Cell Ranger Count - Comprehensive Reference Guide

Overview

Cell Ranger is 10x Genomics' pipeline for processing single-cell RNA-seq data from their Chromium platform. The `cellranger count` command performs alignment, filtering, barcode counting, and UMI counting to generate gene expression profiles for individual cells.

Website: https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/what-is-cell-ranger
Developer: 10x Genomics
Current Version: 8.0+

Applicable to: 10x Genomics Chromium single-cell RNA-seq (3' and 5' gene expression), single-nucleus RNA-seq

---

What Cell Ranger Count Does

Core Functions

1. **Read Processing** - Demultiplexes and processes FASTQ files from 10x libraries
2. **Cell Barcode Correction** - Corrects sequencing errors in cell barcodes
3. **UMI Deduplication** - Collapses PCR duplicates using Unique Molecular Identifiers
4. **Alignment** - Maps reads to transcriptome using STAR aligner
5. **Cell Calling** - Distinguishes real cells from empty droplets
6. **Gene Quantification** - Counts UMIs per gene per cell
7. **Secondary Analysis** - Performs PCA, t-SNE/UMAP, and clustering

Cell Ranger vs Other Tools

Feature	Cell Ranger	STARsolo	alevin-fry	kallisto bustools
Platform	10x only	Universal	Universal	Universal
Speed	Moderate	Fast	Very fast	Very fast
Accuracy	Excellent	Excellent	Excellent	Very good
Cell calling	Sophisticated	Basic	Advanced	Basic
Memory	High (64GB)	High (64GB)	Moderate (32GB)	Low (16GB)
Ease of use	Simple	Complex	Moderate	Moderate
Support	Commercial	Open source	Open source	Open source
Output format	10x standard	Multiple	Multiple	Multiple

Why use Cell Ranger:

- Official 10x Genomics tool (guaranteed compatibility)
- Optimized for 10x chemistry
- Sophisticated cell calling algorithm
- Generates Loupe Browser files
- Extensive documentation and support
- Handles chemistry detection automatically

Why consider alternatives:

- Closed source (some components)
- Higher memory requirements
- Slower than some alternatives
- 10x platform specific

Cell Ranger Workflow

Step-by-Step Process

```
Input: Illumina BCL files OR FASTQ files

Step 1: FASTQ Processing
    Extract cell barcodes (CB) from R1
    Extract UMIs from R1
    Correct barcode errors (1-base substitutions)
    Quality filter reads

Step 2: Alignment (STAR)
    Align R2 (cDNA) to transcriptome
    Map to genome for intronic reads (if --include-introns)
    Generate BAM file (optional)

Step 3: Cell Calling
    Calculate UMI counts per barcode
    Distinguish cells from empty droplets
    Apply EmptyDrops-like algorithm
    Filter cell barcodes

Step 4: UMI Counting
    Deduplicate UMIs per gene per cell
    Correct UMI sequencing errors
    Generate count matrix

Step 5: Secondary Analysis (optional)
    Normalize counts
    PCA dimensionality reduction
    t-SNE and UMAP projection
```

```
    Graph-based clustering
    Differential expression

Output: Count matrices, QC metrics, analysis results
```

Basic Command

```
cellranger count \
    --id=sample1 \
    --transcriptome=/path/to/refdata-gex-GRCh38-2024-A \
    --fastqs=/path/to/fastqs \
    --sample=Sample1 \
    --localcores=16 \
    --localmem=64
```

Command Breakdown

```
cellranger count \                                  # Count pipeline
    --id=sample1 \                                  # Output directory name
    --transcriptome=/path/to/reference \            # Cell Ranger reference
    --fastqs=/path/to/fastqs \                      # Directory with FASTQ files
    --sample=Sample1 \                              # Sample name (FASTQ prefix)
    --localcores=16 \                               # CPU cores to use
    --localmem=64                                   # Memory in GB
```

Key Parameters Explained

Required Parameters

--id **Purpose**: Output directory name

Usage:

```
--id=sample1     # Creates output in ./sample1/
```

Best practices:

- Use sample identifier as ID
- Avoid spaces and special characters
- Use underscores or hyphens: `Sample_1`, `Patient-001`

Note: If directory exists, Cell Ranger will fail unless using `--force-cells` or deleting old output

---

--transcriptome **Purpose**: Path to Cell Ranger reference transcriptome

Format: Directory created by `cellranger mkref`

Pre-built references (download from 10x Genomics):

```
# Human
refdata-gex-GRCh38-2024-A           # Human (GRCh38) latest
refdata-gex-GRCh38-2020-A           # Human (GRCh38) older

# Mouse
refdata-gex-mm10-2024-A             # Mouse (mm10) latest
refdata-gex-mm10-2020-A             # Mouse (mm10) older

# Pre-built with introns
refdata-gex-GRCh38-and-mm10-2024-A # Human + Mouse (barnyard)
```

Custom references:

```
cellranger mkref \
    --genome=custom_ref \
    --fasta=genome.fa \
    --genes=genes.gtf
```

--fastqs **Purpose**: Path to directory containing FASTQ files

Expected structure:

```
fastqs/
  Sample1_S1_L001_R1_001.fastq.gz
  Sample1_S1_L001_R2_001.fastq.gz
  Sample1_S1_L002_R1_001.fastq.gz
  Sample1_S1_L002_R2_001.fastq.gz
  ...
```

FASTQ naming requirements:

- Must follow Illumina naming convention
- Format: `[SampleName]_S[#]_L[Lane]_[R1/R2/I1/I2]_001.fastq.gz`
- R1: Cell barcode + UMI (typically 26-28bp)
- R2: cDNA read (typically 91-98bp)

Multiple FASTQ directories:

```
# Comma-separated paths
--fastqs=/path/to/run1,/path/to/run2

# Or use --fastqs multiple times
```

```
--fastqs=/path/to/run1 \
--fastqs=/path/to/run2
```

--sample **Purpose**: Sample name (must match FASTQ prefix)

Example:

```
# For FASTQs: Sample1_S1_L001_R1_001.fastq.gz
--sample=Sample1     # Matches "Sample1" prefix

# For multiple samples in same directory
--sample=Sample1,Sample2,Sample3

# Or specify multiple times
--sample=Sample1 \
--sample=Sample2
```

Common errors:

```
#  WRONG - case sensitive
--sample=sample1     # When FASTQ is Sample1_*

#  WRONG - includes flowcell info
--sample=Sample1_S1

#  CORRECT
--sample=Sample1
```

Resource Parameters

--localcores **Purpose**: Number of CPU cores to use

Recommendations:

```
--localcores=4      # Minimum (slow, testing)
--localcores=8      # Acceptable (small samples)
--localcores=16     # Recommended (standard)
--localcores=32     # Optimal (large samples)
```

Scaling:

- 8 cores: ~2-3 hours per sample
- 16 cores: ~1-1.5 hours per sample
- 32 cores: ~45-60 minutes per sample

Diminishing returns after 32 cores

--localmem **Purpose**: Maximum memory in GB

Requirements:

```
--localmem=64      # Minimum for human
--localmem=128     # Comfortable for human
--localmem=32      # May work for mouse
```

Memory scaling by species:

- Human (GRCh38): 64GB minimum, 128GB recommended
- Mouse (mm10): 32GB minimum, 64GB recommended
- Zebrafish: 24GB minimum

Why so much memory?:

- STAR index loaded into RAM (~30GB for human)
- Sparse matrix operations
- Cell barcode tracking
- Temporary buffers

Chemistry Parameters

--chemistry **Purpose**: Specify or auto-detect 10x chemistry

Auto-detection (default):

```
# No parameter needed - Cell Ranger detects automatically
cellranger count --id=sample1 ...
```

Cell Ranger examines read structure and barcode whitelist to determine chemistry.

