```
{
    "id": "obj.1481139873",
    "name": "Exome Sequence Analysis",
    "version": "5",
    "digital_signature": "",
    "verification_status": "unreviewed",
    "publication_status": "draft",
    "usability_domain": ["For use in analysis of xome sequence data"],
    "authors": [{"name": "Durga"}],
    "description_domain": {
        "xref":[],
        "keywords": [],
        "pipeline_steps": {
            "FastQC":{
                "description": "A quality control tool for high throughput sequence
data.",
                "version": "sbg:toolkitVersion:0.11.4",
                "step_number": 1,
                "package":
["http://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0.11.4.zip"],
                "input":["sbg://Durga/exome-sequencing/fastq_list.fastq.gz"],
                "output":[
                    "#FastQC.report_zip",
                    "#FastQC.report_html"
                ]
            },
            "SBG_Pair_FASTQs_by_Metadata":{
                "description": "Tool accepts list of FASTQ files for one sample as
the input and groups them into pairs  (two files for each paired end). This
grouping is done using metadata values that are creating unique combination for
each pair or of FASTQ files. Metadata that fields that are uniquely defining one
FASTQ pair are Sample ID, Library ID, Platform unit ID and File segment number.
Listed order of metadata fields is also representing their hierarchy in the
metadata structure. Not all of these four metadata fields are required, but the
present set has to be sufficient to create unique combinations for each pair of
FASTQ files.",
                "version": null,
                "step_number": 1,

"package":["https://cgc.sbgenomics.com/u/stefanristeski/group-fastqs/apps/#sbg-pair-
fastqs-by-metadata/9"],
                "input":["#SBG_Pair_FASTQs_by_Metadata.fastq_list"],
                "output":["#SBG_Pair_FASTQs_by_Metadata.tuple_list"]
            },
            "BWA_INDEX":{
                "description": "BWA INDEX constructs the FM-index (Full-text index
in Minute space) for the reference genome.\nGenerated index files will be used with
BWA MEM, BWA ALN, BWA SAMPE and BWA SAMSE tools.\n\nIf input reference file has TAR
extension it is assumed that BWA indices came together with it. BWA INDEX will only
pass that TAR to the output. If input is not TAR, the creation of BWA indices and
```

```
its packing in TAR file (together with the reference) will be performed.",
                "version": "sbg:toolkitVersion:0.7.13",
                "step_number": 1,
                "package":["http://sourceforge.net/projects/bio-bwa/"],
                "input":["sbg://Durga/exome-sequencing/reference.fasta"],
                "output":["#BWA_INDEX.indexed_reference"]
            },
            "SBG_FASTA_Indices": {
                "description": "Tool allows creating FASTA dictionary and index
simultaneously which is necessary for running GATK tools. This version of tool for
indexing uses SAMtools faidx command (toolkit version0.1.19), while for the FASTA
dictionary is used Picard CreateFastaDictionary (toolkit version 1.140)",
                "version": null,
                "step_number": 1,

"package":["/u/Durga/exome-sequencing/apps/#Durga/exome-sequencing/sbg-fasta-indices
/0"],
                "input":["#SBG_FASTA_Indices.reference"],
                "output":[ "#SBG_FASTA_Indices.fasta_reference",
"#SBG_FASTA_Indices.fasta_index", "#SBG_FASTA_Indices.fasta_dict"]
            },
            "BWA_MEM_Bundle":{
                "description": "**BWA MEM** is an algorithm designed for aligning
sequence reads onto a large reference genome. BWA MEM is implemented as a component
of BWA.",
                "version": "sbg:toolkitVersion:0.7.13",
                "step_number": 2,
                "package":["http://sourceforge.net/projects/bio-bwa/"],
                "input":[
                    "#BWA_INDEX.parameterList",
