systemPipeR's New CWL Command-line Interface

Author: Daniela Cassol (danielac@ucr.edu) and Thomas Girke (thomas.girke@ucr.edu)

Last update: 05 April, 2019

Package

systemPipeR 1.17.7

Contents

Note: if you use `systemPipeR` in published research, please cite: Backman, T.W.H and Girke, T. (2016). *systemPipeR*: NGS Workflow and Report Generation Environment. *BMC Bioinformatics*, 17: 388. 10.1186/s12859-016-1241-0.

1 *systemPipeR*'s New CWL Command-line Interface

Computational workflows are becoming increasingly important for advanced scientific research, mainly because of the amount of data output by next-generation sequencing technologies. Workflows typically are composed of multiple software tools, or pipelines, each with a specific set of parameters, a different configuration of the input data and also the output results. Reproducibility and scalable are the main concern when a research workflow is designed. Although several tools for managing and executing workflow systems are available, they are designed to offer a specific set of functionalities. To address this need, we have developed a new tool (new CWL S4 Class) to create and run workflows, integrating the Common Workflow Language (CWL), which provide a standard for describing analysis workflows in a generic and reproducible manner.

1.1 CWL Introduction

1.2 Structure of `new param` files and `SYSargs2` container

The `.cwl` files defines the parameters of a chosen command-line software. The following shows the format of a sample `hisat2-mapping-se.cwl` file provided by this package.

```
library(systemPipeR)
targets <- system.file("extdata", "targets.txt", package = "systemPipeR")
dir_path <- system.file("extdata/cwl/hisat2-se", package = "systemPipeR")
WF <- loadWorkflow(targets = targets, wf_file = "hisat2-mapping-se.cwl",
    input_file = "hisat2-mapping-se.yml", dir_path = dir_path)

WF <- renderWF(WF, inputvars = c(FileName = "_FASTQ_PATH_", SampleName = "_SampleName_"))
WF
## Instance of 'SYSargs2':
##    Slot names/accessors:
##       targets: 18 (M1A...V12B), targetsheader: 4 (lines)
##       modules: 2
##       wf: 0, clt: 1, yamlinput: 7 (components)
##       input: 18, output: 18
##       cmdlist: 18
##    WF Steps:
##       1. hisat2-mapping-se.cwl (rendered: TRUE)
clt(WF)
## $`hisat2-mapping-se.cwl`
## $`hisat2-mapping-se.cwl`$cwlVersion
## [1] "v1.0"
##
## $`hisat2-mapping-se.cwl`$class
## [1] "CommandLineTool"
```

```
##
## $`hisat2-mapping-se.cwl`$doc
## [1] "[HISAT2](https://ccb.jhu.edu/software/hisat2/index.shtml): graph-based alignment of next generation s
##
## $`hisat2-mapping-se.cwl`$label
## [1] "Last updated 02/2019"
##
## $`hisat2-mapping-se.cwl`$hints
## $`hisat2-mapping-se.cwl`$hints$SoftwareRequirement
## $`hisat2-mapping-se.cwl`$hints$SoftwareRequirement$packages
## $`hisat2-mapping-se.cwl`$hints$SoftwareRequirement$packages[[1]]
## $`hisat2-mapping-se.cwl`$hints$SoftwareRequirement$packages[[1]]$package
## [1] "hisat2"
##
## $`hisat2-mapping-se.cwl`$hints$SoftwareRequirement$packages[[1]]$version
## [1] "2.1.0"
##
##
##
##
##
## $`hisat2-mapping-se.cwl`$baseCommand
## [1] "hisat2"
##
## $`hisat2-mapping-se.cwl`$requirements
## $`hisat2-mapping-se.cwl`$requirements$InitialWorkDirRequirement
## $`hisat2-mapping-se.cwl`$requirements$InitialWorkDirRequirement$listing
## [1] "$(inputs.results_path)"
##
##
##
## $`hisat2-mapping-se.cwl`$arguments
## $`hisat2-mapping-se.cwl`$arguments[[1]]
## $`hisat2-mapping-se.cwl`$arguments[[1]]$prefix
## [1] "-S"
##
## $`hisat2-mapping-se.cwl`$arguments[[1]]$valueFrom
## [1] "$(inputs.results_path.basename)/$(inputs.SampleName).sam"
##
## $`hisat2-mapping-se.cwl`$arguments[[1]]$position
## [1] 5
##
##
## $`hisat2-mapping-se.cwl`$arguments[[2]]
## $`hisat2-mapping-se.cwl`$arguments[[2]]$prefix
## [1] "-x"
##
## $`hisat2-mapping-se.cwl`$arguments[[2]]$valueFrom
## [1] "$(inputs.hisat2_idx_basedir.path)/$(inputs.hisat2_idx_basename)"
##
## $`hisat2-mapping-se.cwl`$arguments[[2]]$position
```

