

API Document

主后端应用 Base URL & Port: <http://127.0.0.1:41888/>

Agent 聊天对话 Base URL & Port: <http://127.0.0.1:41889/>

若无特殊说明，HTTP 请求均发送到**主后端应用**中。

Update

- 1.3 接口新增 `parent_snapshot_id` 返回字段
- 替换所有 `"source_snapshot_id"` 为 `"snapshot_id"`。
-

1. 项目、数据集管理

1.1 用户新建 Dataset & 上传本地 Raw Data

1.1.1 接口概述（Purpose）& 触发逻辑

该接口用于将用户本地的原始数据文件导入到一个**已存在的**项目空间中。后端会处理文件验证、格式标准化（统一转为 `.h5ad`），并创建数据集的数据库记录。

- **核心逻辑：**

1. **校验项目：**根据 `project_id` 检查项目是否存在。
2. **文件处理：**验证 `local_file_path` 是否存在，并调用 Service 层将其转换为 AnnData 格式。
3. **持久化：**将转换后的文件保存至项目工作区的 `raw_data` 目录。
4. **元数据记录：**在 `dataset` 表中创建记录，保存物种、组织类型等生物学元数据。

- **前端触发逻辑**

1. 用户在一个 project workspace 下，点击左上角 create new dataset 按钮
2. 跳出上传 dataset 的弹窗页面

1.1.2 接口路径 (URL / Method / Version)

- **Endpoint:** /api/v1/project/dataset/import
- **Method:** POST

1.1.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	Yes	目标项目的业务 ID (Business ID)	p_20250205_ax921"
local_file_path	string	Yes	服务器磁盘上的原始文件绝对路径	"/data/upload/raw_counts.csv"
dataset_name	string	No	自定义数据集名称, 若为空则默认使用文件名	"Liver_Control_Group_1"
organism	string	No	物种信息	"Homo sapiens"
tissue_type	string	No	组织类型	"Liver"
description	string	No	数据集描述信息	"Raw counts from 10x Genomics"

1.1.4 返回参数 (Response)

参数名	类型	说明
project_id	string	项目业务 ID
project_name	string	项目显示名称
dataset_id	string	新生成的数据集业务 ID
dataset_name	string	数据集名称
workspace_path	string	项目在服务器上的根目录路径

1.1.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
404	N/A	Project {id} does not exist.	检查 project_id 是否正确
404	N/A	File not found: {path}	检查 local_file_path 路径文件是否存在
400	N/A	Failed to import dataset...	检查文件格式是否损坏或不被 Scanpy 支持
500	N/A	System Error: ...	联系后端查看日志

1.1.6 示例 (Example)

Request

```

1  {
2    "project_id": "p_20250205_ax921",
3    "local_file_path": "/tmp/uploads/sample_data.h5ad",
4    "dataset_name": "Patient_001_PBMC",
5    "organism": "Human",
6    "tissue_type": "PBMC",
7    "description": "Baseline sample before treatment"
8 }
```

Response

```

1  {
2    "project_id": "p_20250205_ax921",
3    "project_name": "Immunotherapy Study 2025",
4    "dataset_id": "ds_20250205_fw829",
5    "dataset_name": "Patient_001_PBMC",
6    "workspace_path": "/opt/lotus/projects/p_20250205_ax921"
7 }
```

1.2 用户新建 Project workspace

1.2.1 接口概述 (Purpose)

该接口用于初始化一个空的分析项目空间。它不包含任何数据，仅负责创建项目的元数据记录和对应的物理文件夹结构。

核心逻辑：

- 生成唯一的 `project_id`。
- 在文件系统中创建项目根目录。
- 在 `project_meta` 表中插入一条新记录。

前端触发逻辑：

- 页面右上角，点击新建 Project

接口路径 (URL / Method)

- Endpoint:** `/api/v1/project/create`
- Method:** POST

1.2.2 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
<code>project_name</code>	string	Yes	项目名称	"Alzheimer Analysis Phase 1"
<code>description</code>	string	No	项目描述	"Initial cohort analysis"

1.2.3 返回参数 (Response)

参数名	类型	说明
<code>project_id</code>	string	新生成的项目业务 ID
<code>project_name</code>	string	项目名称
<code>dataset_id</code>	string	固定为空字符串 "" (此时项目内无数据)
<code>dataset_name</code>	string	固定为空字符串 ""
<code>workspace_path</code>	string	新建的项目根目录路径

1.2.4 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
500	N/A	Failed to create project metadata.	数据库写入失败, 请重试
500	N/A	Unexpected error...	系统异常, 检查日志

1.2.5 示例 (Example)

Request

```

1  {
2    "project_name": "Mouse Liver Atlas",
3    "description": "Analysis of liver cell types in wild-type mice."
4 }
```

Response

```

1  {
2    "project_id": "p_20250205_zk992",
3    "project_name": "Mouse Liver Atlas",
4    "dataset_id": "",
5    "dataset_name": "",
6    "workspace_path": "/opt/lotus/projects/p_20250205_zk992"
7 }
```

1.3 访问单个子实验文件夹参数

1.3.1 接口概述 (Purpose) & 前端触发逻辑

用途: 该接口用于获取某个特定分析步骤 (快照/子实验) 的详细信息。它将快照数据结构化为 **Input** (实验入参) 和 **Output** (实验结果指标) , 并自动过滤掉文件路径 (plots, csvs 等) , 只保留对用户有意义的生物学指标 (如细胞数保留率、中位基因数、Cluster 统计等) 。

前端触发逻辑:

1. 用户在项目的左边栏中点击某个具体的实验节点 (例如 "snap 1" 或 "Clustering (Leiden 0.5)") 。
2. 右侧属性面板 (Properties Panel) 或详情弹窗加载时, 调用此接口。

3. 前端根据返回的 `input` 渲染 "Parameters" 表格，根据 `output` 渲染 "Metrics / Results" 表格。

1.3.2 接口路径 (URL / Method / Version)

- **Endpoint:** `/api/v1/project/snapshots/query/{snapshot_id}/`
- **Method:** GET

1.3.3 请求参数 (Request)

参数名	位置 (Location)	类型	必填	说明
snapshot_id	Path	string	Yes	目标快照的业务 ID (e.g., s_pre_20260207_12345) 包含在 Get 请求 路径中

1.3.4 返回参数 (Response)

- Preprocessing

代码块

```
1  {
2      "snapshot_id": "s_pre_1770538360_12669",
3      "branch_name": "Preprocessing",
4      "parent_snapshot_id": None,
5      "create_time": "2026-02-08T03:12:49.425652",
6      "user_notes": "Full pipeline execution. Steps: QC Calculation, QC Filtering, HVG Selection, PCA, Neighborhood Graph",
7      "input": {
8          "organism": "Human",
9          "custom_prefixes": null,
10         "skip_qc_calculation": false,
11         "skip_qc_filter": false,
12         "skip_hvg": false,
13         "skip_pca": false,
14         "skip_neighbors": false,
15         "min_genes": 200,
16         "cell_min_counts": null,
17         "max_genes": null,
```

```

18     "cell_max_counts": null,
19     "min_cells": 3,
20     "gene_min_counts": null,
21     "max_cells": null,
22     "gene_max_counts": null,
23     "pct_mt_max": 20.0,
24     "pct_hb_max": null,
25     "n_top_genes": 2000,
26     "flavor": "seurat",
27     "target_sum": 10000.0,
28     "n_comps": 30,
29     "svd_solver": "arpack",
30     "n_neighbors": 15,
31     "n_pcs": 20
32   },
33   "output": {
34     "summary": {
35       "n_cells_raw": 2700,
36       "n_cells_final": 2699,
37       "retention_rate": "100.0%",
38       "n_genes_final": 2000,
39       "median_genes_per_cell": 817,
40       "median_counts_per_cell": 2197
41     },
42     "qc_details": {
43       "median_pct_mt": 2.03,
44       "filter_stats": {
45         "low_quality_cells_removed": 1
46       }
47     },
48     "analysis_params": {
49       "n_hvg_selected": 2000,
50       "pcs_used": 20
51     }
52   }
53 }
54

```

• Clustering

代码块

```

1  {
2    "snapshot_id": "s_cluster_leiden_1770538770_20518",
3    "parent_snapshot_id": "s_pre_1770599279_52059",
4    "branch_name": "Clustering",

```

```
5     "create_time": "2026-02-08T03:19:33.996854",
6     "user_notes": "Clustering (leiden) performed at res=0.3. Found 5 clusters.",
7     "input": {
8         "method": "leiden",
9         "resolution": 0.3,
10        "run_hierarchical": true
11    },
12    "output": {
13        "summary": {
14            "method": "leiden",
15            "resolution": 0.3,
16            "n_clusters": 5,
17            "n_cells_total": 2699
18        }
19    }
20 }
```

- **DGE**

代码块

```
1  {
2     "snapshot_id": "s_dge_wilcoxon_1770539468_29333",
3     "parent_snapshot_id": "s_cluster_leiden_1770599288_27402",
4
5     "branch_name": "DGE",
6     "create_time": "2026-02-08T03:31:11.592758",
7     "user_notes": "DGE (wilcoxon) on 'leiden'. Identified markers for 5 groups.",
8     "input": {
9         "groupby": "leiden",
10        "method": "wilcoxon",
11        "n_top_genes": 10,
12        "use_raw": true
13    },
14    "output": {
15        "top_markers": {
16            "0": [
17                "LDHB",
18                "RPS12",
19                "RPS25",
20                "RPS27",
21                "RPS6",
22                "RPS3",
23                "TPT1",
24                "RPS14",
25                "RPS18"
26            ]
27        }
28    }
29 }
```

```
25         "CD3D",
26         "RPL31"
27     ],
28     "1": [
29         "FTL",
30         "FTH1",
31         "CST3",
32         "TYROBP",
33         "LYZ",
34         "AIF1",
35         "LST1",
36         "S100A9",
37         "OAZ1",
38         "LGALS1"
39     ],
40     "2": [
41         "NKG7",
42         "CCL5",
43         "GZMA",
44         "CST7",
45         "B2M",
46         "CTSW",
47         "HLA-C",
48         "HLA-A",
49         "PRF1",
50         "GZMM"
51     ],
52     "3": [
53         "CD74",
54         "CD79A",
55         "HLA-DRA",
56         "CD79B",
57         "HLA-DPB1",
58         "HLA-DQA1",
59         "MS4A1",
60         "HLA-DRB1",
61         "HLA-DQB1",
62         "HLA-DPA1"
63     ],
64     "4": [
65         "HLA-DPA1",
66         "HLA-DPB1",
67         "HLA-DRB1",
68         "HLA-DRA",
69         "CST3",
70         "CD74",
71         "HLA-DRB5",
```

```
72         "HLA-DQA1",
73         "FCER1A",
74         "HLA-DQB1"
75     ],
76 }
77 }
78 }
```

