCh38.dna.primary_assembly.fa

awk -v OFS='\t' {'print $1,$2'} Homo_sapiens.GRCh38.dna.primary_assembly.fa.fai > hg38_genomeFile.txt

cat hg38_genomeFile.txt | grep -v KI | grep -v MT | grep -v GL | sed -e "s/^/chr/g" > temp

mv temp hg38_genomeFile.txt
```

Generate Random Intervals and sort all files

```
bedtools shuffle -i pol2.bed -g hg38_genomeFile.txt > pol2shuff.bed
bedtools sort -i pol2shuff.bed > pol2shuff.sort.bed

bedtools shuffle -i h3k4me3.bed -g hg38_genomeFile.txt > h3k4me3shuff.bed
bedtools sort -i h3k4me3shuff.bed > h3k4me3shuff.sort.bed

bedtools sort -i pol2.bed > pol2.sort.bed

bedtools sort -i h3k4me3.bed > h3k4me3.sort.bed

bedtools sort -i hg38.ultraConserved.bed > hg38.ultraConserved.sort.bed
```

```
bedtools sort -i TUCR_intergenic.bed > TUCR_intergenic.sort.bed

bedtools sort -i CHESScoding.bed > CHESScoding.sort.bed

bedtools sort -i CHESSlncRNA.bed > CHESSlncRNA.sort.bed

bedtools sort -i CHESSantisense.bed > CHESSantisense.sort.bed

bedtools sort -i CHESSmisc.bed > CHESSmisc.sort.bed
```

Find closest Pol.II binding site using bedtools closest

```
bedtools closest -a hg38.ultraConserved.sort.bed -b pol2shuff.sort.bed -g hg38_genomeFile.txt -d > close

bedtools closest -a hg38.ultraConserved.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt -d > closest_p

bedtools closest -a TUCR_intergenic.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt -d > closest_pol2_

bedtools closest -a CHESScoding.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt -d > closest_pol2_codi

bedtools closest -a CHESSlncRNA.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt -d > closest_pol2_lncR

bedtools closest -a CHESSantisense..sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt -d > closest_pol2_

bedtools closest -a CHESSmisc.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt -d > closest_pol2_misc_o
```

Find closest H3K4me3 binding site using bedtools closest

```
bedtools closest -a hg38.ultraConserved.sort.bed -b h3k4me3shuff.sort.bed -g hg38_genomeFile.txt -d > c

bedtools closest -a hg38.ultraConserved.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt -d > closes

bedtools closest -a TUCR_intergenic.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt -d > closest_h3

bedtools closest -a CHESScoding.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt -d > closest_h3k4me

bedtools closest -a CHESSlncRNA.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt -d > closest_h3k4me

bedtools closest -a CHESSantisense.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt -d > closest_h3k

bedtools closest -a CHESSmisc.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt -d > closest_h3k4me3_
```

Graphing closest H3K4me3 marks

```
filenames.H3K4me3 <- list.files("./closest_bed/H3K4me3/")
```

```
mastertable.H3k4me3 <- data.frame(matrix(ncol=11,nrow=0, dimnames=list(NULL,c("chromosome.gene","start.g
"chromosome.H3K4me3","start.H3K4me3","end.H3K4me3","H3K4me3","dist"))))

i = 1
for(i in 1:length(filenames.H3K4me3)){
  print(i)
  df <- read.table(paste("./closest_bed/H3K4me3/",filenames.H3K4me3[i],sep=""))
  colnames(df) <- colnames(mastertable.H3k4me3)
  mastertable.H3k4me3 <- rbind(mastertable.H3k4me3,df)
}

mastertable.H3k4me3.sum <- mastertable.H3k4me3 %>%
  mutate(interval = ifelse(dist<500, 500,
                           ifelse(dist<1000, 1000,
                           ifelse(dist<1500, 1500,
                           ifelse(dist<2000, 2000,
                           ifelse(dist<2500, 2500,
                           ifelse(dist<3000, 3000,
                           ifelse(dist<3500, 3500,
                           ifelse(dist<4000, 4000,
                           ifelse(dist<4500, 4500,
                           ifelse(dist<5000, 5000,
                           ifelse(dist<5500, 5500,
                           ifelse(dist<6000, 6000,
                           ifelse(dist<6500, 6500,
                           ifelse(dist<7000, 7000,
                           ifelse(dist<7500, 7500,
                           ifelse(dist<8000, 8000,
                           ifelse(dist<8500, 8500,
                           ifelse(dist<9000, 9000,
                           ifelse(dist<9500, 9500,
                           ifelse(dist<10000, 10000,10500))))))))))))))))))))) %>%
  filter(interval != 10500) %>%
  mutate(order = ifelse(annot=="protein_coding", 1,
                           ifelse(annot=="lncRNA", 2,
                           ifelse(annot=="TUCR", 3,
                           ifelse(annot=="intergenic_TUCR", 7,
                           ifelse(annot=="antisense_RNA", 4,
                           ifelse(annot=="misc_RNA", 5,
                           ifelse(annot=="random", 6,0
                           )))))))) %>%
  mutate(annot = fct_reorder(annot,order)) %>%
  group_by(annot,interval,order) %>%
  summarise(total = n()) %>%
  ungroup() %>%
  group_by(annot) %>%
  mutate(div = sum(total)) %>%
  mutate(prop = total/div) %>%
  mutate(log10 = log10(total)) %>%
  mutate(log2 = log2(total)) %>%
  arrange(order)

p <- ggplot(mastertable.H3k4me3.sum, aes(x=interval,y=log2,fill=annot,order=order)) +
```

```
  geom_bar(stat="identity",
    color = "black")                                                    +
  scale_fill_manual(values = c("red3","green4","blue3","deeppink","yellow2","orange2")) +
  xlab("distance (nt)")                                                  +
  ylab("log(density)")                                                  +
  theme(
    plot.title = element_blank(),
    axis.title = element_text(
      size = rel(1.25),
      face="bold"),
    axis.text.x = element_text(
      angle = -90,
      size = 10),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_line(
      colour = "black"))

ggsave(
    file=paste(outputdir,"/tucr_dist_h3k4me3.png",sep=""),
    plot = print(p),
    width = 10,
    height = 5,
    dpi = 300)
```

Graph closest RNA Pol.II Binding Sites

Have to add strand labels to TUCR output. Have to change random file "annot" column to "random".

```
filenames.pol2 <- list.files("./closest_bed/pol2/")

mastertable.pol2 <- data.frame(matrix(ncol=11,nrow=0, dimnames=list(NULL,                c("chromosome
"chromosome.pol2","start.pol2","end.pol2","pol2","dist"))))

i = 1
for(i in 1:length(filenames.pol2)){
  print(i)
  df <- read.table(paste("./closest_bed/pol2/",filenames.pol2[i],sep=""))
  colnames(df) <- colnames(mastertable.pol2)
  mastertable.pol2 <- rbind(mastertable.pol2,df)
}

mastertable.pol2.sum <- mastertable.pol2 %>%
  mutate(interval = ifelse(dist<500, 500,
                           ifelse(dist<1000, 1000,
                           ifelse(dist<1500, 1500,
                           ifelse(dist<2000, 2000,
                           ifelse(dist<2500, 2500,
                           ifelse(dist<3000, 3000,
                           ifelse(dist<3500, 3500,
                           ifelse(dist<4000, 4000,
```

```
                          ifelse(dist<4500, 4500,
                          ifelse(dist<5000, 5000,
                          ifelse(dist<5500, 5500,
                          ifelse(dist<6000, 6000,
                          ifelse(dist<6500, 6500,
                          ifelse(dist<7000, 7000,
                          ifelse(dist<7500, 7500,
                          ifelse(dist<8000, 8000,
                          ifelse(dist<8500, 8500,
                          ifelse(dist<9000, 9000,
                          ifelse(dist<9500, 9500,
                          ifelse(dist<10000, 10000,10500))))))))))))))))))))) %>%
  filter(interval != 10500) %>%
  mutate(order = ifelse(annot=="protein_coding", 1,
                          ifelse(annot=="lncRNA", 2,
                          ifelse(annot=="TUCR", 3,
                          ifelse(annot=="intergenic_TUCR", 7,
                          ifelse(annot=="antisense_RNA", 4,
                          ifelse(annot=="misc_RNA", 5,
                          ifelse(annot=="random", 6,0
                          )))))))) %>%
  mutate(annot = fct_reorder(annot,order)) %>%
  group_by(annot,interval,order) %>%
  summarise(total = n()) %>%
  ungroup() %>%
  group_by(annot) %>%
  mutate(div = sum(total)) %>%
  mutate(prop = total/div) %>%
  mutate(log10 = log10(total)) %>%
  mutate(log2 = log2(total)) %>%
  arrange(order)

p <- ggplot(mastertable.pol2.sum, aes(x=interval,y=log2,fill=annot,order=order)) +
  geom_bar(stat="identity",
    color = "black")                                                    +
  scale_fill_manual(values = c("red3","green4","blue3","deeppink","yellow2","orange2")) +
  xlab("distance (nt)")                                                  +
  ylab("log(density)")                                                  +
  theme(
    plot.title = element_blank(),
    axis.title = element_text(
      size = rel(1.25),
      face="bold"),
    axis.text.x = element_text(
      angle = -90,
      size = 10),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_line(
      colour = "black"))

ggsave(
```

```
    file=paste(outputdir,"/tucr_dist_pol2.png",sep=""),
    plot = print(p),
    width = 10,
    height = 5,
    dpi = 300)
```

Generate TUCR "Windows"

