THE

RNAseq Pipeline

MANUAL

Timo Lassmann

Dec. 5th 2013

Contents

Chapter 1

Introduction

Analysis of next generation sequencing datasets is complicated and often hard to reproduce. This package contains a pipeline system to process RNAseq data in a *totally* reproducible manner. Moreover, the package contains all the code, scripts and parameters to run the analysis. Typing **make** in the root directory will actually start the entire analysis automatically.

For convenience during development all steps are executed within **makefiles** ensuring that computationally expensive initial steps do not have to be repeated if downstream analysis steps are modified. The makefiles also keep the documentation up to date.

Finally I use the literal programming paradigm to describe and highlight the important steps. This file actually contains the code needed for the analysis.

Chapter 2

Some details

The package is organized as follows:

1. the configure.ac file contains the instructions to check the versions of external programs used in the pipeline. As a default if the installed version is higher than the required version a warning will be produced. Otherwise the configuration will fail.
2. once configured, make will compile the C code used to extract code from this document.
3. After all programs are extracted ro the **src** directory, a makefile will call the **run.mk** makefile and start the pipelines.

Chapter 3

Installation and Usage

To install the package simple follow the steps below:

```
tar -zxvf rnaseq_pipeline-XX.XX.tgz
cd rnaseq_pipeline-XX.XX
./configure
```

WARNING: Typing **make** now will not only compile or extract code but will actually start running pipelines on the input directories specifies below

```
make
```

To apply the same pipeline to different data or change parameter settings simply edit the corresponding sections in this file.

Chapter 4

Results and CODE

The pipeline consists of one main makefile called **run.mk**. Within this files larger analysis units are individually executed using secondary **makefiles**. Output files from one step form the dependency of the next step.

4.1 Pipeline Paramenters and Settings

Input Data

We will run the pipeline on the following two MiSeq runs. Later we repeat with the 150bp paired end HiSeq rapid runs.

Tophat Command

The Tophat command used is:

```
    $(TOPHAT) -p 1 --b2-very-sensitive  --output-dir=$(OUTDIR)/$@  --
       transcriptome-index=$(GENCODE_DIR) $(GENOME) $^
```

Cufflinks Command

The actual Cufflinks command used.

```
    $(CUFFLINKS) -o $(OUTDIR)/$@   -g $(GENCODE_GTF)  $^/accepted_hits.bam
```

Cuffquant Command

The actual Cuffquant command used.

```
    $(CUFFQUANT) -o $(OUTDIR)/$@   $(GENCODE_GTF)  $^/accepted_hits.bam
```

Paths

Set the variables below according to your installation.

```
TOPHAT=/usr/local/bin/tophat
CUFFLINKS=/usr/local/bin/cufflinks
CUFFQUANT=/usr/local/bin/cuffquant

GENCODE_GTF=../src/customgencode.gtf
```

```
GENCODE_DIR=../src/customgencode

GENOME=../src/genome
```

SummaryDirectory

This directory will collect all summary files across multiple runs. This code snippet is used in mulitple post-processing scripts.

```
SUMMARYDIR=../summary/
```

Files

Here is the code of the actual scripts used in the analysis.

File: run.mk

The makefile **run.mk** executes the pipeline **process_run.mk** on all target libraries. Afterwards additional makefiles collect output files and summarize the results.

```
.PHONY: mapping stats genefpkm

all: mapping stats normtable

mapping:
    $(MAKE) -f makegenome.mk
    $(MAKE) -f mirrordata.mk
    $(MAKE) -k -j 80 indir=../data/  -f process_run.mk

normtable:
    $(MAKE) -f make_norm_table.mk

stats:
    $(MAKE) -f get_mapping_stats.mk

genefpkm:
    $(MAKE) -f join_gene_fpkm.mk
```

File: process_run.mk

The makefile to process the output of one HiSeq or MiSeq run. The data is assumed to be already demultiplexed.

```
## Paths

.PHONY: directories alignment_stats
```

Check of input parameteres are set.

```

```

```
ifndef indir
$(error indir is not set)
endif

OUTDIR=../mapping/

dirs := $(indir)

files := $(foreach dir,$(dirs),      $(sort $(wildcard $(dir)/*fq)))
```

Set targets for TopHat and Cufflinks.

```
TOPHAT_OUT=$(notdir $(patsubst %.fq,%_tophat_out,  $(files)))


CUFFQUANT_OUT=$(notdir $(patsubst %.fq,%_cuffquant_out,  $(files)))
```