- **Annotation**

代码块

```
1  {
2      "snapshot_id": "annot_1770543631_13490",
3      "parent_snapshot_id": "s_dge_wilcoxon_1770599292_97740",
4      "branch_name": "Annotation",
5      "create_time": "2026-02-08T04:40:44.939741",
6      "user_notes": "Combined Annotation: GSEAp(2), CellTypist(3)",
7      "input": {
8          "top_n_genes": 50,
9          "categories": [
10              "CellMarker_2024",
11              "BioPlanet_2019"
12          ],
13          "model_names": [
14              "Immune_All_Low.pkl",
15              "Adult_Mouse_Gut.pkl",
16              "Developing_Human_Brain.pkl"
17          ],
18          "majority_voting": true,
19          "target_cluster_col": "leiden"
20      },
21      "output": {
22          "gseapy_CellMarker_2024": {
23              "0": {
24                  "predicted_cell_type": "Goblet Cell Intestinal Crypt Mouse",
25                  "average_confidence": 0.4683,
26                  "color": "#2ca02c"
27              },
28              "1": {
29                  "predicted_cell_type": "Monocyte Spleen Human",
30                  "average_confidence": 0.7137,
31                  "color": "#d62728"
32              }
33          }
34      }
35  }
```

```
32     },
33     "2": {
34         "predicted_cell_type": "CD8+ T Cell Blood Human",
35         "average_confidence": 0.8557,
36         "color": "#ff7f0e"
37     },
38     "3": {
39         "predicted_cell_type": "B Cell Brain Human",
40         "average_confidence": 0.8254,
41         "color": "#1f77b4"
42     },
43     "4": {
44         "predicted_cell_type": "Secretory Cell Lung Human",
45         "average_confidence": 0.765,
46         "color": "#9467bd"
47     }
48 },
49 "gseapy_BioPlanet_2019": {
50     "0": {
51         "predicted_cell_type": "Cytoplasmic ribosomal proteins",
52         "average_confidence": 0.8351,
53         "color": "#ff7f0e"
54     },
55     "1": {
56         "predicted_cell_type": "Iron uptake and transport",
57         "average_confidence": 0.2455,
58         "color": "#d62728"
59     },
60     "2": {
61         "predicted_cell_type": "Endosomal/vacuolar pathway",
62         "average_confidence": 0.5544,
63         "color": "#2ca02c"
64     },
65     "3": {
66         "predicted_cell_type": "Asthma",
67         "average_confidence": 0.8761,
68         "color": "#1f77b4"
69     },
70     "4": {
71         "predicted_cell_type": "Asthma",
72         "average_confidence": 0.9619,
73         "color": "#1f77b4"
74     }
75 },
76 "celltypist_immune_all_low": {
77     "0": {
78         "predicted_cell_type": "Tcm/Naive helper T cells",
```

```
79         "average_confidence":0.7535,
80         "color": "#aec7e8"
81     },
82     "1": {
83         "predicted_cell_type": "Classical monocytes",
84         "average_confidence": 0.9074,
85         "color": "#279e68"
86     },
87     "2": {
88         "predicted_cell_type": "CD16+ NK cells",
89         "average_confidence": 0.7019,
90         "color": "#ff7f0e"
91     },
92     "3": {
93         "predicted_cell_type": "B cells",
94         "average_confidence": 0.9113,
95         "color": "#1f77b4"
96     },
97     "4": {
98         "predicted_cell_type": "DC",
99         "average_confidence": 0.6892,
100        "color": "#d62728"
101    }
102},
103 "celltypist_adult_mouse_gut": {
104     "0": {
105         "predicted_cell_type": "B cell",
106         "average_confidence": 0.0058,
107         "color": "#1f77b4"
108     },
109     "1": {
110         "predicted_cell_type": "B cell",
111         "average_confidence": 0.0058,
112         "color": "#1f77b4"
113     },
114     "2": {
115         "predicted_cell_type": "B cell",
116         "average_confidence": 0.0058,
117         "color": "#1f77b4"
118     },
119     "3": {
120         "predicted_cell_type": "B cell",
121         "average_confidence": 0.0058,
122         "color": "#1f77b4"
123     },
124     "4": {
125         "predicted_cell_type": "B cell",
```

```
126         "average_confidence":0.0058,
127         "color":"#1f77b4"
128     }
129 },
130 "celltypist_developing_human_brain":{
131     "0":{
132         "predicted_cell_type":"Medulla immune cells",
133         "average_confidence":0.0443,
134         "color":"#e377c2"
135     },
136     "1":{
137         "predicted_cell_type":"Forebrain immune cells",
138         "average_confidence":0.1723,
139         "color":"#2ca02c"
140     },
141     "2":{
142         "predicted_cell_type":"Hypothalamus immune cells",
143         "average_confidence":0.1044,
144         "color":"#9467bd"
145     },
146     "3":{
147         "predicted_cell_type":"Head immune cells",
148         "average_confidence":0.0453,
149         "color":"#d62728"
150     },
151     "4":{
152         "predicted_cell_type":"Medulla immune cells",
153         "average_confidence":0.0936,
154         "color":"#e377c2"
155     }
156 }
157 }
158 }
```

1.3.5 错误码 (Error Codes)

HTTP Code	错误信息 (Detail)	说明	
404	"No snapshot found for dataset {id}"	找不到 snapshot id	
500	"Failed to query latest snapshot: ..."	服务器内部错误。	

1.4 获取某个阶段 (eg preprocessing) 最新的实验参数展示

1.4.1 接口概述 (Purpose) & 前端触发逻辑

用途：该接口用于查询指定数据集在某个特定分析阶段 (Branch/Stage) 的最新一次实验快照详情。它主要用于“阶段文件夹”级别的展示。当用户不需要查看历史记录，只想看“Preprocessing”或“Clustering”当前最新的状态（参数和结果指标）时使用。

前端触发逻辑：

1. 用户在左侧实验树状图中，点击 **父级阶段文件夹**（例如 "Preprocessing" 文件夹，而不是具体的某个 `s_pre_xxx` 快照节点）。
2. 前端调用此接口，传入 `dataset_id` 和 `branch_name`（如 "Preprocessing"）。
3. 后端自动查找该分支下时间最近的一个 Snapshot，并返回其 `input`（参数）和 `output`（生物学指标）。

前端触发逻辑：

1. 用户在左侧项目栏中，点击 **父级阶段文件夹**（例如 "Preprocessing" 文件夹，而不是具体的某个 `s_pre_xxx` 快照节点）。
2. 前端调用此接口，传入 `dataset_id` 和 `branch_name`（如 "Preprocessing"）。
3. 后端自动查找该分支下时间最近的一个 Snapshot，并返回其 `input`（参数）和 `output`（生物学指标）。

1.4.2 接口路径 (URL / Method / Version)

- **Endpoint:** /api/v1/snapshots/stage/query/
- **Method:** GET

1.4.3 请求参数 (Request)

参数名	位置 (Location)	类型	必填	说明	示例
dataset_id	Query	string	Yes	数据集业务 ID	ds_20260207_rw
branch_name	Query	string	Yes	分析阶段名称 (Branch Name)	Preprocessing, Clustering, DEG

branch_name 枚举值 (需要大小写完全一致)

代码块

```
1 "Preprocessing"  
2 "Clustering"  
3 "DGE"  
4 "Annotation"
```

1.4.4 返回参数 (Response)

同 “访问单个子实验文件夹参数” 接口

1.4.5 错误码 (Error Codes)

HTTP Code	错误信息 (Detail)	说明
404	"No snapshot found for dataset {id} on branch '{name}'"	该数据集在指定阶段下没有任何实验记录。
422	"Validation Error"	缺少必填参数 (dataset_id 或 branch_name)。
500	"Failed to query latest snapshot: ..."	服务器内部错误。

1.4.6 示例 (Example)

Request

```
1 GET /api/v1/snapshots/stage/query/?  
dataset_id=ds_20260208_001&branch_name=Preprocessing
```

Response

```
1
```

2. 主分析流程 Pipeline (PP, Clustering, DGE, annotation)

2.1 Run Whole Pipeline

2.1.1 接口概述 (Purpose)

该接口用于执行端到端的单细胞分析全流程。它采用嵌套 Request 模式：顶层 preprocessing_params、clustering_params 等字段实际上直接复用了各个子服务的请求

结构。注意：虽然子 Request 定义中包含 `project_id` 和 `dataset_id`，但在调用此全流程接口时，前端无需在内层参数中重复填写这些 ID，后端会自动将顶层的 ID 回填到子对象中。

核心逻辑：

1. **Full Preprocessing:** 集成 QC (质量控制)、Filtering (过滤)、HVG (高变基因)、PCA (降维) 和 Neighbors (邻居图) 为一步，支持参数化配置及部分步骤跳过。
2. **Clustering:** 执行 Leiden 或 Louvain 聚类。
3. **DEG:** 差异基因表达分析。
4. **Annotation:** 执行 GSEApY 功能富集及 CellTypist 自动细胞类型注释。

前端触发逻辑：

Customize Analysis

QC Filtering HVG Selection PCA Analysis Neighbors Graph Clustering DGE Analysis Annotation

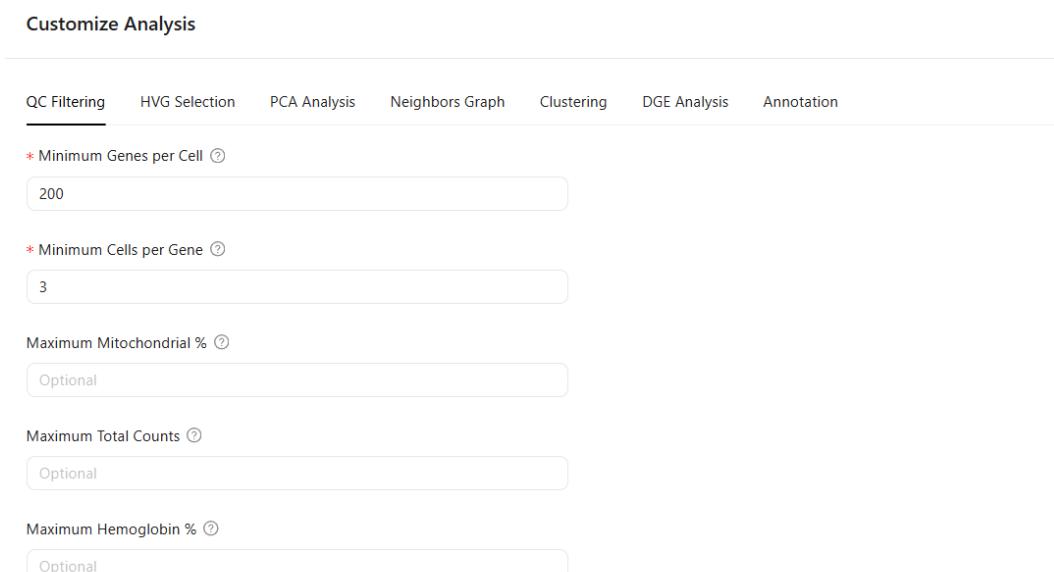
* Minimum Genes per Cell ⑦
200

* Minimum Cells per Gene ⑦
3

Maximum Mitochondrial % ⑦
Optional

Maximum Total Counts ⑦
Optional

Maximum Hemoglobin % ⑦
Optional



2.1.2 接口路径 (URL / Method / Version)

URL: `/api/v1/pipeline/run`

Method: `POST`

Content-Type: `application/json`

Version: `v1`

2.1.3 请求参数 (Request)

参数名	类型	必填	说明
project_id	string	Yes	项目业务 ID
dataset_id	string	Yes	数据集业务 ID
preprocessing_params	Object	No	预处理参数配置 (对应 FullPreprocessingRequest)
clustering_params	Object	Yes	聚类参数 (对应 RunClusteringRequest)
deg_params	Object	No	差异分析参数 (对应 RunDGERequest)
annotation_params	Object	No	统一注释参数 (对应 RunFullAnnotationRequest)，包含 GSEAp 和 CellTypist

2.1.3.1 Preprocessing (preprocessing_params)

参数名	类型	默认值	说明
流程控制 (Skip Flags)			
skip_qc_calculation	bool	FALSE	是否跳过 QC 指标计算
skip_qc_filter	bool	FALSE	是否跳过 QC 过滤
skip_hvg	bool	FALSE	是否跳过高变基因选择
skip_pca	bool	FALSE	是否跳过 PCA 降维
skip_neighbors	bool	FALSE	是否跳过邻居图构建
QC & Filtering			
organism	str	"Human"	物种 (e.g. "Human", "Mouse")
min_genes	int	200	过滤阈值：每个细胞最小基因数
min_cells	int	3	过滤阈值：每个基因最小细胞数
pct_mt_max	float	20	过滤阈值：最大线粒体比例 (%)
max_counts	int	50000	过滤阈值：最大 UMI 计数
HVG			
n_top_genes	int	2000	高变基因数量
flavor	str	"seurat"	算法风格 (seurat, cell_ranger)
target_sum	float	10000	归一化目标总数
PCA & Neighbors			
n_comps	int	50	PCA 主成分数量
svd_solver	str	"arpack"	SVD 求解器
n_neighbors	int	15	邻居数量 (k)
n_pcs	int	30	用于邻居图的 PC 数量

2.1.3.2 clustering_params (聚类 - Core)

参数名	类型	默认值	必填	说明
method	str	"leiden"	No	聚类算法 (leiden, louvain)
resolution	float	0.5	Yes	分辨率列表，如 [0.5, 1.0]
run_hierarchical	bool	TRUE	No	是否计算树状图 (Dendrogram)

2.1.3.3 deg_params (DGE)

参数名	类型	默认值	说明
method	str	"wilcoxon" 补枚举值	检验方法
n_top_genes	int	50 ()	每个 Cluster 提取多少个 Marker Gene
use_raw	bool	TRUE	
groupby	str	leiden	cluster 来做dge

2.1.3.4 annotation_params (GSEApY)

参数名	类型	默认值	说明
GSEApY (Enrichr)			
categories	List[str]	["cellmarker"]	基因集数据库列表。 传空列表或 null 则跳过 GSEApY。
top_n_genes	int	100	每个簇提取多少个基因用于富集分析
CellTypist			
model_names	List[str]	["Immune_All_Low.pkl"]	模型文件列表。 传空列表或 null 则跳过 CellTypist。
majority_voting	bool	TRUE	是否开启多数投票修正
target_cluster_col	str	"leiden"	用于多数投票的聚类列名

