Helmholtz Zentrum München

KNIME4NGS: a comprehensive toolbox for high-throughput Next Generation Sequencing analysis

Maximilian Hastreiter, Tim Jeske, Jonathan Hoser, Michael Kluge, Kaarin Ahomaa,
Marie-Sophie Friedl, Sebastian J. Kopetzky, Jan-Dominik Quell,
H.-Werner Mewes and Robert Küffner

September 7, 2016

Contents

1 Introduction

1.1 Prerequisites

- Linux System (required)
- KNIME or KNIME SDK (required)
- KNIME Cluster Extension[1] (recommended)
- KNIME Virtual Nodes Extension (for parallel execution only, see section 3))
- R installation with packages: gplots, ggplot2, reshape, gsalib; for RNA-Seq pipeline: argparse, edgeR, DESeq, Biobase and limma
- Genome Analysis Toolkit (GATK) 3.x [2] (tested with GATK 3.5 and 3.6)
- Variant Effect Predictor (VEP) script including the Ensembl Core and Variation APIs (for variant annotation, see section 2.6)[3]. VEP itself requires the following perl modules: Archive::Extract, Archive::Zip, Test::More, DBI and CGI

1.2 Installation

1. Start KNIME
2. Specify the KNIME4NGS Update Site as a software repository within KNIME. In order to do that, go to Help –> Install New Software... and add
   `http://ibisngs.github.io/knime4ngs/updateSite`
   to the available software sites as shown in Figure 1.1.
3. Select the new software site. The IBIS KNIME Nodes should now appear. Select and install the displayed package. After restarting KNIME, the KNIME4NGS nodes should appear in the KNIME Node Repository as shown in Figure 1.2.
4. (Optional) Configure the KNIME4NGS Preference Page which can be found at:
   File –> Preferences –> KNIME –> KNIME4NGS.
   - Missing binaries can be obtained by using 'Download missing binaries' and/or search for existing binaries in your file system. This step is necessary since most nodes do not contain the respective execution binaries. Although binaries can be selected in each node independently, setting global binary paths in the preference page is recommended. See section 5 for more details.
   - Activate HTE execution. See section 4 for more details.
5. (Optional) Go to the KNIME GUI Preference Page and set Console View Log Level to DEBUG to obtain more logging information.

[1] https://www.knime.org/cluster-execution
[2] https://www.broadinstitute.org/gatk/
[3] http://www.ensembl.org/info/docs/tools/vep/script/vep_download.html

