Lecture 5:

Transcriptomics and Single-Cell Analysis

From bulk to single-cell • Cell atlas projects • Resolution revolution

Introduction to Biomedical Data Science

Lecture Contents

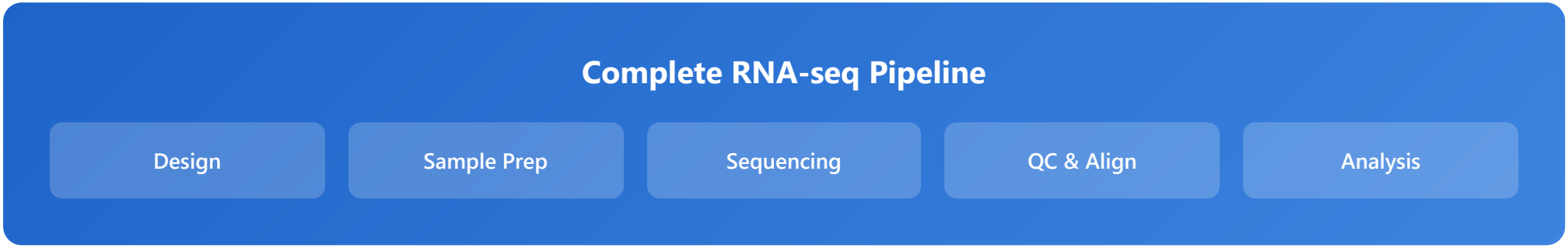
Part 1/3:

Bulk RNA-seq

- Expression profiling
- Differential analysis
- Pathway enrichment
- Time series

RNA-seq Workflow

Comprehensive Guide to Experimental Design and Best Practices

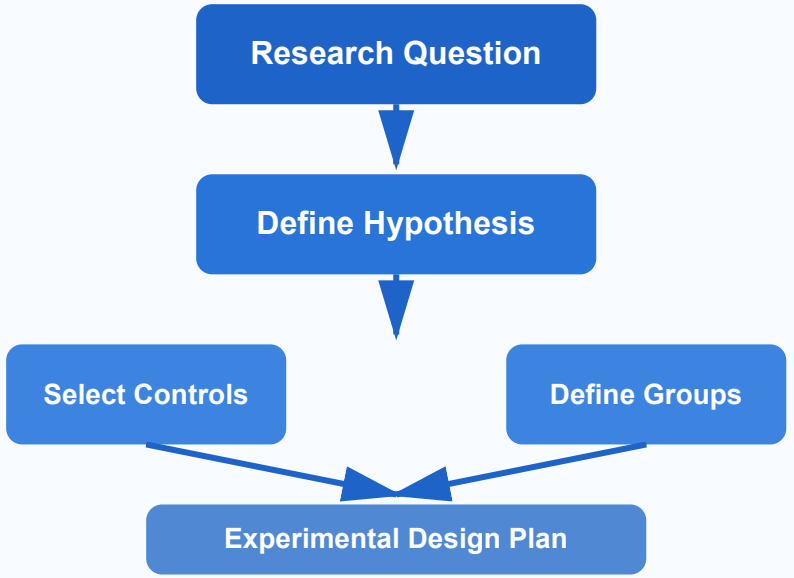


1 Experimental Design

Experimental design is the foundation of any successful RNA-seq study. A well-designed experiment ensures that the biological questions can be answered with statistical confidence while avoiding common pitfalls that can compromise data quality.

Critical considerations:

- Define clear biological hypotheses before starting
- Identify appropriate control groups
- Account for potential confounding variables
- Plan for both technical and biological variation
- Consider the statistical model for downstream analysis



Key Design Principles

- **Randomization:** Randomly assign samples to processing batches to prevent systematic bias
- **Blocking:** Group samples by known sources of variation (e.g., batch, sex, age)
- **Control Selection:** Use appropriate negative and positive controls
- **Sample Size:** Calculate required sample size based on expected effect size and variance

Common Pitfall: Starting sequencing without a clear analysis plan often leads to underpowered studies or inability to test the intended hypothesis.

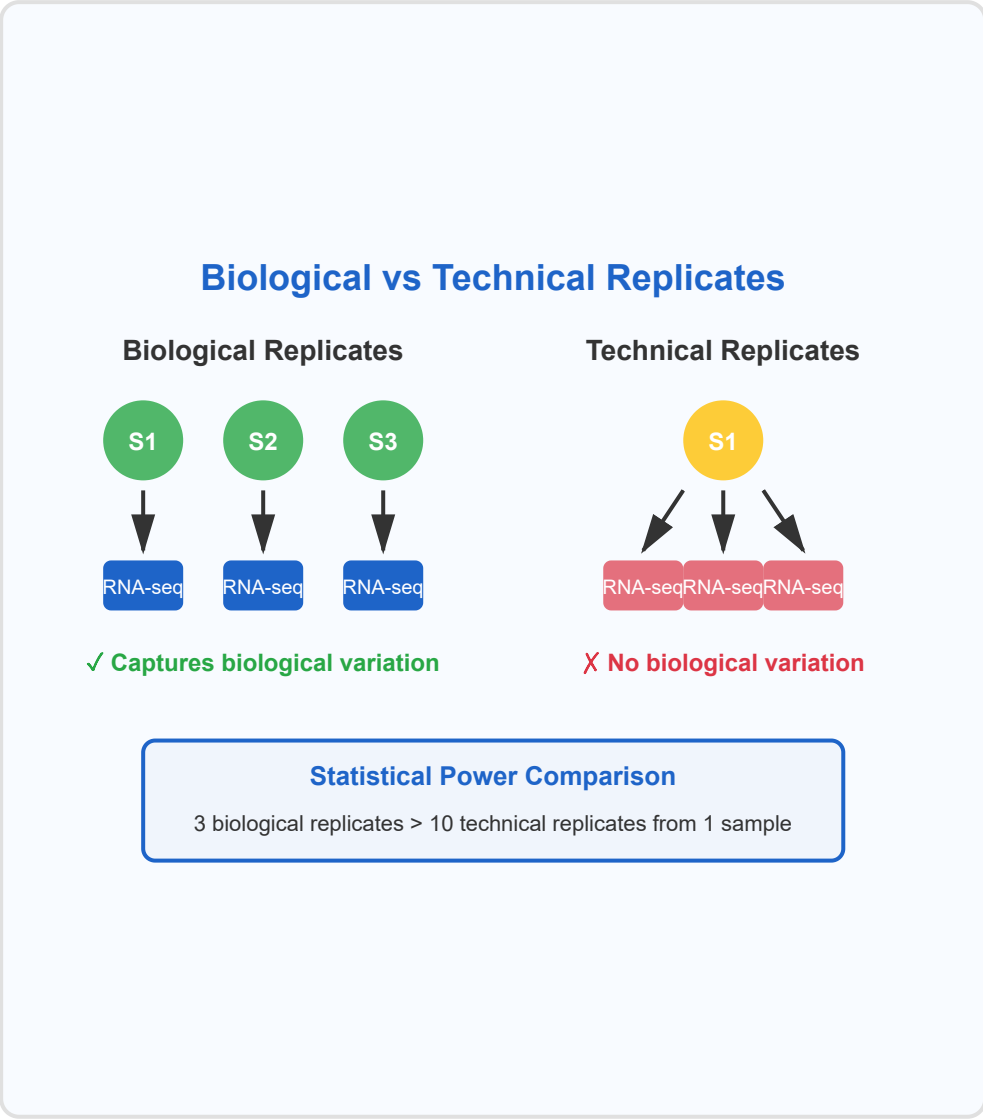
2 Replication Strategies

Replication is crucial for distinguishing true biological variation from technical noise. There are two types of replicates in RNA-seq: **technical replicates** (same sample sequenced multiple times) and **biological replicates** (independent samples from the same condition).

Biological replicates are essential because they capture the natural variation between individuals, which is typically much larger than technical variation. Modern RNA-seq platforms have low technical variation, making technical replicates less necessary.

```
Minimum Recommended: n ≥ 3 biological replicates per condition
```

For optimal statistical power, 5-6 replicates per group is recommended, especially when expecting small effect sizes or high biological variability.



Replication Best Practices

- **Prioritize biological replicates:** They provide information about true biological variation
- **Minimum of 3 replicates:** Required for basic statistical tests (t-tests, DESeq2, edgeR)
- **5-6 replicates optimal:** Provides good power to detect moderate changes (2-fold) with reasonable FDR
- **Avoid pooling:** Pooling samples reduces the ability to assess biological variation
- **Technical replicates rarely needed:** Modern platforms have <5% technical CV

Replicates	Statistical Power	Recommended For
n = 2	Insufficient - No statistical testing	Pilot studies only
n = 3	Minimal - Detects large changes only	Well-defined systems, large effects
n = 5-6	Good - Detects moderate changes (1.5-2 fold)	Most RNA-seq experiments
n ≥ 10	Excellent - Detects subtle changes	Complex studies, small effect sizes

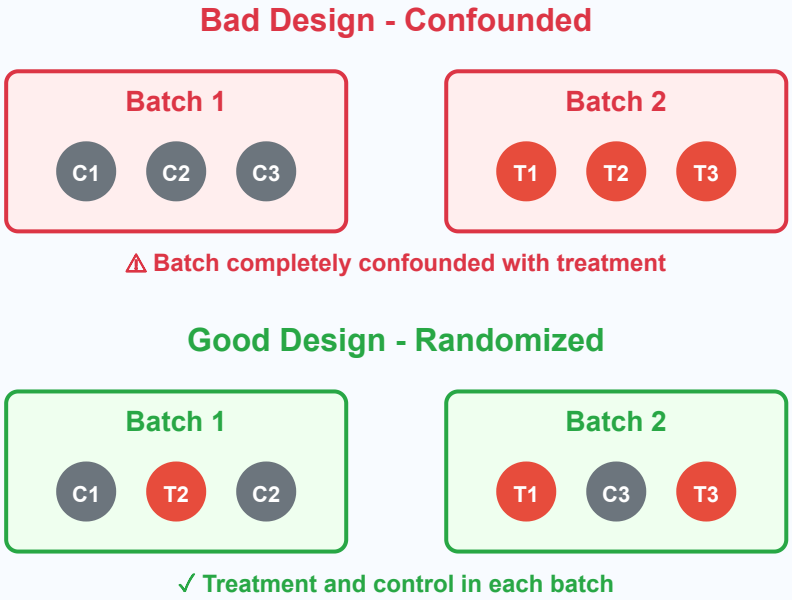
3 Batch Effect Prevention

Batch effects are systematic, non-biological differences between groups of samples processed at different times or under different conditions. They can confound biological signals and lead to false conclusions if not properly controlled.

Common sources of batch effects:

- Different processing days/times
- Different technicians or laboratories
- Different reagent lots or kits
- Different sequencing runs or flow cells
- Seasonal or environmental variations

The gold standard for preventing batch effects is **complete randomization**, where samples from all experimental groups are randomly distributed across all batches.



Batch Effect Prevention Strategies

- **Randomize sample processing:** Distribute conditions evenly across all batches
- **Process all samples together:** When possible, process all samples in a single batch

- **Block design:** If batches are unavoidable, ensure each batch contains all conditions
- **Record metadata:** Document all processing information (date, technician, kit lot, etc.)
- **Use computational correction:** Apply batch correction methods (ComBat, limma) if needed
- **Include controls:** Add control samples to each batch to monitor batch effects

Important Note: Batch effects cannot be fully corrected computationally if they are completely confounded with the biological variable of interest. Prevention through proper experimental design is crucial.

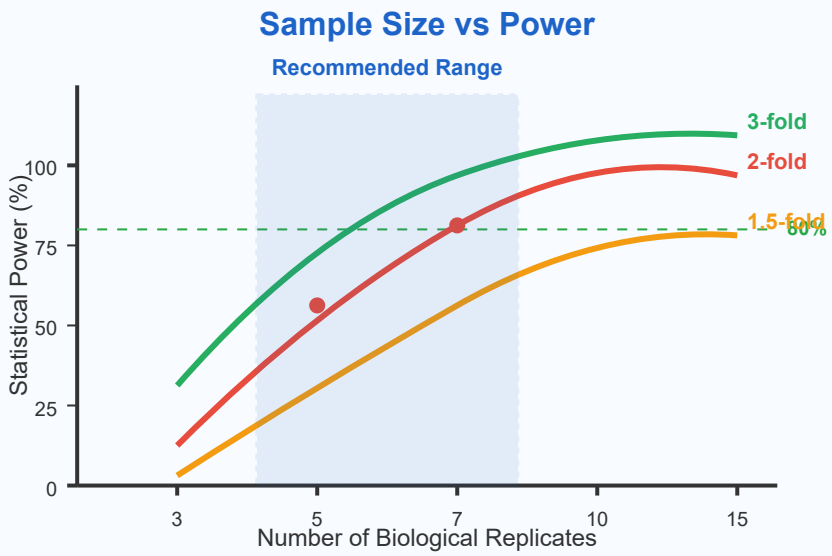
4 Power Analysis

Statistical power is the probability of detecting a true effect when it exists. Power analysis helps determine the appropriate sample size needed to detect biologically meaningful differences with acceptable confidence.

Key factors affecting power:

- **Effect size:** The magnitude of difference you want to detect (e.g., 2-fold change)
- **Biological variability:** Higher variability requires more samples
- **Sample size:** More replicates increase power
- **Significance level (α):** Typically 0.05, adjusted for multiple testing
- **Sequencing depth:** Affects detection of lowly expressed genes

```
Target Power: 80% (β = 0.20) at FDR < 0.05
```



Power Analysis Recommendations

- **Perform power analysis before starting:** Use tools like RNASeqPower, PROPER, or Scotty

- **Use pilot data when available:** Estimate variability from preliminary experiments
- **Consider effect size:** Smaller effects require more samples for detection
- **Account for multiple testing:** Adjust significance threshold for thousands of genes tested
- **Balance depth vs replicates:** More replicates usually better than extreme depth
- **Plan for dropout:** Include extra samples in case of technical failures

Fold Change	Variability (CV)	Samples Needed (80% power)
2-fold	Low (20%)	3-4 per group
2-fold	Moderate (40%)	5-6 per group
1.5-fold	Low (20%)	6-8 per group
1.5-fold	Moderate (40%)	12-15 per group

Rule of Thumb: For detecting 2-fold changes in moderately variable genes, aim for 5-6 biological replicates per condition. This provides good power while remaining cost-effective.

5 Cost Optimization

RNA-seq experiments involve significant costs, and optimizing the allocation of resources between sequencing depth and sample number is crucial for maximizing statistical power within budget constraints.

Key cost considerations:

- **Sequencing depth:** Number of reads per sample
- **Sample number:** Number of biological replicates
- **Library preparation:** Often the most expensive per-sample cost
- **Platform choice:** NovaSeq vs NextSeq vs other platforms
- **Multiplexing:** Pooling multiple samples per lane

```
Recommended: 20-30M reads per sample for standard DE analysis
```

Cost Optimization Strategies

- **Prioritize sample number:** 6 samples at 25M reads beats 3 samples at 100M reads
- **Reduce unnecessary depth:** Beyond 30M reads, returns diminish for standard DE analysis

- **Maximize multiplexing:** Pool compatible samples to reduce per-sample sequencing costs
- **Choose appropriate platform:** Match platform to depth requirements (NovaSeq for many samples)
- **Consider pilot studies:** Small pilot can inform optimal depth for main study
- **Negotiate with core facilities:** Bulk pricing for large projects

Application	Recommended Depth	Rationale
Differential Expression (Human/Mouse)	20-30M reads	Sufficient for ~15K genes with good coverage
Novel Transcript Discovery	50-100M reads	Deeper coverage needed for rare transcripts
Allele-specific Expression	60-100M reads	Higher depth for statistical confidence
Single-cell RNA-seq	50-100K reads/cell	Shallow per cell, many cells
Microbial Transcriptomics	5-10M reads	Smaller transcriptomes require less depth

Cost-Effectiveness Principle: After adequate coverage (~20M reads for mammalian samples), investing in additional biological replicates provides much better statistical power per dollar than increasing sequencing depth.

🎯 Golden Rule of RNA-seq Design

More biological replicates with moderate sequencing depth (5-6 samples × 20-30M reads)
provides superior statistical power compared to
fewer samples with extreme depth (3 samples × 100M reads)

Remember: You cannot compute your way out of poor experimental design!

Library Preparation Methods for RNA-Seq

PolyA Selection

Enriches mRNA by capturing poly-adenylated transcripts

Ribosomal Depletion

Removes rRNA to capture all RNA types including non-coding

Strand Specificity

Preserves information about which DNA strand was transcribed

UMI Incorporation

Unique Molecular Identifiers enable accurate quantification

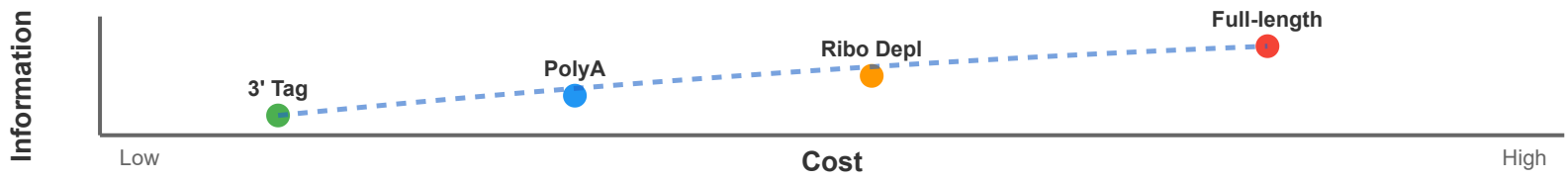
3' Tag-seq

Sequences only 3' ends - cost effective for counting

Full-length Coverage

Complete transcript coverage for isoform analysis

Trade-off: **Cost vs. Information Content**



Detailed Method Explanations

1 PolyA Selection

Overview

PolyA selection is the most commonly used method for mRNA enrichment in RNA-seq. It takes advantage of the poly-adenylated (polyA) tail present on most mature eukaryotic mRNAs. Oligo(dT) beads are used to bind and capture these polyA-tailed transcripts, effectively enriching for coding RNA while removing ribosomal RNA and other non-polyadenylated species.

Key Features

- Captures ~1-2% of total RNA (mostly mRNA)
- Removes ~90% of ribosomal RNA
- Standard method for gene expression studies
- Compatible with degraded samples (with limitations)

✓ Advantages

- Cost-effective
- Simple protocol

✗ Limitations

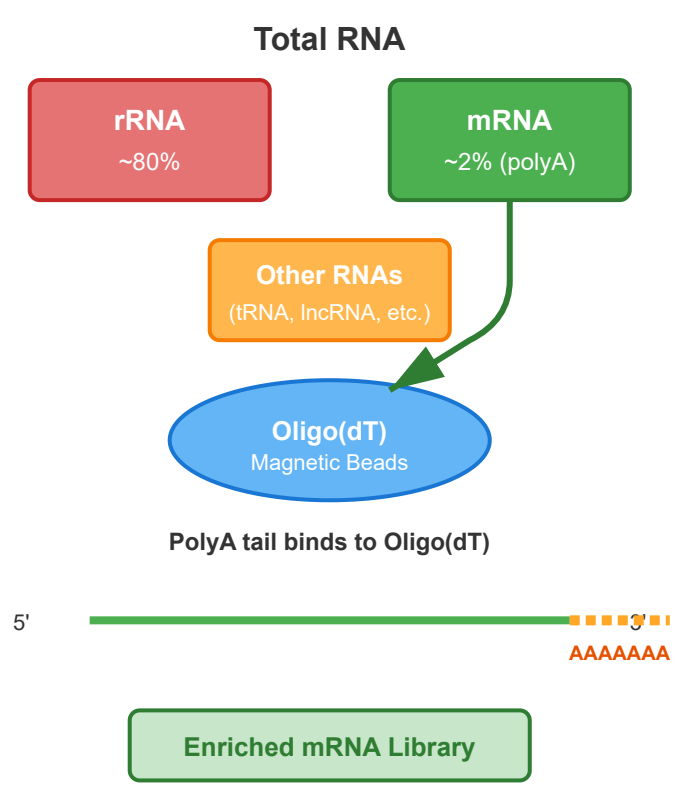
- Misses non-polyA RNAs

- High enrichment efficiency
- Well-established method

- 3' bias with degradation
- Excludes bacterial transcripts
- May lose histone mRNAs

Best Use Cases

Standard gene expression profiling, differential expression analysis, transcript quantification in high-quality eukaryotic samples, and when focusing exclusively on protein-coding genes.



2 Ribosomal RNA Depletion

Overview

Ribosomal depletion removes rRNA (which comprises 80-90% of total RNA) using sequence-specific probes that hybridize to ribosomal sequences. Unlike polyA selection, this method retains all other RNA types including non-polyadenylated transcripts, making it ideal for comprehensive transcriptome

analysis including bacterial samples, long non-coding RNAs, and degraded samples.

Key Features

- Removes rRNA while preserving all other RNA species
- Works with prokaryotic and eukaryotic samples
- Retains non-coding RNAs (lncRNA, circRNA, etc.)
- Better for degraded samples than polyA selection

✓ Advantages

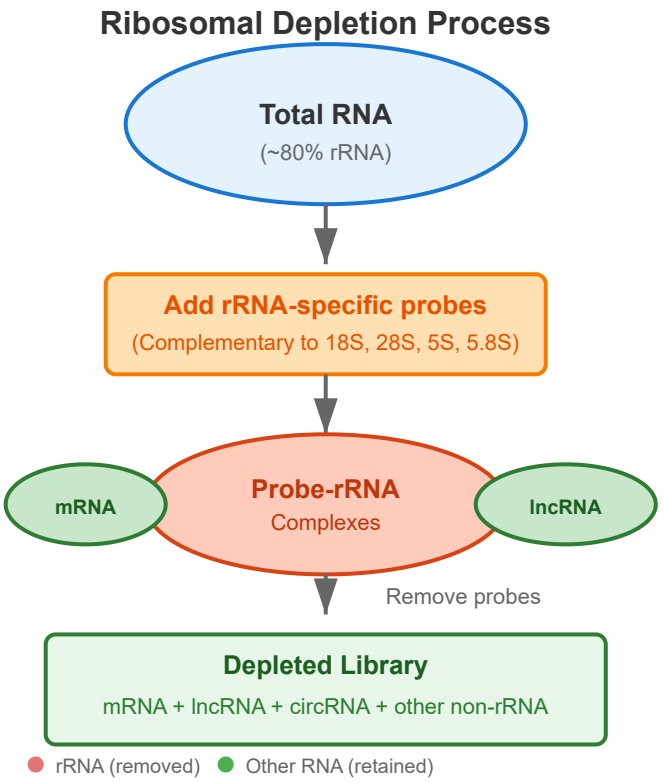
- Captures all RNA types
- No 3' bias
- Works with bacteria
- Better for FFPE samples

✗ Limitations

- More expensive
- Incomplete rRNA removal
- Complex data analysis
- Species-specific probes

Best Use Cases

Non-coding RNA studies, bacterial transcriptomics, degraded or FFPE samples, comprehensive transcriptome profiling including intronic and intergenic regions, and when polyA selection is not suitable.



3 Strand-Specific Library Preparation

Overview

Strand-specific (stranded) RNA-seq preserves information about which DNA strand a transcript originated from. This is crucial because genes can overlap, and antisense transcripts can regulate gene expression. The method typically uses dUTP incorporation during second-strand synthesis, which is later degraded, ensuring only the original strand is sequenced.

Key Features

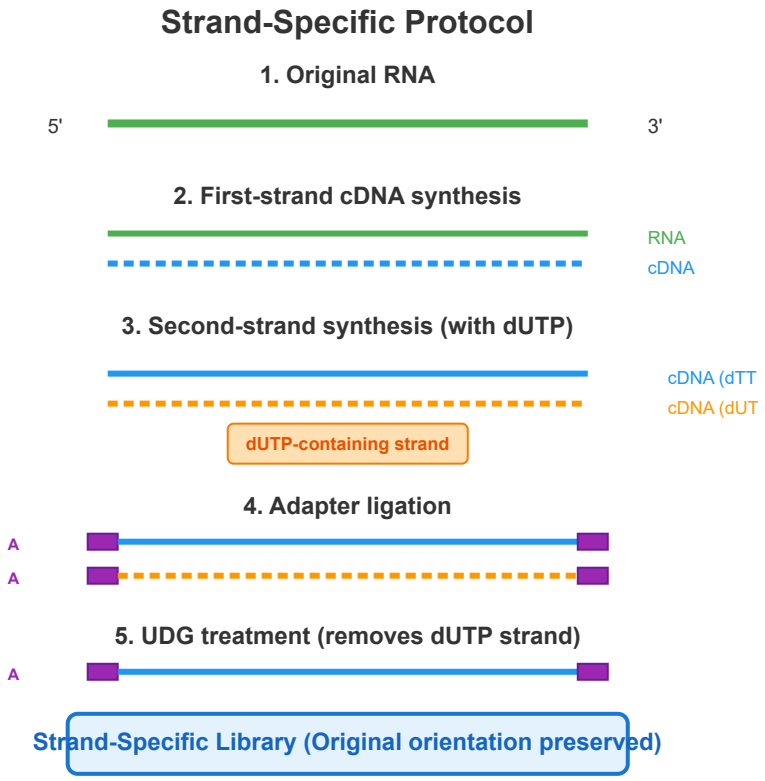
- Maintains strand-of-origin information
- Resolves overlapping genes on opposite strands
- Detects antisense transcription
- Improves accuracy in transcript quantification

✓ Advantages

- Distinguishes sense/antisense
- Better annotation accuracy
- Identifies regulatory RNAs

✗ Limitations

- Slightly higher cost
- More complex protocol
- Additional processing step
- Requires compatible



- Resolves overlapping genes

analysis

Best Use Cases

Discovery of antisense transcripts, analysis of overlapping gene pairs, regulatory RNA studies, transcript annotation improvement, and any study requiring precise strand information for complex genomic regions.

4 UMI (Unique Molecular Identifier) Incorporation

Overview

UMIs are short random nucleotide sequences (typically 8-12 bp) added to each RNA molecule before amplification. During sequencing, PCR duplicates can be identified and removed by their identical UMI sequences, allowing accurate counting of original molecules. This dramatically improves quantification accuracy by distinguishing biological duplicates from PCR artifacts.

Key Features

- Random barcodes tag each individual molecule
- Enables removal of PCR duplicates

- Provides absolute molecule counting
- Essential for single-cell RNA-seq

✓ Advantages

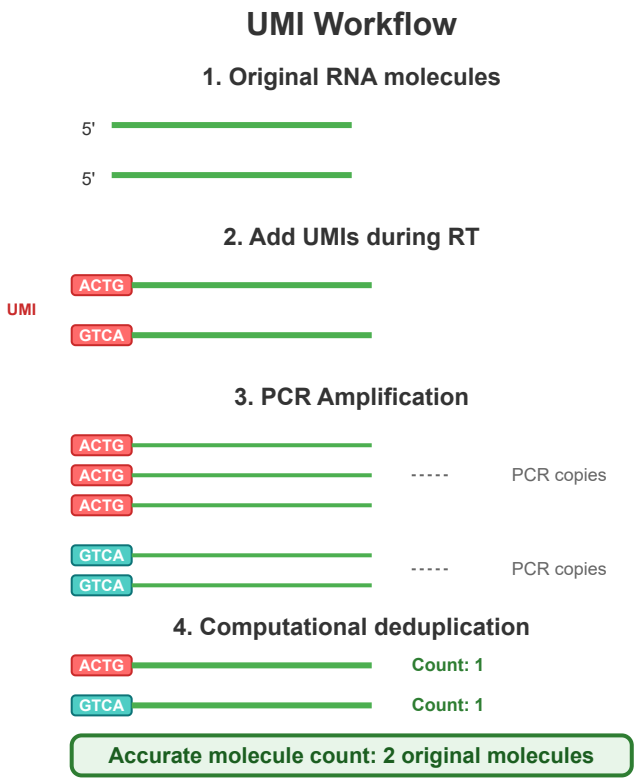
- Accurate quantification
- Removes amplification bias
- Better for low-input samples
- Improves reproducibility

✗ Limitations

- Increases sequencing cost
- Requires specialized analysis
- May reduce mapping rate
- UMI collisions possible

Best Use Cases

Single-cell RNA-seq, low-input RNA samples, precise quantification studies, detection of rare transcripts, clinical applications requiring high accuracy, and any experiment where PCR bias is a concern.



5 3' Tag-seq (3' End Sequencing)

Overview

3' Tag-seq sequences only the 3' end of transcripts (typically 50-100 bp from the polyA tail), making it highly cost-effective for gene expression quantification. Since most genes can be identified from their 3' UTR, this method provides accurate counting at a fraction of the cost of full-length sequencing. It's ideal for differential expression studies where transcript structure information is not needed.

Key Features

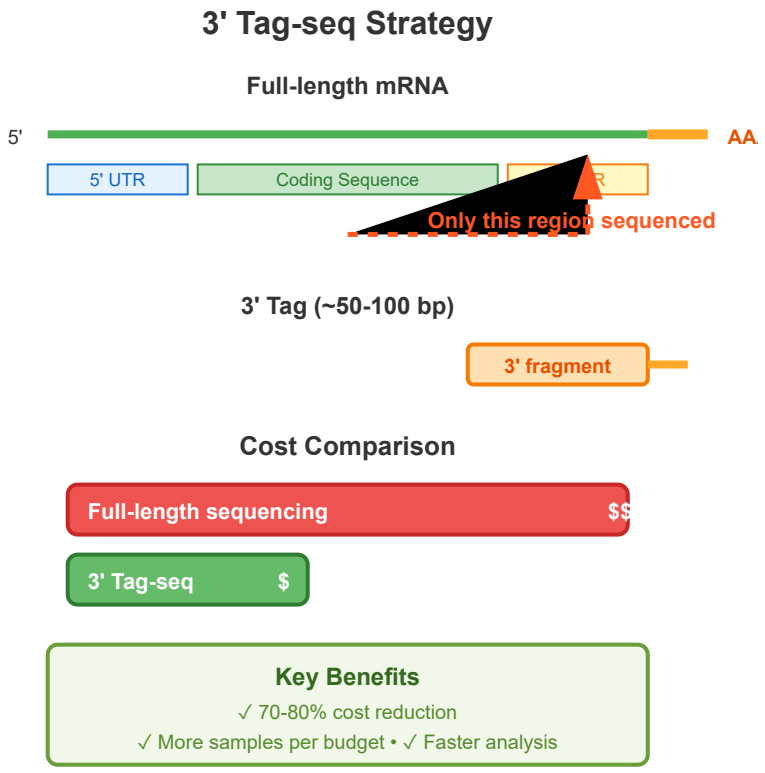
- Sequences only 3' terminal fragments
- Significantly reduced sequencing costs
- Maintains accurate gene-level quantification
- Compatible with UMIs for enhanced accuracy

✓ Advantages

- Highly cost-effective
- Less sequencing depth needed
- Good for large sample sets
- Robust to 5' degradation

✗ Limitations

- No isoform information
- Cannot detect SNPs
- Limited to gene counting
- Requires good 3' annotation



Best Use Cases

Large-scale differential expression studies, population-level gene expression screening, cost-constrained experiments, single-cell RNA-seq, studies focusing solely on gene abundance without requiring isoform or variant information.

6 Full-length Transcript Coverage

Overview

Full-length RNA-seq captures the entire length of transcripts from 5' to 3' end, providing comprehensive information about transcript structure, isoforms, and sequence variants. This method uses specialized protocols to minimize 3' bias and ensure uniform coverage across the entire transcript length, enabling detection of alternative splicing, fusion genes, and RNA editing events.

Key Features

- Complete transcript coverage (5' to 3')
- Detects alternative splicing and isoforms
- Identifies SNPs and RNA editing sites
- Enables allele-specific expression analysis

✓ Advantages

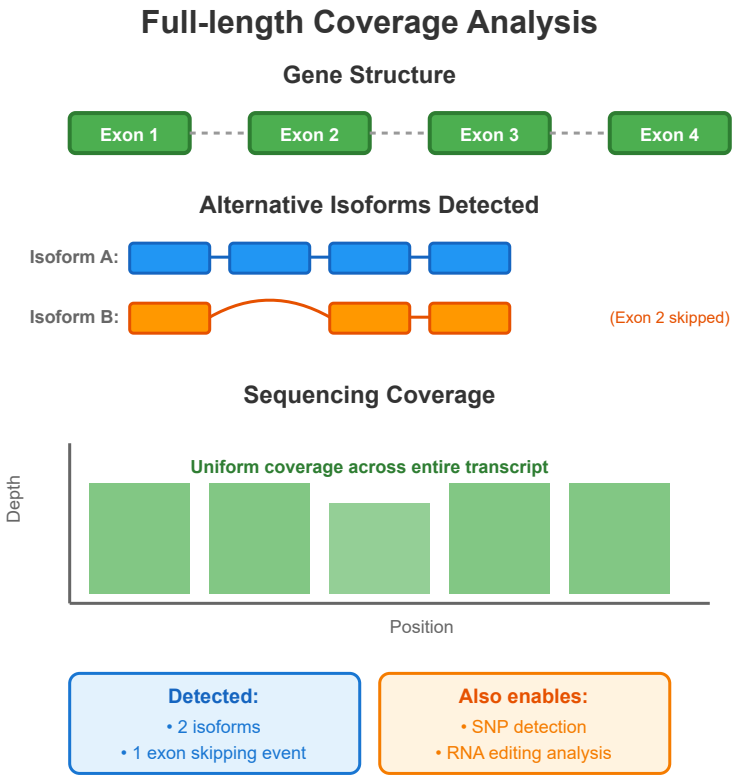
- Complete transcript information
- Isoform identification
- Detects fusion transcripts
- Variant calling capability

✗ Limitations

- Most expensive method
- Requires high sequencing depth
- Complex data analysis
- Needs high-quality RNA

Best Use Cases

Alternative splicing analysis, novel isoform discovery, fusion gene detection in cancer, RNA editing studies, allele-specific expression, comprehensive transcriptome annotation, and any research requiring complete transcript-level information.



Normalization Methods in RNA-Seq Analysis

RPKM/FPKM Issues

Reads/Fragments Per Kilobase Million - biased by composition

TPM Calculation

Transcripts Per Million - better for comparison

DESeq2 Normalization

Median-of-ratios method for differential expression

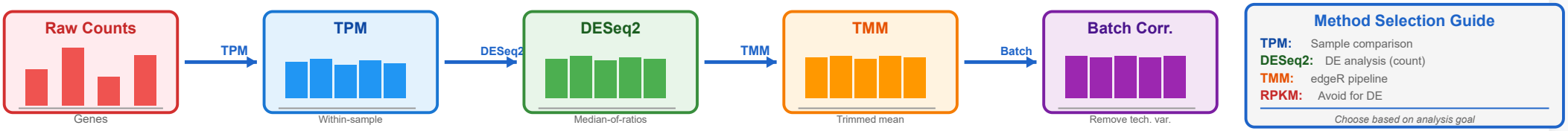
TMM Method

Trimmed Mean of M-values - robust to outliers

Batch Correction

ComBat, limma removeBatchEffect for technical variation

💡 Choose normalization method based on your downstream analysis goals



Detailed Normalization Methods

1 RPKM/FPKM - Reads/Fragments Per Kilobase Million

RPKM (Reads Per Kilobase Million) and FPKM (Fragments Per Kilobase Million) were among the first normalization methods developed for RNA-seq data. They normalize for both sequencing depth and gene length.

Formula

```
RPKM = (Number of reads mapped to gene × 10⁹) /
(Total mapped reads × Gene length in bp)

FPKM uses fragments instead of reads (for paired-
end)
```

Key Concept

Normalizes by scaling counts to account for differences in library size and gene length. The method assumes that most genes are not differentially expressed.

✓ Advantages

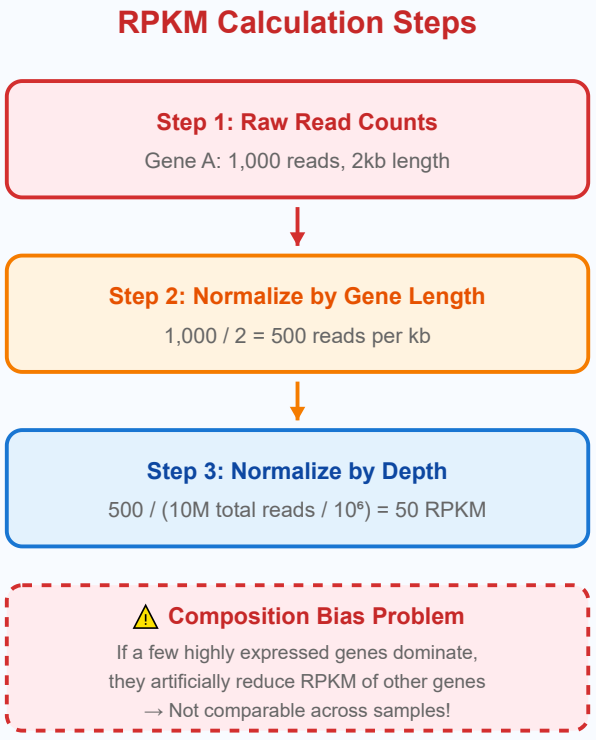
- Intuitive and simple
- Accounts for gene length
- Early standard method

✗ Disadvantages

- Composition bias issue
- Not suitable for DE analysis
- Sample-specific biases
- Values don't sum consistently

⚠ Usage Recommendation

Avoid RPKM/FPKM for differential expression analysis. Better alternatives like TPM, DESeq2, and edgeR are now standard. May still be used for within-sample comparisons or legacy pipelines.



2 TPM - Transcripts Per Million

TPM (Transcripts Per Million) is an improved normalization method that addresses the composition bias issues of RPKM/FPKM. The key difference is the order of normalization steps.

Formula

```
Step 1: RPK = Reads / (Gene length in kb)
Step 2: Sum all RPK values in sample
Step 3: TPM = (RPK / Sum of all RPK) × 10⁶
```

Key Advantage

TPM values always sum to the same total (1 million) in each sample, making cross-sample comparisons more meaningful. The sum of all TPM values in a sample is constant, unlike RPKM.

✓ Advantages

- Better for cross-sample comparison
- Sums to same value per sample
- Reduces composition bias
- Interpretable proportions

✗ Disadvantages

- Still not ideal for DE testing
- Assumes constant RNA output
- Requires gene length info

✓ Usage Recommendation

Excellent for comparing expression levels across samples, visualization, and reporting relative abundances. Preferred over RPKM/FPKM for most visualization purposes.

TPM vs RPKM Comparison

✓ Key Benefit: Consistent Totals

TPM always sums to 1M, enabling direct comparison

TPM Calculation Order

1. Normalize by gene length → RPK
2. Calculate sum of all RPK values
3. Scale to 1 million → TPM

(Order matters! Different from RPKM)

3 DESeq2 - Median-of-Ratios Normalization

DESeq2 uses a sophisticated median-of-ratios method to estimate size factors for normalization. It's specifically designed for differential expression analysis with count data and handles biological variability effectively.

Algorithm Steps

```
1. Calculate geometric mean for each gene across samples
2. For each sample, calculate ratio to geometric mean
3. Take median of ratios → size factor
4. Divide raw counts by size factor
```

Key Innovation

Uses a negative binomial model to account for biological variability. Robust to genes with zero counts and doesn't require gene length information. Automatically filters out genes with all zeros or extreme outliers.

✓ Advantages

- Gold standard for DE analysis
- Handles biological replicates well
- Robust to outliers
- Built-in statistical testing
- No gene length needed

✗ Disadvantages

- Requires raw count data
- Computationally intensive
- Needs biological replicates

✓ Usage Recommendation

DESeq2 Size Factor Calculation

Example: 3 Genes × 3 Samples

Gene	S1	S2	S3	GeoMean
A	100	200	150	145
B	50	100	75	71
C	25	50	37	36

Step 1: Calculate Ratios to Geometric Mean

S1: [100/145, 50/71, 25/36] = [0.69, 0.70, 0.69]
S2: [200/145, 100/71, 50/36] = [1.38, 1.41, 1.39]

Step 2: Take Median → Size Factors

S1 size factor = median([0.69, 0.70, 0.69]) = 0.69
S2 size factor = median([1.38, 1.41, 1.39]) = 1.39

Result: Normalized Counts

Raw counts divided by size factor
• Accounts for library size differences
• Robust to highly expressed genes
• Ready for statistical DE testing

The preferred method for differential gene expression analysis. Use when you have raw count data with biological replicates and want to identify statistically significant changes between conditions.

4 TMM - Trimmed Mean of M-values

TMM (Trimmed Mean of M-values) is the normalization method used by edgeR. It calculates scaling factors based on the weighted trimmed mean of log-ratios between samples, making it highly robust to outliers and composition biases.

Algorithm

```
M-values = log₂(Sample / Reference)
A-values = ½ × log₂(Sample × Reference)
Trim extreme M and A values (default: 30% & 5%)
Calculate weighted mean → TMM factor
```

Key Features

The method trims both extreme fold-changes (M) and extreme absolute expression levels (A), then calculates a weighted mean. This makes it exceptionally robust to outliers and genes with very high or low expression.

✓ Advantages

- Very robust to outliers

✗ Disadvantages

- Needs reference sample selection

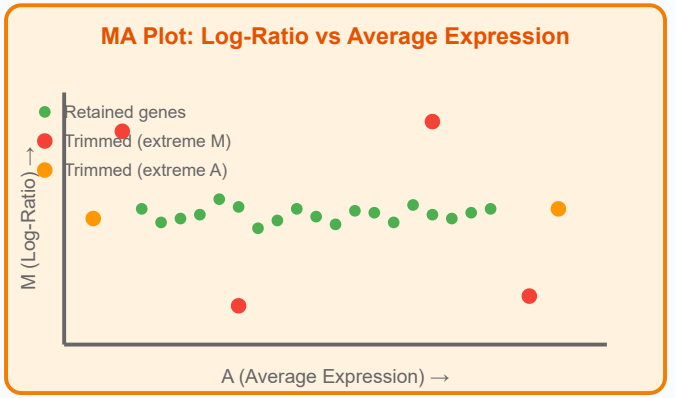
- Handles composition bias well
- Works with edgeR pipeline
- Fast computation
- Effective for many scenarios

- Assumptions may fail with extreme DE
- Less intuitive than DESeq2

✓ Usage Recommendation

Excellent choice for differential expression with edgeR. Particularly good when you have concerns about outliers or composition bias. Often produces similar results to DESeq2 but with faster computation.

TMM Trimming Strategy



Trimming Parameters

- M-trim: Remove top/bottom 30% by fold-change
- A-trim: Remove top/bottom 5% by expression
- Calculate weighted mean of remaining genes
- Weighting reduces influence of low-count genes

Result: Robust normalization factors resistant to outliers

5 Batch Effect Correction

Batch effects are systematic non-biological variations in data caused by technical factors like different sequencing runs, dates, operators, or reagent lots. Batch correction methods remove these technical artifacts while preserving biological variation.

Common Methods

ComBat (sva package):
Empirical Bayes method to adjust for known batches

```
limma::removeBatchEffect:
Linear model-based batch removal

SVA (Surrogate Variable Analysis):
Identifies and removes unknown batch effects
```

Key Considerations

Batch correction should be applied AFTER normalization but BEFORE differential expression testing. Be careful not to remove biological signal along with batch effects. Always visualize data before and after correction using PCA plots.

✓ Advantages

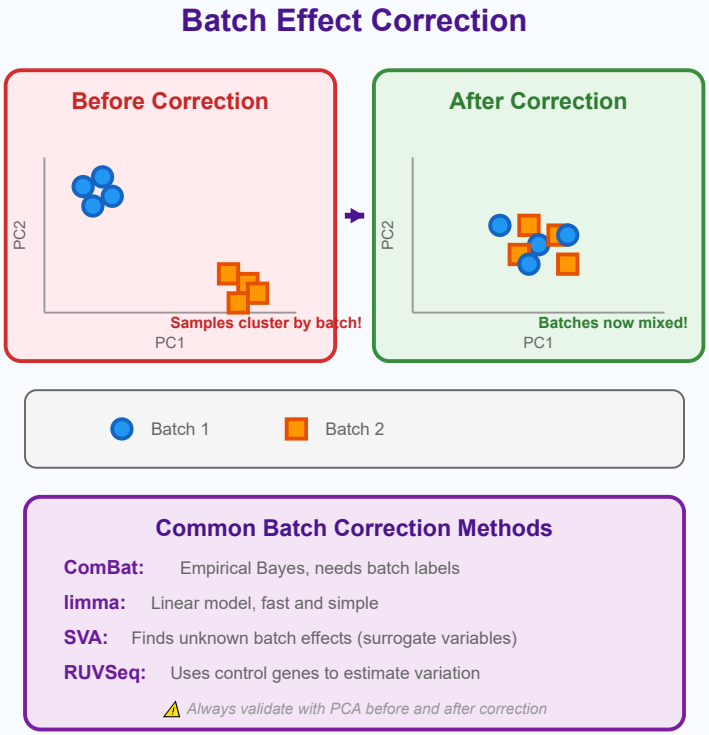
- Removes technical variation
- Improves signal detection
- Enables multi-cohort analysis
- Essential for meta-analysis

✗ Disadvantages

- Can remove biological signal
- Requires careful validation
- Batch info must be known (ComBat)
- May overcorrect

⚠ Usage Recommendation

Use batch correction when you have clear technical batches (different sequencing runs, dates, labs). Always validate with PCA plots. For DE analysis, consider including batch as a covariate in the model instead of pre-correcting.



📊 Quick Selection Guide

For Visualization & Comparison:

For Differential Expression:

Use **TPM** - consistent totals enable direct comparison between samples

Use **DESeq2** or **TMM/edgeR** - statistical models for count data

Avoid for DE Analysis:

RPKM/FPKM - composition bias makes them unsuitable for statistical testing

For Multi-Batch Experiments:

Apply **batch correction** after normalization, validate with PCA

Differential Expression

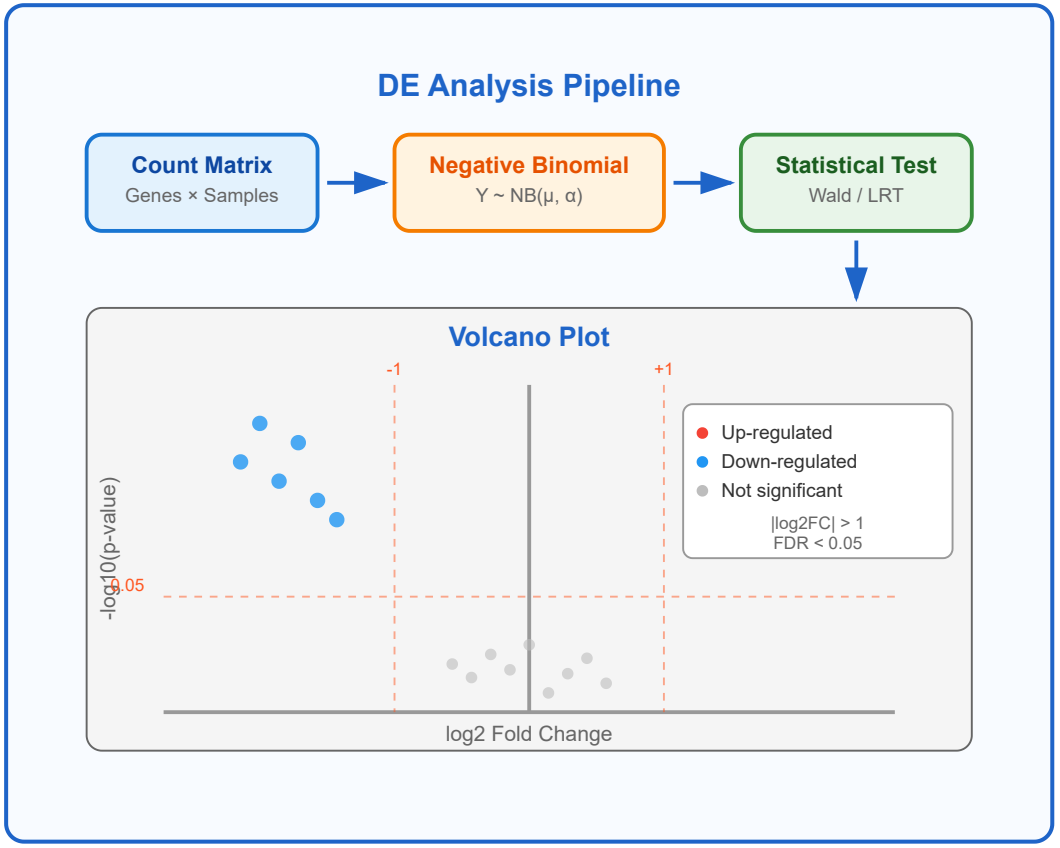
Statistical Models
Account for biological and technical variability

Negative Binomial
Models count data with overdispersion

Fold Change Thresholds
Typically |log2FC| > 1 for biological significance

FDR Control
False Discovery Rate < 0.05 for multiple testing

Volcano Plots
Visualize FC vs statistical significance

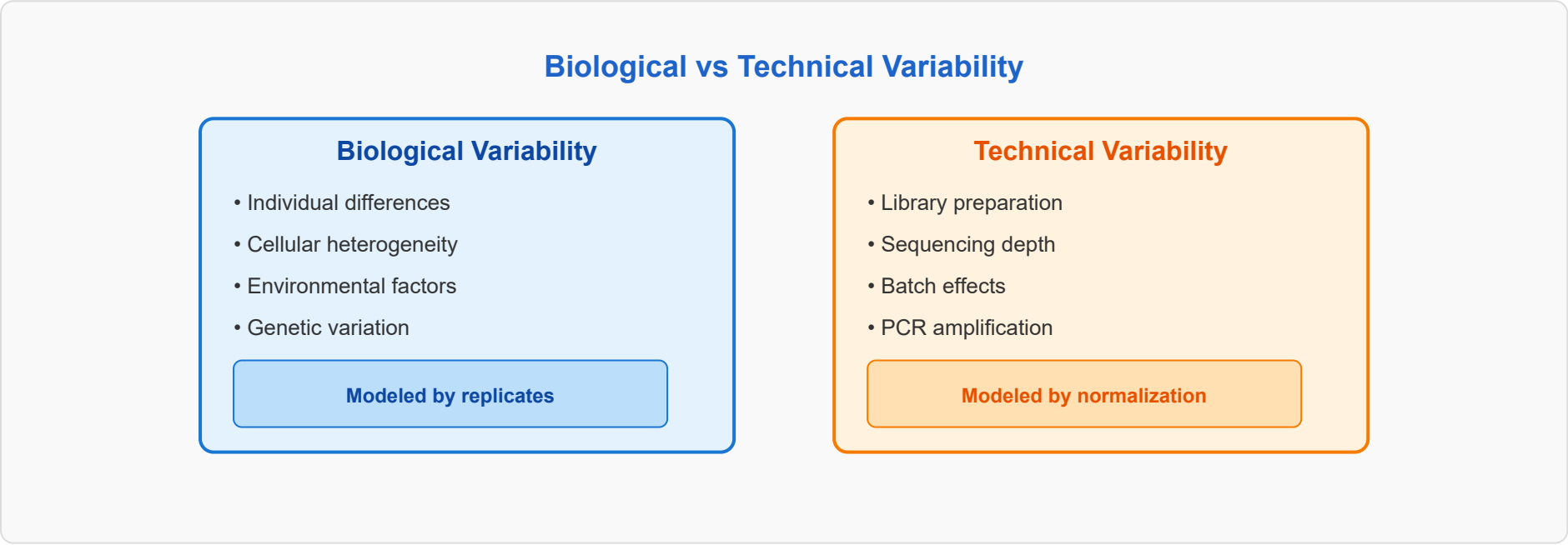


💡 Balance statistical significance with biological relevance

1. Statistical Models for Differential Expression

Statistical models form the foundation of differential expression analysis by providing a mathematical framework to distinguish true biological differences from random variation. In RNA-seq data, we encounter two major sources of variability that must be accounted for.

