

systemPipeR: utilities for building NGS analysis pipelines

Thomas Girke
Email contact: thomas.girke@ucr.edu

May 28, 2014

1 Introduction

systemPipeR is a pipeline for running command-line software, such as NGS aligners, on both single machines or compute clusters. It supports both interactive job submissions or batch submissions to queuing systems of clusters (tested only with Torque). *systemPipeR* can be used with most command-line aligners such as TopHat 2 (Kim et al., 2013) and Bowtie 2 (Langmead and Salzberg, 2012).

Contents

1	Introduction	1
2	Getting Started	2
2.1	Installation	2
2.2	Loading the Package and Documentation	2
3	Structure of targets file	2
4	Structure of param file and SYSargs container	3
5	Workflow	4
5.1	Define environment settings and samples	4
5.2	Alignment with Tophat 2	4
5.3	Create symbolic links for viewing BAM files in IGV	4
5.4	Alignment with Bowtie 2 (here for miRNA profiling experiment)	5
5.5	Read counting for mRNA profiling experiments	5
5.6	Read counting for miRNA profiling experiments	5
5.7	Correlation analysis of samples	5
5.8	DEG analysis with <i>edgeR</i>	6
6	Version Information	6
7	Funding	6
8	References	6

2 Getting Started

2.1 Installation

The R software can be downloaded from CRAN (<http://cran.at.r-project.org/>) and the *systemPipeR* package from GitHub (<https://github.com/tgirke/systemPipeR>). The *systemPipeR* package can be installed from R using the `install.packages` command after downloading and uncompressing the package directory.

```
> system("R CMD build systemPipeR") # Builds package
> install.packages("systemPipeR.X.X.X.tar.gz", repos=NULL, type="source") # Installs the package
```

2.2 Loading the Package and Documentation

```
> library("systemPipeR") # Loads the package
> library(help="systemPipeR") # Lists all functions and classes
> vignette("systemPipeR") # Opens this PDF manual from R
```

3 Structure of targets file

The targets file defines all FASTQ files and sample comparisons of an analysis workflow. The following shows the format of a sample targets file provided by this package.

```
> library(systemPipeR)
> targetspath <- paste0(system.file("extdata", package="systemPipeR"), "/targets.txt")
> read.delim(targetspath, comment.char = "#")
```

	FileName	SampleName	Factor	SampleLong	Experiment	Date
1	/my/path/CGA.fq.gz	C1	C	Ctrl1	1	11-Oct-13
2	/my/path/TTA.fq.gz	C2	C	Ctrl1	1	11-Oct-13
3	/my/path/ACT.fq.gz	B1	B	Treat1	1	11-Oct-13
4	/my/path/GCC.fq.gz	B2	B	Treat1	1	03-Mar-14
5	/my/path/CGA.fq.gz	D1	D	Treat2	1	03-Mar-14
6	/my/path/TTA.fq.gz	D2	D	Treat2	1	03-Mar-14
7	/my/path/TGA.fq.gz	E1	E	Treat3	1	03-Mar-14
8	/my/path/ACA.fq.gz	E2	E	Treat3	1	03-Mar-14

Structure of targets file for paired end (PE) samples.

```
> library(systemPipeR)
> targetspath <- paste0(system.file("extdata", package="systemPipeR"), "/targetsPE.txt")
> read.delim(targetspath, comment.char = "#")
```

	FileName1	FileName2	SampleName	Factor	SampleLong	Experiment	Date
1	/my/path/CGA1.fq.gz	/my/path/CGA2.fq.gz	C1	C	Ctrl1	1	11-Oct-13
2	/my/path/TTA1.fq.gz	/my/path/TTA2.fq.gz	C2	C	Ctrl1	1	11-Oct-13
3	/my/path/ACT1.fq.gz	/my/path/ACT2.fq.gz	B1	B	Treat1	1	11-Oct-13
4	/my/path/GCC1.fq.gz	/my/path/GCC2.fq.gz	B2	B	Treat1	1	03-Mar-14
5	/my/path/CGA1.fq.gz	/my/path/CGA2.fq.gz	D1	D	Treat2	1	03-Mar-14
6	/my/path/TTA1.fq.gz	/my/path/TTA2.fq.gz	D2	D	Treat2	1	03-Mar-14
7	/my/path/TGA1.fq.gz	/my/path/TGA2.fq.gz	E1	E	Treat3	1	03-Mar-14
8	/my/path/ACA1.fq.gz	/my/path/ACA2.fq.gz	E2	E	Treat3	1	03-Mar-14

Comparisons are defined in the header lines of the targets starting with '# <CMP>'. The function `readComp` imports the comparison and stores them in a list.

```
> readComp(file=targetspath, format="vector", delim="-")
```

```
$CMPset1
[1] "C-B" "C-D" "C-E"

$CMPset2
[1] "B-D" "D-E"
```

4 Structure of param file and SYSargs container

The param file defines the parameters of the command-line software. The following shows the format of a sample param file provided by this package.

```
> parampath <- paste0(system.file("extdata", package="systemPipeR"), "/tophat.param")
> read.delim(parampath, comment.char = "#")
```

