Reference Documentation for ODGI v0.6.0

Simon Heumos, Andrea Guarracino, Erik Garrison

Table of Contents

1. odgi (1)

1.1. NAME

odgi - dynamic succinct variation graph tool

1.2. SYNOPSIS

odgi build -g graph.gfa -o graph.og

odgi stats -i graph.og -S

odgi depth -i graph.og

odgi cover -i graph.og -o graph.paths.og

odgi extract -i graph.og -p prefix -r path_name:0-17

odgi explode -i graph.og -p prefix

odgi squeeze -f input_graphs.txt -o graphs.og

odgi position -i target_graph.og -g

odgi sort -i graph.og -o graph.sorted.og -p bSnSnS

odgi view -i graph.og -g

odgi kmers -i graph.og -c -k 23 -e 34 -D 50

odgi unitig -i graph.og -f -t 1324 -l 120

odgi viz -i graph.og -o graph.og.png -x 1920 -y 1080 -R -t 28

odgi paths -i graph.og -f

odgi prune -i graph.og -o graph.pruned.og -c 3 -C 345 -T

odgi unchop -i graph.og -o graph.unchopped.og

odgi normalize -i graph.og -o graph.normalized.og -I 100 -d

odgi bin -i graph.og -j -w 100 -s -g

odgi matrix -i graph.og -e -d

odgi chop -i graph.og -o graph.choped.og -c 1000

odgi groom -i graph.og -o graph.groomed.og

odgi layout -i graph.og -o graph.svg -R 10 -m 100

odgi break -i graph.og -o graph.broken.og -s 100 -d

odgi pathindex -i graph.og -o graph.xp

odgi panpos -i graph.og -p Chr1 -n 4

odgi server -i graph.og -p 4000 -ip 192.168.8.9

odgi test

odgi version

1.3. DESCRIPTION

odgi, the **Optimized Dynamic (genome) Graph Interface**, links a thrifty dynamic in-memory variation graph data model to a set of algorithms designed for scalable sorting, pruning, transformation, and visualization of very large genome graphs. **odgi** includes python bindings that can be used to directly interface with its data model. This **odgi** manual provides detailed information about its features and subcommands, including examples.

1.4. COMMANDS

Each command has its own man page which can be viewed using e.g. **man odgi_build.1**. Below we have a brief summary of syntax and subcommand description.

odgi build [**-g, --gfa**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi build(1) command constructs a succinct variation graph from a GFA. Currently, only GFA1 is supported. For details of the format please see https://github.com/GFA-spec/GFA-spec/blob/master/GFA1.md.

odgi stats [**-i, --idx**=*FILE*] [*OPTION*]...
The odgi stats(1) command produces statistics of a variation graph. Among other metrics, it can calculate the #nodes, #edges, #paths and the total nucleotide length of the graph. Various histogram summary options complement the tool. If [**-B, --bed-multicov**=*BED*] is set, the metrics will be produced for the intervals specified in the BED.

odgi depth [**-i, --input**=*FILE*] [*OPTION*]... The odgi depth(1) command finds the depth of graph as defined by query criteria.

odgi cover [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi cover(1) command finds a path cover of a variation graph, with a specified number of paths per component.

odgi extract [**-f, --input-graphs**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]... The odgi extract(1) command extracts parts of the graph as defined by query criteria.

odgi explode [**-i, --idx**=*FILE*] [**-p, --prefix**=*STRING*] [*OPTION*]...
The odgi explode(1) command breaks a graph into connected components, writing each component in its own file.

odgi squeeze [**-f, --input-graphs**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]... The odgi squeeze(1) command squeezes multiple graphs into the same file.

odgi position [**-i, --target**=*FILE*] [*OPTION*]... The odgi position(1) command position parts of the graph as defined by query criteria.

odgi sort [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi sort(1) command sorts a succinct variation graph. Odgi sort offers a diverse palette of sorting algorithms to determine the node order:

- A topological sort: A graph can be sorted via breadth-first search (BFS) or depth-first search (DFS). Optionally, a chunk size specifies how much of the graph to grab at once in each topological sorting phase. The sorting algorithm will continue the sort from the next node in the prior graph order that has not been sorted, yet. The cycle breaking algorithm applies a DFS sort until a cycle is found. We break and start a new DFS sort phase from where we stopped.
- A random sort: The graph is randomly sorted. The node order is randomly shuffled from Mersenne Twister pseudo-random generated numbers.
- A sparse matrix mondriaan sort: We can partition a hypergraph with integer weights and uniform hyperedge costs using the Mondriaan partitioner.
- A 1D linear SGD sort: Odgi implements a 1D linear, variation graph adjusted, multi-threaded version of the Graph Drawing by Stochastic Gradient Descent algorithm. The force-directed graph drawing algorithm minimizes the graph's energy function or stress level. It applies stochastic gradient descent (SGD) to move a single pair of nodes at a time.
- An eades algorithmic sort: Use Peter Eades' heuristic for graph drawing.

Sorting the paths in a graph my refine the sorting process. For the users' convenience, it is possible to specify a whole pipeline of sorts within one parameter.

odgi view [**-i, --idx**=*FILE*] [*OPTION*]...
The odgi view(1) command can convert a graph in odgi format to GFAv1. It can reveal a graph's internal structures for e.g. debugging processes.

odgi kmers [**-i, --idx**=*FILE*] [**-c, --stdout**] [*OPTION*]...
Given a kmer length, the odgi kmers(1) command can emit all kmers. The output can be refined by setting the maximum number of furcations at edges or by not considering nodes above a given node degree limit.

odgi unitig [**-i, --idx**=*FILE*] [*OPTION*]...
The odgi unitig(1) command can print all unitigs of a given odgi graph to standard output in FASTA format. Unitigs can also be emitted in a fixed sequence quality FASTQ format. Various parameters can refine the unitigs to print.

odgi viz [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi viz(1) command can produce a linear, static visualization of an odgi variation graph. It aggregates the pangenome into bins and directly renders a raster image. The binning level depends on the target width of the PNG to emit. Can be used to produce visualizations for gigabase scale pangenomes. For more information about the binning process, please refer to odgi bin. If reverse coloring was selected, only the bins with a reverse rate of at least 0.5 are colored. Currently, there is

no parameter to color according to the sequence coverage in bins available.

odgi paths [**-i, --idx**=*FILE*] [*OPTION*]…
The odgi paths(1) command allows the investigation of paths of a given variation graph. It can calculate overlap statistics of groupings of paths.

odgi prune [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi prune(1) command can remove complex parts of a graph. One can drop paths, nodes by a certain kind of edge coverage, edges and graph tips. Specifying a kmer length and a maximum number of furcations, the graph can be broken at edges not fitting into these conditions.

odgi unchop [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi unchop(1) command merges each unitig into a single node.

odgi normalize [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi normalize(1) command unchops a given variation graph and simplifies redundant furcations.

odgi matrix [**-i, --idx**=*FILE*] [*OPTION*]…
The odgi matrix(1) command generates a sparse matrix format out of the graph topology of a given variation graph.

odgi bin [**-i, --idx**=*FILE*] [*OPTION*]…
The odgi bin(1) command bins a given variation graph. The pangenome sequence, the one-time traversal of all nodes from smallest to largest node identifier, can be summed up into bins of a specified size. For each bin, the path metainformation is summarized. This enables a summarized view of gigabase scale graphs. Each step of a path is a bin and connected to its next bin via a link. A link has a start bin identifier and an end bin identifier.
The concept of odgi bin is also applied in odgi viz. A demonstration of how the odgi bin JSON output can be used for an interactive visualization is realized in the Pantograph project. Per default, odgi bin writes the bins to stdout in a tab-delimited format: **path.name**, **path.prefix**, **path.suffix**, **bin** (bin identifier), **mean.cov** (mean coverage of the path in this bin), **mean.inv** (mean inversion rate of this path in this bin), **mean.pos** (mean nucleotide position of this path in this bin), **first.nucl** (first nucleotide position of this path in this bin), **last.nucl** (last nucleotide position of this path in this bin). These nucleotide ranges might span positions that are not present in the bin. Example: A range of 1-100 means that the first nucleotide has position 1 and the last has position 100, but nucleotide 45 could be located in another bin. For an exact positional output, please specify [**-j, --json**].

