eCLIP Analysis Pipeline Manual

Romel Baral, Chongming Ni, Arjun Sarathi, Qidi Yang, Yuanqi Zhao

February 29, 2020

Contents

1 Software Introduction and Overview

1.1 Pipeline Thumbnail

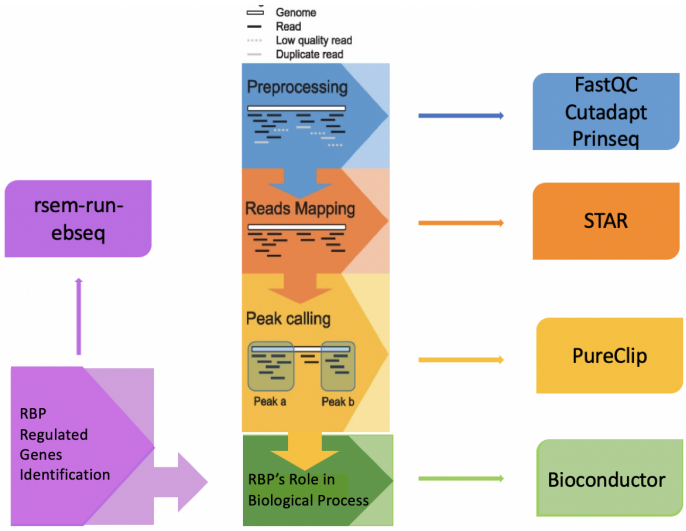

Figure 1: Pipeline Workflow

Our pipeline takes in the gene name of a user-chosen RNA binding protein (RBP), downloads the raw reads from ENCODE, processes the reads, and outputs the RBP's potential roles in cancer progression.

Users can specify our pipeline to be either *default*, *differential expression analysis*, or *cancer analysis* modes. The *default* mode includes steps of preprocessing, reads mapping, and peak calling. The output is a BED file which indicates all possible crosslink sites.

The *differential expression analysis* mode preforms DE analysis with the RBP gene knockdown datasets to hypothesis genes regulated by the user-specified RBP. The result gene list would be taken as an input of the downstream *cancer analysis*.

The *cancer analysis* mode identifies the RBP's role in the user-specified biological process such as cancer progression. By performing differential expression (DE) analysis, we identify genes that are differentially expressed in

the given datasets. Integrating the possible cross-linked sites provided by the *default* mode, we can investigate the potential roles of RBP binding genes in cancer progression.

1.2 Used Programs Description

- **FastQC** *v0.11.5*
  FastQC is a quality control check tool. It provides a quick overview of your data and tells you which part might be problematic. It takes in BAM, SAM, or FASTQ files. It exports the result to an HTML based report.

- **Cutadapt** *2.8*
  Cutadapt finds and removes adapter sequences, primers, poly-A tails and other types of unwanted sequence from your high-throughput sequencing reads.

- **Prinseq-lite** *0.20.4*
  PRINSEQ generates summary statistics of your sequences in graphical format.

- **STAR** *2.7*
  STAR first generates genome index files. It then uses the created index files to map reads in FASTQ and FASTA files and outputs alignment BAM files which will be used later in the pipeline. The index file only needs to be built once.

- **UMI**
  UMI is used to remove PCR duplications to allow for an accurate crosslink site detection.

- **Samtools** *1.10*
  Samtools are used to combine isogenic replicates after reads alignment.

- **PureCLIP**
  PureCLIP simultaneously performs peak-calling and individual crosslink site detection.

- **Bioconductor**
  Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data using the R programming language.
  The Bioconductor libraries used are :
  - **genomation** : A pipeline for summary, annotation, and visualization of genomics data which we use to read in the bed file
  - **TxDb.Hsapiens.UCSC.hg19.knownGene** : Usage of UCSC generated annotation files as a TxDb object within R
  - **CompGO** : A pipeline which we use to annotate the bed file output from Pureclip
  - **org.Hs.eg.db** : To annotate entrez ids to HGNC symbols.
  - **biomaRt** : An interface to Ensembl annotations to convert the output gene lists to hgnc symbols.
  - **limma** : Provides tools to construct linear models for gene expression data, used to obtain DE genes from "sample_data.txt"
- **Differential Gene Expression Detector**
  This Python based script compares gene expression levels between cancer and non-cancer samples. Required Python modules are numpy, os, scipy.stats, sklearn.cluster, sys, csv and collections.

