# Cell Type Annotation in Spatial Transcriptomics

Methods, Tools, and Best Practices

# Course Overview

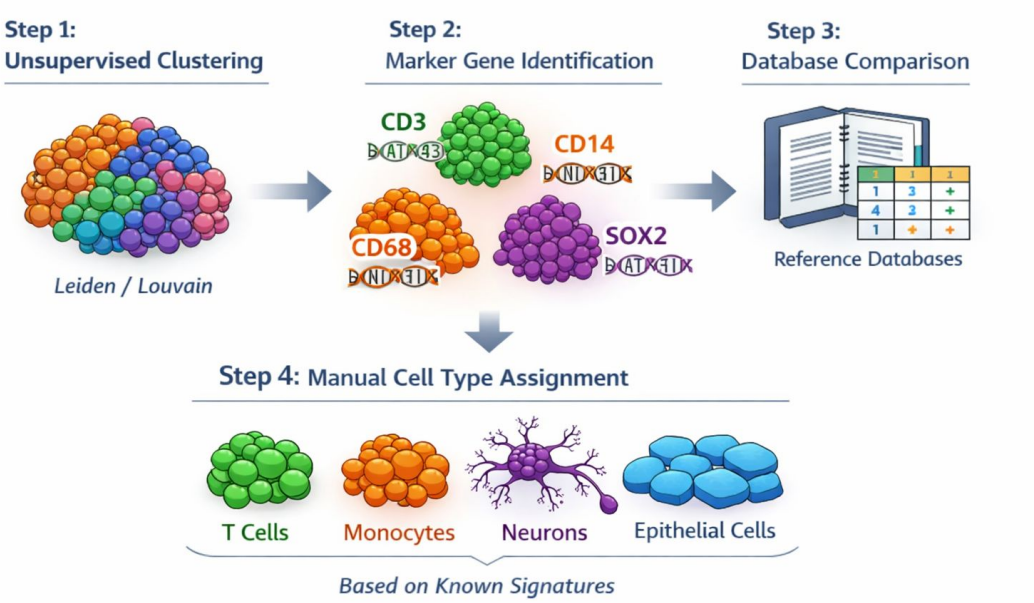
- Introduction to cell type annotation approaches
- Traditional methods: clustering and marker genes
- Reference-based annotation methods
  - Tangram
  - Seurat (Reference Mapping)
  - CoDi (Contrastive Distance)
- Quality assessment strategies
- Ensemble annotation approaches
- Finding or Creating Good Reference scRNA-seq Datasets
- Cell type annotation framework

# Traditional Annotation Approaches

Clustering, Marker Genes, and Reference Databases

# Traditional Cell Type Annotation Workflow

- Step 1: Perform unsupervised clustering (Leiden/Louvain)
- Step 2: Identify cluster-specific marker genes
- Step 3: Compare markers against reference databases
- Step 4: Manual assignment based on known cell type signatures
- This approach relies on expert knowledge and literature
- Works best when cell types have well-characterized markers



# Reference Databases for Marker-Based Annotation

- CellMarker (mouse and human)
  - Curated database of cell type markers
  - Covers human and mouse tissues
- PanglaoDB (mouse and human)
  - Community-curated marker gene database
  - Includes cell type-specific gene lists
- The Human Protein Atlas - to verify markers
  - Single-cell type annotations
  - Tissue-specific expression data
- scType - Automated online annotation tool

# Traditional Methods: Pros and Cons

## Advantages

- No reference scRNA-seq data required
- Interpretable - based on known biology
- Works across species and tissues
- Can discover novel cell types/states
- Low computational requirements
- Flexible - adapts to specific tissues

## Limitations

- Requires expert biological knowledge
- Time-consuming manual curation
- Subjective - depends on analyst
- Markers may not transfer across tissues
- Difficult for rare or novel cell types
- Cluster boundaries don't match cell types

# Reference-trained Cell Type Annotation with CellTypist

- A machine-learning classifier for cell types
- Given a cell's gene expression vector, CellTypist predicts the most likely cell type using a regularized logistic regression model trained on large curated scRNA-seq reference atlases
- Avoid manual marker curation
- Pre-trained models for multiple tissues/species
- Supports majority-voting for smoothing noisy predictions
- Provides cell-level probability matrix → confidence estimation
- Workflow
  - Load and normalize spatial transcriptomics data
  - Select a tissue- and species-specific model (Mouse_Whole_Brain.pkl)
  - Run CellTypist:
    - predicted_labels → raw prediction
    - over_clustering → internal stabilization
    - majority_voting → final smoothed label per spot
- Inspect spatial distribution and marker alignment

# Manual Cell Type Annotation with predifined markers

- Clustering and Marker-Based Cell Type Assignment
  - Feature selection: Select 2,000 highly variable genes
  - Dimensionality reduction: PCA on HVGs (50 components)
  - Graph construction: kNN graph (15 neighbors, 50 PCs)
  - Clustering: Leiden algorithm (resolution = 1.5)
- Cell Type Annotation
  - Define cell-type marker gene sets
  - Compute mean expression per cluster
  - Score clusters by average marker expression
  - Assign each cluster the highest-scoring cell type
  - Map cluster labels back to individual cells

# Reference-Based Annotation Methods

Leveraging High-Quality scRNA-seq References

# Why Reference-Based Annotation?

- Spatial transcriptomics often has lower sensitivity than scRNA-seq
- scRNA-seq provides gold-standard cell type definitions
- Transfer learning: leverage existing annotations
- More objective and reproducible than manual annotation
- Can handle complex mixtures (deconvolution)
- Enables cross-study comparisons with consistent labels

