nature biotechnology

SUPPLEMENTARY INFORMATION

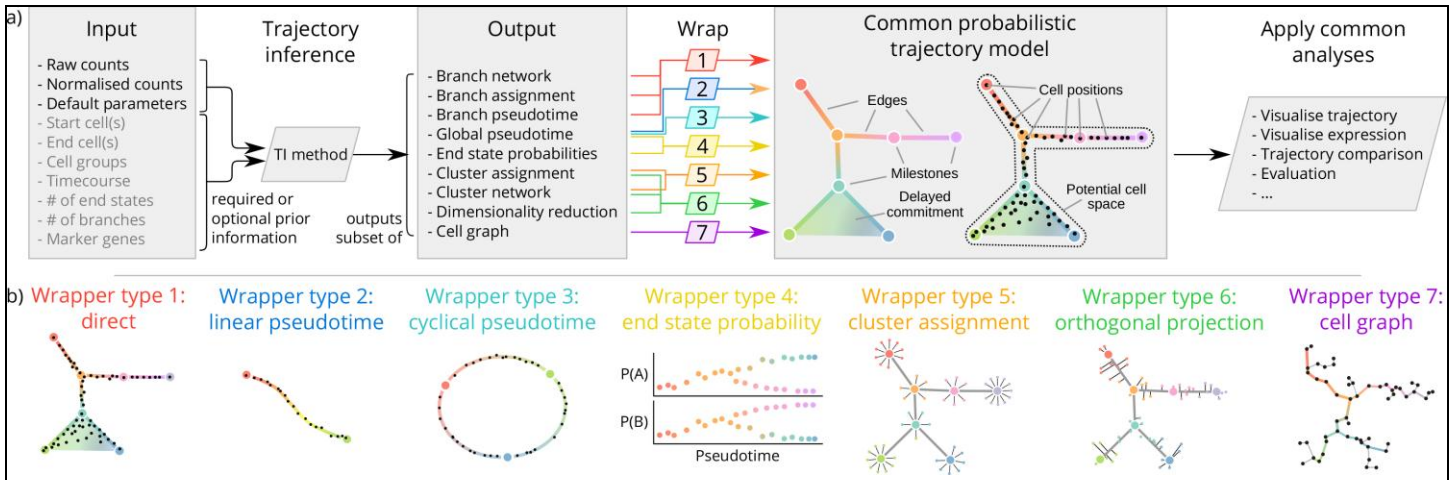
https://doi.org/10.1038/s41587-019-0071-9

In the format provided by the authors and unedited.

A comparison of single-cell trajectory inference methods

Wouter Saelens[1,2,6], Robrecht Cannoodt[1,3,4,6], Helena Todorov[1,2,5] and Yvan Saeys[1,2*]

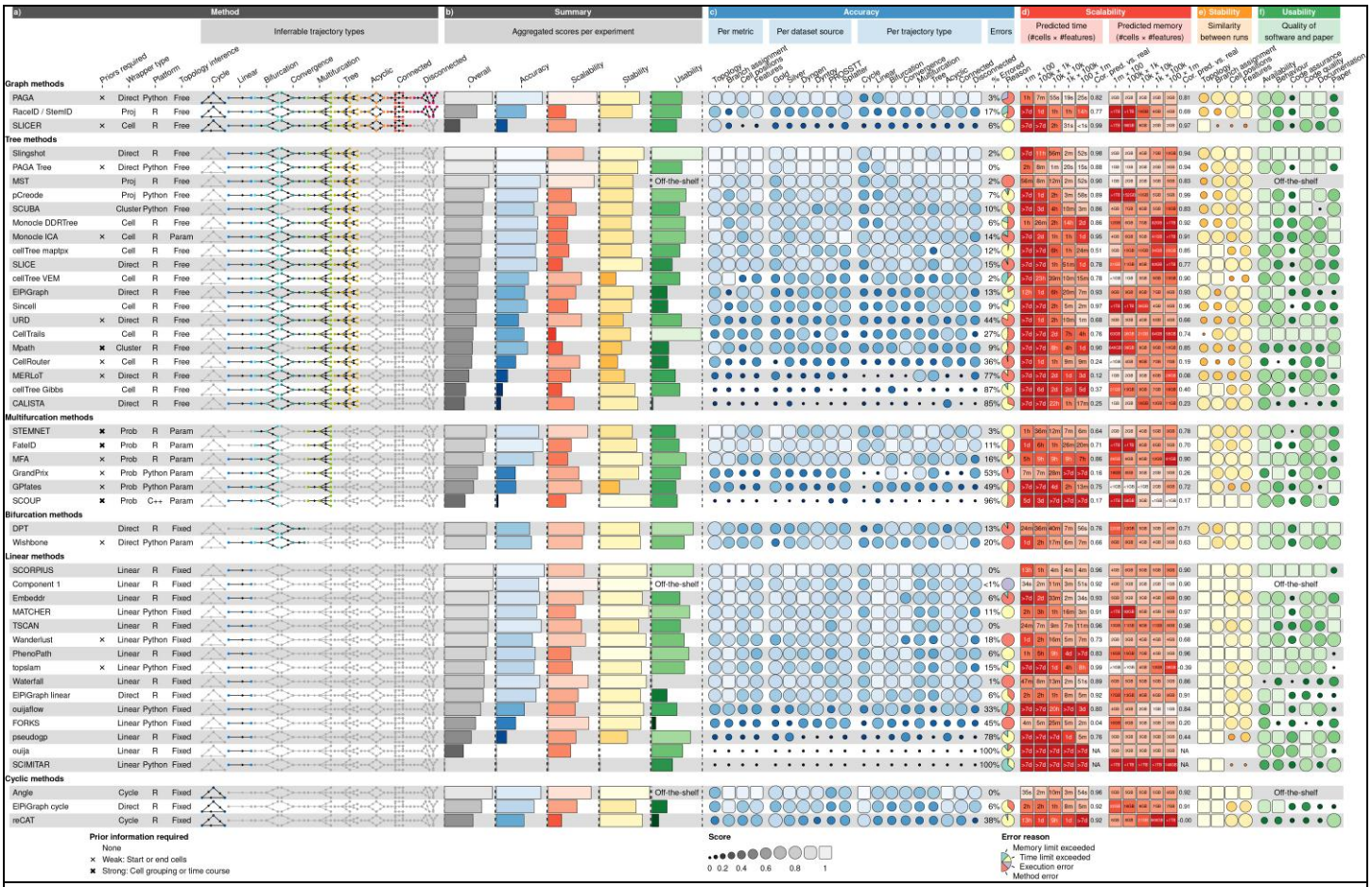
[1]Data mining and Modelling for Biomedicine, VIB Center for Inflammation Research, Ghent, Belgium. [2]Department of Applied Mathematics, Computer Science and Statistics, Ghent University, Ghent, Belgium. [3]Center for Medical Genetics, Ghent University Hospital, Ghent, Belgium. [4]Department of Biomolecular Medicine, Ghent University, Ghent, Belgium. [5]Centre International de Recherche en Infectiologie, Inserm, U1111, Université Claude Bernard Lyon 1, CNRS, UMR5308, École Normale Supérieure de Lyon, Université de Lyon, Lyon, France. [6]These authors contributed equally: Wouter Saelens, Robrecht Cannoodt. *e-mail: yvan.saeys@ugent.be



Supplementary Figure 1

A common interface for TI methods.

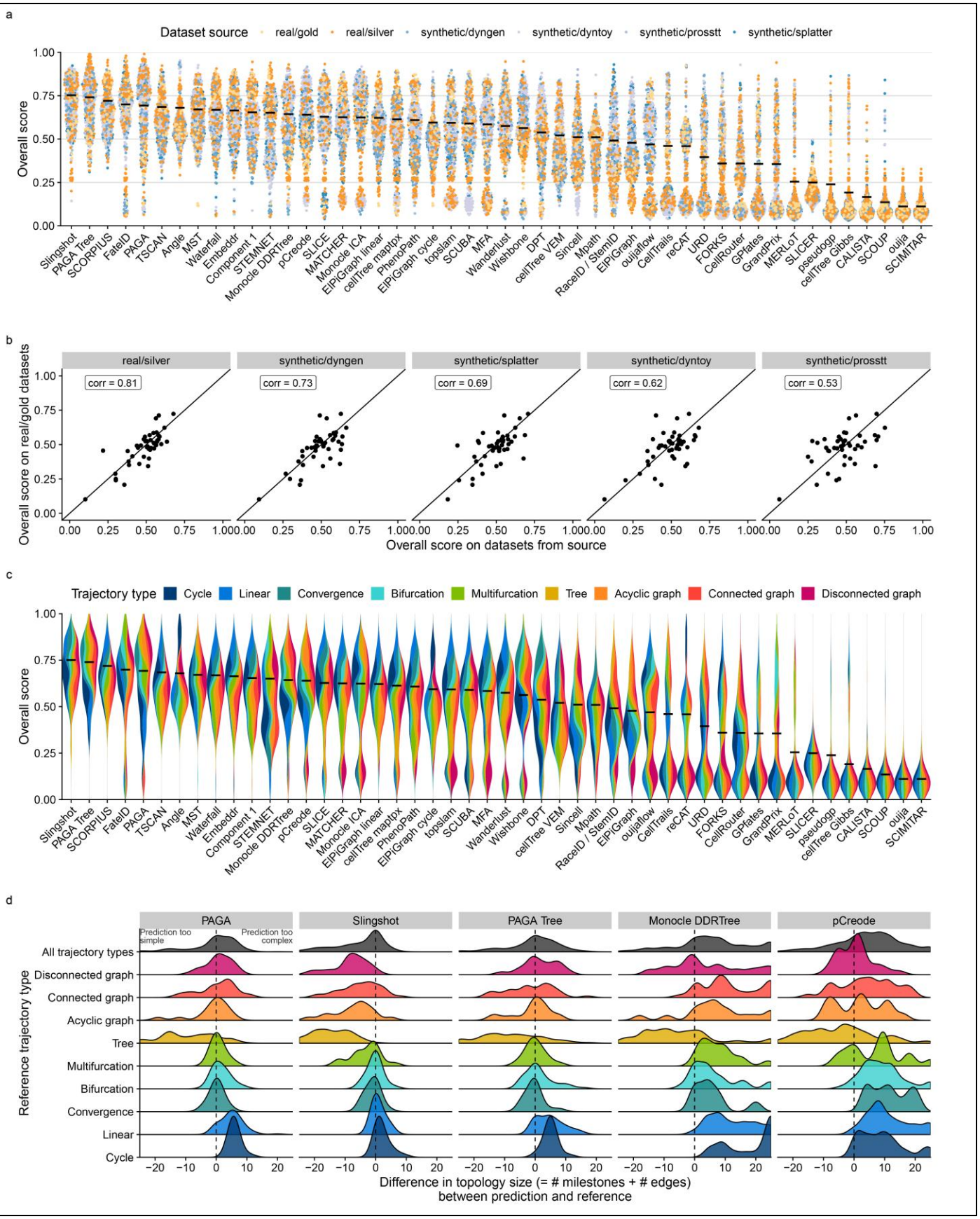
(a) The input and output of each TI method is standardized. As input, each TI method receives either raw or normalized counts, several parameters, and a selection of prior information. After its execution, a method uses one of the seven wrapper functions to transform its output to the common trajectory model. This common model then allows to perform common analysis functions on trajectory models produced by any TI method. (b) Illustrations of the specific transformations performed by each of the wrapper functions.



Supplementary Figure 2

Results from the evaluation, for all methods and across all evaluation criteria.

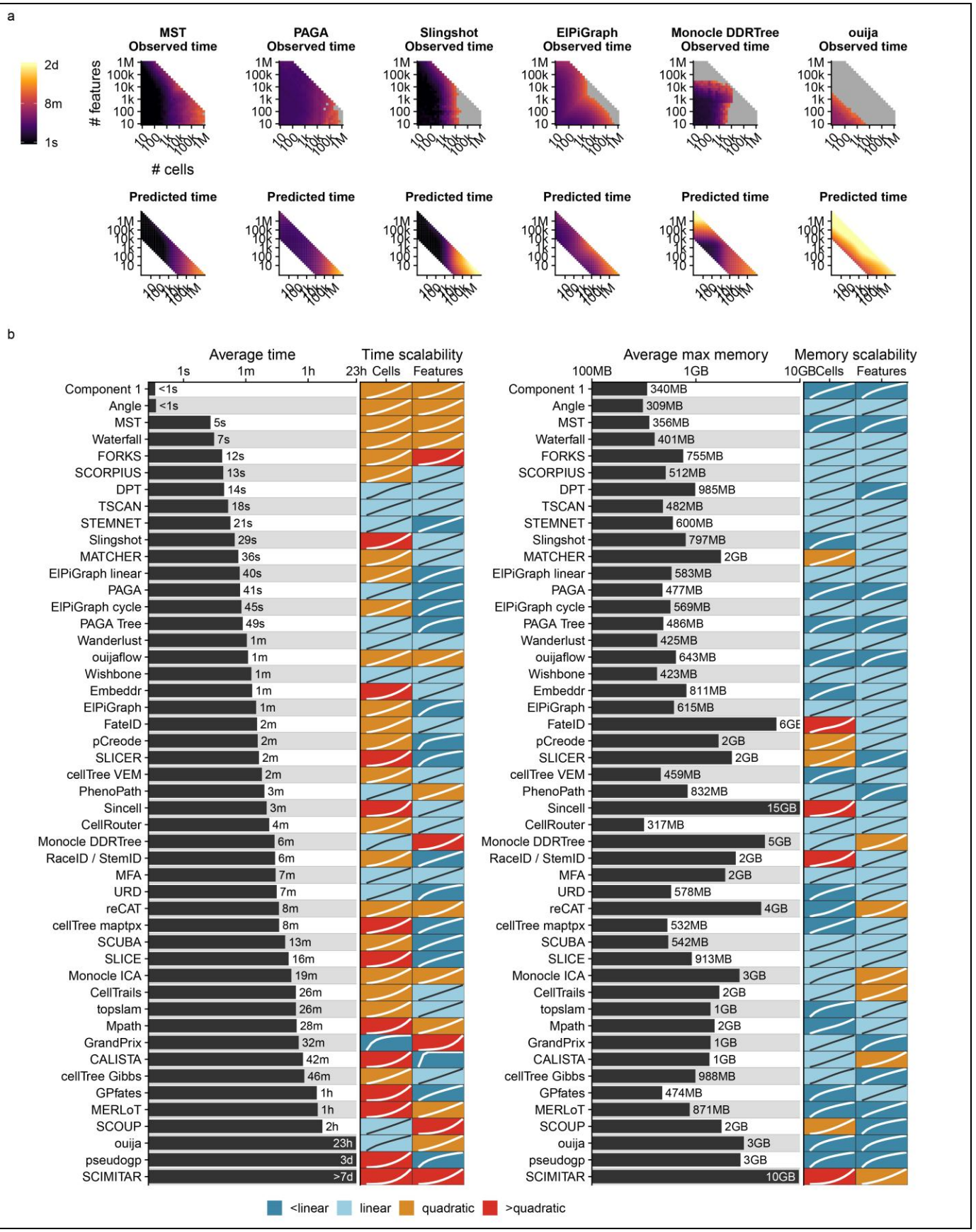
(a) We characterized the methods according to the wrapper type, their required priors, whether the inferred topology is constrained by the algorithm (fixed) or a parameter (param), and the types of inferable topologies. The methods are grouped vertically based on the most complex trajectory type they can infer. (b) The overall results of the evaluation on four criteria: benchmarking using a reference trajectory on real and synthetic data, scalability with increasing number of cells and features, stability across dataset subsamples, and quality of the implementation. (c) Accuracy of trajectory inference methods across metrics, dataset sources and dataset trajectory types. The performance of a method is generally more stable across dataset sources, but very variable depending on the metric and trajectory type. (d) Predicted execution times and memory usage for varying numbers of cells and features (# cells × # features). Predictions were made by training a regression model after running each method on bootstrapped datasets with varying numbers of cells and features. (e) Stability results by calculating the average pairwise similarity between models inferred across multiple runs of the same method. (f) Usability scores of the tool and corresponding manuscript, grouped per category.


a
Dataset source
real/gold
real/silver
synthetic/dyngen
synthetic/dyntoy
synthetic/prosstt
synthetic/splatter
Overall score
Slingshot
PAGA Tree
SCORPIUS
FateID
PAGA
TSCAN
Angle
MST
Waterfall
Embeddr
Component 1
STEMNET
Monocle DDRTree
pCreode
SLICE
MATCHER
Monocle ICA
ElPiGraph linear
cellTree maptpx
PhenoPath
ElPiGraph cycle
topslam
SCUBA
MFA
Wanderlust
Wishbone
DPT
cellTree VEM
Sincell
Mpath
RaceID / StemID
ElPiGraph
ouijaflow
CellTrails
reCAT
URD
FORKS
CellRouter
GPfates
GrandPrix
MERLoT
SLICER
pseudogp
cellTree Gibbs
CALISTA
SCOUP
ouija
SCIMITAR
b
real/silver
synthetic/dyngen
synthetic/splatter
synthetic/dyntoy
synthetic/prosstt
corr = 0.81
corr = 0.73
corr = 0.69
corr = 0.62
corr = 0.53
Overall score on real/gold datasets
Overall score on datasets from source
c
Trajectory type
Cycle
Linear
Convergence
Bifurcation
Multifurcation
Tree
Acyclic graph
Connected graph
Disconnected graph
Overall score
d
PAGA
Slingshot
PAGA Tree
Monocle DDRTree
pCreode
Prediction too simple
Prediction too complex
All trajectory types
Disconnected graph
Connected graph
Acyclic graph
Tree
Multifurcation
Bifurcation
Convergence
Linear
Cycle
Reference trajectory type
Difference in topology size (= # milestones + # edges)
between prediction and reference

Supplementary Figure 3

Accuracy of trajectory inference methods.

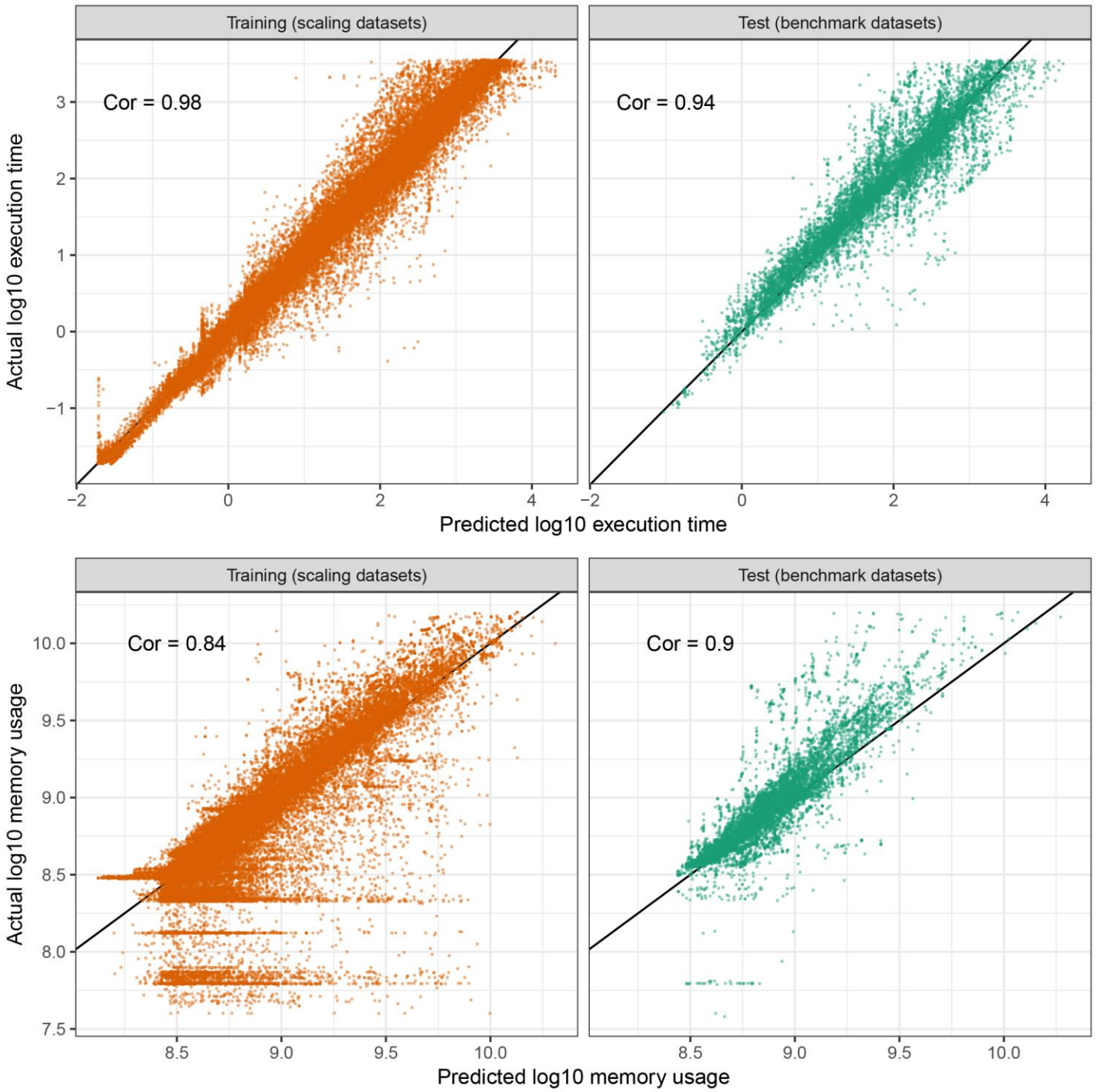
(a) Overall score for all methods across 339 datasets, colored by the source of the datasets. Black line indicates the mean. (b) Similarity between the overall scores of all dataset sources, compared to real datasets with a gold standard, across all methods (n = 46, after filtering out methods that errored too frequently). Shown in the top left is the Pearson correlation. (c) Bias in the overall score towards trajectory types for all methods across 339 datasets. Black line indicates the mean. (d) Distributions of the difference in size between predicted and reference topologies. A positive difference means that the topology predicted by the method is more complex than the one in the reference.


a
MST
Observed time
PAGA
Observed time
Slingshot
Observed time
ElPiGraph
Observed time
Monocle DDRTree
Observed time
ouija
Observed time
2d
8m
1s
features
cells
Predicted time
b
Average time
Time scalability
Average max memory
Memory scalability
Component 1 <1s 340MB
Angle <1s 309MB
MST 5s 356MB
Waterfall 7s 401MB
FORKS 12s 755MB
SCORPIUS 13s 512MB
DPT 14s 985MB
TSCAN 18s 482MB
STEMNET 21s 600MB
Slingshot 29s 797MB
MATCHER 36s 2GB
ElPiGraph linear 40s 583MB
PAGA 41s 477MB
ElPiGraph cycle 45s 569MB
PAGA Tree 49s 486MB
Wanderlust 1m 425MB
ouijaflow 1m 643MB
Wishbone 1m 423MB
Embeddr 1m 811MB
ElPiGraph 1m 615MB
FateID 2m 6GB
pCreode 2m 2GB
SLICER 2m 2GB
cellTree VEM 2m 459MB
PhenoPath 3m 832MB
Sincell 3m 15GB
CellRouter 4m 317MB
Monocle DDRTree 6m 5GB
RaceID / StemID 6m 2GB
MFA 7m 2GB
URD 7m 578MB
reCAT 8m 4GB
cellTree maptpx 8m 532MB
SCUBA 13m 542MB
SLICE 16m 913MB
Monocle ICA 19m 3GB
CellTrails 26m 2GB
topslam 26m 1GB
Mpath 28m 2GB
GrandPrix 32m 1GB
CALISTA 42m 1GB
cellTree Gibbs 46m 988MB
GPfates 1h 474MB
MERLoT 1h 871MB
SCOUP 2h 2GB
ouija 23h 3GB
pseudogp 3d 3GB
SCIMITAR >7d 10GB
<linear
linear
quadratic
>quadratic

Supplementary Figure 4

Scalability of trajectory inference methods.