	PairSet	Name	Value
1	modules	<NA>	bowtie2/2.1.0
2	modules	<NA>	tophat/2.0.8b
3	software	<NA>	tophat
4	cores	-p	4
5	other	<NA>	-g 1 --segment-length 25 -i 30 -I 3000 -G ./data/My_species.gff
6	outfile1	-o	<FileName1>
7	outfile1	path	./results/
8	outfile1	remove	<NA>
9	outfile1	append	.tophat/accepted_hits.bam
10	outfile1	outextension	.tophat/accepted_hits.bam
11	reference	<NA>	./data/My_genome.fasta
12	infile1	<NA>	<FileName1>
13	infile1	path	<NA>
14	infile2	<NA>	<FileName2>
15	infile2	path	<NA>

The systemArgs function imports the definitions of both the param file and the targets file, and stores all relevant information as SYSargs object.

```
> args <- systemArgs(sysma=parampath, mytargets=targetspath)
> args
```

An instance of 'SYSargs' for running 'tophat' on 8 samples

Several accessor functions are available that are named after the slot names of the SYSargs object class.

```
> names(args)

[1] "modules" "software" "cores" "other" "reference" "infile1" "infile2"
[8] "outfile1" "sysargs" "outpaths"

> modules(args)

[1] "bowtie2/2.1.0" "tophat/2.0.8b"

> cores(args)

[1] 4

> outpaths(args)[1]
```

"/rhome/tgirke/Projects/github/systemPipeR/vignettes/results/CGA1.fq.gz.tophat/accepted_hits.bam"

C1

```
> sysargs(args)[1]
```

```
"tophat -p 4 -g 1 --segment-length 25 -i 30 -I 3000 -G ./data/My_species.gff -o /rhome/tgirke/Projects/gitl"
```

The content of the param file can be returned as JSON object as follows (requires *rjson* package).

```
> systemArgs(sysma=parampath, mytargets=targetspath, type="json")
```

```
[1] "{\"modules\":{\"n1\":\"\",\"v2\":\"bowtie2/2.1.0\",\"n1\":\"\",\"v2\":\"tophat/2.0.8b\"},\"software\""
```

5 Workflow

5.1 Define environment settings and samples

Load package:

```
> library(systemPipeR)
```

Subset input targets file as needed. Note: for `qsubRun` the targets file needs to contain absolute paths to FASTQ files in the 'FileName' column.

```
> targets <- read.delim(targetspath, comment.char = "#")
> write.table(targets, "targets_run.txt", row.names=FALSE, quote=FALSE, sep="\t")
```

Construct `SYSargs` object from param and targets files.

```
> args <- systemArgs(sysma=parampath, mytargets="targets_run.txt")
```

5.2 Alignment with Tophat 2

Build Bowtie 2 index.

```
> system("bowtie2-build ./data/mygenome.fa ./data/bowtie2index/mygenome")
```

Execute `SYSargs` on a single machine without submitting to a queuing system of a compute cluster.

```
> bampaths <- runCommandline(args=args)
```

Submit to compute nodes.

```
> qsubargs <- getQsubargs(queue="batch", Nnodes="nodes=1", cores=cores(args), memory="mem=10gb", time="wal")
> (joblist <- qsubRun(appfct="runCommandline(args=args)", appargs=args, qsubargs=qsubargs, Nqsubs=4, submit=TRUE))
```

Alignment Stats

```
> read_statsDF <- alignStats(fqpaths=infile1(args), bampaths=outpaths(args), fqgz=TRUE)
> read_statsDF <- cbind(read_statsDF[targets$FileName,], targets)
> write.table(read_statsDF, "results/alignStats.xls", row.names=FALSE, quote=FALSE, sep="\t")
```

5.3 Create symbolic links for viewing BAM files in IGV

```
> symLink2bam(sysargs=args, htmldir=c("~/html/", "somedir/"),
+             urlbase="http://myserver.edu/~username/",
+             urlfile="IGVurl.txt")
```

5.4 Alignment with Bowtie 2 (here for miRNA profiling experiment)

Run as single process without submitting to cluster, e.g. via qsub -l.

```
> parampath <- paste0(system.file("extdata", package="systemPipeR"), "/bowtieSE.param")
> args <- systemArgs(sysma=parampath, mytargets="targets_run.txt")
> bampaths <- runCommandline(args=args)
```

Submit to compute nodes

```
> qsubargs <- getQsubargs(queue="batch", Nnodes="nodes=1", cores=cores(args), memory="mem=10gb", time="walltime=1000")
> (joblist <- qsubRun(appfct="runCommandline(args=args)", appargs=args, qsubargs=qsubargs, Nqsubs=4, submitargs=qsubargs))
```

5.5 Read counting for mRNA profiling experiments

Create txdb (do only once)

```
> txdb <- makeTranscriptDbFromGFF(file="data/mygenome.gtf", format="gtf", dataSource="ENSEMBL", species="MySpecies")
> saveDb(txdb, file="./data/My_species.sqlite")
```