Add directories to **vpath** to allow for targets and prerequisites to be in different directories.

```
vpath %.fq  $(indir)
vpath %.fq  $(indir)
vpath %_tophat_out  $(OUTDIR)

vpath %_cuffquant_out  $(OUTDIR)

all: directories $(TOPHAT_OUT) $(CUFFQUANT_OUT)
    @echo $(OUTDIR)

%_tophat_out: %.fq
## Tophat Command



%_cuffquant_out: %_tophat_out
## Cuffquant Command


MKDIR_P = mkdir -p

directories: $(OUTDIR)

$(OUTDIR):
    $(MKDIR_P) $(OUTDIR)
```

File: join_gene_fpkm.mk

Makefile to create the basic gene based fpkm expression table. For some reason adding awk code directly into this makefile did not work. Therefore I call the awk script **join_gene_fpkm.awk** from this file.

```
## SummaryDirectory

GENEFPKM=$(shell find ../mapping/ -name 'genes.fpkm_tracking')
```

```
ifeq "$(GENEFPKM)" ""
all:
    @echo "Warning No FPKM files found."
else
all:directories $(SUMMARYDIR)/all_gene.fpkm_tracking

$(SUMMARYDIR)/all_gene.fpkm_tracking: $(GENEFPKM)
    awk -f   join_gene_fpkm.awk $^ > $@
endif

MKDIR_P = mkdir -p

directories: $(SUMMARYDIR)

$(SUMMARYDIR):
    $(MKDIR_P) $(SUMMARYDIR)
```

File: get_mapping_stats.mk

Collects the mapping statistics from all mapped libraries.

```
## SummaryDirectory

ALIGNMENTSTATS=$(shell find ../mapping/ -name 'align_summary.txt')

ifeq "$(ALIGNMENTSTATS)" ""
all:
    @echo "Warning No Mapping Stats"
else
all:directories $(SUMMARYDIR)/all_mapping_stats.csv $(SUMMARYDIR)/
   all_mapping_stats_counts.csv
    @echo "Done."

$(SUMMARYDIR)/all_mapping_stats.csv: $(ALIGNMENTSTATS)
    awk '{if($$2 == "concordant" && $$3 == "pair"){sub("_tophat_out/
       align_summary.txt","",FILENAME); printf "%s\t%s\n" ,FILENAME,$$1
          }}' $(ALIGNMENTSTATS) >$(SUMMARYDIR)/all_mapping_stats.csv

$(SUMMARYDIR)/all_mapping_stats_counts.csv: $(ALIGNMENTSTATS)
    awk '{if($$1 == "Aligned" && $$2 == "pairs:"){sub("_tophat_out/
       align_summary.txt","",FILENAME); printf "%s\t%s\n" ,FILENAME,$$3
       }}' $(ALIGNMENTSTATS) >$(SUMMARYDIR)/all_mapping_stats_counts.csv.
       csv



endif

MKDIR_P = mkdir -p

directories: $(SUMMARYDIR)

$(SUMMARYDIR):
    $(MKDIR_P) $(SUMMARYDIR)
```

File: mirrordata.mk

```
DATADIR=../data


FILES=$(shell find /data/lassmann/benchmark/STRT/extracted  -type f -name
   '*.fq' -size +1M)


FASTQDIRS = $(dir $(FILES))

FASTQFILES = $(notdir $(FILES))

TARGETS = $(addprefix $(DATADIR)/, $(FASTQFILES))



ifeq "$(words  $(FILES))" "0"
all:
    @echo "Warning No Fastq files found."
else
all: directories $(TARGETS)
endif


$(DATADIR)/%.fq:   %.fq
    cp -av $< $@

MKDIR_P = mkdir -p

directories: $(DATADIR)

$(DATADIR):
    $(MKDIR_P) $(DATADIR)

vpath %.fq $(FASTQDIRS)
```

File: makegenome.mk

```



TEST=$(shell find . -name 'getgenomeucsc.sh')

ifeq "$(TEST)" ""
all:
    @echo "Warning No files for de-multiplexing found..."
else
all:  genome.1.bt2 customgencode.1.bt2


endif

GRCm38.p2.genome.fa:
```

```
    wget ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_mouse/release_M2/
       GRCm38.p2.genome.fa.gz
    gunzip GRCm38.p2.genome.fa.gz

gencode.vM2.annotation.gtf:
    wget ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_mouse/release_M2/
       gencode.vM2.annotation.gtf.gz
    gunzip gencode.vM2.annotation.gtf.gz

genome.fa: GRCm38.p2.genome.fa
    ./makecustomgenome GRCm38.p2.genome.fa -nogtf GRCm38.p2.genome.fa -o
       genome

customgencode.gtf: genome.fa  gencode.vM2.annotation.gtf
    cat gencode.vM2.annotation.gtf  | awk '{if($$26~/1|2/){  if($$3 !="
       gene"){ print $$0}}}'  > tmp2.gtf
    grep '>' genome.fa | awk '{x = substr($$0,2) ;print x}' > chromosomes.
       txt
    grep -f  chromosomes.txt tmp2.gtf > customgencode.gtf

genome.1.bt2: customgencode.gtf genome.fa
    bowtie2-build genome.fa genome

customgencode.1.bt2: customgencode.gtf
    tophat -G customgencode.gtf --transcriptome-index=.  genome
```

