VAR-Seq project report template: Some Descriptive Title

Project ID: VARseq_PI_Name_Organism_Mar2015
Project PI: First Last (first.last@inst.edu)
Author of Report: First Last (first.last@inst.edu)

April 4, 2015

Contents

1 Introduction

This report describes the analysis of an VAR-Seq project from Dr. First Last's lab which studies the gene expression changes of ... in *organism* .... The experimental design is as follows...

2 Sample definitions and environment settings

2.1 Environment settings and input data

Typically, the user wants to record here the sources and versions of the reference genome sequence along with the corresponding annotations. In the provided sample data set all data inputs are stored in a `data` subdirectory and all results will be written to a separate `results` directory, while the `systemPipeVARseq.Rnw` script and the `targets` file are expected be located in the parent directory. The R session is expected to run from this parent directory.

To run this sample report, mini sample FASTQ and reference genome files can be downloaded from here. The chosen data set SRP010938 contains 18 paired-end (PE) read sets from *Arabidposis thaliana* Howard et al. (2013). To minimize processing time during testing, each FASTQ file has been subsetted to 90,000-100,000 randomly 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.

2.2 Required packages and resources

The *systemPipeR* package needs to be loaded to perform the analysis steps shown in this report (Girke, 2014).

```
> library(systemPipeR)
```

If applicable load custom functions not provided by *systemPipeR*

```
> source("systemPipeVARseq_Fct.R")
```

2.3 Experiment definition provided by `targets` file

The `targets` file defines all FASTQ files and sample comparisons of the analysis workflow.

```
> targetspath <- system.file("extdata", "targetsPE.txt", package="systemPipeR")
> targets <- read.delim(targetspath, comment.char = "#")[,1:5]
> targets
```

```
                FileName1                FileName2 SampleName Factor SampleLong
1  ./data/SRR446027_1.fastq ./data/SRR446027_2.fastq        M1A     M1  Mock.1h.A
2  ./data/SRR446028_1.fastq ./data/SRR446028_2.fastq        M1B     M1  Mock.1h.B
3  ./data/SRR446029_1.fastq ./data/SRR446029_2.fastq        A1A     A1   Avr.1h.A
4  ./data/SRR446030_1.fastq ./data/SRR446030_2.fastq        A1B     A1   Avr.1h.B
5  ./data/SRR446031_1.fastq ./data/SRR446031_2.fastq        V1A     V1   Vir.1h.A
6  ./data/SRR446032_1.fastq ./data/SRR446032_2.fastq        V1B     V1   Vir.1h.B
```

```
7  ./data/SRR446033_1.fastq ./data/SRR446033_2.fastq        M6A     M6  Mock.6h.A
8  ./data/SRR446034_1.fastq ./data/SRR446034_2.fastq        M6B     M6  Mock.6h.B
9  ./data/SRR446035_1.fastq ./data/SRR446035_2.fastq        A6A     A6   Avr.6h.A
10 ./data/SRR446036_1.fastq ./data/SRR446036_2.fastq        A6B     A6   Avr.6h.B
11 ./data/SRR446037_1.fastq ./data/SRR446037_2.fastq        V6A     V6   Vir.6h.A
12 ./data/SRR446038_1.fastq ./data/SRR446038_2.fastq        V6B     V6   Vir.6h.B
13 ./data/SRR446039_1.fastq ./data/SRR446039_2.fastq       M12A    M12 Mock.12h.A
14 ./data/SRR446040_1.fastq ./data/SRR446040_2.fastq       M12B    M12 Mock.12h.B
15 ./data/SRR446041_1.fastq ./data/SRR446041_2.fastq       A12A    A12  Avr.12h.A
16 ./data/SRR446042_1.fastq ./data/SRR446042_2.fastq       A12B    A12  Avr.12h.B
17 ./data/SRR446043_1.fastq ./data/SRR446043_2.fastq       V12A    V12  Vir.12h.A
18 ./data/SRR446044_1.fastq ./data/SRR446044_2.fastq       V12B    V12  Vir.12h.B
```

3 Read preprocessing

3.1 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. The results are written to a PDF file named `fastqReport.pdf`.

```
> args <- systemArgs(sysma="bwa.param", mytargets="targets.txt")
> fqlist <- seeFastq(fastq=infile1(args), batchsize=100000, klength=8)
> pdf("./results/fastqReport.pdf", height=18, width=4*length(fqlist))
> seeFastqPlot(fqlist)
> dev.off()
```

