TYPE Methods
PUBLISHED 12 October 2023
DOI 10.3389/fimmu.2023.1241283

Using combined single-cell gene expression, TCR sequencing and cell surface protein barcoding to characterize and track CD4+ T cell clones from murine tissues

Annekathrin Silvia Nedwed[1†], Sara Salome Helbich[2,3†], Kathrin Luise Braband[2,3], Michael Volkmar[4], Michael Delacher[2,3*‡] and Federico Marini[1,3*‡]

[1]Institute of Medical Biostatistics, Epidemiology and Informatics (IMBEI), University Medical Center Mainz, Mainz, Germany, [2]Institute of Immunology, University Medical Center Mainz, Mainz, Germany, [3]Research Center for Immunotherapy, University Medical Center Mainz, Mainz, Germany, [4]Helmholtz-Institute for Translational Oncology Mainz (HI-TRON Mainz), Mainz, Germany

Check for updates

OPEN ACCESS

EDITED BY
Matthieu Perreau,
Centre Hospitalier Universitaire Vaudois (CHUV), Switzerland

REVIEWED BY
Ping Zhang,
University of Oxford, United Kingdom
Mark Izraelson,
Institute of Bioorganic Chemistry (RAS), Russia

*CORRESPONDENCE
Michael Delacher
delacher@uni-mainz.de
Federico Marini
marinif@uni-mainz.de

†These authors have contributed equally to this work and share first authorship

‡These authors have contributed equally to this work and share last authorship

RECEIVED 16 June 2023
ACCEPTED 31 August 2023
PUBLISHED 12 October 2023

CITATION
Nedwed AS, Helbich SS, Braband KL, Volkmar M, Delacher M and Marini F (2023) Using combined single-cell gene expression, TCR sequencing and cell surface protein barcoding to characterize and track CD4+ T cell clones from murine tissues.
Front. Immunol. 14:1241283.
doi: 10.3389/fimmu.2023.1241283

Single-cell gene expression analysis using sequencing (scRNA-seq) has gained increased attention in the past decades for studying cellular transcriptional programs and their heterogeneity in an unbiased manner, and novel protocols allow the simultaneous measurement of gene expression, T-cell receptor clonality and cell surface protein expression. In this article, we describe the methods to isolate scRNA/TCR-seq-compatible CD4$^+$ T cells from murine tissues, such as skin, spleen, and lymph nodes. We describe the processing of cells and quality control parameters during library preparation, protocols for multiplexing of samples, and strategies for sequencing. Moreover, we describe a step-by-step bioinformatic analysis pipeline from sequencing data generated using these protocols. This includes quality control, preprocessing of sequencing data and demultiplexing of individual samples. We perform quantification of gene expression and extraction of T-cell receptor alpha and beta chain sequences, followed by quality control and doublet detection, and methods for harmonization and integration of datasets. Next, we describe the identification of highly variable genes and dimensionality reduction, clustering and pseudotemporal ordering of data, and we demonstrate how to visualize the results with interactive and reproducible dashboards. We will combine different analytic R-based frameworks such as *Bioconductor* and *Seurat*, illustrating how these can be interoperable to optimally analyze scRNA/TCR-seq data of CD4$^+$ T cells from murine tissues.

KEYWORDS
scRNA seq, scTCR seq, TCR (T-cell receptor), CD4 T cell, tissue CD4 T cell

Introduction

Single-cell sequencing-based technologies have significantly changed our view on cellular architecture and heterogeneity of samples (1–4). One particular example includes single-cell sequencing-based gene expression profiling (scRNA-seq) of individual cells (5, 6), which is based on the linear amplification of RNA derived from individual cells, followed by complex bioinformatic processing steps and identification of cell types in an unbiased way (7–9). Despite differences in technology and chemistry (benchmarked in (10)), single-cell sequencing experiments generally require four main steps (11).

First, tissues or organs have to be processed and digested to liberate target cells from the extracellular matrix in the tissue network. This yields a single-cell suspension where our target cells are present in varying frequencies, based on the tissue itself and its state, often dependent on the experimental conditions under investigation (Inflamed? Tumor-bearing? Virus-infected? Necrotic? Hypoxic)?. These steps have to be optimized to yield viable, intact cells without causing too much stress or hypoxic damage (12). While experimental procedures are now established for various cell and tissue types, no detailed workflow is available for tissue T cells, covering not only the wet-lab steps but also providing comprehensive guidance on the bioinformatic analyses for the datasets generated. In previous work, we have developed protocols for isolating T cells from a wide array of murine and human tissues such as skin, visceral adipose tissue, colon, lungs, liver, or different lymphoid tissues, and used them for downstream sequencing-based analysis (13–16). In the methods paper presented here, we will describe protocols to isolate target cells from murine skin and secondary lymphoid tissues such as spleen and lymph nodes (LN). To promote best data quality, we pre-enrich for viable, high-quality target cells using fluorescence-activated cell sorting (FACS) before performing single-cell barcoding. This allows the removal of unwanted cells, dead cells, dying cells, and cellular debris that might otherwise compromise quality. We will provide advice on cell sorting and sample multiplexing using barcoded antibodies.

In the second critical step, highly pure target cells are processed ("barcoded") and genetic material is amplified. Single-cell isolation and library preparation can be based on several different technologies. This begins with limiting dilution technologies, magnetic cell sorting, micromanipulation using microscope-guided capillary pipettes or laser microdissection, sorting of single cells into a 96- or 384-well plate using FACS, to microfluidic systems that combine droplets and cells, and new technologies and adaptations are developed rapidly (12, 17). Importantly, all different technologies aim to capture a single cell in an isolated reaction volume to add a unique barcode specific for this cell.

In a third step, a sequencing library is prepared. In our case, we prepare not only one, but three libraries: a gene expression library that contains sequencing reads allowing to identify and quantify genes expressed on a cell-individual level (GEX library); a second library that contains quantitative information about cell surface protein expression and sample multiplexing (hashtag oligo) information (CSP library); and a library that contains the T-cell receptor usage information as nucleotide sequence (VDJ library). We will provide examples of all three libraries including PCR cycles, concentration, and electrophoresis-based size profiles.

The last step of the wet-lab procedure is the sequencing of all three libraries using high-throughput next-generation sequencing technology. At the end of the run, *FastQ* data are demultiplexed and copied from the sequencing instrument, and are now ready to undergo bioinformatic processing. In this methods paper, we provide an example dataset which we generated for this publication, where we applied the above-mentioned protocols to combine single-cell gene expression, TCR sequencing and cell surface protein barcoding to characterize and track CD4$^+$ T-cell clones from murine tissues, and which can be downloaded by the reader for reproducing our bioinformatics workflow. The datasets include several thousand CD4$^+$CD25$^+$ Treg cells from murine spleen, mesenteric LN (mLN), inguinal LN (iLN) as well as CD3$^+$ immune cells from skin, for all of which GEX, CSP and VDJ libraries have been generated and sequenced.

Using this dataset, we will describe a step-by-step bioinformatic workflow to help repeat and reproduce the results achieved using the methods described in this paper. First, we apply *FastQC* and *CellRangerMulti* to enable a combined analysis of all individual samples and determine overall sequencing quality and identify individual cells. Here, we discuss critical quality-related parameters that *CellRanger* delivers, and discuss typical results obtained with CD4$^+$ T cells from tissues. In a next step, we create the count matrix from *CellRangerMulti* output. We describe the pre-processing of scRNA-seq data using a variety of freely available R packages to perform quality control (QC) and filtering, dimensionality reduction, removal of doublets, evaluation of batch effect correction, and generating the final filtered dataset for analysis (following best practices outlined in (8) and (7)). We will also provide guidance on clustering, marker gene detection, cell type annotation, and interactive data exploration, accompanying this manuscript with a notebook containing all code and output from the analysis of our test dataset, which we refer to in the corresponding paragraphs. All essential steps for this end-to-end workflow are summarized in Figure 1.

Methods – experimental procedures

Isolation of T cells from murine spleen, mLN and iLN

To isolate T cells from murine secondary lymphoid tissues such as spleen or lymph nodes, a midline excision is performed to open the skin and abdominal wall, and forceps are used to expose the peritoneal cavity. The spleen is harvested immediately and stored at 4°C until use. To isolate mLNs, the cecum is located, the small intestine is moved to the side and the chain of mLNs are exposed. Using forceps, the tissue is harvested, placed in FACS buffer (Table 1) and stored at 4°C. Inguinal lymph nodes are collected

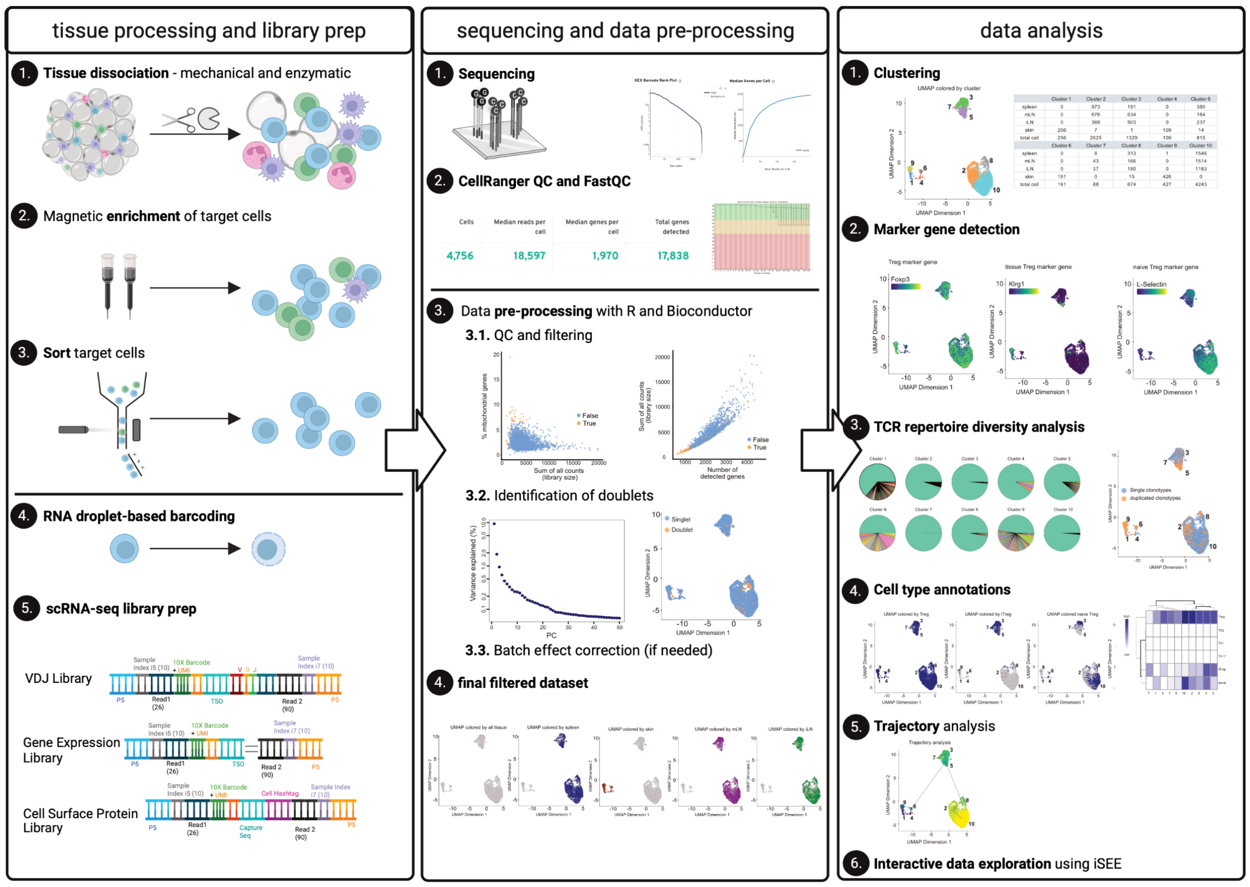

tissue processing and library prep
1. Tissue dissociation - mechanical and enzymatic
2. Magnetic enrichment of target cells
3. Sort target cells
4. RNA droplet-based barcoding
5. scRNA-seq library prep
VDJ Library
Gene Expression Library
Cell Surface Protein Library
sequencing and data pre-processing
1. Sequencing
2. CellRanger QC and FastQC
Cells 4,756
Median reads per cell 18,597
Median genes per cell 1,970
Total genes detected 17,838
3. Data pre-processing with R and Bioconductor
3.1. QC and filtering
3.2. Identification of doublets
3.3. Batch effect correction (if needed)
4. final filtered dataset
data analysis
1. Clustering
2. Marker gene detection
3. TCR repertoire diversity analysis
4. Cell type annotations
5. Trajectory analysis
6. Interactive data exploration using iSEE

FIGURE 1
Graphical abstract. The left panel describes tissue processing and library prep: Tissues harvested from an individual mouse are enzymatically and mechanically digested (1) and material is magnetically enriched for target cells (2) to make cell sorting (3) more efficient. After obtaining a pure target population (3), cells (labelled with Biolegend TotalSeqC anti-mouse Hashtagging antibodies) and 10X beads are loaded on the 10X Chromium controller (4) followed by scRNA-seq library preparation (5). The middle panel describes sequencing (1) quality control using *CellRangerMulti* and *FastQC* (2). Using R and Bioconductor, data can be pre-processed. These steps include QC and filtering (3.1), the identification of doublets (3.2), and, if necessary, batch effect correction (3.3) to yield the final, filtered dataset (4). The right panel describes data analysis, comprising the clustering (1), marker gene detection (2) as well as TCR repertoire diversity analysis (3). Furthermore, cell type annotations (4) and trajectory analysis can be performed (5). Moreover, an interactive data exploration by using iSEE can be done (6). Elements of this figure have been created with Biorender using figures and plots generated in this manuscript.

TABLE 1 Formulation for FACS buffer.

Ingredient	Manufacturer	Final concentration
Phosphate-buffer saline 10X	Gibco #10010023 or other	1X
FCS 100%	Sigma #F7524 or other	2%
Deionized water	NA	Up to final volume

from both hemispheres beneath the skin, placed in FACS buffer and stored at 4°C until use. To process the spleen, it is placed on a 100 µM filter unit and is mechanically dissociated using a plunger or forceps. Following centrifugation (2 min, 1000g, 4°C), red blood cells are lysed using a commercially available ACK lysis buffer (e.g., Thermo Fisher #A1049201). The cell suspension is filtered using a 70 µm strainer, resuspended in 500 µl FACS buffer, and cells are counted. To process LNs, the individual nodes are placed on a 100 µM filter unit and are mechanically dissociated using a plunger or forceps. Following centrifugation (2 min, 1000g, 4°C), the suspension is filtered using a 70 µm strainer, resuspended in 500 µl FACS buffer, and counted.

Afterwards, we add Fc blocking reagent (Miltenyi Biotec #130-092-575) to prevent unspecific binding of antibodies and beads, followed by specific labeling using 1 µg PE-conjugated anti-mouse CD4 (Clone RM4-5, Biolegend #100512) or 1 µg PE-conjugated anti-mouse CD25 (Clone PC61, Biolegend # 102008) antibodies in 500 µl and stain for 20 min at 4°C. After staining, cells are centrifuged (2 min, 1000g, 4°C), washed using 1000 µl of FACS buffer, and resuspended in MACS buffer (Table 2). Next, target cells are bound by anti-PE ultrapure microbeads (Miltenyi Biotec #130-105-639) for 20 min at 4°C, followed again by two centrifugation (2 min, 1000g, 4°C) and washing steps using 1000 µl of FACS buffer. Finally, samples are re-suspended in 500 µl MACS buffer. A 70µl filter unit is placed on an equilibrated MACS column (we recommend working at 4°C to prevent cellular degradation) and the sample is loaded. The column is washed twice with 5 ml MACS buffer.

Afterwards, the sample is eluted in 500 µL FACS buffer and stained for 30 min at 4°C using fluorescence-labelled antibodies as well as TotalSeqC anti-mouse Hashtagging antibodies (Biolegend *#155861* (C1), *#155863* (C2), *#155865* (C3), *#155865* (C4)). To increase TotalSeqC antibody labeling, it is recommended to wash cells 3-5 times with 500 µL FACS buffer after staining. For sorting, cells can be resuspended in 200 µL MACS buffer. In order to prevent aggregates during the co-staining of fluorescence-labeled antibodies and Biolegend TotalSeqC antibodies, it's recommended to centrifuge the antibody mix at 14,000 x g for 10 min at 4°C. Afterwards the supernatant should be transferred to a new tube and maintained at 4°C. The antibody aggregates will stay at the bottom of the original tube. For sorting, an example is shown in Figure 2. We recommend a gating strategy where $CD4^+$ or $CD25^+$ T cells are enriched to high purity using FACS, and dead cells, unwanted cell types and doublets are excluded. The target cells can be sorted into MACS buffer. A small part of the sorted population (target cells) can then be re-analyzed before downstream processing to determine post sort purity, viability, and cell recovery/sort efficiency. If the post sort QC indicates that cells are of good viability and purity (for troubleshooting see Table 3), the sample can be subjected to single-cell barcoding, as described later.

Isolation of T cells from murine skin tissue

To isolate T cells from skin tissue, hair must be removed from the back of the animal with an electric shaver and depilatory cream. The cream is applied for 2 minutes, followed by vigorous washing using tap water to remove hair. It is important that excess hair is completely removed to avoid complications during downstream filtration steps. After cleaning, the skin is separated from the dorsal surface, cut into small pieces, and transferred to a GentleMACS tube (Miltenyi Biotec #130-096-334) containing 10ml of skin digestion buffer (Table 4). We recommend 10ml digestion buffer for 0.5 g of skin tissue.

Then, the sample is digested using the GentleMACS Dissociator (program: *37_C_Multi_H*) or via orbital shaking in a preheated waterbath (37°C). After 90 minutes of digestion or completion of the GentleMACS program, the single-cell suspension can be cut again, centrifuged (10 min, 400g, 4°C), resuspended in 5000 µl FACS buffer and transferred to a 15 ml tube through a 100 µm filter unit. Then, the sample is centrifuged again (2 min, 1000g, 4°C), resuspended in 1000 µl FACS buffer and filtered into a new 1.5 ml tube using a 70 µm filter unit. The sample can now be stained for 30 min at 4°C using fluorescence-labelled antibodies as well as Biolegend TotalSeqC anti-mouse Hashtagging antibodies, as described before. For sorting, cells can be resuspended in 200 µL MACS buffer. An example of the sorting strategy of T cells from murine skin tissue is shown in Figure 2E To increase efficiency, it is beneficial to first enrich for $CD45^+$ immune cells (yield sort) by sorting target cells into MACS buffer, followed by a second purity sorting (4-way purity sort) of target cells (Table 5).

Single droplet barcoding of T cells for combined scRNA/TCR-seq

Target cells from spleen (12,500 $CD3^+CD4^+CD25^+$ Treg cells, TotalSeqC1), mLN (10,000 $CD3^+CD4^+CD25^+$ Treg cells, TotalSeqC2), iLN (7,500 $CD3^+CD4^+CD25^+$ Treg cells, TotalSeqC3) and skin (10,000 $CD3^+$ T cells, TotalSeqC4) have all been sorted into a single 1.5 mL Eppendorf tube containing 350 µL MACS buffer, and the sample collection tube was cooled to 4°C. It is important to process the sample quickly after sorting to decrease the number of dying/dead cells in the collection tube. Therefore, shortly after sorting, cells are pelleted by centrifugation (5 min, 300 xg, 4°C). Supernatant is removed and the sample is supplemented with master mix and beads to a final volume of 70 µL, loaded on a 10X Chromium Next GEM Chip K (10X Genomics #1000287) and processed on the 10X Chromium Controller (10X Genomics #120212), followed by cDNA amplification using the Chromium Next GEM Single Cell 5' Reagent Kit v2 (10X Genomics #1000263) and 5' Feature Barcode Kit (10X Genomics #1000256). Afterwards V(D)J amplification was done

TABLE 2 Formulation for MACS buffer.

Ingredient	Manufacturer	Final concentration
Phosphate-buffer saline 10X	Gibco #10010023 or other	1X
Bovine Serum Albumin 100%	Sigma #A4503 or other	0,5% (w/v)
Ethylenediaminetetraacetic acid	ThermoFisher #15575020	1 mM
Deionized water	NA	Up to final volume

from cDNA by using the Chromium Single Cell Mouse TCR Amplification Kit (10X Genomics #1000254) and GEX, CSP and VDJ library preparation according to the Library Construction protocol (10X Genomics #1000190). In Figure 3A, we show the elements of each library, including the sample indexes i5 and i7, read1 and read2 with their purpose and recommended sequencing length. In Figure 3B, cycle numbers and typical library sizes are shown. Upon completion of cDNA amplification and library preparation, the fragment length composition is usually evaluated using electrophoretic separation of the sample, for which we show examples in Figures 3C–F.

