systemPipeR: NGS workflow and report generation environment

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

June 28, 2014

1 Introduction

systemPipeR provides utilities for building *end-to-end* analysis workflows with automated report generation for next generation sequence (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 `BWA` (**??**), `TopHat 2` (**?**) and `Bowtie 2` (**?**), as well as the R-based NGS aligner *Rsubread* (**?**). Efficient handling of complex sample sets and experimental designs is facilitated by a well-defined sample annotation infrastructure which improves reproducibility and user-friendliness of many typical analysis workflows in the NGS area (**?**).

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

The mini sample FASTQ files used by this overview vignette as well as the associated workflow reporting vignettes can be downloaded from here. The chosen data set SRP010938 contains 18 paired-end (PE) read sets from *Arabidposis thaliana* **?**. 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 annotion files (provided in the same download) 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)
```

Construct SYSargs object from param and targets files.

```
> args <- systemArgs(sysma="tophat.param", mytargets="targetsPE.txt")
```

5.2 FASTQ quality report

The following seeFastq and seeFastqPlot functions generate and plot a series of useful quality statistics for a set of FASTQ files including per cycle quality box plots, base proportions, base-level quality trends, relative k-mer diversity, length and occurrence distribution of reads, number of reads above quality cutoffs and mean quality distribution.

```
> fqlist <- seeFastq(fastq=infile1(args), batchsize=10000, klength=8)
> pdf("./results/fastqReport.pdf", height=18, width=4*length(fqlist))
> seeFastqPlot(fqlist)
> dev.off()
```

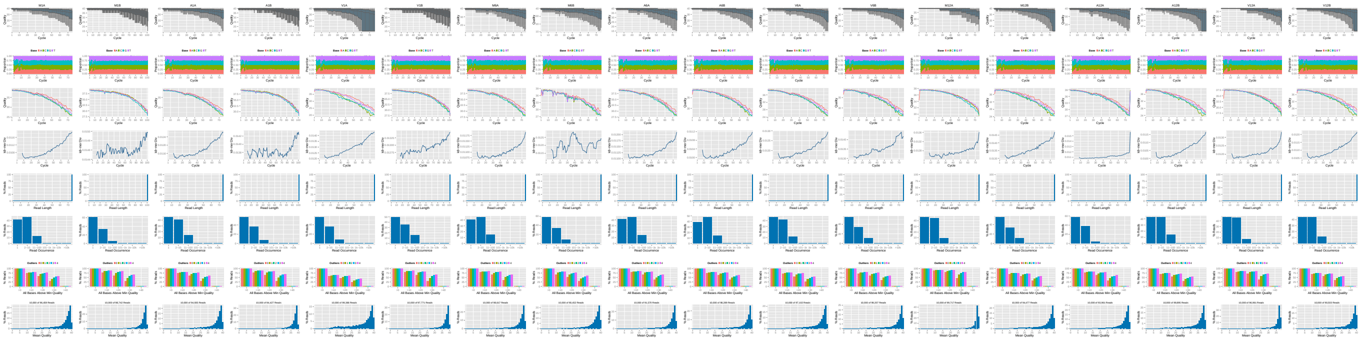

Figure 1: QC report for 18 FASTQ files.

5.3 Alignment with Tophat 2

Build Bowtie 2 index.

```
> moduleload(modules(args)) # Skip if module system is not available
> system("bowtie2-build ./data/tair10.fasta ./data/tair10.fasta")
```

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", cores=cores(args), memory="mem=10gb", time="walltime=20:00:00")
> (joblist <- qsubRun(args=args, qsubargs=qsubargs, Nqsubs=18, package="systemPipeR"))
```

Alignment Stats

```
> read_statsDF <- alignStats(args, fqgz=TRUE)
> write.table(read_statsDF, "results/alignStats.xls", row.names=FALSE, quote=FALSE, sep="\t")
```

5.4 Create symbolic links for viewing BAM files in IGV

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

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

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

```
> args <- systemArgs(sysma="bowtieSE.param", mytargets="targets.txt")
> bampaths <- runCommandline(args=args)
```

Submit to compute nodes

```
> qsubargs <- getQsubargs(queue="batch", cores=cores(args), memory="mem=10gb", time="walltime=20:00:00")
> (joblist <- qsubRun(args=args, qsubargs=qsubargs, Nqsubs=18, package="systemPipeR"))
```

5.6 Read counting for mRNA profiling experiments

Create txdb (do only once)

```
> library(GenomicFeatures)
> txdb <- makeTranscriptDbFromGFF(file="data/tair10.gff", format="gff", dataSource="TAIR", species="A. thal
> saveDb(txdb, file="./data/tair10.sqlite")
```