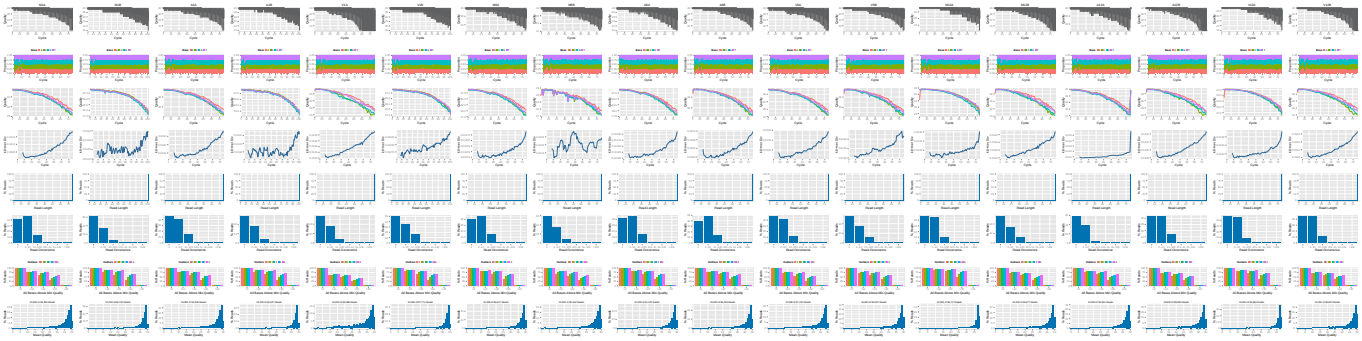

Figure 1: QC report for 18 FASTQ files.

4 Alignments

4.1 Read mapping with `BWA`

The NGS reads of this project will be aligned against the reference genome sequence using `BWA` (Li, 2013; Li and Durbin, 2009). The parameter settings of the aligner are defined in the `bwa.param` file.

```
> args <- systemArgs(sysma="bwa.param", mytargets="targets.txt")
> sysargs(args)[1] # Command-line parameters for first FASTQ file
```

Run alignments sequentially (*e.g.* on a single machine)

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

Alternatively, submit alignment jobs to compute cluster, here using 72 CPU cores (18 qsub processes each with 4 CPU cores).

```
> moduleload(modules(args))
> system("bwa index -a bwtsw ./data/tair10.fasta")
> resources <- list(walltime="20:00:00", nodes=paste0("1:ppn=", cores(args)), memory="10gb")
> reg <- clusterRun(args, conffile=".BatchJobs.R", template="torque.tmpl", Njobs=18, runid="01",
+                   resourceList=resources)
```

Check whether all BAM files have been created

```
> file.exists(outpaths(args))
```

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

```
> read.delim("results/alignStats.xls")
```

4.3 Create symbolic links for viewing BAM files in IGV

The `symLink2bam` function creates symbolic links to view the BAM alignment files in a genome browser such as IGV. The corresponding URLs are written to a file with a path specified under `urlfile`, here IGVurl.txt.

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

5 Variant calling

The following performs variant calling with `GATK` and `BCFtools` in parallel mode on a compute cluster (McKenna et al., 2010; Li, 2011). Alternatively, `runCommandline(args=args)` will run them sequentially on a single machine.

5.1 Variant calling with GATK

```
> writeTargetsout(x=args, file="targets_bam.txt")
> system("java -jar CreateSequenceDictionary.jar R=./data/tair10.fasta O=./data/tair10.dict")
> # system("java -jar /opt/picard/1.81/CreateSequenceDictionary.jar R=./data/tair10.fasta O=./data/tair10.
> args <- systemArgs(sysma="gatk.param", mytargets="targets_bam.txt")
> resources <- list(walltime="20:00:00", nodes=paste0("1:ppn=", 1), memory="10gb")
> reg <- clusterRun(args, conffile=".BatchJobs.R", template="torque.tmpl", Njobs=18, runid="01",
+                   resourceList=resources)
> #unlink(outfile1(args), recursive = TRUE, force = TRUE)
```