# Tangram

Mapping scRNA-seq to Spatial Data via Deep Learning

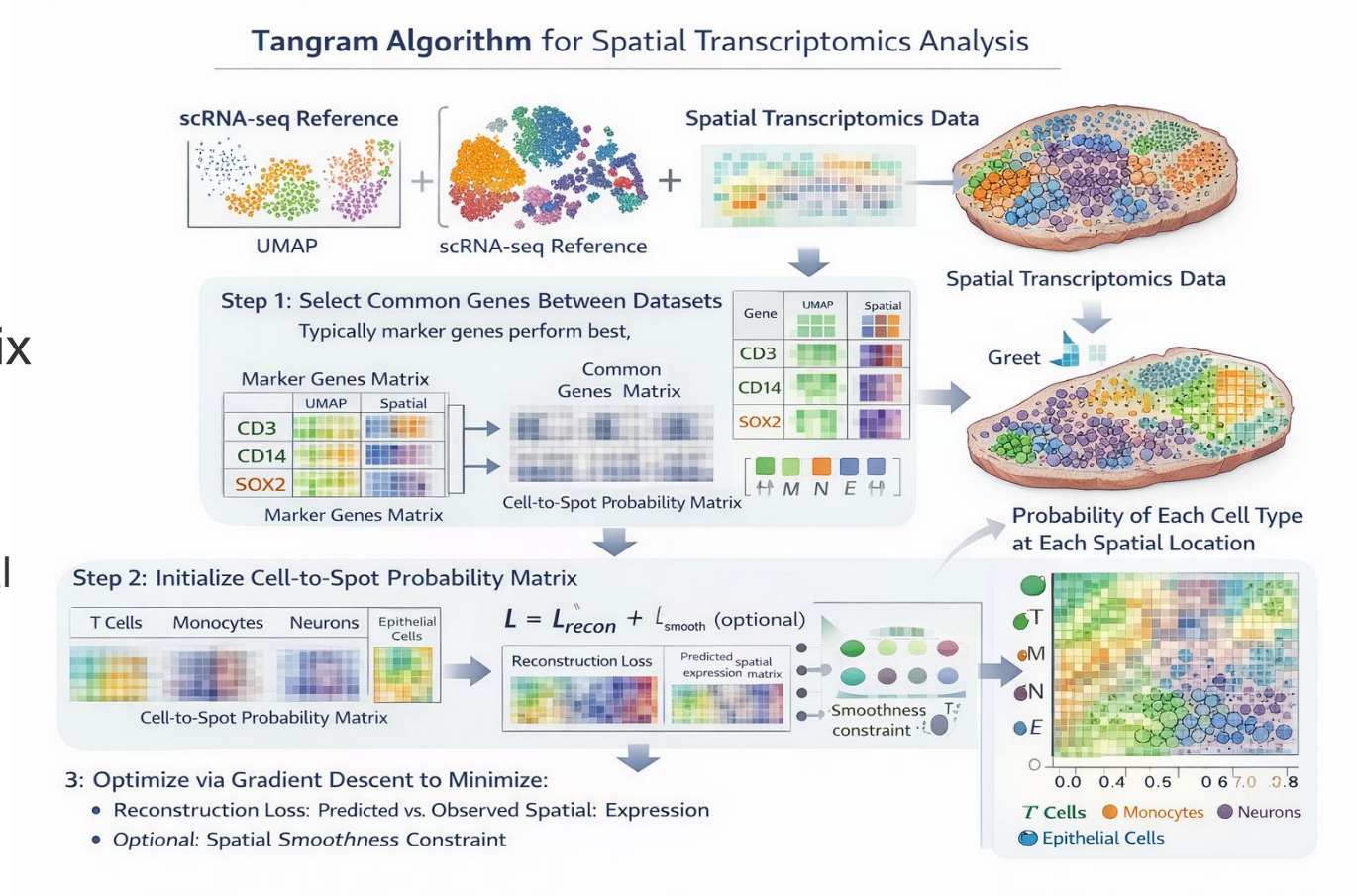
# Tangram

Deep learning-based spatial mapping of scRNA-seq data

- Developed by the Bhattacharyya lab (Broad Institute), Published in Nature Methods (2021)
- Maps single cells to spatial locations probabilistically
- Uses gene expression to learn spatial distributions
- Can transfer cell type annotations AND gene expression
- Particularly good for high-resolution spatial data (Visium, Slide-seq)

# Tangram: How It Works Under the Hood

- Input: scRNA-seq reference + spatial transcriptomics data
- Step 1: Select common genes between datasets
    - Typically marker genes perform best
- Step 2: Initialize cell-to-spot probability matrix
- Step 3: Optimize via gradient descent to minimize:
    - Reconstruction loss: predicted vs. observed spatial expression
    - Optional: spatial smoothness constraint
- Output: Probability of each cell type at each spatial location
- Uses PyTorch for GPU-accelerated optimization



# Tangram: Running the Analysis

Complete Tangram workflow for cell type mapping

```
import tangram as tg
import scanpy as sc

# Load data
adata_sc = sc.read_h5ad('scrna_reference.h5ad')  # scRNA-seq
adata_sp = sc.read_h5ad('spatial_data.h5ad')     # Spatial

# Find marker genes (or use pre-defined)
sc.tl.rank_genes_groups(adata_sc, groupby='cell_type')
markers = tg.get_marker_genes(adata_sc, groupby='cell_type')

# Prepare data
tg.pp_adatas(adata_sc, adata_sp, genes=markers)

# Map cells to spatial locations
ad_map = tg.map_cells_to_space(
    adata_sc, adata_sp,
    mode='cells',        # or 'clusters' for faster
    density_prior='uniform'  # or 'rna_count_based'
)
# Transfer annotations
tg.project_cell_annotations(ad_map, adata_sp, 'cell_type')
```

# Tangram: Key Benefits

- Probabilistic mapping captures uncertainty
- Can map individual cells or clusters (scalability)
- GPU acceleration for large datasets
- Preserves spatial continuity of cell types
- Can also project gene expression (imputation) - see and explore, not to test or claim
- Works well with Visium and other technologies
- Active development and good documentation