odgi chop [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [**-c, --chop-to**=*N*] [*OPTION*]…
The odgi chop(1) command chops long nodes into short ones while preserving the graph topology.

odgi layout [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…
The odgi layout(1) command draws 2D layouts of the graph using stochastic gradient descent (SGD). The input graph must be sorted and id-compacted. The algorithm itself is described in Graph Drawing by Stochastic Gradient Descent. The force-directed graph drawing algorithm minimizes the graph's energy function or stress level. It applies SGD to move a single pair of nodes at a time. The rendered graph is written in SVG format.

odgi flatten [**-i, --idx**=*FILE*] [*OPTION*]…

The odgi flatten(1) command projects the graph sequence and paths into FASTA and BED.

odgi break [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi break(1) command finds cycles in a graph via breadth-first search (BFS) and breaks them, also dropping the graph's paths.

odgi pathindex [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...
The odgi pathindex(1) command generates a path index of a graph. It uses succinct data structures to encode the index. The path index represents a subset of the features of a fully realized xg index. Having a path index, we can use odgi panpos to go from **path:position** → **pangenome:position** which is important when navigating large graphs in an interactive manner like in the Pantograph project.

odgi panpos [**-i, --idx**=*FILE*] [**-p, --path**=*STRING*] [**-n, --nuc-pos**=*N*] [*OPTION*]...
The odgi panpos(1) command give a pangenome position for a given path and nucleotide position. It requires a path index, which can be created with odgi pathindex. Going from **path:position** → **pangenome:position** is important when navigating large graphs in an interactive manner like in the Pantograph project. All input and output positions are 1-based.

odgi server [**-i, --idx**=*FILE*] [**-p, --port**=*N*] [*OPTION*]...
The odgi server(1) command starts an HTTP server with a given path index as input. The idea is that we can go from **path:position** → **pangenome:position** via GET requests to the HTTP server. The server headers do not block cross origin requests. Example GET request: *http://localost:3000/path_name/nucleotide_position*.
The required path index can be created with odgi pathindex. Going from **path:position** → **pangenome:position** is important when navigating large graphs in an interactive manner like in the Pantograph project. All input and output positions are 1-based. If no IP address is specified, the server will run on localhost.

odgi test [<TEST NAME|PATTERN|TAGS> ...] [*OPTION*]...
The odgi test(1) command starts all unit tests that are implemented in odgi. For targeted testing, a subset of tests can be selected. odgi test(1) depends on Catch2. In the default setting, all results are printed to stdout.

odgi version [*OPTION*]...
The odgi version(1) command prints the current git version with tags and codename to stdout (like *v-44-g89d022b "back to old ABI"*). Optionally, only the release, version or codename can be printed.

1.5. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

1.6. AUTHORS

Erik Garrison from the University of California Santa Cruz wrote the whole **odgi** tool. Simon Heumos from the Quantitative Biology Center Tübingen wrote **odgi pathindex**, **odgi panpos**, **odgi server**, and this documentation. Andrea Guarracino from the University of Rome Tor Vergata wrote **odgi viz**, **odgi extract**, **odgi cover**, **odgi explode**, **odgi squeeze**, **odgi depth**, and this

documentation*.

1.7. RESOURCES

Project web site: https://github.com/vgteam/odgi

Git source repository on GitHub: https://github.com/vgteam/odgi

GitHub organization: https://github.com/vgteam

Discussion list / forum: https://github.com/vgteam/odgi/issues

1.8. COPYING

2. odgi build(1)

2.1. NAME

odgi_build - construct a dynamic succinct variation graph

2.2. SYNOPSIS

odgi build [**-g, --gfa**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

2.3. DESCRIPTION

The odgi build(1) command constructs a succinct variation graph from a GFA. Currently, only GFA1

is supported. For details of the format please see https://github.com/GFA-spec/GFA-spec/blob/master/GFA1.md.

2.4. OPTIONS

2.4.1. Graph Files IO

-g, --gfa=*FILE*

GFA1 file containing the nodes, edges and paths to build a dynamic succinct variation graph from.

-o, --out=*FILE*

Write the dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

2.4.2. Graph Sorting

-s, --sort

Apply a general topological sort to the graph and order the node ids accordingly. A bidirected adaptation of Kahn's topological sort (1962) is used, which can handle components with no heads or tails. Here, both heads and tails are taken into account.

2.4.3. Processing Information

-p, --progress

Print progress updates to stdout.

-d, --debug

Verbosely print graph information to stderr. This includes the maximum node_id, the minimum node_id, the handle to node_id mapping, the deleted nodes and the path metadata.

--trace

Include backtrace information when reporting errors.

-v, --verbose

Verbosely print processing information to stderr, including debug-level log messages.

-w, --warnings

Turn on script warnings (applies to executed code).

-t, --threads=*N*

Number of threads to use for the parallel operations.

2.4.4. Program Information

-h, --help

Print a help message for **odgi build**.

2.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

2.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

2.7. AUTHORS

odgi build was written by Erik Garrison.

3. odgi stats(1)

3.1. NAME

odgi_stats - metrics describing variation graphs

3.2. SYNOPSIS

odgi stats [**-i, --idx**=*FILE*] [*OPTION*]...

3.3. DESCRIPTION

The odgi stats(1) command produces statistics of a variation graph. Among other metrics, it can calculate the #nodes, #edges, #paths and the total nucleotide length of the graph. Various histogram summary options complement the tool. If [**-B, --bed-multicov**=*BED*] is set, the metrics will be produced for the intervals specified in the BED.

3.4. OPTIONS

3.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to create statistics from. The file name usually ends with *.og*.

3.4.2. Summary Options

-S, --summarize

Summarize the graph properties and dimensions. Print to stdout the #nucleotides, #nodes, #edges and #paths of the graph.

-W, --weak-connected-components

Shows the properties of the weakly connected components.

-b, --base-content

Describe the base content of the graph. Print to stdout the #A, #C, #G and #T of the graph.

-C, --coverage

Provide a histogram of path coverage over bases in the graph. Print three tab-delimited columns to stdout: **type**, **cov**, **N**. **type** is one of *full* or *uniq* and determines if the histogram corresponds to the full graph or only to a unique paths graph. **cov** implies the #paths. **N** implies the #nucleotides.

-V, --set-coverage

Provide a histogram of coverage over unique set of paths. Print two tab-delimited columns to stdout: **cov**, **sets**. **cov** implies #nucleotides. **sets** lists the unique set of paths in a comma separated list. Sets with a **cov** of one and no paths in **sets** are listed, too.

-M, --multi-coverage

Provide a histogram of coverage over unique multiset, the combination with possible repetition of paths. Print two tab-delimited columns to stdout: **cov**, **sets**. **cov** implies #nucleotides. **sets** lists the unique multisets of paths in a comma separated list. Multisets with a **cov** of one and no paths in **sets** are listed, too.

3.4.3. BED Interval

-B, --bed-multicov=*BED*

For each BED entry, provide a table of path coverage over unique multisets of paths in the graph. Each unique multiset of paths overlapping a given BED interval is described in terms of its length relative to the total interval, the number of path traversals and unique paths involved in these traversals.