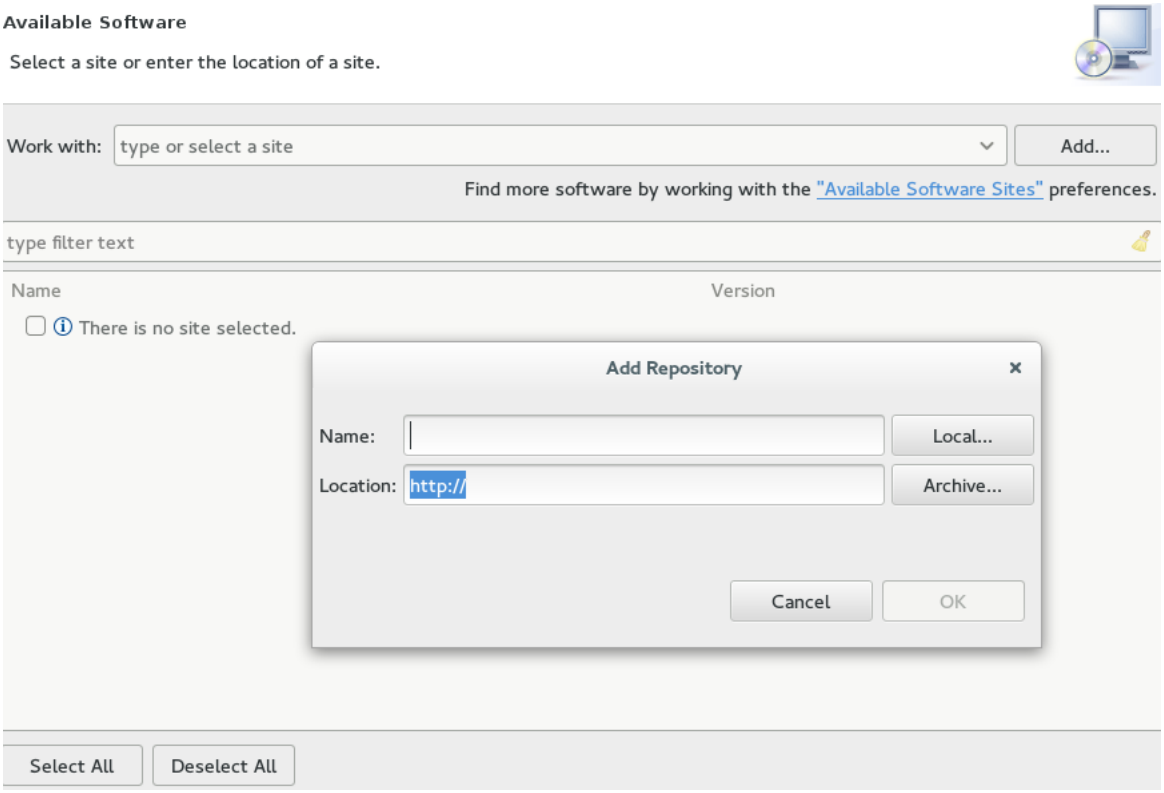

Figure 1.1: Add KNIME4NGS Update Site as a repository site.

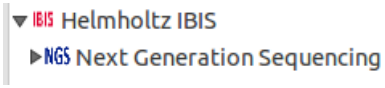

Figure 1.2: KNIME4NGS nodes appear in the KNIME Node Repository.

2 Basic Usage

2.1 File Loading and Pipeline Entry Point

As an entry point to your pipeline, you can and should use the **FileLoader** node for selecting your input files. Depending on your input, this node can be used to enter pipelines at any stage. Possible inputs include FastA, FastQ, SAM/BAM or VCF/gVCF files. After node execution, the selected file path(s) will be displayed in the nodes output table.

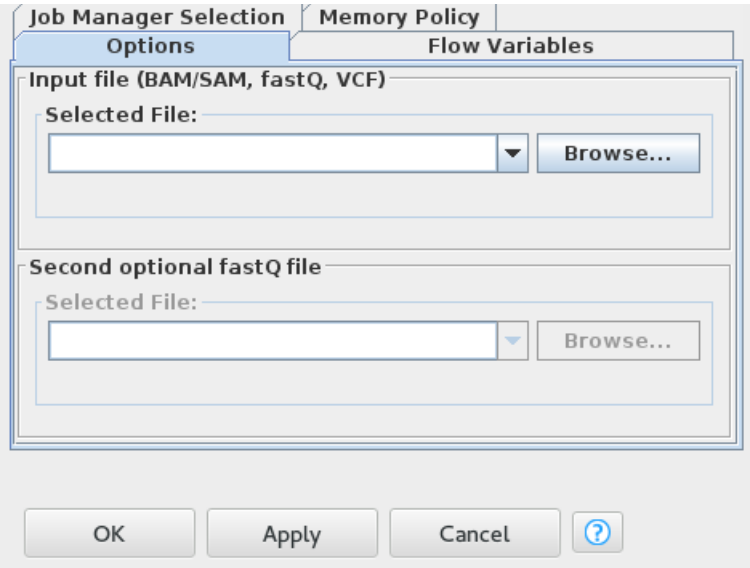

Figure 2.1: Loading files using the FileLoader node.

2.2 Read Filtering and Quality Control

For basic quality control and read filtering, we have included a modified version of **FastQC**, which is designed to work with our proposed read-filtering tool, the **RawReadManipulator**. While the FastQC node will prepare summary statistics about your input sequences, it also provides suggested filtering criteria depending on the input quality. These filter settings are then used by the **RawReadManipulator** to filter the data. Alternatively, you can select your own settings in the node dialog.