Methods – sequencing and QC strategy for scRNA-seq libraries

Next-generation sequencing of GEX, VDJ and CSP libraries

In Figure 3B, we listed the total number tagged and sorted cells and the total number of cells identified after sequencing. The recovery rates were 38.0% for spleen CD25$^+$ Treg cells, 40.1% for mLN CD25$^+$ Treg cells, 41.0% for iLN CD25$^+$ Treg cells, and 33.4%

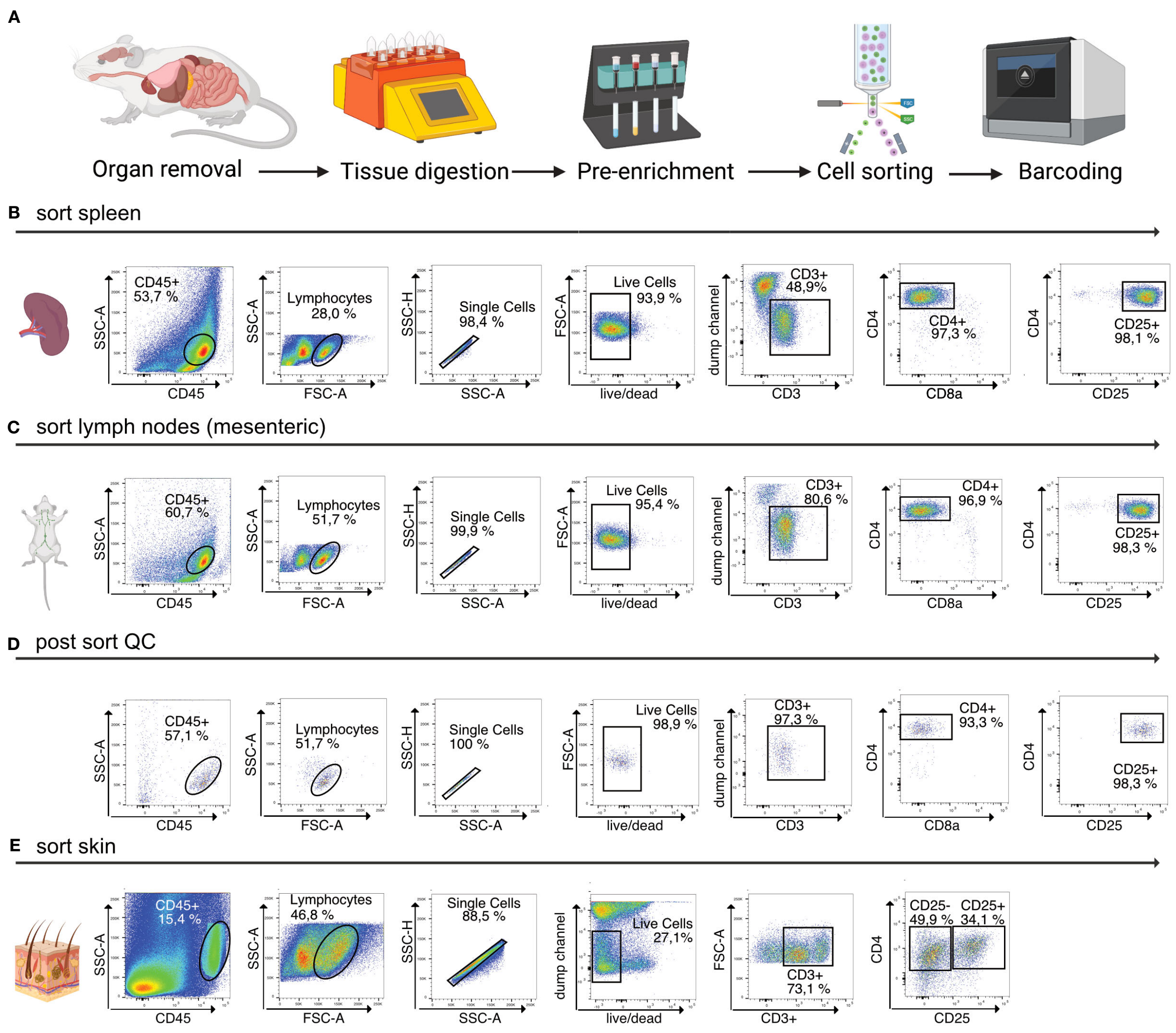


FIGURE 2
Overview of sample preparation for scRNA-seq of CD4$^+$ T cells from murine tissues. **(A)** Procedural overview. Organs are removed, followed by tissue digestion and pre-enrichment for CD4$^+$ T cells. These are then sorted, followed by single-cell barcoding using 10X Chromium controller. **(B, C)** Flow cytometry plots illustrating the gating scheme to isolate T cells from lymphoid tissues such as spleen and mLN. **(D)** Post sort QC of spleen, mLN, iLN CD25$^+$ sorted into the same collection tube. **(E)** Flow cytometry plots illustrating the gating scheme to isolate T cells from murine skin tissue. Figure elements created with Biorender.

TABLE 3 Troubleshooting and Recommendations.

Description	Solution
All cells are dead	Analyze buffer ingredients, optimize erythrocyte lysis procedure
Erythrocyte contamination	Optimize ACK lysis procedure
Low purity of $CD4^+$ or $CD25^+$ T cells	Use Fc blocking reagent, work at 4°C

TABLE 4 Formulation for skin digestion buffer.

Ingredient	Manufacturer	Final concentration
DMEM media	Gibco #41965	1X
Collagenase Type II	Sigma #C6885	4 mg/ml
Bovine Serum Albumin	Sigma #A4503	20 mg/ml
DNAse I	Roche #11284932001	20 µg/ml

TABLE 5 Troubleshooting and Recommendations.

Description	Solution
Clogging caused by hair	Additional filter steps after skin digestion get rid of hair and avoid clogging. Repeat hair removal if patches of hair remain.
Clogging during cell sorting	For cell sorting, samples should be filtered again immediately before acquisition and cooled at 4°C to avoid clogging.
Poor cell recovery after sorting	Use a two-step sorting protocol with a pre-sort ("yield") and a high purity sort (sort strategy "4-way-purity") mode.
Low expression of $CD4^+$ on T cells	Optimize processing time and amount of collagenase enzymes.

for skin $CD4^+$ T cells, with a mean recovery rate of 38.13%. Peripheral tissues that undergo enzymatic digestion, such as skin, liver, lung, or colon tissue, have varying recovery rates based on cell preparation steps, pre-enrichment, duration of processing, sort efficiency and sort setup. This can sometimes lead to recovery rates below 10% and requires optimization. Usually, all samples are sequenced in "one batch", and varying recovery rates can lead to "under- or over-sequencing" of libraries. Therefore, we recommend performing a pre-sequencing using only the gene expression (GEX) library. This reduces the cost for sequencing, allows for the identification and removal of low-quality and degraded samples, and increases the overall comparability of the datasets due to harmonized sequencing depth. Here, using a rough estimate of a projected cell number recovery (in our case, we estimate about 40% of sorted cells to be recovered later for bioinformatic analysis) helps to estimate the total number of reads required to sequence the GEX library to the desired depth. Now, for pre-sequencing, we only run 5%-10% of the estimated required reads to determine the approximate cell number for each library. These values are then used to sequence all libraries with a rather precise estimate of the required numbers of reads per library. In our lab, we routinely sequence 10X 5' scRNA-seq libraries using a paired-end run with 26-10-10-90 sequencing strategy with a 150-cycle high-output cartridge on a NextSeq 500/550 sequencing unit. In a typical run, read 1 identifies the i5 index (cell barcode) with 10 nucleotides and reads 26 nucleotides of 10X Barcode and UMI. On the reverse strand (read 2), primer P7 initiates the i7 read (sample index) with 10 nucleotides and reads 90 nucleotides of the cDNA (Figure 3A. The remaining 90 reads of read 2 are important for calling the gene (*GEX library*), the cell surface protein and/or hashtag oligo (e.g. TotalseqC), which appears at a fixed position (10th base) in read 2 (*CSP library*) or the VDJ information for the TCR (*VDJ library*). For the samples available as open access download alongside this paper, we used a 300-cycle high-output cartridge with a paired-end run and 26-10-10-149 sequencing strategy. In Figures 3C–F examples for library profiles from full length DNA (**c**), GEX (**d**), VDJ (**e**) and CSP (**f**) of a sample containing $CD25^+$ cells from spleen, mLN and iLN as well as CD3 skin T cells is shown. Since we used hashtag oligos (TotalseqC1-4) and pooled the different organs into one sample during sort, we only get one cDNA, GEX, VDJ and CSP library for all 4 samples.

Investigating sequencing quality using *FastQC*

To investigate whether we can estimate library quality, we ran FastQC on all L001 files generated from the different libraries. A plot labeled "per base sequence quality" shows the distribution of quality scores at each position in the read across all reads (Figure 4A). It can alert to whether there were any problems during sequencing. As the read 2 contains the information for the gene expression, we focus on this read in our analysis. Warnings related to "per base sequence content" are common for RNA-seq data and can be safely ignored in most cases. Also, warnings related to "per sequence GC content" has already been observed in literature (18) and can be ignored according to the manufacturer's guidelines. The "sequence duplication level" and "overrepresented sequences" error can indicate a low complexity library which could result from too many cycles of PCR amplification or less cDNA concentration before preparing the library. In this data set, we see a low contamination of a known primer sequence. If this contaminating sequence would be very high, it might be useful to get

A

GEX Library	read 1	index i5	index i7	read 2
purpose	10X Barcode, UMI	Sample Index	Sample Index	cDNA fragment
recommended length	26	10	10	90

VDJ Library	read 1	index i5	index i7	read 2
purpose	10X Barcode, UMI	Sample Index	Sample Index	VDJ info
recommended length	26	10	10	90

CSP Library	read 1	index i5	index i7	read 2
purpose	10X Barcode, UMI	Sample Index	N/A	Multiplex Barcode
recommended length	26	10	10	90

B

Sample	HTO	Sort Gate	% of all events	Total cells	Cells identified	% Identified/Total	Cells with associated TCR	% TCR/Identified
Spleen	C1	CD25+	7,2	12.500	4756	38,05	3412	71,74
mLN	C2	CD25+	25,6	10.000	4010	40,10	3208	80,00
iLN	C3	CD25+	17,4	7.500	3075	41,00	2509	81,59
Skin	C4	CD3+	1,7	10.000	3339	33,39	989	29,62

Library	PCR Cycles	Library (ng/ul)	Average size (bp)
GEX	12	54.8	459
VDJ	12	62.6	511
CSP	8	38.6	213

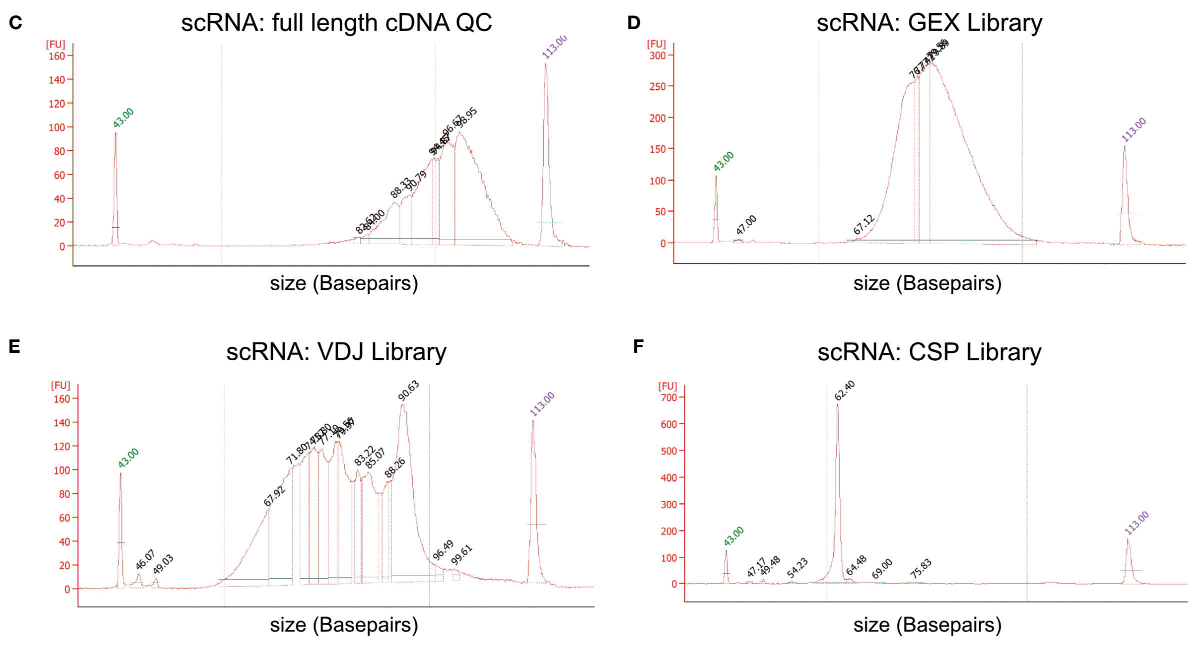
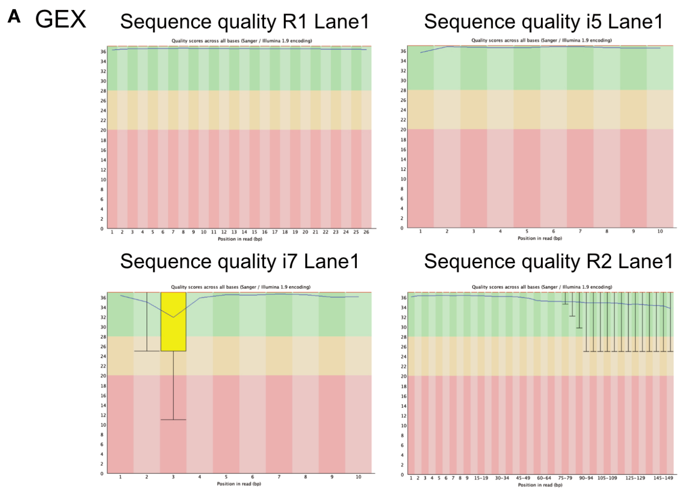
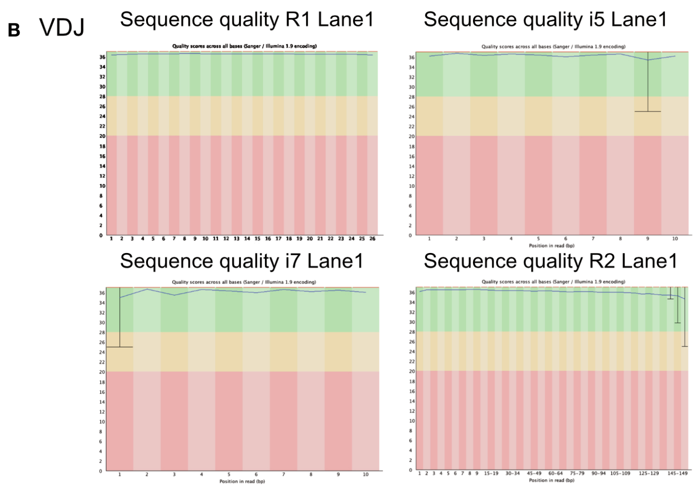


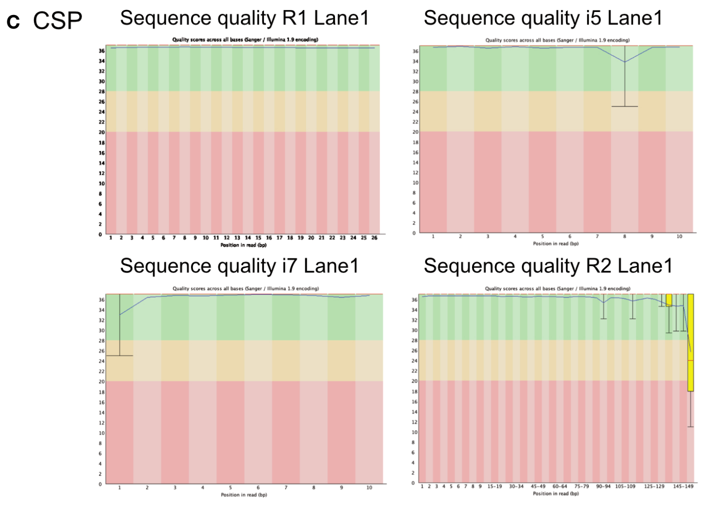
FIGURE 3
Overview of recovery and typical profiles for scRNA-seq libraries. **(A)** Overview of GEX, VDJ and CSP library and recommended sequencing length (source: 10X Genomics). **(B)** Tabular overview of parameters in scRNA-seq experiments. The percentage of all events indicates the total frequency of target cells (either CD4^{+} or CD25^{+} T cells) in all events from the sample. **(C-F)** Examples for library size profiles for samples with a good library profile listed in **(A)** for either **(C)** full length cDNA, **(D)** GEX Library, **(E)** VDJ Library or **(F)** Cell Surface Protein (CSP) library. Electrophoretic separation was performed on a Bioanalyzer.



FastQC GEX-Library	Read 1	Read 2	i5	i7
Basic statistic	OK	OK	OK	OK
Per base sequence quality	OK	OK	OK	OK
Per title sequence quality	OK	OK	OK	OK
Per sequence quality score	OK	OK	OK	OK
Per base sequence content	OK	Warning	ERR	ERR
Per sequence GC content	OK	Warning	ERR	ERR
Per base N content	OK	OK	OK	OK
Sequence length distribution	OK	OK	OK	OK
Sequence duplication levels	Warning	ERR	ERR	ERR
Overrepresented sequences	OK	ERR	ERR	ERR
Adapter content	OK	OK	Warning	Warning



FastQC VDJ-Library	Read 1	Read 2	i5	i7
Basic statistic	OK	OK	OK	OK
Per base sequence quality	OK	OK	OK	OK
Per title sequence quality	OK	OK	OK	OK
Per sequence quality score	OK	OK	OK	OK
Per base sequence content	OK	ERR	ERR	ERR
Per sequence GC content	OK	ERR	ERR	ERR
Per base N content	OK	OK	OK	OK
Sequence length distribution	OK	OK	OK	OK
Sequence duplication levels	ERR	ERR	ERR	ERR
Overrepresented sequences	OK	ERR	ERR	ERR
Adapter content	OK	ERR	Warning	Warning



FastQC CSP Library	Read 1	Read 2	i5	i7
Basic statistic	OK	OK	OK	OK
Per base sequence quality	OK	Warning	OK	OK
Per title sequence quality	OK	OK	OK	OK
Per sequence quality score	OK	OK	OK	OK
Per base sequence content	Warning	ERR	ERR	ERR
Per sequence GC content	OK	Warning	ERR	ERR
Per base N content	OK	OK	OK	OK
Sequence length distribution	OK	OK	OK	OK
Sequence duplication levels	OK	OK	ERR	ERR
Overrepresented sequences	OK	Warning	ERR	ERR
Adapter content	OK	ERR	Warning	Warning

FIGURE 4
FastQC report of the GEX Library, VDJ Library and CSP Library. Statistics of *FastQC* run for the GEX library **(A)**, VDJ library **(B)** and CSP library **(C)** on for read 1 (26 bp), read 2 (149 bp), i5 (10 bp) and i7 (10 bp). Errors and Warnings listed here as reported in *FastQC* documentation. Produced by *FastQC* (version 0.11.9).

rid of it before downstream analysis. As shown in the schematic overview (Figure 3A), the VDJ and CSP Library are very different from the GEX Library because they contain VDJ information and very few cell surface protein barcode sequences. FastQC is not tailored for analysis of such low-complexity libraries, but we included the results for reference (Figures 4B, C).

Methods – use of *CellRanger* to identify cells and investigate quality and quantity

In the previous sections, we described detailed protocols to isolate CD4$^+$ T cell populations from murine tissues such as spleen, LN or skin. Next, we provided advice on cell sorting and sample multiplexing using hashtag oligos (e.g. TotalSeqC), followed by single droplet barcoding and library preparation steps using 5' reagent kits. Sequencing of our three individual libraries (GEX, CSP and VDJ) will generate *FastQ* files ready for analysis using *CellRanger*, a software tool developed for single-cell sequencing-based datasets generated with chemistry from 10X Genomics. In the following paragraphs, we will describe the use of *CellRangerMulti* to extract individual samples and generate output files that allow a first glimpse on data quality and quantity.

