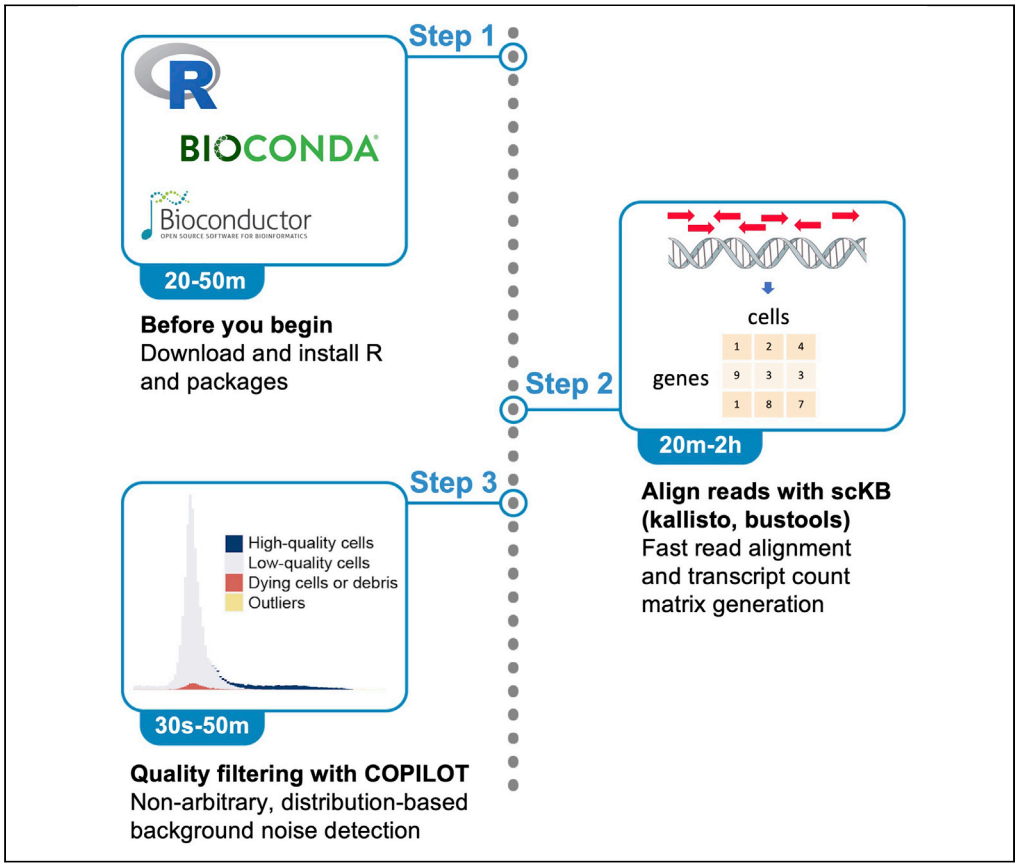
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for fast scRNA-seq raw data processing using scKB and non-arbitrary quality control with COPILOT



We describe a protocol to perform fast and non-arbitrary quality control of single-cell RNA sequencing (scRNA-seq) raw data using scKB and COPILOT. scKB is a wrapper script of kallisto and bustools for accelerated alignment and transcript count matrix generation, which runs significantly faster than the popular tool Cell Ranger. COPILOT then offers non-arbitrary background noise removal by comparing distributions of low-quality and high-quality cells. Together, this protocol streamlines the processing workflow and provides an easy entry for new scRNA-seq users.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Che-Wei Hsu, Rachel Shahan, Trevor M. Nolan, Philip N. Benfey, Uwe Ohler

philip.benfey@duke.edu (P.N.B.)
uwe.ohler@mdc-berlin.de (U.O.)

Highlights

Streamlined processing pipeline for raw scRNA-seq data

Fast alignment of reads enabled by scKB (kallisto and bustools)

Non-arbitrary and distribution-based quality control by COPILOT

Easy entry for novice scRNA-seq users

Hsu et al., STAR Protocols 3, 101729
December 16, 2022
https://doi.org/10.1016/j.xpro.2022.101729

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for fast scRNA-seq raw data processing using scKB and non-arbitrary quality control with COPILOT

Che-Wei Hsu,[1,3,6] Rachel Shahan,[2] Trevor M. Nolan,[2] Philip N. Benfey,[2,5,*] and Uwe Ohler[1,3,4,7,*]

[1]Department of Biology, Humboldt Universität zu Berlin, 10117 Berlin, Germany

[2]Department of Biology, Duke University, Durham, NC 27708, USA

[3]The Berlin Institute for Medical Systems Biology, Max Delbrück Center for Molecular Medicine, 10115 Berlin, Germany

[4]Department of Computer Science, Humboldt Universität zu Berlin, 10117 Berlin, Germany

[5]Howard Hughes Medical Institute, Duke University, Durham, NC 27708, USA

[6]Technical contact: che-wei.hsu@mdc-berlin.de

[7]Lead contact

*Correspondence: philip.benfey@duke.edu (P.N.B.), uwe.ohler@mdc-berlin.de (U.O.)
https://doi.org/10.1016/j.xpro.2022.101729

SUMMARY

We describe a protocol to perform fast and non-arbitrary quality control of single-cell RNA sequencing (scRNA-seq) raw data using scKB and COPILOT. scKB is a wrapper script of kallisto and bustools for accelerated alignment and transcript count matrix generation, which runs significantly faster than the popular tool Cell Ranger. COPILOT then offers non-arbitrary background noise removal by comparing distributions of low-quality and high-quality cells. Together, this protocol streamlines the processing workflow and provides an easy entry for new scRNA-seq users.

For complete details on the use and execution of this protocol, please refer to Shahan et al. (2022).

BEFORE YOU BEGIN

This protocol is designed to process scRNA-seq data from any species, tissue, or system. Please consult the troubleshooting section for potential solutions to common issues.

Processing of scRNA-seq data begins with read alignment against a genome to call a gene-by-cell matrix of transcript counts. Since this step is time-consuming, implementing a fast alignment algorithm largely reduces the running time and saves computational resources. Quality control/filtering, which is performed after calling a gene-by-cell matrix, refers to the removal of low-quality cells in scRNA-seq data by detecting signals characteristic of unhealthy, damaged, dying cells or debris. The typical quality filtering simply removes cells that do not meet an arbitrary UMI (Unique Molecular Identifier) count threshold, assuming a clear-cut separation of background noise from high-quality cells. The background noise could be cells that are unhealthy, dying or damaged, and empty droplets containing ambient RNA or cell debris. Due to the multiple possible sources of background noise, the assumption on which an arbitrary UMI count threshold is based can be easily violated. To address these issues, we describe a protocol to perform fast and non-arbitrary quality control on scRNA-seq data using scKB, a bash script, and COPILOT (Cell preprOcessing PIpeline kaLlistO busTools), an R package that we developed. scKB is a COPILOT-compatible wrapper that takes raw reads in and produces gene-by-cell matrices. scKB incorporates two published tools, kallisto (Bray et al., 2016) and bustools (Melsted et al., 2021). Kallisto provides accelerated alignment of raw reads while bustools offers barcode error correction, UMI collapsing, and count matrix generation. Given kallisto's speedy pseudoalignment algorithm, scKB's run time is significantly faster than

the popular commercial tool 10x Genomics Cell Ranger 6.1.2. Following scKB, COPILOT offers non-arbitrary (distribution-based) and flexible schemes for detecting and removing background noise. Together, this protocol streamlines multiple independent steps from the processing workflow and provides an easy entry for new scRNA-seq users. Improving the ease, efficiency, and accessibility of the scRNA-seq data processing pipeline will ultimately facilitate biological discovery.