```
bedtools slop -b 10000 -i hg38.ultraConserved.sort.bed -g hg38_genomeFile.txt > tucrs_pm10kb.bed

bedtools slop -b 10000 -i TUCR_intergenic.sort.bed -g hg38_genomeFile.txt > intergenic_tucrs_pm10kb.bed

#bedtools slop -b 10000 -i CHESScoding.sort.bed -g hg38_genomeFile.txt > coding_pm10kb.bed

#bedtools slop -b 10000 -i CHESSlncRNA.sort.bed -g hg38_genomeFile.txt > lncRNAs_pm10kb.bed

#bedtools slop -b 10000 -i CHESSantisense.sort.bed -g hg38_genomeFile.txt > antisense_pm10kb.bed

#bedtools slop -b 10000 -i CHESSmisc.sort.bed -g hg38_genomeFile.txt > misc_pm10kb.bed
```

Fisher

```
bedtools fisher -a tucrs_pm10kb.bed -b pol2shuff.sort.bed -g hg38_genomeFile.txt > pol2_random_output.t

bedtools fisher -a tucrs_pm10kb.bed -b pol2.sort.bed -g hg38_genomeFile.txt > pol2_tucrs_output.txt

bedtools fisher -a intergenic_tucrs_pm10kb.bed -b pol2.bed -g hg38_genomeFile.txt > pol2_intergenic_tuc

bedtools fisher -a CHESScoding.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt > pol2_coding_output.tx

bedtools fisher -a CHESSlncRNA.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt > pol2_lncRNA_output.tx

bedtools fisher -a CHESSantisense.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt > pol2_antisense_out

bedtools fisher -a CHESSmisc.sort.bed -b pol2.sort.bed -g hg38_genomeFile.txt > pol2_misc_output.txt

bedtools fisher -a tucrs_pm10kb.bed -b h3k4me3shuff.sort.bed -g hg38_genomeFile.txt > h3k4me3_random_ou

bedtools fisher -a tucrs_pm10kb.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt > h3k4me3_tucrs_output.t

bedtools fisher -a intergenic_tucrs_pm10kb.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt > h3k4me3_int

bedtools fisher -a CHESScoding.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt > h3k4me3_coding_out

bedtools fisher -a CHESSlncRNA.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt > h3k4me3_lncRNA_out

bedtools fisher -a CHESSantisense.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt > h3k4me3_antisens

bedtools fisher -a CHESSmisc.sort.bed -b h3k4me3.sort.bed -g hg38_genomeFile.txt > h3k4me3_misc_output.
```

Show that TUCRs are enriched for hallmarks of lncRNAs

```
if(makeannotate == TRUE){
tucr_lncRNAs <- read.csv("tucr_lncRNAsA.csv")

tucr_lncRNAs <- tucr_lncRNAs %>%
  mutate(row = fct_reorder(row,Order))

p <- ggplot(tucr_lncRNAs, aes(x=row,y=value,fill=as.factor(fill)))                +
  geom_bar(
    stat="identity",
    color = "black",width=0.7)                                                    +
  scale_fill_manual(values = c("red3","green4","blue3","deeppink","yellow2","orange2")) +
  #geom_text(
  #  aes(label=round(value,2)),
  #  vjust=1.6,
  #  color="black",
  #  size=rel(4))                                                                 #+
  ggtitle(paste("TUCRs are enriched for \n RNA Pol.II and H3K4me3"))  +
  xlab("Interval and Mark")                                                       +
  ylab("-log10 of p-value")                                                       +
  theme(
    plot.title = element_text(
      size=rel(1.5),
      face="bold",hjust = 0.5),
    axis.title = element_text(
      size = rel(1.25),
      face="bold"),
    axis.text.x = element_text(
      angle = -90,
      size = 10),
    legend.title = element_blank(),
    legend.position = "none",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_line(
      colour = "black")) #+
    #facet_wrap(~wrap)

ggsave(
    file=paste(outputdir,"/tucr_lncRNAs.png",sep=""),
    plot = print(p),
    width = 5,
    height = 5,
    dpi = 300)
}
```

Section 2: TUCRs are deregulated in GBM and LGG tumors

Results

TUCR expression has not been characterized in glioblastoma (GBM) or low-grade gliomas (LGG). We performed the first comprehensive bioinformatic analysis of TUCR expression in these diseases by comparing GBM and LGG tumor samples from the Cancer Genome Atlas (TCGA) to their normal brain counterparts. Of the 481 TUCRs, we identified 115 and 82 that were up- and downregulated in GBM, respectively. (Figure 3A) We also identified 58 and 91 TUCRs that were up- and downregulated in LGG. (Figure 3B) Of the 157 deregulated TUCRs in LGG, 109 were also deregulated in GBM, a 70% overlap. (Figure 3C) Intergenic TUCRs are of particular interest; they may represent markers for novel genes. Amongst intergenic TUCRs, we identified 6 up- and 6- downregulated TUCR by a magnitude greater that 2-fold and with an FDR of 0.05 or less (Figure 3D). We performed these same analyses in LGG and identified 3 up- and 5 downregulated TUCRs. (Figure 3E) For both disease types, uc.110 and uc.62 represented the most up- and downregulated TUCRs, respectively.

Setup for TCGA RNA-Seq analysis

The next part of this analysis involves using bash and R scripts to identify TUCRs that are deregulated and/or correlated with survival in GBM and LGG. The GBM analyses were stored in a directory named tcga_analysis, while the LGG analyses were stored in the lgg_tcga_analysis. The following script generates these two directories.

```
## Create directory for all outputs from GBM TCGA analysis data
mkdir gbm_tcga_analysis/

## Do the same for LGG
mkdir lgg_tcga_analysis/
```

Extract headers from BAM files

The raw BAM files for GBM and LGG were stored in GBM-RNASEQ-RAW and LGG-RNASEQ-RAW, respectively. Since these files are labeled with unintuitive and encrypted names, I will generate clones of each BAM file using the TCGA ID as the name instead. This will make it easier to see which files are normal brain and which are tumors at a glance. TCGA files are not named this way by default because there is patient information contained within the BAM files that can be identified using the TCGA ID. Therefore, renaming the file in secure storage is preferrable. The first step to doing this is extracting the TCGA ID from the header present in each BAM file.

In the following series of scripts, a bash script was used to extract the header from each encrypted BAM file and generate a text file containing the full header.

```
## Extract header information from GBM BAM files
# Create directory for extracted headerfiles
mkdir tcga_analysis/head/

# Use a for loop to cycle through BAM files and extract header before creating a
#text file containing each BAM file's header.
for bam in $(find GBM-RNASEQ-RAW/* -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
```

```
samtools view -H $bam > $name.headerids.txt
mv $name.headerids.txt ./tcga_analysis/head/
done

## Do the same thing for LGG
# Create directory for extracted headerfiles
mkdir lgg_tcga_analysis/head/

for bam in $(find LGG-RNASEQ-RAW/* -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
samtools view -H $bam > $name.headerids.txt
mv $name.headerids.txt ./lgg_tcga_analysis/head/
done
```

Extract TCGA patient barcodes from BAM headers

Once the headers have been extracted from each BAM file, the TCGA ID is then extracted from each header. A master table is generated that contains the encrypted BAM file name and the new TCGA ID based named, which will be used to generate the renamed clone files in the next step.