3.4.4. Sorting goodness evaluation

-l, --mean-links-length

Calculate the mean links length. This metric is path-guided and computable in 1D and 2D.

-g, --no-gap-links

Don’t penalize gap links in the mean links length. A gap link is a link which connects two nodes that are consecutive in the linear pangenomic order. This option is specifiable only to compute the mean links length in 1D.

-s, --sum-path-nodes-distances

Calculate the sum of path nodes distances. This metric is path-guided and computable in 1D and 2D. For each path, it iterates from node to node, summing their distances, and normalizing by the path length. In 1D, if a link goes back in the linearized viewpoint of the graph, this is

penalized (adding 3 times its length in the sum).

-d, --penalize-different-orientation

If a link connects two nodes which have different orientations, this is penalized (adding 2 times its length in the sum).

-c, --coords-in

File containing the layout coordinates of the succinct variation graph specified as input. The file name usually ends with *.lay*. When the layout coordinates are provided, the mean links length and the sum path nodes distances statistics are evaluated in 2D, else in 1D.

-P, --path-statistics

Display the statistics (mean links length or sum path nodes distances) for each path.

3.4.5. Threading

-t, --threads=***N***

Number of threads to use.

3.4.6. Program Information

-h, --help

Print a help message for **odgi stats**.

3.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

3.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

3.7. AUTHORS

odgi stats was written by Erik Garrison and Andrea Guarracino.

4. odgi depth(1)

4.1. NAME

odgi_depth - find the depth of graph as defined by query criteria

4.2. SYNOPSIS

odgi depth [**-i, --input**=*FILE*] [*OPTION*]...

4.3. DESCRIPTION

The odgi depth(1) command finds the depth of graph as defined by query criteria.

4.4. OPTIONS

4.4.1. Graph Files IO

-i, --input=***FILE***

Compute path depths in this graph.

4.4.2. Depth Options

-r, --path=***STRING***

Compute the depth of the given path in the graph

-R, --paths=***FILE***

Compute depth for the paths listed in FILE.

-g, --graph-pos=***[[node_id]]]***

Compute the depth at the given node, e.g. 7 or 3,4 or 42,10,+ or 302,0,-.

-G, --graph-pos-file=***FILE***

A file with graph path position per line.

-p, --path-pos=***[[path_name]]]***

Return depth at the given path position e.g. chrQ or chr3,42 or chr8,1337,+ or chrZ,3929,-.

-F, --path-pos-file=***FILE***

A file with one path position per line.

-b, --bed-input=***FILE***

A BED file of ranges in paths in the graph.

-d, --graph-depth

Compute the depth on each node in the graph.

-d, --search-radius=***STRING***

Limit coordinate conversion breadth-first search up to DISTANCE bp from each given position [default: 10000].

4.4.3. Threading

-t, --threads=*N*

Number of threads to use.

4.4.4. Program Information

-h, --help

Print a help message for **odgi depth**.

4.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

4.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

4.7. AUTHORS

odgi depth was written by Andrea Guarracino.

5. odgi sort(1)

5.1. NAME

odgi_sort - sort a variation graph

5.2. SYNOPSIS

odgi sort [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

5.3. DESCRIPTION

The odgi sort(1) command sorts a succinct variation graph. Odgi sort offers a diverse palette of sorting algorithms to determine the node order:

- A topological sort: A graph can be sorted via breadth-first search (BFS) or depth-first search (DFS). Optionally, a chunk size specifies how much of the graph to grab at once in each topological sorting phase. The sorting algorithm will continue the sort from the next node in the prior graph order that has not been sorted, yet. The cycle breaking algorithm applies a DFS sort

until a cycle is found. We break and start a new DFS sort phase from where we stopped.

- A random sort: The graph is randomly sorted. The node order is randomly shuffled from Mersenne Twister pseudo-random generated numbers.
- A 1D linear SGD sort: Odgi implements a 1D linear, variation graph adjusted, multi-threaded version of the Graph Drawing by Stochastic Gradient Descent algorithm. The force-directed graph drawing algorithm minimizes the graph's energy function or stress level. It applies stochastic gradient descent (SGD) to move a single pair of nodes at a time.
- A path guided, 1D linear SGD sort: Odgi implements a 1D linear, variation graph adjusted, multi-threaded version of the Graph Drawing by Stochastic Gradient Descent algorithm. The force-directed graph drawing algorithm minimizes the graph's energy function or stress level. It applies stochastic gradient descent (SGD) to move a single pair of nodes at a time. The path index is used to pick the terms to move stochastically. If ran with 1 thread only, the resulting order of the graph is deterministic. The seed is adjustable.

Sorting the paths in a graph my refine the sorting process. For the users' convenience, it is possible to specify a whole pipeline of sorts within one parameter.

5.4. OPTIONS

5.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to sort. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the sorted dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

-s, --sort-order=*FILE*

File containing the sort order. Each line contains one node identifier.

5.4.2. Topological Sorts

-b, --breadth-first

Use a (chunked) breadth first topological sort.

-B, --breadth-first-chunk=*N*

Chunk size for breadth first topological sort. Specify how many nucleotides to grap at once in each BFS phase.

-z, --depth-first

Use a (chunked) depth first topological sort.

-Z, --depth-first-chunk=*N*

Chunk size for the depth first topological sort. Specify how many nucleotides to grap at once in each DFS phace.

-w, --two-way

Use a two-way topological algorithm for sorting. It is a maximum of head-first and tail-first topological sort.

-n, --no-seeds

Don't use heads or tails to seed topological sort.

-c, --cycle-breaking

Use a cycle breaking sort.

5.4.3. Random Sort

-r, --random

Randomly sort the graph.

5.4.4. Path Guided 1D Linear SGD Sort

-Y, --path-sgd

Apply path guided 1D linear SGD algorithm to organize the graph.

-X, --path-index=*FILE*

Load the path index from this *FILE*.

-f, --path-sgd-use-paths=FILE

Specify a line separated list of paths to sample from for the on the fly term generation process in the path guided linear 1D SGD. The default value are *all paths*.

-G, --path-sgd-min-term-updates-paths=*N*

The minimum number of terms to be updated before a new path guided linear 1D SGD iteration with adjusted learning rate eta starts, expressed as a multiple of total path steps. The default value is *1.0*. Can be overwritten by *-U, -path-sgd-min-term-updates-nodes=N*.

-U, --path-sgd-min-term-updates-nodes=*N*

The minimum number of terms to be updated before a new path guided linear 1D SGD iteration with adjusted learning rate eta starts, expressed as a multiple of the number of nodes. Per default, the argument is not set. The default of *-G, path-sgd-min-term-updates-paths=N* is used).

-j, --path-sgd-delta=*N*

The threshold of maximum displacement approximately in bp at which to stop path guided linear 1D SGD. Default values is *0.0*.

-g, --path-sgd-eps=*N*

The final learning rate for path guided linear 1D SGD model. The default value is *0.01*.

-v, --path-sgd-eta-max=*N*

The first and maximum learning rate for path guided linear 1D SGD model. The default value is *squared steps of longest path in graph*.

-a, --path-sgd-zipf-theta=*N*

The theta value for the Zipfian distribution which is used as the sampling method for the second node of one term in the path guided linear 1D SGD model. The default value is *0.99*.

-x, --path-sgd-iter-max=*N*

The maximum number of iterations for path guided linear 1D SGD model. The default value is *30*.

-F, --iteration-max-learning-rate=*N*

The iteration where the learning rate is max for path guided linear 1D SGD model. The default value is *0*.

