Bioinformatics workflow for the detection of eQTL in the cattle genome using Nextflow DSL2

Praveen Krishna Chitneedi
Leibniz Institute for Farm Animal Biology
46 PUBLICATIONS **66** CITATIONS

Bioinformatics workflow for the detection of eQTL in the cattle genome using Nextflow DSL2

P.K. Chitneedi[1]*, F. Hadlich[1] and C. Kuehn[1,2]

[1] Research Institute for Farm Animal Biology (FBN), Wilhelm-Stahl-Allee 2, 18196, Dummerstorf, Germany; [2] Agricultural and Environmental Faculty, University Rostock, Justus-von-Liebig-Weg 6, 18059 Rostock, Germany; * chitneedi@fbn-dummerstorf.de

Abstract

The *in silico* detection of expression quantitative trait loci (eQTL) demands high throughput processing from hundreds of samples, which is often a challenge to handle and run such large datasets. In order to focus on the core analysis, it is convenient to have simple coding and hassle-free installation of different software tools required for the bioinformatics workflow. In this context, the newly available technologies like workflow managers and software containers enabled to develop workflows with less complexity. In this study, we developed an eQTL bioinformatics pipeline with the workflow manager Nextflow and docker container software, for coding and installing the required software tools. This workflow can be portable to a different computer environment, and the results are reproducible. We tested the functionality of our workflow with a sample dataset and the runtime estimates from this demo run will provide important information in planning future analyses with much larger datasets.

Introduction

In recent years, large, cost-effectively sequenced transcriptome (RNA-Seq) and whole genome (WGS) data sets enabled association studies with gene expression as phenotype. These eQTL association studies identify the transcriptomic regulation at a genome wide scale by mapping the expression phenotype to a region on the genome (GILAD *et al.* 2008; MAJEWSKI AND PASTINEN 2011). The eQTL phenotype can be gene, transcript, exon expression or alternative splicing events. After high throughput sequencing, the main challenge in eQTL detection is to handle and process the large amount of generated data. When the data size is small (few gigabytes), it is convenient to implement the bioinformatics workflow with a simple computational script. However, things get complicated when dealing with large data especially in studies like eQTL, where generally thousands of RNA-Seq and WGS samples were analysed. In addition, it is not always possible to install multiple bioinformatics tools with different system requirements on a server maintained by an organization with centralized information technology support. Thus, to run such complex bioinformatics pipelines it is convenient to have a simple syntax declaration for calling the input data, executing the process and standard guidelines for installing all the required software tools. In this context, the concept of workflow managers and containers comes into play. Compared to traditional computational scripts used for bioinformatics workflows, the scripts developed using workflow managers are portable and reproducible on multiple server platforms. The most popular workflow managers for implementing bioinformatics pipelines are Snakemake (MOLDER *et al.* 2021) and Nextflow (DI TOMMASO *et al.* 2017) among others. They use domain specific languages (DSLs), which have simple commands to run complex workflows, and the installation of all required software tools was standardised by using an open source package and environment systems like Conda (ANONYMOUS 2020) or by container software with complete runtime environment like docker (MERKEL 2014) or singularity (KURTZER *et al.* 2017).

Materials & Methods

In this current study, we used Nextflow_v21.04.1.5556 DSL2 for developing the workflow for eQTL detection in the bovine genome. Each step in the workflow was defined as an independent entity called ‘Process’ and they communicate with other process via input and output components called ‘Channel’. In the Nextflow DSL2, different processes can be written on separate files called ‘Modules’ independent of the main workflow script, which offers more flexibility to write complex workflows. All the required software to run the workflow were installed using the container software docker_v20.10.8, build 3967b7d. The docker environment was enabled for Nextflow by declaring in a configuration file ‘nextflow.config’. In order to make the workflow easy to understand and troubleshoot potential errors, we divided the eQTL workflow into four NextFlow scripts as shown in Figure 1. Each script was further divided into individual modules, which perform a specific task within the workflow. In NextFlow DSL2, these modules were defined as individual entities, and they were invoked in the main script.

