BioCompute Object for Regulatory Review

BCO Title: HISAT2-StringTie Workflow

BCO Generation Date: February 16, 2023

BCO Specification Version: v1.3.0

BCO Generator: Seven Bridges

Contents

1 BioCompute Object Domain Entries

1.1 Top Level Fields

["https://w3id.org/biocompute/1.4.2/",
"https://biocompute.sbgenomics.com/bco/57f53382-3b30-4712-ae02-4d57d7721d5b",
"ff2bab981cfc97700456b92211c1946e7ac19693c9b43d3055683225add44568"]

1.2 Provenance Domain

{
"name": "HISAT2-StringTie Workflow",
"version": "1.0.0",
"review": [],
"derived_from":
"https://cgc-api.sbgenomics.com/v2/apps/phil_webster/bco-cwl-examples/hisat2-stringtie/0/raw/"
"obsolete_after": "2023-02-16T00:00:00+0000",
"embargo": ["2023-02-16T00:00:00+0000",
"2023-02-16T00:00:00+0000"],
"created": "2023-02-16T00:00:00+0000",
"modified": "2023-02-16T00:00:00+0000",
"contributors": [],
"license": "https://spdx.org/licenses/CC-BY-4.0.html"
}

1.3 Usability Domain

"The __HISAT2-StringTie Workflow__ can be used to perform a
gene abundance estimation of RNA-Seq data (i.e.
quantification) of a unified set of genes common for all
samples in the analysis. The workflow is based on
the Nature protocol paper [1] with the last step
(differential expression testing) omitted.\n\nThe first
part of the workflow indexes and aligns RNA-Seq reads to
the reference using tools from the __HISAT2 2.2.1__

```
toolkit. The second part performs quantification and, if
specified, transcriptome assembly using tools from
the __StringTie 2.1.3__ toolkit. Depending on the value of
the __Estimate novel isoform abundance?__ required
parameter, analysis will be continued in one of two
possible ways:\n\n1) If set to 'True' then aligned reads
are fed to the first run of __StringTie__ which performs
'reference guided' transcriptome assembly for each sample.
Assembled transcripts and reference annotation transcripts
are then merged into a uniform set of transcripts. The
second run of __StringTie__ performs a quantification of
merged transcripts and genes in each sample. For each
sample, the workflow outputs the abundance estimation in
the __Ballgown__ input format (one TAR bundle containing 5
.ctab tables), two files containing transcript and gene
expression values in the __DESeq2__ input format, and a TAB
file containing FPKM, TPM, and Coverage values for each
gene. In addition to files containing the abundance
estimation, the workflow also outputs a GTF file with
merged transcripts, __GffCompare__ output files with
comparison results between reference annotation transcripts
and merged transcripts.\n\n2) If set to 'False' then
aligned reads are only fed to __StringTie__ once, this run
performs quantification in each sample. For each sample,
the workflow outputs the abundance estimation in the
__Ballgown__ input format (one TAR bundle containing 5
.ctab tables), two files containing transcript and gene
expression values in the __DESeq2__ input format, and a TAB
file containing FPKM, TPM, and Coverage values for each
gene. \n\n*A list of **all inputs and parameters** with
corresponding descriptions can be found at the bottom of
this page.*\n\n***Please note that any cloud infrastructure
```

costs resulting from app and pipeline executions, including the use of public apps, are the sole responsibility of you as a user. To avoid excessive costs, please read the app description carefully and set the app parameters and execution settings accordingly.***\n\n\n### Common Use Cases\n\n- The workflow consists of five steps: __HISAT2 Build__, __HISAT2__, __StringTie__, __StringTie Merge__ and __GffCompare__.\n- __Reads__ is the required input port that accepts raw sequencing reads in FASTQ or FASTA format. Could be also gzip'ed or bzip2'ed.\n- __Reference or Index files__ is also required and accepts a reference FASTA file(s) or a pre-built __HISAT2__ index TAR bundle.\n- __Reference annotation file__ is used for guiding the assembly process and for extracting exons and splice sites to create an index that will enable __HISAT2__ to be a more accurate aligner.\n- This workflow can be used to perform the quantification of RNA-Seq data of a unified set of genomic features across all samples in an analysis suitable for a downstream differential expression step. Once quantification values are obtained, downstream differential expression analysis can be performed using **DESeq2** or **Ballgown** tools on the Seven Bridges Platform.\n\n\n### Changes Introduced by Seven Bridges\n\n* In order to facilitate and accelerate further RNA-Seq analysis, an additional toolkit __Sambamba (0.7.1)__ is integrated into the same Seven Bridges tool with __HISAT2__. Besides the standard __HISAT2__ SAM output, __HISAT2__ has been extended to provide sorted BAM files as well. Thus, the sorting step as described in the protocol paper [1] is not visible in the workflow's main page.\n\n* An additional, Seven Bridges-designed tool, __SBG Pair FASTQs by Metadata__, is added at the beginning of the workflow in

```
order to enable the processing of multiple samples in a
single run.\n\n* The workflow's indexing step is optional.
__HISAT2 Build__ is implemented in a way that it can accept
a TAR bundle containing an already indexed reference
instead of reference FASTA file(s). If a TAR bundle is
provided, the indexing step is skipped and the TAR bundle
is forwarded to the output, thus reducing execution costs.
You can find pre-built index files (grch38\\_tran.tar.gz,
grch37\\_tran.tar.gz, hg38_tran.tar.gz) on the Seven
Bridges Platform under the Public Reference Files section.
Pre-built index files for organisms other than human can be
found
[here](https://daehwankimlab.github.io/hisat2/download/).\n\n*
In order to enable __StringTie__ to produce quantification
tables tailored for __DESeq2__ or
[edgeR](http://bioconductor.org/packages/release/bioc/html/edgeR.html),
the
[`prepDE.py`](https://ccb.jhu.edu/software/stringtie/dl/prepDE.py)
Python script [2] that extracts raw counts from
**Ballgown** input tables is embedded within the
__StringTie__ tool.\n\n### Common Issues and Important
Notes\n\n* For paired-end reads, the __Paired-end__
metadata field should be set appropriately in all FASTQ (or
FASTA) files that are found in the **Reads** input
node.\n\n* All input FASTQ files (or FASTA files depending
on the format of input reads) must have the __Sample ID__
metadata field appropriately set.\n\n* The GTF and
reference FASTA files need to have compatible chromosome
namings (i.e. >1, >2, ... or >chr1, >chr2, ...). If
pre-built HISAT2 index is used (reference FASTA file(s)
unknown), parameter __Remove 'chr' string__
(`--remove-chrname`) or __Add 'chr' string__
```

(`--add-chrname`) should be set to provide the same naming convention with GTF file.\n\n### Limitations\n\nThis workflow is optimised for eukaryotic genomes. When running with prokaryotic samples, whose genomes contain no introns and there are no splicing events, the index building step will fail. Please build the index separately with __HISAT2 Build__ default settings, without __Splice sites file__ (`--ss`) and __Exon file__ (`--exon`) options, and run the workflow with index file in TAR format instead of the genome reference.\n\n### Performance Benchmarking\n\nWorkflow is optimised to be run in scatter mode, so default instance is set to r5.8xlarge with 2 GB of EBS (AWS). The most intensive part in the workflow is reference indexing. For example, building hg38 human reference index using reference transcriptome requires 160 GB of RAM (for more details see the [HISAT2 homepage](https://daehwankimlab.github.io/hisat2/manual/)).\n\nIn the following table you can find estimates of the workflow runtime and its cost when the __Estimate novel isoform abundance?__ parameter is set to 'True'. Indexing is not performed, a TAR index archive (size 5.4 GB) is provided. Size of the gene annotation file is 1.2 GB. All tasks were executed on the on-demand default AWS instance.\n\nNote: Input size and number of reads refer to the average value per sample.\n\n| # of samples | Input format | Input size | Paired-end | # of reads | Read length | Duration | Cost |\n|--------------|--------------|------------|------------|------------|-------------|--------
400 | FASTQ.GZ | 287.5 MB | No | 11M | 36 | 1h 41min | $17.67 |\n| 100 | FASTQ.GZ | 300 MB | No | 11M | 36 | 36min | $5.64 |\n| 1 | FASTQ | 3.8 GB | Yes | 16M | 101 | 13min | $0.53 |\n| 50 | FASTQ.GZ | 3.9 GB | Yes | 60M | 75 | 3h 37min | $20.91 |\n| 200 | FASTQ.GZ | 4 GB | Yes | 60M | 75

```
| 8h 23min | $66.07 |\n| 16 | FASTQ | 21 GB | No | 79M | 75
| 2h 48min | $9.50 |\n| 16 | FASTQ | 35.9 GB | Yes | 160M |
50 | 4h 37min | $15.91 |\n| 1 | FASTQ | 37 GB | Yes | 163M
| 101 | 1h 29 min | $3.44 |\n| 10* | FASTQ | 62.5 GB | Yes
| 190M | 101 | 6h 40min | $21.68 |\n*EBS increased to 3 GB
due to large input size\n\nRuntime and task cost for one of
the samples above when indexing is performed (size of
reference file - 2.9 GB, size of GTF file - 1.2 GB):\n\n| #
of samples | Input format | Input size | Paired-end | # of
reads | Read length | Duration | Cost
|\n|--------------|--------------|------------|------------|------------|-------------|-------
1 | FASTQ | 3.8 GB | Yes | 16M | 101 | 1h 6min | $2.56
|\n\n*The cost of running __HISAT2-StringTie Workflow__ can
be significantly reduced by using **spot instances**. Visit
the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### API Python Implementation\nThe
app's draft task can also be submitted via the **API**. In
order to learn how to get your **Authentication token** and
**API endpoint** for the corresponding Platform visit our
[documentation](https://github.com/sbg/sevenbridges-python#authentication-and-configuration).\n
sevenbridges import Api\n\n# Enter api
credentials\nauthentication_token, api_endpoint =
\"enter_your_token\", \"enter_api_endpoint\"\napi =
Api(token=authentication_token, url=api_endpoint)\n\n# Get
project_id/workflow_id from your address bar. Example:
https://igor.sbgenomics.com/u/your_username/project/workflow\nproject_id,
workflow_id = \"your_username/project\",
\"your_username/project/workflow\"\n\n# Get file names from
files in your project. The file names below are just as an
example.\ninputs = {\n 'in_reads':
list(api.files.query(project=project_id,
```

```
names=['sample_pe1.fq', 'sample_pe2.fq'])),\n
'in_gene_annotation':
list(api.files.query(project=project_id,
names=['gtf_file.gtf'])),\n 'in_references_or_index':
list(api.files.query(project=project_id,
names=['reference_fasta_file.fa'])),\n
'discover_novel_isoform': True\n }\n\n# Run the task\ntask
= api.tasks.create(name='HISAT2-StringTie Workflow - API
Run', project=project_id, app=workflow_id, inputs=inputs,
run=True)\n```\nInstructions for installing and configuring
the API Python client, are provided on
[github](https://github.com/sbg/sevenbridges-python#installation).
For more information about using the API Python client,
consult [sevenbridges-python
documentation](http://sevenbridges-python.readthedocs.io/en/latest/).
**More examples** are available
[here](https://github.com/sbg/okAPI).\n\nAdditionally, [API
R](https://github.com/sbg/sevenbridges-r) and [API
Java](https://github.com/sbg/sevenbridges-java) clients are
available. To learn more about using these API clients
please refer to the [API R client
documentation](https://sbg.github.io/sevenbridges-r/), and
[API Java client
documentation](https://docs.sevenbridges.com/docs/java-library-quickstart).\n\n\n###
References\n\n[1] [HISAT, StringTie, Ballgown protocol
paper](https://www.nature.com/articles/nprot.2016.095)\n\n[2]
[StringTie manual page - using StringTie with DESeq2 and
edgeR](https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual#deseq)"
```

1.4 Extension Domain

```
{
  "fhir_extension": {
```

```
    "fhir_endpoint": "",
    "fhir_version": "",
    "fhir_resources": {}
  },
  "scm_extension": {
    "scm_repository": "",
    "scm_type": "git",
    "scm_commit": "",
    "scm_path": "",
    "scm_preview": ""
  }
}
```