                    "#SBG_Pair_FASTQs_by_Metadata.tuple_list",
                    "#BWA_INDEX.indexed_reference"
                ],
                "output":[
                    "#BWA_MEM_Bundle.bam_index",
                    "#BWA_MEM_Bundle.aligned_reads"
                ]
            },
            "Sambamba_Merge":{
                "description": "Sambamba Merge is used for merging several sorted
BAM files into one. The sorting order of all the files must be the same, and it is
maintained in the output file.",
                "version": "sbg:toolkitVersion:0.5.9",
                "step_number": 3,

"package":["https://github.com/lomereiter/sambamba/releases/tag/v0.5.9"],
                "input":["#BWA_MEM_Bundle.aligned_reads"],
                "output":["#Sambamba_Merge.merged_bam"]
            },
            "Picard_BuildBamIndex":{
                "description": "Picard BuildBamIndex generates a BAM index (.bai)
```

```
file.",
                "version": "sbg:toolkitVersion:1.140",
                "step_number": 4,

"package":["https://github.com/broadinstitute/picard/zipball/master"],
                "input":["#Sambamba_Merge.merged_bam"],

"output":["#Picard_BuildBamIndex.indexed_bam","#Picard_BuildBamIndex.index"]
            },
            "GATK_IndelRealigner":{
                "description": "The local realignment process is designed to
consume one or more BAM files and to locally realign reads such that the number of
mismatching bases is minimized across all the reads. In general, a large percent of
regions requiring local realignment are due to the presence of an insertion or
deletion (indels) in the individual's genome with respect to the reference genome.
Such alignment artifacts result in many bases mismatching the reference near the
misalignment, which are easily mistaken as SNPs. Moreover, since read mapping
algorithms operate on each read independently, it is impossible to place reads on
the reference genome such at mismatches are minimized across all reads.
Consequently, even when some reads are correctly mapped with indels, reads covering
the indel near just the start or end of the read are often incorrectly mapped with
respect the true indel, also requiring realignment. Local realignment serves to
transform regions with misalignments due to indels into clean reads containing a
consensus indel suitable for standard variant discovery approaches. Unlike most
mappers, this walker uses the full alignment context to determine whether an
appropriate alternate reference (i.e. indel) exists. Following local realignment,
the GATK tool Unified Genotyper can be used to sensitively and specifically
identify indels.\n\nThere are 2 steps to the realignment process:\n\n1. Determining
(small) suspicious intervals which are likely in need of realignment (see the
RealignerTargetCreator tool)\n2. Running the realigner over those intervals
(IndelRealigner)\nFor more details, see the indel realignment method
documentation.\n\nInput\nOne or more aligned BAM files and optionally one or more
lists of known indels.\n\nOutput\nA realigned version of your input BAM
file(s).\n\nUsage example:\n java -jar GenomeAnalysisTK.jar \\\n   -T
IndelRealigner \\\n   -R reference.fasta \\\n   -I input.bam \\\n   --known
indels.vcf \\\n   -targetIntervals intervalListFromRTC.intervals \\\n   -o
realignedBam.bam\n \nCaveats\n\nThe input BAM(s), reference, and known indel
file(s) should be the same ones to be used for the IndelRealigner step.\nBecause
reads produced from the 454 technology inherently contain false indels, the
realigner will not work with them (or with reads from similar technologies).\nThis
tool also ignores MQ0 reads and reads with consecutive indel operators in the CIGAR
string.\n\n(IMPORTANT) Reference \".fasta\" Secondary Files\n\nTools in GATK that
require a fasta reference file also look for the reference file's corresponding
.fai (fasta index) and .dict (fasta dictionary) files. The fasta index file allows