2 Setup

2.1 Install Dependencies

Users can install all the softwares and packages using following command.

```
conda create -n eclip python=3.6
conda activate eclip
sh download.sh
pip install -r requirements.txt
```

2.2 Create Directories

Directory structure looks like the following.

```
eclip
├── Predict_Diff_Expression.py
├── cancer
│   ├── eclip_cancer.txt
│   ├── shrna_cancer.txt
│   └── shrna_de_gene_name.txt
├── cancer.py
├── configs.py
├── configs.yml
├── convertBedToGeneNames.R
├── de_analysis.py
├── download.py
├── ensemblToHGNC.R
├── genome
│   ├── gencode.v33.annotation.gtf
│   └── GRCh38.p13.genome.fa
├── hypergeometricPval.R
├── index
├── peak
├── pipeline.py
├── raw_data
│   ├── KHDRBS1.tsv
│   ├── rep1
│   ├── rep2
│   └── shrna
├── read_process.py
├── requirements.txt
├── samtools
├── sbatch.sh
├── sbatch_template.sh
├── shrna_de
│   ├── shrna_de_gene_fdr_0.05.txt
│   └── shrna_de_gene_list.txt
└── tmp
    └── result.ebseq
```

```
├── result.ebseq.normalized_data_matrix
└── result.matrix
```

2.3 Download Datasets

- **Differential Gene Expression Detector**
  Following steps illustrate how to download raw data for Differential Gene Expression Detector and where to save them.

  1. Go to http://firebrowse.org/.
  2. Select one cohort (for example, BRCA, or SARC).
  3. Click on mRNASeq (Red bar).
  4. Download the file(s) having "RSEM_genes_normalized" in name.
  5. Unzip the downloaded file(s). Then unzip the .tar file(s).
  6. For each downloaded file, you will have two unzipped files. Copy the one starting with the name of the cohort (e.g. BRCA, or SARC) to the input directory.

2.4 Run pipline

User can modify configurations in config.yml, then simply run the whole pipline with the following command:

```
python pipeline.py
```

3 Script Details and How To's

3.1 FastQC

Take raw reads and check their quality. The results will be saved in "fastqc/before" folder. After cutting the adaptors, we check the quality again by performing fastqc. The result will be saved in "fastqc/after".

```
fastqc -o $PIPEDIR/rep1/fastqc/before $PIPEDIR/rep1/rawdata/
   read1.fastq $PIPEDIR/rep1/rawdata/read2.fastq
```

```
fastqc -o $PIPEDIR/rep1/fastqc/after $PIPEDIR/rep1/cutadapt/
   round2/read_1_adapterTrim_round2.fastq  $PIPEDIR/rep1/
   cutadapt/round2/read_2.adapterTrim_round2.fastq
```

3.2 Cutadapt

Cutadapt Round 1: Take paired-end raw reads from ENCODE as inputs. Run Cutadapt to trim off both 5' and 3' adapters on both reads.

```
cutadapt -f fastq --match-read-wildcards --times 1 -e 0.1 -O
   1 --quality-cutoff 6 -m 18 -a
   NNNNNAGATCGGAAGAGCACACGTCTGAACTCCAGTCAC -g
   CTTCCGATCTACAAGTT -g CTTCCGATCTTGGTCCT -A AACTTGTAGATCGGA
   -A AGGACCAAGATCGGA -A ACTTGTAGATCGGAA -A GGACCAAGATCGGAA -
   A CTTGTAGATCGGAAG -A GACCAAGATCGGAAG -A TTGTAGATCGGAAGA -A
    ACCAAGATCGGAAGA -A TGTAGATCGGAAGAG -A CCAAGATCGGAAGAG -A
   GTAGATCGGAAGAGC -A CAAGATCGGAAGAGC -A TAGATCGGAAGAGCG -A
   AAGATCGGAAGAGCG -A AGATCGGAAGAGCGT -A GATCGGAAGAGCGTC -A
   ATCGGAAGAGCGTCG -A TCGGAAGAGCGTCGT -A CGGAAGAGCGTCGTG -A
   GGAAGAGCGTCGTGT -o $PIPEDIR/rep1/cutadapt/round1/
   file_1_adapterTrim.fastq -p $PIPEDIR/rep1/cutadapt/round1/
   file_2_adapterTrim.fastq $PIPEDIR/rep1/raw_data/fastq/
   file_1.fastq $PIPEDIR/rep1/raw_data/fastq/file_2.fastq
```

