systemPipeR: utilities for building NGS analysis pipelines

Thomas Girke

Email contact: thomas.girke@ucr.edu

June 21, 2014

1 Introduction

systemPipeR provides utilities for building *end-to-end* analysis pipelines with automated report generation for NGS applications such as RNA-Seq, ChIP-Seq, VAR-Seq and many others. An important feature is support for running command-line software, such as NGS aligners, on both single machines or compute clusters. This includes both interactive job submissions or batch submissions to queuing systems of clusters (tested only with Torque). For instance, *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

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
```

2.3   Sample FASTQ Files

Mini sample FASTQ files used by this overview vignette as well as the associated workflow reporting vignettes can be downloaded from here. This data set contains 18 paired-end (PE) read sets from *Arabidposis thaliana* published by (Howard et al., 2013, SRA: SRP010938). To minimize processing time during testing, each FASTQ file has been subsetted to 90,000-100,000 random sampled PE reads that map to the first 100,000 nucleotides of each chromosome of the *A. thalina* genome. The corresponding reference genome sequence (FASTA) and its GFF annotation files (also provided in the downloadable archive) have been truncated accordingly. This way the entire test sample data set is less than 200MB in storage space. A PE read set has been chosen for this test data set for flexibility, because it can be used for testing both types of analysis routines requiring either SE (single end) reads or PE reads.

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	./data/SRR446027_1.fastq	M1A	M1	Mock.1h.A	1	23-Mar-2012
2	./data/SRR446028_1.fastq	M1B	M1	Mock.1h.B	1	23-Mar-2012
3	./data/SRR446029_1.fastq	A1A	A1	Avr.1h.A	1	23-Mar-2012
4	./data/SRR446030_1.fastq	A1B	A1	Avr.1h.B	1	23-Mar-2012
5	./data/SRR446031_1.fastq	V1A	V1	Vir.1h.A	1	23-Mar-2012
6	./data/SRR446032_1.fastq	V1B	V1	Vir.1h.B	1	23-Mar-2012
7	./data/SRR446033_1.fastq	M6A	M6	Mock.6h.A	1	23-Mar-2012
8	./data/SRR446034_1.fastq	M6B	M6	Mock.6h.B	1	23-Mar-2012
9	./data/SRR446035_1.fastq	A6A	A6	Avr.6h.A	1	23-Mar-2012
10	./data/SRR446036_1.fastq	A6B	A6	Avr.6h.B	1	23-Mar-2012
11	./data/SRR446037_1.fastq	V6A	V6	Vir.6h.A	1	23-Mar-2012
12	./data/SRR446038_1.fastq	V6B	V6	Vir.6h.B	1	23-Mar-2012
13	./data/SRR446039_1.fastq	M12A	M12	Mock.12h.A	1	23-Mar-2012
14	./data/SRR446040_1.fastq	M12B	M12	Mock.12h.B	1	23-Mar-2012
15	./data/SRR446041_1.fastq	A12A	A12	Avr.12h.A	1	23-Mar-2012
16	./data/SRR446042_1.fastq	A12B	A12	Avr.12h.B	1	23-Mar-2012
17	./data/SRR446043_1.fastq	V12A	V12	Vir.12h.A	1	23-Mar-2012
18	./data/SRR446044_1.fastq	V12B	V12	Vir.12h.B	1	23-Mar-2012

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 = "#")[1:2,1:6]

              FileName1                    FileName2 SampleName Factor SampleLong Experiment
1 ./data/SRR446027_1.fastq ./data/SRR446027_2.fastq        M1A     M1  Mock.1h.A          1
2 ./data/SRR446028_1.fastq ./data/SRR446028_2.fastq        M1B     M1  Mock.1h.B          1
```

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] "M1-A1"   "M1-V1"   "A1-V1"   "M6-A6"   "M6-V6"   "A6-V6"   "M12-A12" "M12-V12" "A12-V12"