Use of *CellRangerMulti* to enable a combined analysis of all individual samples

CellRangerMulti is a method for the combined processing scRNA samples by the use of specific multiplexing antibodies and officially supports the analysis of 3' multiplexed data. The 3' and 5' assays capture different ends of the transcript in the final library, and we used the 5' chemistry to generate GEX, CSP and VDJ libraries. Therefore, this type of analysis requires editing of the *CellRangerMulti* pipeline to be compatible with our datasets. Our pooled libraries contain four samples: splenic Treg cells (TotalSeqC1), mLN Treg cells (TotalSeqC2), iLN Treg cells (TotalSeqC3) and skin CD3$^+$ T cells (TotalSeqC4). In the first demultiplexing step, we use *CellRangerMulti* to assign cells to individual samples, a workflow described in Figure 5. First, we

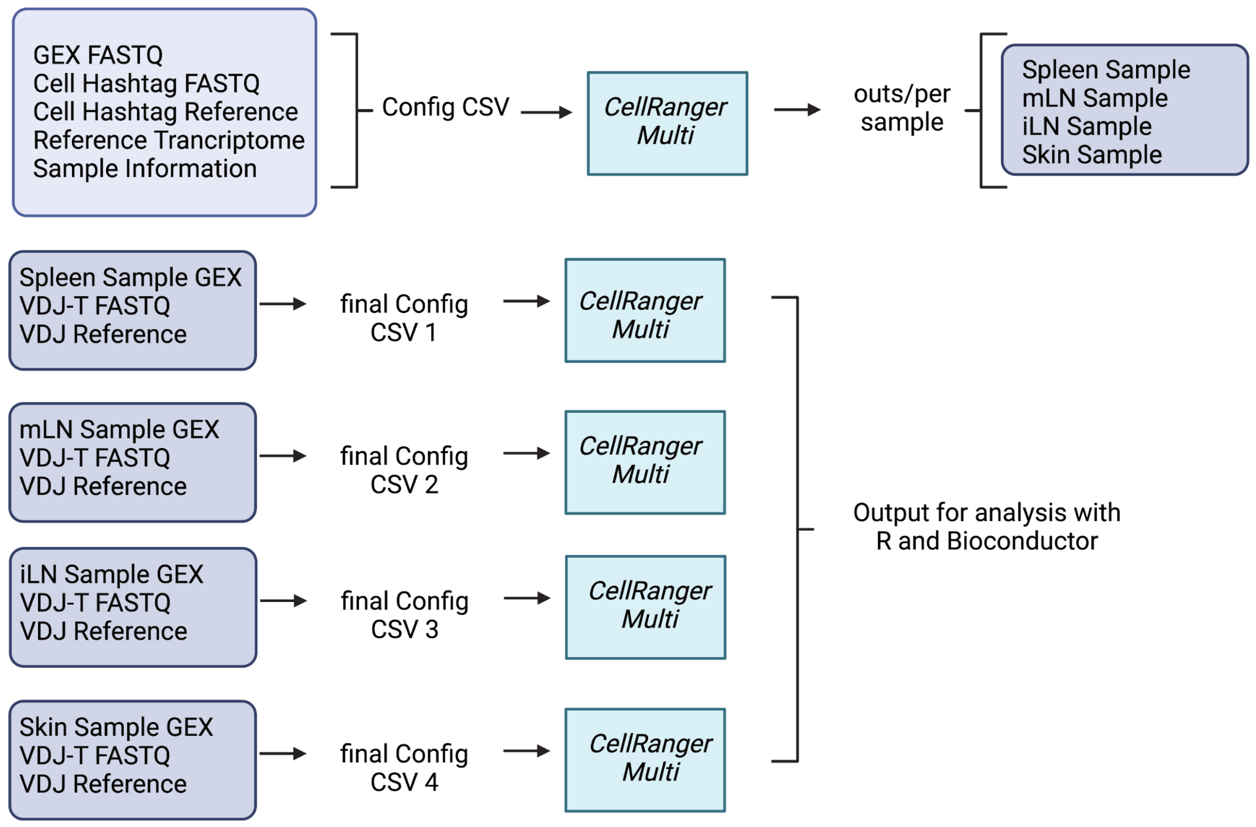


FIGURE 5
Schematic Overview of the *CellRangerMulti* Pipeline for combining 5' Single Cell Gene Expression Analysis with Cell Hashtag and VDJ T Cell Analysis. The 5' Chromium Next GEM Single Cell Immune Profiling cell hashing assay workflow starts with a demultiplexing step to assign pooled cells to individual samples (=hashtags). Afterwards, *CellRangerMulti* can be used to analyze individual samples and combine TCR with the GEX data. Created with Biorender.

BOX 1 Terminal input to run *CellRangerMulti* and assign cells.

Info: In this manuscript, commands to be entered in the terminal are prepended by the “$” symbol.

```
# run multi pipeline (combine GEX Library with Cell Surface Library)
$ cellranger multi\
-id=ddmmyy_multi\
-csv=./configmulti.csv

# configmulti.csv
$ cat configmulti.csv

[gene-expression]
reference,/path_to/refdata-gex-mm10-2020-A
Cmo-set,/path_to/cmo-set.csv
force-cells,
check-library-compatibility,false

[libraries]
fastq_id,fastqs,feature_types

[samples]
sample_id,cmo_ids
spleen,HTO_C0301
mLN,HTO_C0302
iLN, HTO_C0303
skin,HTO_C0304


# cmo-set.csv

$ cat cmo-set.csv

id,name,read,pattern,sequence,feature_type
C1,HTO_C0301,R2,5PNNNNNNNNNN(BC)NNNNNNNNN,ACCCACCAGTAAGAC,Antibody Capture
C2,HTO_C0302,R2,5PNNNNNNNNNN(BC)NNNNNNNNN,GGTCGAGAGCATTCA,Antibody Capture
C3,HTO_C0303,R2,5PNNNNNNNNNN(BC)NNNNNNNNN,CTTGCCGCATGTCAT,Antibody Capture
C4,HTO_C0304,R2,5PNNNNNNNNNN(BC)NNNNNNNNN,AAAGCATTCTTCACG,Antibody Capture



# Command to change to the directory where the CellRanger executable file lives and put it in your $PATH:

$ export PATH=/path_to/cellranger-7.0.1:$PATH
$ export PATH=${PWD}:$PATH

# Command to put other tools bundled with CellRanger in your path:
$ source/path_to/cellranger-7.0.1/sourceme.bash

# Make a new directory
mkdir bamtofastq

# Run bamtofastq
# You will need the path to the individual sample_alignments.bam. In addition, 10X recommends setting the # --reads-per-
fastq= argument higher than the total number of reads recorded.
bamtofastq --reads-per-fastq=2200000000/path_to/sample_alignments.bam/path_to_outputfolder/bamtofastq/name_of_new_folder
# after bam to fastq, identify the FASTQ directory corresponding to GEX:
cd/path_to_outputfolder/bamtofastq/name_of_new_folder

ls -ltsh
# Use samtools to identify the GEX file
source/path_to/cellranger-7.0.1/sourceme.bash

samtools view -H/path_to sample_alignments.bam

# Look for the @CO library info in the bottom

# Run CellRangerMulti final analysis again for each sample (include VDJ Library)
cellranger multi\
-id=ddmmyy_multifinal_organ1\
```

(Continued)

Continued

```
-csv=./configmultifinal.csv

# display the content of configmultifinal.csv
$ cat configmultifinal.csv

[gene-expression]
reference,/path_to/refdata-gex-mm10-2020-A
force-cells,5000
check-library-compatibility,false

[vdj]
reference,/path_to/refdata-cellranger-vdj-GRCm38-alts-ensembl-7.0.0

[libraries]
fastq_id,fastqs,feature_types
bamtofastq,/path_to_FASTQ/, Gene Expression
VDJ_FASTQ,/path_to_FASTQ/, VDJ
```

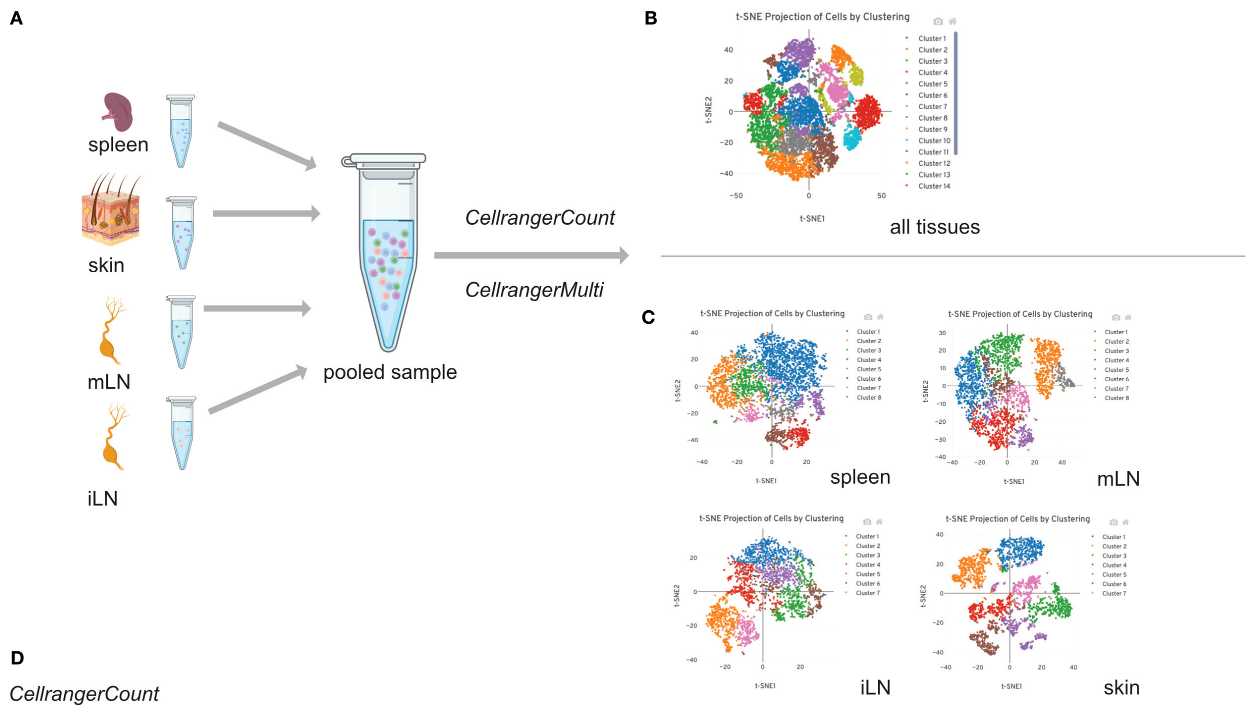
need to create a library comma-separated values (CSV) file which declares the input FASTQ data for the libraries that make up a cell multiplexing experiment (Box 1). Second, we need to create a cell hashtag reference. It declares the molecule structure and unique cell hashtag sequence of each hashtag (=TotalSeq) antibody present in the experiment. Each line of the CSV declares one unique cell hashtag.

The *CellRangerMulti* pipeline first extracts and corrects the cell barcode and UMI from the CSP library using the same methods as gene expression read processing. It then matches the cell hashtag read against the list of features declared in the cell hashtag reference. This is all described in specific sections of the config CSV file which requires the column [gene expression], [libraries] and [samples]. The [gene expression] section specifies the path to the reference transcriptome and the cell hashtag reference. The [libraries] section shows the path to the GEX FASTQs (GEX library) and cell multiplexing FASTQs (CSP library). The [sample] section includes a list of all samples and the corresponding hashtag. After creating these files, we run *CellRangerMulti* and assign cells to samples. By doing so, we also create BAM files of the individual samples in the pool. Those files are located in the individual directories for each sample. Since *CellRangerMulti* requires FASTQ files as the input, we convert the BAM files to individual FASTQ files. This can be done with the *bamtofastq* software tool which is bundled with *CellRanger*. The output of *bamtofastq* will display two directories per sample. After using *samtools*, which is also a part of the *CellRanger* bundle, we can distinguish the gene expression FASTQ from the cell hashtag FASTQ. In a final step, the T-cell receptor library can now be combined with the gene expression data. To do so, we run the *CellRangerMulti* again for every individual sample. We create a new final config CSV file for every individual sample and include the [vdj] section which describes the path to a VDJ reference. Each run produces output files which can then be used for further analysis with R.

Using metrics provided by *CellRanger* to evaluate quality and quantity of cells

In addition to creating outputs files which can be used for further analysis with R, *CellRanger* produces a web summary file in the output folder of the specified analysis directory. It is a good starting point for determining sample quality and quantity before starting with the analysis using R (as described in the next paragraphs in detail). Also, web summaries can be used to determine sample complexity and sequencing need (e.g. how many reads are still required per sample to have good coverage and even sequencing depth distribution between all samples).

Therefore, *CellRanger* is a useful tool for investigating important sample parameters on a first glimpse. In general, we need to distinguish between an output from *CellRangerCount* and *CellRangerMulti*. When performing single cell RNA experiments, it can be useful to first run *CellRangerCount*. This pipeline aligns sequencing reads from the FASTQ files to a reference transcriptome. Then, different filtering steps, barcode counting, and UMI counting allow to determine clusters and perform gene expression analysis. To discriminate *CellRanger count* from *CellRangerMulti*, outputs are shown in Figure 6A. The t-SNE plot derived from *CellRangerCount* (Figure 6B) gives an overview of the heterogeneity of the sample, which, in our case, contains cells from the different lymphoid and peripheral organs (spleen, mLN, iLN, skin). However, *CellRangerCount* cannot assign cells to the organ of origin, since multiplexing info from the CSP library is not processed. The cells in the t-SNE plot are colored by cluster and show cell-associated barcodes. The clustering analysis is based on grouping cells with similar gene expression profiles and allows a first glimpse of data complexity and quality. In our case, with $CD25^+$ or $CD4^+$ T cells from the different lymphoid and peripheral organs (spleen, mLN, iLN, skin), *CellRangerCount* generates a t-SNE with many different clusters, not too surprising because it counts all cells from the different organs (Figure 6B). In contrast to *CellRangerCount*, *CellRangerMulti* can break down individual samples (= organs) using the hashtag oligo information of the CSP Library. The t-SNE after running *CellRangerMulti* shows less heterogeneity for the Treg cell populations in spleen, mLN and iLN, as expected with a very defined cell type (Figure 6C). Within the lymphoid organs, the clustering is more compressed because we enriched and sorted for CD25$^+$ Treg cells for this dataset. In



D

CellrangerCount

	Estimated Cells	Mean Reads per Cell	Mean Genes per Cell	Confidently mapped reads (%)	Valid Barcodes (%)	Sequencing Saturation (%)
total	19.574	35.356	1939	76,4	92,9	75,3

E

CellrangerMulti

	Estimated Cells	Mean Reads per Cell	Mean Genes per Cell	Confidently mapped reads (%)	Valid Barcodes (%)	Sequencing Saturation (%)
spleen	4756	18.597	1970	89,99	100	75,32
mLN	4010	19.452	2003	82,55	100	75,51
iLN	3075	19.545	2015	82,69	100	75,55
skin	3339	17.804	1.696	79,46	100	75,05

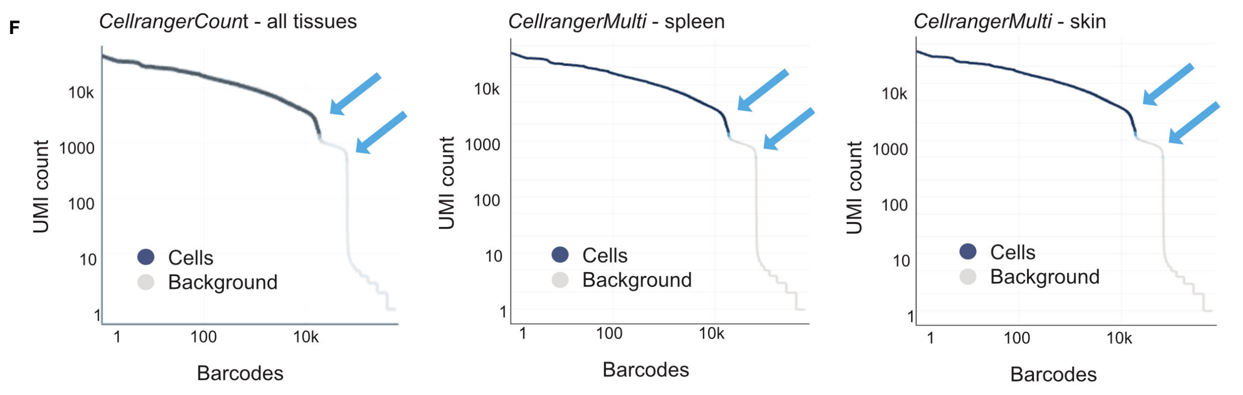


FIGURE 6
Interpretation of *CellRangerCount* and *CellRangerMulti* Output. Schematic overview of the experimental design **(A)** and *CellRangerCount* **(B)** and *CellRangerMulti* **(C)** output. Metric summaries for the *CellRangerCount* **(D)** and *CellRangerMulti* **(E)** and Rank Barcode plots **(F)** for all tissues, spleen and skin. Figure elements created with BioRender.

contrast to this, the clustering of the skin sample looks more heterogenous because it contains a larger subset of cells. If a complete lack of cluster structure appears in a usually rather heterogenous sample, this could indicate low sample quality or loss of single-cell behavior due to massive overloading or system failures.

In a table, we listed some of the web summary metrics which are shown when running *CellRangerCount* (Figure 6D) and *CellRangerMulti* (Figure 6E) on our sample dataset. *CellRanger* estimates the number of cells which are defined as the number of barcodes associated with at least one cell. As listed in Figure 3, using the protocols described in this paper, we should recover around 40% of original cell input as cells that are identified using *CellRanger*. However, a difference between the number of cells when running *CellRangerCount* compared to *CellRangerMulti* appears, which can be explained by the fact that we generally do not achieve 100% binding of the hashtag antibodies (= TotalSeqC barcodes) to the cells. Another important parameter displayed by *CellRanger* is the median reads per cell, which accounts for the total number of sequenced reads divided by the number of barcodes associated with cell-containing partitions. This information is helpful for planning a re-sequencing of the samples if not enough reads have been acquired, so that the recommended minimum of 20.000 reads/cell can be achieved. Another metric, median genes per cell, defines the median number of genes detected per cell-associated barcode. It also depends on sequencing depth and the total number of cells, and a low number of genes per cell can indicate low sequencing depth, low library quality or low transcriptional diversity of the cells. Another parameter linked to sample quality is the fraction of reads mapped confidently to the reference transcriptome. In our dataset, the lowest fraction of reads mapped to the murine genome is observed for the skin sample (79.46%), which, however, still is well above the lower threshold of 30% given by the manufacturer. Another quality-related parameter is the fraction of valid barcodes matching a whitelist. A value lower than 75% may indicate sequencing issues such as low quality of read 1. Finally, *CellRanger* computes sequencing saturation, which is an indicator of library complexity and sequencing depth. Lower sequencing saturation indicates that much of the library complexity was not captured by sequencing and that re-sequencing the sample could potentially increase gene expression coverage.

The *CellRanger* output files also contain a barcode rank plot where all barcodes detected during sequencing are plotted in decreasing order of UMIs associated with the particular barcode (Figure 6F). The shown barcode rank plot originates from the *CellRangerCount* (all tissues) and *CellRangerMulti* (spleen, skin) output. *CellRanger* uses the number of UMIs detected in each gel bead in emulsion (GEM) to determine whether the GEM contains a cell (declared as a cell) or not (declared as background). In a typical sample, a steep drop-off can be found and indicates good separation between the cell-associated barcodes and the barcodes associated with an empty GEMs. As mentioned in manufacturer's guidelines, every barcode plank plot has a distinctive shape with steep drop-offs indicated by blue arrows (Figure 6F). In a very heterogenous sample, the plot can appear bimodal, but a clear separation between the cells and background should always be present. If the separation is not good and the barcode rank plot shows a round curved shape, this may indicate low sample quality or loss of single-cell behavior due to technical failures.

Methods – data processing with R, Bioconductor and Seurat

In the previous paragraph, we discussed the use of *CellRanger* to produce output files which can then be used for further analysis with R. Now, we describe the pre-processing of scRNA-seq data using a variety of openly available R packages, which can be found on CRAN (https://www.R-project.org/) and Bioconductor (8). The pre-processing steps include quality control (QC) and filtering, dimensionality reduction, removal of doublets, evaluation of batch effect correction, which generates the final filtered dataset for analysis. For data pre-processing and analysis, we provide a rendered notebook file containing all code and output from the analysis of our test dataset in the supplement, which we refer to in the corresponding paragraphs (Supplementary Material or downloadable from https://github.com/imbeimainz/scRNAseq_scTCRseq_TissueTcells). In this manuscript, we will mainly discuss the analysis of the data using packages available on Bioconductor. However, the notebook will also provide the code for a pipeline using the Seurat package (19) and discuss the features of this pipeline.