```
## Get new file names for BAM files using TCGA IDs from extracted header
mkdir gbm_tcga_analysis/ids/

mkdir lgg_tcga_analysis/ids/

# Initialize summary file with TCGA IDs
echo "RAWID,TCGAID" > tcga_analysis/ids/tcgaIDs.csv

# Extract IDs from header files
for header in $(find tcga_analysis/head/* -name '*headerids.txt')
do
name=$(echo $header | awk -F ".headerids.txt" '{print $1}')
name2=$(echo $name | cut -c20-)
ids1=$(awk '$1=="@RG"{print $3}' $header)
ids2=$(echo $ids1 | cut -c4-)
echo "$name2.bam,$ids2.bam"
echo "$name2.bam,$ids2.bam" > $name.tcgaid.csv
echo "$name2.bam,$ids2.bam" >> tcga_analysis/ids/tcgaIDs.csv
mv $name.tcgaid.csv ./gbm_tcga_analysis/ids/
done

## Do the same thing for LGG

# Initialize summary file with TCGA IDs
echo "RAWID,TCGAID" > lgg_tcga_analysis/ids/tcgaIDs.csv

# Extract IDs from header files
for header in $(find lgg_tcga_analysis/head/* -name '*headerids.txt')
do
name=$(echo $header | awk -F ".headerids.txt" '{print $1}')
name2=$(echo $name | cut -c24-)
```

```
ids1=$(awk '$1=="@RG"{print $3}' $header)
ids2=$(echo $ids1 | cut -c4-)
echo "$name2.bam,$ids2.bam"
echo "$name2.bam,$ids2.bam" > $name.tcgaid.csv
echo "$name2.bam,$ids2.bam" >> lgg_tcga_analysis/ids/tcgaIDs.csv
mv $name.tcgaid.csv ./lgg_tcga_analysis/ids/
done
```

Reindex TCGA BAM files under new names

```
##Rename BAM files using TCGA ids

# Create folder for new files
mkdir gbm_tcga_analysis/reindex/

# Copy BAM files into new directory with updated names
while IFS=, read orig target; do
orig2=$(find GBM-RNASEQ-RAW/* -name $orig)
echo $orig2
cp $orig2 $target
mv $target gbm_tcga_analysis/reindex/
done < gbm_tcga_analysis/ids/tcgaIDs.csv

# Reindex the renamed files using samtools.
for bam in $(find gbm_tcga_analysis/reindex/ -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
samtools index $bam
done

## Do the same thing for LGG
# Create folder for new files
mkdir lgg_tcga_analysis/reindex

# Copy BAM files into new directory with updated names
while IFS=, read orig target; do
orig2=$(find LGG-RNASEQ-RAW/* -name $orig)
echo $orig2
cp $orig2 $target
mv $target lgg_tcga_analysis/reindex/
done < lgg_tcga_analysis/ids/tcgaIDs.csv

# Reindex the renamed files using samtools.
for bam in $(find lgg_tcga_analysis/reindex/ -name '*.bam')
do
samtools view -H TCGA-CS-4938-01B-11R-1896-07.bam > header
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
samtools reheader header $bam > $name.rehead.bam
samtools index $name.rehead.bam
```

```
done
```

Generating count tables for survival and differential expression.

```
# Making directory
mkdir tcga_analysis/foldchange

#Write header line
echo -e chrom"\t"start"\t"end"\t"id tcga_analysis/reindex/*bam | sed -e "s/ /\t/g"
  > tcga_analysis/foldchange/TUCR_counts_80.txt

#Counts
multiBamCov -bams tcga_analysis/reindex/*bam -bed hg38.ultraConserved.bed -q 10 -f 0.80
  >> tcga_analysis/foldchange/TUCR_counts_80.txt

#Do the same for LGG
mkdir lgg_tcga_analysis/foldchange

#Write header line
echo -e chrom"\t"start"\t"end"\t"id *bam | sed -e "s/ /\t/g"
  > gene_counts.txt

#Counts
multiBamCov -bams lgg_tcga_analysis/reindex/*bam -bed CHESSgenes.bed -q 10
  >> gene_counts.txt
```

Acquiring total reads from RNA-Seq BAM files

```
mkdir tcga_analysis/readcounts/

module load samtools/1.9

for bam in $(find tcga_analysis/reindex/*.bam -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
samtools view -c -F 260 $bam > $name.readcounts.txt
mv $name.readcounts.txt ./tcga_analysis/readcounts/
done

echo "id,counts" > tcga_analysis/readcounts/seqdepth_counts.csv

for txt in $(find tcga_analysis/readcounts/* -name '*.txt')
do
name=$(echo $txt | awk -F ".readcounts.txt" '{print $1}')
echo $name
reads=$(head $txt)
echo "$name,$reads" >> tcga_analysis/readcounts/seqdepth_counts.csv
done
```

Identifying differentially expressed TUCRs with DESeq2

```
tucrcounts <-
  read.table(countfilename,header = TRUE)

metadata <-
  read_csv(file = metadatafile)

tucrcounts2 <-
  tucrcounts[,5:ncol(tucrcounts)]

rownames(tucrcounts2) <-
  tucrcounts[,4]

tucrcounts.info <-
  tucrcounts[,1:4]

tucrcounts.info$length <-
  (tucrcounts$end - tucrcounts$start)/1000

tucrcounts <-
  cbind(tucrcounts.info,tucrcounts2)


dds_TUCR <-
  DESeqDataSetFromMatrix(countData = tucrcounts2,
                                colData = metadata,
                                design = ~dex)
dds_TUCR <-
  DESeq(dds_TUCR)

res_TUCR <-
  results(dds_TUCR, tidy=TRUE)

res_TUCR <-
  as_tibble(res_TUCR)

write.csv(res_TUCR, file = paste(outputdir,"/results_TUCR.csv",sep=""))

res_TUCR_summary <-
  res_TUCR %>%
  mutate(dereg =
           ifelse(res_TUCR$log2FoldChange >=1, "upregulated",
                  ifelse(res_TUCR$log2FoldChange <=-1,"downregulated","unchanged"))) %>%
  filter(dereg == "upregulated" | dereg == "downregulated") %>%
  group_by(dereg) %>%
  summarise(n = n()) %>%
  mutate(count = ifelse(dereg=="downregulated",-n,n))

p <- ggplot(res_TUCR_summary, aes(x=dereg,y=n,fill=dereg)) +
  geom_bar(stat="identity", width = 0.7,
    color = "black") +
  geom_text(aes(label=n), vjust=1.6, color="black", size=7) +
```

```
  scale_fill_manual(values=c("green", "red")) +
  ggtitle(paste("TUCR Deregulation in ",disease)) +
  xlab("Direction of Deregulation") +
  ylab("Number of Deregulated Genes") +
  theme(
    plot.title = element_text(size=rel(1.5), face="bold",hjust = 0.5),
    axis.title = element_text(size = rel(1.25), face="bold"),
    legend.title = element_blank(),
    legend.position = "none",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    panel.background = element_blank(),
    axis.line = element_line(colour = "black"))

ggsave(file=paste(outputdir,"/",disease,"_deregGraph.png",sep=""),
       plot = print(p),
       width = 5,
       height = 5,
       dpi = 300)
```

Writing a script to generate a volcano plot

```
### Volcano plot

volcanoplot <- function (res,
                         genes = "all",
                         title = paste("Deregulated Genes in ",disease,sep=""),
                         output = "results_volcanoplot.png",
                         height = 5,
                         width = 6,
                         dpi = 300){
  #res <- res_TUCR
  #genes = c("uc.110","uc.62")
  #title = "Deregulated Genes"
  vres <- res
  vres <- vres %>%
    mutate(filter1 = abs(log2FoldChange)>=1,filter2 = padj <=0.05,Legend="",gene="")

  for (i in 1:length(vres$row)){
    if(is.na(vres$padj[i])){vres$Legend[i] <- "Not deregulated"}else{
    if(vres$filter1[i] == TRUE && vres$filter2[i] == TRUE){
    vres$Legend[i] <- ">2-Fold & <0.05 FDR"
  }else{
      if(vres$filter1[i] == TRUE){
    vres$Legend[i] <- ">2-Fold"
      }else{vres$Legend[i] <- "Not deregulated"}
  }}
    ifelse(genes!="all",
           ifelse(!is.na(
             match(vres$row[i],genes)),vres$gene[i] <- vres$row[i], ""),
           vres$gene[i] <- vres$row[i])
    }
```

```
  ## plot
if(repel==TRUE){p <- ggplot(vres)
+ geom_point(aes(x=log2FoldChange, y=-log10(padj),col=Legend))
+ scale_color_manual(values = c("blue","red","black"))
+ ggtitle(title)
+ xlab("log2 fold change")
+ ylab("-log10 adjusted p-value")
+ theme(plot.title = element_text(size = rel(1.5), hjust = 0.5, face="bold"),
              axis.title = element_text(size = rel(1.25), face="bold"),
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black"))
+ geom_text_repel(aes(x=log2FoldChange, y=-log10(padj),label = gene),force=50)}
else{p <- ggplot(vres) +
        geom_point(aes(x=log2FoldChange, y=-log10(padj),col=Legend)) +
        scale_color_manual(values = c("blue","red","black")) +
        ggtitle(title) +
        xlab("log2 fold change") +
        ylab("-log10 adjusted p-value") +
        theme(plot.title = element_text(size = rel(1.5), hjust = 0.5, face="bold"),
              axis.title = element_text(size = rel(1.25), face="bold"),
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black"))}
ggsave(output, width = width, height = height, dpi = dpi)
}
```

Generating volcano plot to visualize DE genes

```
volcanoplot(res_TUCR,genes=c("uc.110","uc.210","uc.427","uc.62"),
            title = paste("Deregulated Genes in ",disease,sep=""),
            output = paste(outputdir,"/myron_tucr_results_volcanoplot.png",sep=""))

repel_hold = repel

repel = FALSE

volcanoplot(res_TUCR,
            title = paste("Deregulated Genes in ",disease,sep=""),
            output = paste(outputdir,"/tucr_results_volcanoplot.png",sep=""))

repel = repel_hold
```

Volcano plot for intergenic TUCRs only

```
tucr_annot <- read.csv("tucr_annot.csv")

res_intergenic <- inner_join(res_TUCR,tucr_annot,by = "row") %>%
  filter(annot == "intergenic")

write.csv(res_intergenic, file = paste(outputdir,"/results_intergenicOnly.csv",sep=""))
```