There are three main steps in this protocol: 1) read alignment with scKB; 2) quality filtering cells with COPILOT; and 3) optional downstream analysis with Seurat. Read alignment is performed by the COPILOT-compatible script scKB, which creates gene-by-cell matrices as output. COPILOT takes the raw gene-by-cell matrix as input and filters out low-quality cells and outliers. After quality filtering, COPILOT generates an html summary documenting parameters used for filtering, cell statistics, and sequencing statistics of the sample similar to the summary produced by Cell Ranger from 10x Genomics. Finally, users can choose to output the filtered gene-by-cell matrices in 10x format, the same as produced by the Cell Ranger software suite, or store the filtered matrix as a Seurat object (Butler et al., 2018; Stuart et al., 2019) with COPILOT. When stored as a Seurat object, typical downstream analyses including read count normalization, dimensionality reduction, and doublet removal can automatically be performed with Seurat functions. In addition, if a user applies this protocol to process scRNA-seq data from *Arabidopsis thaliana* roots, COPILOT supports annotation of cell types and developmental stages based on (Shahan et al., 2022). Alternatively, users can apply label transfer (Stuart et al., 2019) to the COPILOT-generated Seurat object from an annotated reference profile (Shahan et al., 2022). We describe the details of each step in the following sections:

1. Align reads with scKB.

A bash script is a plain text file containing a series of commands executed in the terminal of a Unix system. The scKB bash script contains commands from kallisto (Bray et al., 2016) and bustools (Melsted et al., 2021), which perform fast read alignment to a designated genome to produce COPILOT-compatible gene-by-cell matrices. There are two gene-by-cell matrices as output for spliced and unspliced transcripts, respectively. Sequences mapped exclusively to exons of genes are considered spliced, while those mapped to introns are categorized as unspliced. This feature of scKB allows users to calculate RNA velocity in downstream analysis, which leverages the ratio of spliced to unspliced transcripts (Bergen et al., 2021). In addition, scKB summarizes the properties of single-cell RNA-seq samples, such as the number of UMIs per cell barcode, using bustools. The kallisto tool supports read alignment from various single-cell technologies. Here we demonstrate its application to *Arabidopsis* root data generated with 10× Chromium Single Cell 3' GEM Kit v3 technology. Details for available technologies are described in the "quantification and statistical analysis" section.

2. Filter low-quality cells with COPILOT.

The typical quality filtering of scRNA-seq data simply removes cells that do not meet an arbitrary UMI threshold or mitochondrial expression threshold, as demonstrated in the Seurat tutorial (https://satijalab.org/seurat/articles/pbmc3k_tutorial.html). This approach assumes that there is a clear cutoff in terms of UMI count and mitochondrial expression separating low-quality cells from the high-quality cells. Given that published scRNA-seq data varies in quality due to differences between technologies, batches, and independent studies, quality filtering based on arbitrary thresholds can result in either over-filtering or under-filtering individual samples. Over-filtering occurs when valid cells or cell types with relatively low-UMI or high-mitochondrial expression are removed. Conversely, a sample is under-filtered when substantial numbers of low-quality cells are retained in the dataset.

a. Understand the COPILOT algorithm.

The COPILOT quality filtering algorithm is designed to avoid applying an arbitrary threshold. COPILOT identifies the distribution of cell populations showing signals of unhealthy, damaged,

dying cells or debris in a semi-supervised fashion. Steps in the COPILOT algorithm are as follows:

i. The scKB-generated gene-by-cell matrices of spliced and unspliced transcripts are combined into a single gene-by-cell matrix in COPILOT. After removing genes without any UMI counts, the algorithm first identifies a group of cells enriched in user-provided genes characteristic of unhealthy, damaged, dying cells or debris. For example, users can list genes known to be upregulated by apoptosis or cell damage. The most common genes applied are mitochondrial genes, which are excessively expressed during apoptosis (Márquez-Jurado et al., 2018). We call the cells defined by user-provided signals as 'prior cells' given that it is based on the domain knowledge.

ii. The prior cells' UMI count distribution mode is treated as the initial boundary to separate cells into two groups representing low and high-quality cells.

iii. Expression profile references are built for both low and high-quality cells by taking the average of log-normalized counts.

iv. Subsequently, the whole distribution of low-quality cells is recovered by comparing the Pearson correlation coefficient of each high-quality cell to the two references. If cells in the high-quality group have a higher correlation to the low-quality cell profile than the high-quality one, then those cells would be re-labeled as low-quality.

v. COPILOT offers functionality that allows iterative filtering until there are no more cells more similar to the low-quality cell expression profile than the high-quality one. However, for samples where the count distributions of high-quality cells and low-quality cells are not clearly separated, iterative filtering would result in over-filtering. Therefore, to avoid potential over-filtering when processing such samples, the default filtering iteration of COPILOT is set to one.

vi. To generate quality-filtered gene-by-cell matrices, the low-quality cells and prior cells are removed along with the top 1% (by default) of high-quality cells in terms of total UMI counts to address issues associated with outliers. These outliers represent potential multiplets, which refer to at least two different cells sharing the same cell barcode during single-cell library preparation. Since multiplets contain at least two cells (doublets), they have higher UMI counts than singlets on average. This step aims to remove multiplets of extreme cases.

vii. Finally, users have the option to further remove putative doublets that do not have extremely high UMI counts. COPILOT incorporates DoubletFinder (McGinnis et al., 2019) and Seurat to perform doublet removal with default parameters according to the estimated doublet rate (10x Genomics Chromium Single Cell 3′ Reagent Kit User Guide with v3 Chemistry). There is also an option for users to provide an estimated doublet rate for different technologies.

viii. The quality-filtered gene-by-cell matrices of spliced and unspliced transcripts are stored in 10x format. If optional doublet removal and downstream analysis with Seurat is performed, a Seurat object consisting of spliced, unspliced, and combined gene-by-cell matrices will be generated and stored as a .rds file.

b. Summary of the COPILOT algorithm.

The rationale of the COPILOT quality filtering scheme lies in its algorithm. It starts with user-provided information (prior) and then identifies where the low-quality cell population is through iterative search, instead of relying on an arbitrary threshold. In addition, COPILOT offers a comprehensive suite of arguments for generalizability and applicability, allowing users to fine-tune their filtering scheme. Detailed explanations of arguments and their applications are described in the "quantification and statistical analysis" section.

3. Perform optional downstream analysis with Seurat.

This step includes read count normalization, dimensionality reduction, doublet removal, and cell annotation. Normalization is necessary to remove the artificial bias of sequencing depth among cells and retain only the biological heterogeneity. Here, we choose to use SCTransform (Hafemeister and Satija, 2019) incorporated in Seurat to perform normalization.

a. Perform dimensionality reduction and visualization.

Dimensionality reduction is a technique for making high-dimensional data computationally manageable and interpretable. It summarizes the transcriptomic similarity among cells and allows the relationship of cells to be visualized in a human-perceivable way. Before dimensionality reduction, users have options to remove specific genes with the argument "unwanted.genes" or choose only highly-variable genes with arguments "HVG" and "HVGN". With Seurat functions, COPILOT performs PCA (Principal Component Analysis) and UMAP (Uniform Manifold Approximation and Projection) (McInnes et al., 2018) for visualization. If users choose to remove putative doublets with the argument "remove.doublet", one can either provide the doublet rate if known or let COPILOT estimate it according to the 10x Genomics Chromium Single Cell 3′ Reagent Kit User Guide (v3 Chemistry).

b. Perform clustering and annotation.

Finally, cell annotation is performed by clustering using a chosen algorithm and resolution with the arguments "clustering_alg" and "res". For *Arabidopsis thaliana* root data, additional cell type and developmental stage annotation is supported with arguments "do.annotation", "dir_to_bulk" and "dir_to_color_scheme". Annotations are based on established reference profiles, which includes bulk RNA-seq profiles (Li et al., 2016) and microarray data (Brady et al., 2007). The annotation method applied is the correlation-based method described in Shahan et al. (2022). Alternatively, users can perform label transfer from an established reference profile as introduced in Shahan et al. (2022) for cell annotation. Label transfer is our recommended approach for cell annotation, however, it does require more computational resources (128 GB memory and 200 GB storage space for this protocol) given that the annotated reference profiles are usually bulky in size. The legends of arguments are listed in the "quantification and statistical analysis" section.