random access to reference bases and the dictionary file is a dictionary of the
contig names and sizes contained within the fasta reference. These two secondary
files are essential for GATK to work properly. To append these two files to your
fasta reference please use the 'SBG FASTA Indices' tool within your GATK based
workflow before using any of the GATK tools.",
                "version": "sbg:toolkitVersion:2.3.9 Lite",
                "step_number": 5,
```

```
"package":["https://www.broadinstitute.org/gatk/download/auth?package=GATK-archive&version=2.3-9-ge5ebf34"],
                "input":[
                    "sbg://Durga/exome-sequencing/target_intervals.txt",
                    "#SBG_FASTA_Indices.fasta_reference",
                    "#Picard_BuildBamIndex.indexed_bam"
                ],
                "output":["#GATK_IndelRealigner.realigned_bam_file"]
            },
            "Freebayes":{
                "description": "FreeBayes is a Bayesian genetic variant detector designed to find small polymorphisms, specifically SNPs (single-nucleotide polymorphisms), indels (insertions and deletions), MNPs (multi-nucleotide polymorphisms), and complex events (composite insertion and substitution events) smaller than the length of a short-read sequencing alignment.\n\nFreeBayes incorporates a number of features in order to reduce the complexity of variant detection for researchers and developers:\n\n1. Indel realignment is accomplished internally using a read-independent method, and issues resulting from discordant alignments are dramatically reduced through the direct detection of haplotypes.\n2. The need for base quality recalibration is avoided through the direct detection of haplotypes. Sequencing platform errors tend to cluster (e.g. at the ends of reads), and generate unique, non-repeating haplotypes at a given locus.\n3. Variant quality recalibration is avoided by incorporating a number of metrics, such as read placement bias and allele balance, directly into the Bayesian model.\n\n###Common Issues\nFASTA INDEX FILE is not required. If it is not provided (as secondary file), it is generated. When it is provided, the tool runs faster.\nBAM INDEX FILES are not required, but when not provided (as separate input), random access is disabled.\nVARIANT INPUT INDEX FILE is required (as secondary file). It should be generated using Tabix BGZIP and Tabix Index file.\nREGION parameter should match data present in variant input file, both chromosome and positions.",
                "version": "sbg:toolkitVersion:v1.0.2",
                "step_number": 6,
                "package":["https://github.com/ekg/freebayes"],
                "input":["#SBG_FASTA_Indices.fasta_reference",
"#GATK_IndelRealigner.realigned_bam_file"],
                "output":["sbg://Durga/exome-sequencing/Freebayes.variants"]
            }
        }
    },
    "execution_domain": {
        "platform": "SBG",
        "url":
"sbg://Durga/exome-sequencing/exomeseqanalysis02-removesortaddparameters/5/raw/",
        "pipeline_version": "5",
        "env_parameters": ["MemRequirement:2500",
"CPURequirement:Eval_input_read_size"],
        "driver": "SBGdriver",
        "script":
"sbg://Durga/exome-sequencing/exomeseqanalysis02-removesortaddparameters/5/raw/",
        "prerequisites": [
```

```
            {"name":"FastQC","version":"sbg:toolkitVersion:0.11.4"},
            {"name":"SBG_Pair_FASTQs_by_Metadata","version": null},
            {"name":"BWA_INDEX","version":"sbg:toolkitVersion:0.7.13"},
            {"name":"SBG_FASTA_Indices","version":null},
            {"name":"BWA_MEM_Bundle","version":"sbg:toolkitVersion:0.7.13"},
            {"name":"Sambamba_Merge","version":"sbg:toolkitVersion:0.5.9"},
            {"name":"Picard_BuildBamIndex","version":"sbg:toolkitVersion:1.140"},
            {"name":"GATK_IndelRealigner","version":"sbg:toolkitVersion:2.3.9
Lite"},