Sources of Variation



Why Standard Tests Fail

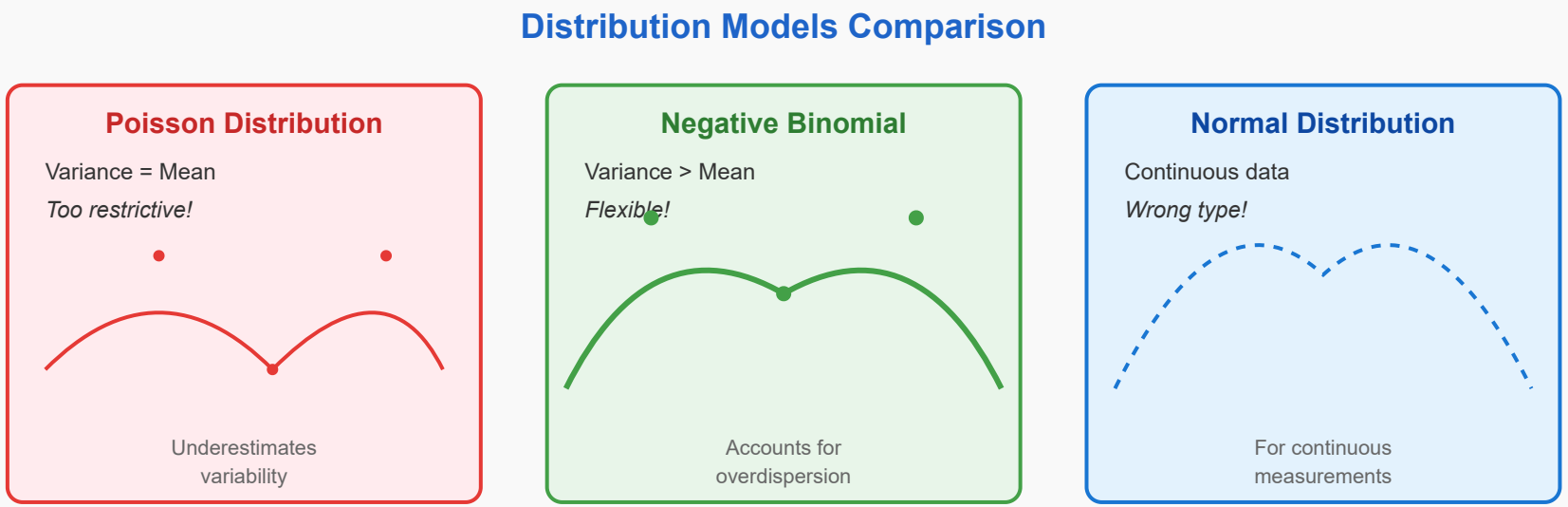
Simple t-tests or ANOVA are inadequate for RNA-seq data because they assume normally distributed data with constant variance. RNA-seq count data violates these assumptions in two critical ways:

Characteristics of RNA-seq count data:

- **Discrete counts** not continuous measurements
- **Mean-variance relationship** variance increases with expression level

- **Overdispersion** variance exceeds what Poisson distribution predicts
- **Zero inflation** many genes have zero counts in some samples

Model Comparison



Modern differential expression tools like DESeq2 and edgeR use **generalized linear models (GLMs)** with negative binomial distributions to properly model RNA-seq count data while accounting for both biological and technical variation.

2. Negative Binomial Distribution

The negative binomial distribution is the cornerstone of modern RNA-seq differential expression analysis. It extends the Poisson distribution by adding a dispersion parameter, allowing the variance to exceed the mean, which is characteristic of biological count data.

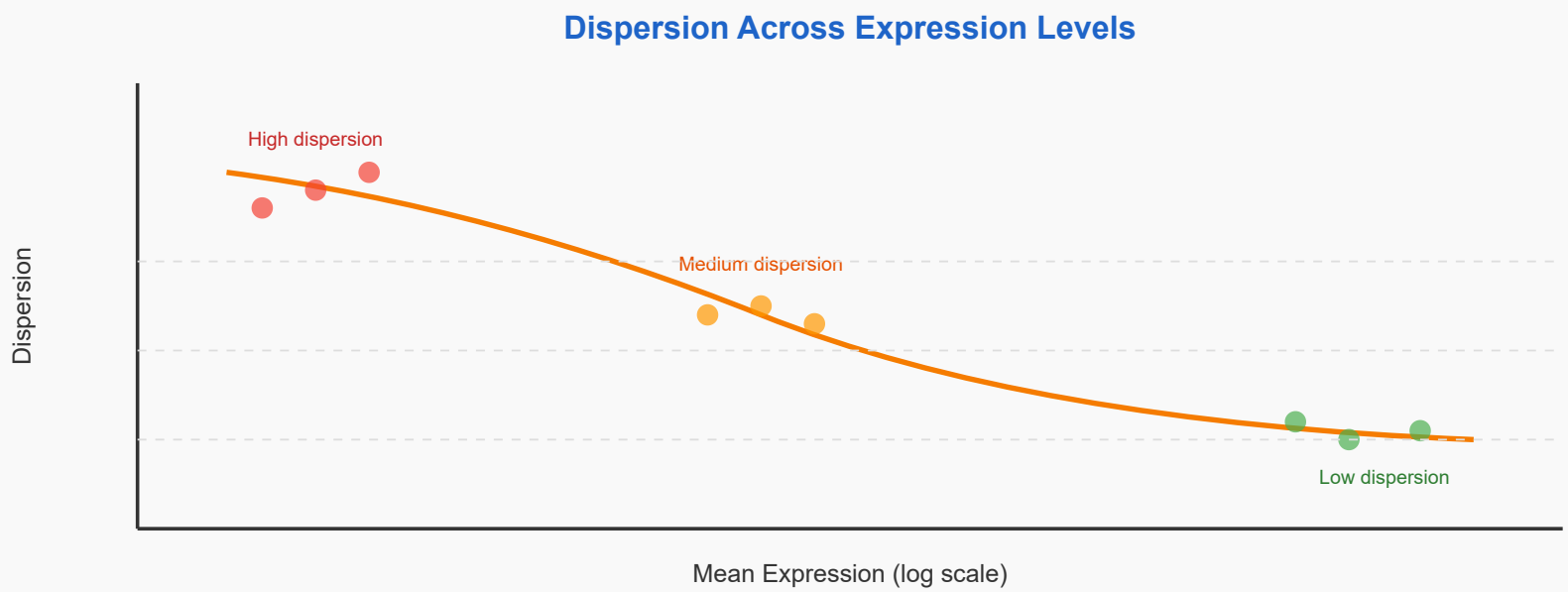
Negative Binomial Formula:

$$Y \sim NB(\mu, \alpha)$$

$$\text{Variance} = \mu + \alpha \times \mu^2$$

where μ = mean expression, α = dispersion parameter

Understanding Dispersion

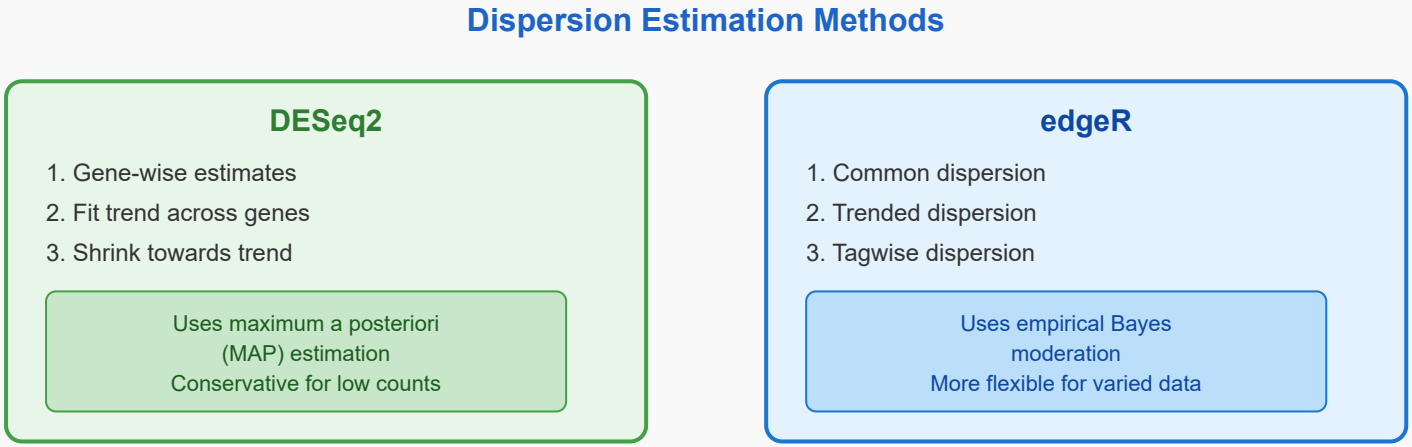


Key properties of dispersion:

- **Gene-specific** each gene has its own dispersion estimate

- **Expression-dependent** typically decreases with higher mean expression
- **Shrinkage estimation** improves accuracy by borrowing information across genes
- **Biological coefficient of variation** BCV = √α represents biological variability

DESeq2 vs edgeR Approaches



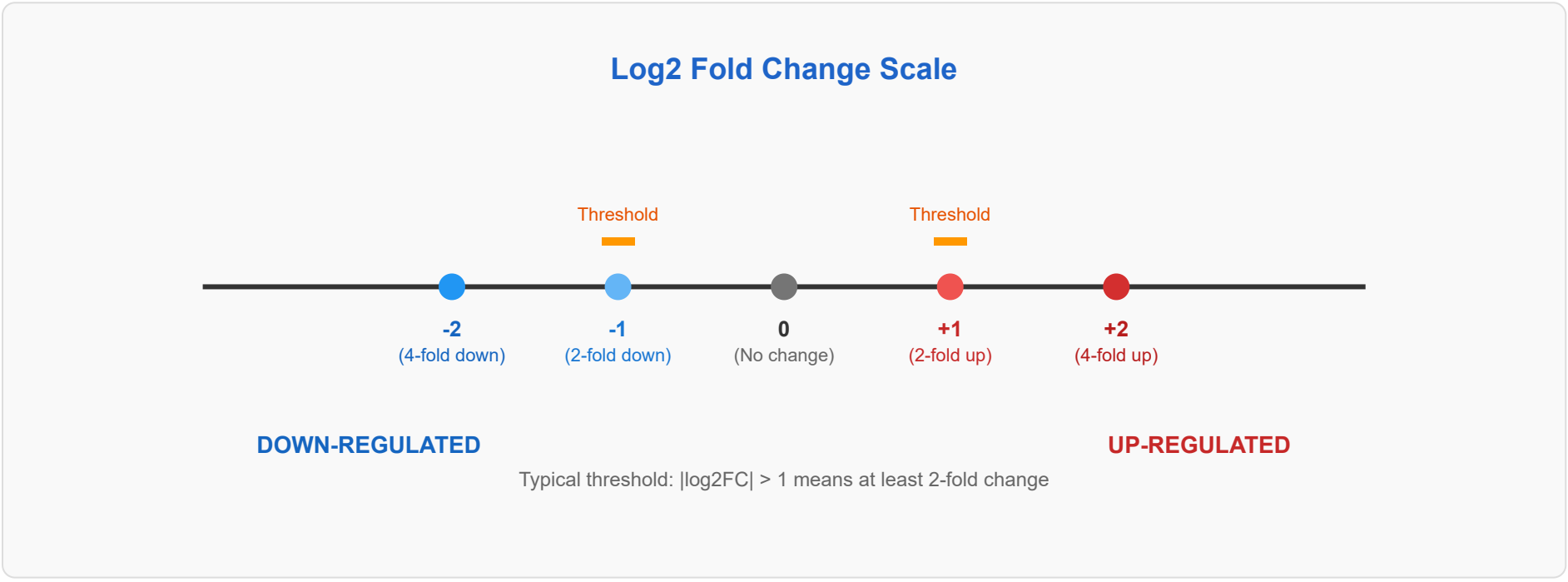

```
# Example R code for dispersion estimation in DESeq2 library(DESeq2) # Create DESeq2 object dds <-
DESeqDataSetFromMatrix( countData = counts, colData = metadata, design = ~ condition ) # Estimate size factors
(normalization) dds <- estimateSizeFactors(dds) # Estimate dispersions dds <- estimateDispersions(dds) # Plot
dispersion estimates plotDispEsts(dds)
```

Both methods share information across genes to improve dispersion estimates, especially for genes with low counts or few replicates. This **shrinkage approach** balances gene-specific estimates with overall trends, reducing false positives while maintaining statistical power.

3. Fold Change Thresholds

While statistical significance tells us that a difference is unlikely to be due to chance, fold change tells us whether that difference is **biologically meaningful**. A gene might be statistically significantly different but only change by 5%, which may not be biologically relevant.

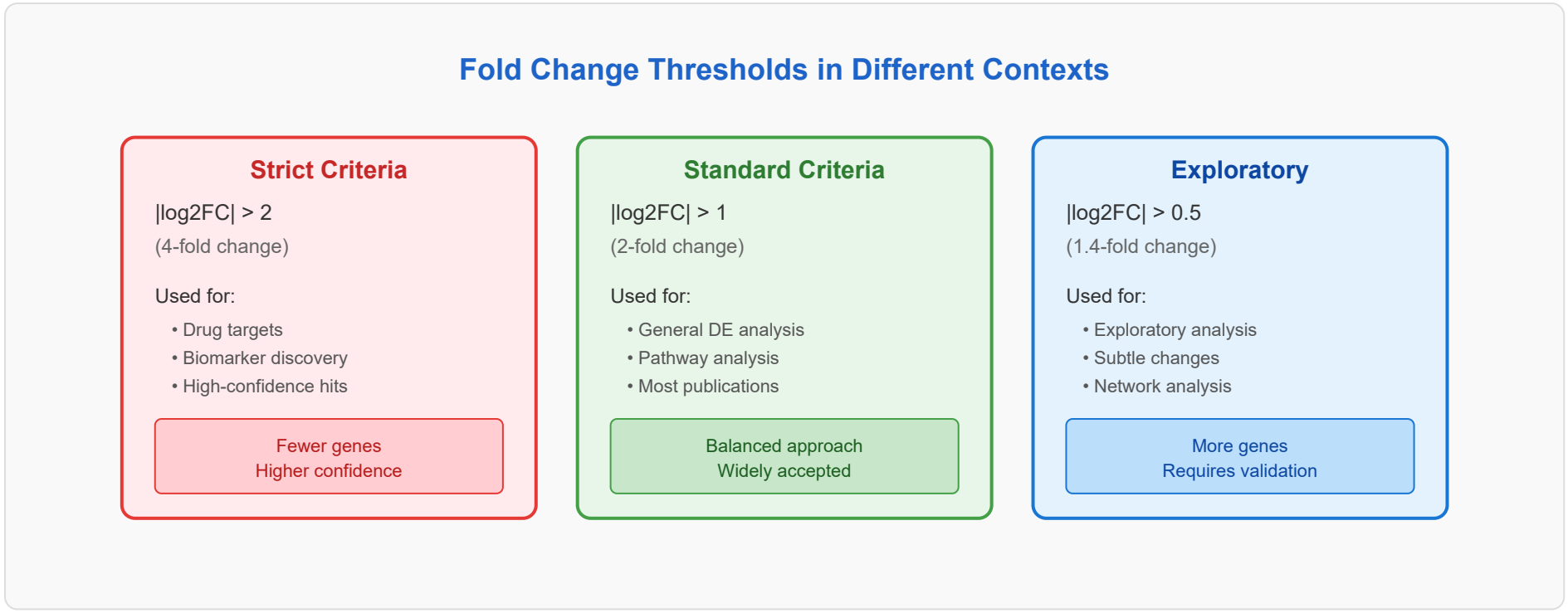
Understanding Log2 Fold Change



Why use log2 fold change?

- **Symmetry** up and down-regulation are symmetric (2-fold up = +1, 2-fold down = -1)
- **Interpretability** each unit represents a doubling or halving
- **Statistical properties** more normally distributed than raw fold changes
- **Easy comparison** magnitude directly comparable across genes

Common Thresholds by Context



Fold Change Calculations:

$$\text{Fold Change (FC)} = \text{Expression}_{\text{Treated}} / \text{Expression}_{\text{Control}}$$

$$\text{Log2 Fold Change} = \log_2(\text{Expression}_{\text{Treated}}) - \log_2(\text{Expression}_{\text{Control}})$$

Example: Gene with 100 counts in control, 400 in treated

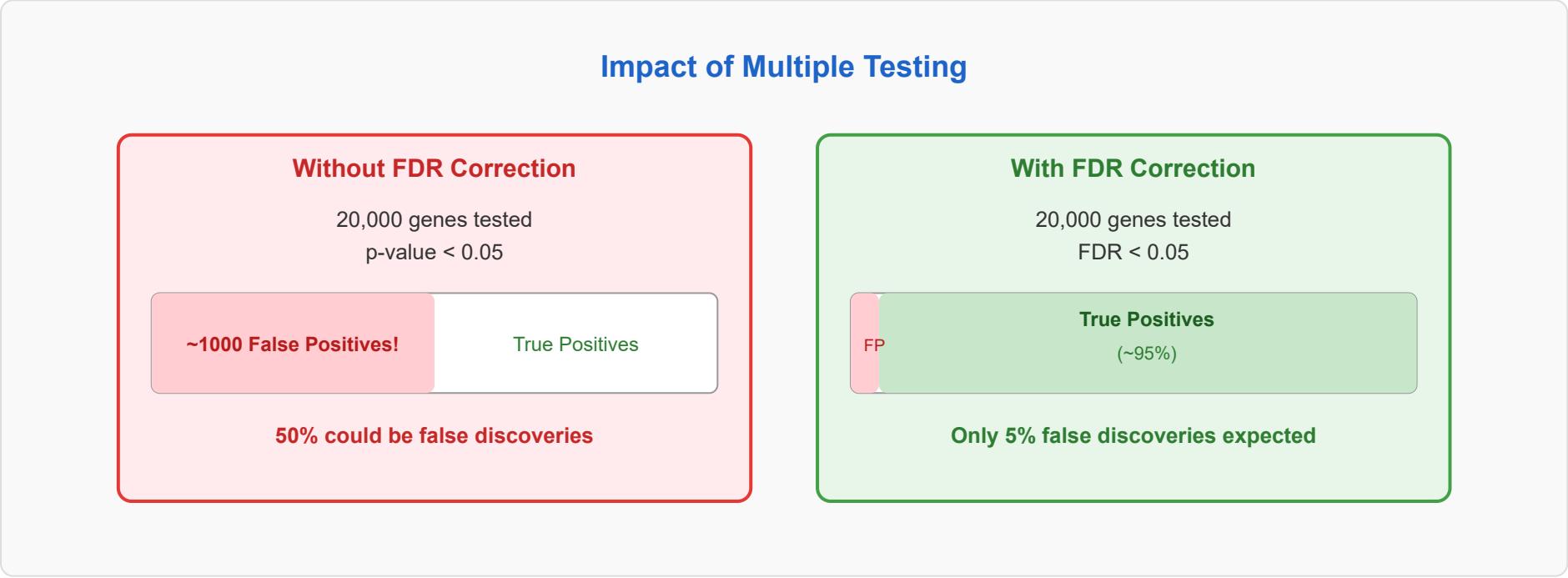
$$FC = 400/100 = 4 \rightarrow \text{log2FC} = \log_2(4) = 2$$

The choice of fold change threshold depends on your experimental context, validation capabilities, and downstream applications. Always consider both statistical significance and biological magnitude when interpreting results.

4. False Discovery Rate (FDR) Control

In RNA-seq experiments, we test thousands of genes simultaneously. This **multiple testing problem** means that even with a p-value threshold of 0.05, we would expect 5% of genes to appear significant by chance alone. For 20,000 genes, that's 1,000 false positives!

The Multiple Testing Problem



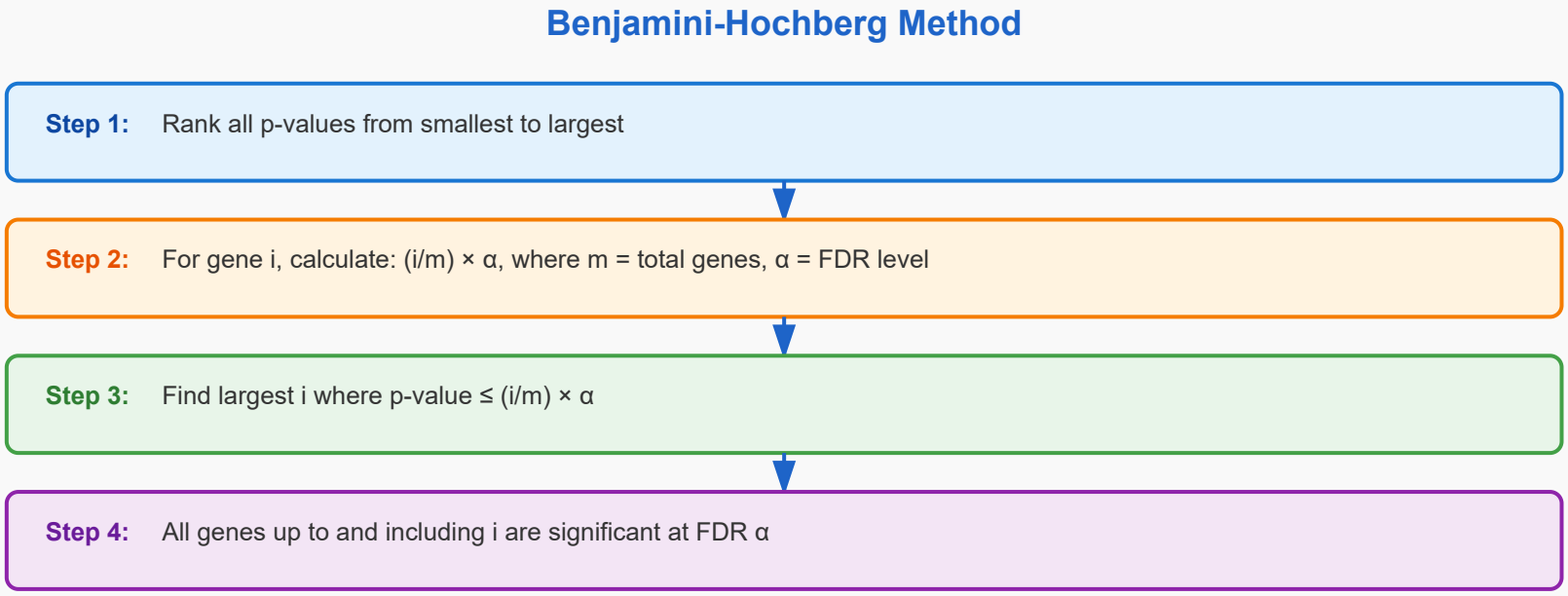
FDR vs P-value

Understanding the difference:

- **P-value** probability of observing data if null hypothesis is true (for a single test)
- **FDR (q-value)** expected proportion of false discoveries among all discoveries
- **Adjusted p-value** corrected p-value accounting for multiple comparisons

- **Benjamini-Hochberg** most common FDR control method in RNA-seq

Benjamini-Hochberg Procedure



Example: With 20,000 genes and α=0.05, the 100th gene needs p < (100/20000)×0.05 = 0.00025

```
# Example R code for FDR correction library(DESeq2) # Run differential expression analysis dds <- DESeq(dds) results
<- results(dds) # Results include both p-value and adjusted p-value (FDR) # padj is the Benjamini-Hochberg adjusted
p-value # Filter for significant genes significant_genes <- results[results$padj < 0.05 &
abs(results$log2FoldChange) > 1, ] # Count significant genes nrow(significant_genes)
```

FDR Interpretation:

FDR = 0.05 means:
"Among all genes called significant,
we expect about 5% to be false discoveries"

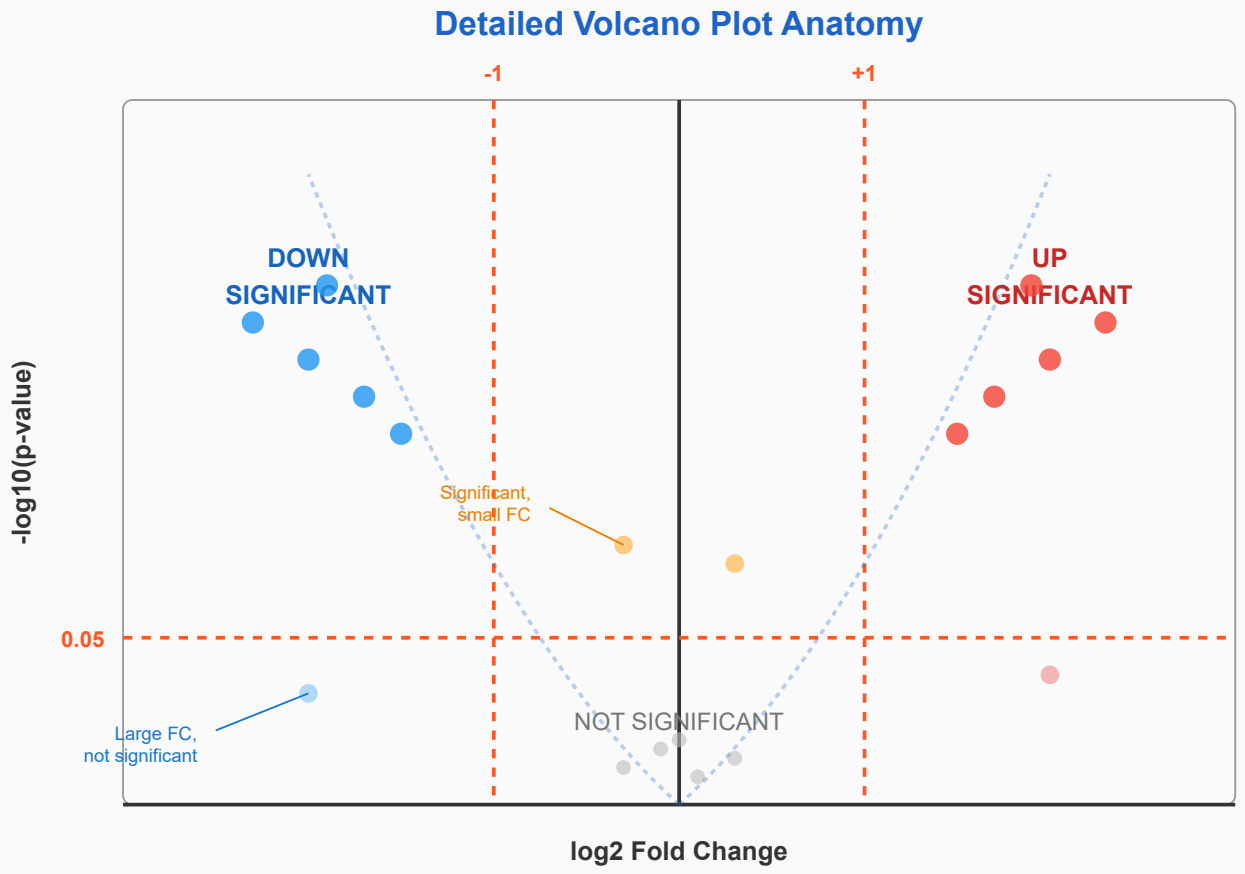
If 1000 genes are significant at FDR < 0.05,
expect ~50 false positives and ~950 true positives

FDR control is essential for reliable differential expression analysis. It provides a more interpretable measure than individual p-values when testing thousands of genes simultaneously, ensuring that your gene list has a predictable proportion of true discoveries.

5. Volcano Plots

Volcano plots are the most widely used visualization for differential expression results. They simultaneously display the **magnitude of change** (fold change) and **statistical confidence** (p-value) for all genes, making it easy to identify the most interesting candidates.

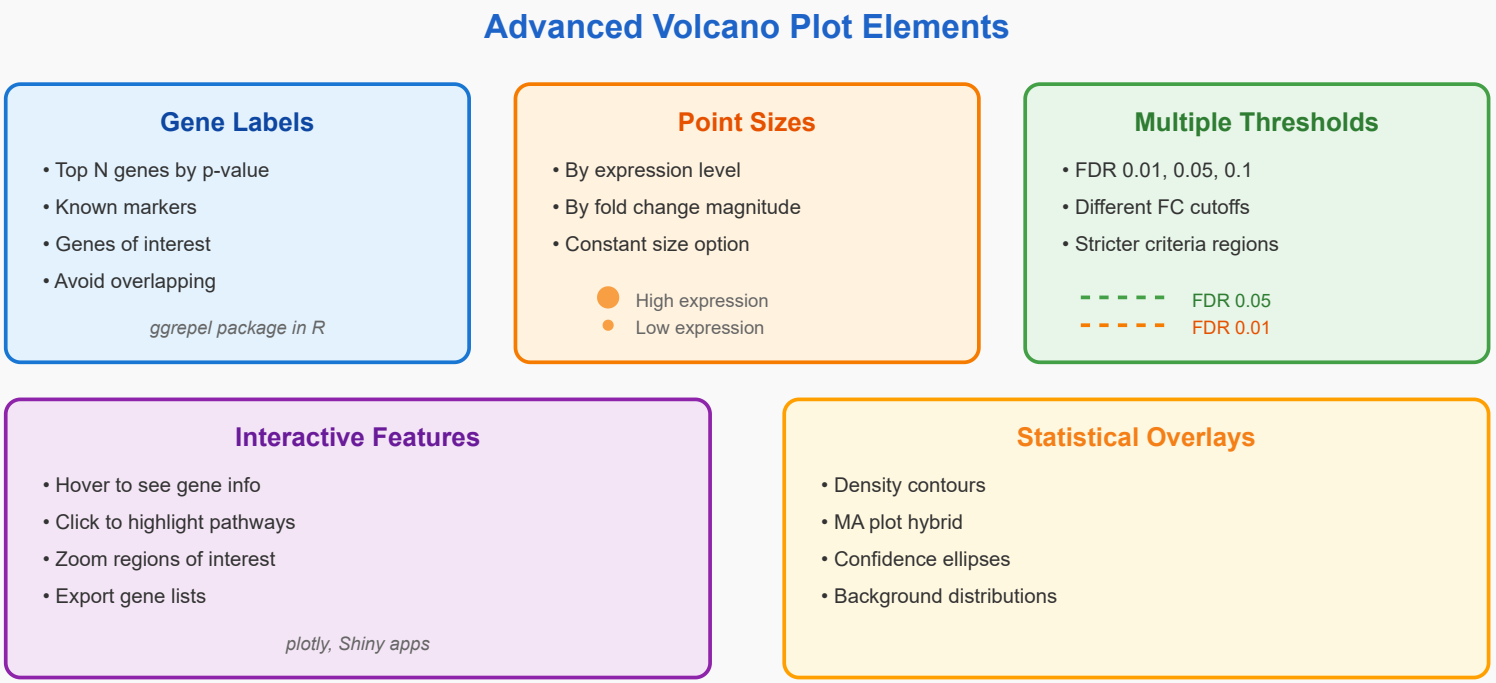
Anatomy of a Volcano Plot



Reading a volcano plot:

- **X-axis** log2 fold change (biological magnitude)
- **Y-axis** -log10(p-value) (statistical significance)
- **Top corners** most interesting genes (large change + significant)
- **Color coding** typically red/up, blue/down, grey/not significant

Enhanced Volcano Plot Features




```
# Creating an enhanced volcano plot in R library(ggplot2) library(ggrepel) # Prepare data volcano_data <- data.frame( gene = rownames(results), log2FC = results$log2FoldChange, pvalue = results$pvalue, padj = results$padj ) # Add significance category volcano_data$significance <- "Not Sig" volcano_data$significance[volcano_data$padj < 0.05 & volcano_data$log2FC > 1] <- "Up" volcano_data$significance[volcano_data$padj < 0.05 & volcano_data$log2FC < -1] <- "Down" # Create plot ggplot(volcano_data, aes(x = log2FC, y = -log10(pvalue))) + geom_point(aes(color = significance), alpha = 0.6, size = 2) + scale_color_manual(values = c("Up" = "#F44336", "Down" = "#2196F3", "Not Sig" = "#BDBDBD")) + geom_vline(xintercept = c(-1, 1), linetype = "dashed", color = "#FF5722") + geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "#FF5722") + labs(title = "Volcano Plot", x = "log2 Fold Change", y = "-log10(p-value)") + theme_minimal() + geom_text_repel(data = subset(volcano_data, padj < 0.001), aes(label = gene), size = 3, max.overlaps = 10)
```

Common Volcano Plot Patterns

Different experimental scenarios produce characteristic volcano plot shapes. A balanced experiment shows genes distributed symmetrically, while strong biological effects create distinct up and down-regulated clusters in the top corners. Technical issues or batch effects often manifest as asymmetric patterns or unexpected clustering.

Interpreting patterns:

- **Symmetric volcano** balanced regulation, good experiment
- **Skewed distribution** may indicate batch effects or normalization issues
- **Few significant genes** weak biological effect or insufficient power
- **Many significant genes** strong biological effect or technical artifact

Volcano plots are invaluable for quality control and hypothesis generation. They allow you to quickly assess the overall experimental results, identify candidate genes for follow-up studies, and communicate findings effectively in presentations and publications.

💡 **Complete Analysis: Combine all five components for robust differential expression analysis**

Statistical Testing in RNA-seq Analysis

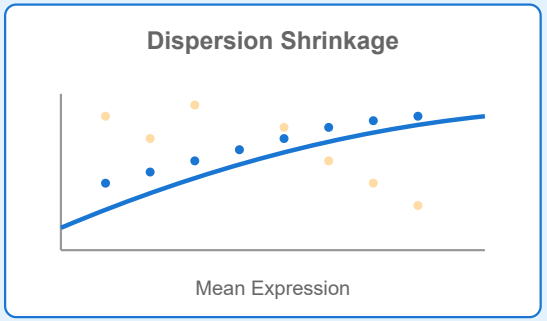
RNA-seq Statistical Methods Comparison

DESeq2

$Y \sim NB(\mu, \alpha)$

Key Features:

- Shrinkage estimation of dispersion
- Size factor normalization
- Wald test / LRT

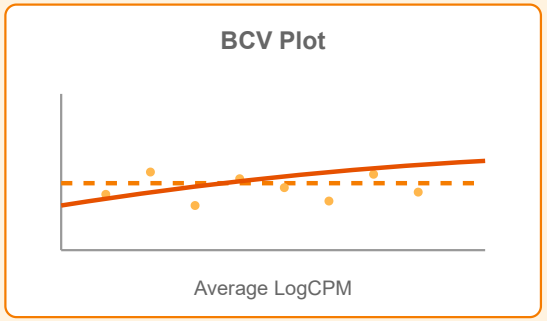


edgeR

$Y \sim NB(\mu, \varphi)$

Key Features:

- Empirical Bayes methods
- TMM normalization
- Quasi-likelihood F-test

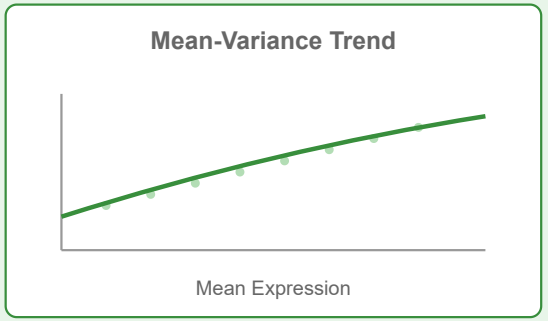


limma-voom

$\log(Y) \sim N(\mu, \sigma^2)$

Key Features:

- Transform to log-space
- Precision weights (voom)
- Linear modeling



Performance: All methods perform similarly with proper use • Choose based on experimental design and analysis goals

💡 DESeq2 and edgeR are most widely used and well-validated

📊 DESeq2 - Detailed Analysis

Overview

DESeq2 is a statistical method for differential gene expression analysis based on the negative binomial distribution. It uses shrinkage estimation to improve dispersion estimates, especially for genes with low counts or small sample sizes. DESeq2 is particularly robust for experiments with small sample sizes (3-5 replicates per condition).

Statistical Model

DESeq2 models RNA-seq count data using a negative binomial (NB) distribution:

$$K \sim NB(\mu, \alpha)$$

NB

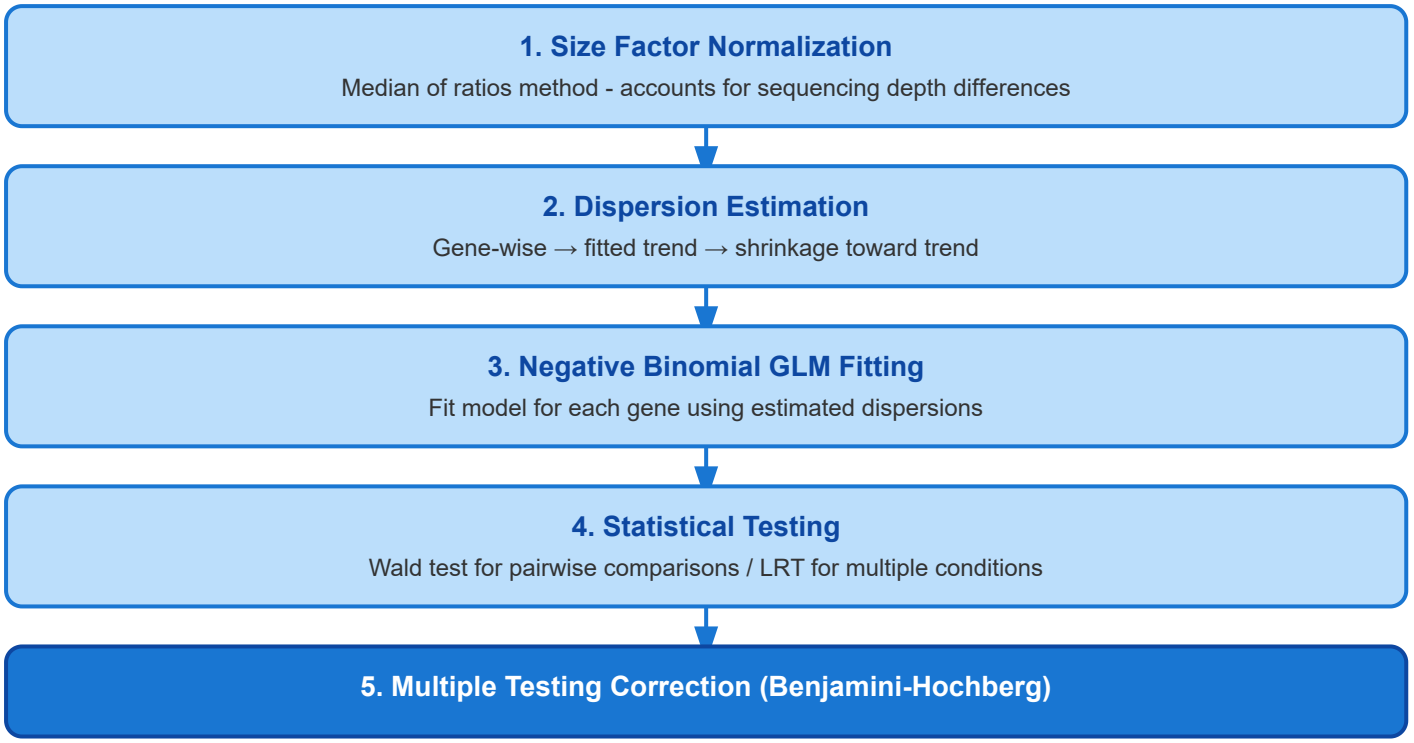
K = Read counts for a gene

μ = Mean expression level (affected by sequencing depth and biological condition)

α = Dispersion parameter (controls variance beyond mean)

The mean μ is modeled as: $\mu = s \times q$, where s is the size factor (normalization) and q is the expected count related to the biological condition.

Analysis Workflow



Key Features & Advantages

- **Shrinkage Estimation:** Borrows information across genes to improve dispersion estimates, particularly beneficial for genes with low counts. This reduces false positives while maintaining sensitivity.

- **Size Factor Normalization:** Uses geometric mean of ratios method, which is robust to outliers and doesn't assume most genes are unchanged between conditions.

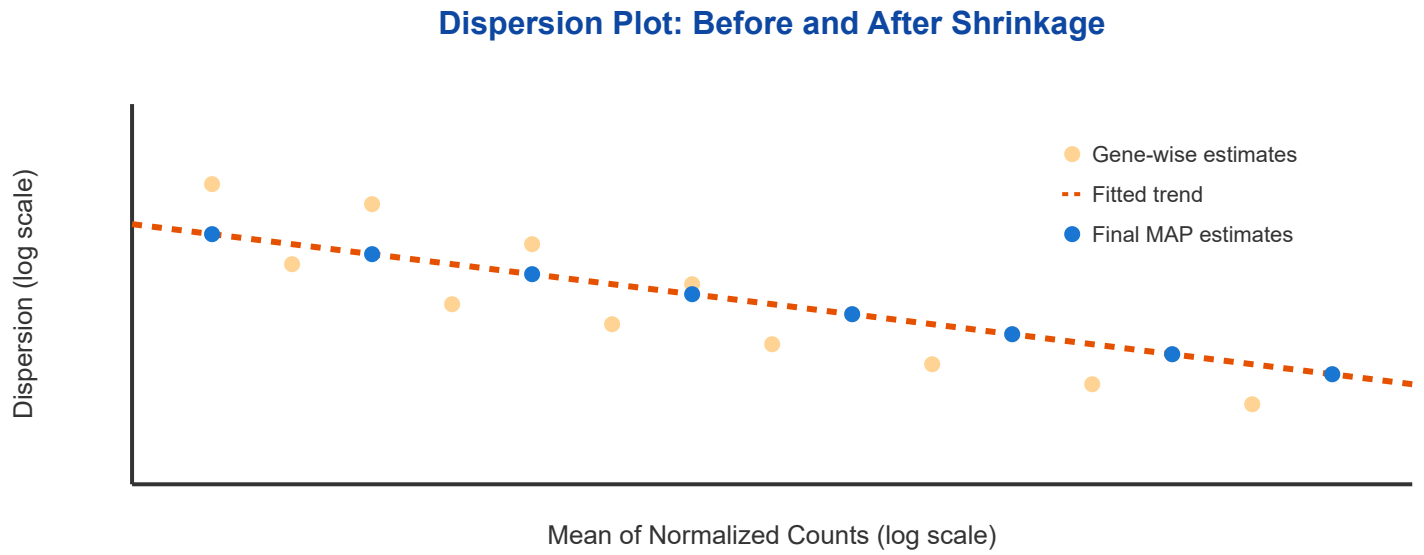
- **Small Sample Performance:** Performs well with as few as 2-3 replicates per condition, though 3-5 is recommended for reliable results.

- **Independent Filtering:** Automatically filters out low-count genes to increase statistical power for detecting truly differentially expressed genes.

Example R Code

```
# Load library library(DESeq2) # Create DESeq2 dataset dds <- DESeqDataSetFromMatrix( countData = counts, colData = metadata, design = ~ condition ) # Run DESeq2 analysis (all steps in one) dds <- DESeq(dds) # Extract results results <- results(dds, contrast = c("condition", "treated", "control")) # Apply shrinkage to log2 fold changes resultsLFC <- lfcShrink(dds, coef = "condition_treated_vs_control", type = "apeglm") # View summary summary(results)
```

Understanding Dispersion Shrinkage



Orange points represent initial gene-wise dispersion estimates (noisy, especially for low-count genes). **Red dashed line** shows the fitted trend across all genes. **Blue points** are the final maximum a posteriori (MAP) estimates after shrinkage toward the trend, providing more stable estimates.

📈 edgeR - Detailed Analysis

Overview

edgeR (empirical analysis of digital gene expression data in R) uses empirical Bayes methods to estimate biological variability. It's particularly well-suited for experiments with multiple factors and complex experimental designs. edgeR is known for its speed and

flexibility in handling various study designs.

Statistical Model

edgeR also uses the negative binomial distribution but parameterizes it slightly differently:

$$Y \sim NB(\mu, \varphi)$$

NB

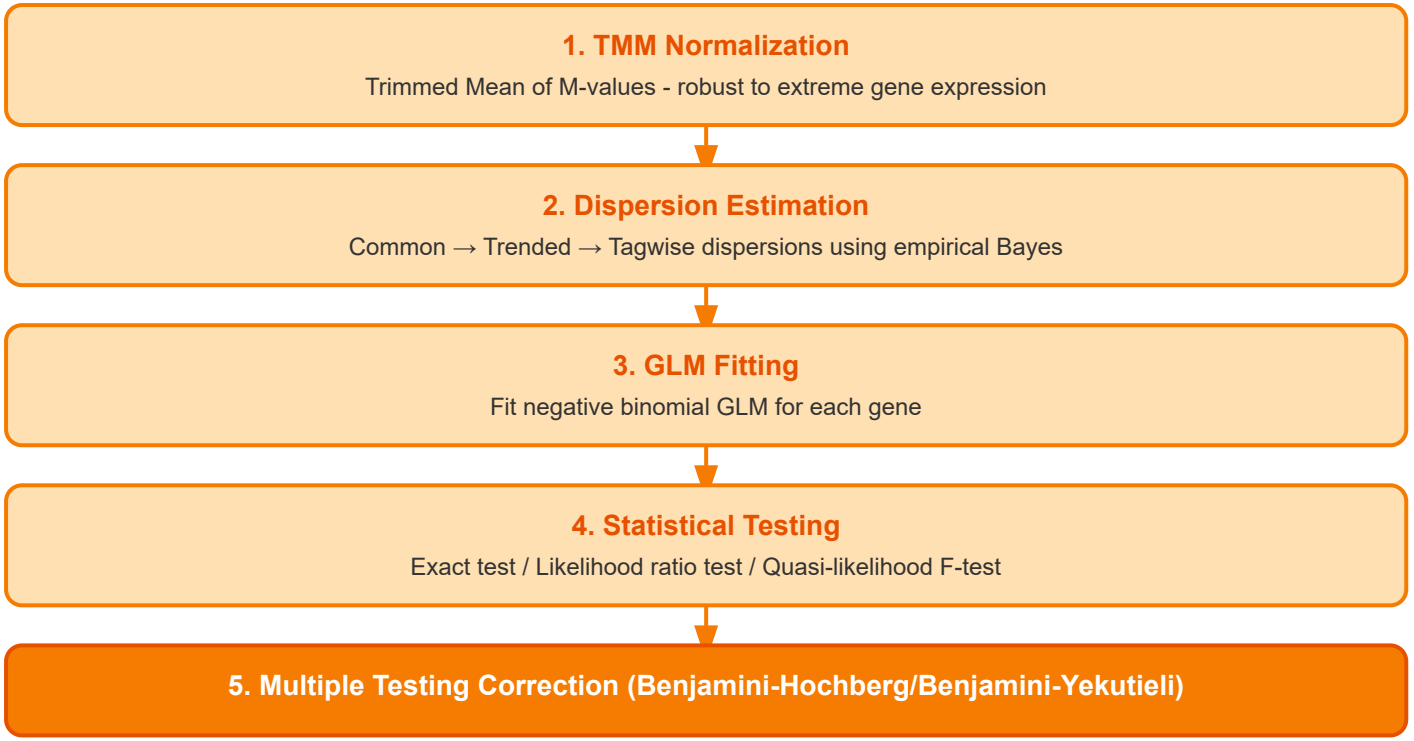
Y = Read counts for a gene

μ = Mean expression level

φ = Dispersion parameter (Variance = $\mu + \varphi\mu^2$)

edgeR models dispersion as a combination of common, trended, and tagwise (gene-specific) components, providing flexibility in capturing biological variability.

Analysis Workflow



Key Features & Advantages

- **TMM Normalization:** Trimmed Mean of M-values is particularly robust when a small proportion of genes are very highly expressed or when there are compositional differences between samples.
- **Flexible Dispersion Estimation:** Three-tier approach (common, trended, tagwise) allows capturing different levels of biological variability across genes.
- **Quasi-likelihood Framework:** The QL F-test provides better type I error control than likelihood ratio tests, especially for small sample sizes.

