Scanpy Wolf, Angerer & Theis, bioRxiv (2017)

Analysis of large-scale scRNA-seq data

F. Alexander Wolf, Institute of Computational Biology, Helmholtz Munich November 7, 2017 - Video talk for Regev Lab - Broad Institute

Scanpy vs. Seurat

Satija et al., Nat. Biotechn. (2015)

Scanpy is benchmarked with Seurat.

- preprocessing: <1 s vs. 14 s
- regressing out unwanted sources of variation: 6 s vs. 129 s
- PCA: <1 s vs. 45 s
- clustering: 1.3 s vs. 65 s
- tSNE: 6 s vs. 96 s
- marker genes (approximation):
 0.8 s vs. 96 s

El theislab / scanpy_usag	e	60 Unwitch + 2 Wister 3 Vice 1
O Corle 🕕 Issues 🖞	1) Pull requests (b) III Projects (b) IIII Wild	🔟 insights 🛛 🗇 Sattings
bands mater - Soanpy_us	age / 170505_searct /	Greate sen file Upload files file file file by
\mathcal{D} falsewelt oddad regressing a	sut ta recolma	Latast commit el detta 10 essends apo
m Toures	upcated for version 0.2.9.1	38 minutes ago
E READMEIND	added regreceleg out to readme	10 econde ago
≧ secret/pyrb	opclated for version 0.1.5.1	38 minutes egu
≦ secrat_R jrynb	updated for version 0.5.9.1	38 minutes ago
III REALINE INT		

First complex: May 5, 2017. See the notebook

Scanpy versus Seural

Scangy provides a number of Secrat's leafures (satis et al., Nat. Rodebhool, 2006), but at significantly righer computationally efficiency. Here, we reproduce most of Secret's guided clustering tutorial as compiled on March 30, 2017. The tutorial starts with preprocessing and ends with the identification of cell types through marker genes of clusters. The data consiste in *Sk PBMCe from a Healthy* Conor and is freely available from 10x (here from this webpage). The profiling information for Secret has been obtained within secret_Ripynb.



Scanpy vs. Cell Ranger for 68k cells

Zheng et al., Nat. Commun. (2017)

Scanpy is benchmarked with Cell Ranger R kit.

- preprocessing: 14 s vs. 300 s
- PCA: 17 s vs. 120 s
- tSNE 5 min vs. 26







Scanpy scales to >1 million cells

Zheng et al., Nat. Commun. (2017)



AnnData github.com/theislab/anndata, pypi/anndata

Simple class for a data matrix with most general annotations.

Nothing like this in Python.

- *.loom* (Python, merely a file format)
- VariantDataset (Python, Java, hail)
- ExpressionSet (R)
- "Seurat Object" (R, Seurat)
- CellDataSet (R, Monocle)
- SingleCellExperiment (R, Scran)





Characteristics of single-cell data Goal Learn abstractions of biology (e.g. cellular identities), Rare subtype associations and mechanisms. Discrete types Data Continuous phenotypes high-dimensional Regulatory Pro-inflammatory Revisiting a unstructured previous state Erythrocyte sparse Source state T-lymphocyte Unidirectional State vacillation noisy temporal progression

• non-linear

figure from Review Wagner et al., Nat Biotechn (2016)

Represent single-cell data

- High-dimensional data → make guess for distance metric d(x, y) → evaluate d locally → generate neighborhood graph of single cells
- Typically, obtain *d* from preprocessing and something like euclidean distance.
- Alternatively, learn d:





DataGraph

Class for representing data as a graph of neighborhood relations between data points.

Simplest case: knn graph.

- much faster than *sklearn.neighbors*
- much faster than R-wrapped C++

One idea: use matrix-multiplication for submatrices of data matrix in parallel.

Also, *DataGraph* offers many functions related to stochastic processes on graphs, absent in *igraph*, *networks*, *graph-tools*.



Scanpy tools operate on DataGraph

A single framework for common analysis tasks.

clustering

Levine et al., Cell (2015), Xu et al. Bioinf (2015) ...

 pseudotime and trajectory inference

Trapnell et al., Bendall et al. (2014), Haghverdi et al. (2016) ...