1.5 Description Domain


```
{
"keywords": [],
"xref": [],
"platform": [
"Seven Bridges Platform"
],
"pipeline_steps": [
{
"step_number": "1",
"name": "sbg_file_selector",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
```

```
},
{
"step_number": "2",
"name": "sbg_file_selector_1",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "3",
"name": "sbg_file_selector_2",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "4",
"name": "stringtie_2_1_3",
"description": "**StringTie** is a fast and highly
efficient assembler of RNA-Seq alignments into potential
transcripts. It uses a novel network flow algorithm as well
as an optional *de novo* assembly step to assemble and
```

quantitate full-length transcripts representing multiple splice variants for each gene locus. Its input can include not only alignments of short reads that can also be used by other transcript assemblers, but also alignments of longer sequences that have been assembled from those reads. In order to identify differentially expressed genes between experiments, **StringTie**'s output can be processed by specialized software like **Ballgown**, **Cuffdiff** or other programs (**DESeq2**, **edgeR**, etc.) [1].\n\n*A list of __all inputs and parameters__ with corresponding descriptions can be found at the bottom of this page.*\n\n*__Please note that any cloud infrastructure costs resulting from app and pipeline executions, including the use of public apps, are the sole responsibility of you as a user. To avoid excessive costs, please read the app description carefully and set the app parameters and execution settings accordingly.__*\n\n### Common Use Cases\n\n* __StringTie__ can be used as a transcriptome assembler. It takes aligned reads as input (note that the provided BAM file has to be sorted by coordinate) and estimates expression level of genes and transcripts as it assembles them [3]. __Reference annotation file__ (`-G`) can be provided to guide the assembly process, though this is optional. If the ultimate goal is differential expression analysis assembled transcripts for each sample have to be merged into unified set of transcripts for all samples using __StringTie Merge__. After the merging step is done additional run of __StringTie__ is required to re-estimate merged transcripts for each sample using __Keep annotated transcripts only__ (`-e`) option that tells __StringTie__ to estimate expression levels only for reference transcripts and the __Create input files for

```
Ballgown and DESeq2__ (`-B`) option that enables creating
count-data input files for __Ballgown__ and __DESeq2__. For
more details refer to StringTie protocol paper [3].\n*
__StringTie__ can be used just for quantification. For this
purpose __Reference annotation file__ (`-G`) has to be
provided and the __Keep annotated transcripts only__ (`-e`)
parameter should be set to True, telling StringTie to only
estimate expression levels of genes and transcripts present
in the annotation file. This mode should be used in
experiments when there is a predefined list of genes or
transcripts of interest or in case of re-estimating merged
transcripts as described above.\n* __Text feature file__
input loads a list of point-features to guide the
transcriptome assembly. Accepted point features are
transcription start sites (TSS) and polyadenylation sites
(CPAS). Details on how this file should look like can be
found in the input's description info.\n* To output a TAB
file in which gene abundance estimation will be reported,
set the __Output gene abundance__ (`-A`) parameter to
True.\n* To output a file with all transcripts in the
provided reference file that are fully covered by the
reads, set the __Output covered reference transcripts__
(`-C`) parameter to True. This option requires **Reference
annotation file** (`-G`) to be provided.\n\n### Changes
Introduced by Seven Bridges\n\n* In order to enable
__StringTie__ to produce quantification tables tailored for
__DESeq2__ or
[EdgeR](http://bioconductor.org/packages/release/bioc/html/edgeR.html),
the
[`prepDE.py`](https://ccb.jhu.edu/software/stringtie/dl/prepDE.py)
python script [2] that extracts raw counts from
**Ballgown** input tables is embedded within the app.
```

```
Parameters of the scripts have default values except names
of gene and transcript count matrices. To obtain raw counts
suitable for __DESeq2__ set **Create input files for
Ballgown and DESeq2** (`-B`) to True. This will also
produce __Ballgown__ input tables as these tables are
needed for obtaining __DESeq2__ raw counts.\n* __Ballgown__
input tables (5 CTAB files) are outputted as an archive TAR
file containing all 5 files. This TAR file can be directly
fed to __Ballgown__ without any further modification given
the metadata field __Sample ID__ is properly set (check the
Common Issues and Important Notes section).\n\n### Common
Issues and Important Notes\n\n* If you want to perform
differential expression analysis using quantification
produced by __StringTie__, make sure that all input BAM
files on the **Aligned reads** input have metadata field
__Sample ID__ properly set. The value of __Sample ID__
metadata field will be used to match corresponding
expression (count) and phenotype data by downstream tools
for differential expression (__DESeq2__ and
__Ballgown__).\n* All BAM files on the **Aligned reads**
input need to be sorted by coordinates.\n\n### Performance
Benchmarking\n\nThe execution time for
quantification/assembly+quantification for human RNA-Seq
data (~ 160M of reads/12GB) takes somewhat less than 15
minutes on the default instance; the price is very low (~
0.05$). Unless specified otherwise, the default instance
used to run the __StringTie__ tool will be c4.xlarge
(AWS).\n\n*Cost can be significantly reduced by using
**spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie home
page](https://ccb.jhu.edu/software/stringtie/index.shtml)\n\n[2]
```

[StringTie manual page - using StringTie with DESeq2 and
EdgeR](https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual#deseq)\n\n[3]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "5",
"name": "stringtie_2_1_4",
"description": "**StringTie** is a fast and highly
efficient assembler of RNA-Seq alignments into potential
transcripts. It uses a novel network flow algorithm as well
as an optional *de novo* assembly step to assemble and
quantitate full-length transcripts representing multiple
splice variants for each gene locus. Its input can include
not only alignments of short reads that can also be used by
other transcript assemblers, but also alignments of longer
sequences that have been assembled from those reads. In
order to identify differentially expressed genes between
experiments, **StringTie**'s output can be processed by
specialized software like **Ballgown**, **Cuffdiff** or
other programs (**DESeq2**, **edgeR**, etc.) [1].\n\n*A
list of __all inputs and parameters__ with corresponding
descriptions can be found at the bottom of this
page.*\n\n*__Please note that any cloud infrastructure
costs resulting from app and pipeline executions, including
the use of public apps, are the sole responsibility of you
as a user. To avoid excessive costs, please read the app
description carefully and set the app parameters and

execution settings accordingly.__*\n\n### Common Use Cases\n\n* __StringTie__ can be used as a transcriptome assembler. It takes aligned reads as input (note that the provided BAM file has to be sorted by coordinate) and estimates expression level of genes and transcripts as it assembles them [3]. __Reference annotation file__ (`-G`) can be provided to guide the assembly process, though this is optional. If the ultimate goal is differential expression analysis assembled transcripts for each sample have to be merged into unified set of transcripts for all samples using __StringTie Merge__. After the merging step is done additional run of __StringTie__ is required to re-estimate merged transcripts for each sample using __Keep annotated transcripts only__ (`-e`) option that tells __StringTie__ to estimate expression levels only for reference transcripts and the __Create input files for Ballgown and DESeq2__ (`-B`) option that enables creating count-data input files for __Ballgown__ and __DESeq2__. For more details refer to StringTie protocol paper [3].\n* __StringTie__ can be used just for quantification. For this purpose __Reference annotation file__ (`-G`) has to be provided and the __Keep annotated transcripts only__ (`-e`) parameter should be set to True, telling StringTie to only estimate expression levels of genes and transcripts present in the annotation file. This mode should be used in experiments when there is a predefined list of genes or transcripts of interest or in case of re-estimating merged transcripts as described above.\n* __Text feature file__ input loads a list of point-features to guide the transcriptome assembly. Accepted point features are transcription start sites (TSS) and polyadenylation sites (CPAS). Details on how this file should look like can be

```
found in the input's description info.\n* To output a TAB
file in which gene abundance estimation will be reported,
set the __Output gene abundance__ (`-A`) parameter to
True.\n* To output a file with all transcripts in the
provided reference file that are fully covered by the
reads, set the __Output covered reference transcripts__
(`-C`) parameter to True. This option requires **Reference
annotation file** (`-G`) to be provided.\n\n### Changes
Introduced by Seven Bridges\n\n* In order to enable
__StringTie__ to produce quantification tables tailored for
__DESeq2__ or
[EdgeR](http://bioconductor.org/packages/release/bioc/html/edgeR.html),
the
[`prepDE.py`](https://ccb.jhu.edu/software/stringtie/dl/prepDE.py)
python script [2] that extracts raw counts from
**Ballgown** input tables is embedded within the app.
Parameters of the scripts have default values except names
of gene and transcript count matrices. To obtain raw counts
suitable for __DESeq2__ set **Create input files for
Ballgown and DESeq2** (`-B`) to True. This will also
produce __Ballgown__ input tables as these tables are
needed for obtaining __DESeq2__ raw counts.\n* __Ballgown__
input tables (5 CTAB files) are outputted as an archive TAR
file containing all 5 files. This TAR file can be directly
fed to __Ballgown__ without any further modification given
the metadata field __Sample ID__ is properly set (check the
Common Issues and Important Notes section).\n\n### Common
Issues and Important Notes\n\n* If you want to perform
differential expression analysis using quantification
produced by __StringTie__, make sure that all input BAM
files on the **Aligned reads** input have metadata field
__Sample ID__ properly set. The value of __Sample ID__
```

metadata field will be used to match corresponding
expression (count) and phenotype data by downstream tools
for differential expression (__DESeq2__ and
__Ballgown__).\n* All BAM files on the **Aligned reads**
input need to be sorted by coordinates.\n\n### Performance
Benchmarking\n\nThe execution time for
quantification/assembly+quantification for human RNA-Seq
data (~ 160M of reads/12GB) takes somewhat less than 15
minutes on the default instance; the price is very low (~
0.05$). Unless specified otherwise, the default instance
used to run the __StringTie__ tool will be c4.xlarge
(AWS).\n\n*Cost can be significantly reduced by using
spot instances. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie home
page](https://ccb.jhu.edu/software/stringtie/index.shtml)\n\n[2]
[StringTie manual page - using StringTie with DESeq2 and
EdgeR](https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual#deseq)\n\n[3]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "6",
"name": "stringtie_merge_2_1_3",
"description": "The **StringTie Merge** tool takes a list
of GTF/GFF files as its input and merges/assembles these
transcripts into a non-redundant set of transcripts [1].
This tool is used as an intermediate step in the new Tuxedo

```
differential expression analysis pipeline described in [2]
to generate a global, unified set of transcripts (isoforms)
across multiple RNA-Seq samples.\n\n*A list of __all inputs
and parameters__ with corresponding descriptions can be
found at the bottom of this page.*\n\n*__Please note that
any cloud infrastructure costs resulting from app and
pipeline executions, including the use of public apps, are
the sole responsibility of you as a user. To avoid
excessive costs, please read the app description carefully
and set the app parameters and execution settings
accordingly.__*\n\n### Common Use Cases\n\n* This tool
should be used after __StringTie__ transcript assembling of
each sample in the experiment. This tool creates a set of
transcripts that is consistent across all samples, so that
the transcripts can be compared in subsequent steps of the
analysis [2]. For more details refer to the StringTie
protocol paper [2].\n\n### Changes Introduced by Seven
Bridges\n\n* No modifications to the original tool
representation have been made.\n\n### Common Issues and
Important Notes\n\n* No common issues specific to the
tool's execution on the Seven Bridges Platform have been
detected. \n\n### Performance Benchmarking\n\n* The
execution time for merging 8 assembled transcripts takes
several minutes on the default instance; the price is
negligible (~ 0.01$). Unless specified otherwise, the
default instance used to run the __StringTie Merge__ tool
will be c4.large (AWS).\n\n*Cost can be significantly
reduced by using **spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie
manual
page](http://ccb.jhu.edu/software/stringtie/index.shtml?t=manual)\n\n[2]
```

[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "7",
"name": "gffcompare_0_11_6",
"description": "__GffCompare__ [1] is a part of the GFF
utility toolkit and can be used to:\n\n- compare and
evaluate the accuracy of RNA-Seq transcript assemblers
(__Cufflinks__, __Stringtie__).\n- collapse (merge)
duplicate transcripts from multiple GTF/GFF3 files (e.g.
resulted from assembly of different samples)\n- classify
transcripts from one or multiple GTF/GFF3 files as they
relate to reference transcripts provided in an annotation
file (also in GTF/GFF3 format)\n\nThe original form of this
program is also distributed as part of the Cufflinks suite,
under the name \"__CuffCompare__\". Most of the options and
parameters of __CuffCompare__ are supported by
__GffCompare__, while new features will likely be added to
__GffCompare__ in the future.\n\n*A list of __all inputs
and parameters__ with corresponding descriptions can be
found at the bottom of this page.*\n\n*__Please note that
any cloud infrastructure costs resulting from app and
pipeline executions, including the use of public apps, are
the sole responsibility of you as a user. To avoid
excessive costs, please read the app description carefully
and set the app parameters and execution settings
accordingly.__*\n\n### Common Use Cases\n\n* This tool

should be used after __StringTie__ and __StringTie Merge__
transcript assembling and merging. The GffCompare program
then compares the genes and transcripts with the annotation
and reports statistics on this comparison. For more details
refer to the StringTie protocol paper [2].\n\n### Changes
Introduced by Seven Bridges\n\n* No modifications to the
original tool representation have been made.\n\n### Common
Issues and Important Notes\n\n* No common issues specific
to the tool's execution on the Seven Bridges Platform have
been detected.\n\n### Performance Benchmarking\n\n* The
execution time for comparing 8 assembled transcripts takes
several minutes on the default instance; the price is
negligible (~ 0.01$). Unless specified otherwise, the
default instance used to run the __GffCompare__ tool will
be c4.large (AWS).\n\n*Cost can be significantly reduced by
using **spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [GffCompare
manual
page](http://ccb.jhu.edu/software/stringtie/gff.shtml#gffcompare)\n\n[2]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "0.11.6",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "8",
"name": "hisat2_extractsplicesites_2_2_1",
"description": "__HISAT2 ExtractSpliceSites__ extracts a
list of splice sites from a GTF file in the **HISAT2**'s

own format as follows (4 tab-delimited columns): \n(1)
chromosome name; (2) zero-offset based genomic position of
the flanking base on the left side of an intron; (3)
zero-offset based genomic position of the flanking base on
the right; (4) strand.\n\n*A list of **all inputs and
parameters** with corresponding descriptions can be found
at the end of the page.*\n\n### Common Use Cases\n\n* This
tool can be used as a preprocessing step for __HISAT2
Build__. It is used to create a file containing splice
sites that can be forwarded to __HISAT2 Build__ in order to
create an index that can improve alignment accuracy.\n\n###
Changes Introduced by Seven Bridges\n\n* Output file will
be prefixed by the input gene annotation filename, unless
the __Output file prefix__ option is explicitly
specified.\n\n### Common Issues and Important Notes\n\n* No
common issues specific to the tool's execution on the Seven
Bridges platform have been detected. \n\n### Performance
Benchmarking\n\nThe execution time for human gene
annotation takes several minutes on the default instance;
the price is negligible (~ $0.02) using on-demand AWS
instances.\n\n*Cost can be significantly reduced by **spot
instance** usage. Visit [knowledge
center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "9",
"name": "hisat2_extractexons_2_2_1",