Creating the count matrix from *CellRangerMulti* output

scRNA-seq data analysis is performed on a count matrix, containing the counts (i.e. number of UMI or reads) per gene in each cell. scRNA-seq data is usually very sparse due to several factors such as dropout events, low mRNA abundance in the cells, and a combination of biological and technical variation (20, 21). In our workflow, the count matrix is constructed from the feature-barcode matrix information generated by *CellRangerMulti*. For scRNA-seq data, *CellRanger* provides an unfiltered feature-barcode matrix and a filtered feature-barcode matrix. The unfiltered feature-barcode matrix contains every barcode from a fixed list of known barcodes that have at least one count. These can contain background and cell-associated barcodes. The filtered feature-barcode matrix, however, only includes detected cell-associated barcodes. In our experience, unfiltered data contain a lot of cellular debris and background noise. However, if desired by the user, there are also R packages like *DropletUtils* (22) which provide methods to process the unfiltered count matrix to remove the unwanted noise. In our workflow, we will present the approach working on the filtered data and refer users to the *DropletUtils* documentation on how to work with unfiltered data. We use the *Read10X()* function from the *Seurat* package (19) to read in the filtered feature-barcode matrix information (Box 2, Table 6). This function returns a sparse matrix which stores the count information

BOX 2 R code for creating SingleCellExperiment objects.

```
# Function to read in the data
# provide all the filepaths to the count data as a list
# as well as a list of the respective tissues
  readDataset <- function(filepath_list, tissue) {
    sceRNA <- list()
    # iterate over each sample of the input data
    for (i in 1:length(filepath_list)) {
      # read the count data
      counts <- Read10X(filepath_list[[i]])
      # generate a SingleCellExperiment object
      sce = SingleCellExperiment(assays = list(counts = counts))
      # Add the tissue type information as meta data
      sce$tissue <- rep(tissue[[i]], ncol(sce))
      sceRNA <- c(sceRNA, sce)
  }
  # return the list of SingleCellExperiment objects
  return(sceRNA)
}
# input data is stored in a folder called data
filepaths <- c("./data/iLN",
               "./data/mLN",
               "./data/skin",
               "./data/spleen")

sceRNA <-
  readDataset(filepaths, tissue = c("iLN", "mLN", "skin", "spleen"))
# set the names of the objects in the list so that we can easily identify and
# access the different tissues
names(sceRNA) <- c("iLN", "mLN", "skin", "spleen")
# Now have a look at the data
sceRNA
```

with genes as rows and samples as columns. We further process the resulting count matrix using the *SingleCellExperiment()* constructor from the *SingleCellExperiment* package (8). We repeat this process for all samples in the experiment. In addition to the counts, we store the tissue of origin for each sample as metadata in the respective *SingleCellExperiment* object. This information will be essential for some of the presented downstream analyses steps, especially for compelling and informative data visualizations. See section "1 Create SingleCellExperiment" in the notebook for the respective code of this analysis.

Gene level annotation

In a processing step before data analysis, we perform a gene-level annotation based on the input data (Box 3, Table 7). This gene-level annotation is used to facilitate the downstream applied analysis steps. During the annotation, the gene identifiers of the input data are mapped to their respective gene name using the *AnnotationHub* package (23). Gene names are usually more widely used and discernible and hence facilitate many of the downstream analysis steps, such as marker gene detection and cluster marker

TABLE 6 Troubleshooting and Recommendations.

Description	Solution
Directory provided does not exist	It seems like the directory stated does not exist or is not found. Check if the directory location is spelled correctly and directory hierarchy matches the current working location.
filtered feature-barcode matrix folder does not contain features.tsv file	In the used version of CellRanger (v. 7.1) the features.tsv files is called genes.tsv. Please use this file as features file. Please note that you have to rename the file to features.tsv as the Read10X()expects this filename.
… file not found	The Read10X() function is rather stringent concerning filenames (at least as of v. 4.3.0) and expects the files to be named *matrix.mtx, barcode.tsv* and *features.tsv.* If the files have any other name (e.g. a sample prefix), the function will not find the files. Please rename the files following the mentioned naming convention.

BOX 3 R code for gene level annotation.

```
sce <- sceRNA$iLN

# set up the annotation hub
ah <- AnnotationHub()
# extract the indentifiers and names for mouse data
query(ah, c("musculus", "Ensembl", "EnsDb"))
ens.mm.v102 <- ah[["AH89211"]]
genes(ens.mm.v102)[, 2]

# search for the mitochondrial genes
is.mito <- grepl("^mt-", rownames(sce))

  chr.loc <- mapIds(
  ens.mm.v102,
  keys = rownames(sce),
  keytype = "GENENAME",
  column = "SEQNAME"
)
is.mito <- which(chr.loc == "MT")
is.mito
```

identification. Besides the annotation of gene names, we also determine which genes of the input data map to the mitochondrial portion of the genome as this is later used for filtering and quality control. See section "2 Gene level annotation" in the notebook for the respective code of this step.

Extracting T cells from the data using linked TCR information

Before we apply quality control procedures to our data, we would like to filter our dataset for T cells with productive TCR chain information. For this, we have to use the information of the T-cell receptor (TCR) stored in the VDJ library. Only cells with TCR information will be kept in our data. In order to filter our data set for T cells, we add the information on the TCR chains and the clonotype of each cell to our *SingleCellExperiment* objects (Box 4). In our specific workflow, we also must transform the clonotypes as we have processed each sample individually using *CellRanger*. In order to work with shared clonotypes between tissues, we first apply a transformation step to assign identical TCR chains the same clonotype id (Box 5). Afterwards, we save the harmonized TCR chain and clonotype information as meta data in our *SingleCellExperiment* objects. We also provide a list of the transformed TCR chain and clonotype information with the data of this manuscript for follow-up. If the information of the TCR is not available, but an analysis of solely T cells is desired, users can follow this presented workflow up until the cell type annotation step. After this step, the data can be filtered for cells which were annotated as T cells and the workflow can be repeated from the beginning. For more information, please see section "3 Extracting T cells using T chain receptor information" in the notebook.

Per sample quality control and filtering of low-quality cells

A well-defined filtering strategy to select for high-quality cells is highly recommended before analysis. Different quality parameters and metrics can be used to filter out cells of low quality (24). In this workflow, we mainly use a combination of three quality parameters: the library size, the number of features and the percentage of

TABLE 7 Troubleshooting and Recommendations.

Description	Solution
No genes map to the mitochondrial genome	It could be that the pattern used to search for mitochondrial genes does not match the pattern of mitochondrial genes in the data. Please check that these two patterns are identical (usually follow the lines of 'MT', 'mt', 'Mt' or 'chrM').
No gene names found/all gene names are 'NA'	It could be that the species you are using for the annotation does not match your data. Please check that the correct species is specified. Another reason could be that the wrong id type was specified. Please check that the id type matches your gene ids.

BOX 4 R code for extraction of T Cells using TCRs.

```
addTCRMetaData <- function(sce, tcr_filepath, clonotypes_filepath) {
  # Read in the information about the TCRs
  tcr <- read.csv(tcr_filepath)
  clonotypes <- read.csv(clonotypes_filepath)

  # Remove duplicated barcodes as the information is identical.
  tcr <- tcr[!duplicated(tcr$barcode)],

  # Subset to only barcode and raw clonotype column as we only use those.
  tcr <- tcr[, c("barcode", "raw_clonotype_id")]
  # Rename column to match to the clonotypes file
  names(tcr)[names(tcr) == "raw_clonotype_id"] <- "clonotype_id"

  # Extract the TCR chain information from the clonotypes file through matching
  # of the clonotypes.
  tcr <- merge(tcr, clonotypes[, c("clonotype_id", "cdr3s_aa")])

  # Reorder columns, set barcodes as rownames (to match the scRNA data)
  # and remove the barcode column as it is no longer necessary.
  tcr <- tcr[, c(2, 1, 3)]
  rownames(tcr) <- tcr[, 1]
  tcr[, 1] <- NULL

  # Add the TCR chain and clonotype information as metadata to the data
  clonotype <-
  tcr$clonotype_id[match(colnames(sce), rownames(tcr))]
  sce$clonotype <- clonotype
  cdr3s_aa <- tcr$cdr3s_aa[match(colnames(sce), rownames(tcr))]
  sce$cdr3s_aa <- cdr3s_aa

  # filter out those cells without a clonotype because they are not of interest
  # for us
  sce <- sce[,!is.na(sce$clonotype)]
  return(sce)
}

# Add the information of the TCR chains and the clonotypes to our data
sceRNA$iLN <- addTCRMetaData(
  sce = sceRNA$iLN,
  tcr_filepath = "./data/iLN/filtered_contig_annotations.csv",
  clonotypes_filepath = "./data/iLN/clonotypes.csv"
)
sceRNA$iLN

sceRNA$mLN <- addTCRMetaData(
  sce = sceRNA$mLN,
  tcr_filepath = "./data/mLN/filtered_contig_annotations.csv",
  clonotypes_filepath = "./data/mLN/clonotypes.csv"
)
sceRNA$mLN

sceRNA$skin <- addTCRMetaData(
  sce = sceRNA$skin,
  tcr_filepath = "./data/skin/filtered_contig_annotations.csv",
  clonotypes_filepath = "./data/skin/clonotypes.csv"
)
sceRNA$skin

sceRNA$spleen <- addTCRMetaData(
  sce = sceRNA$spleen,
  tcr_filepath = "./data/spleen/filtered_contig_annotations.csv",
  clonotypes_filepath = "./data/spleen/clonotypes.csv"
)
sceRNA$spleen
```

mitochondrial DNA. All of these can be used to determine the quality of the cells. The library size is the sum of all counts in one cell, which should be sufficiently high for each cell. A small/low library size indicates possible cell death of the respective cell. However, an unusually large library size could also indicate doublets (i.e., multiple cells sequenced in one droplet). The number of detected features (in this case, genes) in each individual cell should as well be sufficiently high to ensure adequate sequencing of the cells. The last quality parameter, the percentage of mitochondrial DNA captures the percentage of reads

BOX 5 R code for harmonization of clonotypes.

```
# set up clonotype data frame
df_clonotypes <- data.frame(
  clonotype = sceRNA$iLN$clonotype,
  clonotype_n = as.numeric(gsub("clonotype", "", sceRNA$iLN$clonotype)),
  cdr3s_aa = sceRNA$iLN$cdr3s_aa
)
df_clonotypes <- df_clonotypes[order(df_clonotypes$clonotype_n)],
filter <-!duplicated(df_clonotypes$clonotype)
df_clonotypes <- df_clonotypes[filter],

# function to transform the clonotypes
addClonotypesToDataFrame <- function(clonotypes_df, sce) {
  n_last_clonotype <- max(clonotypes_df$clonotype_n)
  for (i in 1:ncol(sce)) {
    chain <- sce$cdr3s_aa[[i]]
    # if there is no clonotype with the same chain, the clonotype is new
    # and should be added to the data frame
    if (!any(which(clonotypes_df$cdr3s_aa == chain))) {
      n_last_clonotype <- n_last_clonotype + 1
      clonotypes_df <- rbind(clonotypes_df,
                              c(
                              paste("clonotype", n_last_clonotype, sep = ""),
                              as.numeric(n_last_clonotype),
                              chain
                              ))
    }
  }
  # transform the clonotype number back to a numeric
  clonotypes_df$clonotype_n <-
    as.numeric(clonotypes_df$clonotype_n)
  return(clonotypes_df)
}


df_clonotypes <-
  addClonotypesToDataFrame(df_clonotypes, sceRNA$mLN)
df_clonotypes <-
  addClonotypesToDataFrame(df_clonotypes, sceRNA$skin)
df_clonotypes <-
  addClonotypesToDataFrame(df_clonotypes, sceRNA$spleen)
# function to change clonotypes for all samples

changeClonotypes <- function(sce, clonotypes_df) {
  for (i in 1:ncol(sce)) {
    chain <- sce$cdr3s_aa[[i]]
    new_clonotype <-
      clonotypes_df[which(clonotypes_df$cdr3s_aa == chain)[1]],
    sce$clonotype[[i]] <- new_clonotype$clonotype
  }
  return(sce)
}


sceRNA$mLN <- changeClonotypes(sceRNA$mLN, df_clonotypes)
sceRNA$skin <- changeClonotypes(sceRNA$skin, df_clonotypes)
sceRNA$spleen <- changeClonotypes(sceRNA$spleen, df_clonotypes)
```

in a cell that map to the mitochondrial genome. An unusually large number of reads assigned to mitochondrial genes in a cell indicates cell death and hence low-quality cells. For the quality control, it is advisable to operate on a per-sample level instead of applying the quality control metrics for all samples combined. The individual samples might have different levels of quality due to being sequenced or processed individually or different biological prerequisites such as tissue specific properties. Hence, only one run of quality control metrics combined on all samples could falsely indicate cells of low quality because of the above-mentioned characteristics. Furthermore, also samples that were generated in different batches should be handled separately. The sequencing properties of the individual batches can greatly differ and hence as well influence the resulting quality metrics (25). In our workflow, we use the *addPerCellQC()* function of the *scater* package (24), which follows a data-driven approach for determining adequate threshold values (Box 6). This function first determines the median across all cells for the above-mentioned quality control parameters.

BOX 6 R code for quality control and filtering (identical for all samples, showcase iLN).

```
iLN <- sceRNA$iLN

rowData(iLN)$gene_name <- rownames(iLN)
rowData(iLN)$location <- chr.loc
iLN <- addPerFeatureQC(iLN)

rowData(iLN)

iLN <- addPerCellQC(iLN, subsets = list(Mito = is.mito))
qcstats <- perCellQCMetrics(iLN, subsets = list(Mito = is.mito))
filtered <-
quickPerCellQC(qcstats, percent_subsets = "subsets_Mito_percent")
filtered
colSums(as.data.frame(filtered))

table(filtered$low_n_features, filtered$high_subsets_Mito_percent)

# Flag the low quality cells as discard
iLN$discard <- filtered$discard

# Plot the percent of mitochondrial RNA for each cell, color the cells by
# whether they should be discarded or not
plotColData(iLN, y = "subsets_Mito_percent", colour_by = "discard")

# Plot the library size
plotColData(iLN, y = "sum", colour_by = "discard")

# Plot the number of detected genes
plotColData(iLN, y = "detected", colour_by = "discard")

# Plot mitochondrial RNA percentage against library size
plotColData(iLN,
            x = "sum",
            y = "subsets_Mito_percent",
            colour_by = "discard") +
  labs(x = "Sum of all counts (library size)",
       y = "Percent mitochondrial genes")

# Plot library size against number of detected genes
plotColData(iLN,
            x = "detected",
            y = "sum",
            colour_by = "discard") +
  labs(x = "Number of detected genes",
       y = "Sum of all counts (library size)")

# Assign the data back to our object
sceRNA$iLN <- iLN
# As this report is only for exploratory analyses we do not filter out any cells
# Otherwise you could do
# sceRNA$iLN <- iLN[,!discard]
```

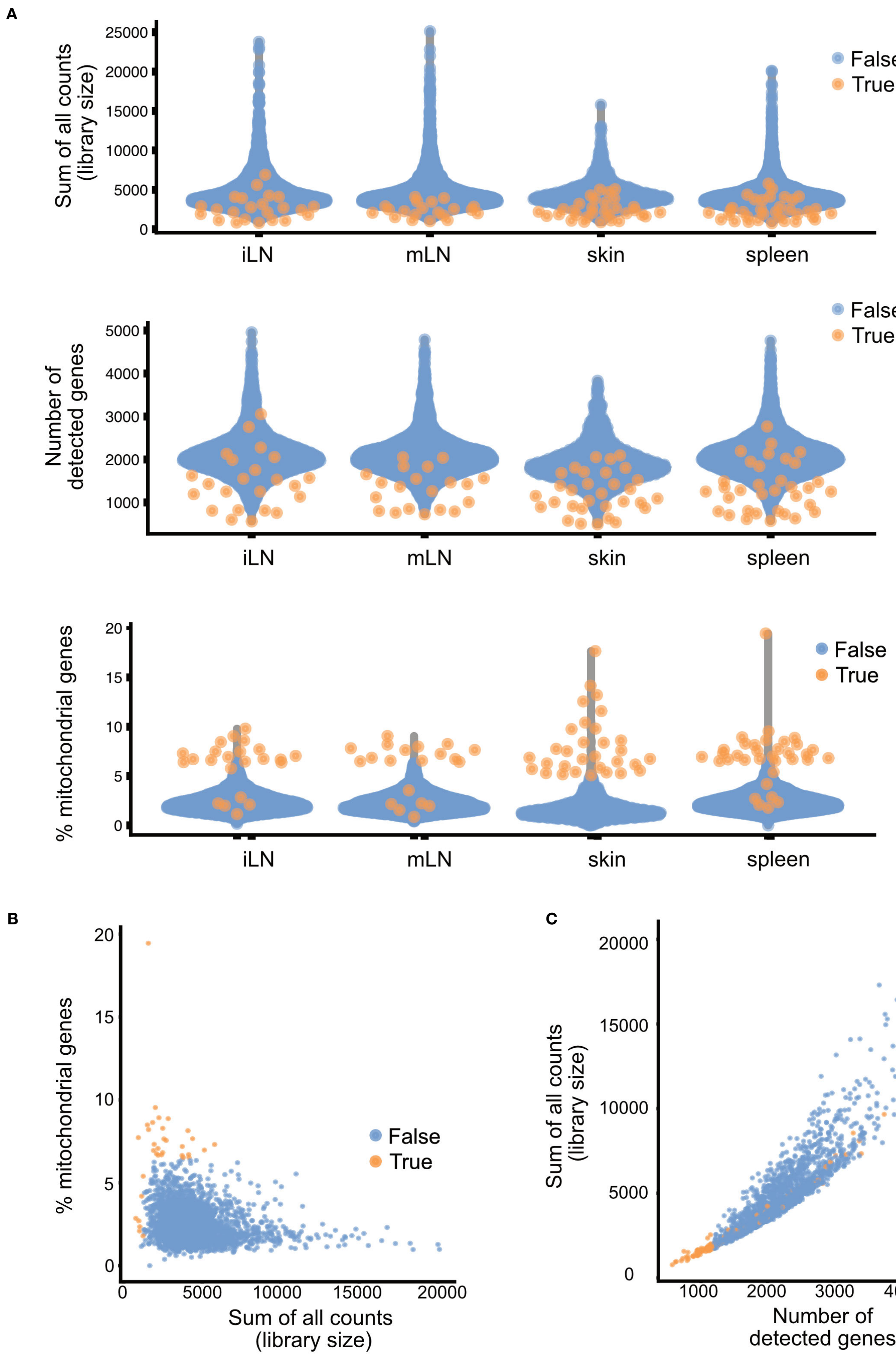


FIGURE 7
Summary of quality control metrics. **(A)** Plots of the library size, number of detected genes and mitochondrial content for each of the samples. **(B)** Scatter plots of the library size and mitochondrial content and **(C)** library size and number of detected genes. Each dot in the plot represents a cell, blue cells are of high quality, orange cells are of low quality and should be filtered out.

Following, for each cell the median absolute deviation (MAD) is calculated. If a quality control parameter of a cell deviates more than 3 MAD from the median in an undesired direction, the cell is considered an outlier. All cells which are considered outliers in at least one of the quality parameters are marked as low-quality cells.

After identification of low-quality cells, these cells can either be removed from the data or just marked as such. The removal ensures that these cells do not interfere downstream analyses and interpretation. However, it could also be the case that interesting subpopulations of cells are marked as low-quality cells because they exhibit one of the quality control parameters. One of such examples would be hepatocytes. These cells are highly metabolically active and hence will have a high number of mitochondrial genes. Hence, it is important to check for accidental removal of high-quality cells by plotting the different quality metrics against each other and evaluating how well the different quality metrics correlate for each sample. In Figure 7, the different quality metrics of our samples are displayed. Figure 7A shows the different quality control metrics of each sample, first the library size, then the number of detected genes and lastly the number of mitochondrial genes in the data. In Figure 7B, we plotted for the skin sample the library size against the percentage of reads mapping to mitochondrial genes, while Figure 7C plots the number of genes detected against the library size. Such a multivariate approach by considering different metrics simultaneously, can lead to better decision on which cells to retain for further steps and which cells to remove. However, as the workflow presented in this paper is only of explorative nature, we will not exclude cells of low quality here. For more information, please see section “4 Per sample Quality Control and filtering of low-quality cells” in the notebook.

Quality metrics and their correlation with TCR calling

In our analysis, we were also interested in whether the quality control metrics differed between cells with TCR and cells without TCR. Especially the mitochondrial content could be of interest. Hence, we compared the cells with TCR with those without (Table 8). These data illustrate that around 70% or more cells of the samples have TCRs. One exception being the cells of the skin, where only around 30% of cells have associated TCRs. We can also see that the percentage of cells with a high mitochondrial content (i.e low quality cells) is nearly doubled in the cells without TCR compared to the cells with TCR. This shows that filtering of cells with associated TCR also seems to work as a way of quality control and filtering of low-quality cells. Since the VDJ library is generated from cDNA, results here also depend on the quality of the cDNA library.

Doublet detection

In a single cell experiment, doublets are artificial observations in which two cells are sequenced as one cell. Those are especially

TABLE 8 Different summary statistics on the input data such as number of cells per sample, number of cells with and without TCR and percentage of cells with high mitochondrial content in cells with and without TCR.