```
## [1] 6
##
##
## $`hisat2-mapping-se.cwl`$arguments[[3]]
## $`hisat2-mapping-se.cwl`$arguments[[3]]$prefix
## [1] "-k"
##
## $`hisat2-mapping-se.cwl`$arguments[[3]]$valueFrom
## [1] "1"
##
## $`hisat2-mapping-se.cwl`$arguments[[3]]$position
## [1] 1
##
##
## $`hisat2-mapping-se.cwl`$arguments[[4]]
## $`hisat2-mapping-se.cwl`$arguments[[4]]$prefix
## [1] "--min-intronlen"
##
## $`hisat2-mapping-se.cwl`$arguments[[4]]$valueFrom
## [1] "30"
##
## $`hisat2-mapping-se.cwl`$arguments[[4]]$position
## [1] 2
##
##
## $`hisat2-mapping-se.cwl`$arguments[[5]]
## $`hisat2-mapping-se.cwl`$arguments[[5]]$prefix
## [1] "--max-intronlen"
##
## $`hisat2-mapping-se.cwl`$arguments[[5]]$valueFrom
## [1] "3000"
##
## $`hisat2-mapping-se.cwl`$arguments[[5]]$position
## [1] 3
##
##
##
## $`hisat2-mapping-se.cwl`$inputs
## $`hisat2-mapping-se.cwl`$inputs$hisat2_idx_basedir
## $`hisat2-mapping-se.cwl`$inputs$hisat2_idx_basedir$label
## [1] "Path to the directory the index for the reference genome"
##
## $`hisat2-mapping-se.cwl`$inputs$hisat2_idx_basedir$type
## [1] "Directory"
##
##
## $`hisat2-mapping-se.cwl`$inputs$hisat2_idx_basename
## $`hisat2-mapping-se.cwl`$inputs$hisat2_idx_basename$label
## [1] "Basename of the hisat2 index files"
##
## $`hisat2-mapping-se.cwl`$inputs$hisat2_idx_basename$type
```

```
## [1] "string"
##
##
## $`hisat2-mapping-se.cwl`$inputs$fq1
## $`hisat2-mapping-se.cwl`$inputs$fq1$label
## [1] "Comma-separated list of files containing unpaired reads to be aligned"
##
## $`hisat2-mapping-se.cwl`$inputs$fq1$type
## [1] "File"
##
## $`hisat2-mapping-se.cwl`$inputs$fq1$inputBinding
## $`hisat2-mapping-se.cwl`$inputs$fq1$inputBinding$prefix
## [1] "-U"
##
## $`hisat2-mapping-se.cwl`$inputs$fq1$inputBinding$itemSeparator
## [1] ","
##
## $`hisat2-mapping-se.cwl`$inputs$fq1$inputBinding$position
## [1] 7
##
##
##
## $`hisat2-mapping-se.cwl`$inputs$thread
## $`hisat2-mapping-se.cwl`$inputs$thread$label
## [1] "Launch NTHREADS parallel search threads"
##
## $`hisat2-mapping-se.cwl`$inputs$thread$type
## [1] "int"
##
## $`hisat2-mapping-se.cwl`$inputs$thread$inputBinding
## $`hisat2-mapping-se.cwl`$inputs$thread$inputBinding$prefix
## [1] "--threads"
##
##
##
## $`hisat2-mapping-se.cwl`$inputs$SampleName
## $`hisat2-mapping-se.cwl`$inputs$SampleName$label
## [1] "Filename to write output to"
##
## $`hisat2-mapping-se.cwl`$inputs$SampleName$type
## [1] "string"
##
##
## $`hisat2-mapping-se.cwl`$inputs$results_path
## $`hisat2-mapping-se.cwl`$inputs$results_path$label
## [1] "Path to the results directory"
##
## $`hisat2-mapping-se.cwl`$inputs$results_path$type
## [1] "Directory"
##
##
```

```
##
## $`hisat2-mapping-se.cwl`$outputs
## $`hisat2-mapping-se.cwl`$outputs$hisat2_sam
## $`hisat2-mapping-se.cwl`$outputs$hisat2_sam$type
## [1] "File"
##
## $`hisat2-mapping-se.cwl`$outputs$hisat2_sam$outputBinding
## $`hisat2-mapping-se.cwl`$outputs$hisat2_sam$outputBinding$glob
## [1] "$(inputs.results_path.basename)/$(inputs.SampleName).sam"
```