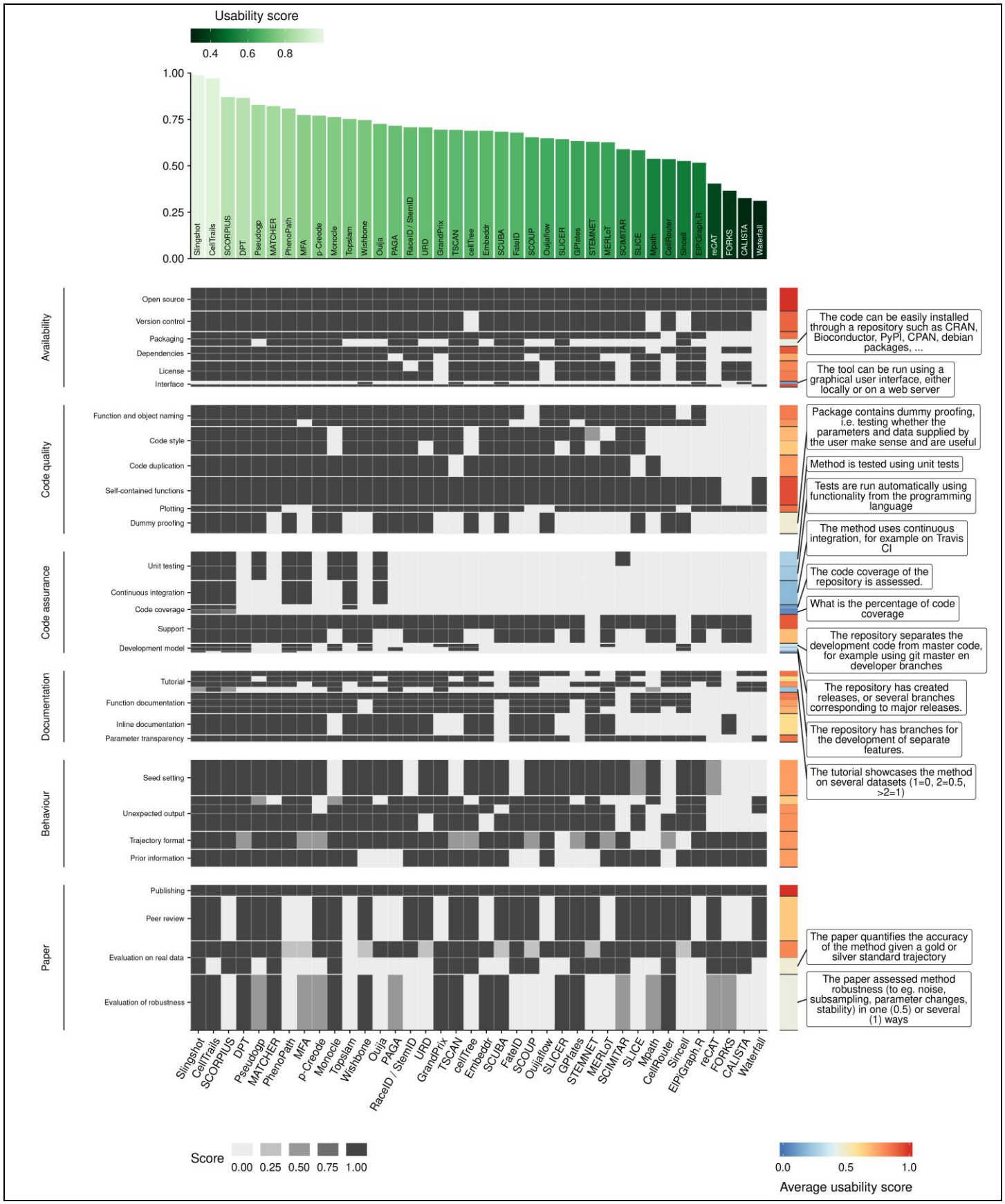
(a) Three examples of average observed running times across five datasets (left) and the predicted running time (right). (b) Overview of the scalability results of all methods, ordered by their average predicted running time from (a). We predicted execution times and memory usage for each method with increasing number of features or cells, and used these values to classify each method into sublinear, linear, quadratic and superquadratic based on the shape of the curve.



Supplementary Figure 5

Agreement between actual values and predictions for execution times and memory usage.

We created a predictive model of the running time and memory usage based on a set of scaling datasets (left), and validated this model based on the similarity of the predictions and actual values on all benchmark datasets (right). Shown are the values for each method and dataset (n = 65618 for training, n = 11939 for test). Top left indicates the Pearson correlation coefficient.


Usability score
0.4
0.6
0.8
1.00
0.75
0.50
0.25
0.00
Slingshot
CellTrails
SCORPIUS
DPT
Pseudogp
MATCHER
PhenoPath
MFA
p-Creode
Monocle
Topslam
Wishbone
Oulja
PAGA
RaceID / StemID
URD
GrandPrix
TSCAN
cellTree
Embeddr
SCUBA
FateID
SCOUP
Oujaflow
SLICER
GPfates
STEMNET
MERLoT
SCIMITAR
SLICE
Mpath
CellRouter
Sincell
EPiGraph.R
reCAT
FORKS
CALISTA
Waterfall
Availability
Open source
Version control
Packaging
Dependencies
License
Interface
Code quality
Function and object naming
Code style
Code duplication
Self-contained functions
Plotting
Dummy proofing
Code assurance
Unit testing
Continuous integration
Code coverage
Support
Development model
Documentation
Tutorial
Function documentation
Inline documentation
Parameter transparency
Behaviour
Seed setting
Unexpected output
Trajectory format
Prior information
Paper
Publishing
Peer review
Evaluation on real data
Evaluation of robustness
The code can be easily installed through a repository such as CRAN, Bioconductor, PyPI, CPAN, debian packages, ...
The tool can be run using a graphical user interface, either locally or on a web server
Package contains dummy proofing, i.e. testing whether the parameters and data supplied by the user make sense and are useful
Method is tested using unit tests
Tests are run automatically using functionality from the programming language
The method uses continuous integration, for example on Travis CI
The code coverage of the repository is assessed.
What is the percentage of code coverage
The repository separates the development code from master code, for example using git master en developer branches
The repository has created releases, or several branches corresponding to major releases.
The repository has branches for the development of separate features.
The tutorial showcases the method on several datasets (1=0, 2=0.5, >2=1)
The paper quantifies the accuracy of the method given a gold or silver standard trajectory
The paper assessed method robustness (to eg. noise, subsampling, parameter changes, stability) in one (0.5) or several (1) ways
Score
0.00 0.25 0.50 0.75 1.00
0.0
0.5
1.0
Average usability score

Supplementary Figure 6

Usability of trajectory inference methods.

Shown is the score given for each method on every item from the usability score sheet (Supplementary Table 3). Each aspect of the quality control was part of a category, and each category was weighted so that it contributed equally to the final quality score. Within each category, each aspect also received a weight depending on how often it was mentioned in a set of papers discussing good practices in tool development and evaluation. This is represented in the plot as the height on the y-axis. Top: Average usability score for each method. Right: The average score of each quality control item. Shown into more detail are those items which had an average score lower than 0.5.

Metrics to compare two trajectories

A trajectory, as defined in our evaluation, is a model with multiple abstractions. The top abstraction is the topology which contains information about the paths each cell can take from their starting point. Deeper abstractions involve the mapping of each cell to a particular branch within this network, and the position (or ordering) of each cells within these branches. Internally, the topology is represented by the milestone network and regions of delayed commitment, the branch assignment and cellular positions are represented by the milestone percentages (**Figure S1**).

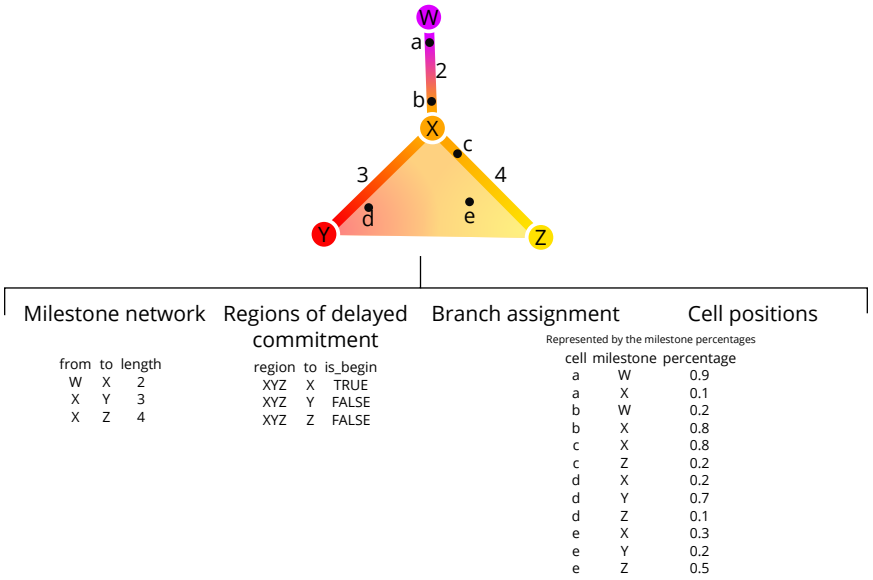


Figure S1: An example trajectory that will be used throughout this section. It contains contains four milestones (W to Z) and five cells (a to e).

Given the multilayered complexity of a trajectory model, it is not trivial to compare the similarity of two trajectory models using only one metric. We therefore sought to use different comparison metrics, each serving a different purpose:

- **Specific metrics** investigate one particular aspect of the trajectory. Such metrics make it possible to find particular weak points for methods, e.g. that a method is very good at ordering but does not frequently find the correct topology. Moreover, having multiple individual metrics allow personalised rankings of methods, for example for users which are primarily interested in using the method correct topology.
- **Application metrics** focus on the quality of a downstream analysis using the trajectory. For example, it measures whether the trajectory can be used to find accurate differentially expressed genes.
- **Overall metrics** should capture all the different abstractions, in other words such metrics measure whether the resulting trajectory has a good topology, that the cells belong to similar branches *and* that they are ordered correctly.

Here, we first describe and illustrate several possible specific, application and overall metrics. Next, we test these metrics on several test cases, to make sure they robustly identify "wrong" trajectory predictions.

All metrics described here were implemented within the *dyneval* R package (https://github.com/dynverse/dyneval).

Metric characterisation and testing

Specific metrics

isomorphic, edgeflip and HIM: Edit distance between two trajectory topologies

We used three different scores to assess the similarity in the topology between two trajectories, iregardless of where the cells were positioned.

For all three scores, we first simplified the topology of the trajectory to make both graph structures comparable:

- As we are only interested in the main structure of the topology without start or end, the graph was made undirected.
- All milestones with degree 2 were removed. For example in the topology A ⇨ B ⇨ C ⇨ D, C ⇨ D, the B milestone was removed
- A linear topology was converted to A ⇨ B ⇨ C
- A cyclical topology such as A ⇨ B ⇨ C ⇨ D or A ⇨ B ⇨ A were all simplified to A ⇨ B ⇨ C ⇨ A

- Duplicated edges such as A ⇨ B, A ⇨ B were decoupled to A ⇨ B, A ⇨ C ⇨ B

The isomorphic score returns 1 if two graphs are isomorphic, and 0 if they were not. For this, we used the used the BLISS algorithm[1], as implemented in the R *igraph* package.

The edgeflip score was defined as the minimal number of edges which should be added or removed to convert one network into the other, divided by the total number of edges in both networks. This problem is equivalent to the maximum common edge subgraph problem, a known NP-hard problem without a scalable solution[2]. We implemented a branch and bound approach for this problem, using several heuristics to speed up the search:

- First check all possible edge additions and removals corresponding to the number of different edges between the two graphs.
- For each possible solution, first check whether:
  - The maximal degree is the same
  - The minimal degree is the same
  - All degrees are the same after sorting
- Only then check if the two graphs are isomorphic as described earlier.
- If no solution is found, check all possible solutions with two extra edge additions/removals.

The HIM metric (Hamming-Ipsen-Mikhailov distance)[3] which was adopted from the R nettools package (https://github.com/filosi/nettools). It uses an adjacency matrix which was weighted according to the lengths of each edges within the milestone network. Conceptually, HIM is a linear combination of:

- The normalised Hamming distance[4], which calculates the distance between two graphs by matching individual edges in the adjacency matrix, but disregards overall structural similarity.
- The normalised Ipsen-Mikhailov distance[5], which calculates the overall distance of two graphs based on matches between its degree and adjacency matrix, while disregarding local structural similarities. It requires a γ parameter, which is usually estimated based on the number of nodes in the graph, but which we fixed at 0.1 so as to make the score comparable across different graph sizes.

We compared the three scores on several common topologies (**Figure S2**). While conceptually very different, the edgeflip and HIM still produce similar scores (**Figure S2b**). The HIM tends to punish the detection of cycles, while the edgeflip is more harsh for differences in the number of bifurcations (**Figure S2b**). The main difference however is that the HIM takes into account edge lengths when comparing two trajectories, as illustrated in (**Figure S2c**). Short "extra" edges in the topology are less punished by the HIM than by the edgeflip.

To summarise, the different topology based scores are useful for different scenarios:

- If the two trajectories should only be compared when the topology is exactly the same, the isomorphic should be used.
- If it is important that the topologies are similar, but not necessarily isomorphic, the edgeflip is most appropriate.
- If the topologies should be similar, but shorter edges should not be punished as hard as longer edges, the HIM is most appropriate.

$F1_{branches}$ and $F1_{milestones}$: Comparing how well the cells are clustered in the trajectory

Perhaps one of the simplest ways to calculate the similarity between the cellular positions of two topologies is by mapping each cell to its closest milestone *or* branch (**Figure S3**). These clusters of cells can then be compared using one of the many external cluster evaluation measures[6]. When selecting a cluster evaluation metric, we had two main conditions:

- Because we allow methods to filter cells in the trajectory, the metric should be able to handle "non-exhaustive assignment", where some cells are not assigned to any cluster.
- The metric should give each cluster equal weight, so that rare cell stages are equally important as large stages.

The F1 score between the Recovery and Relevance is a metric which conforms to both these conditions. This metric will map two clustersets by using their shared members based on the Jaccard similarity. It then calculates the Recovery as the average maximal Jaccard for every cluster in the first set of clusters (in our case the reference trajectory). Conversely, the Relevance is calculated based on the average maximal similarity in the second set of clusters (in our case the prediction). Both the Recovery and Relevance are then given equal weight in a harmonic mean (F1). Formally, if C and C' are two cell clusters:

Jaccard$(c, c') = \frac{|c \cap c'|}{|c \cup c'|}$

Recovery $= \frac{1}{|C|} \sum_{c \in C} \max_{c' \in C'}$ Jaccard(c, c')

Relevance $= \frac{1}{|C'|} \sum_{c' \in C'} \max_{c \in C}$ Jaccard(c, c')

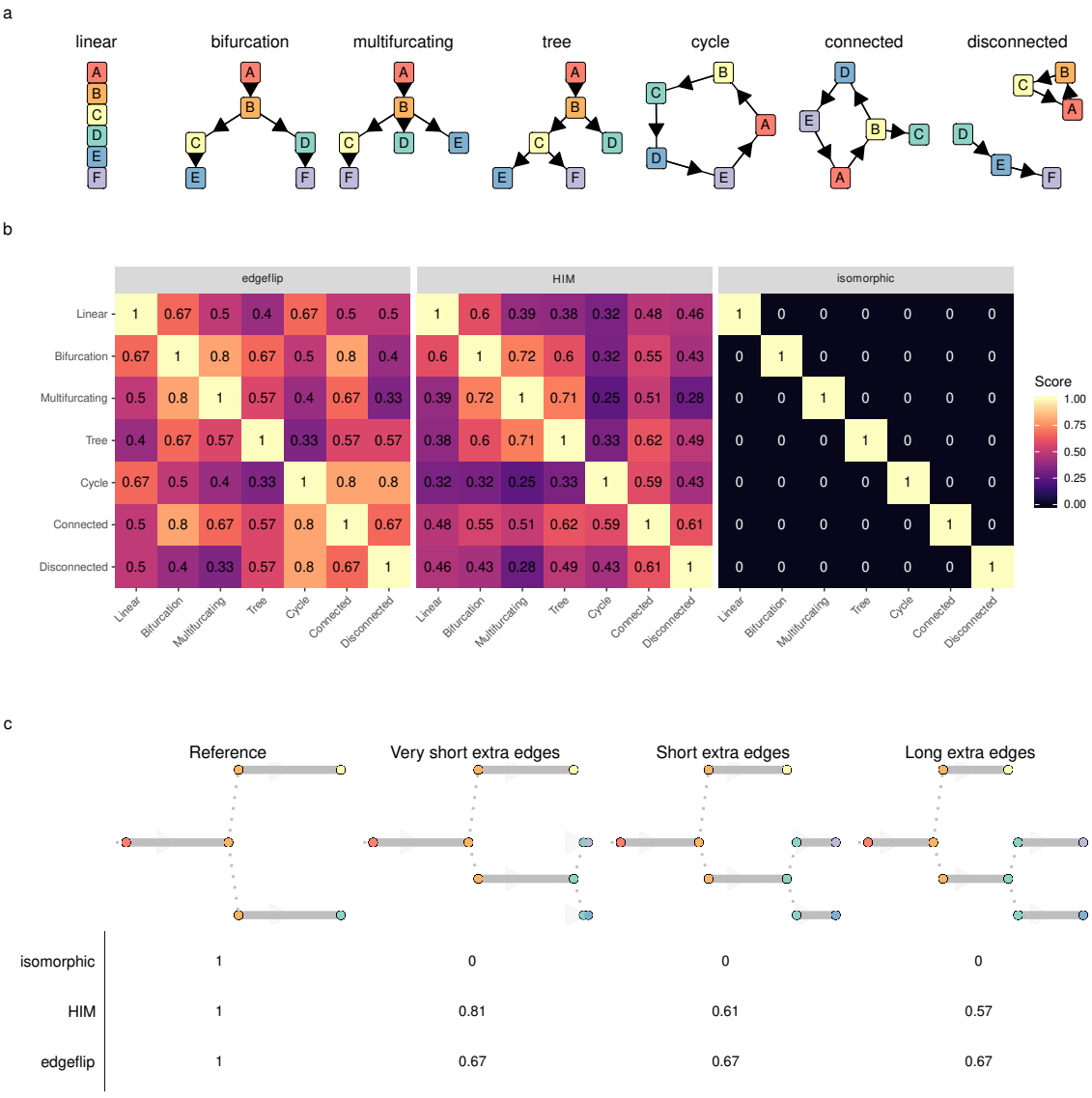


Figure S2: Showcase of three metrics to evaluate topologies: isomorphic, edgeflip and HIM. (a) The used topologies. (b) The scores when comparing each pair of trajectory types. (c) Four datasets in which aan extra edge is added and made progressively longer. This shows how the HIM can take into account edge lengths.

$$\text{F1} = \frac{2}{\frac{1}{\text{Recovery}} + \frac{1}{\text{Relevance}}}$$

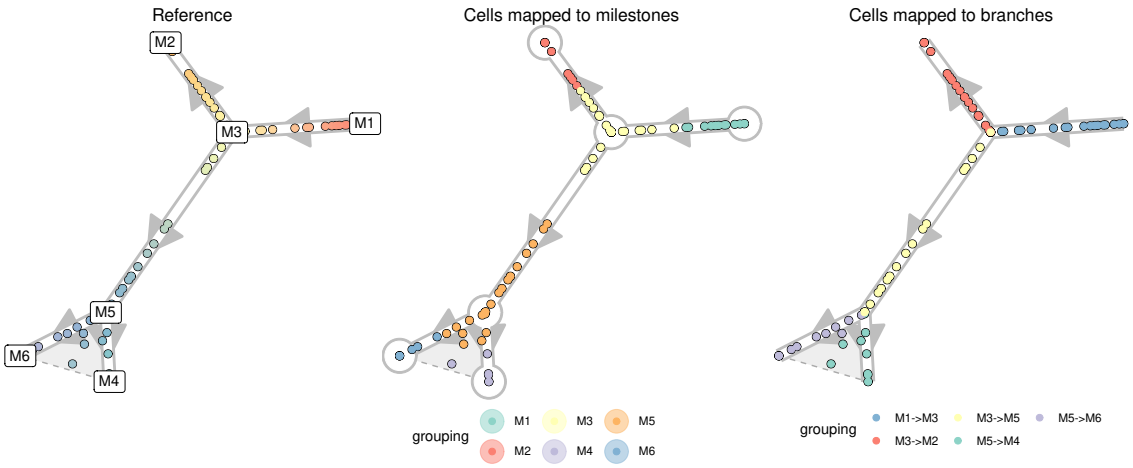


Figure S3: Mapping cells to their closest milestone or branch for the calculation of the $F1_{milestones}$ and $F1_{branches}$. To calculate the $F1_{milestones}$, cells are mapped towards the nearest milestone, i.e. the milestone with the highest milestone percentage. For the $F1_{branches}$, the cells are mapped to the closest edge.

cor_{dist}: Correlation between geodesic distances

When the position of a cell is the same in both the reference and the prediction, its *relative* distances to all other cells in the trajectory should also be the same. This observation is the basis for the cor_{dist} metric.

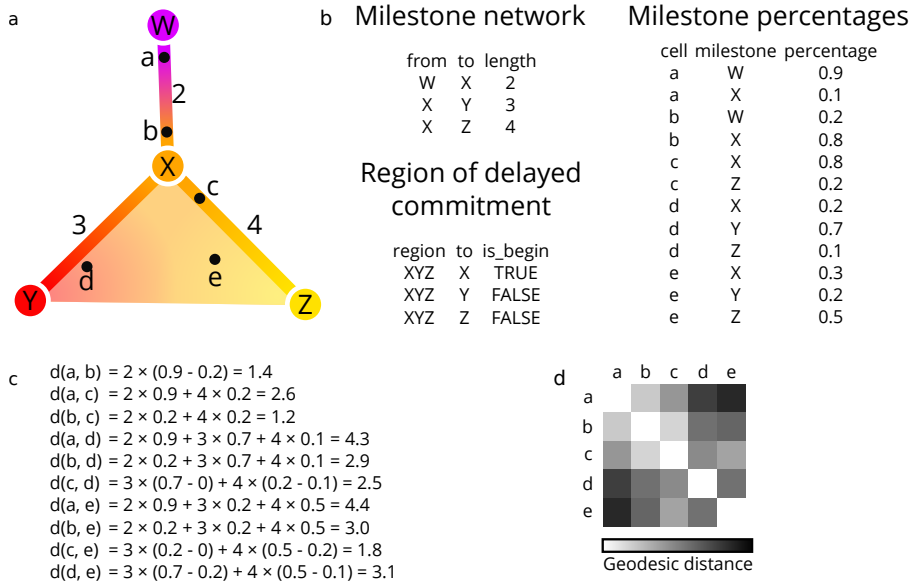


Figure 4: The calculation of geodesic distances on a small example trajectory. a) A toy example containing four milestones (W to Z) and five cells (a to e). b) The corresponding milestone network, milestone percentages and regions of delayed commitment, when the toy trajectory is converted to the common trajectory model. c) The calculations made for calculating the pairwise geodesic distances. d) A heatmap representation of the pairwise geodesic distances.

The geodesic distance is the distance a cell has to go through the trajectory space to get from one position to another. The way this distance is calculated depends on how two cells are positioned, showcased by an example in **Figure 4**:

- **Both cells are on the same edge in the milestone network.** In this case, the geodesic distance is defined as the product of the difference in milestone percentages and the length of their shared edge. For cells a and b in the example, $d(a, b)$ is equal to $1 \times (0.9 - 0.2) = 0.7$.
- **Cells reside on different edges in the milestone network.** First, the distance of the cell to all its nearby milestones is calculated, based on its percentage within the edge and the length of the edge. These distances in combination with the milestone network are used to calculate the shortest path distance between the two cells. For cells a and c in the example, $d(a, X) = 1 \times 0.9$ and $d(c, X) = 3 \times 0.2$, and therefore $d(a, c) = 1 \times 0.9 + 3 \times 0.2$.