"description": "__HISAT2 ExtractExons__ extracts a list of exons from a GTF file in the **HISAT2**'s own format which is a tab-delimited file with the following columns: (1) chromosome name; (2) zero-offset based left genomic position of an exon; and (3) zero-offset based right genomic position of an exon.\n\n*A list of **all inputs and parameters** with corresponding descriptions can be found at the end of the page.*\n\n### Common Use Cases\n\n* This tool can be used as a preprocessing step for __HISAT2 Build__. It is used to create a file containing exons that can be further fed to __HISAT2 Build__ in order to create an index that can improve alignment accuracy.\n\n### Changes Introduced by Seven Bridges\n\n* Output file will be prefixed by the input gene annotation filename, unless the __Output file prefix__ option is explicitly specified.\n\n### Common Issues and Important Notes\n\n* No common issues specific to the tool's execution on the Seven Bridges platform have been detected.\n\n### Performance Benchmarking\n\nThe execution time for human gene annotation takes several minutes on the default instance; the price is negligible (~ $0.02) using on-demand AWS instances. \n\n*Cost can be significantly reduced by **spot instance** usage. Visit [knowledge center](https://docs.sevenbridges.com/docs/about-spot-instances) for more details.*",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "10",

"name": "hisat2_build_2_2_1",
"description": "__HISAT2 Build__ tool builds a HISAT2 index necessary for the __HISAT2__ alignment method. To create an index, __HISAT2 Build__ uses a genome reference file(s) in FASTA format and outputs a set of 8 files with suffixes .1.ht2, .2.ht2, .3.ht2, .4.ht2, .5.ht2, .6.ht2, .7.ht2, and .8.ht2. In the case of a large index these suffixes will have an ht2l termination. These files together constitute the index: they are all that is needed to align reads to that reference [1].\n\nThe __HISAT2__ indexing scheme is called Hierarchical Graph Ferragina Manzini (HGFM) index. In addition to using one global GFM index that represents the general population, __HISAT2__ uses a large set of small GFM indexes that collectively cover the whole genome (each index representing a genomic region of 56 Kbp, with 55,000 indexes needed to cover the human population). These small indexes (called local indexes) combined with several alignment strategies enable effective alignment of sequencing reads [1].\n\n*A list of **all inputs and parameters** with corresponding descriptions can be found at the bottom of this page.*\n\n***Please note that any cloud infrastructure costs resulting from app and pipeline executions, including the use of public apps, are the sole responsibility of you as a user. To avoid excessive costs, please read the app description carefully and set the app parameters and execution settings accordingly.***\n\n### Common Use Cases\n\n* This tool is used to create an index for the __HISAT2__ splice aware aligner.\n* __HISAT2 Build__ can take advantage of known splice sites, exons, SNPs and haplotypes to create an index that will enable the __HISAT2__ aligner to create more accurate alignments. These files can be created using __HISAT2 ExtractExons__,

```
__HISAT2 ExtractSpliceSites__ and __HISAT2 Extract
SNPsHaplotypes__. Pre-built index files using these options
can be also found on [HISAT2 home
page](https://daehwankimlab.github.io/hisat2/download/).\n*
__HISAT2 Build__ can generate either small or large
indexes. The wrapper will decide which one to generate
based on the length of the input genome. If the reference
does not exceed 4 billion characters but a large index is
preferred, the user can specify `--large-index` to force
__HISAT2 Build__ to build a large index instead [1].\n\n###
Changes Introduced by Seven Bridges\n\n* The directory
containing the index files will be outputted as a TAR
bundle (the __Index files__ output). This bundle can then
be provided to the __HISAT2__ aligner, which will
automatically take care of untarring it and preparing it to
run successfully without further issues.\n* __HISAT2
Build__ can accept a TAR bundle containing an already
indexed reference instead of a reference FASTA file(s), to
skip indexing and reduce processing time if this tool is a
part of a workflow.\n* Output file will be prefixed by the
input reference file name, unless the **Output file
prefix** option is explicitly specified.\n\n### Common
Issues and Important Notes\n\n* __Exon__ (`--exon`) and
__Splice sites__ (`--ss`) files have to be used together.
If exactly one of these options is used, __HISAT2 Build__
will fail. Input files for these two options can be
generated from __HISAT2 ExtractExons__ and __HISAT2
ExtractSpliceSites__ tools.\n* The **Max bucket size**
(`--bmax`), **Max bucket size (as divisior)**
(`--bmaxdivn`), and **Diff-cover period** (`--dcv`) options
are governing the trade off between running time and memory
usage and should only be set by advanced users.  __HISAT2
```

```
Build__ automatically sets these parameters to their
optimal values, that yield the best running time without
exhausting memory. This behavior can be disabled using
the -a/--noauto option [1].\n\n### Performance
Benchmarking\n\nIn the following table you can find
estimates of __HISAT2 Build__ duration and cost. Two
different instances were used for benchmarking because
__HISAT2 Build__ requires significantly more memory (~
200GB for human reference [1]) when using some of the
following options: __Splice sites file__ (`--ss`), __Exon
file__ (`--exon`) and/or __SNP file__ (`--snp`),
__Haplotype file__ (`--haplotype`) opposed to situations
where these options are not used. The Seven Bridges version
of the tool will dynamically choose an appropriate instance
based on the provided inputs. The results shown here are
obtained by indexing the human reference genome.  Execution
time and cost can vary for other genomes.\n\n*Cost can be
significantly reduced by **spot instances** usage. Visit
the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n| Settings | Duration | Cost |
Instance
(AWS)|\n|------------------------------|-------------|-----------------|--------------|\n|
Default (no additional files) | 18min.  | $0.25 |
c5.4xlarge |\n| Using `--ss` and `--exon` options | 51min.
| $1.84 | r5.8xlarge |\n| Using `--ss`, `--exon` ,`--snp`
and `--haplotype` options | 1h 12min.  | $2.60 | r5.8xlarge
|\n\n### References\n\n[1] [HISAT2 manual
page](https://daehwankimlab.github.io/hisat2/manual/)",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
```

"output_list": []
},
{
"step_number": "11",
"name": "hisat2_2_2_1",
"description": "**HISAT2** is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes as well as to a single reference genome [1]. \n\n__HISAT2__ (hierarchical indexing for spliced alignment of transcripts 2) aligns both DNA and RNA sequences using a graph Ferragina Manzini (GFM) index. In addition to using one global GFM index that represents the general population, __HISAT2__ uses a large set of small GFM indexes that collectively cover the whole genome (each index representing a genomic region of 56 Kbp, with 55,000 indexes needed to cover the human population). These small indexes (called local indexes) combined with several alignment strategies enable effective alignment of sequencing reads. This new indexing scheme is called Hierarchical Graph FM (HGFM) index [1].\n\n*A list of **all inputs and parameters** with corresponding descriptions can be found at the bottom of this page.*\n\n***Please note that any cloud infrastructure costs resulting from app and pipeline executions, including the use of public apps, are the sole responsibility of you as a user. To avoid excessive costs, please read the app description carefully and set the app parameters and execution settings accordingly.***\n\n### Common Use Cases\n\n- Though **HISAT2** can be used for all types of next-generation sequencing reads, it is mostly used for aligning RNA-Seq data.\n- The __Downstream transcriptome assembly__

(`--downstream-transcriptome-assembly/--dta`) parameter tells __HISAT2__ to report alignments tailored for transcript assemblers including **StringTie**. With this option, __HISAT2__ requires longer anchor lengths for de novo discovery of splice sites. This leads to fewer alignments with short-anchors, which helps transcript assemblers significantly improve computation and memory usage.\n- The __Downstream transcriptome assembly__ - __Cufflinks__ (`--dta-cufflinks`) parameter tells __HISAT2__ to report alignments tailored specifically for **Cufflinks**. In addition to what __HISAT2__ does with the above option (__Downstream transcriptome assembly__), it also looks for novel splice sites with three signals (GT/AG, GC/AG, AT/AC). Nonetheless, all user-provided splice sites are used irrespective of their signals. __HISAT2__ produces an optional field, `XS:A:[+-]`, for every spliced alignment.\n\n\n### Changes Introduced by Seven Bridges\n\n* In order to facilitate and accelerate further RNA-Seq analysis, an additional toolkit __Sambamba (0.7.1)__ is integrated into the same Seven Bridges tool representation alongside __HISAT2 (2.2.1)__. Besides the standard __HISAT2__ SAM output, HISAT2 has been extended to provide two additional output file options: \n * an **unsorted BAM** file created by piping standard __HISAT2__ output to __Sambamba view__ and \n * a **coordinate-sorted BAM** file and its Index (BAI) file created by additional piping of the output through __Sambamba sort__ and __Sambamba index__. \n To select the desired output, use the __Output type__ parameter. The default output is a **coordinate-sorted BAM** file.\n* If the __Read group ID__ parameter is not defined, by default it will be set to ‘1’, unless the __No @RG line__ option is explicitly specified.

```
If the tool is scattered within a workflow, it will assign
the __Read group ID__ according to the order of the
scattered folders. This also ensures a unique __Read group
ID__ when processing multi-read group input data from one
sample.\n* All output files will be prefixed by the input
sample ID (inferred from the __Sample ID__ metadata field
if existent, or from filename otherwise), unless the
__Output prefix__ option is explicitly specified.\n\n###
Common Issues and Important Notes\n\n- To run __HISAT2,__
properly indexed reference files are required. These files
can be either created using the __HISAT2 Build__ tool or
downloaded from the [HISAT2 home
page](https://daehwankimlab.github.io/hisat2/download/).\n-
Some __HISAT2__ options (i.e. __Maximum number of ambiguous
characters__ (`--n-ceil`) or __Long introns with canonical
splice sites penalty__ (`--pen-canintronlen`)) specify a
function rather than an individual number or setting. In
these cases, the user specifies three parameters: (a) a
function type F, (b) a constant term B, and (c) a
coefficient A. The available function types are constant
(C), linear (L), square-root (S), and natural log (G). The
parameters are specified as F,B,A - that is, the function
type, the constant term, and the coefficient are separated
by commas with no whitespace. The constant term and
coefficient may be negative and/or floating-point numbers
[1].\n\n         
Example 1: if the function specification is L,-0.4,-0.6,
then the function defined is:\n`f(x) = -0.4 + -0.6 *
x`\n\n         
Example 2: if the function specification is G,1,5.4, then
the function defined is:\n`f(x) = 1.0 + 5.4 * ln(x)`\n\n-
In a single run, __Reads__ should be either paired or
```

```
unpaired. __Reads__ may be a mix of different lengths.\n-
For paired-end __Reads__, the __Paired-end__ metadata field
has to be set to 1 or 2. Paired-end __Reads__ must
correspond to each other file-for-file and
read-for-read.\n\n### Performance Benchmarking\n\nFor the
human reference genome, __HISAT2__ requires about 9 GB of
RAM to run properly. In the following table you can find
estimates of __HISAT2__ running time and cost. All samples
are aligned onto the hg38 human reference index built using
reference transcriptome (i.e. using __Splice sites__
(`--ss`) and __Exon__ (`--exon`) options). \n\n*The cost of
running __HISAT2__ can be significantly reduced by using
**spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n| Experiment type | Input size |
Paired-end | # of reads | Read length | Duration | Cost |
Instance
(AWS)|\n|----------------------|-----------------|------------|-----------------|------------
RNA-Seq | 2 x 21.5 GB | Yes | 95M | 101 | 1h 50min.  |
$0.99 | c4.2xlarge |\n| RNA-Seq | 2 x 10.8 GB | Yes | 47.5M
| 101 | 57min.  | $0.51 | c4.2xlarge |\n| RNA-Seq | 2 x 2.2
GB | Yes | 9.5M | 101 | 16min.  | $0.14 | c4.2xlarge |\n|
RNA-Seq | 3.9 GB | No | 17.6M | 51 | 12min.  | $0.11 |
c4.2xlarge |\n\n### References\n\n[1] [HISAT2 manual
page](https://daehwankimlab.github.io/hisat2/manual/)",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "12",
```

```
"name": "sbg_pair_fastqs_by_metadata",
"description": "Tool accepts list of FASTQ files groups
them into separate lists. This grouping is done using
metadata values and their hierarchy (Sample ID > Library ID
> Platform unit ID > File segment number) which should
create unique combinations for each pair of FASTQ files.
Important metadata fields are Sample ID, Library ID,
Platform unit ID and File segment number. 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. Files with no paired end metadata are
grouped in the same way as the ones with paired end
metadata, generally they should be alone in a separate
list. Files with no metadata set will be grouped together.
\n\nIf there are more than two files in a group, this might
create errors further down most pipelines and the user
should check if the metadata fields for those files are set
properly.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
}
]
}
```