2.1.4 返回参数 (Response)

参数名	类型	说明
pipeline_run_id	str	此次全流程运行的唯一 ID
final_snapshot_id	str	最终状态（包含注释结果）的快照 ID
steps_completed	List[str]	已完成步骤的列表 (e.g. ["QC", "HVG", ...])
snapshots	Dict[str, str]	关键步骤对应的快照 ID 映射表
status	str	"SUCCESS"

2.1.5 错误码 (Error Codes)

HTTP Code	Error Code	Description
400	PARAM_ERROR	参数错误 (如 resolutions 为空)
404	PROJECT_NOT_FOUND	Project ID 或 Dataset ID 不存在
500	PIPELINE_FAILED	流程执行中途失败 (如内存溢出, 算法收敛失败)

2.1.6 示例 (Example)

Request

```

1   {
2     "project_id": "p_20260206_demo_01",
3     "dataset_id": "ds_20260206_raw_data",
4
5     "preprocessing_params": {
6       "skip_qc_calculation": false,
7       "skip_qc_filter": false,
8       "organism": "Human",
9       "min_genes": 300,
10      "min_cells": 5,
11      "pct_mt_max": 15.0,
12      "max_counts": 40000,
13
14      "skip_hvg": false,
15      "n_top_genes": 2500,
16      "flavor": "seurat",
17      "target_sum": 10000.0,

```

```

18     "skip_pca": false,
19     "n_comps": 50,
20     "svd_solver": "arpack",
21
22     "skip_neighbors": false,
23     "n_neighbors": 20,
24     "n_pcs": 30
25   },
26
27
28   "clustering_params": {
29     "method": "leiden",
30     "resolutions": [0.3, 0.5, 0.8, 1.0, 1.5],
31     "primary_resolution": 0.5,
32     "run_hierarchical": true
33   },
34
35   "deg_params": {
36     "method": "wilcoxon",
37     "n_top_genes": 100
38   },
39
40   "annotation_params": {
41     "categories": [
42       "CellMarker_2024",
43       "BioPlanet_2019",
44       "NCI-Nature_2016",
45       "GO_Biological_Process_2023"
46     ],
47     "top_n_genes": 50,
48     "model_names": ["Immune_All_Low.pkl"],
49     "majority_voting": true,
50     "target_cluster_col": "leiden"
51   }
52 }
```

Response

```

1  {
2    "pipeline_run_id": "run_pipe_20260206123045_12345",
3    "final_snapshot_id": "s_annot_ct_20260206123510_99887",
4    "status": "SUCCESS",
5    "msg": "Whole pipeline executed successfully.",
6    "steps_completed": [
```

```

7     "QC Calculation",
8     "QC Filtering",
9     "HVG Selection",
10    "PCA",
11    "Neighborhood Graph",
12    "Clustering",
13    "DEG",
14    "Annotation (GSEApY)",
15    "Annotation (CellTypist)"
16  ],
17  "snapshots": {
18    "preprocessing": "s_pre_20260206123100_11223",
19    "clustering": "s_cluster_leiden_20260206123200_33445",
20    "deg": "s_deg_20260206123300_55667",
21    "annotation_gseapy": "s_annotation_gsea_20260206123400_77889",
22    "annotation_celltypist": "s_annotation_ct_20260206123510_99887"
23  }
24 }

```

2.2 预处理 (Preprocessing) 统一大接口

2.2.1 接口概述 (Purpose) & 前端触发逻辑

该接口用于一次性执行完整的单细胞数据预处理流程。为了提高效率，所有计算步骤（QC计算 -> 过滤 -> 高变基因 -> PCA -> 邻域图）均在内存中连续执行，**不保存中间产物**，仅在最后保存一个包含所有处理结果的 Snapshot（快照）。

前端触发逻辑： 用户在 project id -> dataset id folder -> preprocessing 数据详情页点击 “**Run New Preprocessing**” 按钮。前端允许用户在一个表单中配置所有参数（如过滤阈值、PCA组件数等），或者使用默认参数，然后一次性提交请求。

2.2.2 接口路径 (URL / Method / Version)

- **Endpoint:** /api/v1/preprocessing/run/full
- **Method:** POST

2.2.3 请求参数 (Request)

参数名	类型	必填	说明	默认值/示例
核心定位参数				
project_id	string	是	项目业务 ID	"p_20250205_ax921"
dataset_id	string	是	数据集业务 ID	"ds_20250205_fw829"
QC 计算参数				
organism	string	否	物种 (Human/Mouse)	"Human"
custom_prefixes	dict	否	自定义基因前缀	null
skip_qc_calculation	bool	否	跳过 QC 计算	FALSE
QC 过滤参数 (Cell级别)				
min_genes	int	否	单细胞最小基因数	200
cell_min_counts	int	否	单细胞最小 Total Counts	null
max_genes	int	否	单细胞最大基因数	null
cell_max_counts	int	否	单细胞最大 Total Counts	null
QC 过滤参数 (Gene级别)				
min_cells	int	否	基因最少在多少细胞表达	3
gene_min_counts	int	否	基因最小总 Counts	null
max_cells	int	否	基因最多在多少细胞表达	null
gene_max_counts	int	否	基因最大总 Counts	null
QC 过滤参数 (指标级别)				
pct_mt_max	float	否	线粒体最大比例 (0-100)	null
pct_hb_max	float	否	血红蛋白最大比例 (0-100)	null
skip_qc_filter	bool	否	跳过过滤步骤	FALSE
HVG & Scaling 参数				
n_top_genes	int	否	高变基因数量 (500-10000)	2000
flavor	string	否	算法 (seurat, cell_ranger, seurat_v3)	"seurat"
target_sum	float	否	归一化目标总数	10000
skip_hvg	bool	否	跳过 HVG 选择	FALSE
PCA 参数				

n_comps	int	否	PCA 主分数量	50
svd_solver	string	否	SVD 求解器 (arpack)	"arpack"
skip_pca	bool	否	跳过 PCA 计算	FALSE
Neighbors 参数				
n_neighbors	int	否	邻居数量 (K值)	15
n_pcs	int	否	建图使用的 PC 数量	30
skip_neighbors	bool	否	跳过邻域图构建	FALSE

2.2.4 返回参数 (Response)

参数名	类型	说明
project_id	string	项目 ID
dataset_id	string	数据集 ID
final_stage	string	流程最终完成的阶段 (e.g., "Neighborhood Graph")
executed_steps	list[str]	实际执行了哪些步骤的列表
skipped_steps	list[str]	跳过了哪些步骤的列表
message	string	成功消息提示
snapshot_id	string	新生成的快照 ID (包含合并后的结果)
snapshot_path	string	新生成的 .h5ad 文件路径

2.2.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
404	DATASET_NOT_FOUND	Dataset {id} not found.	检查 dataset_id 是否正确或文件是否存在。
500	PIPELINE_ERR OR	PIPELINE_ERR OR: Failed to save...	IO 错误或算法计算崩溃 (如内存不足) , 检查后端日志。
500	SYSTEM_ERR OR	SYSTEM_ERR OR: {details}	未知系统异常。

2.2.6 示例 (Example)

Request

```

1  {
2      "project_id": "p_20250205_ax921",

```

```
3     "dataset_id": "ds_20250205_fw829",
4
5     // 基础 QC 配置
6     "organism": "Human",
7
8     // 细胞过滤条件
9     "min_genes": 500,
10    "max_genes": 6000,
11    "cell_min_counts": 1000,
12    "cell_max_counts": 50000,
13    "pct_mt_max": 15.0,
14    "pct_hb_max": 5.0,
15
16    // 基因过滤条件
17    "min_cells": 5,
18    "gene_min_counts": 10,
19
20    // 归一化与高变基因
21    "target_sum": 10000.0,
22    "n_top_genes": 2500,
23    "flavor": "seurat",
24
25    // 降维配置
26    "n_comps": 40,
27    "svd_solver": "arpack",
28
29    // 邻域图配置
30    "n_neighbors": 20,
31    "n_pcs": 30,
32
33    // 流程控制 (默认 false, 按需开启)
34    "skip_qc_calculation": false,
35    "skip_qc_filter": false,
36    "skip_hvg": false,
37    "skip_pca": false,
38    "skip_neighbors": false
39 }
```

Response

1

2.3 新建聚类 (New Clustering)

2.3.1 接口概述 (Purpose) & 前端触发逻辑

该接口用于对经过预处理的单细胞数据执行聚类分析 (Clustering)。支持多种聚类算法 (Leiden, Louvain, cplearn)，并自动计算 UMAP 降维以便于可视化。

- **Purpose:** 识别细胞亚群 (Cluster)，生成 UMAP 可视化图，并在特定条件下生成层级聚类 (Dendrogram)。
- **前端触发逻辑:** 用户在分析流程中完成预处理 (QC/PCA/Neighbors) 后，点击 project -> dataset -> clustering 页面的“Run New Clustering”按钮。前端需提供聚类算法、分辨率 (Resolution) 等参数。

2.3.2 接口路径 (URL / Method / Version)

URL: /api/v1/clustering/create

Method: POST

2.3.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	是	项目业务 ID	"p_20250205_ax921"
dataset_id	string	是	数据集业务 ID	"ds_20250205_fw829"
snapshot_id	string	否	上游 Snapshot ID (通常是 Preprocessing 结果)。若为空，后端会自动查找最新的预处理结果。	"snap_pre..."
method	string	否	聚类算法。可选值: 'leiden', 'louvain', 'cplearn'。	"leiden"
resolution	float	否	聚类分辨率。值越大，生成的 Cluster 越多；值越小，Cluster 越少。	0.5
run_hierarchical	boolean	否	是否计算层级聚类 (树状图)。建议仅在 Cluster 数量适中时开启。	TRUE

2.3.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	本次聚类分析生成的 Snapshot ID
snapshot_path	string	结果文件 (.h5ad) 在服务器上的存储路径
umap_plot_path	string	UMAP 聚类可视化图的路径
dendrogram_path	string	层级聚类树状图的路径 (若 run_hierarchical=false 或计算失败则为 null)
clusters_summary	object	聚类结果摘要 (key: 算法名, value: 发现的 Cluster 数量)
msg	string	操作成功提示信息

2.3.5 错误码 (Error Codes)

HTTP Code	Error Code	说明	解决方案
400	BAD_REQUEST	参数错误或前置条件未满足	检查 resolution 是否为空, 或确认是否已运行 Neighbors 建图步骤。
404	NOT_FOUND	上游 Snapshot 未找到	检查 project_id/dataset_id 是否正确, 或确认预处理是否成功完成。
500	CLUSTERING_ERROR	聚类计算失败	通常是数据问题 (如无有效邻域图) 或算法异常, 需查看后端日志。

2.3.6 示例 (Example)

Request:

```

1  {
2      "project_id": "p_20250205_ax921",
3      "dataset_id": "ds_20250205_fw829",
4      "method": "leiden",
5      "resolution": 0.8,
6      "run_hierarchical": true
7  }

```

Response:

```
1  {
2    "snapshot_id": "snap_cluster_20250205_ab123",
3    "snapshot_path":
4      "projects/p_20250205_ax921/snapshots_anndata/snap_cluster_20250205_ab123.h5ad",
5    "umap_plot_path":
6      "projects/p_20250205_ax921/ds_20250205_fw829/clustering/snap_cluster_.../umap_c
luster.png",
7    "dendrogram_path":
8      "projects/p_20250205_ax921/ds_20250205_fw829/clustering/snap_cluster_.../dendro
gram.png",
9    "clusters_summary": {
10      "leiden": 12
11    },
12    "msg": "Clustering and UMAP projection complete."
13  }
```

2.4 Merge Multiple clusters

2.4.1 接口概述 (Purpose) & 前端触发逻辑

用途：该接口用于手动合并 (Merge) 多个特定的簇 (Clusters) 为一个新的群体。它通常用于“人工校正”阶段，当研究人员通过生物学知识判断某些被算法分开的簇实际上属于同一种细胞类型时调用。

逻辑流程：

- 加载数据：**根据 `snapshot_id` 加载之前的聚类结果。
- 合并标签：**将 `clusters_to_merge` 列表中指定的所有簇 ID，在 `method` (如 `leiden`) 列中统一重命名为 `new_label`。
- 更新可视化：**基于合并后的新分组，重新生成 UMAP 图和图例。

前端触发场景：

- 交互式合并：**用户在 UMAP 可视化界面上选中多个簇 (例如 Cluster 3 和 Cluster 5)，点击“Merge Clusters”按钮。
- 重命名/归并：**用户认为 Cluster 0, 1, 2 都是 "T-cells"，手动输入这些 ID 并指定新名称 "T-cells"。

2.4.2 接口路径 (URL / Method / Version)

URL: `/api/v1/clustering/merge`

Method: POST

2.4.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	Yes	项目业务 ID	"p_20230101_1234"
dataset_id	string	Yes	数据集业务 ID	"ds_20230101_5678"
snapshot_id	string	Yes	源聚类快照 ID。即想要修改的那个聚类结果的 ID。	"s_cluster_leiden_2023..."
method	string	Yes	数据中存储聚类结果的列名，通常是之前的算法名。	"leiden", "louvain"
clusters_to_merge	List	Yes	需要合并的簇 ID 列表。支持字符串或数字。	["1", "4"] 或 [1, 4]
new_label	string	No	合并后的新组名。如果不传，后端可能会生成默认名称（如 "Merged_Group"）。	"T-Cells", "Merged_1_4"

2.4.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	新生成的快照 ID (包含合并后的结果)
snapshot_path	string	新生成的 .h5ad 文件路径
umap_plot_path	string	更新后的 UMAP 静态图路径 (反映了合并后的图例)
dendrogram_path	string	(可选) 更新后的树状图路径
clusters_summary	Dict	聚类统计摘要 (e.g. {"leiden": 5}), 显示剩余簇的数量
msg	string	操作结果消息 (e.g. "Clusters merged successfully.")