Cutadapt Round 2: Take outputs from Round 1. Run Cutadapt to trim off the 3' adapters on read 2 in order to control for double ligation events.

```
cutadapt -f fastq --match-read-wildcards --times 1 -e 0.1 -O
   5 --quality-cutoff 6 -m 18 -A AACTTGTAGATCGGA -A
   AGGACCAAGATCGGA -A ACTTGTAGATCGGAA -A GGACCAAGATCGGAA -A
   CTTGTAGATCGGAAG -A GACCAAGATCGGAAG -A TTGTAGATCGGAAGA -A
   ACCAAGATCGGAAGA -A TGTAGATCGGAAGAG -A CCAAGATCGGAAGAG -A
   GTAGATCGGAAGAGC -A CAAGATCGGAAGAGC -A TAGATCGGAAGAGCG -A
   AAGATCGGAAGAGCG -A AGATCGGAAGAGCGT -A GATCGGAAGAGCGTC -A
   ATCGGAAGAGCGTCG -A TCGGAAGAGCGTCGT -A CGGAAGAGCGTCGTG -A
   GGAAGAGCGTCGTGT -o $PIPEDIR/rep1/cutadapt/round2/
   read_1_adapterTrim.fastq -p $PIPEDIR/rep1/cutadapt/round2/
   read_2_adapterTrim.fastq  $PIPEDIR/rep1/cutadapt/round1/
   read_1_adapterTrim.fastq $PIPEDIR/rep1/cutadapt/round1/
   read_2_adapterTrim.fastq
```

3.3 Prinseq-lite

Take in the reads after removing adaptors and check the statistics and quality summaries and show in graphic formats. It will provide distribution of length, occurrence of N's, tag sequence check and Dinucleotide odds ratios, includes the PCA plots.

```
perl prinseq-lite5.20.4/prinseq-lite.pl -verbose -fastq
   $PIPEDIR/rep1/cutadapt/read1_cutadaptR2_1.fastq -
   graph_data read1.gd -out_good null -out_bad null -
   graph_stats ld,ns,ts,dn


perl prinseq-lite-0.20.4/prinseq-lite.pl -verbose -fastq
   $PIPEDIR/rep1/cutadapt/read2_cutadaptR2_2.fastq -
   graph_data read2.gd -out_good null -out_bad null -
   graph_stats ld,ns,ts,dn
```

3.4 STAR

Preprocess: After alignment, we remove PCR duplicates based on the mapping position and random barcodes using UMI, which requires the read ID in the format $\_@HISEQ:87:00000000\_BARCODEread1\_$. Therefore we append the barcode to the read ID prior the mapping.

```
awk -v l=10 'BEGIN{OFS=FS=" "} substr($1, 1, 1) == "@" {print
    "@" substr($1, (l+3), 500) "_" substr($1, 2, l) " " $2 };
    substr($1, 1, 1) != "@" {print}; ' $PIPEDIR/rep1/cutadapt
   /round2/file_1_adapterTrim_round2.fastq > $PIPEDIR/rep1/
   cutadapt/round2/file_1_adapterTrim_round2.bc.fastq

awk -v l=10 'BEGIN{OFS=FS=" "} substr($1, 1, 1) == "@" {print
    "@" substr($1, (l+3), 500) "_" substr($1, 2, l) " " $2 };
    substr($1, 1, 1) != "@" {print}; ' $PIPEDIR/rep1/cutadapt
   /round2/file_2_adapterTrim_round2.fastq > $PIPEDIR/rep1/
   cutadapt/round2/file_2_adapterTrim_round2.bc.fastq
```

Build index: Generate genome indexes.

```
STAR --runThreadN 20 --runMode genomeGenerate --genomeDir
   $PIPEDIR/index --genomeFastaFiles $PIPEDIR/genome/
   GRCh38_p13_genome.fasta --sjdbGTFfile $PIPEDIR/genome/
   gencode_v33_annotation.gtf
```

