Biomni Agent Conversation History

Human Prompt

*Given these single cell RNA-seq data /data/lep/BaisBench/Task2_data/h5ad_file/task2 - Wang et al. (2020) Journal of Experimental Medicine - Ileum.h5ad, /data/lep/BaisBench/Task2_data/h5ad_file/task2 - Wang et al. (2020) Journal of Experimental Medicine - Rectum.h5ad, /data/lep/BaisBench/Task2_data/h5ad_file/task2 - Wang et al. (2020) Journal of Experimental Medicine - Colon.h5ad, and the background information: In this research, I aimed to investigate the nutrient absorption functions of the human intestine by analyzing intestinal epithelial cells from the human ileum, colon, and rectum. The human intestine plays a crucial role in the digestion and absorption of nutrients, but the specific cellular mechanisms, especially differences across intestinal regions, remain largely unclear. Using single-cell RNA sequencing (scRNA-seq), I profiled 14,537 epithelial cells from human ileum, colon, and rectum obtained from six donors (two per region) to better understand nutrient absorption functions in the small and large intestine and to identify potential new marker genes for various cell types involved in these processes.

The sequencing was done using the 10X Genomics platform, producing high-quality transcriptomic data, which allowed me to perform detailed analyses of gene expression in different regions of the human gut. The key findings from these analyses help define the roles of different intestinal cell types in nutrient absorption, including Paneth-like cells (PLCs) in the large intestine, and new markers for transient-amplifying (TA) cells and goblet cells. , analysis the data to answer the following questions: Q1: Which cell type was most abundant in the human ileum according to the single-cell transcriptomic data? A) Goblet cells B) Enterocytes C) Paneth cells D) Stem cells Q2: Which new marker was identified for transient-amplifying (TA) cells in the human intestine? A) NUSAP1 B) CDX2 C) SOX9 D) LGR5 Q3: Which gene was identified as being enriched in goblet cells in all three regions of the human intestine (ileum, colon, and rectum)? A) ITLN1 B) TFF1 C) MYC D) APOA1 Q4: The data identified Paneth-like cells (PLCs) in the large intestine. What gene can NOT be used to identify these cells? A) LYZ B) CA4 C) APOA4 D) SPIB . Provide the letter options as answers.*

Step 1

I'll analyze the single-cell RNA-seq data from the human intestine (ileum, colon, and rectum) to answer these questions about cell types, marker genes, and abundance patterns.

Plan

Plan

- *[ ]* Load and examine the structure of all three datasets (ileum, colon, rectum)
- *[ ]* Perform quality control and preprocessing on the datasets
- *[ ]* Integrate the datasets and perform clustering analysis
- *[ ]* Annotate cell types using known markers and automated methods
- *[ ]* Analyze cell type abundance across regions to answer Q1
- *[ ]* Identify marker genes for transient-amplifying (TA) cells to answer Q2
- *[ ]* Find genes enriched in goblet cells across all regions to answer Q3
- *[ ]* Analyze Paneth-like cell markers in large intestine to answer Q4
- *[ ]* Provide final answers

Let me start by examining the data structure:

Code Execution

```
import scanpy as sc
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from pathlib import Path
import warnings
warnings.filterwarnings('ignore')

# Set up scanpy settings
sc.settings.verbosity = 3
sc.settings.set_figure_params(dpi=80, facecolor='white')

# Load the three datasets
data_dir = "/data/lep/BaisBench/Task2_data/h5ad_file/"
ileum_file = "task2 - Wang et al. (2020) Journal of Experimental Medicine - Ileum.h5ad"
colon_file = "task2 - Wang et al. (2020) Journal of Experimental Medicine - Colon.h5ad"
rectum_file = "task2 - Wang et al. (2020) Journal of Experimental Medicine - Rectum.h5ad"

print("Loading datasets...")
adata_ileum = sc.read_h5ad(data_dir + ileum_file)
adata_colon = sc.read_h5ad(data_dir + colon_file)
adata_rectum = sc.read_h5ad(data_dir + rectum_file)

print(f"Ileum dataset: {adata_ileum.shape}")
print(f"Colon dataset: {adata_colon.shape}")
print(f"Rectum dataset: {adata_rectum.shape}")

# Examine the structure of each dataset
print("\n=== Ileum Dataset Structure ===")
print(f"Observations (cells): {adata_ileum.n_obs}")
print(f"Variables (genes): {adata_ileum.n_vars}")
print(f"Observation keys: {list(adata_ileum.obs.columns)}")
print(f"Variable keys: {list(adata_ileum.var.columns)}")

print("\n=== Colon Dataset Structure ===")
print(f"Observations (cells): {adata_colon.n_obs}")
print(f"Variables (genes): {adata_colon.n_vars}")
print(f"Observation keys: {list(adata_colon.obs.columns)}")

print("\n=== Rectum Dataset Structure ===")
print(f"Observations (cells): {adata_rectum.n_obs}")
print(f"Variables (genes): {adata_rectum.n_vars}")
print(f"Observation keys: {list(adata_rectum.obs.columns)}")
```

Tools Used: Python REPL

Observation

```
Loading datasets...
Ileum dataset: (5980, 16795)
Colon dataset: (4329, 16986)
Rectum dataset: (3797, 17465)

=== Ileum Dataset Structure ===
Observations (cells): 5980
Variables (genes): 16795
Observation keys: ['CellType', 'n_counts', 'log1p_n_counts', 'n_genes', 'log1p_n_genes', 'percent_mito',
'percent_ribo', 'percent_hb', 'percent_top50', 'assay_ontology_term_id', 'cell_type_ontology_term_id',
'development_stage_ontology_term_id', 'disease_ontology_term_id',
'self_reported_ethnicity_ontology_term_id', 'is_primary_data', 'organism_ontology_term_id',
'sex_ontology_term_id', 'tissue_ontology_term_id', 'donor_id', 'suspension_type', 'tissue_type', 'assay',
'disease', 'organism', 'sex', 'tissue', 'self_reported_ethnicity', 'development_stage',
'observation_joinid', 'cellxgene_cell_type']
Variable keys: ['mito', 'ribo', 'hb', 'n_counts', 'n_cells', 'n_genes', 'highly_variable', 'means',
'dispersions', 'dispersions_norm', 'gene_symbols', 'feature_is_filtered', 'feature_name',
'feature_reference', 'feature_biotype', 'feature_length', 'feature_type', 'ensembl_id', 'gene_symbol']

=== Colon Dataset Structure ===
Observations (cells): 4329
Variables (genes): 16986
Observation keys: ['CellType', 'n_counts', 'log1p_n_counts', 'n_genes', 'log1p_n_genes', 'percent_mito',
'percent_ribo', 'percent_hb', 'percent_top50', 'assay_ontology_term_id', 'cell_type_ontology_term_id',
'development_stage_ontology_term_id', 'disease_ontology_term_id',
'self_reported_ethnicity_ontology_term_id', 'is_primary_data', 'organism_ontology_term_id',
'sex_ontology_term_id', 'tissue_ontology_term_id', 'donor_id', 'suspension_type', 'tissue_type', 'assay',
'disease', 'organism', 'sex', 'tissue', 'self_reported_ethnicity', 'development_stage',
'observation_joinid', 'cellxgene_cell_type']

=== Rectum Dataset Structure ===
Observations (cells): 3797
Variables (genes): 17465
Observation keys: ['CellType', 'n_counts', 'log1p_n_counts', 'n_genes', 'log1p_n_genes', 'percent_mito',
'percent_ribo', 'percent_hb', 'percent_top50', 'assay_ontology_term_id', 'cell_type_ontology_term_id',
'development_stage_ontology_term_id', 'disease_ontology_term_id',
'self_reported_ethnicity_ontology_term_id', 'is_primary_data', 'organism_ontology_term_id',
'sex_ontology_term_id', 'tissue_ontology_term_id', 'donor_id', 'suspension_type', 'tissue_type', 'assay',
'disease', 'organism', 'sex', 'tissue', 'self_reported_ethnicity', 'development_stage',
'observation_joinid', 'cellxgene_cell_type']
```

Step 2

Now let me examine the cell type annotations that are already present:

Code Execution

```
# Check existing cell type annotations
print("=== Cell Type Annotations ===")
print("\nIleum cell types:")
print(adata_ileum.obs['CellType'].value_counts())

print("\nColon cell types:")
print(adata_colon.obs['CellType'].value_counts())

print("\nRectum cell types:")
print(adata_rectum.obs['CellType'].value_counts())

# Also check cellxgene_cell_type if available
print("\n=== CellxGene Cell Type Annotations ===")
print("\nIleum cellxgene cell types:")
print(adata_ileum.obs['cellxgene_cell_type'].value_counts())

print("\nColon cellxgene cell types:")
print(adata_colon.obs['cellxgene_cell_type'].value_counts())

print("\nRectum cellxgene cell types:")
print(adata_rectum.obs['cellxgene_cell_type'].value_counts())
```

Tools Used: Python REPL

Observation

```
=== Cell Type Annotations ===

Ileum cell types:
CellType
Enterocyte        4243
Progenitor         782
Goblet             303
TA                 291
Stem Cell          229
Paneth-like         94
Enteriendocrine     38
Name: count, dtype: int64

Colon cell types:
CellType
Progenitor        1784
Enterocyte         870
Goblet             765
TA                 417
Paneth-like        349
Stem Cell          131
Enteriendocrine     13
Name: count, dtype: int64

Rectum cell types:
CellType
Progenitor        1435
Goblet             899
Enterocyte         584
Paneth-like        299
Stem Cell          258
TA                 256
Enteriendocrine     66
Name: count, dtype: int64

=== CellxGene Cell Type Annotations ===

Ileum cellxgene cell types:
cellxgene_cell_type
colonocyte                          4243
progenitor cell                      782
colon goblet cell                    303
transit amplifying cell of colon     291
stem cell                            229
paneth cell of colon                  94
enteroendocrine cell of colon         38
Name: count, dtype: int64

Colon cellxgene cell types:
cellxgene_cell_type
progenitor cell                     1784
colonocyte                           870
colon goblet cell                    765
transit amplifying cell of colon     417
paneth cell of colon                 349
stem cell                            131
```

```
enteroendocrine cell of colon        13
Name: count, dtype: int64

Rectum cellxgene cell types:
cellxgene_cell_type
progenitor cell                    1435
colon goblet cell                   899
colonocyte                          584
paneth cell of colon                299
stem cell                           258
transit amplifying cell of colon    256
enteroendocrine cell of colon        66
Name: count, dtype: int64
```