2.4.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
400	PARAM_ERROR	"Cluster ID 'X' not found in data"	检查 clusters_to_merge 中的 ID 是否确实存在于当前数据的聚类结果中。
400	PARAM_ERROR	"Method 'leiden' not found"	检查 method 参数是否拼写错误，或者该快照是否未运行过该聚类算法。
404	SNAPSHOT_NOT_FOUND	"Source snapshot not found"	检查 snapshot_id 是否正确，或该快照文件是否已被删除。
500	MERGE_ERROR	"Unexpected error..."	服务器内部错误，请查看日志。

2.4.6 示例 (Example)

Request

```

1  {
2    "project_id": "p_20260205124022_58920",
3    "dataset_id": "ds_20260205124031_48692",
4    "snapshot_id": "s_cluster_leiden_1770418367_26821",
5    "method": "leiden",
6    "clusters_to_merge": ["3", "5", "8"],
7    "new_label": "Macrophages"
8  }

```

Response

```

1  {
2    "snapshot_id": "s_cluster_merge_20260206153000_12345",
3    "snapshot_path": "projects/.../s_cluster_merge_... .h5ad",
4    "umap_plot_path": "projects/.../umap_cluster.png",
5    "dendrogram_path": null,
6    "clusters_summary": {
7      "leiden": 12
8    },
9    "msg": "Clusters merged. New group 'Macrophages' created."
10 }

```

2.5 Continue clustering on sub cluster (继续分簇)

2.5.1 接口概述 (Purpose) & 前端触发逻辑

2.5.2 接口路径 (URL / Method / Version)

- Endpoint:
- Method:

2.5.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例

2.5.4 返回参数 (Response)

参数名	类型	说明

2.5.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案

2.5.6 示例 (Example)

Request

1

Response

1

2.6 差异表达基因分析 (DGE Analysis)

2.6.1 接口概述 (Purpose) & 前端触发逻辑

该接口用于执行差异基因表达分析 (Differential Gene Expression Analysis)。它通过统计检验方法 (如 Wilcoxon 或 t-test) 找出在特定分组 (如聚类结果) 中显著高表达的标记基因 (Marker Genes)，并生成用于细胞类型注释的统计表和可视化图表。

- Purpose:** 识别每个 Cluster 的特异性标记基因，生成统计表格 (CSV) 及可视化图表 (Rank Plot, Dot Plot, Heatmap, Violin Plot)。
- 前端触发逻辑:** 用户在完成聚类分析后，进入 “DGE” 模块，选择上个 clustering 实验结果 id 并点击 “Run new DGE”。

2.6.2 接口路径 (URL / Method / Version)

URL: /api/v1/dge/new

Method: POST

2.6.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	是	项目业务 ID	"p_20250205_ax921"
dataset_id	string	是	数据集业务 ID	"ds_20250205_fw829"
snapshot_id	string	否	上游 Snapshot ID (通常是 Clustering 结果)。若为空，后端会自动查找最新的聚类结果。	"snap_cluster_.."
groupby	string	否	用于分组的列名 (在 adata.obs 中)，聚类结果列名。	"leiden", "louvain", "cplearn"
method	string	否	统计检验方法。可选值: 'wilcoxon', 't-test', 'logreg'。	"wilcoxon"
n_top_genes	int	否	每个组要在图中展示的 Top 基因数量 (范围 5-100)。	25
use_raw	boolean	否	是否使用原始数据 (adata.raw) 进行计算。推荐为 true 以避免归一化偏差。	TRUE

2.6.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	本次 DGE 分析生成的 Snapshot ID
snapshot_path	string	结果文件 (.h5ad) 在服务器上的存储路径
csv_path	string	包含完整统计结果 (LogFoldChange, P-values 等) 的 CSV 文件下载路径
rank_genes_plot	string	基因排名可视化图路径 (Rank Plot)
dotplot	string	气泡图路径 (Dot Plot)
heatmap	string	热图路径 (Heatmap)
violin	string	小提琴图路径 (Violin Plot)
top_markers	object	简单的 Marker 预览 (key: 组名, value: Top 5 基因名列表)。 例如: {"0": ["GeneA", "GeneB"], ...}
msg	string	操作成功提示信息

2.6.5 错误码 (Error Codes)

HTTP Code	Error Code	说明	解决方案
400	BAD_REQUEST	参数错误或分组无效	检查 groupby 列是否存在, 或确认该列是否至少包含 2 个分组。
404	NOT_FOUND	上游 Snapshot 未找到	检查 project_id/ dataset_id 是否正确, 或确认聚类步骤是否已完成。
500	CALCULATION_ERROR	计算失败	通常由于数据稀疏、矩阵奇异或内存不足导致, 需查看后端日志。

2.6.6 示例 (Example)

Request

```

1  {
2    "project_id": "p_20250205_ax921",
3    "dataset_id": "ds_20250205_fw829",
4    "groupby": "leiden",
5    "method": "wilcoxon",
6    "n_top_genes": 20,
7    "use_raw": true
8  }

```

Response

```
1  {
2      "snapshot_id": "snap_dge_wilcoxon_20250205_xy998",
3      "snapshot_path": "projects/p_20250205_ax921/snapshots_anndata/snap_dge_wilcoxon_20250205_xy998.h5ad",
4      "csv_path": "projects/p_20250205_ax921/ds_20250205_fw829/differential_gene_expression/snap_dge_.../markers_all.csv",
5      "rank_genes_plot": "projects/p_20250205_ax921/ds_20250205_fw829/differential_gene_expression/snap_dge_.../dge_rank.png",
6      "dotplot": "projects/p_20250205_ax921/ds_20250205_fw829/differential_gene_expression/snap_dge_.../dge_dotplot.png",
7      "heatmap": "projects/p_20250205_ax921/ds_20250205_fw829/differential_gene_expression/snap_dge_.../dge_heatmap.png",
8      "violin": "projects/p_20250205_ax921/ds_20250205_fw829/differential_gene_expression/snap_dge_.../dge_violin.png",
9      "top_markers": {
10          "0": ["CD3D", "IL7R", "LDHB", "CCR7", "CD3E"],
11          "1": ["CD79A", "MS4A1", "CD79B", "HLA-DRA", "LYZ"]
12      },
13      "msg": "Differential expression analysis complete."
14 }
```

2.7 Annotation 统一接口 (Run Full Annotation Pipeline)

2.7.1 接口概述 (Purpose) & 前端触发逻辑

用途：该接口是生物学注释的统一入口。它支持在一个请求中同时执行（或单独执行）**功能富集分析 (GSEApY)** 和 **自动细胞类型注释 (CellTypist)**。核心特性是**统一输出**：无论运行多少个数据库或模型，所有结果（Label 信息、P-value、统计数据）都聚合在一个 JSON 报告中，并且每个数据库/模型都会生成一张对应的 UMAP 静态图。

前端触发逻辑：

1. 用户在 "Annotation" 步骤页面，看到两个面板：

- **Enrichr Databases:** 复选框 (e.g., CellMarker, GO Biological Process) 。
 - **CellTypist Models:** 复选框 (e.g., Immune_All_Low, Human_Lung_Atlas) 。
2. 用户勾选所需的数据库和模型 (至少勾选一项) 。
3. 点击 "Run Annotation" 按钮调用此接口。
4. 前端收到响应后，遍历 `result_keys`，使用 key 既可以从 JSON 报告中获取详细数据，也可以拼接 `.png` 后缀直接加载对应的 UMAP 图片。

2.7.2 接口路径 (URL / Method / Version)

- **Endpoint:** /api/v1/annotation/full
- **Method:** POST

2.7.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	Yes	项目业务 ID	"p_20260206_01"
dataset_id	string	Yes	数据集业务 ID	"ds_20260206_raw"
snapshot_id	string	No	上一步分析 (DEG 或 Clustering) 的快照 ID。	"s_deg..."
GSEApY 参数				
categories	List[str]	No	Enrichr 数据库列表。传空列表或 null 则跳过 GSEApY 流程。	["CellMarker_2024", "BioPlanet_2019"]
top_n_genes	int	No	每个簇提取多少个 Marker 基因用于富集。	100
CellTypist 参数				
model_names	List[str]	No	CellTypist 模型文件列表。传空列表或 null 则跳过 CellTypist 流程。	["Immune_All_Low.pkl"]
majority_voting	bool	No	是否开启多数投票修正（利用聚类信息优化预测）。	TRUE
target_cluster_col	string	No	用于多数投票的聚类列名（需存在于 obs 中）。	"leiden"

2.7.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	本次全流程注释生成的唯一快照 ID。
snapshot_path	string	生成的 .h5ad 文件路径。
cluster_id	string	固定为 "ALL"。
msg	string	成功消息摘要 (e.g., "Full annotation completed: GSEAp(y(2 DBs), CellTypist(1 Models))")。
enrichment_results	Object	核心结果对象 , 包含前端展示所需的所有元数据路径和 Key 列表。
└ report_path	string	统一 JSON 报告文件路径 。包含所有数据库/模型的详细结果。
└ result_keys	List[str]	结果 Key 列表 。例如 ["gseapy_CellMarker_2024", "celltypist_Immune_All_Low"]。前端可使用这些 Key 在 Report 中查找数据，并拼接 .png 加载图片。
└ plots	List[str]	生成的所有图片文件的相对路径列表（用于缩略图展示或验证）。

2.7.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
400	PARAM_ERROR	"No annotation tasks selected. Please provide 'categories' or 'model_names'."	请求中 categories 和 model_names 全为空。必须至少选择一个任务。
400	PREREQUISITE_MISSING	"GSEAp(y requested but 'rank_genes_groups' not found."	选择了 GSEAp(y 但源快照中没有 DEG 差异分析数据。请先运行 DEG 分析。
500	PIPELINE_FAILED	"Annotation pipeline failed: [Error Detail]"	服务器内部错误 (如 Enrichr API 超时、模型加载失败)。

2.7.6 示例 (Example)

Request

```

1  {
2      "project_id": "p_20260207_8899",
3      "dataset_id": "ds_20260207_raw",

```

```

4   "snapshot_id": "s_deg_wilcoxon_20260207_12345",
5
6   "categories": [
7     "CellMarker_2024",
8     "GO_Biological_Process_2023"
9   ],
10  "top_n_genes": 50,
11
12  "model_names": [
13    "Immune_All_Low.pkl"
14  ],
15  "majority_voting": true,
16  "target_cluster_col": "leiden"
17 }

```

Response

```

1  {
2    "snapshot_id": "s_annot_full_20260207_998877",
3    "snapshot_path": "projects/.../snapshots/s_annot_full_20260207_998877.h5ad",
4    "cluster_id": "ALL",
5    "msg": "Full annotation completed: GSEAp(2 DBs), CellTypist(1 Models)",
6    "enrichment_results": {
7      "report_path":
8        "projects/p_.../annotation/s_annot_full_.../s_annot_full_..._full_report.json",
9
10     "result_keys": [
11       "gseapy_CellMarker_2024",
12       "gseapy_GO_Biological_Process_2023",
13       "celltypist_immune_all_low"
14     ],
15
16     "plots": [
17       "projects/.../annotation/.../gseapy_CellMarker_2024.png",
18       "projects/.../annotation/.../gseapy_GO_Biological_Process_2023.png",
19       "projects/.../annotation/.../celltypist_immune_all_low.png"
20     ]
21   }
22 }