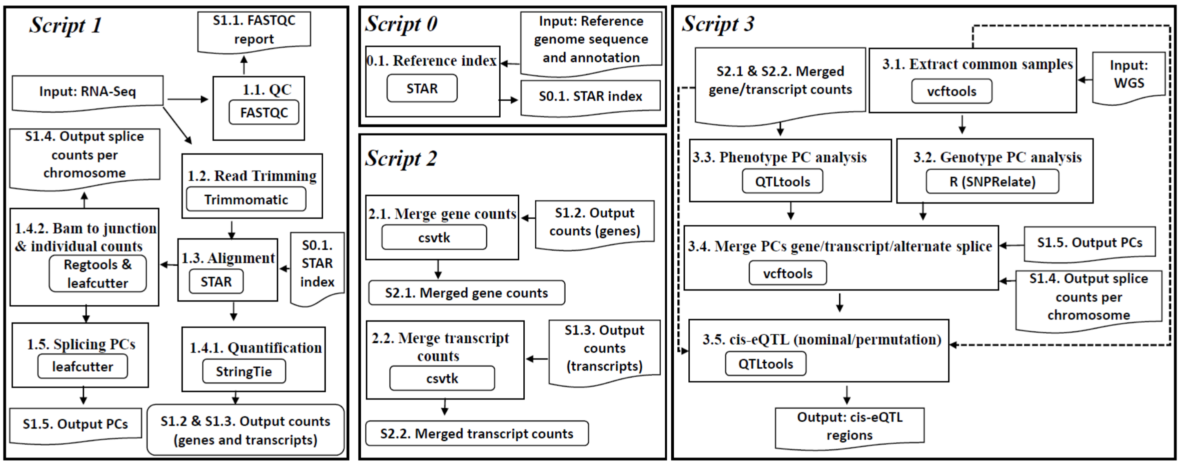
Script 1
S1.1. FASTQC report
Input: RNA-Seq
1.1. QC
FASTQC
S1.4. Output splice counts per chromosome
1.2. Read Trimming
Trimmomatic
1.4.2. Bam to junction & individual counts
Regtools & leafcutter
1.3. Alignment
STAR
S0.1. STAR index
1.5. Splicing PCs
leafcutter
1.4.1. Quantification
StringTie
S1.5. Output PCs
S1.2 & S1.3. Output counts (genes and transcripts)
Script 0
Input: Reference genome sequence and annotation
0.1. Reference index
STAR
S0.1. STAR index
Script 2
2.1. Merge gene counts
csvtk
S1.2. Output counts (genes)
S2.1. Merged gene counts
2.2. Merge transcript counts
csvtk
S1.3. Output counts (transcripts)
S2.2. Merged transcript counts
Script 3
S2.1 & S2.2. Merged gene/transcript counts
3.1. Extract common samples
vcftools
Input: WGS
3.3. Phenotype PC analysis
QTLtools
3.2. Genotype PC analysis
R (SNPRelate)
3.4. Merge PCs gene/transcript/alternate splice
vcftools
S1.5. Output PCs
S1.4. Output splice counts per chromosome
3.5. cis-eQTL (nominal/permutation)
QTLtools
Output: cis-eQTL regions

Figure 1. Diagrammatic representation of Bioinformatics workflow for eQTL detection designed using Nextflow DSL2. Each block represents individual scripts with different modules, input and output data flow.

Script 0: Bovine genome STAR index.

As a pre requisite to perform alignment using STAR (DOBIN *et al.* 2013), it is required to perform the indexing of a reference genome, which was used repeatedly for aligning multiple samples. We built the STAR index for the latest bovine genome ARS-UCD1.2 and Ensembl annotation v102 using the STAR command “--runMode genomeGenerate”.