Organ	Identified cells	Cells with associated TCR	% cells with high mitochondrial content	Cells without TCR	% cells with high mitochondrial content
spleen	4,756	3,412	3.66	1,344	6.18
mLN	4,010	3,208	3.11	802	5.11
iLN	3,075	2,509	3.5	566	6.71
skin	3,339	989	7.89	2,350	16.47

BOX 7 R code for doublet detection (identical for all samples, showcase iLN).

```
iLN <- sceRNA$iLN
# Doublet detection
iLN <- scDblFinder(iLN)
# Print a statistics table
table(iLN$scDblFinder.class)
# Assign the object back to save the information
sceRNA$iLN <- iLN
# Or you can assign the object back without the cells marked as doublets
# sceRNA$iLN <- iLN[, iLN$scDblFinder.class == “singlet”]
```

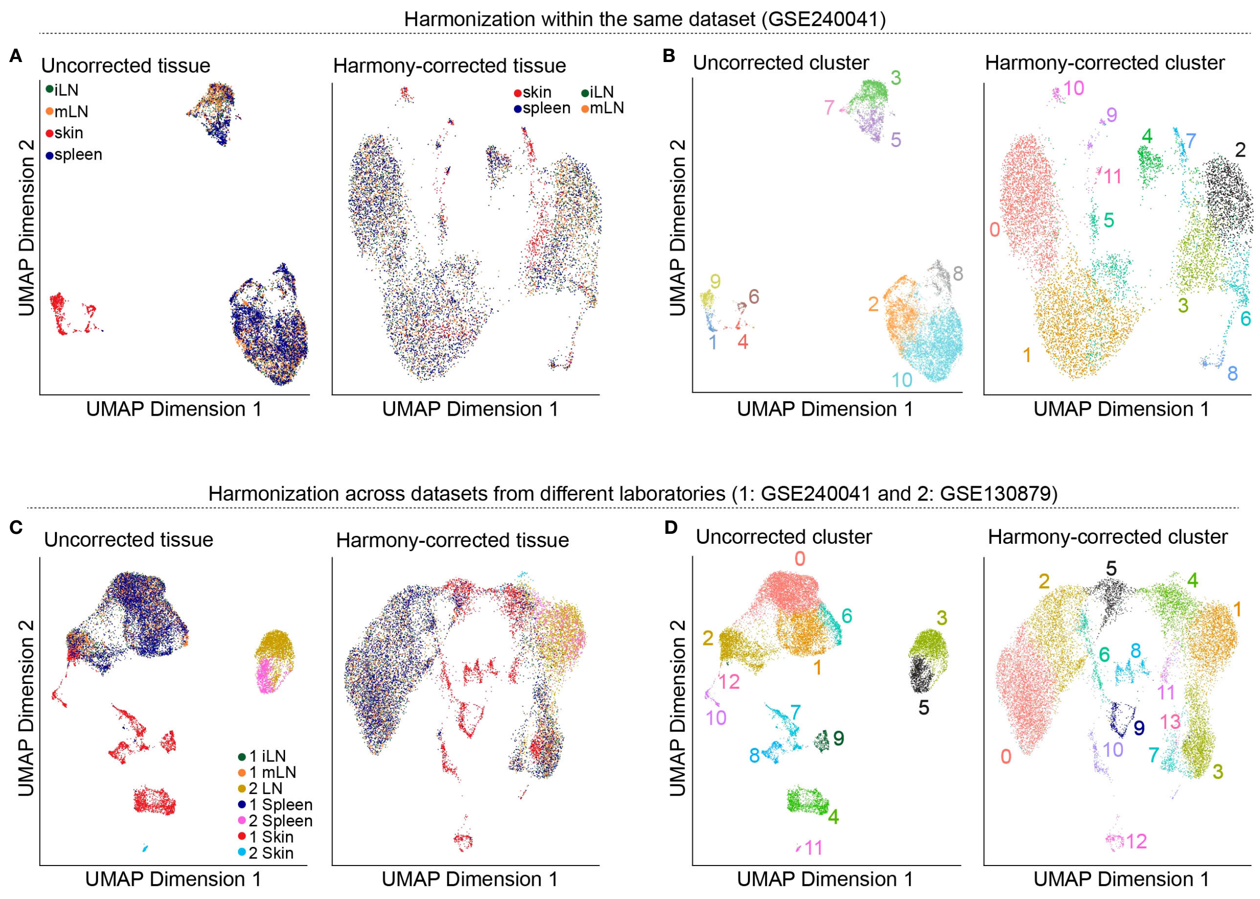


FIGURE 8
Harmonization of the data. **(A)** UMAP representation of the data before and after batch-correction using *harmony* colored by the tissue of the sample. **(B)** UMAP representation of the data before and after batch-correction using *harmony* colored by the clustering results. **(C)** UMAP representation of the integrated dataset with the publicly available data before and after batch-correction using *harmony* colored by the tissue of the sample. **(D)** UMAP representation of the integrated dataset with the publicly available data before and after batch-correction using *harmony* colored by the clustering results.

BOX 8 R code per-sample Normalization.

```
sceRNA <- lapply(sceRNA, logNormCounts)
```

common in droplet-based scRNA-seq protocols and usually arise from errors in cell sorting or capturing (26, 27). Doublets usually do not represent meaningful biological states and can influence the analysis of the data. For example, a mixture of two cells which were sequenced as one could be characterized as a transitionary state between two cell types or an intermediate population. The general approach for doublet detection in scRNA-seq data is the use of expression profiles of the cells. Based on their expression profile, doublets are computationally inferred from the data. In our workflow, we use the *scDblFinder()* function from the

BOX 9 R code feature selection.

```
all.dec <- lapply(sceRNA, modelGeneVar)
all.hvgs <- lapply(all.dec, getTopHVGs, prop = 0.1)
```

BOX 10 R code uncorrected integration.

```
# normalize counts across the samples
rescaled <- multiBatchNorm(sceRNA)

# extract the individual samples
iLN <- rescaled$iLN
mLN <- rescaled$mLN
skin <- rescaled$skin
spleen <- rescaled$spleen

# combine the selected features
combined.dec <- combineVar(all.dec)
chosen.hvgs <- combined.dec$bio > 0
sum(chosen.hvgs)

# Synchronizing the metadata for cbind()ing.
rowData(iLN) <-
  rowData(iLN)[, c("gene_name", "location")]
rowData(mLN) <-
  rowData(mLN)[, c("gene_name", "location")]
rowData(skin) <-
  rowData(skin)[, c("gene_name", "location")]
rowData(spleen) <-
  rowData(spleen)[, c("gene_name", "location")]

# merge individual objects into one final object
sce_merged <- cbind(
  iLN,
  mLN,
  skin,
  spleen
)
```

BOX 11 R code batch correction using harmony.

```
# Read in the data as seurat object as shown in "1 Create SingleCellExperiment"
seurat <- NormalizeData(seurat)
seurat <- FindVariableFeatures(seurat)
seurat <- ScaleData(seurat)
seurat <- RunPCA(seurat)
DimPlot(seurat, reduction = "pca")

seurat <- RunHarmony(seurat, group.by.vars = "tissue", plot_convergence = FALSE)

seurat <- RunUMAP(seurat,
                  reduction = 'harmony',
                  dims = 1:20)
seurat <- FindNeighbors(seurat,
                       reduction = "harmony",
                       dims = 1:20)
seurat <- FindClusters(seurat, resolution = 0.5)


DimPlot(seurat, reduction = "umap")

DimPlot(seurat, reduction = "umap", group.by = "tissue",
        cols = c("springgreen4", "darkmagenta", "tomato4", "darkblue"))
```

corresponding package (28) (Box 7). This function simulates expression profiles of possible doublets by randomly combining two cells of the data together before assigning each cell a doublet score based on its likelihood to be a double. Further details on the method and computation can be found in the *scDblFinder* documentation. Once doublets have been identified in the data, users can decide to either flag these cells or remove them completely from the data. In this context, it can be helpful to overlay the doublet classification over downstream computed clustering results to evaluate if the considered doublets are forming a distinguished cluster or display any relevant pattern. During the exploration of the data, we recommend to simply flag doublet cells but advocate for removal of the cells once the processed dataset is created. In Figure 9C, we can see that the identified doublets in our data to not follow a specific pattern. Overall, the number of detected doublets was also very low in our samples, less than 5% of all cells (see Figure 9B). For the doublet detection step, we refer readers to the provided notebook section "5 Doublet detection in the individual samples".

Per-sample normalization

In scRNA-seq data, often differences in the sequencing coverage between libraries arise (29). The cause for these variations is typically technical variation in cDNA capture or PCR amplification efficiency. Since this variability does not depict true biological signal in the data, it can distort the interpretation of expression profiles. In order to prevent the influence of the technical variation on data analysis, the data is normalized (30, 31).

Usually, normalization is applied to the different batches of the data at hand. The data presented in this paper does not consist of different batches but only of different tissues. However, treating the different tissues as individual batches and normalization across tissues at this point would be detrimental to downstream analysis steps. Hence, we decided to postpone the across tissue normalization to a later point of the workflow. Nevertheless, there are intra-sample normalization methods which should be applied at this point in the analysis. One of these normalizations is a log-scaling of the expression values, as implemented in the *logNormCounts* function of the *scran* package (32) (Box 8). This is beneficial for downstream analysis steps such as dimensionality reduction and clustering, as the expression values become more comparable without having too extreme values. For the normalization of the counts see section "6 Per-Sample Normalization".

Feature selection

In an exploratory scRNA-seq analysis, characterization of heterogeneity across individual cells is often one of the major goals. In order to quantify the differences in gene expression between cells, a subset of genes is selected such that this set contains useful information about the biological variation, while removing random noise and technical differences. This process of feature selection majorly impacts the performance of downstream analyses and methods. A commonly used approach of feature selection is the selection of the most variable genes across the cells (32). The approach is based on the assumption that the biological variation of the data will manifest as an increased variation in the affected genes, hence overshadowing technical noise and irrelevant biological variation (8). In our workflow, we use the *modelGeneVar()* function of the *scran* package

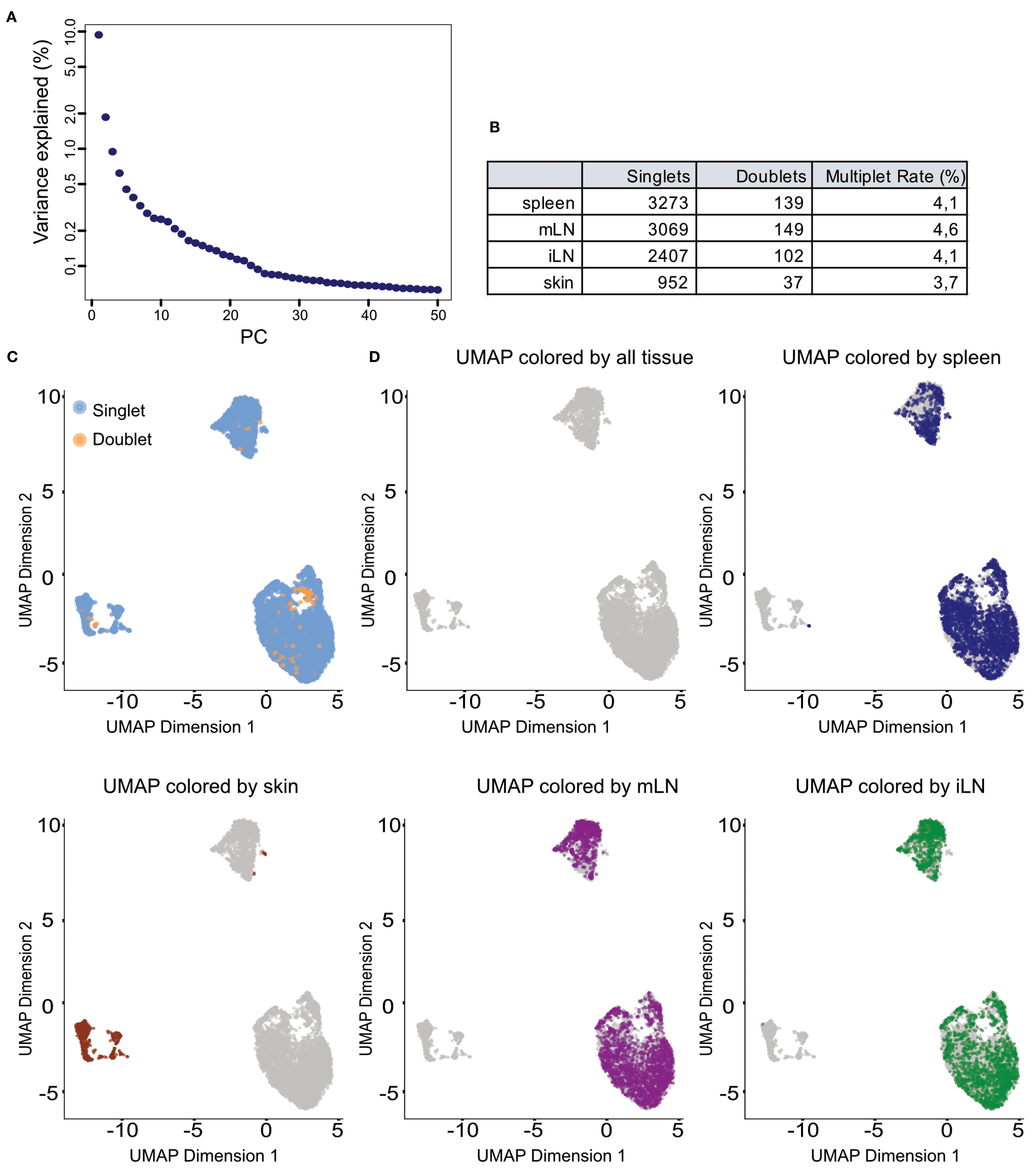
BOX 12 R code PCA.

```
set.seed(42)
sce_merged <- runPCA(sce_merged,
                     subset_row = chosen.hvgs,
                     BSPARAM = BiocSingular::RandomParam())

# Plot scree plot of the variance explained by each PC
percent.var <- attr(reducedDim(sce_merged), "percentVar")
plot(percent.var,
     log = "y",
     xlab = "PC",
     ylab = "Variance explained (%)")

# calculate UMAP and tSNE representation of the data
set.seed(42)
sce_merged <- runTSNE(sce_merged, dimred = "PCA")

set.seed(42)
sce_merged <- runUMAP(sce_merged, dimred = "PCA")
```

(32) for the computation of the variation in the genes (Box 9). We then use the *getTopHVGs()* function of the same package to extract the top 10% of highly variable genes (HVG) for each sample. These HVGs are then used as features for downstream steps. For the feature selection for each sample, see section "7 Feature Selection".

Data integration and merging of samples

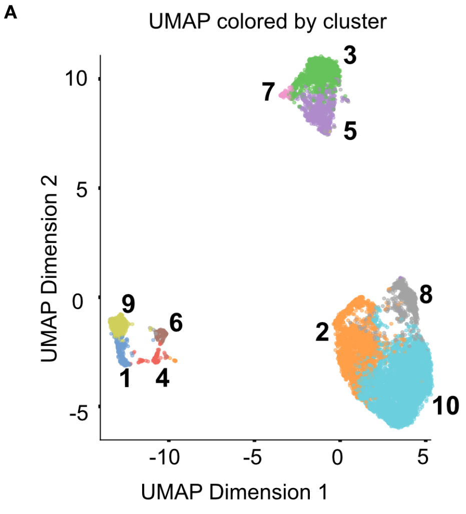
So far, we worked on each of our tissue samples individually as the presented steps yield more meaningful results if applied in a sample-specific manner. However, methods such as dimensionality



	Singlets	Doublets	Multiplet Rate (%)
spleen	3273	139	4,1
mLN	3069	149	4,6
iLN	2407	102	4,1
skin	952	37	3,7

FIGURE 9
Dimensionality reduction and clustering results. **(A)** Scree plot of the variance explained by each of the calculated principal components (PC). **(B)** Summary table of detected doublets in each of the tissues. **(C)** UMAP representation of the data colored by doublet status of each cell. **(D)** UMAP representation of the data colored by the different tissue types in the input data.

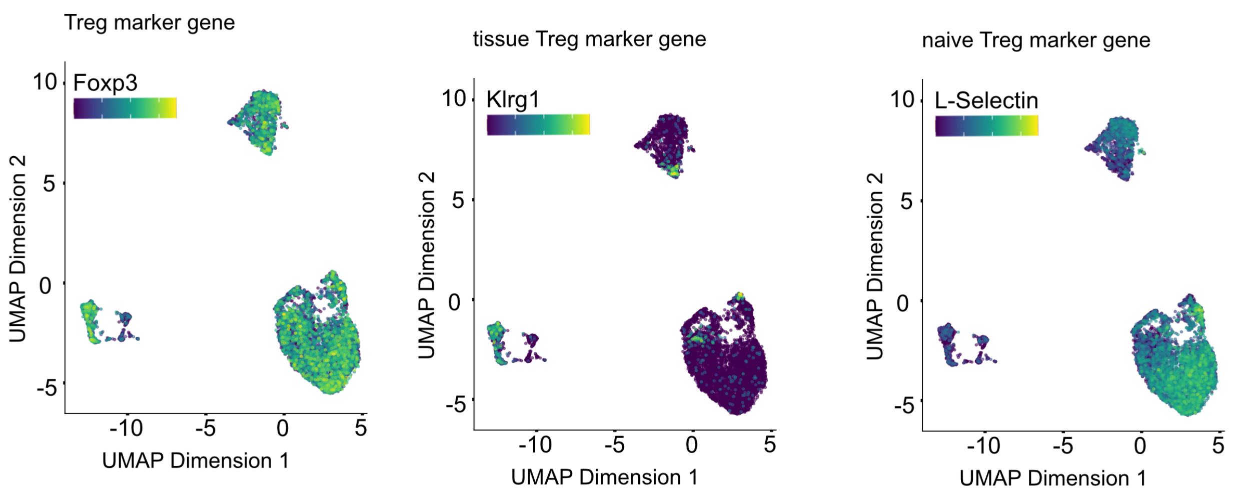
A



B

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
spleen	0	973	191	0	380
mLN	0	676	634	0	184
iLN	0	369	503	0	237
skin	256	7	1	109	14
total cell	256	2025	1329	109	815
	Cluster 6	**Cluster 7**	**Cluster 8**	**Cluster 9**	**Cluster 10**
spleen	0	8	313	1	1546
mLN	0	43	166	0	1514
iLN	0	37	180	0	1183
skin	161	0	15	426	0
total cell	161	88	674	427	4243

C



D

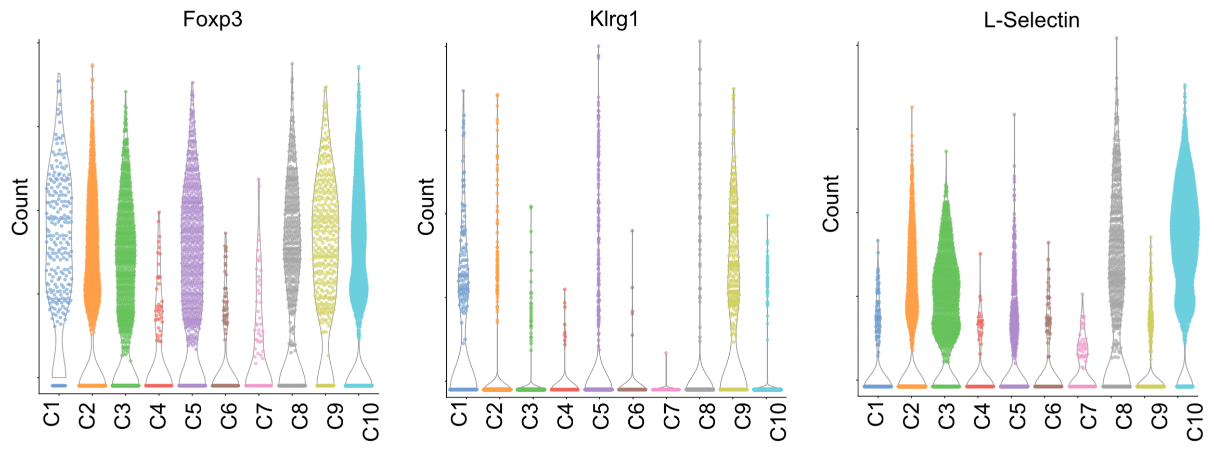


FIGURE 10
Clustering results and marker gene detection. **(A)** UMAP colored by the clusters found in the data. **(B)** Summary table of cluster composition. **(C)** UMAP representation of the data colored by the expression of different marker genes. **(D)** Violin plots showing the expression of different marker genes in the individual clusters.

reduction, clustering, marker gene detection and cell type annotation should be applied on the data set as a whole. This is why we will merge the individual *SingleCellExperiment* objects into one single object. For this, there are generally two approaches available: merging the samples without batch correction and merging after applying batch correction (33, 34). Usually, scRNA-seq data sets do not only contain different samples and tissues but also different batches. As previously discussed in this manuscript, there are technical differences between samples of different batches which can influence the results. We would like to filter out these technical differences to focus on biological variation between samples. In our workflow, we will present both approaches, batch-corrected and -uncorrected. In the uncorrected approach, we first apply the across sample normalization using the *multiBatchNorm()* function of the *batchelor* package (33). Afterwards, the metadata of the individual samples is synchronized before merging the objects into one *SingleCellExperiment* object (Box 10). In the batch-corrected

BOX 13 R code clustering.

```
# Calculate the clusters
snn.gr <- buildSNNGraph(sce_merged,
                        k = 25,
                        use.dimred = "PCA")
clusters <- igraph::cluster_walktrap(snn.gr)$membership

# See which tissue can be found in which cluster
tab <- table(Cluster = clusters, Batch = sce_merged$tissue)
tab

# Set the cluster as colLabels of the SingleCellExperiment
colLabels(sce_merged) <- factor(clusters)
plotTSNE(sce_merged, colour_by = "label")
plotUMAP(sce_merged, colour_by = "label")

# color tSNE by tissue
tsne <- plotTSNE(sce_merged, colour_by = "tissue")
# set custom colors, because with the original chosen colors of the method,
# the individual tissues are hard to distinguish.
tsne <- tsne + scale_fill_manual(
  values = c(
    skin = "tomato4",
    spleen = "darkblue",
    iLN = "springgreen4",
    mLN = "darkmagenta"
  ),
  aesthetics = "colour"
)
# plot the tSNE
tsne

# color UMAP by tissue
umap <- plotUMAP(sce_merged, colour_by = "tissue")
# set custom colors, because with the original chosen colors of the method,
# the individual tissues are hard to distinguish.
umap <- umap + scale_fill_manual(
  values = c(
    skin = "tomato4",
    spleen = "darkblue",
    iLN = "springgreen4",
    mLN = "darkmagenta"
  ),
  aesthetics = "colour"
)
# plot the UMAP
Umap

# plot tSNE and UMAP colored by doublet identification of cells
plotTSNE(sce_merged, colour_by = "scDblFinder.class")
plotUMAP(sce_merged, colour_by = "scDblFinder.class")
```

approach, we use the *RunHarmony* function of the *harmony* package (https://CRAN.R-project.org/package=harmony), after transforming our *SingleCellExperiment* object to a *Seurat* object (Box 11). Here, the data is already merged at read-in and processed as a whole, following the usual Seurat workflow (19). When inspecting the data further after merging, we realized that the batch correction was too stringent on our data and overcorrected for reasonable and important biological characteristics of the skin sample (Figures 8A, B). Hence, we will use the uncorrected, merged *SingleCellExperiment*. The code for uncorrected as well as batch-corrected merging of the data is shown in section "8 Data integration and merging of samples". To further showcase the effect of batch correction, we tried to integrate our data with a publicly available dataset presented in (15). From this dataset, we used the skin, spleen and LN sample to match the data presented in this paper. After downloading and reading the data as presented earlier in this paper, we tried to integrate and harmonize the two datasets using the *harmony* package. Figures 8C, D show the results of the integrated dataset. The code for these steps can be found in the notebook in section "8.3 Integration with publicly available data".