```

2.8 回All Type & Cell typist 和 GSEAPY 关键词搜索

2.8.1 接口概述（Purpose）

用于查询系统支持的注释参考方法（Annotation Reference Methods）。支持查询 GSEApY 基因集库和 CellTypist 模型。**逻辑说明：**

- 支持通过 `keyword` 模糊搜索（匹配名称、描述、物种或器官）。
- 支持通过 `type` 筛选特定类型（GSEApY 或 CellTypist）。
- 如果 `keyword` 为空（或未传），则返回指定类型下的所有方法信息。

2.8.2 接口路径（URL / Method / Version）

Endpoint: `/api/v1/methods/annotation/search`

Method: `GET`

2.8.3 请求参数（Request）

参数名	位置 (Location)	类型	必填	默认值	说明
keyword	Query中	string	No	""	搜索关键字。匹配字段包括：方法名、描述、物种、器官。传空字符串则返回全部。
type	Query中	string	No	"all"	方法类型筛选。可选值："all"（全部）, "gseapy", "celltypist"。

2.8.4 返回参数（Response）

参数名	类型	说明
id	int	数据库主键 ID
method_name	string	方法唯一标识名 (e.g., "CellMarker_20 24", "Immune_All_L ow.pkl")
type	string	方法类型 ("gseapy" 或 "celltypist")
description	string	方法描述 (可能 为 null)
species	string	适用物种 (e.g., "Human")
organ	string	适用器官/组织 (可能为 null)

2.8.5 错误码 (Error Codes)

HTTP Code	错误信息 (Detail)	说明
500	"An error occurred while searching annotation methods: ..."	服务器内部错误 (如数据库连接失 败)
422	"Validation Error"	参数校验错误 (FastAPI 自动处 理)

2.8.6 示例 (Example)

场景 1：获取所有 GSEApay 库 (无关键字) GET /api/v1/methods/annotation/search?
type=gseapy

场景 2：搜索包含 "Immune" 的所有方法 GET /api/v1/methods/annotation/search?
keyword=Immune&type=all

```
1  Response
2      [
3          {
4              "id":61,
5              "method_name": "ARCHS4_Cell-lines",
6              "type": "gseapy",
7              "description": "Cell type / tissue-related gene sets for Human",
8              "species": "Human",
9              "organ": "None"
10         },
11         {
12             "id":62,
13             "method_name": "ARCHS4_IDG_Coexp",
14             "type": "gseapy",
15             "description": "ARCHS4 IDG Coexp gene sets for Human",
16             "species": "Human",
17             "organ": "None"
18         },
19         {
20             "id":63,
21             "method_name": "ARCHS4_Kinases_Coexp",
22             "type": "gseapy",
23             "description": "ARCHS4 Kinases Coexp gene sets for Human",
24             "species": "Human",
25             "organ": "None"
26         }
27     ]
```

2.9 手动更改 Annotation 标签

2.9.1 接口概述 (Purpose) & 前端触发逻辑

用途：允许用户手动修改/覆盖特定 Snapshot 中某个聚类 (Cluster) 的细胞类型注释标签。这通常用于修正算法预测不准确的标签，或者将通用的标签（如 "T cell"）细化为更具体的亚型（如 "CD4+ Naive T cell"）。

前端触发逻辑：

1. 用户在注释结果页面 (Annotation Result Page) 查看聚类表格或 UMAP 图。
2. 用户点击某个 Cluster 的“重命名”按钮或直接编辑表格中的“Predicted Cell Type”列。
3. 用户输入新的名称并确认保存，前端调用此接口提交修改。

2.9.2 接口路径 (URL / Method / Version)

Endpoint: /api/v1/annotation/labels/update

Method: PATCH

2.9.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
snapshot_id	string	Yes	目标 Annotation Snapshot 的业务 ID。	"snap_annot_20260210_1234"
file_name	string	Yes	目标注释结果在元数据中的 Key (通常是去除 .png 后缀的文件名)。	"gseapy_CellMarker_2024" 或 "celltypist_immune_all_low"
updated_annotation	Object	Yes	需要更新的 Cluster ID 与新标签的映射字典。Key 为 Cluster ID, Value 为新名称。	{"0": "new cell", "1": "modified cell name 2"}

2.9.4 返回参数 (Response)

参数名	类型	说明
msg	string	操作结果提示信息
snapshot_id	string	被修改的 Snapshot ID
updated_count	int	实际成功更新的标签数量
target_file	string	被修改的目标 Key (对应请求中的 file_name)

2.9.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
400	-	"Annotation category '{target_key}' not found..."	提供的 file_name 在该 Snapshot 的元数据中不存在。请检查前端传参是否与后端存储的 Key 一致。
404	-	"Snapshot '{snapshot_id}' not found."	提供的 snapshot_id 在数据库中不存在。
500	-	"Failed to update labels: {error}"	服务器内部错误 (如数据库写入失败)。

2.9.6 示例 (Example)

Request

```
1  {
2      "snapshot_id": "annot_1770714992_67399",
3      "file_name": "gseapy_CellMarker_2024",
4      "updated_annotation": {
5          "0": "Test Modify label",
6          "1": "New cell name"
7      }
8  }
```

Response

```
1  {
2      "msg": "Annotation labels updated successfully.",
3      "snapshot_id": "annot_1770714992_67399",
4      "updated_count": 2,
5      "target_file": "gseapy_CellMarker_2024"
6  }
```

以下多个接口（直至第三部分 Agent 前）为备用子接口，前端逻辑暂时无需显式触发

2.10 计算 QC 指标 (QC Calculation)

2.10.1 接口概述 (Purpose)

计算线粒体、核糖体、血红蛋白等基因的比例及细胞的总计数统计。关键点：此步骤不修改原始数据，也不生成新的 AnnData 快照。它只生成用于前端绘制小提琴图 (Violin Plot) 和散点图的统计文件 (`qc_metrics.json`)。

计算增量列，只在 cache 储存新增的列。缓存只包含增量数据的 AnnData (增量)。

2.10.2 接口路径 (URL / Method / Version)

- URL: `/api/v1/preprocessing/qc/calculate`
- Method: `POST`

2.10.3 请求参数 (Request)

参数名	类型	必选	说明	示例
project_id	string	是	项目业务 ID	p_20250101_88321
dataset_id	string	是	数据集业务 ID	ds_raw_001
organism	string	否	物种 (决定正则前缀)	Human, Mouse
custom_prefixes	object	否	自定义前缀 (高级模式)	{"mt": "MT-", "hb": "^HB"}

2.10.4 返回参数 (Response)

参数名	类型	说明
dataset_id	string	数据集 ID
metrics_file_path	string	生成的指标统计文件路径 (供前端读取)
violin_plot_path	string	Path to the static Violin PDF
scatter_mt_path	string	Path to the static Scatter (MT vs Counts) PDF
scatter_genes_path	string	Path to the static Scatter (Genes vs Counts) PDF

2.10.5 错误码 (Error Codes)

2.10.6 示例 (Example)

代码块

```

1  {
2    "project_id": "p_20250101_88321",
3    "dataset_id": "ds_raw_001",
4    "organism": "Human"
5  }

```

2.10.7 时序图 (Sequence Diagram)

2.11 应用过滤 (Apply Filtering)

2.11.1 接口概述 (Purpose)

根据用户设定的阈值（基于接口一的可视化结果），对细胞和基因进行物理删除。关键点：此步骤会生成第一个分析快照 `snapshots_anndata/node_root_qc.h5ad`。（全量）

2.11.2 接口路径 (URL / Method / Version)

URL: `/api/v1/preprocessing/qc/filter`

Method: POST

2.11.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
<code>project_id</code>	string	是	项目 ID	p_20250101_88321
<code>dataset_id</code>	string	是	数据集 ID	
<code>min_genes</code>	int	是	最小基因数 (filter_cells)	200
<code>min_cells</code>	int	是	最小细胞数 (filter_genes)	3
<code>max_counts</code>	int	否	(可选) 最大 Count 数	
<code>pct_mt_max</code>	float	否	(可选) 最大线粒体比例	
<code>pct_hb_max</code>	float	否	(可选) 最大血红蛋白比例	

代码块

```
1  {
2      "min_genes": 200,          // 最小基因数 (filter_cells)
3      "min_cells": 3,           // 最小细胞数 (filter_genes)
4      "max_counts": 50000,       // (可选) 最大 Count 数
5      "pct_mt_max": 20.0,       // (可选) 最大线粒体比例
6      "pct_hb_max": 5.0        // (可选) 最大血红蛋白比例
7  }
```

2.11.4 返回参数 (Response)

参数名	类型	说明
snapshot_path	string	新生成的快照路径
n_obs_remaining	int	过滤后剩余细胞数
n_vars_remaining	int	过滤后剩余基因数

2.11.5 错误码 (Error Codes)

2.11.6 示例 (Example)

Request

```
1  {
2    "project_id": "p_20250101_88321",
3    "filters": { "min_genes": 200, "min_cells": 3, "pct_mt_max": 15 }
4 }
```

2.12 特征选择与标准化 (Feature Selection & Scaling)

2.12.1 接口概述 (Purpose)

对经过 QC 过滤的数据进行下游分析前的预处理。包含四个原子步骤：

1. **Normalization & Log1p**: 标准化测序深度并对数化。
2. **Identify HVGs**: 识别高变基因（通常 2000 个）。
3. **Backup Raw**: 将标准化后但未切片的数据备份到 `adata.raw`（用于后续差异分析）。
4. **Subset & Scale**: 只保留高变基因，并进行 Z-Score 缩放（为 PCA 准备）。

生成结果将包含一个 Dispersion Plot（离散度图）用于前端展示，以及一个新的 Snapshot（全量）。

2.12.2 接口路径 (URL / Method / Version)

URL: `/api/v1/preprocessing/hvg`

Method: `POST`

2.12.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	是	项目业务 ID	"p_20250101_88321"
dataset_id	string	是	数据集业务 ID	"ds_20260103_0f5c3"
snapshot_id	string	否	上一步(QC Filter)生成的快照ID。如果不传，后端默认取该 Dataset 下最新的 QC 快照。	"snap_20260108_qc"
n_top_genes	int	否	期望保留的高变基因数量。最常用的调节参数。	2000
flavor	string	否	选择算法风格。可选 'seurat', 'cell_ranger', 'seurat_v3'。	'seurat'
target_sum	float	否	标准化时的总计数目标 (Normalize Total)。	10000.0 (1e4)

2.12.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	新生成的快照 ID (e.g., snap_xxx_hvg)
snapshot_path	string	新生成的 .h5ad 文件路径 (Relative Path)
hvg_plot_path	string	可视化路径：高变基因分布图 (Mean vs Dispersion Scatter Plot)
n_genes_found	int	实际找到的高变基因数量 (通常等于 n_top_genes)
msg	string	操作成功提示

2.12.5 错误码 (Error Codes)

HTTP Code	Error Code	说明	解决方案
404	DATASET_NOT_FOUND	指定的 dataset_id 或 snapshot_id 不存在	检查 ID 是否正确，或是否先运行了 QC Filter
400	INVALID_PARAMS	参数错误，例如 n_top_genes 设置过大超过了总基因数	调整 n_top_genes (e.g. < 总基因数)
500	CALCULATION_ERROR	Scanpy 计算过程中报错 (如内存不足，数据全为0等)	检查上一步过滤是否太狠导致没有细胞剩余

2.12.6 示例 (Example)

Request

```
1 POST /api/v1/preprocessing/hvg
2
3 {
4     "project_id": "p_1767495922_c912e",
5     "dataset_id": "ds_20260103_220522_0f5c3",
6     "n_top_genes": 2000,
7     "flavor": "seurat"
8 }
```

Response

```
1 {
2     "snapshot_id": "snap_20260108_2301_hvg1",
3     "snapshot_path": "projects/p_.../snapshots_anndata/node_hvg_snap_...h5ad",
4     "hvg_plot_path": "projects/p_.../qc/hvg_dispersion.pdf",
5     "n_genes_found": 2000,
6     "msg": "HVG selection and scaling complete."
7 }
```

2.13 运行 PCA 分析 (Run PCA)

2.13.1 接口概述 (Purpose)

对高变基因数据进行主成分分析 (PCA)。关键点：

- 读取 `node_processed.h5ad`。
- 执行 `pca`。
- 核心功能：**提取 `adata.uns['pca']['variance_ratio']` 并保存为 JSON，供前端绘制 **Elbow Plot**。这是用户判断数据复杂度的关键依据。
- 保存中间快照 `node_pca_{snap_id}.h5ad`。

2.13.2 接口路径 (URL / Method / Version)

