

---

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

---

Maximilian Hastreiter, Jonathan Hoser, Tim Jeske, Michael Kluge, Marie-Sophie Friedl,  
Sebastian Kopetzky, Jan-Dominik Quell, Robert Küffner and H.-Werner Mewes  
June 10, 2016

## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Prerequisites . . . . .	2
1.2	Installation . . . . .	2
<b>2</b>	<b>Basic Usage</b>	<b>3</b>
2.1	File Loading and Pipeline Entry Point . . . . .	3
2.2	Read Filtering and Quality Control . . . . .	3
2.3	Mapping and SAM-BAM Conversion . . . . .	4
2.4	Alignment Refinement . . . . .	4
2.5	Variant Calling and Filtering . . . . .	5
2.6	Variant Annotation . . . . .	6
2.7	Differential Expression . . . . .	6
<b>3</b>	<b>Parallel Execution</b>	<b>6</b>
<b>4</b>	<b>The HTE</b>	<b>7</b>
<b>5</b>	<b>Preference Page</b>	<b>8</b>
<b>6</b>	<b>FAQ</b>	<b>10</b>

# 1 INTRODUCTION

## 1.1 PREREQUISITES

- Linux System (required)
- KNIME or KNIME SDK (required)
- KNIME Cluster Extension<sup>1</sup> (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 <sup>2</sup>
- Variant Effect Predictor (VEP) script including the Ensembl Core and Variation APIs (for variant annotation, see Section 2.6) <sup>3</sup> VEP itself requires the following perl modules: Archive::Extract, Archive::Zip, Test::More, DBI and CGI

**To do** specify supported versions of KNIME, R, GATK, VEP (1)

## 1.2 INSTALLATION

1. Start KNIME
2. Add the KNIME4NGS Update Site by selecting Help -> Install New Software... -> Add (Figure 1.2). The Update Site is located at:  
`/home/share/tmp/knime/de.helmholtz_muenchen.ibis.knimenodes.update.zip`
3. Select the new Update 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:
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.

**To do** specify publication platform for update site, sample data (2) **To do** licensing -> ask ICB group (3) **To do** define example workf

---

<sup>1</sup><https://www.knime.org/cluster-execution>

<sup>2</sup><https://www.broadinstitute.org/gatk/>

<sup>3</sup>[http://www.ensembl.org/info/docs/tools/vep/script/vep\\_download.html](http://www.ensembl.org/info/docs/tools/vep/script/vep_download.html)