# Seurat Reference Mapping

Label Transfer via Anchor-Based Integration

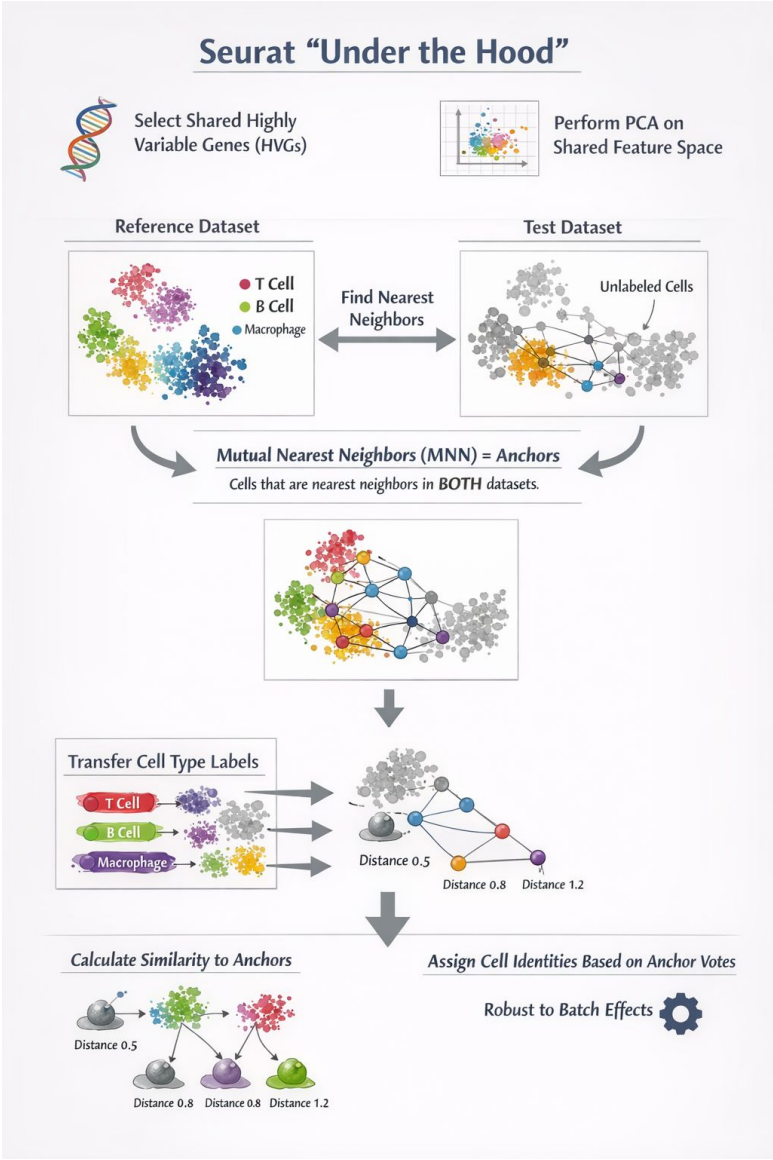
# Seurat

## Anchor-based integration and label transfer

- Part of Seurat v4/v5 R integration framework
- Uses mutual nearest neighbors (MNN) to find anchors
- An anchor is a pair of cells from two datasets that are mutual nearest neighbors of each other in a shared feature space (typically PCA computed on shared highly variable genes).
- Transfers labels from reference to query dataset
- Works for any single-cell technology, including spatial
- Widely used, well-documented, large community

# Seurat: How It Works Under the Hood

- Select shared highly variable genes (HVGs) between datasets
- Perform PCA to embed all cells in a shared feature space
- For each cell in Reference Dataset → find nearest neighbors in Test Dataset
- For each cell in Test Dataset → find nearest neighbors in Reference Dataset
- Define anchor = cells that are mutual nearest neighbors (MNN) in both ways
- Anchors link transcriptionally similar cells across datasets
- Transfer cell type labels from annotated dataset to unannotated dataset
- Assign cell identities based on anchor-supported similarity
    - For every unannotated cell, calculate similarity to each anchor (distance in PCA space)
- Key strength: robust to batch effects between datasets



# Seurat: Running Label Transfer (R)

Seurat label transfer workflow

```
# Find transfer anchors
anchors <- FindTransferAnchors(
    reference = ref,
    query = spatial,
    normalization.method = "SCT",
    reference.reduction = "pca",
    dims = 1:30
)

# Transfer labels
predictions <- TransferData(
    anchorset = anchors,
    refdata = ref$cell_type,
    weight.reduction = spatial[["pca"]],
    dims = 1:30
)

# Add to spatial object
spatial <- AddMetaData(spatial, predictions)
SpatialDimPlot(spatial, group.by = "predicted.id")
```

# Seurat: Key Benefits

- Battle-tested on thousands of datasets
- Excellent documentation and tutorials
- Prediction scores help assess confidence
- Can integrate multiple references
- Built-in spatial visualization functions
- Active development and community support

# CoDi

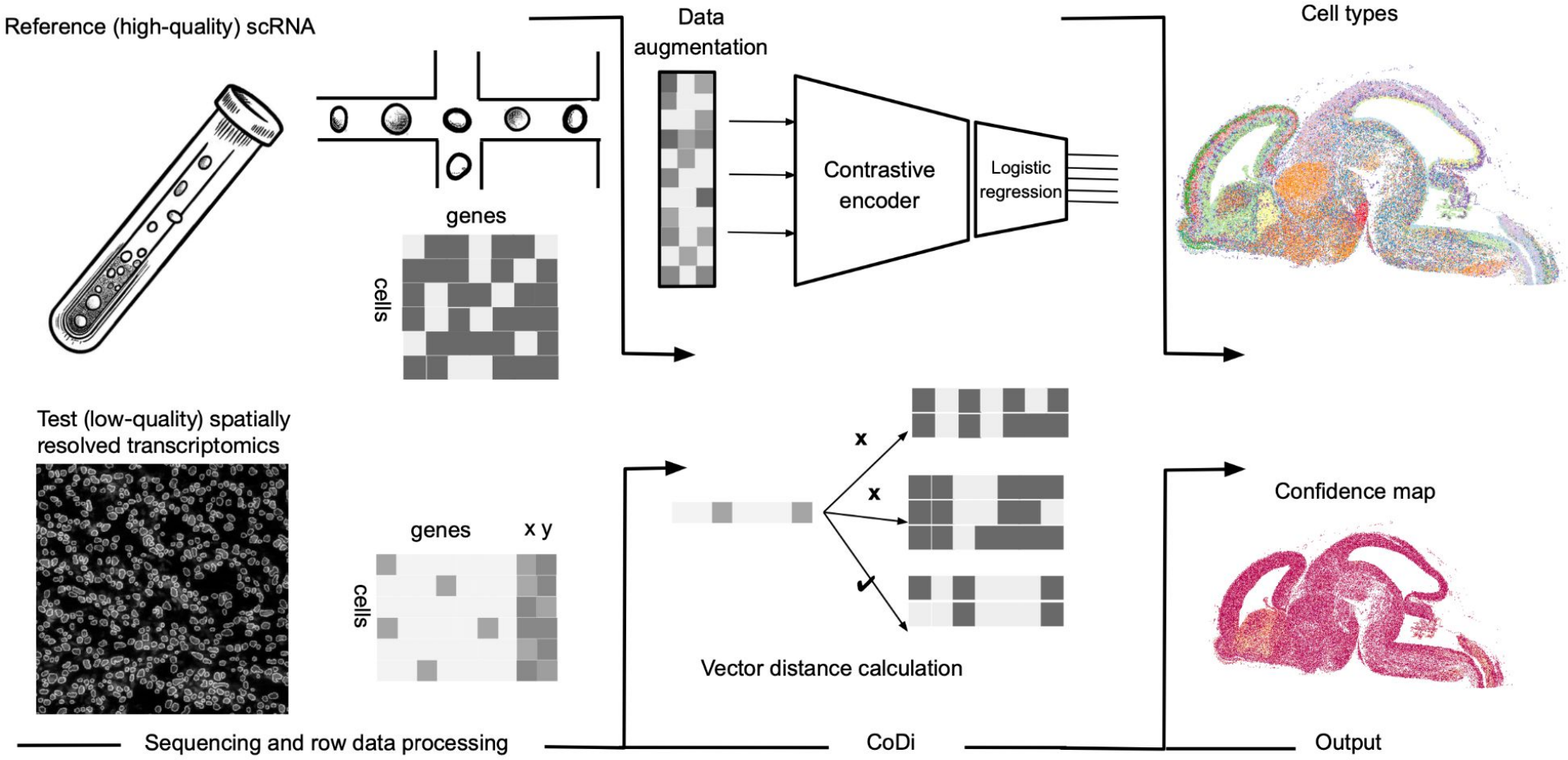
Contrastive Distance Cell Type Annotation