1.6 Execution Domain


```
{
"keywords": [],
"xref": [],
"platform": [
"Seven Bridges Platform"
```

```
],
"pipeline_steps": [
{
"step_number": "1",
"name": "sbg_file_selector",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "2",
"name": "sbg_file_selector_1",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "3",
"name": "sbg_file_selector_2",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
```

to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "4",
"name": "stringtie_2_1_3",
"description": "**StringTie** is a fast and highly efficient assembler of RNA-Seq alignments into potential transcripts. It uses a novel network flow algorithm as well as an optional *de novo* assembly step to assemble and quantitate full-length transcripts representing multiple splice variants for each gene locus. Its input can include not only alignments of short reads that can also be used by other transcript assemblers, but also alignments of longer sequences that have been assembled from those reads. In order to identify differentially expressed genes between experiments, **StringTie**'s output can be processed by specialized software like **Ballgown**, **Cuffdiff** or other programs (**DESeq2**, **edgeR**, etc.) [1].\n\n*A list of __all inputs and parameters__ with corresponding descriptions can be found at the bottom of this page.*\n\n*__Please note that any cloud infrastructure costs resulting from app and pipeline executions, including the use of public apps, are the sole responsibility of you as a user. To avoid excessive costs, please read the app description carefully and set the app parameters and execution settings accordingly.__*\n\n### Common Use Cases\n\n* __StringTie__ can be used as a transcriptome assembler. It takes aligned reads as input (note that the

provided BAM file has to be sorted by coordinate) and estimates expression level of genes and transcripts as it assembles them [3]. __Reference annotation file__ (`-G`) can be provided to guide the assembly process, though this is optional. If the ultimate goal is differential expression analysis assembled transcripts for each sample have to be merged into unified set of transcripts for all samples using __StringTie Merge__. After the merging step is done additional run of __StringTie__ is required to re-estimate merged transcripts for each sample using __Keep annotated transcripts only__ (`-e`) option that tells __StringTie__ to estimate expression levels only for reference transcripts and the __Create input files for Ballgown and DESeq2__ (`-B`) option that enables creating count-data input files for __Ballgown__ and __DESeq2__. For more details refer to StringTie protocol paper [3].\n* __StringTie__ can be used just for quantification. For this purpose __Reference annotation file__ (`-G`) has to be provided and the __Keep annotated transcripts only__ (`-e`) parameter should be set to True, telling StringTie to only estimate expression levels of genes and transcripts present in the annotation file. This mode should be used in experiments when there is a predefined list of genes or transcripts of interest or in case of re-estimating merged transcripts as described above.\n* __Text feature file__ input loads a list of point-features to guide the transcriptome assembly. Accepted point features are transcription start sites (TSS) and polyadenylation sites (CPAS). Details on how this file should look like can be found in the input's description info.\n* To output a TAB file in which gene abundance estimation will be reported, set the __Output gene abundance__ (`-A`) parameter to

```
True.\n* To output a file with all transcripts in the
provided reference file that are fully covered by the
reads, set the __Output covered reference transcripts__
(`-C`) parameter to True. This option requires **Reference
annotation file** (`-G`) to be provided.\n\n### Changes
Introduced by Seven Bridges\n\n* In order to enable
__StringTie__ to produce quantification tables tailored for
__DESeq2__ or
[EdgeR](http://bioconductor.org/packages/release/bioc/html/edgeR.html),
the
[`prepDE.py`](https://ccb.jhu.edu/software/stringtie/dl/prepDE.py)
python script [2] that extracts raw counts from
**Ballgown** input tables is embedded within the app.
Parameters of the scripts have default values except names
of gene and transcript count matrices. To obtain raw counts
suitable for __DESeq2__ set **Create input files for
Ballgown and DESeq2** (`-B`) to True. This will also
produce __Ballgown__ input tables as these tables are
needed for obtaining __DESeq2__ raw counts.\n* __Ballgown__
input tables (5 CTAB files) are outputted as an archive TAR
file containing all 5 files. This TAR file can be directly
fed to __Ballgown__ without any further modification given
the metadata field __Sample ID__ is properly set (check the
Common Issues and Important Notes section).\n\n### Common
Issues and Important Notes\n\n* If you want to perform
differential expression analysis using quantification
produced by __StringTie__, make sure that all input BAM
files on the **Aligned reads** input have metadata field
__Sample ID__ properly set. The value of __Sample ID__
metadata field will be used to match corresponding
expression (count) and phenotype data by downstream tools
for differential expression (__DESeq2__ and
```

__Ballgown__).\n* All BAM files on the **Aligned reads**
input need to be sorted by coordinates.\n\n### Performance
Benchmarking\n\nThe execution time for
quantification/assembly+quantification for human RNA-Seq
data (~ 160M of reads/12GB) takes somewhat less than 15
minutes on the default instance; the price is very low (~
0.05$). Unless specified otherwise, the default instance
used to run the __StringTie__ tool will be c4.xlarge
(AWS).\n\n*Cost can be significantly reduced by using
spot instances. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie home
page](https://ccb.jhu.edu/software/stringtie/index.shtml)\n\n[2]
[StringTie manual page - using StringTie with DESeq2 and
EdgeR](https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual#deseq)\n\n[3]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "5",
"name": "stringtie_2_1_4",
"description": "**StringTie** is a fast and highly
efficient assembler of RNA-Seq alignments into potential
transcripts. It uses a novel network flow algorithm as well
as an optional *de novo* assembly step to assemble and
quantitate full-length transcripts representing multiple
splice variants for each gene locus. Its input can include
not only alignments of short reads that can also be used by

```
other transcript assemblers, but also alignments of longer
sequences that have been assembled from those reads.  In
order to identify differentially expressed genes between
experiments, **StringTie**'s output can be processed by
specialized software like **Ballgown**, **Cuffdiff** or
other programs (**DESeq2**, **edgeR**, etc.) [1].\n\n*A
list of __all inputs and parameters__ with corresponding
descriptions can be found at the bottom of this
page.*\n\n*__Please note that any cloud infrastructure
costs resulting from app and pipeline executions, including
the use of public apps, are the sole responsibility of you
as a user. To avoid excessive costs, please read the app
description carefully and set the app parameters and
execution settings accordingly.__*\n\n### Common Use
Cases\n\n* __StringTie__ can be used as a transcriptome
assembler. It takes aligned reads as input (note that the
provided BAM file has to be sorted by coordinate) and
estimates expression level of genes and transcripts as it
assembles them [3]. __Reference annotation file__ (`-G`)
can be provided to guide the assembly process, though this
is optional. If the ultimate goal is differential
expression analysis assembled transcripts for each sample
have to be merged into unified set of transcripts for all
samples using __StringTie Merge__. After the merging step
is done additional run of __StringTie__ is required to
re-estimate merged transcripts for each sample using __Keep
annotated transcripts only__ (`-e`) option that tells
__StringTie__ to estimate expression levels only for
reference transcripts and the __Create input files for
Ballgown and DESeq2__ (`-B`) option that enables creating
count-data input files for __Ballgown__ and __DESeq2__. For
more details refer to StringTie protocol paper [3].\n*
```

```
__StringTie__ can be used just for quantification. For this
purpose __Reference annotation file__ (`-G`) has to be
provided and the __Keep annotated transcripts only__ (`-e`)
parameter should be set to True, telling StringTie to only
estimate expression levels of genes and transcripts present
in the annotation file. This mode should be used in
experiments when there is a predefined list of genes or
transcripts of interest or in case of re-estimating merged
transcripts as described above.\n* __Text feature file__
input loads a list of point-features to guide the
transcriptome assembly. Accepted point features are
transcription start sites (TSS) and polyadenylation sites
(CPAS). Details on how this file should look like can be
found in the input's description info.\n* To output a TAB
file in which gene abundance estimation will be reported,
set the __Output gene abundance__ (`-A`) parameter to
True.\n* To output a file with all transcripts in the
provided reference file that are fully covered by the
reads, set the __Output covered reference transcripts__
(`-C`) parameter to True. This option requires **Reference
annotation file** (`-G`) to be provided.\n\n### Changes
Introduced by Seven Bridges\n\n* In order to enable
__StringTie__ to produce quantification tables tailored for
__DESeq2__ or
[EdgeR](http://bioconductor.org/packages/release/bioc/html/edgeR.html),
the
[`prepDE.py`](https://ccb.jhu.edu/software/stringtie/dl/prepDE.py)
python script [2] that extracts raw counts from
**Ballgown** input tables is embedded within the app.
Parameters of the scripts have default values except names
of gene and transcript count matrices. To obtain raw counts
suitable for __DESeq2__ set **Create input files for
```

```
Ballgown and DESeq2** (`-B`) to True. This will also
produce __Ballgown__ input tables as these tables are
needed for obtaining __DESeq2__ raw counts.\n* __Ballgown__
input tables (5 CTAB files) are outputted as an archive TAR
file containing all 5 files. This TAR file can be directly
fed to __Ballgown__ without any further modification given
the metadata field __Sample ID__ is properly set (check the
Common Issues and Important Notes section).\n\n### Common
Issues and Important Notes\n\n* If you want to perform
differential expression analysis using quantification
produced by __StringTie__, make sure that all input BAM
files on the **Aligned reads** input have metadata field
__Sample ID__ properly set. The value of __Sample ID__
metadata field will be used to match corresponding
expression (count) and phenotype data by downstream tools
for differential expression (__DESeq2__ and
__Ballgown__).\n* All BAM files on the **Aligned reads**
input need to be sorted by coordinates.\n\n### Performance
Benchmarking\n\nThe execution time for
quantification/assembly+quantification for human RNA-Seq
data (~ 160M of reads/12GB) takes somewhat less than 15
minutes on the default instance; the price is very low (~
0.05$). Unless specified otherwise, the default instance
used to run the __StringTie__ tool will be c4.xlarge
(AWS).\n\n*Cost can be significantly reduced by using
**spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie home
page](https://ccb.jhu.edu/software/stringtie/index.shtml)\n\n[2]
[StringTie manual page - using StringTie with DESeq2 and
EdgeR](https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual#deseq)\n\n[3]
[HISAT, StringTie, Ballgown protocol
```

paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "6",
"name": "stringtie_merge_2_1_3",
"description": "The **StringTie Merge** tool takes a list
of GTF/GFF files as its input and merges/assembles these
transcripts into a non-redundant set of transcripts [1].
This tool is used as an intermediate step in the new Tuxedo
differential expression analysis pipeline described in [2]
to generate a global, unified set of transcripts (isoforms)
across multiple RNA-Seq samples.\n\n*A list of __all inputs
and parameters__ with corresponding descriptions can be
found at the bottom of this page.*\n\n*__Please note that
any cloud infrastructure costs resulting from app and
pipeline executions, including the use of public apps, are
the sole responsibility of you as a user. To avoid
excessive costs, please read the app description carefully
and set the app parameters and execution settings
accordingly.__*\n\n### Common Use Cases\n\n* This tool
should be used after __StringTie__ transcript assembling of
each sample in the experiment. This tool creates a set of
transcripts that is consistent across all samples, so that
the transcripts can be compared in subsequent steps of the
analysis [2]. For more details refer to the StringTie
protocol paper [2].\n\n### Changes Introduced by Seven
Bridges\n\n* No modifications to the original tool
representation have been made.\n\n### Common Issues and

Important Notes\n\n* No common issues specific to the tool's execution on the Seven Bridges Platform have been detected. \n\n### Performance Benchmarking\n\n* The execution time for merging 8 assembled transcripts takes several minutes on the default instance; the price is negligible (~ 0.01$). Unless specified otherwise, the default instance used to run the __StringTie Merge__ tool will be c4.large (AWS).\n\n*Cost can be significantly reduced by using **spot instances**. Visit the [Knowledge Center](https://docs.sevenbridges.com/docs/about-spot-instances) for more details.*\n\n### References\n\n[1] [StringTie manual page](http://ccb.jhu.edu/software/stringtie/index.shtml?t=manual)\n\n[2] [HISAT, StringTie, Ballgown protocol paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "7",
"name": "gffcompare_0_11_6",
"description": "__GffCompare__ [1] is a part of the GFF utility toolkit and can be used to:\n\n- compare and evaluate the accuracy of RNA-Seq transcript assemblers (__Cufflinks__, __Stringtie__).\n- collapse (merge) duplicate transcripts from multiple GTF/GFF3 files (e.g. resulted from assembly of different samples)\n- classify transcripts from one or multiple GTF/GFF3 files as they relate to reference transcripts provided in an annotation file (also in GTF/GFF3 format)\n\nThe original form of this

```
program is also distributed as part of the Cufflinks suite,
under the name \"__CuffCompare__\". Most of the options and
parameters of __CuffCompare__ are supported by
__GffCompare__, while new features will likely be added to
__GffCompare__ in the future.\n\n*A list of __all inputs
and parameters__ with corresponding descriptions can be
found at the bottom of this page.*\n\n*__Please note that
any cloud infrastructure costs resulting from app and
pipeline executions, including the use of public apps, are
the sole responsibility of you as a user. To avoid
excessive costs, please read the app description carefully
and set the app parameters and execution settings
accordingly.__*\n\n### Common Use Cases\n\n* This tool
should be used after __StringTie__ and __StringTie Merge__
transcript assembling and merging. The GffCompare program
then compares the genes and transcripts with the annotation
and reports statistics on this comparison. For more details
refer to the StringTie protocol paper [2].\n\n### Changes
Introduced by Seven Bridges\n\n* No modifications to the
original tool representation have been made.\n\n### Common
Issues and Important Notes\n\n* No common issues specific
to the tool's execution on the Seven Bridges Platform have
been detected.\n\n### Performance Benchmarking\n\n* The
execution time for comparing 8 assembled transcripts takes
several minutes on the default instance; the price is
negligible (~ 0.01$). Unless specified otherwise, the
default instance used to run the __GffCompare__ tool will
be c4.large (AWS).\n\n*Cost can be significantly reduced by
using **spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [GffCompare
manual
```

page](http://ccb.jhu.edu/software/stringtie/gff.shtml#gffcompare)\n\n[2]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "0.11.6",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "8",
"name": "hisat2_extractsplicesites_2_2_1",
"description": "__HISAT2 ExtractSpliceSites__ extracts a
list of splice sites from a GTF file in the **HISAT2**'s
own format as follows (4 tab-delimited columns): \n(1)
chromosome name; (2) zero-offset based genomic position of
the flanking base on the left side of an intron; (3)
zero-offset based genomic position of the flanking base on
the right; (4) strand.\n\n*A list of **all inputs and
parameters** with corresponding descriptions can be found
at the end of the page.*\n\n### Common Use Cases\n\n* This
tool can be used as a preprocessing step for __HISAT2
Build__. It is used to create a file containing splice
sites that can be forwarded to __HISAT2 Build__ in order to
create an index that can improve alignment accuracy.\n\n###
Changes Introduced by Seven Bridges\n\n* Output file will
be prefixed by the input gene annotation filename, unless
the __Output file prefix__ option is explicitly
specified.\n\n### Common Issues and Important Notes\n\n* No
common issues specific to the tool's execution on the Seven
Bridges platform have been detected. \n\n### Performance
Benchmarking\n\nThe execution time for human gene
annotation takes several minutes on the default instance;

