RNA-seq 与多智能体功能框架横向比较报告

生成时间：2026-05-22 15:00:09
本地项目：`/home/xia-lab/wutx/00test/bbav5_0502_Yang_Tihuan.7880`
外部参考：`BioScriptHub/RNAseq_pipeline`，本地读取副本 `/tmp/RNAseq_pipeline_BioScriptHub`，commit `5cde0c3`，提交日期 2026-05-16
外部仓库地址：https://github.com/BioScriptHub/RNAseq_pipeline

1. 结论先行

这次通读后，需要纠正一个前提：本项目的 RNA-seq 和多智能体设计并不只是一个线性五步流程。它已经具备：

- 自然语言入口到阶段执行的多智能体链路；
- `PlanningAgentV2` 内置 preflight；
- `/api/execute` 的语义 preflight；
- `DiagnosisAgent` 的 preflight/postflight/failure 门控诊断；
- `stage_capabilities + ArtifactResolver` 的 manifest/上下文驱动输入解析；
- Celery 多队列和 ResourceGovernor 的 CPU 令牌调度；
- RNA-seq 主流程外的 `sample_qc / fgsea / wgcna / candidate_family` 解释层；
- 面向基因家族系统的候选家族/结构域排序能力。

外部 BioScriptHub pipeline 的优势不在“多智能体”或“分析深度”，而在 RNA-seq 命令行流水线的工程收口：一键 `run_all.sh`、显式 `00_preflight_check.sh`、MultiQC、项目级 HTML 报告、featureCounts 属性自动试探、样本并行/单样本线程拆分、HISAT2 到 sorted BAM 的流式落盘、resume 开关。这些是本项目可以吸收的工程做法，但不应该照搬它的命名、目录和报告结构。

2. 本项目 RNA-seq 功能框架

2.1 阶段总览

层级	阶段	脚本	主要作用	当前定位
原始数据处理	`rnaseq_qc`	`apps/cli/run_rnaseq_qc.py`	FASTQ 输入校验、FastQC、cutadapt、clean FASTQ、QC markdown/manifest	核心执行层
比对	`rnaseq_align`	`apps/cli/run_rnaseq_align.py`	HISAT2 比对、samtools view/sort/index、sorted BAM、比对统计	核心执行层
定量	`rnaseq_quant`	`apps/cli/run_rnaseq_quant.py`	StringTie、GFF/GFF3 转 GTF、featureCounts、count/FPKM/TPM	核心执行层
样本质控	`rnaseq_sample_qc`	`apps/cli/run_rnaseq_sample_qc.py`	DESeq2 VST、PCA、样本相关性、距离热图、聚类	解释/质量层

层级	阶段	脚本	主要作用	当前定位
差异表达	`rnaseq_de`	`apps/cli/run_rnaseq_de.py`	DESeq2、比较组、MA/火山图、全基因/家族 foreground	核心统计层
富集	`rnaseq_enrich` / `rnaseq_go_enrich` / `rnaseq_kegg_enrich`	`apps/cli/run_rnaseq_enrich.py`	GO/KEGG 富集、gene2go/gene2ko/OrgDb、植物 KEGG 过滤、pathview	生物解释层
GSEA	`rnaseq_fgsea`	`apps/cli/run_rnaseq_fgsea.py`	基于排序统计量的 KEGG fgsea，支持植物通路过滤	生物解释层
共表达	`rnaseq_wgcna`	`apps/cli/run_rnaseq_wgcna.py`	VST/count 输入、WGCNA 模块、软阈值、模块性状关联	高阶解释层
候选家族	`rnaseq_candidate_family`	`apps/cli/run_rnaseq_candidate_family.py`	DEG + InterPro/Pfam + GO/KEGG 上下文，排序候选家族/结构域	基因家族系统特色层

2.2 `rnaseq_qc`

当前实现：

- 输入来自 `samples.yaml`，包含样本、R1/R2 路径。
- 执行前检查 R1/R2 是否存在、是否非空、R1/R2 文件大小是否严重不平衡。
- 使用 `FastQC` 对 raw FASTQ 做质控。
- 使用 `cutadapt` 做接头和质量过滤，默认 adapter 为 `AGATCGGAAGAGC`，质量阈值 20，最短长度 50。
- 生成 clean FASTQ、FastQC HTML/zip、cutadapt log、Markdown 报告和 manifest。
- manifest 记录 artifacts、metrics、工具版本、sha256、路径。