SYSargs2 class stores all the information and instructions needed for processing a set of input files with a specific command-line or a series of command-line within a workflow. The *SYSargs2* S4 class object is created from the *loadWorkflow* and *renderWF* function, which populates all the command-line for each sample in each step of the particular workflow. Each sample level input/outfile operation uses its own *SYSargs2* instance. The output of *SYSargs2* define all the expected output files for each step in the workflow, which usually it is the sample input for the next step in an *SYSargs2* instance. Between different instances, this connectivity is established by writing the subsetting output with the *writeTargetsout* function to a new targets file that serves as input to the next *loadWorkflow* and *renderWF* call. By chaining several *SYSargs2* steps together one can construct complex workflows involving many sample-level input/output file operations with any combination of command-line or R-based software.

Several accessor methods are available that are named after the slot names of the `SYSargs2` object.

```
names(WF)
##  [1] "targets"       "targetsheader" "modules"
##  [4] "wf"            "clt"           "yamlinput"
##  [7] "cmdlist"       "input"         "output"
## [10] "cwlfiles"      "inputvars"
```

Of particular interest is the `cmdlist()` method. It constructs the system commands for running command-lined software as specified by a given `.cwl` file combined with the paths to the input samples (*e.g.* FASTQ files) provided by a `targets` file. The example below shows the `cmdlist()` output for running HISAT2 on the first SE read sample. Evaluating the output of `cmdlist()` can be very helpful for designing and debugging `.cwl` files of new command-line software or changing the parameter settings of existing ones.

```
cmdlist(WF)[1]
## $M1A
## $M1A$`hisat2-mapping-se.cwl`
## [1] "hisat2 -S results/M1A.sam  -x ./data/tair10.fasta  -k 1  --min-intronlen 30  --max-intronlen 3000  -U
modules(WF)
##        module1        module2
## "hisat2/2.0.1" "samtools/1.9"
targets(WF)[1]
## $M1A
## $M1A$FileName
## [1] "./data/SRR446027_1.fastq.gz"
##
## $M1A$SampleName
```