# CoDi

Contrastive learning for robust cell type annotation

- Leverages:
  - Contrastive learning
  - Advanced distance calculation with reference single-cell datasets
- Designed for accuracy and scalability in large ST datasets
- Robust to batch effects and technical variation
- Can handle imbalanced cell type distributions
- Designed specifically for transfer learning scenarios
- Superior performance across multiple evaluation metrics
  - Marker gene retention: 5–25% higher than existing tools
  - Annotation accuracy: 0.1–30% higher on downsampled datasets
  - Capable of identifying rare cell types (e.g., neurons in the heart)
- Enhances understanding of cellular environments
- Represents a next-generation solution for ST-based cell type annotation

# CoDi: Under the Hood



# CoDi: Running the Analysis (Python)

CoDi workflow for contrastive annotation

```
python CoDi.py --sc_path <single_cell_dataset.h5ad>
               --st_path <spatial_dataset.h5ad>
               -a <sc_annotation>

docker run --rm \
    -v "${PWD}:/data" \
    vladimirkovacevic/codi \
    python /opt/codi/CoDi.py \
    --sc_path /data/your_single_cell_dataset.h5ad \
    --st_path /data/your_spatial_dataset.h5ad \
    -a "sc_annotation"
```

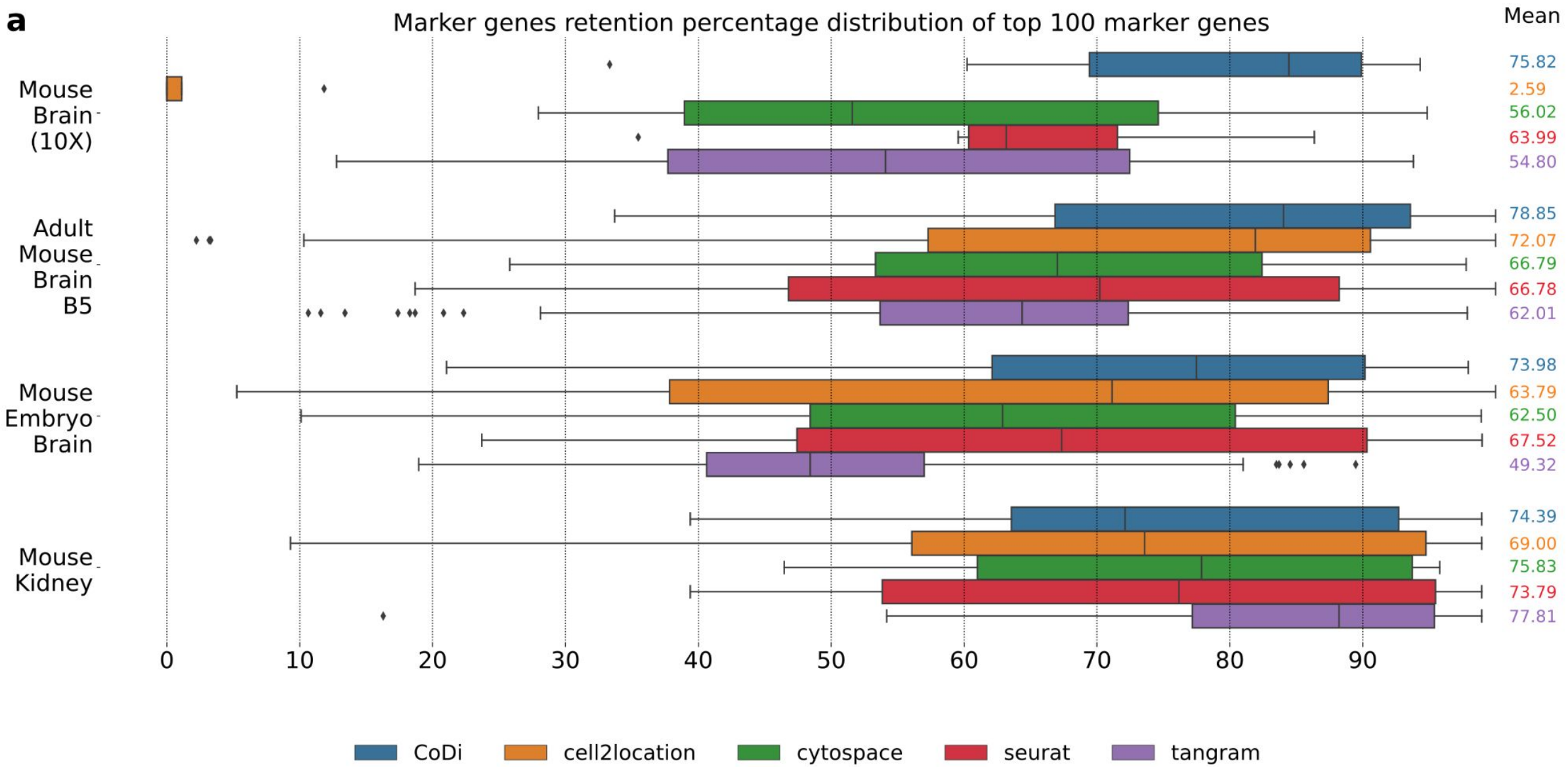
# Quality Assessment

Evaluating Annotation Accuracy

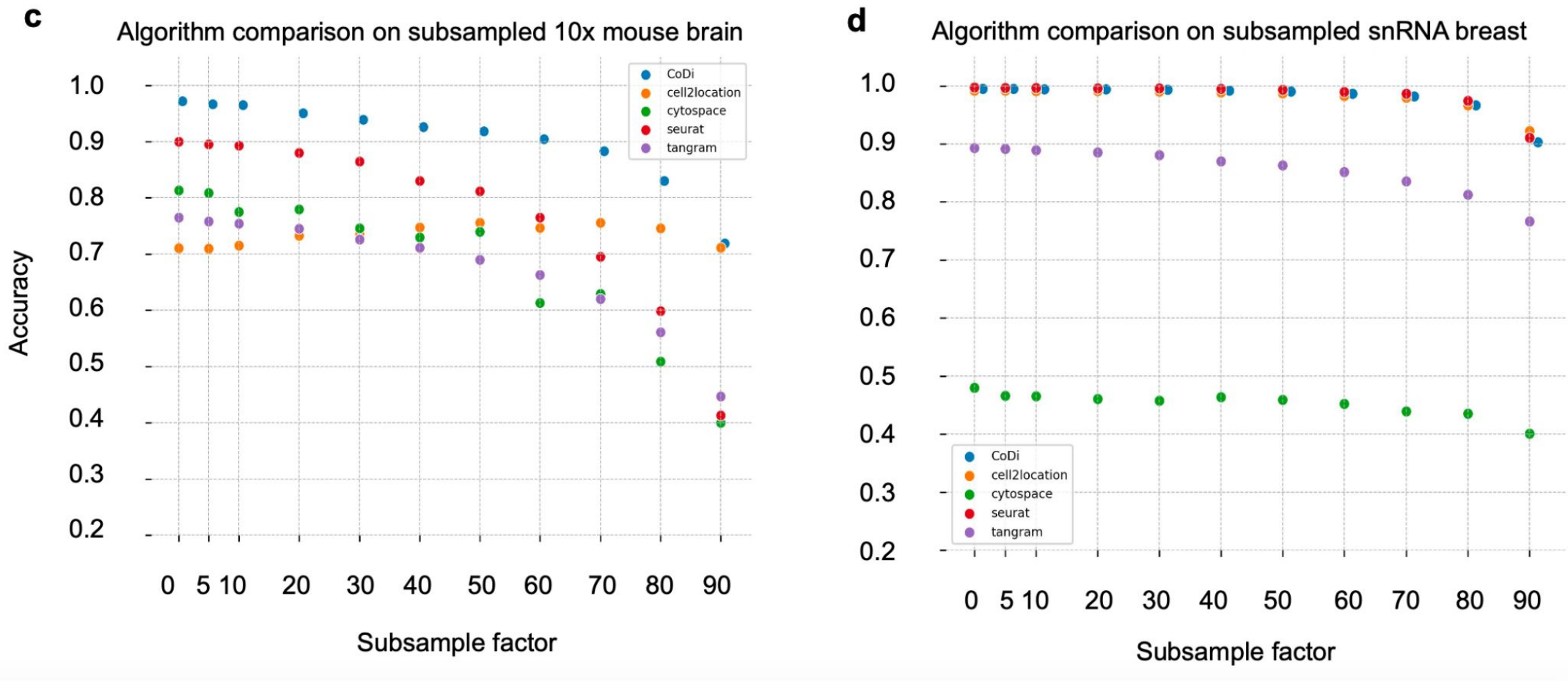
# Assessing Annotation Quality

- Critical step often overlooked!
- Two main validation strategies:
  - Marker gene retention analysis
  - Accuracy on downsampled scRNA dataset
  - Spatial distribution validation

# Quality comparison - Retention of marker genes



# Quality comparison - Accuracy on downsampled reference



# Performance Requirements

## ⏱️ Execution Time

| Execution time (min) | Tangram | Cytospace | Seurat | cell2 location | CoDi |
|---|---|---|---|---|---|
| Adult mouse brain B5 (Stereo-seq) | 36.66 | 26.11 | 30.85 | 83.07 | 156.30 |
| Whole brain mouse embryo (Stereo-seq) | 44.52 | 35.15 | 31.93 | 88.30 | 193.03 |
| Mouse brain (10x Visium) | 4.45 | 10.51 | 6.10 | 18.23 | 12.53 |
| Mouse kidney (Slide-seq) | 6.22 | 18.76 | 8.49 | 37.38 | 10.62 |

## 💾 Memory Requirements

| Memory (GB) | Tangram | Cytospace | Seurat | cell2 location | CoDi |
|---|---|---|---|---|---|
| Adult mouse brain B5 (Stereo-seq) | 7.072 | 122.68 | 167.169 | 44.942 | 46.178 |
| Whole brain mouse embryo (Stereo-seq) | 8.367 | 124.78 | 142.223 | 55.738 | 46.418 |
| Mouse brain (10x Visium) | 3.272 | 24.11 | 44.415 | 11.482 | 17.108 |
| Mouse kidney (Slide-seq) | 2.812 | 44.04 | 36.168 | 20.896 | 11.969 |