当前短板：

- 没有 MultiQC 汇总。
- 没有样本级并行参数，多个样本基本按循环执行。
- 质控工具用 cutadapt，而外部 pipeline 用 fastp；两者都常规，但 fastp 的 JSON/HTML 与 MultiQC 集成更顺。
- 部分 manifest 参数仍有硬编码倾向，例如报告中记录的 CPU 参数不一定完全等于调度后的有效 CPU。

2.3 `rnaseq_align`

当前实现：

- 有非常重要的"基因组索引锁定"逻辑：从 prefix 提取物种 token，只接受 `{species}.genome.*.ht2` 或对应 genome FASTA 构建出的 index。
- 避免误用蛋白 HISAT2 index，这是本项目比一般脚本更稳的一点。
- 支持自动查找 clean FASTQ，包括 `final/` 中带 prefix 的 QC 产物。
- HISAT2 输出 SAM，然后 `samtools view` 转 BAM，`samtools sort` 生成 sorted BAM，最后 index。
- final 里已经是 sorted BAM：`{sample}.sorted.bam` 和 `{sample}.sorted.bam.bai`。
- 解析 HISAT2 log 得到 total reads、aligned reads、unaligned reads、alignment rate、unique/multiple alignments。

当前短板：

- 虽然 final 是 sorted BAM，但中间会先落地大 SAM 和 unsorted BAM；外部 pipeline 使用 `hisat2 | samtools sort` 流式直接得到 sorted BAM，磁盘压力更小。
- 当前没有显式暴露 `--rna-strandness`、splice-site 模式、min/max intron length。
- 没有 `samtools flagstat/stats` 作为标准 BAM QC，也没有 MultiQC BAM 汇总。

2.4 `rnaseq_quant`

当前实现：

- 输入 sorted BAM 和 GTF/GFF/GFF3。
- 每个 BAM 跑 StringTie，得到 sample-level StringTie GTF 和 coverage。
- 如果输入是 GFF/GFF3，会尝试 `gffread -T` 转 GTF；失败时用 Python fallback 根据 gene/mRNA/exon/CDS 层级补 `gene_id` 和 `transcript_id`。
- featureCounts 当前固定参数为 `-t exon -g gene_id -p -B -C -T cpu`。
- 输出 count matrix、FPKM、TPM、gene length、normalization info、report、manifest。

当前短板：

- 没有 `feature_type` 和 `group_attribute` 的自动探测。
- 没有根据 Assigned 低比例自动换属性重试。
- featureCounts summary 和 assigned ratio 没有作为一等指标进入 manifest。
- 对 NCBI GFF3，固定 `gene_id` 可能不如 `gene / locus_tag / Parent / ID` 自动候选稳。

不会打乱后续分析的关键原则：

- 可以在 quant 阶段内部试探不同 `-t/-g` 组合，但最终只落一个标准 count matrix。
- manifest 记录最终选择的 `feature_type`、`group_attribute`、Assigned 率、候选尝试列表。
- 后续 `rnaseq_de` 只依赖 count matrix 的第一列 gene_id 和样本列，不关心 featureCounts 曾用哪个 attribute。
- 风险在于不同 attribute 会改变 gene_id 命名体系，所以必须把最终 gene_id universe 固化，并在 enrich/candidate 阶段做 ID overlap 检查。

2.5 `rnaseq_sample_qc`

当前实现：

- 这是我之前容易看漏的一层。PCA、样本相关性热图、样本距离热图和聚类并不是没有，而是放在独立阶段 `rnaseq_sample_qc`。
- 优先读取 count matrix，也可从配置/YAML 推断样本分组。
- R 端使用 DESeq2 VST，输出 VST matrix、PCA 数据、PCA 方差、PCA 图、sample correlation matrix/heatmap、sample distance matrix/heatmap、sample clustering。
- 生成 manifest、report、commands.sh、environment.txt、R script。

与外部 pipeline 的区别：

- 外部把 VST/PCA/correlation 放在 DESeq2 主脚本里，并由最终 HTML 报告收口。
- 本项目拆成单独阶段，模块边界更清晰，也方便只做样本 QC；但总报告没有把它统一串起来时，用户会觉得"DE 阶段没给 PCA"。

2.6 `rnaseq_de`

当前实现：

- 输入 count matrix 和 group/sample YAML。
- 支持从 `groups` 或 sample metadata 中读取分组，按 count matrix 列顺序对齐。
- 有 replicate policy，默认允许单重复但记录 replicate violations。
- 支持 pairwise comparisons，也有 time LRT / spline LRT 的计划能力。
- 输出每个比较的 DE_results、DE_genes、MA plot、volcano plot。
- 支持 family foreground：如果提供 `final_members.id`，可输出 family-specific DE 结果。
- all-zero count matrix 等边界条件会写空表和 `NO_EXPRESSION_SIGNAL.txt`，避免直接崩掉。
- manifest 记录 comparison stats、family analysis、replicate counts、replicate violations、de_plan。