Note: For first-time users to quickly test this protocol with minimal computation resource requirements, we demonstrate the utility using an 80 MB toy data subset from a wild-type *Arabidopsis thaliana* root scRNA-seq dataset (GEO accession GSE152766, sample GSM4626009). This dataset was generated using the 10x Genomics Chromium Single Cell 3' GEM Kit v3 technology. The toy data can be found under directory "toy_data" in the GitHub repository for scKB. For demonstration of scalability and generality, we run the scKB-COPILOT pipeline on a full wild-type *Arabidopsis thaliana* root scRNA-seq dataset (col0) and a PBMC dataset published by 10x Genomics (https://www.10xgenomics.com/resources/datasets/1-k-pbm-cs-from-a-healthy-donor-v-2-chemistry-3-standard-3-0-0). Finally, we benchmark the run time of the scKB-COPILOT pipeline against 10x Genomics Cell Ranger 6.1.2 on all three datasets. For the tool/package versions used to verify this protocol, please refer to the "key resources table".

Install scKB and COPILOT on your machine

Timing: 20–50 min

This protocol is verified on Linux machines (Ubuntu 22.04 and CentOS 7) with 2 TB storage space, 256 GB memory, and a 32-core CPU.

⚠ CRITICAL: To run the whole protocol with toy col0, full col0 and PBMC data, we recommend users use a machine with 128 GB memory, 500 GB of storage and a CPU of 16-cores. At least 2 GB memory and 10 GB storage is required for toy col0 data. At least 16 GB memory and 250 GB storage is required for full col0 data without label transfer. 128 GB memory and 250 GB storage is recommended for both full col0 data with label transfer and PBMC data. The Linux operating system should have four command line tools installed, which are "git", "unzip", "curl" and "rename". Depending on Linux distribution

types, users can use "sudo apt install" (Ubuntu, Linux Mint), "sudo yum install" (Fedora, CentOS) or "sudo pacman -S" (Manjaro, Arco Linux) to install the command line tools.

Note: To skip the installation procedure, users can begin with steps 7b or 7c for instructions on Singularity (Kurtzer et al., 2017, 2021) or Docker (Merkel, 2014) usage. Users can pull the scKB-COPILOT container image from Cloud Library or Docker Hub and build a computational environment capable of running this protocol. A container image is a recipe for creating a container of a specific computational environment. Therefore, a container built from the scKB-COPILOT image would include every package and tool needed to run this protocol. The Singularity and Docker engines can be installed following instructions from their github or website.

For users who are new to bioinformatics and value reproducibility of this protocol, containers are always the best choice. When using shared computational resources on a high-performance computing (HPC) cluster, users usually do not have root privileges. Therefore, we recommend using the singularity container for this protocol as there is no requirement of root privileges to run.

4. Download scKB.
   a. The bash script scKB can be downloaded from github or cloned using the "git clone" command. Git clone is a command for downloading existing source code from a remote repository such as GitHub.

```
>git clone https://github.com/ohlerlab/scKB.git
```

   b. Extract cell white lists and annotation files. The white list includes the expected cell barcodes of a given technology. For *Arabidopsis thaliana* data, the gene annotation file used here is TAIR10, while the genome file used in the following sections is TAIR9. Notice that the number implies versions of gene annotation files, not the genome. That is, TAIR9 and TAIR10 share the same genome file but different gene annotation files. Finally, scKB script should be made executable before use.

```
>cd ./scKB

>unzip 10xv2_whitelist.txt.zip

>unzip 10xv3_whitelist.txt.zip

>gunzip Arabidopsis_thaliana.TAIR10.43.gtf.gz

>chmod 777 scKB
```

5. Install kallisto (Bray et al., 2016) and bustools (Melsted et al., 2021) via conda.

Note: Conda is a package and environment manager that enables easy installation of many bioinformatic tools. The tool kallisto is developed for speedy read alignment against a genome. bustools offers functionalities to convert the output from kallisto to gene-by-cell matrices of transcript counts.

   a. Install conda. Download the installer script following instructions from https://bioconda.github.io/user/install.html.
   b. Install R, kallisto and bustools via conda. The users can specify package versions to install with conda.

```
>conda create -n scKB -c conda-forge -c bioconda r-base=4.1.3 kallisto=0.48.0
bustools=0.41.0
```

c. Alternatively, install R, kallisto, and bustools via conda using the yml file, which specifies the tools and versions to be installed.

```
>conda env create -f scKB.yml
```

6. Install BSgenome (Pagès, 2017) and BUSpaRse (Moses, n.d.) in R.

Note: It is required for users to convert their genome file (.fasta) into a BSgenome object (see problem 1 in the "troubleshooting" section). If the genome you want to use is already available as a BSgenome object, simply install it from Bioconductor.

Note: Make sure that the name of chromosomes/sequence is the same in the gene annotation file (.gtf) and the BSgenome object. BUSpaRse is required to extract kallisto-compatible intron files using the BSgenome object and user-provided gene annotation file (.gtf) as inputs.

a. Activate scKB environment and start R.

```
>conda activate scKB

>R
```

b. Install BUSpaRse, BSgenome via BiocManager in the R environment. BiocManager allows users to install and manage packages from the Bioconductor project (Gentleman et al., 2004), which disseminates tools from current and emerging biological assays. Install desired genome file via BiocManager if already available as a BSgenome object.

```
# Install BiocManager

>install.packages("BiocManager")

# Install BUSpaRse and BSgenome

>BiocManager::install("BUSpaRse")

# Install desired genome if available as BSgenome object

>BiocManager::install("BSgenome.Athaliana.TAIR.TAIR9")

# To bash

>quit()
```

7. Install COPILOT via devtools in R, which allows users to install R packages hosted on a GitHub repository.
   a. Install COPILOT and its dependencies in R environment.

```
# In bash

# Install umap-learn for Seurat to run umap under COPILOT

>conda install -c conda-forge umap-learn

# Activate R and install dependencies

>R

>install.packages(c("devtools","Seurat'','rjson","R2HTML"))

>BiocManager::install("DropletUtils")
```

```
>devtools::install_github('chris-mcginnis-ucsf/DoubletFinder')

# Install COPILOT

>devtools::install_github('ohlerlab/COPILOT')
```

b. Alternatively, apply a singularity container. The container includes most software and tools listed above (From steps 5 to 7a). The input data and output results are better stored "outside" of the container, as they would be deleted once the user exits the container after use. For this, a directory in the file system should be mounted to the container, which serves as storage for data and output results. These procedures take about 3 min.

```
# In bash, make sure singularity is installed on your machine

# Pull container from Cloud Library

>singularity pull library://cheweihsu/collection/sckb-copilot.sif:lastest

# Run the container and mount (with –bind) the directory where you store data (left to the colon) to a directory in the container (right to the colon)

>singularity shell –bind /where/you/store/data/on/filesystem/:/data/ sckb-copilot.sif_latest.sif

# Source conda bash script

>source /usr/local/etc/profile.d/conda.sh

# Activate conda environment "scKB", move to writable mounted directory and clone the scKB repo following step 4

>conda activate scKB

>cd /data/

Steps 4a and b

# Exit the container when finished running this protocol

>exit
```

c. For users with root privileges, a docker container is also an option. Although a docker container includes all software and tools listed above (From steps 4 to 7a), to avoid storing data in the container, we strongly recommend users to re-clone the scKB repository following step 4 to the mounted directory "/data/". Notice that docker should be ran with the "sudo" command if root privilege is required. These procedures take about 3 min.

```
# In bash, make sure docker is installed on your machine

# Pull image from docker hub

>sudo docker pull cheweihsu/sckb-copilot:latest

# Run the image with 128 GB memory applied and mount the directory where you store the data (source) to a directory in the container (target)

>sudo docker run -it –memory 128G –mount type=bind,source=/where/you/store/data/on/filesystem/,target=/data/ -d cheweihsu/sckb-copilot:latest

# Check running container ID

>sudo docker ps

# Enter the container with the corresponding container ID

>sudo docker exec -it {container ID} bash
```

```
# In the container, activate conda environment "scKB" and move to the scKB repo or to the mounted
/data/ directory and repeat step 4

>conda activate scKB

>cd /scKB/

# Or

>cd /data/

Steps 4a and b

# Exit the container when finished running this protocol

>exit

# (Optional) To stop and remove a container

>sudo docker stop {container ID}

>sudo docker rm {container ID}

# (Optional) Copy file to and from a container

>sudo docker cp foo.txt {container ID}:/foo.txt

>sudo docker cp {container ID}:/foo.txt foo.txt
```