            {"name":"Freebayes","version":"sbg:toolkitVersion:v1.0.2"}
        ]
    },
    "parametric_domain": {
        "FastQC_kmers": "7",
        "FastQC_threads": "1",
        "BWA_INDEX_bwt_construction": "auto",
        "BWA_INDEX_block_size": "10000000",
        "BWA_MEM_Bundle_threads": "8",
        "BWA_MEM_Bundle_minimum_seed_length": "19",
        "BWA_MEM_Bundle_dropoff": "100",
        "BWA_MEM_Bundle_select_seeds": "1.5",
        "BWA_MEM_Bundle_seed_occurrence_for_the_3rd_round": "20",
        "BWA_MEM_Bundle_skip_seeds": "500",
        "BWA_MEM_Bundle_drop_chains_fraction": "0.5",
        "BWA_MEM_Bundle_discard_chain_length": "0",
        "BWA_MEM_Bundle_mate_rescue_rounds": "50",
        "BWA_MEM_Bundle_score_for_a_sequence_match": "1",
        "BWA_MEM_Bundle_mismatch_penalty": "4",
        "BWA_MEM_Bundle_verbose_level": "3",
        "BWA_MEM_Bundle_minimum_output_score": "30",
        "BWA_MEM_Bundle_total_memory": "15",
        "Sambamba_Merge_mem_mb": "1024",
        "Sambamba_Merge_reserved_threads": "1",
        "Picard_BuildBamIndex_max_records_in_ram": "500000",
        "Picard_BuildBamIndex_compression_level": "5",
        "Picard_BuildBamIndex_validation_stringency": "SILENT",
        "Picard_BuildBamIndex_quiet": "FALSE",
        "Picard_BuildBamIndex_verbosity": "INFO",
        "Picard_BuildBamIndex_memory_per_job": "2048",
        "Picard_BuildBamIndex_output_index": "FALSE",
        "GATK_IndelRealigner_disable_radnomization": "FALSE",
        "GATK_IndelRealigner_allow_potentailly_misencoded_quals": "FALSE",
        "GATK_IndelRealigner_baq": "OFF",
        "GATK_IndelRealigner_baq_gap_open_penalty": "40",
        "GATK_IndelRealigner_default_base_qualities": "-1",
        "GATK_IndelRealigner_disable_indel_quals": "FALSE",
        "GATK_IndelRealigner_use_legacy_downsampler": "FALSE",
        "GATK_IndelRealigner_use_original_qualities": "FALSE",
        "GATK_IndelRealigner_validation_strictness": "SILENT",
        "GATK_IndelRealigner_memory_per_job": "2048",
        "GATK_IndelRealigner_cpu_per_job": "1",
```

```
        "GATK_IndelRealigner_consensus_determination_model": "USE_READS",
        "GATK_IndelRealigner_lod_threshold_for_cleaning": "5",
        "GATK_IndelRealigner_entropy_threshold": "0.15",
        "GATK_IndelRealigner_max_consensuses": "30",
        "GATK_IndelRealigner_max_isize_for_movement": "3000",
        "GATK_IndelRealigner_max_positional_move_allowed": "200",
        "GATK_IndelRealigner_max_reads_for_consensuses": "120",
        "GATK_IndelRealigner_max_reads_for_realignment": "20000",
        "GATK_IndelRealigner_max_reads_in_memory": "150000",
        "GATK_IndelRealigner_no_original_alignment_tags": "FALSE",
        "Freebayes_min_mapping_quality": "1",
        "Freebayes_min_base_quality": "0",
        "Freebayes_min_supporting_allele_qsum": "0",
        "Freebayes_min_supporting_mapping_qsum": "0",
        "Freebayes_mismatch_base_quality_threshold": "10",
        "Freebayes_read_mismatch_limit": "unbounded",
        "Freebayes_read_max_mismatch_fraction": "1",
        "Freebayes_genotyping_max_iterations": "1000",
        "Freebayes_genotyping_max_banddepth": "6",
        "Freebayes_posterior_integration_limits": "1,3",
        "Freebayes_gvcf": "TRUE"
    },
    "io_domain": {
        "reference_uri": [
            "sbg://Durga/exome-sequencing/reference.fasta",
            "sbg://Durga/exome-sequencing/target_intervals.txt"
        ],
        "input_uri_list": [
            "sbg://Durga/exome-sequencing/fastq_list.fastq.gz"
        ],
        "output_uri_list": [
            "sbg://Durga/exome-sequencing/BWA_MEM_Bundle.aligned_reads",
            "sbg://Durga/exome-sequencing/FastQC.report_zip",
            "sbg://Durga/exome-sequencing/FastQC.report_html",
            "sbg://Durga/exome-sequencing/Freebayes.variants",
            "sbg://Durga/exome-sequencing/Sambamba_Merge.merged_bam"
        ]
        }
}
```