-k, --path-sgd-zipf-space=*N*

The maximum space size of the Zipfian distribution which is used as the sampling method for the second node of one term in the path guided linear 1D SGD model. The default value is the *longest path length*.

-I, --path-sgd-zipf-space-max=*N*

The maximum space size of the Zipfian distribution beyond which quantization occurs. Default value is *100*.

-l, --path-sgd-zipf-space-quantization-step=*N*

Quantization step size when the maximum space size of the Zipfian distribution is exceeded. Default value is *100*.

-y, --path-sgd-zipf-max-num-distributions=*N*

Approximate maximum number of Zipfian distributions to calculate. The default value is *100*.

-q, --path-sgd-seed=*N*

Set the seed for the deterministic 1-threaded path guided linear 1D SGD model. The default value is *pangenomic!*.

-u, --path-sgd-snapshot=***STRING***

Set the prefix to which each snapshot graph of a path guided 1D SGD iteration should be written to. This is turned off per default. This argument only works when *-Y, --path-sgd* was specified. Not applicable in a pipeline of sorts.

5.4.5. Path Sorting Options

-L, --paths-min

Sort paths by their lowest contained node identifier.

-M, --paths-max

Sort paths by their highest contained node identifier.

-A, --paths-avg

Sort paths by their average contained node identifier.

-R, --paths-avg-rev

Sort paths in reverse by their average contained node identifier.

-D, --path-delim=*path-delim*

Sort paths in bins by their prefix up to this delimiter.

5.4.6. Pipeline Sorting

-p, --pipeline=*STRING*

Apply a series of sorts, based on single character command line arguments given to this command. The default sort is *s*. The reverse sort would be specified via *f*.

5.4.7. Additional Parameters

-d, --dagify-sort

Sort on the basis of a DAGified graph.

-O, --Optimize

Use the MutableHandleGraph::optimize method to compact the node identifier space.

5.4.8. Threading

-t, --threads=*N*

Number of threads to use for the parallel operations.

5.4.9. Processing Information

-P, --progress

Print sort progress to stdout.

5.4.10. Program Information

-h, --help

Print a help message for **odgi sort**.

5.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

5.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

5.7. AUTHORS

odgi sort was written by Erik Garrison, Simon Heumos, and Andrea Guarracino.

6. odgi groom(1)

6.1. NAME

odgi_groom - resolve spurious inverting links

6.2. SYNOPSIS

odgi groom [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

6.3. DESCRIPTION

The odgi groom(1) command resolves spurious inverting links.

6.4. OPTIONS

6.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to groom. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the groomed succinct variation graph to *FILE*. The file name usually ends with *.og*.

6.4.2. Processing Information

-P, --progress

Display progress of the grooming to stderr.

6.4.3. Program Information

-h, --help

Print a help message for **odgi groom**.

6.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

6.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

6.7. AUTHORS

odgi groom was written by Erik Garrison and Andrea Guarracino.

7. odgi cover(1)

7.1. NAME

odgi_cover - find a path cover of the variation graph

7.2. SYNOPSIS

odgi cover [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

7.3. DESCRIPTION

The odgi cover(1) command finds a path cover of a variation graph, with a specified number of paths per component.

7.4. OPTIONS

7.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph where find a path cover. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the succinct variation graph with the generated paths to *FILE*. The file name usually ends with *.og*.

7.4.2. Cover Options

-n, --num-paths-per-component=*N*

Number of paths to generate per component.

-k, --node-windows-size=*N*

Size of the node window to check each time a new path is extended (it has to be greater than or equal to 2).

-c, --min-node-coverage=*N*

Minimum node coverage to reach (it has to be greater than 0). There will be generated paths until the specified minimum node coverage is reached, or until the maximum number of allowed generated paths is reached (number of nodes in the input variation graph).

-i, --ignore-paths

Ignore the paths already embedded in the graph during the nodes coverage initialization.

-w, --write-node-coverages=*FILE*

Write the node coverages at the end of the paths generation to FILE. The file will contain tab-separated values (header included), and have 3 columns: *component_id*, *node_id*, and *coverage*.

7.4.3. Threading

-t, --threads=*N*

Number of threads to use for the parallel sorter.

7.4.4. Processing Information

-P, --progress

Print information about the components and the progress to stderr.

7.4.5. Program Information

-h, --help

Print a help message for **odgi cover**.

7.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

7.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

7.7. AUTHORS

odgi cover was written by Andrea Guarracino.

8. odgi explode(1)

8.1. NAME

odgi_explode - breaks a graph into connected components in their own files

8.2. SYNOPSIS

odgi explode [**-i, --idx**=*FILE*] [**-p, --prefix**=*STRING*] [*OPTION*]...

8.3. DESCRIPTION

The odgi explode(1) command breaks a graph into connected components, writing each component in its own file.

8.4. OPTIONS

8.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to break in its components. The file name usually ends with *.og*.

8.4.2. Explode Options

-p, --prefix=***STRING***

Write each connected component in a file with the given prefix. The file for the component `i` will be named `STRING.i.og` (default: `component`)

-O, --optimize

Compact the node ID space in each connected component.

8.4.3. Threading

-t, --threads=***N***

Number of threads to use (to write the components in parallel).

8.4.4. Processing Information

-P, --progress

Print information about the components and the progress to stderr.

8.4.5. Program Information

-h, --help

Print a help message for **odgi explode**.

8.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

8.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

8.7. AUTHORS

odgi explode was written by Andrea Guarracino.

9. odgi squeeze(1)

9.1. NAME

odgi_squeeze - squeezes multiple graphs into the same file

9.2. SYNOPSIS

odgi squeeze [**-f, --input-graphs**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

9.3. DESCRIPTION

The odgi squeeze(1) command merges multiple graphs into the same file.

9.4. OPTIONS

9.4.1. Graph Files IO

-f, --input-graphs=***FILE***

Input file containing the list of graphs to squeeze into the same file. The file must contain one path per line.

-o, --out=***FILE***

Store all the input graphs in this file. The file name usually ends with *.og*.

9.4.2. Squeeze Options

-s, --rank-suffix=*STRING*

Add the separator and the input file rank as suffix to the path names (to avoid path name collisions).

-O, --optimize

Compact the node ID space in each input file before imploding.

9.4.3. Processing Information

-P, --progress

Print information about the progress to stderr.

9.4.4. Program Information

-h, --help

Print a help message for **odgi squeeze**.

9.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

9.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

9.7. AUTHORS

odgi squeeze was written by Andrea Guarracino.

10. odgi extract(1)

10.1. NAME

odgi_extract - extract parts of the graph as defined by query criteria

10.2. SYNOPSIS

odgi extract [**-f, --input-graphs**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

10.3. DESCRIPTION

The odgi extract(1) command extracts parts of the graph as defined by query criteria.

10.4. OPTIONS

10.4.1. Graph Files IO

-f, --input-graphs=*FILE*

File containing the succinct variation graph. The file name usually ends with *.og*.

-o, --out=*FILE*

Store all subgraph in this file. The file name usually ends with *.og*.

10.4.2. Extract Options

-s, --split-subgraphs=*STRING*

Instead of writing the target subgraphs into a single graph, write one subgraph per given target to a separate file named `path:start-end.og` (0-based coordinates).

-l, --node-list::_FILE_

A file with one node id per line. The node specified will be extracted from the input graph.

-n, --node::_STRING_

A single node from which to begin our traversal.

-c, --context::_NUMBER_

The number of steps away from our initial subgraph that we should collect.

-L, --use-length

Treat the context size as a length in bases (and not as a number of steps).