The geodesic distance can be easily extended towards cells within regions of delayed commitment. When both cells are part of the same region of delayed commitment, the geodesic distance was defined as the manhattan distances between the milestone percentages weighted by the lengths from the milestone network. For cells d and e in the example, $d(d, e)$ is equal to $0 \times (0.3 - 0.2) + 2 \times (0.7 - 0.2) + 3 \times (0.4 - 0.1) = 1.9$. The distance between two cells where only one is part of a region of delayed commitment is calculated similarly to the previous paragraph, by first calculating the distance between the cells and their neighbouring milestones first, then calculating the shortest path distances between the two.

Calculating the pairwise distances between cells scales quadratically with the number of cells, and would therefore not be scaleable for large datasets. For this reason, a set of waypoint cells are defined *a priori*, and only the distances between the waypoint cells and all other cells is calculated, in order to calculate the correlation of geodesic distances of two trajectories (**Figure S5a**). These cell waypoints are determined by viewing each milestone, edge and region of delayed commitment as a collection of cells. We do stratified sampling from each collection of cells by weighing them by the total number of cells within that collection. For calculating the cor_{dist} between two trajectories, the distances between all cells and the union of both waypoint sets is computed.

To select the number of cell waypoints, we need to find a trade-off between the accuracy versus the time to calculate cor_{dist}. To select an optimal number of cell waypoints, we used the synthetic dataset with the most complex topology, and determined the cor_{dist} at different levels of both cell shuffling and number of cell waypoints (**Figure S5b**). We found that using cell waypoints does not induce a systematic bias in the cor_{dist}, and that its variability was relatively minimal when compared to the variability between different levels of cell shuffling when using 100 or more cell waypoints.

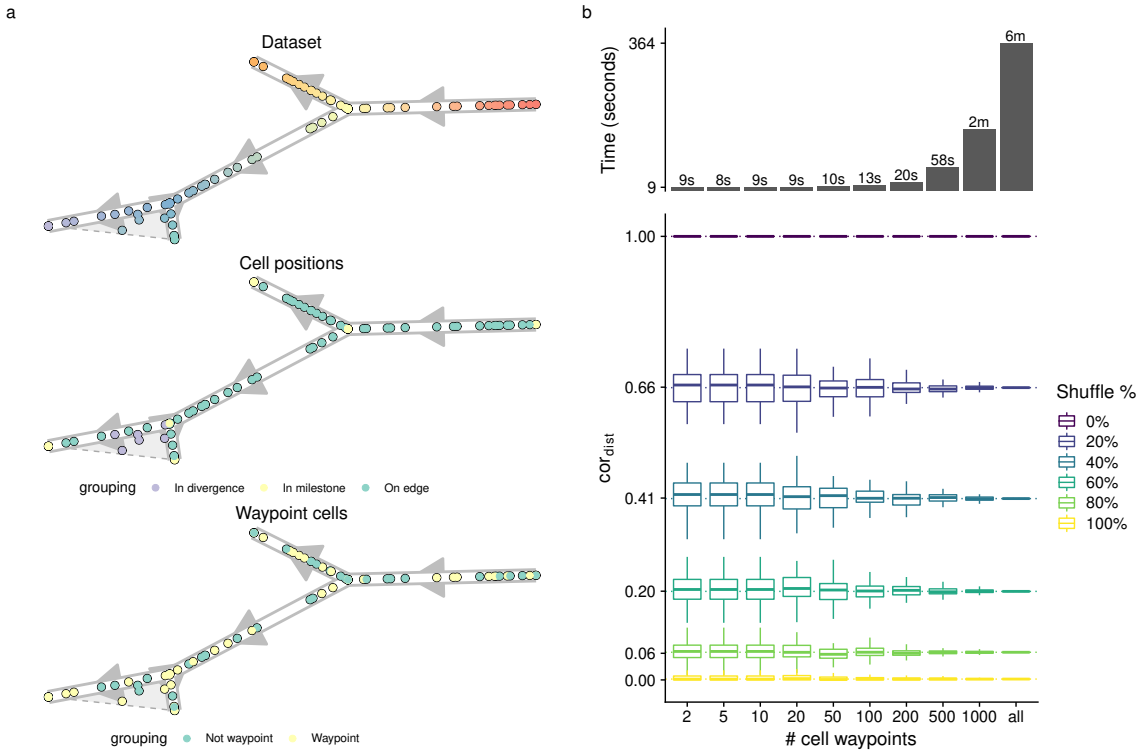


Figure S5: Determination of cell waypoints a) Illustration of the stratified cell sampling using an example dataset (top). Each milestone, edge between two milestones and region of delayed commitment is seen as a collection of cells (middle), and the number of waypoints (100 in this case) are divided over each of these collection of cells (bottom). b) Accuracy versus time to calculate cor_{dist}. Shown are distributions over 100 random waypoint samples. The upper whisker of the boxplot extends from the hinge (75

Although the cor_{dist}'s main characteristic is that it looks at the positions of the cells, other features of the trajectory are also (partly) captured. To illustrate this, we used the geodesic distances themselves as input for dimensionality reduction (**Figure S6**) with varying topologies. This reduced space captures the original trajectory structure quite well, including the overall topology and branch lengths. Only some structures, not easily visualisabe

$NMSE_{rf}$ and $NMSE_{lm}$: Using the positions of the cells within one trajectory to predict the cellular positions in the other trajectory

An alternative approach to detect whether the positions of cells are similar between two trajectories, is to use the positions of one trajectory to predict the positions within the other trajectory. If the cells are at similar positions in the trajectory (relative to its nearby cells), the prediction error should be low.

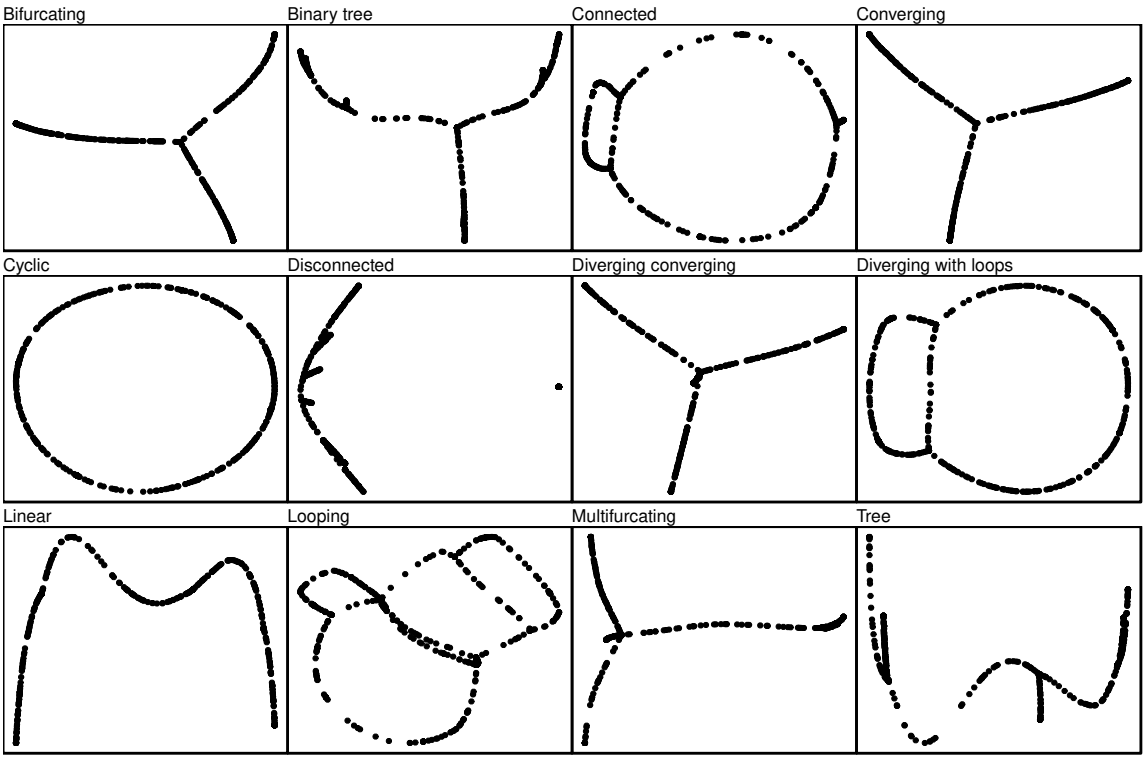


Figure S6: The geodesic distances can be used to reconstruct the original trajectory structure We generated different toy trajectory datasets with varying topologies and calculated the geodesic distances between all cells within the trajectory. We then used these distances as input for classical multidimensional scaling. This shows that the geodesic distances do not only contain information regarding the cell's positions, but also information on the lengths and wiring of the topology.

Specifically, we implemented two metrics which predict the milestone percentages from the reference by using the predicted milestone percentages as features (**Figure S7**). We did this with two regression methods, linear regression (*lm*, using the R `lm` function) and Random Forest (*rf*, implemented in the *ranger* package[7]). In both cases, the accuracy of the prediction was measured using the Mean Squared error (MSE), in the case of Random forest we used the out-of-bag mean-squared error. Next, we calculated MSE_{worst} equal to the MSE when predicting all milestone percentages as the average. We used this to calculate the normalised mean squared error as $NMSE = 1 - \frac{MSE}{MSE_{worst}}$. We created a regression model for every milestone in the gold standard, and averaged the $NMSE$ values to finally obtain the $NMSE_{rf}$ and $NMSE_{lm}$ scores.

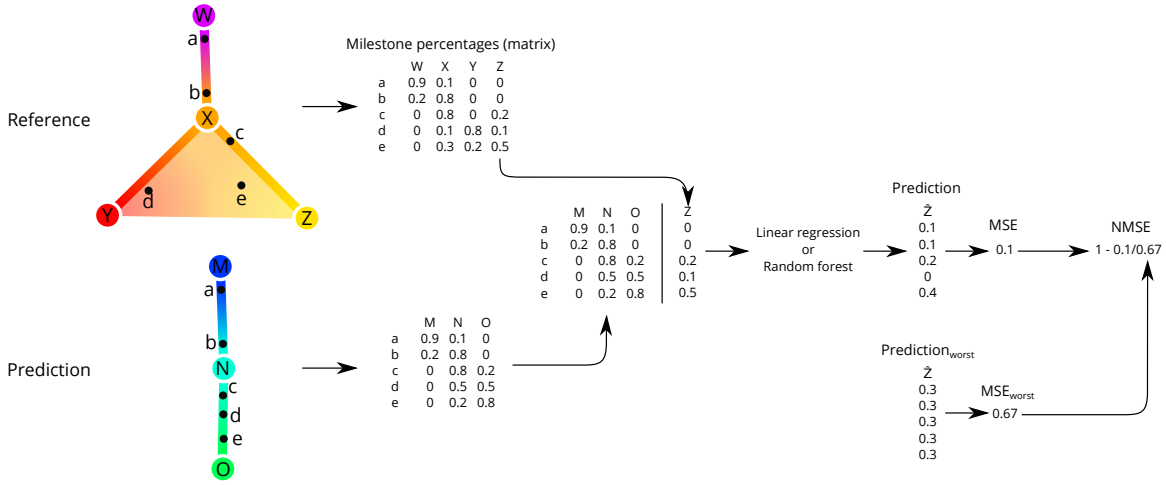


Figure S7: The calculation of $NMSE_{lm}$ distances on a small example trajectory. The milestone percentages of the reference are predicted based on the milestone percentages of the prediction, using regression models such as linear regression or random forests. The predicted trajectory is then scored by comparing the mean-squared error (MSE° of this regression model with the baseline MSE where the prediction is the average milestone percentage

Application metrics

Although most metrics described above already assess some aspects directly relevant to the user, such as whether the method is good at finding the right topology, these metrics do not assess the quality of downstream analyses and hypotheses which can be generated from these models.

cor_{features} and $wcor_{\text{features}}$: The accuracy of dynamical differentially expressed features/genes.

Perhaps the main advantage of studying cellular dynamic processes using single-cell -omics data is that the dynamics of gene expression can be studied for the whole transcriptome. This can be used to construct other models such as dynamic regulatory networks and gene expression modules. Such analyses rely on a "good-enough" cellular ordering, so that it can be used to identify dynamical differentially expressed genes.

To calculate the cor_{features} we used Random forest regression to rank all the features according to their importance in predicting the positions of cells in the trajectory. More specifically, we first calculated the geodesic distances for each cell to all milestones in the trajectory. Next, we trained a Random Forest regression model (implemented in the R *ranger* package[7], https://github.com/imbs-hl/ranger) to predict these distances for each milestone, based on the expression of genes within each cell. We then extracted feature importances using the Mean Decrease in Impurity (`importance = 'impurity'` parameter of the `ranger` function), as illustrated in (**Figure S8**). The overall importance of a feature (gene) was then equal to the mean importance over all milestones. Finally, we compared the two rankings by calculating the Pearson correlation, with values between -1 and 0 clipped to 0.

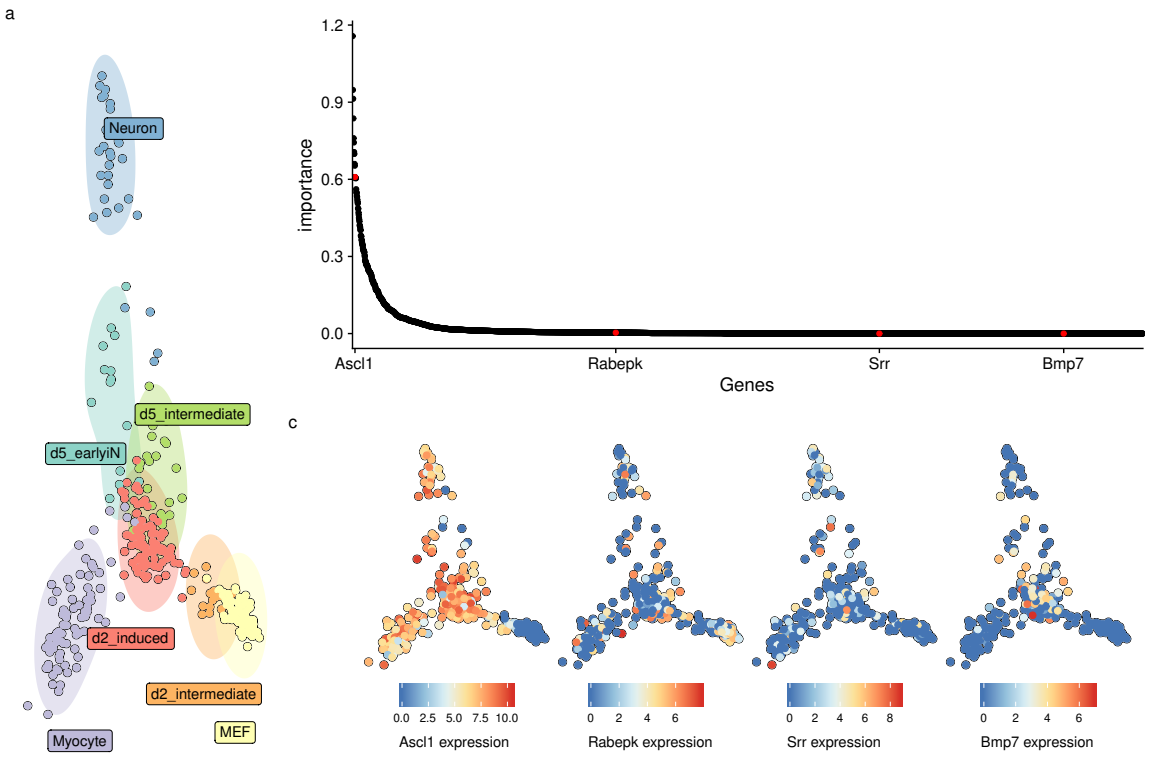


Figure S8: An illustration of ranking features based on their importance in a trajectory. (a) A MDS dimensionality reudction of a real dataset in which mouse embryonic fibroblasts (MEF) differentiate into Neurons and Myocytes. (b) The ranking of feature importances from high to low. The majority of features have a very low importance. (c) Some examples, which were also highlighted in b. Higher features in the ranking are clearly specific to certain parts of the trajectory, while features lower on the ranking have a more dispersed expression pattern.

Random forest regression has two main hyperparameters. The number of trees to be fitted (`num_trees` parameter) was fixed to 10000 to provide accurate and stable estimates of the feature importance (**Figure S9**). The number of features on which can be split (`mtry` parameter) was set to 1% of all available features (instead of the default square-root of the number of features), as to make sure that predictive but highly correlated features, omnipresent in transcriptomics data, are not suppressed in the ranking.

For most datasets, only a limited number of features will be differentially expressed in the trajectory. For example, in the dataset used in **Figure S9** only the top 10%-20% show a clear pattern of differential expression. The correlation will weight each of these features equally, and will therefore give more weight to the bottom, irrelevant features. To prioritise the top

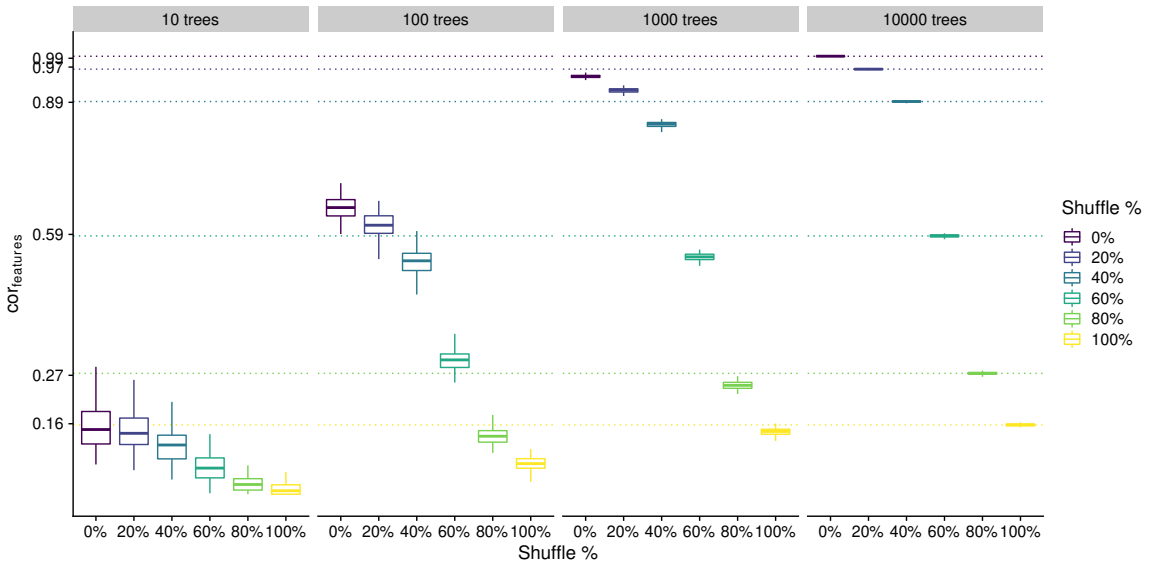


Figure S9: Effect of the number of trees parameter on the accuracy and variability of the cor_{features}. We used the dataset from **Figure S8** and calculated the cor_{features} after shuffling a percentage of cells.

differentially expressed features, we also implemented the $wcor_{\text{features}}$, which will weight the correlation using the feature importance scores in the reference so that the top features have relatively more impact on the score (**Figure S10**).

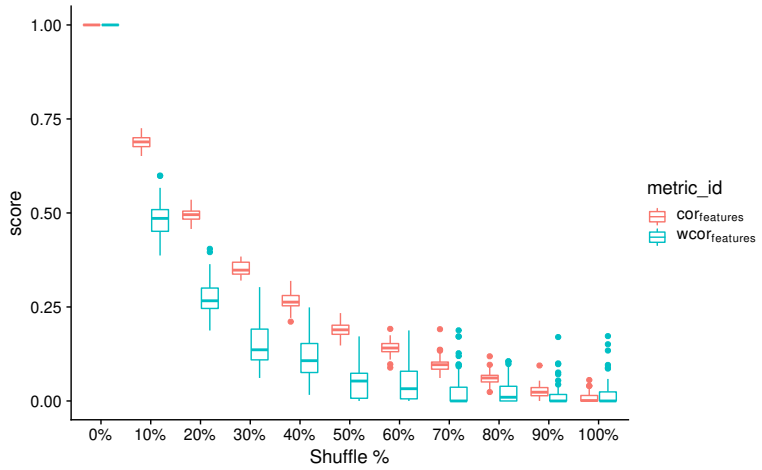


Figure S10: Effect of weighting the features based on their feature importance in the reference. We used the same dataset as in **Figure S8**, and calculated the cor_{features} after shuffling a percentage of cells.

Metric conformity

Although most metrics described in the previous section make sense intuitively, this does not necessarily mean that these metrics are robust and will generate reasonable results when used for benchmarking. This is because different methods and datasets will all lead to a varied set of trajectory models:

- Real datasets have all cells grouped onto milestones
- Some methods place all cells in a region of delayed commitment, others never generate a region of delayed commitment
- Some methods always return a linear trajectory, even if a bifurcation is present in the data
- Some methods filter cells

A good metric, especially a good overall metric, should work in all these circumstances. To test this, we designed a set of rules to which a good metric should conform, and assessed empirically whether a metric conforms to these rules.

We generated a panel of toy datasets (using our *dyntoy* package, https://github.com/dynverse/dyntoy) with all possible combinations of:

- # cells: 10, 20, 50, 100, 200 and 500
- # features: 200
- topologies: linear, bifurcation, multifurcating, tree, cycle, connected graph and disconnected graph
- Whether cells are placed on the milestones (as in real data) or on the edges/regions of delayed commitment between the milestones (as in synthetic data)

We then perturbed the trajectories in these datasets in certain ways, and tested whether the scores follow an expected pattern. An overview of the conformity of every metric is first given in **Table S1**. The individual rules and metric behaviour are discussed more into detail after that.

name	cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
Same score on identity	✔	✖	✔	✔	✔	✔	✔	✖	✔	✔	✔
Local cell shuffling	✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔
Edge shuffling	✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔
Local and global cell shuffling	✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔
Changing positions locally and/or globally	✔	✔	✔	✖	✖	✖	✔	✔	✖	✖	✔
Cell filtering	✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔
Removing divergence regions	✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔
Move cells to start milestone	✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔
Move cells to closest milestone	✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔
Length shuffling	✔	✖	✔	✖	✔	✖	✖	✖	✖	✔	✔
Cells into small subedges	✖	✔	✖	✔	✔	✔	✔	✖	✔	✔	✔
New leaf edges	✔	✔	✖	✔	✔	✔	✖	✖	✔	✔	✔
New connecting edges	✔	✔	✖	✔	✔	✔	✔	✔	✔	✔	✔
Changing topology and cell position	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖	✔
Bifurcation merging	✔	✖	✔	✔	✔	✔	✔	✔	✔	✔	✔
Bifurcation merging and changing cell positions	✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔
Bifurcation concatentation	✔	✖	✔	✔	✔	✔	✔	✔	✔	✔	✔
Cycle breaking	✔	✖	✔	✔	✔	✔	✔	✔	✖	✔	✔
Linear joining	✔	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔
Linear splitting	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Change of topology	✔	✖	✖	✔	✔	✔	✔	✔	✖	✔	✔
Cells on milestones vs edges	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table S1: Overview of whether a particular metric conforms to a particular rule

Rule 1: Same score on identity

Description: The score should be approximately the same when comparing the trajectory to itself.

A metric conforms to this rule if: $0.99 \leqslant score \leqslant 1$.

Conclusion(s): Metrics which contain some stochasticity (random forest based metrics in particular), usually do not conform to this rule, even though their scores are still consistently high.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✖	✔	✔	✔	✔	✔	✖	✔	✔	✔

Table S2: Which metrics conform to rule 1.

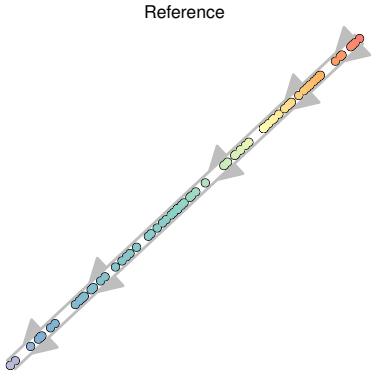


Figure S11: Example trajectory that was used to assess this rule.

