Technical Report

pyinfercnv: a Python implementation of inferCNV for copy-number inference from single-cell RNA-seq data

OmicVerse Team[1,*]
[1]OmicVerse, Inc., San Francisco, CA 94114, USA.
[2]School of Public Health, Guangzhou Medical University, Guangzhou 511436, China.
[3]Department of Genetics, Stanford University, Stanford, CA 94305, USA.
*Correspondence: contact@omicos.cn
OmicVerse Team members who contributed to this report: Jieshen Xian[1,2], Zehua Zeng[1,3].

Summary

We present pyinfercnv, a Python implementation of inferCNV, which infers copy-number alterations of single cells from their expression relative to reference cells and calls copy-number states with a hidden Markov model. pyinfercnv follows inferCNV's code path, including R's random number generator, R's smoothing spline and Seurat's steps for subclustering, so that the same seed gives the same result as in R. On the oligodendroglioma example of inferCNV, pyinfercnv returned the same subclusters and copy-number states as inferCNV 1.26.0 and recovered the 1p/19q co-deletion. On six breast tumours from 10x Genomics (10,424 cells), its relative expression agreed with inferCNV's to 1.2×10^{-7} and 98.2 to 99.5% of its copy-number states were the same; the remaining differences arise from the approximate nearest-neighbour search that inferCNV uses for subclustering. pyinfercnv ran 41 to 64 times faster than inferCNV on one thread. It is used by OmicVerse for copy-number inference and is released at `https://github.com/omicverse/py-inferCNV` and on PyPI.

Keywords inferCNV, copy-number variation, hidden Markov model, tumour heterogeneity, single-cell RNA-seq, Python, reproducibility

Introduction

Copy-number alterations distinguish malignant from normal cells in most tumours, and in single-cell RNA sequencing they leave a trace in expression: the average expression of many genes along a chromosome arm rises with a gain and falls with a loss. Inferring copy number from expression has become a routine step to identify malignant cells and their subclones in tumour atlases [1–3]. inferCNV, developed at the Broad Institute, implements this inference [4]. It compares the expression of observation cells with that of reference normal cells along the genome and calls copy-number states with a hidden Markov model. Other methods include CopyKAT [5] and Numbat [6].

Single-cell analysis is increasingly done in Python. Workflows are built on scanpy and the scverse ecosystem [7,8], and OmicVerse [9] brings bulk, single-cell and spatial analyses together in one Python framework. Python offers infercnvpy [10], a scanpy plugin that follows inferCNV's approach, but inferCNV itself is available only in R, as a Bioconductor package.

Python users currently reach inferCNV through rpy2 [11]. This requires a parallel R installation with JAGS and Seurat, converts the expression matrix between the two languages and exposes inferCNV's options only through the binding layer. A native implementation removes all three costs, as PyDESeq2 [12] has shown for DESeq2 [13] and pyedger [14] for edgeR [15]. We set the same target for pyinfercnv: an inference in Python should return the same copy-number calls as the same inference in R.

Here we present pyinfercnv, a Python implementation of inferCNV that is versioned against a specific inferCNV release. We describe its implementation and compare it with inferCNV 1.26.0 on the oligodendroglioma example of inferCNV and on six breast tumours from 10x Genomics.

Implementation

Method

inferCNV takes the counts of observation and reference cells and the genomic positions of the genes [4]. After removing lowly expressed genes, it normalises the counts by sequencing depth, log-transforms them and subtracts the mean expression of the reference cells from every gene. The relative expression is smoothed along each chromosome with a moving pyramidal window of 101 genes, centred within each cell and corrected again by the references, which gives each cell's relative expression along the genome (Fig. 1d). The cells of each annotation group are then divided into subclusters by Leiden clustering [16] of a shared-nearest-neighbour graph built with Seurat [17]. A hidden Markov model with six states, from complete loss to amplification, is fitted to each subcluster's mean profile; its emission parameters are calibrated on simulated cells with known copy numbers, the hidden spike. Optional steps filter states with a Bayesian network, mask genes that are not differentially expressed and denoise the relative expression

© 2026 The Authors
Published at omicverse.org/articles/pyinfercnv, version 1.

by the variation in the reference cells.

Features and code structure

pyinfercnv 0.2.1 implements `infercnv::run` of inferCNV 1.26.0 (Table 1): the relative expression, the subclustering, the six- and three-state hidden Markov models with the hidden-spike calibration, the Bayesian-network filter, the masking of non-differentially expressed genes and the denoising. `infercnv` takes an AnnData object[18] with gene positions and stores its results in the object. pyinfercnv builds on NumPy[19], SciPy[20], pandas[21] and python-igraph; compute-heavy kernels are compiled with Numba.