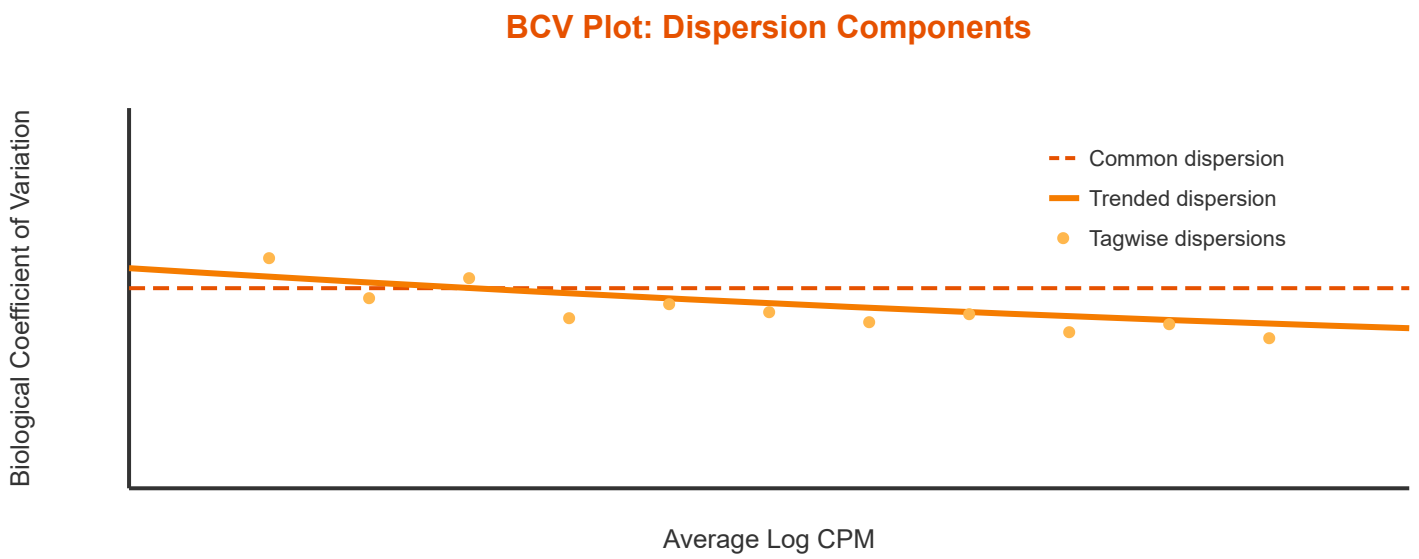
- **Exact Test:** For simple two-group comparisons, edgeR offers an exact test analogous to Fisher's exact test, which doesn't require asymptotic approximations.

- **Complex Design Support:** Excellent support for multi-factor experiments, paired designs, and batch effect correction through its GLM framework.

Example R Code

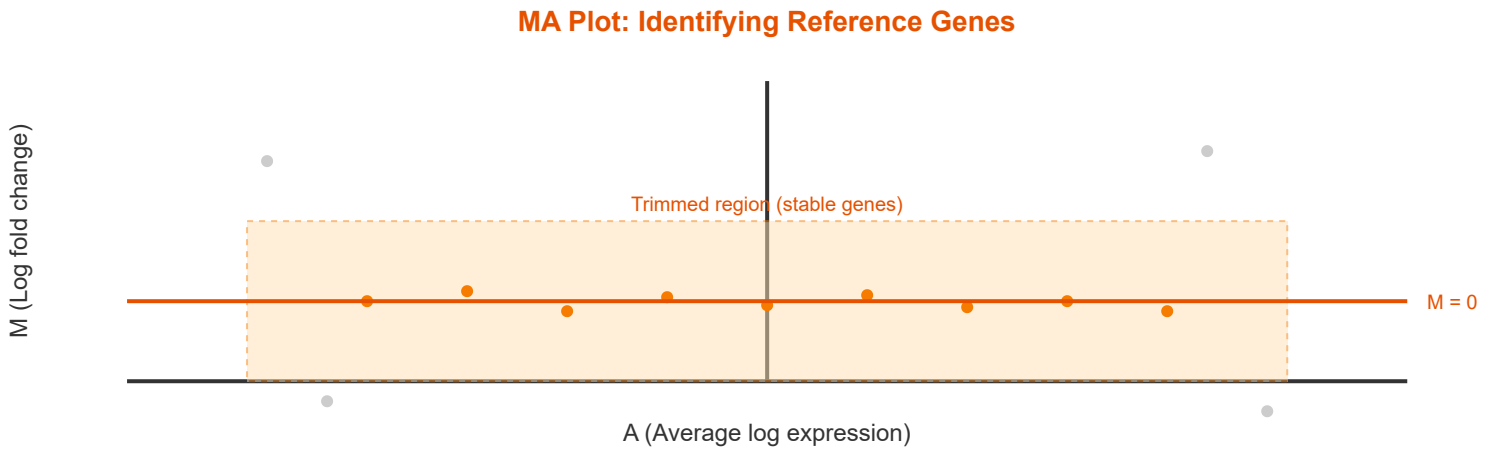
```
# Load library library(edgeR) # Create DGEList object y <- DGEList(counts = counts, group = group) # TMM normalization y <- calcNormFactors(y, method = "TMM") # Design matrix design <- model.matrix(~ group) # Estimate dispersions y <- estimateDisp(y, design) # Fit GLM fit <- glmQLFit(y, design) # Conduct quasi-likelihood F-test qlf <- glmQLFTest(fit, coef = 2) # Extract results results <- topTags(qlf, n = Inf)
```

Understanding BCV (Biological Coefficient of Variation)



Red dashed line represents common dispersion (average across all genes). **Orange curve** shows trended dispersion (expression-dependent). **Orange points** are tagwise (gene-specific) dispersions that are moderated toward the trend using empirical Bayes.

TMM Normalization Principle



TMM trims extreme M-values and A-values (typically 30% and 10% respectively) before calculating the normalization factor. This makes it robust to genes with very high expression or extreme differential expression that could bias the normalization.

📉 limma-voom - Detailed Analysis

Overview

limma-voom transforms RNA-seq count data to log-counts per million (log-CPM) with associated precision weights. This allows the use of limma's linear modeling framework, which was originally developed for microarray data. The voom transformation estimates the mean-variance relationship and computes precision weights for each observation.

Statistical Model

Unlike DESeq2 and edgeR, limma-voom operates in log-space with a normal distribution:

$$\log_2(\text{CPM} + 0.5) \sim N(\mu, \sigma^2_i)$$

N

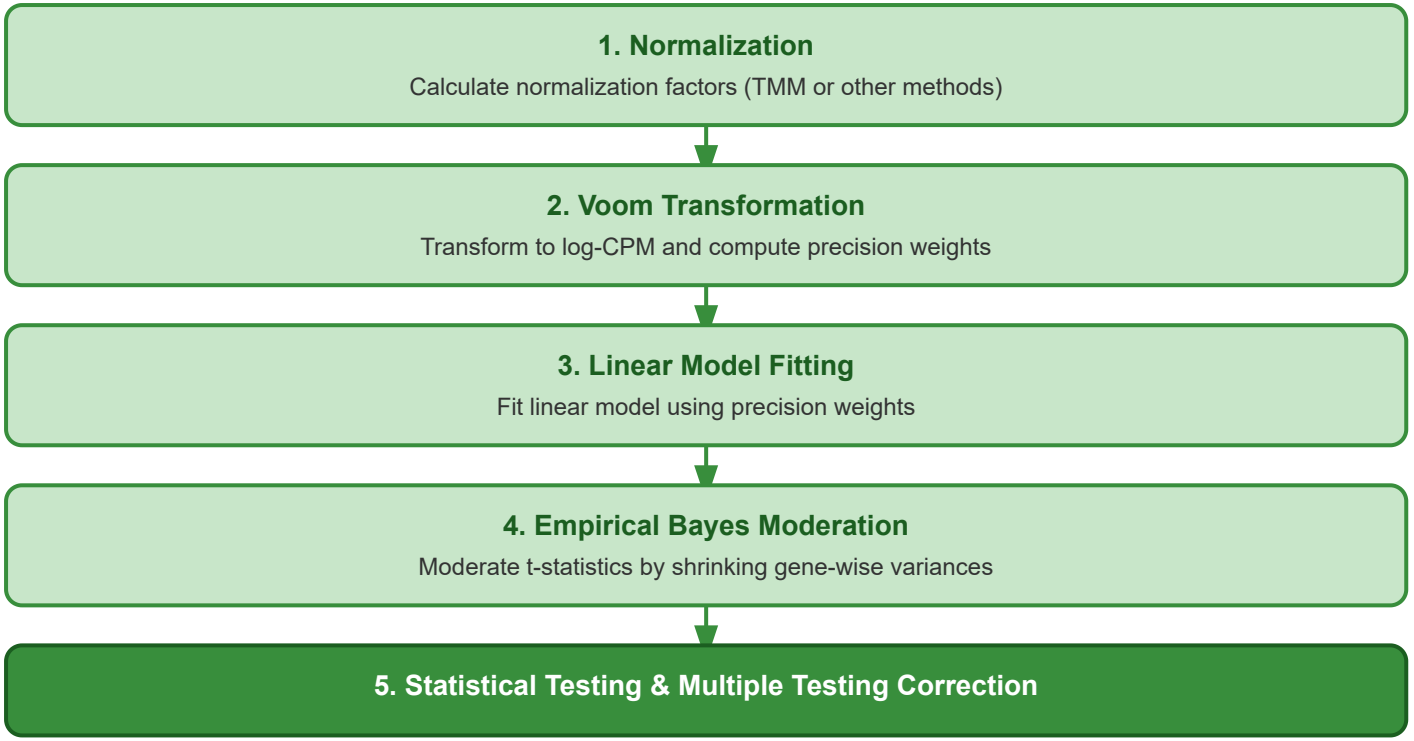
CPM = Counts per million (normalized counts)

μ = Mean log-expression for the gene

σ^2_i = Variance for observation i (varies by expression level)

Precision weights $w_i = 1/\sigma^2_i$ capture mean-variance relationship

Analysis Workflow



Key Features & Advantages

- **Log-space Analysis:** Working in log-space makes the data more normally distributed and allows using the mature statistical framework of linear models.
- **Precision Weights:** voom accurately models the mean-variance relationship by computing observation-specific precision weights, addressing heteroscedasticity in the log-transformed data.
- **Speed:** Linear modeling is computationally faster than iterative GLM fitting, making limma-voom particularly efficient for large datasets.

- **Empirical Bayes Moderation:** Borrows information across genes to moderate standard errors, improving statistical power especially with small sample sizes.

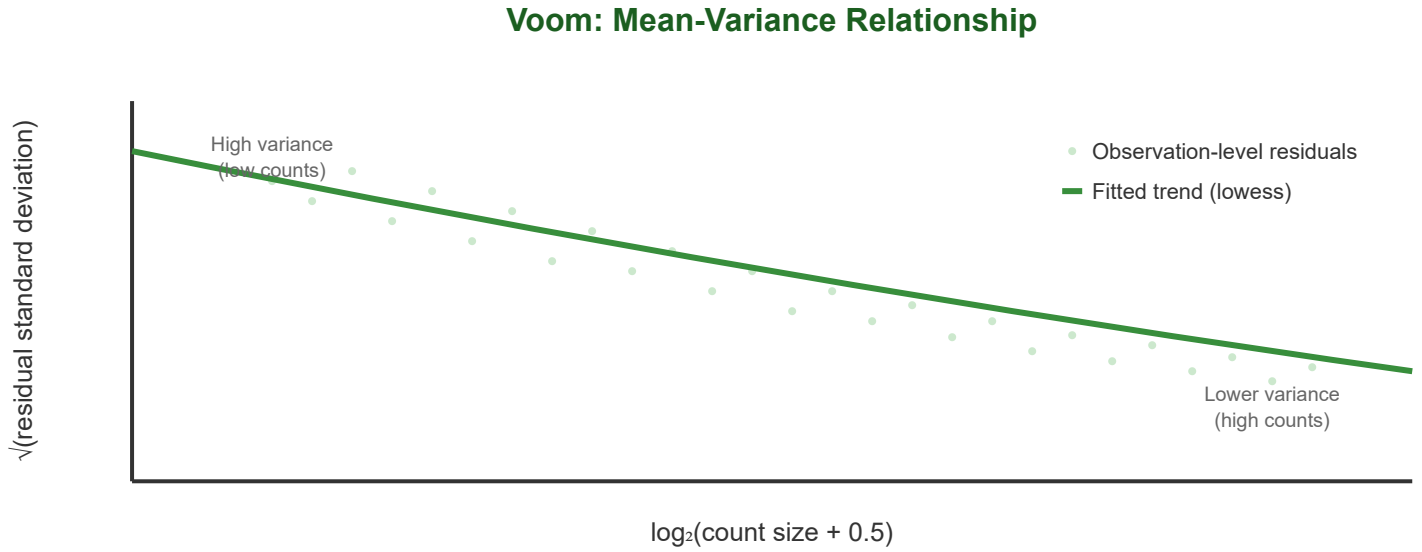
- **Flexible Contrasts:** Easy specification of complex contrasts and multiple comparisons within a single modeling framework.

- **Quality Weights:** Can incorporate sample-level quality weights (voomWithQualityWeights) to down-weight poor-quality samples automatically.

Example R Code

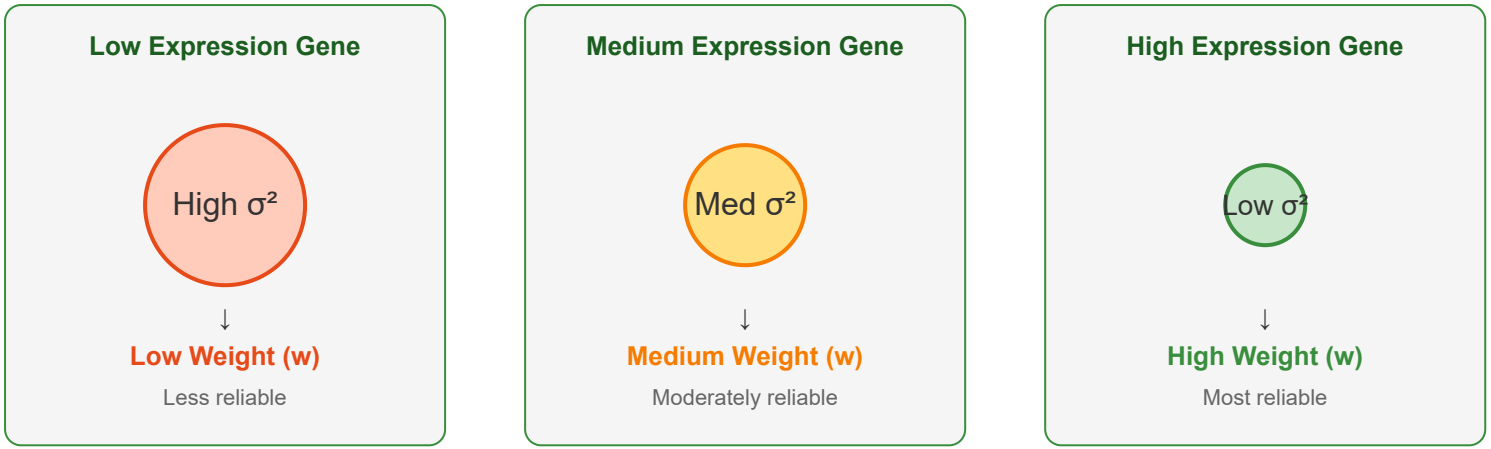
```
# Load libraries library(limma) library(edgeR) # Create DGEList and normalize dge <- DGEList(counts = counts) dge <- calcNormFactors(dge, method = "TMM") # Design matrix design <- model.matrix(~ group) # Voom transformation v <- voom(dge, design, plot = TRUE) # Fit linear model fit <- lmFit(v, design) # Empirical Bayes moderation fit <- eBayes(fit) # Extract results results <- topTable(fit, coef = 2, number = Inf)
```

Understanding the Voom Mean-Variance Trend



The voom plot shows the square root of residual standard deviation versus average $\log_2$ count. The trend captures the mean-variance relationship, which is then used to calculate precision weights. **Low-count genes** (left) have higher variance and lower weights, while **high-count genes** (right) have lower variance and higher weights in the analysis.

How Precision Weights Work



Precision weight $w = 1/\sigma^2$. Genes with low expression have high variance (large circle) and receive low weights, meaning they contribute less to the statistical inference. High-expression genes have low variance (small circle) and receive high weights, making them more influential in the analysis. This appropriately accounts for the different reliability of measurements across the expression range.

📋 Method Comparison Summary

Feature	DESeq2	edgeR	limma-voom
Statistical Model	Negative Binomial GLM	Negative Binomial GLM	Linear model (log-space)
Normalization	Size factors (geometric mean)	TMM (Trimmed Mean of M-values)	TMM or other methods

Feature	DESeq2	edgeR	limma-voom
Dispersion Estimation	Shrinkage toward fitted trend	Common, trended, tagwise	Precision weights from mean-variance
Statistical Test	Wald test / LRT	Exact test / LRT / QL F-test	Moderated t-test
Best For	Small sample sizes (2-5 per group)	Complex designs, many samples	Large datasets, speed priority
Computation Speed	Moderate	Moderate	Fast
Memory Usage	Moderate	Low-Moderate	Low
Handling of Low Counts	Excellent (shrinkage)	Very good (empirical Bayes)	Good (precision weights)
Complex Designs	Good	Excellent	Excellent
Documentation	Extensive	Extensive	Extensive
Typical Use Case	Standard DE analysis, medical research	Multi-factor experiments, large studies	Large-scale screens, time-course

💡 Choosing the Right Method

Use DESeq2 if: You have small sample sizes (2-5 replicates), prioritize conservative estimates, or are following standard genomics workflows.

Use edgeR if: You have complex experimental designs with multiple factors, need flexible statistical tests, or want the most established NB-based approach.

Use limma-voom if: You have large datasets requiring fast computation, are comfortable with log-transformation assumptions, or want easy integration with microarray analysis pipelines.

General advice: All three methods produce similar results when used appropriately. The most important factors are proper experimental design, adequate biological replicates (≥3 per group), and appropriate quality control. Choose based on your specific needs and computational resources.

These methods are continuously updated. Always refer to the official documentation and recent literature for the latest recommendations.

Key References: Love et al. (2014) - DESeq2 | Robinson et al. (2010) - edgeR | Law et al. (2014) - limma-voom

Multiple Testing Correction: A Comprehensive Guide

The Multiple Testing Problem

✗ Without Correction

Scenario: Testing 20,000 genes at α = 0.05

When we test thousands of hypotheses simultaneously, even with a reasonable significance level, we end up with an unacceptable number of false positives.

Expected false positives = 20,000 × 0.05 = 1,000 genes!

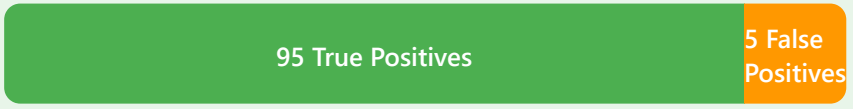
Problem: Among 1,000 "significant" results, we cannot distinguish true discoveries from false alarms!

✓ With FDR Control

Solution: Benjamini-Hochberg procedure at FDR = 0.05

By controlling the False Discovery Rate, we ensure that among all discoveries we make, no more than 5% are false positives.

Controls: False discoveries / Total discoveries ≤ 5%



Benefit: Among 100 called significant, we expect ~95 are truly associated with the phenotype!

Multiple Testing Correction Methods

Bonferroni Correction (FWER)

$$p_{adjusted} = p \times n$$
or
$$\alpha_{threshold} = \alpha / n$$

Key Characteristics:

- ✓ **Controls Family-Wise Error Rate (FWER):** Probability of making at least one false positive ≤ α
- ✓ **Very Conservative:** Suitable when even a single false positive is unacceptable
- ✓ **Simple to Calculate:** Just multiply p-values by number of tests
- ✓ **Low Power:** May miss many true discoveries (high false negative rate)
- ✓ **Assumes Independence:** Overly conservative when tests are correlated

📊 Practical Example:

Study: Testing 100 genes for disease association
Original α = 0.05
Bonferroni threshold: 0.05 / 100 = 0.0005

Benjamini-Hochberg Procedure (FDR)

Sort p-values: $p_{(1)} \leq p_{(2)} \leq \ldots \leq p_{(m)}$
Find largest i where: $p_{(i)} \leq (i/m) \times \alpha$
Reject $H_{(1)}, \ldots, H_{(i)}$

Key Characteristics:

- ✓ **Controls False Discovery Rate (FDR):** Expected proportion of false positives among discoveries ≤ α
- ✓ **Balanced Approach:** More discoveries than Bonferroni while controlling error rate
- ✓ **Standard in Genomics:** Widely accepted in high-throughput studies
- ✓ **Sequential Procedure:** Tests are evaluated in order of significance
- ✓ **Robust:** Works well even with correlated tests

📊 Practical Example:

Study: Same 100 genes, FDR = 0.05

Rank (i)	p-value	BH threshold (i/100) × 0.05	Significant?

Gene	Raw p-value	Bonferroni p	Significant?
Gene A	0.0001	0.01	✓ Yes
Gene B	0.001	0.10	✗ No
Gene C	0.01	1.00	✗ No

Result: Only the most extreme p-values pass the threshold. Minimal false positives but potentially many missed discoveries.

🎯 When to Use:

- Clinical trials where false positives could lead to harmful treatments
- Small number of tests (n < 100)
- When stringent control of Type I error is critical
- Regulatory or diagnostic applications

1	0.0001	0.0005	✓ Yes
2	0.0008	0.0010	✓ Yes
3	0.0012	0.0015	✓ Yes
4	0.0020	0.0020	✓ Yes (largest i)
5	0.0030	0.0025	✗ No

Result: First 4 genes are called significant. Among these, we expect ~5% (0.2 genes) to be false positives.
Comparison: Bonferroni found 1 gene, BH found 4 genes - more discoveries with controlled error!

🎯 When to Use:

- **Genomics studies:** RNA-seq, GWAS, microarray analysis
- **Exploratory research:** When some false positives are acceptable
- **Large-scale testing:** Thousands to millions of hypotheses
- **Discovery phase:** Identifying candidates for follow-up studies

Q-value Method

$$q\text{-value} = \min_{t \geq p} \{ FDR(t) \}$$
$$\text{where } FDR(t) = \pi_0 \times t \times m / \#\{p_i \leq t\}$$
π_0 = estimated proportion of true nulls

Key Characteristics:

- ✓ **Minimum FDR:** Q-value is the minimum FDR at which a test would be called significant
- ✓ **Estimates π_0:** Adaptively estimates proportion of true null hypotheses
- ✓ **More Powerful:** Can discover more true positives than BH when $\pi_0 < 1$
- ✓ **Data-Driven:** Uses the data structure to inform correction
- ✓ **Interpretable:** Direct probability interpretation for each test

📊 Practical Example:

Study: 10,000 gene expression tests

Estimated π_0 = 0.60 (60% are truly null)

Gene	p-value	BH FDR	Q-value	Interpretation

Permutation-Based Testing

1. Randomly permute sample labels
2. Recalculate all test statistics
3. Repeat B times (e.g., B = 1000)
4. $p_{perm} = \#\{t_{perm} \geq t_{obs}\} / B$

Key Characteristics:

- ✓ **Non-Parametric:** No distributional assumptions required
- ✓ **Empirical Null:** Null distribution derived directly from data
- ✓ **Accounts for Correlation:** Preserves dependency structure
- ✓ **Flexible:** Works with any test statistic
- ✓ **Computationally Intensive:** Requires many iterations
- ✓ **Gold Standard:** For complex experimental designs

📊 Practical Example:

Study: Compare gene expression between Case (n=20) and Control (n=20) for 5,000 genes

Procedure:

1. **Observed data:** Calculate t-statistic for each gene
2. **Permutation 1:** Randomly shuffle case/control labels → recalculate all 5,000 t-statistics
3. **Permutation 2-1000:** Repeat shuffling and calculation

Gene X	0.001	0.08	0.05	Significant at 5% FDR
Gene Y	0.005	0.12	0.08	Borderline at 10% FDR
Gene Z	0.02	0.25	0.18	Not significant

Key Insight: Q-value accounts for the fact that 40% of genes are truly differentially expressed, making it less conservative than standard BH.

Advantage: If $\pi_0 = 0.60$, the correction is less severe than assuming all nulls are true ($\pi_0 = 1.0$), resulting in more discoveries.

🎯 When to Use:

- **Enriched datasets:** When expecting many true positives
- **Differential expression:** Comparing treatment vs. control
- **Maximum power needed:** When false negatives are costly
- **Well-powered studies:** Large sample sizes with strong effects

4. **P-value:** For Gene A with $t_{obs} = 3.5$, count how many permutations yielded $t \geq 3.5$

Gene	t_{obs}	Parametric p	Permutation p	Difference
Gene A	4.2	0.0001	0.0002	Similar
Gene B	2.8	0.008	0.015	More conservative
Gene C	2.1	0.042	0.068	Not significant

Advantage: Permutation accounts for actual correlation between genes, avoiding overly liberal or conservative corrections.

🎯 When to Use:

- **Small sample sizes:** When parametric assumptions are questionable
- **Complex designs:** Time series, paired samples, block designs
- **Correlated features:** Gene co-expression, spatial data
- **Validation:** Confirming results from other methods
- **When computational resources allow:** Can be parallelized

Method Selection Guide

⚖️ Power vs. Control Trade-off

Most Conservative → Most Liberal:

Bonferroni (Strictest control, fewest discoveries) → **Benjamini-Hochberg** (Balanced FDR control) → **Q-value** (Adaptive, more discoveries) → **Permutation** (Data-driven, context-specific)

General Recommendations:

- **Use Bonferroni when:** You need absolute certainty, testing few hypotheses, or false positives are extremely costly
- **Use BH when:** Standard genomics analysis, thousands of tests, exploratory phase
- **Use Q-value when:** You expect many true signals, need maximum power, follow-up validation planned
- **Use Permutation when:** Complex designs, correlated data, small samples, or validating other methods

💡 **Important Note:** In large-scale genomics studies (e.g., testing 20,000+ genes), **multiple testing correction is essential**. Without correction, the overwhelming majority of "significant" results would be false positives, making the findings unreliable and unreproducible.

Best Practice: Always report which correction method was used, the threshold applied (e.g., FDR < 0.05), and the number of discoveries made. Consider using multiple methods and examining their concordance for robustness.

Pathway Analysis: Comprehensive Guide

From Genes to Biological Insights

Pathway analysis transforms lists of differentially expressed genes into interpretable biological knowledge. Instead of examining hundreds of individual genes, we identify enriched biological pathways, processes, and functions.

Analysis Methods

Over-Representation Analysis (ORA)

How it works:

- Tests if DE genes are over-represented in a pathway
- Uses Fisher's exact test on 2×2 contingency table
- Simple and interpretable results

Limitations:

- Requires arbitrary threshold (e.g., $p < 0.05$, $|FC| > 2$)
- Loses information about genes near threshold
- Ignores magnitude of gene expression changes

Gene Set Enrichment Analysis (GSEA)

How it works:

- Uses all genes ranked by expression change
- Calculates enrichment score (ES) using Kolmogorov-Smirnov test
- No arbitrary threshold needed

Advantages:

- More sensitive than ORA
- Considers all genes and their ranks

- Identifies coordinated changes in pathways

Feature	ORA	GSEA
Input	List of significant genes	All genes with expression values
Threshold	Required (e.g., $p < 0.05$, $\|FC\| > 2$)	Not required
Statistical Test	Fisher's exact test / Hypergeometric test	Kolmogorov-Smirnov-like statistic
Sensitivity	Lower (loses borderline genes)	Higher (uses all genes)
Computational Cost	Fast	Slower (permutation testing)
Best For	Quick exploratory analysis	Comprehensive pathway analysis

Pathway Databases

Gene Ontology (GO)

Gene Ontology is a standardized vocabulary system that describes gene functions across all species. It provides a hierarchical, structured representation of biological knowledge organized into three independent ontologies: Biological Process (BP), Molecular Function (MF), and Cellular Component (CC). GO is the most widely used annotation system in bioinformatics and is continuously updated by the research community.

Biological Process (BP)

Series of molecular events with a defined beginning and end. Examples: "cell cycle", "immune response", "metabolic process". Represents biological objectives accomplished by ordered assemblies of molecular functions.

Molecular Function (MF)

Activities at the molecular level. Examples: "kinase activity", "DNA binding", "receptor activity". Describes tasks performed by individual gene products without specifying where or when they occur.

Cellular Component (CC)

Locations where gene products are active. Examples: "nucleus", "mitochondrion", "plasma membrane". Describes subcellular structures and macromolecular complexes.

Hierarchical Structure

Terms organized in directed acyclic graph (DAG) from general to specific. Child terms inherit relationships from parents, enabling analysis at different granularities.

Species-Independent

Applies to all organisms from bacteria to humans. Same GO term can annotate orthologous genes across species, facilitating comparative analyses.

Evidence Codes

Each annotation has evidence code (IEA: inferred, IDA: direct assay, etc.) indicating confidence level and source of annotation.

💡 Example Use Case: Cell Cycle Analysis

Scenario: You identified 150 upregulated genes in cancer cells vs. normal cells.

GO Analysis Results:

- **BP:** "Cell cycle" (GO:0007049, p = 1.2e-15) - 45 genes
- **BP:** "DNA replication" (GO:0006260, p = 3.4e-12) - 28 genes
- **MF:** "DNA-binding transcription factor activity" (GO:0003700, p = 2.1e-8) - 22 genes
- **CC:** "Nucleus" (GO:0005634, p = 5.6e-10) - 78 genes

Interpretation: Cancer cells show dysregulation of cell cycle control and DNA replication, with many transcription factors activated in the nucleus - consistent with uncontrolled proliferation.

🎯 When to use GO: Best for broad functional categorization, understanding where and how gene products function, and comparing results across different studies and organisms.

KEGG (Kyoto Encyclopedia of Genes and Genomes)

KEGG is a comprehensive database that integrates genomic, chemical, and systemic functional information. It focuses on molecular interaction networks including metabolic pathways, signaling cascades, and disease pathways. KEGG is particularly valuable for understanding how genes work together in biological systems and for visualizing pathway diagrams showing enzymes, substrates, and products.

Visual Pathway Maps

Detailed, hand-drawn pathway diagrams showing molecular interactions, substrates, and products with spatial organization. KEGG Mapper allows overlaying expression data onto these maps.

Metabolic Pathways

Comprehensive coverage of metabolic processes including glycolysis, TCA cycle, amino acid metabolism, and lipid metabolism with chemical structures and enzyme reactions.

Signaling Pathways

Well-curated signal transduction pathways (MAPK, PI3K-Akt, Wnt, etc.) showing protein-protein interactions, phosphorylation cascades, and transcriptional regulation.

Disease Pathways

Comprehensive disease pathway collection showing genetic and environmental factors in cancer, neurodegenerative diseases, metabolic disorders, and infectious diseases.

Drug Information

Includes drug-target relationships and pharmacological pathways. Useful for drug discovery, drug repurposing studies, and understanding mechanism of action.

Multi-Omics Integration

Links genes, proteins, compounds, reactions, and diseases. Enables integration of genomics, transcriptomics, metabolomics, and disease data in single framework.

💡 Example Use Case: Drug Response Study

Scenario: Analyzing gene expression changes in cells treated with a kinase inhibitor.

KEGG Analysis Results:

- **hsa04010** - MAPK signaling pathway (p = 2.3e-18) - 45 genes affected
- **hsa04151** - PI3K-Akt signaling pathway (p = 1.5e-14) - 38 genes affected
- **hsa04115** - p53 signaling pathway (p = 4.2e-12) - 22 genes affected
- **hsa05200** - Pathways in cancer (p = 8.7e-10) - 52 genes affected

Visualization: KEGG Mapper colored the affected genes on pathway diagrams, showing ERK and downstream targets downregulated (blue), apoptosis genes upregulated (red), and cell cycle checkpoint genes activated.

Interpretation: The drug effectively inhibits MAPK signaling, leading to cell cycle arrest and apoptosis activation - confirming mechanism of action.

```
# Example: Mapping expression data to KEGG pathway library(pathview) # Gene expression data (log2 fold changes)
gene.data <- setNames(deg_results$log2FoldChange, deg_results$entrez_id) # Visualize on MAPK pathway
pathview(gene.data = gene.data, pathway.id = "04010", # MAPK pathway species = "hsa", # Homo sapiens out.suffix =
"MAPK_drug_treatment")
```

🎯 When to use KEGG: Best for understanding molecular mechanisms, visualizing expression changes on pathway diagrams, drug target analysis, and integrating metabolomics with genomics data.

Reactome

Reactome is a peer-reviewed, expert-curated database of biological pathways with a focus on human biology. It represents pathways as a series of molecular events (reactions) organized hierarchically. Reactome provides detailed pathway diagrams with emphasis on molecular interactions, post-translational modifications (PTMs), protein complexes, and subcellular localization. The database is particularly strong in signal transduction and immune system pathways.

Reaction-Based Model

Pathways decomposed into individual molecular reactions. Each reaction explicitly shows inputs, outputs, catalysts, and regulators with stoichiometry and compartmentalization.

Peer-Reviewed Content

All pathways reviewed by experts in respective fields. Higher confidence in accuracy compared to automated annotations. Each pathway linked to supporting literature.

Detailed Annotations

Includes specific protein modifications (phosphorylation sites), complex formations, subcellular locations, and disease variants with precise molecular detail.

Hierarchical Organization

Top-level pathways subdivided into smaller units. Enables analysis at different granularity levels from broad processes to specific molecular events.

Cross-References

Extensive links to external databases (UniProt, ChEBI, Ensembl, GO). Enables comprehensive data integration across multiple biological resources.

Pathway Browser

Interactive visualization tool with zoom capabilities, overlay of expression data, and exploration of related pathways with intuitive interface.

💡 Example Use Case: Immune Response Analysis

Scenario: RNA-seq data from T cells stimulated with antigen compared to unstimulated controls.

Reactome Analysis Results:

- **R-HSA-202403** - TCR signaling (p = 1.5e-22) - 58 entities

- **R-HSA-202424** - Downstream TCR signaling (p = 3.2e-20) - 42 entities
- **R-HSA-1280215** - Cytokine signaling in immune system (p = 8.1e-18) - 67 entities
- **R-HSA-983705** - Interleukin-2 family signaling (p = 2.4e-15) - 28 entities

Detailed Insights:

- ZAP70 phosphorylation events increased (specific sites: Y319, Y493)
- LAT-SLP76 complex formation upregulated
- Nuclear translocation of NFAT transcription factors detected
- Reactions in cytoplasm and nucleus compartments identified

Interpretation: Complete activation of TCR signaling cascade with proper complex formation and nuclear translocation - indicating functional T cell activation.

```
# Example: Reactome pathway analysis library(ReactomePA) library(org.Hs.eg.db) # Convert gene symbols to Entrez IDs gene.entrez <- bitr(de.genes, fromType="SYMBOL", toType="ENTREZID", OrgDb=org.Hs.eg.db) # Enrichment analysis reactome.result <- enrichPathway(gene = gene.entrez$ENTREZID, pvalueCutoff = 0.05, pAdjustMethod = "BH", readable = TRUE) # Visualize as network library(enrichplot) cnetplot(reactome.result, categorySize="pvalue")
```

🎯 When to use Reactome: Best for detailed mechanistic understanding, analyzing protein modifications and complexes, studying immune system pathways, and when you need expert-curated, peer-reviewed pathway information.

MSigDB (Molecular Signatures Database)

MSigDB is a comprehensive collection of annotated gene sets developed at the Broad Institute. It aggregates gene sets from multiple sources including pathway databases, published literature, and computational analyses. MSigDB is organized into collections (C1-C8 and Hallmark), each serving different analysis purposes. It's the standard resource for Gene Set Enrichment Analysis (GSEA) and is particularly valuable for its curated "Hallmark" gene sets representing well-defined biological states and processes.

Hallmark Gene Sets (H)

50 refined sets representing well-defined biological states. Reduced redundancy and noise. Most commonly used for initial GSEA analysis. Examples: HALLMARK_HYPOXIA, HALLMARK_APOPTOSIS.

Curated Gene Sets (C2)

~6,300 gene sets from pathway databases (KEGG, Reactome, BioCarta), chemical and genetic perturbations. Most comprehensive collection for exploring diverse hypotheses.

Ontology Gene Sets (C5)

Gene Ontology terms organized by BP, CC, and MF. Provides GO analysis within MSigDB framework. Useful for functional annotation with GSEA method.

Oncogenic Signatures (C6)

Gene sets representing signatures of cellular pathways disrupted in cancer. Includes signatures for oncogenes (MYC, RAS) and tumor suppressors (TP53, RB1).

Immunologic Signatures (C7)

Gene sets representing cell types and states of immune system. Derived from microarray and RNA-seq studies of immune cells. Useful for immune profiling.

Cell Type Signatures (C8)

Gene sets from single-cell RNA-seq studies. Cell type markers for various tissues and cell types. Essential for deconvolution and cell type identification.

💡 Example Use Case: Cancer Subtype Classification

Scenario: Gene expression data from breast cancer samples - understanding biological differences between subtypes.

MSigDB GSEA Results (Hallmark):

- **Basal-like subtype enriched:**
  - HALLMARK_EPITHELIAL_MESENCHYMAL_TRANSITION (NES=2.45, FDR=0.001)
  - HALLMARK_HYPOXIA (NES=2.12, FDR=0.002)

- HALLMARK_GLYCOLYSIS (NES=1.98, FDR=0.003)

- **Luminal subtype enriched:**
  - HALLMARK_ESTROGEN_RESPONSE_EARLY (NES=2.87, FDR<0.001)
  - HALLMARK_ESTROGEN_RESPONSE_LATE (NES=2.34, FDR=0.001)

Using C6 (Oncogenic Signatures):

- MYC_TARGETS highly enriched in aggressive tumors
- TP53_DN signatures in basal-like subtype

Interpretation: Clear molecular distinction - basal-like shows aggressive phenotype with EMT and metabolic reprogramming, while luminal shows estrogen dependence.

```
# Example: MSigDB analysis with fgsea library(fgsea) library(msigdbr) # Get Hallmark gene sets hallmark_sets <- msigdbr(species = "Homo sapiens", category = "H") hallmark_list <- split(hallmark_sets$gene_symbol, hallmark_sets$gs_name) # Prepare ranked gene list gene_ranks <- setNames(deg_results$stat, deg_results$gene) gene_ranks <- sort(gene_ranks, decreasing = TRUE) # Run GSEA fgsea_results <- fgsea(pathways = hallmark_list, stats = gene_ranks, minSize = 15, maxSize = 500, nPermSimple = 10000) # Filter significant pathways sig_pathways <- fgsea_results[padj < 0.05]
```

🎯 When to use MSigDB: Best for GSEA analysis, exploring diverse biological hypotheses, cancer research, when you want both broad (Hallmark) and comprehensive (C2) options, and for standardized analysis workflows.

Choosing the Right Database

Database	Best For	Strengths	Considerations

Gene Ontology	Broad functional annotation, cross-species comparisons	• Universal applicability • Most comprehensive • Hierarchical structure • Widely adopted	• Can be redundant • Varying evidence quality • Less detailed mechanisms
KEGG	Metabolic pathways, visual interpretation, drug studies	• Beautiful pathway maps • Metabolic focus • Drug information • Easy visualization	• Reference pathways • License restrictions • Update frequency
Reactome	Detailed mechanisms, immune pathways, PTM analysis	• Peer-reviewed • Reaction-level detail • Excellent for signal transduction • Well-annotated complexes	• Mainly human-focused • Smaller than others • Can be very detailed
MSigDB	GSEA, cancer research, hypothesis generation	• Hallmark sets (refined) • Comprehensive collections • Regular updates • GSEA-optimized	• Overlapping sets in C2 • Varying quality • Can be overwhelming

Best Practices for Pathway Analysis

1. Multiple Testing Correction

Always apply FDR or Bonferroni correction. Testing thousands of pathways inflates false positives. Use adjusted p-value < 0.05 as cutoff.

2. Consider Gene Set Size

Filter pathways with 15-500 genes. Very small sets are unstable; very large sets are too general and lack specificity.

3. Validate Results

Cross-validate findings across multiple databases. Examine individual genes in significant pathways for biological sense.

4. Use Background Appropriately

5. Report Methodology

6. Interpret Biologically

For ORA, use all genes measured in experiment as background, not entire genome. Affects statistical significance dramatically.

Clearly state: database version, analysis method (ORA/GSEA), p-value cutoffs, correction method, and software version.

Statistical significance ≠ biological importance. Consider effect sizes, biological context, and literature support.

⚠️ Common Pitfalls to Avoid

- **Cherry-picking results:** Don't select only pathways that fit your hypothesis
- **Ignoring redundancy:** Multiple related pathways may represent same biology
- **Over-interpreting p-values:** p=0.049 vs p=0.051 is not meaningful difference
- **Wrong background:** Using wrong gene universe inflates false positives
- **Outdated databases:** Use current versions - biology knowledge evolves
- **Ignoring directionality:** Check if genes are up or down-regulated in pathway

Summary & Key Takeaways

🎯 Quick Reference Guide

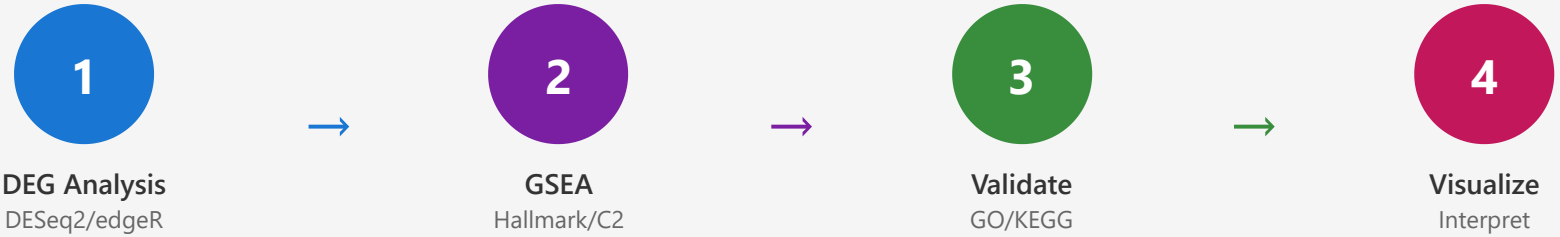
Analysis Methods

- **ORA:** Fast, simple, requires thresholds
- **GSEA:** More powerful, uses all genes
- **Recommendation:** Use GSEA when possible
- **Why:** GSEA detects subtle coordinated changes

Database Selection

- **Starting point:** MSigDB Hallmark or GO BP
- **Mechanism:** KEGG or Reactome
- **Validation:** Use multiple databases
- **Species:** Non-human → prioritize GO

📊 Typical Workflow



✅ Analysis Checklist

- ☑ Multiple testing correction applied
- ☑ Appropriate background used
- ☑ Gene set size filtered (15-500)
- ☑ Multiple databases consulted
- ☑ Individual genes examined
- ☑ Biological context considered
- ☑ Methods clearly documented
- ☑ Results validated

📚 Additional Resources

GO Consortium
geneontology.org
~44,000 terms

KEGG
genome.jp/kegg
~500 pathways

Reactome
reactome.org
~2,500 pathways

MSigDB
gsea-msigdb.org
~25,000 gene sets

🧬 Remember

Pathway analysis transforms gene lists into biological insights.

Choose your tools wisely, validate thoroughly, and always think biologically!

The goal is understanding, not just statistics.

Part 2/3:

Single-Cell Technologies

- Technology overview
- Cell isolation
- Quality control
- Analysis challenges

scRNA-seq Overview

Evolution & Comparison of Single-Cell Technologies

Throughput vs Depth Trade-off

Cell Atlases	Development	Disease	Drug Discovery
Human Cell Atlas	Lineage tracing	Tumor heterogeneity	Target identification

💡 Revolution in understanding cellular heterogeneity

Droplet-based Single-Cell RNA Sequencing Methods

Droplet-based methods represent a revolutionary approach to single-cell RNA sequencing, enabling the analysis of thousands to millions of individual cells in a single experiment. These platforms use microfluidic technology to encapsulate individual cells with barcoded beads in nanoliter-sized droplets, allowing for massive parallelization and cost-effective profiling of cellular heterogeneity at unprecedented scales.

10X Genomics Platform

Most widely used - Chromium platform with GEMs

Drop-seq Principles

Co-encapsulation of cells and barcoded beads

InDrop Technology

Hydrogel beads with photocleavable barcodes

Barcode Design

Cell barcode + UMI for molecular counting

Doublet Detection

Computational and experimental QC for multiplets

1 10X Genomics Chromium Platform

Overview

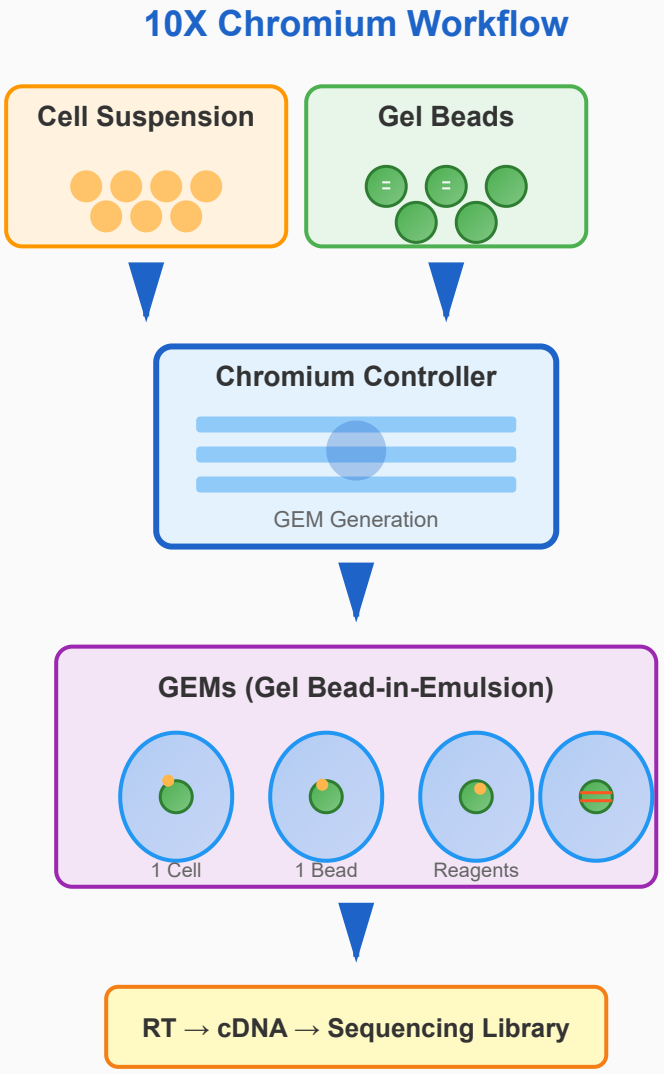
The 10X Genomics Chromium platform is the gold standard in droplet-based scRNA-seq, utilizing Gel Bead-in-Emulsion (GEM) technology. The system can process up to 80,000 cells per run, with each cell receiving a unique barcode from gel beads containing millions of oligonucleotides.

Key Features

The platform uses a specialized microfluidic chip that partitions cells into individual GEMs along with gel beads and reagents. Each gel bead is covered with approximately 100 million copies of a unique oligonucleotide, ensuring robust barcode capture. The system achieves high cell capture rates (typically 50-65%) with low doublet rates (<1% per 1,000 cells).

Technical Specifications

- **Throughput:** 500-80,000 cells per sample
- **Capture rate:** 50-65% of input cells
- **Genes detected:** ~1,000-5,000 per cell
- **UMIs per cell:** ~10,000-50,000
- **Processing time:** ~7-8 minutes for partitioning
- **Doublet rate:** ~0.8% per 1,000 cells loaded



Advantages

User-friendly automated workflow, standardized protocols, comprehensive computational tools (Cell Ranger, Loupe Browser), and extensive community support make 10X the preferred choice for most research applications.

2 Drop-seq Principles

Fundamental Concept

Drop-seq, developed at Harvard Medical School, pioneered the co-encapsulation approach for droplet-based scRNA-seq. The method uses custom microfluidic devices to pair individual cells with uniquely barcoded microparticles (beads) in nanoliter-volume aqueous droplets suspended in oil. This enables massively parallel single-cell transcriptomics at low cost.

Technical Implementation

The Drop-seq workflow involves flowing cells and barcoded beads through separate channels that merge at a junction where oil is introduced, creating droplets. Each bead contains ~100 million copies of a unique 12-nucleotide barcode sequence. After droplet formation, cells are lysed, mRNA is captured on beads, and reverse transcription occurs within droplets.

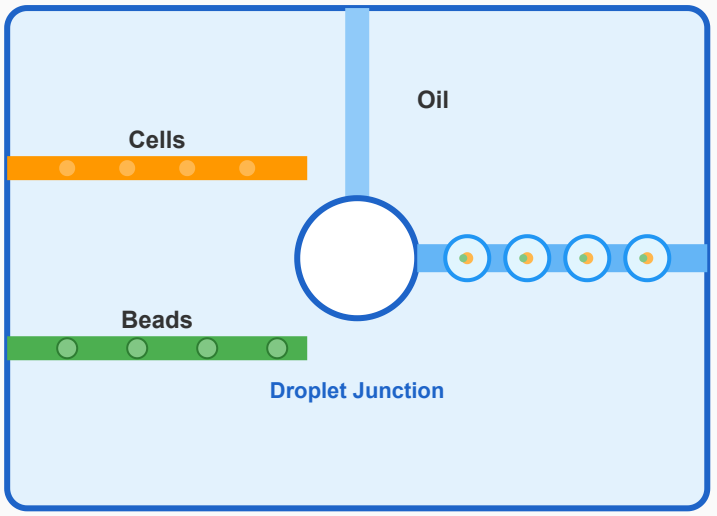
🔬 Key Advantages

- **Cost-effective:** DIY approach with open-source protocols
- **Scalable:** Can profile 10,000+ cells per experiment
- **Flexible:** Customizable for different applications
- **Accessible:** Detailed protocols publicly available
- **Compatible:** Standard molecular biology reagents

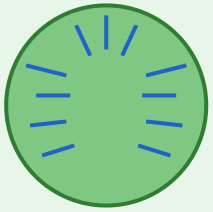
📊 Performance Metrics

Typical Drop-seq experiments capture ~1,000-3,000 genes per cell with median UMI counts around 1,000-5,000. While lower than plate-based methods, the massive throughput enables comprehensive characterization of cellular populations and rare cell types.