Step 3

Perfect! The datasets already have cell type annotations. Let me update my plan and continue with the analysis:

Plan

- *[x]* Load and examine the structure of all three datasets (ileum, colon, rectum)
- *[ ]* Analyze cell type abundance across regions to answer Q1
- *[ ]* Examine marker genes for different cell types to answer Q2, Q3, and Q4
- *[ ]* Integrate datasets if needed for comparative analysis
- *[ ]* Provide final answers

Code Execution

```
# Q1: Which cell type was most abundant in the human ileum?
print("=== Q1: Cell type abundance in human ileum ===")
ileum_counts = adata_ileum.obs['CellType'].value_counts()
print("Cell type counts in ileum:")
print(ileum_counts)
print(f"\nMost abundant cell type in ileum: {ileum_counts.index[0]} ({ileum_counts.iloc[0]} cells)")

# Calculate percentages
ileum_percentages = (ileum_counts / ileum_counts.sum() * 100).round(2)
print(f"\nCell type percentages in ileum:")
for cell_type, percentage in ileum_percentages.items():
    print(f"{cell_type}: {percentage}%")

# Check if this matches the options
options_q1 = ['Goblet cells', 'Enterocytes', 'Paneth cells', 'Stem cells']
print(f"\nOptions: {options_q1}")
print(f"Answer: The most abundant cell type in ileum is 'Enterocyte' which corresponds to option B)
Enterocytes")
```

Tools Used: Python REPL

Observation

```
=== Q1: Cell type abundance in human ileum ===
Cell type counts in ileum:
CellType
```

```
Enterocyte        4243
Progenitor         782
Goblet             303
TA                 291
Stem Cell          229
Paneth-like         94
Enteriendocrine     38
Name: count, dtype: int64

Most abundant cell type in ileum: Enterocyte (4243 cells)

Cell type percentages in ileum:
Enterocyte: 70.95%
Progenitor: 13.08%
Goblet: 5.07%
TA: 4.87%
Stem Cell: 3.83%
Paneth-like: 1.57%
Enteriendocrine: 0.64%

Options: ['Goblet cells', 'Enterocytes', 'Paneth cells', 'Stem cells']
Answer: The most abundant cell type in ileum is 'Enterocyte' which corresponds to option B) Enterocytes
```

Step 4

Great! Now let me combine all datasets and analyze marker genes to answer the remaining questions:

Code Execution

```
# Add tissue information to each dataset
adata_ileum.obs['tissue_region'] = 'Ileum'
adata_colon.obs['tissue_region'] = 'Colon'
adata_rectum.obs['tissue_region'] = 'Rectum'

# Find common genes across all datasets for integration
common_genes = set(adata_ileum.var.index) & set(adata_colon.var.index) & set(adata_rectum.var.index)
print(f"Common genes across all datasets: {len(common_genes)}")

# Subset to common genes
adata_ileum_common = adata_ileum[:, list(common_genes)].copy()
adata_colon_common = adata_colon[:, list(common_genes)].copy()
adata_rectum_common = adata_rectum[:, list(common_genes)].copy()

# Concatenate datasets
adata_combined = adata_ileum_common.concatenate(adata_colon_common, adata_rectum_common,
                                                batch_categories=['Ileum', 'Colon', 'Rectum'])

print(f"Combined dataset shape: {adata_combined.shape}")
print(f"Total cells: {adata_combined.n_obs}")

# Check cell type distribution in combined dataset
print("\n=== Combined dataset cell type distribution ===")
combined_counts = adata_combined.obs['CellType'].value_counts()
print(combined_counts)
```

Tools Used: Python REPL

Observation

```
Common genes across all datasets: 15422
Combined dataset shape: (14106, 15422)
Total cells: 14106

=== Combined dataset cell type distribution ===
CellType
Enterocyte         5697
Progenitor         4001
Goblet             1967
TA                  964
Paneth-like         742
Stem Cell           618
Enteriendocrine     117
Name: count, dtype: int64
```

Step 5

Now let me analyze marker genes for each cell type to answer the remaining questions:

Code Execution

```
# Normalize and log transform for marker gene analysis
sc.pp.normalize_total(adata_combined, target_sum=1e4)
sc.pp.log1p(adata_combined)

# Calculate marker genes for each cell type
sc.tl.rank_genes_groups(adata_combined, 'CellType', method='wilcoxon', n_genes=50)

# Get the results
marker_results = adata_combined.uns['rank_genes_groups']

# Function to get top markers for a cell type
def get_top_markers(cell_type, n_genes=20):
    cell_types = marker_results['names'].dtype.names
    if cell_type in cell_types:
        idx = list(cell_types).index(cell_type)
        markers = [marker_results['names'][i][idx] for i in range(n_genes)]
        scores = [marker_results['scores'][i][idx] for i in range(n_genes)]
        return list(zip(markers, scores))
    return []

# Analyze TA cell markers for Q2
print("=== Q2: Transient-Amplifying (TA) cell markers ===")
ta_markers = get_top_markers('TA', 20)
print("Top 20 markers for TA cells:")
for i, (gene, score) in enumerate(ta_markers):
    print(f"{i+1:2d}. {gene:12s} (score: {score:.3f})")

# Check the specific options
options_q2 = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
print(f"\nChecking Q2 options in TA markers: {options_q2}")
ta_marker_genes = [gene for gene, score in ta_markers]
for option in options_q2:
    if option in ta_marker_genes:
        rank = ta_marker_genes.index(option) + 1
        score = ta_markers[ta_marker_genes.index(option)][1]
        print(f"  {option}: Rank {rank}, Score {score:.3f}")
    else:
        print(f"  {option}: Not in top 20 TA markers")
```

Tools Used: Python REPL

Observation

```
=== Q2: Transient-Amplifying (TA) cell markers ===
Top 20 markers for TA cells:
 1. ENSG00000164104 (score: 49.287)
 2. ENSG00000164032 (score: 47.658)
 3. ENSG00000117632 (score: 47.558)
 4. ENSG00000123416 (score: 47.469)
 5. ENSG00000196230 (score: 45.994)
 6. ENSG00000189403 (score: 45.752)
 7. ENSG00000198830 (score: 45.654)
 8. ENSG00000187514 (score: 42.942)
 9. ENSG00000166803 (score: 42.901)
10. ENSG00000167900 (score: 42.107)
11. ENSG00000203760 (score: 41.450)
12. ENSG00000099901 (score: 41.333)
13. ENSG00000124795 (score: 40.707)
14. ENSG00000176890 (score: 40.483)
15. ENSG00000182054 (score: 40.478)
16. ENSG00000069275 (score: 40.415)
17. ENSG00000132341 (score: 40.364)
18. ENSG00000106355 (score: 39.436)
19. ENSG00000173207 (score: 39.405)
20. ENSG00000125835 (score: 39.348)

Checking Q2 options in TA markers: ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
  NUSAP1: Not in top 20 TA markers
  CDX2: Not in top 20 TA markers
  SOX9: Not in top 20 TA markers
  LGR5: Not in top 20 TA markers
```

Step 6

I need to check if the gene symbols are properly mapped. Let me examine the gene annotation:

Code Execution

```
# Check gene symbol mapping
print("Checking gene symbol mapping...")
print(f"Gene symbols available: {'gene_symbols' in adata_combined.var.columns}")
print(f"Gene symbol column: {'gene_symbol' in adata_combined.var.columns}")

if 'gene_symbols' in adata_combined.var.columns:
    print("\nSample gene symbols:")
    print(adata_combined.var['gene_symbols'].head(10))

    # Map Ensembl IDs to gene symbols
    gene_id_to_symbol = dict(zip(adata_combined.var.index, adata_combined.var['gene_symbols']))

    # Convert TA markers to gene symbols
    print("\n=== TA cell markers with gene symbols ===")
    ta_markers_with_symbols = []
    for i, (ensembl_id, score) in enumerate(ta_markers):
        symbol = gene_id_to_symbol.get(ensembl_id, ensembl_id)
        ta_markers_with_symbols.append((symbol, score))
        print(f"{i+1:2d}. {symbol:12s} (score: {score:.3f})")

    # Check Q2 options again
    print(f"\nChecking Q2 options in TA markers: {options_q2}")
    ta_symbol_genes = [gene for gene, score in ta_markers_with_symbols]
    for option in options_q2:
        if option in ta_symbol_genes:
            rank = ta_symbol_genes.index(option) + 1
            score = ta_markers_with_symbols[ta_symbol_genes.index(option)][1]
            print(f"  {option}: Rank {rank}, Score {score:.3f}")
        else:
            print(f"  {option}: Not in top 20 TA markers")

elif 'gene_symbol' in adata_combined.var.columns:
    gene_id_to_symbol = dict(zip(adata_combined.var.index, adata_combined.var['gene_symbol']))
    # Similar processing...
```

Tools Used: Python REPL

Observation

```
Checking gene symbol mapping...
Gene symbols available: False
Gene symbol column: True
```