Manual specification (only if auto-detection fails):

```
--chemistry=SC3Pv3     # Single Cell 3' v3 (most common)
--chemistry=SC3Pv2     # Single Cell 3' v2 (older)
--chemistry=SC5P-PE    # Single Cell 5' paired-end
--chemistry=SC5P-R2    # Single Cell 5' R2-only
--chemistry=ARC-v1     # Multiome ATAC + Gene Expression
```

Chemistry versions:

Chemistry	Release	Read Structure	Use Case
SC3Pv3	2018	R1: 28bp, R2: 91bp	Current standard 3'
SC3Pv2	2016	R1: 26bp, R2: 98bp	Older 3' protocol
SC5P-PE	2017	R1: 26bp, R2: 98bp	Full-length transcripts
SC5P-R2	2019	R1: 26bp, R2: 90bp	5' gene expression
ARC-v1	2021	Multiple	Joint ATAC + RNA

When to specify manually:

- Mixed chemistries in same run (rare)
- Auto-detection fails
- Custom chemistry
- Standard runs (auto-detect is robust)

Cell Calling Parameters

--expect-cells **Purpose**: Expected number of recovered cells

Usage:

```
--expect-cells=5000     # Expecting ~5,000 cells
```

How it affects cell calling:

- Used as prior for EmptyDrops-like algorithm
- Influences UMI threshold for cell detection
- Does NOT hard-cap the number of cells called

Recommendations:

Loading Target	--expect-cells
Standard	Don't specify (auto)
High loading	10000
Low loading	1000-2000
Nuclei	5000-10000

When to use:

- Known loading concentration
- Nuclei samples (higher background)
- Difficult cell calling
- Standard whole-cell experiments

Default behavior: If not specified, Cell Ranger estimates from data

---

--force-cells **Purpose**: Force Cell Ranger to call exactly N cells

Usage:

```
--force-cells=5000     # Forces exactly 5,000 cells
```

WARNING: This bypasses sophisticated cell calling algorithm!

When to use (rarely):

- Known exact cell number from separate count
- Benchmarking different thresholds
- Rescue low-quality samples

Risks:

- May include empty droplets (low genes/cell)
- May exclude real cells (on threshold edge)
- Defeats purpose of UMI-based cell calling

Recommendation: Avoid unless necessary. Use `--expect-cells` instead.

Analysis Parameters

--nosecondary **Purpose**: Skip secondary analysis (PCA, clustering, etc.)

Usage:

```
--nosecondary     # Faster, no dimensionality reduction
```

What gets skipped:

- PCA
- t-SNE and UMAP projection
- Graph-based clustering
- Differential expression
- Feature selection

Output still includes:

- Count matrices (raw and filtered)
- Cell calling
- UMI counts
- QC metrics
- Web summary

When to use:

- Quick QC check
- Downstream analysis in R/Python
- Batch processing (analyze later)
- Saving compute time
- Need immediate exploratory analysis

Time savings: ~20-30% faster

--include-introns **Purpose**: Count intronic reads (unspliced pre-mRNA)

Usage:

```
--include-introns=true     # Count intronic + exonic reads
```

Default: Only exonic reads counted

Why count introns?:

Single-nucleus RNA-seq:

```
Nuclei contain:
    Mature mRNA (exonic)
    Pre-mRNA (intronic + exonic)
    Unspliced transcripts

Without --include-introns:
  → Many reads discarded
  → Lower UMI counts per cell
  → Biased toward highly spliced genes

With --include-introns:
  → 2-3x more UMIs per cell
  → Better detection of lowly expressed genes
  → More accurate quantification
```

RNA velocity analysis:

```
Requires both:
  - Spliced reads (mature mRNA)
  - Unspliced reads (pre-mRNA)

--include-introns captures nascent transcription
→ Enables trajectory inference
```

When to use:

- **Always use for nuclei**
- RNA velocity analysis
- Nascent RNA studies
- Standard whole-cell RNA-seq (introduces noise)

Trade-offs:

- Higher UMI counts (2-3x for nuclei)
- Better gene detection
- Potential genomic DNA contamination
- Slightly longer runtime

--no-bam **Purpose**: Skip BAM file generation

Usage:

```
--no-bam    # Don't create possorted_genome_bam.bam
```

What gets skipped:

- Coordinate-sorted BAM file
- BAM index (.bai)

Output still includes:

- All count matrices
- All QC metrics
- All analysis results

BAM file uses:

- IGV visualization
- Read-level QC (RSeQC, Picard)
- Reanalysis with different parameters
- Alternative alignment viewers

When to use --no-bam:

- Disk space limited (BAM ~5-10GB per sample)
- Only need count matrices
- Batch processing many samples
- Need read-level visualization
- Detailed QC required

Disk savings: ~5-10GB per sample

Performance Parameters

--localvmem **Purpose**: Maximum virtual memory in GB

Usage:

```
--localmem=128 \          # Physical memory
--localvmem=192           # Virtual memory (1.5x physical)
```

Default: Automatically calculated from `--localmem`

When to specify:

- System has swap space
- Memory-constrained environment
- Need precise control

Recommendation: Usually not needed

Output Files

Directory Structure

```
sample1/
  outs/
      web_summary.html                    #  Main QC report
      metrics_summary.csv                 # Key metrics table

      filtered_feature_bc_matrix/         #  Main output (cells only)
          barcodes.tsv.gz                 # Cell barcodes
          features.tsv.gz                 # Gene IDs and names
          matrix.mtx.gz                   # Sparse count matrix

      raw_feature_bc_matrix/              # Unfiltered (all barcodes)
          barcodes.tsv.gz
          features.tsv.gz
          matrix.mtx.gz

      filtered_feature_bc_matrix.h5       # HDF5 format (filtered)
      raw_feature_bc_matrix.h5            # HDF5 format (raw)
      molecule_info.h5                    # Molecule-level data

      possorted_genome_bam.bam            # Aligned reads (optional)
      possorted_genome_bam.bam.bai        # BAM index

      cloupe.cloupe                       #  Loupe Browser file

      analysis/                           # Secondary analysis
          pca/                            # PCA results
          tsne/                           # t-SNE projection
          umap/                           # UMAP projection
          clustering/                     # Graph-based clusters
          diffexp/                        # Differential expression

      [Other files]

  SC_RNA_COUNTER_CS/                       # Internal pipeline files
      ...
```

Key Output Files

1. web_summary.html **Description**: Interactive HTML report with plots and metrics

Sections:

1. **Summary**
    - Estimated cells
    - Mean reads per cell
    - Median genes per cell
    - Total genes detected
    - Median UMI counts per cell
2. **Sequencing**
    - Number of reads
    - Valid barcodes
    - Sequencing saturation
    - Q30 bases in barcode/UMI/read
3. **Mapping**
    - Reads mapped to genome
    - Reads mapped to transcriptome
    - Reads mapped confidently to transcriptome
    - Reads mapped to intronic regions (if --include-introns)
4. **Cells**
    - Barcode rank plot (knee plot)
    - Fraction of reads in cells
5. **Gene Expression**
    - Median genes per cell
    - Total genes detected
    - Median UMI counts per cell

Interactive features:

- Hover for exact values
- Zoom on plots
- Download images

---

2. filtered_feature_bc_matrix/ Description: Main count matrix (genes × cells)

Format: 10x Genomics Matrix Market format

Files:

barcodes.tsv.gz:

```
AAACCTGAGAAACCAT-1
AAACCTGAGAAACCGC-1
AAACCTGAGAAACCTA-1
```

...

- One barcode per line
- Suffix `-1` indicates GEM group (usually all `-1`)
- Number of lines = number of cells

features.tsv.gz:

```
ENSG00000243485 MIR1302-2HG Gene Expression
ENSG00000237613 FAM138A Gene Expression
ENSG00000186092 OR4F5   Gene Expression
...
```

- Column 1: Gene ID (Ensembl)
- Column 2: Gene symbol
- Column 3: Feature type (always "Gene Expression" for count)
- Number of lines = number of genes

matrix.mtx.gz:

```
%%MatrixMarket matrix coordinate integer general
32738 5000 10000000
1 1 5
1 2 3
2 1 1
3 1 12
...
```

- Line 1: Header (Matrix Market format)
- Line 2: Dimensions (genes, cells, non-zero entries)
- Remaining lines: `gene_index cell_index count`
- **Sparse format**: Only non-zero counts stored