Drop-seq Microfluidic Design



Barcoded Bead



~10^8 oligonucleotides
Unique barcode

Process Steps

1. Cell lysis in droplet
2. mRNA capture on bead
3. Reverse transcription
4. Break emulsion
5. PCR amplification
6. Library preparation

3 InDrop Technology

Innovation Overview

InDrop (Indexing Droplets) represents a unique approach using hydrogel microspheres with photocleavable primers. Developed at Harvard, this system employs UV light to release barcoded primers after encapsulation, enabling precise control of the reverse transcription reaction timing. The technology supports high-throughput analysis with improved flexibility.

Photocleavable Mechanism

The hallmark of InDrop is its use of hydrogel beads containing photocleavable primers. These primers remain inactive until UV exposure, which cleaves a photolabile group and releases the primers into the droplet. This design prevents premature reactions and allows for better temporal control of the workflow, reducing background and improving data quality.

🔬 Technical Features

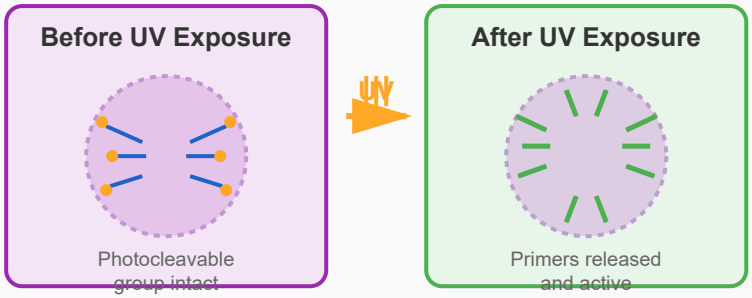
- **Hydrogel beads:** Soft, porous structure for better reagent diffusion
- **Photocleavable primers:** UV-activated barcode release
- **Cell viability:** Gentle encapsulation preserves cell integrity
- **Barcode diversity:** 147,456 unique cell barcodes
- **Throughput:** Thousands of cells per run

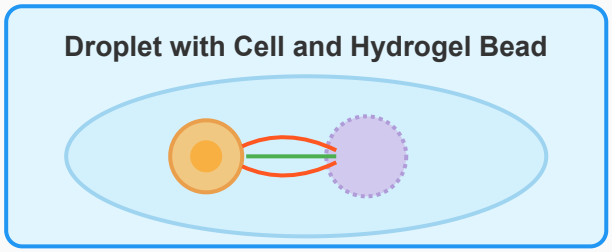
- **Flexibility:** Compatible with various cell types and conditions

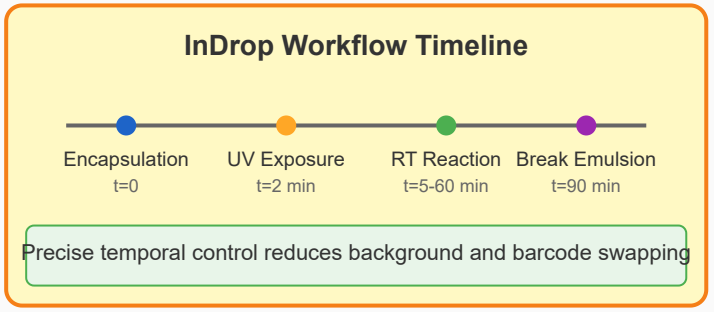
Unique Advantages

The photocleavable design provides superior control over reaction timing, reducing premature primer extension and barcode swapping. The hydrogel matrix allows better diffusion of lysate and reagents compared to solid beads, potentially improving capture efficiency for long transcripts.

InDrop Photocleavable Technology







4 Barcode Design and UMI Strategy

Barcode Architecture

The barcode design in droplet-based scRNA-seq is critical for accurate molecular counting and cell identification. A typical barcode structure consists of multiple functional elements: a cell barcode (identifying individual cells), a Unique Molecular Identifier (UMI) for counting individual mRNA molecules, and a polyT sequence for mRNA capture. This hierarchical design enables precise quantification while minimizing technical artifacts.

UMI Functionality

UMIs are random nucleotide sequences (typically 10-12 bp) that tag individual mRNA molecules before amplification. By collapsing PCR duplicates back to unique molecules based on UMI sequences, researchers can accurately count original transcript numbers rather than amplified copies. This dramatically improves quantification accuracy and reduces amplification bias, particularly important for detecting low-abundance transcripts.

🔬 Design Specifications

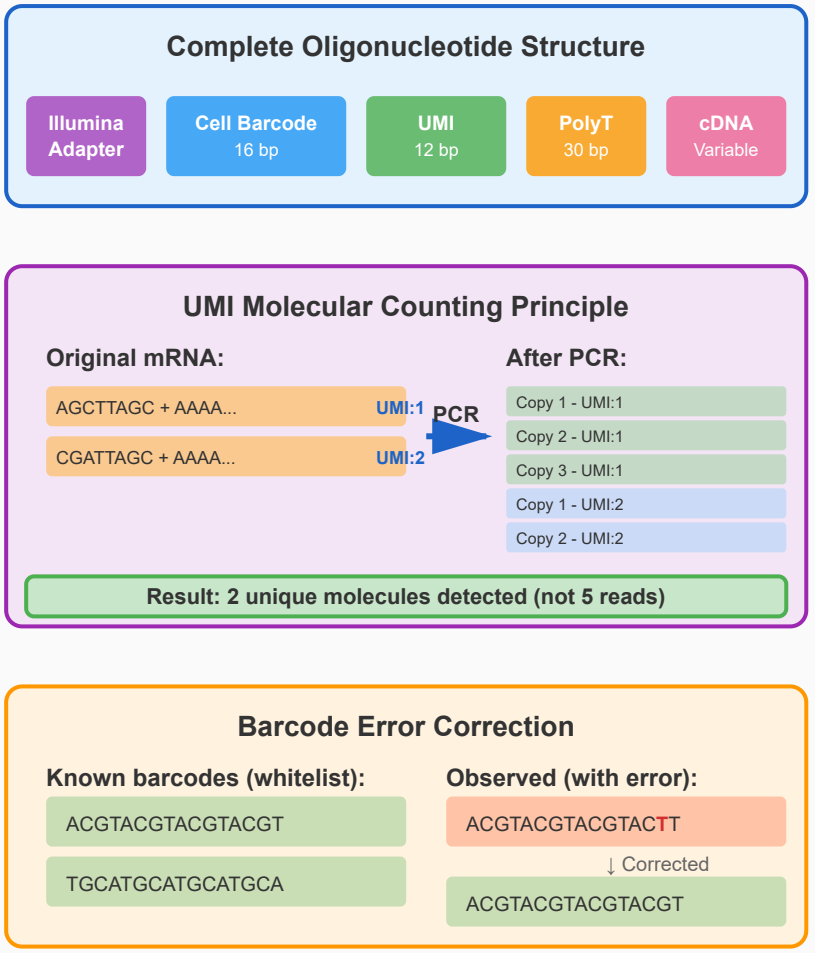
- **Cell barcode:** 12-16 bp, 147K-16M unique combinations
- **UMI length:** 10-12 bp, ~1-4 million unique tags
- **Error correction:** Hamming distance algorithms for barcode errors
- **PolyT tail:** 20-30 T's for mRNA poly(A) capture

- **Read structure:** R1 = barcode+UMI, R2 = cDNA insert
- **Multiplexing:** Sample barcodes for pooled sequencing

📊 UMI Benefits

UMI-based counting reduces quantification bias by 3-10 fold compared to read counting, particularly for highly expressed genes. It enables accurate detection of fold-changes as low as 1.5x and improves identification of differentially expressed genes in low-input samples.

Barcode Structure and UMI Strategy



5 Doublet Detection and Quality Control

Doublet Formation

Doublets occur when two or more cells are captured in a single droplet, resulting in merged transcriptomes that can be misinterpreted as novel cell states. The doublet rate typically increases linearly with cell loading concentration, ranging from 0.4% per 1,000 cells in 10X Genomics to higher rates in other platforms. Doublets are particularly problematic as they can create artificial cell types or mask true biological variation.

Detection Strategies

Multiple computational approaches exist for doublet detection. DoubletFinder uses artificial doublets created by averaging expression profiles to train a classifier. Scrublet generates simulated doublets and compares them to observed cells using nearest-neighbor distances. DoubletDecon uses deconvolution to identify cells with mixed expression signatures. Experimental approaches include cell hashing with oligonucleotide-conjugated antibodies and genetic demultiplexing using natural genetic variation.

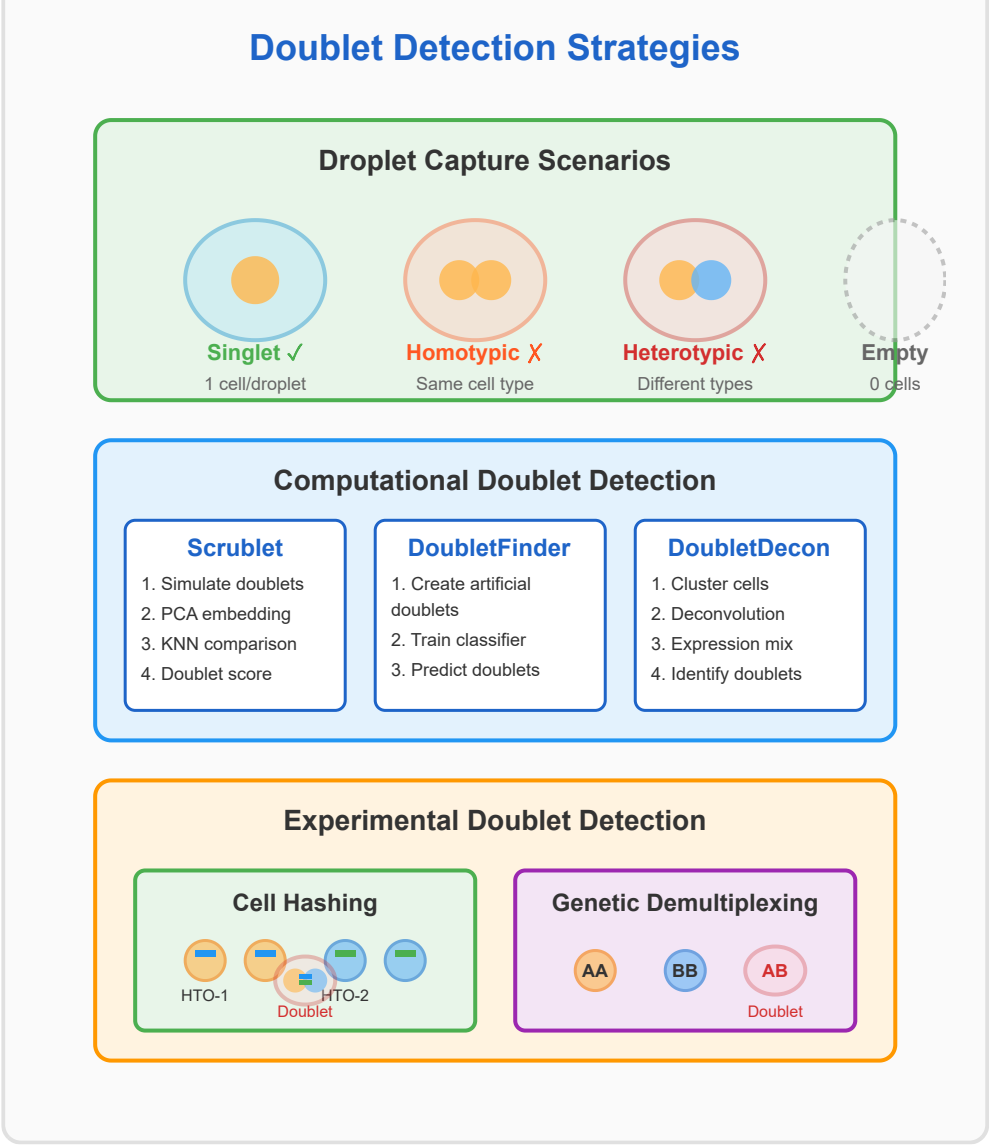
🔬 Detection Methods

- **Computational:** DoubletFinder, Scrublet, DoubletDecon algorithms
- **Cell hashing:** Lipid-tagged oligonucleotides for sample multiplexing

- **Genetic demux:** SNP-based identification of mixed genotypes
- **Expression patterns:** Dual marker gene detection
- **Library size:** Elevated UMI counts indicate doublets
- **Heterotypic doublets:** Most detectable (different cell types)

⚠ Quality Control Thresholds

Standard QC includes filtering cells with <200 or >5,000 genes, >10% mitochondrial reads, and doublet scores above threshold. Multiplet rate estimation: doublet rate ≈ 0.008 × (cells loaded / 1,000). Combined computational and experimental approaches provide highest accuracy.



Method	Type	Sensitivity	Specificity	Limitations
Scrublet	Computational	70-85%	90-95%	Misses homotypic doublets
DoubletFinder	Computational	75-90%	92-97%	Computationally intensive
Cell Hashing	Experimental	95-99%	98-99%	Requires sample multiplexing

Method	Type	Sensitivity	Specificity	Limitations
Genetic Demux	Experimental	90-95%	95-98%	Requires genetic diversity

💡 Key Takeaways

- **High Throughput:** Droplet methods enable profiling of 10,000-80,000 cells per experiment
- **Cost-Effective:** Lower per-cell costs compared to plate-based methods
- **Trade-off:** Higher throughput at the expense of per-cell sensitivity (1,000-5,000 genes/cell)
- **Barcode Design:** Cell barcodes + UMIs enable accurate molecular counting
- **Quality Control:** Computational and experimental doublet detection methods are essential
- **Platform Choice:** 10X Genomics offers user-friendliness; Drop-seq/InDrop offer flexibility

Plate-based Methods

Plate-based scRNA-seq Workflow

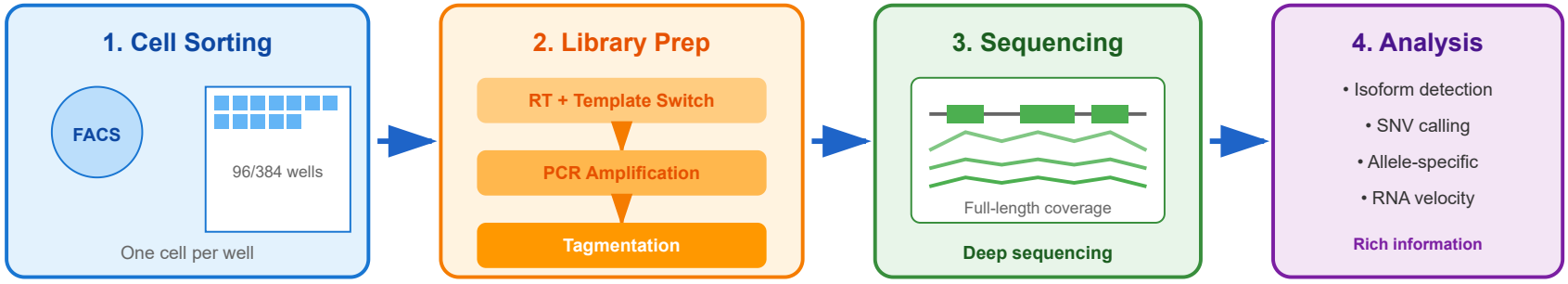


Plate-based Methods Comparison

Smart-seq2/3

- ✓ Full-length transcripts
- ✓ Highest sensitivity
- ✓ Isoform analysis
- ✗ No UMIs
- ✗ Higher cost/cell

MARS-seq

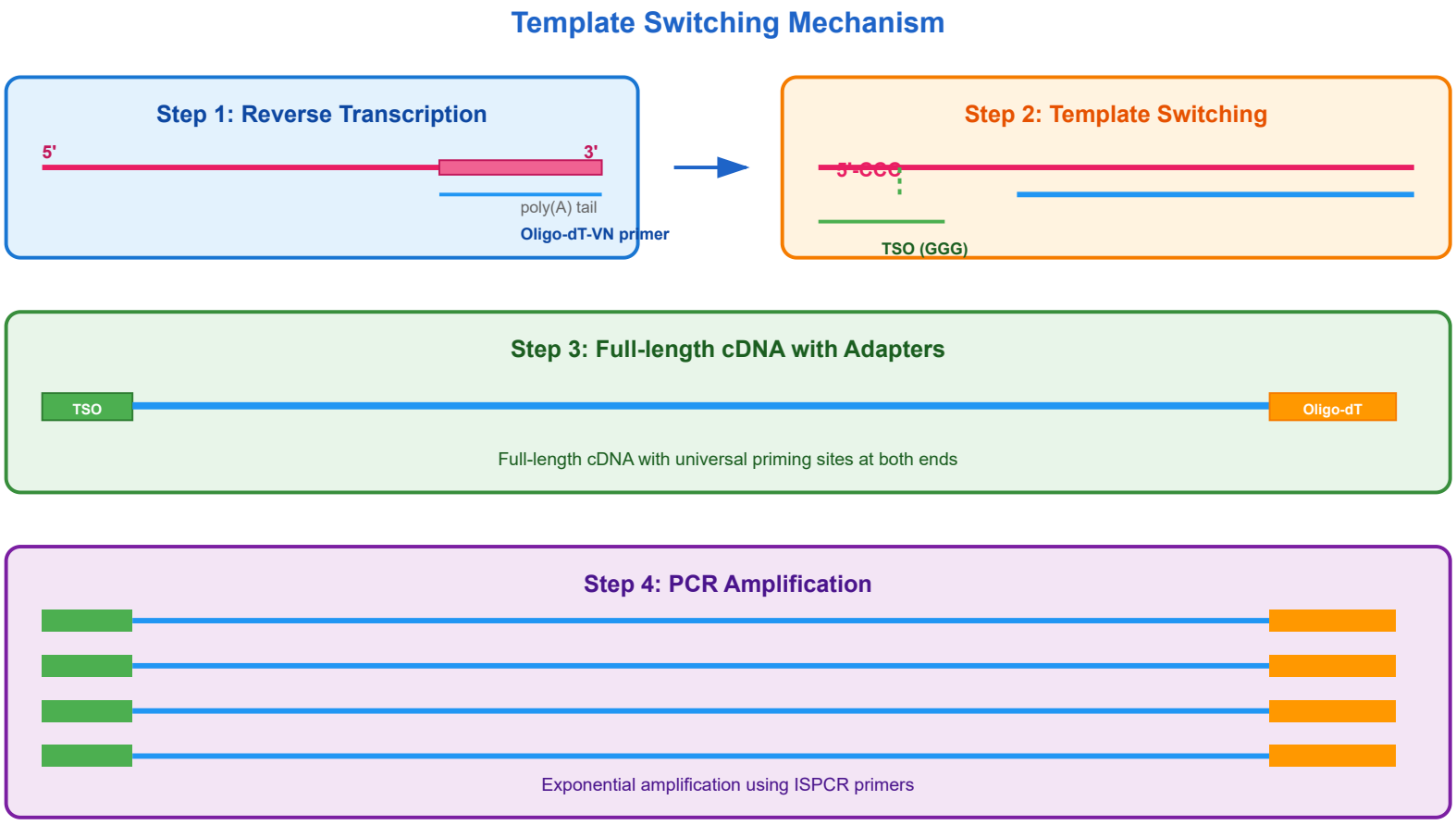
- ✓ UMI incorporation
- ✓ Automated
- ✓ 3' counting
- ✓ Cost-effective
- ✗ 3' bias

CEL-seq2

- ✓ Linear amplification
- ✓ UMIs
- ✓ Low bias
- ✓ Multiplexing
- ✗ Complex protocol

> 💡 Lower throughput but deeper sequencing per cell

Smart-seq2: Molecular Principle

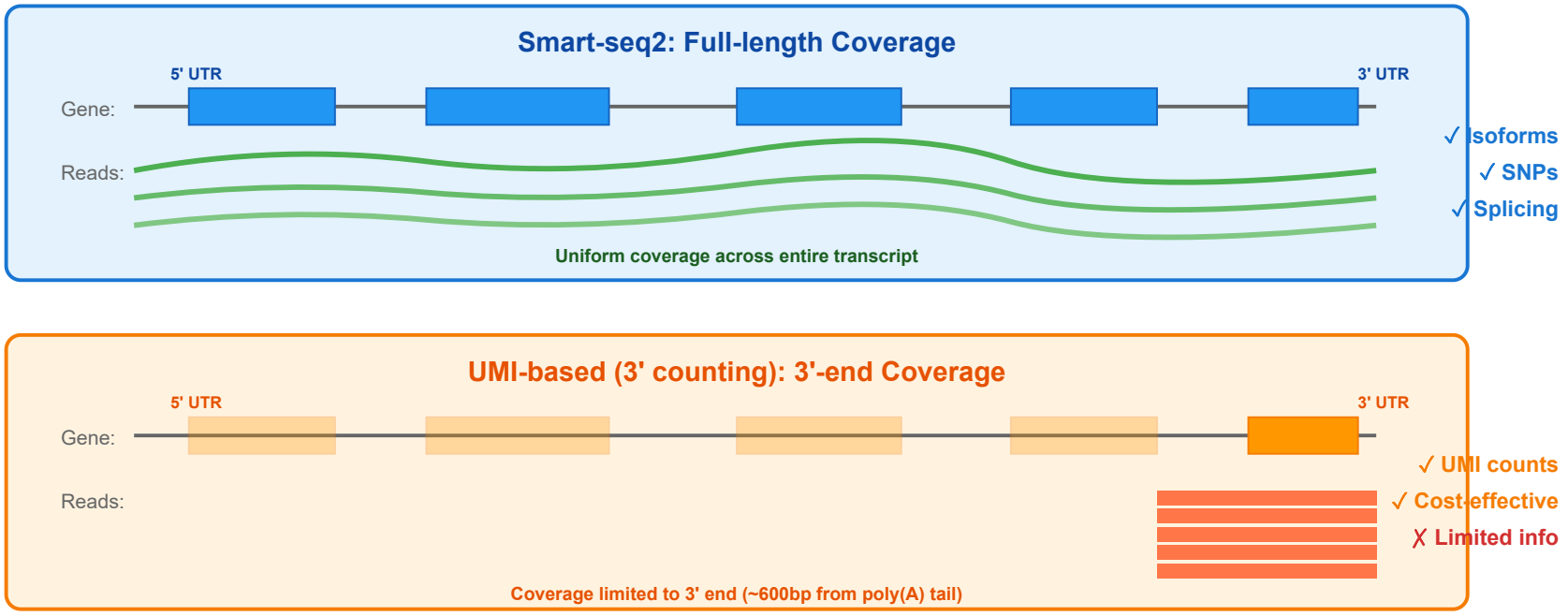


Key Innovation: Template Switching

- Reverse transcriptase adds non-templated CCC nucleotides at the 3' end of first-strand cDNA
- Template Switching Oligo (TSO) with GGG sequence hybridizes to CCC overhang
- RT switches templates and extends along TSO, adding universal priming site

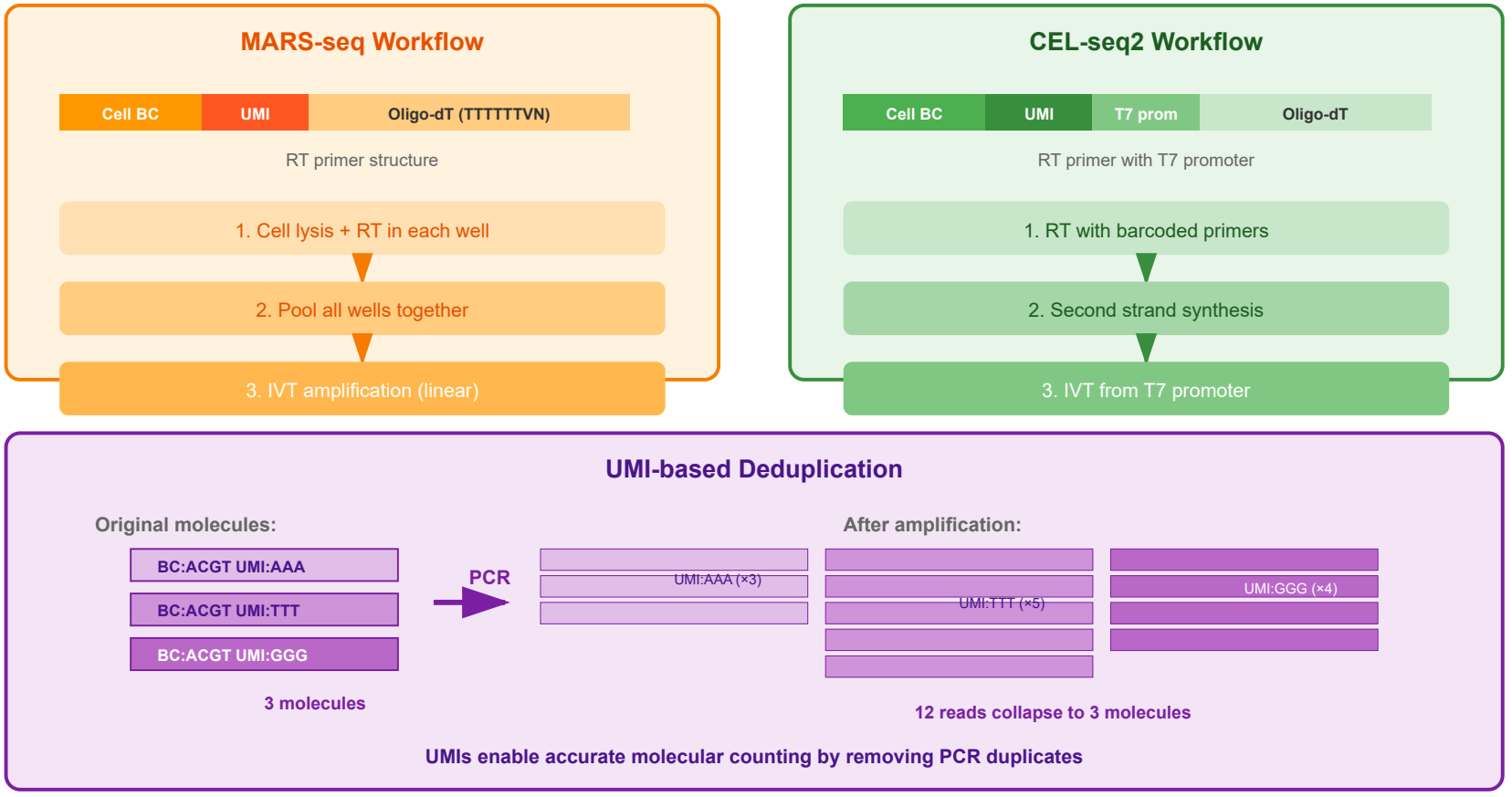
- Enables full-length transcript coverage from 5' to 3' end

Coverage Comparison: Smart-seq vs UMI-based Methods



Trade-off: Smart-seq provides rich molecular information but lacks UMIs for absolute quantification. UMI-based methods provide accurate molecule counting but limited transcript information.

UMI-based Plate Methods: MARS-seq & CEL-seq2



UMI Principle

Each original mRNA molecule is tagged with a unique random barcode (UMI) during reverse transcription. After PCR amplification, all reads with the same UMI are collapsed to count as a single molecule, eliminating amplification bias and enabling accurate quantification.

Detailed Method Comparison

Feature	Smart-seq2/3	MARS-seq	CEL-seq2
Coverage	Full-length transcripts	3' end only (~600bp)	3' end only
UMI	❌ No	✅ Yes (8bp)	✅ Yes (6bp)
Sensitivity	Highest (~10,000 genes/cell)	Moderate (~5,000 genes/cell)	Moderate (~5,000 genes/cell)
Amplification	PCR (exponential)	IVT (linear) then PCR	IVT (linear)
Bias	PCR amplification bias, 3' bias with degraded RNA	Strong 3' bias	Reduced bias (linear amp)
Cost/cell	High ($5-10)	Moderate ($1-3)	Moderate ($1-3)
Throughput	96-384 cells/batch	384-1536 cells/batch	96-384 cells/batch
Isoform detection	✅ Excellent	❌ Limited	❌ Limited
SNV/mutation	✅ Yes	❌ No	❌ No
RNA velocity	✅ Possible (intronic reads)	✅ Possible	✅ Possible
Automation	Moderate	High (robotic)	Moderate
Best use case	Deep characterization, rare cells, isoform analysis	Large-scale studies, cost-sensitive projects	Balanced approach, reduced bias requirements

Selection Guide: Choose Smart-seq for detailed molecular characterization and isoform analysis. Choose MARS-seq or CEL-seq2 for larger studies where accurate molecule counting is more important than full-length coverage.

Applications and Technical Considerations

When to Use Plate-based Methods

- **Rare cell populations:** When working with limited cell numbers or precious samples (e.g., 50-500 cells)
- **Quality control needs:** When microscopic inspection and selection of individual cells is critical
- **Alternative splicing studies:** Full-length coverage enables isoform detection and splice variant analysis
- **Genetic variation analysis:** SNP calling and allele-specific expression require full-length reads
- **Deep sequencing requirements:** When high read depth per cell is needed (e.g., 1-10M reads/cell)

Technical Considerations

Cell Viability and Quality

• FACS sorting can stress cells; minimize sort time and use gentle settings

• Cell viability should be >90% for optimal results

• Sort directly into lysis buffer to preserve RNA integrity

RNA Quality Requirements

• Smart-seq2: Very sensitive to RNA degradation; RIN >7 recommended

• UMI methods: More tolerant of degradation due to 3'-end focus

• Minimize freeze-thaw cycles; work quickly after cell sorting

Batch Effects

• Plate-based methods process cells in batches (96-384 wells)

• Include biological replicates across multiple plates

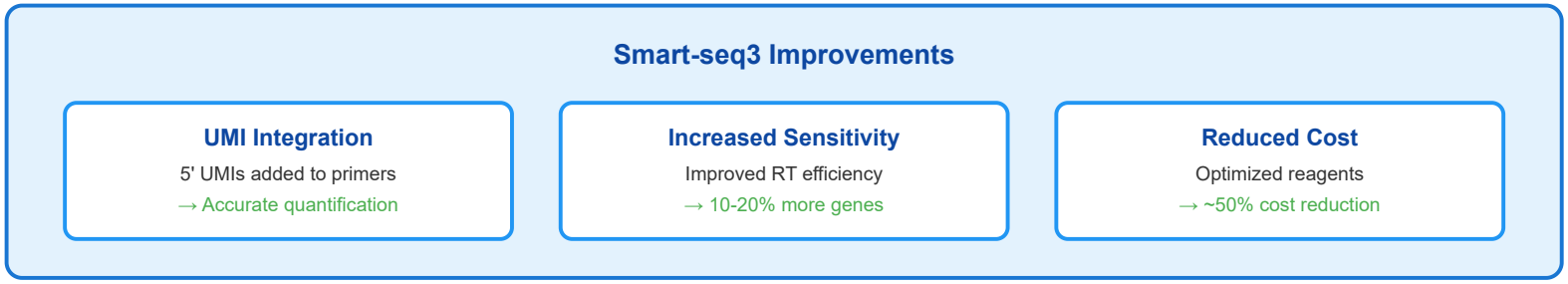
• Use spike-in controls (ERCC) to monitor technical variation

• Computational batch correction may be necessary

Cost-Throughput Trade-off: Plate-based methods cost more per cell but provide deeper information. For studies requiring >10,000 cells, consider droplet-based methods. For <1,000 cells with rich molecular detail needs, plate-based methods are optimal.

Recent Advances and Emerging Technologies

Smart-seq3



Multimodal Plate-based Methods

- **scNMT-seq:** Simultaneous measurement of RNA, DNA methylation, and chromatin accessibility in single cells
- **SMART-seqTOTAL:** Captures both poly(A) and non-poly(A) RNAs, including enhancer RNAs and lncRNAs
- **scGET-seq:** Combined genomic DNA and transcriptome sequencing for clonal tracking
- **PLATE-seq:** High-throughput plate-based method with increased automation and reduced costs

Future Directions

Integration with Spatial Information

Combining FACS-based spatial sorting with plate-based deep sequencing to link spatial context with detailed molecular profiles

Long-read Sequencing

Adaptation of plate-based methods for Oxford Nanopore and PacBio platforms, enabling full-length isoform characterization without fragmentation

Enhanced Automation

Development of fully automated liquid handling systems to increase throughput while maintaining quality, bridging the gap with droplet-based methods

> 💡 **The field is moving toward methods that combine the molecular depth of plate-based approaches with the throughput advantages of droplet-based systems**

Data Preprocessing for Single-Cell RNA-seq

Essential Steps for High-Quality Analysis

1. Cell Filtering

Remove low-quality cells, doublets, and empty droplets to ensure data integrity

2. Gene Filtering

Exclude lowly expressed genes and problematic gene categories

3. Normalization

Account for technical variations in sequencing depth and composition

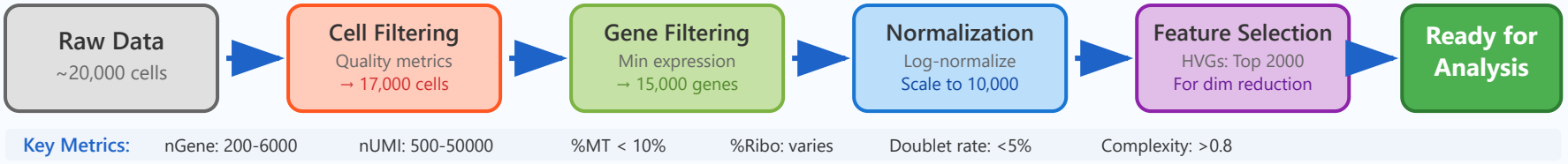
4. Imputation

Handle dropout events and technical zeros (use with caution)

5. Batch Effects

Identify and correct technical variations from sample processing

Quality Control Pipeline

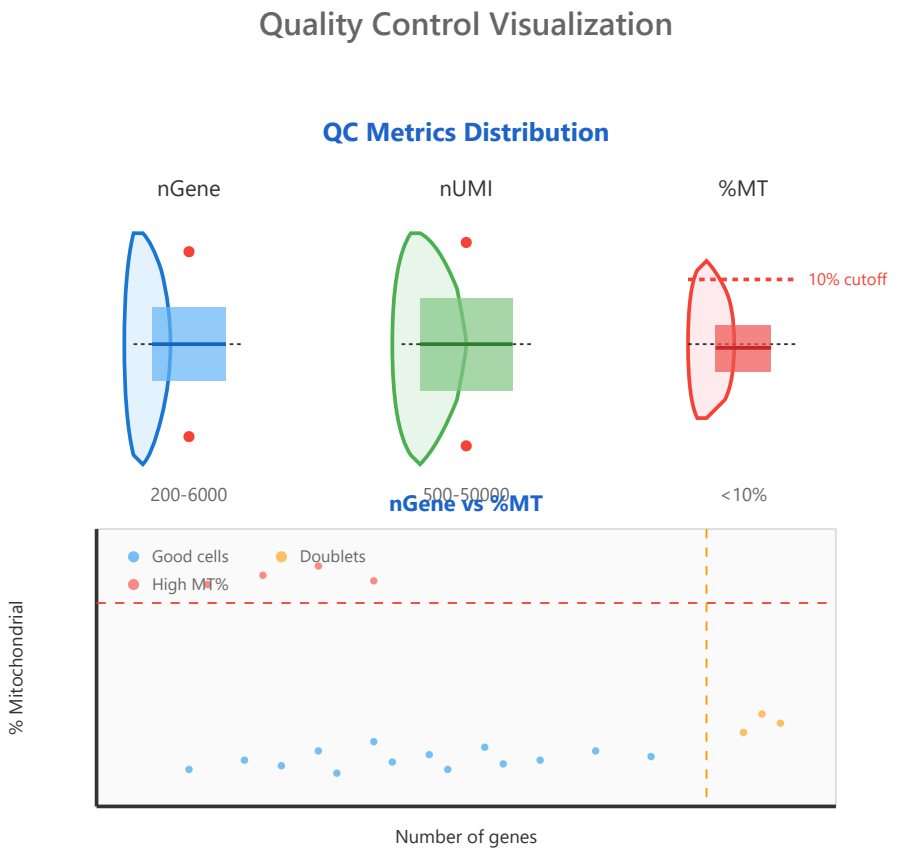


1 Cell Filtering

Cell filtering is the critical first step in single-cell RNA-seq analysis. This process removes low-quality cells, empty droplets, and doublets/multiplets that could compromise downstream analysis. High-quality cells are identified based on multiple quality control metrics that assess cellular integrity and technical artifacts.

Key Quality Metrics

- ✓ **nGene/nFeature:** Number of detected genes (typically 200-6000)
- ✓ **nUMI/nCount:** Total number of unique molecular identifiers
- ✓ **%Mitochondrial:** Percentage of reads mapping to MT genes (<10%)
- ✓ **%Ribosomal:** Percentage of reads from ribosomal genes
- ✓ **Complexity:** Library complexity (nGene/nUMI ratio)



Example Code (Scanpy/Python):

```
# Calculate QC metrics import scanpy as sc
sc.pp.calculate_qc_metrics(adata, qc_vars=['mt',
'ribo'], percent_top=None, log1p=False, inplace=True)
# Filter cells sc.pp.filter_cells(adata,
min_genes=200) sc.pp.filter_cells(adata,
max_genes=6000) adata =
adata[adata.obs['pct_counts_mt'] < 10, :]
```

⚠ **Important:** Thresholds should be dataset-specific. Always visualize distributions before setting cutoffs to avoid removing biologically relevant populations.

2 Gene Filtering

Gene filtering removes features that provide little information or could introduce noise into the analysis. This includes genes expressed in very few cells, housekeeping genes that show minimal variation, and genes from specific categories that might bias the analysis (e.g., mitochondrial, ribosomal genes for certain analyses).

Filtering Criteria

- ✓ **Minimum cells:** Remove genes expressed in <3-10 cells
- ✓ **Mitochondrial genes:** Often excluded from HVG selection

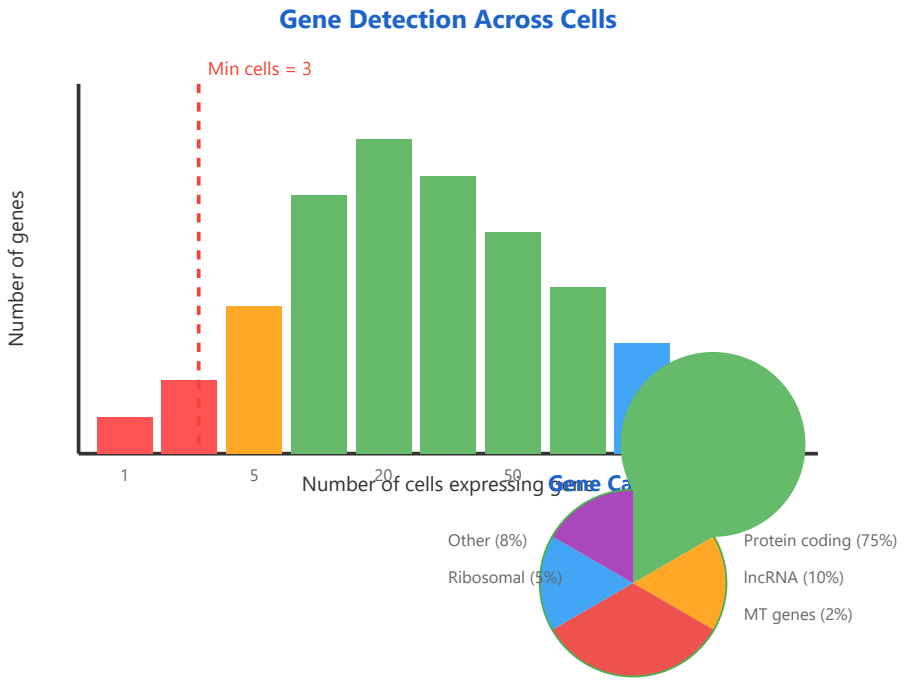
Gene Expression Distribution

- ✓ **Ribosomal genes:** May be removed for specific analyses
- ✓ **Cell cycle genes:** Can be regressed out if needed
- ✓ **Sex-specific genes:** Consider removing for certain studies

Example Code (Seurat/R):

```
# Filter genes library(Seurat) # Keep genes expressed in at least 3 cells min.cells <- 3 genes.use <- rowSums(counts > 0) >= min.cells counts <- counts[genes.use, ] # Remove MT and ribosomal genes mt.genes <- grep("^MT-", rownames(counts)) ribo.genes <- grep("^RP[SL]", rownames(counts)) counts <- counts[-c(mt.genes, ribo.genes), ]
```

💡 **Tip:** Consider keeping mitochondrial and ribosomal genes in the initial dataset for QC purposes, then exclude them during feature selection for dimensionality reduction.



3 Normalization Methods

Normalization corrects for technical differences between cells, particularly sequencing depth variations. Different normalization methods are suited for different data types and analysis goals. The choice of normalization can significantly impact downstream results, especially clustering and differential expression.

Normalization Effects

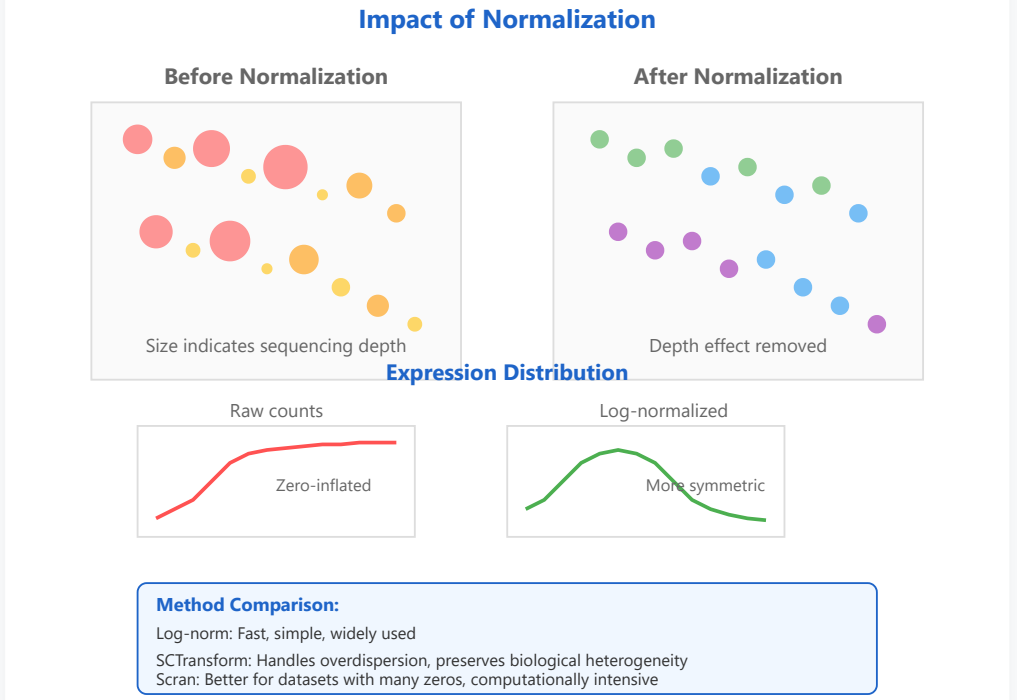
Common Methods

- ✓ **Log-normalization:** Scale to 10,000 UMIs + log transform (most common)
- ✓ **SCTransform:** Regularized negative binomial regression
- ✓ **Scran pooling:** Pool-based size factor estimation
- ✓ **TMM/DESeq2:** Methods adapted from bulk RNA-seq
- ✓ **Pearson residuals:** Variance-stabilizing transformation

Example Code (Multiple Methods):

```
# Method 1: Standard log-normalization
sc.pp.normalize_total(adata, target_sum=1e4)
sc.pp.log1p(adata) # Method 2: SCTransform (Seurat)
library(Seurat) seurat_obj <- SCTransform(seurat_obj,
vars.to.regress = "percent.mt", verbose = FALSE) #
Method 3: Scran normalization library(scran) sce <-
computeSumFactors(sce) sce <- logNormCounts(sce)
```

⚠ **Note:** Different normalization methods make different assumptions. Log-normalization assumes similar total RNA content across cells, which may not hold for all cell types.



4 Imputation Strategies

Imputation methods attempt to recover missing expression values (zeros) that may represent technical dropouts rather than true biological absence. However, imputation is controversial as it can introduce false signals and should be used cautiously, particularly for differential expression analysis.

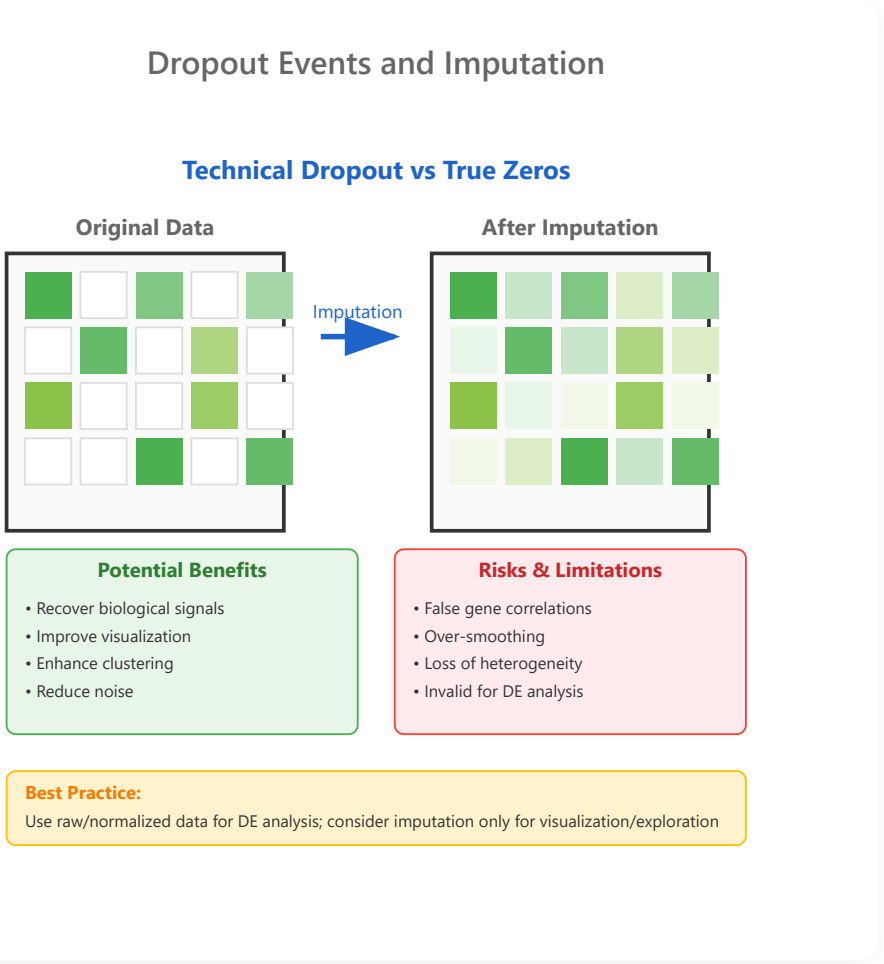
Imputation Approaches

- ✓ **MAGIC:** Markov affinity-based graph imputation
- ✓ **scImpute:** Statistical model-based imputation
- ✓ **SAVER:** Bayesian approach using gene-gene relationships
- ✓ **DCA:** Deep count autoencoder
- ✓ **No imputation:** Often the safest choice

Example Code (MAGIC):

```
import magic import scanpy as sc # Create MAGIC
operator magic_op = magic.MAGIC() # Run imputation
adata_magic = adata.copy()
magic_op.fit_transform(adata_magic.X) # Compare
before/after sc.pl.scatter(adata, x='gene1',
y='gene2', title='Original')
sc.pl.scatter(adata_magic, x='gene1', y='gene2',
title='After MAGIC')
```

⚠ **Caution:** Imputation can create false gene-gene correlations and should generally be avoided for differential expression analysis. Consider using methods designed to handle sparsity instead.



5 Batch Effect Correction

Batch effects are systematic technical variations introduced during sample processing, sequencing, or data generation. These can confound biological signals and must be carefully identified and corrected. Modern methods can integrate multiple datasets while preserving biological variation.

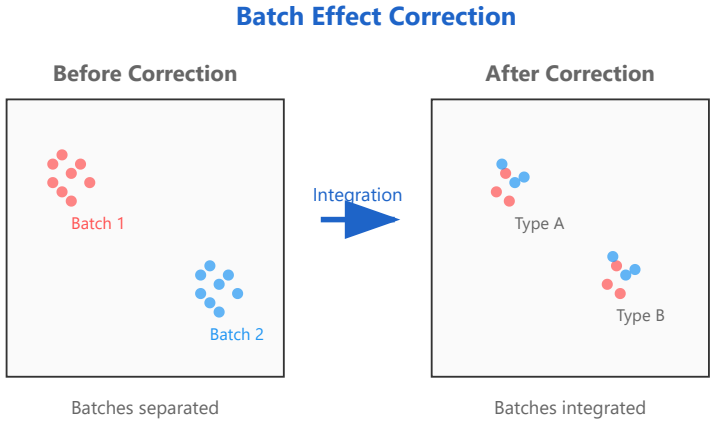
Integration Methods

- ✓ **Harmony:** Fast iterative correction in PCA space
- ✓ **Seurat Integration:** Anchor-based integration using CCA
- ✓ **scVI:** Variational autoencoder approach
- ✓ **MNN:** Mutual nearest neighbors correction
- ✓ **ComBat:** Empirical Bayes batch correction
- ✓ **LIGER:** Integrative non-negative matrix factorization

Example Code (Harmony):

```
import scanpy as sc import scanpy.external as sce # Compute PCA first sc.pp.pca(adata) # Run Harmony integration sce.pp.harmony_integrate( adata, key='batch', # batch variable in obs basis='X_pca', adjusted_basis='X_pca_harmony' ) # Use corrected
```

Batch Effect Visualization



Integration Method Comparison

Method	Speed	Scalability	Preservation
Harmony	Fast	Excellent	Good
Seurat CCA	Medium	Good	Excellent
scVI	Slow	Excellent	Excellent
ComBat	Fast	Limited	Variable

Key Consideration:
Always check biological signals are preserved post-correction using known markers

```
embeddings for UMAP sc.pp.neighbors(adata,
use_rep='X_pca_harmony') sc.tl.umap(adata)
```

💡 **Best Practice:** Always visualize data before and after batch correction. Check that biological variation (e.g., cell types) is preserved while technical variation is removed.