Dimensionality reduction using principal component analysis

In scRNA-seq analyses dimensionality reduction is used to achieve different objectives in the workflow. First, it greatly reduces the runtime of the following steps as calculations only need to be computed for a small number of dimensions compared to the large number of genes in the input data. Secondly, the procedure can reduce noise in the data by using average of genes rather than individual gene expression values. Lastly, it can also improve plotting of the data as 2/3-dimensional plots are usually easier to visualize and interpret as higher dimensional visualizations. A common approach for dimensionality reduction in scRNA-seq is Principal Component Analysis (PCA) (Box 12). As the first couple of principal components (PC) capture the largest

BOX 14 R code marker gene detection.

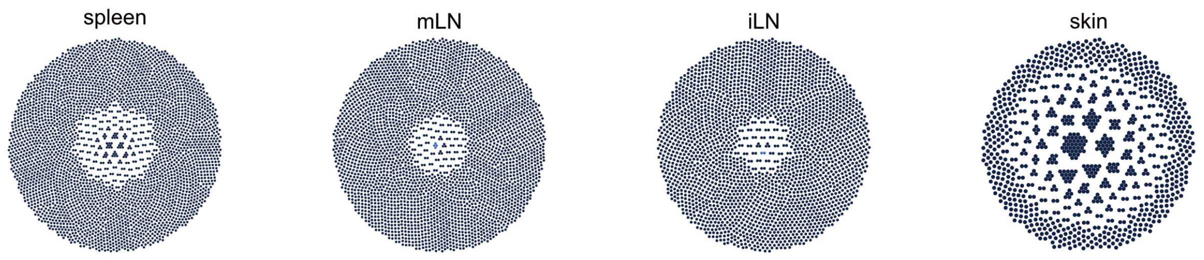
```
# score the marker genes between the individual pairs of clusters
markerGenes <- scoreMarkers(sce_merged, colLabels(sce_merged))

# extract marker genes for cluster 1, 2 and 9
markerGenes_cluster1 <- as.data.frame(markerGenes[[1]])
markerGenes_cluster2 <- as.data.frame(markerGenes[[2]])
markerGenes_cluster9 <- as.data.frame(markerGenes[[9]])

# generate a data table of the top 20 marker for each of the selected clusters
DT::datatable(head(markerGenes_cluster1[order(markerGenes_cluster1$mean.logFC.detected, decreasing = TRUE)], n =
20))
DT::datatable(head(markerGenes_cluster2[order(markerGenes_cluster2$mean.logFC.detected, decreasing = TRUE)], n =
20))
DT::datatable(head(markerGenes_cluster9[order(markerGenes_cluster9$mean.logFC.detected, decreasing = TRUE)], n =
20))

# plot the expression of the top 6 marker genes for each cluster in every of
# the clusters
plotExpression(
  sce_merged,
  features = head(rownames(markerGenes_cluster1)),
  x = "label",
  colour_by = "label"
)
plotExpression(
  sce_merged,
  features = head(rownames(markerGenes_cluster2)),
  x = "label",
  colour_by = "label"
)
plotExpression(
  sce_merged,
  features = head(rownames(markerGenes_cluster9)),
  x = "label",
  colour_by = "label"
)
```

amount of variance in the data, it can be assumed that these PC represent a considerable amount of biological variation of the data at hand. This way, the biological signal can be concentrated in a smaller number of PCs which can help with interpretation and visualization of the high-dimensional scRNA-seq data. In our analysis we use the *runPCA()* function from the *BiocSingular* package (8), https://doi.org/10.18129/B9.bioc.BiocSingular). The function calculates the principal components for the given data. In the shown code, we calculate the PCs based on the HVGs we determined previously, ensuring a reduced computation time while

A

	iLN	mLN	skin	spleen
1x TRA	1686	2127	774	2238
1x TRB	2281	2936	749	3101
2 x TRA	254	336	99	377
2x TRB	193	230	150	242
1x TRA + 1x TRB	1490	1893	556	1977
1x TRA + 2x TRB	161	183	128	192
1x TRA + 3x TRB	0	0	26	0
2x TRA + 1x TRB	222	289	77	327
3x TRA + 1x TRB	0	0	5	2

B



C

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6	Cluster 7	Cluster 8	Cluster 9	Cluster 10
1x TRA	193	1324	990	77	641	122	66	467	354	2591
1x TRB	195	1839	1250	89	764	134	82	567	299	3848
2 x TRA	33	224	157	8	96	16	14	78	39	401
2x TRB	36	135	72	10	43	22	5	96	79	317
1x TRA + 1x TRB	139	1168	921	59	598	97	61	383	237	2253
1x TRA + 2x TRB	29	105	62	8	35	20	4	73	68	260
1x TRA + 3x TRB	6	0	0	0	0	0	0	0	20	0
2x TRA + 1x TRB	26	194	147	6	88	14	13	55	28	344
3x TRA + 1x TRB	1	2	0	0	0	0	0	0	4	0

D

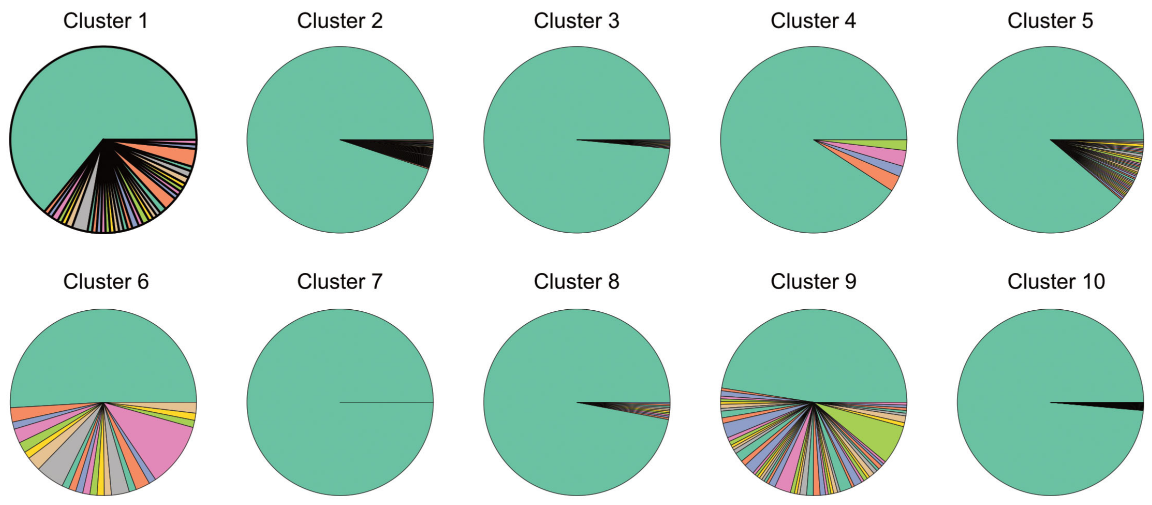


FIGURE 11
TCR diversity in tissue CD4+ T Cells of an individual animal. **(A)** Table of number of cells with different combinations of TCR chains in the individual tissues. **(B)** Visualization of the TCR diversity in the individual tissues. **(C)** Table of number of cells with different combinations of TCR chains in the individual clusters. **(D)** Pie charts visualizing the clonality of TCR in the clusters. TCR chains found only once per cluster were grouped and colored in green, while the remaining portion of the pie chart visualizes TCR chains with multiple occurrences.

at the same time reducing the high-dimensional noise. A critical choice in the context of PCA is the choice of the number of top PCs used for downstream analyses. A helpful visualization to decide on this number is shown in Figure 9A. The figure plots the PCs against the percentage of variance each PC explains/captures. We see that there is a notable drop in the amount of variance explained by the PCs after the 25th PC. Hence, we decided to use the first 25 PCs for downstream analyses as these capture most of the variance of our data at hand. For the PCA analysis see section "9 Dimensionality reduction using Principal Component Analysis". Once dimensionality reduction is applied, we can also calculate a t-SNE or UMAP representation of our data (35, 36). Both visualization techniques are suitable for high-dimensional datasets such as scRNA-seq data. The t-stochastic neighborhood embedding (t-SNE) aims to find a low-dimensionality representation of the data that preserves the distances between points from the high-dimensionality space. Uniform manifold approximation and projection (UMAP, (36)) is another non-linear visualization technique for high-dimensionality data, similar to t-SNE. It should be mentioned that both methods are non-deterministic, meaning that they yield slightly different results each time the function is run on the data. We can prevent this by using the R function *set.seed()* using the same seed each time. In Figures 9C, D and Figures 10A–C we show the UMAP representation of our data colored by different properties of the data. We also calculated the t-SNE representations of our data colored by the same properties, the results are shown in the notebook accompanying this manuscript. For the plotting of the UMAP and t-SNE see Box 13.

Clustering

Clustering is adopted for scRNA-seq data to summarize the high-dimensional, complex data by dividing the cells into individual groups based on gene expression profiles. This greatly eases interpretation and exploration of the data, as the cells are then represented as discrete groups rather than the complex, high-dimensional space that is the origin of the data. In its nature, clustering is an explorative step of the analysis, possibly run in different iterations. In our workflow, we use the *buildSNNGraph()* function of the *scran* package (32) followed by the *cluster_walktrap ()* function of the *igraph* package (Box 13). This function implements a graph-based clustering approach. Other approaches are for example Louvain clustering (37), vector quantization like k-means or hierarchical clustering (8). Once clusters have been calculated, they can be visualized as UMAP or t-SNE. In Figure 10A, we color the UMAP by the detected clusters. Together with Figure 9B, this shows that the skin cells form individual clusters which are clearly separated from the rest of the data. The remaining tissue types intermingle in their clusters with the separation being driven by factors other than tissue type. Figure 10B also highlights in a table that the skin forms exclusive

BOX 15 R code TCR repertoire analysis.

```
chains_frequency <- table(sce_merged$cdr3s_aa)
chains_duplicated <- chains_frequency > 1

is_duplicated_chains <- sapply(sce_merged$cdr3s_aa, function(x) chains_duplicated[[x]])

which_chain <- sapply(sce_merged$cdr3s_aa, function(x) if(chains_duplicated[[x]]){
  x
}else{NA})

sce_merged$duplicated_chains <- is_duplicated_chains
sce_merged$which_duplicated_chain <- which_chain

clonotype_frequency <- table(sce_merged$clonotype)
clonotype_duplicated <- clonotype_frequency > 1

is_duplicated_clonotype <- sapply(sce_merged$clonotype, function(x) clonotype_duplicated[[x]])

which_clonotype <- sapply(sce_merged$clonotype, function(x) if(clonotype_duplicated[[x]]){
  x
}else{NA})

sce_merged$duplicated_clonotype <- is_duplicated_clonotype
sce_merged$which_duplicated_clonotype <- which_clonotype
plotUMAP(sce_merged, color_by = "duplicated_clonotype", order_by = "duplicated_clonotype")
```

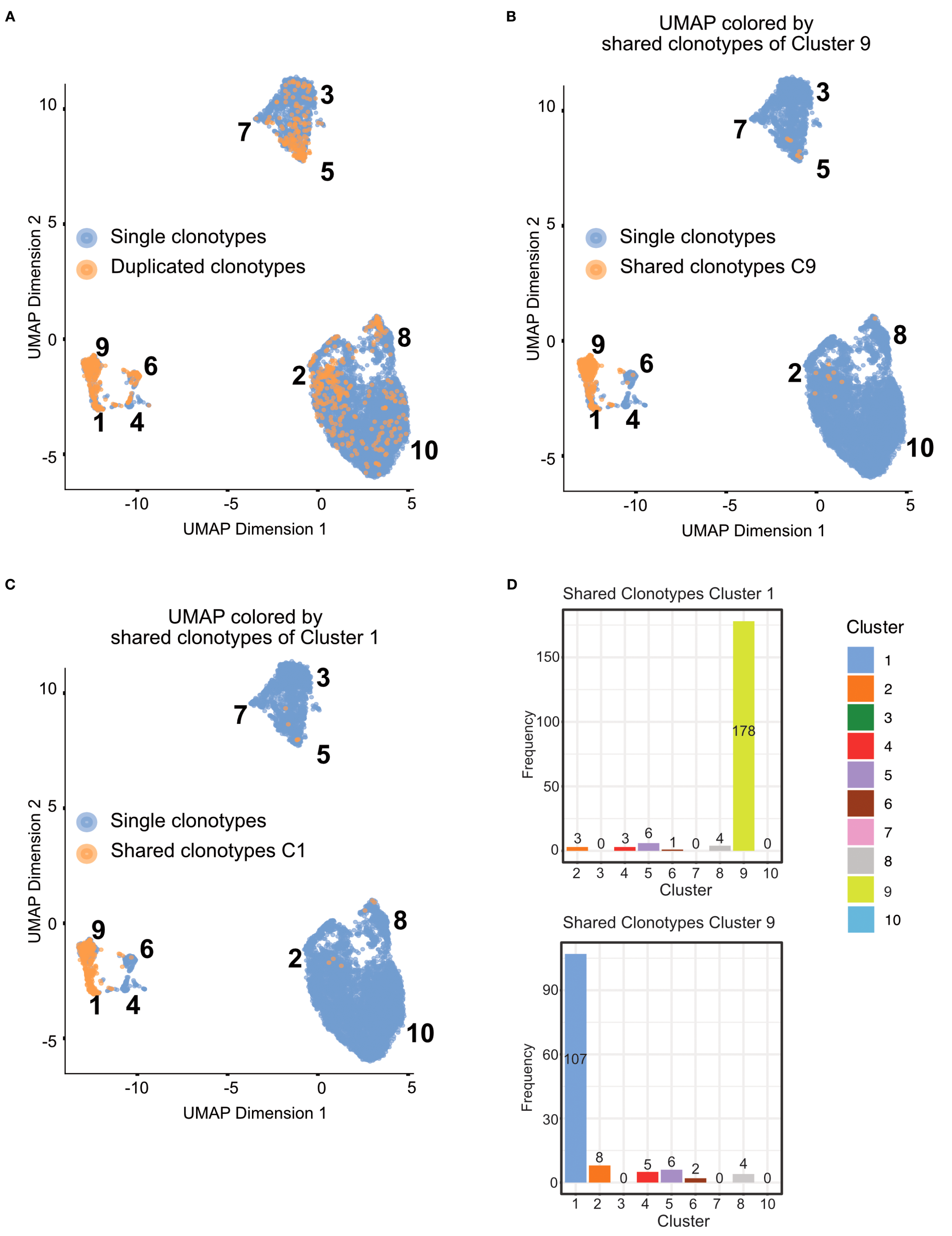


FIGURE 12
UMAP plotted by shared clonotypes. **(A)** UMAP representation of the data plotted by whether a clonotype is shared between cells of different tissues. The overlaying numbers represent the clusters of the cells shown in Figure 9A. **(B)** UMAP representation of the data colored by TCR chains shared with cells in cluster 9. **(C)** UMAP representation of the data colored by TCR chains shared with cells in cluster 1. **(D)** Barplot of the number of shared clonotypes of cluster 1 (upper) and 9 (lower) with the other remaining clusters. .

BOX 16 R code Clonotypes shared between clusters (identical for both, showcase cluster 1).

```
clonotype_cluster1 <- sce_merged[, colLabels(sce_merged) == "1"]$clonotype
clono_cluster1_other_clusters <- sce_merged$clonotype %in% clonotype_cluster1
sce_merged$clonotype_cluster1_shared <- clono_cluster1_other_clusters

# overlay shared clonotypes on the UMAP
plotUMAP(sce_merged, color_by = "clonotype_cluster1_shared", order_by = "clonotype_cluster1_shared")

# plot shared clonotypes as barplot
data <- as.data.frame(table(sce_merged$clonotype_cluster1_shared, colLabels(sce_merged)))
data <- data[data$Var1 == TRUE],
data <- data[!data$Var2 == 1, c(2:3)]
colnames(data) <- c("Cluster", "Frequency")

ggplot(data, aes(x = Cluster, y = Frequency, fill = Cluster, label = Frequency)) +
  geom_bar(stat = "identity") +
  geom_text(size = 5, position = position_stack(vjust = 0.5)) +
  theme_bw()
```

clusters with only incidental, individual cells being part of tissue-mixed clusters. For more details on clustering, see section "10 Clustering" in the notebook.

Marker gene detection

After clustering the data in the previous workflow step, the interpretation of the data can be further facilitated by characterizing marker genes (38). Marker genes are genes that drive the separation between the individual clusters, and the identification of such genes helps identifying possible functions and biological meaning of the individual clusters. The general strategy to determine marker genes of individual clusters is a pairwise comparison of all the clusters to calculate scores which quantify the differences in gene expression. In our analysis we use the *scoreMarkers()* function from the *scran* package (32) for this analysis step (Box 14). The function compares each of the clusters in pairs. Pairwise comparisons provide the advantage of providing more information about the markers which is beneficial to the interpretation. Also, in contrast to the approach of comparing one cluster against the average of all remaining cells, pairwise comparisons are more robust against population composition and uneven subpopulation sizes. The *scoreMarkers()* function calculates different effect size summaries to quantify the difference in gene expression between the clusters. The one we use in our workflow, visualized in Figure 10D, is the log fold-change, where we use the genes with the highest log fold-change between clusters as our marker genes for each cluster. There are also other metrics available in the function. For users interested in those, we refer to the documentation of the *scoreMarkers()* function. For the marker gene detection, see section "11 Marker gene det

TCR repertoire diversity

TCR V(D)J sequencing coupled with single-cell RNA sequencing enables profiling of paired TCRα and TCRβ chains at single-cell resolution with coupled global gene expression in the same cell (39, 40). This analysis makes it possible to characterize T-cell clonal expansion in steady state and in disease, as well as tracking shared T-cell clonotypes between different tissues. In our analysis, we wanted to use this information to evaluate if there are shared TCR chains between different tissues as well as different clusters. We also wanted to evaluate for each tissue and cluster which chains were only found once compared to chains found multiple times.