Reconciling both:

• graph abstraction

Wolf et al., bioRxiv (2017)

Scanpy's API

Modular and intuitive API.

- sc.preprocessing
- sc.tools
- sc.plotting
- sc.datasets
- sc.settings

. . .

Command-line interface...

Docs #Exphires/VBI

OF deep GH lub

Features/API

Scopy's high-level API provides on overview of all frohums relevant to protict luse

import scorpy spi as so

Preprocessing tools

Filtering of highly-variable genes, batch reflect correction, periodi (UMI) normalization, proposessing recipes

Basic Preprocessing

getmater_cause (data), min_counts,)	Piter cell celliers based on counts and numbers of gene		
ge Hiltor, genes (dort (min_ (ells, -)	Filter genes have don ministral number of only encounts		
ge Hilter, generalit sparales (1900), flavor, -]	Filter grows have don elegention extract highly variable		
ee.cogaa (Catal, copy)	Logarithmise the data matrix.		
to the (and characteristic fit	Principal removement analysis (Peringes all 1)		
gg.marmax.ton_per_cect.(dista),]	Normalize each cail.		
$g_{ij} = e_{ij} e_{ij$	Regression and an anticipation set of variation		
the ways (care) second states to activative code()	Scale data to unit variance and zero mean.		
estrances a kisk fredien sed)	Subsample to a fraction of the number of samples.		

Recipes

pp.res.(pr.steng#" (addta), r_bop_gares,)	Normalization and Nitering as of [Zhang 37].		
ast region perimeters (adats), mean threshold (Normalization and filtering as of (Weinrebt27).		

Machine Learning and Statistics tools

Visualization

$[\mathbf{x}_1]_{1 \leq n} [(\mathbf{x}_1)_{i \in \mathbb{N}} (\mathbf{x}_1)_{i \in \mathbb{N}} (\mathbf{x}_2)_{i \in \mathbb{N}} (\mathbf{x}_$	Principal component analysis [Pertragos at 1]
st. sawe (adulta), rupos, perpiesitg	1-2N2 (Master/00) (Amir13) (Pedragorr11),
studitimas (odata), nuomps nuncightors	Diffusion Maps [Collman05] (Haghverdid5) (Walta7),
stration_graph [addts]; layout, root,]	Force-cirected graph drawing (Fruchterman 71) (Weine

Branching trajectories and pseudotime, clustering, differential expression

$(1, \eta_{\rm BR}) ({\rm Matrix}) = ({\rm matrix}) ({\rm matrix}) ({\rm matrix}) ({\rm matrix}) ({\rm matrix}) = 0$	Generate cell (an users of differentiation medicities with
$\label{eq:states} w_{i}, w_{$	Cluster cells into subgroups [Closeds 60] [Lewins 15] [Th
$[\pi_1, a_2, [a_2, a_3]]$ is branching a run lightwise J	Infer progression of odis, identify insuching subgroups
(1) sant genes, groups [2data group(n)], [1]	Rank genes according to differential expression (Wolff

Simulations

(t six (model) max, hearthing = [) Simulate dynamic gene expression data (with manol/?) [whif17].

Generic methods

Reading and Writing

 could blear regregation (shore, shore, shore)
 Read blear regregation (shore, shore)

 write (the range of the shore, shore,

Data Structures

Andrea (1977), fo	np.sor, and, drype, single, only	Steer on an entropy data matrix.
managem beiete	lik kanan laha mana Ji	Data service techa smith of maximilarity as

Plotting

Generic plotting with AnnData

putaceter (ideta) a y color alpha ut	Satter plot.
$p_{1,whete} = [adata, keys], group, by, (ther,)$	Wolin plot.
p. anneng (adeta attr. keys), lebels, a ?	Plot no Ning.

Plotting tool results

Methods that extract and visualize tool-specific amotation in an AmData object.