Step 7

Code Execution

```
# Use gene_symbol column instead
print("Using gene_symbol column...")
print("\nSample gene symbols:")
print(adata_combined.var['gene_symbol'].head(10))

# Map Ensembl IDs to gene symbols
gene_id_to_symbol = dict(zip(adata_combined.var.index, adata_combined.var['gene_symbol']))

# Convert TA markers to gene symbols
print("\n=== TA cell markers with gene symbols ===")
ta_markers_with_symbols = []
for i, (ensembl_id, score) in enumerate(ta_markers):
    symbol = gene_id_to_symbol.get(ensembl_id, ensembl_id)
    ta_markers_with_symbols.append((symbol, score))
    print(f"{i+1:2d}. {symbol:12s} (score: {score:.3f})")

# Check Q2 options again
print(f"\nChecking Q2 options in TA markers: {options_q2}")
ta_symbol_genes = [gene for gene, score in ta_markers_with_symbols]
for option in options_q2:
    if option in ta_symbol_genes:
        rank = ta_symbol_genes.index(option) + 1
        score = ta_markers_with_symbols[ta_symbol_genes.index(option)][1]
        print(f"  {option}: Rank {rank}, Score {score:.3f}")
    else:
        print(f"  {option}: Not in top 20 TA markers")

# Let's also check if these genes are expressed at all
print(f"\nChecking if Q2 options are present in the dataset:")
for option in options_q2:
    if option in adata_combined.var['gene_symbol'].values:
        ensembl_id = adata_combined.var[adata_combined.var['gene_symbol'] == option].index[0]
        print(f"  {option}: Present (Ensembl ID: {ensembl_id})")
    else:
        print(f"  {option}: Not found in dataset")
```

Tools Used: Python REPL

Observation

```
Using gene_symbol column...

Sample gene symbols:
DIP2B            DIP2B
SLC4A1          SLC4A1
TMEM187        TMEM187
USF1              USF1
PKD2L2-DT    PKD2L2-DT
FAM89B          FAM89B
MMD                MMD
MIER1            MIER1
PTPN14          PTPN14
CRTC3-AS1    CRTC3-AS1
Name: gene_symbol, dtype: object

=== TA cell markers with gene symbols ===
 1. ENSG00000164104 (score: 49.287)
 2. ENSG00000164032 (score: 47.658)
 3. ENSG00000117632 (score: 47.558)
 4. ENSG00000123416 (score: 47.469)
 5. ENSG00000196230 (score: 45.994)
 6. ENSG00000189403 (score: 45.752)
 7. ENSG00000198830 (score: 45.654)
 8. ENSG00000187514 (score: 42.942)
 9. ENSG00000166803 (score: 42.901)
10. ENSG00000167900 (score: 42.107)
11. ENSG00000203760 (score: 41.450)
12. ENSG00000099901 (score: 41.333)
13. ENSG00000124795 (score: 40.707)
14. ENSG00000176890 (score: 40.483)
15. ENSG00000182054 (score: 40.478)
16. ENSG00000069275 (score: 40.415)
17. ENSG00000132341 (score: 40.364)
18. ENSG00000106355 (score: 39.436)
19. ENSG00000173207 (score: 39.405)
20. ENSG00000125835 (score: 39.348)

Checking Q2 options in TA markers: ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
  NUSAP1: Not in top 20 TA markers
  CDX2: Not in top 20 TA markers
  SOX9: Not in top 20 TA markers
  LGR5: Not in top 20 TA markers

Checking if Q2 options are present in the dataset:
  NUSAP1: Present (Ensembl ID: NUSAP1)
  CDX2: Present (Ensembl ID: CDX2)
  SOX9: Present (Ensembl ID: SOX9)
  LGR5: Present (Ensembl ID: LGR5)
```

Step 8

There seems to be an issue with the gene symbol mapping. Let me fix this and check the actual gene symbols:

Code Execution

```
# The issue is that the gene symbols are not being mapped properly
# Let's check the actual mapping
print("Checking gene symbol mapping issue...")

# Check a few examples
sample_ids = list(adata_combined.var.index[:10])
for ensembl_id in sample_ids:
    symbol = adata_combined.var.loc[ensembl_id, 'gene_symbol']
    print(f"{ensembl_id} -> {symbol}")

# It looks like the index might already be gene symbols or there's a mapping issue
# Let's check if the index contains gene symbols
print(f"\nFirst 10 gene indices: {list(adata_combined.var.index[:10])}")

# Let's recalculate markers using gene symbols as index if available
# First, let's check if we can find our target genes directly
target_genes = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
print(f"\nLooking for target genes in var index:")
for gene in target_genes:
    if gene in adata_combined.var.index:
        print(f"  {gene}: Found in var.index")
    elif gene in adata_combined.var['gene_symbol'].values:
        matching_rows = adata_combined.var[adata_combined.var['gene_symbol'] == gene]
        print(f"  {gene}: Found in gene_symbol column, index: {matching_rows.index.tolist()}")
    else:
        print(f"  {gene}: Not found")
```

Tools Used: Python REPL

Observation

```
Checking gene symbol mapping issue...
DIP2B -> DIP2B
SLC4A1 -> SLC4A1
TMEM187 -> TMEM187
USF1 -> USF1
PKD2L2-DT -> PKD2L2-DT
FAM89B -> FAM89B
MMD -> MMD
MIER1 -> MIER1
PTPN14 -> PTPN14
CRTC3-AS1 -> CRTC3-AS1

First 10 gene indices: ['DIP2B', 'SLC4A1', 'TMEM187', 'USF1', 'PKD2L2-DT', 'FAM89B', 'MMD', 'MIER1',
'PTPN14', 'CRTC3-AS1']
```

```
Looking for target genes in var index:
  NUSAP1: Found in var.index
  CDX2: Found in var.index
  SOX9: Found in var.index
  LGR5: Found in var.index
```

Step 9

Great! The gene symbols are already in the index. Let me recalculate the marker genes properly:

Code Execution

```
# Now let's get the TA markers with proper gene symbols
print("=== Q2: TA cell markers (corrected) ===")

# Get marker genes results
marker_results = adata_combined.uns['rank_genes_groups']
cell_types = marker_results['names'].dtype.names

# Get TA markers
ta_idx = list(cell_types).index('TA')
ta_markers_genes = [marker_results['names'][i][ta_idx] for i in range(50)]  # Get top 50
ta_markers_scores = [marker_results['scores'][i][ta_idx] for i in range(50)]

print("Top 20 TA cell markers:")
for i in range(20):
    print(f"{i+1:2d}. {ta_markers_genes[i]:12s} (score: {ta_markers_scores[i]:.3f})")

# Check Q2 options
options_q2 = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
print(f"\nChecking Q2 options in TA markers: {options_q2}")
for option in options_q2:
    if option in ta_markers_genes:
        rank = ta_markers_genes.index(option) + 1
        score_idx = ta_markers_genes.index(option)
        score = ta_markers_scores[score_idx]
        print(f"  {option}: Rank {rank}, Score {score:.3f}")
    else:
        print(f"  {option}: Not in top 50 TA markers")

# Let's also check expression levels directly
print(f"\nExpression levels of Q2 options in TA cells:")
ta_cells = adata_combined[adata_combined.obs['CellType'] == 'TA']
for gene in options_q2:
    if gene in adata_combined.var.index:
        mean_expr = ta_cells[:, gene].X.mean()
        print(f"  {gene}: Mean expression in TA cells: {mean_expr:.3f}")
```

Tools Used: Python REPL

Observation

```
=== Q2: TA cell markers (corrected) ===
Top 20 TA cell markers:
 1. ENSG00000164104 (score: 49.287)
 2. ENSG00000164032 (score: 47.658)
 3. ENSG00000117632 (score: 47.558)
 4. ENSG00000123416 (score: 47.469)
 5. ENSG00000196230 (score: 45.994)
 6. ENSG00000189403 (score: 45.752)
 7. ENSG00000198830 (score: 45.654)
 8. ENSG00000187514 (score: 42.942)
 9. ENSG00000166803 (score: 42.901)
10. ENSG00000167900 (score: 42.107)
11. ENSG00000203760 (score: 41.450)
12. ENSG00000099901 (score: 41.333)
13. ENSG00000124795 (score: 40.707)
14. ENSG00000176890 (score: 40.483)
15. ENSG00000182054 (score: 40.478)
16. ENSG00000069275 (score: 40.415)
17. ENSG00000132341 (score: 40.364)
18. ENSG00000106355 (score: 39.436)
19. ENSG00000173207 (score: 39.405)
20. ENSG00000125835 (score: 39.348)

Checking Q2 options in TA markers: ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
  NUSAP1: Not in top 50 TA markers
  CDX2: Not in top 50 TA markers
  SOX9: Not in top 50 TA markers
  LGR5: Not in top 50 TA markers

Expression levels of Q2 options in TA cells:
  NUSAP1: Mean expression in TA cells: 0.789
  CDX2: Mean expression in TA cells: 1.242
  SOX9: Mean expression in TA cells: 0.655
  LGR5: Mean expression in TA cells: 0.112
```

Step 10

There seems to be an issue with the marker gene results still showing Ensembl IDs. Let me recalculate the markers properly:

Code Execution

```
# Let's recalculate markers from scratch to ensure proper gene symbol usage
print("Recalculating marker genes from scratch...")

# Make a fresh copy and ensure we're working with the right data
adata_fresh = adata_combined.copy()

# Recalculate markers
sc.tl.rank_genes_groups(adata_fresh, 'CellType', method='wilcoxon', n_genes=50)

# Extract results properly
results_df = sc.get.rank_genes_groups_df(adata_fresh, group='TA')
print("=== Q2: Top TA cell markers ===")
print(results_df.head(20))

# Check our target genes
options_q2 = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
print(f"\nChecking Q2 options in TA markers:")
for option in options_q2:
    if option in results_df['names'].values:
        row = results_df[results_df['names'] == option].iloc[0]
        print(f"  {option}: Rank {results_df[results_df['names'] == option].index[0] + 1}, Score
{row['scores']:.3f}, Adj p-val {row['pvals_adj']:.2e}")
```

```
    else:
        # Check if it's in the top 50 at all
        full_results = sc.get.rank_genes_groups_df(adata_fresh, group='TA')
        if option in full_results['names'].values:
            row = full_results[full_results['names'] == option].iloc[0]
            rank = full_results[full_results['names'] == option].index[0] + 1
            print(f"  {option}: Rank {rank}, Score {row['scores']:.3f}, Adj p-val {row['pvals_adj']:.2e}")
        else:
            print(f"  {option}: Not in top markers")
```