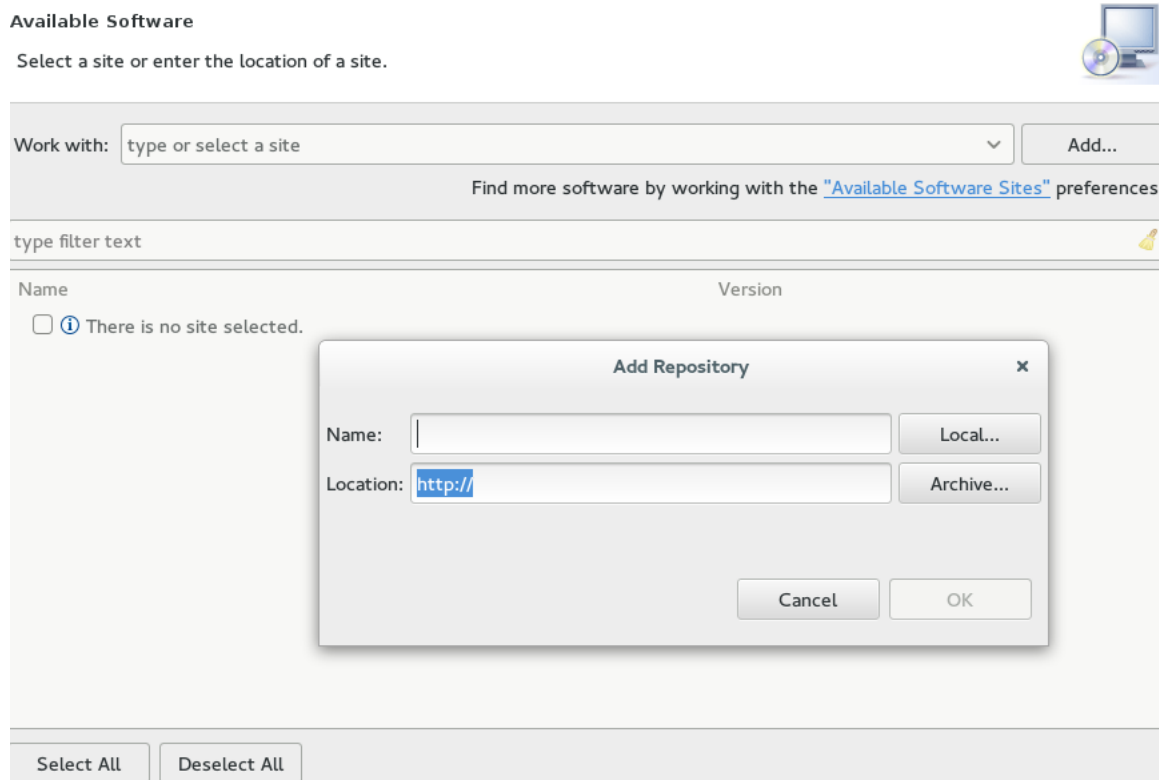


Figure 1.1: Add KNIME4NGS Update Site as repository

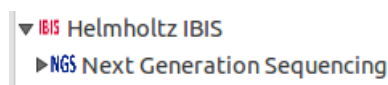


Figure 1.2: KNIME4NGS nodes appear in the KNIME Node Repository

## 2 BASIC USAGE

**To do high priority:** final iteration: sabotage, configure method, node description, minimum one node a day! (6)

### 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 FastQ, SAM/BAM or VCF/gVCF files. After node execution, the selected file path(s) will be displayed in the nodes output table.

### 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.

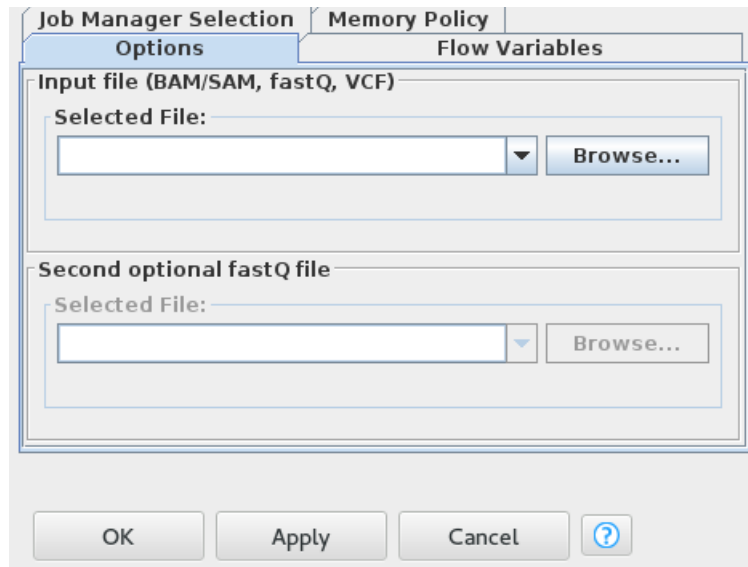


Figure 2.1: Loading files using the FileLoader node.

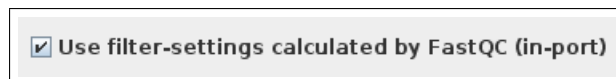


Figure 2.2: The RawReadManipulator can use proposed filter settings created by FastQC.

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.