- URL:** `/api/v1/dim-reduction/pca`
- Method:** `POST`

2.13.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	是	项目 ID	p_20250101_88321
n_comps	int	否	计算的主成分数量	50
svd_solver	string	否	SVD 求解器	arpack

2.13.4 返回参数 (Response)

参数名	类型	说明
variance_plot_data	object	用于绘制 Elbow Plot 的数据
pca_scatter_path	string	PC1/PC2 散点图数据文件路径
snapshot_path	string	PCA 快照路径

2.13.5 错误码 (Error Codes)

2.13.6 示例 (Example)

代码块

```
1  {
2      "project_id": "p_20250101_88321",
3      "n_comps": 50
4 }
```

2.13.7 时序图 (Sequence Diagram)

2.14 构建邻居图 (Build Neighborhood Graph)

2.14.1 接口概述 (Purpose)

根据用户在 Elbow Plot 上选定的 PC 数量 (`n_pcs`), 计算细胞间的邻居关系图。关键点：读取 `node_pca.h5ad` , 计算后保存为 `node_graph_{snap_id}.h5ad`。

2.14.2 接口路径 (URL / Method / Version)

- **URL:** `/api/v1/dim-reduction/neighbors`
- **Method:** `POST`
- Port:

2.14.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
<code>project_id</code>	string	是	项目 ID	p_20250101_88321
<code>n_neighbors</code>	int	否	邻居数量 (影响局部/全局结构)	1
<code>n_pcs</code>	int	是	用户决策参数：使用的 PC 数量	30 (通常根据拐点图决定)

2.14.4 返回参数 (Response)

参数名	类型	说明
<code>snapshot_path</code>	string	包含图结构的快照路径
<code>n_connectivities</code>	int	连接数统计 (Debug用)

2.14.5 错误码 (Error Codes)

2.14.6 示例 (Example)

代码块

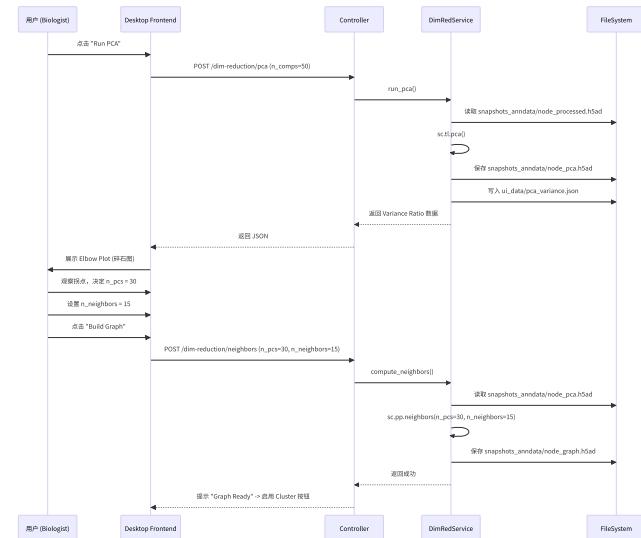
```
1  {
2      "project_id": "p_20250101_88321",
3      "n_neighbors": 15,
4      "n_pcs": 30
5 }
```

2.14.7 时序图 (Sequence Diagram)

```

1 sequenceDiagram
2     participant User as 用户 (Biologist)
3     participant UI as Desktop Frontend
4     participant CTL as Controller
5     participant SVC as DimRedService
6     participant FS as FileSystem
7
8     %% Step 1: PCA Calculation
9     User->>UI: 点击 "Run PCA"
10    UI->>CTL: POST /dim-reduction/pca (n_compo=50)
11    CTL->>SVC: run_pca()
12    SVC->>FS: 读取 snapshots_anndata/node_processed.h5ad
13    SVC->>SVC: sc.tl.pca()
14    SVC->>FS: 保存 snapshots_anndata/node_pca.h5ad
15    SVC->>FS: 写入 ui_data/pca_variance.json
16    SVC-->>CTL: 返回 Variance Ratio 数据
17    CTL-->>UI: 返回 JSON
18    UI->>User: 展示 Elbow Plot (碎石图)
19
20    %% Step 2: Human Decision
21    User->>UI: 观察拐点, 决定 n_pcs = 30
22    User->>UI: 设置 n_neighbors = 15
23    User->>UI: 点击 "Build Graph"
24
25    %% Step 3: Neighbors
26    UI->>CTL: POST /dim-reduction/neighbors (n_pcs=30, n_neighbors=15)
27    CTL->>SVC: compute_neighbors()
28    SVC->>FS: 读取 snapshots_anndata/node_processed.h5ad
29    SVC->>SVC: sc.pp.neighbors(n_pcs=30)
30    SVC->>FS: 保存 snapshots_anndata/node_graph.h5ad
31    SVC-->>CTL: 返回成功
32    CTL-->>UI: 提示 "Graph Ready" -> 启用 Cluster 按钮
33

```



2.15 单个簇功能富集分析 (GSEAp Single)

2.15.1 接口概述 (Purpose)

对指定的单个簇 (Cluster) 执行功能富集分析 (基于 GSEAp/Enrichr)。核心逻辑：

1. 基于之前的 DEG (差异表达分析) 结果提取指定簇的 Marker Genes。

2. 调用 Enrichr API 对这些基因进行富集分析（如 CellMarker, GO, KEGG）。
3. 返回该簇的潜在细胞类型或通路信息。

2.15.2 接口路径 (URL / Method / Version)

URL: /api/v1/annotation/gseapy/single

Method: POST

2.15.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	Yes	项目业务 ID	"p_20260206_1234"
dataset_id	string	Yes	数据集业务 ID	"ds_20260206_5678"
snapshot_id	string	No	DEG 分析结果的快照 ID。若不填，系统将尝试查找最新的 DEG 快照。	"s_deg_wilcoxon..."
cluster_id	string	Yes	目标簇的 ID。	"1" 或 "CD8_T_cells"
top_n_genes	int	No	每个簇提取多少个顶部 Marker 基因用于富集。	100
categories	List[str]	No	Enrichr 的基因集库名称列表。	["CellMarker_2024", "BioPlanet_2019"]

2.15.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	新生成的注释结果快照 ID
snapshot_path	string	结果文件存储路径
cluster_id	string	当前分析的簇 ID (e.g., "1")
enrichment_results	Dict	富集分析结果详情。Key 为库名，Value 为富集条目列表。
msg	string	操作结果消息

2.15.5 示例 (Example)

代码块

```

1  {
2      "project_id": "p_20260206_1001",
3      "dataset_id": "ds_20260206_raw_data",
4      "snapshot_id": "s_deg_wilcoxon_20260206_556677",
5      "cluster_id": "0",
6      "top_n_genes": 50,
7      "categories": [
8          "CellMarker_2024",
9          "BioPlanet_2019"
10     ]
11 }
```

代码块

```

1  {
2      "snapshot_id": "s_annot_gsea_single_20260206_112233",
3      "snapshot_path": "projects/p_20260206_1001/ds_20260206_raw_data/snapshots/s_annot_gsea_single_20260206_112233.json",
4      "cluster_id": "0",
5      "msg": "Annotation for Cluster 0 completed successfully.",
6      "enrichment_results": {
7          "CellMarker_2024": [
8              {
9                  "Term": "CD8+ T cell:Lung",
10                 "Overlap": "12/50",
```

```

11     "P-value": 1.25e-14,
12     "Adjusted P-value": 3.45e-12,
13     "Old P-value": 0,
14     "Old Adjusted P-value": 0,
15     "Odds Ratio": 154.32,
16     "Combined Score": 5231.45,
17     "Genes": ["CD8A", "CD8B", "GZMK", "NKG7", "CCL5"]
18   },
19   {
20     "Term": "Natural Killer cell:Peripheral Blood",
21     "Overlap": "8/50",
22     "P-value": 4.56e-8,
23     "Adjusted P-value": 1.23e-6,
24     "Old P-value": 0,
25     "Old Adjusted P-value": 0,
26     "Odds Ratio": 89.12,
27     "Combined Score": 1204.56,
28     "Genes": ["NKG7", "GNLY", "GZMB"]
29   }
30 ],
31 "BioPlanet_2019": [
32   {
33     "Term": "T cell receptor signaling pathway",
34     "Overlap": "5/50",
35     "P-value": 0.00012,
36     "Adjusted P-value": 0.0045,
37     "Odds Ratio": 45.2,
38     "Combined Score": 340.1,
39     "Genes": ["CD3D", "CD3E", "CD247", "LCK"]
40   }
41 ],
42 }
43 }
```

2.16 批量全簇功能富集分析 (GSEAp Batch)

2.16.1 接口概述 (Purpose)

对 DEG 分析中识别出的**所有簇**执行自动化的批量功能富集分析。核心逻辑：

1. 遍历所有簇，提取每个簇的 Marker Genes。
2. 批量查询 Enrichr API。
3. 生成一份包含所有簇潜在身份的汇总映射表。

2.16.2 接口路径 (URL / Method / Version)

接口路径 (URL / Method / Version)

- **URL:** /api/v1/annotation/gseapy/all
- **Method:** POST
-

2.16.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	Yes	项目业务 ID	"p_20260206_1234"
dataset_id	string	Yes	数据集业务 ID	"ds_20260206_5678"
snapshot_id	string	No	DEG 分析结果的快照 ID。	"s_deg_wilcoxon..."
cluster_id	string	No	此处应传 "ALL" 或忽略，后端会自动处理所有簇。	"ALL"
top_n_genes	int	No	每个簇提取多少个顶部 Marker 基因。	100
categories	List[str]	No	Enrichr 的基因集库名称列表。	["CellMarker_2024"]

2.16.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	新生成的注释结果快照 ID
snapshot_path	string	结果文件存储路径
cluster_id	List/String	返回 "ALL" 或包含所有已处理簇 ID 的列表。
enrichment_results	Dict	汇总结果。Key 为 Cluster ID，Value 为该簇的最佳匹配注释。
msg	string	操作结果消息

2.16.5 错误码 (Error Codes)

该错误码表适用于 `/api/v1/annotation/` 下的所有接口 (GSEAp Single, GSEAp Batch, CellTypist)。

HTTP Code	业务 Error Code	错误信息 (Message Example)	适用接口	解决方案
400	PARAM_ERROR	Invalid parameter: top_n_genes must be positive.	All	检查请求体中的数值参数是否合法 (如 <code>top_n > 0</code>)。
400	PREREQUISITE_MISSING	DEG analysis results not found in source snapshot.	GSEAp	关键错误。GSEAp 注释依赖于差异表达分析 (DEG)。请先运行 DEG 分析接口，并传入正确的 <code>snapshot_id</code> 。
400	PREREQUISITE_MISSING	Clustering data (obs) not found in snapshot.	CellTypist	CellTypist 依赖聚类结果 (如 <code>leiden</code> 列)。请确认上一步聚类已完成。
400	CLUSTER_NOT_FOUND	Cluster ID '99' not found in DEG results.	GSEAp	用户请求了一个不存在的簇 ID。请检查前端展示的簇列表是否与当前数据一致。
400	COLUMN_NOT_FOUND	Target column 'leiden' not found in adata.obs.	CellTypist	多数投票 (Majority Voting) 指定的聚类列名不存在。请检查 <code>target_cluster_col</code> 参数。
400	MODEL_NOT_SUPPORTED	Model 'Brain_Atlas_High.pkl' is not available.	CellTypist	请求了后端未下载或不支持的 CellTypist 模型。请使用支持的模型列表。
400	INVALID_CATEGORY	Enrichr category 'Bad_Lib' is invalid.	GSEAp	传入的基因集库名称 (categories) Enrichr 不支持。请参考 Enrichr 官方库列表。

404	RESOURCE_NOT_FOUND	Project 'p_xxx' or Dataset 'ds_xxx' not found.	All	检查 URL 或 Body 中的 ID 是否正确，或者项目是否已被删除。
404	SNAPSHOT_NOT_FOUND	Source snapshot 's_deg...' does not exist.	All	检查 snapshot_id 是否正确，或者该文件是否在磁盘上丢失。

HTTP Code	业务 Error Code	错误信息 (Message Example)	适用接口	解决方案
502	ENRICHR_API_ERROR	Failed to connect to Enrichr API: Timeout.	GSEApay	GSEApay 需调用外部 API。可能是网络问题或 Enrichr 服务器宕机。建议稍后重试。
500	CELLTYPIST_EXCEPTION_ERROR	CellTypist prediction failed: Memory Error.	CellTypist	机器学习模型运行失败（如内存不足）。建议检查服务器资源或减少数据量。
500	IO_ERROR	Failed to write annotation result to disk.	All	磁盘读写权限或空间不足。请联系系统管理员。
500	INTERNAL_ERROR	Unexpected error: [Stack Trace]	All	代码逻辑导致的未捕获异常。请将错误日志发送给开发人员。