Figure 2.2: The RawReadManipulator can use proposed filter settings created by FastQC.

A suggested workflow is depicted in Figure 2.3. Running **FastQC** for a second time after read filtering allows you to check the quality of the filtered FastQ files.

2.3 Mapping and SAM-BAM Conversion

For mapping the reads against a reference genome, you can choose between several mapping nodes like **BWA**, **Segemehl** or **Star**, depending on your type of analysis. If you have already used the previous nodes for QC, you can directly connect them to your preferred mapping node, as shown in Figure 2.4. Otherwise use the **FileLoader** to load your file(s).
SAM files can be converted to BAM format using the **FastSam2Bam** node, an implementation

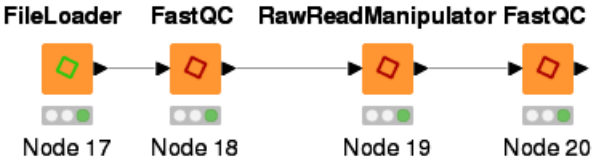

Figure 2.3: Quality Check and Filtering Workflow

based on Samtools and Picard, allowing fast conversion times by using multiple processing threads.

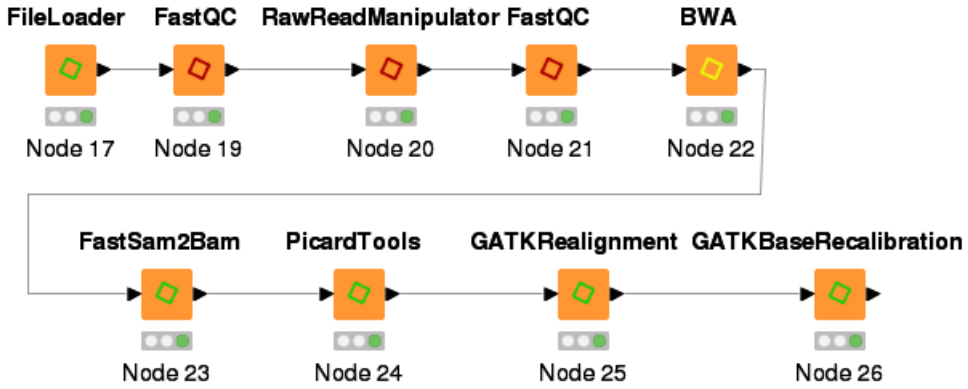

Figure 2.4: Mapping, Conversion and Alignment Refinement

2.4 Alignment Refinement

In order to improve alignment quality and therefore the quality of all following analysis steps, there are several ways to increase alignment reliability. We have included several nodes, following the GATK Base Practices, for PCR Duplicate marking (**PicardTools**), Indel Realignment (**GATKRealignment**) and Base Quality Score Recalibration (**GATKBaseRecalibration**) as shown in Figure 2.4. Note, that the latter two GATK nodes require read groups. If there are no read groups in your BAM files you can add some by using the *AddOrReplaceReadGroups* tool of the **PicardTools** node.

2.5 Variant Calling and Filtering

Variant Calling can be done by different ways, depending on the field of interest. For SNP and short Indel calling, we provide the **UnifiedGenotyper** and **HaplotypeCaller** (gvcf mode only). The resulting GVCF files of the **HaplotypeCaller** can be converted to VCF files by using **GenotypeGVCFs**. Furthermore, we included **Pindel**, an algorithm for calling long Indels, structural variants and more. (Currently, the node only supports single-sample calling) **Pindel** requires a config file which can easily be generated by the *CollectInsertMetrics* tool of **PicardTools**. All mentioned variant calling nodes are shown in Figure 2.5.

To exclude low quality variants that arise from sequencing artifacts we suggest to apply Variant Quality Score Recalibration (**VQSR**) based on GATK's VariantRecalibration and ApplyRecalibration tool. Additionally, the **VCFToolsFilter** enables filtering for further sequencing quality parameters.