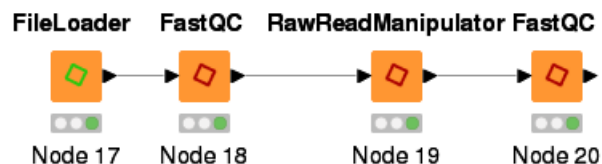


Figure 2.3: Quality Check and Filtering Workflow

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

**To do** FastQC "sh mergeFsettings" funktioniert so unter Ubuntu nicht (Max Home Pc). Geht nur ohne "sh". (8)

## 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

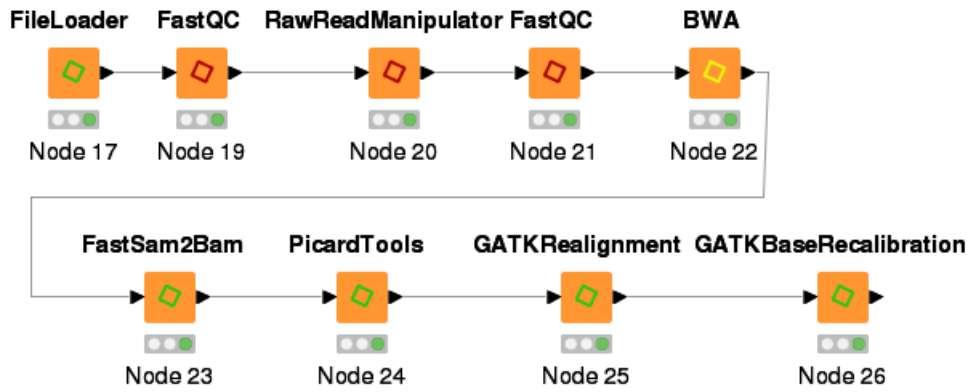


Figure 2.4: Mapping, Conversion and Alignment Refinement

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.

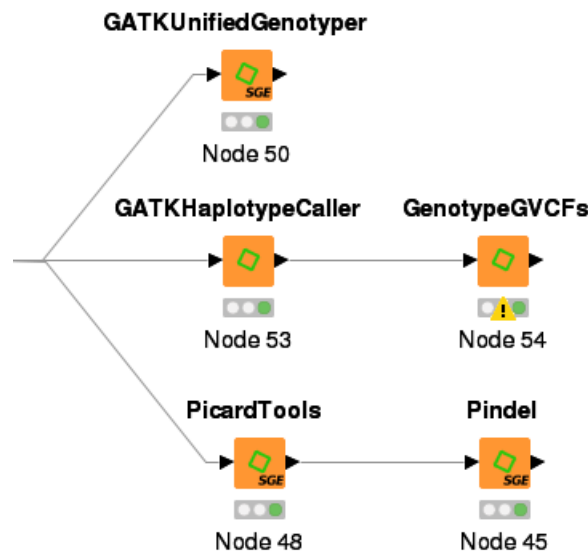


Figure 2.5: For Variant Calling, three different nodes are provided, each based on different calling algorithms.