File: make_norm_table.mk

Collects the mapping statistics from all mapped libraries.

```
.PHONY:cuffnorm

## SummaryDirectory

CUFFNORM=/usr/local/bin/cuffnorm

GENCODE_GTF=../src/customgencode.gtf
QUANTDIR=$(SUMMARYDIR)/quantknowntable

INFILES=$(shell find . -name 'abundances.cxb' | awk 'BEGIN{print "
   sample_name group"} {split($$1,a,"_cuffquant_out"); printf "%s\t%s\n",
   $$1 ,substr(a[1],3,50 ) }')


ifeq "$(INFILES)" ""
all:
    @echo "Warning No Qumant found"
else
all: $(QUANTDIR)/sample_sheet.txt cuffnorm
    echo "Done."

endif

cuffnorm:
```

```
    $(CUFFNORM) -p 80 --use-sample-sheet -o $(QUANTDIR) $(GENCODE_GTF) $(
        QUANTDIR)/sample_sheet.txt

$(QUANTDIR)/sample_sheet.txt: directories
    find ../mapping/ -name 'abundances.cxb' | awk 'BEGIN{print "
        sample_name group"} {split($$1,a,"_cuffquant_out"); printf "%s\t%s\
        n" ,$$1 ,substr(a[1],3,50 ) }' > $(QUANTDIR)/sample_sheet.txt



MKDIR_P = mkdir -p

directories: $(SUMMARYDIR) $(QUANTDIR)

$(QUANTDIR):
    $(MKDIR_P)  $(QUANTDIR)

$(SUMMARYDIR):
    $(MKDIR_P) $(SUMMARYDIR)
```