# Marker Gene Retention Analysis

- Compare marker expression: assigned cells vs. reference
- For each predicted cell type, check:
  - Are known markers expressed?
  - Is expression level similar to reference?
- Metrics to compute:
  - Correlation of marker expression profiles
  - Fraction of markers expressed above threshold
  - Log fold change consistency
- Red flag: markers absent in annotated cells

# Marker Gene Retention: Example Analysis

Checking marker gene expression in annotated cells
Calculate the fraction of cells where the gene is detected

```
# Define known markers per cell type
markers = {
    'T_cells': ['CD3D', 'CD3E', 'CD4', 'CD8A'],
    'B_cells': ['CD19', 'MS4A1', 'CD79A'],
    'Macrophages': ['CD68', 'CD163', 'CSF1R']
}
# For each cell type, check marker expression
def check_marker_retention(adata, cell_type_col, markers):
    results = []
    for ct, genes in markers.items():
        ct_cells = adata[adata.obs[cell_type_col] == ct]
        for gene in genes:
            if gene in adata.var_names:
                expr = ct_cells[:, gene].X.mean()
                frac = (ct_cells[:, gene].X > 0).mean()
                results.append({
                    'cell_type': ct, 'marker': gene,
                    'mean_expr': expr, 'frac_expressed': frac
                })
    return pd.DataFrame(results)

retention = check_marker_retention(adata_spatial, 'annotation', markers)
print(retention)
```

# Spatial Distribution Validation

- Check if annotations make biological sense spatially
- Key questions to ask:
  - Do immune cells localize to expected regions?
  - Are epithelial cells at expected boundaries?
  - Do stromal cells show expected patterns?
- Visual inspection of spatial plots essential
- Compare to H&E histology if available
- Use pathologist/domain expert input

# Spatial Validation: Visualization

Visual and statistical spatial validation

```
import scanpy as sc
import squidpy as sq
import matplotlib.pyplot as plt

# Visualize annotations spatially
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# Plot 1: Cell type annotations
sc.pl.spatial(adata, color='annotation', ax=axes[0], show=False)
axes[0].set_title('Cell Type Annotations')

# Plot 2: Known marker for validation
# e.g., CD3D should be high in T cell regions
sc.pl.spatial(adata, color='CD3D', ax=axes[1], show=False)
axes[1].set_title('CD3D Expression (T cell marker)')


# Neighborhood enrichment analysis
sq.gr.spatial_neighbors(adata)
sq.gr.nhood_enrichment(adata, cluster_key='annotation')
sq.pl.nhood_enrichment(adata, cluster_key='annotation')
```