the price is negligible (~ $0.02) using on-demand AWS instances.\n\n*Cost can be significantly reduced by **spot instance** usage. Visit [knowledge center](https://docs.sevenbridges.com/docs/about-spot-instances) for more details.*",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "9",
"name": "hisat2_extractexons_2_2_1",
"description": "__HISAT2 ExtractExons__ extracts a list of exons from a GTF file in the **HISAT2**'s own format which is a tab-delimited file with the following columns: (1) chromosome name; (2) zero-offset based left genomic position of an exon; and (3) zero-offset based right genomic position of an exon.\n\n*A list of **all inputs and parameters** with corresponding descriptions can be found at the end of the page.*\n\n### Common Use Cases\n\n* This tool can be used as a preprocessing step for __HISAT2 Build__. It is used to create a file containing exons that can be further fed to __HISAT2 Build__ in order to create an index that can improve alignment accuracy.\n\n### Changes Introduced by Seven Bridges\n\n* Output file will be prefixed by the input gene annotation filename, unless the __Output file prefix__ option is explicitly specified.\n\n### Common Issues and Important Notes\n\n* No common issues specific to the tool's execution on the Seven Bridges platform have been detected.\n\n### Performance Benchmarking\n\nThe execution time for human gene

annotation takes several minutes on the default instance;
the price is negligible (~ $0.02) using on-demand AWS
instances. \n\n*Cost can be significantly reduced by **spot
instance** usage. Visit [knowledge
center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "10",
"name": "hisat2_build_2_2_1",
"description": "__HISAT2 Build__ tool builds a HISAT2 index
necessary for the __HISAT2__ alignment method. To create
an index, __HISAT2 Build__ uses a genome reference file(s)
in FASTA format and outputs a set of 8 files with suffixes
.1.ht2, .2.ht2, .3.ht2, .4.ht2, .5.ht2, .6.ht2, .7.ht2, and
.8.ht2. In the case of a large index these suffixes will
have an ht2l termination. These files together constitute
the index: they are all that is needed to align reads to
that reference [1].\n\nThe __HISAT2__ indexing scheme is
called Hierarchical Graph Ferragina Manzini (HGFM) index.
In addition to using one global GFM index that represents
the general population, __HISAT2__ uses a large set of
small GFM indexes that collectively cover the whole genome
(each index representing a genomic region of 56 Kbp, with
55,000 indexes needed to cover the human population). These
small indexes (called local indexes) combined with several
alignment strategies enable effective alignment of
sequencing reads [1].\n\n*A list of **all inputs and

```
parameters** with corresponding descriptions can be found
at the bottom of this page.*\n\n***Please note that any
cloud infrastructure costs resulting from app and pipeline
executions, including the use of public apps, are the sole
responsibility of you as a user. To avoid excessive costs,
please read the app description carefully and set the app
parameters and execution settings accordingly.***\n\n###
Common Use Cases\n\n* This tool is used to create an index
for the __HISAT2__ splice aware aligner.\n* __HISAT2
Build__ can take advantage of known splice sites, exons,
SNPs and haplotypes to create an index that will enable the
__HISAT2__ aligner to create more accurate alignments.
These files can be created using __HISAT2 ExtractExons__,
__HISAT2 ExtractSpliceSites__ and __HISAT2 Extract
SNPsHaplotypes__. Pre-built index files using these options
can be also found on [HISAT2 home
page](https://daehwankimlab.github.io/hisat2/download/).\n*
__HISAT2 Build__ can generate either small or large
indexes. The wrapper will decide which one to generate
based on the length of the input genome. If the reference
does not exceed 4 billion characters but a large index is
preferred, the user can specify `--large-index` to force
__HISAT2 Build__ to build a large index instead [1].\n\n###
Changes Introduced by Seven Bridges\n\n* The directory
containing the index files will be outputted as a TAR
bundle (the __Index files__ output). This bundle can then
be provided to the __HISAT2__ aligner, which will
automatically take care of untarring it and preparing it to
run successfully without further issues.\n* __HISAT2
Build__ can accept a TAR bundle containing an already
indexed reference instead of a reference FASTA file(s), to
skip indexing and reduce processing time if this tool is a
```

```
part of a workflow.\n* Output file will be prefixed by the
input reference file name, unless the **Output file
prefix** option is explicitly specified.\n\n### Common
Issues and Important Notes\n\n* __Exon__ (`--exon`) and
__Splice sites__ (`--ss`) files have to be used together.
If exactly one of these options is used, __HISAT2 Build__
will fail. Input files for these two options can be
generated from __HISAT2 ExtractExons__ and __HISAT2
ExtractSpliceSites__ tools.\n* The **Max bucket size**
(`--bmax`), **Max bucket size (as divisior)**
(`--bmaxdivn`), and **Diff-cover period** (`--dcv`) options
are governing the trade off between running time and memory
usage and should only be set by advanced users.  __HISAT2
Build__ automatically sets these parameters to their
optimal values, that yield the best running time without
exhausting memory. This behavior can be disabled using
the -a/--noauto option [1].\n\n### Performance
Benchmarking\n\nIn the following table you can find
estimates of __HISAT2 Build__ duration and cost. Two
different instances were used for benchmarking because
__HISAT2 Build__ requires significantly more memory (~
200GB for human reference [1]) when using some of the
following options: __Splice sites file__ (`--ss`), __Exon
file__ (`--exon`) and/or __SNP file__ (`--snp`),
__Haplotype file__ (`--haplotype`) opposed to situations
where these options are not used. The Seven Bridges version
of the tool will dynamically choose an appropriate instance
based on the provided inputs. The results shown here are
obtained by indexing the human reference genome.  Execution
time and cost can vary for other genomes.\n\n*Cost can be
significantly reduced by **spot instances** usage. Visit
the [Knowledge
```

```
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n| Settings | Duration | Cost |
Instance
(AWS)|\n|------------------------------|--------------|------------------|---------------|\n|
Default (no additional files) | 18min.  | $0.25 |
c5.4xlarge |\n| Using `--ss` and `--exon` options | 51min.
| $1.84 | r5.8xlarge |\n| Using `--ss`, `--exon` ,`--snp`
and `--haplotype` options | 1h 12min.  | $2.60 | r5.8xlarge
|\n\n### References\n\n[1] [HISAT2 manual
page](https://daehwankimlab.github.io/hisat2/manual/)",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "11",
"name": "hisat2_2_2_1",
"description": "**HISAT2** is a fast and sensitive
alignment program for mapping next-generation sequencing
reads (both DNA and RNA) to a population of human genomes
as well as to a single reference genome [1]. \n\n__HISAT2__
(hierarchical indexing for spliced alignment of transcripts
2) aligns both DNA and RNA sequences using a graph
Ferragina Manzini (GFM) index. In addition to using one
global GFM index that represents the general
population, __HISAT2__ uses a large set of small GFM
indexes that collectively cover the whole genome (each
index representing a genomic region of 56 Kbp, with 55,000
indexes needed to cover the human population). These small
indexes (called local indexes) combined with several
alignment strategies enable effective alignment of
```

```
sequencing reads. This new indexing scheme is called
Hierarchical Graph FM (HGFM) index [1].\n\n*A list of **all
inputs and parameters** with corresponding descriptions can
be found at the bottom of this page.*\n\n***Please note
that any cloud infrastructure costs resulting from app and
pipeline executions, including the use of public apps, are
the sole responsibility of you as a user. To avoid
excessive costs, please read the app description carefully
and set the app parameters and execution settings
accordingly.***\n\n### Common Use Cases\n\n- Though
**HISAT2** can be used for all types of next-generation
sequencing reads, it is mostly used for aligning RNA-Seq
data.\n- The __Downstream transcriptome assembly__
(`--downstream-transcriptome-assembly/--dta`) parameter
tells __HISAT2__ to report alignments tailored for
transcript assemblers including **StringTie**. With this
option, __HISAT2__ requires longer anchor lengths for de
novo discovery of splice sites. This leads to fewer
alignments with short-anchors, which helps transcript
assemblers significantly improve computation and memory
usage.\n- The __Downstream transcriptome assembly__ -
__Cufflinks__ (`--dta-cufflinks`) parameter tells
__HISAT2__ to report alignments tailored specifically for
**Cufflinks**. In addition to what __HISAT2__ does with the
above option (__Downstream transcriptome assembly__), it
also looks for novel splice sites with three signals
(GT/AG, GC/AG, AT/AC). Nonetheless, all user-provided
splice sites are used irrespective of their signals.
__HISAT2__ produces an optional field, `XS:A:[+-]`, for
every spliced alignment.\n\n\n### Changes Introduced by
Seven Bridges\n\n* In order to facilitate and accelerate
further RNA-Seq analysis, an additional toolkit __Sambamba
```

```
(0.7.1)__ is integrated into the same Seven Bridges tool
representation alongside __HISAT2 (2.2.1)__. Besides the
standard __HISAT2__ SAM output, HISAT2 has been extended to
provide two additional output file options: \n * an
**unsorted BAM** file created by piping standard __HISAT2__
output to __Sambamba view__ and \n * a **coordinate-sorted
BAM** file and its Index (BAI) file created by additional
piping of the output through __Sambamba sort__ and
__Sambamba index__. \n To select the desired output, use
the __Output type__ parameter. The default output is a
**coordinate-sorted BAM** file.\n* If the __Read group ID__
parameter is not defined, by default it will be set to '1',
unless the __No @RG line__ option is explicitly specified.
If the tool is scattered within a workflow, it will assign
the __Read group ID__ according to the order of the
scattered folders. This also ensures a unique __Read group
ID__ when processing multi-read group input data from one
sample.\n* All output files will be prefixed by the input
sample ID (inferred from the __Sample ID__ metadata field
if existent, or from filename otherwise), unless the
__Output prefix__ option is explicitly specified.\n\n###
Common Issues and Important Notes\n\n- To run __HISAT2,__
properly indexed reference files are required. These files
can be either created using the __HISAT2 Build__ tool or
downloaded from the [HISAT2 home
page](https://daehwankimlab.github.io/hisat2/download/).\n-
Some __HISAT2__ options (i.e. __Maximum number of ambiguous
characters__ (`--n-ceil`) or __Long introns with canonical
splice sites penalty__ (`--pen-canintronlen`)) specify a
function rather than an individual number or setting. In
these cases, the user specifies three parameters: (a) a
function type F, (b) a constant term B, and (c) a
```

```
coefficient A. The available function types are constant
(C), linear (L), square-root (S), and natural log (G). The
parameters are specified as F,B,A - that is, the function
type, the constant term, and the coefficient are separated
by commas with no whitespace. The constant term and
coefficient may be negative and/or floating-point numbers
[1].\n\n         
Example 1: if the function specification is L,-0.4,-0.6,
then the function defined is:\n`f(x) = -0.4 + -0.6 *
x`\n\n         
Example 2: if the function specification is G,1,5.4, then
the function defined is:\n`f(x) = 1.0 + 5.4 * ln(x)`\n\n-
In a single run, __Reads__ should be either paired or
unpaired. __Reads__ may be a mix of different lengths.\n-
For paired-end __Reads__, the __Paired-end__ metadata field
has to be set to 1 or 2. Paired-end __Reads__ must
correspond to each other file-for-file and
read-for-read.\n\n### Performance Benchmarking\n\nFor the
human reference genome, __HISAT2__ requires about 9 GB of
RAM to run properly. In the following table you can find
estimates of __HISAT2__ running time and cost. All samples
are aligned onto the hg38 human reference index built using
reference transcriptome (i.e. using __Splice sites__
(`--ss`) and __Exon__ (`--exon`) options). \n\n*The cost of
running __HISAT2__ can be significantly reduced by using
**spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n| Experiment type | Input size |
Paired-end | # of reads | Read length | Duration | Cost |
Instance
(AWS)|\n|----------------------|----------------|------------|-----------------|------------
RNA-Seq | 2 x 21.5 GB | Yes | 95M | 101 | 1h 50min.  |
```

$0.99 | c4.2xlarge |\n| RNA-Seq | 2 x 10.8 GB | Yes | 47.5M | 101 | 57min. | $0.51 | c4.2xlarge |\n| RNA-Seq | 2 x 2.2 GB | Yes | 9.5M | 101 | 16min. | $0.14 | c4.2xlarge |\n| RNA-Seq | 3.9 GB | No | 17.6M | 51 | 12min. | $0.11 | c4.2xlarge |\n\n### References\n\n[1] [HISAT2 manual page](https://daehwankimlab.github.io/hisat2/manual/)",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "12",
"name": "sbg_pair_fastqs_by_metadata",
"description": "Tool accepts list of FASTQ files groups them into separate lists. This grouping is done using metadata values and their hierarchy (Sample ID > Library ID > Platform unit ID > File segment number) which should create unique combinations for each pair of FASTQ files. Important metadata fields are Sample ID, Library ID, Platform unit ID and File segment number. 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. Files with no paired end metadata are grouped in the same way as the ones with paired end metadata, generally they should be alone in a separate list. Files with no metadata set will be grouped together. \n\nIf there are more than two files in a group, this might create errors further down most pipelines and the user should check if the metadata fields for those files are set properly.",
"version": "NA",

```
"prerequisite": [],
"input_list": [],
"output_list": []
}
]
}
```