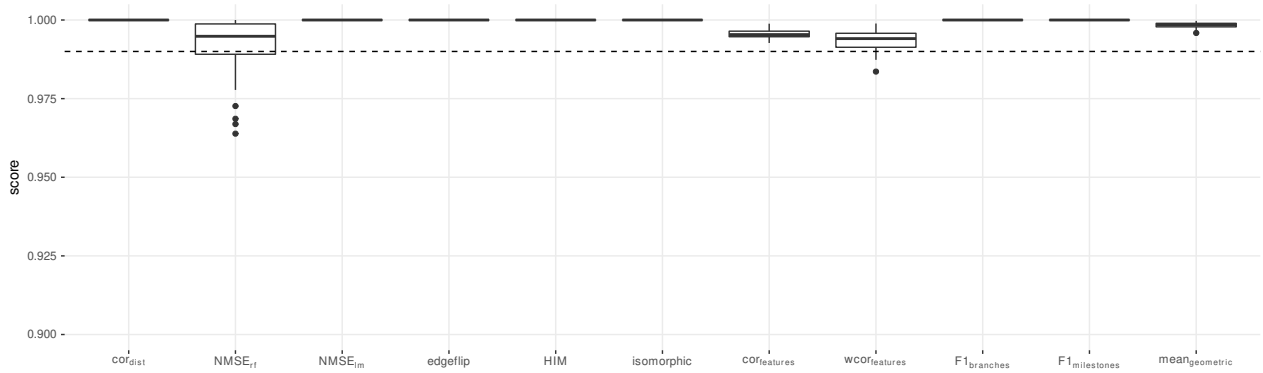
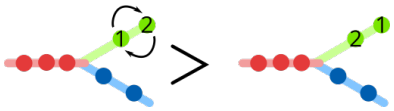


Figure S12: Score values of the different metrics across 42 datasets.

Rule 2: Local cell shuffling



Description: Shuffling the positions of cells within each edge should lower the score. This is equivalent to changing the cellular position locally.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): Metrics which do not look at the cellular positioning, or group the cells within branches or milestones, do not conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔

Table S3: Which metrics conform to rule 2.

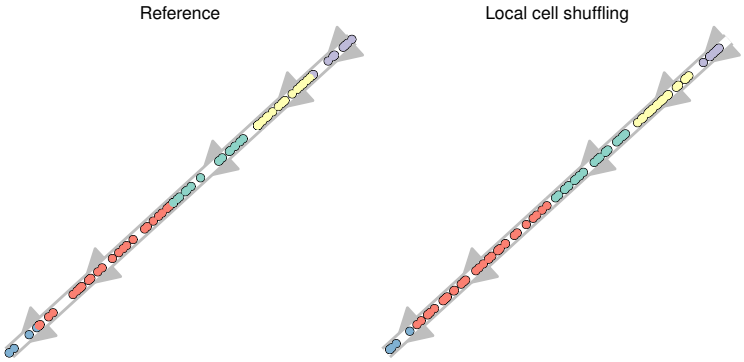


Figure S13: Example dataset before and after perturbation.

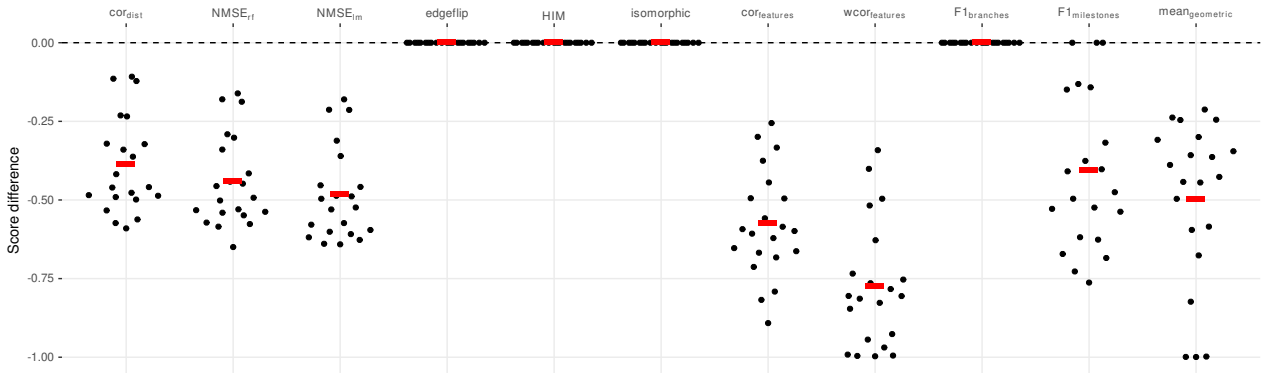


Figure S14: Differences in scores of 231 datasets before and after perturbation. Red bar gives the mean.

Rule 3: Edge shuffling

Description: Shuffling the edges in the milestone network should lower the score. This is equivalent to changing the cellular positions only globally.

A metric conforms to this rule if: $monotonic\left(\textit{shuffled edges}, \overline{score}_{\textit{shuffled edges}}\right)$.

Conclusion(s): Metrics which only look at the topology do not conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔

Table S4: Which metrics conform to rule 3.

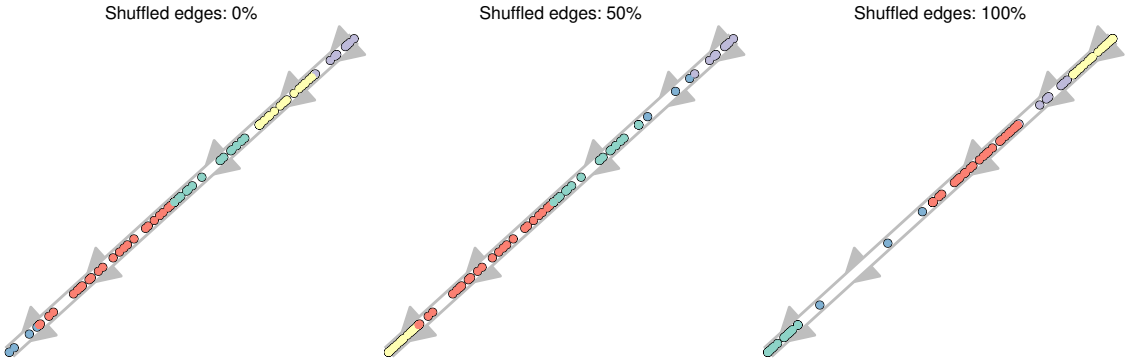


Figure S15: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

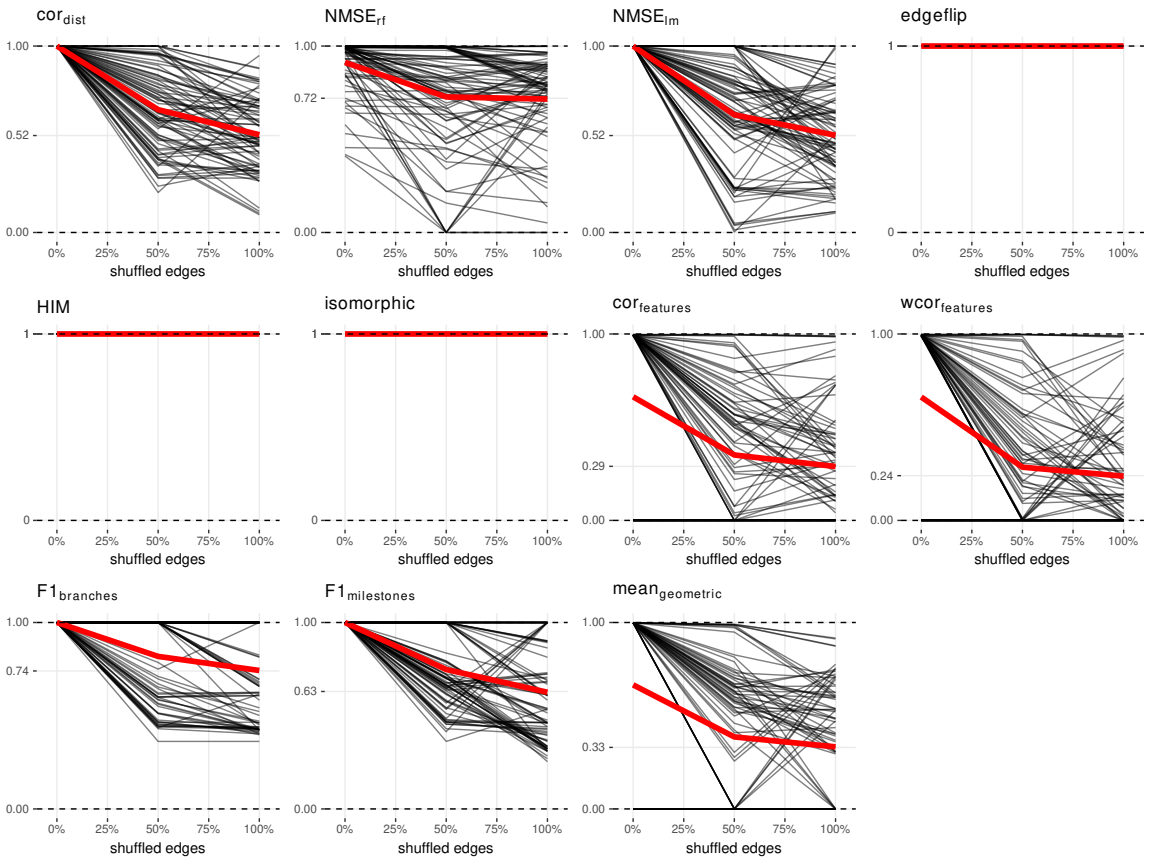


Figure S16: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 4: Local and global cell shuffling

Description: Shuffling the positions of cells should lower the score. This is equivalent to changing the cellular position both locally and globally.

A metric conforms to this rule if: $monotonic\left(\textit{shuffled cells}, \overline{score}_{\textit{shuffled cells}}\right)$.

Conclusion(s): Most metrics that look at the position of each cell conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔

Table S5: Which metrics conform to rule 4.

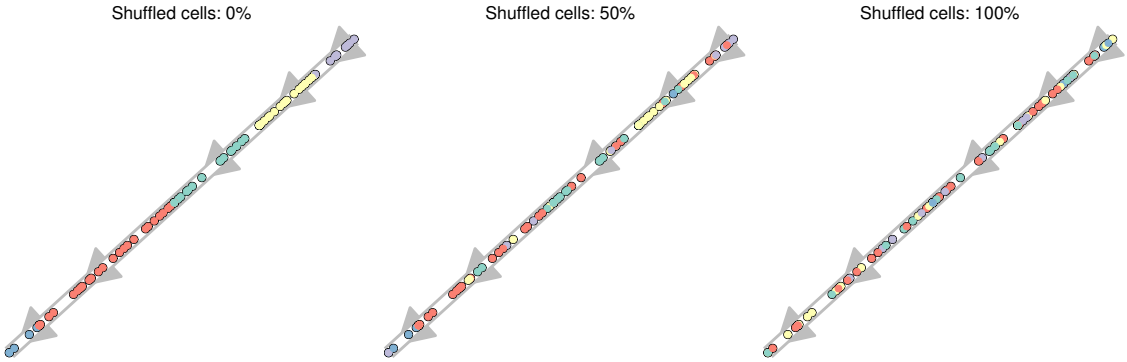


Figure S17: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

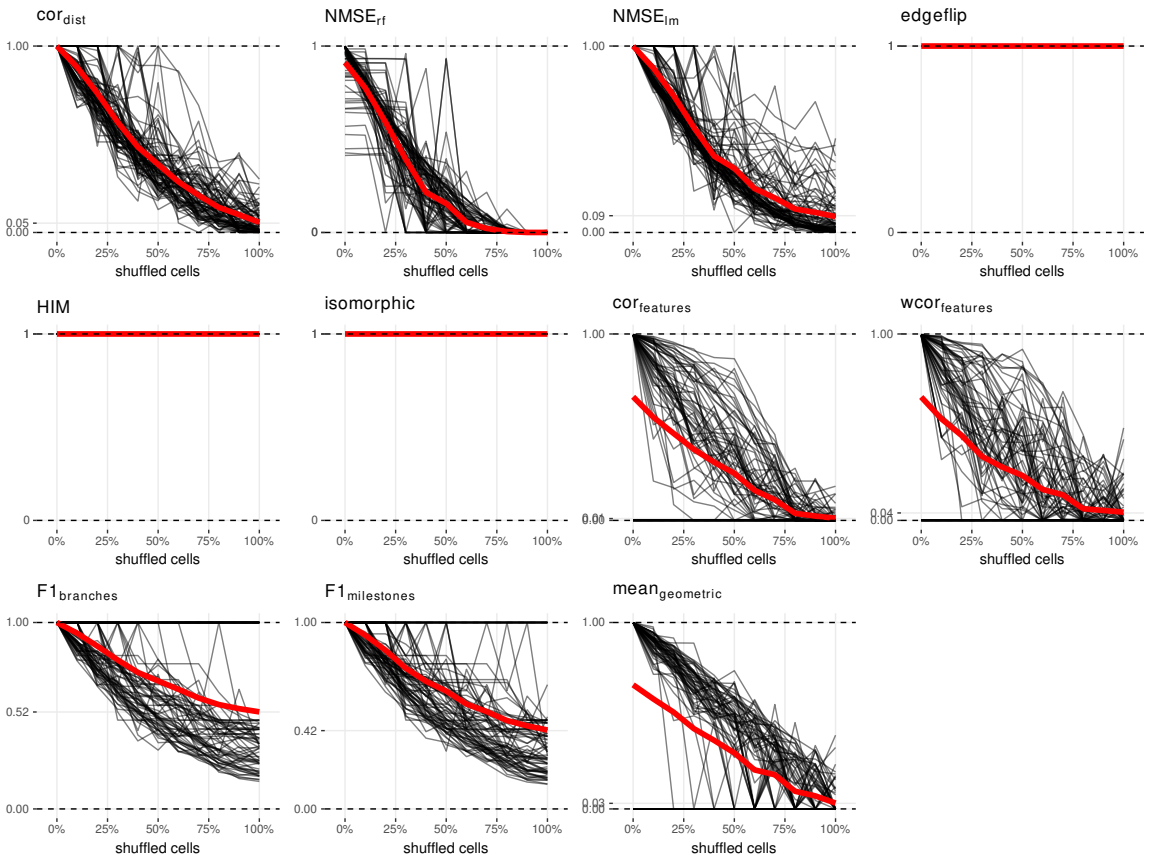
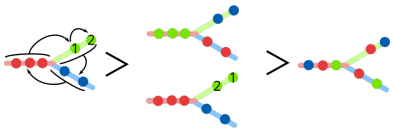


Figure S18: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 5: Changing positions locally and/or globally



Description: Changing the cellular position locally AND globally should lower the score more than any of the two individually.

A metric conforms to this rule if: $score_{identity} > score_a \wedge score_{identity} > score_b \wedge score_a > score_{a+b} \wedge score_b > score_{a+b}$.

Conclusion(s): Because the topology remains the same, the topology scores do not conform to this rule. Also the clustering based scores have some difficulties with this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✔	✔	✖	✖	✖	✔	✔	✖	✖	✔

Table S6: Which metrics conform to rule 5.

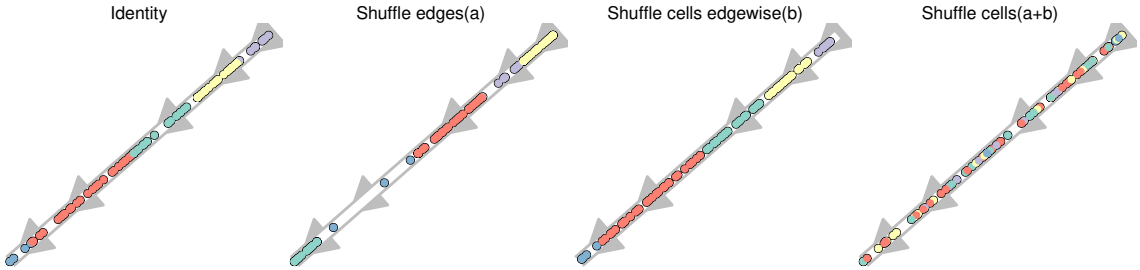
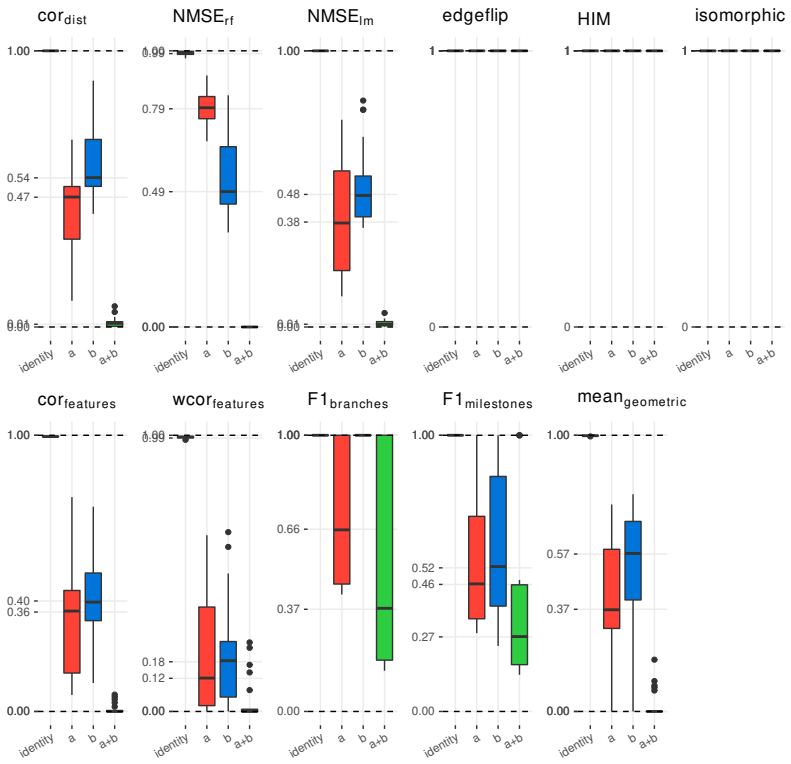


Figure S19: Example dataset before perturbation (identity), with any of the two perturbations (a and b) and both perturbations combined (a+b).

\begin{myfigure}{!htbp}



\textbf{**Figure S20**: Score values before perturbation (identity), with any of the two perturbations (a and b) and both perturbations combined (a+b). The upper whisker of the boxplot extends from the hinge (75% percentile) to the largest

value, no further than 1.5× the IQR from the hinge. The lower whisker extends from the hinge (25% percentile) to the smallest value, at most 1.5× the IQR of the hinge. We used 84 different datasets.}

\end{myfigure}

Rule 6: Cell filtering

Description: Removing cells from the trajectory should lower the score.

A metric conforms to this rule if: $monotonic\,(\textit{Filtered cells}, \overline{score}_{\textit{Filtered cells}})$.

Conclusion(s): Metrics which look at the topology do not conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔

Table S7: Which metrics conform to rule 6.

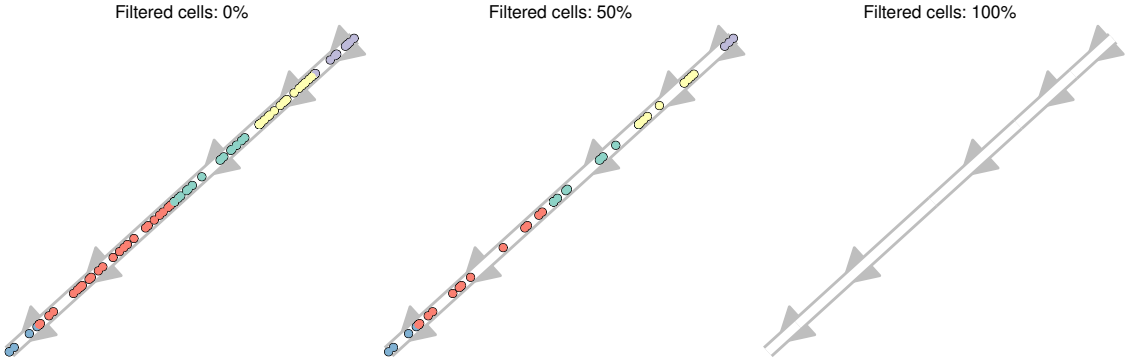


Figure S21: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

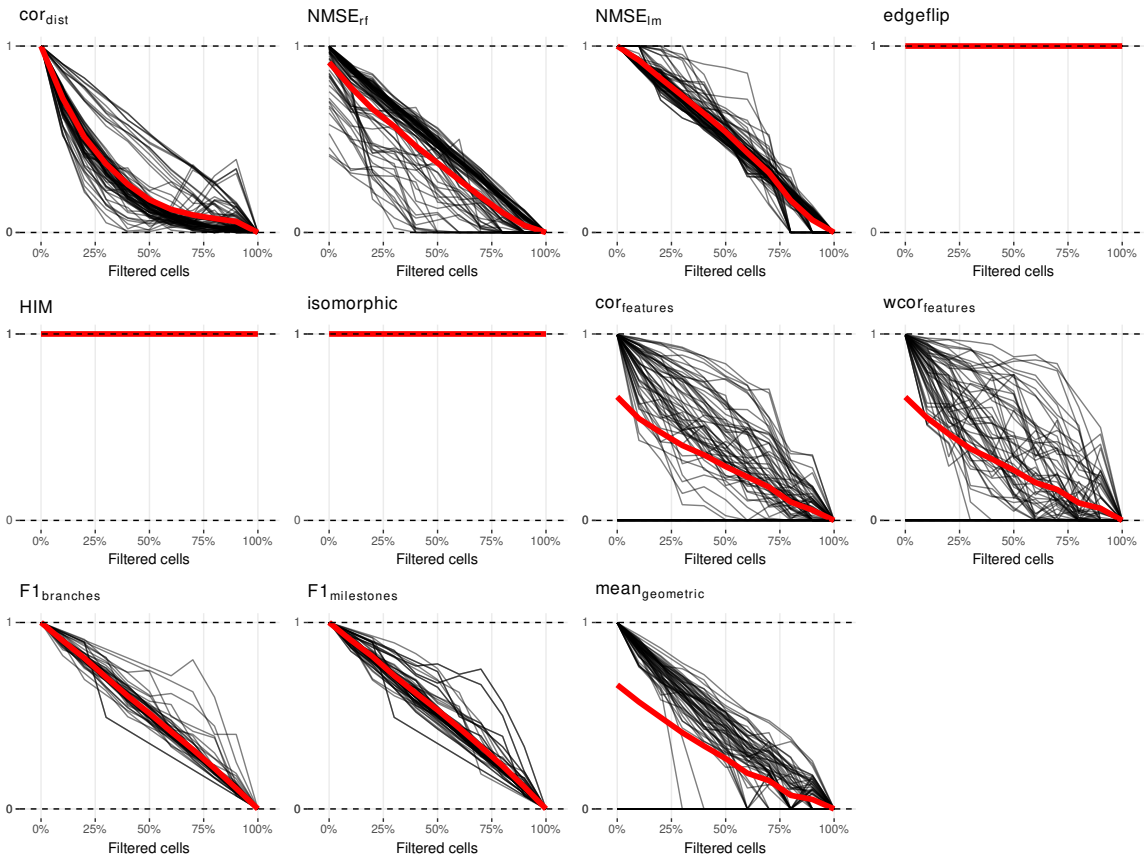


Figure S22: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 7: Removing divergence regions

Description: Removing divergence regions should lower the score.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): Both $F1_{branches}$ and edgeflip fail here because neither the topology nor the branch assignment changes. Moreover, the decreases in score are relatively minor for all metrics, given that the impact of the positions of the cells is only minimal.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔

Table S8: Which metrics conform to rule 7.

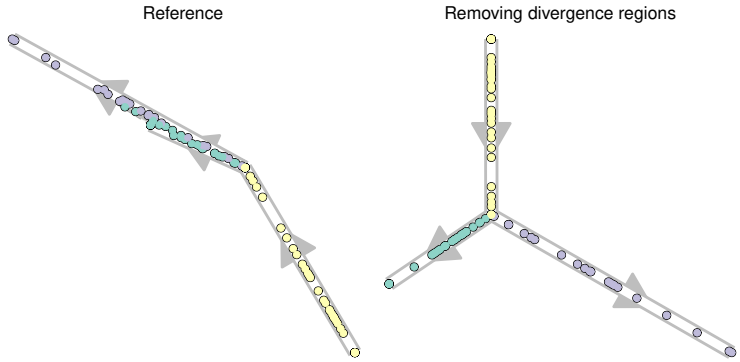


Figure S23: Example dataset before and after perturbation.