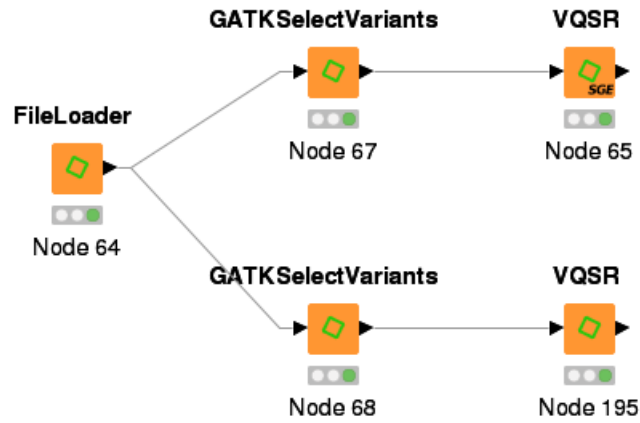


Figure 2.6: VQSR

## 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.

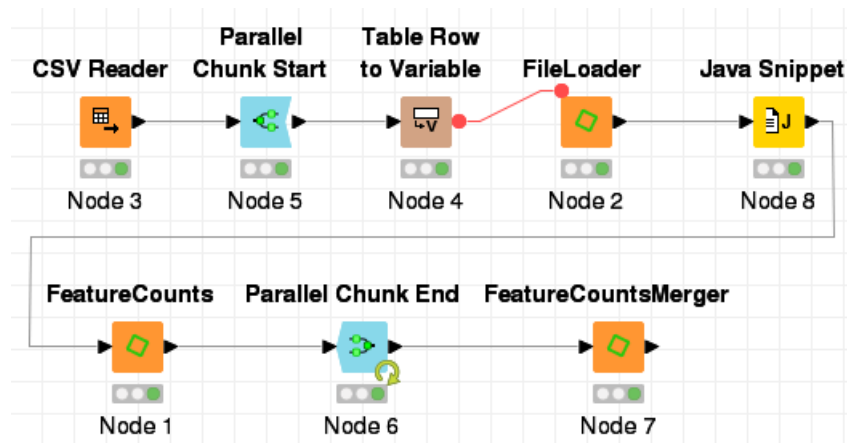


Figure 2.7: Differential Expression

## 3 PARALLEL EXECUTION

Analysing 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.

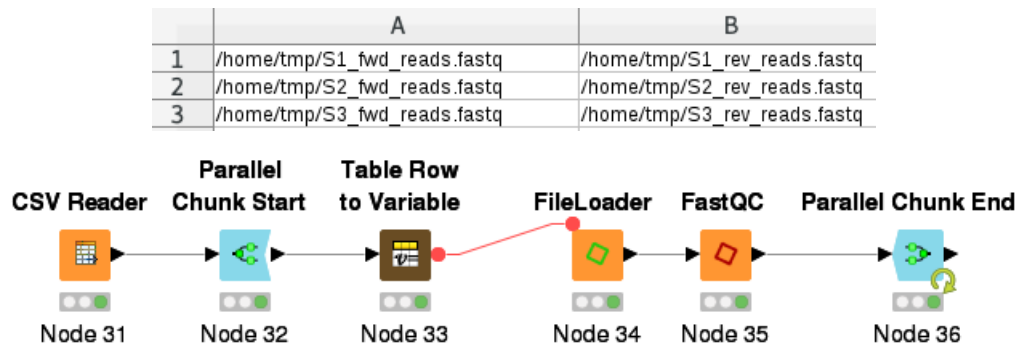


Figure 3.1: Parallel execution

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). By using the **Parallel Chunk Start**, each row will be handled separately. Thus, the input files for each chunk can be imported via the FlowVariables of **Table Row to Variable** node which is shown in Figure 3.1. In this example, for each row of the input csv-file, the **FastQC** quality control is performed.