1.7 Parametric Domain


```
{
"keywords": [],
"xref": [],
"platform": [
"Seven Bridges Platform"
],
"pipeline_steps": [
{
"step_number": "1",
"name": "sbg_file_selector",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "2",
"name": "sbg_file_selector_1",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
```

```
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "3",
"name": "sbg_file_selector_2",
"description": "SBG File selector selects which input file
will be propagated to output. If 'Propagate first input'
parameter is set to 'True', input 'in_1' will be propagated
to output, else 'in_2' is chosen.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "4",
"name": "stringtie_2_1_3",
"description": "**StringTie** is a fast and highly
efficient assembler of RNA-Seq alignments into potential
transcripts. It uses a novel network flow algorithm as well
as an optional *de novo* assembly step to assemble and
quantitate full-length transcripts representing multiple
splice variants for each gene locus. Its input can include
not only alignments of short reads that can also be used by
other transcript assemblers, but also alignments of longer
sequences that have been assembled from those reads.  In
order to identify differentially expressed genes between
```

```
experiments, **StringTie**'s output can be processed by
specialized software like **Ballgown**, **Cuffdiff** or
other programs (**DESeq2**, **edgeR**, etc.) [1].\n\n*A
list of __all inputs and parameters__ with corresponding
descriptions can be found at the bottom of this
page.*\n\n*__Please note that any cloud infrastructure
costs resulting from app and pipeline executions, including
the use of public apps, are the sole responsibility of you
as a user. To avoid excessive costs, please read the app
description carefully and set the app parameters and
execution settings accordingly.__*\n\n### Common Use
Cases\n\n* __StringTie__ can be used as a transcriptome
assembler. It takes aligned reads as input (note that the
provided BAM file has to be sorted by coordinate) and
estimates expression level of genes and transcripts as it
assembles them [3]. __Reference annotation file__ (`-G`)
can be provided to guide the assembly process, though this
is optional. If the ultimate goal is differential
expression analysis assembled transcripts for each sample
have to be merged into unified set of transcripts for all
samples using __StringTie Merge__. After the merging step
is done additional run of __StringTie__ is required to
re-estimate merged transcripts for each sample using __Keep
annotated transcripts only__ (`-e`) option that tells
__StringTie__ to estimate expression levels only for
reference transcripts and the __Create input files for
Ballgown and DESeq2__ (`-B`) option that enables creating
count-data input files for __Ballgown__ and __DESeq2__. For
more details refer to StringTie protocol paper [3].\n*
__StringTie__ can be used just for quantification. For this
purpose __Reference annotation file__ (`-G`) has to be
provided and the __Keep annotated transcripts only__ (`-e`)
```

```
parameter should be set to True, telling StringTie to only
estimate expression levels of genes and transcripts present
in the annotation file. This mode should be used in
experiments when there is a predefined list of genes or
transcripts of interest or in case of re-estimating merged
transcripts as described above.\n* __Text feature file__
input loads a list of point-features to guide the
transcriptome assembly. Accepted point features are
transcription start sites (TSS) and polyadenylation sites
(CPAS). Details on how this file should look like can be
found in the input's description info.\n* To output a TAB
file in which gene abundance estimation will be reported,
set the __Output gene abundance__ (`-A`) parameter to
True.\n* To output a file with all transcripts in the
provided reference file that are fully covered by the
reads, set the __Output covered reference transcripts__
(`-C`) parameter to True. This option requires **Reference
annotation file** (`-G`) to be provided.\n\n### Changes
Introduced by Seven Bridges\n\n* In order to enable
__StringTie__ to produce quantification tables tailored for
__DESeq2__ or
[EdgeR](http://bioconductor.org/packages/release/bioc/html/edgeR.html),
the
[`prepDE.py`](https://ccb.jhu.edu/software/stringtie/dl/prepDE.py)
python script [2] that extracts raw counts from
**Ballgown** input tables is embedded within the app.
Parameters of the scripts have default values except names
of gene and transcript count matrices. To obtain raw counts
suitable for __DESeq2__ set **Create input files for
Ballgown and DESeq2** (`-B`) to True. This will also
produce __Ballgown__ input tables as these tables are
needed for obtaining __DESeq2__ raw counts.\n* __Ballgown__
```

input tables (5 CTAB files) are outputted as an archive TAR
file containing all 5 files. This TAR file can be directly
fed to __Ballgown__ without any further modification given
the metadata field __Sample ID__ is properly set (check the
Common Issues and Important Notes section).\n\n### Common
Issues and Important Notes\n\n* If you want to perform
differential expression analysis using quantification
produced by __StringTie__, make sure that all input BAM
files on the **Aligned reads** input have metadata field
__Sample ID__ properly set. The value of __Sample ID__
metadata field will be used to match corresponding
expression (count) and phenotype data by downstream tools
for differential expression (__DESeq2__ and
__Ballgown__).\n* All BAM files on the **Aligned reads**
input need to be sorted by coordinates.\n\n### Performance
Benchmarking\n\nThe execution time for
quantification/assembly+quantification for human RNA-Seq
data (~ 160M of reads/12GB) takes somewhat less than 15
minutes on the default instance; the price is very low (~
0.05$). Unless specified otherwise, the default instance
used to run the __StringTie__ tool will be c4.xlarge
(AWS).\n\n*Cost can be significantly reduced by using
spot instances. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie home
page](https://ccb.jhu.edu/software/stringtie/index.shtml)\n\n[2]
[StringTie manual page - using StringTie with DESeq2 and
EdgeR](https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual#deseq)\n\n[3]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],

```
"input_list": [],
"output_list": []
},
{
"step_number": "5",
"name": "stringtie_2_1_4",
"description": "**StringTie** is a fast and highly
efficient assembler of RNA-Seq alignments into potential
transcripts. It uses a novel network flow algorithm as well
as an optional *de novo* assembly step to assemble and
quantitate full-length transcripts representing multiple
splice variants for each gene locus. Its input can include
not only alignments of short reads that can also be used by
other transcript assemblers, but also alignments of longer
sequences that have been assembled from those reads.  In
order to identify differentially expressed genes between
experiments, **StringTie**'s output can be processed by
specialized software like **Ballgown**, **Cuffdiff** or
other programs (**DESeq2**, **edgeR**, etc.) [1].\n\n*A
list of __all inputs and parameters__ with corresponding
descriptions can be found at the bottom of this
page.*\n\n*__Please note that any cloud infrastructure
costs resulting from app and pipeline executions, including
the use of public apps, are the sole responsibility of you
as a user. To avoid excessive costs, please read the app
description carefully and set the app parameters and
execution settings accordingly.__*\n\n### Common Use
Cases\n\n* __StringTie__ can be used as a transcriptome
assembler. It takes aligned reads as input (note that the
provided BAM file has to be sorted by coordinate) and
estimates expression level of genes and transcripts as it
assembles them [3]. __Reference annotation file__ (`-G`)
```

can be provided to guide the assembly process, though this is optional. If the ultimate goal is differential expression analysis assembled transcripts for each sample have to be merged into unified set of transcripts for all samples using __StringTie Merge__. After the merging step is done additional run of __StringTie__ is required to re-estimate merged transcripts for each sample using __Keep annotated transcripts only__ (`-e`) option that tells __StringTie__ to estimate expression levels only for reference transcripts and the __Create input files for Ballgown and DESeq2__ (`-B`) option that enables creating count-data input files for __Ballgown__ and __DESeq2__. For more details refer to StringTie protocol paper [3].\n* __StringTie__ can be used just for quantification. For this purpose __Reference annotation file__ (`-G`) has to be provided and the __Keep annotated transcripts only__ (`-e`) parameter should be set to True, telling StringTie to only estimate expression levels of genes and transcripts present in the annotation file. This mode should be used in experiments when there is a predefined list of genes or transcripts of interest or in case of re-estimating merged transcripts as described above.\n* __Text feature file__ input loads a list of point-features to guide the transcriptome assembly. Accepted point features are transcription start sites (TSS) and polyadenylation sites (CPAS). Details on how this file should look like can be found in the input's description info.\n* To output a TAB file in which gene abundance estimation will be reported, set the __Output gene abundance__ (`-A`) parameter to True.\n* To output a file with all transcripts in the provided reference file that are fully covered by the reads, set the __Output covered reference transcripts__

(`-C`) parameter to True. This option requires **Reference annotation file** (`-G`) to be provided.\n\n### Changes Introduced by Seven Bridges\n\n* In order to enable __StringTie__ to produce quantification tables tailored for __DESeq2__ or [EdgeR](http://bioconductor.org/packages/release/bioc/html/edgeR.html), the [`prepDE.py`](https://ccb.jhu.edu/software/stringtie/dl/prepDE.py) python script [2] that extracts raw counts from **Ballgown** input tables is embedded within the app. Parameters of the scripts have default values except names of gene and transcript count matrices. To obtain raw counts suitable for __DESeq2__ set **Create input files for Ballgown and DESeq2** (`-B`) to True. This will also produce __Ballgown__ input tables as these tables are needed for obtaining __DESeq2__ raw counts.\n* __Ballgown__ input tables (5 CTAB files) are outputted as an archive TAR file containing all 5 files. This TAR file can be directly fed to __Ballgown__ without any further modification given the metadata field __Sample ID__ is properly set (check the Common Issues and Important Notes section).\n\n### Common Issues and Important Notes\n\n* If you want to perform differential expression analysis using quantification produced by __StringTie__, make sure that all input BAM files on the **Aligned reads** input have metadata field __Sample ID__ properly set. The value of __Sample ID__ metadata field will be used to match corresponding expression (count) and phenotype data by downstream tools for differential expression (__DESeq2__ and __Ballgown__).\n* All BAM files on the **Aligned reads** input need to be sorted by coordinates.\n\n### Performance Benchmarking\n\nThe execution time for

```
quantification/assembly+quantification for human RNA-Seq
data (~ 160M of reads/12GB) takes somewhat less than 15
minutes on the default instance; the price is very low (~
0.05$). Unless specified otherwise, the default instance
used to run the __StringTie__ tool will be c4.xlarge
(AWS).\n\n*Cost can be significantly reduced by using
**spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie home
page](https://ccb.jhu.edu/software/stringtie/index.shtml)\n\n[2]
[StringTie manual page - using StringTie with DESeq2 and
EdgeR](https://ccb.jhu.edu/software/stringtie/index.shtml?t=manual#deseq)\n\n[3]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "6",
"name": "stringtie_merge_2_1_3",
"description": "The **StringTie Merge** tool takes a list
of GTF/GFF files as its input and merges/assembles these
transcripts into a non-redundant set of transcripts [1].
This tool is used as an intermediate step in the new Tuxedo
differential expression analysis pipeline described in [2]
to generate a global, unified set of transcripts (isoforms)
across multiple RNA-Seq samples.\n\n*A list of __all inputs
and parameters__ with corresponding descriptions can be
found at the bottom of this page.*\n\n*__Please note that
any cloud infrastructure costs resulting from app and
```

pipeline executions, including the use of public apps, are
the sole responsibility of you as a user. To avoid
excessive costs, please read the app description carefully
and set the app parameters and execution settings
accordingly.__*\n\n### Common Use Cases\n\n* This tool
should be used after __StringTie__ transcript assembling of
each sample in the experiment. This tool creates a set of
transcripts that is consistent across all samples, so that
the transcripts can be compared in subsequent steps of the
analysis [2]. For more details refer to the StringTie
protocol paper [2].\n\n### Changes Introduced by Seven
Bridges\n\n* No modifications to the original tool
representation have been made.\n\n### Common Issues and
Important Notes\n\n* No common issues specific to the
tool's execution on the Seven Bridges Platform have been
detected. \n\n### Performance Benchmarking\n\n* The
execution time for merging 8 assembled transcripts takes
several minutes on the default instance; the price is
negligible (~ 0.01$). Unless specified otherwise, the
default instance used to run the __StringTie Merge__ tool
will be c4.large (AWS).\n\n*Cost can be significantly
reduced by using **spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n### References\n\n[1] [StringTie
manual
page](http://ccb.jhu.edu/software/stringtie/index.shtml?t=manual)\n\n[2]
[HISAT, StringTie, Ballgown protocol
paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "2.1.3",
"prerequisite": [],
"input_list": [],
"output_list": []

},
{
"step_number": "7",
"name": "gffcompare_0_11_6",
"description": "__GffCompare__ [1] is a part of the GFF utility toolkit and can be used to:\n\n- compare and evaluate the accuracy of RNA-Seq transcript assemblers (__Cufflinks__, __Stringtie__).\n- collapse (merge) duplicate transcripts from multiple GTF/GFF3 files (e.g. resulted from assembly of different samples)\n- classify transcripts from one or multiple GTF/GFF3 files as they relate to reference transcripts provided in an annotation file (also in GTF/GFF3 format)\n\nThe original form of this program is also distributed as part of the Cufflinks suite, under the name \"__CuffCompare__\". Most of the options and parameters of __CuffCompare__ are supported by __GffCompare__, while new features will likely be added to __GffCompare__ in the future.\n\n*A list of __all inputs and parameters__ with corresponding descriptions can be found at the bottom of this page.*\n\n*__Please note that any cloud infrastructure costs resulting from app and pipeline executions, including the use of public apps, are the sole responsibility of you as a user. To avoid excessive costs, please read the app description carefully and set the app parameters and execution settings accordingly.__*\n\n### Common Use Cases\n\n* This tool should be used after __StringTie__ and __StringTie Merge__ transcript assembling and merging. The GffCompare program then compares the genes and transcripts with the annotation and reports statistics on this comparison. For more details refer to the StringTie protocol paper [2].\n\n### Changes Introduced by Seven Bridges\n\n* No modifications to the