Alignment: Map reads to the genome.

```
STAR --outFilterType BySJout --genomeDir $PIPEDIR/index --
   readFilesIn $PIPEDIR/rep1/cutadapt/round2/
   file_1_adapterTrim_round2.bc.fastq,$PIPEDIR/rep1/cutadapt/
   round2/file_2_adapterTrim_round2.bc.fastq --
   outFileNamePrefix $PIPEDIR/rep1/star/align/rep1_ --
   outFilterMultimapNmax 1 --alignSJoverhangMin 8 --
   alignSJDBoverhangMin 1 --outFilterMismatchNmax 999 --
   outFilterMismatchNoverLmax 0.04 --scoreDelOpen -1 --
   alignIntronMin 20 --alignIntronMax 1000000 --
   alignMatesGapMax 1000000 --outSAMtype BAM
   SortedByCoordinate --runThreadN 10 --alignEndsType
   EndToEnd
```

3.5 UMI

Preprocess: Build indexed file for the aligned bam file.

```
samtools  index  $PIPEDIR/rep1/star/align/rep1_Aligned.
   sortedByCoord.out.bam
```

Remove PCR duplicate: Groups PCR duplicates and deduplicates reads to yield one read per group.

```
. $HOME/.bashrc
conda activate umi_tools_env

umi_tools dedup -I $PIPEDIR/rep1/star/align/
   sam68_rep1_Aligned.sortedByCoord.out.bam --paired -S
   $PIPEDIR/rep1/prepeak/aligned.f.duplRm.bam

umi_tools dedup -I $PIPEDIR/rep2/star/align/
   sam68_rep2_Aligned.sortedByCoord.out.bam --paired -S
   $PIPEDIR/rep2/prepeak/aligned.f.duplRm.bam
```

3.6 Samtools

If the experiment involved more than two replicates, merge the bam files after alignment.

```
samtools merge -f $PIPEDIR/samtools/aligned.f.duplRm.pooled.
   bam $PIPEDIR/rep1/prepeak/aligned.f.duplRm.bam $PIPEDIR/
   rep2/prepeak/aligned.f.duplRm.bam
```

Index the merged bam file for next step - peakcalling.

```
samtools index $PIPEDIR/samtools/aligned.f.duplRm.pooled.bam
```

3.7 PureCLIP

Run PureCLIP for peak-calling in basic mode by using BAM and BAI files, the reference genome and a specified output file.

```
pureclip -i $PIPEDIR/samtools/rep1_2_sortedByCoord_merged.bam
    -bai  $PIPEDIR/samtools/rep1_2_sortedByCoord_merged.bam.
   bai -g  $PIPEDIR/genome/GRCh38_p13_genome.fasta -iv 'chr1;
   chr2;chr3;' -nt 10 -o $PIPEDIR/peaks/
   rep1_2_pureclip_crosslink_sites.bed
```

3.8 RSEM

Run RSEM to do differential expression analysis for shRNA knockdown targeting a specific gene. First, generate data matrix from gene quantifications files. Then do differential expression analysis with generated matrix. Finally, users can specify the FDR threshold to find genes which are differentially expressed after shRNA knockdown.

```
rsem-generate-data-matrix control_rep1.tsv control_rep2.tsv
   rep1.tsv rep2.tsv > result.matrix
rsem-run-ebseq result.matrix 2,2 result.ebseq
rsem-control-fdr result.ebseq 0.05 de_gene_fdr_0.05.txt
awk 'NR!=1{ print $1}' de_gene_fdr_0.05.txt > de_gene_list.
   txt
```

3.9 Bioconductor

Usage of these scripts requires the installation of R, a version of Java with the same arch (x64 R requires a version of x64 Java), and the installation of the six Bioconductor libraries mentioned above. All scripts below do have installation codes, but these vary depending on user OS and other settings, so it is assumed all required libraries are installed prior to running the scripts.

Please refer to: https://www.bioconductor.org/install/index.html#install-bioconductor-packages for further help in installing these libraries.