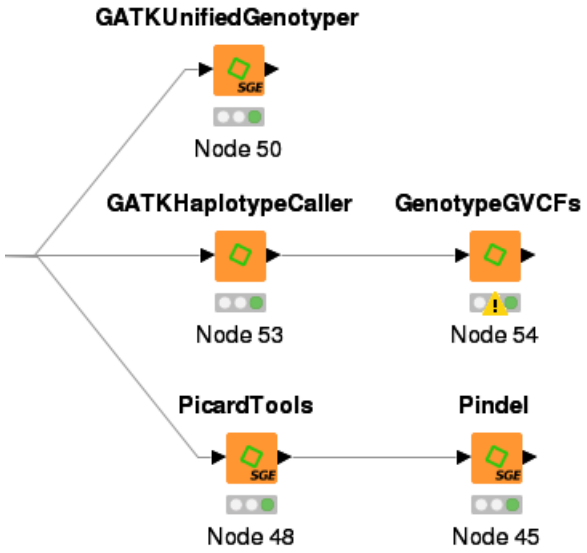

Figure 2.5: For Variant Calling, three different nodes are provided, each based on different calling algorithms.

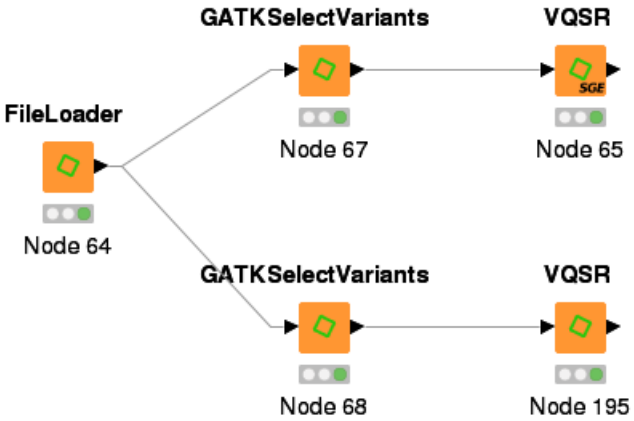

Figure 2.6: Exemplary workflow showing the use of the Variant Quality Score Recalibration.

2.6 Variant Annotation

Our toolbox provides the **VEP** node to annotate the functional impact of called variants. The node wraps Ensembl's VEP script and allows you to set the most important flags directly via the node dialog. Nevertheless, it is possible to specify further flags or plugins in an extra text field. After applying **VEP**, the variants can be filtered according to the annotations by applying the **VEPFilter**.

2.7 Differential Expression

For differential expression analysis, we provide a pipeline based on the Subread package **featureCounts**. It can be used for generating raw read counts that are the input for typical differential expression analysis like **DESeq**, **Limma** or **EdgeR**. An exemplary workflow is shown in Figure 2.7.
Using the parallel chunk environment, multiple BAM files are processed at the same time.