```
volcanoplot(res_intergenic,genes=c("uc.62","uc.110"),
            title = paste("Intergenic TUCR deregulation in ",disease,sep=""),
            output = paste(outputdir,"/intergenic_tucr_results_volcanoplot.png",sep=""))
```

Determining sequencing depth for TCGA files

```
mkdir tcga_analysis/readcounts/

for bam in $(find tcga_analysis/reindex/*.bam -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
samtools view -c -F 260 $bam > $name.readcounts.txt
mv $name.readcounts.txt ./tcga_analysis/readcounts/
done

##Do the same for lgg

mkdir lgg_tcga_analysis/readcounts/

for bam in $(find lgg_tcga_analysis/reindex/*.bam -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
echo $name
samtools view -c -F 260 $bam > $name.readcounts.txt
mv $name.readcounts.txt ./lgg_tcga_analysis/readcounts/
done
```

Identifying and comparing TUCR absolute expression in GBM and LGG

```
if(!dir.exists(paste(outputdir,"/RPKM",sep=""))){
  dir.create(paste(outputdir,"/RPKM",sep=""))
}

mergedcounts <-
  read.table(mergedcountfile,header = TRUE)

metadata <-
  read_csv(file = metadatafile)

CHESSantisense <- read.table("CHESSantisense.bed",header=FALSE)
CHESSantisense <- CHESSantisense %>%
  dplyr::select("V4","V6")


CHESScoding <- read.table("CHESScoding.bed",header=FALSE)
CHESScoding <- CHESScoding %>%
  dplyr::select("V4","V6")
```

```
CHESSlncRNA <- read.table("CHESSlncRNA.bed",header=FALSE)
CHESSlncRNA <- CHESSlncRNA %>%
  dplyr::select("V4","V6")


CHESSmisc <- read.table("CHESSmisc.bed",header=FALSE)
CHESSmisc <- CHESSmisc %>%
  dplyr::select("V4","V6")


TUCRids <- read.table("hg38.ultraconserved.bed",header=FALSE,fill=TRUE)
TUCRids <- TUCRids %>%
  dplyr::select("V4","V6"="V5")


count_annot <- rbind(TUCRids,CHESSantisense,CHESScoding,CHESSlncRNA,CHESSmisc)
colnames(count_annot) <- c("id","annot")


mergedcounts <- mergedcounts %>%
  left_join(count_annot,by="id")

NAchecker_mergedcounts<- mergedcounts[is.na(mergedcounts$annot),]

mergedcounts <- mergedcounts[complete.cases(mergedcounts),]

i <- 1

for(i in 1:length(unique(mergedcounts$annot))){


annot_unique <- as.character(unique(mergedcounts$annot))

annotation <- as.character(annot_unique[i])

print(annotation)

rpkmcounts <- mergedcounts %>%
  filter(annot == annotation) %>%
  mutate(length = end - start)

#rpkmcounts <- rpkmcounts[481,]

rpkmcounts.info <- rpkmcounts %>%
  dplyr::select(chrom,start,end,length,id,annot)

rpkmcounts <- rpkmcounts %>%
  dplyr::select(-chrom,-start,-end,-id,-annot)

#sum(is.na(mergedcounts$annot))

rownames(rpkmcounts) <- as.character(rpkmcounts.info$id)
```

```
genelength <- rpkmcounts.info$length/1000

seqdepth <- read.csv(file = seqdepthfile)

seqdepth <- as.vector(seqdepth$counts/1000000)

rpkm <- function(counts,len,dep){
  x <- counts/len
  return(t(t(x)/dep))
}

rpkm.df <- rpkm(rpkmcounts,genelength,seqdepth)
rpkm.df2 <- cbind(rpkmcounts.info,rpkm.df)

rpkm.median <- as.data.frame(rpkm.df)
rpkm.median <- rpkm.median %>%
  dplyr::summarize(median_RPKM = rowMedians(rpkm.df)) %>%
  dplyr::mutate(countif = ifelse(median_RPKM >=1,1,0),proportion = sum(countif,na.rm = TRUE)/n())
rpkm.median <- cbind(rpkmcounts.info,rpkm.median)
rpkm.median <- rpkm.median %>% dplyr::select(id,median_RPKM,countif,proportion)

proportion.df <- round(as.numeric(rpkm.median$proportion[i]),3)*100

write.csv(rpkm.df2,paste(outputdir,"/RPKM","/rpkm.",annotation,".csv",sep=""))

write.csv(rpkm.median,paste(outputdir,"/RPKM","/medianrpkm.",annotation,".csv",sep=""))

heatmap.df <- rpkm.df
geneids <- rpkm.df2 %>%
  dplyr::select(id)
means <- apply(heatmap.df,1,mean)

heatmap.df2 <- as.data.frame(apply(heatmap.df,1:2,function(x) {ifelse(x>=10,10,x)}))
heatmap.df2 <- cbind(geneids,means,heatmap.df2)

heatmap.df2 <- heatmap.df2 %>%
  gather(key = "Sample", value = "RPKM",-id,-means)

p <- ggplot(data = heatmap.df2,mapping = aes(x = Sample,y = reorder(id,means), fill = RPKM)) +
  geom_tile() +
  scale_fill_gradientn(colours = c("blue", "white", "red"), values = c(0,0.1,1)) + ggtitle(paste(annota
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())

ggsave(p,file=paste(outputdir,"/RPKM","/",annotation,"_heatmap.png",sep=""), width = 2.5, height = 7, d

}
```

Generate RPKM Box Plots for each TUCR

```
if(makeRPKMboxplots == TRUE){

if(!dir.exists(paste(outputdir,"/RPKM","/RPKMboxplots",sep=""))){
  dir.create(paste(outputdir,"/RPKM","/RPKMboxplots",sep=""))
}
rpkm.dotplot <- rpkm.TUCRs2[,6:ncol(rpkm.TUCRs2)]
TUCRids <- rpkm.TUCRs2[,4]
rpkm.dotplot <- cbind(TUCRids,rpkm.dotplot)

i = 1
for(i in 1:length(TUCRids)){
#TUCR <- "uc.110"
TUCR <- TUCRids[i]
print(i)
rpkm.dotplot2 <- rpkm.dotplot %>%
  gather(key = "Sample", value = "RPKM",-TUCRids) %>%
  dplyr::filter(TUCRids == TUCR)

p<- ggplot(rpkm.dotplot2, aes(x=TUCRids, y=RPKM)) +
  geom_boxplot() +
  geom_dotplot(binaxis='y', stackdir='center',
               stackratio=0.75, dotsize=0.20) +
  ggtitle(paste("RPKM expression of ",TUCR)) +
  theme(plot.title = element_text(hjust = 0.5),
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black")) +
  labs(y="Reads per kilobase million (RPKM)", x = "TUCR") +
  stat_summary(fun.y=mean, geom="point", shape=20, size=10, color="red", fill="red")

ggsave(p,file=paste(outputdir,"/RPKM","/RPKMboxplots/",TUCR,"_rpkm_boxplot.png",sep = ""))
}

}
```

Section 3: TUCR expression correlates with survival in LGG and GBM

Results

Gene deregulation can sometimes result in a disparity in clinical outcomes. We sought to identify whether deregulation of any of the 481 TUCRs correlate with patient outcomes in LGG and GBM. For this exercise, we obtained LGG and GBM patient survival data from TCGA. We then matched these patient outcomes to the patient RNA-Seq samples that were included in our differential expression analysis. For each TUCR, we produced a Kaplan-Meier plot tracking differences in survival for high expressing and low expressing groups. The high expression group represents the top quartile (25%) of expression values across the RNA-Seq samples; the low expression group represents the bottom quartile (25%) of expression values.

Of the 310 TUCRs that were expressed in GBM TCGA RNA-Seq data, only 36 were correlated with survival in manner that was statistically significant (p-value < 0.05). Of these 36 TUCRs, 9 had a statistically

significant Cox proportional hazard that was associated with a good prognosis, while 7 had one that was associated with a poor prognosis. The remaining 20 displayed a statistically significant difference in survival in our Kaplan-Meier analysis. (Figure 3A)

We were initially surprised by the relatively (11.6%) small fraction of TUCRs that correlate with survival in GBM. However, glioblastoma has a median survival of 15 months, which does not leave much room for dramatic differences in patient outcomes. We also studied survival differences across LGG patients, as they have a much longer median survival (X months). Because of this, we hypothesized that more TUCRs would correlate with survival in LGG. Indeed, of the 293 TUCRs that were expressed in LGG TCGA RNA-Seq data, over half (n = 167) were correlated with survival in at least one of our studies. Of these 167 TUCRs, 19 had a statistically significant Cox proportional hazard that was associated with a good prognosis, while 18 had one that was associated with a poor prognosis. In this analysis, we also identified TUCRs that were statistically significant in our Cox Proportional Hazards and Kaplan Meier studies. Of these, 49 were associated with a good prognosis while only 3 were associated with a poor prognosis. The remaining 78 displayed a statistically significant difference in survival in our Kaplan-Meier analysis only (Figure 3B). We have highlighted two TUCRs that represent a statistically significant correlation with good (uc.443, Figure 3C) or poor (uc.75, Figure 3D) prognosis using both methods.

Methodology

Acquiring and parsing clinical survival data