Reading in R:

```r
library(Matrix)
library(Seurat)

# Method 1: Seurat
data <- Read10X(data.dir = "sample1/outs/filtered_feature_bc_matrix")
seurat <- CreateSeuratObject(counts = data, project = "sample1")

# Method 2: Manual
mat <- readMM("filtered_feature_bc_matrix/matrix.mtx.gz")
features <- read.delim("filtered_feature_bc_matrix/features.tsv.gz",
                       header = FALSE)
barcodes <- read.delim("filtered_feature_bc_matrix/barcodes.tsv.gz",
                       header = FALSE)
rownames(mat) <- features$V2  # Gene symbols
colnames(mat) <- barcodes$V1  # Cell barcodes
```

Reading in Python:

```python
import scanpy as sc

# Load 10x data
adata = sc.read_10x_mtx('sample1/outs/filtered_feature_bc_matrix',
                        var_names='gene_symbols',
                        cache=True)
```

---

3. raw_feature_bc_matrix/ **Description**: Unfiltered count matrix (all barcodes)

Includes:

- Called cells (same as filtered matrix)
- Empty droplets
- Debris
- Background

Uses:

- Custom cell calling with EmptyDrops
- Cell calling threshold evaluation
- Ambient RNA profiling
- Contamination detection

Size: 10-100x larger than filtered matrix

Example R code for custom cell calling:

```r
library(DropletUtils)

# Load raw matrix
raw <- read10xCounts("sample1/outs/raw_feature_bc_matrix")

# Run EmptyDrops
e.out <- emptyDrops(counts(raw), lower=100, niters=10000)

# Filter cells (FDR < 0.01)
is.cell <- e.out$FDR <= 0.01
cells <- raw[, which(is.cell)]
```

---

4. HDF5 Files (.h5) **Description**: Same data as feature-barcode matrices, but in HDF5 format

Files:

- `filtered_feature_bc_matrix.h5`: Filtered matrix
- `raw_feature_bc_matrix.h5`: Unfiltered matrix
- `molecule_info.h5`: Molecule-level information

Advantages:

- Single file (easier to transfer)
- Faster to read for some tools
- Smaller file size

Reading:

```r
library(Seurat)
data <- Read10X_h5("sample1/outs/filtered_feature_bc_matrix.h5")
```

```python
import scanpy as sc
adata = sc.read_10x_h5("sample1/outs/filtered_feature_bc_matrix.h5")
```

---

5. cloupe.cloupe **Description**: File for 10x Loupe Browser visualization

Loupe Browser features:

- Interactive t-SNE/UMAP exploration
- Gene expression visualization
- Cluster annotation
- Differential expression
- Export publication-quality figures

Download: https://support.10xgenomics.com/single-cell-gene-expression/software/visualization/latest/what-is-loupe-cell-browser

---

6. metrics_summary.csv **Description**: Key metrics in CSV format

Example:

```
Metric,Value
Estimated Number of Cells,"5,234"
Mean Reads per Cell,"50,123"
Median Genes per Cell,"2,456"
Total Genes Detected,"18,234"
Median UMI Counts per Cell,"8,901"
Sequencing Saturation,78.5%
Reads Mapped Confidently to Transcriptome,85.3%
Fraction Reads in Cells,91.2%
...
```

Uses:

- Programmatic QC filtering
- Batch comparison
- Summary tables

Understanding Key Metrics

Estimated Number of Cells

What it is: Number of cell barcodes called as real cells

Expected range:

- Good: Matches loading target (±20%)
- Poor: <50% of target or >2x target

Low cell count causes:

1. **Low viability** - Dead cells don't capture
2. **Low loading** - Fewer cells than intended
3. **Low RNA content** - Cells below detection threshold
4. **Clogging** - Microfluidic chip blockage

High cell count causes:

1. **High loading** - More cells than intended
2. **Doublets** - Two cells per droplet
3. **Debris** - Cell-free RNA incorrectly called

Median Genes per Cell

What it is: Median number of genes detected (>0 UMIs) per cell

Expected range:

Cell Type	Good	Acceptable	Poor
PBMCs	>1000	500-1000	<500
Cell lines	>2000	1000-2000	<1000
Neurons	>3000	2000-3000	<2000
Tissue	>1500	800-1500	<800

Low genes/cell causes:

1. **Low sequencing depth** - Need more reads
2. **Poor RNA quality** - RNA degradation
3. **Low complexity** - Limited cell types
4. **Wrong cell calling threshold** - Including empties

Median UMI Counts per Cell

What it is: Median total UMI count per cell (all genes summed)

Expected range:

Sequencing Depth	UMIs/Cell
Shallow (~20K reads/cell)	1,000-3,000
Standard (~50K reads/cell)	3,000-10,000
Deep (~100K reads/cell)	10,000-30,000

Relationship to depth:

```
More reads → More unique UMIs detected
But: Diminishing returns due to saturation
```

Sequencing Saturation

What it is: Fraction of reads that are duplicates of already-captured UMIs

Formula: `1 - (unique UMIs / total reads)`

Interpretation:

Saturation	Status	Action
<50%	Undersaturated	Can sequence more
50-70%	Good	Optimal
70-90%	High	Diminishing returns
>90%	Oversaturated	Wasted sequencing

Why it matters:

```
Low saturation (<50%):
  → More sequencing increases UMI counts proportionally
  → Cost-effective to sequence deeper

High saturation (>80%):
  → More sequencing yields minimal new UMIs
  → Wasteful to sequence deeper
```

Example:

```
Sample A: 40% saturation, 5,000 UMIs/cell
  → Doubling depth → ~8,000 UMIs/cell (60% increase)

Sample B: 85% saturation, 10,000 UMIs/cell
  → Doubling depth → ~10,800 UMIs/cell (8% increase)
```

Reads Mapped Confidently to Transcriptome

What it is: Percentage of reads that map uniquely to exonic regions

Expected range:

Quality	Percentage
Excellent	>70%
Good	60-70%
Acceptable	50-60%
Poor	<50%

Low mapping causes:

1. **Wrong reference** - Species mismatch
2. **Contamination** - Bacterial, viral
3. **Genomic DNA** - Intronic/intergenic reads
4. **Low library quality** - Adapter dimers

Fraction Reads in Cells

What it is: Percentage of reads from called cell barcodes (vs background)

Expected range:

Quality	Percentage
Excellent	>80%
Good	70-80%
Acceptable	60-70%
Poor	<60%

Why it matters:

```
High fraction (>80%):
  → Most reads from real cells
  → Little ambient RNA
```

```
  → Efficient cell capture

Low fraction (<60%):
  → Many reads from empty droplets
  → High ambient RNA contamination
  → Poor cell loading or viability
```

Resource Requirements

Memory (RAM)

Component	Requirement
STAR index	30GB (human)
Sparse matrices	10-20GB
Cell calling	5-10GB
Buffer	10GB
Total	**64GB minimum**

Recommendations:

- Human: 64GB minimum, 128GB comfortable
- Mouse: 32GB minimum, 64GB comfortable
- Smaller organisms: 24-32GB

CPU Cores

Scaling:

- 8 cores: ~2-3 hours
- 16 cores: ~1-1.5 hours (recommended)
- 32 cores: ~45-60 minutes
- 64 cores: ~35-45 minutes (diminishing returns)

Optimal: 16-32 cores

Disk Space

Per sample:

Component	Size
FASTQ files	5-15GB

Component	Size
BAM file	5-10GB
Count matrices	0.5-2GB
Temporary files	10-30GB
Total workspace	**30-60GB**

Recommendation: 100GB free space per sample

Time

Sample Size	16 cores	32 cores
5K cells, 50K reads/cell	60 min	40 min
10K cells, 50K reads/cell	80 min	50 min
5K cells, 100K reads/cell	90 min	55 min
10K cells, 100K reads/cell	120 min	70 min

Troubleshooting

Low Cell Count (<50% of Target)

Diagnostic steps:

1. **Check cell loading**:
   - Look at barcode rank plot in web summary
   - Steep drop-off → Good separation
   - Gradual slope → Poor loading
2. **Check viability**:
   - If loaded 10K cells but got 2K → ~20% viability
   - Trypan blue staining before loading
   - Check for clumping
3. **Check RNA quality**:
   - Low genes/cell → RNA degradation
   - Run BioAnalyzer or TapeStation
4. **Try --expect-cells**:

```
# If auto-calling too conservative
--expect-cells=10000
```

Low Genes per Cell (<500)

Causes and solutions:

1. **Insufficient sequencing depth**:

   ```
   # Check reads per cell
   # If <20K, need more sequencing
   ```

2. **RNA degradation**:

   - Check RIN score (should be >7)
   - Check fragment length distribution
   - Use fresh/frozen tissue, not FFPE

3. **Wrong cell type**:

   - Some cells naturally have low complexity
   - Differentiated cells: Lower
   - Stem cells: Higher

4. **Cell stress**:

   - Long dissociation → Stress response
   - Optimize tissue dissociation protocol

Low Mapping Rate (<50%)

Diagnostic steps:

1. **Verify reference species**:

   ```
   # Check FASTQ headers for organism
   zcat R2.fastq.gz | head -n 10

   # Check reference
   grep "genome" transcriptome/reference.json
   ```

2. **Check for contamination**:

   - Run FastQC
   - Look for overrepresented sequences
   - Check for bacterial content

3. **Genomic DNA contamination**:

   - High intronic reads
   - Check DNase treatment
   - May need better RNA purification

High Mitochondrial Reads (>20%)

What it means:

- Cells with high mt% are stressed/dying
- Released cytoplasmic RNA, but intact mitochondria
- Common in tissue dissociation

Solutions:

1. **Filter in downstream analysis**:

   ```
   # Calculate mt %
   seurat[["percent.mt"]] <- PercentageFeatureSet(seurat, pattern = "^MT-")

   # Filter cells
   seurat <- subset(seurat, subset = percent.mt < 20)
   ```

2. **Optimize dissociation**:
   - Reduce dissociation time
   - Lower temperature
   - Gentler mechanical disruption

3. **Accept if biological**:
   - Neurons naturally have high mt%
   - Some tissues inherently stressed

Sample Name Mismatch Error

Error:

```
No FASTQs found for requested sample: "Sample1"
```

Causes:

1. **Case sensitivity**:

   ```
   #   WRONG
   --sample=sample1     # When FASTQ is Sample1_*

   #   CORRECT
   --sample=Sample1
   ```

2. **Wrong prefix**:

   ```
   # FASTQ: MyExperiment_Sample1_S1_L001_R1_001.fastq.gz

   #   WRONG
   --sample=Sample1
   ```

```
#   CORRECT
--sample=MyExperiment_Sample1
```

3. **FASTQ in wrong location**:

```
# Check FASTQs are in --fastqs directory
ls -lh /path/to/fastqs/Sample1*.fastq.gz
```

Out of Memory

Error: "STAR process was killed (probably due to memory limits)"

Solutions:

1. **Increase memory**:

```
--localmem=128     # If you have 128GB RAM
```

2. **Use smaller reference**:

```
# Remove non-primary chromosomes
# Custom reference with cellranger mkref
```

3. **Close other programs**:
   - Free up RAM before running
   - Check with `free -h`

Best Practices

Standard 3' Gene Expression (Cells)

```
cellranger count \
    --id=sample1 \
    --transcriptome=/ref/refdata-gex-GRCh38-2024-A \
    --fastqs=/path/to/fastqs \
    --sample=Sample1 \
    --localcores=16 \
    --localmem=64
```

Single-Nucleus RNA-seq

```
cellranger count \
    --id=sample1 \
    --transcriptome=/ref/refdata-gex-GRCh38-2024-A \
    --fastqs=/path/to/fastqs \
    --sample=Sample1 \
```

```
    --include-introns=true \    # Critical for nuclei!
    --localcores=16 \
    --localmem=64
```