Numerical parity with inferCNV

pyinfercnv follows the code that inferCNV runs and computes in double precision; its matrices are returned in single precision, which bounds their differences from R's. The random parts of inferCNV draw from R's generator in a fixed order: the hidden-spike simulation, the subclustering and the calibration of the emission variance. pyinfercnv ports R's Mersenne-Twister generator[22] with its sampling and normal and uniform draws, and R's smoothing spline with generalised cross-validation, which the simulation uses for the mean–variance and dropout trends. Before each subclustering, Seurat's PCA sets the seed to 42, so the subclustering does not depend on the user's seed. The Leiden algorithm runs in python-igraph on a generator that delivers R's random numbers as R igraph does, so the same graph gives R's partition. Seurat finds nearest neighbours with Annoy, an approximate search; pyinfercnv uses exact neighbours, so a few cells can join other subclusters. The steps are tested against inferCNV 1.26.0 (216 tests, of which 19 compare with the output of R runs and 3 with stored R values). pyinfercnv versions follow inferCNV releases (0.2.x tracks inferCNV 1.26.0).

Use in OmicVerse

pyinfercnv is the inferCNV backend of OmicVerse[9] copy-number inference: `ov.single.CNV(adata, method='infercnv')`, whose `run` method takes the reference annotation and the platform, runs it on an annotated AnnData and stores the copy-number matrix and a per-cell score.

Comparison with inferCNV

Data and design

We used the oligodendroglioma example of inferCNV, 184 cells profiled with Smart-seq2 from oligodendrogliomas[3], with microglia and non-malignant oligodendrocytes as reference. We also used six breast tumours profiled with 10x Genomics, a ductal carcinoma in situ and five triple-negative breast cancers[5] (532 to 3,225 cells per tumour, 10,424 in total), with the cells that CopyKAT called normal as reference. Both packages ran inferCNV with subclusters, the six-state hidden Markov model and denoising, from the same seed (Supplementary methods). We compared the relative expression and the denoised matrices, the subclusters, the hidden Markov model states and run time.

Relative expression, subclusters and copy-number states

The relative expression and the denoised matrices agreed with inferCNV in every dataset, to a largest absolute difference of 1.2×10^{-7} (Fig. 1a). In the oligodendroglioma example, pyinfercnv returned the same six subclusters and the same hidden Markov model state for every gene in every cell. Its 142 malignant cells showed the 1p/19q co-deletion that defines oligodendroglioma: relative expression was 22.8% lower on 1p and 18.9% lower on 19q than in the reference cells, and 1.8% higher on the rest of the genome, in R and in Python (Fig. 1d, e).

In the breast tumours, the hidden Markov model states of 98.2 to 99.5% of the genes and cells agreed with inferCNV, and the subclusters had adjusted Rand indices of 0.79 to 0.98 (Fig. 1b). The two packages found similar numbers of subclusters in every tumour, from 18 and 19 in TNBC3 to 60 and 56 in TNBC4. The remaining differences come from the nearest-neighbour graphs, in which Seurat's approximate search differs from the exact one for some cells; on the same graph, the partitions are identical. In TNBC1, the per-cell copy-number score separated the cells that CopyKAT called tumour from the normal cells in the same way in both packages (Fig. 1f–i).

Speed

pyinfercnv ran 41 to 64 times faster than inferCNV (Fig. 1c). For TNBC5, the largest tumour (3,225 cells), inferCNV took 751 s and pyinfercnv 16 s; for the oligodendroglioma example, 201 s and 3.2 s.

Conclusion and future perspectives

pyinfercnv brings inferCNV to Python with the same results as R. From the same seed, it reproduces inferCNV 1.26.0's relative expression and denoised matrices, and its subclusters and copy-number states wherever the neighbour graphs are the same, while running one to two orders of magnitude faster. Because pyinfercnv works with AnnData objects and is part of OmicVerse, copy-number inference can be done in the same Python session as the rest of a single-cell workflow.

Future work includes the Annoy neighbour search, so that R's default subclustering is reproduced on all data, the reassignment of copy-number states by the Bayesian network, inferCNV's sigmoidal denoising and its heatmaps. pyinfercnv will continue to follow inferCNV releases, with each version pinned to the release it reproduces.