```
wget http://gdac.broadinstitute.org/runs/stddata__2016_01_28/data/GBM/20160128/gdac.broadi
nstitute.org_GBM.Merge_Clinical.Level_1.2016012800.0.0.tar.gz

tar -xvzf gdac.broadinstitute.org_GBM.Merge_Clinical.Level_1.2016012800.0.0.tar.gz

mv gdac.broadinstitute.org_GBM.Merge_Clinical.Level_1.2016012800.0.0 GBM.Merge_Clinical
```

Completing survival analysis for TUCRs

```
if(!file.exists("GBM.clin.merged.txt")){unzip("GBM.clin.merged.zip")}

countdata <- read.table(countfilename,header = TRUE)
metadata <- read_csv(file = metadatafile)

posdata <- countdata[,1:4]
countdata <- countdata[,5:length(colnames(countdata))]

countdata <- as.matrix(countdata)

n_index <- c(seq(1,5))
t_index <- c(seq(6,ncol(countdata)))

vm <- function(x){
  cond <- factor(ifelse(seq(1,dim(x)[2],1) %in% t_index, 1,  0))
  d <- model.matrix(~1+cond)
  x <- t(apply(x,1,as.numeric))
  ex <- voom(x,d,plot=F)
  return(ex$E)
```

```
}

count_vm <- vm(countdata)
colnames(count_vm) <- metadata$barcode

scal <- function(x,y){
  mean_n <- rowMeans(y)  # mean of normal
  sd_n <- apply(y,1,sd)  # SD of normal
  # z score as (value - mean normal)/SD normal
  res <- matrix(nrow=nrow(x), ncol=ncol(x))
  colnames(res) <- colnames(x)
  rownames(res) <- rownames(x)
  for(i in 1:dim(x)[1]){
    for(j in 1:dim(x)[2]){
      res[i,j] <- (x[i,j]-mean_n[i])/sd_n[i]
    }
  }
  return(res)
}

z_rna <- scal(count_vm[,t_index],count_vm[,n_index])
rownames(z_rna) <- posdata[,4]

clinical <- read.table("GBM.clin.merged.txt",header = TRUE)
#clinical <- read.table("GBM.oncolnc.merged.txt",header = TRUE)

clinical$time <- as.numeric(clinical$time)

sum(clinical$patient %in% colnames(z_rna))

ind_tum <- which(unique(colnames(z_rna)) %in% clinical$patient)
ind_clin <- which(clinical$patient %in% colnames(z_rna))

out.tab <- c()
for(x in 1:nrow(count_vm)){
  ind_gene <- x
  s <- Surv(clinical$time[ind_clin],clinical$status[ind_clin])
  cx <- coxph(formula = s ~ count_vm[ind_gene,ind_tum])
  cx <- tidy(cx)
  out.tab <- rbind(out.tab,cx)
}

surv_TUCR <- cbind(posdata,out.tab)
surv_TUCR <- surv_TUCR %>% dplyr::select(-term)

write_csv(surv_TUCR,paste(outputdir,"/survival_TUCRs.csv",sep=""))

if(makekpmplots == TRUE){

if(!dir.exists(paste(outputdir,"/tucrKPMplots",sep=""))){
  dir.create(paste(outputdir,"/tucrKPMplots",sep=""))
}
```

```
km_countdata <- countdata
colnames(km_countdata) <- metadata$barcode
posdata$median <- rowMedians(countdata)

probs <- c(0.25,0.5,0.75)
q <- rowQuantiles(countdata, probs = probs)
posdata$n25 <- q[,1]
posdata$n75 <- q[,3]
km_TUCRs <- cbind(posdata,km_countdata)
km_TUCRs <- km_TUCRs[,4:ncol(km_TUCRs)]
km_TUCRs <- km_TUCRs %>%
  gather(key = "sample", value = "count",-id,-median,-n75,-n25) %>%
  mutate(group = ifelse(count>=n75,"high",ifelse(count<=n25,"low",NA))) %>%
  dplyr::select(id,median,"patient"=sample,group) %>%
  left_join(clinical,by="patient") %>%
  dplyr::filter(median != 0)

TUCRids <- posdata$id
i = 1

ptable <- data.frame(matrix(ncol=3,nrow=0, dimnames=list(NULL, c("id","pvalue","method"))))

for(i in 1:length(TUCRids)){
  print(i)
  skip_to_next <- FALSE
  TUCR <- TUCRids[i]
  #TUCR <- "uc.1"
  kmdata <- km_TUCRs %>%
    dplyr::filter(id == TUCR)
  fit <- survfit(Surv(time, status) ~ group, data = kmdata)
  p <- tryCatch(ggsurvplot(fit,data=kmdata,conf.int = TRUE,pval = TRUE,risk.table = TRUE),
                error = function(e) { skip_to_next <<- TRUE})
  if(skip_to_next) {
    rbinder <- cbind(as.character(TUCR),NA,NA)
    ptable[i,] <- rbinder
    next }
  ggsave(file=paste(outputdir,"/tucrKPMplots/",TUCR,"_kpmplot.png",sep = ""),
         plot = print(p))
    p2value <- surv_pvalue(fit,  data = kmdata,method = "survdiff")
  rbinder <- cbind(as.character(TUCR),p2value$pval,p2value$method)
  ptable[i,] <- rbinder
}}

write_csv(ptable,paste(outputdir,"/kpm_summary_TUCRs.csv",sep=""))
```

Summarize Survival Data

```
surv_TUCR <- read.csv(paste(outputdir,"/survival_TUCRs.csv",sep=""),header=TRUE)

ptable <- read.csv(paste(outputdir,"/kpm_summary_TUCRs.csv",sep=""),header=TRUE)

res_surv <- inner_join(ptable,surv_TUCR,by="id")
```

```
res_surv_sum <- res_surv

res_surv_sum <- res_surv_sum %>% mutate(gene="",Survival="")

for (i in 1:length(res_surv_sum$id)){
  ifelse(is.na(res_surv_sum$pvalue[i]),res_surv_sum$pvalue[i] <- 1,res_surv_sum$pvalue[i]
         <- res_surv_sum$pvalue[i])

  ifelse(genes!="all",ifelse(!is.na(match(res_surv_sum$id[i],genes)),
                             res_surv_sum$gene[i] <- as.character(res_surv_sum$id[i]), ""),
                                  res_surv_sum$gene[i] <- res_surv_sum$id[i])

  ifelse((res_surv_sum$pvalue[i] <0.05 & res_surv_sum$p.value[i] <0.05 &
            res_surv_sum$estimate[i] < 0),
         res_surv_sum$Survival[i] <- "Significant (Both, Good Prognosis)",
  ifelse((res_surv_sum$pvalue[i] <0.05 & res_surv_sum$p.value[i] <0.05 &
            res_surv_sum$estimate[i] > 0),
         res_surv_sum$Survival[i] <- "Significant (Both, Poor Prognosis)",
  ifelse((res_surv_sum$pvalue[i] >0.05 & res_surv_sum$p.value[i] <0.05 &
            res_surv_sum$estimate[i] < 0),
         res_surv_sum$Survival[i] <- "Significant (CH, Good Prognosis)",
  ifelse((res_surv_sum$pvalue[i] >0.05 & res_surv_sum$p.value[i] <0.05 &
            res_surv_sum$estimate[i] > 0),
         res_surv_sum$Survival[i] <- "Significant (CH, Poor Prognosis)",
  ifelse((res_surv_sum$pvalue[i] <0.05 & res_surv_sum$p.value[i] >0.05),
         res_surv_sum$Survival[i] <- "Significant (KM)",
  ifelse(res_surv_sum$Survival[i] <- "Not significant"))))))}

res_surv_sum <- res_surv_sum %>%
  group_by(Survival) %>%
  summarise(count = n())

write.csv(res_surv_sum,file=paste(outputdir,"/tucr_surv_table.csv",sep=""))

p <- ggplot(res_surv_sum, aes(x=Survival,y=count,fill=Survival)) + geom_bar(stat="identity", width = 0.
plot.title = element_text(size=rel(1.5), face="bold",hjust = 0.5),
axis.title = element_text(size = rel(1.25), face="bold"),
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black"),
axis.text.x = element_blank())

ggsave(file=paste(outputdir,"/tucr_surv_bar.png",sep=""),plot = print(p), width = 5, height = 5, dpi = 3
```

Writing a script to generate a volcano plot for survival