-r, --path-range

Find the node(s) in the specified path range TARGET=path[:pos1[-pos2]] (0-based coordinates)

-r, --bed-file::_FILE_

Find the node(s) in the path range(s) specified in the given BED FILE

-E, --full-range

Collects all nodes in the sorted order of the graph in the min and max position touched by the given path ranges. Be careful to use it with very complex graphs.

10.4.3. Threading

-t, --threads=*N*

Number of threads to use (to embed the subpaths in parallel).

10.4.4. Processing Information

-P, --progress

Print information to stderr.

10.4.5. Program Information

-h, --help

Print a help message for **odgi extract**.

10.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

10.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

10.7. AUTHORS

odgi extract was written by Andrea Guarracino.

11. odgi view(1)

11.1. NAME

odgi_view - projection of graphs into other formats

11.2. SYNOPSIS

odgi view [**-i, --idx**=*FILE*] [*OPTION*]...

11.3. DESCRIPTION

The odgi view(1) command can convert a graph in odgi format to GFAv1. It can reveal a graph's internal structures for e.g. debugging processes.

11.4. OPTIONS

11.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

-g, --to-gfa

Write the graph in GFAv1 format to standard output.

11.4.2. Summary Options

-d, --display

Show the internal structures of a graph. Print to stdout the maximum node identifier, the minimum node identifier, the nodes vector, the delete nodes bit vector and the path metadata, each in a separate line.

11.4.3. Program Information

-h, --help

Print a help message for **odgi view**.

11.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

11.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

11.7. AUTHORS

odgi view was written by Erik Garrison.

12. odgi kmers(1)

12.1. NAME

odgi_kmers - show and characterize the kmer space of the graph

12.2. SYNOPSIS

odgi kmers [**-i, --idx**=*FILE*] [**-c, --stdout**] [*OPTION*]...

12.3. DESCRIPTION

Given a kmer length, the odgi kmers(1) command can emit all kmers. The output can be refined by setting the maximum number of furcations at edges or by not considering nodes above a given node degree limit.

12.4. OPTIONS

12.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

-c, --stdout=

Write the kmers to standard output. Kmers are line-separated.

12.4.2. Kmer Options

-k, --kmer-length=*N*

The kmer length to generate kmers from.

-e, --max-furcations=*N*

Break at edges that would induce this many furcations when generating a kmer.

-D, --max-degree=*N*

Don't take nodes into account that have a degree greater than *N*.

12.4.3. Threading

-t, --threads=*N*

Number of threads to use.

12.4.4. Program Information

-h, --help

Print a help message for **odgi kmers**.

12.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

12.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

12.7. AUTHORS

odgi kmers was written by Erik Garrison.

13. odgi unitig(1)

13.1. NAME

odgi_unitig - output unitigs of the graph

13.2. SYNOPSIS

odgi unitig [**-i, --idx**=*FILE*] [*OPTION*]...

13.3. DESCRIPTION

The odgi unitig(1) command can print all unitigs of a given odgi graph to standard output in FASTA format. Unitigs can also be emitted in a fixed sequence quality FASTQ format. Various parameters can refine the unitigs to print.

13.4. OPTIONS

13.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

13.4.2. FASTQ Options

-f, --fake-fastq

Write the unitigs in FASTQ format to stdout with a fixed quality value of *I*.

13.4.3. Unitig Options

-t, --sample-to=*N*

Continue unitigs with a random walk in the graph so that they have at least the given *N* length.

-p, --sample-plus=*N*

Continue unitigs with a random walk in the graph by *N* past their natural end.

-l, --min-begin-node-length=*N*

Only begin unitigs collection from nodes which have at least length *N*.

13.4.4. Program Information

-h, --help

Print a help message for **odgi unitig**.

13.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

13.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

13.7. AUTHORS

odgi unitig was written by Erik Garrison.

14. odgi viz(1)

14.1. NAME

odgi_viz - variation graph visualizations

14.2. SYNOPSIS

odgi viz [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

14.3. DESCRIPTION

The odgi viz(1) command can produce a linear, static visualization of an odgi variation graph. It can aggregate the pangenome into bins and directly renders a raster image. The binning level can be specified in input or it is calculated from the target width of the PNG to emit. Can be used to produce visualizations for gigabase scale pangenomes. For more information about the binning process, please refer to odgi bin.

14.4. OPTIONS

14.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to convert from. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the visualization in PNG format to this file.

14.4.2. Visualization Options

-x, --width=*N*

Set the width in pixels of the output image.

-y, --height=*N*

Set the height in pixels of the output image.

-P, --path-height=*N*

The height in pixels for a path.

-X, --path-x-padding=*N*

The padding in pixels on the x-axis for a path.

-R, --pack-paths

Pack all paths rather than displaying a single path per row.

-L, --link-path-pieces=*FLOAT*

Show thin links of this relative width to connect path pieces.

-A, --alignment-prefix=*STRING*

Apply alignment related visual motifs to paths which have this name prefix. It affects the [**-S, --show-strand**] and [**-d, --change-darkness**] options.

-S, --show-strand

Use red and blue coloring to display forward and reverse alignments. This parameter can be set in combination with [**-A, --alignment-prefix**=*STRING*].

-z, --color-by-mean-inversion-rate

Change the color respect to the node strandness (black for forward, red for reverse); in binned mode (**-b, --binned-mode**), change the color respect to the mean inversion rate of the path for each bin, from black (no inversions) to red (bin mean inversion rate equals to 1).

-s, --color-by-prefix

Colors paths by their names looking at the prefix before the given character C.

14.4.3. Intervals selection

-r, --pangenomic-range

Nucleotide range to visualize: `STRING=[PATH:]start-end`. `*-end` for `[0,end]`; `start-*` for `[start,pangenome_length]`. If no PATH is specified, the nucleotide positions refer to the pangenome's sequence (i.e., the sequence obtained arranging all the graph's node from left to right).

14.4.4. Paths selection

-p, --paths-to-display

List of paths to display in the specified order; the file must contain one path name per line and a subset of all paths can be specified.

14.4.5. Path names visualization Options

-H, --hide-path-names

Hide the path names on the left of the generated image.

-C, --color-path-names-background

Color path names background with the same color as paths

-c, --max-num-of-characters

Maximum number of characters to display for each path name (max 128 characters). The default value is *the length of the longest path name* (up to 32 characters).

14.4.6. Binned Mode Options

-b, --binned-mode

The variation graph is binned before its visualization. Each pixel in the output image will correspond to a bin. For more information about the binning process, please refer to odgi bin.

-w, --bin-width=*N*

The bin width specifies the size of each bin in the binned mode. If it is not specified, the bin width is calculated from the width in pixels of the output image.

-g, --no-gap-links

We divide links into 2 classes:

1. the links which help to follow complex variations. They need to be drawn, else one could not follow the sequence of a path.
2. the links helping to follow simple variations. These links are called **gap-links**. Such links solely connecting a path from left to right may not be relevant to understand a path's traversal through the bins. Therefore, when this option is set, the gap-links are not drawn in binned mode.

-m, --color-by-mean-coverage

Change the color respect to the mean coverage of the path for each bin, from black (no coverage) to blue (max bin mean coverage in the entire graph).

14.4.7. Gradient Mode (also known as Position Mode) Options

-d, --change-darkness

Change the color darkness based on nucleotide position in the path. When it is used in binned mode, the mean inversion rate of the bin node is considered to set the color gradient starting position: when this rate is greater than 0.5, the bin is considered inverted, and the color gradient starts from the right-end of the bin. This parameter can be set in combination with [**-A, --alignment-prefix**=*STRING*].

-l, --longest-path

Use the longest path length to change the color darkness.

-u, --white-to-black

Change the color darkness from white (for the first nucleotide position) to black (for the last nucleotide position).

14.4.8. Program Information

-h, --help

Print a help message for **odgi viz**.