Viscalization

pt pre (adam, injurant)	Plot PCA results.	
as any manage (added, components above $\mathrm{sys}($	Renkgama according to contributions to PCs.	
ρ^{2} produces the (column) reside alpha $-\beta$	Scattor plot in PCA scordinates	
an analysements (more (ladeta), log show save)	Not the verter constitu-	
pt , case (adults), caller, sights, groups,	Scatter plot in tSNE basis.	
et entrese (odata) coincialoite, groups, - 3	Scatter plot in Diffusion Map has's	
p^{1} -sime groups (backets), inspect, radius, alpha, - \underline{J}	Scatter plot in graph deaxing locks	

Branching trajectories and pseudotime, clustering, differential expression

at tags (atlate), bush, color a(dra.groups,))	Summary figura for appreciately graph abstraction.
phage parts (March With Network)	First the observation graph
at applying (induct) works keys _])	One opression durigos along paths in the abstractor
pt. Asseste () adata(, basis, color, alpha,)	Fibilities the of Louvein chartening
$\mu^{*} \to \mu^{*}$ (where) have, every signal groups -1	Flat results of DPT analysis
al sector (deta), basis, color alpha,)	Scattor plot of DPT results.
plusipingeoups, paralleline (Addite), triker, and $\tau=0$	First groups and presidentine
ρ^{1} -spectra sectors (addres) entry map, shows , ,]	Heatmap of previous instances
a statute prove (statut provincia)	Pibli ranking of years.
pi mani, grow, groups, pintin (tdalaj, provot)	Fibil ranking of gener for all hested comparisons.

Simulations

Builtin datasets

Simple functions that provide another distances for benchmarking. See here for extensive documented to torials and use cases.

All of these functions return an Annotated Data object.

datasets parties ()	Get logarithmized data for development of Myelold Progeni
Automoto togglicavitete.))	Simple traggleswitch term simulated data
акониссализиот окоз ()	Simulated mycloid progeniter data.
decreases in the $[0,100000,00000,0001,0]$	Minise Calendary Bloke.

Scanpy's use cases

Docs a Examples

O Edition GitHub

Examples

Good starting points are the following examples, which build on established results from the literature. All examples are versioned on GitHub.

Example 1: Secrat's [Setija15] guided clustering tutorial.



Example 2: The Diffusion Pseudotime (DP1) analyses of [Disgloverdi16] for date of [Paul16] and [Molgrand15]. Note that DPT has recently been very favorably discussed by the authors of Monodo.



Example 3: Analyzing AB 300 cells from [Zheng12], we find that Scanpy is about a factor bito 14factor and more memory efficient than the Cell Ranger R kit for secondary analysis.



Example 4: Visualizing 1.9 mio brain cells.



Example 5: Simulating single cells using literature-cursted gene regulatory betworks (Wirtmann122): here, myeloid differentiation (Reumaink 11).



Example 6: Pseudotime-based vs. deec-learning based reconstruction of cell cycle from image data [Eulenberg17].



theislab / graph_abstracti	ion	I Poleta B	O Unweith - 3	# Star 8 V Tank
enerate celular maps of differ	entiation manife	olds with complex topologie	s.	
② 26 commits	ji 1 branch	⊖ @ releases	AL 1 contributor	⊕ MIT
insich meeters Nex pull request			Greate new file Upload files fit	of the Charace downlo
talescell updated readme			Later	1 commit 955/338 12 days
lt deep,learning		updated everything		13 days
minimal_avamples		matching the preprint		13 days
is restorowalts		updated readmes		12 days
B paul 5		updated readmes		12 days
a phones		removed 33k dataset		13 days
la planaria		matching the preprint		13 days
anightere		initial commit		2 months
E LICENSE		add license		2 months

Graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells

This repository allows to reproduce analyses and figures of the preprint.

Graph abstraction is available within Scangy. Central toplevel functions are:

- scampy.api.tools.apa
- scampy.api.plotting.apa_prash
- scampy.api.plotting.apa_path

Minimal examples with known ground truth





and simple datasets that illustrate connectivity patterns of clusters.



Differentiation manifolds in hematopolesis

Here, we consider two well-studied datasets on hematopoietic differentiation.