```
### Volcano plot

volcanosurv <- function (res,genes = "all",title = paste("TUCRs correlated with survival in ",disease,se
  #res <- res_surv
  #genes = c("uc.110","uc.62")
  #title = paste("TUCR correlation with patient survival in ",disease,sep="")
```

```
  #output = paste(outputdir,"/intergenic_tucr_results_volcanosurv.png",sep="")
  i = 9
  vres <- res
  vres <- vres %>% mutate(gene="",Survival="")
  for (i in 1:length(vres$id)){
  ifelse(is.na(vres$pvalue[i]),vres$pvalue[i] <- 1,vres$pvalue[i] <- vres$pvalue[i])

  ifelse(genes!="all",ifelse(!is.na(match(vres$id[i],genes)),vres$gene[i] <- as.character(vres$id[i]),

  ifelse((vres$pvalue[i] <0.05 & vres$p.value[i] <0.05 & vres$estimate[i] < 0),vres$Survival[i] <- "Sign
  ifelse((vres$pvalue[i] <0.05 & vres$p.value[i] <0.05 & vres$estimate[i] > 0),vres$Survival[i] <- "Sign
  ifelse((vres$pvalue[i] >0.05 & vres$p.value[i] <0.05 & vres$estimate[i] < 0),vres$Survival[i] <- "Sign
  ifelse((vres$pvalue[i] >0.05 & vres$p.value[i] <0.05 & vres$estimate[i] > 0),vres$Survival[i] <- "Sign
  ifelse((vres$pvalue[i] <0.05 & vres$p.value[i] >0.05),vres$Survival[i] <- "Significant (KM)",
  ifelse(vres$Survival[i] <- "Not significant"))))))}


  ## plot
if(repel==TRUE){p <- ggplot(vres) +
        geom_point(aes(x=estimate, y=-log10(as.numeric(p.value)),col=Survival)) +
        scale_color_manual(values = c("gray","#619cff","#f8766d","#00ba38","#c77cff","black")) +
        ggtitle(title) +
        xlab("Cox Estimated Proportional Hazard") +
        ylab("-log10 P-Value (CH)") +
        theme(plot.title = element_text(size = rel(1.5), hjust = 0.5, face="bold"),
              axis.title = element_text(size = rel(1.25), face="bold"),
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black")) +
        geom_text_repel(aes(x=estimate, y=-log10(as.numeric(p.value)),label = gene),force=50)
}else{
  p <- ggplot(vres) +
        geom_point(aes(x=estimate, y=-log10(as.numeric(pvalue)),col=Survival)) +
    scale_color_manual(values = c("gray","#619cff","#f8766d","#00ba38","#c77cff","black")) +
        + ggtitle(title) +
        xlab("Cox Estimated Proportional Hazard") +
        ylab("-log10 P-Value (CH)") +
        theme(plot.title = element_text(size = rel(1.5), hjust = 0.5, face="bold"),
              axis.title = element_text(size = rel(1.25), face="bold"),
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black"))}
ggsave(output, width = width, height = height, dpi = dpi)
}
```

Draw Volcano plot for TUCRs

```
volcanosurv(res_surv,genes=c("uc.174","uc.325","uc.363"), title = paste("TUCR correlation with patient s
```

Section 4: TUCRs are coregulated with genes that have known functions

Results

Many genes have published functions and mechanisms of action. Gene that are coregulated have correlated differential expression. Coregulated genes may share a biological function. We have predicted TUCR function by identifying coregulated genes with known functions. Using a guilt-by-association analysis pipeline, we first identified determined the deregulation of all known coding and non-coding genes using the CHESS annotation database. This dataset contains known Refseq and Ensembl genes in additional to some novel genes that were discovered in 2018. We the constructed a correlation matrix of TUCRs to other TUCRs and "CHESS genes" using the Spearman correlation. This correlation measures the relationship of the gene to the TUCR as TUCR deregulation decreases. If the TUCR increases and the coregulated gene decreases, the correlation will increase and approach a perfect positive correlated value of 1. If the TUCR increases and the coregulated gene decreases, the correlation will decrease and approach a perfect negative correlation of -1. Genes with no correlation to the TUCR are those with a correlation close to zero, specifically between -0.3 and 0.3.

Coregulated genes lists for each TUCR were subset into separate lists by directionality (positive/negative). These lists were subject to gene ontology (GO) term analysis. For each TUCR, lists of biological processes, molecular functions, or cellular compartments were predicted for the list of coregulated genes. This list was cross referenced and filtered to a control list of "cancer" related GO Terms; these terms were generated by performing the same guilt-by-association analysis on known oncogenes and tumor suppressors from the CancerMine database. The final matrix was filtered to statistically significant (FDR <=0.05) GO Terms only. Then a heatmap of the directionality of significant positive (red), negative (blue), and uncorrelated biological processes (white) was generated for GBM (Figure 5A) and LGG (Figure 6A). These heatmaps were sorted by frequency of occurrence, and the 10 most represented positive (Figure 5B, Figure 6B) and negative (Figure 5C, Figure 6C) associations are highlighted. These analyses were also performed to determine TUCR molecular functions (MF, Supplementary Figures 2 and 3) and cellular compartments (CC, Supplementary Figures 4 and 5). The full list of all TUCR GO Terms was also compiled. (Supplementary Table 1)

Methodology

Identify DE TUCRs that are correlated with each other

```
if(!dir.exists(paste(outputdir,"/correlations",sep=""))){
  dir.create(paste(outputdir,"/correlations",sep=""))
}

if(useintermediate==FALSE){
TUCRids <- read.table(countfilename,header = TRUE)
TUCRids <- as.character(TUCRids[,4])
normcounts <- read.table("normal_mergedcounts.txt",header = TRUE)
expcounts <- read.table(mergedcounts,header = TRUE)
posdata <- normcounts[,1:4]
normcounts <- normcounts[4:ncol(normcounts)]
expcounts <- expcounts[4:ncol(expcounts)]
tucrcounts <- inner_join(posdata,normcounts,by="id")
tucrcounts <- inner_join(tucrcounts,expcounts,by="id")

metadata <- read_csv(file = metadatafile)
```

```
tucr.cor <- tucrcounts[,5:ncol(tucrcounts)]
rownames(tucr.cor) <- as.character(tucrcounts[,4])

posdata <- tucrcounts[,1:4]

countdata <- as.matrix(tucr.cor)

n_index <- c(seq(1,5))
t_index <- c(seq(6,ncol(countdata)))

vm <- function(x){
  cond <- factor(ifelse(seq(1,dim(x)[2],1) %in% t_index, 1,  0))
  d <- model.matrix(~1+cond)
  x <- t(apply(x,1,as.numeric))
  ex <- voom(x,d,plot=F)
  return(ex$E)
}

count_vm <- vm(countdata)
colnames(count_vm) <- metadata$barcode
posdata$median <- rowMedians(as.matrix(count_vm))

tucr.cor <- cbind(posdata[,4:5],count_vm)

tucr.cor.matrix <- data.frame(row.names = posdata[,4])


i=3

for(i in 3:ncol(tucr.cor)){
  print(i)
  tucr.cor.matrix[,i-2] <- tucr.cor[,i]/tucr.cor[,2]


colnames(tucr.cor.matrix) <- metadata$id
tucr.cor.matrix2 <- t(as.data.frame(apply(as.data.frame(tucr.cor.matrix),1:2,function(x) {ifelse(x>=1,x
tucr.cor.matrix2 <- as.data.frame(tucr.cor.matrix2)
tucr.cor.matrix2 <- tucr.cor.matrix2[1:180,]
geneindex <- rownames(tucr.cor.matrix)}

save(tucr.cor.matrix2,geneindex,TUCRids,file="inter.rData")
rm(list=ls())

load(file="inter.rData")

tucr.cor.matrix3 <- cor(tucr.cor.matrix2,method="spearman", use = "complete.obs")

save(tucr.cor.matrix3,geneindex,TUCRids,file=intermediatefile)
rm(list=ls())
load(file=intermediatefile)}else{
  load("lgg_matrixintermediate.rData")
```