4.2 Postanalysis

```
library(ggplot2)
library(reshape)
library(gplots)
library(randomForest)
library("foreach")
library("doSNOW")
registerDoSNOW(makeCluster(80, type="SOCK"))

set.seed(42)




fpkm_matrix <- read.delim("genes.fpkm_table", row.names=1)

dat = as.matrix(fpkm_matrix[,2:ncol(fpkm_matrix)])
class(dat) <- "numeric"
dat = t(dat)


spikes = as.data.frame(t(fpkm_matrix[1:10,2:ncol(fpkm_matrix)]) )
spikes = cbind(spikes, substr( rownames(spikes) ,11,19))


colnames(spikes) = c("Cherry","Venus","Spike2", "Spike6","Spike4","Spike5"
    ,"Spike7","Spike1", "Spike3","Spike8","Run")

ggplot(spikes, aes(Spike1+ 1e-05, Spike4+ 1e-05)) + geom_point() + facet_
    wrap("Run", scale = "free") + scale_x_log10() + scale_y_log10()
```

```
ggsave("Spike1_4.pdf")

ggplot(spikes, aes(Cherry+ 1e-05, Venus+ 1e-05)) + geom_point() + facet_
    wrap("Run", scale = "free") + scale_x_log10() + scale_y_log10()
ggsave("Cherry_Venus.pdf")


l = melt(spikes)

ggplot(data = l, aes(x = variable, y = value + 1e-05)) + geom_boxplot() +
    coord_flip() + scale_y_log10("Unit is fpkm") + facet_wrap("Run") + xlab
    ("")
ggsave("spikefpkm.pdf")







library(ggplot2)
library(gplots)
library(randomForest)
library("foreach")
library("doSNOW")
registerDoSNOW(makeCluster(80, type="SOCK"))

set.seed(42)


nzmean <- function(x) {
    if (all(x==0)) 0 else mean(x)
}


nzcount <- function(x) {
    sum(x !=0);
}

#mean = apply(x,1,nzmean)

#nzcount <- apply(x,1, nzcount)


#either cell_cycle_genes_image.csv

#mat = read.table("cell_cycle_genes_image.csv",header =T,row.names = 1,sep
    = "\t")

#or

mat = read.table("all_gene.fpkm_tracking_and_Imaging.csv",header =T,row.
    names = 1,sep = "\t")
mat = t(mat)
```

```
#end

mat = mat[rownames(mat) != "Description",];

imagenamelist <- c("SpotTotalAreaCh2","SpotAvgAreaCh2","SpotTotalIntenCh2"
   ,"SpotAvgIntenCh2","SpotTotalAreaCh3","SpotAvgAreaCh3","
   SpotTotalIntenCh3","SpotAvgIntenCh3","ErrorType","Description","
   ImageKeep","tracking_id.gene_short_name");

rnadata = mat[!(rownames(mat) %in% imagenamelist),];


imagedata = mat[(rownames(mat) %in% imagenamelist),];



dat = as.matrix(rnadata)
class(dat) <- "numeric"
dat = t(dat)


rf <- foreach(ntree = rep(125, 80), .combine = combine, .packages = "
   randomForest") %dopar%  randomForest(dat,      proximity = TRUE,ntree =
    ntree)



fit = cmdscale(1 - rf$proximity)


kmeans = kmeans(fit,2)

plot(fit,col = kmeans$cluster)

smallest_cluster_size = min(kmeans$size)


outlier = kmeans$cluster [kmeans$size[kmeans$cluster] ==smallest_cluster_
   size]


names(outlier);

rnadata_filtered =rnadata[,!(colnames(rnadata) %in% names(outlier))]

#heatmap.2( log(dat+0.01),density.info="none",trace = c("none"))


imagedata_filtered = imagedata[,!(colnames(rnadata) %in% names(outlier))]

good_image = imagedata_filtered["ImageKeep",]  == 1;


rnadata_filtered = rnadata_filtered[,good_image]
imagedata_filtered = imagedata_filtered[,good_image]
```

```
dat = as.matrix(rnadata_filtered)
class(dat) <- "numeric"



###extra filtering = remove genes with lower than 10 mean fpkm expression


dat_mean = apply(dat,1,nzmean)


rnadata_filtered2 = rnadata_filtered[dat_mean > 10,]

###


SpotAvgIntenCh2 = imagedata_filtered["SpotAvgIntenCh2",]
SpotAvgIntenCh3 = imagedata_filtered["SpotAvgIntenCh3",]

class(SpotAvgIntenCh2) <- "numeric"

class(SpotAvgIntenCh3) <- "numeric"

image_ratio = log2((SpotAvgIntenCh2+0.01) / (SpotAvgIntenCh3+0.01))
rownames(image_ratio) = "image_ratio"
dat = as.matrix(rnadata_filtered2)
class(dat) <- "numeric"




dat= rbind(dat,image_ratio)
frame = data.frame(t(dat))
y <- frame[, ncol(frame)]
x <- frame[,1:(ncol(frame)-1)]
rf <- randomForest(x,y, do.trace = 100,      importance = TRUE, ntree = 1,
   type=regression)

#> library("foreach")
#> library("doSNOW")
#> registerDoSNOW(makeCluster(80, type="SOCK"))

#> x <- matrix(runif(500), 100)
#> y <- gl(2, 50)

#> rf <- foreach(ntree = rep(250, 4), .combine = combine, .packages = "
   randomForest") %dopar%
#+    randomForest(x, y, ntree = ntree)
#> rf
#Call:
#randomForest(x = x, y = y, ntree = ntree)
#Type of random forest: classification
```

```
rf <- foreach(ntree = rep(125, 80), .combine = combine, .packages = "
    randomForest") %dopar%    randomForest(x, y, do.trace = 100,
    importance = TRUE, type=regression,ntree = ntree)




jpeg(filename="ALLRFaccuracy.jpg", width=800, height=800, pointsize=12,bg=
    "white");
plot( predict(rf), y)
abline(c(0,1),col=2)

dev.off();

jpeg(filename="ALLRFimpvar.jpg", width=800, height=800, pointsize=12,bg="
    white");

varImpPlot(rf)
dev.off();

good_genes = rf$importance[,1]  > 0.1

plotdata = rnadata_filtered2[good_genes,]

#plotdata = log(plotdata+0.1)

plotdata = rbind(plotdata,image_ratio)
gaga = as.matrix(plotdata);

class(gaga) <- "numeric"
heatmap.2( gaga,density.info="none",trace = c("none"))

test = ( cor(t(gaga),method="spearman"))


jpeg(filename="ImportantGenesHeatmap.jpg", width=1200, height=800,
    pointsize=12,bg="white");
heatmap.2( test,density.info="none",trace = c("none"),col=redgreen(1000))
dev.off()

## negative contriolll...




sapply (1:ncol(x), function (col) x[,col]<<-sample(x[,col]))



###ok
```