Tools Used: Python REPL

Observation

```
Recalculating marker genes from scratch...
=== Q2: Top TA cell markers ===
              names     scores  logfoldchanges  pvals  pvals_adj
0   ENSG00000164104  49.287342       22.807138    0.0        0.0
1   ENSG00000164032  47.658443       31.773426    0.0        0.0
2   ENSG00000117632  47.557880       14.543709    0.0        0.0
3   ENSG00000123416  47.468521       29.164425    0.0        0.0
4   ENSG00000196230  45.993614       17.044569    0.0        0.0
5   ENSG00000189403  45.751892       32.029449    0.0        0.0
6   ENSG00000198830  45.653923       27.071512    0.0        0.0
7   ENSG00000187514  42.941772      114.271042    0.0        0.0
8   ENSG00000166803  42.901146       11.213116    0.0        0.0
9   ENSG00000167900  42.107391        8.832512    0.0        0.0
10  ENSG00000203760  41.449654        6.627507    0.0        0.0
11  ENSG00000099901  41.332695       10.953118    0.0        0.0
12  ENSG00000124795  40.707226        7.722132    0.0        0.0
13  ENSG00000176890  40.483170        7.409595    0.0        0.0
14  ENSG00000182054  40.477501        9.233834    0.0        0.0
15  ENSG00000069275  40.414684        8.030207    0.0        0.0
16  ENSG00000132341  40.364231       11.444024    0.0        0.0
17  ENSG00000106355  39.435669        6.972369    0.0        0.0
18  ENSG00000173207  39.404640        6.924662    0.0        0.0
19  ENSG00000125835  39.348309        7.799319    0.0        0.0

Checking Q2 options in TA markers:
  NUSAP1: Not in top markers
  CDX2: Not in top markers
  SOX9: Not in top markers
  LGR5: Not in top markers
```

Step 11

I need to fix the gene symbol mapping issue. Let me check the original datasets to see how the gene names are stored:

Code Execution

```
# Let's check the original datasets to understand the gene naming
print("Checking original dataset gene naming...")
print(f"Ileum var columns: {list(adata_ileum.var.columns)}")
print(f"First 10 ileum gene indices: {list(adata_ileum.var.index[:10])}")
print(f"First 10 ileum gene symbols: {list(adata_ileum.var['gene_symbol'][:10])}")
```

```
# Check if there's a mismatch between index and gene_symbol
print(f"\nChecking for mismatches...")
mismatches = 0
for i in range(min(100, len(adata_ileum.var))):
    idx = adata_ileum.var.index[i]
    symbol = adata_ileum.var['gene_symbol'].iloc[i]
    if idx != symbol:
        mismatches += 1
        if mismatches <= 5:  # Show first 5 mismatches
            print(f"  {idx} != {symbol}")

print(f"Total mismatches in first 100 genes: {mismatches}")

# Let's try to create a proper mapping and redo the analysis
# First, let's see if we can find NUSAP1 in the original data
target_genes = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
print(f"\nLooking for target genes in original ileum data:")
for gene in target_genes:
    if gene in adata_ileum.var.index:
        print(f"  {gene}: Found in var.index")
    elif gene in adata_ileum.var['gene_symbol'].values:
        ensembl_id = adata_ileum.var[adata_ileum.var['gene_symbol'] == gene].index[0]
        print(f"  {gene}: Found in gene_symbol column, Ensembl ID: {ensembl_id}")
    else:
        print(f"  {gene}: Not found")
```

Tools Used: Python REPL

Observation

```
Checking original dataset gene naming...
Ileum var columns: ['mito', 'ribo', 'hb', 'n_counts', 'n_cells', 'n_genes', 'highly_variable', 'means',
'dispersions', 'dispersions_norm', 'gene_symbols', 'feature_is_filtered', 'feature_name',
'feature_reference', 'feature_biotype', 'feature_length', 'feature_type', 'ensembl_id', 'gene_symbol']
First 10 ileum gene indices: ['ENSG00000238009', 'WASH9P', 'RPL23AP21', 'ENSG00000237094', 'LINC01409',
'LINC00115', 'FAM41C', 'SAMD11', 'NOC2L', 'KLHL17']
First 10 ileum gene symbols: ['ENSG00000238009', 'WASH9P', 'RPL23AP21', 'ENSG00000237094', 'LINC01409',
'LINC00115', 'FAM41C', 'SAMD11', 'NOC2L', 'KLHL17']

Checking for mismatches...
Total mismatches in first 100 genes: 0

Looking for target genes in original ileum data:
  NUSAP1: Found in var.index
  CDX2: Found in var.index
  SOX9: Found in var.index
  LGR5: Found in var.index
```

Step 12

Good! The gene symbols are properly stored in the index. Now let me create a new analysis using the original data with proper gene symbols:

Code Execution

```
# Let's work with the original datasets that have proper gene symbols
# Combine them properly
```

```
adata_ileum.obs['tissue_region'] = 'Ileum'
adata_colon.obs['tissue_region'] = 'Colon'
adata_rectum.obs['tissue_region'] = 'Rectum'

# Find common genes (using gene symbols)
common_genes = set(adata_ileum.var.index) & set(adata_colon.var.index) & set(adata_rectum.var.index)
print(f"Common genes across all datasets: {len(common_genes)}")

# Subset to common genes and concatenate
adata_ileum_sub = adata_ileum[:, list(common_genes)].copy()
adata_colon_sub = adata_colon[:, list(common_genes)].copy()
adata_rectum_sub = adata_rectum[:, list(common_genes)].copy()

# Concatenate
adata_all = adata_ileum_sub.concatenate(adata_colon_sub, adata_rectum_sub,
                                        batch_categories=['Ileum', 'Colon', 'Rectum'])

print(f"Combined dataset shape: {adata_all.shape}")

# Normalize and find markers
sc.pp.normalize_total(adata_all, target_sum=1e4)
sc.pp.log1p(adata_all)

# Calculate marker genes
sc.tl.rank_genes_groups(adata_all, 'CellType', method='wilcoxon', n_genes=100)

# Get TA markers
ta_markers_df = sc.get.rank_genes_groups_df(adata_all, group='TA')
print("\n=== Q2: Top 20 TA cell markers ===")
print(ta_markers_df.head(20))

# Check Q2 options
options_q2 = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
print(f"\nChecking Q2 options in TA markers:")
for option in options_q2:
    if option in ta_markers_df['names'].values:
        row = ta_markers_df[ta_markers_df['names'] == option].iloc[0]
        rank = ta_markers_df[ta_markers_df['names'] == option].index[0] + 1
        print(f"  {option}: Rank {rank}, Score {row['scores']:.3f}")
    else:
        print(f"  {option}: Not in top 100 TA markers")
```

Tools Used: Python REPL

Observation

```
Common genes across all datasets: 15422
Combined dataset shape: (14106, 15422)

=== Q2: Top 20 TA cell markers ===
              names     scores  logfoldchanges  pvals  pvals_adj
0   ENSG00000164104  49.287342       22.807138    0.0        0.0
1   ENSG00000164032  47.658443       31.773426    0.0        0.0
2   ENSG00000117632  47.557880       14.543709    0.0        0.0
3   ENSG00000123416  47.468521       29.164425    0.0        0.0
4   ENSG00000196230  45.993614       17.044569    0.0        0.0
5   ENSG00000189403  45.751892       32.029449    0.0        0.0
6   ENSG00000198830  45.653923       27.071512    0.0        0.0
7   ENSG00000187514  42.941772      114.271042    0.0        0.0
```

```
8   ENSG00000166803  42.901146       11.213116    0.0        0.0
9   ENSG00000167900  42.107391        8.832512    0.0        0.0
10  ENSG00000203760  41.449654        6.627507    0.0        0.0
11  ENSG00000099901  41.332695       10.953118    0.0        0.0
12  ENSG00000124795  40.707226        7.722132    0.0        0.0
13  ENSG00000176890  40.483170        7.409595    0.0        0.0
14  ENSG00000182054  40.477501        9.233834    0.0        0.0
15  ENSG00000069275  40.414684        8.030207    0.0        0.0
16  ENSG00000132341  40.364231       11.444024    0.0        0.0
17  ENSG00000106355  39.435669        6.972369    0.0        0.0
18  ENSG00000173207  39.404640        6.924662    0.0        0.0
19  ENSG00000125835  39.348309        7.799319    0.0        0.0

Checking Q2 options in TA markers:
  NUSAP1: Not in top 100 TA markers
  CDX2: Not in top 100 TA markers
  SOX9: Not in top 100 TA markers
  LGR5: Not in top 100 TA markers
```

Step 13

There's still an issue with the concatenation changing the gene names back to Ensembl IDs. Let me try a different approach and map the Ensembl IDs back to gene symbols:

Code Execution

```
# Let's create a mapping from Ensembl ID to gene symbol using the original data
ensembl_to_symbol = {}

# Get mapping from ileum data (they should all have the same mapping)
for idx in adata_ileum.var.index:
    ensembl_id = adata_ileum.var.loc[idx, 'ensembl_id']
    symbol = idx  # The index is the gene symbol
    ensembl_to_symbol[ensembl_id] = symbol

print(f"Created mapping for {len(ensembl_to_symbol)} genes")

# Now map the TA markers back to gene symbols
print("\n=== Q2: TA cell markers with gene symbols ===")
ta_markers_with_symbols = []
for i, row in ta_markers_df.head(30).iterrows():
    ensembl_id = row['names']
    symbol = ensembl_to_symbol.get(ensembl_id, ensembl_id)
    score = row['scores']
    ta_markers_with_symbols.append((symbol, score))
    print(f"{i+1:2d}. {symbol:12s} (score: {score:.3f})")

# Check Q2 options
options_q2 = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
print(f"\nChecking Q2 options in TA markers:")
ta_genes = [gene for gene, score in ta_markers_with_symbols]
for option in options_q2:
    if option in ta_genes:
        rank = ta_genes.index(option) + 1
        score = ta_markers_with_symbols[ta_genes.index(option)][1]
        print(f"  {option}: Rank {rank}, Score {score:.3f}")
    else:
        # Check if it's anywhere in the full results
        full_ta_df = sc.get.rank_genes_groups_df(adata_all, group='TA')
        found = False
        for j, row in full_ta_df.iterrows():
            ensembl_id = row['names']
            symbol = ensembl_to_symbol.get(ensembl_id, ensembl_id)
            if symbol == option:
                print(f"  {option}: Rank {j+1}, Score {row['scores']:.3f}")
                found = True
                break
        if not found:
            print(f"  {option}: Not found in TA markers")
```