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

```
## 18          1 23-Mar-2012
output(WF)
## $M1A
## $M1A$`hisat2-mapping-se.cwl`
## [1] "results/M1A.sam"
##
##
## $M1B
## $M1B$`hisat2-mapping-se.cwl`
## [1] "results/M1B.sam"
##
##
## $A1A
## $A1A$`hisat2-mapping-se.cwl`
## [1] "results/A1A.sam"
##
##
## $A1B
## $A1B$`hisat2-mapping-se.cwl`
## [1] "results/A1B.sam"
##
##
## $V1A
## $V1A$`hisat2-mapping-se.cwl`
## [1] "results/V1A.sam"
##
##
## $V1B
## $V1B$`hisat2-mapping-se.cwl`
## [1] "results/V1B.sam"
##
##
## $M6A
## $M6A$`hisat2-mapping-se.cwl`
## [1] "results/M6A.sam"
##
##
## $M6B
## $M6B$`hisat2-mapping-se.cwl`
## [1] "results/M6B.sam"
##
##
## $A6A
## $A6A$`hisat2-mapping-se.cwl`
## [1] "results/A6A.sam"
##
##
## $A6B
## $A6B$`hisat2-mapping-se.cwl`
## [1] "results/A6B.sam"
##
```

```
##
## $V6A
## $V6A$`hisat2-mapping-se.cwl`
## [1] "results/V6A.sam"
##
##
## $V6B
## $V6B$`hisat2-mapping-se.cwl`
## [1] "results/V6B.sam"
##
##
## $M12A
## $M12A$`hisat2-mapping-se.cwl`
## [1] "results/M12A.sam"
##
##
## $M12B
## $M12B$`hisat2-mapping-se.cwl`
## [1] "results/M12B.sam"
##
##
## $A12A
## $A12A$`hisat2-mapping-se.cwl`
## [1] "results/A12A.sam"
##
##
## $A12B
## $A12B$`hisat2-mapping-se.cwl`
## [1] "results/A12B.sam"
##
##
## $V12A
## $V12A$`hisat2-mapping-se.cwl`
## [1] "results/V12A.sam"
##
##
## $V12B
## $V12B$`hisat2-mapping-se.cwl`
## [1] "results/V12B.sam"
cwlfiles(WF)
## $cwl
## [1] "/home/dcassol/R/x86_64-pc-linux-gnu-library/3.7/systemPipeR/extdata/cwl/hisat2-se/hisat2-mapping-se.c
##
## $yml
## [1] "/home/dcassol/R/x86_64-pc-linux-gnu-library/3.7/systemPipeR/extdata/cwl/hisat2-se/hisat2-mapping-se.y
inputvars(WF)
## $FileName
## [1] "_FASTQ_PATH_"
##
## $SampleName
## [1] "_SampleName_"
```

```
infile1(WF)[1:4]
##
## "/home/dcassol/danielac@ucr.edu/github/systemPipeR_Workflows/systemPipeR_overview/version-cwl/data/SRR446
##
## "/home/dcassol/danielac@ucr.edu/github/systemPipeR_Workflows/systemPipeR_overview/version-cwl/data/SRR446
##
## "/home/dcassol/danielac@ucr.edu/github/systemPipeR_Workflows/systemPipeR_overview/version-cwl/data/SRR446
##
## "/home/dcassol/danielac@ucr.edu/github/systemPipeR_Workflows/systemPipeR_overview/version-cwl/data/SRR446
```