In Figure 11, we displayed several different statistics of the TCR chains in our data. Figure 11A shows a summary table of the occurrences of different combinations of TCR chains in the different tissues. We can see that most cells in our data have at least one TCRβ chain, followed by cells with at least one TCRα chain and cells with one TCRα and one TCRβ. Figure 11B visualizes the clonality of the TCR chains in the individual tissues. Here, we can see that most chains of the lymphoid organs only occur once, while some chains can be found multiple times. The highest TCR diversity can be found in the skin. Figure 11C shows a similar summary table as part Figure 11A, this time separated into the individual clusters calculated for our data set. Here, we can observe similar patterns as for the distribution of TCRs in the individual tissues. Lastly, Figure 11D shows pie charts of the clonality of the TCR in the individual clusters, in which we grouped all the TCR which occurred only once in the clusters. These are shown in green, while the remaining proportion of each pie chart is composed of TCR which have multiple occurrences in a cluster. Here, we can see that nearly each cluster has TCR chains which can be found more than once except for cluster 7. In a second step, we also wanted to analyze if there are TCR chains which were shared by cells of different tissues (Box 15). In Figure 12A, we plotted the UMAP representation of our data colored by whether TCR chains are shared by cells of different tissue origin. We can see that there are a lot of TCR chains shared between different tissues. In Figures 12B, C, we colored the UMAP by the occurrence of TCR chains found in cells of either cluster 9 or cluster 1 (Box 16). In Figure 12B, we can

BOX 17 R code cell type annotation using a reference data set.

```
ref_annot_immgen <- ImmGenData()
# Calculate cell type annotations
celltype_immgen_main <- SingleR(test = sce_merged,
                                ref = ref_annot_immgen,
                                labels = ref_annot_immgen$label.main,
                                BPPARAM = BiocParallel::MulticoreParam(6))
celltype_immgen_fine <- SingleR(test = sce_merged,
                                ref = ref_annot_immgen,
                                labels = ref_annot_immgen$label.fine,
                                BPPARAM = BiocParallel::MulticoreParam(6))

# summarize cell type annotation results
table(celltype_immgen_main$labels)
table(celltype_immgen_fine$labels)

# save results as meta data in the SingleCellExperiment object
sce_merged$celltype_immgen_main <- celltype_immgen_main$labels
sce_merged$celltype_immgen_fine <- celltype_immgen_fine$labels

# plot UMAP and tSNE representation colored by the assigned cell types from the
# main labels
plotTSNE(sce_merged,
         colour_by = "celltype_immgen_main",
         text_by = "celltype_immgen_main")
plotUMAP(sce_merged,
         colour_by = "celltype_immgen_main",
         text_by = "celltype_immgen_main")

# plot UMAP and tSNE representation colored by the assigned cell types from the
# fine labels
plotTSNE(sce_merged,
colour_by = "celltype_immgen_fine",
text_by = "celltype_immgen_fine")
plotUMAP(sce_merged,
colour_by = "celltype_immgen_fine",
text_by = "celltype_immgen_fine")

# plot a heatmap of the degree of matching of the individual cells to the
# available cell type labels in the reference data
plotScoreHeatmap(celltype_immgen_main)

# plot a heatmap of cluster to cell types, showing which cell type can found in
# the individual clusters
tab <- table(Assigned = celltype_immgen_main$pruned.labels,
             Cluster = colLabels(sce_merged))
# Adding a pseudo-count of 10 to avoid strong color jumps with just 1 cell.
pheatmap(log2(tab + 10),
         color = colorRampPalette(c("white", "darkblue"))(101))
```

see that there are a lot of TCR chains shared between cluster 9 and 1 which are both composed of exclusively skin cells. However, there are also TCR shared with cells in cluster 5 and 2. Figure 12C shows that TCR chains of cluster 1 are also shared with cells in cluster 2, 5 and 8. Figure 12D shows the shared clonotypes of Cluster 1 and 9 between the other clusters. For the marker gene detection, see section "12 TCR repertoire diversity" in the notebook.

Cell type annotation

Cell type annotation is arguably one of the most critical yet challenging step of a scRNA-seq analysis (41–43), as the concept of a cell type itself and the distinction of different cell types is a highly discussed topic (44, 45) Transcriptomic profiles of single cells still make it possible to assign cell types to the individual cells of a scRNA-seq data set (46). Usually, this is done using an appropriate reference data set with each cell being assigned a cell type based on the most similar cell in the reference data. In our workflow, we will present the methods of *SingleR* for cell type annotation (47) (Box 17). Technically, any published and carefully labeled bulk or single-cell RNA-seq data set can be used as reference data set. However, the quality of the resulting assigned cell types heavily depends on the compatibility of the data at hand and the reference data. Also, the reference data should ideally contain a variety of cells which comprises all the cell types

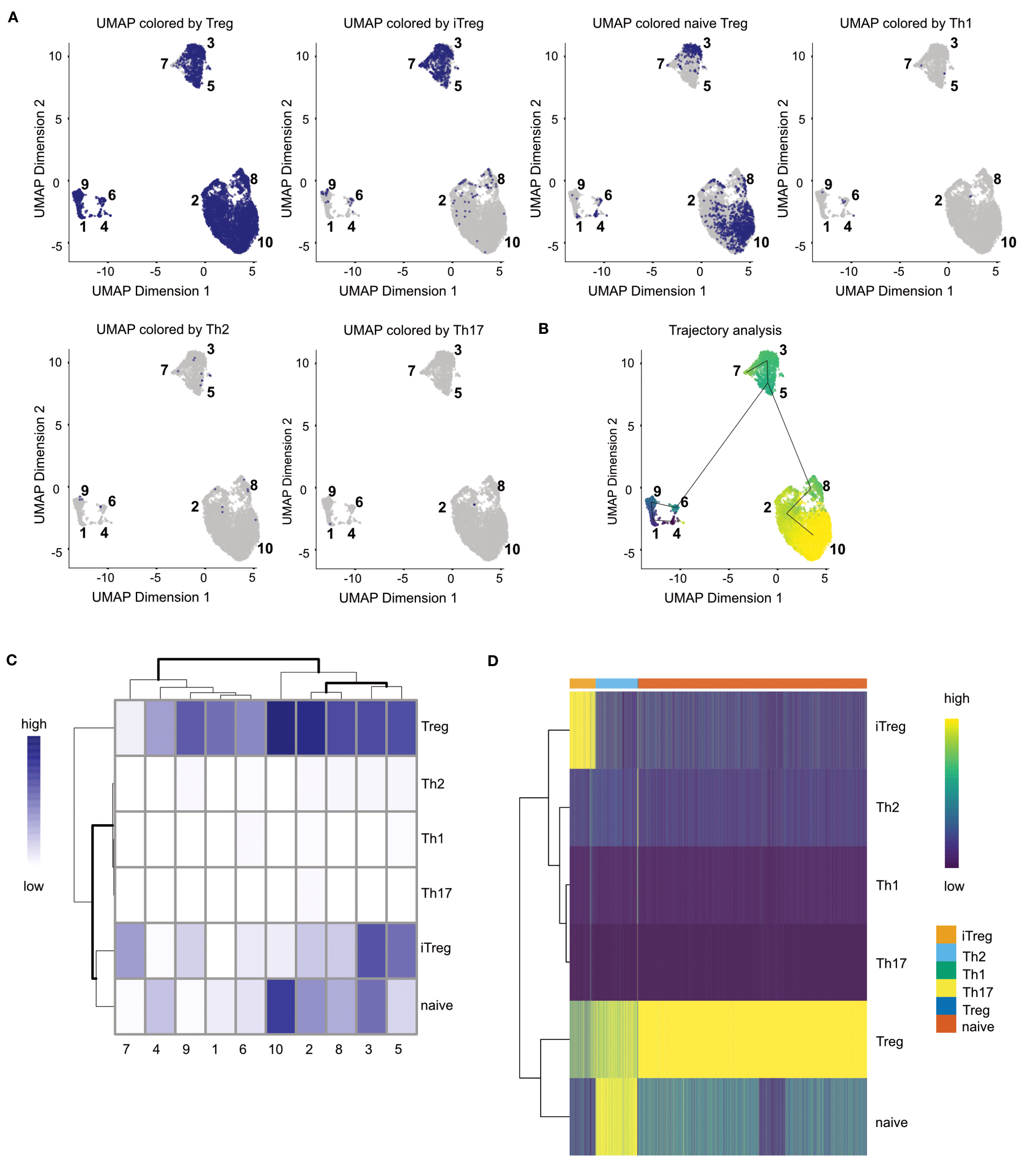


FIGURE 13
Cell type annotation results. **(A)** UMAP colored by assigned cell types for each cell. **(B)** Trajectory analysis of the data plotted on the UMAP **(C)** Heatmap of cell type distribution across clusters. **(D)** Heatmap of matching similarity of each cell to the different cell types in the reference.

BOX 18 R code cell type annotation using custom markers.

```
# set up a list of known marker genes for certain cell types, e.g. Treg cells
treg <- c("Foxp3", "Il2")
# p Treg cells
p_treg <- c("Rorc", "Gata3")
# t Treg cellst_treg <- c("Ikzf2")
# Tissue Treg
tissue_treg <- c("Batf", "Klrg1", "Areg", "Ccr8", "Il10")
# Th1 cells
th1 <- c("Tbx21", "Ifng")
# Naive T-cells
naive <- c("Ccr7", "Sell", "Irf4")

# repeat these two steps for all markers of interest
plotExpression(sce_merged, features = "Foxp3",
    x = "label", colour_by = "label")
plotUMAP(sce_merged, color_by = "Foxp3", order_by = "Foxp3")
```

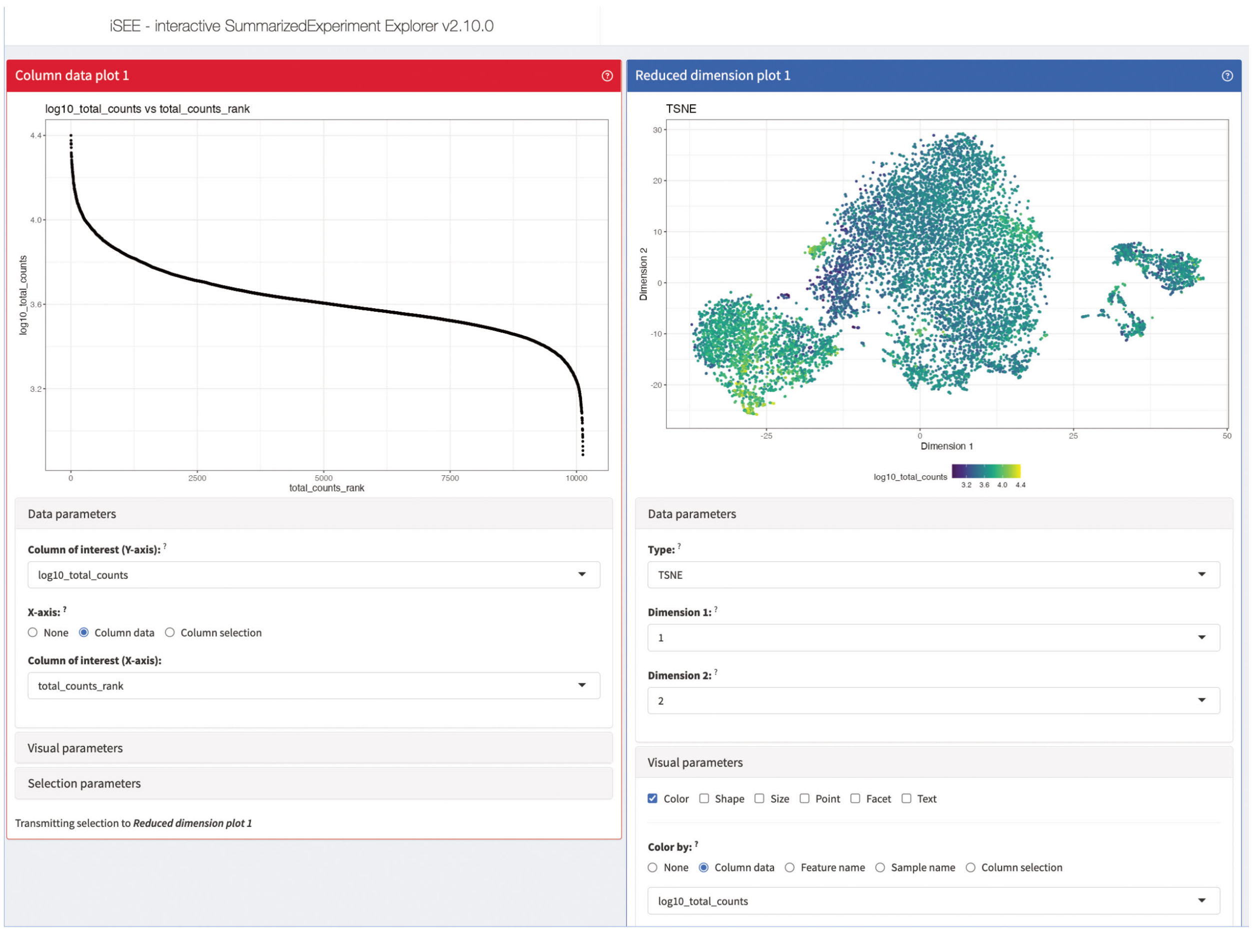
BOX 19 R code trajectory analysis.

```
by.cluster <- aggregateAcrossCells(sce_merged,
                                   ids = colLabels(sce_merged))
centroids <- reducedDim(by.cluster, "PCA")

# Set clusters = NULL as we have already aggregated above.
mst <- createClusterMST(centroids, clusters = NULL)
mst
line.data <- reportEdges(by.cluster,
                         mst = mst,
                         clusters = NULL,
                         use.dimred = "UMAP")

plotUMAP(sce_merged, colour_by = "label") +
     geom_line(data = line.data,
               mapping = aes(x = dim1,
                             y = dim2,
                             group = edge))

map.tscan <- mapCellsToEdges(sce_merged,
                             mst = mst,
                             use.dimred = "PCA")
tscan.pseudo <- orderCells(map.tscan, mst)
head(tscan.pseudo)

common.pseudo <- averagePseudotime(tscan.pseudo)
plotUMAP(sce_merged, colour_by = I(common.pseudo),
        text_by = "label", text_colour = "red") +
   geom_line(data = line.data, mapping = aes(x = dim1, y = dim2, group = edge))
```

expected in the scRNA-seq data at hand. A large variety of suitable reference data sets can be found in the R package *celldex* (47). In our workflow, we use an unpublished, in-house reference data set consisting of different T-cell subpopulations for cell type annotation and the visualizations shown in Figure 13. However, we also present in the HTML report how to use reference data sets from the *celldex* package (Box 17). After a suitable reference data set has been selected, the cell types can simply be annotated by calling the *SingleR()* function with the input data and the reference data as shown in our workflow. The results can be plotted in a heatmap as scores of the different labels to cells. An example can be seen in Figure 13C. Ideally, each cell should have one label with a high score compared to all other labels. Figure 13D we plot the composition of the individual clusters with the available cell types. We see that most clusters mainly consist of one to two cell types, with all clusters including Tregs. In Figure 13 we plot the same results as an overlay over the UMAP representation of our data. Here as well we can see a nice distribution and clustering of the individual cell types, with all clusters having Treg cells. As mentioned above, another approach to cell type annotation is the use of marker genes (Box 18). In our workflow, we also did cell type annotation based on known marker genes for specific T cell subpopulations. In Figure 10C some of the markers are showcased and we can see the expression of individual selected marker genes in the UMAP representation of the data. Figure 10C shows violin expression plots of the marker genes in the individual samples. Combined with automated reference-based methods, this can support the interpretation and identification of cell types of the data at hand. For the cell type annotation see also section "12 Cell type annotation using reference data and custom markers" in the notebook.

Trajectory analysis

A large variety of biological processes can be represented as a continuum of biological changes in the cellular state. This is especially true of cell type differentiation which can for example be observed in different T-cell subpopulations. In our high dimensional scRNA-seq data, we want to characterize this process of differentiation by finding a trajectory. Associated with a trajectory is the pseudotime, which is the position of each cell along the trajectory and could for example represent the state of differentiation of a cell along a continuous process. Pseudotime helps us answer questions about the global population structure of our data. In our workflow, we use a cluster-based approach for identifying the trajectory in the data (Box 19). The *TSCAN* (48) algorithm implemented in the corresponding package first computes cluster centroids of the determined clusters before forming a minimum spanning tree (MST). Figure 12B shows the results of our trajectory analysis. The pseudotime ranges from dark to light colors, meaning cells with a dark blue color have an early pseudotime than yellow-colored cells. In the case of the presented data, a trajectory



iSEE - interactive SummarizedExperiment Explorer v2.10.0

Column data plot 1

Data parameters

Column of interest (Y-axis): log10_total_counts

X-axis: None [✓ Column data] Column selection

Column of interest (X-axis): total_counts_rank

Visual parameters

Selection parameters

Transmitting selection to *Reduced dimension plot 1*

Reduced dimension plot 1

Data parameters

Type: TSNE

Dimension 1: 1

Dimension 2: 2

Visual parameters

[✓ Color] Shape Size Point Facet Text

Color by: None [✓ Column data] Feature name Sample name Column selection

log10_total_counts

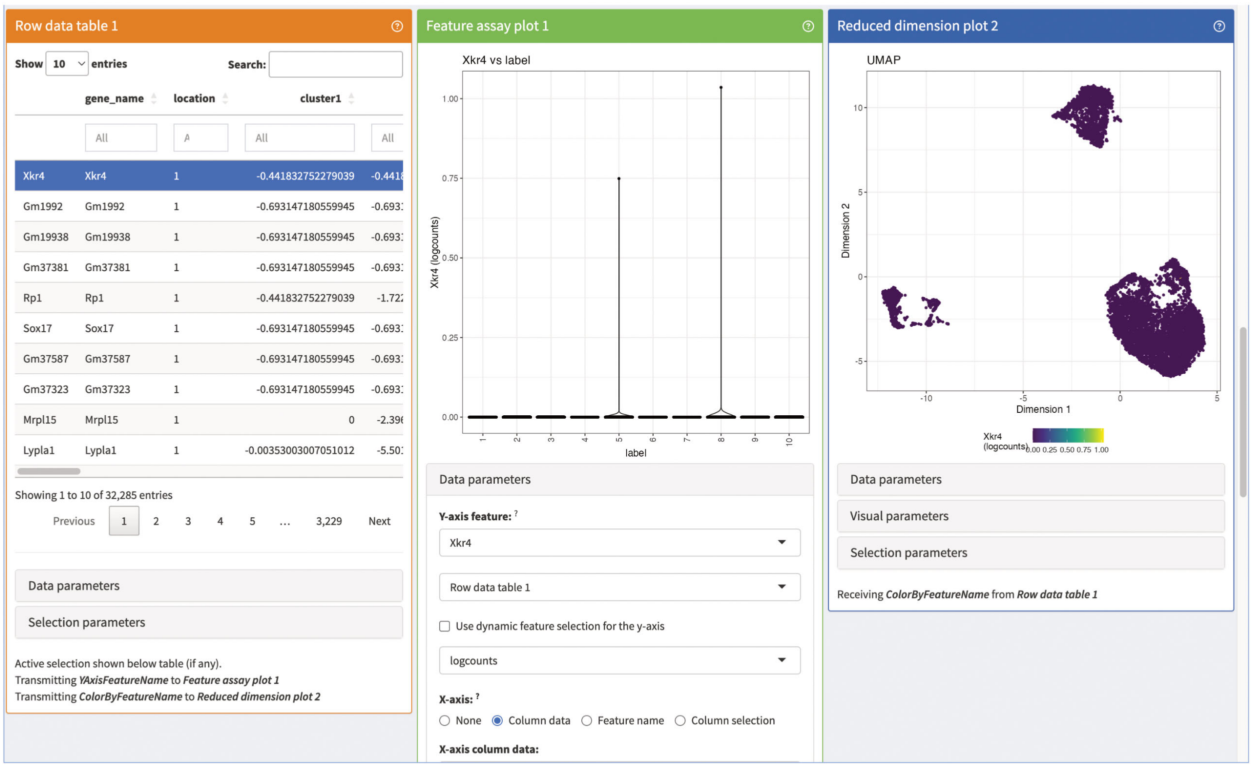
FIGURE 14
Quality control panels of our iSEE instance. The column data plot 1 on the left plots the library size of each cell in decreasing order. The reduced dimension plot 1 on the right shows the t-SNE presentation of our data colored by the log-normalized library size of each cell. Dark cells have a small library size, while yellow cells have a large library size.

analysis might not yield too many additional insights on the data because of the overall composition of the data. However, in projects and datasets where continuous processes are under investigation, a trajectory analysis might yield additional insight of the data. For the trajectory analysis, see section "13 Trajectory Analysis" in the notebook.