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Single-Cell mRNA Sequencing data	(Shahan et al., 2022); 10x Genomics	*Arabidopsis thaliana* root: GEO accession GSE152766, sample GSM4626009;PBMC dataset: https://www.10xgenomics.com/resources/datasets/1-k-pbm-cs-from-a-healthy-donor-v-2-chemistry-3-standard-3-0-0
COPILOT and CellRanger v6.1.2 summary files	(Hsu, 2022a)	https://data.mendeley.com/datasets/sm6tnbthxy/1
Software and algorithms		
singularity (v3.8.7)	(Kurtzer et al., 2017, 2021)	https://github.com/apptainer/apptainer
docker (v20.10.14)	(Merkel, 2014)	https://www.docker.com/
scKB	(Shahan et al., 2022) (Hsu, 2022b)	https://github.com/ohlerlab/scKB
conda (v4.12.0)	(Anon, 2020)	https://anaconda.org/anaconda/conda
kallisto (v0.48.0)	(Bray et al., 2016)	https://pachterlab.github.io/kallisto/
bustools (v0.41.0)	(Melsted et al., 2021)	https://github.com/BUStools/bustools
R (v4.1.3)	(R Core Team, 2020)	https://www.r-project.org/
BiocManager (v3.14)	(Gentleman et al., 2004)	https://www.bioconductor.org/install/
devtools (v2.4.3)	(Wickham et al., 2021)	https://github.com/r-lib/devtools
BUSpaRse (v1.8.0)	(Moses, n.d.)	https://bioconductor.org/packages/release/bioc/html/BUSpaRse.html
BSgenome (v1.62.0)	(Pagès, 2017)	https://bioconductor.org/packages/release/bioc/html/BSgenome.html
COPILOT	(Shahan et al., 2022) (Hsu, 2021)	https://github.com/ohlerlab/COPILOT
umap-learn (v0.5.2)	(McInnes et al., 2018)	https://umap-learn.readthedocs.io/en/latest/
rjson (v0.2.21)	(Couture-Beil, 2022)	https://cran.r-project.org/web/packages/rjson/index.html
R2HTML (v2.3.2)	(Lecoutre et al., 2016)	https://cran.r-project.org/web/packages/R2HTML/index.html
DropletUtils (v1.14.2)	(Aaron Lun, 2018)	https://bioconductor.org/packages/release/bioc/html/DropletUtils.html
DoubletFinder (v2.0.3)	(McGinnis et al., 2019)	https://github.com/chris-mcginnis-ucsf/DoubletFinder
Seurat (v4.1.0)	(Butler et al., 2018; Stuart et al., 2019)	https://satijalab.org/seurat/

STEP-BY-STEP METHOD DETAILS

For first-time users, we recommend first running the protocol on toy data following steps 1, 2, 5, and 6. For those who wish to learn how to transform typical raw scRNA-seq data into a normalized and annotated gene-by-cell matrix, please follow steps 1, 3, and 7. For users who would like to try this protocol on scRNA-seq data from a species other than *Arabidopsis thaliana*, (e.g., human PBMC data), please see steps 4 and 8.

Align raw reads to genome via scKB

Timing: 2 min for toy data. For the full datasets demonstrated in this protocol, the running time ranged between 25 min to 120 min, depending on data size and species

This step aligns raw reads to the designated genome and calls transcript UMI counts for each cell barcode and gene. The expected output of this step includes gene-by-cell matrices of spliced and unspliced UMI counts of transcripts, reads/aligning stats of the sample, cell barcodes, and gene ID files. These outputs are COPILOT-compatible and can serve as direct inputs to COPILOT for quality filtering and generation of the summary file.

1. Extract intron information with the R package BUSpaRse and make kallisto index. This step is important for kallisto to distinguish spliced and unspliced reads when mapping against the designated genome.

   Note: Extraction of intron information is done by running BUSpaRse::get_velocity_files(). Notice that argument "X" should point to the directory of your annotation file. Argument "L" specifies the length of reads, which varies with single-cell technologies. For example, 10x Genomics v1: 98 nucleotides, 10x v2: 98 nucleotides, 10x v3:91 nucleotides, Drop-seq: 50 nucleotides. In this example, we set "L" to 91 since our toy data was generated with 10x v3 chemistry. We should set the argument "style" to "Ensembl" since our gene annotation (.gtf) file is downloaded from Ensembl (https://plants.ensembl.org/index.html). For details related to get_velocity_files() arguments, please refer to the BUSpaRse manual (https://bioconductor.org/packages/release/bioc/manuals/BUSpaRse/man/BUSpaRse.pdf).

   ⚠ CRITICAL: After running get_velocity_files(), the working directory should have files "cDNA_introns.fa", "cDNA_tx_to_capture.txt", "introns_tx_to_capture.txt", and "tr2g.tsv". Preparing intron files for the *Arabidopsis thaliana* genome takes about 1 min.

```
# In R

>setwd("~/to/where/you/clone/the/repo/scKB")

# Load libraries and genome

>library(BUSpaRse)

>library(BSgenome.Athaliana.TAIR.TAIR9)

# Extract intron information

>get_velocity_files(X = "./Arabidopsis_thaliana.TAIR10.43.gtf", L = 91, Genome =
BSgenome.Athaliana.TAIR.TAIR9, out_path = "./", isoform_action = "separate", chrs_only=-
FALSE, style="Ensembl")

# Index the intron file with kallisto and exit R

>system("kallisto index -i ./cDNA_introns_10xv3.idx ./cDNA_introns.fa")

>quit()
```

2. Run scKB.

Note: Name the output directory as your sample name and set the number of computational threads/cores available accordingly with argument "t".

Note: After running scKB, the output directory (in this case "./col0_toy") should have 8 files: a "inspect.json", a "run_info.json", a "spliced.barcodes.txt", a "spliced.genes.txt", a "spliced.mtx", a"unspliced.barcodes.txt", a "unspliced.genes.txt" and a "unspliced.mtx".

Note: The json files document the sample stats of reads and alignment; the mtx files represent gene-by-cell matrices; the txt files contain cell barcodes and gene IDs information.

Note: The toy data contain 1 million reads and the run takes about 30 s.

```
# In bash

>cd ~/to/where/you/clone/the/repo/scKB

# Run bash script scKB. Please name the output directory as your sample name using "-n" flag

>./scKB -f ./toy_data -i ./cDNA_introns_10xv3.idx -d ./ -s 10xv3 -t 16 -w ./10xv3_white-
list.txt -n ./col0_toy
```

3. Run scKB on *Arabidopsis thaliana* wild-type Columbia-0 ("col0") full data.

Note: The col0 full data contains 386 million reads and downloading can take 2–3 h.

Note: Here, we store the fastq files for the full dataset under the folder named "col0_data".

Note: Notice that if containers were applied, it is recommended to set the output directories of sra files and the fastqs files under the mounted directory "/data/" to avoid potential data loss (see steps 7b and c in the section "install scKB and COPILOT on your machine").

Note: The scKB run takes 22 min.

```
# In bash

>cd ~/to/where/you/clone/the/repo/scKB

# Install sra-tools and download sra from sample GSM4626009 (col0 full data)

>conda install -c bioconda sra-tools

>prefetch -v SRR12046119 SRR12046120

# Convert sra to fastqs (sras are downloaded to /home/[USER]/ncbi/public/sra/ by default)

>fastq-dump -outdir ./col0_data/ -split-files -gzip /home/[USER]/ncbi/public/sra/
SRR12046119.sra

>fastq-dump -outdir ./col0_data/ -split-files -gzip /home/[USER]/ncbi/public/sra/
SRR12046120.sra

#Rename fastq files for scKB compatibility

>rename 's/_1.fastq.gz/_R1_001.fastq.gz/' ./col0_data/*.fastq.gz

>rename 's/_2.fastq.gz/_R2_001.fastq.gz/' ./col0_data/*.fastq.gz
```

```
>rename 's/_3.fastq.gz/_I1_001.fastq.gz/' ./col0_data/*.fastq.gz

# Run scKB on the full data col0 from which the toy_data is subsetted

>./scKB -f ./col0_data -i ./cDNA_introns_10xv3.idx -d ./ -s 10xv3 -t 16 -w ./10xv3_white-
list.txt -n ./col0
```

4. Run scKB on PBMC dataset generated with 10x v2 technology.

   Note: The codes cover data download, intron file extraction, kallisto index building, and scKB execution.