2 Showcase Workflow with `HISAT2`

2.1 Read mapping with `HISAT2`

The NGS reads of this project will be aligned against the reference genome sequence using `Hisat2` (Kim, Langmead, and Salzberg 2015). The parameter settings of the aligner are defined in the `workflow_hisat2-se.cwl` and `workflow_hisat2-se.yml` files.

```
###################################### Example: hisat2 WF with samtools ##
targets <- system.file("extdata", "targets.txt", package = "systemPipeR")
dir_path <- system.file("extdata/cwl/workflow-hisat2-se", package = "systemPipeR")
WF <- loadWorkflow(targets = targets, wf_file = "workflow_hisat2-se.cwl",
    input_file = "workflow_hisat2-se.yml", dir_path = dir_path)
WF <- renderWF(WF, inputvars = c(FileName = "_FASTQ_PATH_", SampleName = "_SampleName_"))
WF

## Paired-End HISAT2 only
targetsPE <- system.file("extdata", "targets.txt", package = "systemPipeR")
dir_path <- system.file("extdata/cwl/hisat2-pe", package = "systemPipeR")
WF <- loadWorkflow(targets = targetsPE, wf_file = "hisat2-mapping-pe.cwl",
    input_file = "hisat2-mapping-pe.yml", dir_path = dir_path)
WF <- renderWF(WF, inputvars = c(FileName1 = "_FASTQ_PATH1_",
    FileName2 = "_FASTQ_PATH2_", SampleName = "_SampleName_"))
WF

## Paired-End HISAT2 WF
dir_path <- system.file("extdata/cwl/workflow-hisat2-pe", package = "systemPipeR")
WF <- loadWorkflow(targets = targetsPE, wf_file = "workflow_hisat2-pe.cwl",
    input_file = "workflow_hisat2-pe.yml", dir_path = dir_path)
WF <- renderWF(WF, inputvars = c(FileName1 = "_FASTQ_PATH1_",
    FileName2 = "_FASTQ_PATH2_", SampleName = "_SampleName_"))
WF
```

Subsetting *SYSargs2* class slots for each workflow step.

```
## Testing subset_wf function
subsetWF(WF, slot = "input", subset = "FileName")
subsetWF(WF, slot = "output", subset = 2)
```

```
subsetWF(WF, slot = "step", subset = 1)  ## subset all the HISAT2 commandline
subsetWF(WF, slot = "output", subset = "samtools-index.cwl")
subsetWF(WF, slot = "output", subset = 1, delete = TRUE)  ##DELETE
```

Execute *SYSargs2* on a single machine without submitting to a queuing system of a compute cluster. This way the input FASTQ files will be processed sequentially.

```
## runCommandLine
library(systemPipeR)
runCommandline(WF)  ## creates the files in the ./results folder
runCommandline(WF, dir = TRUE)  ## creates the files in the ./results/workflowName/Samplename folder
runCommandline(WF, dir = TRUE, make_bam = TRUE)  ##if it uses the workflow with samtools, should not uses mak
```

Check and update the output location if necessary.

```
WF <- output_update(WF, dir = TRUE)  ## Updates the output(WF) to the right location in the subfolders
WF <- output_update(WF, dir = TRUE, replace = ".bam")  ## Updates the output(WF) to the right location in the
output(WF)
## Add to runCommandline
```

Check whether all BAM files have been created.

```
WF_track <- run_track(WF_ls = c(WF))
names(WF_track)
WF_steps(WF_track)
track(WF_track)
summaryWF(WF_track)
```

Parallelization of read/alignment stats via scheduler (*e.g.* Slurm) across several compute nodes.

```
## clusterRun
library(batchtools)
resources <- list(walltime = 120, ntasks = 1, ncpus = 4, memory = 1024)
reg <- clusterRun2(WF, FUN = runCommandline2, conffile = ".batchtools.conf.R",
    template = "batchtools.slurm.tmpl", Njobs = 4, runid = "01",
    resourceList = resources)
getStatus(reg = reg)

WF <- output_update(WF, dir = TRUE)  ## Updates the output(WF) to the right location in the subfolders
output(WF)
```

2.1.1 Read and alignment stats

The following provides an overview of the number of reads in each sample and how many of them aligned to the reference.

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

2.1.1.1 Write `new targets files`

To establish the connectivity between different instances, it is possible by writing the subsetting output with the `writeTargetsout` function to a new targets file that serves as input to the next `loadWorkflow` and `renderWF` call.

```
names(WF$clt)
writeTargetsout(x = WF, file = "default", step = 1)
```

2.2 Building a custom workflow with CWL

TODO: Describe the rules and standards for `systemPipeR` and how to create a new template using the `create.clt`.