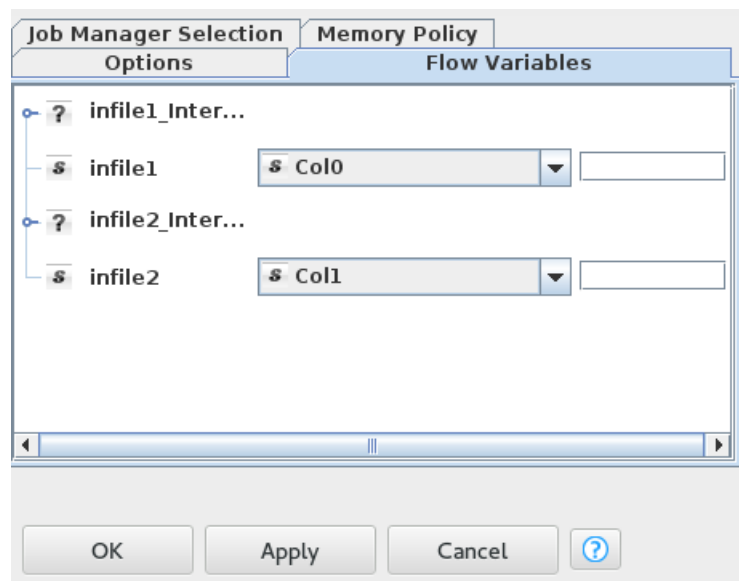


Figure 3.2: Using FlowVariables to select the input files of the FileLoader node.

## 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.

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. To use the HTE mode you also have to create a HTE database (the hte.db file) which logs the errors

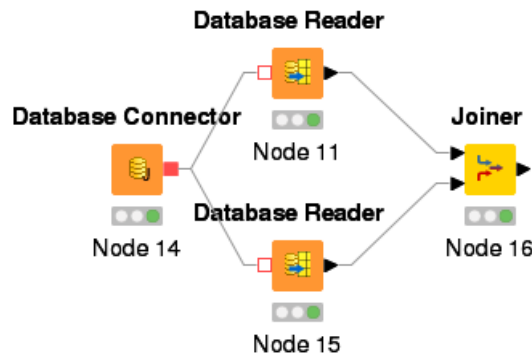


Figure 4.1: Read the HTE database

occurring during the execution of several workflows. The HTE database consists of two tables: the HTEExecution table comprises entries for each executed node and the HTEError\_History table contains the error messages if the execution failed at least once.

You can easily access the database using KNIME nodes that are included in the basic repository. An exemplary workflow is shown in figure 4.1. Before configuring the Database Connector you have to load a database driver file to connect to the SQLite data base. The driver file is included in the KNIME4NGS extension and can be found in your KNIME installation: `knime_*/plugins/de.helmholtz_muenchen.ibis.knimenodes_*/libs/sqlite-jdbc-*.jar`. In order to load the driver file, go to "Preferences > KNIME > Databases > New...". Afterwards, you have to choose "org.sqlite.JDBC" in the configuration dialog of the Database Connector and set the database url as "jdbc:sqlite:YOUR\_PATH\_TO\_hte.db". The subsequent Database Reader node allows you to inspect the fields of both tables via "Fetch Metadata" and to execute SQL queries. The query results of both tables can be joined by using the Join node provided by KNIME.

The termination state of a node is saved in files ending with klock. These klock 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 klock file.

**To do** review database connection section if SQLiteConnector is available in eclipse KNIME (10)

## 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.

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. Note, that not all binaries can be download due to license issues. Finally, 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. **GC overhead limit exceeded while running Picard node.**
  - Increase memory when starting KNIME using the knime.ini file e.g. by adding -Xmx8G
6. **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.

Node	Description
<b>Bcftools</b>	A set of utilities that manipulate variant calls in the Variant Call Format (VCF) and its binary counterpart BCF.
<b>Bowtie2</b>	An ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences.
<b>BWA</b>	The Burrows-Wheeler Alignment Tool (BWA) aligns relatively short sequences (queries) to a sequence database (target), such as the human reference genome.
<b>DepthOfCoverage</b>	Assess sequence coverage by a wide array of metrics, partitioned by sample, read group, or library.
<b>DESeq</b>	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.
<b>EdgeR</b>	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.