   Note: Notice that if containers were applied, the data downloaded and the output directory are recommended to be put under the mounted directory "/data/" to avoid potential data loss (see steps 7b and c in the section "install scKB and COPILOT on your machine").

   Note: The whole process takes about 120 min (preparing genome and intron files take 85 min, scKB takes 45 min).

```
# In bash

>cd ~/to/where/you/clone/the/repo/scKB

# Download PBMC dataset from 10x Genomics website

>mkdir pbmc

>cd pbmc

>curl -O https://cf.10xgenomics.com/samples/cell-exp/3.0.0/pbmc_1k_v2/pbmc_1k_v2_fastqs.tar

>tar xvf pbmc_1k_v2_fastqs.tar

>cd ../

# Download gene annotation file from 10x Genomics website

>curl -O https://cf.10xgenomics.com/supp/cell-exp/refdata-gex-GRCh38-2020-A.tar.gz

>tar zxvf refdata-gex-GRCh38-2020-A.tar.gz

# Activate R and set working directory in R

>R

>setwd("~/to/where/you/clone/the/repo/scKB")

# Install Human BSgenome object

>BiocManager::install("BSgenome.Hsapiens.UCSC.hg38")

# Load libraries and genome

>library(BUSpaRse)

>library(BSgenome.Hsapiens.UCSC.hg38)

# Extract intron information

>get_velocity_files(X = "./refdata-gex-GRCh38-2020-A/genes/genes.gtf", L = 98, Genome =
BSgenome.Hsapiens.UCSC.hg38, out_path = "./", isoform_action = "separate", chrs_only=TRUE,
style="Ensembl")

# Index the intron file with kallisto
```

```
>system("kallisto index -i ./cDNA_introns_10xv2.idx ./cDNA_introns.fa")

>quit()

# In bash

# Run scKB

>./scKB -f ./pbmc/pbmc_1k_v2_fastqs -i ./cDNA_introns_10xv2.idx -d ./ -s 10xv2 -t 16 -w
./10xv2_whitelist.txt -n ./pbmc_1k_v2
```

Run COPILOT for quality filtering

Timing: between 30 s to 50 min, depending on data size and whether downstream analysis with Seurat is performed

This step includes quality filtering on cells, optional doublet removal, and cell type/developmental stage annotation (optional). Notice that if containers were used, the data downloaded and the output directory are recommended to be put under the mounted directory "/data/" to avoid potential data loss (see steps 7b and c under the section "install scKB and COPILOT on your machine").

5. Run COPILOT for non-iterative quality filtering without doing downstream analysis with Seurat.

Note: The toy data contain only 1 million reads. Therefore, the argument "min.UMI.low.quality" and "min.UMI.high.quality" are adjusted accordingly for expected low sequencing depth.

Note: The non-iterative filtering is achieved by setting the argument "filtering.ratio" to 1.

Note: The run takes about 30 s.

```
# In R

>setwd("~/to/where/you/clone/the/repo/scKB")

# Load COPILOT

>library(COPILOT)

# Run COPILOT

>copilot(sample.name = "col0_toy", species.name = "Arabidopsis thaliana", transcriptome.-
name = "TAIR10", sample.stats = NULL, mt.pattern = "ATMG",

     mt.threshold = 5, cp.pattern = "ATCG", remove.doublet = FALSE, do.seurat = FALSE, do.an-
notation = FALSE, unwanted.genes = NULL, filtering.ratio = 1, min.UMI.low.quality = 1, min.U-
MI.high.quality = 3)
```

6. Run COPILOT for iterative quality filtering without downstream analysis.

Note: The iterative filtering is achieved by setting the argument "filtering.ratio" to 0, which will run until there are no cells more similar to the low-quality cell expression profile than the high-quality cell expression profile.

Note: The run takes about 2 min.

```
# Run COPILOT

>copilot(sample.name = "col0_toy", species.name = "Arabidopsis thaliana", transcriptome.name = "TAIR10", sample.stats = NULL, mt.pattern = "ATMG",

    mt.threshold = 5, cp.pattern = "ATCG", remove.doublet = FALSE, do.seurat = FALSE, do.annotation = FALSE, unwanted.genes = NULL, filtering.ratio = 0, min.UMI.low.quality = 1, min.UMI.high.quality = 3)
```

7. Run COPILOT for non-iterative quality filtering with downstream analysis (do.seurat = TRUE), including doublet removal (remove.doublet = TRUE) and annotation of cell types and developmental stages based on bulk-RNAseq and microarray data (do.annotation = TRUE).

   Note: If users wish to annotate cells based on an established reference profile as described in Shahan et al. (2022), example code for label transfer is also provided.

   Note: Notice that the code for label transfer is suitable only for reference objects created by Seurat version 4. Label transfer demonstrated here requires users to have sufficient amount of space storage and memory (128 GB) to host and load the reference object.

   Note: Here we also demonstrate how to remove genes from downstream analysis with the argument "unwanted.genes". In this case, the protoplasting-induced genes are removed from the *Arabidopsis* data. Protoplasting refers to a process of removing cell walls from plant cells, which alters the expression of a subset of genes (Denyer et al., 2019). Therefore, to correct for artificial factors in downstream analysis, genes known to be induced by protoplasting during library preparation are removed. The run takes about 50 min without label transfer and 60 min with label transfer.

```
# In R

>setwd("~/to/where/you/clone/the/repo/scKB")

# Load COPILOT

>library(COPILOT)

# Load unwanted genes (optional)

>pp.genes   <-as.character(read.table("./supp_data/Protoplasting_DEgene_FC2_list.txt", header=F)$V1)

# Run COPILOT

>copilot(sample.name = "col0", species.name = "Arabidopsis thaliana", transcriptome.name = "TAIR10", sample.stats = NULL, mt.pattern = "ATMG",

mt.threshold = 5, cp.pattern = "ATCG", remove.doublet = TRUE, do.seurat = TRUE, do.annotation = TRUE, unwanted.genes = pp.genes, filtering.ratio = 1, dir_to_color_scheme = "./supp_data/color_scheme_at.RData", dir_to_bulk = "./supp_data/Root_bulk_arabidopsis_curated.RD")

# Label transfer from Shahan et al. (2022)

# After downloading supplementary file Root_Atlas_seu4.rds.gz directly from GEO GSE152766, decompress Root_Atlas_seu4.rds.gz in bash

>gunzip Root_Atlas_seu4.rds.gz

# Load Seurat, reference atlas and query data in R

>R

>library(Seurat)
```

```
>rc.integrated <- readRDS("./Root_Atlas_seu4.rds")

>seu <- readRDS("./col0/col0_COPILOT.rds")

# Find anchors and transfer annotation from reference in Shahan et al.,2022

>lt.anchors <- FindTransferAnchors(reference = rc.integrated, query = seu, normalization.-
method = "SCT", npcs = 50, dims = 1:50)

>predictions <- TransferData(anchorset = lt.anchors, refdata = rc.integrated$cellty-
pe.anno, dims = 1:50, weight.reduction = "pcaproject")

>seu <- AddMetaData(seu, metadata = predictions)

>seu@meta.data$celltype.anno <- seu@meta.data$predicted.id

>predictions <- TransferData(anchorset = lt.anchors, refdata = rc.integrated$time.anno,
dims = 1:50, weight.reduction = "pcaproject")

>seu <- AddMetaData(seu, metadata = predictions)

>seu@meta.data$time.anno <- seu@meta.data$predicted.id

# Visualize the transferred labels and save them as plots (optional)

>pdf("col0_transferred_celltype_anno.pdf", height=8, width=8)

>DimPlot(seu, reduction = "umap", group.by = "celltype.anno")

>dev.off()

>pdf("col0_transferred_time_anno.pdf", height=8, width=8)

>DimPlot(seu, reduction = "umap", group.by = "time.anno")

>dev.off()

# Save the query data with transferred labels

>saveRDS(seu, file = "./col0/col0_COPILOT.rds")
```

8. Run COPILOT on the PBMC dataset generated with 10x v2 technology for non-iterative quality filtering without downstream analysis.