original tool representation have been made.\n\n### Common Issues and Important Notes\n\n* No common issues specific to the tool's execution on the Seven Bridges Platform have been detected.\n\n### Performance Benchmarking\n\n* The execution time for comparing 8 assembled transcripts takes several minutes on the default instance; the price is negligible (~ 0.01$). Unless specified otherwise, the default instance used to run the __GffCompare__ tool will be c4.large (AWS).\n\n*Cost can be significantly reduced by using **spot instances**. Visit the [Knowledge Center](https://docs.sevenbridges.com/docs/about-spot-instances) for more details.*\n\n### References\n\n[1] [GffCompare manual page](http://ccb.jhu.edu/software/stringtie/gff.shtml#gffcompare)\n\n[2] [HISAT, StringTie, Ballgown protocol paper](http://www.nature.com/nprot/journal/v11/n9/full/nprot.2016.095.html)",
"version": "0.11.6",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "8",
"name": "hisat2_extractsplicesites_2_2_1",
"description": "__HISAT2 ExtractSpliceSites__ extracts a list of splice sites from a GTF file in the **HISAT2**'s own format as follows (4 tab-delimited columns): \n(1) chromosome name; (2) zero-offset based genomic position of the flanking base on the left side of an intron; (3) zero-offset based genomic position of the flanking base on the right; (4) strand.\n\n*A list of **all inputs and parameters** with corresponding descriptions can be found

at the end of the page.*\n\n### Common Use Cases\n\n* This tool can be used as a preprocessing step for __HISAT2 Build__. It is used to create a file containing splice sites that can be forwarded to __HISAT2 Build__ in order to create an index that can improve alignment accuracy.\n\n### Changes Introduced by Seven Bridges\n\n* Output file will be prefixed by the input gene annotation filename, unless the __Output file prefix__ option is explicitly specified.\n\n### Common Issues and Important Notes\n\n* No common issues specific to the tool's execution on the Seven Bridges platform have been detected. \n\n### Performance Benchmarking\n\nThe execution time for human gene annotation takes several minutes on the default instance; the price is negligible (~ $0.02) using on-demand AWS instances.\n\n*Cost can be significantly reduced by **spot instance** usage. Visit [knowledge center](https://docs.sevenbridges.com/docs/about-spot-instances) for more details.*",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "9",
"name": "hisat2_extractexons_2_2_1",
"description": "__HISAT2 ExtractExons__ extracts a list of exons from a GTF file in the **HISAT2**'s own format which is a tab-delimited file with the following columns: (1) chromosome name; (2) zero-offset based left genomic position of an exon; and (3) zero-offset based right genomic position of an exon.\n\n*A list of **all inputs and

parameters** with corresponding descriptions can be found
at the end of the page.*\n\n### Common Use Cases\n\n* This
tool can be used as a preprocessing step for __HISAT2
Build__. It is used to create a file containing exons that
can be further fed to __HISAT2 Build__ in order to create
an index that can improve alignment accuracy.\n\n###
Changes Introduced by Seven Bridges\n\n* Output file will
be prefixed by the input gene annotation filename, unless
the __Output file prefix__ option is explicitly
specified.\n\n### Common Issues and Important Notes\n\n* No
common issues specific to the tool's execution on the Seven
Bridges platform have been detected.\n\n### Performance
Benchmarking\n\nThe execution time for human gene
annotation takes several minutes on the default instance;
the price is negligible (~ $0.02) using on-demand AWS
instances. \n\n*Cost can be significantly reduced by **spot
instance** usage. Visit [knowledge
center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "10",
"name": "hisat2_build_2_2_1",
"description": "__HISAT2 Build__ tool builds a HISAT2 index
necessary for the __HISAT2__ alignment method. To create
an index, __HISAT2 Build__ uses a genome reference file(s)
in FASTA format and outputs a set of 8 files with suffixes
.1.ht2, .2.ht2, .3.ht2, .4.ht2, .5.ht2, .6.ht2, .7.ht2, and

```
.8.ht2. In the case of a large index these suffixes will
have an ht2l termination. These files together constitute
the index: they are all that is needed to align reads to
that reference [1].\n\nThe __HISAT2__ indexing scheme is
called Hierarchical Graph Ferragina Manzini (HGFM) index.
In addition to using one global GFM index that represents
the general population, __HISAT2__ uses a large set of
small GFM indexes that collectively cover the whole genome
(each index representing a genomic region of 56 Kbp, with
55,000 indexes needed to cover the human population). These
small indexes (called local indexes) combined with several
alignment strategies enable effective alignment of
sequencing reads [1].\n\n*A list of **all inputs and
parameters** with corresponding descriptions can be found
at the bottom of this page.*\n\n***Please note that any
cloud infrastructure costs resulting from app and pipeline
executions, including the use of public apps, are the sole
responsibility of you as a user. To avoid excessive costs,
please read the app description carefully and set the app
parameters and execution settings accordingly.***\n\n###
Common Use Cases\n\n* This tool is used to create an index
for the __HISAT2__ splice aware aligner.\n* __HISAT2
Build__ can take advantage of known splice sites, exons,
SNPs and haplotypes to create an index that will enable the
__HISAT2__ aligner to create more accurate alignments.
These files can be created using __HISAT2 ExtractExons__,
__HISAT2 ExtractSpliceSites__ and __HISAT2 Extract
SNPsHaplotypes__. Pre-built index files using these options
can be also found on [HISAT2 home
page](https://daehwankimlab.github.io/hisat2/download/).\n*
__HISAT2 Build__ can generate either small or large
indexes. The wrapper will decide which one to generate
```

```
based on the length of the input genome. If the reference
does not exceed 4 billion characters but a large index is
preferred, the user can specify `--large-index` to force
__HISAT2 Build__ to build a large index instead [1].\n\n###
Changes Introduced by Seven Bridges\n\n* The directory
containing the index files will be outputted as a TAR
bundle (the __Index files__ output). This bundle can then
be provided to the __HISAT2__ aligner, which will
automatically take care of untarring it and preparing it to
run successfully without further issues.\n* __HISAT2
Build__ can accept a TAR bundle containing an already
indexed reference instead of a reference FASTA file(s), to
skip indexing and reduce processing time if this tool is a
part of a workflow.\n* Output file will be prefixed by the
input reference file name, unless the **Output file
prefix** option is explicitly specified.\n\n### Common
Issues and Important Notes\n\n* __Exon__ (`--exon`) and
__Splice sites__ (`--ss`) files have to be used together.
If exactly one of these options is used, __HISAT2 Build__
will fail. Input files for these two options can be
generated from __HISAT2 ExtractExons__ and __HISAT2
ExtractSpliceSites__ tools.\n* The **Max bucket size**
(`--bmax`), **Max bucket size (as divisior)**
(`--bmaxdivn`), and **Diff-cover period** (`--dcv`) options
are governing the trade off between running time and memory
usage and should only be set by advanced users.  __HISAT2
Build__ automatically sets these parameters to their
optimal values, that yield the best running time without
exhausting memory. This behavior can be disabled using
the -a/--noauto option [1].\n\n### Performance
Benchmarking\n\nIn the following table you can find
estimates of __HISAT2 Build__ duration and cost. Two
```

different instances were used for benchmarking because
__HISAT2 Build__ requires significantly more memory (~
200GB for human reference [1]) when using some of the
following options: __Splice sites file__ (`--ss`), __Exon
file__ (`--exon`) and/or __SNP file__ (`--snp`),
__Haplotype file__ (`--haplotype`) opposed to situations
where these options are not used. The Seven Bridges version
of the tool will dynamically choose an appropriate instance
based on the provided inputs. The results shown here are
obtained by indexing the human reference genome. Execution
time and cost can vary for other genomes.\n\n*Cost can be
significantly reduced by **spot instances** usage. Visit
the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n| Settings | Duration | Cost |
Instance
(AWS)|\n|------------------------------|--------------|------------------|---------------|\n|
Default (no additional files) | 18min. | $0.25 |
c5.4xlarge |\n| Using `--ss` and `--exon` options | 51min.
| $1.84 | r5.8xlarge |\n| Using `--ss`, `--exon` ,`--snp`
and `--haplotype` options | 1h 12min. | $2.60 | r5.8xlarge
|\n\n### References\n\n[1] [HISAT2 manual
page](https://daehwankimlab.github.io/hisat2/manual/)",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "11",
"name": "hisat2_2_2_1",
"description": "**HISAT2** is a fast and sensitive

```
alignment program for mapping next-generation sequencing
reads (both DNA and RNA) to a population of human genomes
as well as to a single reference genome [1]. \n\n__HISAT2__
(hierarchical indexing for spliced alignment of transcripts
2) aligns both DNA and RNA sequences using a graph
Ferragina Manzini (GFM) index. In addition to using one
global GFM index that represents the general
population, __HISAT2__ uses a large set of small GFM
indexes that collectively cover the whole genome (each
index representing a genomic region of 56 Kbp, with 55,000
indexes needed to cover the human population). These small
indexes (called local indexes) combined with several
alignment strategies enable effective alignment of
sequencing reads. This new indexing scheme is called
Hierarchical Graph FM (HGFM) index [1].\n\n*A list of **all
inputs and parameters** with corresponding descriptions can
be found at the bottom of this page.*\n\n***Please note
that any cloud infrastructure costs resulting from app and
pipeline executions, including the use of public apps, are
the sole responsibility of you as a user. To avoid
excessive costs, please read the app description carefully
and set the app parameters and execution settings
accordingly.***\n\n### Common Use Cases\n\n- Though
**HISAT2** can be used for all types of next-generation
sequencing reads, it is mostly used for aligning RNA-Seq
data.\n- The __Downstream transcriptome assembly__
(`--downstream-transcriptome-assembly/--dta`) parameter
tells __HISAT2__ to report alignments tailored for
transcript assemblers including **StringTie**. With this
option, __HISAT2__ requires longer anchor lengths for de
novo discovery of splice sites. This leads to fewer
alignments with short-anchors, which helps transcript
```

```
assemblers significantly improve computation and memory
usage.\n- The __Downstream transcriptome assembly__ -
__Cufflinks__ (`--dta-cufflinks`) parameter tells
__HISAT2__ to report alignments tailored specifically for
**Cufflinks**. In addition to what __HISAT2__ does with the
above option (__Downstream transcriptome assembly__), it
also looks for novel splice sites with three signals
(GT/AG, GC/AG, AT/AC). Nonetheless, all user-provided
splice sites are used irrespective of their signals.
__HISAT2__ produces an optional field, `XS:A:[+-]`, for
every spliced alignment.\n\n\n### Changes Introduced by
Seven Bridges\n\n* In order to facilitate and accelerate
further RNA-Seq analysis, an additional toolkit __Sambamba
(0.7.1)__ is integrated into the same Seven Bridges tool
representation alongside __HISAT2 (2.2.1)__. Besides the
standard __HISAT2__ SAM output, HISAT2 has been extended to
provide two additional output file options: \n * an
**unsorted BAM** file created by piping standard __HISAT2__
output to __Sambamba view__ and \n * a **coordinate-sorted
BAM** file and its Index (BAI) file created by additional
piping of the output through __Sambamba sort__ and
__Sambamba index__. \n To select the desired output, use
the __Output type__ parameter. The default output is a
**coordinate-sorted BAM** file.\n* If the __Read group ID__
parameter is not defined, by default it will be set to '1',
unless the __No @RG line__ option is explicitly specified.
If the tool is scattered within a workflow, it will assign
the __Read group ID__ according to the order of the
scattered folders. This also ensures a unique __Read group
ID__ when processing multi-read group input data from one
sample.\n* All output files will be prefixed by the input
sample ID (inferred from the __Sample ID__ metadata field
```

if existent, or from filename otherwise), unless the
__Output prefix__ option is explicitly specified.\n\n###
Common Issues and Important Notes\n\n- To run __HISAT2,__
properly indexed reference files are required. These files
can be either created using the __HISAT2 Build__ tool or
downloaded from the [HISAT2 home
page](https://daehwankimlab.github.io/hisat2/download/).\n-
Some __HISAT2__ options (i.e. __Maximum number of ambiguous
characters__ (`--n-ceil`) or __Long introns with canonical
splice sites penalty__ (`--pen-canintronlen`)) specify a
function rather than an individual number or setting. In
these cases, the user specifies three parameters: (a) a
function type F, (b) a constant term B, and (c) a
coefficient A. The available function types are constant
(C), linear (L), square-root (S), and natural log (G). The
parameters are specified as F,B,A - that is, the function
type, the constant term, and the coefficient are separated
by commas with no whitespace. The constant term and
coefficient may be negative and/or floating-point numbers
[1].\n\n
Example 1: if the function specification is L,-0.4,-0.6,
then the function defined is:\n`f(x) = -0.4 + -0.6 *
x`\n\n
Example 2: if the function specification is G,1,5.4, then
the function defined is:\n`f(x) = 1.0 + 5.4 * ln(x)`\n\n-
In a single run, __Reads__ should be either paired or
unpaired. __Reads__ may be a mix of different lengths.\n-
For paired-end __Reads__, the __Paired-end__ metadata field
has to be set to 1 or 2. Paired-end __Reads__ must
correspond to each other file-for-file and
read-for-read.\n\n### Performance Benchmarking\n\nFor the
human reference genome, __HISAT2__ requires about 9 GB of

```
RAM to run properly. In the following table you can find
estimates of __HISAT2__ running time and cost. All samples
are aligned onto the hg38 human reference index built using
reference transcriptome (i.e. using __Splice sites__
(`--ss`) and __Exon__ (`--exon`) options). \n\n*The cost of
running __HISAT2__ can be significantly reduced by using
**spot instances**. Visit the [Knowledge
Center](https://docs.sevenbridges.com/docs/about-spot-instances)
for more details.*\n\n| Experiment type | Input size |
Paired-end | # of reads | Read length | Duration | Cost |
Instance
(AWS)|\n|----------------------|-----------------|------------|-----------------|------------
RNA-Seq | 2 x 21.5 GB | Yes | 95M | 101 | 1h 50min.  |
$0.99 | c4.2xlarge |\n| RNA-Seq | 2 x 10.8 GB | Yes | 47.5M
| 101 | 57min.  | $0.51 | c4.2xlarge |\n| RNA-Seq | 2 x 2.2
GB | Yes | 9.5M | 101 | 16min.  | $0.14 | c4.2xlarge |\n|
RNA-Seq | 3.9 GB | No | 17.6M | 51 | 12min.  | $0.11 |
c4.2xlarge |\n\n### References\n\n[1] [HISAT2 manual
page](https://daehwankimlab.github.io/hisat2/manual/)",
"version": "2.2.1",
"prerequisite": [],
"input_list": [],
"output_list": []
},
{
"step_number": "12",
"name": "sbg_pair_fastqs_by_metadata",
"description": "Tool accepts list of FASTQ files groups
them into separate lists. This grouping is done using
metadata values and their hierarchy (Sample ID > Library ID
> Platform unit ID > File segment number) which should
create unique combinations for each pair of FASTQ files.
```

```
Important metadata fields are Sample ID, Library ID,
Platform unit ID and File segment number. 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. Files with no paired end metadata are
grouped in the same way as the ones with paired end
metadata, generally they should be alone in a separate
list. Files with no metadata set will be grouped together.
\n\nIf there are more than two files in a group, this might
create errors further down most pipelines and the user
should check if the metadata fields for those files are set
properly.",
"version": "NA",
"prerequisite": [],
"input_list": [],
"output_list": []
}
]
}
```

1.8 Input/Output Domain

```
{
  "input_subdomain": [
    {
      "uri": [
        {
          "filename": "",
          "uri": "",
          "access_time": ""
        }
      ]
    }
```

```
  ],
  "output_subdomain": [
    {
      "mediatype": "",
      "uri": [
        {
          "uri": "",
          "access_time": ""
        }
      ]
    }
  ]
}
```


1.9 Error Domain


```
{
  "empirical_error": [],
  "algorithmic_error": []
}
```

2 Funding

The Seven Bridges Cancer Genomics Cloud has been funded in whole or in part with Federal funds from the National Cancer Institute, National Institutes of Health, Contract No. HHSN261201400008C and ID/IQ Agreement No. 17X146 under Contract No. HHSN261201500003I.