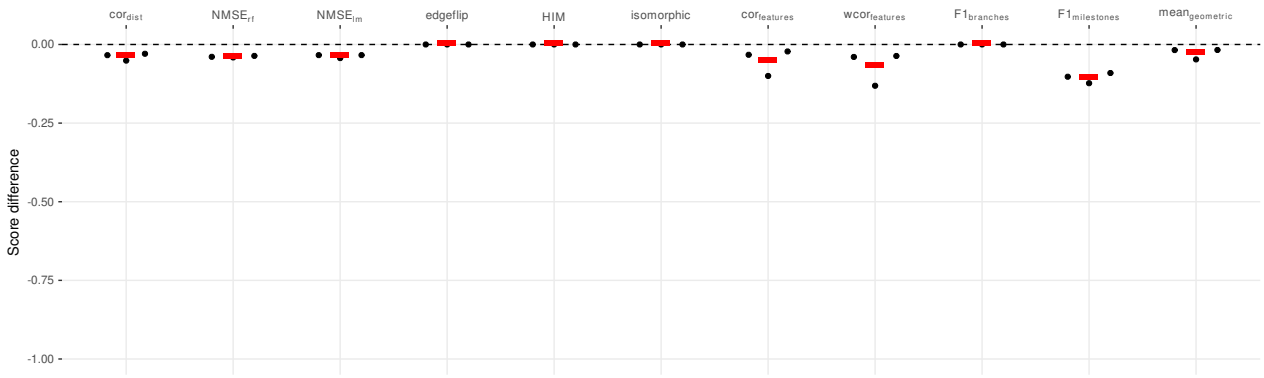
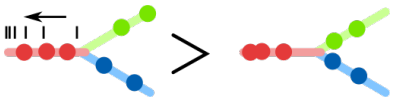


Figure S24: Differences in scores of 33 datasets before and after perturbation. Red bar gives the mean.

Rule 8: Move cells to start milestone



Description: Moving the cells closer to their start milestone should lower the score. Cells were moved closer to the start milestone using $percentage_{new} = percentage^{warp\ magnitude}$.

A metric conforms to this rule if: $monotonic\,(\textit{Warp magnitude}, \overline{score}_{\textit{Warp magnitude}})$.

Conclusion(s): Both $F1_{branches}$ and topology scores fail here because neither the topology nor the branch assignment changes. The score decreases only slightly for all the other metrics, given that only the relative distances change between cells, but not their actual ordering.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔

Table S9: Which metrics conform to rule 8.

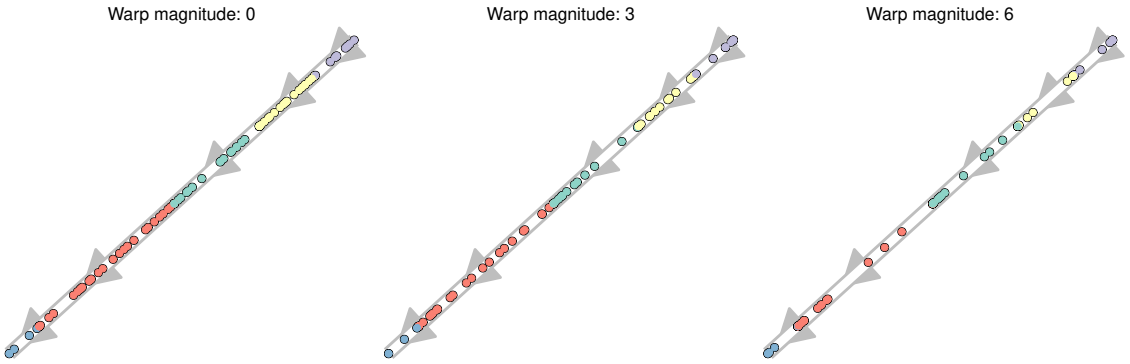


Figure S25: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

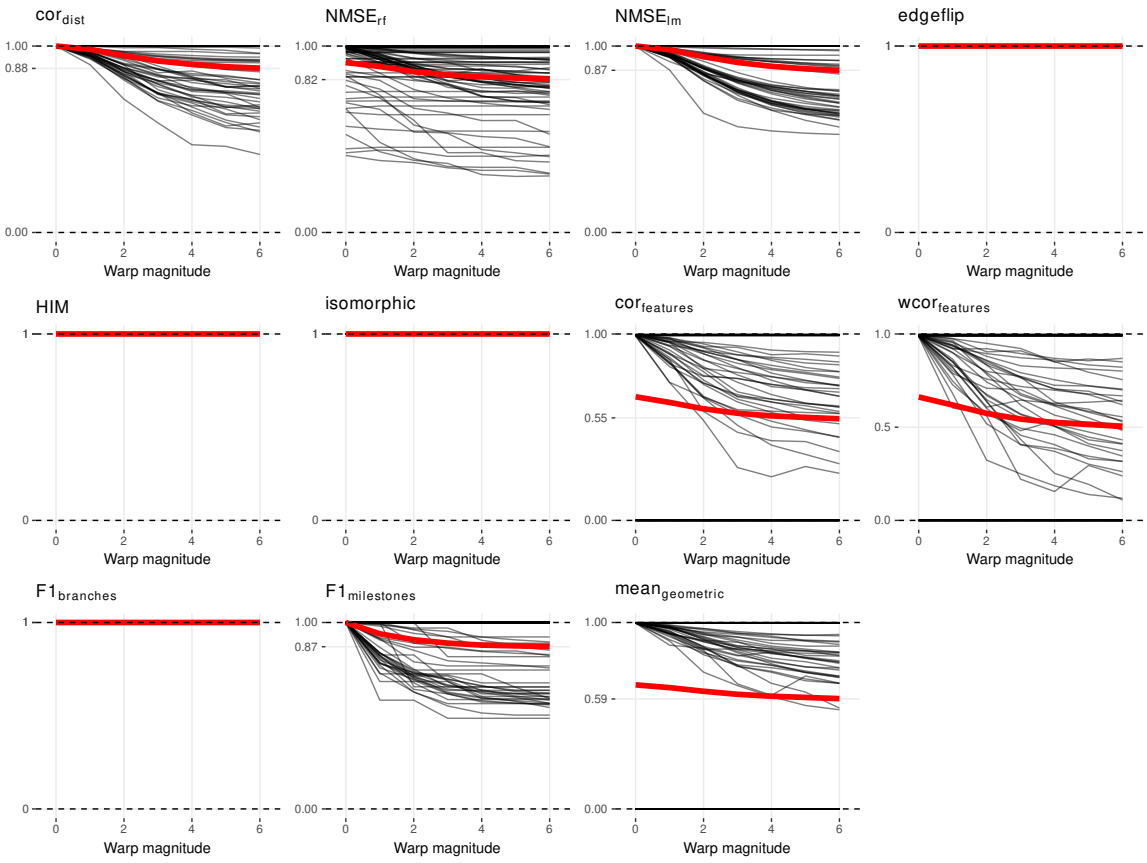


Figure S26: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 9: Move cells to closest milestone

Description: Moving the cells closer to their nearest milestone should lower the score.

A metric conforms to this rule if: $monotonic\left(\textit{Warp magnitude}, \overline{score}_{\textit{Warp magnitude}}\right)$.

Conclusion(s): Both $F1_{branches}$ and topology scores fail here because neither the topology nor the branch assignment changes. The score decreases only slightly for all the other metrics, given that only the relative distances change between cells, but not their actual ordering.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✔	✔	✖	✖	✖	✔	✔	✖	✔	✔

Table S10: Which metrics conform to rule 9.

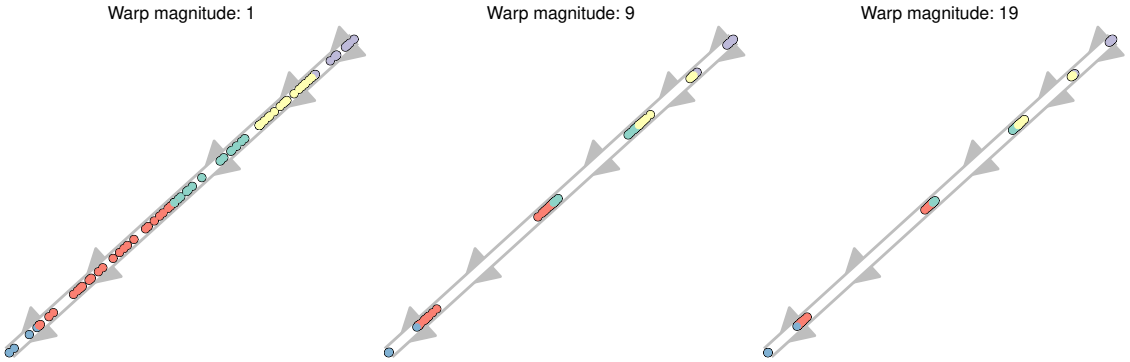


Figure S27: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

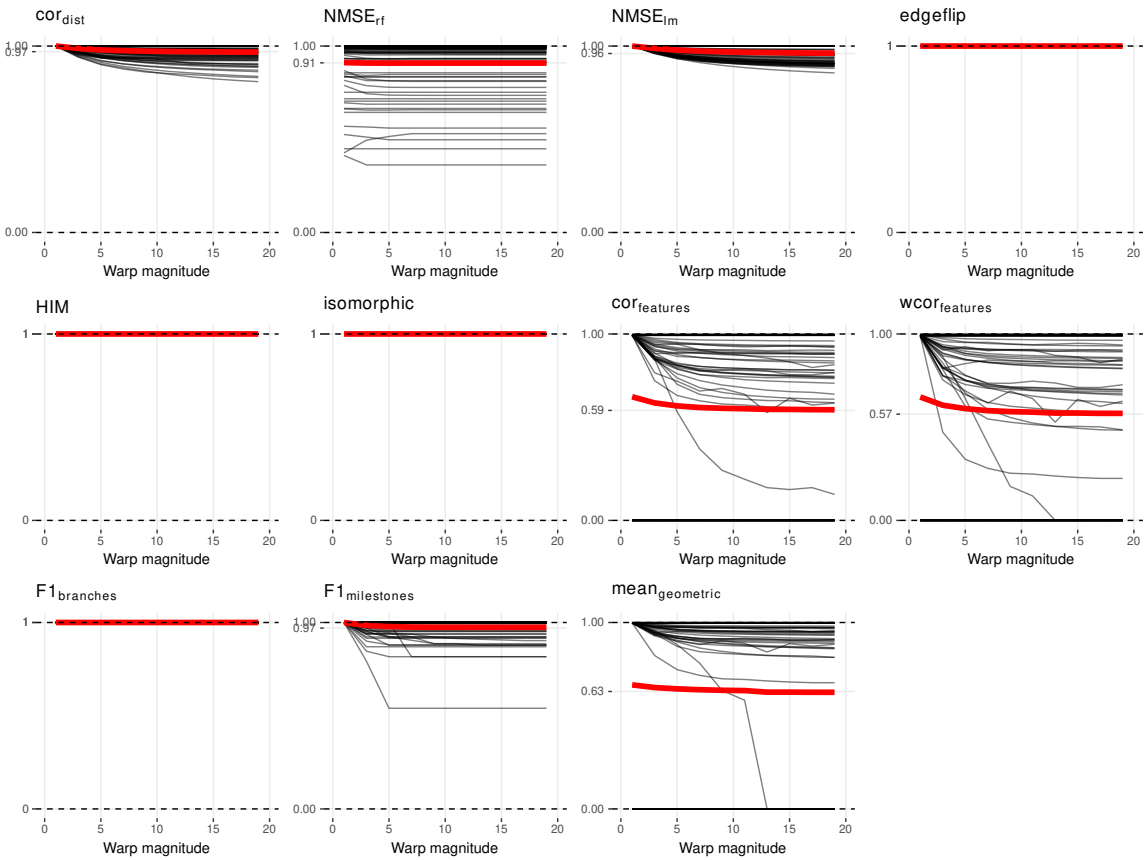
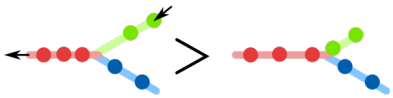


Figure S28: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 10: Length shuffling



Description: Shuffling the lengths of the edges of the milestone network should lower the score.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): Only the correlation between geodesic distances is consistently decreases when the lengths of the edges change.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✖	✔	✖	✔	✖	✖	✖	✖	✔	✔

Table S11: Which metrics conform to rule 10.

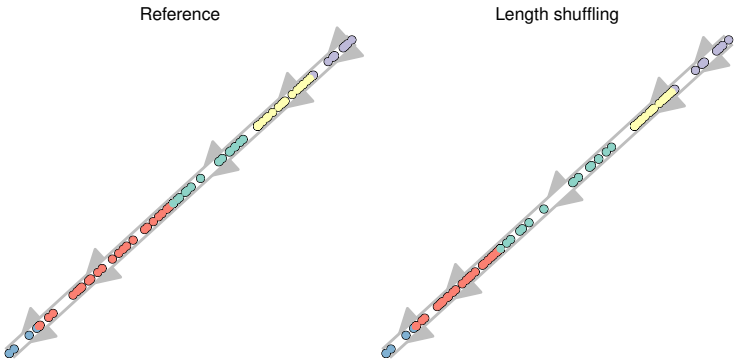


Figure S29: Example dataset before and after perturbation.

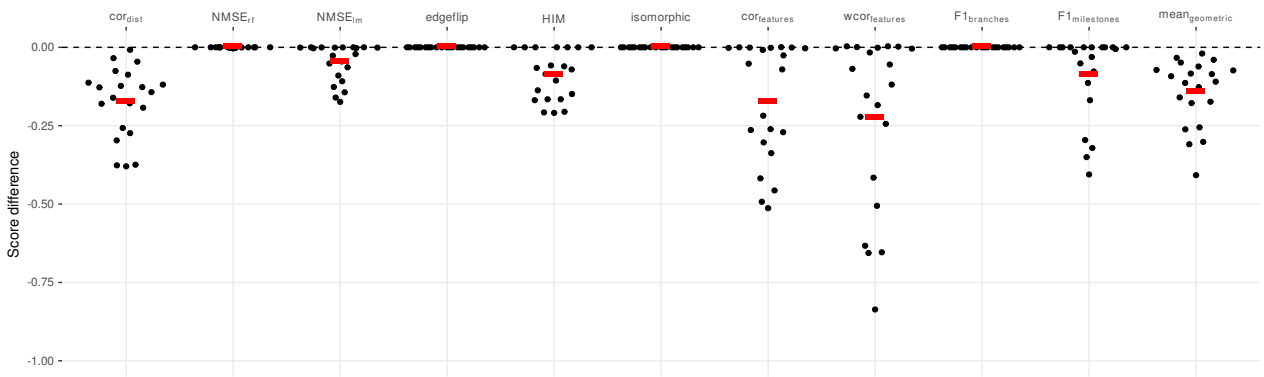
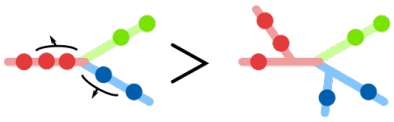


Figure S30: Differences in scores of 231 datasets before and after perturbation. Red bar gives the mean.

Rule 11: Cells into small subedges



Description: Moving some cells into short subedges should lower the score.

A metric conforms to this rule if: $monotonic\left(\textit{Number of added edges}, \overline{score}_{\textit{Number of added edges}}\right)$.

Conclusion(s): This rule is primarily captured by the scores looking at the topology and clustering quality.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✖	✔	✖	✔	✔	✔	✔	✖	✔	✔	✔

Table S12: Which metrics conform to rule 11.

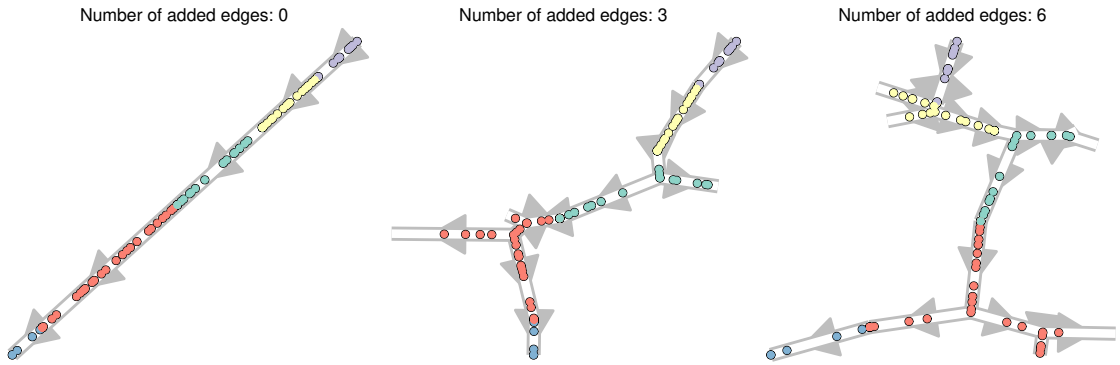


Figure S31: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

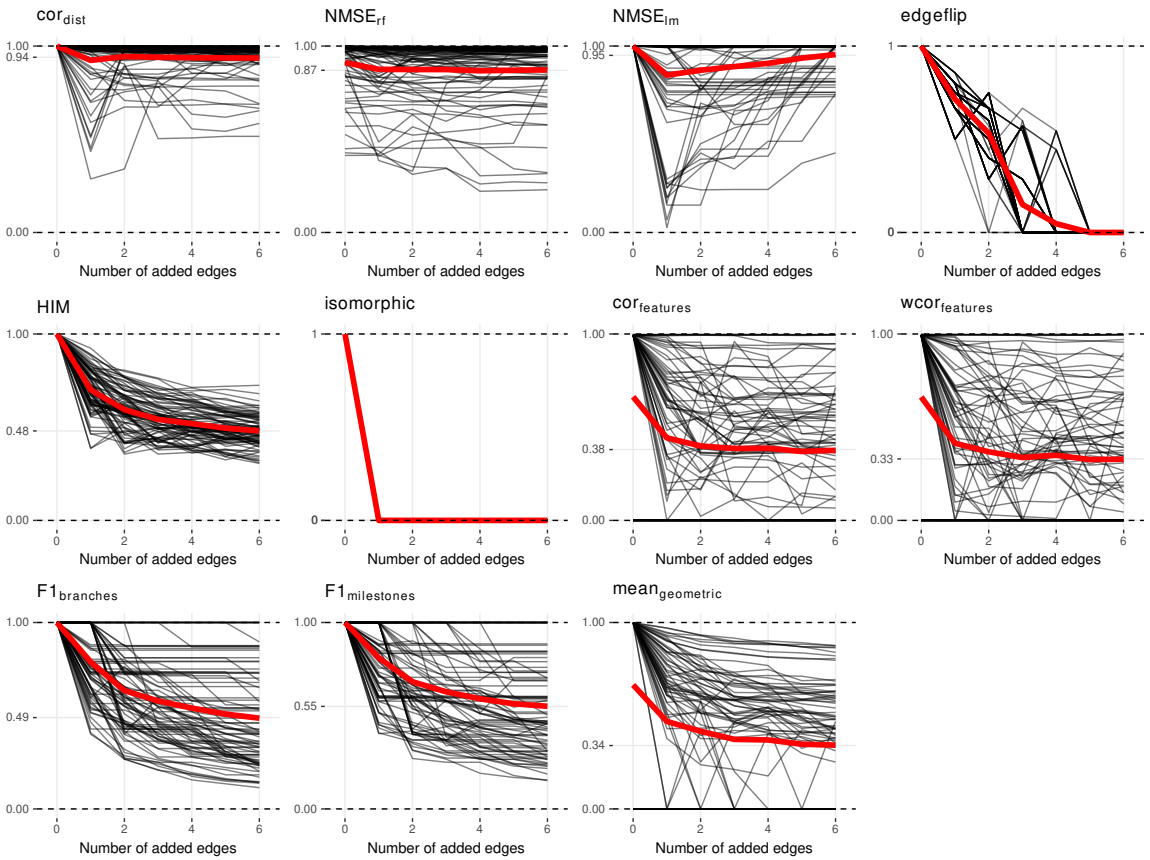


Figure S32: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 12: New leaf edges

Description: Adding new edges only connected to one existing milestone should lower the score.

A metric conforms to this rule if: $monotonic\left(\textit{Number of edges}, \overline{score}_{\textit{Number of edges}}\right)$.

Conclusion(s): As the positions of the cells are not affected, only metrics which investigate the clustering quality and topology conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✖	✔	✔	✔	✖	✖	✔	✔	✔

Table S13: Which metrics conform to rule 12.

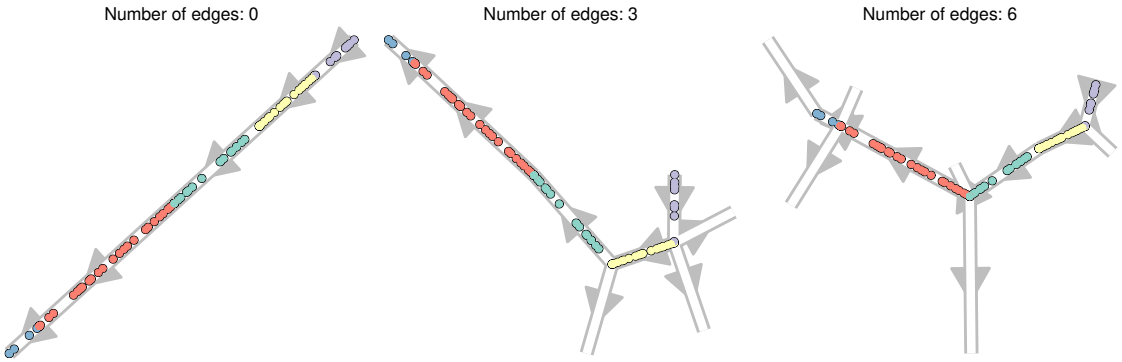


Figure S33: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

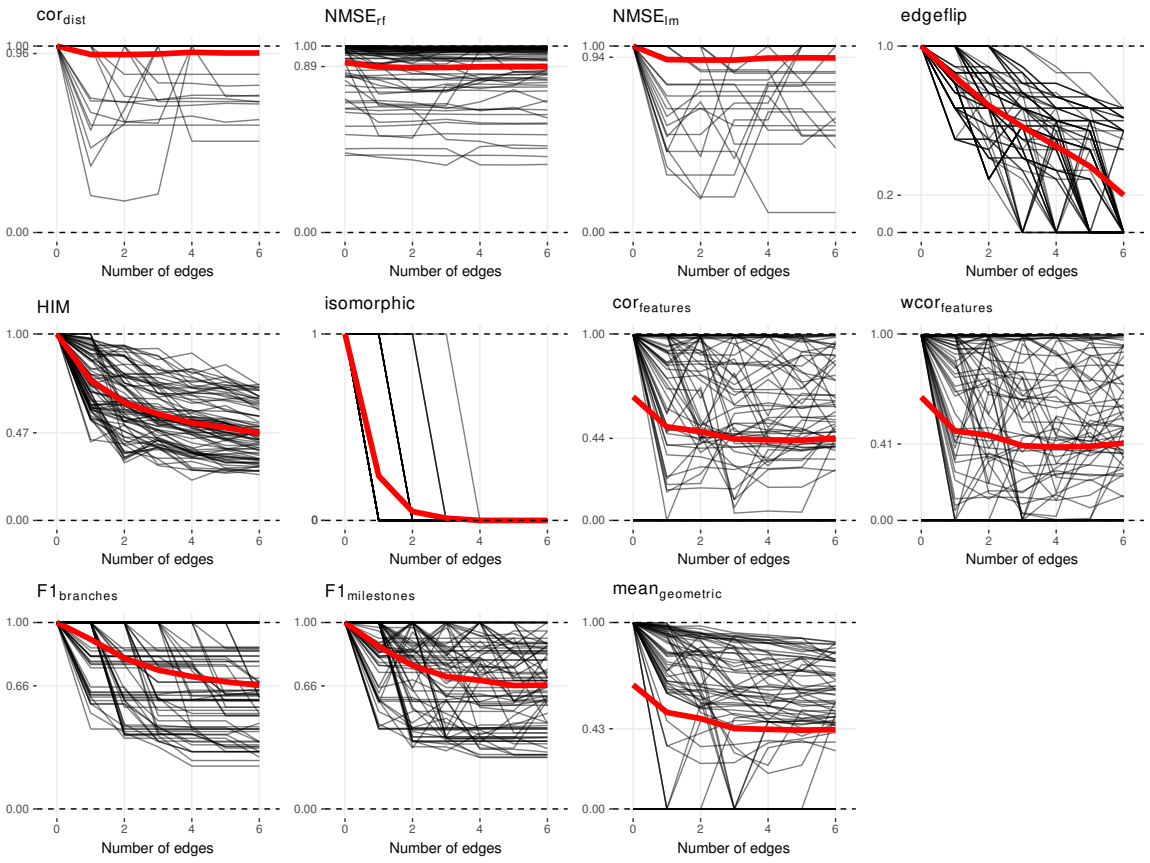
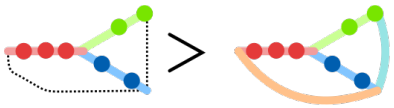


Figure S34: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 13: New connecting edges



Description: Adding new edges between existing milestones should lower the score.