For example, windows users can try running the following command to run a R script from command line.

```
"C:\Program Files\R\R-3.6.1\bin\Rscript.exe" script.R
   args1 args2 ....
```

Replace "C:/Program Files/R/R-3.6.1/bin/Rscript.exe" with the path to Rscript.exe on your system, and script.R with one of three scripts below. There are three R scripts that the user should be familiar with

- **convertBedToGeneNames.R**

```
"C:\Program Files\R\R-3.6.1\bin\Rscript.exe"
   convertBedToGeneNames.R control.bed crosslink.
   bed crosslink_genes.txt
```

  This script takes three arguments: control.bed (input file), crosslink.bed (input file), and crosslink_genes.txt (output file). This script reads in the two bed file inputs, annotates genes to all the locations, and removes the false positive hits of the control bed file from the crosslink.bed genes and saves the list of genes in crosslink_genes.txt

- **ensemblToHGNC.R**

```
"C:\Program Files\R\R-3.6.1\bin\Rscript.exe"
   ensemblToHGNC.R shrna_de_genes_list.txt
   shrna_de_converted.txt
```

  This script reads the txt file "shrna_de_genes_list.txt" (output from *differential expression analysis* mode) and converts the ENSG gene ids to gene symbols and saves it in "shrna_de_converted.txt". This represents the list of genes that are differentially expressed before and knockdown of the RBP of interest (in this case sam68).

- **hypergeometricPval.R**

```
"C:\Program Files\R\R-3.6.1\bin\Rscript.exe"
   hypergeometricPval.R crosslink_genes.txt
   shrna_de_converted.txt newdat.txt
```

  This function computes the hypergeometric intersection p value for the intersection between differentially expressed in the gene expression dataset provided by the user (which is the 3rd argument - newdat.txt), and the gene lists we get as outputs from the previous two R scripts (the first argument is crosslink_genes.txt and the 2nd is shrna_de_converted.txt). If no user-defined 3rd argument is provided, the default "sample_data.txt"

is used.

The user provided input file, however, should have all samples corresponding to rows, and genes (HGNC symbol) in the columns, with one additional column called "label" that denotes the class label of the sample.

For example, "sample_data.txt" uses processed BRCA data, and the label column indicates if the sample is a control or cancer sample, and this information is used to conduct differential expression analysis. Generation of "sample_data.txt" is elucidated in the script **sampledatgeneration.R**.

3.10 Differential Gene Expression Detector

Given a list of genes and raw gene expression files downloaded according to section 2.3, 'Differential Gene Expression Detector' compares the gene expression between cancer and non-cancer samples and based on the analysis, predicts if each individual gene in the gene list is related to cancer or not.

This part requires a single python script "Predict_Diff_Expression.py" in base directory. User needs to provide a raw gene expression data in the input directory. Any data, including file with gene names that has been converted in section 3.9, must also be in the input directory. For directory structure, please refer to section 2.2.

The script detects anomaly in gene expression levels and compares the cancer cases to non-cancer ones to predict differential expression. It analyzes the files in the input directory and outputs the differentially expressed genes. Simultaneously, it also saves differentially expressed genes for each cancer category into input directory for future use. If the differentially expressed genes for each cancer category are already saved in the directory, the script does not look at the raw files and analyze afresh (unless given *–force-analyze* option, which we will discuss below). Thus, after the first run, the following runs are much faster.

The script is equipped with two manadatory inputs: (1) '–input-file=x' and (2) '–output-file=y.csv' (each option has two '-'s at start). x and y denote

input and output file paths. In both cases, we consider relative path to the files. Output file has to be specified in csv format.

The script is equipped with three options: (1) '–help' (2) '–use-clustering' and (3) '–force-analyze' (each option has two '-'s).

First option '–help' shows the manual of the script.

Second option '–use-clustering' is useful when a gene does not have differential expression, but belongs to a cluster with genes having differential expression levels in cancer vs non-cancer samples.

Third option '–force-analyze' reanalyzes gene expressions even though the enriched data is present in directory. The script ignores any other option provided and throws error accordingly.

The final output of the script is saved as specified by user. Each row signifies each query gene in the gene list. Each column signifies one cancer category. If a particular cell under row 'y' and column 'z' is empty, that indicates the gene 'y' does not have differential expression related to the cancer category 'z'. Otherwise, the related genes with names starting as 'y' having differential expressions in cancer vs non-cancer sample are concatenated with ';'.

```
python Predict_Diff_Expression.py --help

python Predict_Diff_Expression.py --input-file=x --output-
    file=y.csv

python Predict_Diff_Expression.py --input-file=x --output-
    file=y.csv --force-analyze

python Predict_Diff_Expression.py --input-file=x --output-
    file=y.csv --use-clustering

python Predict_Diff_Expression.py --input-file=x --output-
    file=y.csv --force-analyze --use-clustering
```

Things to remember:

1. Clustering will take a lot of time and memory because of kernel applied distance calculation. If the number of genes are more than 20,000 and number of samples are more than 10000, clustering can go on for upto six

hours(no GPU library was used). The creator has observed upto 28 GB of RAM usage during clustering.