14.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

14.6. BUGS

```
Refer to the *odgi* issue tracker at https://github.com/vgteam/odgi/issues.
```

14.7. AUTHORS

odgi viz was written by Erik Garrison and Andrea Guarracino.

15. odgi paths(1)

15.1. NAME

odgi_paths - embedded path interrogation

15.2. SYNOPSIS

odgi paths [**-i, --idx**=*FILE*] [*OPTION*]...

15.3. DESCRIPTION

The odgi paths(1) command allows the investigation of paths of a given variation graph. It can calculate overlap statistics of groupings of paths.

15.4. OPTIONS

15.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to investigate the paths from. The file name usually ends with *.og*.

-O, --overlaps=***FILE***

Read in the path grouping file to generate the overlap statistics from. The file must be tab-delimited. The first column lists a grouping and the second the path itself. Each line has one path entry. For each group the pairwise overlap statistics for each pairing will be calculated and printed to stdout.

15.4.2. Investigation Options

-L, --list-paths

Print the paths in the graph to stdout. Each path is printed in its own line.

-H, --haplotypes

Print to stdout the paths in an approximate binary haplotype matrix based on the graph's sort order. The output is tab-delimited: **path.name**, **path.length**, **node.count**, **node.1**, **node.2**, **node.n**. Each path entry is printed in its own line.

-D, --delim=***CHAR***

The part of each path name before this delimiter is a group identifier. This parameter should only be set in combination with [**-H, --haplotypes**]. Prints an additional, first column **group.name** to stdout.

-d, --distance

Provides a sparse distance matrix for paths. If [**-D, --delim**] is set, it will be path groups distances.

-f, --fasta

Print paths in FASTA format to stdout.

15.4.3. Threading

-t, --threads=*N*

Number of threads to use.

15.4.4. Program Information

-h, --help

Print a help message for **odgi paths**.

15.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

15.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

15.7. AUTHORS

odgi paths was written by Erik Garrison.

16. odgi prune(1)

16.1. NAME

odgi_prune - remove complex parts of the graph

16.2. SYNOPSIS

odgi prune [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

16.3. DESCRIPTION

The odgi prune(1) command can remove complex parts of a graph. One can drop paths, nodes by a certain kind of edge coverage, edges and graph tips. Specifying a kmer length and a maximum number of furcations, the graph can be broken at edges not fitting into these conditions.

16.4. OPTIONS

16.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to load in. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the pruned graph to *FILE*. The file name should end with *.og*.

16.4.2. Kmer Options

-k, --kmer-length=*N*

The length of the kmers to consider.

-e, --max-furcations=*N*

Break at edges that would induce *N* many furcations in a kmer.

16.4.3. Node Options

-d, --max-degree=*N*

Remove nodes that have a higher node degree than *N*.

-c, --min-coverage=*N*

Remove nodese covered by fewer than *N* number of path steps.

-C, --max-coverage=*N*

Remove nodes covered by more than *N* number of path steps.

-T, --cut-tips=*N*

Remove nodes which are graph tips.

16.4.4. Edge Options

-E, --edge-coverage

Remove edges outside of the minimum and maximum coverage rather than nodes. Only set this argument in combination with [**-c, --min-coverage**=*N*] and [**-C, --max-coverage**=*N*].

-b, --best-edges=*N*

Only keep the *N* most covered inbound and output edges of each node.

16.4.5. Path Options

-D, --drop-paths

Remove the paths from the graph.

16.4.6. Threading

-t, --threads=*N*

Number of threads to use.

16.4.7. Program Information

-h, --help

Print a help message for **odgi prune**.

16.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

16.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

16.7. AUTHORS

odgi prune was written by Erik Garrison.

17. odgi unchop(1)

17.1. NAME

odgi_unchop - merge unitigs into single nodes

17.2. SYNOPSIS

odgi unchop [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

17.3. DESCRIPTION

The odgi unchop(1) command merges each unitig into a single node preserving the node order.

17.4. OPTIONS

17.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to unchop. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the unchopped dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

17.4.2. Processing Information

-d, --debug

Print information about the process to stderr.

17.4.3. Threading

-t, --threads=*N*

Number of threads to use for the parallel operations.

17.4.4. Program Information

-h, --help

Print a help message for **odgi unchop**.

17.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

17.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

17.7. AUTHORS

odgi unchop was written by Erik Garrison and Andrea Guarracino.

18. odgi normalize(1)

18.1. NAME

odgi_normalize - compact unitigs and simplify redundant furcations

18.2. SYNOPSIS

odgi normalize [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]…

18.3. DESCRIPTION

The odgi normalize(1) command unchops a given variation graph and simplifies redundant furcations.

18.4. OPTIONS

18.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to normalize. The file name usually ends with *.og*.

-o, --out=***FILE***

Write the normalized dynamic succinct variation graph to this file. A file ending with *.og* is recommended.

-I, --max-iterations=***N***

Iterate the normalization up to *N* many times. The default is *10*.

18.4.2. Program Debugging

-d, --debug

Print information about the normalization process to stdout.

18.4.3. Program Information

-h, --help

Print a help message for **odgi normalize**.

18.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

18.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

18.7. AUTHORS

odgi normalize was written by Erik Garrison.

19. odgi bin(1)

19.1. NAME

odgi_bin - binning of pangenome sequence and path information in the graph

19.2. SYNOPSIS

odgi bin [**-i, --idx**=*FILE*] [*OPTION*]...

19.3. DESCRIPTION

The odgi bin(1) command bins a given variation graph. The pangenome sequence, the one-time traversal of all nodes from smallest to largest node identifier, can be summed up into bins of a specified size. For each bin, the path metainformation is summarized. This enables a summarized view of gigabase scale graphs. Each step of a path is a bin and connected to its next bin via a link. A link has a start bin identifier and an end bin identifier.
The concept of odgi bin is also applied in odgi viz. A demonstration of how the odgi bin JSON output can be used for an interactive visualization is realized in the Pantograph project. Per default, odgi bin writes the bins to stdout in a tab-delimited format: **path.name**, **path.prefix**, **path.suffix**, **bin** (bin identifier), **mean.cov** (mean coverage of the path in this bin), **mean.inv** (mean inversion rate of this path in this bin), **mean.pos** (mean nucleotide position of this path in this bin), **first.nucl** (first nucleotide position of this path in this bin), **last.nucl** (last nucleotide position of this path in this bin). These nucleotide ranges might span positions that are not present in the bin. Example: A range of 1-100 means that the first nucleotide has position 1 and the last has position 100, but nucleotide 45 could be located in another bin. For an exact positional output, please specify **[-j, --json]**.
Running odgi bin in HaploBlocker mode, only arguments **[-b, --haplo-blocker]**, **[-p[N], --haplo-blocker-min-paths[N]]**, and **[-c[N], --haplo-blocker-min-coverage[N]]** are required. A TSV is printed to stdout: Each row corresponds to a node. Each column corresponds to a path. Each value is the coverage of a specific node of a specific path.

19.4. OPTIONS

19.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to investigate the bin from. The file name usually ends with *.og*.

19.4.2. FASTA Options

-f, --fasta=*FILE*

Write the pangenome sequence to *FILE* in FASTA format.

19.4.3. Bin Options

-n, --number-bins=*N*

The number of bins the pangenome sequence should be chopped up to.

-w, --bin-width=*N*

The bin width specifies the size of each bin.

-D, --path-delim=*STRING*

Annotate rows by prefix and suffix of this delimiter.

-a, --aggregate-delim

Aggregate on path prefix delimiter. Argument depends on [**-D, --path-delim**=*STRING*].