A metric conforms to this rule if: $monotonic\left(\textit{Number of edges}, \overline{score}_{\textit{Number of edges}}\right)$.

Conclusion(s): Even though the positions of the cells do not change, the cor_{dist} still conforms to this rule because new edges can create shortcuts which will affect the geodesic distances between cells. Apart from this, metrics which investigate the clustering quality and topology also conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✖	✔	✔	✔	✔	✔	✔	✔	✔

Table S14: Which metrics conform to rule 13.

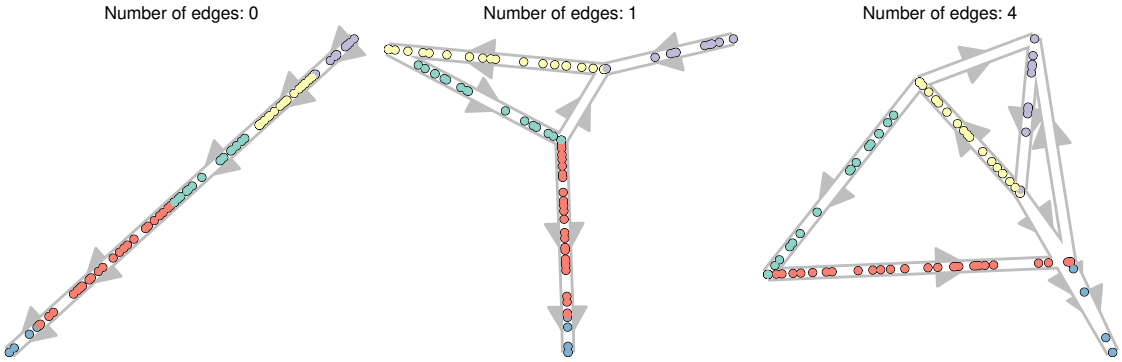


Figure S35: Lowly (left), moderatly (middle) and highly (right) perturbed example dataset.

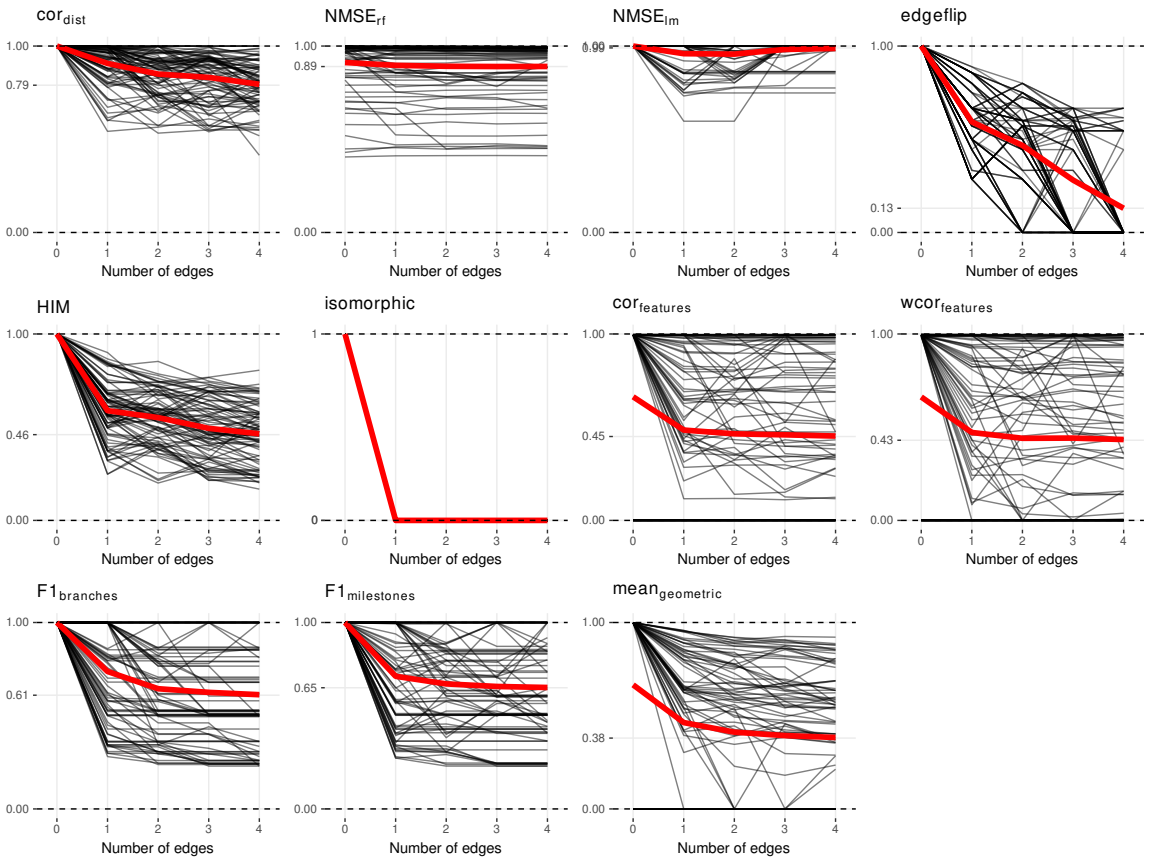
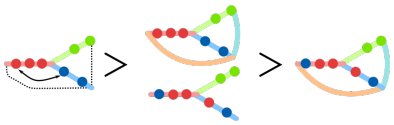


Figure S36: Score values at different extents of the perturbation across 84 datasets. Red line denotes the mean values.

Rule 14: Changing topology and cell position



Description: Changing both the topology and the cell positions should lower the score more than any of the two individually.

A metric conforms to this rule if: $score_{identity} > score_a \wedge score_{identity} > score_b \wedge score_a > score_{a+b} \wedge score_b > score_{a+b}$.

Conclusion(s): Most metrics have problems with this rule as they focus on either the cellular positions or the topology individually. Only the cor[dist] and $mean_{geometric}$ consistently conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✖	✖	✖	✖	✖	✖	✖	✖	✖	✖	✔

Table S15: Which metrics conform to rule 14.

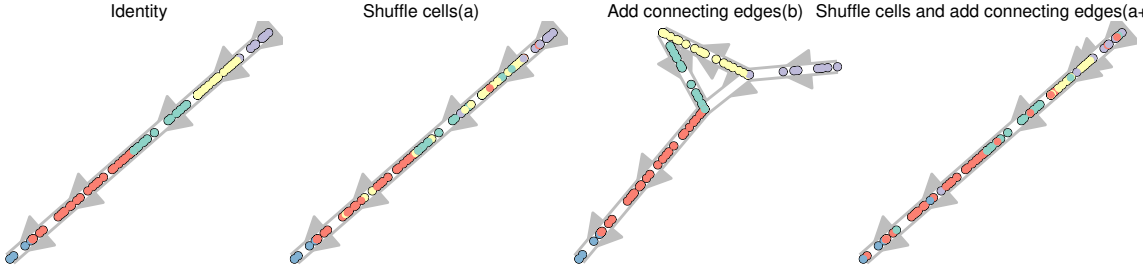
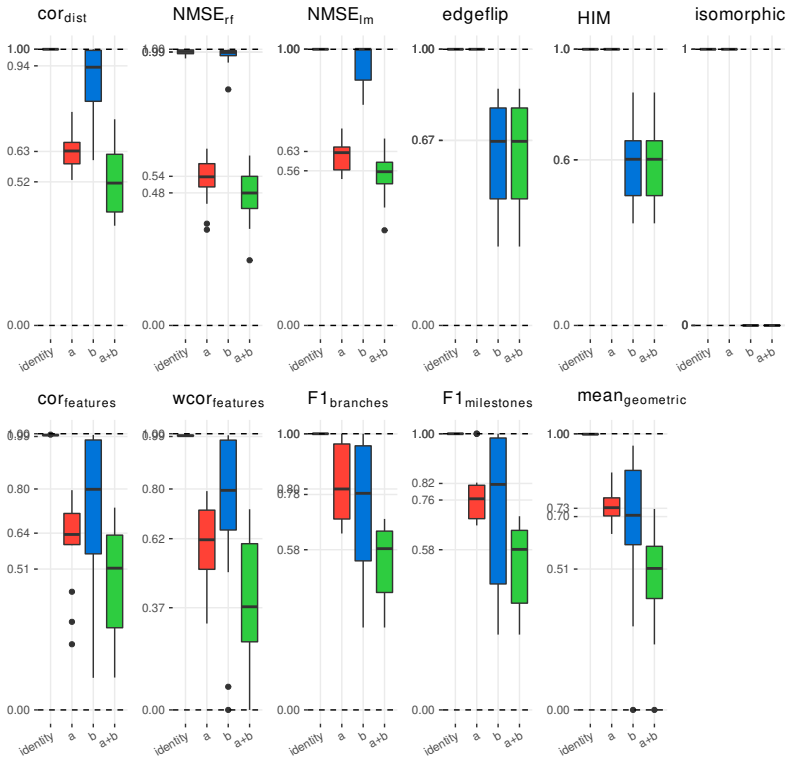


Figure S37: Example dataset before perturbation (identity), with any of the two perturbations (a and b) and both perturbations combined (a+b).

\begin{myfigure}{!htbp}

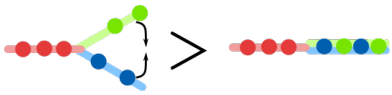


\textbf{Figure S38: Score values before perturbation (identity), with any of the two perturbations (a and b) and both perturbations combined (a+b). The upper whisker of the boxplot extends from the hinge (75% percentile) to the largest

value, no further than 1.5× the IQR from the hinge. The lower whisker extends from the hinge (25% percentile) to the smallest value, at most 1.5× the IQR of the hinge. We used 56 different datasets.}

\end{myfigure}

Rule 15: Bifurcation merging



Description: Merging the two branches after a bifurcation point should lower the score.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): This changes both the cellular ordering and the topology so most metrics are affected.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✖	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table S16: Which metrics conform to rule 15.

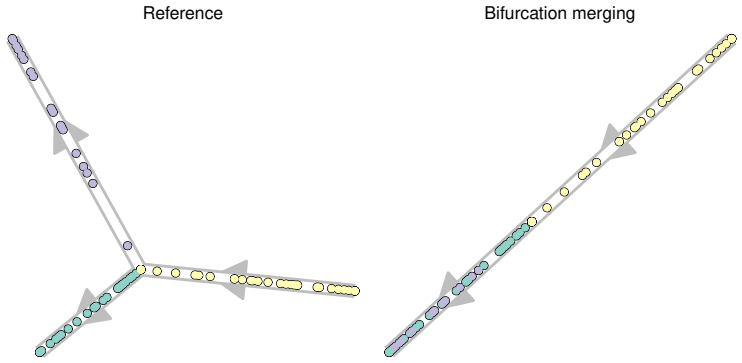


Figure S39: Example dataset before and after perturbation.

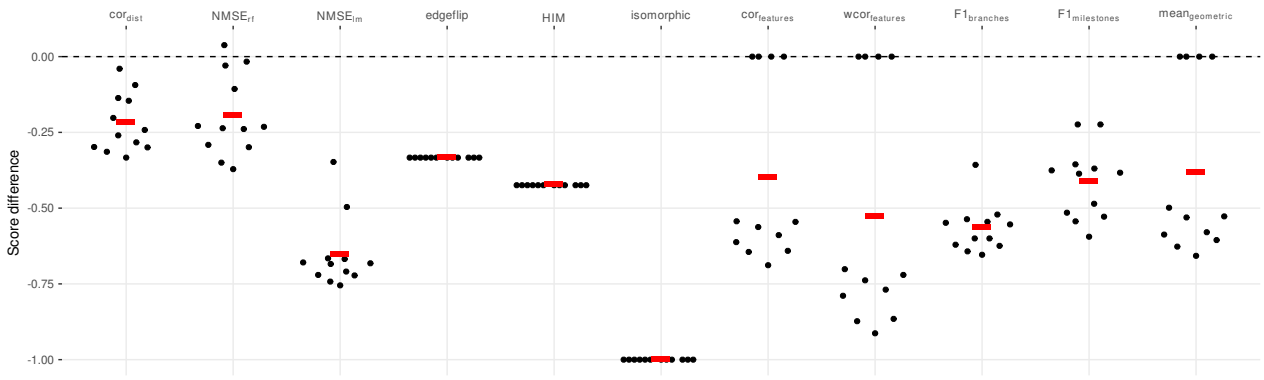
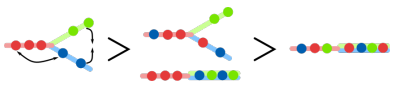


Figure S40: Differences in scores of 132 datasets before and after perturbation. Red bar gives the mean.

Rule 16: Bifurcation merging and changing cell positions



Description: Merging the two branches of a bifurcation and changing the cells positions should lower the score more than any of the two individually.

A metric conforms to this rule if: $score_{identity} > score_a \wedge score_{identity} > score_b \wedge score_a > score_{a+b} \wedge score_b > score_{a+b}$.

Conclusion(s): Only metrics which look at the topology do not conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✔	✖	✖	✖	✔	✔	✔	✔	✔

Table S17: Which metrics conform to rule 16.

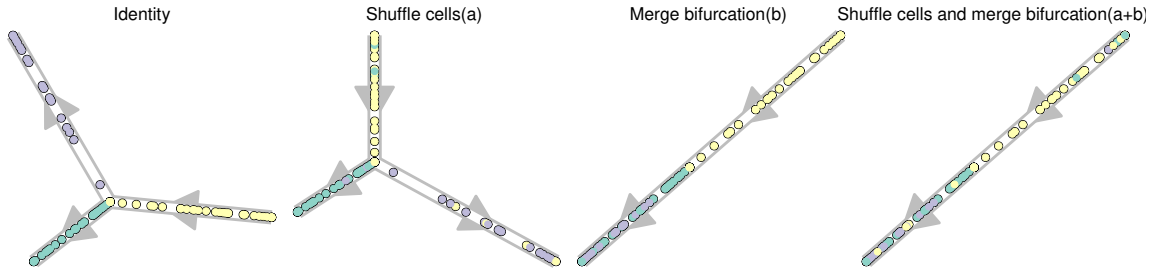
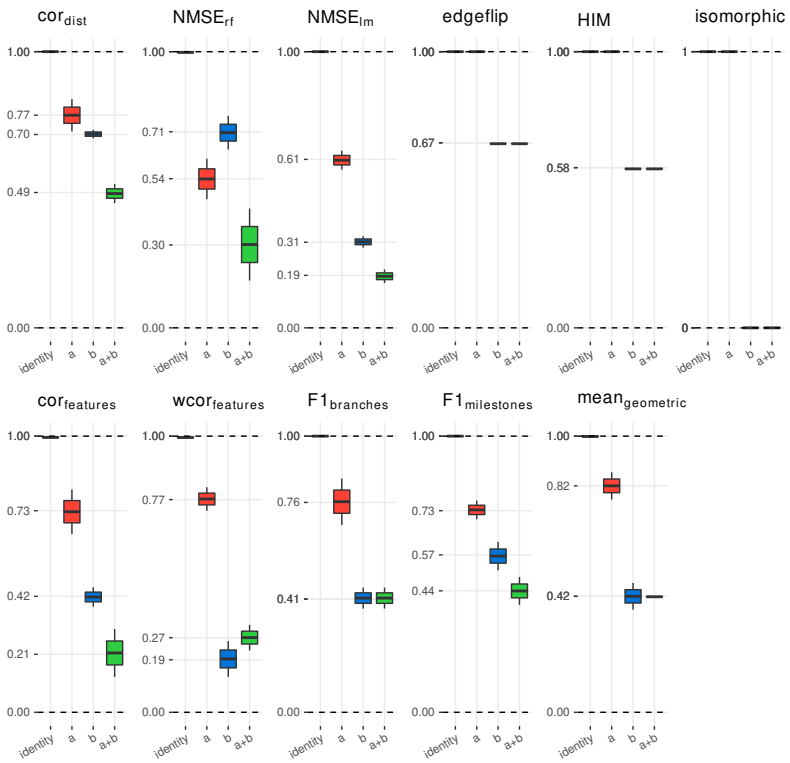


Figure S41: Example dataset before perturbation (identity), with any of the two perturbations (a and b) and both perturbations combined (a+b).

\begin{myfigure}{!htbp}



\textbf{**Figure S42**: Score values before perturbation (identity), with any of the two perturbations (a and b) and both perturbations combined (a+b). The upper whisker of the boxplot extends from the hinge (75% percentile) to the largest value, no further than 1.5× the IQR from the hinge. The lower whisker extends from the hinge (25% percentile) to the smallest value, at most 1.5× the IQR of the hinge. We used 8 different datasets.}

\end{myfigure}

Rule 17: Bifurcation concatentation

Description: Concatenating one branch of a bifurcation to the other bifurcation branch should lower the score.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): This changes both the cellular ordering and the topology so most metrics conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✖	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table S18: Which metrics conform to rule 17.

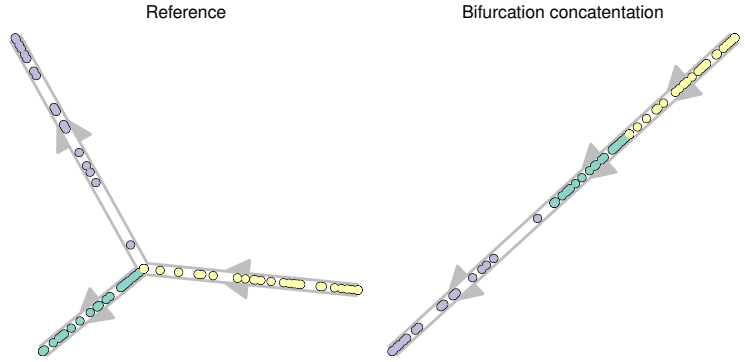


Figure S43: Example dataset before and after perturbation.

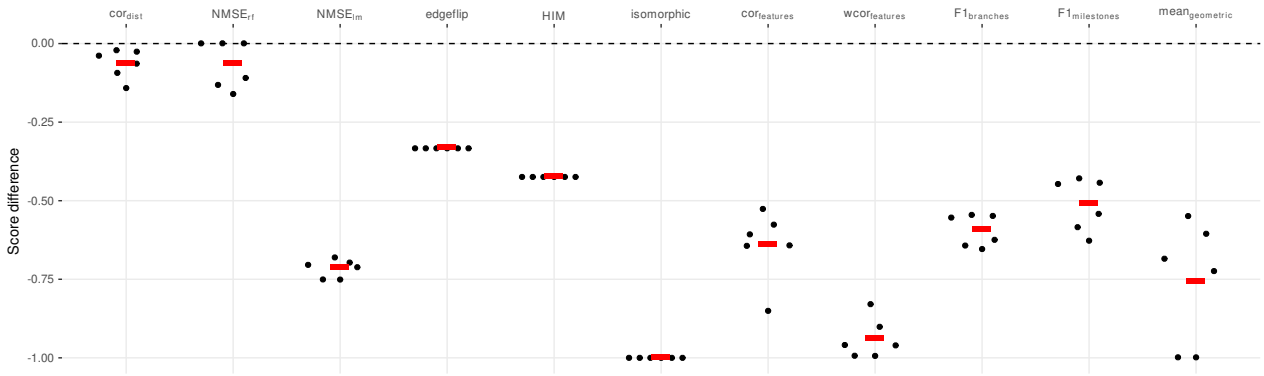


Figure S44: Differences in scores of 66 datasets before and after perturbation. Red bar gives the mean.

Rule 18: Cycle breaking

Description: Breaking a cyclic trajectory should lower the score.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): Because the actual positions of the cells nor the branch assignment change, both the MSE metrics and the $F1_{branches}$ do not conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✖	✔	✔	✔	✔	✔	✔	✖	✔	✔

Table S19: Which metrics conform to rule 18.

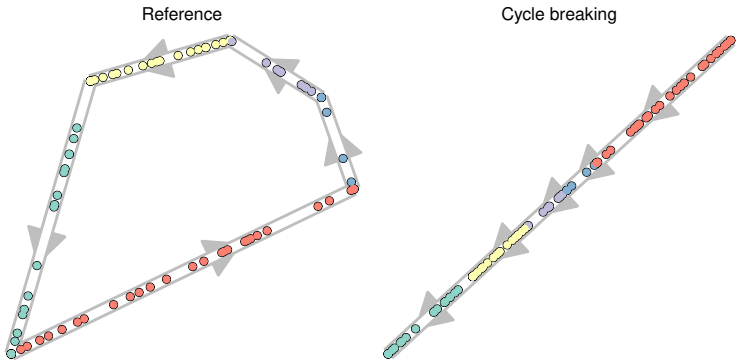


Figure S45: Example dataset before and after perturbation.

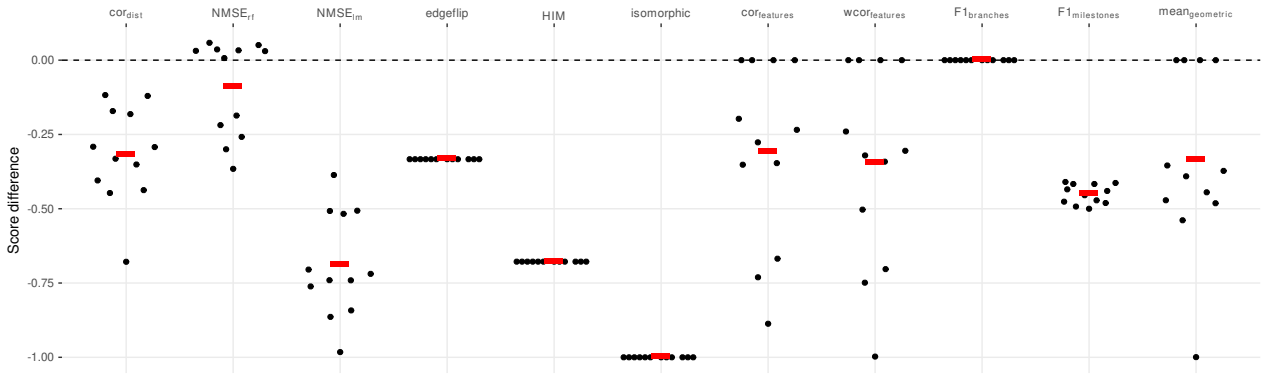
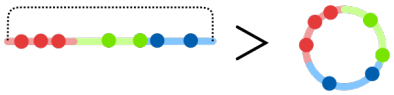


Figure S46: Differences in scores of 132 datasets before and after perturbation. Red bar gives the mean.

Rule 19: Linear joining



Description: Joining the two ends of a linear trajectory should lower the score.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): Because the positions of the cells can be perfectly predicted, the MSE metrics do not conform to this rule. Furthermore, because the branch assignment change stays the same, the $F1_{branches}$ also does not conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	$cor_{features}$	$wcor_{features}$	$F1_{branches}$	$F1_{milestones}$	$mean_{geometric}$
✔	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔

Table S20: Which metrics conform to rule 19.