Processing Workflow Summary

1. **Quality Control:** Assess cell and gene quality metrics
2. **Cell Filtering:** Remove low-quality cells based on thresholds
3. **Gene Filtering:** Exclude lowly expressed and problematic genes
4. **Normalization:** Account for technical variations
5. **Feature Selection:** Identify highly variable genes
6. **Batch Correction:** Integrate multiple datasets if needed
7. **Dimensionality Reduction:** PCA, then UMAP/tSNE for visualization

> **Remember:** Each dataset is unique. Always visualize your data at each step and adjust parameters based on your specific biological system and experimental design.

Dimensionality Reduction Techniques

A Comprehensive Guide for Single-Cell RNA-seq Analysis

PCA for scRNA-seq

Linear dimensionality reduction technique that identifies directions of maximum variance in high-dimensional data

t-SNE Principles

Non-linear technique that preserves local structure through probability distributions

UMAP Advantages

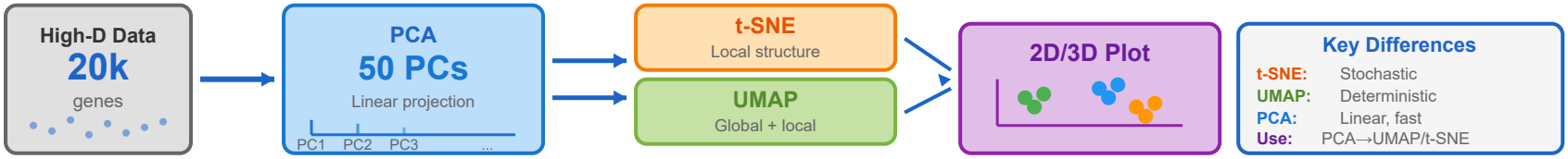
Fast manifold learning technique preserving both local and global data structure

Diffusion Maps

Captures continuous trajectories and developmental processes in biological data

Parameter Selection

Critical parameters like perplexity and n_neighbors significantly affect visualization results



PCA Principal Component Analysis (PCA)

Overview

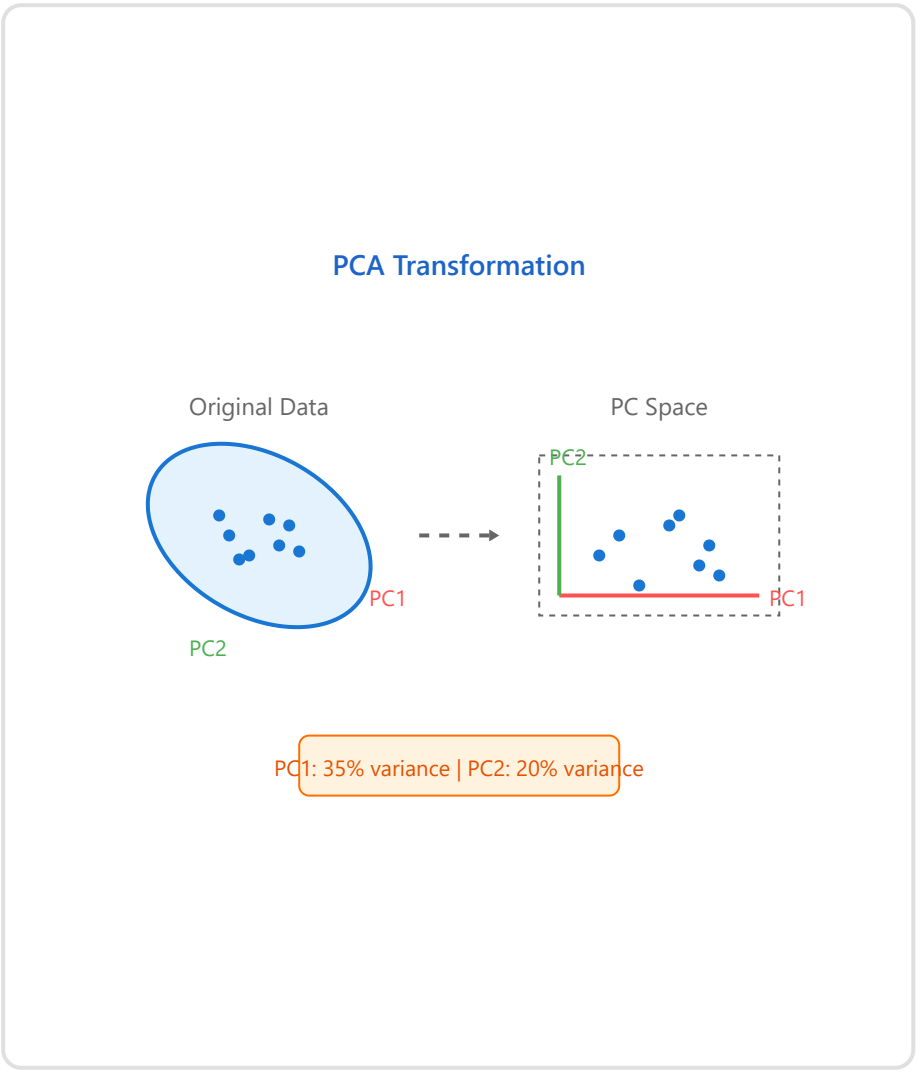
PCA is a linear dimensionality reduction technique that identifies orthogonal axes (principal components) along which the data shows maximum variance. In scRNA-seq analysis, PCA is typically the first step after normalization and feature selection, reducing thousands of genes to 10-50 principal components while preserving most biological variation.

Mathematical Foundation

PCA performs eigendecomposition of the covariance matrix or singular value decomposition (SVD) of the data matrix. Each PC is a linear combination of original features, with coefficients (loadings) indicating each gene's contribution to that component.

Key Parameters

- n_components: Number of PCs to compute (typically 10-50)
- svd_solver: Algorithm choice ('auto', 'full', 'arpack', 'randomized')
- whiten: Whether to normalize PC scores



Advantages

✓ Computationally efficient for large datasets
✓ Mathematically well-understood

Limitations

✗ Assumes linear relationships
✗ May miss non-linear patterns

✓ Preserves global structure
✓ Deterministic results

✗ Sensitive to outliers
✗ Not ideal for visualization

```python
# Python example using Scanpy
import scanpy as sc

# Perform PCA on normalized data
sc.pp.pca(adata, n_comps=50, svd_solver='arpack')

# Visualize variance ratio
sc.pl.pca_variance_ratio(adata, n_pcs=50)

# Plot PCA results
sc.pl.pca(adata, color='cell_type', components=['1,2', '2,3'])
```

tSNE t-Distributed Stochastic Neighbor Embedding (t-SNE)

Overview

t-SNE is a non-linear dimensionality reduction technique that models pairwise similarities between data points in high-dimensional space and iteratively adjusts their representation in low-dimensional space to preserve these similarities. It excels at revealing local structure and finding clusters in complex datasets.

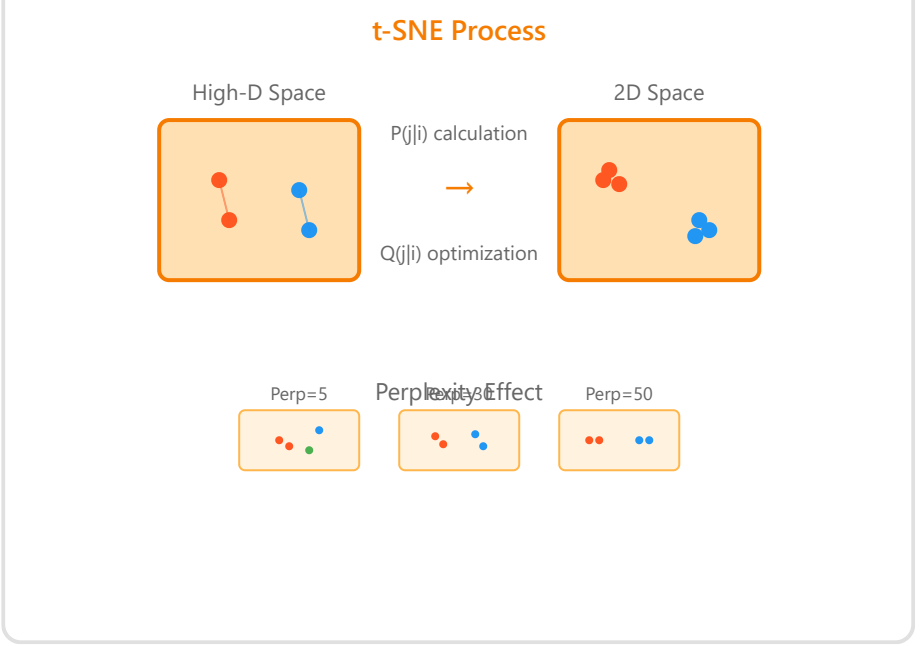
Algorithm Process

t-SNE converts high-dimensional Euclidean distances into conditional probabilities representing similarities. It uses a Student

t-distribution in the low-dimensional space to allow dissimilar objects to be modeled far apart, addressing the "crowding problem" of traditional SNE.

Critical Parameters

- perplexity: Balance between local and global aspects (5-50)
- learning_rate: Step size for gradient descent (10-1000)
- n_iter: Number of iterations (250-1000)
- early_exaggeration: Factor for spacing clusters (4-12)



Advantages

✓ Excellent for visualizing clusters
✓ Preserves local structure well
✓ Reveals complex patterns
✓ Good for exploratory analysis

Limitations

✗ Computationally expensive
✗ Non-deterministic (stochastic)
✗ Distorts global structure
✗ Cannot embed new data

```python
# Python example using Scanpy
import scanpy as sc

# Run t-SNE on PCA results
sc.tl.tsne(adata,
    perplexity=30,
    learning_rate=200,
    n_iter=1000,
    random_state=42)

# Visualize results
sc.pl.tsne(adata, color=['cell_type', 'n_genes'])
```

Uniform Manifold Approximation and Projection (UMAP)

Overview

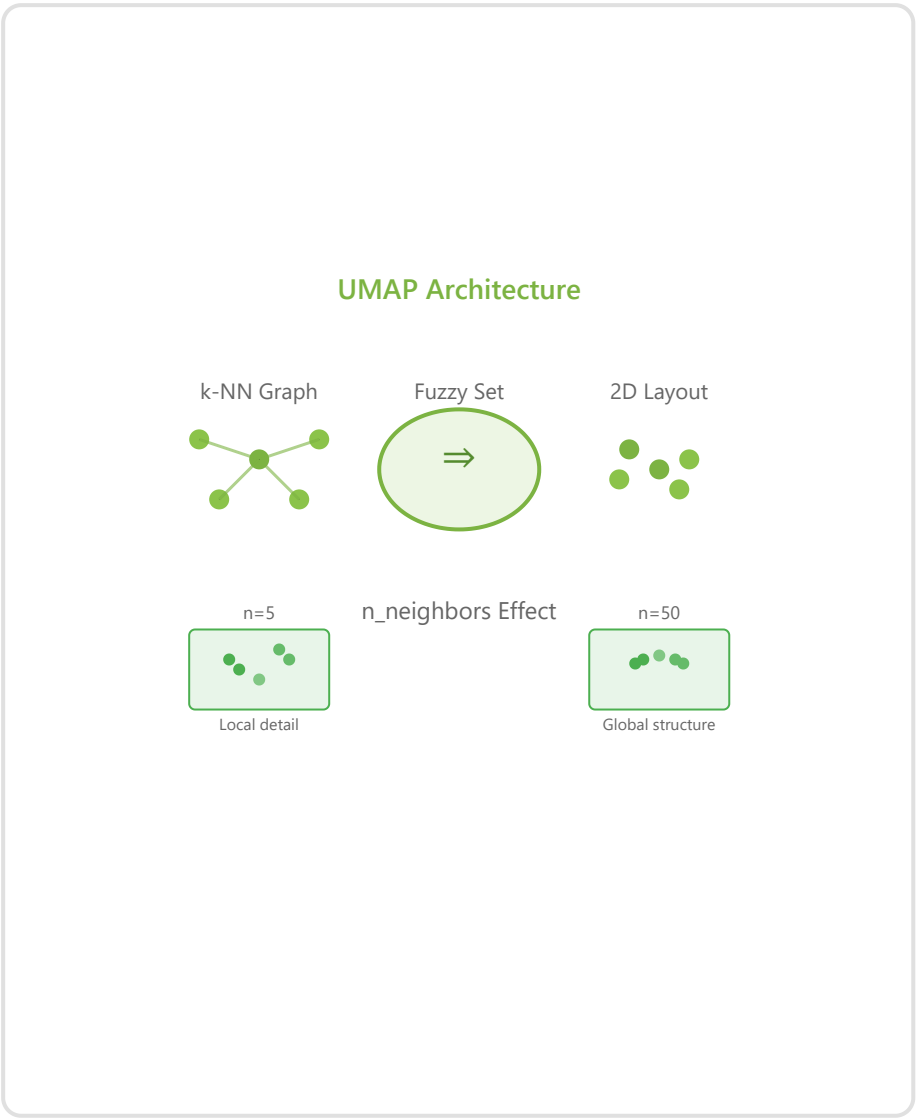
UMAP is a manifold learning technique based on Riemannian geometry and algebraic topology. It constructs a high-dimensional graph representation of the data and optimizes a low-dimensional graph to be as structurally similar as possible, preserving both local and global structure better than t-SNE.

Theoretical Foundation

UMAP assumes data is uniformly distributed on a Riemannian manifold and models the manifold with a fuzzy topological structure. The algorithm uses a novel fuzzy simplicial set representation and optimizes the cross-entropy between high and low-dimensional representations.

Important Parameters

- n_neighbors: Size of local neighborhood (5-50)
- min_dist: Minimum distance between points (0.0-1.0)
- metric: Distance metric ('euclidean', 'manhattan', etc.)
- n_components: Output dimensions (typically 2 or 3)



Advantages

✓ Faster than t-SNE

Limitations

✗ Newer, less established method

✓ Preserves global structure better
✓ More deterministic results
✓ Can handle larger datasets
✓ Supports various distance metrics

✗ Can create artificial connections
✗ Sensitive to hyperparameters
✗ Interpretation can be challenging

```python
# Python example using Scanpy
import scanpy as sc

# Compute neighborhood graph
sc.pp.neighbors(adata, n_neighbors=15, n_pcs=50)

# Run UMAP
sc.tl.umap(adata,
    min_dist=0.3,
    spread=1.0,
    n_components=2,
    random_state=42)

# Visualize with different parameters
sc.pl.umap(adata, color=['leiden', 'n_counts'], legend_loc='on data')
```

DM Diffusion Maps

Overview

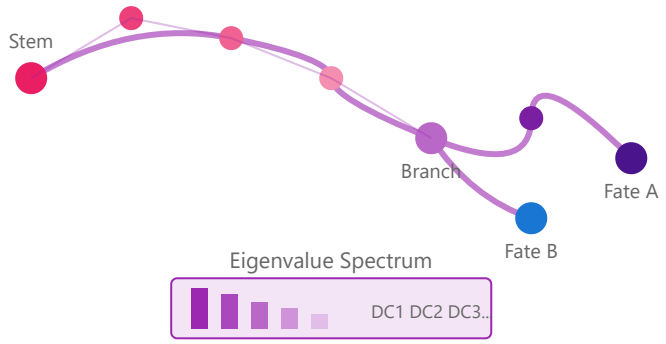
Diffusion Maps is a non-linear dimensionality reduction technique that models data as lying on a manifold and uses a diffusion process to discover its underlying geometry. It's particularly powerful for capturing continuous trajectories and branching

processes in biological systems, making it ideal for developmental studies.

Mathematical Principle

The method constructs a Markov chain on the data, where transition probabilities reflect local geometry. Eigenvectors of the transition matrix provide coordinates that respect the manifold's intrinsic geometry, with diffusion distance capturing connectivity along the manifold rather than Euclidean distance.

Diffusion Process



Key Parameters

- n_components: Number of diffusion components
- knn: Number of nearest neighbors for graph
- decay: Rate of kernel decay (bandwidth)
- t: Diffusion time parameter

Advantages

✓ Captures continuous processes well
✓ Robust to noise
✓ Preserves branching structure
✓ Good for trajectory inference
✓ Respects data geometry

Limitations

✗ Computationally intensive
✗ Parameter selection crucial
✗ Less intuitive visualization
✗ May oversmooth data

```
# Python example using Scanpy/Palantir
import scanpy as sc
import palantir

# Compute diffusion maps
sc.tl.diffmap(adata, n_comps=10)

# Run Palantir for trajectory analysis
```

```
dm_res = palantir.utils.run_diffusion_maps(adata.obsm['X_pca'], n_components=5)

# Compute pseudotime
pr_res = palantir.core.run_palantir(
    dm_res['EigenVectors'],
    start_cell,
    num_waypoints=500)

# Visualize diffusion components
sc.pl.diffmap(adata, color='pseudotime', components=['1,2', '2,3'])
```

Parameter Selection Guidelines

Overview

Proper parameter selection is crucial for meaningful dimensionality reduction results. Parameters control the balance between local and global structure preservation, computational efficiency, and biological interpretability. Understanding how parameters affect results helps avoid artifacts and reveals true biological patterns.

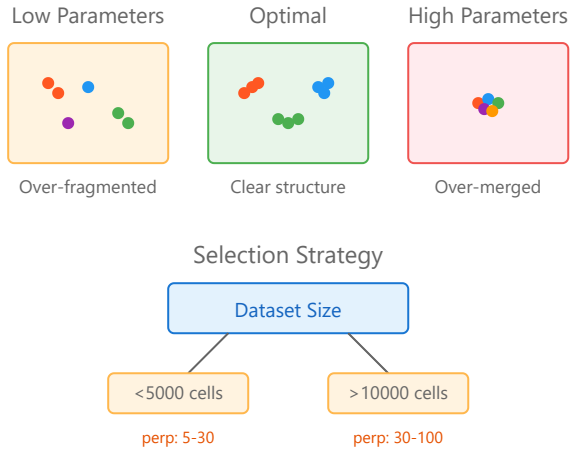
General Principles

Start with default parameters and systematically vary them to understand their effects. Consider your biological question: local structure (cell types) vs global structure (trajectories). Dataset size and sparsity influence optimal parameter choices. Always validate results using known markers and biological knowledge.

Critical Decisions

- Number of PCs: Elbow plot, technical variance
- Perplexity (t-SNE): 5-50, scale with N^0.5
- n_neighbors (UMAP): 5-50, affects connectivity
- min_dist (UMAP): 0.0-1.0, cluster tightness

Parameter Impact Comparison



Best Practices

✓ Test multiple parameter sets
✓ Use biological validation
✓ Consider computational resources
✓ Document parameter choices
✓ Check stability across runs

Common Pitfalls

✗ Using default parameters blindly
✗ Over-interpreting distances
✗ Ignoring batch effects
✗ Not checking multiple seeds
✗ Forcing biological interpretation

```
# Parameter optimization example
import scanpy as sc
import numpy as np

# Determine optimal number of PCs
sc.pp.pca(adata, n_comps=100)
sc.pl.pca_variance_ratio(adata, n_pcs=100, log=True)

# Test different perplexity values for t-SNE
perplexities = [5, 15, 30, 50]
for perp in perplexities:
    sc.tl.tsne(adata, perplexity=perp, use_rep='X_pca')
    adata.obsm[f'X_tsne_perp{perp}'] = adata.obsm['X_tsne'].copy()

# Grid search for UMAP parameters
```

```
n_neighbors_list = [5, 15, 30, 50]
min_dist_list = [0.0, 0.1, 0.3, 0.5]

for nn in n_neighbors_list:
    for md in min_dist_list:
        sc.pp.neighbors(adata, n_neighbors=nn)
        sc.tl.umap(adata, min_dist=md)
        # Evaluate clustering quality
        sc.tl.leiden(adata)
        # Calculate silhouette score or other metrics
```

Method Comparison Matrix

Method	Type	Speed	Global Structure	Local Structure	Deterministic	Best Use Case
PCA	Linear	Very Fast	Excellent	Poor	Yes	Initial reduction, QC
t-SNE	Non-linear	Slow	Poor	Excellent	No	Cluster visualization
UMAP	Non-linear	Fast	Good	Excellent	Mostly	General visualization
Diffusion Maps	Non-linear	Medium	Good	Good	Yes	Trajectory analysis
Force Atlas 2	Graph-based	Medium	Fair	Good	No	Network visualization

Key Takeaways

- **Visualization ≠ Analysis:** Use these methods for exploration, not quantitative conclusions

- **Pipeline approach:** PCA → Clustering → UMAP/t-SNE for visualization
- **Validate findings:** Always confirm patterns with known markers and biology
- **Parameter sensitivity:** Test multiple settings and report those used

Clustering Methods in Single-Cell Analysis

Graph-based Clustering

Build kNN graph then find communities

Leiden Algorithm

Improved Louvain with better guarantees

K-means Adaptations

SC3 uses consensus clustering

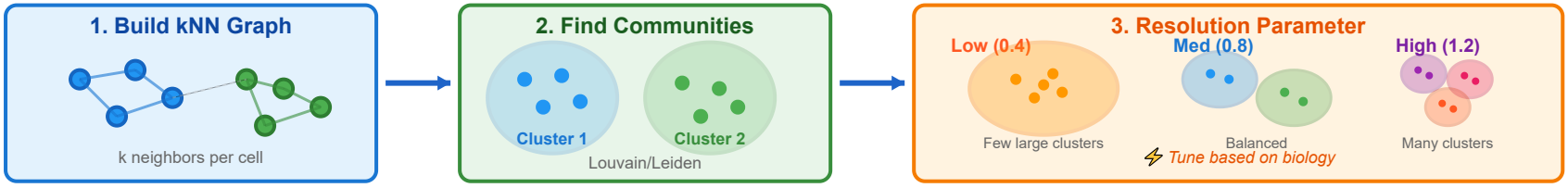
Resolution Selection

Higher resolution = more clusters

Stability Analysis

Bootstrap to assess cluster robustness

💡 No single correct clustering - depends on biological question



Detailed Method Explanations

1 Graph-based Clustering

Graph-based clustering represents cells as nodes in a network where edges connect similar cells. This approach naturally captures the manifold structure of single-cell data.

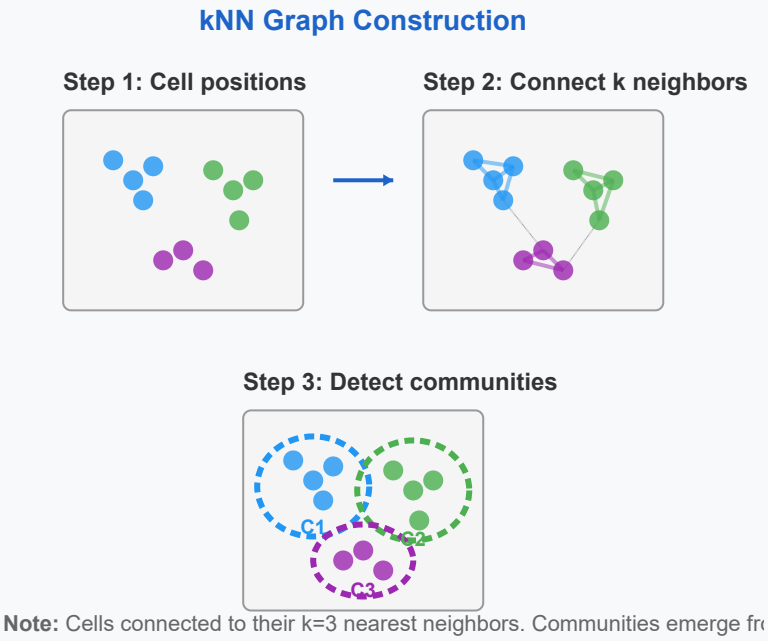
Key Steps:

- **Graph Construction:** Build a k-nearest neighbor (kNN) graph where each cell is connected to its k most similar cells based on gene expression distances
- **Edge Weighting:** Edges are weighted by similarity (e.g., Euclidean distance in PCA space or cosine similarity)
- **Community Detection:** Apply algorithms like Louvain or Leiden to identify densely connected groups (communities)

Advantages:

- Handles complex, non-convex cluster shapes naturally
- Computationally efficient for large datasets
- Robust to noise and outliers
- Standard approach in Seurat and Scanpy

> **Typical Parameter:** k = 10-30 neighbors is common for most scRNA-seq datasets. Higher k creates more connections but may merge distinct populations.



Note: Cells connected to their k=3 nearest neighbors. Communities emerge fr

Leiden Algorithm

The Leiden algorithm is an improved community detection method that addresses key limitations of the popular Louvain algorithm, particularly ensuring well-connected communities.

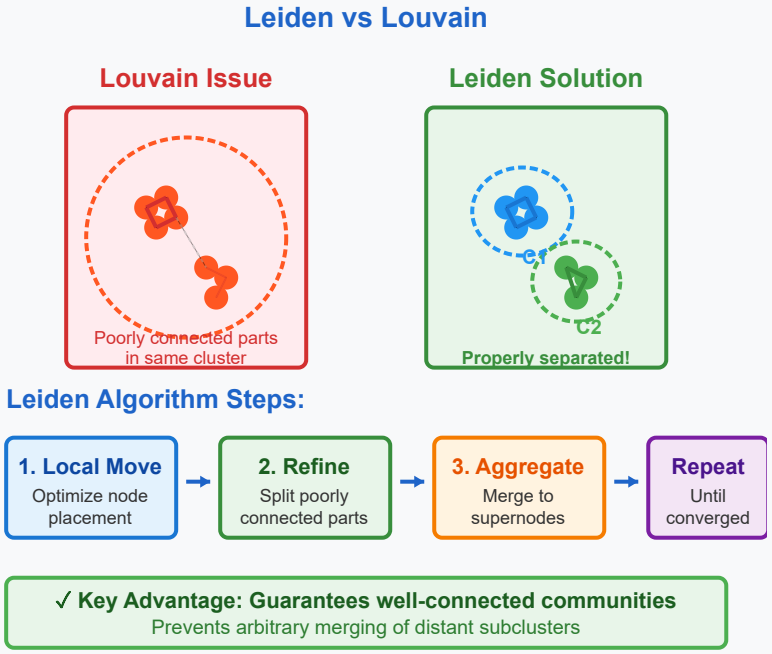
Improvements over Louvain:

- **Guarantees Connectivity:** Ensures communities are internally connected, preventing poorly connected subcommunities
- **Faster Convergence:** Typically requires fewer iterations to reach optimal partitioning
- **Quality Function:** Optimizes modularity while maintaining community quality
- **Refinement Phase:** Includes additional refinement step to split disconnected parts

Algorithm Phases:

- **Local Moving:** Move nodes between communities to increase modularity
- **Refinement:** Split weakly connected subcommunities
- **Aggregation:** Collapse communities into supernodes
- **Repeat:** Iterate until no improvement

Why it matters: Leiden is the recommended algorithm in Scanpy and provides more reliable clustering results,



especially for complex datasets with hierarchical structure.

```
# Python (Scanpy)
sc.tl.leiden(adata, resolution=0.8)

# R (Seurat) - uses Louvain by default
adata <- FindClusters(adata, resolution=0.8,
algorithm=4) # 4 = Leiden
```

3 K-means Adaptations (SC3)

SC3 (Single-Cell Consensus Clustering) adapts k-means clustering for single-cell data through consensus approaches and careful parameter selection.

SC3 Methodology:

- **Multiple Transformations:** Apply different distance metrics and dimensionality reductions
- **Consensus Clustering:** Run k-means multiple times with different initializations
- **Similarity Matrix:** Build consensus across multiple runs to stabilize results

- **Optimal k Selection:** Uses silhouette width and other metrics to estimate cluster number

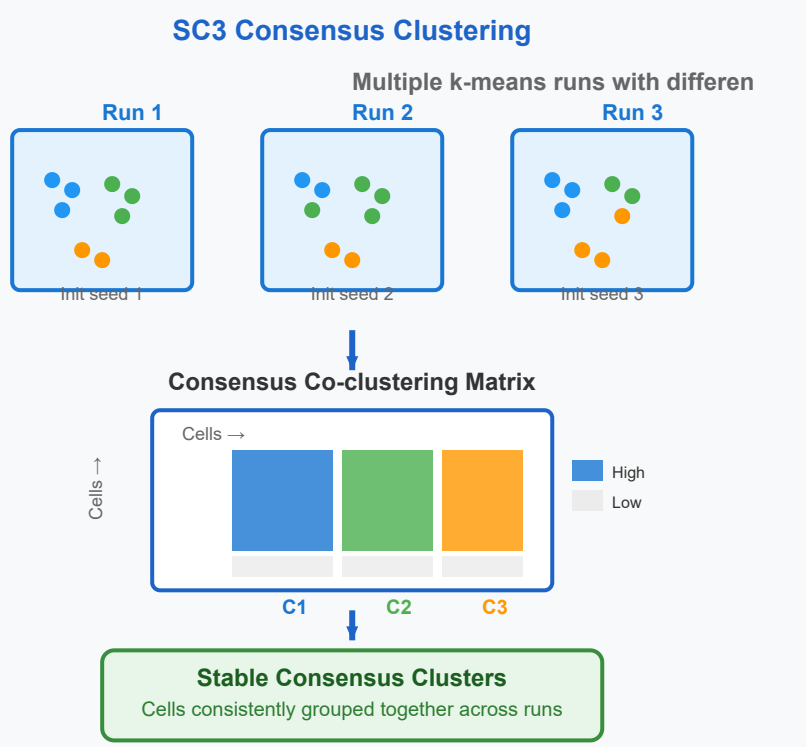
Why Consensus?

- K-means is sensitive to initialization (local minima problem)
- Different random starts can produce different results
- Consensus approach averages over many runs for stability
- Identifies core cluster memberships that persist across runs

Limitations:

- Assumes spherical clusters (unlike graph-based methods)
- Requires specifying k (number of clusters) in advance
- Computationally intensive for large datasets
- Less popular than graph-based methods for scRNA-seq

> **When to use:** SC3 is useful for smaller datasets where you want robust cluster number estimation and are willing to invest computational time.



4 Resolution Parameter Selection

The resolution parameter controls the granularity of clustering in community detection algorithms. This is one of the most important tuning parameters in single-cell clustering.

How Resolution Works:

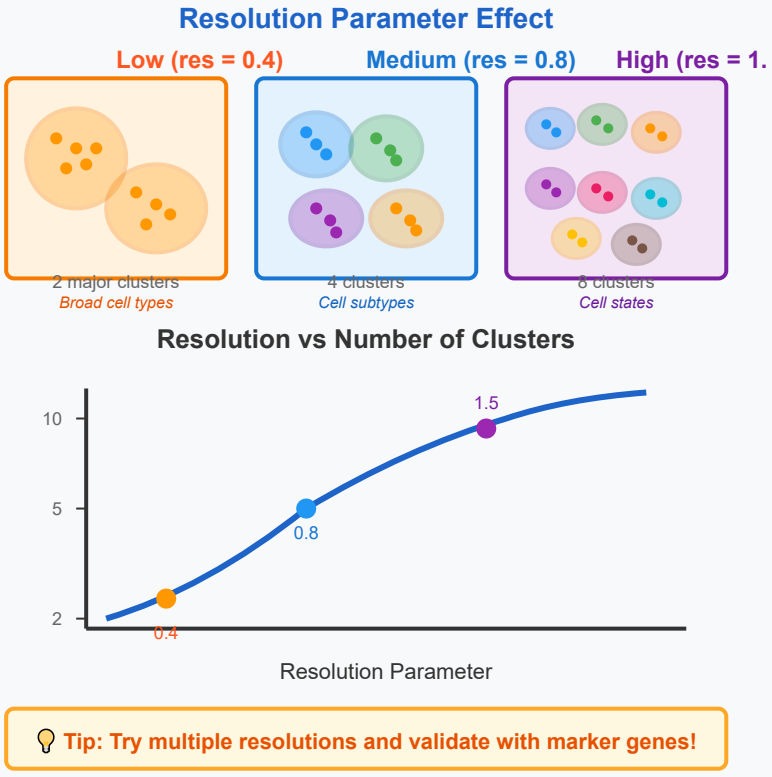
- **Low Resolution (0.1-0.5):** Favors larger communities, merges similar populations
- **Medium Resolution (0.6-1.0):** Balanced granularity, commonly used default
- **High Resolution (1.0-2.0+):** Produces more fine-grained clusters, separates subtle differences

Biological Interpretation:

- **Cell Type Level:** Lower resolution (0.4-0.6) for major cell types (T cells, B cells, monocytes)
- **Subtype Level:** Medium resolution (0.8-1.2) for cell subtypes (CD4+ T cells, CD8+ T cells)
- **State Level:** Higher resolution (1.5-2.0) for cell states (activated, resting, proliferating)

Selection Strategies:

- Start with default (0.8-1.0) and adjust based on results
- Use multiple resolutions and compare with biological knowledge
- Check cluster stability at each resolution
- Consider the biological question being asked



Tip: Try multiple resolutions and validate with marker genes!

Important: There is no "correct" resolution - it depends on your research question. Are you interested in broad cell types or fine-grained states?

5 Cluster Stability Analysis

Stability analysis assesses how robust clustering results are to perturbations in the data. This helps identify reliable clusters versus those that may be artifacts.

Bootstrap Resampling:

- **Process:** Randomly sample cells with replacement, recluster multiple times
- **Stability Score:** Measure how often cells cluster together across bootstrap runs
- **Interpretation:** High stability (>0.8) indicates robust clusters; low stability suggests uncertain boundaries

Subsampling Analysis:

- Remove random subsets of cells and recluster
- Check if major clusters persist
- Identifies clusters dependent on specific cells

Metrics for Stability:

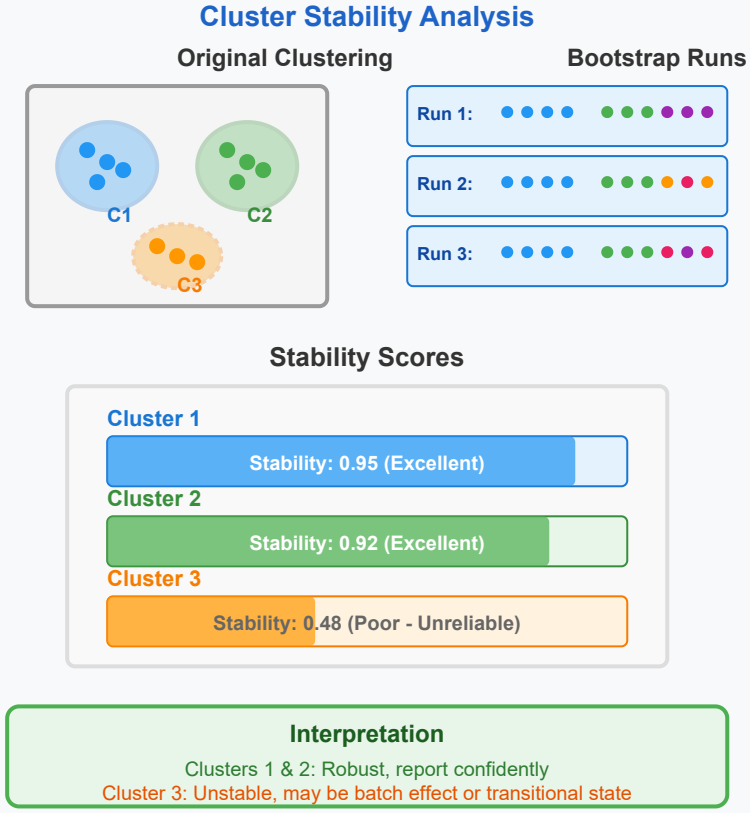
- **Adjusted Rand Index (ARI):** Measures agreement between clusterings (0=random, 1=perfect)
- **Jaccard Index:** Overlap between cluster memberships
- **Co-clustering Frequency:** How often cell pairs are grouped together

Practical Applications:

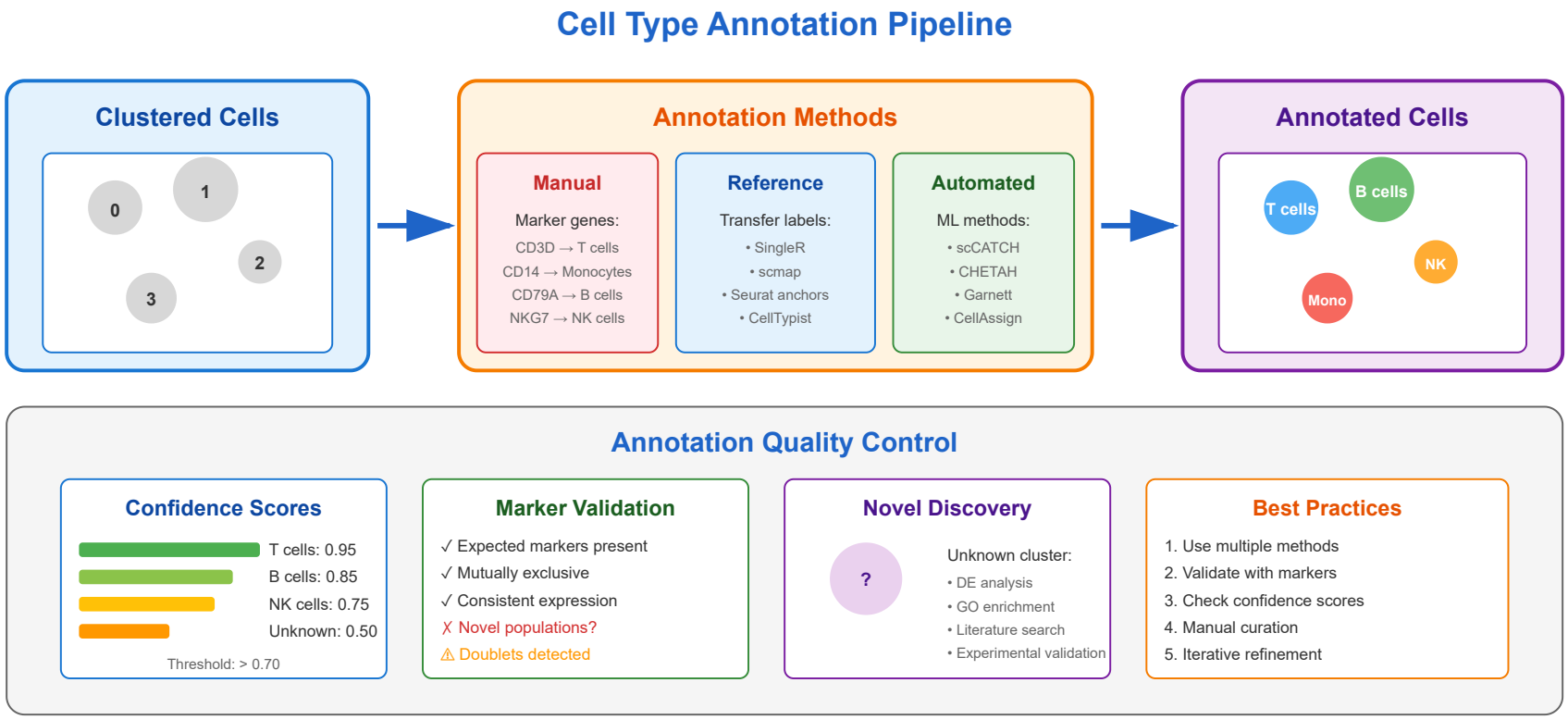
- Identify stable core clusters to report in publications
- Flag uncertain cells at cluster boundaries
- Compare stability across different resolutions
- Validate that biological conclusions don't depend on unstable clusters

> **Best Practice:** Always perform stability analysis before biological interpretation. Don't over-interpret unstable clusters!

```
# Python example with clustree-like approach
for i in range(100): # 100 bootstrap runs
    subsample =
adata[np.random.choice(adata.n_obs,
                size=adata.n_obs, replace=True)]
    sc.tl.leiden(subsample, resolution=0.8)
    # Calculate co-clustering matrix
```



Cell Type Annotation



💡 Combine automated tools with manual curation for optimal results

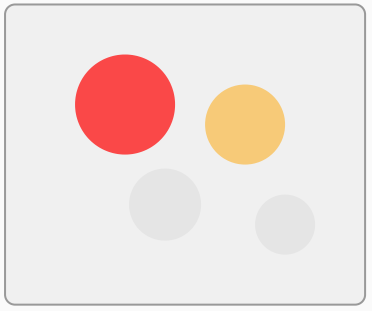
1. Manual Annotation Methods

Marker Gene-Based Identification

Manual annotation relies on expert knowledge to identify cell types based on the expression patterns of known marker genes. This approach requires deep biological understanding and literature review but provides high-quality, interpretable results.

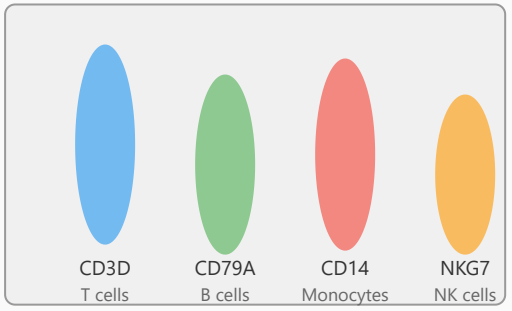
1. **Identify Differential Markers**
   Run differential expression analysis to find genes enriched in each cluster
2. **Visualize Gene Expression**
   Generate feature plots, violin plots, and heatmaps for known markers
3. **Literature Comparison**
   Cross-reference expression patterns with published cell type signatures
4. **Assign Cell Type Labels**
   Manually label clusters based on marker combinations and biological knowledge

Feature Plot: CD3D Expression

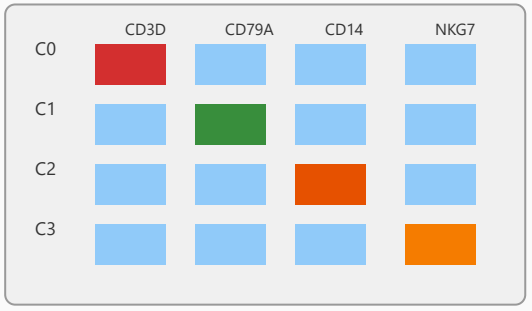


High CD3D = T cells

Violin Plot: Marker Genes



Heatmap: Top Markers



✅ Advantages

- Highly interpretable and biologically meaningful
- Leverages existing biological knowledge
- Flexible and adaptable to novel cell types
- No need for reference datasets

⚠️ Limitations

- Time-consuming and labor-intensive
- Requires extensive domain expertise
- Subjective and prone to bias
- Not scalable for large datasets

- Direct visual inspection possible

- Difficult to maintain consistency across studies

💻 Example: Manual Annotation in Seurat (R)

```
# Find cluster markers cluster_markers <- FindAllMarkers(seurat_obj, only.pos = TRUE) # Visualize key markers FeaturePlot(seurat_obj, features = c("CD3D", "CD79A", "CD14", "NKG7")) # Assign cell type labels based on markers new.cluster.ids <- c("T cells", "B cells", "Monocytes", "NK cells") names(new.cluster.ids) <- levels(seurat_obj) seurat_obj <- RenameIdents(seurat_obj, new.cluster.ids)
```

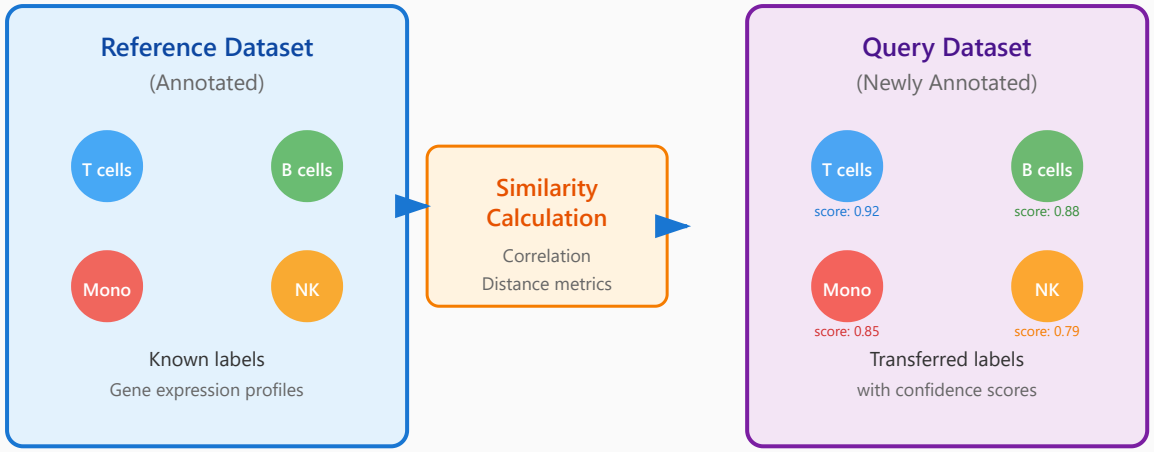
2. Reference-Based Annotation Methods

🔗 Label Transfer from Annotated References

Reference-based methods leverage well-annotated datasets to automatically transfer cell type labels to new query datasets. These approaches compare gene expression profiles between query and reference cells to predict cell identities based on similarity.

1. **Select Reference Dataset**
   Choose a high-quality, well-annotated reference from similar tissue/conditions
2. **Compute Similarity Scores**
   Calculate correlation or distance between query and reference cells
3. **Transfer Labels**
   Assign labels based on nearest neighbors or ensemble voting
4. **Quality Assessment**
   Evaluate confidence scores and validate assignments

Reference-Based Label Transfer Workflow



Popular Tools & Methods

SingleR
Correlation-based method using curated reference datasets

Seurat Anchors
Integration-based label transfer with CCA/RPCA

scmap
Fast nearest neighbor search for cell type projection

CellTypist
Machine learning classifier with pre-trained models

✅ Advantages

- Fast and scalable for large datasets
- Consistent annotations across studies
- Leverages community-curated references
- Provides confidence scores
- Minimal manual intervention required

⚠️ Limitations

- Limited by reference dataset quality
- Cannot identify novel cell types
- Batch effects can reduce accuracy
- May fail for rare cell populations
- Requires appropriate reference selection

💻 Example: SingleR Annotation (R)

```
library(SingleR) library(celldex) # Load reference dataset ref <- HumanPrimaryCellAtlasData() # Run SingleR
annotation predictions <- SingleR(test = query_data, ref = ref, labels = ref$label.main) # Add annotations to
```

```
Seurat object seurat_obj$celltype <- predictions$labels seurat_obj$annotation_score <- predictions$scores
```

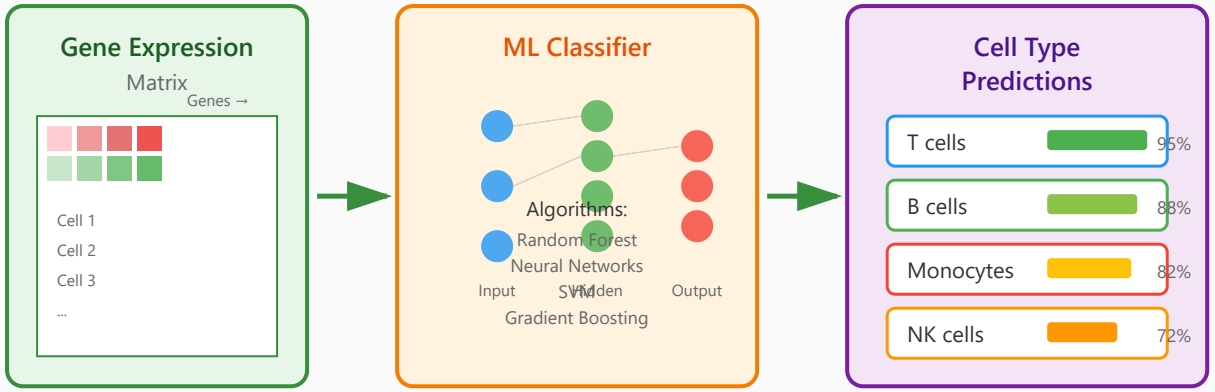
3. Automated Machine Learning Methods

AI-Powered Cell Type Prediction

Automated methods use machine learning algorithms to classify cell types based on gene expression patterns. These tools can be trained on existing data or use pre-defined marker databases to make predictions without requiring manual inspection.

1. **Data Preprocessing**
   Normalize and prepare expression matrix for ML algorithms
2. **Model Training/Selection**
   Train classifier or use pre-trained model on marker databases
3. **Prediction & Scoring**
   Classify cells and generate probability scores for each type
4. **Validation & Refinement**
   Assess predictions quality and refine low-confidence calls

Machine Learning Classification Pipeline



🛠️ Automated Annotation Tools

scCATCH
Evidence-based scoring from marker databases

Garnett
Supervised classifier with marker file specification

CHETAH
Hierarchical classification with confidence scoring

CellAssign
Probabilistic model for marker-based assignment

✅ Advantages

- Highly scalable and reproducible
- Minimal manual curation needed
- Handles complex datasets efficiently
- Can detect subtle expression patterns
- Built-in confidence metrics

⚠️ Limitations

- Black-box nature reduces interpretability
- Requires high-quality training data
- May overfit to training dataset biases
- Limited by predefined marker databases
- Difficult to validate novel predictions

💻 Example: CellTypist Annotation (Python)

```
import celltypist from celltypist import models # Load pre-trained model model =
models.Model.load(model='Immune_All_Low.pkl') # Predict cell types predictions = celltypist.annotate(adata,
```

```
model=model, majority_voting=True) # Add predictions to AnnData adata.obs['predicted_labels'] =
predictions.predicted_labels adata.obs['conf_score'] = predictions.probability
```

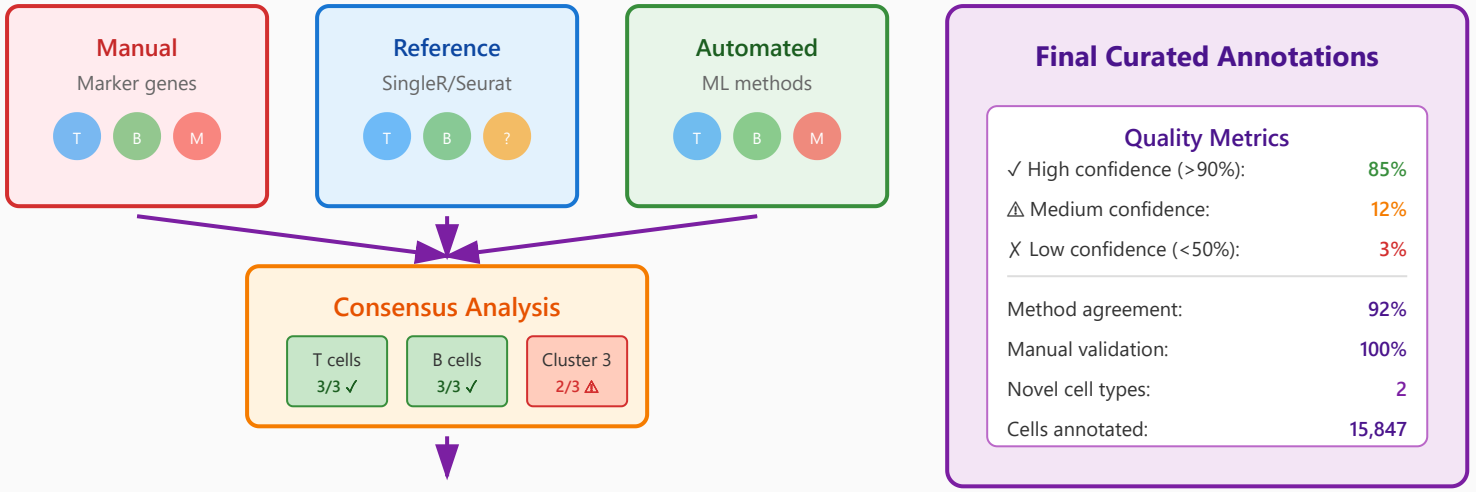
4. Hybrid & Multi-Method Approaches

Combining Multiple Annotation Strategies

The most robust approach combines manual curation, reference-based methods, and automated tools to leverage the strengths of each strategy. This iterative workflow produces high-confidence annotations while maintaining biological interpretability and scalability.