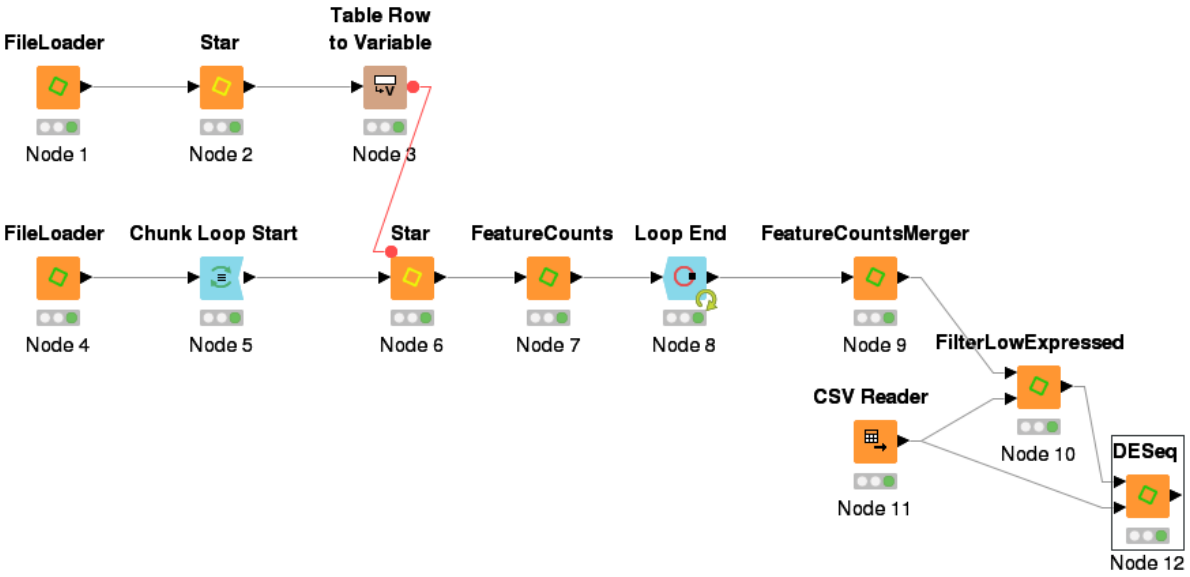

Figure 2.7: Differential Expression Workflow using Star and DESeq

3 Parallel Execution

Analysing a large number of samples typically requires parallel processing to ensure reasonable runtimes. In KNIME, this can be achieved by, for example, using the **Parallel Chunk** nodes (**KNIME Virtual Nodes Extension**). As shown in Figure 3.1, these nodes can be combined with the KNIME4NGS nodes to create a parallel execution pipeline for large scale NGS analysis.

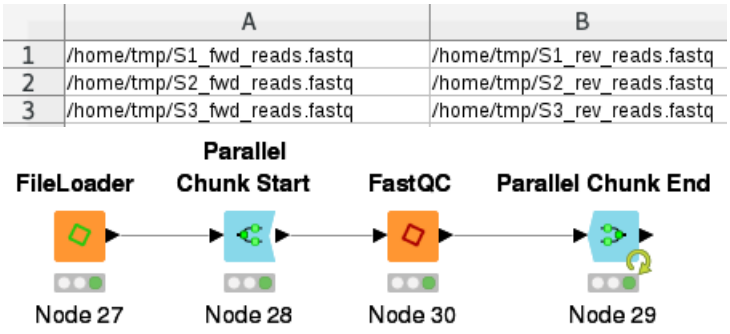

Figure 3.1: Exemplary parallel execution workflow

To run a pipeline in parallel mode, first prepare a file containing all input filepaths (e.g. Col0: forward read files, Col1: reverse read files, one row for each sample). In this case you have to check the box: "Handle input file as list of file paths?". By using the **Parallel Chunk Start**, each row will be handled separately if the chunk count equals the number of rows.

4 The HTE

Several nodes of our toolbox can be run in the High Throughput Execution (HTE) mode which has three main benefits. Firstly, nodes are automatically restarted if they fail for some reason. Secondly, the occurring errors are reported in a database and you are notified by Email if your workflow fails. Thirdly, HTE remembers the successful execution of a node which prevents the time-consuming re-execution.

In some cases, tools used by the KNIME4NGS nodes fail because of errors that could be solved by a restart the tool. HTE is capable of restarting nodes automatically as often as you wish. You can globally set a threshold for the number of restarts for all nodes in the preference page (see section 5) or set it individually for each node in the configuration dialog.

4.1 HTE Database

If you use the HTE mode you can create a HTE database which logs errors occurring during the execution of your workflows. The database is stored locally as a file and requires no database server. The HTE database consists of two tables: the HTExecution table comprises entries for each executed node and the HTError_History table contains the error messages if the execution failed at least once. If you want to use a HTE database in your KNIME installation, go to the KNIME4NGS preference page and create a new database file (hte.db) or select an existing one.