Read counting with summarizeOverlaps in parallel mode with multiple cores

```
> library(BiocParallel)
> txdb <- loadDb("./data/tair10.sqlite")
> eByg <- exonsBy(txdb, by="gene")
> bfl <- BamFileList(outpaths(args), yieldSize=50000, index=character())
> multicoreParam <- MulticoreParam(workers=4); register(multicoreParam); registered()
> counteByg <- bplapply(bfl, function(x) summarizeOverlaps(eByg, 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, ranges=eByg))
> write.table(countDFeByg, "results/countDFeByg.xls", col.names=NA, quote=FALSE, sep="\t")
> write.table(rpkmDFeByg, "results/rpkmDFeByg.xls", col.names=NA, quote=FALSE, sep="\t")
```

5.7 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")
```

5.8 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)
```

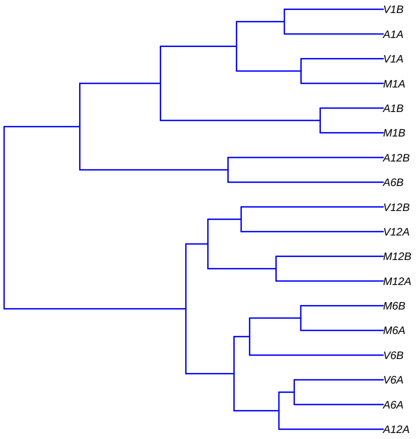

Figure 2: Correlation dendrogram of samples.

5.9 DEG analysis with edgeR

```
> library(edgeR)
> targets <- read.delim(targetspath, comment="#")
> cmp <- readComp(file=targetspath, format="matrix", delim="-")
> cmp[[1]]
```

```
      [,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"
```

Run *edgeR*

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

Filter and plot DEG results for up and down regulated genes. Because of the toy sample set used in this vignette, the FDR value has been set to a relatively high threshold (here 10%). More commonly used FDR cutoffs are 1% or 5%.

```
> DEG_list <- filterDEGs(degDF=edgeDF, filter=c(Fold=2, FDR=10))
```

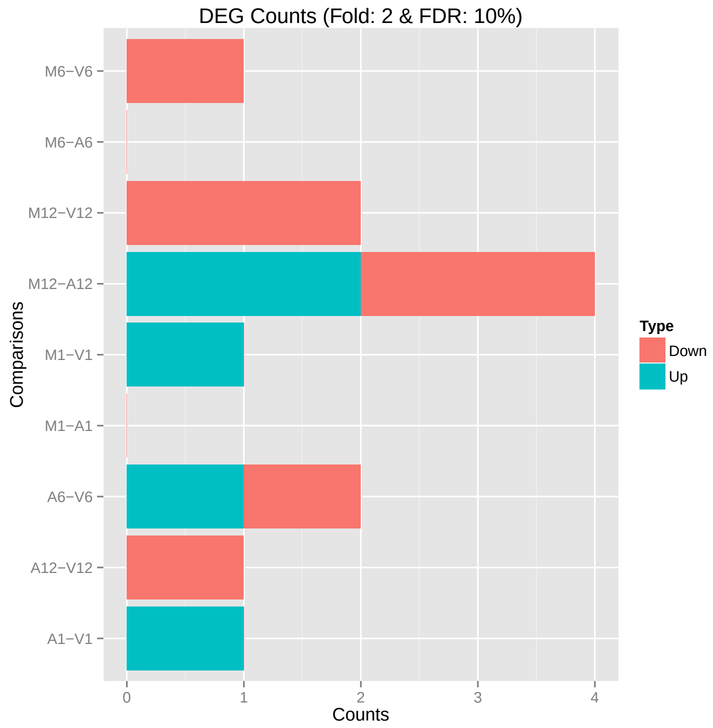

Figure 3: Up and down regulated DEGs.

```
> names(DEG_list)
> DEG_list$Summary
```

5.10 GO term enrichment analysis of DEGs

Instructions for obtaining GO annotations from *biomaRt*. Note there are usually various alternative sources from where those annotations can be obtained, including the `*.db` genome annotation packages from Bioconductor.

```
> library("biomaRt")
> listMarts() # Choose BioMart databases, here vb_mart_22 (VectorBase)
> m <- useMart("ENSEMBL_MART_PLANT"); listDatasets(m) # Choose genome from VectorBase, here aaegypti_eg_ge
> m <- useMart("ENSEMBL_MART_PLANT", dataset="athaliana_eg_gene")
> listAttributes(m) # Choose data types you want to download
> go <- getBM(attributes=c("go_accession", "tair_locus", "go_namespace_1003"), mart=m)
> go <- go[go[,3]!="",]; go[,3] <- as.character(go[,3])
> go[go[,3]=="molecular_function", 3] <- "F"; go[go[,3]=="biological_process", 3] <- "P"; go[go[,3]=="cellu
> go[1:4,]
> dir.create("./data/GO")
> write.table(go, "data/GO/GOannotationsBiomart_mod.txt", quote=FALSE, row.names=FALSE, col.names=FALSE, se
> readGOorg(myfile = "data/GO/GOannotationsBiomart_mod.txt", outdir="data/GO", colno = c(1,2,3))
> gene2GOlist(outdir="data/GO", rootUK=FALSE)
```