1. **Initial Automated Annotation**
   Apply multiple automated/reference methods for rapid first-pass labeling
2. **Consensus Analysis**
   Compare predictions across methods, identify agreements and conflicts
3. **Manual Validation**
   Expert review of low-confidence or conflicting predictions using markers
4. **Iterative Refinement**
   Update annotations, re-cluster if needed, and validate final assignments

Integrated Multi-Method Annotation Workflow



📋 Best Practice Checklist

- ✓ Run 2-3 different annotation methods
- ✓ Compare results for consistency
- ✓ Validate with canonical markers
- ✓ Review confidence scores
- ✓ Manually inspect low-confidence cells
- ✓ Document annotation decisions
- ✓ Check for doublets/multiplets
- ✓ Investigate novel populations

✅ Advantages

- Maximizes accuracy through consensus
- Balances speed with quality
- Identifies method-specific biases
- Enables discovery of novel cell types
- Provides comprehensive confidence metrics
- Maintains biological interpretability

⚠️ Considerations

- Requires more computational resources
- Longer analysis time investment
- Needs expertise across multiple tools
- Resolving conflicts can be subjective
- More complex workflow management

💻 Example: Multi-Method Consensus Workflow

```
# Step 1: Run multiple methods manual_labels <- ManualAnnotation(seurat_obj, marker_genes) singler_labels <-
SingleR(seurat_obj, ref_data) automated_labels <- CellTypist(seurat_obj, model) # Step 2: Create consensus
matrix consensus <- CompareAnnotations( list(manual = manual_labels, singler = singler_labels, automated =
automated_labels) ) # Step 3: Identify high-confidence consensus final_labels <- consensus %>%
filter(agreement >= 2/3) %>% select(consensus_label, confidence_score) # Step 4: Manual review of conflicts
conflicts <- consensus %>% filter(agreement < 2/3) reviewed_labels <- ManualReview(seurat_obj, conflicts)
```

Key Takeaways

Successful cell type annotation requires a strategic combination of methods. Start with automated tools for efficiency, validate with reference datasets for consistency, and refine with manual curation for accuracy. Always assess confidence scores, validate with known markers, and document your annotation decisions for reproducibility.

Trajectory Analysis

Pseudotime Inference

Order cells along developmental paths

Branching Processes

Identify cell fate decisions

Monocle Algorithm

Reverse graph embedding

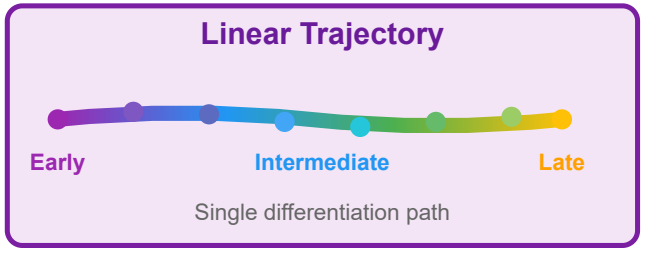
Slingshot Method

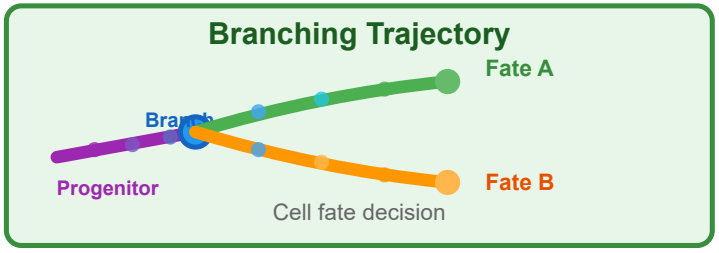
Cluster-based trajectory inference

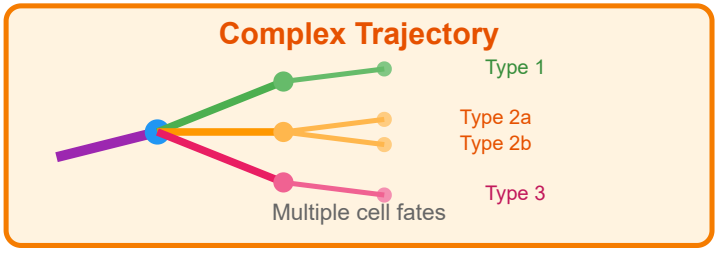
Validation Approaches

Known genes, time-series data

💡 Assumes continuous progression - verify biological relevance







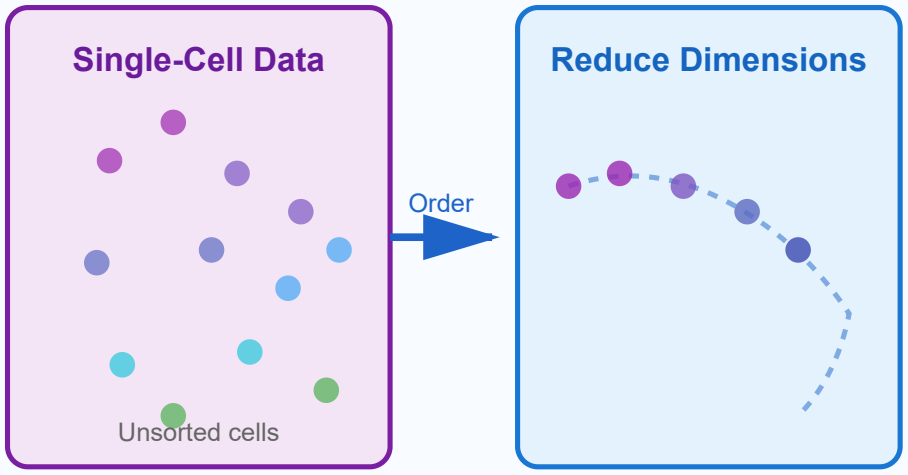
1 Pseudotime Inference

Pseudotime inference is a computational method that orders cells along a continuous trajectory based on their transcriptional similarity, creating a "virtual timeline" of cellular development. Unlike chronological time, pseudotime represents the progression of cells through a biological process such as differentiation, development, or response to stimuli.

Core Principles

- **Dimensionality Reduction:** High-dimensional gene expression data is projected into lower-dimensional space using PCA, t-SNE, or UMAP
- **Trajectory Construction:** Cells are ordered along a path representing progression through a biological process
- **Temporal Ordering:** Each cell receives a pseudotime value indicating its relative position along the trajectory
- **Gene Expression Dynamics:** Identifies genes whose expression changes smoothly along pseudotime

Pseudotime Inference Process



⚠ Key Considerations

- Assumes cells progress along a smooth, continuous trajectory
- Requires sufficient cellular diversity to capture the full process
- May not capture complex, non-linear dynamics or cyclic processes
- Results should be validated with known biological markers

2 Branching Processes

Branching processes identify critical decision points where progenitor cells commit to distinct developmental fates. These bifurcation events represent moments when cellular populations diverge into separate lineages, each characterized by unique gene expression programs.

Key Components

- **Branch Point Detection:** Algorithms identify locations where cell populations split in trajectory space
- **Lineage Assignment:** Cells are assigned to specific branches based on transcriptional profiles
- **Bifurcation Analysis:** Statistical methods determine the significance and timing of branching events
- **Branch-Specific Genes:** Identification of genes that drive or mark distinct cell fates

📊 Real-World Example: T Cell Development

During T cell maturation in the thymus, double-positive (CD4+CD8+) thymocytes reach a critical branch point where they commit to either CD4+ helper T cells or CD8+ cytotoxic T cells. This decision is marked by differential expression of transcription factors like ThPOK (CD4+ fate) and Runx3 (CD8+ fate).

⚠️ Key Considerations

- Branch points must be validated with functional assays or known biology
- Technical noise can create false branching patterns
- Requires adequate cell numbers in each branch for reliable inference
- Temporal resolution affects ability to detect transient branch points

3 Monocle Algorithm

Monocle is a pioneering computational tool that uses reverse graph embedding to reconstruct complex cellular trajectories from single-cell RNA-seq data. Developed by the Trapnell lab, Monocle learns a principal graph structure that captures the underlying developmental or temporal progression of cells.

Algorithm Workflow

- **Feature Selection:** Identifies highly variable genes that capture biological variation
- **Dimensionality Reduction:** Uses reversed graph embedding or UMAP to project cells
- **Principal Graph Learning:** Constructs a tree-like structure representing the trajectory
- **Pseudotime Calculation:** Assigns each cell a position along the learned trajectory
- **Differential Expression:** Identifies genes with significant expression changes along pseudotime

Monocle Versions

- **Monocle 1:** Introduced ICA for dimensionality reduction and minimum spanning tree for trajectory construction
- **Monocle 2:** Implemented reversed graph embedding and DDRTree algorithm for complex branching
- **Monocle 3:** Utilizes UMAP and partition-based graph abstraction for scalability on large datasets

📊 Real-World Example: Myoblast Differentiation

Monocle has been successfully applied to study skeletal muscle development, ordering myoblasts through their differentiation into mature myocytes. The algorithm identified key regulatory genes such as MYOD1, MYOG, and MYH3 that are upregulated during differentiation, while

proliferation markers like MKI67 decrease.

⚠ Key Considerations

- Requires careful parameter tuning for optimal trajectory learning
- Computational complexity increases with dataset size (Monocle 3 addresses this)
- Root cell selection impacts pseudotime ordering—use biological priors when possible
- Works best with datasets containing clear developmental progression

4 Slingshot Method

Slingshot is a cluster-based trajectory inference method that learns smooth, continuous lineage structures from single-cell data. Unlike methods that learn trajectories directly from individual cells, Slingshot first identifies cell clusters and then fits smooth curves through these clusters, making it robust to noise and computationally efficient.

Algorithm Components

- **Cluster Identification:** Uses existing clustering methods like k-means or hierarchical clustering
- **Minimum Spanning Tree:** Constructs a tree connecting cluster centroids to represent global structure
- **Lineage Construction:** Identifies paths from root to terminal clusters as potential lineages

- **Principal Curves:** Fits smooth curves through cells along each lineage for refined representation
- **Cell Weights:** Assigns weights to cells for each lineage based on distance to the curve

Key Advantages

- **Cluster-Based Approach:** Reduces noise by working with cluster summaries
- **Flexible Integration:** Works with any clustering and dimensionality reduction method
- **Multiple Lineages:** Can identify and model multiple diverging trajectories simultaneously
- **Cell Weights:** Provides uncertainty estimates for cell assignment to lineages
- **Computational Efficiency:** Scales well to large datasets

📊 Real-World Example: Pancreatic Development

Slingshot has been applied to study pancreatic endocrine cell differentiation, where pancreatic progenitors diverge into multiple hormone-producing cell types (alpha, beta, delta, and PP cells). The method successfully identified the branching structure and revealed transcriptional programs driving each cell fate decision, including key roles of NKX6.1, ARX, and PAX4.

⚠️ Key Considerations

- Trajectory quality depends heavily on the quality of initial clustering
- Requires specification or inference of root cluster (starting point)
- May oversimplify trajectories if clusters don't capture biological transitions well

- Best suited for datasets with clear cluster structure along developmental paths

5 Validation Approaches

Validating trajectory inference results is critical to ensure that computational predictions reflect true biological processes rather than technical artifacts. Since pseudotime is inferred rather than directly measured, multiple complementary validation strategies should be employed.

Validation Strategies

- **Known Marker Genes:** Check if genes with established temporal expression patterns match pseudotime predictions
- **Time-Series Data:** Compare pseudotime ordering with actual collection time points when available
- **Functional Validation:** Perform perturbation experiments on key predicted regulators
- **Cross-Method Comparison:** Run multiple trajectory inference algorithms and compare results
- **RNA Velocity:** Use splicing information to validate trajectory directionality
- **Lineage Tracing:** Compare with experimental lineage tracking data when available

Marker Gene Validation

- **Early Markers:** Should have high expression in early pseudotime (e.g., stem cell markers like OCT4, NANOG)

- **Late Markers:** Should have high expression in late pseudotime (e.g., differentiation markers)
- **Stage-Specific Markers:** Should show expected temporal patterns at specific developmental stages
- **Correlation Analysis:** Calculate correlation between known marker expression and pseudotime

Time-Series Validation

- **Collection Time Comparison:** Verify that pseudotime ordering correlates with actual sample collection time
- **Temporal Coherence:** Cells collected at similar times should have similar pseudotime values
- **Directionality Check:** Ensure trajectory direction matches biological progression
- **Kendall's Tau:** Statistical measure of concordance between pseudotime and real time

📊 Real-World Example: Neural Differentiation Validation

In a study of neural differentiation from embryonic stem cells, researchers validated pseudotime trajectories by confirming that: (1) pluripotency markers (OCT4, SOX2) decreased along pseudotime, (2) neural progenitor markers (PAX6, SOX1) peaked at intermediate pseudotime, and (3) mature neuronal markers (TUBB3, MAP2) increased in late pseudotime. This pattern matched both time-series experiments and known biology.

Advanced Validation Methods

- **RNA Velocity:** Uses spliced vs unspliced mRNA ratios to predict future cell states and validate trajectory direction
- **CRISPR Screens:** Perturbation of predicted key regulators should alter trajectory structure
- **Genetic Lineage Tracing:** Direct comparison with barcode-based lineage tracking experiments
- **Multi-Omics Integration:** Validate with protein expression, chromatin accessibility, or metabolomics data

⚠ Key Considerations

- Use multiple independent validation approaches for robust conclusions
- Consider biological context—not all predicted patterns may be functionally relevant
- Validation with time-series data is gold standard but not always available
- Negative results (mismatches) can reveal technical issues or unexpected biology
- Document validation methods thoroughly for reproducibility

Best Practice: Always combine multiple validation approaches and document assumptions about biological processes when interpreting trajectory analysis results.

Part 3/3:

Advanced Methods

- Spatial context
- Multi-modal data
- Velocity analysis
- Communication inference

Spatial Transcriptomics Technologies

A Comprehensive Guide to Methods, Applications, and Trade-offs

🔬 Visium Technology

10X Genomics spatial transcriptomics platform with 55µm spots providing whole transcriptome profiling

🎯 MERFISH Principles

Multiplexed error-robust FISH achieving subcellular resolution with combinatorial barcoding

🧬 seqFISH Evolution

Sequential fluorescence in situ hybridization profiling 10,000+ genes at single-molecule resolution

⚡ Slide-seq Methods

Bead-based spatial barcoding achieving 10µm near-cellular resolution with whole transcriptome

⚖️ Resolution Trade-offs

Understanding the balance between gene coverage, spatial resolution, and experimental throughput

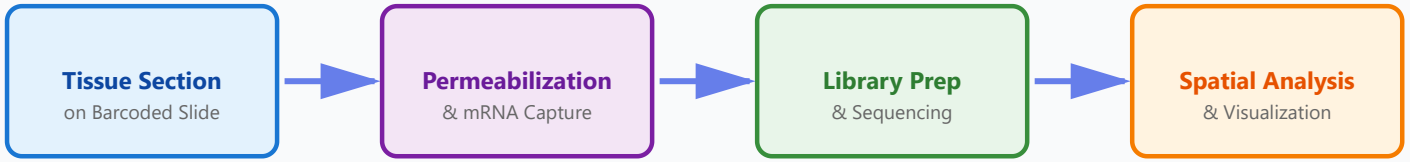
💡 Spatial context reveals tissue architecture, cell-cell interactions, and microenvironmental influences on gene expression

1 Visium Spatial Gene Expression

Array-Based Whole Transcriptome Spatial Profiling

Visium, developed by 10x Genomics, represents the most widely adopted spatial transcriptomics platform for unbiased, whole-transcriptome analysis. The technology employs a glass slide containing approximately 5,000 barcoded spots arranged in a hexagonal array, with each spot measuring 55 micrometers in diameter. Fresh-frozen or FFPE tissue sections are placed onto these slides, where tissue permeabilization releases mRNA that is captured by spatially-barcoded oligonucleotides. Each spot contains millions of capture probes with unique spatial barcodes, enabling downstream sequencing to map gene expression back to specific locations within the tissue architecture. This approach bridges the gap between traditional histology and single-cell genomics, providing spatial context to transcriptomic data while maintaining compatibility with standard next-generation sequencing workflows.

Visium Workflow Overview



Parameter	Specification
Spot Size	55 µm diameter (captures 1-10 cells per spot)
Spot Spacing	100 µm center-to-center distance
Total Spots	~5,000 spots per slide (6.5mm × 6.5mm capture area)
Gene Detection	Whole transcriptome (18,000+ genes typically detected)
Sensitivity	~5,000-10,000 UMIs (Unique Molecular Identifiers) per spot
Tissue Compatibility	Fresh-frozen (FF) and formalin-fixed paraffin-embedded (FFPE)

Processing Time	2-3 days (library preparation + sequencing)

🔍 Unbiased Discovery

Captures the entire transcriptome without prior gene selection, enabling discovery of unexpected expression patterns and novel tissue organization principles.

📊 Standardized Platform

Commercial platform with optimized reagents and automated analysis pipelines (Space Ranger), ensuring reproducibility across laboratories.

🧬 Integration Ready

Seamlessly integrates with single-cell RNA-seq datasets for cell type deconvolution using computational methods.

💻 Rich Ecosystem

Extensive analysis tools including Seurat, Scanpy, Giotto, and Squidpy for spatial domain identification and cell-cell communication inference.

✓ Advantages

- ✓ Comprehensive genome-wide coverage without gene panel limitations
- ✓ Well-established commercial platform with strong technical support

✗ Limitations

- ✗ Lower spatial resolution (55µm spots) captures multiple cells, losing single-cell information
- ✗ Requires computational deconvolution to infer cell type composition

- ✓ Compatible with both fresh-frozen and archived FFPE tissues
- ✓ Large and active research community sharing protocols
- ✓ Cost-effective for whole-transcriptome discovery experiments
- ✓ Straightforward integration with existing scRNA-seq workflows

- ✗ Fixed array geometry may not align optimally with tissue structure
- ✗ Lower sensitivity compared to targeted approaches (typical 20-30% gene detection)
- ✗ Requires relatively large tissue sections (minimum 6.5mm × 6.5mm)
- ✗ Difficult to resolve fine cellular boundaries and subcellular localization

🎯 Key Applications

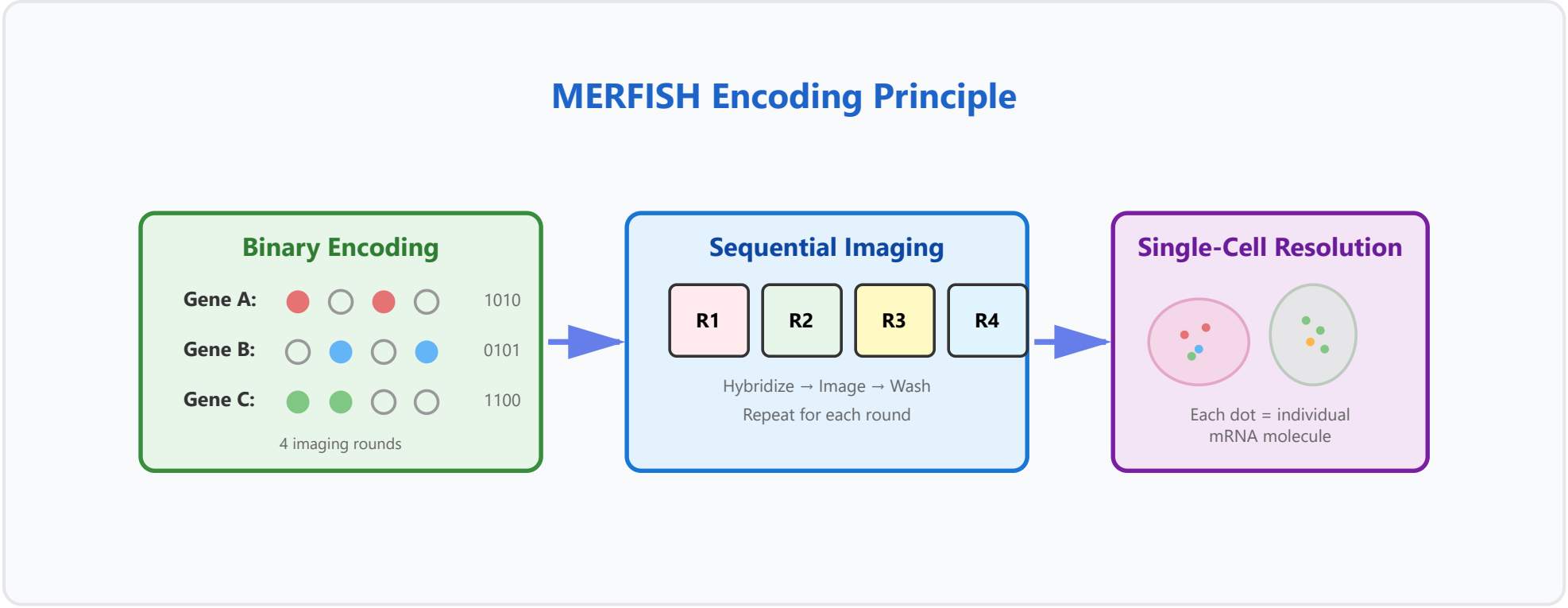
- → **Cancer Research:** Tumor microenvironment mapping, identification of spatial immune infiltration patterns, and characterization of tumor-stroma interactions
- → **Neuroscience:** Brain region parcellation, mapping of cortical layers, and spatial organization of neural cell types across brain structures
- → **Developmental Biology:** Tissue morphogenesis tracking, spatial gene expression dynamics during organogenesis, and embryonic patterning
- → **Immunology:** Tertiary lymphoid structure identification, immune cell spatial organization in lymphoid organs and inflamed tissues

→ **Disease Pathology:** Spatial characterization of fibrotic tissues, inflammatory responses, and tissue architecture disruption in disease states

2 MERFISH Technology

Multiplexed Error-Robust Fluorescence In Situ Hybridization

MERFISH revolutionizes spatial transcriptomics through an ingenious application of error-correcting codes borrowed from information theory. Developed by Xiaowei Zhuang's laboratory, MERFISH enables the simultaneous imaging of hundreds to thousands of RNA species at subcellular resolution. The technology employs combinatorial labeling where each target RNA is encoded by a specific binary barcode (e.g., 16-bit encoding scheme). Through sequential rounds of imaging with different fluorescent probe combinations, each RNA molecule's identity is determined by its unique fluorescence pattern across multiple imaging rounds. The incorporation of Hamming distance-based error correction allows robust RNA identification even in the presence of technical noise or imperfect hybridization, achieving >95% accuracy in gene assignment.



Parameter	Specification
Spatial Resolution	~100-200 nm (subcellular, near-optical diffraction limit)
Gene Panel Size	100-10,000 genes (typically 300-500 for standard experiments)
Detection Efficiency	10-20% of cellular mRNA molecules per gene
Encoding Scheme	Modified Hamming Distance-4 (MHD4) code with 16-bit barcodes
Imaging Rounds	Typically 10-20 rounds for 300-1000 genes

Throughput	Hundreds to thousands of cells per experiment
Imaging Time	2-5 days for complete workflow

🎯 Subcellular Precision

Resolves individual mRNA molecules within single cells, enabling analysis of RNA localization patterns, nuclear vs cytoplasmic distribution.

🛡️ Error Correction

Built-in error correction based on Hamming distance allows robust gene identification even with technical imperfections, achieving >95% accuracy.

📈 Scalable Multiplexing

Binary combinatorial encoding enables exponential scaling: N imaging rounds can theoretically detect 2^N different RNA species.

🔬 3D Capability

Can be extended to three-dimensional tissue volumes through z-stack imaging, revealing spatial organization in intact tissue.

✓ Advantages

- ✓ True single-cell and subcellular resolution for precise cellular analysis

✗ Limitations

- ✗ Requires pre-selection of target gene panel (not unbiased)

- ✓ High multiplexing capacity (hundreds to thousands of genes)
- ✓ Robust error correction ensures reliable gene identification
- ✓ Absolute quantification of mRNA copy numbers per cell
- ✓ Compatible with immunofluorescence for protein colocalization
- ✓ Can image thick tissue sections and 3D samples
- ✓ Reveals subcellular RNA localization and compartmentalization

- ✗ Complex experimental workflow with multiple hybridization cycles
- ✗ Requires specialized microscopy equipment
- ✗ Labor-intensive protocol with long acquisition times (days per sample)
- ✗ Computationally demanding for image processing and spot decoding
- ✗ Higher cost per gene compared to sequencing-based methods
- ✗ Limited field of view in single imaging sessions
- ✗ Tissue autofluorescence can interfere with detection

🎯 Key Applications

→ **Neuroscience:** Mapping cell type organization in brain regions, neuronal connectivity inference, molecular characterization of neuronal subtypes

→ **Cancer Biology:** Tumor heterogeneity analysis at single-cell level, characterization of rare cancer cell subpopulations, spatial analysis of drug resistance markers

→ **Cell Biology:** Subcellular RNA localization studies, analysis of RNA trafficking and compartmentalization, investigation of local translation sites

→ **Developmental Biology:** High-resolution fate mapping during embryogenesis, molecular characterization of tissue boundaries, analysis of morphogen gradients

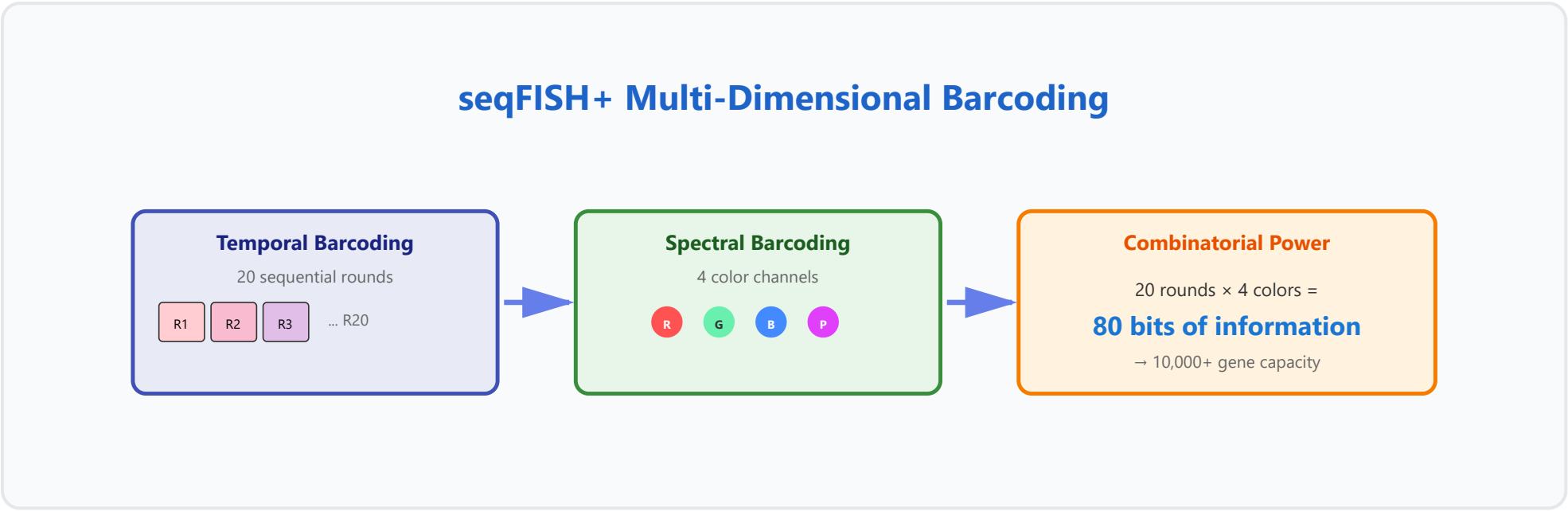
→ **Immunology:** Spatial organization of immune cells, characterization of cell interactions at contacts, immune checkpoint expression patterns

3 seqFISH+ Technology

Sequential Fluorescence In Situ Hybridization with Ultra-High Multiplexing

seqFISH+ represents an evolution of sequential FISH technologies that achieves unprecedented multiplexing capacity through super-resolution imaging and combinatorial barcoding strategies. Originally developed by Long Cai's laboratory, seqFISH+ can profile 10,000+ genes with subcellular resolution. The method employs pseudocoloring strategies where multiple rounds of hybridization with different colored probes create unique combinatorial codes for each target RNA. By combining temporal barcoding (sequential rounds), spectral barcoding (different fluorophores), and spatial barcoding (subcellular localization), seqFISH+ achieves massive multiplexing while maintaining single-molecule sensitivity. The latest iterations incorporate super-

resolution microscopy techniques to push resolution beyond the optical diffraction limit, achieving near-whole-transcriptome coverage at nanoscale precision.



Parameter	Specification
Spatial Resolution	~100 nm with super-resolution (diffraction-limited: ~200-300 nm)
Gene Coverage	10,000+ genes; scalable to whole transcriptome theoretically
Detection Sensitivity	~10-15% of cellular mRNA copies detected per target
Imaging Rounds	80 hybridization cycles for 10,000 gene panel

Fluorescent Channels	3-4 color channels per round (pseudo-coloring strategy)
Field of View	100s-1000s of cells per imaging field
Total Experiment Time	3-7 days for complete workflow with large gene panels

🎨 Pseudo-Coloring

Uses sequential rounds with limited color channels to create high-dimensional barcodes, achieving massive multiplexing without many simultaneous fluorophores.

🔬 Super-Resolution

Can be combined with STORM, PAINT, or other super-resolution techniques to achieve resolution beyond the diffraction limit (~20-50 nm).

📊 Complete Coverage

Achieves near-whole-transcriptome coverage (10,000+ genes), approaching comprehensive profiling of scRNA-seq with spatial information preserved.

🧊 3D Tissue Mapping

Optimized for volumetric imaging of intact tissues, enabling reconstruction of 3D cell organization and tissue architecture.

✓ Advantages

✗ Limitations

- ✓ Unprecedented multiplexing capacity (10,000+ genes)
- ✓ Maintains subcellular resolution with single-molecule sensitivity
- ✓ Compatible with super-resolution microscopy for nanoscale imaging
- ✓ Efficient probe design allows broad gene coverage
- ✓ 3D imaging capability for volumetric tissue analysis
- ✓ Can be combined with protein imaging (immunofluorescence)
- ✓ Provides absolute mRNA quantification per cell

- ✗ Extremely long experimental time (multiple days for large panels)
- ✗ Very labor-intensive with 80+ sequential hybridization rounds
- ✗ Requires highly stable imaging system (minimal stage drift)
- ✗ Computationally intensive image registration and spot decoding
- ✗ High cost due to extensive probe sets and imaging time
- ✗ Risk of sample degradation over multiple rounds
- ✗ Limited throughput (typically single fields per experiment)
- ✗ Requires significant expertise in microscopy and image analysis

🎯 Key Applications

→ **Systems Biology:** Comprehensive mapping of tissue-level gene expression programs, revealing coordinated spatial patterns across thousands of genes

→ **Developmental Biology:** High-resolution spatiotemporal mapping during development, tracking cell fate decisions with

near-complete transcriptome coverage

→ **Neuroscience:** Brain cell type classification and spatial organization, mapping transcriptional gradients across brain regions with cellular resolution

→ **Single-Cell Spatial Analysis:** Detailed characterization of cellular heterogeneity within tissues, revealing rare cell states with spatial context

→ **3D Tissue Architecture:** Reconstruction of tissue organization in three dimensions, understanding how gene expression relates to physical tissue structure

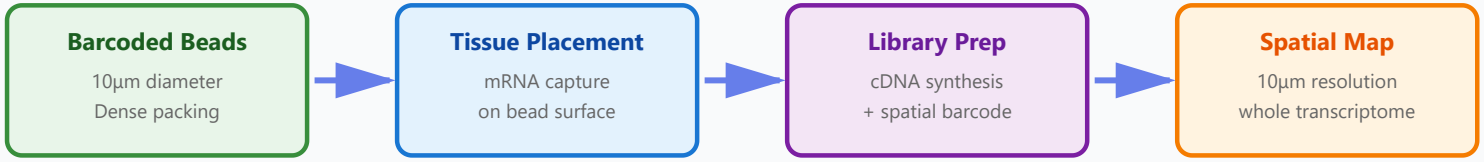
4 Slide-seq and Slide-seqV2

High-Resolution Bead-Based Spatial Barcoding

Slide-seq, developed by Evan Macosko and Fei Chen, introduces a bead-based approach to spatial transcriptomics that achieves near-cellular resolution (10 μm) with unbiased whole-transcriptome profiling. The technology employs DNA-barcoded beads (10 μm diameter) that are randomly packed onto a surface at high density, creating a two-dimensional array where each bead's position is recorded and associated with a unique spatial barcode. Tissue sections are placed onto this bead array, and mRNA is

captured directly on the beads through poly(dT) oligonucleotides. Unlike array-based methods with pre-defined spot positions, Slide-seq's random bead packing enables flexible spatial resolution limited only by bead size. Slide-seqV2 improved upon the original with enhanced bead chemistry, achieving 10-fold higher sensitivity while maintaining the same spatial resolution, making it a powerful tool for high-resolution spatial transcriptomics.

Slide-seq Technology Workflow



Parameter	Slide-seq	Slide-seqV2
Spatial Resolution	10 μm (near-cellular resolution)	
Bead Size	10 μm diameter	
Bead Density	~100 million beads/cm^2 (densely packed)	
UMIs per Bead	~150 UMIs	**~1,500 UMIs (10-fold ↑)**

Genes Detected	~600 genes/bead	**~3,000 genes/bead (5-fold ↑)**
Gene Coverage	Whole transcriptome (unbiased)	
Tissue Compatibility	Fresh-frozen tissues primarily	

Random Bead Packing

Unlike fixed arrays, random bead distribution enables flexible spatial sampling that can adapt to any tissue geometry without pre-defined positions.

Near-Cellular Resolution

10µm bead size approaches single-cell dimensions for many cell types, enabling more precise spatial mapping than larger spot-based methods.

Whole Transcriptome

Unbiased capture of all mRNA species without pre-selection, enabling discovery-driven research and comprehensive gene expression profiling.

Scalable Platform

Bead-based approach is highly scalable and can be adapted to large tissue sections or multiple samples in parallel experiments.

✓ Advantages

✗ Limitations

- ✓ Higher spatial resolution (10μm) than Visium, approaching cellular scale
- ✓ Unbiased whole-transcriptome profiling without gene panel selection
- ✓ Flexible spatial sampling through random bead distribution
- ✓ Slide-seqV2 achieves 10-fold sensitivity improvement over original
- ✓ Compatible with standard sequencing workflows and analysis pipelines
- ✓ Can profile irregular tissue shapes and edges effectively
- ✓ Lower cost per data point compared to imaging-based methods
- ✓ Suitable for large tissue sections and whole-organ mapping

- ✗ Still captures multiple cells per bead in dense tissues (not true single-cell)
- ✗ Requires computational deconvolution for cell type assignment
- ✗ Random bead placement creates irregular spatial sampling
- ✗ Lower gene detection efficiency compared to optimized scRNA-seq
- ✗ Primarily optimized for fresh-frozen tissues, FFPE compatibility limited
- ✗ Complex bead preparation and quality control requirements
- ✗ Spatial barcode assignment requires computational image analysis
- ✗ Variable bead packing density can create sampling artifacts

🎯 Key Applications

→ **Brain Mapping:** High-resolution spatial transcriptomics of brain regions, cell type mapping across cortical layers, analysis of neuroanatomical structures

→ **Developmental Biology:** Spatiotemporal profiling during organogenesis with near-cellular resolution, tracking cell fate transitions and tissue boundaries

→ **Tumor Microenvironment:** Characterization of cancer-immune interfaces, spatial heterogeneity analysis, mapping of invasive margins with improved resolution

→ **Comparative Studies:** Cross-tissue and cross-species spatial transcriptome comparisons, evolutionary analysis of tissue organization

→ **Organ-Scale Mapping:** Whole-organ spatial profiling to understand tissue-level organization, regional specialization, organ-wide gene expression gradients

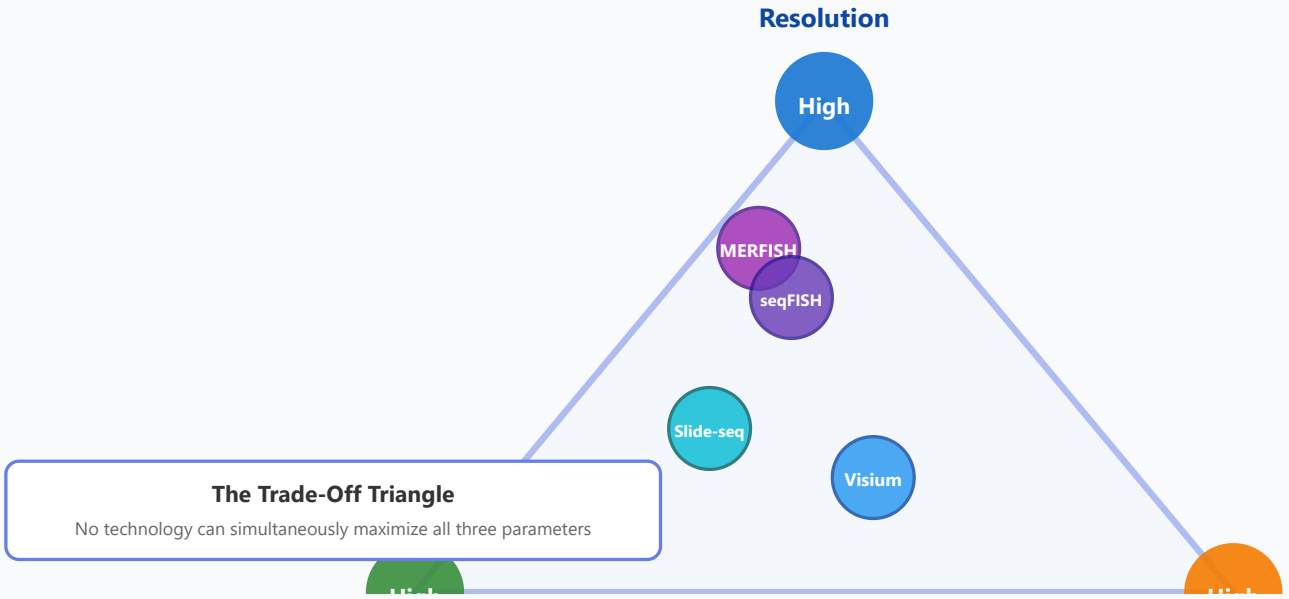
5 Resolution Trade-offs & Technology Selection

Balancing Spatial Resolution, Gene Coverage, and Throughput

The field of spatial transcriptomics presents researchers with a fundamental three-way trade-off between **spatial resolution**, **gene coverage**, and **experimental throughput**. This triangle of constraints forces strategic decisions based on specific biological questions. High-resolution imaging methods like MERFISH and seqFISH provide subcellular precision but require targeted gene

panels and extended acquisition times. Sequencing-based approaches like Visium and Slide-seq offer unbiased whole-transcriptome profiling with faster turnaround but at lower spatial resolution. Understanding these trade-offs is essential for selecting the optimal technology for each research question, tissue type, and experimental scale. No single technology excels in all three dimensions—the key is matching the method to your scientific priorities.

Technology Comparison Matrix



Technology	Resolution	Gene Count	Throughput	Best Use Case

Visium	55 µm (multi-cell)	Whole transcriptome (18,000+ genes)	★★★★★ Very High	Discovery studies, large tissue surveys, unbiased profiling
Slide-seqV2	10 µm (near-cellular)	Whole transcriptome (3,000+ genes)	★★★★ High	Higher resolution discovery, brain mapping, organ-scale studies
MERFISH	~100-200 nm (subcellular)	Targeted panel (100-1,000 genes)	★★ Medium	Hypothesis-driven studies, cell type ID, subcellular RNA
seqFISH+	~100 nm (subcellular)	Ultra-high mux (10,000+ genes)	★ Low	Comprehensive high-res mapping, 3D architecture, systems biology

🎯 Resolution vs Coverage

Inverse relationship: Higher spatial resolution typically requires targeted approaches, while whole-transcriptome methods sacrifice resolution for comprehensive coverage.

⚡ Throughput Constraints

Imaging-based methods require extensive acquisition time (days), limiting throughput. Sequencing-based approaches enable parallel processing of many samples.

💰 Cost Considerations

Cost per gene varies dramatically: sequencing-based methods are cost-effective for many genes, while imaging methods excel for specific targeted genes.

🔬 Technical Expertise

Commercial platforms (Visium) offer standardization and ease of use. Advanced imaging methods require specialized microscopy expertise.

✓ Strategic Advantages of Each Approach

- ✓ **Visium:** Best for exploratory studies, hypothesis generation, when comprehensive gene coverage is critical
- ✓ **Slide-seq:** Optimal when you need better resolution than Visium but still want whole-transcriptome profiling
- ✓ **MERFISH:** Perfect for hypothesis testing with known gene panels, validating scRNA-seq findings spatially
- ✓ **seqFISH:** Ultimate choice when both high gene coverage AND subcellular resolution are required

✗ Common Pitfalls to Avoid

- ✗ Using imaging methods when genes of interest are unknown - wastes time and resources
- ✗ Choosing Visium for rare cell type analysis when cells are smaller than 55µm spots
- ✗ Attempting seqFISH for time-sensitive projects - requires weeks of optimization
- ✗ Ignoring tissue type compatibility (FFPE vs fresh-frozen requirements)
- ✗ Underestimating computational requirements for high-resolution imaging data
- ✗ Not considering integration with existing scRNA-seq data early in planning

🎯 Integrated Workflow Strategies

→ **Discovery → Validation Workflow:** Start with Visium for unbiased discovery, identify key genes and regions, then validate with MERFISH at subcellular resolution

→ **scRNA-seq Integration:** Generate single-cell reference atlas first, then use Visium or Slide-seq for spatial mapping with cell type deconvolution algorithms

→ **Multi-Technology Approach:** Combine complementary methods on serial sections - Visium for broad coverage, MERFISH for specific regions, immunofluorescence for proteins

→ **Iterative Refinement:** Use lower-resolution methods to identify spatial domains, then apply higher-resolution techniques to boundary regions and interfaces

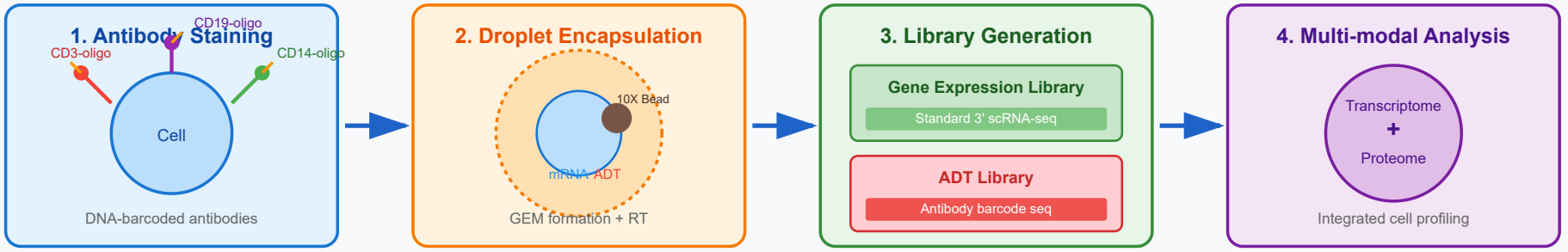
→ **3D Reconstruction:** Stack seqFISH or MERFISH z-sections for true 3D tissue maps, or use Slide-seq on serial sections for organ-scale 3D atlases

🔬 **Key Takeaway:** There is no single "best" spatial transcriptomics technology. The optimal choice depends on your specific biological question, required resolution, gene coverage needs, sample type, budget, and timeline. Understanding the fundamental trade-offs enables strategic technology selection for maximum scientific impact.

CITE-seq

Cellular Indexing of Transcriptomes and Epitopes by Sequencing

CITE-seq Workflow Overview



Key Benefits of CITE-seq

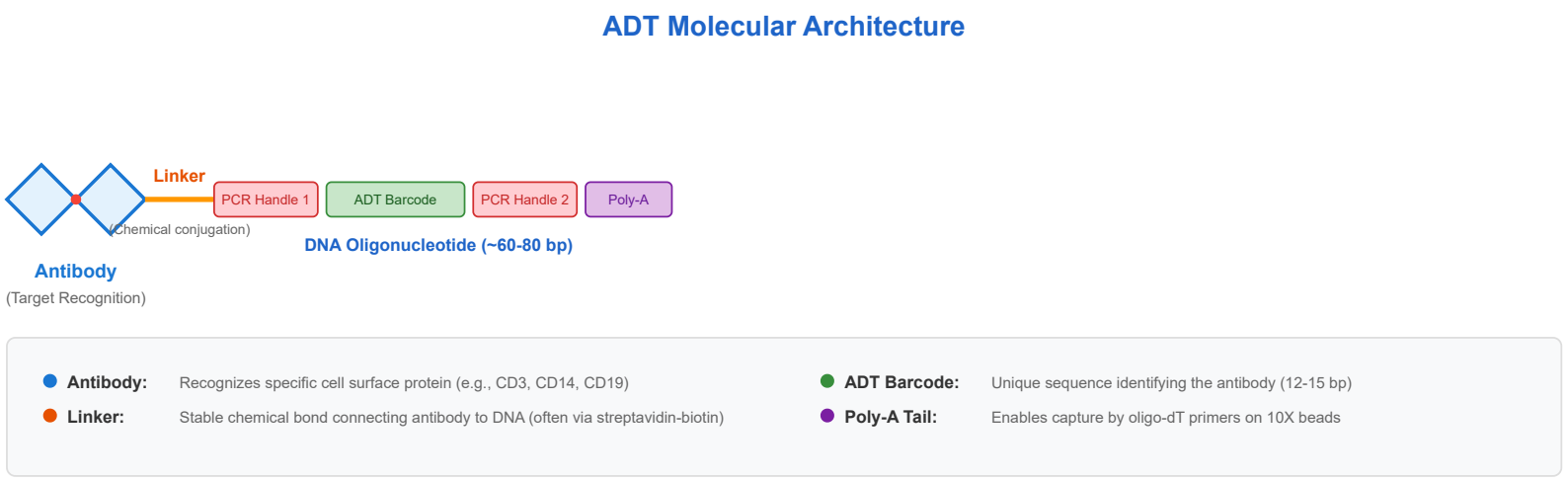
1. **Enhanced Cell Type Resolution**
   Protein markers provide clearer cell identity
2. **Validation of Gene Expression**
   Direct protein measurement validates RNA data
3. **Post-translational Information**
   Captures protein modifications not visible in RNA
4. **Low-abundance Transcripts**
   Proteins detect markers missed by RNA-seq
5. **Platform Compatibility**
   Works with standard 10X Genomics workflow
6. **Same Cell Barcode**
   RNA and protein data share cell identity

> 💡 CITE-seq bridges the gap between transcriptomics and proteomics at single-cell resolution

1. Antibody-Derived Tags (ADT) Technology

CITE-seq's core innovation lies in the use of oligonucleotide-conjugated antibodies, known as Antibody-Derived Tags (ADTs). These specialized reagents enable simultaneous measurement of surface protein expression alongside gene expression in individual cells.

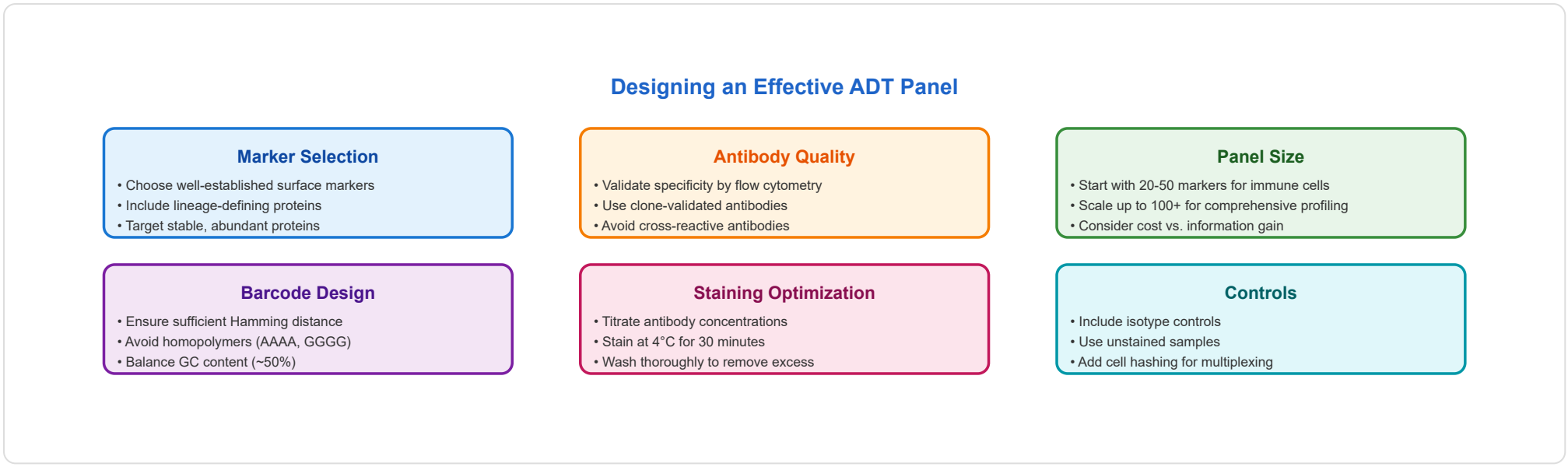
ADT Structure and Design



How ADTs Work

ADTs function by combining the specificity of antibody-antigen recognition with the quantitative power of DNA sequencing. When cells are stained with a cocktail of ADT-conjugated antibodies, each antibody binds to its cognate surface protein. During droplet encapsulation and reverse transcription, the poly-A tail of the ADT oligonucleotide is captured by the same oligo-dT primers that capture mRNA, ensuring that protein and RNA measurements share the same cell barcode.