-j, --json

Print bins and links to stdout in pseudo JSON format. Each line is a valid JSON object, but the whole file is not a valid JSON! First, each bin including its pangenome sequence is printed to stdout per line. Second, for each path in the graph, its traversed bins including metainformation: **bin** (bin identifier), **mean.cov** (mean coverage of the path in this bin), **mean.inv** (mean inversion rate of this path in this bin), **mean.pos** (mean nucleotide position of this path in this bin), and an array of ranges determining the nucleotide position of the path in this bin. Switching first and last nucleotide in a range represents a complement reverse orientation of that particular sequence.

-s, --no-seqs

If [**-j, --json**] is set, no nucleotide sequences will be printed to stdout in order to save disk space.

-g, --no-gap-links

We divide links into 2 classes:

1. the links which help to follow complex variations. They need to be drawn, else one could not follow the sequence of a path.
2. the links helping to follow simple variations. These links are called **gap-links**. Such links solely connecting a path from left to right may not be relevant to understand a path's traversal through the bins. Therefore, when this option is set, the gap-links are left out saving disk space

19.4.4. HaploBlocker Options

-b, --haplo-blocker

Write a TSV to stdout formatted in a way ready for HaploBlocker: Each row corresponds to a node. Each column corresponds to a path. Each value is the coverage of a specific node of a specific path.

-p[N], --haplo-blocker-min-paths[N]

Specify the minimum number of paths that need to be present in the bin to actually report that bin. The default value is 1.

-c[N], --haplo-blocker-min-coverage[N]

Specify the minimum coverage a path needs to have in a bin to actually report that bin. The default value is 1.

19.4.5. Program Information

-h, --help

Print a help message for **odgi bin**.

-P, --progress

Write the current progress to stderr.

19.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

19.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

19.7. AUTHORS

odgi bin was written by Erik Garrison and Simon Heumos

20. odgi matrix(1)

20.1. NAME

odgi_matrix - write the graph topology in sparse matrix formats

20.2. SYNOPSIS

odgi matrix [**-i, --idx**=*FILE*] [*OPTION*]...

20.3. DESCRIPTION

The odgi matrix(1) command generates a sparse matrix format out of the graph topology of a given variation graph.

20.4. OPTIONS

20.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to create the sparse matrix from. The file name usually ends with *.og*.

20.4.2. Matrix Options

-e, --edge-depth-weight

Weigh edges by their path depth.

-d, --delta-weight

Weigh edges by their inverse id delta.

20.4.3. Program Information

-h, --help

Print a help message for **odgi matrix**.

20.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

20.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

20.7. AUTHORS

odgi matrix was written by Erik Garrison.

21. odgi chop(1)

21.1. NAME

odgi_chop - divide nodes into smaller pieces

21.2. SYNOPSIS

odgi chop [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [**-c, --chop-to**=*N*] [*OPTION*]...

21.3. DESCRIPTION

The odgi chop(1) command chops long nodes into short ones while preserving the graph topology and node order.

21.4. OPTIONS

21.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to chop. The file name usually ends with *.og*.

-o, --out=***FILE***

Write the choped succinct variation graph to *FILE*. The file name usually ends with *.og*.

21.4.2. Chop Options

-c, --chop-to=***N***

Divide nodes that longer than *N* into nodes no longer than *N* while maintaining graph topology.

21.4.3. Threading

-t, --threads=***N***

Number of threads to use for the parallel operations.

21.4.4. Processing Information

-d, --debug

Print information about the process to stderr.

21.4.5. Program Information

-h, --help

Print a help message for **odgi chop**.

21.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

21.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

21.7. AUTHORS

odgi chop was written by Erik Garrison and Andrea Guarracino.

22. odgi layout(1)

22.1. NAME

odgi_layout - use SGD to make 2D layouts of the graph

22.2. SYNOPSIS

odgi layout [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

22.3. DESCRIPTION

The odgi layout(1) command draws 2D layouts of the graph using stochastic gradient descent (SGD). The input graph must be sorted and id-compacted. The algorithm itself is described in Graph Drawing by Stochastic Gradient Descent. The force-directed graph drawing algorithm minimizes the graph's energy function or stress level. It applies SGD to move a single pair of nodes at a time. The rendered graph is written in SVG format.

22.4. OPTIONS

22.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to layout. The file name usually ends with *.og*.

-o, --out=***FILE***

Write the rendered layout in SVG format to *FILE*.

22.4.2. SGD Options

-m, --iter-max=*N*

The maximum number of iterations to run the layout. Default is *30*.

-p, --n-pivots=*N*

The number of pivots for sparse layout. Default is *0* leading to a non-sparse layout.

-e, --eps=*N*

The learning rate for SGD layout. Default is *0.01*.

22.4.3. SVG Options

-x, --x-padding=*N*

The padding between the connected component layouts. Default is *10.0*.

-R, --render-scale=*N*

SVG scaling Default is *5.0*.

22.4.4. Processing Information

-d, --debug

Print information about the components to stdout.

22.4.5. Program Information

-h, --help

Print a help message for **odgi layout**.

22.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

22.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

22.7. AUTHORS

odgi layout was written by Erik Garrison, Andrea Guarracino, and Simon Heumos.

23. odgi flatten(1)

23.1. NAME

odgi_flatten - generate linearization of the graph

23.2. SYNOPSIS

odgi flatten [**-i, --idx**=*FILE*] [*OPTION*]...

23.3. DESCRIPTION

The odgi flatten(1) command projects the graph sequence and paths into FASTA and BED.

23.4. OPTIONS

23.4.1. Graph Files IO

-i, --idx=***FILE***

File containing the succinct variation graph to flatten. The file name usually ends with *.og*.

23.4.2. Output Options

-f, --fasta=***FILE***

Write the concatenated node sequences in FASTA format to *FILE*.

-n, --name-seq=***STRING***

The name to use for the concatenated graph sequence. Default is the name of the input file which was specified via [**-i, --idx**=*FILE*].

-b, --bed=***FILE***

Write the mapping between graph paths and the linearized FASTA sequence in BED format to *FILE*.

23.4.3. Program Information

-h, --help

Print a help message for **odgi flatten**.

23.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

23.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

23.7. AUTHORS

odgi flatten was written by Erik Garrison.

24. odgi break(1)

24.1. NAME

odgi_break - break cycles in the graph and drop its paths

24.2. SYNOPSIS

odgi break [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

24.3. DESCRIPTION

The odgi break(1) command finds cycles in a graph via breadth-first search (BFS) and breaks them, also dropping the graph's paths.

24.4. OPTIONS

24.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to break. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the broken graph to *FILE*.

24.4.2. Cycle Options

-c, --cycle-max-bp=*N*

The maximum cycle length at which to break.

-s, --max-search-bp=*N*

The maximum search space of each BFS given in number of base pairs.

-u, --repeat-up-to=*N*

Iterate cycle breaking up to *N* times or stop if no new edges are removed.

-d, --show

Print the edges we would remove to stdout.

24.4.3. Program Information

-h, --help

Print a help message for **odgi break**.

24.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

24.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

24.7. AUTHORS

odgi break was written by Erik Garrison.

25. odgi pathindex(1)

25.1. NAME

odgi_pathindex - create a path index for a given path

25.2. SYNOPSIS

odgi pathindex [**-i, --idx**=*FILE*] [**-o, --out**=*FILE*] [*OPTION*]...

25.3. DESCRIPTION

The odgi pathindex(1) command generates a path index of a graph. It uses succinct data structures to encode the index. The path index represents a subset of the features of a fully realized xg index. Having a path index, we can use odgi panpos to go from **path:position → pangenome:position** which is important when navigating large graphs in an interactive manner like in the Pantograph project.