You can easily access the database using two KNIME nodes that are included in the basic repository. An exemplary workflow is shown in Figure 4.1. Simply add the path to your HTE database (the hte.db file) in the configuration dialog of the SQLite Connector node. The subsequent Database Reader allows you to inspect the fields of both tables via "Fetch Metadata" and to execute SQL queries.

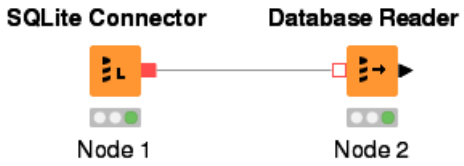

Figure 4.1: Read the HTE database

4.2 Klock Files

Sometimes, it may happen that you reset a node or a complete workflow unintentionally. In general, nodes are completely rexecuted after resetting. This behavior can cause you to wait for a long time to get the same results although you have not changed the configuration of a node. For that reason, most of our nodes generate klock files which save the termination state of time-consuming nodes. These files prevent the re-execution of successfully executed nodes if their configuration is not changed. If you want to re-execute a node without altering its configuration you can simply delete the corresponding klock file which is located in the same directory as the output file.

5 Preference Page

The toolbox comes along with its own preference page shown in Figure 5.1. This page allows you to set several options which simplifies and accelerates the configuration of your workflows. All nodes will also work without the preference page settings but they are much more comfortable to use if you take the time to configure the preference page.