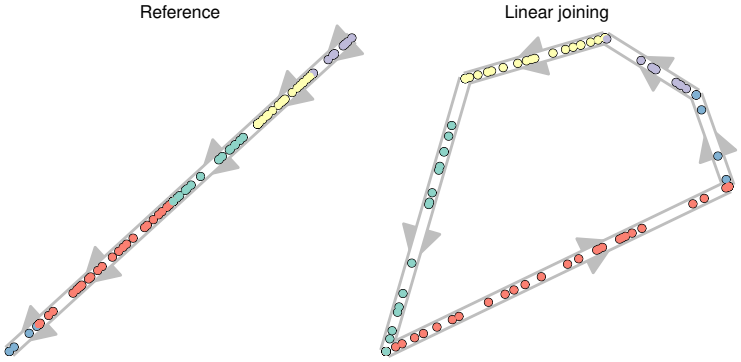


Figure S47: Example dataset before and after perturbation.

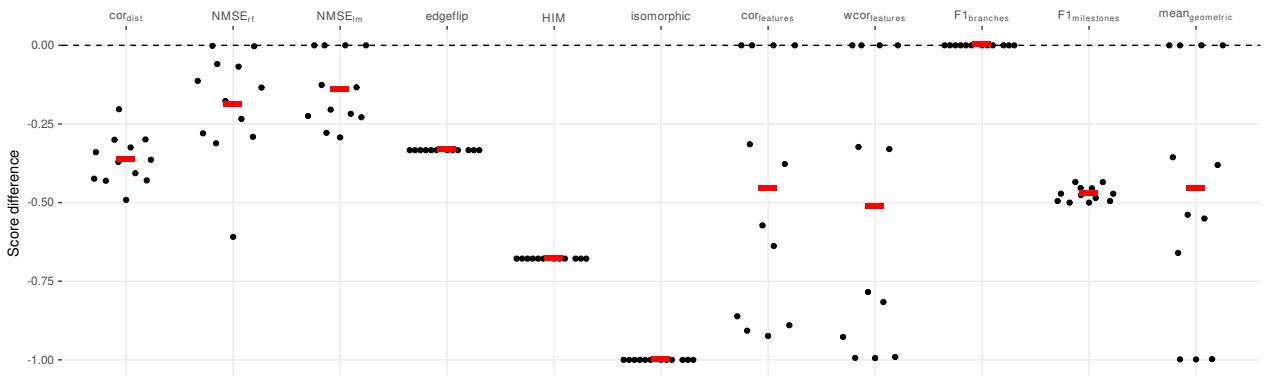
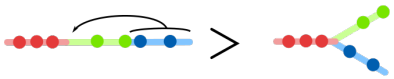


Figure S48: Differences in scores of 132 datasets before and after perturbation. Red bar gives the mean.

Rule 20: Linear splitting



Description: Splitting a linear trajectory into a bifurcation should lower the score.

A metric conforms to this rule if: $score_{identity} > score_{prediction}$.

Conclusion(s): Only the MSE metrics do not conform to this rule as the positions of the cells can be perfectly predicted in the gold standard given the prediction.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table S21: Which metrics conform to rule 20.

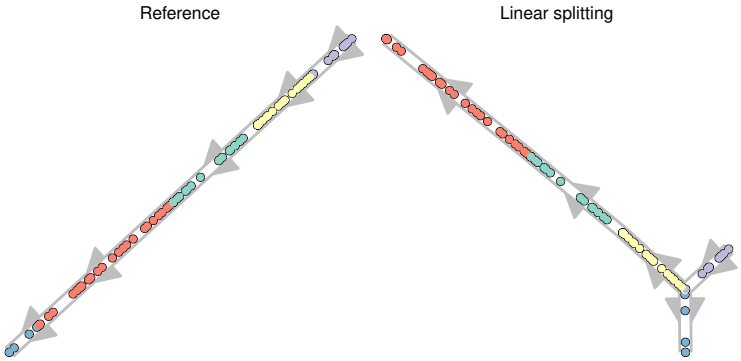


Figure S49: Example dataset before and after perturbation.

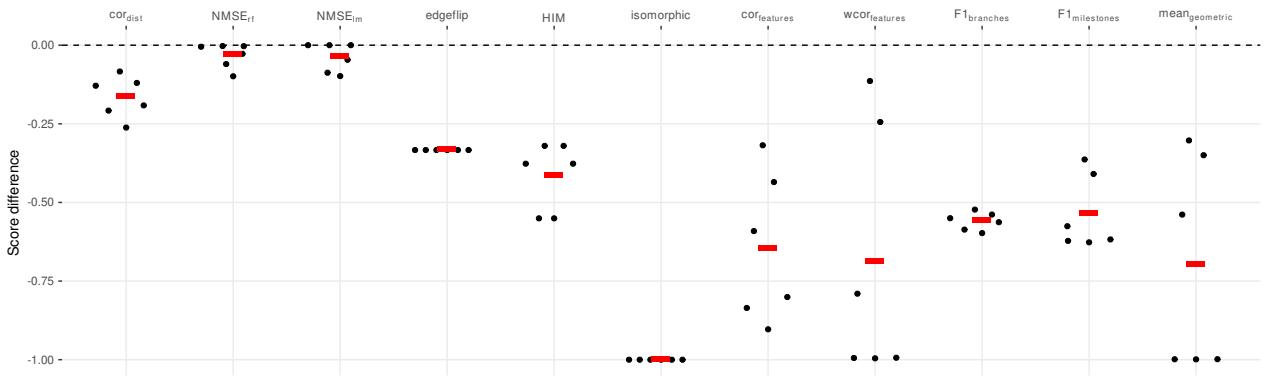


Figure S50: Differences in scores of 66 datasets before and after perturbation. Red bar gives the mean.

Rule 21: Change of topology

Description: Changing the topology of the trajectory should lower the score.

A metric conforms to this rule if: $score_{\text{same topology}} > score_{\text{different topology}}$.

Conclusion(s): Because the positions of the cells can be perfectly predicted, the MSE metrics do not conform to this rule. Furthermore, because the branch assignment change stays the same, the $F1_{\text{branches}}$ also does not conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✖	✖	✔	✔	✔	✔	✔	✖	✔	✔

Table S22: Which metrics conform to rule 21.

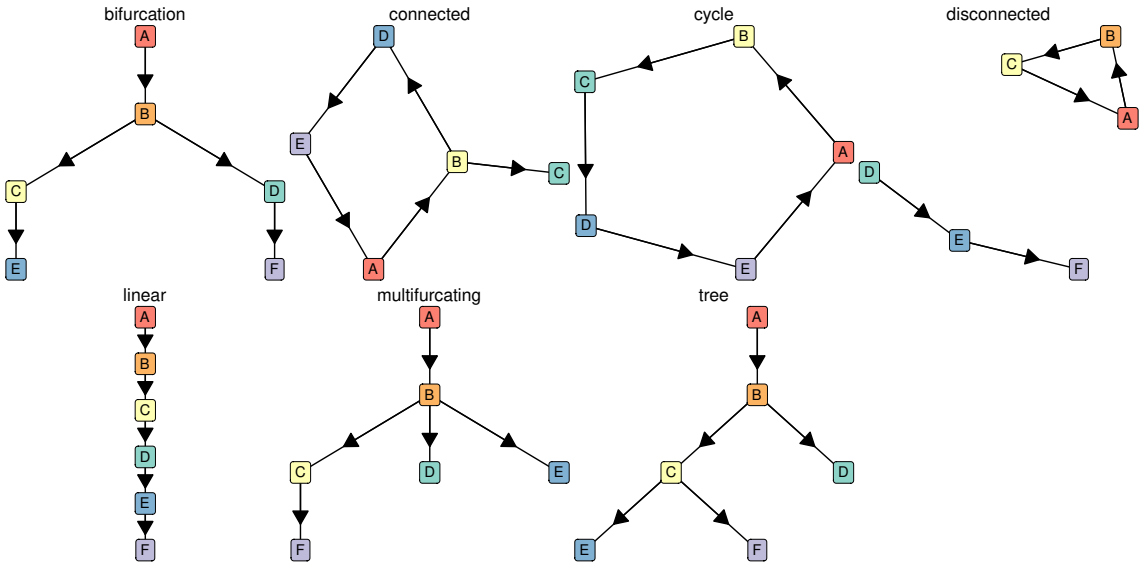


Figure S51: The different trajectory topologies that were used to compare the metrics.

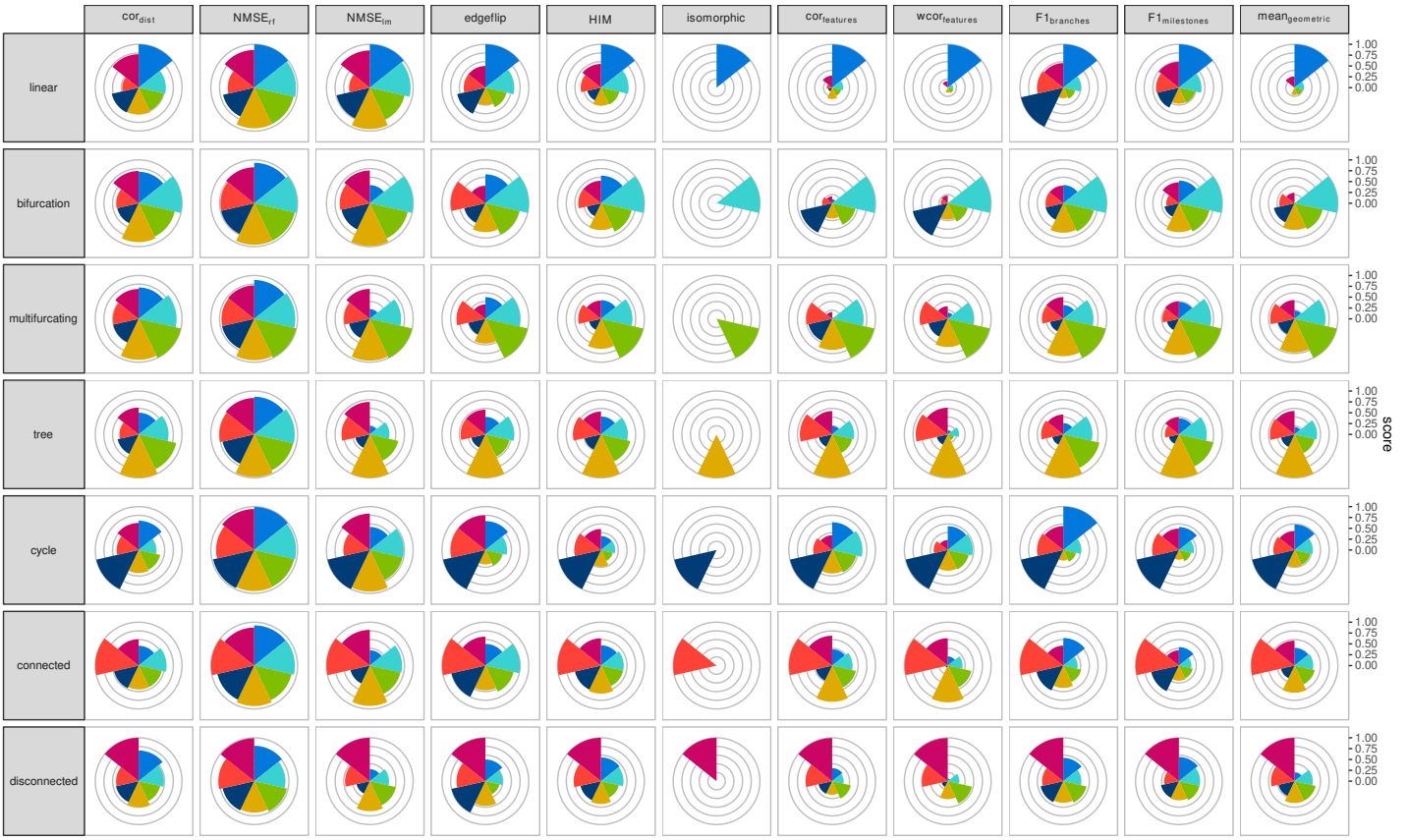


Figure S52: Score values on different topologies (left).

Rule 22: Cells on milestones vs edges

Description: A score should behave similarly both when cells are located on the milestones (as is the case in real datasets) or on the edges between milestones (as is the case in synthetic datasets).

A metric conforms to this rule if: $corr\left(score_{edges}, score_{milestones}\right) > 0.8$.

Conclusion(s): All scores conform to this rule.

cor_{dist}	$NMSE_{rf}$	$NMSE_{lm}$	edgeflip	HIM	isomorphic	cor_{features}	$wcor_{\text{features}}$	$F1_{\text{branches}}$	$F1_{\text{milestones}}$	$mean_{\text{geometric}}$
✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table S23: Which metrics conform to rule 22.

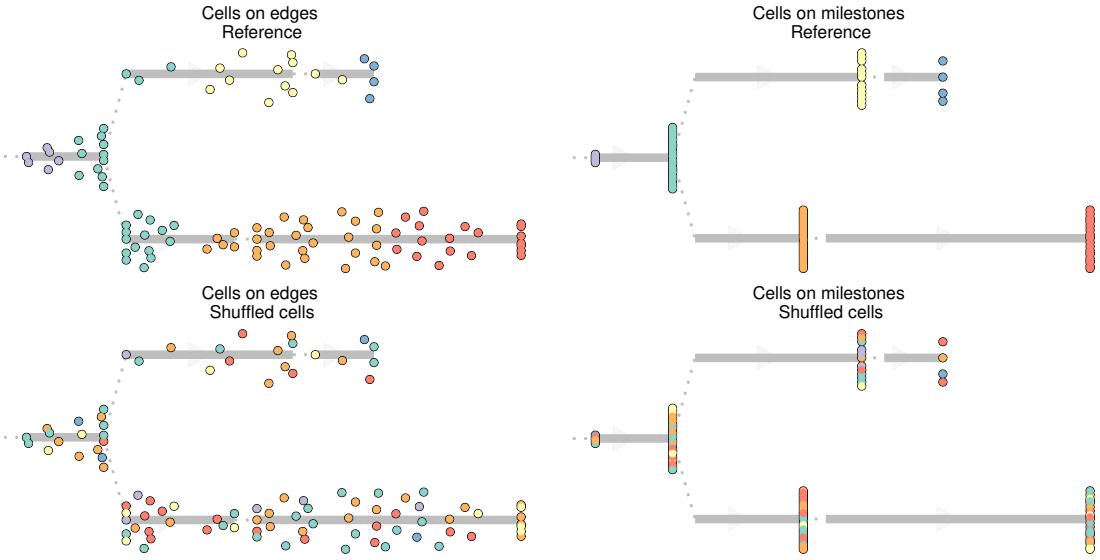


Figure S53: Example dataset in which cells are placed on the edges (left) or on the milestones (right), and with their original positions (top) or shuffled (bottom).

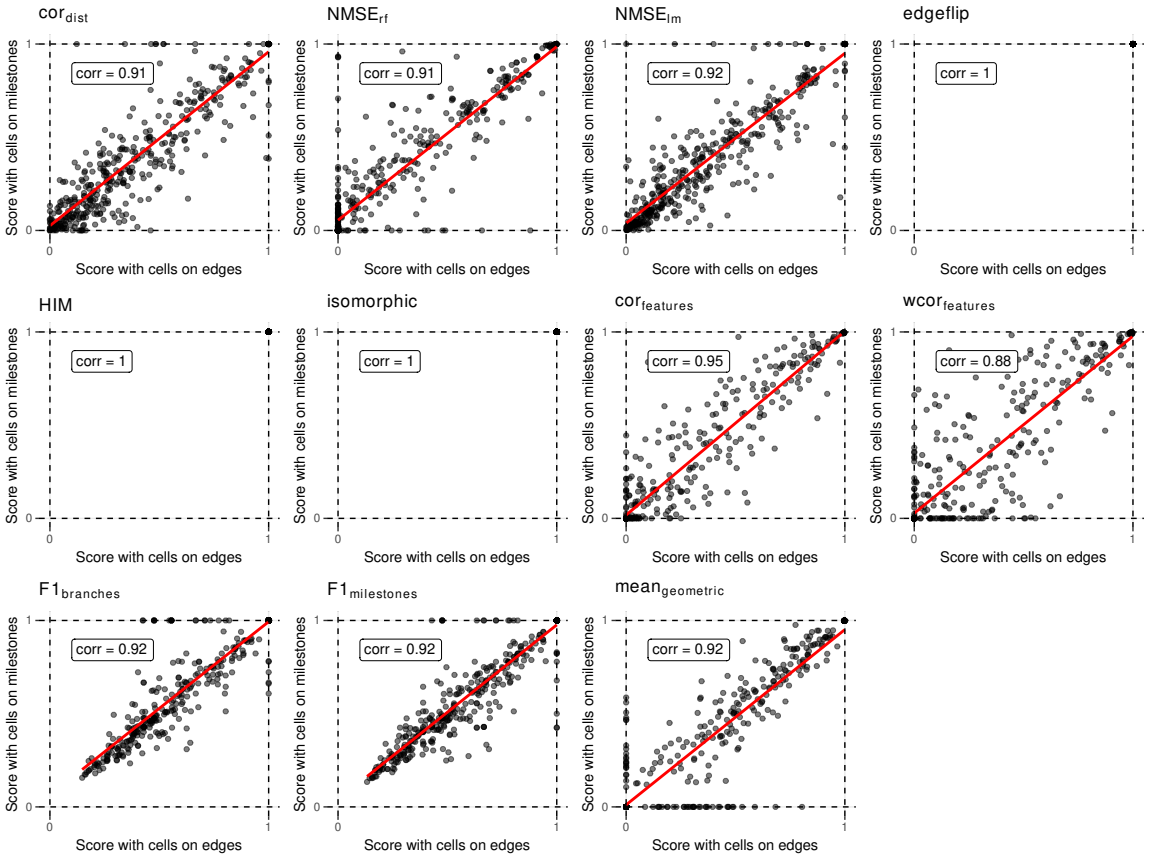


Figure S54: Score values of the same datasets (n = 84) in which cells were put either on the edges or on the milestones. Shown in the top left is the Spearman rank correlation.

Score aggregation

To rank the methods, we need to aggregate on two levels: across **datasets** and across specific/application metrics to calculate an **overall metric**.

Aggregating over datasets

When combining different datasets, it is important that the biases in the datasets does not influence the overall score. In our study, we define three such biases, although there are potentially many more:

- **Difficulty of the datasets**: Some datasets are more difficult than others. This can have various reasons, such as the complexity of the topology, the amount of biological and technical noise, or the dimensions of the data. It is important that a small increase in performance on a more difficult dataset has an equal impact on the final score as a large increase in performance on easier datasets.
- **Dataset sources**: It is much easier to generate synthetic datasets than real datasets, and this bias is reflected in our set of datasets. However, given their higher biological relevance, real datasets should be given at least equal importance than synthetic datasets.
- **Trajectory types**: There are many more linear and disconnected real datasets, and only a limited number of tree or graph datasets. This imbalance is there because historically most datasets have been linear datasets, and because it is easy to create disconnected datasets by combining different datasets. However, this imbalance in trajectory types does not necessarily reflect the general importance of that trajectory type.

We designed an aggregation scheme which tries to prevent these biases from influencing the ranking of the methods.

The difficulty of a dataset can easily have an impact on how much weight the dataset gets in an overall ranking. We illustrate this with a simple example in **Figure S55**. One method consistently performs well on both the easy and the difficult datasets. But because the differences are small in the difficult datasets, the mean would not give this method a high score. Meanwhile, a variable method which does not perform well on the difficult dataset gets the highest score, because it scored so high on the easier dataset.

To avoid this bias, we normalise the scores of each dataset by first scaling and centering to $\mu = 0$ and $\sigma = 1$, and then moving the score values back to $[0, 1]$ by applying the unit normal density distribution function. This results in scores which are comparable across different datasets (**Figure S55**). In contrast to other possible normalisation techniques, this will still retain some information on the relative difference between the scores, which would have been lost when using the ranks for normalisation. An example of this normalisation, which will also be used in the subsequent aggregation steps, can be seen in **Figure S56**.

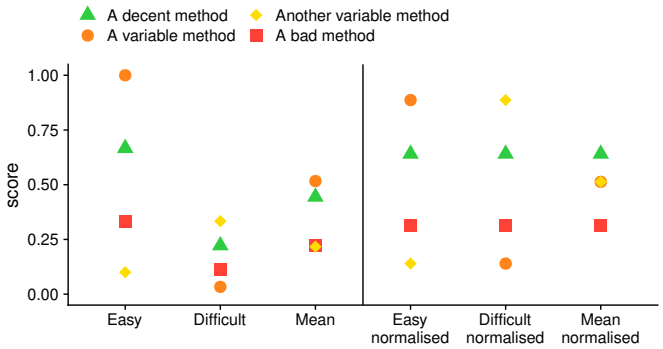


Figure S55: An illustration of how the difficulty of a dataset can influence the overall ranking. A decent method, which consistently ranks high on an easy and difficult dataset, does not get a high score when averaging. On the other hand, a method which ranks high on the easy dataset, but very low on the difficult dataset does get a high score on average. After normalising the scores (right), this problem dissapears.

After normalisation, we aggregate step by step the scores from different datasets. We first aggregate the datasets with the same dataset source and trajectory type using an arithmetic mean of their scores **Figure S57a**. Next, the scores are averaged over different dataset sources, using a arithmetic mean which was weighted based on how much the synthetic and silver scores correlated with the real gold scores **Figure S57b**. Finally, the scores are aggregated over the different trajectory types again using a arithmetic mean **Figure S57c**.

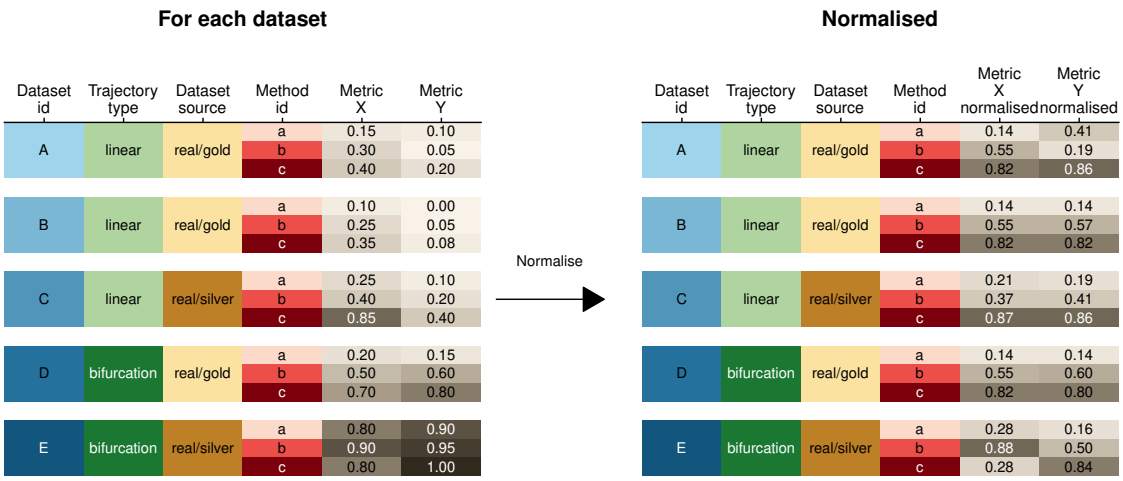


Figure S56: An example of the normalisation procedure. Shown are some results of a benchmarking procedure, where every row contains the scores of a particular method (red shading) on a particular dataset (blue shading), with a trajectory type (green shading) and dataset source (orange shading). In this example, we first split the datasets

Overall metrics

Undoubtedly, a single optimal overall metric does not exist for trajectories, as different users may have different priorities:

- A user may be primarily interested in defining the correct topology, and only use the cellular ordering when the topology is correct
- A user may be less interested in how the cells are ordered within a branch, but primarily in which cells are in which branches
- A user may already know the topology, and may be primarily interested in finding good features related to a particular branching point
- …

Each of these scenarios would require a combinations of *specific* and *application* metrics with different weights. To provide an "overall" ranking of the metrics, which is impartial for the scenarios described above, we therefore chose a metric which weighs every aspect of the trajectory equally:

- Its **ordering**, using the cor_{dist}
- Its **branch assignment**, using the $F1_{branches}$
- Its **topology**, using the HIM
- The accuracy of **differentially expressed features**, using the $wcor_{\text{features}}$

Next, we considered three different ways of averaging different scores: the arithmetic mean, geometric mean and harmonic mean. Each of these types of mean have different use cases. The harmonic mean is most appropriate when the scores would all have a common denominator (as is the case for the Recovery and Relevance described earlier). The arithmetic mean would be most appropriate when all the metrics have the same range. For our use case, the geometric mean is the most appropriate, because it is low if one of the values is low. For example, this means that if a method is not good at inferring the correct topology, it will get a low overall score, even if it performs better at all other scores. This ensures that a high score will only be reached if a prediction has a good ordering, branch assignment, topology, and set of differentially expressed features.