Running the enrichment analysis

```
> edgeDF <- read.delim("results/edgeRglm_allcomp.xls", row.names=1, check.names=FALSE)
> DEGlist <- filterDEGs(degDF=edgeDF, filter=c(Fold=2, FDR=5))
> up_down <- DEGlist$UporDown; names(up_down) <- paste(names(up_down), "_up_down", sep="")
> up <- DEGlist$Up; names(up) <- paste(names(up), "_up", sep="")
> down <- DEGlist$Down; names(down) <- paste(names(down), "_down", sep="")
> DEGlist <- c(up_down, up, down)
> DEGlist <- DEGlist[sapply(DEGlist, length) > 0]
> loadData("data/GO")
> BatchResult <- GOCluster_Report(setlist=DEGlist, method="all", id_type="gene", CLSZ=10, cutoff=0.9, goca
> library("biomaRt"); vb <- useMart("vb_mart_22", dataset="aaegypti_eg_gene")
> goslimvec <- as.character(getBM(attributes=c("goslim_goa_accession"), mart=vb)[,1])
> BatchResultslim <- GOCluster_Report(setlist=DEGlist, method="slim", id_type="gene", myslimv=goslimvec, CL
```

Sample plots of GO slim terms.

```
> gos <- gos[grep("mycomp", gos$CLID), ]
> goBarplot(gos, gocat="MF")
> goBarplot(gos, gocat="BP")
> goBarplot(gos, gocat="CC")
```

6 Version Information

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

- R version 3.1.0 (2014-04-10), `x86_64-unknown-linux-gnu`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, utils
- Other packages: AnnotationDbi 1.26.0, BSgenome 1.32.0, Biobase 2.24.0, BiocGenerics 0.10.0, BiocParallel 0.6.1, Biostrings 2.32.0, Category 2.30.0, DBI 0.2-7, GO.db 2.14.0, GOstats 2.30.0, GenomeInfoDb 1.0.2, GenomicAlignments 1.0.1, GenomicRanges 1.16.3, IRanges 1.22.9, Matrix 1.1-4, RSQLite 0.11.4, Rsamtools 1.16.1, ShortRead 1.22.0, XVector 0.4.0, annotate 1.42.0, edgeR 3.6.2, ggplot2 1.0.0, graph 1.42.0, limma 3.20.6, rjson 0.2.14, systemPipeR 1.0.11
- Loaded via a namespace (and not attached): AnnotationForge 1.6.1, BBmisc 1.7, BatchJobs 1.2, BiocStyle 1.2.0, GSEABase 1.26.0, MASS 7.3-33, RBGL 1.40.0, RColorBrewer 1.0-5, Rcpp 0.11.2, XML 3.98-1.1, bitops 1.0-6, brew 1.0-6, checkmate 1.0, codetools 0.2-8, colorspace 1.2-4, digest 0.6.4, fail 1.2, foreach 1.4.2, genefilter 1.46.1, gtable 0.1.2, hwriter 1.3, iterators 1.0.7, lattice 0.20-29, latticeExtra 0.6-26, munsell 0.4.2, plyr 1.8.1, proto 0.3-10, reshape2 1.4, scales 0.2.4, sendmailR 1.1-2, splines 3.1.0, stats4 3.1.0, stringr 0.6.2, survival 2.37-7, tools 3.1.0, xtable 1.7-3, zlibbioc 1.10.0

7 Funding

This software was developed with funding from the National Science Foundation: MCB-1021969 .

8 References

Thomas Girke. systemPipeR: NGS workflow and report generation environment, 28 June 2014. URL https://github.com/tgirke/systemPipeR.

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.

Michael Lawrence, Wolfgang Huber, Hervé Pagès, Patrick Aboyoun, Marc Carlson, Robert Gentleman, Martin T Morgan, and Vincent J Carey. Software for computing and annotating genomic ranges. *PLoS Comput. Biol.*, 9(8):e1003118, 8 August 2013. ISSN 1553-734X. doi: 10.1371/journal.pcbi.1003118. URL http://dx.doi.org/10.1371/journal.pcbi.1003118.

H Li and R Durbin. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14): 1754–1760, July 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp324. URL http://dx.doi.org/10.1093/bioinformatics/btp324.

Heng Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. 03 2013. URL http://arxiv.org/abs/1303.3997.

Yang Liao, Gordon K Smyth, and Wei Shi. The subread aligner: fast, accurate and scalable read mapping by seed-and-vote. *Nucleic Acids Res.*, 41(10):e108, 4 April 2013. ISSN 0305-1048. doi: 10.1093/nar/gkt214. URL http://dx.doi.org/10.1093/nar/gkt214.