当前短板：

- 候选基因排序还没有成为 DE 阶段标准 artifact。
- UpSet、top candidate gene ranking、标准化 count/VST 目前没有作为 `rnaseq_de` 的固定输出集合；其中 VST/PCA 在 `rnaseq_sample_qc`，FPKM/TPM 在 `rnaseq_quant`。
- 多比较之间的“同一基因跨比较证据”还没有固化成候选排序。

2.7 `rnaseq_enrich`

当前实现：

- 支持 GO、KEGG 或 both。
- 背景可以来自上传的 `gene2go/gene2ko`，也支持部分 OrgDb 物种。
- KEGG 支持 KO 到 pathway 映射、KEGG REST cache、植物兼容过滤、pathview 图。
- 有 foreground/background ID mismatch 的强检查，重叠为 0 时会明确失败并给示例。
- 支持 global 和 family foreground。
- manifest 记录 DE gene 输入、GO terms、KEGG pathways、背景基因数、overlap、raw/filtered KEGG 等。

本项目优势：

- 对植物和非模式物种更友好。
- 不强行套人类 OrgDb/MSigDB。
- 与 annotation 阶段、候选家族阶段能衔接。

2.8 `rnaseq_fgsea`

当前实现定位：

- 用全量 DESeq2 排序结果做 fgsea，而不是只对显著 DEG 做 over-representation。
- pathway 集合来自 `gene2ko` 和 KEGG KO/species cache。
- 支持按物种名推断 KEGG organism，例如 Arabidopsis、rice、maize、soybean、tomato、wheat 等。
- 默认过滤明显动物/人类疾病通路，更适合植物数据。

2.9 `rnaseq_wgcna`

当前实现：

- 可直接复用 `rnaseq_sample_qc` 的 VST matrix。
- 如果没有 VST，也可从 count matrix 构建 DESeq2 VST。
- 支持 `min_mean`、`max_genes`、`min_module_size`、`merge_cut_height`、`powers`。
- 做 goodSamplesGenes、样本聚类、pickSoftThreshold、模块识别、模块 eigengene、模块-性状关联等 WGCNA 常规流程。
- 通过 `pipeline_backend.py` 自动寻找 VST/count matrix、sample_table 或 samples.yaml。

定位：

- 外部 BioScriptHub pipeline 没有 WGCNA。
- 这是本项目 RNA-seq 解释层的扩展优势，尤其适合“候选基因家族”和“调控模块”问题。

2.10 `rnaseq_candidate_family`

当前实现：

- 这不是普通 DE gene ranking，而是候选家族/结构域排序。

- 输入 global DE genes、全蛋白 FASTA、可选 InterProScan TSV、GO/KEGG 富集上下文。
- 支持 TF 家族规则：SPL/SBP、MYB、WRKY、bHLH、bZIP、NAC、AP2/ERF、MADS、TCP、GRAS、C2H2、Homeobox、GATA、HSF、ARF。
- 支持常见功能结构域：kinase/RLK、LRR、NBS-LRR、F-box、ubiquitin、P450、PPR、ABC transporter、transporter、AAA ATPase 等。
- 若有已有 InterProScan，优先复用；否则可按 `auto/always/never` 决定是否运行。
- 输出 candidate family ranking、gene annotations、enrichment context、report、manifest。

本项目优势：

- 与“基因家族分析系统”定位高度一致。
- 外部 pipeline 只做 candidate genes；本项目可以进一步回答“下一步应该跑哪个家族 identify/phylogeny/motif/domain/expression”。

建议补齐：

- 在 `rnaseq_de` 增加 gene-level candidate ranking。
- 在 `rnaseq_candidate_family` 保持 family/domain-level ranking。
- 总报告里并列放 `Top Candidate Genes` 和 `Top Candidate Families/Domains`。