2.16.6 示例 (Example)

```

req

1  {
2      "project_id": "p_20260206_1001",
3      "dataset_id": "ds_20260206_raw_data",
4      "snapshot_id": "s_deg_wilcoxon_20260206_556677",
5      "cluster_id": "ALL",
6      "top_n_genes": 100,
7      "categories": [
8          "CellMarker_2024"
9      ]
10 }

```

Res

```
1  {
2      "snapshot_id": "s_annot_gsea_batch_20260206_998877",
3      "snapshot_path":
4          "projects/p_20260206_1001/ds_20260206_raw_data/snapshots/s_annot_gsea_batch_202
5          60206_998877.json",
6      "cluster_id": [
7          "0",
8          "1",
9          "2"
10     ],
11     "msg": "Batch annotation for all clusters completed.",
12     "enrichment_results": {
13         "0": {
14             "top_match": "CD8+ T cell",
15             "database": "CellMarker_2024",
16             "confidence_score": 5231.45,
17             "top_terms": [
18                 {
19                     "Term": "CD8+ T cell:Lung",
20                     "Adjusted P-value": 3.45e-12
21                 }
22             ],
23         },
24         "1": {
25             "top_match": "Macrophage",
26             "database": "CellMarker_2024",
27             "confidence_score": 4100.22,
28             "top_terms": [
29                 {
30                     "Term": "Macrophage:Liver",
31                     "Adjusted P-value": 1.12e-10
32                 }
33             ],
34         },
35         "2": {
36             "top_match": "B cell",
37             "database": "CellMarker_2024",
38             "confidence_score": 3890.15,
39             "top_terms": [
40                 {
41                     "Term": "B cell:Spleen",
42                     "Adjusted P-value": 5.67e-09
43                 }
44             ]
45     }
```

```
44      }
45  }
```

2.17 CellTypist 自动注释 (CellTypist Auto)

2.17.1 接口概述 (Purpose)

使用 CellTypist 机器学习模型对整个数据集进行自动细胞类型预测。核心逻辑：

1. 加载 AnnData (通常是聚类后的结果)。
2. 应用预训练模型 (如 Immune_All_Low.pkl) 预测每个细胞的类型。
3. (可选) 结合聚类结果进行多数投票 (Majority Voting)，修正预测结果。

2.17.2 接口路径 (URL / Method / Version)

URL: /api/v1/annotation/celltypist

Method: POST

2.17.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
project_id	string	Yes	项目业务 ID	"p_20260206_1234"
dataset_id	string	Yes	数据集业务 ID	"ds_20260206_5678"
snapshot_id	string	No	聚类分析结果的快照 ID (AnnData)。	"s_cluster_leiden..."
model_names	List[str]	No	CellTypist 预训练模型名称列表。	["Immune_All_Low.pkl"]
majority_voting	bool	No	是否开启多数投票修正 (利用聚类信息优化预测)。	TRUE
target_cluster_col	string	No	用于多数投票的聚类列名 (需存在于 obs 中)。	"leiden"

2.17.4 返回参数 (Response)

参数名	类型	说明
snapshot_id	string	新生成的 CellTypist 结果快照 ID
snapshot_path	string	结果 AnnData 文件路径
cluster_id	None	此接口针对全数据，通常返回 null
enrichment_results	Dict	包含预测概览，例如每种细胞类型的计数统计。
msg	string	操作结果消息

2.17.5 示例 (Example)

```
req
1  {
2      "project_id": "p_20260206_1001",
3      "dataset_id": "ds_20260206_raw_data",
4      "snapshot_id": "s_cluster_leiden_20260206_334455",
5      "model_names": [
6          "Immune_All_Low.pkl"
7      ],
8      "majority_voting": true,
9      "target_cluster_col": "leiden"
10 }
```

```
res
1  {
2      "snapshot_id": "s_annot_celltypist_20260206_667788",
3      "snapshot_path":
4          "projects/p_20260206_1001/ds_20260206_raw_data/snapshots/s_annot_celltypist_202
60206_667788.h5ad",
5      "cluster_id": null,
6      "msg": "CellTypist annotation completed. Majority voting applied using
'leiden'.",
7      "enrichment_results": {
8          "model_used": "Immune_All_Low.pkl",
```

```
8     "summary_stats": {
9         "T cells": 1200,
10        "Monocytes": 800,
11        "B cells": 400,
12        "NK cells": 150
13    },
14    "majority_voting_map": {
15        "0": "T cells",
16        "1": "Monocytes",
17        "2": "B cells",
18        "3": "T cells",
19        "4": "NK cells"
20    },
21    "conf_score_distribution": {
22        "mean": 0.89,
23        "min": 0.45,
24        "max": 0.99
25    }
26 }
27 }
```

3. Agent 对话

3.1 发送 AI chat message 请求

3.1.1 接口概述 (Purpose)

⚠ 注意：本接口运行在独立端口，不是主应用端口！ 端口号为 41889.

本接口用于向 LLM 发送一条用户消息，并触发完整的智能体工作流。

3.1.2 接口路径 (URL / Method / Version)

- URL: `/api/agent/chat`
- Method: `POST`
- URL: `http://127.0.0.1:41889/api/agent/chat`

3.1.3 请求参数 (Request)

字段名	类型	必填	默认值	说明
message	string	是	无	用户输入的自然语言消息，例如：“Merge clusters 3 and 4”
snapshot_id	string	是	无	当前前端正在使用的 snapshot ID
project_id	string	是	无	项目 ID，智能体规划必须依赖此字段
dataset_id	string	否	null	数据集 ID，可选
session_id	string	否	null	会话 ID，用于多轮对话追踪

3.1.4 返回参数 (Response)

字段名	类型	说明
reply	string	智能体最终生成的 Markdown 文本（总结、报告或追问）
final_snapshot_id	string / null	执行后更新的 snapshot ID；若无更新则可能为

3.1.5 错误码 (Error Codes)

HTTP 状态码	说明	可能原因
500	Lotus encountered a technical issue	LangGraph 执行失败、内部异常、参数缺失等
422	Unprocessable Entity	请求 JSON 字段格式不正确或缺失必填字段
200	OK	请求成功

3.1.6 示例 (Example)

Request

```

1  {
2      "message": "Run PCA on the dataset",
3      "snapshot_id": "snap_0001",

```

```
4     "project_id": "proj_123",
5     "dataset_id": "ds_001",
6     "session_id": "sess_001"
7 }
8
```

Response

```
1  {
2     "reply": "### PCA Completed\nThe PCA step has been executed successfully...",
3     "final_snapshot_id": "snap_0002"
4 }
5
```

!!! 以下接口皆是主应用端口上运行的

3.2 获取某一项目会话列表 (List Project Sessions)

3.2.1 接口概述 (Purpose) & 前端触发逻辑

Purpose: 获取指定项目下的所有会话记录，通常按更新时间从新到旧排序。

Frontend Trigger:

- 当用户进入某个具体项目页面，或者在界面侧边栏展开“历史会话”列表时触发。用于渲染侧边栏菜单，让用户在不同对话之间切换。
- 或者在用户打开对话框时，先查询项目会话列表，然后获取列表中第一个（最新）的 session id。

3.2.2 接口路径 (URL / Method / Version)

URL: /api/v1/agent/history/project/session

Method: GET

3.2.3 请求参数 (Request)

参数名称	类型	描述
session_id	string	会话的唯一业务 ID。
name	string	会话的显示名称（例如：“第一次分析”）。
agent	string	该会话使用的 Agent 名称（例如：“BioAgent-Pro”）。
updated_at	datetime	会话最后一次更新或活跃的时间戳。

3.2.4 返回参数 (Response)

参数名称	类型	描述
session_id	string	会话的唯一业务 ID。
name	string	会话的显示名称（例如：“第一次分析”）。
agent	string	该会话使用的 Agent 名称（例如：“BioAgent-Pro”）。
updated_at	datetime	会话最后一次更新或活跃的时间戳。

3.2.5 错误码 (Error Codes)

HTTP Status	描述
200	成功获取列表。
422	参数校验错误 (如缺少 project_id)。
500	服务器内部错误。

3.2.6 示例 (Example)

Req

```
1 GET /api/v1/agent/history/project/session?project_id=proj_123456
```

Res

```
1 [
2   {
3     "session_id": "sess_001",
```

```
4      "name": "Protein Analysis Run 1",
5      "agent": "BioAgent-Pro",
6      "updated_at": "2023-10-27T10:00:00Z"
7  },
8  {
9      "session_id": "sess_002",
10     "name": "New Analysis",
11     "agent": "BioAgent-Pro",
12     "updated_at": "2023-10-26T14:30:00Z"
13 }
14 ]
```

3.3 创建 chat 会话 session

3.3.1 接口概述（Purpose） & 前端触发逻辑

Purpose: 在指定项目下初始化一个新的空白会话。

Frontend Trigger:

- 当用户点击侧边栏的“新建对话”或“+”按钮时触发。后端会生成 Session ID 并初始化状态，前端收到响应后跳转或切换至新会话窗口。
- 或者在调用“[获取某一项目会话列表 \(List Project Sessions\)](#)”接口后，发现该 project 没有对应 session id，可主动调用此接口创建新 session。

3.3.2 接口路径（URL / Method / Version）

URL: `/api/v1/agent/history/sessions/create`

Method: `POST`

3.3.3 请求参数（Request）

参数名称	类型	是否必填	描述
project_id	string	是	关联的项目 ID。
agent_name	string	否	指定使用的 Agent 名称，默认为 "BioAgent-Pro"。
initial_name	string	否	会话的初始名称，默认为 "New Analysis"。

3.3.4 返回参数 (Response)

参数名称	类型	描述
session_id	string	新创建的会话唯一 ID。
project_id	string	关联的项目 ID。
name	string	会话名称。
status	string	会话初始状态 (如 "INIT")。
created_at	datetime	创建时间。

3.3.5 错误码 (Error Codes)

HTTP Status	描述
201	Created. 会话创建成功。
404	未找到对应的项目 ID，创建失败。
422	请求参数格式错误。
500	服务器内部错误。

3.3.6 示例 (Example)

```
req
```

```
1 { "project_id": "proj_123456", "initial_name": "Experiment #3 Discussion" }
```

res

```
1 {
2   "session_id": "sess_new_789",
3   "project_id": "proj_123456",
4   "name": "Experiment #3 Discussion",
5   "status": "INIT",
6   "created_at": "2023-10-27T12:00:00Z"
7 }
```

3.4 重命名会话 (Rename Chat Session)

3.4.1 接口概述 (Purpose)

Purpose: 修改现有会话的显示名称。

Frontend Trigger: 用户在会话列表中点击某个会话旁的“编辑/重命名”图标，输入新名称并确认后触发。

3.4.2 接口路径 (URL / Method / Version)

URL: /api/v1/agent/history/sessions/rename/{session_id}

Method: PATCH

3.4.3 请求参数 (Request)

- **Path Parameters (路径参数)**

参数名称	类型	是否必填	描述
session_id	string	是	目标会话的 ID。

- **Body Parameters (JSON)**

参数名称	类型	是否必填	描述
new_name	string	是	新的会话名称。

3.4.4 返回参数 (Response)

参数名称	类型	描述
msg	string	操作结果提示信息。

3.4.5 错误码 (Error Codes)

HTTP Status	描述
200	重命名成功。
404	会话未找到 (Session not found)。
422	参数校验错误 (如新名称为空)。

3.4.6 示例 (Example)

REQ

```
1 PATCH /api/v1/agent/history/sessions/rename/sess_001
2 Content-Type: application/json
3
4 {
5     "new_name": "Final Review"
6 }
```

res

```
1 {
2     "msg": "Successfully Renamed"
3 }
```

3.5 删除会话 (Delete Chat Session)

3.5.1 接口概述 (Purpose) & 前端触发逻辑

Purpose: 永久删除指定的会话及其关联的历史记录。

Frontend Trigger: 用户在会话列表中点击“删除”图标，并在确认弹窗中点击“确认”后触发。

3.5.2 接口路径 (URL / Method / Version)

URL: /api/v1/agent/sessions/delete/{session_id}

Method: PATCH

3.5.3 请求参数 (Request)

参数名称	类型	是否必填	描述	默认值/示例
session_id	string	是	需要删除的会话ID。	

3.5.4 返回参数 (Response)

Content-Type: N/A (No Content)

Body: 空

3.5.5 错误码 (Error Codes)