```
}

header <- head(tucr.cor.matrix3)

summarycor <- data.frame(searchTUCR=character(0),ngenes=integer(0),meancor=numeric(0),mediancor=numeric

i = 1

for(i in 1:length(TUCRids)){
  #print(i)
  searchTUCR <- TUCRids[i]
  #searchTUCR <- "uc.110"
###uc.110 is at row 20972 in tucr.cor.matrix3 via manual confirmation.
tucr.cor.matrix.match <- tucr.cor.matrix3[match(searchTUCR,geneindex),]
tucr.cor.matrix.match <- as.data.frame(tucr.cor.matrix.match)
tucr.cor.matrix.match.plus <- tucr.cor.matrix.match %>%
  mutate(partner = geneindex) %>%
  mutate(gene = searchTUCR) %>%
  dplyr::select(gene,partner,"cor"=tucr.cor.matrix.match) %>%
  dplyr::filter(cor>=0.3 & partner != searchTUCR) %>%
  dplyr::mutate(direction = "positive")

write.csv(tucr.cor.matrix.match.plus,
          paste(outputdir,"/correlations/",searchTUCR,"_plus_correlations.csv",sep=""))

tucr.cor.matrix.match.minus <- tucr.cor.matrix.match %>%
  mutate(partner = geneindex) %>%
  mutate(gene = searchTUCR) %>%
  dplyr::select(gene,partner,"cor"=tucr.cor.matrix.match) %>%
  dplyr::filter(cor<=-0.3 & partner != searchTUCR) %>%
  dplyr::mutate(direction = "negative")

write.csv(tucr.cor.matrix.match.minus,
          paste(outputdir,"/correlations/",searchTUCR,"_minus_correlations.csv",sep=""))

meancor <- mean(tucr.cor.matrix.match$tucr.cor.matrix.match,na.rm=TRUE)
mediancor <- median(tucr.cor.matrix.match$tucr.cor.matrix.match,na.rm=TRUE)
ngenes <- length(tucr.cor.matrix.match$tucr.cor.matrix.match)
mincor <- min(tucr.cor.matrix.match$tucr.cor.matrix.match,na.rm=TRUE)
maxcor <- max(tucr.cor.matrix.match$tucr.cor.matrix.match,na.rm=TRUE)
newrow <- cbind(searchTUCR,ngenes,meancor,mediancor,mincor,maxcor)
summarycor <- rbind(summarycor,newrow)}

summarycor %>%
readr::write_excel_csv(here::here(paste(outputdir,"/correlations",sep=""),paste("cor_summary.csv",sep="
```

Generate GO Terms for coregulated genes

```
filenames <- list.files(paste(outputdir,"/correlations",sep=""))

if(!dir.exists(paste(outputdir,"/GO_terms",sep=""))){
```

```
  dir.create(paste(outputdir,"/GO_terms",sep=""))
}

if(!dir.exists(paste(outputdir,"/GO_terms_bp",sep=""))){
  dir.create(paste(outputdir,"/GO_terms_bp",sep=""))
}

if(!dir.exists(paste(outputdir,"/GO_terms_mf",sep=""))){
  dir.create(paste(outputdir,"/GO_terms_mf",sep=""))
}

if(!dir.exists(paste(outputdir,"/GO_terms_cc",sep=""))){
  dir.create(paste(outputdir,"/GO_terms_cc",sep=""))
}

allgo <- data.frame(matrix(ncol=7,nrow=0, dimnames=list(NULL, c("GO Term","Ont","N","DE","P.DE","direct

allgo_bp <- data.frame(matrix(ncol=7,nrow=0, dimnames=list(NULL, c("GO Term","Ont","N","DE","P.DE","dir

allgo_mf <- data.frame(matrix(ncol=7,nrow=0, dimnames=list(NULL, c("GO Term","Ont","N","DE","P.DE","dir

allgo_cc <- data.frame(matrix(ncol=7,nrow=0, dimnames=list(NULL, c("GO Term","Ont","N","DE","P.DE","dir

i = 2

for(i in 2:length(filenames)){
  #print(i)
  skip_to_next <- FALSE
  df <- read.csv(paste(outputdir,"/correlations/",filenames[i],sep=""))
  gene <- as.character(df$gene[1])

  if(length(df[,1] > 0)){

  df2 <- cSplit(df, 'partner', sep="___", type.convert=FALSE)

  df2 <- df2 %>% arrange(desc(cor))

  IDs <- df2$partner_1

  entrezIDs <- tryCatch(mapIds(org.Hs.eg.db, IDs, 'ENTREZID', 'SYMBOL'), error = function(e) { skip_to_

  if(skip_to_next) { next }

  df2$entrez <- entrezIDs
  if(str_detect(filenames[i],"minus")){direction <- -1}else{
    if(str_detect(filenames[i],"plus")){direction <- 1}else{direction <- 0}
  }

  Genes<-entrezIDs
  g <- goana(Genes)
  topGO <- topGO(g)

  topGO_all <- topGO %>%
```

```
    mutate(direction = direction) %>%
    mutate(gene = gene)

  write.csv(topGO_all,paste(outputdir,"/GO_terms/",filenames[i],sep=""))
  topGO_bp <- topGO %>%
    filter(Ont == "BP") %>%
    mutate(direction = direction) %>%
    mutate(gene = gene)

  write.csv(topGO_bp,paste(outputdir,"/GO_terms_bp/",filenames[i],sep=""))
  topGO_mf <- topGO %>%
    filter(Ont == "MF") %>%
    mutate(direction = direction) %>%
    mutate(gene = gene)

  write.csv(topGO_mf,paste(outputdir,"/GO_terms_mf/",filenames[i],sep=""))
  topGO_cc <- topGO %>%
    filter(Ont == "CC") %>%
    mutate(direction = direction) %>%
    mutate(gene = gene)

  write.csv(topGO_cc,paste(outputdir,"/GO_terms_cc/",filenames[i],sep=""))
  }else{}
  allgo <- rbind(allgo,topGO)
  allgo_bp <- rbind(allgo_bp,topGO_bp)
  allgo_mf <- rbind(allgo_mf,topGO_mf)
  allgo_cc <- rbind(allgo_cc,topGO_cc)
}

write_csv(allgo,paste(outputdir,"/GOterm_all_summary_TUCRs.csv",sep=""))

write_csv(allgo_bp,paste(outputdir,"/GOterm_bp_summary_TUCRs.csv",sep=""))

write_csv(allgo_mf,paste(outputdir,"/GOterm_mf_summary_TUCRs.csv",sep=""))

write_csv(allgo_cc,paste(outputdir,"/GOterm_cc_summary_TUCRs.csv",sep=""))
```

Summarize GO term data for oncogenes and tumor suppressors

```
if(!dir.exists(paste(outputdir,"/Cancermine/",sep=""))){
  dir.create(paste(outputdir,"/Cancermine/",sep=""))
}

GSEA <- read.csv("GSEA.csv",header = TRUE)

annotations <- as.character(unique(GSEA$annot))

topGO_onctsg <- ""

i <- 1
```

```
for(i in 1:length(annotations)){

#print(annotations[i])

group <- GSEA %>%
  filter(annot == annotations[i])

entrezIDs <- mapIds(org.Hs.eg.db, as.character(group$gene), 'ENTREZID', 'SYMBOL')

Genes <- entrezIDs
g <- goana(Genes)
topGO <- topGO(g)
topGO$annot <- annotations[i]

topGO_onctsg <- rbind(topGO_onctsg,topGO)
}

topGO_cancermine <- topGO_onctsg %>%
    filter(Ont == "BP") %>%
    dplyr::select(Term,P.DE,annot)

cancerTerms <- topGO_cancermine %>%
  dplyr::select(Term) %>%
  unique()

write.csv(topGO_cancermine,paste(outputdir,"/Cancermine/topGO_cancermine.csv",sep=""))
```

Summarize GO Term Data (new)

```
cancerTerms <- read.csv(paste(outputdir,"/Cancermine/topGO_cancermine.csv",sep=""),header=TRUE) %>%
  dplyr::select(Term) %>%
  distinct()

TUCRids <- read.table(countfilename,header = TRUE)
TUCRids <- as.character(TUCRids[,4])

allgo <- read.csv(paste(outputdir,"/GOterm_all_summary_TUCRs.csv",sep=""),header=TRUE,stringsAsFactors=
allgo <- allgo %>%
  dplyr::select(Term)

rbinder.allgo <- allgo
heatmap.allgo <- data.frame(matrix(ncol=3,nrow=0, dimnames=list(NULL, c("Term","gene","direction"))))

i <- 1

for(i in 1:length(TUCRids)){
#for(i in 1:30){
  print(i)
  TUCR <- TUCRids[i]
  skip_to_next_p <<- FALSE
  skip_to_next_n <<- FALSE
```

```
  positive <- tryCatch(read.csv(paste(outputdir,"/GO_Terms/",TUCR,"_plus_correlations.csv",sep=""),
                       header=TRUE), error = function(e) { skip_to_next_p <<- TRUE})

  negative <- tryCatch(read.csv(paste(outputdir,"/GO_Terms/",TUCR,"_minus_correlations.csv",sep=""),
                       header=TRUE), error = function(e) { skip_to_next_n <<- TRUE})


  if(skip_to_next_p & skip_to_next_n) { next }

  pos_neg <- rbind(positive,negative)
  pos_neg <- pos_neg %>%
    filter(P.DE <= 0.05) %>%
    dplyr::select(Term,gene,direction)

  rbinder.allgo.TUCR <- dplyr::left_join(rbinder.allgo,pos_neg,by="Term")
  rbinder.allgo.TUCR$gene <- TUCR
  heatmap.allgo <- rbind(heatmap.allgo,rbinder.allgo.TUCR)
}


heatmap.allgo[is.na(heatmap.allgo)] <- 0
heatmap.allgo2 <- heatmap.allgo[which(as.character(heatmap.allgo$Term) %in% as.character(cancerTerms$Te