2. Suppose you have run the script once. Corresponding enriched differential gene lists are created while the script was running. Now you have downloaded one more raw expression file and you want to run the script. If you run it without any options, it will take the enriched differential gene lists, which it saved in the first run, from the input path and calculate output based on that. If you want to analyze afresh, use '–force-analyze' option.

3. There is no verbose toggle in the script, i.e., logs will be printed on terminal. This is done to ensure smooth progress of the script.

4 Outputs

4.1 Default Mode Output

```
chr1    629930  629931  3       7.98235 +       [score_CL=7.98235;score_E=121.741;score_B=46.1763;score_UC=7.98235]
chr1    1317037 1317038 3       3.0966  +       [score_CL=3.0966;score_E=109.332;score_B=23.8956;score_UC=3.0966]
chr1    1318575 1318576 3       11.7122 +       [score_CL=11.7122;score_E=125.684;score_B=50.6248;score_UC=11.7122]
chr1    1318576 1318577 3       11.9687 +       [score_CL=11.9687;score_E=125.673;score_B=50.8707;score_UC=11.9687]
```

Figure 2: BED file after peakcalling

The main output from the default mode is a BED file containing a crosslink site associated with a score in each row. The columns from left to right are chromosome number, start, end, state, score, and strand.

4.2 Differential Expression Analysis Mode Output

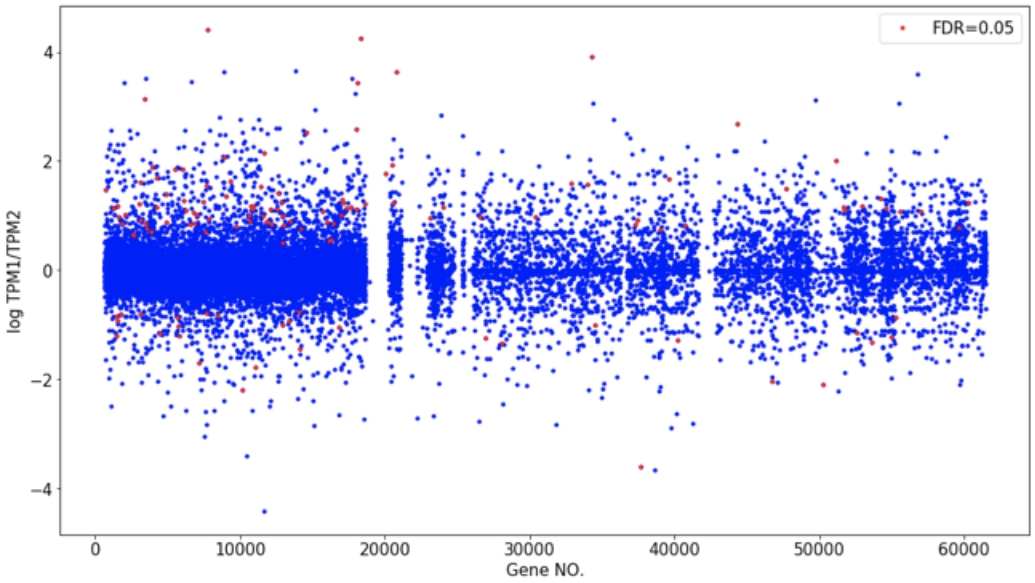

Figure 3: shRNA knowdown differential expression analysis

The figure provides an example of running differential expression analysis software (EBSeq) to identify genes related with the RBP - SAM68. When the false discovery rate (FDR) is set to 0.05, the red dots represent the genes that are significantly differentially expressed between the RBP gene knockout dataset and the control dataset.