25.4. OPTIONS

25.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph to generate a path index from. The file name usually ends with *.og*.

-o, --out=*FILE*

Write the path index to *FILE*.

25.4.2. Program Information

-h, --help

Print a help message for **odgi pathindex**.

25.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

25.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

25.7. AUTHORS

odgi pathindex was written by Simon Heumos.

26. odgi panpos(1)

26.1. NAME

odgi_panpos - get the pangenome position of a given path and nucleotide position (1-based)

26.2. SYNOPSIS

odgi panpos [**-i, --idx**=*FILE*] [**-p, --path**=*STRING*] [**-n, --nuc-pos**=*N*] [*OPTION*]...

26.3. DESCRIPTION

The odgi panpos(1) command give a pangenome position for a given path and nucleotide position. It requires a path index, which can be created with odgi pathindex. Going from **path:position** → **pangenome:position** is important when navigating large graphs in an interactive manner like in the Pantograph project. All input and output positions are 1-based.

26.4. OPTIONS

26.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph index to find the pangenome position in. The file name usually ends with *.xp*.

26.4.2. Position Options

-p, --path=*STRING*

The path name of the query.

-n, --nuc-pos=*STRING*

The nucleotide sequence of the query.

26.4.3. Program Information

-h, --help

Print a help message for **odgi panpos**.

26.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

26.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

26.7. AUTHORS

odgi panpos was written by Simon Heumos.

27. odgi position(1)

27.1. NAME

odgi_position - position parts of the graph as defined by query criteria

27.2. SYNOPSIS

odgi position [**-i, --target**=*FILE*] [*OPTION*]...

27.3. DESCRIPTION

The odgi position(1) command position parts of the graph as defined by query criteria.

27.4. OPTIONS

27.4.1. Graph Files IO

-i, --target=***FILE***

Describe positions in this graph.

-x, --source=***FILE***

Translate positions from this graph into the target graph using common `--lift-paths` shared between both graphs [default: use the same source/target graph]

27.4.2. Position Options

-r, --ref-paths=***STRING***

Translate the given positions into positions relative to this reference path.

-R, --ref-paths=***FILE***

Use the ref-paths in FILE.

-l, --lift-path=***STRING***

Lift positions from `--source` to `--target` via coordinates in this path common to both graphs [default: all common paths between `--source` and `--target`].

-g, --graph-pos=***[[node_id]]]***

A graph position, e.g. 42,10,+ or 302,0,-.

-F, --path-pos-file=***FILE***

A file with one path position per line.

-b, --bed-input=***FILE***

A BED file of ranges in paths in the graph to lift into the target graph `-v`, `--give-graph-pos` emit

graph positions.

-v, --give-graph-pos

Emit graph positions (node,offset,strand) rather than path positions.

-d, --search-radius=*STRING*

Limit coordinate conversion breadth-first search up to DISTANCE bp from each given position [default: 10000].

27.4.3. Threading

-t, --threads=*N*

Number of threads to use.

27.4.4. Program Information

-h, --help

Print a help message for **odgi position**.

27.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

27.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

27.7. AUTHORS

odgi position was written by Erik Garrison.

28. odgi server(1)

28.1. NAME

odgi_server - start a HTTP server with a given index file to query a pangenome position

28.2. SYNOPSIS

odgi server [**-i, --idx**=*FILE*] [**-p, --port**=*N*] [*OPTION*]...

28.3. DESCRIPTION

The odgi server(1) command starts an HTTP server with a given path index as input. The idea is that we can go from **path:position → pangenome:position** via GET requests to the HTTP server. The server headers do not block cross origin requests. Example GET request: **http://localost:3000/path_name/nucleotide_position**.
The required path index can be created with odgi pathindex. Going from **path:position → pangenome:position** is important when navigating large graphs in an interactive manner like in the Pantograph project. All input and output positions are 1-based. If no IP address is specified, the server will run on localhost.

28.4. OPTIONS

28.4.1. Graph Files IO

-i, --idx=*FILE*

File containing the succinct variation graph index to host in a HTTP server. The file name usually ends with *.xp*.

28.4.2. HTTP Options

-p, --port=*N*

Run the server under this port.

-a, --ip=*IP*

Run the server under this IP address. If not specified, *IP* will be *localhost*.

28.4.3. Program Information

-h, --help

Print a help message for **odgi server**.

28.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

28.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

28.7. AUTHORS

odgi server was written by Simon Heumos.

29. odgi test(1)

29.1. NAME

odgi_test - run odgi unit tests

29.2. SYNOPSIS

odgi test [<TEST NAME|PATTERN|TAGS> ...] [*OPTION*]...

29.3. DESCRIPTION

The odgi test(1) command starts all unit tests that are implemented in odgi. For targeted testing, a subset of tests can be selected. odgi test(1) depends on Catch2. In the default setting, all results are printed to stdout.

29.4. OPTIONS

29.4.1. Testing Options

-l, --list-tests

List all test cases. If a pattern was specified, all matching test cases are listed.

-t, --list-tags

List all tags. If a pattern was specified, all matching tags are listed.

-s, --success

Include successful tests in output.

-b, --break

Break into debugger mode upon failed test.

-e, --nothrow

Skip exception tests.

-i, --invisibles

Show invisibles like tabs or newlines.

-o, --out=*FILE*

Write all output to *FILE*.

-r, --reporter=*STRING*

Reporter to use. Default is console.

-n, --name=*STRING*

Suite name.

-a, --abort

Abort at first failure.

-x, --abortx=*N*

Abort after *N* failures.

-w, --warn=*STRING*

Enable warnings.

-d, --durations=*yes*|*no*

Show test durations. Default is *no*.

-f, --input-file=*FILE*

Load test names from a file.

-#, --filenames-as-tags

Adds a tag for the file name.

-c, --section=*STRING*

Specify the section to run the tests on.

-v, --verosity=*quiet*|*normal*|*high*

Set output verbosity. Default is *normal*.

--list-test-names-only

List all test cases names only. If a pattern was specified, all matching test cases are listed.

--list-reporters

List all reporters.

--order=*decl*|*lex*|*rand*

Test case order. Default ist *decl*.

--rng-seed=*time*|*number*

Set a specific seed for random numbers.

--use-color=*yes*|*no*

Should the output be colorized? Default is *yes*.

--libidentify

Report name and version according to libidentify.

--wait-for-keypress=*start*|*exit*|*both*

Waits for a keypress before *start*|*exit*|*both*.

--benchmark-resolution-multiple

Multiple of clock resolution to run benchmarks.

29.4.2. Program Information

-?, -h, --help

Print a help message for **odgi test**.

29.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

29.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

29.7. AUTHORS

odgi test was written by Erik Garrison, Simon Heumos, and Andrea Guarracino.

30. odgi version(1)

30.1. NAME

odgi_version - display the version of odgi

30.2. SYNOPSIS

odgi version [*OPTION*]...

30.3. DESCRIPTION

The odgi version(1) command prints the current git version with tags and codename to stdout (like *v-44-g89d022b "back to old ABI"*). Optionally, only the release, version or codename can be printed.

30.4. OPTIONS

30.4.1. Version Options

-v, --version=

Print only the version (like *v-44-g89d022b*).

-c, --codename

Print only the codename (like *back to old ABI*).

-r, --release

Print only the release (like *v*).

30.4.2. Program Information

-h, --help

Print a help message for **odgi version**.

30.5. EXIT STATUS

0

Success.

1

Failure (syntax or usage error; parameter error; file processing failure; unexpected error).

30.6. BUGS

Refer to the **odgi** issue tracker at https://github.com/vgteam/odgi/issues.

30.7. AUTHORS

odgi version was written by Simon Heumos.