   Note: The run takes about 15 s.

```
# In bash

# Extract mitochondrial gene names from genome annotation file

>grep chrM ./refdata-gex-GRCh38-2020-A/genes/genes.gtf | awk '{print $10}' | uniq | sed -e
's/"//g' -e 's/;//g' > mt_genes.txt

# Activate R and set working directory in R

>R

>setwd("~/to/where/you/clone/the/repo/scKB")

# Load COPILOT

>library(COPILOT)

# Run COPILOT

>copilot(sample.name = "pbmc_1k_v2", species.name = "Homo sapiens", transcriptome.name =
"hg38", sample.stats = NULL, mt.pattern = read.table("mt_genes.txt")$V1, mt.threshold = 5,
cp.pattern = NULL, remove.doublet = FALSE, do.seurat = FALSE, do.annotation = FALSE, unwan-
ted.genes = NULL, filtering.ratio = 1, legend.position=c(0.2,0.8))
```

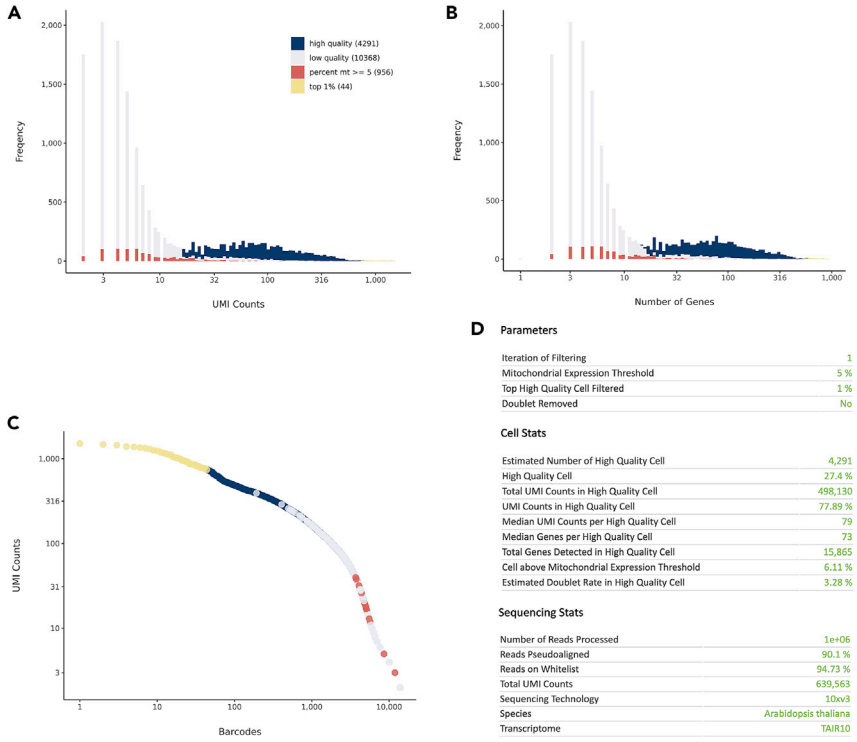


Figure 1. Distribution of cell populations for col0 toy data

(A) UMI count histogram.

(B) Gene number histogram.

(C) Barcode rank plot ranked by UMI count.

(D) Parameters, sample stats of cells and sequencing.

EXPECTED OUTCOMES

After running this protocol, one should expect to have quality-filtered and unfiltered gene-by-cell matrices for both spliced and unspliced transcripts stored as 10x format. The two matrices should have the same dimensions of genes and cells. Other than the matrices, one should also expect a COPILOT summary file in html format documenting parameters used, cell stats, sequencing stats, and histograms showing the distribution of low and high-quality cell populations (Figure 1, Dataset S1 (Hsu, 2022a)). As a rule of thumb, a high quality sample should have the estimated number of high quality cells (in Cell Stats) close to the number of cells users expect to recover. In addition, it should have a high percentage of reads mapped to the genome (Reads Pseudoaligned in the section Sequencing Stats) and high percentage of reads that belong to cells with a valid cell barcode (reads on Whitelist in the section Sequencing Stats).

If the COPILOT argument "do.seurat" is set to TRUE, one should also expect to see feature plots in the "Analysis" panel of the summary file (Hsu, 2022a). In addition, one should expect a quality-filtered Seurat object stored in the rds format under the working directory. If label transfer from the reference profile (Shahan et al., 2022) is performed, the meta data of the Seurat object should contain transferred labels with names "celltype.anno" (cell types) and "time.anno" (developmental stages).

QUANTIFICATION AND STATISTICAL ANALYSIS

1. Benchmark the run time against 10x Genomics Cell Ranger 6.1.2.

Given the same version of the gene annotation and genome files, we documented the runtimes of CellRanger v6.1.2 and this protocol. We ran CellRanger using "cellranger count" command with "nosecondary" flag to skip dimensionality reduction, clustering, and visualization. This procedure is equivalent to running the scKB-COPILOT pipeline for non-iterative quality filtering without downstream analysis. The CellRanger summary files for each data set are available in Dataset S2 (Che-Wei Hsu, 2022). The runtime covers the read alignment, quality-filtering, and output file generation.

scRNA-seq data	Species	Single-cell technology	Number of reads	CellRanger run time	scKB-COPILOT run time
Col0_toy_data	*Arabidopsis thaliana*	10x Genomics v3	1 million	3.5 min	1 min (scKB 30 s, COPILOT 30 s).
Col0_full_data	*Arabidopsis thaliana*	10x Genomics v3	386 million	640 min	25 min (scKB 22 min, COPILOT 3 min)
PBMC	*Homo sapiens*	10x Genomics v2	77 million	185 min	45 min (scKB 45 min, COPILOT 15 s)

2. Argument legends for scKB:

-h (–help): Software usage.

-f (–file): Directory to fastq files.

-i (–index): Directory to intron index files (includes the .idx file itself).

-d (–dir_intermediate): Directory to intermediate files output.

-s (–supported_tech): Single cell technique supported, which is equivalent to argument "x" in kallisto. The argument "x" of kallisto can be adapted to process any existing and new technologies. It accepts a string specifying a technology in the format of "bc:umi:seq" (for cell barcodes, UMI, and the transcript), where each of bc, umi, and seq are a triplet of integers separated by a comma, denoting the file index, start and end position of the sequence. An example demonstrating how to specify a 10x Chromium Single Cell 3' GEM Kit v2 single-cell library is provided in kallisto manual.

-t (–thread): Number of threads used. Maximum 16.

-w (–whitelist): Directory to whitelist file (includes the text file itself).

-n (–dir_final): Directory to final output that includes gene cell matrix.

3. Argument legends for COPILOT:

COPILOT accepts inputs that are not generated by scKB. Users can provide the unfiltered gene-by-cell matrices generated by Cell Ranger or other tools (see arguments "spliced.mtx", "unspliced.mtx", and "total.mtx"). Other than commonly-used mitochondrial gene expression, users have options to use any signature of gene expression to decide the initial boundary separating low and high-quality populations (see arguments "mt.pattern", "mt.threshold", and "cp.pattern"). In cases where the given signature of gene expression fails to determine the initial boundary, users can specify the boundary manually by providing thresholds for low and high-quality cells in terms of UMI counts (see arguments "min.UMI.low.quality" and "min.UMI.high.quality"). For iterative filtering, users can decide whether or not to filter until there are no cells more similar to the low-quality cell expression profile than the high-quality cell expression profile (see argument "filtering.ratio"). Users can specify the percentage using argument "top.percent" to remove potential outliers.