5.2 Variant calling with BCFtools

```
> args <- systemArgs(sysma="sambcf.param", mytargets="targets_bam.txt")
> resources <- list(walltime="20:00:00", nodes=paste0("1:ppn=", 1), memory="10gb")
> reg <- clusterRun(args, conffile=".BatchJobs.R", template="torque.tmpl", Njobs=18, runid="01",
+                   resourceList=resources)
> #unlink(outfile1(args), recursive = TRUE, force = TRUE)
```

6 Filtering variants

6.1 Filtering variants from GATK

```
> args <- systemArgs(sysma="filter_gatk.param", mytargets="targets_gatk.txt")
> filter <- "totalDepth(vr) >= 2 & (altDepth(vr) / totalDepth(vr) >= 0.8) & rowSums(softFilterMatrix(vr))==
> # filter <- "totalDepth(vr) >= 20 & (altDepth(vr) / totalDepth(vr) >= 0.8) & rowSums(softFilterMatrix(vr
> filterVars(args, filter, varcaller="gatk", organism="A. thaliana")
> writeTargetsout(x=args, file="targets_gatk_filtered.txt")
```

6.2 Filtering variants from BCFtools

```
> args <- systemArgs(sysma="filter_sambcf.param", mytargets="targets_sambcf.txt")
> filter <- "rowSums(vr) >= 2 & (rowSums(vr[,3:4])/rowSums(vr[,1:4]) >= 0.8)"
> # filter <- "rowSums(vr) >= 20 & (rowSums(vr[,3:4])/rowSums(vr[,1:4]) >= 0.8)"
> filterVars(args, filter, varcaller="bcftools", organism="A. thaliana")
> writeTargetsout(x=args, file="targets_sambcf_filtered.txt")
```

7 Annotate filtered variants

7.1 Annotate filtered variants from GATK

```
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_gatk_filtered.txt")
> txdb <- loadDb("./data/tair10.sqlite")
> fa <- FaFile(systemPipeR::reference(args))
> variantReport(args=args, txdb=txdb, fa=fa, organism="A. thaliana")
```

7.2 Annotate filtered variants from BCFtools

```
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_sambcf_filtered.txt")
> txdb <- loadDb("./data/tair10.sqlite")
> fa <- FaFile(systemPipeR::reference(args))
> variantReport(args=args, txdb=txdb, fa=fa, organism="A. thaliana")
```

8 Combine annotation results among samples

8.1 Combine results from GATK

```
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_gatk_filtered.txt")
> combineDF <- combineVarReports(args, filtercol=c(Consequence="nonsynonymous"))
> write.table(combineDF, "./results/combineDF_nonsyn_gatk.xls", quote=FALSE, row.names=FALSE, sep="\t")
```

8.2 Combine results from BCFtools

```
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_sambcf_filtered.txt")
> combineDF <- combineVarReports(args, filtercol=c(Consequence="nonsynonymous"))
> write.table(combineDF, "./results/combineDF_nonsyn_sambcf.xls", quote=FALSE, row.names=FALSE, sep="\t")
```

9 Summary statistics of variants

9.1 Summary for GATK

```
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_gatk_filtered.txt")
> write.table(varSummary(args), "./results/variantStats_gatk.xls", quote=FALSE, col.names = NA, sep="\t")
```

9.2 Summary for BCFtools

```
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_sambcf_filtered.txt")
> write.table(varSummary(args), "./results/variantStats_sambcf.xls", quote=FALSE, col.names = NA, sep="\t",
```

10 Venn diagram of variants

```
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_gatk_filtered.txt")
> varlist <- sapply(names(outpaths(args))[1:4], function(x) as.character(read.delim(outpaths(args)[x])$VARI
> vennset_gatk <- overLapper(varlist, type="vennsets")
> args <- systemArgs(sysma="annotate_vars.param", mytargets="targets_sambcf_filtered.txt")
> varlist <- sapply(names(outpaths(args))[1:4], function(x) as.character(read.delim(outpaths(args)[x])$VARI
> vennset_bcf <- overLapper(varlist, type="vennsets")
> pdf("./results/vennplot_var.pdf")
> vennPlot(list(vennset_gatk, vennset_bcf), mymain="", mysub="GATK: red; BCFtools: blue", colmode=2, ccol=c
> dev.off()
```