Quick QC (Speed Priority)

```
cellranger count \
    --id=sample1 \
    --transcriptome=/ref/refdata-gex-GRCh38-2024-A \
    --fastqs=/path/to/fastqs \
    --sample=Sample1 \
    --nosecondary \             # Skip clustering
    --no-bam \                  # Skip BAM
    --localcores=32 \           # More cores
    --localmem=64
```

Disk Space Limited

```
cellranger count \
    --id=sample1 \
    --transcriptome=/ref/refdata-gex-GRCh38-2024-A \
    --fastqs=/path/to/fastqs \
    --sample=Sample1 \
    --no-bam \                  # Saves 5-10GB per sample
    --localcores=16 \
    --localmem=64
```

Downstream Analysis

Seurat (R)

```
library(Seurat)
library(dplyr)

# Load data
data <- Read10X(data.dir = "sample1/outs/filtered_feature_bc_matrix")
seurat <- CreateSeuratObject(counts = data, project = "sample1",
                             min.cells = 3, min.features = 200)

# QC filtering
seurat[["percent.mt"]] <- PercentageFeatureSet(seurat, pattern = "^MT-")
seurat <- subset(seurat, subset = nFeature_RNA > 200 &
```

```r
                                nFeature_RNA < 5000 &
                                percent.mt < 20)

# Normalization and scaling
seurat <- NormalizeData(seurat)
seurat <- FindVariableFeatures(seurat, selection.method = "vst",
                               nfeatures = 2000)
seurat <- ScaleData(seurat)

# Dimensionality reduction
seurat <- RunPCA(seurat, npcs = 50)
seurat <- RunUMAP(seurat, dims = 1:30)

# Clustering
seurat <- FindNeighbors(seurat, dims = 1:30)
seurat <- FindClusters(seurat, resolution = 0.5)

# Visualization
DimPlot(seurat, reduction = "umap", label = TRUE)
FeaturePlot(seurat, features = c("CD3D", "CD8A", "CD4", "CD14"))

# Find markers
markers <- FindAllMarkers(seurat, only.pos = TRUE, min.pct = 0.25,
                          logfc.threshold = 0.25)
```

---

scanpy (Python)

```python
import scanpy as sc
import numpy as np

# Load data
adata = sc.read_10x_mtx('sample1/outs/filtered_feature_bc_matrix',
                        var_names='gene_symbols', cache=True)

# QC filtering
sc.pp.calculate_qc_metrics(adata, qc_vars=['mt'],
                           percent_top=None, log1p=False,
                           inplace=True)
adata = adata[adata.obs.n_genes_by_counts < 5000, :]
adata = adata[adata.obs.pct_counts_mt < 20, :]

# Normalization
sc.pp.normalize_total(adata, target_sum=1e4)
sc.pp.log1p(adata)
```

```
# Variable genes
sc.pp.highly_variable_genes(adata, min_mean=0.0125, max_mean=3,
                            min_disp=0.5)
adata = adata[:, adata.var.highly_variable]

# Scaling and PCA
sc.pp.scale(adata, max_value=10)
sc.tl.pca(adata, svd_solver='arpack')

# UMAP and clustering
sc.pp.neighbors(adata, n_neighbors=10, n_pcs=40)
sc.tl.umap(adata)
sc.tl.leiden(adata, resolution=0.5)

# Visualization
sc.pl.umap(adata, color=['leiden', 'CD3D', 'CD8A'])

# Find markers
sc.tl.rank_genes_groups(adata, 'leiden', method='wilcoxon')
sc.pl.rank_genes_groups(adata, n_genes=20, sharey=False)
```

Related Documentation

- **Cell Ranger mkref**: https://support.10xgenomics.com/single-cell-gene-expression/software/pipelines/latest/using/mkref
- **Cell Ranger aggr**: Aggregating multiple samples
- **Seurat**: https://satijalab.org/seurat/
- **scanpy**: https://scanpy.readthedocs.io/
- **10x Support**: https://support.10xgenomics.com/

Document Version: 1.0
Last Updated: February 2026
Cell Ranger Version: 8.0+
Applicable to: 10x Genomics Chromium single-cell RNA-seq (3' and 5'), single-nucleus RNA-seq