ADT Panel Design Considerations



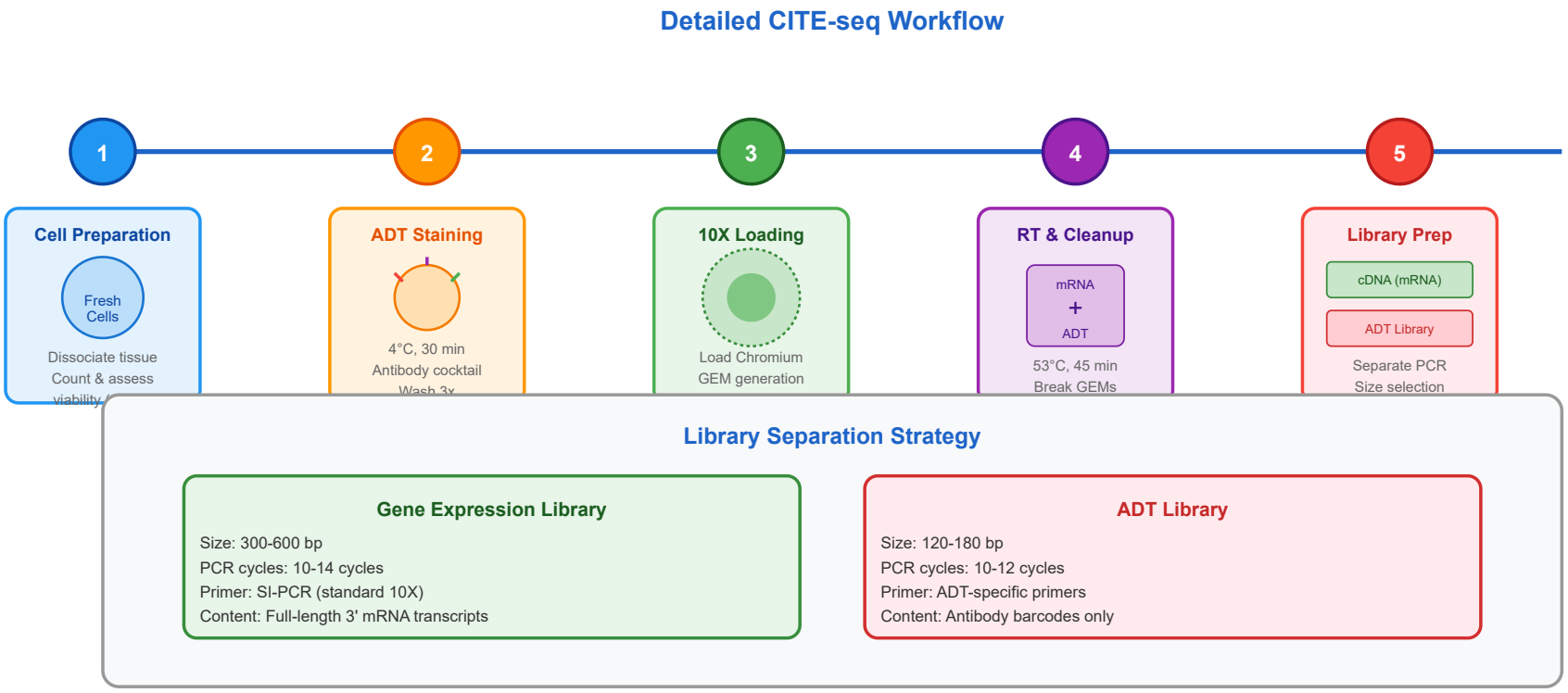
Key Advantages of ADT Technology

- **Direct Protein Measurement:** Captures actual protein expression, not just mRNA abundance
- **Higher Sensitivity:** Antibodies can detect low-abundance proteins that may not be captured in RNA data
- **Same Cell Resolution:** Both RNA and protein data are linked by the same cell barcode
- **Reduced Dropout:** Protein measurements are less affected by technical dropout compared to sparse scRNA-seq data
- **Functional States:** Captures activation markers and post-translational modifications

2. Technical Workflow and Implementation

CITE-seq integrates seamlessly with standard droplet-based single-cell RNA-seq platforms, particularly the 10X Genomics Chromium system. The workflow requires minimal modifications to existing protocols while adding a powerful protein measurement dimension.

Step-by-Step Protocol

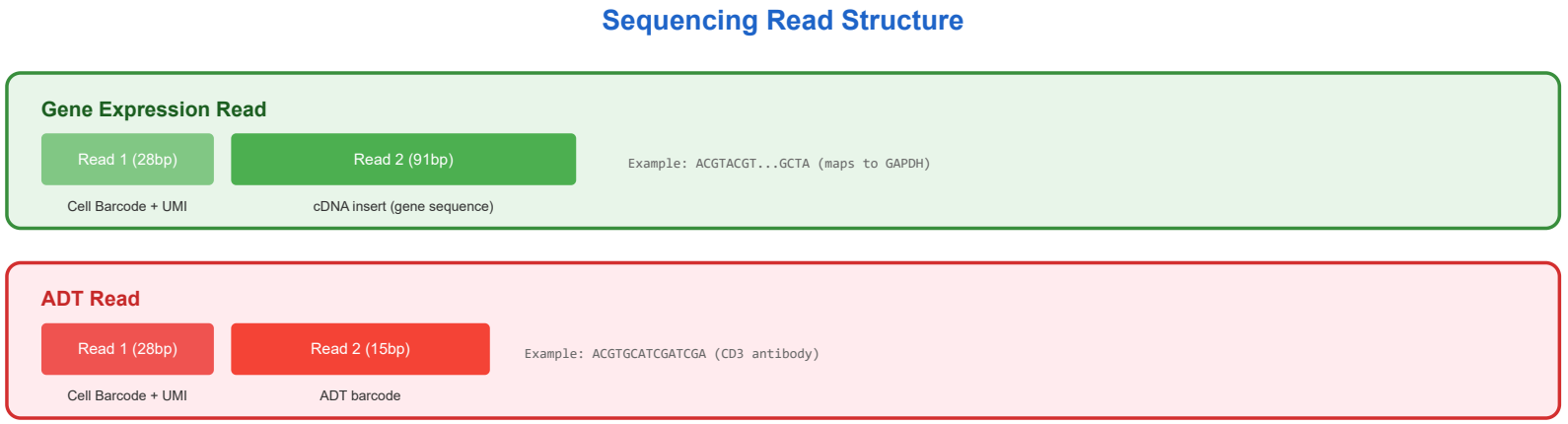


Critical Technical Considerations

- **ADT Library Optimization:** The ADT library is much smaller and simpler than the cDNA library, requiring careful titration to avoid over-amplification. Typically, ADT libraries are sequenced with 5-10% of total sequencing reads.

- **Size Selection:** Proper size selection is crucial to separate ADT libraries (120-180 bp) from cDNA libraries (300-600 bp). SPRI bead cleanup at 0.6x ratio effectively separates these populations.
- **Washing Stringency:** Thorough washing after antibody staining is essential to remove unbound ADTs, which can create high background signal. Three washes with PBS + 1% BSA are recommended.
- **Cell Concentration:** Maintain cell concentration at recommended levels (700-1,200 cells/µL) to ensure proper encapsulation efficiency and minimize doublets.

Sequencing and Data Output



Sequencing Depth Recommendations

Gene Expression Library

Recommended depth: 20,000-50,000 reads per cell

ADT Library

Recommended depth: 1,000-5,000 reads per cell

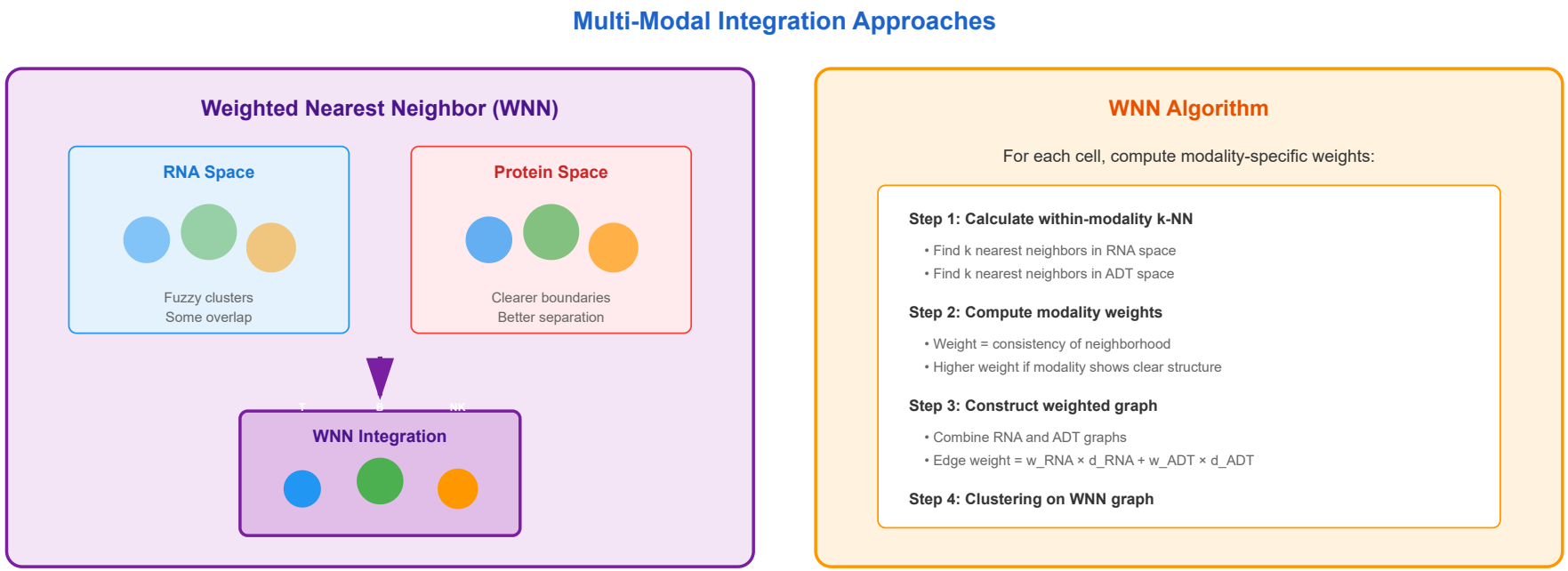
Purpose: Capture transcriptome diversity
Typical output: 2,000-5,000 genes per cell

Purpose: Quantify protein markers
Typical output: 20-200 proteins per cell

3. Applications and Multi-Modal Data Integration

CITE-seq enables sophisticated multi-modal analysis by integrating transcriptomic and proteomic data from the same single cells. This integration reveals biological insights that would be impossible to obtain from either data type alone.

Data Integration Methods



Key Applications

CITE-seq Applications Across Biology

1 Immune Cell Profiling

- Resolve fine T cell subsets (CD4+, CD8+, Tregs, exhausted T cells)
- Characterize activation states (CD25, CD69, HLA-DR)
- Identify rare populations (double-negative T cells, γδ T cells)
- Study immune responses in vaccination, infection, and cancer

Example: COVID-19 immune profiling, CAR-T cell monitoring

2 Tumor Microenvironment

- Map tumor-infiltrating lymphocytes (TILs)
- Identify immunosuppressive cells (MDSCs, Tregs, TAMs)
- Measure checkpoint molecules (PD-1, PD-L1, CTLA-4, TIM-3)
- Correlate immune infiltration with tumor genetics

Example: Predicting immunotherapy response, resistance mechanisms

3 Developmental Biology

- Track cell lineage markers during differentiation
- Validate developmental trajectories with surface markers
- Study stem cell heterogeneity (HSCs, MSCs, iPSCs)
- Measure differentiation stage markers (CD34, CD38, CD117)

Example: Hematopoiesis mapping, organoid development

4 Autoimmune & Inflammatory Disease

- Profile immune dysregulation in disease vs. healthy tissue
- Identify pathogenic cell states
- Track inflammatory markers (TNF-α, IL-6, IFN-γ receptors)
- Monitor treatment response over time

Example: Rheumatoid arthritis, inflammatory bowel disease, lupus

5 Neuroscience

- Classify neuronal subtypes using surface receptors
- Study neurotransmitter receptor expression
- Profile microglia activation states
- Investigate neuroinflammation in disease

Example: Alzheimer's disease, neurodegeneration, brain development

6 Sample Multiplexing (Cell Hashing)

- Pool multiple samples in one 10X run
- Use hashtag oligos (HTOs) for sample identification
- Reduce batch effects and costs
- Detect and remove doublets computationally

Example: Patient cohorts, treatment time courses, genetic perturbations

Advantages Over RNA-seq Alone

- **Resolving Ambiguous Clusters:** Protein markers can distinguish cell types that appear similar at the transcriptome level. For example, CD4+ and CD8+ T cells may cluster together based on RNA but are cleanly separated by CD4/CD8 protein expression.

- **Validation of Annotations:** Surface protein measurements provide independent validation of RNA-based cell type assignments, increasing confidence in biological interpretations.
- **Functional Markers:** Many activation markers (PD-1, Ki-67, phospho-proteins) are post-translationally regulated and not reliably detected at the RNA level.
- **Low-Dropout Measurements:** Protein counts are less sparse than RNA counts, providing more robust quantification of key markers that may be subject to dropout in scRNA-seq.

Computational Tools for CITE-seq Analysis

- **Seurat (R):** Integrated workflow with WNN analysis, multi-modal clustering, and visualization. Most widely used for CITE-seq data.
- **MUON (Python):** Multi-modal omics analysis framework with support for CITE-seq integration.
- **CiteFuse (R):** Specialized package for CITE-seq analysis including doublet detection and data normalization.
- **totalVI (Python/scvi-tools):** Probabilistic model for joint analysis of RNA and protein data with batch correction.
- **Azimuth:** Reference-based cell type annotation using CITE-seq reference atlases.

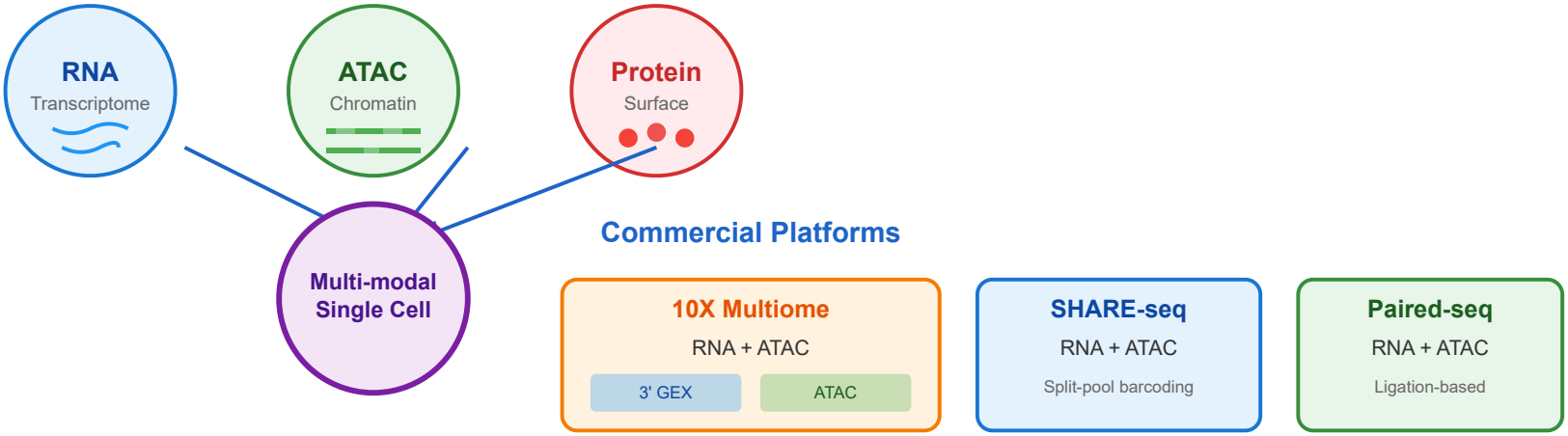
Future Directions and Extensions

CITE-seq technology continues to evolve with new applications and extensions including REAP-seq (RNA expression and protein sequencing), ASAP-seq (ATAC-seq with protein), TEA-seq (Transcriptome, Epitope, and Accessibility sequencing), and DOGMA-seq (DNA, Oligo, Gene expression, Methylation, and Accessibility sequencing). These multi-omic approaches promise even deeper insights into cellular identity and function by layering additional modalities onto single-cell measurements.

🎯 CITE-seq represents a paradigm shift in single-cell biology by enabling simultaneous measurement of transcriptome and proteome, providing unprecedented resolution for cell type identification, functional state characterization, and disease profiling.

Multimodal Omics

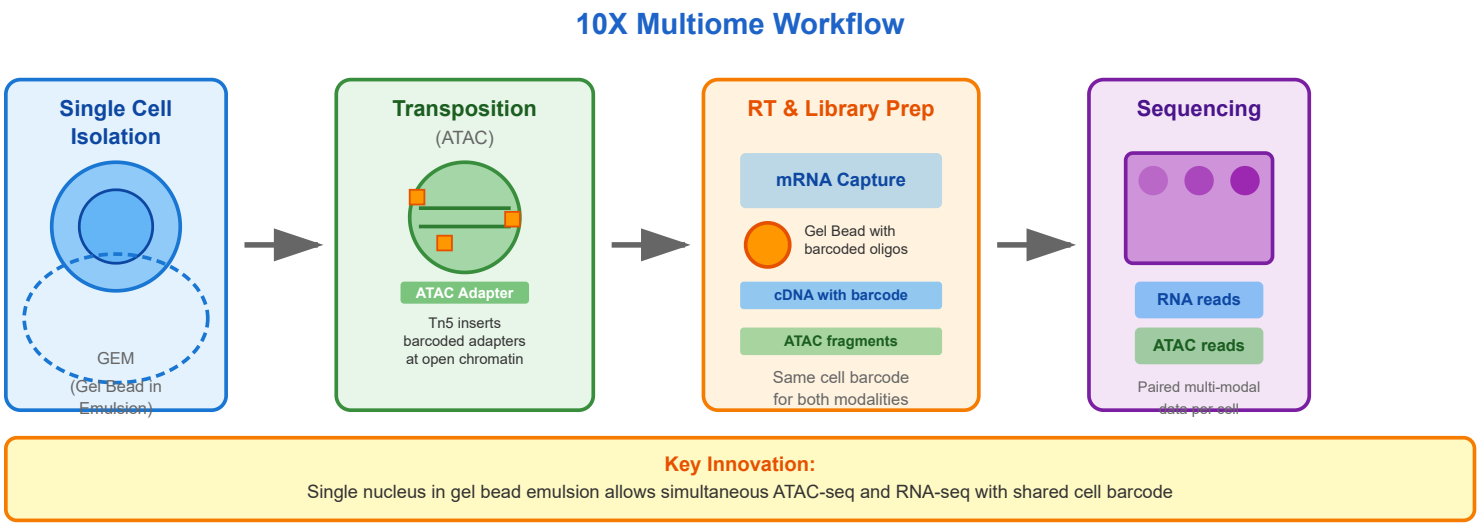
Single-Cell Multimodal Technologies



💡 Multi-omics reveals regulatory mechanisms

Methodological Principles & Examples

1. 10X Genomics Multiome ATAC + Gene Expression



The 10X Genomics Multiome platform enables simultaneous profiling of gene expression and chromatin accessibility from the same single cell. The workflow begins with single-cell or single-nucleus isolation into gel bead-in-emulsion (GEM) droplets containing barcoded oligonucleotides. Within each droplet, Tn5 transposase performs ATAC-seq by inserting sequencing adapters at accessible chromatin regions, while poly(A)-tailed mRNAs are captured by barcoded oligonucleotides on the gel bead. Both modalities share the same cellular barcode, enabling direct linkage of chromatin accessibility and gene expression profiles.

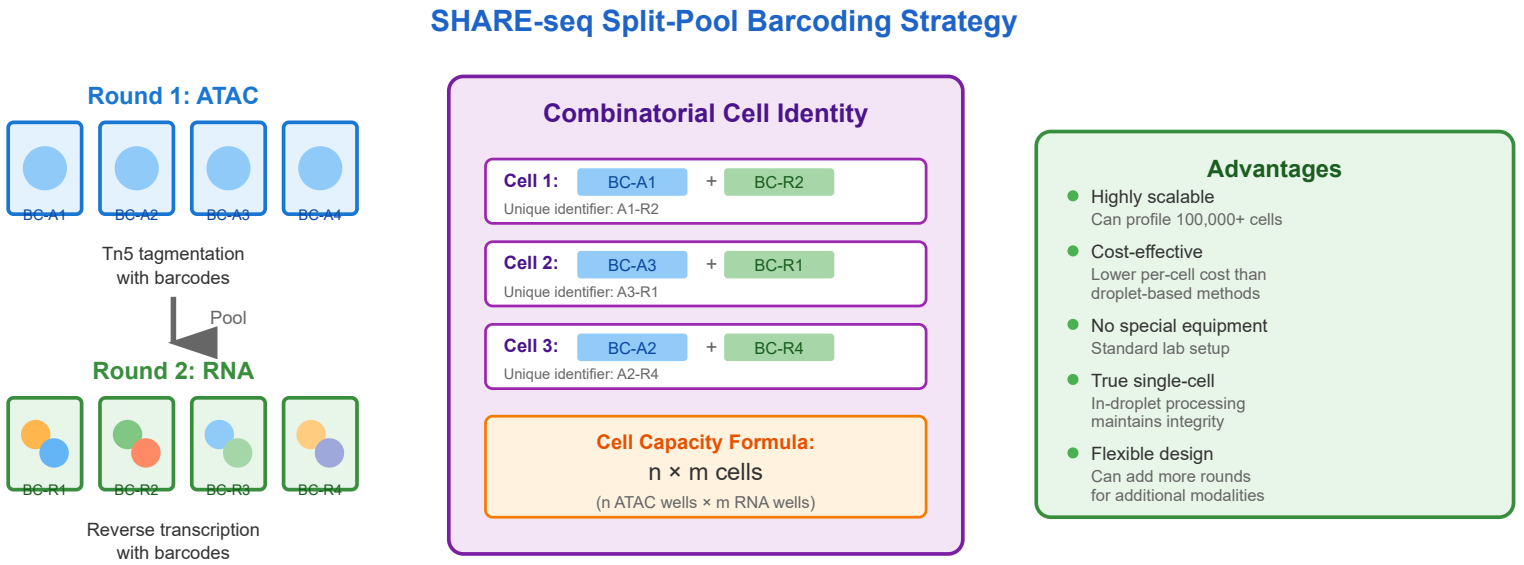
Key Features:

- **High throughput:** Process thousands of cells in parallel using microfluidic technology
- **Linked modalities:** Same cell barcode for RNA and ATAC data ensures accurate pairing
- **Single-nucleus compatible:** Works with frozen samples and difficult-to-dissociate tissues
- **Automated workflow:** Standardized protocol with commercial reagents reduces technical variability
- **Data integration:** Built-in bioinformatics tools for joint analysis of chromatin and transcription

🎯 **Applications:**

Ideal for studying gene regulatory mechanisms, cell differentiation trajectories, and linking genetic variants to gene expression through integrated chromatin accessibility and transcriptome analysis.

2. SHARE-seq (Shared Chromatin and RNA seq)



SHARE-seq employs a split-pool combinatorial indexing strategy to jointly profile chromatin accessibility and gene expression. Cells are first distributed across wells for ATAC-seq with well-specific barcodes, then pooled and redistributed for RNA-seq with a second round of barcoding. Each cell receives a unique combination of ATAC and RNA barcodes, enabling bimodal data pairing without physical cell isolation. This approach dramatically increases throughput while reducing per-cell costs.

Technical Details:

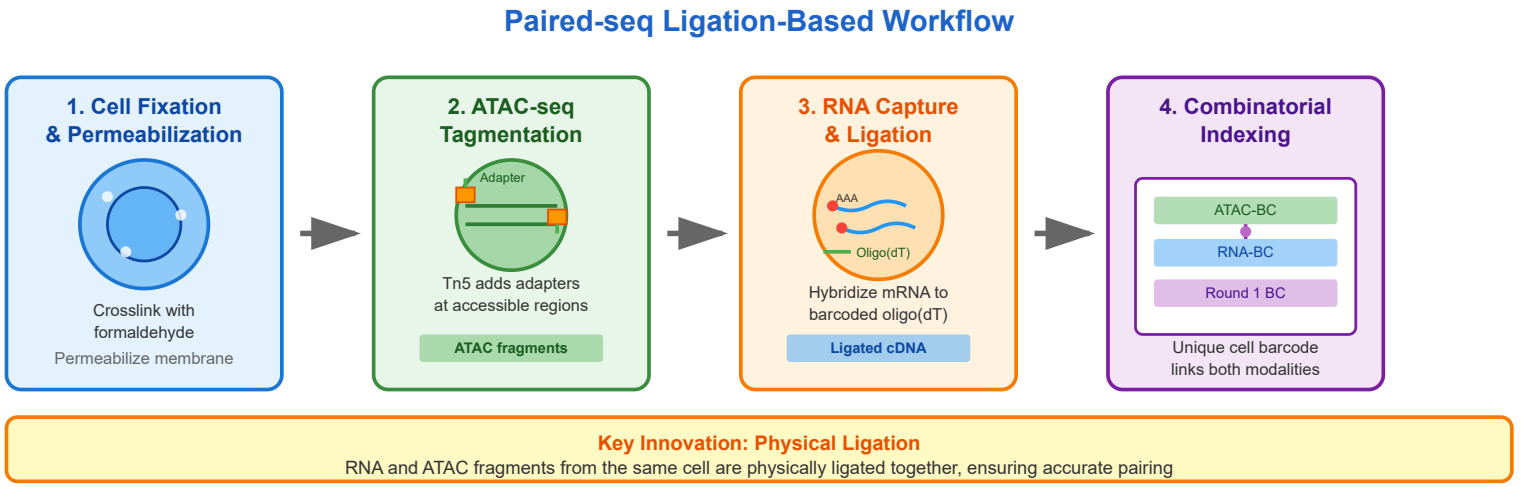
- **Split-pool strategy:** Multiple rounds of barcoding create unique cell identifiers through combinatorial indexing

- **Ultra-high throughput:** Can process >100,000 cells in a single experiment
- **In-droplet processing:** Maintains single-cell resolution without microfluidic devices
- **Scalable barcoding:** n wells in round 1 × m wells in round 2 = n×m unique cell barcodes
- **Adaptable protocol:** Can be extended to include additional modalities with extra barcoding rounds

🎯 **Best Use Cases:**

Large-scale population studies, atlas generation, rare cell type discovery, and experiments requiring cost-effective profiling of tens of thousands of cells.

3. Paired-seq (Paired-end sequencing)



Paired-seq uses a ligation-based approach to physically link RNA and ATAC-seq libraries from the same cell. After fixation and permeabilization, Tn5 transposase tags accessible chromatin regions. Subsequently, poly(A) mRNAs are captured by barcoded

oligo(dT) primers. A key innovation is the physical ligation of ATAC and RNA fragments, followed by combinatorial indexing rounds. This physical linkage ensures robust pairing of both modalities from individual cells.

Distinguishing Features:

- **Ligation-based pairing:** Physical linkage of ATAC and RNA fragments from the same cell ensures high confidence in data pairing
- **Fixed cells:** Formaldehyde crosslinking preserves cellular structure during processing
- **Nucleus-friendly:** Works well with both single cells and isolated nuclei
- **Sequential processing:** ATAC and RNA steps performed on the same cell population
- **High data quality:** Ligation step reduces barcode collisions and improves data reliability

🎯 Optimal Applications:

Studies requiring high-confidence multimodal pairing, archived tissue samples (compatible with fixed samples), and experiments where data quality is prioritized over throughput.

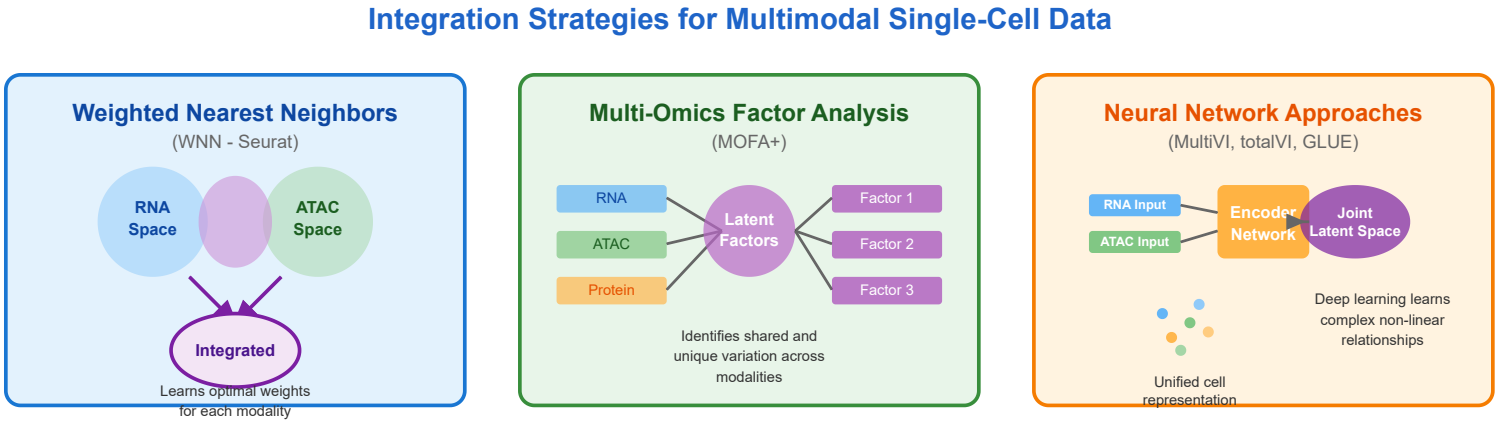
Platform Comparison

Feature	10X Multiome	SHARE-seq	Paired-seq
Barcoding Strategy	Droplet-based (GEM)	Split-pool combinatorial	Ligation + combinatorial
Throughput	1,000-10,000 cells	100,000+ cells	10,000-50,000 cells
Equipment Required	Chromium Controller	Standard lab equipment	Standard lab equipment
Cost per Cell	$$$ (Higher)	$ (Lower)	$$ (Medium)
Sample Type	Fresh/frozen nuclei	Fresh cells/nuclei	Fixed or fresh cells

Feature	10X Multiome	SHARE-seq	Paired-seq
Protocol Complexity	Low (automated)	Medium (manual steps)	High (multi-step)
Data Quality	High consistency	Good with higher noise	High with physical linkage
Cell Barcode Collisions	Very low	Low (combinatorial)	Very low (ligation)
Scalability	Limited by device	Highly scalable	Moderately scalable
Best Application	Routine multimodal analysis	Large-scale atlas projects	High-confidence pairing

Computational Data Integration

Multimodal Analysis Approaches



Integration of multimodal single-cell data requires sophisticated computational approaches that can handle the distinct characteristics of each data type. Modern methods range from weighted neighbor graphs to deep learning architectures, each with strengths for different biological questions.

Common Integration Challenges:

- **Scale differences:** RNA and ATAC data have vastly different dynamic ranges and sparsity levels
- **Feature spaces:** Different modalities measure fundamentally different molecular features
- **Missing data:** Dropout events and technical noise vary across modalities
- **Batch effects:** Technical variation can confound biological signals
- **Interpretation:** Balancing complexity with biological interpretability

Research Applications & Biological Insights

Real-World Applications

Development & Differentiation

Track chromatin remodeling during cell fate decisions. Multimodal data reveals how enhancer accessibility precedes gene activation during embryonic development and organogenesis.

Disease Mechanisms

Identify dysregulated gene regulatory networks in cancer, autoimmune diseases, and neurological disorders by linking genetic variants to altered chromatin states and gene expression.

Drug Response

Understand heterogeneous drug responses by profiling epigenetic and transcriptional changes across cell populations, identifying resistance mechanisms and therapeutic targets.

Cell Atlas Projects

Generate comprehensive reference maps of tissues and organs with integrated chromatin and transcriptome profiles, enabling discovery of rare cell types and transitional states.

🌟 Future Directions:

Emerging technologies aim to add more modalities (methylation, protein, spatial information) to create truly comprehensive single-cell profiles. Integration with spatial transcriptomics and live-cell imaging will provide temporal and spatial context to regulatory mechanisms.

RNA Velocity

Inferring Cell State Dynamics from Single-Cell RNA-seq Data

🧬 Spliced/Unspliced Ratio

Infer transcriptional dynamics from steady-state snapshots

📊 Velocity Estimation

Predict future cell states and trajectories

⚙️ Dynamic Models

Account for transcription, splicing, and degradation

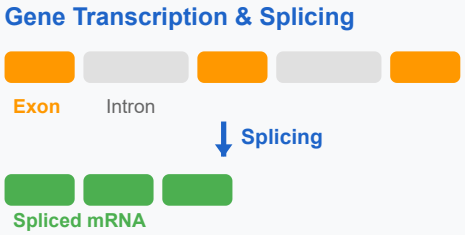
🚀 scVelo Improvements

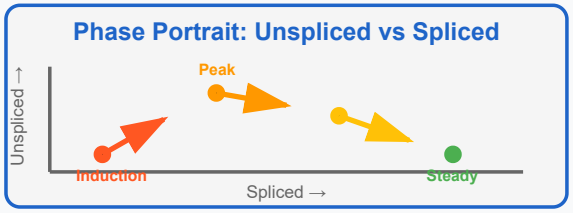
Dynamical model with latent time inference

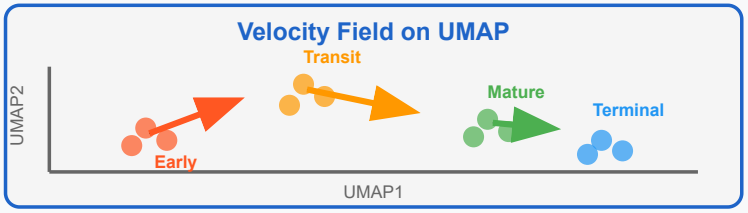
🎯 Interpretation

Direction and magnitude of cell state changes

💡 RNA velocity adds a temporal dimension to static single-cell snapshots





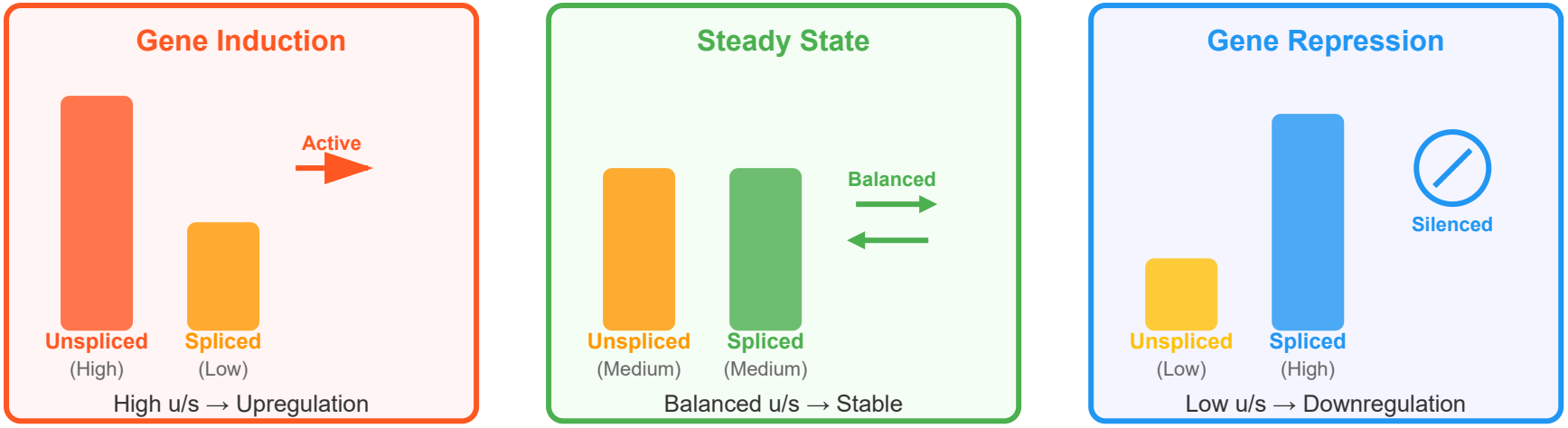


1 Spliced/Unspliced Ratio Analysis

RNA velocity leverages the natural biology of gene expression to infer cellular dynamics. During transcription, genes produce pre-mRNA containing both exons (coding sequences) and introns (non-coding sequences). Through splicing, introns are removed to create mature mRNA. By measuring both unspliced (pre-mRNA) and spliced (mature mRNA) counts in single cells, we can infer whether a gene is being actively induced, repressed, or in steady state.

Key Concepts

- → **Unspliced mRNA:** Newly transcribed pre-mRNA containing introns, indicates recent transcriptional activity
- → **Spliced mRNA:** Mature mRNA after intron removal, represents the functional transcript pool
- → **Ratio dynamics:** High unspliced/spliced ratio suggests gene induction; low ratio suggests repression
- → **Temporal inference:** The relationship between unspliced and spliced abundance reveals transcriptional state
- → **Steady-state assumption:** Most scRNA-seq captures cells at equilibrium, but the ratio reveals direction of change



Velocity Inference from Ratios

```
If u/s >> expected: positive velocity (gene turning ON)
    If u/s ≈ expected: zero velocity (steady state)
If u/s << expected: negative velocity (gene turning OFF)
```

2 Velocity Estimation Methods

RNA velocity estimation transforms the observed unspliced and spliced counts into predicted future states. The core idea is to fit a model that captures the relationship between unspliced and spliced abundances, then use deviations from steady-state to predict directional changes in gene expression. This velocity vector for each gene can then be projected into low-dimensional space (like UMAP or t-SNE) to visualize cell trajectories.

Estimation Workflow

→ **Phase portrait construction:** Plot unspliced vs spliced counts for each gene across all cells

→ **Steady-state line fitting:** Identify the expected equilibrium relationship (typically linear)

→ **Deviation measurement:** Calculate how far each cell deviates from steady state

→ **Velocity vector computation:** Convert deviations into directional change predictions

→ **Dimensionality reduction projection:** Project gene-level velocities into embedding space

→ **Velocity field smoothing:** Average velocities across similar cells for robustness

Mathematical Foundation

Steady State Model: At equilibrium, transcription rate α balances degradation: $\alpha = \beta s + \gamma u$

Velocity Equation: $v = du/dt - \gamma u$, where du/dt represents change in unspliced abundance

Linear Regression: Fit $u = \gamma s$ to identify steady-state relationship from data

Residual-based Velocity: $v \propto u - \gamma s$
- positive = above line → gene increasing
- negative = below line → gene decreasing

Projection: Gene velocities projected to low-dimensional space: $v_{emb} = \Sigma\ (\partial x/\partial g) \times v_g$

3 Dynamic Models of RNA Kinetics

Dynamic models explicitly capture the kinetics of transcription, splicing, and degradation. Rather than assuming steady state, these models account for how unspliced and spliced abundances change over time through differential equations. This allows for more accurate velocity estimation, especially in systems with rapid transcriptional changes or where genes haven't reached equilibrium.

Model Components

→ **Transcription rate (α):** Rate at which new unspliced mRNA is produced from the gene

→ **Splicing rate (β):** Rate at which unspliced mRNA is converted to spliced mRNA

→ **Degradation rates (γ_u, γ_s):** Rates at which unspliced and spliced mRNA are degraded

→ **Gene states:** Genes can be in induction, repression, or steady-state phases

→ **Time-dependent dynamics:** Models capture temporal evolution of mRNA populations

→ **Parameter estimation:** Infer kinetic parameters from observed u and s distributions

Differential Equations for RNA Dynamics

Unspliced mRNA:

$$du/dt = \alpha(t) - \beta u - \gamma_u \cdot u$$

Spliced mRNA:

$$ds/dt = \beta u - \gamma_s \cdot s$$

Gene States:

- Induction: α switches from $0 \rightarrow \alpha_{max}$
- Steady: α constant, $du/dt \approx 0$, $ds/dt \approx 0$
- Repression: α switches from $\alpha_{max} \rightarrow 0$

Steady-State Model (velocyto)

Assumption: Cells are at equilibrium
Method: Linear regression on u vs s
Pros: Simple, fast computation
Cons: Less accurate for transient states

Dynamical Model (scVelo)

Assumption: Cells in different phases
Method: Fit full ODE system
Pros: Captures transitions accurately
Cons: More complex, computationally intensive

4 scVelo: Advanced Dynamic Modeling

scVelo represents a major advancement in RNA velocity analysis by implementing a fully dynamic model that accounts for gene-specific kinetics across all cells. Unlike the original velocyto approach which assumes steady state, scVelo models the complete transcription-splicing-degradation dynamics and infers a "latent time" variable for each cell, representing its progression through biological processes. This enables more accurate velocity estimates and better resolution of complex differentiation trajectories.

Key Innovations

→ **Dynamical mode:** Jointly models induction, steady-state, and repression phases for each gene

→ **Latent time inference:** Estimates internal biological time for each cell along trajectories

→ **Gene-specific kinetics:** Learns individual α, β, γ parameters for each gene

→ **Likelihood-based fitting:** Uses EM algorithm to optimize parameters and latent time simultaneously

→ **Velocity confidence:** Provides uncertainty estimates for velocity predictions

→ **Improved projection:** Better velocity field smoothing and embedding projection methods

scVelo Dynamical Model

Likelihood-based Optimization:
Maximize P(u, s | α, β, γ, t) across all cells

For each gene:
• Estimate switching time points (when α changes)
• Fit rate parameters (α, β, γ) that best explain observed u/s distributions
• Infer latent time t for each cell

Velocity from Model:
v = du/dt = α(t) - (β + γ_u)·u
Computed from fitted parameters and cell's position in latent time

5 Interpretation and Applications

Understanding RNA velocity results requires careful interpretation of both the direction and magnitude of velocity vectors in the context of biological processes. Velocity arrows indicate the predicted short-term future state of cells, enabling inference of differentiation trajectories, cell fate decisions, and regulatory dynamics. However, velocities should be validated against known biology and complemented with other analyses for robust conclusions.

Interpretation Guidelines

→ **Arrow direction:** Points toward predicted future cell state along differentiation or cell cycle

→ **Arrow length:** Indicates magnitude of change; longer arrows suggest faster transitions

→ **Velocity consistency:** Coherent flow patterns indicate robust trajectories

→ **Terminal states:** Cells with short or inward-pointing arrows may represent endpoints

→ **Branching points:** Diverging velocities reveal cell fate bifurcations

→ **Confidence assessment:** Check velocity confidence scores and gene-level contributions

→ **Biological validation:** Verify predictions with known markers and experimental perturbations

🧬 Developmental Trajectories

Map cell fate decisions during embryogenesis and organogenesis. Velocity analysis reveals the directionality of differentiation pathways and identifies progenitor populations.

🔄 Cell Cycle Analysis

Distinguish cycling from non-cycling cells and determine cell cycle phase progression. Velocity vectors show circular patterns for proliferating cells.

🦠 Disease Progression

Study cancer evolution, immune response dynamics, and disease state transitions. Identify aberrant differentiation patterns in pathological conditions.

🧪 Drug Response

Assess how perturbations alter cell trajectories. Compare velocity fields before and after treatment to identify mechanistic effects.

🧠 Neuronal Differentiation

Trace neural stem cell differentiation into diverse neuronal subtypes. Resolve complex branching patterns in neurogenesis.

☆ Best Practices

✓ **Quality control:** Filter low-quality cells and genes before velocity analysis

✓ **Multiple approaches:** Compare steady-state and dynamical models

✓ **Confidence metrics:** Examine velocity confidence and gene contributions

✓ **Biological context:** Interpret results in light of known markers and biology

✓ **Validation:** Cross-validate with trajectory inference methods (Monocle, Slingshot)

✓ **Visualization:** Use multiple embeddings (UMAP, PCA, force-directed) to assess robustness

⚠ Common Pitfalls to Avoid

✗ **Over-interpretation:** RNA velocity predicts SHORT-TERM futures, not long-term fates

✗ **Batch effects:** Technical variation can create spurious velocity patterns

✗ **Low coverage:** Sparse data reduces velocity reliability, especially for rare cell types

✗ **Assuming causality:** Velocity shows correlation with trajectories, not causal mechanisms

✗ **Ignoring biology:** Computational predictions must align with experimental evidence

✗ **Single-gene focus:** Analyze velocity across many genes, not individual genes in isolation

Summary: RNA velocity is a powerful computational method that adds temporal resolution to single-cell RNA-seq data. By leveraging the natural dynamics of RNA splicing, it enables prediction of cell state transitions, identification of differentiation

trajectories, and discovery of regulatory relationships. When combined with careful quality control, appropriate modeling choices (steady-state vs. dynamical), and biological validation, RNA velocity provides unique insights into cellular dynamics that are invisible in static gene expression measurements.

Key References

- **velocyto:** La Manno et al. (2018). "RNA velocity of single cells." Nature.
- **scVelo:** Bergen et al. (2020). "Generalizing RNA velocity to transient cell states through dynamical modeling." Nature Biotechnology.
- **RNA velocity analysis:** Lange et al. (2022). "CellRank for directed single-cell fate mapping." Nature Methods.
- **Velocity interpretation:** Gorin et al. (2022). "RNA velocity unraveled." PLoS Computational Biology.