Data and code availability

pyinfercnv is released under the BSD 3-Clause License on PyPI (`pip install pyinfercnv`) and at `https://github.com/omicverse/py-inferCNV`; this report describes version 0.2.1. The benchmark (`report/scripts` in the repository) reproduces every number and figure in this report. The oligodendroglioma example is distributed with inferCNV; the breast tumour count matrices with CopyKAT's tumour and normal calls were obtained from the Gene Expression Omnibus (GSE148673: DCIS1 and TNBC1 to TNBC5), accessed on 12 November 2026. Gene positions for the breast tumours were taken from GENCODE releases 24 and 32. Software: pyinfercnv 0.2.1 (Python 3.10, NumPy 2.2.6, SciPy 1.15.2, pandas 2.3.3, python-igraph 1.0.0; scanpy 1.11.5 for Fig. 1f–i); inferCNV 1.26.0 with Seurat 5.5.1 and igraph 2.3.4 on R 4.5.3 (bioconda). Hardware: one node of the Stanford Sherlock cluster (AMD EPYC 9754).

Table 1. inferCNV steps and their pyinfercnv counterparts.

inferCNV 1.26.0	pyinfercnv	Agreement on the benchmark data
Relative expression (steps 1–14)	infercnv	absolute difference at most 1.2×10^{-7}
Hidden spike (i6 calibration)	hmm.calibrate_i6_emission	identical simulated cells and parameters
Leiden subclusters (step 15)	subcluster.leiden_subcluster_pca	identical in the example; ARI 0.79–0.98
HMM states (step 17)	hmm.predict_i6, predict_i3	identical in the example; 98.2–99.5%
Denoising (step 22)	denoise.denoise_by_ref_mean_sd	absolute difference at most 1.2×10^{-7}