Methods – interactive data exploration using *iSEE*

For most data analysis workflows, one of the most crucial and time-consuming steps is the data exploration, usually accompanied by a lot of different data visualizations (49). This is also the case for scRNA-seq where the data usually is not only complex, but also large in size. Reiterating data exploration and visualizations steps can be beneficial to the data analysis and can help to compact and facilitate data interpretation. An excellent tool for interactive and iterative data exploration and visualization for scRNA-seq data is *iSEE* (50). *iSEE* provides a flexible framework which is compatible with a lot of different data types and can be dynamically adapted to the respective data set at hand. Each instance of *iSEE* can be customized to the individual data set by selecting the most suitable visualization and exploration techniques in form of different panels provided by *iSEE* (Figures 14–16). As an input to *iSEE*, users have to provide a *SummarizedExperiment* object (*SingleCellExperiment* being a derivative class, with features tailored to single cell assays). This format is commonly returned by most packages in the *Bioconductor* ecosystem. In our workflow, the data is also already saved as a *SingleCellExperiment* object from the beginning, so the data presented here can easily and directly be explored with *iSEE*. In our workflow, we will present different panels of *iSEE* to demonstrate the possibilities of the application. For this, we present a customized panel layout that can be achieved using the code shown in "14 Interactive data exploration using iSEE". The first two panels we add to our *iSEE* instance are quality control-related and plot the library size as well as a t-SNE of the log-normalized library size (Figure 14). The plots help to identify clusters of low-quality cells and can also be used to detect quality control or normalization errors.

Next, we add panels to visualize the marker genes of individual clusters (Figure 15). The panels consist of a summarization table, an expression plot of individual marker genes in the clusters as well as an UMAP of the expression of selected marker genes. All three panels are interactive and connected, so that users can evaluate different marker genes. Lastly, we present summaries on the counts of individual genes in the panels shown in Figure 16. The panels summarize the expression of the genes in the data as a



Row data table 1

Show 10 entries Search:

	gene_name	location	cluster1	
	All	A	All	All
Xkr4	Xkr4	1	-0.441832752279039	-0.4418
Gm1992	Gm1992	1	-0.693147180559945	-0.693
Gm19938	Gm19938	1	-0.693147180559945	-0.693
Gm37381	Gm37381	1	-0.693147180559945	-0.693
Rp1	Rp1	1	-0.441832752279039	-1.72
Sox17	Sox17	1	-0.693147180559945	-0.693
Gm37587	Gm37587	1	-0.693147180559945	-0.693
Gm37323	Gm37323	1	-0.693147180559945	-0.693
Mrpl15	Mrpl15	1	0	-2.39
Lypla1	Lypla1	1	-0.00353003007051012	-5.50

Showing 1 to 10 of 32,285 entries

Previous 1 2 3 4 5 … 3,229 Next

Data parameters

Selection parameters

Active selection shown below table (if any).
Transmitting *YAxisFeatureName* to *Feature assay plot 1*
Transmitting *ColorByFeatureName* to *Reduced dimension plot 2*

Feature assay plot 1

Data parameters

Y-axis feature: Xkr4

Row data table 1

☐ Use dynamic feature selection for the y-axis

logcounts

X-axis: ○ None [✓ Column data] ○ Feature name ○ Column selection

X-axis column data:

Reduced dimension plot 2

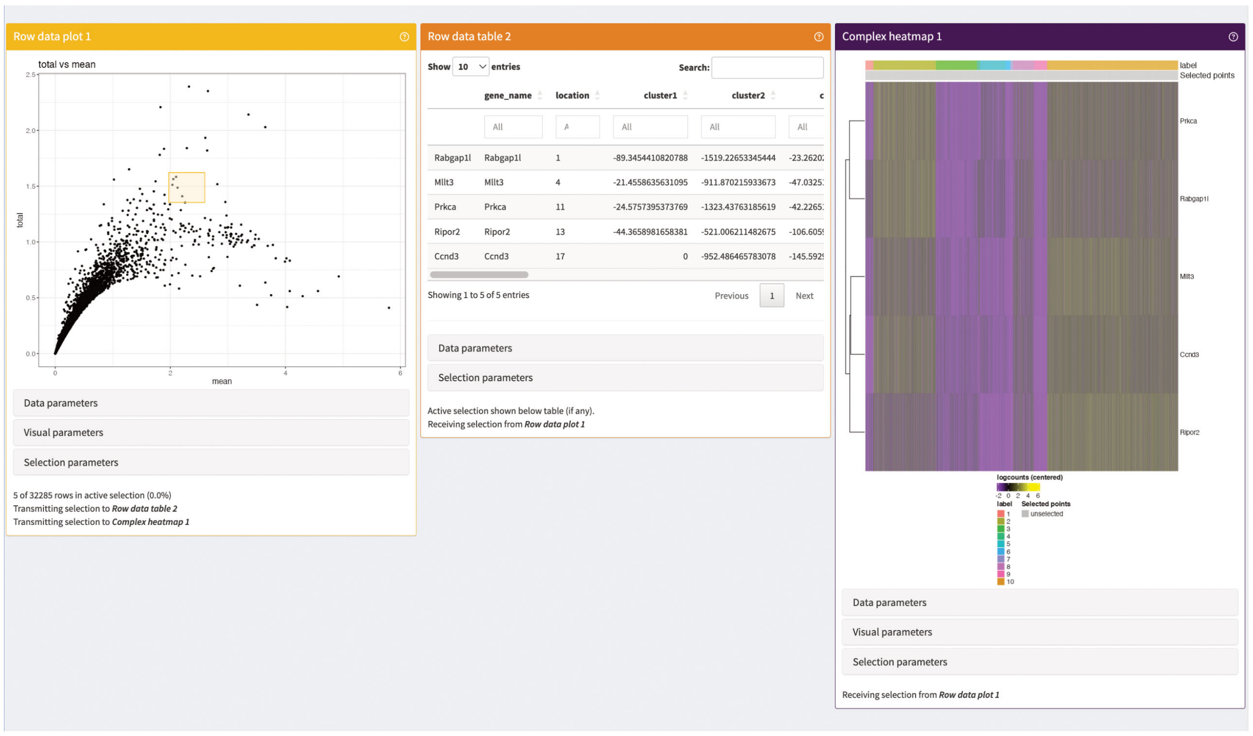
Data parameters

Visual parameters

Selection parameters

Receiving *ColorByFeatureName* from *Row data table 1*

FIGURE 15
Marker gene panels of our iSEE instance. The row data Table 1 contains a table of the different marker genes of the individual clusters. The feature assay plot 1 shows a violin plot of the expression of the selected marker gene of the row data Table 1. Lastly, the reduced dimensions plot 2 shows the UMAP representation of our data colored by the expression of the selected marker in the row data Table 1.



5 of 32285 rows in active selection (0.0%)
Transmitting selection to *Row data table 2*
Transmitting selection to *Complex heatmap 1*

Show 10 entries Search:

	gene_name	location	cluster1	cluster2	c
	All	A	All	All	All
Rabgap1l	Rabgap1l	1	-89.3454410820788	-1519.22653345444	-23.2620
Mllt3	Mllt3	4	-21.4558635631095	-911.870215933673	-47.0325
Prkca	Prkca	11	-24.5757395373769	-1323.43763185619	-42.2265
Ripor2	Ripor2	13	-44.3658981658381	-521.006211482675	-106.605
Ccnd3	Ccnd3	17	0	-952.486465783078	-145.592

Showing 1 to 5 of 5 entries Previous 1 Next

Active selection shown below table (if any).
Receiving selection from *Row data plot 1*

Receiving selection from *Row data plot 1*

FIGURE 16
Gene summary panels of our iSEE instance. The Row data plot visualizes the mean-variances trend of the genes in our data. The Row data table is an interactive table displaying statistics on the genes selected in Row data plot (yellow square). The Complex heatmap shows a heatmap of the expression of the selected genes (yellow square) in the individual cells of the data.

table as well as an expression heatmap and can help explore different genes of interest in the data. As shown here, *iSEE* provides several different summary statistics and visualizations for the data. Besides the showcased panels here, there is a variety of other different panels available. This can greatly benefit the data analysis by being an interactive and reproducible way for data exploration and visualization.

Data availability statement

Publicly available datasets analyzed in this manuscript can be found at the Gene Expression Omnibus (GEO) database, with accession code GSE240041. We included previously published datasets, also available at GEO under the accession code GSE130879. The code generated throughout this manuscript is available at the GitHub repository https://github.com/imbeimainz/scRNAseq_scTCRseq_TissueTcells. The rendered HTML notebooks accompanying the analyses presented can be found on Zenodo (https://zenodo.org/record/8338200).

Ethics statement

Murine organs and tissues were harvested according to the regulations of the German Animal Welfare Act (§4 Tierschutzgesetz).

Author contributions

Conceptualization: MD and FM; Methodology: ASN, SSH, FM, and MD; Software: ASN, SSH, KLB, MV, MD, and FM; Investigation and Resources: ASN, SSH, KLB, MD, and FM; Writing – Original Draft: ASN, SSH, MD, and FM; Writing – Review and Editing: ASN, SSH, MD, and FM; Visualization: ASN and SSH; Supervision: MD and FM; Project Administration: MD and FM; Funding Acquisition: MD and FM. All authors contributed to the article and approved the submitted version.

Funding

This work was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Projektnummer 318346496 – SFB1292/2 TP19N (to FM and MD) and Projektnummer 490846870 – TRR355/1 TPA01 and TPZ02 (to MD).

Acknowledgments

We thank the FZI flow cytometry core facility, the FZI NGS facility, and the University Medical Center Mainz animal facility for technical support. This work has also been supported by the computing infrastructure provided by the Core Facility Bioinformatics at the University Medical Center Mainz.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Supplementary material

The Supplementary Material for this article can be found online at: https://github.com/imbeimainz/scRNAseq_scTCRseq_TissueTcells.

References

1. Shalek AK, Satija R, Adiconis X, Gertner RS, Gaublomme JT, Raychowdhury R, et al. Single-cell transcriptomics reveals bimodality in expression and splicing in immune cells. *Nature* (2013) 498:236–40. doi: 10.1038/nature12172

2. Satija R, Shalek AK. Heterogeneity in immune responses: from populations to single cells. *Trends Immunol* (2014) 35:219–29. doi: 10.1016/j.it.2014.03.004

3. Papalexi E, Satija R. Single-cell RNA sequencing to explore immune cell heterogeneity. *Nat Rev Immunol* (2018) 18:35–45. doi: 10.1038/nri.2017.76

4. Efremova M, Vento-Tormo R, Park JE, Teichmann SA, James KR. Immunology in the era of single-cell technologies. *Annu Rev Immunol* (2020) 38:727–57. doi: 10.1146/annurev-immunol-090419-020340

5. Nieto P, Elosua-Bayes M, Trincado JL, Marchese D, Massoni-Badosa R, Salvany M, et al. A single-cell tumor immune atlas for precision oncology. *Genome Res* (2021) 31:1913–26. doi: 10.1101/gr.273300.120

6. Rodriguez-Ubreva J, Arutyunyan A, Bonder MJ, Del Pino-Molina L, Clark SJ, de la Calle-Fabregat C, et al. Single-cell Atlas of common variable immunodeficiency shows germinal center-associated epigenetic dysregulation in B-cell responses. *Nat Commun* (2022) 13:1779. doi: 10.1038/s41467-022-29450-x

7. Luecken MD, Theis FJ. Current best practices in single-cell RNA-seq analysis: a tutorial. *Mol Syst Biol* (2019) 15:e8746. doi: 10.15252/msb.20188746

8. Amezquita RA, Lun ATL, Becht E, Carey VJ, Carpp LN, Geistlinger L, et al. Orchestrating single-cell analysis with Bioconductor. *Nat Methods* (2020) 17:137–45. doi: 10.1038/s41592-019-0654-x

9. Virshup I, Bredikhin D, Heumos L, Palla G, Sturm G, Gayoso A, et al. The scverse project provides a computational ecosystem for single-cell omics data analysis. *Nat Biotechnol* (2023) 41:604–6. doi: 10.1038/s41587-023-01733-8

10. Mereu E, Lafzi A, Moutinho C, Ziegenhain C, Mccarthy DJ, Alvarez-Varela A, et al. Benchmarking single-cell RNA-sequencing protocols for cell atlas projects. *Nat Biotechnol* (2020) 38:747–55. doi: 10.1038/s41587-020-0469-4

11. Lafzi A, Moutinho C, Picelli S, Heyn H. Tutorial: guidelines for the experimental design of single-cell RNA sequencing studies. *Nat Protoc* (2018) 13:2742–57. doi: 10.1038/s41596-018-0073-y

12. Jovic D, Liang X, Zeng H, Lin L, Xu F, Luo Y. Single-cell RNA sequencing technologies and applications: A brief overview. *Clin Transl Med* (2022) 12:e694. doi: 10.1002/ctm2.694

13. Delacher M, Imbusch CD, Weichenhan D, Breiling A, Hotz-Wagenblatt A, Trager U, et al. Genome-wide DNA-methylation landscape defines specialization of regulatory T cells in tissues. *Nat Immunol* (2017) 18:1160–72. doi: 10.1038/ni.3799

14. Delacher M, Schmidl C, Herzig Y, Breloer M, Hartmann W, Brunk F, et al. Rbpj expression in regulatory T cells is critical for restraining T(H)2 responses. *Nat Commun* (2019) 10:1621. doi: 10.1038/s41467-019-09276-w

15. Delacher M, Imbusch CD, Hotz-Wagenblatt A, Mallm JP, Bauer K, Simon M, et al. Precursors for nonlymphoid-tissue treg cells reside in secondary lymphoid organs and are programmed by the transcription factor BATF. *Immunity* (2020) 52:295–312 e11. doi: 10.1016/j.immuni.2019.12.002

16. Delacher M, Simon M, Sanderink L, Hotz-Wagenblatt A, Wuttke M, Schambeck K, et al. Single-cell chromatin accessibility landscape identifies tissue repair program in human regulatory T cells. *Immunity* (2021) 54:702–720 e17. doi: 10.1016/j.immuni.2021.03.007

17. Hwang B, Lee JH, Bang D. Single-cell RNA sequencing technologies and bioinformatics pipelines. *Exp Mol Med* (2018) 50:1–14. doi: 10.1038/s12276-018-0071-8

18. Romiguier J, Ranwez V, Douzery EJ, Galtier N. Contrasting GC-content dynamics across 33 mamMalian genomes: relationship with life-history traits and chromosome sizes. *Genome Res* (2010) 20:1001–9. doi: 10.1101/gr.104372.109

19. Hao Y, Hao S, Andersen-Nissen E, Mauck WM, Zheng S, Butler A, et al. Integrated analysis of multimodal single-cell data. *Cell* (2021) 184:3573–3587 e29. doi: 10.1016/j.cell.2021.04.048

20. Kolodziejczyk AA, Kim JK, Svensson V, Marioni JC, Teichmann SA. The technology and biology of single-cell RNA sequencing. *Mol Cell* (2015) 58:610–20. doi: 10.1016/j.molcel.2015.04.005

21. Svensson V. Droplet scRNA-seq is not zero-inflated. *Nat Biotechnol* (2020) 38:147–50. doi: 10.1038/s41587-019-0379-5

22. Lun ATL, Riesenfeld S, Andrews T, Dao TP, Gomes T, Marioni JC. Participants In The 1st Human Cell Atlas EmptyDrops: distinguishing cells from empty droplets in droplet-based single-cell RNA sequencing data. *Genome Biol* (2019) 20:63. doi: 10.1186/s13059-019-1662-y

23. Huber W, Carey VJ, Gentleman R, Anders S, Carlson M, Carvalho BS, et al. Orchestrating high-throughput genomic analysis with Bioconductor. *Nat Methods* (2015) 12:115–21. doi: 10.1038/nmeth.3252

24. Mccarthy DJ, Campbell KR, Lun AT, Wills QF. Scater: pre-processing, quality control, norMalization and visualization of single-cell RNA-seq data in R. *Bioinformatics* (2017) 33:1179–86. doi: 10.1093/bioinformatics/btw777

25. Hippen AA, Falco MM, Weber LM, Erkan EP, Zhang K, Doherty JA, et al. miQC: An adaptive probabilistic framework for quality control of single-cell RNA-sequencing data. *PloS Comput Biol* (2021) 17:e1009290. doi: 10.1371/journal.pcbi.1009290

26. Mcginnis CS, Murrow LM, Gartner ZJ. DoubletFinder: doublet detection in single-cell RNA sequencing data using artificial nearest neighbors. *Cell Syst* (2019) 8:329–337 e4. doi: 10.1016/j.cels.2019.03.003

27. Xi NM, Li JJ. Benchmarking computational doublet-detection methods for single-cell RNA sequencing data. *Cell Syst* (2021) 12:176–194 e6. doi: 10.1016/j.cels.2020.11.008

28. Germain PL, Lun A, Garcia Meixide C, Macnair W, Robinson MD. Doublet identification in single-cell sequencing data using scDblFinder. *F1000Res* (2021) 10:979. doi: 10.12688/f1000research.73600.1

29. Stegle O, Teichmann SA, Marioni JC. Computational and analytical challenges in single-cell transcriptomics. *Nat Rev Genet* (2015) 16:133–45. doi: 10.1038/nrg3833

30. Vallejos CA, Risso D, Scialdone A, Dudoit S, Marioni JC. NorMalizing single-cell RNA sequencing data: challenges and opportunities. *Nat Methods* (2017) 14:565–71. doi: 10.1038/nmeth.4292

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

32. Lun AT, Mccarthy DJ, Marioni JC. A step-by-step workflow for low-level analysis of single-cell RNA-seq data with Bioconductor. *F1000Res* (2016) 5:2122. doi: 10.12688/f1000research.9501.2

33. Haghverdi L, Lun ATL, Morgan MD, Marioni JC. Batch effects in single-cell RNA-sequencing data are corrected by matching mutual nearest neighbors. *Nat Biotechnol* (2018) 36:421–7. doi: 10.1038/nbt.4091

34. Korsunsky I, Millard N, Fan J, Slowikowski K, Zhang F, Wei K, et al. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nat Methods* (2019) 16:1289–96. doi: 10.1038/s41592-019-0619-0

35. Van Der Maaten L, Hinton G. Visualizing Data using t-SNE. *J Mach Learn Res* (2008) 9:2579–605.

36. Becht E, Mcinnes L, Healy J, Dutertre CA, Kwok IWH, Ng LG, et al. Dimensionality reduction for visualizing single-cell data using UMAP. *Nat Biotechnol* (2018) 37:38–44. doi: 10.1038/nbt.4314

37. Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. *J Stat Mechanics-Theory Experiment* (2008) 2008: P10008. doi: 10.1088/1742-5468/2008/10/P10008

38. Pullin JM, Mccarthy DJ. A comparison of marker gene selection methods for single-cell RNA sequencing data. *bioRxiv* (2022) 2022. doi: 10.1101/2022.05.09.490241

39. De Simone M, Rossetti G, Pagani M. Single cell T cell receptor sequencing: techniques and future challenges. *Front Immunol* (2018) 9:1638. doi: 10.3389/fimmu.2018.01638

40. Cordes M, Pike-Overzet K, Van Den Akker EB, Staal FJT, Cante-Barrett K. Multi-omic analyses in immune cell development with lessons learned from T cell development. *Front Cell Dev Biol* (2023) 11:1163529. doi: 10.3389/fcell.2023.1163529

41. Abdelaal T, Michielsen L, Cats D, Hoogduin D, Mei H, Reinders MJT, et al. A comparison of automatic cell identification methods for single-cell RNA sequencing data. *Genome Biol* (2019) 20:194. doi: 10.1186/s13059-019-1795-z

42. Xia B, Yanai I. A periodic table of cell types. *Development* (2019) 12:dev169854. doi: 10.1242/dev.169854

43. Huang Q, Liu Y, Du Y, Garmire LX. Evaluation of cell type annotation R packages on single-cell RNA-seq data. *Genomics Proteomics Bioinf* (2021) 19:267–81. doi: 10.1016/j.gpb.2020.07.004

44. Osumi-Sutherland D, Xu C, Keays M, Levine AP, Kharchenko PV, Regev A, et al. Cell type ontologies of the Human Cell Atlas. *Nat Cell Biol* (2021) 23:1129–35. doi: 10.1038/s41556-021-00787-7

45. Zeng H. What is a cell type and how to define it? *Cell* (2022) 185:2739–55. doi: 10.1016/j.cell.2022.06.031

46. Clarke ZA, Andrews TS, Atif J, Pouyabahar D, Innes BT, Macparland SA, et al. Tutorial: guidelines for annotating single-cell transcriptomic maps using automated and manual methods. *Nat Protoc* (2021) 16:2749–64. doi: 10.1038/s41596-021-00534-0

47. Aran D, Looney AP, Liu L, Wu E, Fong V, Hsu A, et al. Reference-based analysis of lung single-cell sequencing reveals a transitional profibrotic macrophage. *Nat Immunol* (2019) 20:163–72. doi: 10.1038/s41590-018-0276-y

48. Ji Z, Ji H. TSCAN: Pseudo-time reconstruction and evaluation in single-cell RNA-seq analysis. *Nucleic Acids Res* (2016) 44:e117. doi: 10.1093/nar/gkw430

49. Akhmedov M, Martinelli A, Geiger R, Kwee I. Omics Playground: a comprehensive self-service platform for visualization, analytics and exploration of Big Omics Data. *NAR Genom Bioinform* (2020) 2:lqz019. doi: 10.1093/nargab/lqz019

50. Rue-Albrecht K, Marini F, Soneson C, Lun ATL. iSEE: interactive summarizedExperiment explorer. *F1000Res* (2018) 7:741. doi: 10.12688/f1000research.14966.1