4.3 Cancer Analysis Mode Output

4.3.1 Converted gene list outputs

The figure below represents the txt file output from convertBedToGeneNames.R (pureclip_converted.txt) and ensemblToHGNC.R (sam68degs_converted.txt). These files are used as inputs to compute the hypergeometric p.value with the gene expression data and for the additional cancer downstream analysis

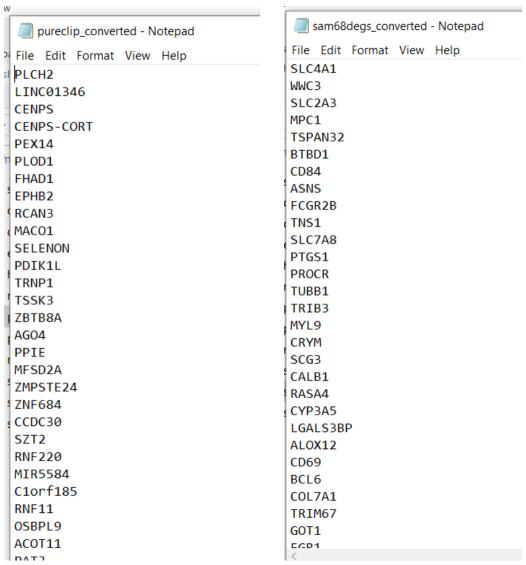

Figure 4: Converted gene lists from the R script

4.3.2 p.value txt file

The below figure represents the txt file output "pvalues.txt" from running the script hypergeometricPval.R

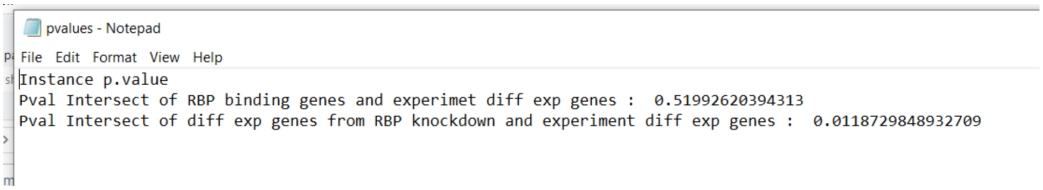

Figure 5: p.value output

4.3.3 Differential Gene Expression Detector

In the following figure, we have an example of final csv file generated after running Predict_Diff_Expression.py. Gene names in the first column are the query genes. Top header entries contain cancer categories, e.g. BRCA, GBMLGG, KIPAN and so on. Cells contain gene names. Our pipeline can

identify the gene names with prefixes. For example, BRCA instead of BRCA1 suffices.

	A	B	C	D	E	F	G
1	gene_names	BRCA	GBMLGG	KIPAN	KIRC	STES	
2	SLC4A1	SLC4A10;SLC4A11;SLC4A1AP;		SLC4A10;SLC4A11	SLC4A10;SLC4A11	SLC4A10;SLC4A11	
3	WWC3	WWC3		WWC3	WWC3	WWC3	
4	SLC2A3	SLC2A3		SLC2A3	SLC2A3		
5	MPC1						
6	TSPAN32	TSPAN32		TSPAN32	TSPAN32	TSPAN32	
7	BTBD1	BTBD10;BTBD11;BTBD12;BTE		BTBD10;BTBD11;	BTBD10;BTBD11;	BTBD12;BTBD16;E	
8	CD84	CD84		CD84	CD84	CD84	
9	ASNS	ASNSD1;ASNS		ASNSD1;ASNS	ASNSD1;ASNS		
10	FCGR2B	FCGR2B		FCGR2B	FCGR2B	FCGR2B	
11	TNS1	TNS1		TNS1	TNS1	TNS1	
12	SLC7A8	SLC7A8		SLC7A8	SLC7A8	SLC7A8	
13	PTGS1					PTGS1	
14	PROCR	PROCR		PROCR	PROCR	PROCR	
15	TUBB1	TUBB1		TUBB1	TUBB1		
16	TRIB3	TRIB3		TRIB3	TRIB3	TRIB3	
17	MYL9	MYL9		MYL9		MYL9	
18	CRYM	CRYM		CRYM	CRYM		
19	SCG3	SCG3				SCG3	
20	CALB1	CALB1		CALB1	CALB1		
21	RASA4	RASA4			RASA4	RASA4P;RASA4	

Figure 6: Final CSV output denoting genes and cancer categories