Cell-Cell Communication

Inferring Cell-Cell Interactions from scRNA-seq

Ligand-Receptor Interaction

TNF→TNFR | IL6→IL6R | VEGFA→FLT1

Communication Inference Tools

CellPhoneDB
- Statistical framework
- Permutation test
- Multi-subunit
- Spatial optional

NicheNet
- Gene regulatory
- Prior knowledge
- Target prediction
- Prioritization

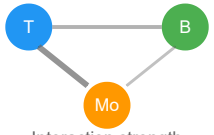
CellChat
- Network analysis
- Pattern recognition
- Visualization
- Comparison

Typical Analysis Outputs

Interaction Heatmap

Cell type pairs

Network Graph

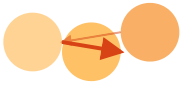


Interaction strength

Dot Plot

Size: p-value
Color: mean expr

Spatial Context

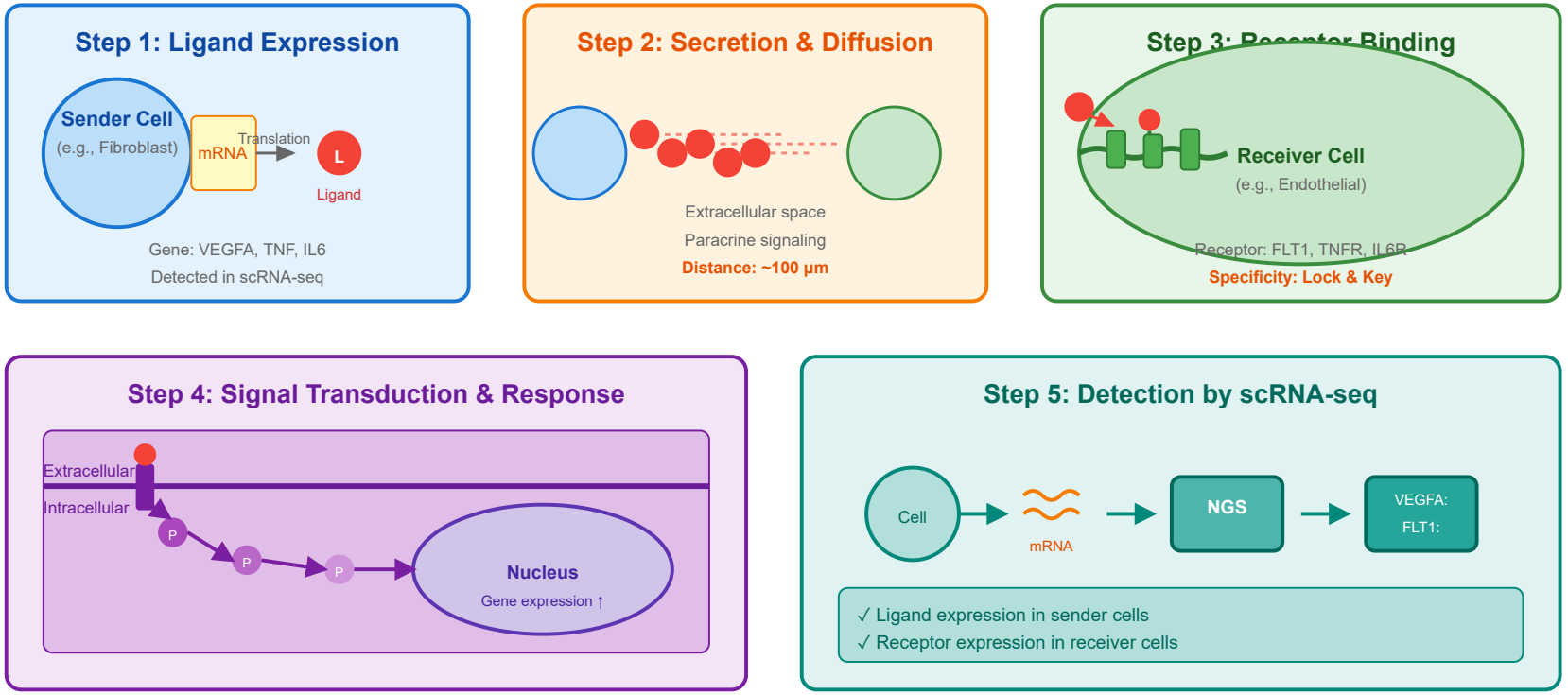


Physical proximity
→ Higher confidence

💡 Infer cellular communication from expression patterns

Detailed Mechanism of Ligand-Receptor Interaction

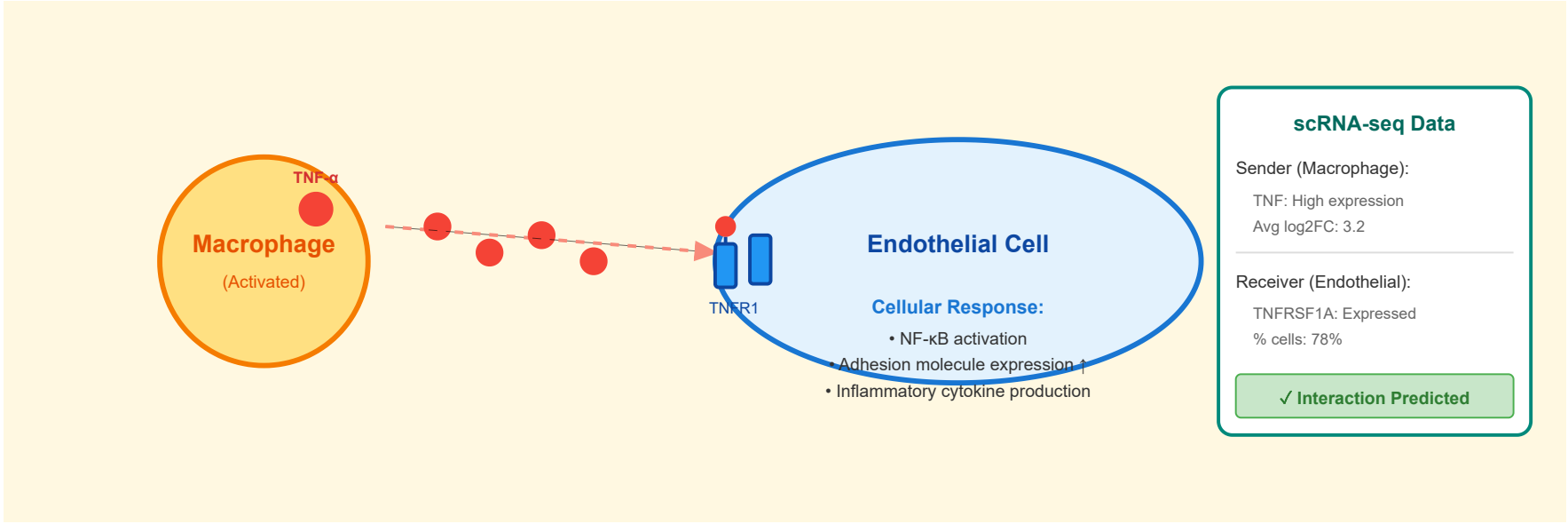
Step-by-Step Communication Process



🔬 From gene expression to cellular response: A complete signaling cascade

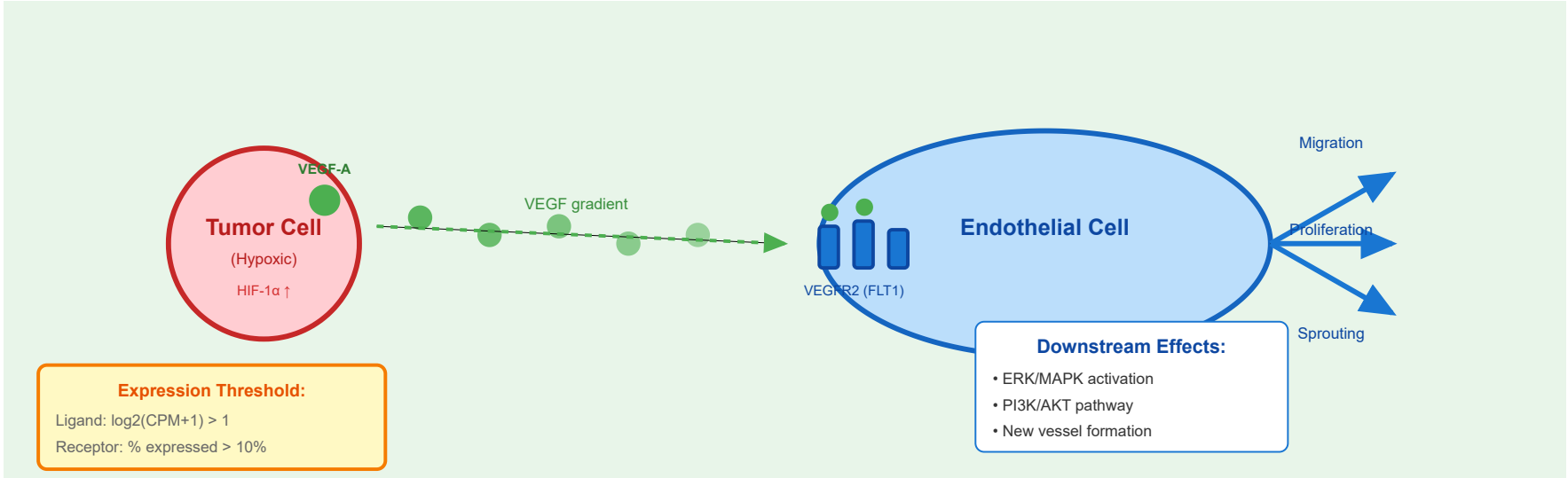
Real-World Examples of Cell-Cell Communication

Example 1: Immune Response - TNF Signaling



Biological Context: During inflammation, activated macrophages secrete TNF-α, which binds to TNFR1 on endothelial cells. This triggers the expression of adhesion molecules (ICAM-1, VCAM-1), facilitating leukocyte recruitment to inflamed tissues. This is a classic example of paracrine signaling in immune responses.

Example 2: Angiogenesis - VEGF Signaling

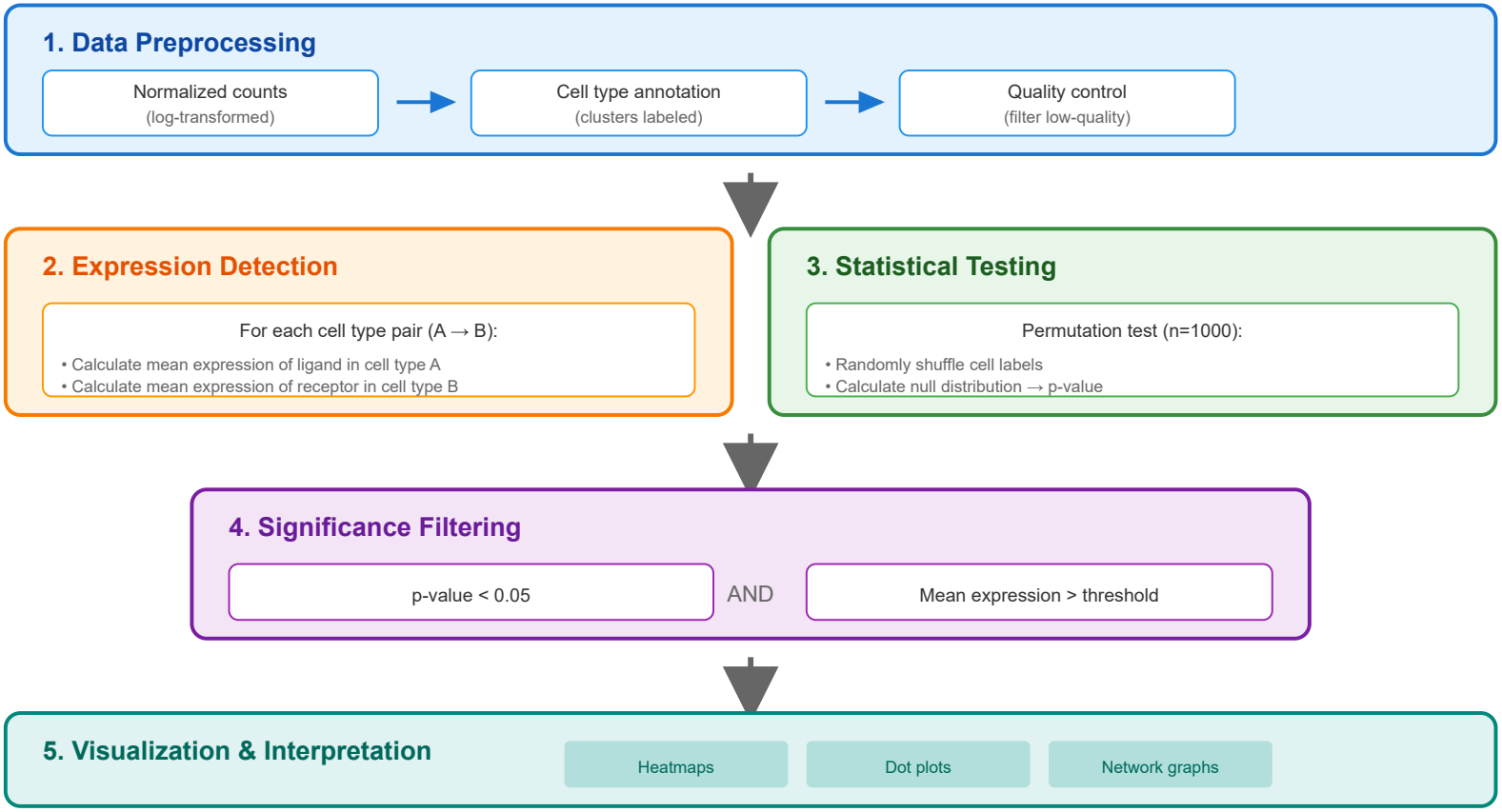


Biological Context: In hypoxic tumor microenvironments, tumor cells upregulate VEGF-A through HIF-1α. VEGF-A binds to VEGFR2 (KDR/FLK1) on endothelial cells, promoting angiogenesis. scRNA-seq can detect this interaction and help identify pro-angiogenic cell populations, making it valuable for cancer research and anti-angiogenic therapy development.

Computational Analysis Workflow

Step-by-Step Analysis Pipeline

CellPhoneDB Analysis Pipeline



Key Parameters

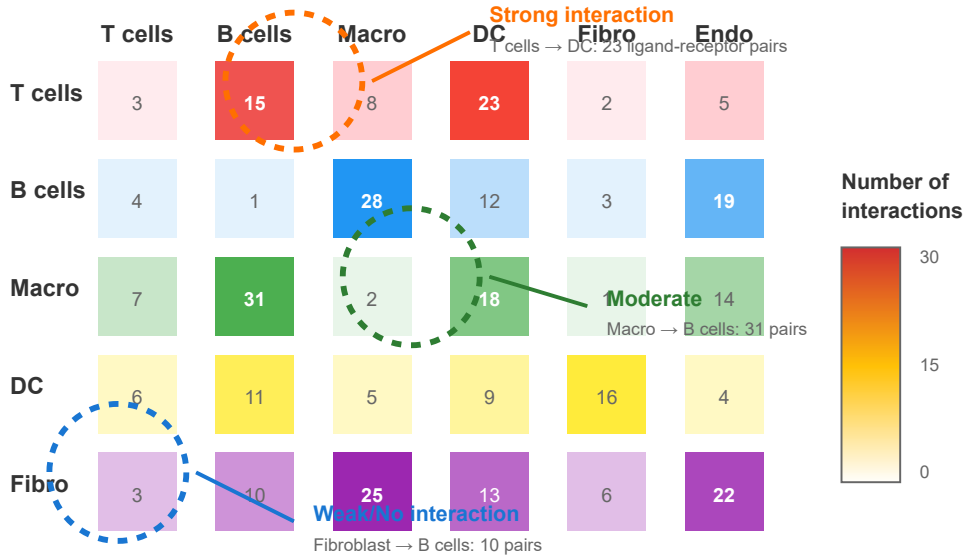
- **Expression threshold:** Minimum expression level (e.g., log2(CPM+1) > 0.1)
- **Percentage threshold:** Min % of cells expressing (e.g., >10%)
- **p-value cutoff:** Significance level (typically 0.05)
- **Permutations:** Number of randomizations (usually 1000)

Output Interpretation

- **Mean values:** Average expression of ligand and receptor
- **p-value:** Statistical significance of interaction
- **Interaction score:** Combined metric of expression and significance
- **Specificity:** How unique the interaction is to cell type pair

Interpreting Analysis Results

Understanding Heatmap Outputs



Key Interpretation Points:

- **Diagonal patterns:** Autocrine signaling (cells talking to themselves)
- **Off-diagonal hotspots:** Strong paracrine interactions between different cell types
- **Asymmetry:** Direction matters – A→B may differ from B→A (different ligands/receptors)
- **Biological validation:** Always validate computationally predicted interactions with literature or experiments
- **Spatial context:** Physical proximity increases confidence in predicted interactions

Common Pitfalls to Avoid:

- Don't assume all significant interactions are biologically relevant
- Consider expression thresholds carefully – too low leads to false positives
- Check for batch effects that might create artificial cell-cell communication signals

- Remember that correlation doesn't imply causation – experimental validation is essential

Batch Effect Correction

Comprehensive Guide to Single-Cell Integration Methods

MNN Correction

Mutual nearest neighbors for batch alignment

Harmony Algorithm

Iterative clustering and correction

LIGER Integration

Integrative non-negative matrix factorization

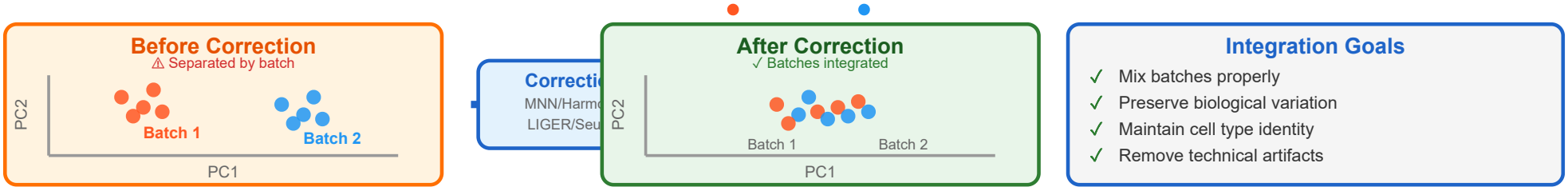
Seurat Integration

Canonical correlation analysis + anchors

Benchmark Studies

Compare methods on simulated and real data

💡 **Critical for multi-sample and multi-technology integration**



1 MNN Correction (Mutual Nearest Neighbors)

Overview

MNN correction identifies pairs of cells from different batches that are mutual nearest neighbors in high-dimensional space. These pairs represent cells of the same type across batches and are used to calculate correction vectors.

How It Works

- **Step 1:** Identify mutual nearest neighbors between batch pairs
- **Step 2:** Calculate correction vectors from MNN pairs
- **Step 3:** Apply correction to all cells using weighted averaging
- **Step 4:** Preserve local structure within each batch

> **Key Concept:** If cell A in batch 1 is among the nearest neighbors of cell B in batch 2, AND cell B is among the nearest neighbors of cell A, they form an MNN pair.

Best Used For

- Integration of datasets with similar cell type compositions
- Scenarios where batch effects are relatively mild

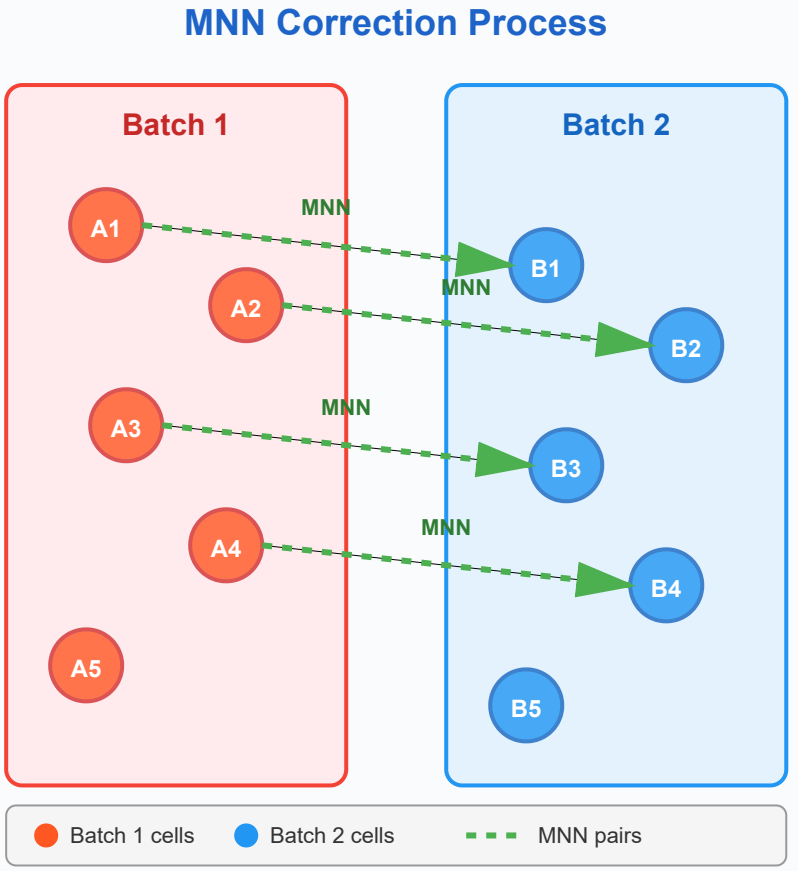
- When preserving rare cell populations is important

✓ Advantages

- Preserves biological variation well
- No overcorrection of rare populations
- Computationally efficient

✗ Limitations

- Requires overlapping cell types
- Less effective for strong batch effects
- May struggle with very different batches



2 Harmony Algorithm

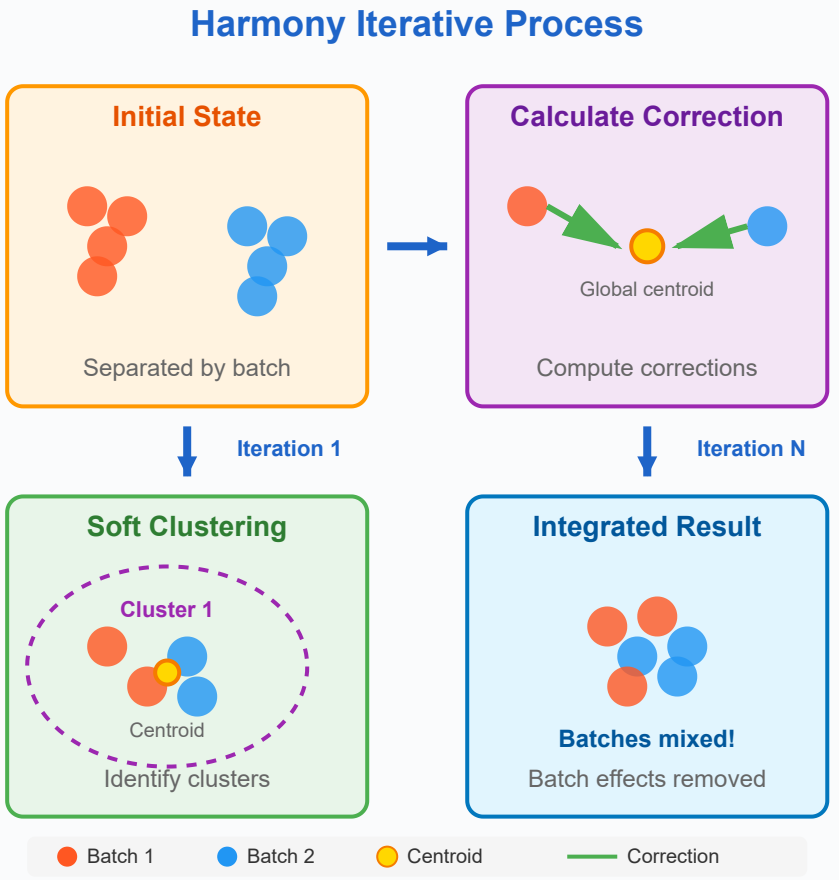
Overview

Harmony is an iterative algorithm that performs batch correction by soft clustering cells and then correcting their

positions to remove batch-specific effects while preserving biological structure. It works directly on PCA embeddings.

How It Works

- **Step 1:** Start with PCA-reduced data
- **Step 2:** Perform soft clustering to identify cell groups
- **Step 3:** Calculate batch-specific centroids for each cluster
- **Step 4:** Correct cell positions toward global centroids
- **Step 5:** Iterate until convergence



> **Key Advantage:** Harmony uses a diversity penalty to ensure that clusters are balanced across batches, preventing overclustering of any single batch.

Best Used For

- Large-scale datasets with multiple batches
- Strong batch effects requiring aggressive correction
- Fast integration of many samples (computationally efficient)
- Integration across different sequencing platforms

✓ Advantages

- Very fast and scalable
- Works on existing PCA embeddings

✗ Limitations

- May overcorrect biological variation

- Handles strong batch effects well
- Simple to implement

- Less control over correction strength
- Can merge distinct cell states

3 LIGER Integration (iNMF)

Overview

LIGER (Linked Inference of Genomic Experimental Relationships) uses integrative non-negative matrix factorization (iNMF) to identify shared and dataset-specific factors. It decomposes gene expression into shared biological signals and batch-specific technical variations.

How It Works

- **Step 1:** Factorize each dataset's expression matrix
- **Step 2:** Identify shared factors (W) across datasets
- **Step 3:** Learn dataset-specific factors (V_1, V_2, ...)
- **Step 4:** Use shared factors for integration
- **Step 5:** Quantile normalize factor loadings

Mathematical Framework: For dataset i: $X_i \approx W \times H_i + V_i \times H_i$, where W contains shared factors and V_i contains dataset-specific factors.

Best Used For

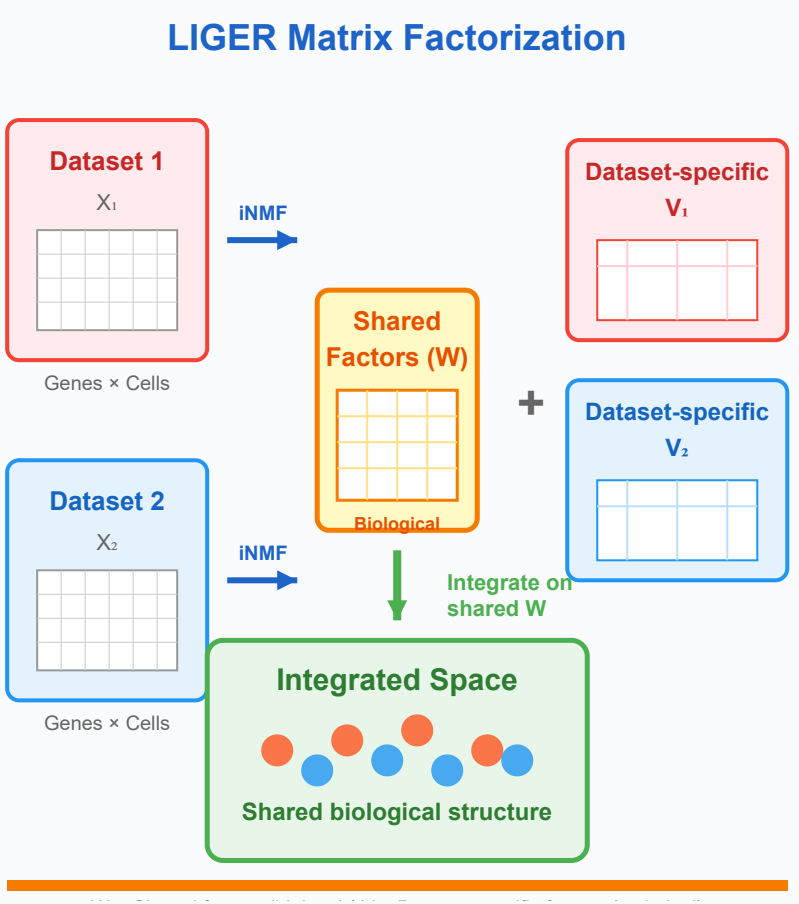
- Multi-modal data integration (e.g., scRNA-seq + scATAC-seq)
- Cross-species comparisons
- Datasets with fundamentally different feature sets
- When you need to identify shared vs. unique biological signals

✓ Advantages

- Works with different feature spaces
- Identifies shared biology explicitly
- Great for multi-modal integration
- Preserves dataset-specific signals

✗ Limitations

- Computationally intensive
- Requires parameter tuning (k factors)
- More complex to implement
- Memory intensive for large datasets



4 Seurat Integration (CCA + Anchors)

Overview

Seurat integration uses Canonical Correlation Analysis (CCA) to identify shared correlation structures between datasets, then finds "anchor" cells that represent correspondences across batches. These anchors guide the integration process.

How It Works

- **Step 1:** Identify highly variable genes in each dataset
- **Step 2:** Perform CCA to find shared correlation structures
- **Step 3:** Identify mutual nearest neighbors as "anchors"
- **Step 4:** Score and filter anchors based on similarity
- **Step 5:** Use anchors to harmonize datasets

> **Key Innovation:** Anchors are high-confidence cell pairs that serve as reference points for integration, allowing precise correction while preserving biological heterogeneity.

Best Used For

- Standard scRNA-seq integration workflows

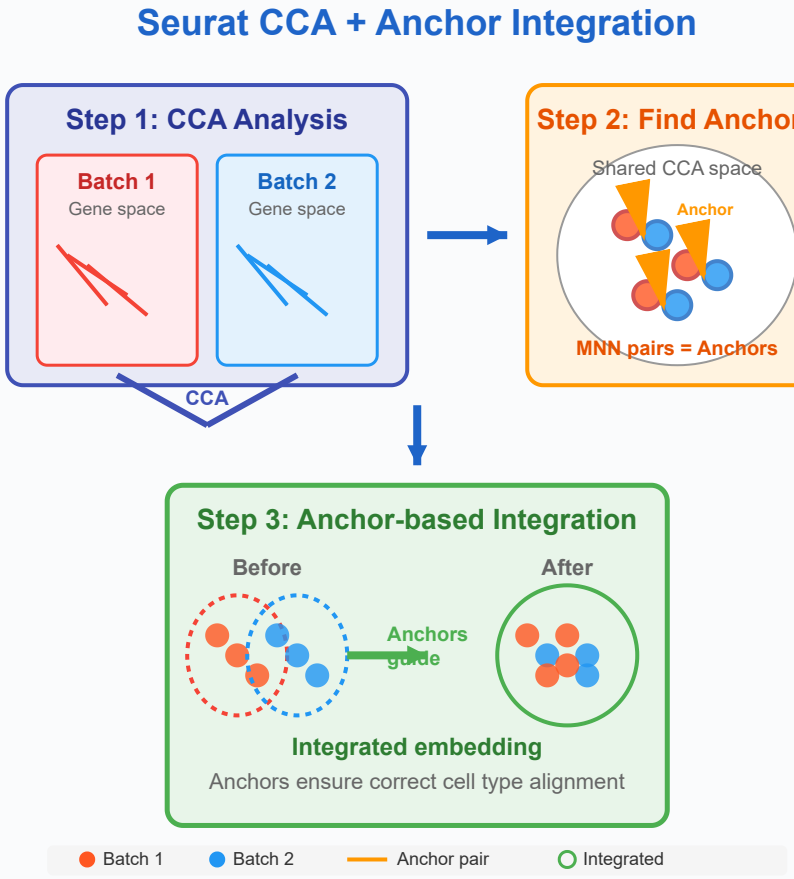
- Datasets with good cell type overlap
- When you want fine control over integration
- Reference-based integration (map query to reference)

✓ Advantages

- Well-established and validated
- Excellent documentation
- Works well with Seurat ecosystem
- Flexible anchor filtering

✗ Limitations

- Computationally demanding
- Memory intensive for large datasets
- Requires overlapping cell populations
- Can be slow with many datasets



5 Benchmark Studies & Method Comparison

Overview

Benchmark studies systematically compare integration methods using standardized metrics on both simulated and real datasets.

These studies help researchers choose the most appropriate method for their specific use case.

Key Evaluation Metrics

- **Batch Mixing:** How well cells from different batches are mixed (e.g., kBET, LISI)
- **Bio-conservation:** Preservation of biological variation (e.g., ARI, NMI, ASW)
- **Cell Type Purity:** Maintenance of distinct cell populations
- **Computational Efficiency:** Runtime and memory usage
- **Scalability:** Performance with increasing dataset size

> **Important Finding:** No single method is universally best. Method selection depends on batch effect strength, dataset characteristics, and analysis goals.

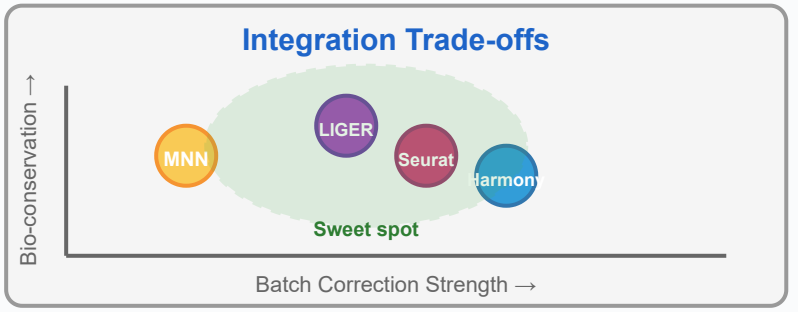
General Guidelines

- **Mild batch effects:** MNN, fastMNN
- **Strong batch effects:** Harmony, Seurat
- **Multi-modal data:** LIGER, MultiVI
- **Large-scale data:** Harmony (fastest), scVI
- **Reference mapping:** Seurat, Symphony

Notable Benchmark Papers

Method Performance Comparison

Method	Batch Mixing	Bio-conservation	Speed
MNN	75%	90%	70%
Harmony	95%	80%	95%
LIGER	85%	95%	60%
Seurat	90%	85%	55%



- Luecken et al. (2021) - Comprehensive scRNA-seq integration comparison
- Tran et al. (2020) - Evaluation across 77 batches
- Chazarra-Gil et al. (2021) - Flexible benchmarking framework

✓ Best Practices

- Test multiple methods on your data
- Check both mixing AND biology
- Visualize before/after integration
- Use appropriate metrics for evaluation

⚠ Common Pitfalls

- Overcorrection removes biology
- Undercorrection leaves batch effects
- Ignoring method assumptions
- Not validating integration quality

💡 **Remember: Always validate your integration results by checking both batch mixing and biological signal preservation!**

Integration Methods for Single-Cell Analysis

Anchor-based Methods

Seurat, LIGER find correspondence between datasets

Deep Learning Approaches

scVI, scGAN learn shared latent space

Reference Building

Create comprehensive cell atlases

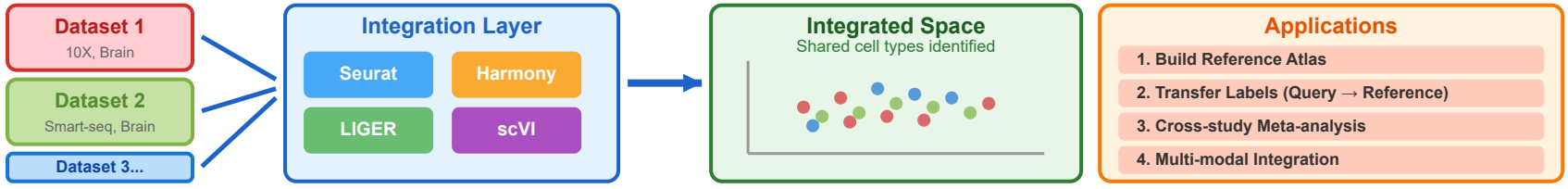
Query Mapping

Project new data onto reference

Performance Metrics

Biological conservation vs batch mixing

💡 Integration enables meta-analysis and transfer learning



Detailed Method Descriptions

1 Anchor-based Methods

Overview

Anchor-based methods identify mutual nearest neighbors (MNNs) or "anchors" between datasets to align cells across different batches or technologies. These anchors serve as reference points for integration.

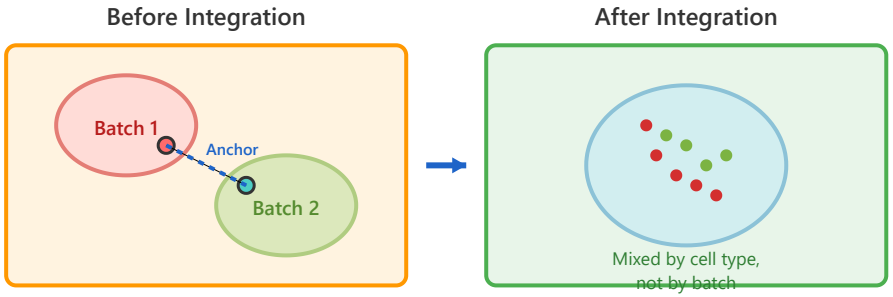
Key Algorithms

- **Seurat v3/v4:** Uses canonical correlation analysis (CCA) to find shared correlation structures, then identifies anchors via mutual nearest neighbors
- **LIGER:** Uses integrative non-negative matrix factorization (iNMF) to discover shared and dataset-specific factors
- **Harmony:** Iteratively clusters cells and corrects for batch effects using soft k-means clustering

Advantages

- Computationally efficient for large datasets
- Preserves biological variation while removing batch effects
- Well-validated and widely adopted

Anchor-based Integration Process



Algorithm Steps

1. **Feature Selection**
   Identify highly variable genes across datasets
2. **Anchor Identification**
   Find mutual nearest neighbors (MNNs) between datasets
3. **Integration**
   Use anchors to align and integrate datasets into shared space

⭐ Key Points

- Anchors are pairs of cells from different datasets that represent the same cell type or state
- MNN approach ensures bidirectional nearest neighbors, reducing false positives

- Works well when datasets share common cell types but differ in technology or batch

2 Deep Learning Approaches

Overview

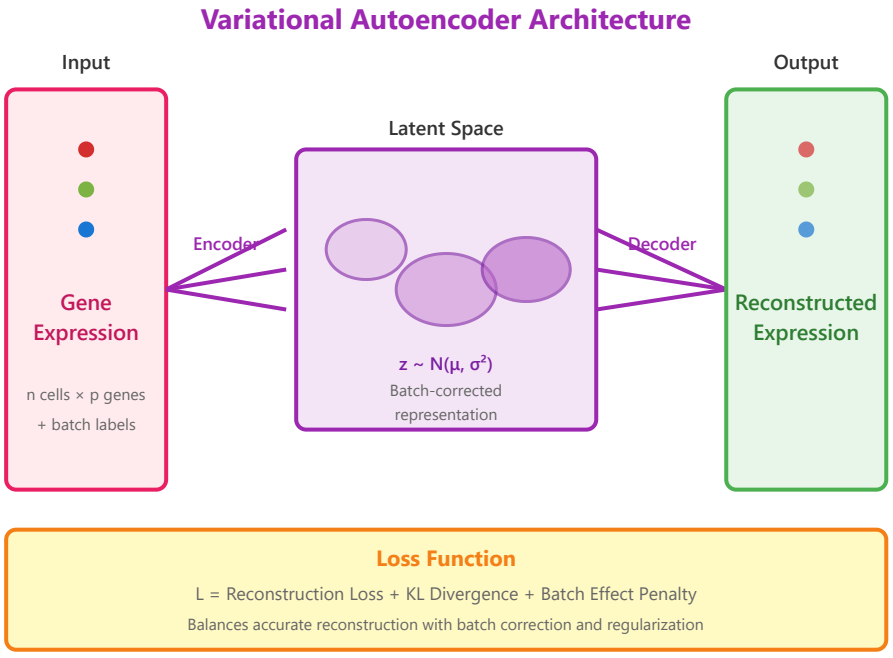
Deep learning methods use neural networks to learn a shared latent representation that captures biological variation while removing technical effects. These models can handle complex non-linear relationships.

Key Algorithms

- **scVI (Single-Cell Variational Inference):** Uses variational autoencoders (VAE) to model gene expression with explicit batch correction
- **scGAN:** Generative adversarial network that learns batch-invariant representations
- **scANVI:** Semi-supervised version of scVI that incorporates cell type labels
- **SAUCIE:** Autoencoder with explicit regularization for batch effects

Advantages

- Captures complex non-linear relationships
- Probabilistic framework provides uncertainty estimates
- Can incorporate multiple data modalities



- Scales to millions of cells

Key Points

- VAE learns a probabilistic latent representation with explicit uncertainty quantification
- Batch information is provided as input but removed from latent space through adversarial training or regularization
- Can impute missing values and denoise expression data as part of the integration process
- Requires GPU acceleration for large datasets but provides state-of-the-art performance

3 Reference Building

Overview

Reference building involves creating comprehensive cell atlases by integrating multiple high-quality datasets. These references serve as standardized maps of cell types and states for a given tissue or organism.

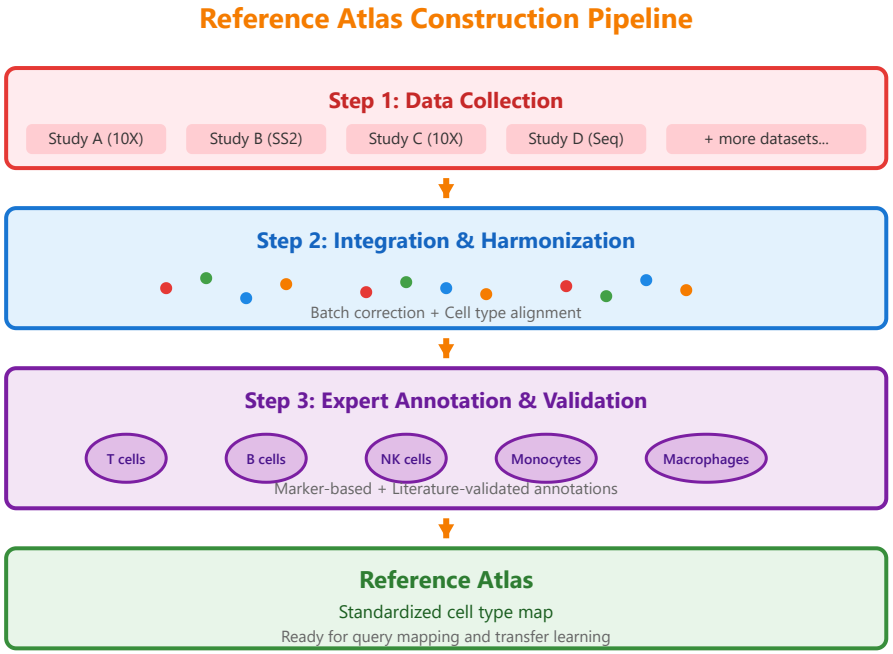
Construction Process

- **Data Collection:** Aggregate datasets from multiple sources, technologies, and conditions
- **Quality Control:** Rigorous filtering and validation of cells and annotations
- **Integration:** Harmonize datasets while preserving biological heterogeneity
- **Annotation:** Comprehensive cell type labeling by expert curators

- **Validation:** Cross-validation and benchmarking against known markers

Major Reference Atlases

- **Human Cell Atlas (HCA):** Comprehensive map of human cells
- **Tabula Sapiens:** Reference for human organ systems
- **Mouse Cell Atlas:** Complete mouse cell type map



⭐ Key Points

- References require diverse, high-quality datasets spanning multiple conditions and technologies
- Expert curation ensures accurate and consistent cell type annotations
- Regular updates incorporate new data and refined annotations
- Enables standardized analysis and comparison across studies

4 Query Mapping

Overview

Query mapping projects new datasets onto existing reference atlases, enabling rapid cell type annotation and

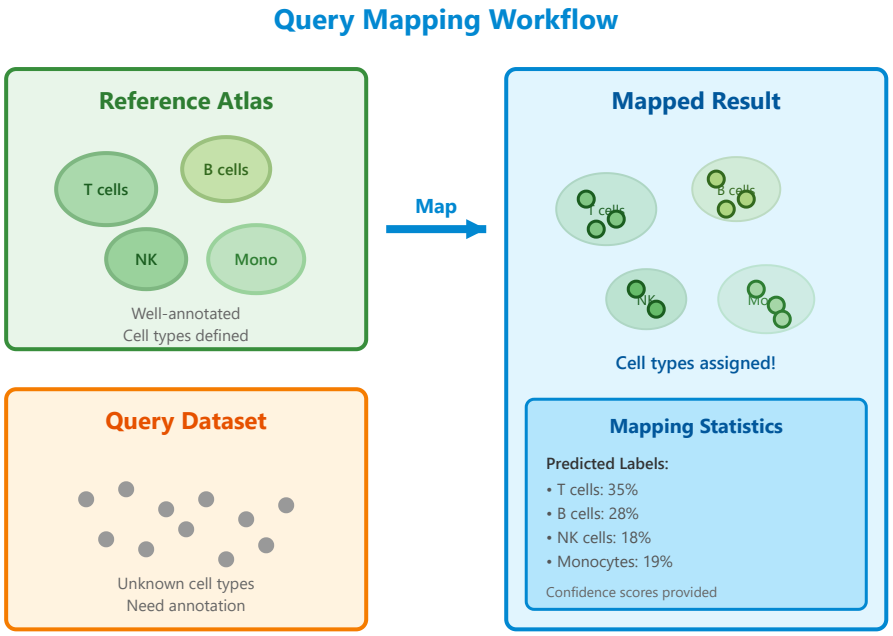
comparison without full re-integration. This is a form of transfer learning for single-cell analysis.

Mapping Methods

- **Seurat Reference Mapping:** Projects query onto PCA space of reference using supervised anchors
- **Symphony:** Fast reference mapping using harmonized coordinates
- **scArches:** Transfer learning with neural networks, updates model with query data
- **SingleR:** Correlation-based cell type assignment from reference

Applications

- Rapid annotation of new datasets
- Cross-study comparisons
- Disease vs. healthy comparison
- Time-series or perturbation studies



Key Points

- Much faster than full integration (seconds to minutes vs. hours)
- Provides confidence scores for cell type predictions
- Works best when query and reference share common cell types
- Novel cell types in query may be assigned to nearest reference type (limitation)
- Enables large-scale studies by reusing well-curated references

5 Performance Metrics

Overview

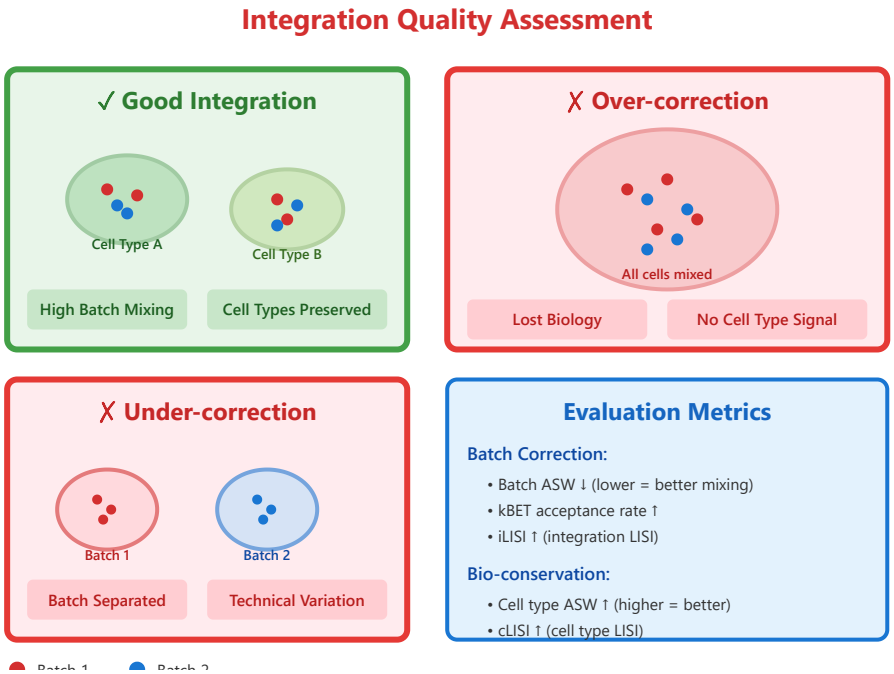
Evaluating integration quality requires balancing two competing objectives: removing technical variation (batch mixing) while preserving biological variation (cell type separation).

Key Metrics

- **Batch Mixing Metrics:**
  - Batch ASW (Average Silhouette Width): Measures batch separation
  - kBET (k-nearest neighbor Batch Effect Test): Tests batch mixing
  - LISI (Local Inverse Simpson's Index): Quantifies local diversity
- **Bio-conservation Metrics:**
  - Cell type ASW: Measures cell type separation
  - ARI (Adjusted Rand Index): Compares clustering to labels
  - NMI (Normalized Mutual Information): Information preservation
- **Trajectory Preservation:** Ensures developmental/temporal relationships maintained

Benchmarking Studies

- No single method dominates all scenarios



- Performance depends on dataset characteristics and goals

⭐ Key Points

- Integration is a balancing act: remove batch effects without losing biology
- Multiple metrics needed to assess both batch mixing and bio-conservation
- Over-correction merges distinct cell types; under-correction leaves technical variation
- Optimal method depends on dataset characteristics (number of batches, cell types, technologies)
- Visual inspection (UMAP plots) combined with quantitative metrics provides best assessment

Hands-on: Seurat Tutorial

Seurat v5 Standard Workflow

1. Data Loading & QC

```
# Read 10X data
data <- Read10X("filtered_feature_bc_matrix/")
seurat <- CreateSeuratObject(data, min.cells=3)
```

2. QC Filtering

```
# Calculate mitochondrial %
seurat[["percent.mt"]] <- PercentageFeatureSet(seurat, "^MT-")
seurat <- subset(seurat, nFeature_RNA > 200 & percent.mt < 10)
```

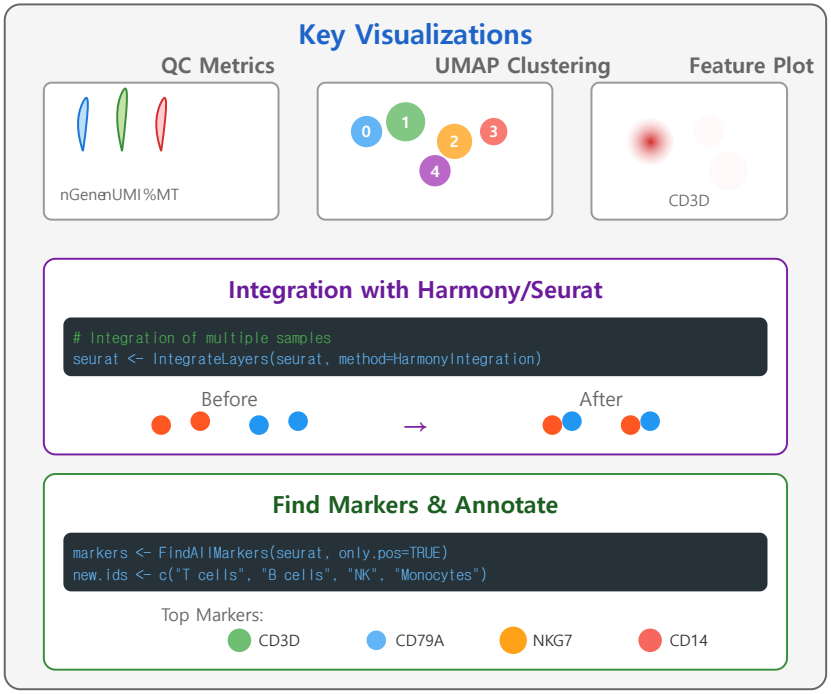
3. Normalization & Scaling

```
seurat <- NormalizeData(seurat)
seurat <- FindVariableFeatures(seurat, nfeatures=2000)
seurat <- ScaleData(seurat)
```

4. Dimension Reduction & Clustering

```
seurat <- RunPCA(seurat) %>% RunUMAP(dims=1:30)
seurat <- FindNeighbors(seurat) %>% FindClusters(res=0.5)
DimPlot(seurat, label=TRUE) + NoLegend()
```

Key Visualizations

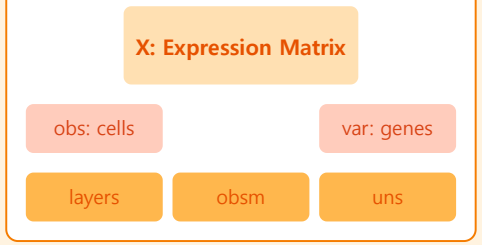


💡 Most widely used R package for scRNA-seq analysis

Hands-on: Scanpy Analysis

Scanpy: Python-based Single Cell Analysis

AnnData Structure



Standard Workflow

```python
# Import libraries
import scanpy as sc
import pandas as pd
adata = sc.read_10x_h5('file.h5')
```

```python
# QC and filter
sc.pp.calculate_qc_metrics(adata)
sc.pp.filter_cells(adata, min_genes=200)
sc.pp.filter_genes(adata, min_cells=3)
```

```python
# Normalize and find HVGs
sc.pp.normalize_total(adata)
sc.pp.log1p(adata)
sc.pp.highly_variable_genes(adata)
```

```python
# Dimension reduction
sc.tl.pca(adata)
sc.pp.neighbors(adata)
sc.tl.umap(adata)
```

Advanced Scanpy Features

Trajectory Analysis

```python
sc.tl.paga(adata)
sc.tl.dpt(adata)
```

RNA Velocity

```python
import scvelo as scv
scv.tl.velocity(adata)
```

GPU Acceleration

```python
import rapids_singlecell
# 100x speedup!
```

Large datasets

Data Integration

```python
sc.external.pp.harmony_
integrate(adata, 'batch')
```

💡 Python ecosystem with extensive documentation

Thank you!

Key Applications

- Disease studies - Cell type changes in pathology
- Development biology - Cell fate trajectories
- Drug discovery - Target identification and validation
- Clinical futures - Diagnostic and therapeutic applications