HTTP Status	描述
204	No Content. 删除成功。
404	会话未找到 (Session not found), 无法删除。

3.5.6 示例 (Example)

Request

```
1  DELETE /api/v1/agent/sessions/delete/sess_001
```

Response

```
1  (HTTP 204 No Content, Body is empty)
```

3.6 获取单个会话历史记录 (History Retrieval)

3.6.1 接口概述 (Purpose) & 前端触发逻辑

用途：获取指定会话（Session）的完整或分页历史记录。该接口通常返回经过 UI 格式化处理的数据，包含用户消息、Agent 回复以及中间的工具调用步骤。

前端触发逻辑：

- 初始化加载：** 用户点击某个会话或进入聊天页面时，调用此接口加载第一页数据。
- 加载更多：** 用户在聊天窗口向上滚动（Infinite Scroll）查看更早的消息时，传入下一页的 `page` 参数。

3.6.2 接口路径 (URL / Method / Version)

- Endpoint:** /api/v1/agent/history/{session_id}/history/all
- Method:** GET

3.6.3 请求参数 (Request)

参数名	位置 (Location)	类型	必选	说明	默认值
session_id	Path	string	Yes	会话的业务 ID	/
page	Query	int	No	分页页码，必须 <code>>= 1</code>	1
size	Query	int	No	每页条数，范围 1-100	20

3.6.4 返回参数 (Response)

响应是一个 JSON 对象 (Object)，包含分页信息和消息列表：

参数名	类型	说明
session_id	string	当前查询的会话 ID
total_messages	int	该会话下的消息总条数
page	int	当前页码
size	int	当前页大小
messages	List[Object]	消息列表 (结构见下表)

messages 列表中的对象结构:

参数名	类型	说明
id	int	消息的数据库主键 ID
type	string	消息类型 (user_input, agent_thought, tool_call, tool_result, agent_final)
content	object/string	消息的具体内容 Payload (结构同写入接口的数据字段)
timestamp	string	创建时间 (ISO 8601 格式字符串)

3.6.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
422	-	"Validation Error"	请求参数校验失败 (如 page 小于 1)。
404	-	"Not Found"	指定的 session_id 不存在。
500	-	"Internal Server Error"	数据库查询失败。

3.6.6 示例 (Example)

Request: GET /api/v1/agent/history/session_p_default/history/all?
page=1&size=2

Response (Success - 200 OK):

```
1  {
2      "session_id": "session_p_default_initial",
3      "total_messages": 42,
4      "page": 1,
5      "size": 2,
6      "messages": [
7          {
8              "id": 105,
9              "type": "agent_final",
10             "content": {
11                 "text": "Analysis complete. Here are the UMAP plots.",
12                 "attachments": ["/files/umap.png"]
13             },
14             "timestamp": "2026-02-09T10:05:00.123456"
15         },
16         {
17             "id": 104,
18             "type": "tool_result",
19             "content": {
20                 "tool_name": "run_clustering",
21                 "result_data": "Clustering finished with 12 groups.",
22                 "is_error": false
23             },
24             "timestamp": "2026-02-09T10:04:58.654321"
25         }
26     ]
27 }
```

4. 全局偏好管理 (Global Preference)

4.1 创建用户偏好 (Create New User Preference)

4.1.1 接口概述 (Purpose) & 前端触发逻辑

用途： 创建一个新的全局用户偏好设置。该设置通常包含特定的一组分析参数（如 QC 阈值、聚类分辨率等），以便用户在后续分析中快速复用。

前端触发逻辑： 用户在左下角设置面板调整好参数后，点击“Save Preference”按钮。前端将当前的参数配置序列化为 JSON 字符串并调用此接口。

4.1.2 接口路径 (URL / Method / Version)

Endpoint: /api/v1/user/preference/create

Method: POST

4.1.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
preference_name	string	No	自定义偏好名称。如果为空，后端会自动生成(如preference_1)。	"Standard QC & Clustering"
settings	string	Yes	存储具体的管道参数，必须是合法的 JSON 字符串。	{"\"qc_min_genes\": 500, \"resolution\": 1.0}"

4.1.4 返回参数 (Response)

参数名	类型	说明
id	int	数据库生成的主键 ID
preference_name	string	偏好名称
settings	Object	存储的配置 JSON Object
created_at	string	创建时间 (ISO 8601)
updated_at	string	更新时间 (ISO 8601)

4.1.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
400	-	"Failed to create preference. Name '{name}' might already exist."	该名称已存在，请提示用户更换名称。
500	-	"Failed to create preference: {error}"	服务器内部错误，请检查日志。

4.1.6 示例 (Example)

Request

```

1   {
2     "preference_name": "My Custom Settings",
3     "settings": {
4       "preprocessing_params": {
5         "skip_qc_calculation": false,
6         "skip_qc_filter": false,
7         "skip_hvg": false,
8         "skip_pca": false,
9         "skip_neighbors": false,
10        "organism": "Human",
11        "min_genes": 200,
12        "min_cells": 3,
13        "pct_mt_max": 15,
14        "max_counts": 40000,
15        "n_top_genes": 2000,
16        "flavor": "seurat",
17        "target_sum": 10000,
18        "n_comps": 50,
19        "svd_solver": "arpack",
20        "n_neighbors": 15,
21        "n_pcs": 30
22      },
23      "clustering_params": {
24        "method": "leiden",
25        "resolutions": [
26          0.5,
27          1
28        ],

```

```

29         "primary_resolution": 0.5,
30         "run_hierarchical": true
31     },
32     "deg_params": {
33         "method": "wilcoxon",
34         "n_top_genes": 50
35     },
36     "annotation_params": {
37         "cluster_id": "ALL",
38         "categories": [
39             "CellMarker_2024",
40             "BioPlanet_2019"
41         ]
42     },
43     "celltypist_params": {
44         "model_name": "Immune_All_Low.pkl",
45         "majority_voting": true
46     }
47 }
48 }
```

Response

```

1  {
2      "id":1,
3      "preference_name": "CI_Test_Preference",
4      "settings": {
5          "preprocessing_params": {
6              "skip_qc_calculation": false,
7              "skip_qc_filter": false,
8              "skip_hvg": false,
9              "skip_pca": false,
10             "skip_neighbors": false,
11             "organism": "Human",
12             "min_genes": 200,
13             "min_cells": 3,
14             "pct_mt_max": 15.0,
15             "max_counts": 40000,
16             "n_top_genes": 2000,
17             "flavor": "seurat",
18             "target_sum": 10000.0,
19             "n_comps": 50,
20             "svd_solver": "arpack",
21             "n_neighbors": 15,
22             "n_pcs": 30
23     },
```

```
24     "clustering_params":{  
25         "method":"leiden",  
26         "resolutions":[  
27             0.5,  
28             1.0  
29         ],  
30         "primary_resolution":0.5,  
31         "run_hierarchical":true  
32     },  
33     "deg_params":{  
34         "method":"wilcoxon",  
35         "n_top_genes":50  
36     },  
37     "annotation_params":{  
38         "cluster_id":"ALL",  
39         "categories":[  
40             "CellMarker_2024",  
41             "BioPlanet_2019"  
42         ]  
43     },  
44     "celltypist_params":{  
45         "model_name":"Immune_All_Low.pkl",  
46         "majority_voting":true  
47     }  
48 },  
49 "created_at":"2026-02-10T05:40:04.793577",  
50 "updated_at":"2026-02-10T05:40:04.793577"  
51 }
```

4.2 查询用户偏好 (Get User Preference by Name)

4.2.1 接口概述 (Purpose) & 前端触发逻辑

用途：根据名称获取特定的偏好设置详情。**注意：**调用此接口会自动更新该记录的 `updated_at` 时间戳，表示该偏好最近被访问过。

前端触发逻辑： 用户在下拉菜单中选择某个预设配置时，或者系统尝试加载特定名称的默认配置时。

4.2.2 接口路径 (URL / Method / Version)

- **Endpoint:** /api/v1/user/preference/query

- Method: GET

4.2.3 请求参数 (Request)

参数名	类型	必选	说明	默认值/示例
name	string	Yes	要查询的唯一偏好名称。	"Standard QC & Clustering"

4.2.4 返回参数 (Response)

参数名	类型	说明
id	int	数据库生成的主键 ID
preference_name	string	偏好名称
settings	Object	存储的配置 JSON Object
created_at	string	创建时间 (ISO 8601)
updated_at	string	更新时间 (ISO 8601)

4.2.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
400	-	"Failed to create preference. Name '{name}' might already exist."	该名称已存在，请提示用户更换名称。
500	-	"Failed to create preference: {error}"	服务器内部错误，请检查日志。

4.2.6 示例 (Example)

Request

1 Request: GET /api/v1/user/preference/query?name=Standard%20QC

Response

```
1  {
2      "id":1,
3      "preference_name":"CI_Test_Preference_Renamed",
4      "settings":{
5          "qc_min_genes":500,
6          "clustering_resolution":1.2
7      },
8      "created_at":"2026-02-10T05:42:49.804753",
9      "updated_at":"2026-02-10T05:42:53.987221"
10 }
```

4.3 获取最新偏好 (Get Latest User Preference)

4.3.1 接口概述 (Purpose) & 前端触发逻辑

用途：

获取系统中最近一次更新或创建的那个偏好设置。

前端触发逻辑：

用户打开分析页面时，界面默认加载用户上次使用的配置（“Last Used Settings”）。

4.3.2 接口路径 (URL / Method / Version)

- **Endpoint:** /api/v1/user/preference/latest/one
- **Method:** GET
- **Version:** v1

4.3.3 请求参数 (Request)

无。

4.3.4 返回参数 (Response)

同上 (UserPreferenceResponse 结构)。

4.3.5 错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
404	-	"No preferences found."	数据库中没有任何有效的偏好记录。

4.3.6 示例 (Example)

Request:

```
GET /api/v1/user/preference/latest/one
```

4.4 更新用户偏好 (Update User Preference)

4.4.1 接口概述 (Purpose) & 前端触发逻辑

用途:

更新现有偏好设置的参数内容，或者对其进行重命名。

前端触发逻辑:

用户修改了当前预设的参数并点击“更新/保存”，或者用户在管理列表中修改了预设的名称。

4.4.2 接口路径 (URL / Method / Version)

- **Endpoint:** /api/v1/user/preference/update
- **Method:** PATCH
- **Version:** v1

请求参数 (Request)

Location: Body (JSON)

参数名	类型	必选	说明	默认值/示例
name	string	Yes	当前的偏好名称 (用于定位记录)。	"Old Name"
settings	string	Yes	新的参数配置 JSON 字符串。	"{\"qc_min_genes\": 600}"
new_name	string	No	(可选) 新的偏好名称。如果提供且不为空，将执行重命名。	"New Renamed Name"

返回参数 (Response)

同上 (UserPreferenceResponse 结构)，返回更新后的对象。

错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
400	-	"Update failed. Preference '{name}' not found or new name '{new_name}' already exists."	检查原名称是否存在，或新名称是否已与其他记录冲突。
404	-	"Error retrieving updated preference data."	更新成功但回查数据失败（极少见）。

示例 (Example)

Request:

代码块

```

1  {
2      "name": "CI_Test_Preference",
3      "settings": {
4          "qc_min_genes": 500,
5          "clustering_resolution": 1.2
6      },
7      "new_name": "CI_Test_Preference_Renamed"
8 }
```

代码块

```

1  {
2      "id":1,
3      "preference_name":"CI_Test_Preference_Renamed",
4      "settings":{
5          "qc_min_genes":500,
6          "clustering_resolution":1.2
7      },
8      "created_at":"2026-02-10T05:42:49.804753",
9      "updated_at":"2026-02-10T05:42:53.987221"
10 }
```

4.5 删除用户偏好 (Delete User Preference)

接口概述 (Purpose) & 前端触发逻辑

用途:

软删除指定名称的偏好设置。

前端触发逻辑:

用户在偏好管理列表中点击“删除”按钮。

接口路径 (URL / Method / Version)

- Endpoint:** /api/v1/user/preference/delete
- Method:** DELETE
- Version:** v1

请求参数 (Request)

Location: Query Parameters (URL)

参数名	类型	必选	说明	示例
name	string	Yes	要删除的偏好名称。	"Outdated Config"

返回参数 (Response)

无内容 (Body 为空)。

错误码 (Error Codes)

HTTP Code	业务 Error Code	错误信息 (Message)	解决方案
204	-	(No Content)	删除成功。
404	-	"Preference '{name}' not found or already deleted."	尝试删除不存在或已删除的记录。

示例 (Example)

Request:

```
DELETE /api/v1/user/preference/delete?name=Outdated%20Config
```

Response:

Status: 204 No Content