2.3 Showcase small RNA-Seq workflow

3 Version Information

```
sessionInfo()
## R Under development (unstable) (2019-04-03 r76310)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.2 LTS
##
## Matrix products: default
## BLAS:   /usr/local/lib/R/lib/libRblas.so
## LAPACK: /usr/local/lib/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices
## [6] utils     datasets  methods   base
##
## other attached packages:
##  [1] batchtools_0.9.11           data.table_1.12.0
##  [3] ape_5.3                     ggplot2_3.1.0
##  [5] systemPipeR_1.17.7          ShortRead_1.41.0
##  [7] GenomicAlignments_1.19.1    SummarizedExperiment_1.13.0
##  [9] DelayedArray_0.9.9          matrixStats_0.54.0
## [11] Biobase_2.43.1              BiocParallel_1.17.18
## [13] Rsamtools_1.99.5            Biostrings_2.51.5
```

```
## [15] XVector_0.23.2              GenomicRanges_1.35.1
## [17] GenomeInfoDb_1.19.3         IRanges_2.17.4
## [19] S4Vectors_0.21.22           BiocGenerics_0.29.2
## [21] BiocStyle_2.11.0
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-137              Category_2.49.1
##  [3] bitops_1.0-6              bit64_0.9-7
##  [5] RColorBrewer_1.1-2        progress_1.2.0
##  [7] httr_1.4.0                Rgraphviz_2.27.0
##  [9] tools_3.7.0               backports_1.1.3
## [11] R6_2.4.0                  DBI_1.0.0
## [13] lazyeval_0.2.2            colorspace_1.4-1
## [15] withr_2.1.2               prettyunits_1.0.2
## [17] bit_1.1-14                compiler_3.7.0
## [19] graph_1.61.1              formatR_1.6
## [21] rtracklayer_1.43.3        bookdown_0.9
## [23] scales_1.0.0              checkmate_1.9.1
## [25] genefilter_1.65.0         RBGL_1.59.5
## [27] rappdirs_0.3.1            stringr_1.4.0
## [29] digest_0.6.18             rmarkdown_1.12
## [31] AnnotationForge_1.25.0    pkgconfig_2.0.2
## [33] htmltools_0.3.6           BSgenome_1.51.0
## [35] limma_3.39.14             rlang_0.3.3
## [37] RSQLite_2.1.1             GOstats_2.49.0
## [39] hwriter_1.3.2             VariantAnnotation_1.29.24
## [41] RCurl_1.95-4.12           magrittr_1.5
## [43] GO.db_3.7.0               GenomeInfoDbData_1.2.1
## [45] Matrix_1.2-17             Rcpp_1.0.1
## [47] munsell_0.5.0             stringi_1.4.3
## [49] yaml_2.2.0                edgeR_3.25.3
## [51] zlibbioc_1.29.0           plyr_1.8.4
## [53] grid_3.7.0                blob_1.1.1
## [55] crayon_1.3.4              lattice_0.20-38
## [57] splines_3.7.0             GenomicFeatures_1.35.9
## [59] annotate_1.61.1           hms_0.4.2
## [61] locfit_1.5-9.1            knitr_1.22
## [63] pillar_1.3.1              rjson_0.2.20
## [65] base64url_1.4             codetools_0.2-16
## [67] biomaRt_2.39.2            XML_3.98-1.19
## [69] evaluate_0.13             latticeExtra_0.6-28
## [71] BiocManager_1.30.4        gtable_0.3.0
## [73] assertthat_0.2.1          xfun_0.6
## [75] xtable_1.8-3              survival_2.44-1.1
## [77] tibble_2.1.1              pheatmap_1.0.12
## [79] AnnotationDbi_1.45.1      memoise_1.1.0
## [81] brew_1.0-6                GSEABase_1.45.0
```

4 Funding

This project is funded by NSF award ABI-1661152.

References

Kim, Daehwan, Ben Langmead, and Steven L Salzberg. 2015. "HISAT: A Fast Spliced Aligner with Low Memory Requirements." *Nat. Methods* 12 (4): 357–60.