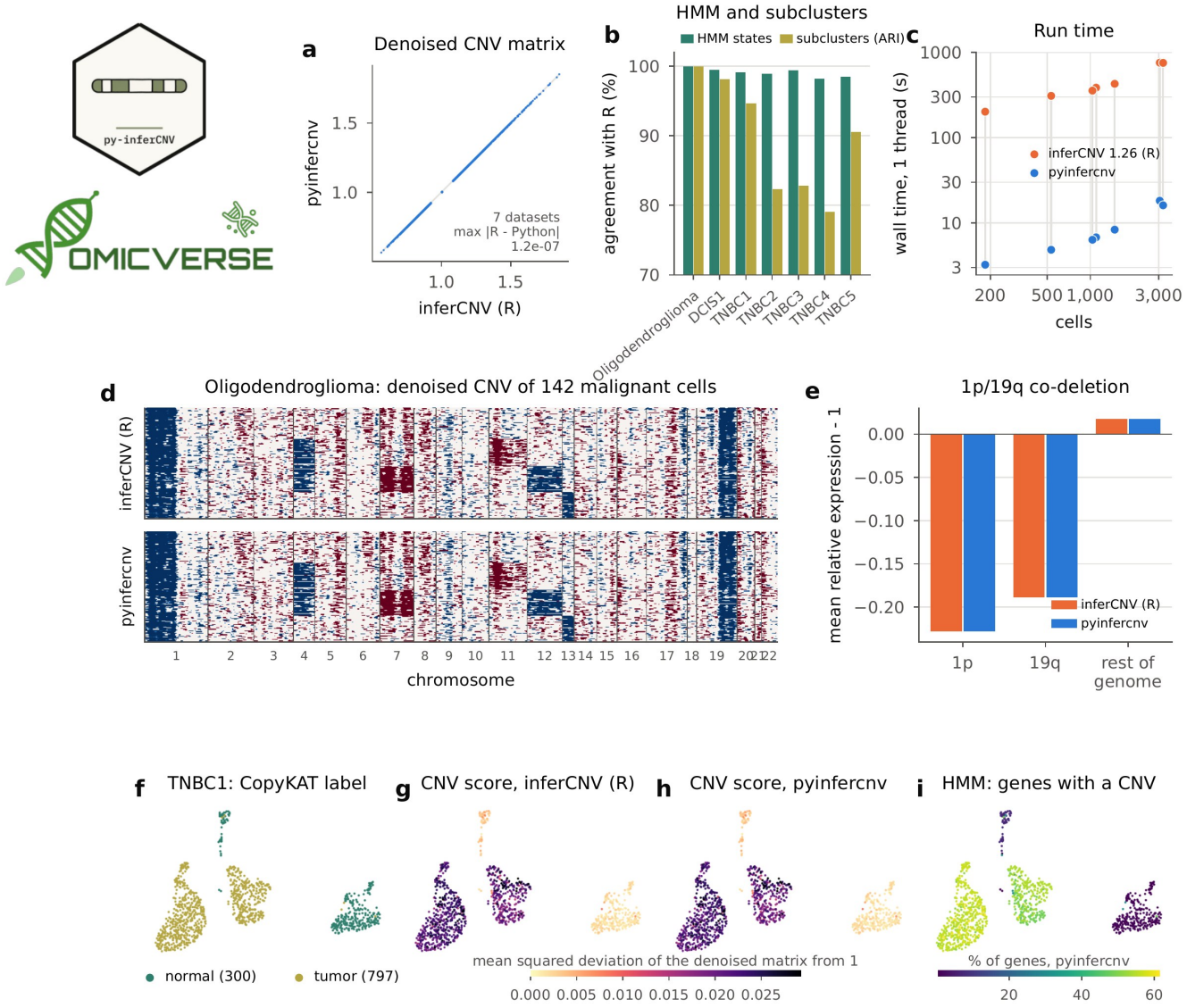


Figure 1. pyinfercnv versus inferCNV 1.26.0 on the oligodendroglioma example and six breast tumours.

(a) Values of the denoised copy-number matrix, pyinfercnv against inferCNV, for 4,000 random entries of each dataset. **(b)** Agreement of the hidden Markov model states (share of genes and cells with the same state) and of the subclusters (adjusted Rand index, weighted over annotation groups). **(c)** Wall time on one thread, the faster of two runs, against the number of cells. **(d)** Denoised relative expression of the 142 malignant cells of the oligodendroglioma example along the genome, from inferCNV and from pyinfercnv. **(e)** Mean relative expression on 1p, 19q and the rest of the genome in those cells. **(f–i)** UMAP[23] of the 1,097 cells of TNBC1, computed with scanpy from the counts: **(f)** tumour and normal cells as called by CopyKAT; **(g, h)** copy-number score (mean squared deviation of the denoised matrix from 1) from inferCNV and pyinfercnv; **(i)** share of genes in a copy-number state in pyinfercnv's hidden Markov model.

Declaration of generative AI and AI-assisted technologies

The authors used Claude Opus 5.5, GPT-6 Astra, and OmicOS to proofread grammar and syntax, improve readability. Any AI-assisted edits were reviewed by the authors, who take full responsibility for the content of this publication.

Conflicts of interest

Contributors Jieshen Xian and Zehua Zeng are affiliated with OmicVerse, Inc.

References

[1] Patel AP, Tirosh I, Trombetta JJ, et al. (2014). Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. Science 344, 1396-1401. 10.1126/science.1254257

[2] Tirosh I, Izar B, Prakadan SM, et al. (2016). Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. Science 352, 189-196. 10.1126/science.aad0501

[3] Tirosh I, Venteicher AS, Hebert C, et al. (2016). Single-cell RNA-seq supports a developmental hierarchy in human oligodendroglioma. Nature 539, 309-313. 10.1038/nature20123

[4] Tickle T, Tirosh I, Georgescu C, Brown M, Haas B. inferCNV of the Trinity CTAT Project. Klarman Cell Observatory, Broad Institute. https://github.com/broadinstitute/inferCNV

[5] Gao R, Bai S, Henderson YC, et al. (2021). Delineating copy number and clonal substructure in human tumors from single-cell transcriptomes. Nature Biotechnology 39, 599-608. 10.1038/s41587-020-00795-2

[6] Gao T, Soldatov R, Sarkar H, et al. (2023). Haplotype-aware analysis of somatic copy number variations from single-cell transcriptomes. Nature Biotechnology 41, 417-426. 10.1038/s41587-022-01468-y

[7] Wolf FA, Angerer P, Theis FJ (2018). SCANPY: large-scale single-cell gene expression data analysis. Genome Biology 19, 15. 10.1186/s13059-017-1382-0

[8] Virshup I, Bredikhin D, Heumos L, et al. (2023). The scverse project provides a computational ecosystem for single-cell omics data analysis. Nature Biotechnology 41, 604-606. 10.1038/s41587-023-01733-8

[9] Zeng Z, Ma Y, Hu L, et al. (2024). OmicVerse: a framework for bridging and deepening insights across bulk and single-cell sequencing. Nature Communications 15, 5983. 10.1038/s41467-024-50194-3

[10] Sturm G. infercnvpy: infer copy number variation from single-cell transcriptomics data. https://github.com/icbi-lab/infercnvpy

[11] Gautier L. rpy2: Python interface to the R language. https://rpy2.github.io

[12] Muzellec B, Teleńczuk M, Cabeli V, Andreux M (2023). PyDESeq2: a python package for bulk RNA-seq differential expression analysis. Bioinformatics 39, btad547. 10.1093/bioinformatics/btad547