# Ensemble Annotation

Combining Multiple Methods for Robust Results

# Why Use Ensembl Annotation?

- Different methods have different strengths/weaknesses
- No single method is best for all scenarios
- Ensemble approaches provide:
  - Higher accuracy through consensus
  - Uncertainty estimation (method agreement)
  - Robustness to method-specific biases
- Cells with consistent annotation = high confidence
- Cells with disagreement = require review

# Ensemble Annotation Strategies

- Strategy 1: Majority Voting
  - Assign label agreed upon by most methods
  - Simple but effective
- Strategy 2: Weighted Voting
  - Weight methods by prediction confidence
  - Or by validation performance
- Strategy 3: Hierarchical Consensus
  - Agree on broad types first, then refine
  - Reduces label noise at fine resolution
- Strategy 4: Meta-learning (train classifier on method outputs)

# Recommended Ensemble Workflow

- Run multiple methods independently using same reference annotated dataset
  - Tangram, Seurat, CoDi
- Compute ensembl annotation (majority/weighted)
- Calculate confidence (method agreement)
- Validate high-confidence annotations
  - Marker retention, spatial patterns
- Review low-confidence cells manually
- Iterate if needed with adjusted parameters

# Reference dataset

Finding or Creating Good Reference scRNA-seq Datasets

# Finding or Creating Good Reference scRNA-seq Datasets

Important public repositories and reference atlases:

- GEO – Gene Expression Omnibus - Example: How to download
- ArrayExpress – EBI repository
- Human Cell Atlas – comprehensive multi-organ cell atlas
- Single Cell Portal – Broad Institute
- ENA / NCBI SRA – raw scRNA-seq data
- Tabula Muris – mouse whole-organ scRNA atlas (data via HCA Data Explorer, GEO)
- Tabula Sapiens – human multi-organ atlas (~1.1M cells)
- PanglaoDB – integrated mouse & human scRNA data + markers
- Cell type - specific atlases – e.g., brain, immune cell atlases, organ-specific collections

# Selecting Reference scRNA-seq Datasets

- Know your tissue - which cell types are expected (tissue compatibility with your experiment)
- Cell type diversity – cover expected populations and states
- Sequencing depth / UMI counts – ensure adequate sensitivity
- Quality metrics – low % mitochondrial reads, enough genes per cell
- Combine more than one dataset? Yes, leverage multiple datasets to broaden reference scope
- If combining datasets, harmonize annotations and consider integration if needed
- Avoid mixing gene expressions of same cell type from different datasets

# Cell Type Annotation Framework

scripts/cell_annotation_framework.py — multi-tool spatial annotation + ensemble voting

## What it does

- Runs 3 reference-based annotation tools on one spatial dataset
- Each tool labels spots using an scRNA-seq reference
- Saves per-tool CSVs and a merged wide-format table
- Produces an ensemble prediction via majority voting
- Generates spatial-panel figure

## Design goals

- Error handling
- Memory-optimal: backed mode, hard-links
- Docker isolation for Seurat & CoDi containers
- All tool parameters exposed & documented on the CLI

## Three tools

- Tangram: deep-learning mapping
- Seurat: Docker, R anchor label transfer
- CoDi : Docker / contrastive-distance annotation
- Ensemble = majority vote; confidence = agreement %

## Pipeline Flow

1. Parse & Validate
2. Detect Matrix State
3. Marker Genes
4. Run 3 Tools
5. Merge & Ensemble
6. Figure & Output

# Input Validation & Matrix-State Detection

Every input is checked before a single tool runs

## Validation checks

- File exists, readable, .h5ad extension
- Annotation field present in reference .obs
  - warns if missing values or < 2 cell types
- Gene overlap ref ∩ query ≥ 50 %
- var_names made unique (handles scanpy duplicates)
- Docker images checked; auto-pulled if absent

## Expression Matrix state

- Any negative value → log
  - batch-corrected / SCTransform residuals
- All non-zero integers → raw
- Floats, max > 20 → normalized
  - e.g. normalize_total target_sum = 1e4
- Floats, max ≤ 20 → log
  - log1p of 1e4 data caps at ≈ 9; 20 is safe ceiling

## Why raw counts matter

- All 3 tools do their own normalisation internally :
  - Tangram – pp_adatas() normalises
  - Seurat – NormalizeData() in the R script
  - CoDi – container handles preprocessing
- Feeding pre-normalised data → double normalisation
- Framework guarantees raw counts reach every tool

## Tool selection (--tools)

- Accepts: all | tangram | tangram,codi
- Unavailable tools skipped with a warning
- backed='r' mode when only Docker tools selected
  - expression matrix stays on disk - zero RAM
- Pipeline exits only if ALL tools fail

## Raw-count restoration

- If .X ≠ raw AND .raw exists :
  - .X ← .raw.X (in-memory swap)
  - warning logged so user knows
- If .X ≠ raw AND no .raw :
  - warning: results may be unreliable
- CoDi hard-links the on-disk file :
  - in-memory swap does NOT apply to CoDi!

## Marker-gene save / restore pattern

- rank_genes_groups needs log data; Tangram needs raw
- Solution :
  - save .X (sparse copy - cheap)
  - normalise → run rank_genes_groups
  - restore .X from saved copy
- Only runs when Tangram is selected
- Skipped if result already cached in .uns

# The Three Annotation Tools

Each tool has its own data path, Docker strategy and exposed parameters

## Tangram

### Approach

- Runs entirely in-process (Python API)
- Deep-learning spatial cell-to-spot mapping
- No container - fastest startup

### Key steps

- Slice ref & query to pre-computed marker genes
- tg.pp_adatas() - align gene sets
- map_cells_to_space() - train the mapping
- project_cell_annotations() - transfer labels

### Exposed parameters

- --tangram_epochs (default 1 000)
- --tangram_mode cells | genes
- --tangram_density_prior
  - rna_count_based | uniform

## Seurat (Docker)

### Approach

- R-based anchor label transfer (FindTransferAnchors)

### Key steps

- Host writes matrix.mtx.gz + barcodes + features
- ref_mtx & query_mtx mounted read-only (:ro)
- Auto-generated R script executes in container
- FindTransferAnchors → TransferData → CSV

### Docker mounts & params

- /data/ref_mtx/ ← reference (ro)
- /data/query_mtx/ ← query (ro)
- /scripts/ ← R script + output
- --seurat_nfeatures (default 2 000)
- --seurat_dims (default 30)