After quality filtering, users can continue with Seurat downstream analysis directly with COPILOT (see arguments "do.seurat", "do.annotation", "unwanted.genes", "HVG", "HVGN", "dir_to_bulk", "dir_to_color_scheme", "clustering_alg" and "res"). Notice that the Seurat analysis is only optional and is not a dependency for non-arbitrary quality filtering. In addition, users can choose whether or not to remove putative doublets and by what rate (see arguments "estimate.doublet.rate", "doublet.rate" and "remove.doublet"). Removing doublets depends on Seurat as it is part of optional downstream analysis. Finally, to edit contents shown in COPILOT summary files, users can adjust the arguments "species.name", "transcriptome.name", "sample.stats", and "legend.position".

sample.name: User defined sample name (character). This should be the same as the name of the directory that contains spliced and unspliced matrices if you use scKB to produce raw UMI count matrices.

spliced.mtx: Gene by cell matrix of spliced counts, which should have column and row names. Default is NULL.

unspliced.mtx: Gene by cell matrix of unspliced counts, which should have column and row names. Default is NULL.

total.mtx: Gene by cell matrix of total counts, which should have column and row names. Default is NULL.

filtered.mtx.output.dir: Output directory for quality filtered matrices. Default is NULL.

species.name: Species name (character). Default is "Not Provided".

transcriptome.name: Name of transcriptome annotation file (e.g., TAIR10 for Arabidopsis). Default is "Not Provided".

sample.stats: Meta data of the sample in R data.frame format. Default is NULL.

mt.pattern: Pattern of gene names/IDs (character; e.g., "ATMG") or list of genes (character vector). Here we demonstrate COPILOT usage with mitochondrial genes. This argument is mandatory to run COPILOT.

mt.threshold: Threshold of expression percentage. Cell would be treated as prior cell if it has expression percentage of genes provided in "mt.pattern" higher than this threshold (numeric). Default is 5.

cp.pattern: Pattern of chloroplast gene names/IDs (character; e.g., "ATCG") or list of chloroplast genes (character vector). Default is NULL.

top.percent: Percentage of cells that contain a high number of UMIs filtered (numeric). Default is 1.

filtering.ratio: Metric that controls the stringency of cell filtering (lenient: 1; strict:0; moderate: 0 < filtering.ratio < 1; numeric). Default is 1.

estimate.doublet.rate: Whether or not to estimate doublet rate according to 10x Genomics' estimation (boolean). Default is TRUE.

doublet.rate: User specified doublet rate (numeric). Default is NULL.

remove.doublet: Whether or not to remove doublets after quality filtering (boolean). Default is TRUE.

do.seurat: Whether or not to perform normalization, PCA, UMAP dimensionality reduction, and clustering using Seurat and output a Seurat object (boolean). Default is TRUE.

do.annotation: Whether or not to perform cell annotation (boolean). COPILOT only supports annotation for Arabidopsis thaliana root scRNA-seq data. Default is FALSE.

unwanted.genes: Gene IDs/names of unwanted genes (character vector, e.g., cell cycle related genes, organelle genes, etc). Default is NULL.

HVG: Whether or not to select highly variable genes (boolean). Default is FALSE.

HVGN: Number of highly variable genes selected (numeric). Defalut is 200.

dir_to_bulk: Directory to reference expression profile for annotation. Default is NULL.

dir_to_color_scheme: Directory to color scheme file for annotation. Default is NULL.

clustering_alg: Algorithm for clustering (1 = original Louvain algorithm; 2 = Louvain algorithm with multilevel refinement; 3 = SLM algorithm; 4 = Leiden algorithm, which requires the python module leidenalg; numeric). Default is 3.

res: Resolution used for clustering (numeric). Default is 0.5.

min.UMI.low.quality: Minimum UMIs for a barcode to be considered a cell (numeric). Default is 100.

min.UMI.high.quality: Minimum UMIs for a cell to be considered a high quality cell (numeric). Default is 300.

legend.position: x y position of the legend on UMI histogram plot (numeric vector of length 2). Default is c(0.8,0.8).

LIMITATIONS

The current version of COPILOT only supports cell type and developmental stage annotation for the root of *Arabidopsis thaliana*.

TROUBLESHOOTING

Problem 1

The required genome is not available as a BSgenome object or users need to convert a customized genome into a BSgenome object. (Related to step 6b under the section 'install scKB and COPILOT on your machine').

Potential solution

Users can convert the genome fasta file into BSgenome object following sections 2.2.5 and 2.3 in the BSgenome manual.

Problem 2

COPILOT dependencies fail to be installed (related to step 7 under the section 'install scKB and COPILOT on your machine').

Potential solution

If dependencies fail to be installed via the R command "install.packages()", BiocManager, or devtools, try using conda to install them. For example, we encountered problems installing Seurat v4.1.0 because one of its dependencies, "shiny", failed to be installed. We solved it by installing shiny via conda before installing Seurat v4.1.0 with the install.packages() function.

Problem 3

scKB fails to run properly (related to the section 'align raw reads to genome via scKB').

Potential solution

- Under the directory that the user provides to the argument "f", make sure to have at least one sequence run (R1 and R2 gzipped fastq). Note that the file names should contain strings "_R1_" and "_R2_" for read 1 and read 2 respectively. Multiple runs of the same sample should be put under the same directory. Each run of scKB will call gene-by-cell matrices for one sample.
- The directory to intron file index file (-i), whitelist (-w), and intermediate files (-d) can be the same, yet the final output directory (-n) should be different.
- Although this protocol only demonstrates the processing of 10x Genomics v2 and v3 chemistry scRNA-seq data, scKB supports various technologies such as CELSeq and DropSeq. One can adjust the argument "s" to meet sequencing conditions of different techniques as documented in the kallisto manual. The argument "x" in kallisto bus is analogous to the argument "s" in scKB.

Problem 4

COPILOT fails to locate the input data. Related to the section 'run COPILOT for quality filtering.'

Potential solution

The name of the output directory for scKB should be the same as the COPILOT argument "sample.name". Alternatively, the user can directly feed gene-by-cell matrices to COPILOT by using arguments "spliced.mtx", "unspliced.mtx", and "total.mtx". Note that if "total.mtx" is provided, then please let "spliced.mtx" and "unspliced.mtx" remain as NULL.

Problem 5

COPILOT fails to start with quality filtering. Related to the section 'run COPILOT for quality filtering.'

Potential solution

- COPILOT relies on detecting a distribution of cells enriched in signals characteristic of unhealthy, damaged, dying cells or debris (prior cells) to identify the initial filtering threshold for low-quality cells. Thus, it is required to provide a list of gene IDs or a character pattern shared among these gene IDs (see argument "mt.pattern").
- If the COPILOT algorithm detects no prior cells, the initial filtering threshold will be set as specified in the argument "min.UMI.high.quality". If the initial boundary determined by the prior cells distribution mode is lower than what the user provides to the argument "min.UMI.high.quality", then it will be replaced by "min.UMI.high.quality".

Problem 6

When running COPILOT with downstream analysis, users encounter an error message (related to step 7 under the section 'run COPILOT for quality filtering'):

```
Error in .External2(C_X11, d$display, d$width, d$height, d$pointsize,:
  unable to start device X11cairo.
```

```
In addition: Warning message:

In (function (display = "", width, height, pointsize, gamma, bg, :

  unable to open connection to X11 display''
```

Potential solution

This error message is caused by the unavailability of x11 device required to generate feature plots for the COPILOT summary file. If the R environment is installed via conda as demonstrated in this protocol, it may not locate x11 device. To solve the issue, we recommend running COPILOT in JupyterLab (https://jupyterlab.readthedocs.io/en/stable/), a web-based user interface for programming. In addition, to let JupyterLab recognize the R kernel, one needs to install the R package "IRkernel" and run the function "installspec()". An R kernel activated via JupyterLab will be able to locate x11 device.

```
# In bash

# Install jupyterlab

>conda install -c conda-forge jupyterlab

# Activate R and install IRkernel

>R

>install.packages("IRkernel")

# Make R kernel recognizable by jupyterlab then exit

>IRkernel::installspec()

>quit()

# In bash

# Start a jupyterlab session, it will open automatically in your browser

>jupyter lab

# For a jupyter lab started on a remote cluster, specify a port number which ranged from 1111 to
9999

>jupyter lab -port=8888

# In terminal of local machine

# Listen to jupyter lab session on remote cluster through the port via ssh command, the user
should specify the user and cluster/node name with -J

>ssh -N -L 127.0.0.1:8888:127.0.0.1:8888 -J [user@]host[:port]
```

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Uwe Ohler (uwe.ohler@mdc-berlin.de).