Tools Used: Python REPL

Observation

```
Created mapping for 16795 genes

=== Q2: TA cell markers with gene symbols ===
 1. ENSG00000164104 (score: 49.287)
 2. ENSG00000164032 (score: 47.658)
 3. ENSG00000117632 (score: 47.558)
 4. ENSG00000123416 (score: 47.469)
 5. ENSG00000196230 (score: 45.994)
 6. ENSG00000189403 (score: 45.752)
 7. ENSG00000198830 (score: 45.654)
 8. ENSG00000187514 (score: 42.942)
 9. ENSG00000166803 (score: 42.901)
10. ENSG00000167900 (score: 42.107)
11. ENSG00000203760 (score: 41.450)
12. ENSG00000099901 (score: 41.333)
13. ENSG00000124795 (score: 40.707)
14. ENSG00000176890 (score: 40.483)
15. ENSG00000182054 (score: 40.478)
16. ENSG00000069275 (score: 40.415)
17. ENSG00000132341 (score: 40.364)
18. ENSG00000106355 (score: 39.436)
19. ENSG00000173207 (score: 39.405)
20. ENSG00000125835 (score: 39.348)
21. ENSG00000105968 (score: 39.311)
22. ENSG00000128951 (score: 39.156)
23. ENSG00000168393 (score: 38.829)
24. ENSG00000196262 (score: 38.575)
25. ENSG00000136938 (score: 38.540)
26. ENSG00000122952 (score: 38.516)
27. ENSG00000113810 (score: 38.455)
28. ENSG00000167088 (score: 38.453)
29. ENSG00000102007 (score: 38.140)
30. ENSG00000205581 (score: 37.999)

Checking Q2 options in TA markers:
  NUSAP1: Not found in TA markers
  CDX2: Not found in TA markers
  SOX9: Not found in TA markers
  LGR5: Not found in TA markers
```

Step 14

I think there's an issue with the mapping. Let me check the actual Ensembl IDs to see what genes these are:

Code Execution

```
# Let me check what these top Ensembl IDs actually correspond to
print("Checking what the top TA marker Ensembl IDs correspond to:")

# Create reverse mapping (symbol to ensembl)
symbol_to_ensembl = {}
for idx in adata_ileum.var.index:
    ensembl_id = adata_ileum.var.loc[idx, 'ensembl_id']
    symbol = idx
    symbol_to_ensembl[symbol] = ensembl_id

# Check the top markers
top_ensembl_ids = ta_markers_df.head(10)['names'].tolist()
print("\nTop 10 TA marker genes:")
for i, ensembl_id in enumerate(top_ensembl_ids):
    # Find the gene symbol that corresponds to this Ensembl ID
    found_symbol = None
    for symbol, ens_id in symbol_to_ensembl.items():
        if ens_id == ensembl_id:
            found_symbol = symbol
            break

    if found_symbol:
        score = ta_markers_df.iloc[i]['scores']
        print(f"{i+1:2d}. {found_symbol:12s} (Ensembl: {ensembl_id}, Score: {score:.3f})")
    else:
        score = ta_markers_df.iloc[i]['scores']
        print(f"{i+1:2d}. {ensembl_id:12s} (Score: {score:.3f}) - Symbol not found")

# Now check our target genes
print(f"\nChecking target genes in symbol_to_ensembl mapping:")
target_genes = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
for gene in target_genes:
    if gene in symbol_to_ensembl:
        ensembl_id = symbol_to_ensembl[gene]
        print(f"  {gene}: {ensembl_id}")

        # Check if this Ensembl ID is in TA markers
        if ensembl_id in ta_markers_df['names'].values:
            row = ta_markers_df[ta_markers_df['names'] == ensembl_id].iloc[0]
            rank = ta_markers_df[ta_markers_df['names'] == ensembl_id].index[0] + 1
            print(f"    -> Rank {rank}, Score {row['scores']:.3f}")
    else:
        print(f"  {gene}: Not found in mapping")
```

Tools Used: Python REPL

Observation

```
Checking what the top TA marker Ensembl IDs correspond to:

Top 10 TA marker genes:
 1. ENSG00000164104 (Score: 49.287) - Symbol not found
 2. ENSG00000164032 (Score: 47.658) - Symbol not found
 3. ENSG00000117632 (Score: 47.558) - Symbol not found
 4. ENSG00000123416 (Score: 47.469) - Symbol not found
 5. ENSG00000196230 (Score: 45.994) - Symbol not found
 6. ENSG00000189403 (Score: 45.752) - Symbol not found
 7. ENSG00000198830 (Score: 45.654) - Symbol not found
 8. ENSG00000187514 (Score: 42.942) - Symbol not found
 9. ENSG00000166803 (Score: 42.901) - Symbol not found
10. ENSG00000167900 (Score: 42.107) - Symbol not found

Checking target genes in symbol_to_ensembl mapping:
  NUSAP1: NUSAP1
  CDX2: CDX2
  SOX9: SOX9
  LGR5: LGR5
```

Step 15

I see the issue - the gene symbols are being used as their own Ensembl IDs in some cases. Let me try a different approach and look up these Ensembl IDs directly:

Code Execution

```
# Let me try to use external database to identify these genes
from biomni.tool.database import query_ensembl

# Check a few of the top Ensembl IDs
top_ensembl_ids = ['ENSG00000164104', 'ENSG00000164032', 'ENSG00000117632', 'ENSG00000123416']

print("Looking up top TA marker Ensembl IDs:")
for ensembl_id in top_ensembl_ids:
    try:
        result = query_ensembl(f"gene symbol for {ensembl_id}")
        print(f"{ensembl_id}: {result}")
    except:
        print(f"{ensembl_id}: Could not query")

# Let me also try a more direct approach - check if NUSAP1 appears anywhere in the marker results
print(f"\nDirect search for target genes in all marker results:")
all_markers = sc.get.rank_genes_groups_df(adata_all, group=None)

target_genes = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
for gene in target_genes:
    # Check if gene appears in any group's markers
    gene_rows = all_markers[all_markers['names'] == gene]
    if len(gene_rows) > 0:
        print(f"\n{gene} found in markers:")
        for _, row in gene_rows.iterrows():
            print(f"  Group: {row['group']}, Rank: {row.name + 1}, Score: {row['scores']:.3f}")
    else:
        print(f"\n{gene}: Not found in any group's markers")

# Let's also check if these genes are highly expressed in TA cells specifically
print(f"\nExpression analysis of target genes in TA cells:")
ta_cells = adata_all[adata_all.obs['CellType'] == 'TA']
for gene in target_genes:
    if gene in adata_all.var.index:
        ta_expr = ta_cells[:, gene].X.mean()
        all_expr = adata_all[:, gene].X.mean()
        fold_change = ta_expr / all_expr if all_expr > 0 else 0
        print(f"  {gene}: TA mean={ta_expr:.3f}, All mean={all_expr:.3f}, FC={fold_change:.2f}")
```

Tools Used: database → query_ensembl

Observation

```
Looking up top TA marker Ensembl IDs:
ENSG00000164104: {'success': False, 'error': "Error querying LLM: Error code: 503 - {'error': {'message':
'当前分组 default 下对于模型 claude-sonnet-4-5 计费模式 [按量计费,按次计费] 无可用渠道 (request id:
2026010703145815468368272734515)', 'type': 'shell_api_error'}}"}
ENSG00000164032: {'success': False, 'error': "Error querying LLM: Error code: 503 - {'error': {'message':
'当前分组 default 下对于模型 claude-sonnet-4-5 计费模式 [按量计费,按次计费] 无可用渠道 (request id:
2026010703150073260260910421587)', 'type': 'shell_api_error'}}"}
ENSG00000117632: {'success': False, 'error': "Error querying LLM: Error code: 503 - {'error': {'message':
'当前分组 default 下对于模型 claude-sonnet-4-5 计费模式 [按量计费,按次计费] 无可用渠道 (request id:
2026010703150319121457609324956)', 'type': 'shell_api_error'}}"}
ENSG00000123416: {'success': False, 'error': "Error querying LLM: Error code: 503 - {'error': {'message':
'当前分组 default 下对于模型 claude-sonnet-4-5 计费模式 [按量计费,按次计费] 无可用渠道 (request id:
2026010703150577807143055754501)', 'type': 'shell_api_error'}}"}

Direct search for target genes in all marker results:

NUSAP1: Not found in any group's markers

CDX2: Not found in any group's markers

SOX9: Not found in any group's markers

LGR5: Not found in any group's markers

Expression analysis of target genes in TA cells:
  NUSAP1: TA mean=0.789, All mean=0.080, FC=9.86
  CDX2: TA mean=1.242, All mean=1.201, FC=1.03
  SOX9: TA mean=0.655, All mean=0.451, FC=1.45
  LGR5: TA mean=0.112, All mean=0.058, FC=1.95
```

Step 16

Excellent! Now I can see that NUSAP1 has a very high fold change (9.86x) in TA cells compared to all cells, which suggests it's a strong marker for TA cells. Let me verify this by calculating the actual statistical significance:

Code Execution

```
# Let's manually calculate the marker gene statistics for our target genes
import scipy.stats as stats

print("=== Manual marker gene analysis for Q2 options ===")

target_genes = ['NUSAP1', 'CDX2', 'SOX9', 'LGR5']
ta_cells_idx = adata_all.obs['CellType'] == 'TA'
non_ta_cells_idx = adata_all.obs['CellType'] != 'TA'

for gene in target_genes:
    if gene in adata_all.var.index:
        # Get expression values
        ta_expr = adata_all[ta_cells_idx, gene].X.toarray().flatten()
        non_ta_expr = adata_all[non_ta_cells_idx, gene].X.toarray().flatten()

        # Calculate statistics
        ta_mean = ta_expr.mean()
        non_ta_mean = non_ta_expr.mean()
        fold_change = ta_mean / non_ta_mean if non_ta_mean > 0 else float('inf')

        # Perform statistical test
        try:
            stat, pval = stats.mannwhitneyu(ta_expr, non_ta_expr, alternative='greater')

            # Calculate what percentage of TA cells express this gene
            ta_expressing = (ta_expr > 0).sum()
            ta_total = len(ta_expr)
            ta_percent = ta_expressing / ta_total * 100

            # Calculate what percentage of non-TA cells express this gene
            non_ta_expressing = (non_ta_expr > 0).sum()
            non_ta_total = len(non_ta_expr)
            non_ta_percent = non_ta_expressing / non_ta_total * 100

            print(f"\n{gene}:")
            print(f"  TA cells expressing: {ta_expressing}/{ta_total} ({ta_percent:.1f}%)")
            print(f"  Non-TA cells expressing: {non_ta_expressing}/{non_ta_total} ({non_ta_percent:.1f}
%)")
            print(f"  Mean expression - TA: {ta_mean:.3f}, Non-TA: {non_ta_mean:.3f}")
            print(f"  Fold change: {fold_change:.2f}")
            print(f"  P-value: {pval:.2e}")

        except Exception as e:
            print(f"  Error in statistical test: {e}")

print(f"\n=== Summary for Q2 ===")
print("Based on fold change analysis, NUSAP1 shows the highest enrichment in TA cells (9.86x fold
change)")
print("This suggests NUSAP1 is a new marker for transient-amplifying cells.")
print("Answer: A) NUSAP1")
```

Tools Used: Python REPL

Observation

```
=== Manual marker gene analysis for Q2 options ===

NUSAP1:
  TA cells expressing: 613/964 (63.6%)
  Non-TA cells expressing: 306/13142 (2.3%)
  Mean expression - TA: 0.789, Non-TA: 0.028
  Fold change: 28.18
  P-value: 0.00e+00

CDX2:
  TA cells expressing: 856/964 (88.8%)
  Non-TA cells expressing: 9559/13142 (72.7%)
  Mean expression - TA: 1.242, Non-TA: 1.198
  Fold change: 1.04
  P-value: 1.00e+00

SOX9:
  TA cells expressing: 568/964 (58.9%)
  Non-TA cells expressing: 4317/13142 (32.8%)
  Mean expression - TA: 0.655, Non-TA: 0.436
  Fold change: 1.50
  P-value: 1.07e-32

LGR5:
  TA cells expressing: 116/964 (12.0%)
  Non-TA cells expressing: 556/13142 (4.2%)
  Mean expression - TA: 0.112, Non-TA: 0.054
  Fold change: 2.09
  P-value: 1.70e-26

=== Summary for Q2 ===
Based on fold change analysis, NUSAP1 shows the highest enrichment in TA cells (9.86x fold change)
This suggests NUSAP1 is a new marker for transient-amplifying cells.
Answer: A) NUSAP1
```

Step 17

Perfect! Now let me analyze goblet cell markers for Q3:

Code Execution

```
# Q3: Gene enriched in goblet cells across all three regions
print("=== Q3: Goblet cell markers across all regions ===")

# Get goblet cell markers
goblet_markers_df = sc.get.rank_genes_groups_df(adata_all, group='Goblet')
print("Top 20 goblet cell markers:")
print(goblet_markers_df.head(20))

# Check Q3 options
options_q3 = ['ITLN1', 'TFF1', 'MYC', 'APOA1']
print(f"\nChecking Q3 options in goblet cell markers:")

# Check if these genes are in the top markers
for option in options_q3:
    if option in goblet_markers_df['names'].values:
        row = goblet_markers_df[goblet_markers_df['names'] == option].iloc[0]
        rank = goblet_markers_df[goblet_markers_df['names'] == option].index[0] + 1
        print(f"  {option}: Rank {rank}, Score {row['scores']:.3f}")
    else:
        print(f"  {option}: Not in top markers")

# Let's also check expression in goblet cells vs other cells for these options
print(f"\nExpression analysis of Q3 options in goblet cells:")
goblet_cells_idx = adata_all.obs['CellType'] == 'Goblet'
non_goblet_cells_idx = adata_all.obs['CellType'] != 'Goblet'

for gene in options_q3:
    if gene in adata_all.var.index:
        # Get expression by tissue region for goblet cells
        for region in ['Ileum', 'Colon', 'Rectum']:
            region_goblet_idx = (adata_all.obs['CellType'] == 'Goblet') & (adata_all.obs['batch'] ==
region)
            if region_goblet_idx.sum() > 0:
                region_expr = adata_all[region_goblet_idx, gene].X.mean()
                print(f"  {gene} in {region} goblet cells: {region_expr:.3f}")

        # Overall statistics
        goblet_expr = adata_all[goblet_cells_idx, gene].X.mean()
        non_goblet_expr = adata_all[non_goblet_cells_idx, gene].X.mean()
        fold_change = goblet_expr / non_goblet_expr if non_goblet_expr > 0 else float('inf')
        print(f"  {gene} overall FC in goblet cells: {fold_change:.2f}")
        print()
```

Tools Used: Python REPL

Observation

```
=== Q3: Goblet cell markers across all regions ===
Top 20 goblet cell markers:
              names     scores  logfoldchanges          pvals      pvals_adj
0   ENSG00000160180  69.290100      422.050293   0.000000e+00   0.000000e+00
1   ENSG00000275395  67.207191       70.878181   0.000000e+00   0.000000e+00
2   ENSG00000167748  60.740082       17.457697   0.000000e+00   0.000000e+00
3   ENSG00000184709  57.664539        6.549780   0.000000e+00   0.000000e+00
4   ENSG00000179914  56.117908       16.369087   0.000000e+00   0.000000e+00
5   ENSG00000174236  54.157074        6.412893   0.000000e+00   0.000000e+00
6   ENSG00000070526  51.333145        6.370056   0.000000e+00   0.000000e+00
7   ENSG00000016490  50.647903       30.707626   0.000000e+00   0.000000e+00
8   ENSG00000188175  48.290699        4.461995   0.000000e+00   0.000000e+00
9   ENSG00000129538  45.820595        6.320205   0.000000e+00   0.000000e+00
10  ENSG00000122711  45.154137       48.071217   0.000000e+00   0.000000e+00
11  ENSG00000172238  44.926689        5.580137   0.000000e+00   0.000000e+00
12  ENSG00000107281  42.161896        3.831885   0.000000e+00   0.000000e+00
13  ENSG00000116176  41.209175        5.420337   0.000000e+00   0.000000e+00
14  ENSG00000142669  38.882439        7.491322   0.000000e+00   0.000000e+00
15  ENSG00000157613  38.451515        2.996501   0.000000e+00   0.000000e+00
16  ENSG00000106541  38.446861       31.811293   0.000000e+00   0.000000e+00
17  ENSG00000148180  38.116783        6.134699   0.000000e+00   0.000000e+00
18  ENSG00000124664  37.460396        3.603790  4.068443e-307  1.530132e-304
19  ENSG00000101443  36.607941       15.843679  2.138350e-293  6.869449e-291

Checking Q3 options in goblet cell markers:
  ITLN1: Not in top markers
  TFF1: Not in top markers
  MYC: Not in top markers
  APOA1: Not in top markers

Expression analysis of Q3 options in goblet cells:
  ITLN1 in Ileum goblet cells: 1.575
  ITLN1 in Colon goblet cells: 1.786
  ITLN1 in Rectum goblet cells: 2.167
  ITLN1 overall FC in goblet cells: 10.54

  TFF1 in Ileum goblet cells: 1.273
  TFF1 in Colon goblet cells: 0.577
  TFF1 in Rectum goblet cells: 0.983
  TFF1 overall FC in goblet cells: 2.22

  MYC in Ileum goblet cells: 0.284
  MYC in Colon goblet cells: 0.396
  MYC in Rectum goblet cells: 0.486
  MYC overall FC in goblet cells: 1.15

  APOA1 in Ileum goblet cells: 2.059
  APOA1 in Colon goblet cells: 0.074
```

```
APOA1 in Rectum goblet cells: 0.005
APOA1 overall FC in goblet cells: 0.29
```

Step 18

Great! I can see that ITLN1 shows consistent high expression across all three regions (Ileum: 1.575, Colon: 1.786, Rectum: 2.167) and has a high fold change (10.54x) in goblet cells. Let me verify this further:

Code Execution

```
# Let's verify ITLN1 is indeed a strong goblet cell marker across all regions
print("=== Detailed analysis of ITLN1 in goblet cells ===")

# Check ITLN1 expression specifically
gene = 'ITLN1'
if gene in adata_all.var.index:
    # Expression by cell type
    print("Expression by cell type:")
    for cell_type in adata_all.obs['CellType'].unique():
        cell_type_idx = adata_all.obs['CellType'] == cell_type
        if cell_type_idx.sum() > 0:
            mean_expr = adata_all[cell_type_idx, gene].X.mean()
            expressing_cells = (adata_all[cell_type_idx, gene].X > 0).sum()
            total_cells = cell_type_idx.sum()
            percent_expressing = expressing_cells / total_cells * 100
            print(f"  {cell_type:15s}: Mean={mean_expr:.3f}, Expressing={expressing_cells}/{total_cells}
({percent_expressing:.1f}%)")

    # Expression in goblet cells by region
    print(f"\n{gene} expression in goblet cells by region:")
    for region in ['Ileum', 'Colon', 'Rectum']:
        region_goblet_idx = (adata_all.obs['CellType'] == 'Goblet') & (adata_all.obs['batch'] == region)
        if region_goblet_idx.sum() > 0:
            mean_expr = adata_all[region_goblet_idx, gene].X.mean()
            expressing_cells = (adata_all[region_goblet_idx, gene].X > 0).sum()
            total_cells = region_goblet_idx.sum()
            percent_expressing = expressing_cells / total_cells * 100
            print(f"  {region:6s}: Mean={mean_expr:.3f}, Expressing={expressing_cells}/{total_cells}
({percent_expressing:.1f}%)")

# Let's also check the other options for comparison
print(f"\n=== Comparison of all Q3 options ===")
options_q3 = ['ITLN1', 'TFF1', 'MYC', 'APOA1']

for gene in options_q3:
    if gene in adata_all.var.index:
        print(f"\n{gene}:")
        goblet_cells_idx = adata_all.obs['CellType'] == 'Goblet'
        non_goblet_cells_idx = adata_all.obs['CellType'] != 'Goblet'

        goblet_mean = adata_all[goblet_cells_idx, gene].X.mean()
        non_goblet_mean = adata_all[non_goblet_cells_idx, gene].X.mean()
        fold_change = goblet_mean / non_goblet_mean if non_goblet_mean > 0 else float('inf')

        # Check consistency across regions
        region_means = []
        for region in ['Ileum', 'Colon', 'Rectum']:
            region_goblet_idx = (adata_all.obs['CellType'] == 'Goblet') & (adata_all.obs['batch'] ==
region)
```

```
            if region_goblet_idx.sum() > 0:
                region_mean = adata_all[region_goblet_idx, gene].X.mean()
                region_means.append(region_mean)

        consistency_score = min(region_means) / max(region_means) if max(region_means) > 0 else 0

        print(f"  Fold change in goblet cells: {fold_change:.2f}")
        print(f"  Regional means: {region_means}")
        print(f"  Consistency score: {consistency_score:.2f} (higher = more consistent)")

print(f"\n=== Q3 Answer ===")
print("ITLN1 shows high and consistent expression across all three intestinal regions in goblet cells")
print("Answer: A) ITLN1")
```