Read counting with summarizeOverlaps in parallel mode with multiple cores

```
> library(BiocParallel)
> txdb <- loadDb("./data/My_species.sqlite")
> eByg <- exonsBy(txdb, by="gene")
> bams <- names(bampaths); names(bams) <- targets$SampleName
> bfl <- BamFileList(bams, yieldSize=50000, index=character())
> multicoreParam <- MulticoreParam(workers=4); register(multicoreParam); registered()
> counteByg <- bplapply(bfl, function(x) summarizeOverlaps(gff, x, mode="Union", ignore.strand=TRUE, intersect="all"))
> countDFeByg <- sapply(seq(along=counteByg), function(x) assays(counteByg[[x]])$counts)
> rownames(countDFeByg) <- names(rowData(counteByg[[1]])); colnames(countDFeByg) <- names(bfl)
> rpkmDFeByg <- apply(countDFeByg, 2, function(x) returnRPKM(counts=x, gffsub=eByg))
> write.table(assays(countDFeByg)$counts, "results/countDFeByg.xls", col.names=NA, quote=FALSE, sep="\t")
> write.table(rpkmDFeByg, "results/rpkmDFeByg.xls", col.names=NA, quote=FALSE, sep="\t")
```

5.6 Read counting for miRNA profiling experiments

Download miRNA genes from miRBase

```
> system("wget ftp://mirbase.org/pub/mirbase/19/genomes/My_species.gff3 -P ./data/")
> gff <- import.gff("./data/My_species.gff3", asRangedData=FALSE)
> gff <- split(gff, elementMetadata(gff)$ID)
> bams <- names(bampaths); names(bams) <- targets$SampleName
> bfl <- BamFileList(bams, yieldSize=50000, index=character())
> countDFmiR <- summarizeOverlaps(gff, bfl, mode="Union", ignore.strand=FALSE, intersect="all", feature="miR") # Note
> rpkmDFmiR <- apply(countDFmiR, 2, function(x) returnRPKM(counts=x, gffsub=gff))
> write.table(assays(countDFmiR)$counts, "results/countDFmiR.xls", col.names=NA, quote=FALSE, sep="\t")
> write.table(rpkmDFmiR, "results/rpkmDFmiR.xls", col.names=NA, quote=FALSE, sep="\t")
```

5.7 Correlation analysis of samples

```
> library(ape)
> rpkmDFeByg <- read.table("./results/rpkmDFeByg.xls", check.names=FALSE)
> rpkmDFeByg <- rpkmDFeByg[rowMeans(rpkmDFeByg) > 50,]
> d <- cor(rpkmDFeByg, method="spearman")
> hc <- hclust(as.dist(1-d))
> plot.phylo(as.phylo(hc), type="p", edge.col="blue", edge.width=2, show.node.label=TRUE, no.margin=TRUE)
```

5.8 DEG analysis with edgeR

```
> targetspath <- paste0(system.file("extdata", package="systemPipeR"), "/targets.txt")
> targets <- read.delim(targetspath, comment.char = "#")
> (cmp <- readComp(file=targetspath, format="matrix", delim="-"))

$CMPset1
      [,1] [,2]
[1,] "C"  "B"
[2,] "C"  "D"
[3,] "C"  "E"

$CMPset2
      [,1] [,2]
[1,] "B"  "D"
[2,] "D"  "E"

> edgeDF <- run_edgeR(countDF=countDF, targets=targets, cmp=cmp[[1]], independent=TRUE, mdsplot="")
```

6 Version Information

```
> toLatex(sessionInfo())
```

- R version 3.0.2 (2013-09-25), x86_64-unknown-linux-gnu
- Locale: C
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, utils
- Other packages: BiocGenerics 0.8.0, Biostrings 2.30.1, GenomicRanges 1.14.2, IRanges 1.20.1, Rsamtools 1.14.2, ShortRead 1.20.0, XVector 0.2.0, edgeR 3.4.2, lattice 0.20-24, limma 3.18.11, rjson 0.2.13, systemPipeR 1.0.5
- Loaded via a namespace (and not attached): Biobase 2.22.0, BiocStyle 1.0.0, RColorBrewer 1.0-5, bitops 1.0-6, grid 3.0.2, hwriter 1.3, latticeExtra 0.6-26, stats4 3.0.2, tools 3.0.2, zlibbioc 1.8.0

7 Funding

This software was developed with funding from the National Science Foundation: [MCB-1021969](#) .

8 References

- Daehwan Kim, Geo Pertea, Cole Trapnell, Harold Pimentel, Ryan Kelley, and Steven L Salzberg. TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biol.*, 14(4):R36, 25 April 2013. ISSN 1465-6906. doi: 10.1186/gb-2013-14-4-r36. URL <http://dx.doi.org/10.1186/gb-2013-14-4-r36>.
- Ben Langmead and Steven L Salzberg. Fast gapped-read alignment with bowtie 2. *Nat. Methods*, 9(4):357–359, April 2012. ISSN 1548-7091. doi: 10.1038/nmeth.1923. URL <http://dx.doi.org/10.1038/nmeth.1923>.