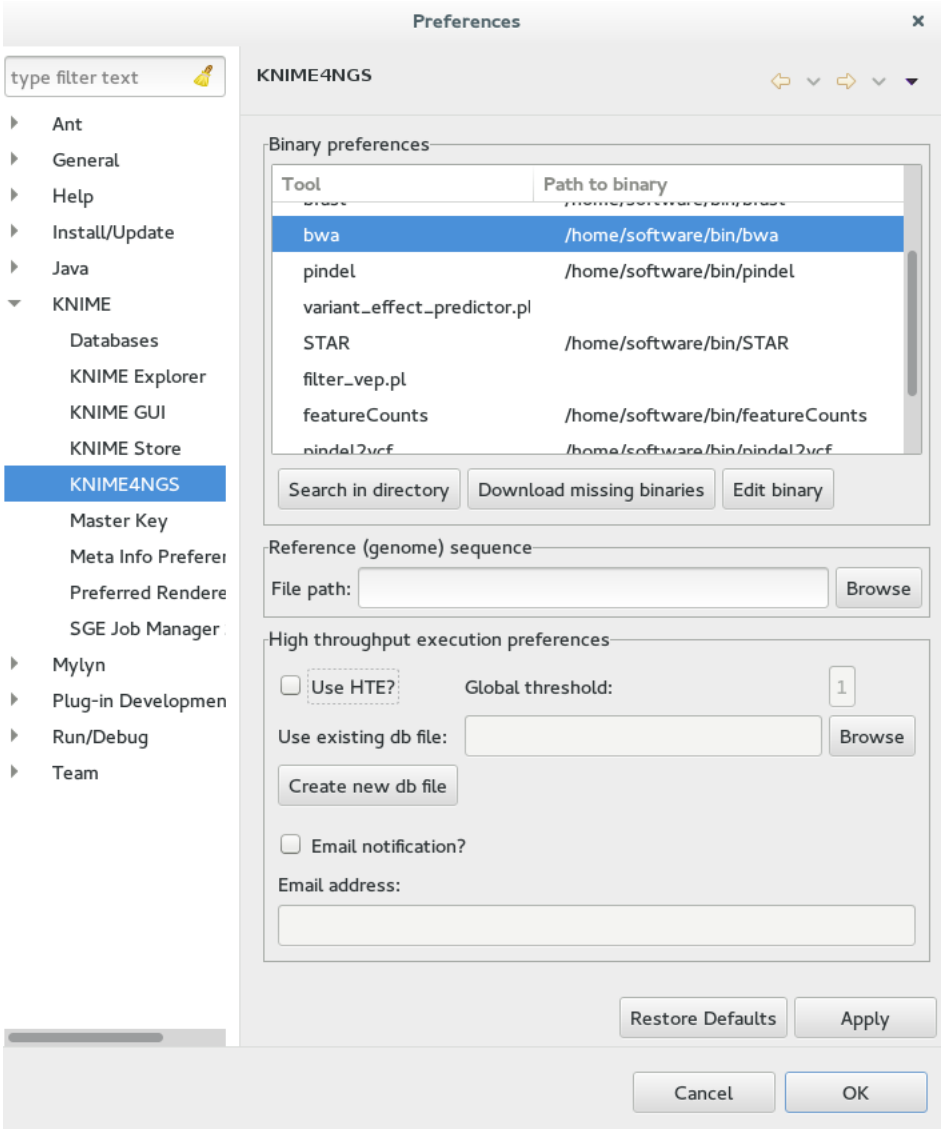

Figure 5.1: KNIME4NGS Preference Page. It can be found in "File > Preferences > KNIME > KNIME4NGS"

The first part of the preference page offers you three possibilities to set paths to binaries required by the nodes of the toolkit. First, you can specify a directory where missing binaries will be searched. Please consider that also subdirectories are included and thus the searching process may take a while depending on the selected starting directory. Second, it is also possible to download missing binaries in your directory of choice. Executable binaries will be saved into the selected folder and automatically integrated into the preference page without the need for further configuration. Note, that not all binaries can be download due to license issues. If required, you can set each binary path manually.

In the middle part of the page you can set the reference genome. As all GATK and some other nodes require a reference genome, setting it only once will save you a lot of time.

The last part of the page allows you to set HTE preferences (see also section 4). If you decide to use HTE, you have to create a new HTE database file (hte.db) or you can select an existing database file. Further, you can set the threshold for the repeated execution in the preference page which applies to all nodes. If you want to get an Email whenever a node fails after repeated execution, you have to set an Email host and an address to send and receive Emails.

6 FAQ

1. **Which nodes can work on compressed input files?**
   –> Compatible with compressed FastQ files: FastQC, RawReadManipulator
   –> Compatible with compressed vcf files: VEP

2. **No active session error in parallel chunk environment.**
   –> Reset parallel chunk start and re-execute.

3. **The node wont execute and always returns to "orange configure-state" without error message.**
   –> Reset the last successfully executed predecessor node and re-execute the nodes.

4. **Update has encountered a problem.**
   –> Restart KNIME before updating.

5. **Node is in EXECUTING state and cannot be canceled although it has been successfully executed on the cluster.**
   –> Go to the corresponding workflow folder in the knime-workspace. Edit the xml file of the executing node and replace EXECUTING by IDLE or CONFIGURED.

GATK errors

1. **VQSR: Unable to retrieve results.**
   –> Use VQSR in single-threaded mode (Threads 1).

2. **VQSR: No data found.**
   –> Reduce maximum number of Gaussians to 4.

7 Overview of all KNIME4NGS nodes

Node	Description
Bcftools	A set of utilities that manipulate variant calls in the Variant Call Format (VCF) and its binary counterpart BCF.
Bowtie2	An ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences.
BWA	The Burrows-Wheeler Alignment Tool (BWA) aligns relatively short sequences (queries) to a sequence database (target), such as the human reference genome.
DepthOfCoverage	Assess sequence coverage by a wide array of metrics, partitioned by sample, read group, or library.
DESeq	This node uses the DESeq package to test for differential expression based on a model using the negative binomial distribution. The node tries to automatically install the missing R packages during execution. If this fails, we recommend to manually install the packages via the Bioconductor.
EdgeR	Assessing differential expression in comparative RNA-seq experiments. The node tries to automatically install the missing R packages during execution. If this fails, we recommend to manually install the packages via the Bioconductor.
FastaSelector	Selects multiple fasta files.
FastQC	Quality check for NGS read data.
FastqcStatisticMerger	Merges statistics of FastQC modules.
FeatureCounts	An efficient general purpose program for assigning sequence reads to genomic features.
FeatureCountsMerger	Merges the results of multiple feature count runs.
FeatureCountsStatisticMerger	Merges statistics of the FeatureCounts node.
FileLoader	Loads either one or two separate input file(s) or a list of files whereby it serves as a starting point for any NGS workflow.
FilterLowExpressed	Filters low expressed genes before differential expression analysis. The node tries to automatically install the missing R packages during execution. If this fails, we recommend to manually install the packages via the Bioconductor.
GATKBaseRecalibration	Recalibration of sequencing quality scores.
CombineGVCFs	Combines any number of gVCF files that were produced by the Haplotype Caller into a single joint gVCF file.
CombineVCFs	Combines any number of VCF lists that were produced by Pindel into a single joint VCF file.
GenotypeGVCFs	Performs joint genotyping on gVCF files produced by HaplotypeCaller.
GATKHaplotypeCaller	Calls germline SNPs and indels via local re-assembly of haplotypes.