## CoDi (Docker)

### Approach

- Contrastive-distance cell type annotation
- Hard-links original h5ad → zero-copy input

### Key steps

- Runs contrastive learning encoder and distance calculation in parallel
- Output CSV discovered by scanning the dir

### Docker mounts & params

- /data/ ← hard-linked inputs + output CSV
- --codi_epochs (for contrastive encoder)
- Output columns auto-normalised to standard schema

# Ensemble Prediction, Output & Results

## Merge & ensemble voting

- Per-tool CSVs merged into wide DataFrame :
  - cell_id | tangram | seurat | codi | ensemble
- Majority vote across all successful tools
- Ties broken alphabetically → deterministic
- Single tool → its predictions used directly
- Confidence = winner_votes / n_tools

## Output files

- {sample}_{tool}_predictions.csv (×3)
- {sample}_merged_predictions.csv
  - ensemble is the last column
- {sample}_annotation_summary.json
  - runtime, confidence, gene overlap, metadata
- {sample}_annotation_panels.png
- cell_annotation_framework.log

## Publication-ready figure

- One spatial scatter panel per tool + ensemble
- Unified colour palette (tab10 / tab20)
- Arial font
- No spines, no ticks
- Shared legend centred below panels
- Saved at 200 dpi, white background

### Example command

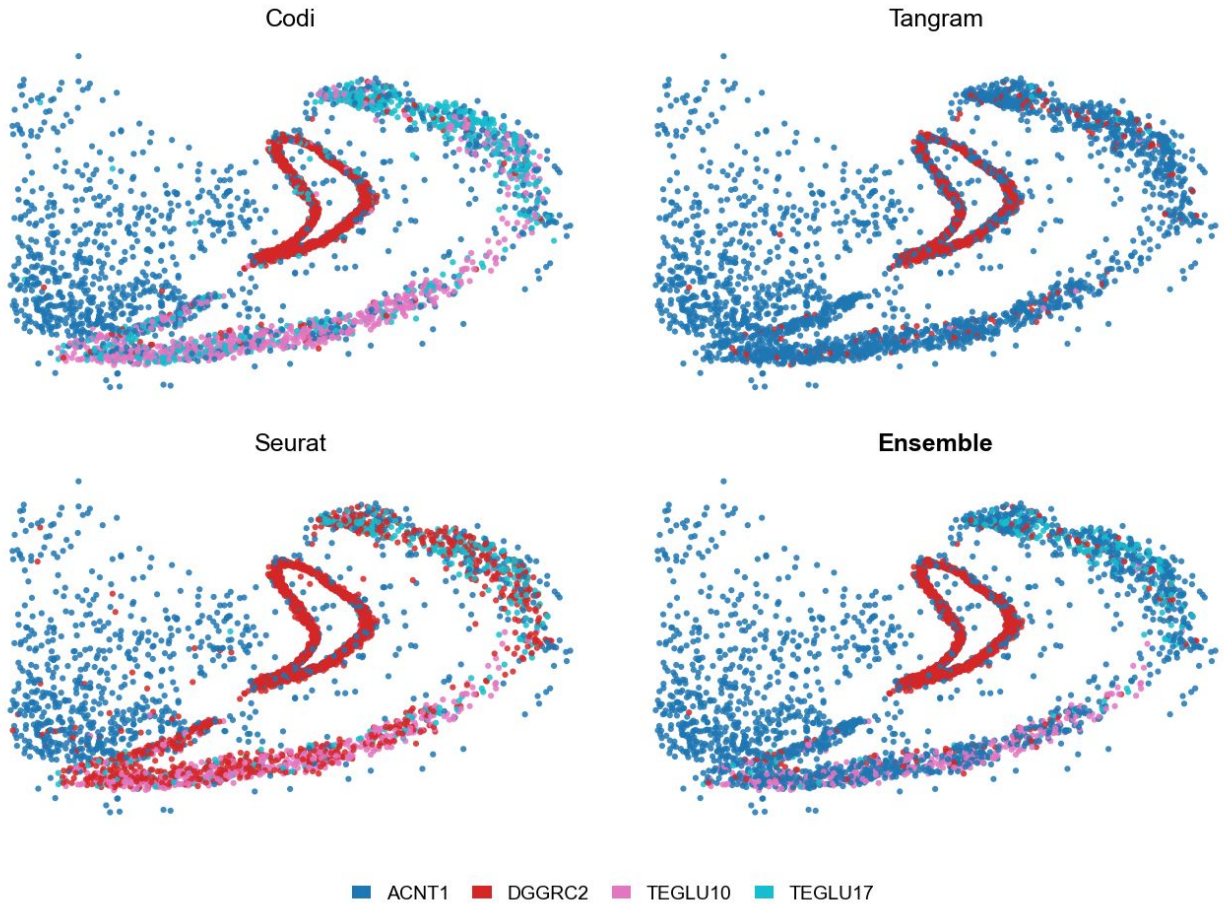
```
python scripts/cell_annotation_framework.py \
  --reference   data/adult_mouse_brain_SC4k.h5ad \
  --query       data/adult_mouse_brain_ST4k.h5ad \
  --annotation-field  cell_subclass \
  --tools  tangram,seurat,codi \
  --tangram_epochs 500  --seurat_dims 20 \
  --output  results  --verbose
```

## Logging & error handling

- Dual logging: console (INFO) + file (DEBUG)
- Per-tool execution time reported
- Total pipeline runtime stored in summary JSON
- Tool failure caught → pipeline continues
- Pipeline exits (code 1) only if ALL tools fail
- Full traceback written to log for debugging
- Keyboard-interrupt (Ctrl-C) exits cleanly

# Cell Type Annotation Framework Outputs

```
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ================================================================================
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | CELL TYPE ANNOTATION FRAMEWORK
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ================================================================================
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | Output directory
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | Sample name: bra
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline |
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ================
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | TOOL AVAILABILIT
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ----------------
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline |   tangram
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline |   seurat
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline |   codi
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline |
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ================
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | TOOL SELECTION
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ----------------
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | Requested tools:
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | Tools to run: co
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | At least one Pyt
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline |
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ================
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | LOADING INPUT DA
2026-02-04 00:44:23 | INFO     | run_annotation_pipeline | ----------------------------------------
```



# Summary

Key Takeaways

# Method Comparison Summary

- Tangram: Best for high-resolution spatial, fast, GPU-friendly
- Seurat: Most established, great docs, no GPU needed
- CoDi: Robust to domain shift, handles imbalance well
- Choice depends on:
  - Spatial technology (resolution, sensitivity)
  - Reference quality and coverage
  - Computational resources available
  - Need for uncertainty quantification