Data from Paul et al. (2015)

In paul15, we analyze data for myeloid progenitor development. This is the same data, which has served as benchmark for Monocle 2 (Qiu et al., Nat. Meth., 2017) and DPT (Haghverdi et al., Nat. Meth., 2016).



Data from Nestorowa, Harney et al. (2016)

In nestoroward, we analyze data for early hematopoletic differentation



Lineage tree for whole cell atlas of an adult animal

In planaria, we reconstruct the lineage tree of the whole cell atlas of planaria (Plass, Jordi et al., submitted, 2017).



Deep Learning

In deep_learning, we use deep learning to generate a feature space and, by that, a distance metric, which induces a nearest-neighbor graph. For the problem of reconstructing cell-cycle Eulenberg, Köhler, et al., Nat. Commun. (2017), we find that graph abstraction correctly separates a small cluster of dead cells from the cell evolution through 05, S and 02 phese.





PBMC cells

For all of the following scRNA-seq datasets (3K and 88K PBMC cells, all 10X Genomics), graph abstraction reconstructs correct lineage motifs. As the data is disconnected in large parts, a global lineage tree cannot be informed.





Outlook

Very-short term

- common file format for backing AnnData
- AnnData based on pandas dataframes instead of structured arrays

Mid-term

- aggregation of datasets
- better correction for confounders
- include any standard, canonical analysis method...
- module-wise installation (reduce dependencies?)

Long-term

• mini-batch learning

Thanks to

Machine Learning group at Helmholtz Munich, in particular, Philipp Angerer and Fabian Theis.

Thank you for your attention!

Scanpy code snippets

In [3]: filename_data = './data/pbmc3k_filtered_gene_bc_matrices/hg19/matrix.mtx'
filename_genes = './data/pbmc3k_filtered_gene_bc_matrices/hg19/genes.tsv'
filename_barcodes = './data/pbmc3k_filtered_gene_bc_matrices/hg19/barcodes.tsv'
adata = sc.read(filename_data).transpose()
adata.var_names = np.loadtxt(filename_genes, dtype='S')[:, 1]
adata.smp_names = np.loadtxt(filename_barcodes, dtype='S')

reading file ./write/data/pbmc3k_filtered_gene_bc_matrices/hg19/matrix.h5

Basic filtering.

```
In [4]: adata.smp['n_counts'] = np.sum(adata.x, axis=1).Al
sc.pp.filter_cells(adata, min_genes=200)
sc.pp.filter_genes(adata, min_cells=3)
```

- ... filtered out 0 outlier cells
- ... filtered out 19024 genes that are detected in less than 3 cells

Plot some information about mitochondrial genes, important for quality control

A violin plot of the computed quality measures.

In [6]: sc.pl.violin(adata, ['n_genes', 'n_counts', 'percent_mito'], jitter=0.4, show=True)



- In (9): sc.pp.normalize_per_cell(adata, scale_factor=le4)
 result = sc.pp.filter_genes_dispersion(adata.X, log=True,
 sc.pl.filter_genes_dispersion(result)
 - ... filter highly varying genes by dispersion and mean using `min_disp`, `max_disp`, `min_mean` and `max_mea --> set `n_top_genes` to simply select top-scoring genes



In (11): adata_corrected = sc.pp.regress_out(adata,

amp_keya=['n	counts	, 'per	cent_mi
copy= True)			

0:00:00.000 - regress out ['n_counts', 'percent_mito'] ... sparse input is densified and may lead to huge memory consumption

0:00:09.418 - finished

Compute PCA and make a scatter plot.

In [12]: sc.pp.scale(adata_corrected, max_value=10)

clipping at max_value 10

In [13]: sc.tl.pca(adata_corrected)

adata_corrected.smp['X_pca'] *= -1 # multiply by 1 for correspondence sc.pl.pca_scatter(adata_corrected, color='CST3', right_margin=0.2)

0:00:00.000 - compute PCA with n_comps = 10 0:00:00.668 - finished, added the data representation "X_pca" (adata.smp) the loadings "PC1", "PC2", ... (adata.var) and "pca variance ratio" (adata.add)