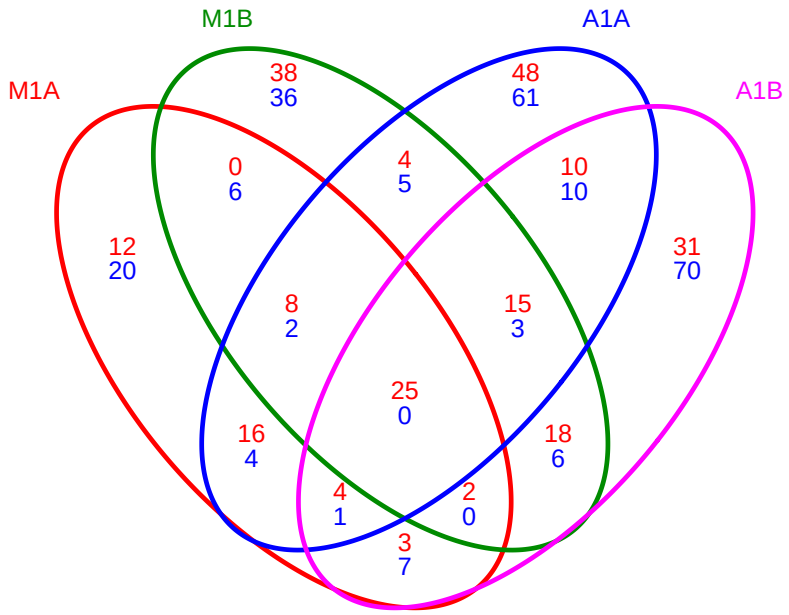

Figure 2: Venn Diagram for 4 samples from GATK and BCFtools.

11 Version Information

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

- R version 3.1.2 (2014-10-31), `x86_64-unknown-linux-gnu`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.28.2, Biobase 2.26.0, BiocGenerics 0.12.1, BiocParallel 1.0.2, Biostrings 2.34.1, DBI 0.3.1, GenomeInfoDb 1.2.4, GenomicAlignments 1.2.1, GenomicRanges 1.18.4, IRanges 2.0.1, RSQLite 1.0.0, Rsamtools 1.18.2, S4Vectors 0.4.0, ShortRead 1.24.0, XVector 0.6.0, systemPipeR 1.0.12
- Loaded via a namespace (and not attached): AnnotationForge 1.8.2, BBmisc 1.9, BatchJobs 1.5, BiocStyle 1.4.1, Category 2.32.0, GO.db 3.0.0, GOstats 2.32.0, GSEABase 1.28.0, MASS 7.3-37, Matrix 1.1-5, RBGL 1.42.0, RColorBrewer 1.1-2, Rcpp 0.11.4, XML 3.98-1.1, annotate 1.44.0, base64enc 0.1-2, bitops 1.0-6, brew 1.0-6, checkmate 1.5.1, codetools 0.2-11, colorspace 1.2-6, digest 0.6.8, edgeR 3.8.5, fail 1.2, foreach 1.4.2, genefilter 1.48.1, ggplot2 1.0.0, graph 1.44.1, grid 3.1.2, gtable 0.1.2, hwriter 1.3.2, iterators 1.0.7, lattice 0.20-30, latticeExtra 0.6-26, limma 3.22.4, munsell 0.4.2, pheatmap 1.0.2, plyr 1.8.1, proto 0.3-10,

reshape2 1.4.1, rjson 0.2.15, scales 0.2.4, sendmailR 1.2-1, splines 3.1.2, stringr 0.6.2, survival 2.38-1, tools 3.1.2, xtable 1.7-4, zlibbioc 1.12.0

12 Funding

This project was supported by funds from the National Institutes of Health (NIH).

13 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.

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. A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data. *Bioinformatics*, 27(21):2987–2993, 1 November 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr509. URL http://bioinformatics.oxfordjournals.org/content/27/21/2987.abstract.

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

Aaron McKenna, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella, David Altshuler, Stacey Gabriel, Mark Daly, and Mark A DePristo. The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, 20(9):1297–1303, 19 July 2010. ISSN 1088-9051. doi: 10.1101/gr.107524.110. URL http://dx.doi.org/10.1101/gr.107524.110.