3 References

Lau et al (2017) The Cancer Genomics Cloud: Collaborative, Reproducible, and Democratized—A New Paradigm in Large-Scale Computational Research. Cancer Res. 77(21):e3-e6. doi: 10.1158/0008-5472.CAN-17-0387.

4 Appendix 1: BioCompute Object Specification v1.3.0

Name	ID	Description
Top Level Fields		
BioCompute Object Indetifier	BCO_id	Unique identifier that should be apllied to each BCO instance. Assigned by a BCO database engine, like URL. It never be reused.
Type	type	As any object of the type, it has its own fields.
Digital signature	digital_signature	A string-type, read-only generated and stored by a BCO database, protecting the object from internal or external alterations without proper validation. It can be used for validation, downloading, and transferring BCOs.
BCO version	bco_spec_version	The version of the BCO specification used to define this document.
Provenance Domain		
Name	name	Name of the BCO.
Structured name	structured_name	Computable text field designed to represent a BCO instance name in visible interfaces
Version	version	Records the versioning of this BCO instance object. A change in the BCO affecting the outcome of the computation should be deposited as a new BCO, not as a new version.
Review	review	Describes the status of an object in the review process. Status flags: unreviewed, in-review, approved, suspended, rejected.
Inheritance/derivation	derived_from	If the object is derived from another, this field will specify the parent object, in the form of the objectid. It is null, if inherits only from the base BioCompute Object or a type definition.
Obsolescence	obsolete	If the object has an expiration date this field will specify that using the datetime type.
Embargo	embargo	If the object has a period of time that it is not public, that range can be specified using these fields. Using the datetime type a start and end time are specified for the embargo.
Created	created	Using the datetime type the time of initial creation of the BCO is recorded.
Modification	modified	Using the datetime type the time of most recent modification of the BCO is recorded.
Contributors	contributors	List to hold contributor identifiers and a description of their type of contribution, including a field for ORCIDs to record author information, as they allow for the author to curate their information after submission.
License	license	A space for Creative commons licence or other licence information. The default or recommended licence can be Attribution 4.0 International.
Usability Domain		
Usability Domain	usability_domain	Provides a space for the author to define the usability domain of the BCO. It is an array of free text values. This field is to aid in search-ability and provide a specific description of the object. It helps determine when and how the BCO can be used.
Extension Domain		

(continued)

Name	ID	Description
Extension Domain	extension_domain	For a user to add more structured information that is defined in the type definition. This section is not evaluated by checks for BCO validity or computational correctness.
Extension to External References: SMART on FHIR Genomics	Extension to External References: SMART on FHIR Genomics	SMART on FHIR Genomics provides a framework for HER-based apps to built on FHIR that integrate clinical and genomics information.
Extension to External References: GitHub	Extension to External References: GitHub	Include an extension to GitHub repositories where HTS computational analysis pipelins, workflows, protocols, and tool or software source code can be stored, deposited, downloaded.
Description Domain		
Description Domain	description_domain	Structured field for description of external references, the pipeline steps, and the relationship of IO objects. Information in this domain is not used for computation. Capture information that is currently being provided in FDA submission in journal format.
Keywords	keywords	List of key map fields to hold a list of keywords to aid in search-ability and description of the object.
External References	xref	It contains a list of the databases and/or ontology IDs that are cross-referenced in the BCO. It provides more specificity in the information related to BCO entries.
Pipeline tools	pipeline_steps	For recording the specifics of a pipeline. Each individual tool is represented as step, at the discretion of the author. Step Number (step_number), Name (name), Tool Description (description), Tool Version (version), Tool Prerequisites (prerequisite), Input List (input_list), Output List (output_list).
Execution Domain		
Execution Domain	execution_domain	The filelds required for execution of the BCO have been encapsulated together in order to clearly separate information needed for deployment, software configuration, and running applications in a dependent enviroment.
Script Access Type	script_access_type	This field indicates whether the code of the script to execute the BioCompute Object is access as an external file via HTTP or in-line text in the script field.
Script	script	Points to an internal or external reference to a script object that was used to perform computations for this BCO instance. This may be reference to Galaxy Project or Seven Bridges Genomics pipeline, a Common Workflow Language (CWL) object in GitHub, HIVE computational service or any other type of script.
Pipeline Version	pipeline_version	This field records the version of the pipeline implementation.
Platform/Environment	platform	The multi-value reference to a particular deployment of an existing platform where this BCO can be reproduced (Galaxy or HIVE or CASAVA).
Script Driver	script_driver	The reference to an executable that can be launched in order to perform a sequence of commands described in the script. For example if the pipeline is driven by a HIVE script, the script driver is the hive execution engine. For CWL based scripts specify cwl-runner. Another very general commonly used in Linux based operating systems is shell.

(continued)

Name	ID	Description
Algorithmic tools and Software Prerequisites	software_prerequisites	Field listing the minimal necessary prerequisites, library, tool versions needed to successfully run the script to produce BCO.
Domain Prerequisites	domain_prerequisites	Listing the minimal necessary domain specific external data source access in order to successfully run the script to produce BCO.
Enviromental parameters	env_parameters	Multi-value additional key value pairs useful to configure the execution environment on the target platform, like compute cores, available memory use of the script.
Parametric Domain		
Parametric Domain	parametric_domain	List of parameters customizing the computational flow which can affect the output of the calculations. These fields are custom to each type of analysis and are tied to a particular pipeline implementation.
Input and Output Domain		
Input and output Domain	io_domain	This represents the list of global input and output files created by the computational workflow, excluding the intermediate files.
Input Subdomain	input_subdomain	This field records the references and input files for the entire pipeline. Each type of input file is listed under a key for that type.
Output Subdomain	output_subdomain	This field records the outputs for the entire pipeline .
Error Domain, acceptable range of variability		
Error Domain, acceptable range of variability	error_domain	Consists of two subdomains: empirical and algorithmic._The empirical subdomain contains the limits of_detectability_ fps, fns, statistical confidence of outcomes, etc. The algorithmic subdomain is descriptive of errors that originated by fuzziness of the algorithms, driven by stochastic processes, in dynamically parallelized multi-threaded executions, or in machine learning methodologies where the state of the machine can affect the outcomeConsists of two subdomains: empirical and algorithmic. The empirical subdomain contains the limits of detectability FPs, FNs, statistical confidence of outcomes, etc. The algorithmic subdomain is descriptive of errors that originated by fuzziness of the algorithms, driven by stochastic processes, in dynamically parallelized multi-threaded executions, or in machine learning methodologies where the state of the machine can affect the outcome.

5 Appendix 2: The Complete BioCompute Object


```
{
  "spec_version": "https://w3id.org/biocompute/1.4.2/",
  "object_id": "https://biocompute.sbgenomics.com/bco/57f53382-3b30-4712-ae02-4d57d7721d5b",
  "etag": "ff2bab981cfc97700456b92211c1946e7ac19693c9b43d3055683225add44568",
  "provenance_domain": {
    "name": "HISAT2-StringTie Workflow",
    "version": "1.0.0",
    "review": [],
    "derived_from": "https://cgc-api.sbgenomics.com/v2/apps/phil_webster/bco-cwl-examples/hisat
    "obsolete_after": "2023-02-16T00:00:00+0000",
    "embargo": ["2023-02-16T00:00:00+0000", "2023-02-16T00:00:00+0000"],
    "created": "2023-02-16T00:00:00+0000",
    "modified": "2023-02-16T00:00:00+0000",
    "contributors": [],
    "license": "https://spdx.org/licenses/CC-BY-4.0.html"
  },
  "usability_domain": "The __HISAT2-StringTie Workflow__ can be used to perform a gene abundanc
  "extension_domain": {
    "fhir_extension": {
      "fhir_endpoint": "",
      "fhir_version": "",
      "fhir_resources": {}
    },
    "scm_extension": {
      "scm_repository": "",
      "scm_type": "git",
      "scm_commit": "",
      "scm_path": "",
      "scm_preview": ""
    }
```

```
  },
  "description_domain": {
    "keywords": [],
    "xref": [],
    "platform": [
      "Seven Bridges Platform"
    ],
    "pipeline_steps": [
      {
        "step_number": "1",
        "name": "sbg_file_selector",
        "description": "SBG File selector selects which input file will be propagated to output
        "version": "NA",
        "prerequisite": [],
        "input_list": [],
        "output_list": []
      },
      {
        "step_number": "2",
        "name": "sbg_file_selector_1",
        "description": "SBG File selector selects which input file will be propagated to output
        "version": "NA",
        "prerequisite": [],
        "input_list": [],
        "output_list": []
      },
      {
        "step_number": "3",
        "name": "sbg_file_selector_2",
        "description": "SBG File selector selects which input file will be propagated to output
        "version": "NA",
        "prerequisite": [],
```

```
      "input_list": [],
      "output_list": []
    },
    {
      "step_number": "4",
      "name": "stringtie_2_1_3",
      "description": "**StringTie** is a fast and highly efficient assembler of RNA-Seq alig
      "version": "2.1.3",
      "prerequisite": [],
      "input_list": [],
      "output_list": []
    },
    {
      "step_number": "5",
      "name": "stringtie_2_1_4",
      "description": "**StringTie** is a fast and highly efficient assembler of RNA-Seq alig
      "version": "2.1.3",
      "prerequisite": [],
      "input_list": [],
      "output_list": []
    },
    {
      "step_number": "6",
      "name": "stringtie_merge_2_1_3",
      "description": "The **StringTie Merge** tool takes a list of GTF/GFF files as its input
      "version": "2.1.3",
      "prerequisite": [],
      "input_list": [],
      "output_list": []
    },
    {
      "step_number": "7",
```

```
    "name": "gffcompare_0_11_6",
    "description": "__GffCompare__ [1] is a part of the GFF utility toolkit and can be used
    "version": "0.11.6",
    "prerequisite": [],
    "input_list": [],
    "output_list": []
  },
  {
    "step_number": "8",
    "name": "hisat2_extractsplicesites_2_2_1",
    "description": "__HISAT2 ExtractSpliceSites__ extracts a list of splice sites from a GT
    "version": "2.2.1",
    "prerequisite": [],
    "input_list": [],
    "output_list": []
  },
  {
    "step_number": "9",
    "name": "hisat2_extractexons_2_2_1",
    "description": "__HISAT2 ExtractExons__ extracts a list of exons from a GTF file in the
    "version": "2.2.1",
    "prerequisite": [],
    "input_list": [],
    "output_list": []
  },
  {
    "step_number": "10",
    "name": "hisat2_build_2_2_1",
    "description": "__HISAT2 Build__ tool builds a HISAT2 index necessary for the __HISAT2_
    "version": "2.2.1",
    "prerequisite": [],
    "input_list": [],
```

```
        "output_list": []
      },
      {
        "step_number": "11",
        "name": "hisat2_2_2_1",
        "description": "**HISAT2** is a fast and sensitive alignment program for mapping next-g
        "version": "2.2.1",
        "prerequisite": [],
        "input_list": [],
        "output_list": []
      },
      {
        "step_number": "12",
        "name": "sbg_pair_fastqs_by_metadata",
        "description": "Tool accepts list of FASTQ files groups them into separate lists. This
        "version": "NA",
        "prerequisite": [],
        "input_list": [],
        "output_list": []
      }
    ]
  },
  "execution_domain": {
    "script": [
      "https://cgc-api.sbgenomics.com/v2/apps/phil_webster/bco-cwl-examples/hisat2-stringtie/0/
    ],
    "script_driver": "Seven Bridges Common Workflow Language Executor",
    "software_prerequisites": [],
    "external_data_endpoints": [],
    "environment_variables": []
  },
  "parametric_domain": [],
```

```
  "io_domain": {
    "input_subdomain": [
      {
        "uri": [
          {
            "filename": "",
            "uri": "",
            "access_time": ""
          }
        ]
      }
    ],
    "output_subdomain": [
      {
        "mediatype": "",
        "uri": [
          {
            "uri": "",
            "access_time": ""
          }
        ]
      }
    ]
  },
  "error_domain": {
    "empirical_error": [],
    "algorithmic_error": []
  }
}
```