Materials availability

This study did not generate new unique reagents.

Data and code availability

The toy dataset, codes, and supplementary data needed to reproduce the demonstrated protocol are available on GEO repository GSE152766 and GitHub: https://github.com/ohlerlab/scKB and https://github.com/ohlerlab/COPILOT.

The single-cell RNA-seq data used to generate the toy data have been deposited at GEO with accession number GSE152766 and sample number GSM4626009. The PBMC single-cell RNA-seq data is published by 10x Genomics.

The supplementary summary files of COPILOT and CellRanger v6.1.2 are available at Mendeley Data.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.xpro.2022.101729.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (International Research Training Group 2403) to C.-W.H. and U.O.; US National Institutes of Health (NRSA postdoctoral fellowship 1F32GM136030-01 and MIRA 1R35GM131725) to R.S. and P.N.B., respectively; the US National Science Foundation (Postdoctoral Research Fellowships in Biology Program Grant No. IOS-2010686) to T.M.N.; and by the Howard Hughes Medical Institute to P.N.B. as an investigator.

AUTHOR CONTRIBUTIONS

Conceptualization, C.-W.H. and U.O.; protocol elaboration, C.-W.H.; software development, C.-W.H.; scientific evaluation, R.S., T.M.N., P.B.N., and U.O.; writing, C.-W.H.; review and editing, C.-W.H., R.S., T.M.N., P.N.B., and U.O.

DECLARATION OF INTERESTS

P.N.B. is a member of the *Developmental Cell* advisory board and is the co-founder and chair of the scientific advisory board of Hi Fidelity Genetics, Inc, a company that works on crop root growth.

REFERENCES

Aaron Lun, J.G. (2018). DropletUtils (Bioconductor). https://doi.org/10.18129/B9.BIOC.DROPLETUTILS.

Anon. (2020). Anaconda Software Distribution (Anaconda Inc.). https://docs.anaconda.com/.

Bergen, V., Soldatov, R.A., Kharchenko, P.V., and Theis, F.J. (2021). RNA velocity—current challenges and future perspectives. Mol. Syst. Biol. *17*, e10282. https://doi.org/10.15252/msb.202110282.

Brady, S.M., Orlando, D.A., Lee, J.-Y., Wang, J.Y., Koch, J., Dinneny, J.R., Mace, D., Ohler, U., and Benfey, P.N. (2007). A high-resolution root spatiotemporal map reveals dominant expression patterns. Science *318*, 801–806. https://doi.org/10.1126/science.1146265.

Bray, N.L., Pimentel, H., Melsted, P., and Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. Nat. Biotechnol. *34*, 525–527. https://doi.org/10.1038/nbt.3519.

Butler, A., Hoffman, P., Smibert, P., Papalexi, E., and Satija, R. (2018). Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat. Biotechnol. *36*, 411–420. https://doi.org/10.1038/nbt.4096.

Couture-Beil, A. (2022). Rjson: JSON for R (cran.r-project.org).

Denyer, T., Ma, X., Klesen, S., Scacchi, E., Nieselt, K., and Timmermans, M.C.P. (2019). Spatiotemporal Developmental Trajectories in the Arabidopsis Root Revealed Using High-Throughput Single-Cell RNA Sequencing. Developmental Cell *48*, 840–852.e5. https://doi.org/10.1016/j.devcel.2019.02.022.

Gentleman, R.C., Carey, V.J., Bates, D.M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., et al. (2004). Bioconductor: open software development for computational biology and bioinformatics. Genome Biol. *5*, R80. https://doi.org/10.1186/gb-2004-5-10-r80.

Hafemeister, C., and Satija, R. (2019). Normalization and variance stabilization of single-cell RNA-seq data using regularized negative binomial regression. Genome Biol. *20*, 296. https://doi.org/10.1186/s13059-019-1874-1.

Hsu, C.-W. (2022a). Summary files. https://doi.org/10.17632/SM6TNBTHXY.1.

Hsu, C.-W. (2022b). Hsu-Che-Wei/scKB: scKB (Zenodo). https://doi.org/10.5281/ZENODO.6654379.

Hsu, C.-W. (2021). Hsu-Che-Wei/COPILOT: COPILOT (Cell preprOcessing PIpeline kaListO busTools) (Zenodo). https://doi.org/10.5281/ZENODO.5775927.

Kurtzer, G.M., Cclerget, Bauer, M., Kaneshiro, I., Trudgian, D., and Godlove, D. (2021). Hpcng/Singularity: Singularity 3.7.3 (Zenodo). https://doi.org/10.5281/ZENODO.1310023.

Kurtzer, G.M., Sochat, V., and Bauer, M.W. (2017). Singularity: scientific containers for mobility of compute. PLoS One *12*, e0177459. https://doi.org/10.1371/journal.pone.0177459.

Lecoutre, E., Bouchet-Valat, M., and Friedrichsmeier, T. (2016). R2HTML: HTML Exportation for R Objects (cran.r-project.org).

Li, S., Yamada, M., Han, X., Ohler, U., and Benfey, P.N. (2016). High-resolution expression map of the arabidopsis root reveals alternative splicing and lincRNA regulation. Dev. Cell *39*, 508–522. https://doi.org/10.1016/j.devcel.2016.10.012.

n.d. L. Moses, n.d. BUSpaRse. Bioconductor. https://doi.org/10.18129/B9.BIOC.BUSPARSE

Márquez-Jurado, S., Díaz-Colunga, J., das Neves, R.P., Martinez-Lorente, A., Almazán, F., Guantes, R., and Iborra, F.J. (2018). Mitochondrial levels determine variability in cell death by modulating apoptotic gene expression. Nat. Commun. *9*, 389. https://doi.org/10.1038/s41467-017-02787-4.

McGinnis, C.S., Murrow, L.M., and Gartner, Z.J. (2019). DoubletFinder: doublet detection in single-cell RNA sequencing data using artificial nearest neighbors. Cell Syst. *8*, 329–337.e4. https://doi.org/10.1016/j.cels.2019.03.003.

McInnes, L., Healy, J., Saul, N., and Großberger, L. (2018). UMAP: uniform Manifold approximation

and projection. J. Open Source Softw. *3*, 861. https://doi.org/10.21105/joss.00861.

Melsted, P., Booeshaghi, A.S., Liu, L., Gao, F., Lu, L., Min, K.H.J., da Veiga Beltrame, E., Hjörleifsson, K.E., Gehring, J., and Pachter, L. (2021). Modular, efficient and constant-memory single-cell RNA-seq preprocessing. Nat. Biotechnol. *39*, 813–818. https://doi.org/10.1038/s41587-021-00870-2.

Merkel, D. (2014). Docker: lightweight Linux containers for consistent development and deployment. Linux J. *239*, 2.

Pagès, H. (2017). BSgenome (Bioconductor). https://doi.org/10.18129/B9.BIOC.BSGENOME.

R Core Team (2020). R: A language and environment for statistical computing (Vienna, Austria: R Foundation for Statistical Computing). https://www.R-project.org/.

Shahan, R., Hsu, C.-W., Nolan, T.M., Cole, B.J., Taylor, I.W., Greenstreet, L., Zhang, S., Afanassiev, A., Vlot, A.H.C., Schiebinger, G., et al. (2022). A single-cell Arabidopsis root atlas reveals developmental trajectories in wild-type and cell identity mutants. Dev. Cell *57*, 543–560. https://doi.org/10.1016/j.devcel.2022.01.008.

Stuart, T., Butler, A., Hoffman, P., Hafemeister, C., Papalexi, E., Mauck, W.M., Hao, Y., Stoeckius, M., Smibert, P., and Satija, R. (2019). Comprehensive integration of single-cell data. Cell *177*, 1888–1902.e21. https://doi.org/10.1016/j.cell.2019.05.031.

Wickham, H., Hester, J., Chang, W., Bryan, J., and RStudio. (2021). devtools: Tools to Make Developing R Packages Easier (cran.r-project.org).