$CMPset2
 [1] "M1-A1"   "M1-V1"   "M1-M6"   "M1-A6"   "M1-V6"   "M1-M12"  "M1-A12"  "M1-V12"  "A1-V1"
[10] "A1-M6"   "A1-A6"   "A1-V6"   "A1-M12"  "A1-A12"  "A1-V12"  "V1-M6"   "V1-A6"   "V1-V6"
[19] "V1-M12"  "V1-A12"  "V1-V12"  "M6-A6"   "M6-V6"   "M6-M12"  "M6-A12"  "M6-V12"  "A6-V6"
[28] "A6-M12"  "A6-A12"  "A6-V12"  "V6-M12"  "V6-A12"  "V6-V12"  "M12-A12" "M12-V12" "A12-V12"
```

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
6    outfile1          -o                                      <FileName1>
7    outfile1        path                                        ./results/
8    outfile1      remove                                             <NA>
9    outfile1      append                                           .tophat
10   outfile1 outextension          .tophat/accepted_hits.bam
11  reference        <NA>                              ./data/tair10.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 18 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" "results"    "infile1"
 [8] "infile2"   "outfile1" "sysargs"    "outpaths"
> modules(args)
[1] "bowtie2/2.1.0" "tophat/2.0.8b"
> cores(args)
[1] 4
> outpaths(args)[1]
                                                                                                M1A
"/rhome/tgirke/Projects/github/systemPipeR/vignettes/results/SRR446027_1.fastq.tophat/accepted_hits.bam"
> sysargs(args)[1]

"tophat -p 4 -g 1 --segment-length 25 -i 30 -I 3000 -o /rhome/tgirke/Projects/github/systemPipeR/vignettes/
```

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="wall
> (joblist <- qsubRun(args=args, qsubargs=qsubargs, Nqsubs=4, package="systemPipeR"))
```

Alignment Stats

```
> read_statsDF <- alignStats(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 -I.

```
> 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="wal
> (joblist <- qsubRun(args=args, qsubargs=qsubargs, Nqsubs=4, package="systemPipeR"))
```

5.5   Read counting for mRNA profiling experiments

Create txdb (do only once)

```
> txdb <- makeTranscriptDbFromGFF(file="data/mygenome.gtf", format="gtf", dataSource="ENSEMBL", species="My
> 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$SampleName
> bfl <- BamFileList(bams, yieldSize=50000, index=character())
> multicoreParam <- MulticoreParam(workers=4); register(multicoreParam); registered()
> counteByg <- bplapply(bfl, function(x) summarizeOverlaps(gff, x, mode="Union", ignore.strand=TRUE, inter
> countDFeByg <- sapply(seq(along=counteByg), function(x) assays(counteByg[[x]])$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, inter.feature=FALSE) # 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,] "M1"   "A1"
 [2,] "M1"   "V1"
 [3,] "A1"   "V1"
 [4,] "M6"   "A6"
 [5,] "M6"   "V6"
 [6,] "A6"   "V6"
 [7,] "M12"  "A12"
 [8,] "M12"  "V12"
 [9,] "A12"  "V12"

$CMPset2
      [,1]   [,2]
 [1,] "M1"   "A1"
 [2,] "M1"   "V1"
 [3,] "M1"   "M6"
 [4,] "M1"   "A6"
 [5,] "M1"   "V6"
 [6,] "M1"   "M12"
 [7,] "M1"   "A12"
 [8,] "M1"   "V12"
 [9,] "A1"   "V1"
[10,] "A1"   "M6"
[11,] "A1"   "A6"
[12,] "A1"   "V6"
[13,] "A1"   "M12"
[14,] "A1"   "A12"
[15,] "A1"   "V12"
[16,] "V1"   "M6"
[17,] "V1"   "A6"
[18,] "V1"   "V6"
[19,] "V1"   "M12"
[20,] "V1"   "A12"
[21,] "V1"   "V12"
[22,] "M6"   "A6"
[23,] "M6"   "V6"
[24,] "M6"   "M12"
[25,] "M6"   "A12"
[26,] "M6"   "V12"
```

```
[27,] "A6"  "V6"
[28,] "A6"  "M12"
[29,] "A6"  "A12"
[30,] "A6"  "V12"
[31,] "V6"  "M12"
[32,] "V6"  "A12"
[33,] "V6"  "V12"
[34,] "M12" "A12"
[35,] "M12" "V12"
[36,] "A12" "V12"
```

```
> 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 1.0.7
- 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

8   References

Brian E Howard, Qiwen Hu, Ahmet Can Babaoglu, Manan Chandra, Monica Borghi, Xiaoping Tan, Luyan He, Heike Winter-Sederoff, Walter Gassmann, Paola Veronese, and Steffen Heber. High-throughput RNA sequencing of pseudomonas-infected arabidopsis reveals hidden transcriptome complexity and novel splice variants. *PLoS One*, 8 (10):e74183, 1 October 2013. ISSN 1932-6203. doi: 10.1371/journal.pone.0074183. URL `http://dx.doi.org/10.1371/journal.pone.0074183`.

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`.