<b>FastaSelector</b>	Selects multiple fasta files.
<b>FastQC</b>	Quality check for NGS read data.
<b>FastqcStatisticMerger</b>	Merges statistics of FastQC modules.
<b>FeatureCounts</b>	An efficient general purpose program for assigning sequence reads to genomic features.
<b>FeatureCountsMerger</b>	Merges the results of multiple feature count runs.
<b>FeatureCountsStatisticMerger</b>	Merges statistics of the FeatureCounts node.
<b>FileLoader</b>	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.
<b>FilterLowExpressed</b>	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.
<b>GATKBaseRecalibration</b>	Recalibration of sequencing quality scores.
<b>CombineGVCFs</b>	Combines any number of gVCF files that were produced by the Haplotype Caller into a single joint gVCF file.
<b>CombineVCFs</b>	Combines any number of VCF lists that were produced by Pindel into a single joint VCF file.
<b>GenotypeGVCFs</b>	Performs joint genotyping on gVCF files produced by HaplotypeCaller.
<b>GATKHaplotypeCaller</b>	Calls germline SNPs and indels via local re-assembly of haplotypes.
<b>MergeTwoVCFs</b>	Merges two VCF files into one combined VCF file.
<b>GATKPhaseByTransmission</b>	Computes the most likely genotype combination and phases trios and parent/child pairs.
<b>GATKRealignment</b>	Realignment around insertions and deletions.
<b>GATKSelectVariants</b>	Selects a subset of variants from a larger callset.
<b>GATKUnifiedGenotyper</b>	Calls SNPs and indels.
<b>GATKVariantFiltration</b>	Filters variant calls based on INFO and FORMAT annotations.
<b>KGGSeq</b>	Filters and prioritizes genetic variants from whole exome sequencing data.
<b>Limma</b>	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.
<b>PicardTools</b>	Tools to manipulate SAM/BAM files.
<b>Pindel</b>	A pattern growth approach to detect breakpoints of large deletions and medium sized insertions from paired-end reads.
<b>RawReadManipulator</b>	Filters reads from fastQ/fastA files.

<b>RawReadManipulator StatisticMerger</b>	Merges statistics of the RawReadManipulator node.
<b>SamTools</b>	Provides access to many functionalities of SAMtools.
<b>Segemehl</b>	Heuristic mapping of short read sequences with gaps.
<b>Snpeff</b>	Annotates and predicts the effects of genetic variants.
<b>Snpsift</b>	A wrapper for the Snpsift toolbox.
<b>Star</b>	Aligns RNA-seq reads to a reference genome using uncompressed suffix arrays and is "splice aware".
<b>StarStatisticMerger</b>	Merges statistics of the STAR node.
<b>VCFMerger</b>	Combines VCF records from different sources.
<b>VCFSorter</b>	Sorts the chromosomes of an input VCF file analogous to a given reference sequence.
<b>VCFToolsFilter</b>	Filters annotated VCF files.
<b>VEP</b>	The Variant Effect Predictor (VEP) annotates genomic variants.
<b>VEPFilter</b>	Filters VCF files annotated by VEP.
<b>VQSR</b>	Creates a Gaussian mixture model by looking at the annotations values over a high quality subset of the input call set and then evaluate all input variants.

Table 6.1:

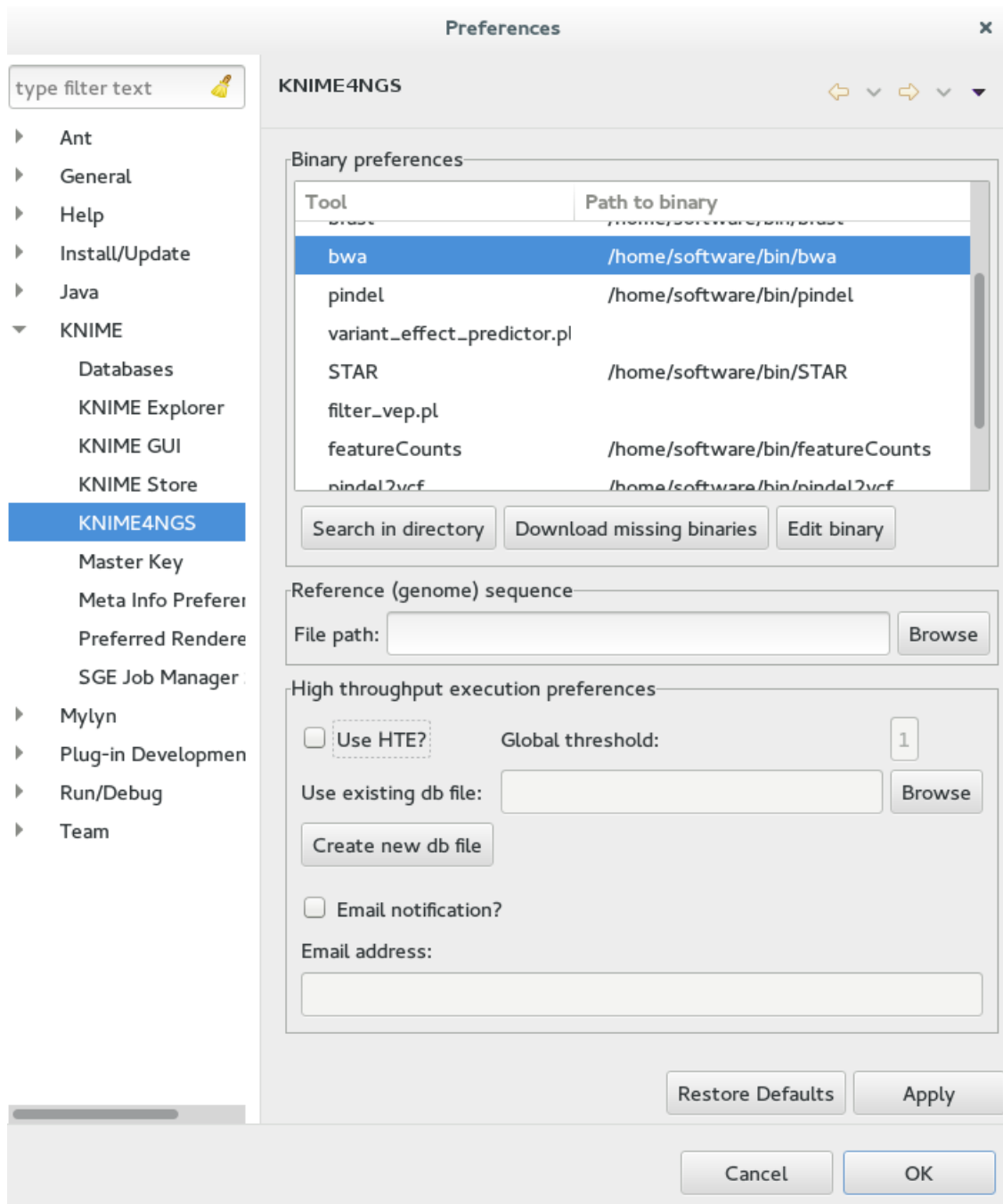


Figure 5.1: KNIME4NGS Preference Page. It can be found in "File > Preferences > KNIME > KNIME4NGS"

## TO DO...

- ☐ 1 (p. 2): specify supported versions of KNIME, R, GATK, VEP
- ☐ 2 (p. 2): specify publication platform for update site, sample data
- ☐ 3 (p. 2): licensing → ask ICB group
- ☐ 4 (p. 2): define example workflow including input files and reference genome
- ☒ 5 (p. 3): define nodes to publish → 47 nodes
- ☐ 6 (p. 3): **high priority**: final iteration: sabotage, configure method, node description, minimum one node a day!
- ☒ 7 (p. 3): **high priority**: enable FileLoader to handle lists of input files
- ☐ 8 (p. 4): FastQC "sh mergeFsettings" funktioniert so unter Ubuntu nicht (Max Home Pc). Geht nur ohne "sh".
- ☒ 9 (p. 6): **high priority**: modify FeatureCounts node to avoid JavaSnippet in workflows
- ☐ 10 (p. 8): review database connection section if SQLiteConnector is available in eclipse KNIME