# Best Practices for Cell Type Annotation

- Always validate with marker genes and spatial patterns
- Use high-quality, tissue-matched reference when possible
- Consider ensemble methods for robust annotations
- Report confidence/uncertainty in annotations
- Document all preprocessing steps for reproducibility
- Be cautious with rare cell types (low power)
- Collaborate with domain experts for validation
- Don't over-interpret low-confidence annotations

# Assignments

Create Pull Request (PR) on https://github.com/vladimirkovacevic/st-course:

1. Introduce both CoDi distance and CoDi contrastive - extract annotations and include in ensemble and visualisations
2. Create docker image for Tangram and run it in the same way as CoDi and Seurat
3. Calculate marker genes for reference scRNA, for each cell type and calculate for each ST annotation to decide which one retains markers in the highest percentage (example)

Find annotated reference scRNA for zebrafish embryo and perform 3-methods reference based cell type annotation

# Resources and Documentation

- Tangram: github.com/broadinstitute/Tangram
- Seurat: satijalab.org/seurat/
- CoDi: github.com/Stomics/CoDi
- CellMarker: bio-bigdata.hrbmu.edu.cn/CellMarker/
- PanglaoDB: panglaodb.se/

# Questions?

Cell Type Annotation in Spatial Transcriptomics

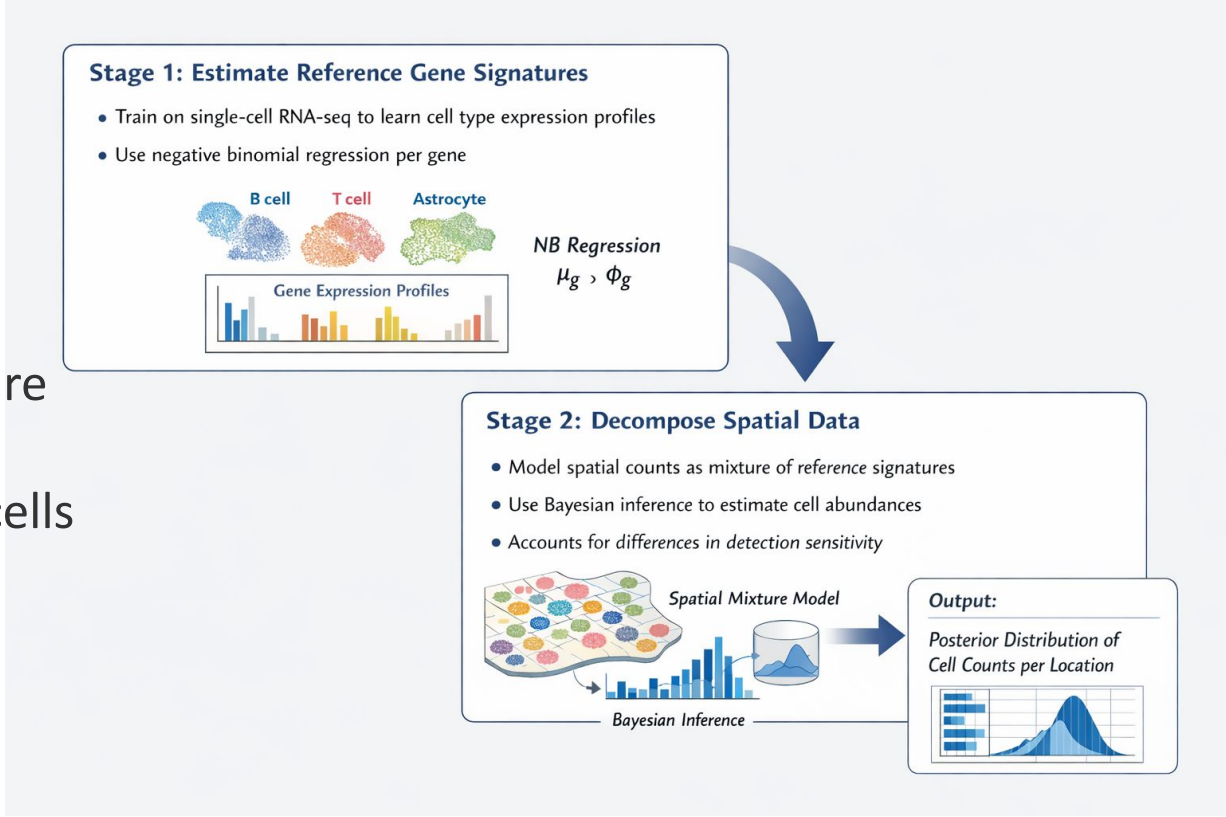
# cell2location

Bayesian model for mapping cell types to spatial locations

- Developed by the Teichmann lab (Wellcome Sanger Institute, Nature Biotechnology, 2022)
- Estimates absolute cell abundance at each location
- Accounts for platform-specific technical effects
- Uses negative binomial model with learned priors

# cell2location: Under the Hood

- Stage 1 – Estimate reference gene signatures
  - Train on single-cell RNA-seq to learn typical expression profiles for each cell type
  - Use negative binomial regression per gene to capture variability
- Stage 2 – Decompose spatial data
  - Model spatial transcriptomics counts as a mixture of reference signatures
  - Use Bayesian inference to estimate how many cells of each type are in each spot
  - Accounts for differences in detection sensitivity between datasets
- Output:
  Posterior distribution of cell counts per location (gives uncertainty and mean estimates)



# cell2location: Key Benefits

- Estimates absolute cell numbers, not just proportions
- Full uncertainty quantification (Bayesian posteriors)
- Handles platform effects between scRNA-seq and spatial
- Can detect rare cell types with low abundance
- Built on scvi-tools framework (robust, well-tested)
- Works well even with partial reference coverage
- Integrates smoothly with scanpy/anndata ecosystem

# cell2location: Running the Analysis

Two-stage cell2location workflow

```
import cell2location
import scanpy as sc

# Load data
adata_sc = sc.read_h5ad('scrna_reference.h5ad')
adata_vis = sc.read_h5ad('visium_data.h5ad')

# Stage 1: Estimate reference signatures
cell2location.models.RegressionModel.setup_anndata(adata_sc,
    labels_key='cell_type')
ref_model = cell2location.models.RegressionModel(adata_sc)
ref_model.train(max_epochs=250)

# Export estimated signatures
adata_sc = ref_model.export_posterior(adata_sc)
inf_aver = adata_sc.varm['means_per_cluster_mu_fg']

# Stage 2: Spatial mapping
cell2location.models.Cell2location.setup_anndata(adata_vis)
mod = cell2location.models.Cell2location(adata_vis,
    cell_state_df=inf_aver, N_cells_per_location=30)
mod.train(max_epochs=30000)

# Export results
```