The final overall score (**Figure S58**) for a method was thus defined as:

$$Overall = mean_{geometric} = \sqrt[4]{cor_{\text{dist}} \times \text{HIM} \times wcor_{\text{features}} \times F1_{branches}}$$

We do however want to stress that different use cases will require a different overall score to order the methods. Such a context-dependent ranking of all methods is provided through the dynguidelines app (guidelines.dynverse.org).

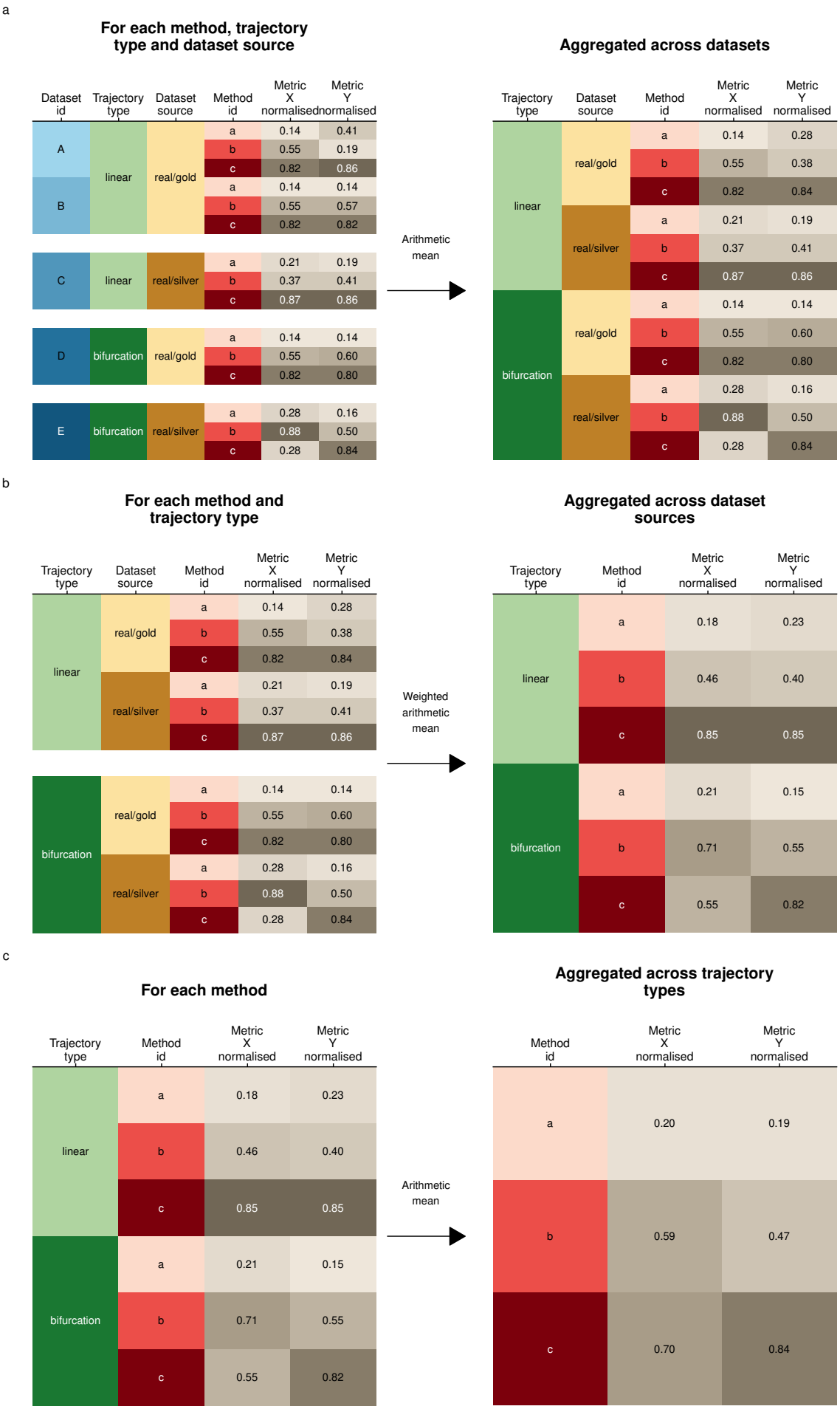


Figure S57: An example of the aggregation procedure. In consecutive steps we aggregated across (a) different datasets with the same source and trajectory type, (b) different dataset sources with the same trajectory type (weighted for the correlation of the dataset source with the real gold dataset source) and (c) all trajectory types.

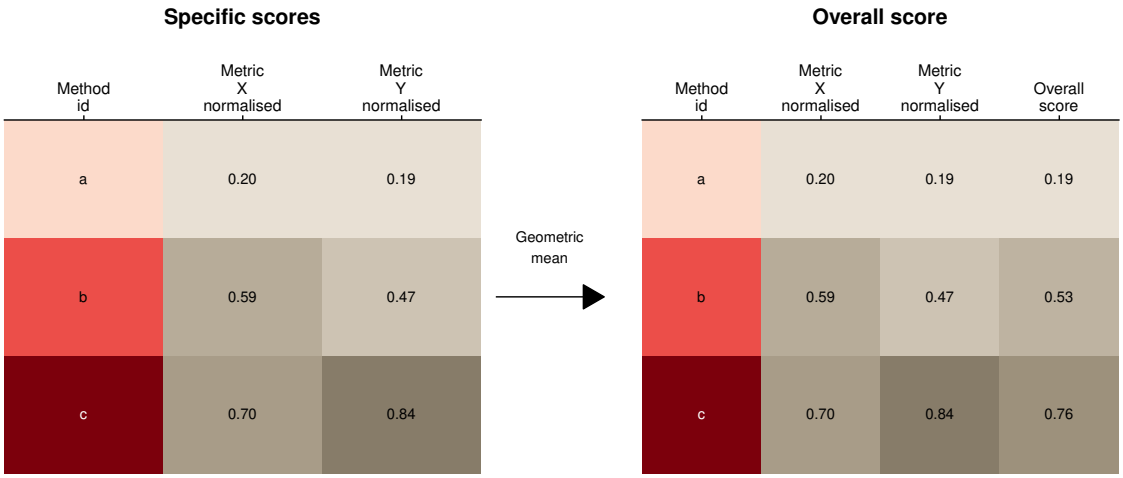


Figure S58: An example of the averaging procedure. For each method, we calculated the geometric mean between its normalised and aggregated scores

Supplementary References

1. Junttila, T. & Kaski, P. Engineering an Efficient Canonical Labeling Tool for Large and Sparse Graphs. in *2007 Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX)* 135–149 (Society for Industrial and Applied Mathematics, 2007). doi:10.1137/1.9781611972870.13

2. Bahiense, L., Manić, G., Piva, B. & de Souza, C. C. The maximum common edge subgraph problem: A polyhedral investigation. *Discrete Applied Mathematics* **160**, 2523–2541 (2012).

3. Jurman, G., Visintainer, R., Filosi, M., Riccadonna, S. & Furlanello, C. The HIM glocal metric and kernel for network comparison and classification. in *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* 1–10 (2015). doi:10.1109/DSAA.2015.7344816

4. Dougherty, E. R. Validation of gene regulatory networks: Scientific and inferential. *Briefings in Bioinformatics* **12**, 245–252 (2011).

5. Ipsen, M. & Mikhailov, A. S. Evolutionary reconstruction of networks. *Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics* **66**, 046109 (2002).

6. Saelens, W., Cannoodt, R. & Saeys, Y. A comprehensive evaluation of module detection methods for gene expression data. *Nature Communications* **9**, 1090 (2018).

7. Wright, M. N. & Ziegler, A. Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R | Wright | Journal of Statistical Software. *Journal of Statistical Software* **77**, (2017).

Synthetic datasets

To generate synthetic datasets, we used four different synthetic data simulators:

- **dyngen:** Simulations of gene regulatory networks, available at github.com/dynverse/dyngen
- **dyntoy:** Random gradients of expression in the reduced space, available at github.com/dynverse/dyntoy
- **PROSSTT:** Expression is sampled from a linear model which depends on pseudotime[1]
- **Splatter:** Simulations of non-linear paths between different expression states[2]

For every simulator, we took great care to make the datasets as realistic as possible. To do this, we extracted several parameters from all real datasets. We calculated the number of differentially expressed features within a trajectory using a two-way Mann–Whitney U test between every pair of cell groups. These values were corrected for multiple testing using the Benjamini-Hochberg procedure (FDR < 0.05) and we required that a gene was expressed in at least 5% of cells, and had at least a fold-change of 2. We also calculated several other parameters, such as drop-out rates and library sizes using the Splatter package[2]. These parameters were then given to the simulators when applicable, as described for each simulator below. Not every real dataset was selected to serve as a reference for a synthetic dataset. Instead, we chose a set of ten distinct reference real datasets by clustering all the parameters of each real dataset, and used the reference real datasets at the cluster centers from a pam clustering (with $k = 10$, implemented in the R cluster package) to generate synthetic data.

dyngen

The dyngen (github.com/dynverse/dyngen) workflow to generate synthetic data is based on the well established workflow used in the evaluation of network inference methods[3,4] and consists of four main steps: network generation, simulation, gold standard extraction and simulation of the scRNA-seq experiment. At every step, we tried to mirror real regulatory networks, while keeping the model simple and easily extendable. We simulated a total of 110 datasets, with 11 different topologies.

Network generation

One of the main processes involved in cellular dynamic processes is gene regulation, where regulatory cascades and feedback loops lead to progressive changes in expression and decision making. The exact way a cell chooses a certain path during its differentiation is still an active research field, although certain models have already emerged and been tested in vivo. One driver of bifurcation seems to be mutual antagonism, where genes[5] strongly repress each other, forcing one of the two to become inactive[6]. Such mutual antagonism can be modelled and simulated[7,8]. Although such a two-gene model is simple and elegant, the reality is frequently more complex, with multiple genes (grouped into modules) repressing each other[9].

To simulate certain trajectory topologies, we therefore designed module networks in which the cells follow a particular trajectory topology given certain parameters. Two module networks generated linear trajectories (linear and linear long), one generated a bifurcation, one generated a convergence, one generated a multifurcation (trifurcating), two generated a tree (consecutive bifurcating and binary tree), one generated an acyclic graph (bifurcating and converging), one generated a complex fork (trifurcating), one generated a rooted tree (consecutive bifurcating) and two generated simple graph structures (bifurcating loop and bifurcating cycle). The structure of these module networks is available at https://github.com/dynverse/dyngen/tree/master/inst/ext_data/modulenetworks.

From these module networks we generated gene regulatory networks in two steps: the main regulatory network was first generated, and extra target genes from real regulatory networks were added. For each dataset, we used the same number of genes as were differentially expressed in the real datasets. 5% of the genes were assigned to be part of the main regulatory network, and were randomly distributed among all modules (with at least one gene per module). We sampled edges between these individual genes (according to the module network) using a uniform distribution between 1 and the number of possible targets in each module. To add additional target genes to the network, we assigned every regulator from the network to a real regulator in a real network (from regulatory circuits[10]), and extracted for every regulator a local network around it using personalized pagerank (with damping factor set to 0.1), as implemented in the `page_rank` function of the *igraph* package.

Simulation of gene regulatory systems using thermodynamic models

To simulate the gene regulatory network, we used a system of differential equations similar to those used in evaluations of gene regulatory network inference methods[4]. In this model, the changes in gene expression (x_i) and protein expression

(y_i) are modeled using ordinary differential equations[3] (ODEs):

$$\frac{dx_i}{dt} = \underbrace{m \times f(y_1, y_2, ...)}_{\text{production}} - \underbrace{\lambda \times x_i}_{\text{degradation}}$$

$$\frac{dy_i}{dt} = \underbrace{r \times x_i}_{\text{production}} - \underbrace{\Lambda \times y_i}_{\text{degradation}}$$

where m, λ, r and Λ represent production and degradation rates, the ratio of which determines the maximal gene and protein expression. The two types of equations are coupled because the production of protein y_i depends on the amount of gene expression x_i, which in turn depends on the amount of other proteins through the activation function $f(y_1, y_2, ...)$.

The activation function is inspired by a thermodynamic model of gene regulation, in which the promoter of a gene can be bound or unbound by a set of transcription factors, each representing a certain state of the promoter. Each state is linked with a relative activation α_j, a number between 0 and 1 representing the activity of the promoter at this particular state. The production rate of the gene is calculated by combining the probabilities of the promoter being in each state with the relative activation:

$$f(y_1, y_2, ..., y_n) = \sum_{j \in \{0,1,...,n^2\}} \alpha_j \times P_j$$

The probability of being in a state is based on the thermodynamics of transcription factor binding. When only one transcription factor is bound in a state:

$$P_j \propto \nu = \left(\frac{y}{k}\right)^n$$

where the hill coefficient n represents the cooperativity of binding and k the transcription factor concentration at half-maximal binding. When multiple regulators are bound:

$$P_j \propto \nu = \rho \times \prod_j \left(\frac{y_j}{k_j}\right)^{n_j}$$

where ρ represents the cooperativity of binding between the different transcription factors.

P_i is only proportional to ν because ν is normalized such that $\sum_i P_i = 1$.

To each differential equation, we added an additional stochastic term:

$$\frac{dx_i}{dt} = m \times f(y_1, y_2, ...) - \lambda \times x_i + \eta \times \sqrt{x_i} \times \Delta W_t$$
$$\frac{dy_i}{dt} = r \times x_i - \Lambda \times y_i + \eta \times \sqrt{y_i} \times \Delta W_t$$

with $\Delta W_t \sim \mathcal{N}(0, h)$.

Similar to GeneNetWeaver[3], we sample the different parameters from random distributions, defined as follows. e defines whether a transcription factor activates (1) or represses (-1), as defined within the regulatory network network.

$$
\begin{aligned}
r &= \mathcal{U}(10, 200) \\
d &= \mathcal{U}(2, 8) \\
p &= \mathcal{U}(2, 8) \\
q &= \mathcal{U}(1, 5) \\
a_0 &= \begin{cases} 1 & \text{if } |e| = 0 \\ 1 & \text{if } \forall x \in e,\, x = -1 \\ 0 & \text{if } \forall x \in e,\, x = 1 \\ 0.5 & \text{otherwise} \end{cases} \\
a_i &= \begin{cases} 0 & \text{if } \exists x \in e_i,\, x = -1 \\ 1 & \text{otherwise} \end{cases} \\
s &= \mathcal{U}(1, 20) \\
k &= y_{max}/(2 * s), \\
&\quad \text{where } y_{max} = r/d \times p/q \\
c &= \mathcal{U}(1, 4)
\end{aligned}
$$

We converted each ODE to an SDE by adding a chemical Langevin equation, as described in[3]. These SDEs were simulated using the Euler–Maruyama approximation, with time-step $h = 0.01$ and noise strength $\eta = 8$. The total simulation time varied between 5 for linear and bifurcating datasets, 10 for consecutive bifurcating, trifurcating and converging datasets, 15 for bifurcating converging datasets and 30 for linear long, cycle and bifurcating loop datasets. The burn-in period was for each simulation 2. Each network was simulated 32 times.

Simulation of the single-cell RNA-seq experiment

For each dataset we sampled the same number of cells as were present in the reference real dataset, limited to the simulation steps after burn-in. These cells were sampled uniformly across the different steps of the 32 simulations. Next, we used the Splatter package[2] to estimate the different characteristics of a real dataset, such as the distributions of average gene expression, library sizes and dropout probabilities. We used Splatter to simulate the expression levels $\lambda_{i,j}$ of housekeeping genes i (to match the number of genes in the reference dataset) in every cell j. These were combined with the expression levels of the genes simulated within a trajectory. Next, true counts were simulated using $Y'_{i,j} \sim \text{Poisson}(\lambda_{i,j})$. Finally, we simulated dropouts by setting true counts to zero by sampling from a Bernoulli distribution using a dropout probability $\pi^D_{i,j} = \frac{1}{1+e^{-k(\ln(\lambda_{i,j})-x_0)}}$. Both x_0 (the midpoint for the dropout logistic function) and k (the shape of the dropout logistic function) were estimated by Splatter.

This count matrix was then filtered and normalised using the pipeline described below.

Gold standard extraction

Because each cellular simulation follows the trajectory at its own speed, knowing the exact position of a cell within the trajectory topology is not straightforward. Furthermore, the speed at which simulated cells make a decision between two or more alternative paths is highly variable. We therefore first constructed a backbone expression profile for each branch within the trajectory. To do this, we first defined in which order the expression of the modules is expected to change, and then generated a backbone expression profile in which the expression of these modules increases and decreases smoothly between 0 and 1. We also smoothed the expression in each simulation using a rolling mean with a window of 50 time steps, and then calculated the average module expression along the simulation. We used dynamic time warping, implemented in the dtw R package[11,12], with an open end to align a simulation to all possible module progressions, and then picked the alignment which minimised the normalised distance between the simulation and the backbone. In case of cyclical trajectory topologies, the number of possible milestones a backbone could progress through was limited to 20.

dyntoy

For more simplistic data generation (“toy” datasets), we created the dyntoy workflow (github.com/dynverse/dyntoy) . We created 12 topology generators (described below), and with 10 datasets per generator, this lead to a total of 120 datasets.

We created a set of topology generators, were $B(n, p)$ denotes a binomial distribution, and $U(a, b)$ denotes a uniform distribution:

- Linear and cyclic, with number of milestones $\sim B(10, 0.25)$
- Bifurcating and converging, with four milestones
- Binary tree, with number of branching points $\sim U(3, 6)$
- Tree, with number of branching points $\sim U(3, 6)$ and maximal degree $\sim U(3, 6)$

For more complex topologies we first calculated a random number of "modifications" $\sim U(3, 6)$ and a $deg_{max} \sim B(10, 0.25) + 1$. For each type of topology, we defined what kind of modifications are possible: divergences, loops, convergences and divergence-convergence. We then iteratively constructed the topology by uniformly sampling from the set of possible modifications, and adding this modification to the existing topology. For a divergence, we connected an existing milestone to a number of a new milestones. Conversely, for a convergence we connected a number of new nodes to an existing node. For a loop, we connected two existing milestones with a number of milestones in between. Finally for a divergence-convergence we connected an existing milestone to several new milestones which again converged on a new milestone. The number of nodes was sampled from $\sim B(deg_{max} - 3, 0.25) + 2$

- Looping, allowed loop modifications
- Diverging-converging, allowed divergence and converging modifications
- Diverging with loops, allowed divergence and loop modifications
- Multiple looping, allowed looping modifications
- Connected, allowed looping, divergence and convergence modifications
- Disconnected, number of components sampled from $\sim B(5, 0.25) + 2$, for each component we randomly chose a topology from the ones listed above

After generating the topology, we sampled the length of each edge $\sim U(0.5, 1)$. We added regions of delayed commitment to a divergence in a random half of the cases. We then placed the number of cells (same number as from the reference real dataset), on this topology uniformly, based on the length of the edges in the milestone network.

For each gene (same number as from the reference real dataset), we calculated the Kamada-Kawai layout in 2 dimensions, with edge weight equal to the length of the edge. For this gene, we then extracted for each cell a density value using a bivariate normal distribution with $\mu \sim U(x_{min}, x_{min})$ and $\sigma \sim U(x_{min}/10, x_{min}/8)$. We used this density as input for a zero-inflated negative binomial distribution with $\mu\, U(100, 1000) \times$ *density*, $k\, U(\mu/10, \mu/4)$ and pi from the parameters of the reference real dataset, to get the final count values.

This count matrix was then filtered and normalised using the pipeline described below.

PROSSTT

PROSSTT is a recent data simulator[1], which simulates expression using linear mixtures of expression programs and random walks through the trajectory. We used 5 topology generators from dyntoy (linear, bifurcating, multifurcating, binary tree and tree), and simulated for each topology generator 10 datasets using different reference real datasets. However, due to frequent crashes of the tool, only 19 datasets created output and were thus used in the evaluation.

Using the `simulate_lineage` function, we simulated the lineage expression, with parameters $U(0.01, 0.1)$, *branch-tol*$_{intra} \sim U(0, 0.9)$ and *branch-tol*$_{inter} \sim U(0, 0.9)$. These parameter distributions were chosen very broad so as to make sure both easy and difficult datasets are simulated. After simulating base gene expression with `simulate_base_gene_exp`, we used the `sample_density` function to finally simulate expression values of a number of cells (the same as from the reference real dataset), with $\alpha \sim$ *Lognormal* ($\mu = 0.3$ and $\sigma = 1.5$) and $\beta \sim$ *Lognormal* ($\mu = 2$ and $\sigma = 1.5$). Each of these parameters were centered around the default values of PROSSTT, but with enough variability to ensure a varied set of datasets.

This count matrix was then filtered and normalised using the pipeline described below.

Splatter

Splatter[2] simulates expression values by constructing non-linear paths between different states, each having a distinct expression profile. We used 5 topology generators from dyntoy (linear, bifurcating, multifurcating, binary tree and tree), and simulated for each topology generator 10 datasets using different reference real datasets, leading to a total of 50 datasets.

We used the `splatSimulatePaths` function from Splatter to simulate datasets, with number of cells and genes equal to those in the reference real dataset, and with parameters *nonlinearProb*, *sigmaFac* and *skew* all sampled from $U(0, 1)$.

Supplementary References

1. Papadopoulos, N., Parra, R. G. & Soeding, J. PROSSTT: Probabilistic simulation of single-cell RNA-seq data for complex differentiation processes. *bioRxiv* 256941 (2018). doi:10.1101/256941

2. Zappia, L., Phipson, B. & Oshlack, A. Splatter: Simulation of single-cell RNA sequencing data. *Genome Biology* **18**, 174 (2017).

3. Schaffter, T., Marbach, D. & Floreano, D. GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods. *Bioinformatics (Oxford, England)* **27**, 2263–2270 (2011).

4. Marbach, D. *et al.* Wisdom of crowds for robust gene network inference. *Nature methods* **9**, 796–804 (2012).

5. Xu, H. *et al.* Regulation of bifurcating B cell trajectories by mutual antagonism between transcription factors IRF4 and IRF8. *Nature Immunology* **16**, 1274–1281 (2015).

6. Graf, T. & Enver, T. Forcing cells to change lineages. *Nature* **462**, 587 (2009).

7. Wang, J., Zhang, K., Xu, L. & Wang, E. Quantifying the Waddington landscape and biological paths for development and differentiation. *Proceedings of the National Academy of Sciences* **108**, 8257–8262 (2011).

8. Ferrell, J. E. Bistability, Bifurcations, and Waddington's Epigenetic Landscape. *Current Biology* **22**, R458–R466 (2012).

9. Yosef, N. *et al.* Dynamic regulatory network controlling Th17 cell differentiation. *Nature* **496**, 461–468 (2013).

10. Marbach, D. *et al.* Tissue-specific regulatory circuits reveal variable modular perturbations across complex diseases. *Nature Methods* **13**, 366 (2016).

11. Giorgino, T. Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. *Journal of Statistical Software* **31**, (2009).

12. Tormene, P., Giorgino, T., Quaglini, S. & Stefanelli, M. Matching incomplete time series with dynamic time warping: An algorithm and an application to post-stroke rehabilitation. *Artificial Intelligence in Medicine* **45**, 11–34 (2009).