MergeTwoVCFs	Merges two VCF files into one combined VCF file.
GATKPhaseByTransmission	Computes the most likely genotype combination and phases trios and parent/child pairs.
GATKRealignment	Realignment around insertions and deletions.
GATKSelectVariants	Selects a subset of variants from a larger callset.
GATKUnifiedGenotyper	Calls SNPs and indels.
GATKVariantFiltration	Filters variant calls based on INFO and FORMAT annotations.
KGGSeq	Filters and prioritizes genetic variants from whole exome sequencing data.
Limma	Detects differential expressed genes by using the limma package. The node tries to automatically install the missing R packages during execution. If this fails, we recommend to manually install the packages via the Bioconductor.
PicardTools	Tools to manipulate SAM/BAM files.
Pindel	A pattern growth approach to detect breakpoints of large deletions and medium sized insertions from paired-end reads.
RawReadManipulator	Filters reads from fastQ/fastA files.
RawReadManipulator StatisticMerger	Merges statistics of the RawReadManipulator node.
SamTools	Provides access to many functionalities of SAMtools.
Segemehl	Heuristic mapping of short read sequences with gaps.
SnpEff	Annotates and predicts the effects of genetic variants.
SnpSift	A wrapper for the SnpSift toolbox.
Star	Aligns RNA-seq reads to a reference genome using uncompressed suffix arrays and is "splice aware".
StarStatisticMerger	Merges statistics of the STAR node.
VCFMerger	Combines VCF records from different sources.
VCFSorter	Sorts the chromosomes of an input VCF file analogous to a given reference sequence.
VCFToolsFilter	Filters annotated VCF files.
VEP	The Variant Effect Predictor (VEP) annotates genomic variants.
VEPFilter	Filters VCF files annotated by VEP.
VQSR	Variant Quality Score Recalibration: creates a Gaussian mixture model by looking at the annotations values over a high quality subset of the input call set and then evaluates all input variants.

Table 7.1: Overview of all KNIME4NGS nodes

7.1 In-house tools

7.1.1 FastQCmod

FastQCmod is an extension to the great FastQC (currently build on release v0.10.1). The extension includes a logic to extract the analysis results of the individual FastQC analysis methods and write a more computer readable output, to allow for a simplified reusal by e.g. RawReadManipulator. Furtheron, in the light of the fact that CPU power is currently more widely available than I/O-Bandwidth, FastQCmod contains code for parallelization of the analysis of single fastq-files, compared to the per-sample/per-file parallelization of the original FastQC. A backport of recent FastQC versions to FastQCmod is in preparation. The source-code of FastQCmod (based on FastQC v0.10.1) is available under the GPLv3 license.

7.1.2 RawReadManipulator

The RawReadManipulator (RRM) is an in-house development of a modular, highly parallelized filtering pipeline for raw reads. The java application is easily extensible with new or custom filtering-modules as necessary, and makes use of the data-independence that reads (or read-pairs) have towards each other. All available filters have many available configuration options, such that the most common filtering needs should be met when using RRM. Tests have shown the parallelization to be scaling well with the number of CPUs given to the task, with the bottleneck of available data-I/O-bandwith as the limiting factor in terms of filtering-speed.