Tools Used: Python REPL

Observation

```
=== Detailed analysis of ITLN1 in goblet cells ===
Expression by cell type:
  Enterocyte     : Mean=0.109, Expressing=432/5697 (7.6%)
  Goblet         : Mean=1.928, Expressing=1668/1967 (84.8%)
  Paneth-like    : Mean=0.160, Expressing=109/742 (14.7%)
  Enteriendocrine: Mean=0.235, Expressing=19/117 (16.2%)
  Progenitor     : Mean=0.227, Expressing=772/4001 (19.3%)
  TA             : Mean=0.335, Expressing=247/964 (25.6%)
  Stem Cell      : Mean=0.359, Expressing=173/618 (28.0%)

ITLN1 expression in goblet cells by region:
  Ileum : Mean=1.575, Expressing=212/303 (70.0%)
  Colon : Mean=1.786, Expressing=638/765 (83.4%)
  Rectum: Mean=2.167, Expressing=818/899 (91.0%)

=== Comparison of all Q3 options ===

ITLN1:
  Fold change in goblet cells: 10.54
  Regional means: [np.float32(1.574517), np.float32(1.786372), np.float32(2.1667128)]
  Consistency score: 0.73 (higher = more consistent)

TFF1:
  Fold change in goblet cells: 2.22
  Regional means: [np.float32(1.2734404), np.float32(0.5766294), np.float32(0.98267186)]
  Consistency score: 0.45 (higher = more consistent)

MYC:
  Fold change in goblet cells: 1.15
  Regional means: [np.float32(0.28400573), np.float32(0.39579472), np.float32(0.48576304)]
  Consistency score: 0.58 (higher = more consistent)

APOA1:
  Fold change in goblet cells: 0.29
  Regional means: [np.float32(2.0591497), np.float32(0.07445932), np.float32(0.0054119714)]
  Consistency score: 0.00 (higher = more consistent)

=== Q3 Answer ===
ITLN1 shows high and consistent expression across all three intestinal regions in goblet cells
Answer: A) ITLN1
```

Step 19

Now let me analyze Paneth-like cells for Q4:

Code Execution

```
# Q4: Paneth-like cells (PLCs) markers - which gene can NOT be used to identify these cells?
print("=== Q4: Paneth-like cell markers ===")

# Get Paneth-like cell markers
paneth_markers_df = sc.get.rank_genes_groups_df(adata_all, group='Paneth-like')
print("Top 20 Paneth-like cell markers:")
print(paneth_markers_df.head(20))

# Check Q4 options
options_q4 = ['LYZ', 'CA4', 'APOA4', 'SPIB']
print(f"\nChecking Q4 options in Paneth-like cell markers:")

# Check if these genes are in the top markers
for option in options_q4:
    if option in paneth_markers_df['names'].values:
        row = paneth_markers_df[paneth_markers_df['names'] == option].iloc[0]
        rank = paneth_markers_df[paneth_markers_df['names'] == option].index[0] + 1
        print(f"  {option}: Rank {rank}, Score {row['scores']:.3f}")
    else:
        print(f"  {option}: Not in top markers")

# Let's check expression levels and specificity
print(f"\nExpression analysis of Q4 options in Paneth-like cells:")
paneth_cells_idx = adata_all.obs['CellType'] == 'Paneth-like'
non_paneth_cells_idx = adata_all.obs['CellType'] != 'Paneth-like'

for gene in options_q4:
    if gene in adata_all.var.index:
        paneth_expr = adata_all[paneth_cells_idx, gene].X.mean()
        non_paneth_expr = adata_all[non_paneth_cells_idx, gene].X.mean()
        fold_change = paneth_expr / non_paneth_expr if non_paneth_expr > 0 else float('inf')

        # Check expression in Paneth-like cells
        paneth_expressing = (adata_all[paneth_cells_idx, gene].X > 0).sum()
        paneth_total = paneth_cells_idx.sum()
        paneth_percent = paneth_expressing / paneth_total * 100

        print(f"\n{gene}:")
        print(f"  Mean expression in Paneth-like: {paneth_expr:.3f}")
        print(f"  Mean expression in non-Paneth-like: {non_paneth_expr:.3f}")
        print(f"  Fold change: {fold_change:.2f}")
        print(f"  Expressing Paneth-like cells: {paneth_expressing}/{paneth_total} ({paneth_percent:.1f}
%)")

        # Check if it's also highly expressed in other cell types
        print(f"  Expression in other cell types:")
        for cell_type in ['Enterocyte', 'Goblet', 'TA', 'Stem Cell', 'Progenitor', 'Enteriendocrine']:
            ct_idx = adata_all.obs['CellType'] == cell_type
            if ct_idx.sum() > 0:
                ct_expr = adata_all[ct_idx, gene].X.mean()
                ct_expressing = (adata_all[ct_idx, gene].X > 0).sum()
                ct_total = ct_idx.sum()
                ct_percent = ct_expressing / ct_total * 100
                print(f"    {cell_type:15s}: Mean={ct_expr:.3f}, Expressing={ct_percent:.1f}%")
```

Tools Used: Python REPL

Observation

```
=== Q4: Paneth-like cell markers ===
Top 20 Paneth-like cell markers:
              names     scores  logfoldchanges          pvals      pvals_adj
0   ENSG00000269404  43.155617       13.981342   0.000000e+00   0.000000e+00
1   ENSG00000168748  42.798031       52.415188   0.000000e+00   0.000000e+00
2   ENSG00000167434  37.838772       40.320015   0.000000e+00   0.000000e+00
3   ENSG00000088832  37.564850       17.130270  8.062434e-309  3.108068e-305
4   ENSG00000142959  37.085739       16.722857  4.770042e-301  1.471081e-297
5   ENSG00000106211  36.608276       11.648397  2.112169e-293  5.428273e-290
6   ENSG00000188290  34.519527        6.054525  4.086580e-261  9.002152e-258
7   ENSG00000196188  33.828579        7.289050  7.497031e-251  1.445053e-247
8   ENSG00000270629  33.537952        5.618852  1.349111e-246  2.311477e-243
9   ENSG00000188175  32.917889        6.070938  1.219316e-237  1.880185e-234
10  ENSG00000107954  32.402645        5.265451  2.519138e-230  3.531374e-227
11  ENSG00000086548  32.137276        7.424510  1.330077e-226  1.709149e-223
12  ENSG00000271383  32.043877        5.334399  2.671769e-225  3.169129e-222
13  ENSG00000034510  31.899368      185.436752  2.724526e-223  3.000870e-220
14  ENSG00000259823  31.101435       14.510640  2.303251e-212  2.367742e-209
15  ENSG00000125968  30.485912       13.948833  4.005961e-204  3.860745e-201
16  ENSG00000105388  30.469761        9.373098  6.557548e-204  5.948082e-201
17  ENSG00000112312  30.119112        5.869996  2.723576e-199  2.333197e-196
18  ENSG00000090382  30.112841        9.907763  3.290341e-199  2.670371e-196
19  ENSG00000183034  29.647354       10.410186  3.668730e-193  2.828591e-190

Checking Q4 options in Paneth-like cell markers:
  LYZ: Not in top markers
  CA4: Not in top markers
  APOA4: Not in top markers
  SPIB: Not in top markers

Expression analysis of Q4 options in Paneth-like cells:

LYZ:
  Mean expression in Paneth-like: 1.328
  Mean expression in non-Paneth-like: 0.225
  Fold change: 5.90
  Expressing Paneth-like cells: 551/742 (74.3%)
  Expression in other cell types:
    Enterocyte     : Mean=0.146, Expressing=9.6%
    Goblet         : Mean=0.424, Expressing=27.8%
    TA             : Mean=0.366, Expressing=29.0%
    Stem Cell      : Mean=0.316, Expressing=23.6%
    Progenitor     : Mean=0.193, Expressing=16.6%
    Enteriendocrine: Mean=0.175, Expressing=12.0%

CA4:
  Mean expression in Paneth-like: 2.131
  Mean expression in non-Paneth-like: 0.364
  Fold change: 5.86
  Expressing Paneth-like cells: 668/742 (90.0%)
  Expression in other cell types:
    Enterocyte     : Mean=0.410, Expressing=19.4%
    Goblet         : Mean=0.288, Expressing=20.9%
    TA             : Mean=0.316, Expressing=31.4%
    Stem Cell      : Mean=0.250, Expressing=24.3%
    Progenitor     : Mean=0.366, Expressing=30.9%
    Enteriendocrine: Mean=0.303, Expressing=21.4%

SPIB:
  Mean expression in Paneth-like: 1.954
  Mean expression in non-Paneth-like: 0.090
  Fold change: 21.61
  Expressing Paneth-like cells: 715/742 (96.4%)
  Expression in other cell types:
    Enterocyte     : Mean=0.052, Expressing=3.3%
    Goblet         : Mean=0.085, Expressing=6.4%
    TA             : Mean=0.220, Expressing=17.2%
    Stem Cell      : Mean=0.119, Expressing=10.2%
    Progenitor     : Mean=0.111, Expressing=9.1%
    Enteriendocrine: Mean=0.115, Expressing=7.7%
```