[13] Love MI, Huber W, Anders S (2014). Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. Genome Biology 15, 550. 10.1186/s13059-014-0550-8

[14] Zeng Z, Meng X (2026). pyedger: a Python implementation of edgeR for differential expression analysis. Preprints.org. 10.20944/preprints202611.1133.v1

[15] Robinson MD, McCarthy DJ, Smyth GK (2010). edgeR: a Bioconductor package for differential expression analysis of digital gene expression data. Bioinformatics 26, 139-140. 10.1093/bioinformatics/btp616

[16] Traag VA, Waltman L, van Eck NJ (2019). From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports 9, 5233. 10.1038/s41598-019-41695-z

[17] Hao Y, Stuart T, Kowalski MH, et al. (2024). Dictionary learning for integrative, multimodal and scalable single-cell analysis. Nature Biotechnology 42, 293-304. 10.1038/s41587-023-01767-y

[18] Virshup I, Rybakov S, Theis FJ, Angerer P, Wolf FA (2024). anndata: access and store annotated data matrices. Journal of Open Source Software 9, 4371. 10.21105/joss.04371

[19] Harris CR, Millman KJ, van der Walt SJ, et al. (2020). Array programming with NumPy. Nature 585, 357-362. 10.1038/s41586-020-2649-2

[20] Virtanen P, Gommers R, Oliphant TE, et al. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17, 261-272. 10.1038/s41592-019-0686-2

[21] McKinney W (2010). Data structures for statistical computing in Python. Proceedings of the 9th Python in Science Conference, 56-61. 10.25080/Majora-92bf1922-00a

[22] Matsumoto M, Nishimura T (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation 8, 3-30. 10.1145/272991.272995

[23] Becht E, McInnes L, Healy J, et al. (2019). Dimensionality reduction for visualizing single-cell data using UMAP. Nature Biotechnology 37, 38-44. 10.1038/nbt.4314

Supplementary information

Supplementary methods

Data. The oligodendroglioma example of inferCNV contains 184 cells from four tumours[3] (142 malignant cells in four groups, 19 microglia or macrophages and 23 non-malignant oligodendrocytes) and its gene positions. For the breast tumours, the count matrices of GSE148673 were read with the tumour and normal calls that CopyKAT gives for every cell, and the normal cells formed the reference; genes were placed with their GENCODE 24 positions (32 for names missing in release 24).

Runs. In R: `set.seed(1)` and `infercnv::run(object, cutoff, cluster_by_groups = TRUE, denoise = TRUE, HMM = TRUE, BayesMaxPNormal = 0, num_threads = 1, no_plot = TRUE)` with cutoff 1 for Smart-seq2 and 0.1 for 10x Genomics; everything else at its defaults, including the six-state model, subclustering with Leiden on Seurat's PCA and denoising by the standard deviation of the reference cells. In Python: `infercnv(adata, config=InferCNVConfig(cutoff=..., cluster_by_groups=True, denoise=True, HMM=True, BayesMaxPNormal=0, random_state=1))` with the same reference groups.

Comparison. The relative expression after `invert_log2` (step 14) and the denoised matrix (step 22) were compared entry by entry; the HMM states (step 17) by the share of genes and cells with the same state; the subclusters by the adjusted Rand index within each annotation group, weighted by the number of cells.

Chromosome arms (Fig. 1e). Genes on chromosome 1 starting before 121.1 Mb and genes on chromosome 19 starting after 24.2 Mb (the largest gaps in the gene positions, at the centromeres) form 1p and 19q.

Embedding (Fig. 1f–i). The cells of TNBC1 were embedded with scanpy (counts normalised to 10,000 per cell and log-transformed, 2,000 variable genes, 30 principal components, 15 nearest neighbours) and UMAP; the embedding is for display only.

Run time. Run times are wall-clock times of `infercnv::run` and `infercnv`, without reading and writing files. Each run was pinned to its own idle processor core with BLAS and OpenMP limited to one thread, R and Python alternating over 12 cores of one node. Each dataset ran twice per package, and the faster run is reported.

Notes for users coming from R

1. **Versions.** pyinfercnv 0.2.x tracks inferCNV 1.26.0.
2. **Input.** `infercnv` takes an AnnData object with counts and the chromosome, start and end of each gene in `var`; `reference_key` and `reference_cat` name the annotation column and the reference groups.
3. **Seeds.** `random_state=s` gives the result of `set.seed(s)` before `infercnv::run`.
4. **Defaults.** `BayesMaxPNormal` defaults to 0 (R: 0.5), and the Bayesian network's reassignment of states (`reassignCNVs`) is not ported; HMM states are numbered from 0 (R from 1).