Script 1: RNA-Seq mapping and quantification of expression counts

Quality control and trimming of reads from samples with paired-end reads was carried out with FASTQC (http://www.bioinformaticsbabraham.ac.uk/projects/fastqc/) and trimmomatic v_0.36 (BOLGER *et al.* 2014). The cleaned paired reads of each sample were forwarded to alignment using STAR_v2.7.0d. For the gene and transcript level counts, the reads were sorted by coordinates and for detecting splice junctions the reads were unsorted and we used “--outSAMstrandField intronMotif ” to determine the read origin either from intron or from splice junction. Using the StringTie_v1.3.3 (PERTEA *et al.* 2015) assembler, we carried out the quantification and normalization at gene and transcript level, and where appropriate the strandedness of the RNA-Seq library was addressed with the StringTie options --rf for the first and --fr for the second stranded libraries. For detecting splice junctions the unsorted reads were indexed and sorted using Samtools_v1.9-4 and then exon-exon junctions were extracted from these bam files using regtools_v0.6.0 (COTTO *et al.* 2021) ‘junctions extract’, which was followed by clustering of introns found in the junction files to generate an across sample expression count matrix per each chromosome. Using leafcutter_v0.2.9 (LI *et al.* 2018) we estimated the top ten principal components (PCs). The count matrices and PCs were used in the model to detect splice eQTL in script 3.

Script 2: Merging count matrices at gene level and transcript level.

After StringTie quantification, the output read count files are stored as .tsv (gene level) and .gtf (transcript level) output files. The TPM normalized expression counts of each sample at gene and transcript level were merged into matrix tables using csvtk --merge (https://bioinf.shenwei.me/csvtk/usage/). These tables were provided as input files to script 3 for the detection of eQTL.

Script 3: Estimating genotype and phenotype covariates and eQTL detection.

The eQTL mapping was carried out with QTLtools_v1.3 (DELANEAU *et al.* 2017). It expects three input files for mapping, genotype data in VCF format, phenotype data in .bed format and the principal components (PCs) as covariates. The input genotype data was filtered for the samples common to the phenotype input. Additionally, we filtered out the SNPs with minor allele frequency less 0.05. The population stratification present due to systematic ancestry differences in the data set was addressed by performing a PC analysis with genotype input using the R package SNPRelate (ZHENG *et al.* 2012). Similarly, to address the outlier samples and batch effects in the RNA-Seq data at gene and transcript level, a phenotype PC analysis was performed with the QTLtools 'pca' command. For the eQTL mapping, the covariates include the top ten genotype PCs and top ten phenotype PCs. Finally, the nominal cis-eQTL mapping was carried out by QTLtools with the parameter --nominal 0.01, which provides as output only phenotype-variant pairs with a nominal P-value below 0.01, and the parameter --normal enforced the phenotypes to match a normal distribution N(0,1). In order to obtain the adjusted P-value for the cis associated phenotype and top variants, we performed permutation based QTL mapping also using QTLtools with the parameters --normal and --permute 1000.

Results & Discussion

We developed this eQTL workflow with the primary goal of eQTL detection in the BovReg (https://www.bovreg.eu/) project with RNA-Seq phenotype and WGS data. To check the functionality of this eQTL workflow, we tested the workflow with 88 bovine liver RNA-Seq and the corresponding WGS data. The analysis was ran on Intel(R) Xeon(R) @ 2.10GHz server with 144 CPUs, 18 cores and 1TB RAM memory. Tests with a simple dataset will give an idea of how much runtime is required and potential pitfalls to consider, when running this workflow with much larger sample size. For the current test run, the WGS include 19,590,389 bi-allelic, polymorphic (MAF > 0.005) variants across 29 autosomal chromosome, which were imputed using a stepwise strategy from 6k to 50k to HD to WGS (PAUSCH *et al.* 2016) with a mean imputation accuracy of 0.97 across different steps. The RNA-Seq samples consist of paired end libraries sequenced on the Illumina HiSeq 2500 system (NOLTE *et al.* 2019; HEIMES *et al.* 2020). The average number of input reads across 88 RNA samples were 56 million read pairs varying between 43 to 74 million read pairs. Although for this demo we used only paired-end reads, this pipeline will also work for single-read RNA-samples. The eQTL mapping will be carried out per chromosome by taking the WGS data for each chromosome for each iteration. By providing the configuration file and the scripts, this workflow can be portable, the analysis can reproducible on any Nextflow and docker installed computer. When executing the scripts the Nextflow installs all the required tools and users just have to take care of declaring the correct paths of input data. The Nextflow stores all log files and output data in a directory named as 'work'. This workflow was designed to copy all the important results from each step in a separate directory path. To run a workflow with large samples requires a rough estimation of runtime on a given computer infrastructure, as it takes few days to months to finish the analyses for large sample sizes. Thus, with the current demo of our eQTL workflow, we also checked the average runtime for 88 RNA-samples in the different processes on our local server. The alignment is the most time consuming process and took 108 hours for aligning all the 88 samples, while the runtime of the other processes in Script 1 varied from 1 to 10 hours. The runtime for Script 2 was just 2 minutes 41seconds, this was expected as this script only merges the available count information of different samples into a table. The final Script 3 with the PC analysis, QTL mapping and permutation across different levels (gene, transcript and splicing events) completed in 2 hours 28minutes. This runtime estimates provided us with valuable information to plan future analyses with higher sample size. Although this workflow was analysed using bovine datasets it can used for detecting eQTL in other species. In the future, the iteration of this workflow also includes allele specific expression and detection of trans-eQTL across different bovine tissues.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 815668.