#heatmap.allgo2 <- heatmap.allgo

heatmap.allgo2 <- heatmap.allgo2 %>%
  distinct() %>%
  dplyr::group_by(Term,gene) %>%
  mutate(n=n(),onedirection = ifelse(n>1,0,direction)) %>%
  dplyr::select(-n,-direction) %>%
  distinct() %>%
  group_by(Term,onedirection) %>%
  mutate(ysums = ifelse(onedirection==0,0,sum(onedirection))) %>%
  ungroup() %>%
  group_by(Term) %>%
  mutate(y = sum(abs(onedirection))) %>%
  ungroup() %>%
  group_by(gene) %>%
  mutate(x = sum(onedirection)) %>%
  ungroup() %>%
  arrange(desc(y))
  #spread(Term,onedirection) %>%
  #ungroup()



colors <- colorRampPalette(c("red","white","blue"))(3)

heatmap.allgo2$onedirection <- factor(heatmap.allgo2$onedirection)
N <- nlevels(heatmap.allgo2$onedirection)

p <- ggplot(data = heatmap.allgo2,mapping = aes(x = reorder(gene,x),y = reorder(Term,y), fill = onedire
guides(fill = guide_legend(reverse=TRUE)) +
```

```
  geom_tile() +
   scale_fill_manual(values=colors, breaks=levels(heatmap.allgo2$onedirection)[seq(1, N, by=1)]) +
  ggtitle(paste("TUcR cancer associated functions in ",disease,sep="")) +
  xlab(paste("TUCRs (n=",length(unique(heatmap.allgo2$gene)),")",sep = "")) +
  ylab(paste("Cancer associated GO Terms (n=",length(unique(heatmap.allgo2$Term)),")",sep = "")) +
  labs(fill = "Correlation") +
  theme(plot.title = element_text(hjust = 0.5),
        panel.grid = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank())

ggsave(p,file=paste(outputdir,"/GOTerms_heatmap.png",sep=""))

chart.mostrep <- heatmap.allgo2 %>%
  dplyr::select(Term,y) %>%
  distinct() %>%
  arrange(desc(y))

chart.allgo.mostrep <- chart.mostrep[1:8,]

chart.allgo.up <- heatmap.allgo2 %>%
  filter(ysums>0) %>%
  dplyr::select(Term,ysums) %>%
  distinct() %>%
  arrange(desc(ysums))

chart.allgo.up.top10 <- chart.allgo.up[1:10,]

chart.allgo.down <- heatmap.allgo2 %>%
  filter(ysums<0) %>%
  dplyr::select(Term,ysums) %>%
  distinct() %>%
  arrange(ysums)

chart.allgo.down.bottom10 <- chart.allgo.down[1:10,]

p <- ggplot(chart.allgo.mostrep, aes(x=reorder(Term,y),y=y,fill=Term)) +
geom_bar(stat="identity", width = 0.7,color="black") + geom_text(aes(label=round(y,2)), hjust=2, color=
  theme(
  plot.title = element_text(size=rel(1.5), face="bold",hjust = 0.5),
axis.title = element_text(size = rel(1.25), face="bold"),
legend.title = element_blank(),
legend.position = "none",
axis.text.x = element_text(size=rel(2.0)),
axis.text.y = element_text(size=rel(2.0)),
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black")) + coord_flip()

ggsave(file=paste(outputdir,"/tucr_allgo.png",sep=""),plot = print(p), width = 15, height = 10, dpi = 3
```

Section 5: TUCRs are contained within mRNA or lncRNA transcripts

Results

Of the 481 TUCRs, 186 were previously characterized as intergenic, or not being contained within a known coding or non-coding transcript. We chose to focus on these TUCRs, as they potentially represent a partial or full novel long non-coding RNA (lncRNA) transcripts. To identify the RNA transcript(s) that contain these TUCRs, we performed a de novo transcript reassembly using GBM RNA-Seq data from TCGA. Using this method, we identified 2636 transcripts that contain TUCRs. We prioritized transcripts that did not overlap with known genes, as these are likely to represent novel genes. Transcripts that were greater than 10,000 nucleotide (nt) were also eliminated. Using these criteria, 8 transcripts were identified in GBM, LGG, or both disease types. (Figure 7A, Table 1) These transcripts were submitted to the Coding Potential Assessing Tool (CPAT) to determine whether they express any coding potential. Of the 8 transcripts, five represented no coding potential, while three contained coding potential. In particular, one of these transcripts with coding potential (TUC110-) overlaps with the most highly upregulated TUCR in both LGG and GBM (uc.110, Figures 7B and 7C).

Methodology

Stringtie GBM

The hg19 genomic coordinates for each TUCR were acquired from _____ and were lifted over to hg38 coordinates. The stringtie bioinformatics package was then used to perform a de novo transcript reassembly of all transcripts in 161 GBM, 400 LGG, and 5 normal brain TCGA Cancer RNA-Seq datasets. First, transcripts present in each RNA-Seq BAM file were compiled into individual BED/GTF format genomic coordinate files using Stringtie. These files were then merged into one master file containing all transcripts in all samples using Stringtie and an hg38 genome reference file. The bedtools package was then used to find the genomic coordinates of all transcripts containing ultraconserved regions. The _____ group used stringtie to analyze the entirety of the GTex RNA-Seq normal tissue catalog, so bedtools was also used to determine the intersection of TUCRs with their results. This methodology provides insight into TUCR transcript presence in normal tissues and in brain cancer.

```
wget ftp://ftp.ensembl.org/pub/release-99/gtf/homo_sapiens/Homo_sapiens.GRCh38.99.gtf.gz
mv Homo_sapiens.GRCh38.99.gtf.gz stringtie
gunzip stringtie/Homo_sapiens.GRCh38.99.gtf.gz

module load gcc/7.1.0
module load stringtie/2.0.6

for bam in $(find GBM-RNASEQ-RAW/* -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
name2=$(basename "$name")
if [ ! -f stringtie/$name2.gtf ]
then
    echo "$name2.gtf does not exist"
    stringtie $bam -o $name.gtf
    mv $name.gtf ./stringtie
else
    echo "$name2.gtf exists... skipping"
fi
```

```
if [ ! -f $name.gtf ]; then
  stringtie $bam -o $name.gtf
fi
mv $name.gtf ./stringtie
done

cd stringtie

stringtie --merge *rehead.gtf -G Homo_sapiens.GRCh38.99.gtf -o stringtie_merged.gtf

intersectBed -a hg38.ultraConserved.bed -b CHESSgenes.bed -wa -wb
  > intersectchess_TUCRs.bed

intersectBed -a hg38.ultraConserved.bed -b stringtie_merged.gtf -wa -wb
  > intersectstringtie_TUCRs.bed

cp stringtie_merged.gtf stringtie_merged.bed
```

Stringtie LGG

```
wget ftp://ftp.ensembl.org/pub/release-99/gtf/homo_sapiens/Homo_sapiens.GRCh38.99.gtf.gz
mv Homo_sapiens.GRCh38.99.gtf.gz stringtie
gunzip stringtie/Homo_sapiens.GRCh38.99.gtf.gz

module load gcc/7.1.0
module load stringtie/2.0.6

for bam in $(find LGG-RNASEQ-RAW/* -name '*.bam')
do
name=$(echo $bam | awk -F ".bam" '{print $1}')
name2=$(basename "$name")
if [ ! -f stringtie/$name2.gtf ]
then
    echo "$name2.gtf does not exist"
    stringtie $bam -o $name.gtf
    mv $name.gtf ./stringtie
else
    echo "$name2.gtf exists... skipping"
fi

if [ ! -f $name.gtf ]; then
  stringtie $bam -o $name.gtf
fi
mv $name.gtf ./stringtie
done

cd stringtie

stringtie --merge *rehead.gtf -G Homo_sapiens.GRCh38.99.gtf -o stringtie_merged.gtf

intersectBed -a hg38.ultraConserved.bed -b CHESSgenes.bed -wa -wb
```

```
  > intersectchess_TUCRs.bed

intersectBed -a hg38.ultraConserved.bed -b stringtie_merged.gtf -wa -wb
  > intersectstringtie_TUCRs.bed

cp stringtie_merged.gtf stringtie_merged.bed
```

Show that TUCRs are contained within novel genes

```
tucr_novelgenes <- read.csv("tucr_novelgenes.csv")

p <- ggplot(tucr_novelgenes, aes(x=row,y=value,fill=row)) + geom_bar(stat="identity", width = 0.7) +
  geom_text(aes(label=round(value,2)), vjust=1.6, color="black", size=rel(4)) + ggtitle(paste("Some int
plot.title = element_text(size=rel(1.5), face="bold",hjust = 0.5),
axis.title = element_text(size = rel(1.25), face="bold"),
axis.text.x = element_text(angle = -90, size = 10),
legend.title = element_blank(),
legend.position = "none",
panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
panel.background = element_blank(), axis.line = element_line(colour = "black"))

ggsave(file=paste(outputdir,"/tucr_novelgenes.png",sep=""),plot = print(p), width = 5, height = 5, dpi =
```