Step 20

I notice that APOA4 wasn't analyzed. Let me check if it exists in the dataset and complete the analysis:

Code Execution

```
# Check if APOA4 exists in the dataset
print("Checking for APOA4...")
if 'APOA4' in adata_all.var.index:
    print("APOA4 found in dataset")

    gene = 'APOA4'
    paneth_cells_idx = adata_all.obs['CellType'] == 'Paneth-like'

    paneth_expr = adata_all[paneth_cells_idx, gene].X.mean()
    non_paneth_expr = adata_all[~paneth_cells_idx, gene].X.mean()
    fold_change = paneth_expr / non_paneth_expr if non_paneth_expr > 0 else float('inf')

    paneth_expressing = (adata_all[paneth_cells_idx, gene].X > 0).sum()
    paneth_total = paneth_cells_idx.sum()
    paneth_percent = paneth_expressing / paneth_total * 100

    print(f"\nAPOA4:")
    print(f"  Mean expression in Paneth-like: {paneth_expr:.3f}")
    print(f"  Mean expression in non-Paneth-like: {non_paneth_expr:.3f}")
    print(f"  Fold change: {fold_change:.2f}")
    print(f"  Expressing Paneth-like cells: {paneth_expressing}/{paneth_total} ({paneth_percent:.1f}%)")

    # Check expression in other cell types
    print(f"  Expression in other cell types:")
    for cell_type in ['Enterocyte', 'Goblet', 'TA', 'Stem Cell', 'Progenitor', 'Enteriendocrine']:
        ct_idx = adata_all.obs['CellType'] == cell_type
        if ct_idx.sum() > 0:
            ct_expr = adata_all[ct_idx, gene].X.mean()
            ct_expressing = (adata_all[ct_idx, gene].X > 0).sum()
            ct_total = ct_idx.sum()
            ct_percent = ct_expressing / ct_total * 100
            print(f"    {cell_type:15s}: Mean={ct_expr:.3f}, Expressing={ct_percent:.1f}%")
else:
    print("APOA4 not found in dataset")

# Summary analysis for Q4
print(f"\n=== Q4 Summary ===")
print("Analysis of potential Paneth-like cell markers:")
print("LYZ: FC=5.90, expressed in 74.3% of Paneth-like cells")
print("CA4: FC=5.86, expressed in 90.0% of Paneth-like cells")
print("SPIB: FC=21.61, expressed in 96.4% of Paneth-like cells - STRONGEST marker")
if 'APOA4' in adata_all.var.index:
    apoa4_expr = adata_all[adata_all.obs['CellType'] == 'Paneth-like', 'APOA4'].X.mean()
    apoa4_non = adata_all[adata_all.obs['CellType'] != 'Paneth-like', 'APOA4'].X.mean()
    apoa4_fc = apoa4_expr / apoa4_non if apoa4_non > 0 else float('inf')
    apoa4_pct = (adata_all[adata_all.obs['CellType'] == 'Paneth-like', 'APOA4'].X > 0).sum() /
(adata_all.obs['CellType'] == 'Paneth-like').sum() * 100
    print(f"APOA4: FC={apoa4_fc:.2f}, expressed in {apoa4_pct:.1f}% of Paneth-like cells")

# The question asks which gene can NOT be used to identify Paneth-like cells
# We need to find the gene with the lowest specificity or fold change
print(f"\nBased on fold changes and specificity:")
print("SPIB has the highest fold change (21.61x) and highest expression (96.4%)")
print("LYZ and CA4 have moderate fold changes (~6x)")
print("The gene that would be LEAST useful for identifying Paneth-like cells would be the one with lowest
fold change")
```

Tools Used: Python REPL

Observation

```
Checking for APOA4...
APOA4 not found in dataset

=== Q4 Summary ===
Analysis of potential Paneth-like cell markers:
LYZ: FC=5.90, expressed in 74.3% of Paneth-like cells
CA4: FC=5.86, expressed in 90.0% of Paneth-like cells
SPIB: FC=21.61, expressed in 96.4% of Paneth-like cells - STRONGEST marker

Based on fold changes and specificity:
SPIB has the highest fold change (21.61x) and highest expression (96.4%)
LYZ and CA4 have moderate fold changes (~6x)
The gene that would be LEAST useful for identifying Paneth-like cells would be the one with lowest fold
change
```

Step 21

Let me check if APOA4 might be present under a different name or if there are any genes similar to it:

Code Execution

```
# Search for APOA4 or similar genes
print("Searching for APOA4 or related genes...")
apoa_genes = [gene for gene in adata_all.var.index if 'APOA' in gene]
print(f"Genes containing 'APOA': {apoa_genes}")

# Let's also check if there might be alternative names
apo_genes = [gene for gene in adata_all.var.index if gene.startswith('APO')]
```

```
print(f"Genes starting with 'APO': {apo_genes[:20]}")  # Show first 20

# Since APOA4 is missing, let's analyze what we have
print(f"\n=== Final Q4 Analysis ===")
print("Available genes and their suitability as Paneth-like cell markers:")

markers_data = {
    'LYZ': {'fc': 5.90, 'pct': 74.3},
    'CA4': {'fc': 5.86, 'pct': 90.0},
    'SPIB': {'fc': 21.61, 'pct': 96.4}
}

for gene, data in markers_data.items():
    print(f"{gene}: Fold change = {data['fc']:.2f}, Expression = {data['pct']:.1f}%")

print(f"\nAmong the available genes:")
print("- SPIB is an excellent marker (highest FC and expression)")
print("- LYZ is a good marker")
print("- CA4 is a good marker")
print("- APOA4 is not present in the dataset")

print(f"\nSince APOA4 is not found in the dataset, it cannot be used to identify Paneth-like cells")
print("Answer: D) APOA4")

# Let's double-check by looking at the original question context
print(f"\n=== Double-checking with literature context ===")
print("In intestinal biology:")
print("- LYZ (Lysozyme) is a well-known Paneth cell marker")
print("- CA4 (Carbonic Anhydrase 4) can be expressed in intestinal cells")
print("- SPIB is a transcription factor that can mark Paneth-like cells")
print("- APOA4 (Apolipoprotein A4) is primarily involved in lipid metabolism")
print("APOA4 would be the least specific for Paneth-like cells even if present")
```

Tools Used: Python REPL

Observation

```
Searching for APOA4 or related genes...
Genes containing 'APOA': ['APOA1']
Genes starting with 'APO': ['APOL3', 'APOBEC3D', 'APOBEC3C', 'APOH', 'APOBEC3F', 'APOA1', 'APOL2', 'APOE',
'APOBR', 'APOL6', 'APOC1', 'APOOL', 'APOL1', 'APOM', 'APOLD1', 'APOL4', 'APOO', 'APOBEC3G', 'APOBEC1']

=== Final Q4 Analysis ===
Available genes and their suitability as Paneth-like cell markers:
LYZ: Fold change = 5.90, Expression = 74.3%
CA4: Fold change = 5.86, Expression = 90.0%
SPIB: Fold change = 21.61, Expression = 96.4%

Among the available genes:
- SPIB is an excellent marker (highest FC and expression)
- LYZ is a good marker
- CA4 is a good marker
- APOA4 is not present in the dataset

Since APOA4 is not found in the dataset, it cannot be used to identify Paneth-like cells
Answer: D) APOA4

=== Double-checking with literature context ===
In intestinal biology:
```

```
- LYZ (Lysozyme) is a well-known Paneth cell marker
- CA4 (Carbonic Anhydrase 4) can be expressed in intestinal cells
- SPIB is a transcription factor that can mark Paneth-like cells
- APOA4 (Apolipoprotein A4) is primarily involved in lipid metabolism
APOA4 would be the least specific for Paneth-like cells even if present
```

Step 22

Perfect! Now let me update my progress and provide the final answers:

Plan

- *[x]* Analyze cell type abundance across regions to answer Q1
- *[x]* Examine marker genes for different cell types to answer Q2, Q3, and Q4
- *[x]* Integrate datasets and perform comparative analysis
- *[x]* Provide final answers

Summary and Solution

Based on my comprehensive analysis of the single-cell RNA-seq data from human intestinal epithelial cells across ileum, colon, and rectum, here are the answers: **Q1: Which cell type was most abundant in the human ileum according to the single-cell transcriptomic data?** Answer: **B) Enterocytes** Analysis showed that in the ileum dataset: - Enterocytes: 4,243 cells (70.95%) - Progenitor cells: 782 cells (13.08%) - Goblet cells: 303 cells (5.07%) - Other cell types: <5% each **Q2: Which new marker was identified for transient-amplifying (TA) cells in the human intestine?** Answer: **A) NUSAP1** NUSAP1 showed exceptional specificity for TA cells with: - 28.18-fold enrichment in TA cells vs other cell types - Expression in 63.6% of TA cells vs only 2.3% of non-TA cells - Highly significant differential expression ($p < 0.001$) **Q3: Which gene was identified as being enriched in goblet cells in all three regions of the human intestine (ileum, colon, and rectum)?** Answer: **A) ITLN1** ITLN1 demonstrated consistent high expression across all regions: - Ileum goblet cells: 1.575 mean expression - Colon goblet cells: 1.786 mean expression - Rectum goblet cells: 2.167 mean expression - 10.54-fold enrichment in goblet cells overall - Expressed in 70-91% of goblet cells across regions **Q4: The data identified Paneth-like cells (PLCs) in the large intestine. What gene can NOT be used to identify these cells?** Answer: **D) APOA4** Analysis revealed: - SPIB: Excellent marker (21.61-fold change, 96.4% expression) - LYZ: Good marker (5.90-fold change, 74.3% expression) - CA4: Good marker (5.86-fold change, 90.0% expression) - APOA4: Not present in the dataset, therefore cannot be used for identification