References

Anonymous (2020) Available online at https://docs.anaconda.com/

Bolger A.M., Lohse M. and Usadel B. (2014) Bioinformatics 30(15)**:** 2114-2120. https://doi.org/10.1093/bioinformatics/btu170

Cotto K.C., Feng Y.-Y., Ramu A., Skidmore Z.L., Kunisaki J. *et al.* (2021) BioRxiv https://doi.org/10.1101/436634 (

Delaneau O., Ongen H., Brown A.A., Fort A., Panousis N.I. *et al.* (2017) Nat Commun 8:15452. https://doi.org/10.1038/ncomms15452

Di Tommaso P., Chatzou M., Floden E.W., Barja P.P., Palumbo E. *et al.* (2017) Nat Biotechnol 35(4)**:** 316-319. https://doi.org/10.1038/nbt.3820

Dobin A., Davis C.A., Schlesinger F., Drenkow J., Zaleski C. *et al.* (2013) Bioinformatics 29(1)**:** 15-21. https://doi.org/10.1093/bioinformatics/bts635

Gilad Y., Rifkin S.A. and Pritchard J.K. (2008) Trends Genet 24(8)**:** 408-415. https://doi.org/10.1016/j.tig.2008.06.001

Heimes A., Brodhagen J., Weikard R., Seyfert H.M., Becker D. *et al.* (2020) Front Immunol 11:715. https://doi.org/10.3389/fimmu.2020.00715

Kurtzer G.M., Sochat V. and Bauer M.W. (2017) Plos One 12(5). https://doi.org/10.1371/journal.pone.0177459

Li Y.I., Knowles D.A., Humphrey J., Barbeira A.N., Dickinson S.P. *et al.* (2018) Nature Genetics 50(1)**:** 151-158. https://doi.org/10.1038/s41588-017-0004-9

Majewski J., and Pastinen T. (2011) Trends Genet 27(2)**:** 72-79. https://doi.org/10.1016/j.tig.2010.10.006

Merkel D. (2014) Linux journal 2014(239)**:** 2.

Molder F., Jablonski K.P., Letcher B., Hall M.B., Tomkins-Tinch C.H. *et al.* (2021) F1000Res 10:33. https://doi.org/10.12688/f1000research.29032.2

Nolte W., Weikard R., Brunner R.M., Albrecht E., Hammon H.M. *et al.* (2019) Front Genet 10:1130. https://doi.org/10.3389/fgene.2019.01130

Pausch H., Emmerling R., Schwarzenbacher H. and Fries R. (2016) Genet Sel Evol 48:14. https://doi.org/10.1186/s12711-016-0190-4

Pertea M., Pertea G.M., Antonescu C.M., Chang T.C., Mendell J.T. *et al.* (2015) Nature Biotechnology 33(3)**:** 290-295. https://doi.org/10.1038/nbt.3122

Zheng X., Levine D., Shen J., Gogarten S.M., Laurie C. *et al.* (2012) Bioinformatics 28(24)**:** 3326-3328. https://doi.org/10.1093/bioinformatics/bts606