3. 本项目多智能体功能框架

3.1 总体链路

本项目不是简单“用户点按钮跑脚本”。实际执行链路大致是：

1. 前端/API 接收自然语言或阶段执行请求。
2. Router 判断 FAQ、咨询、分析、解读、可行性、报告增强等模式。
3. IntentAgentV2 解析目标阶段、物种、家族、job、prefix、参数。
4. PlanningAgentV2 做 preflight、补依赖、生成阶段计划、应用学习参数。
5. `/api/execute` 对明确阶段执行再做语义 preflight 和 artifact resolve。
6. Celery 将任务分配到 default/heavy/medium/light/rnaseq 队列。
7. ResourceGovernor 根据 CPU token 和最大并发数发放资源租约。
8. StageToolRegistry 调用 `pipeline_backend.run_stage()`。
9. 每个 stage 执行对应 `apps/cli/run_*.py`，产出 final artifacts 和 manifest。
10. DiagnosisAgent 在 preflight/postflight/failure 三个门控点做审计和部分自动修复。
11. ResultParser/Discovery/Knowledge 负责结果解读、发现建议和报告增强。

3.2 LangGraph 编排器

核心文件：`gfa_core/mas/core/langgraph_orchestrator.py`

已定义节点：

- `router`
- `intent`
- `knowledge`
- `learning_recommend`
- `planning`
- `preflight`
- `execution`
- `validator`
- `diagnosis`
- `result_parser`
- `discovery`
- `learning`

主要流程：

- FAQ/咨询/解读/可行性/报告增强：走 Knowledge。
- 分析任务：Router -> Intent -> Learning recommend -> Planning -> Preflight -> Execution。

- 执行失败：Execution -> Diagnosis -> 可 retry/等待确认/skip/fail。
- 结果解析后：ResultParser -> Validator -> Discovery -> Learning 或 Replan。
- Discovery 可触发重新规划，最多有限次数。

注意：

- 代码注释说“9-Agent”，实际初始化后是合并后的 7 个核心 agent，并在图里拆出多个节点。
- PlanningAgentV2 注释明确写了“已合并 PreFlight”，所以你的“多智能体有 preflight”这个判断是对的。

3.3 Router 与自然语言触发

核心文件：`gfa_core/mas/agents/router.py`

Router 使用规则 + LLM fallback：

- 执行动词包括“帮我、执行、运行、开始、启动”等。
- 分析关键词包括“分析、鉴定、构建、计算、生成、提取、下载、ncbi”等。
- 问询动词优先归为 FAQ，避免“有没有”触发执行。
- 低置信度时调用 LLM 分类。

因此，“通过自然语言执行 RNA-seq 全流程”在架构上是可以触发的；关键取决于 IntentAgent 和 PlanningAgent 是否把“执行 rnaseq 分析”展开成 `rnaseq_qc -> rnaseq_align -> rnaseq_quant -> rnaseq_sample_qc -> rnaseq_de -> enrich/fgsea/candidate` 这种计划，以及前端是否要求用户确认必要参数。

3.4 PlanningAgentV2 的 preflight

核心文件：`gfa_core/mas/agents/planning_v2.py`

它做的 preflight 包括：

- 根据 intent stages 调用 `complete_dependencies()` 补齐依赖。
- 检查 stage 是否已接入执行器。
- 检查已有 manifest；成功或 partial_success 的阶段会标记可跳过。
- 检查依赖阶段状态。
- 检查输入文件。
- 优先使用 `ProjectContextIndex + ArtifactResolver + stage_capabilities` 做语义输入解析。
- 无上下文时回退到旧版 glob 文件检查。
- 检查资源。

和“scoped”的关系：

- preflight 是执行前检查：确认输入、依赖、阶段状态、资源、是否需要用户确认。
- scoped 是分析边界/物种/成员集上下文：例如 target species、reference species、member_set_id、scope_file。
- scoped 不是 preflight 的替代品，它提供 preflight 和执行阶段需要理解的上下文。
- 你的系统里二者是叠加关系：preflight 会读取 scope/context 来避免多物种任务中误选一个物种。

3.5 `/api/execute` 的语义 preflight

核心文件：`apps/web/vue3/main.py`

关键逻辑：

- 构建 `ProjectContextIndex`。
- 用 `ArtifactResolver` 解析 stage capability 的 required inputs。
- 如果缺 artifact，会返回 `preflight_failed`，附带 suggested upstream stages。
- 如果存在多目标物种 identify 这类歧义，会返回 `needs_confirmation`。
- 通过后把 `semanticPreflight` 和 `resolvedArtifacts` 写入任务参数，交给 Celery。

这和 PlanningAgentV2 的 preflight 有重叠，但入口不同：

- PlanningAgentV2 更偏“自然语言规划前的全流程检查”。
- `/api/execute` 更偏“用户已经选择某个 stage 后的执行门控”。

- Celery 里还会再跑 DiagnosisAgent 的 preflight gate，这是执行前最后一道保险。

3.6 DiagnosisAgent 的三段门控

核心文件：`gfa_core/mas/agents/diagnosis.py`

覆盖 RNA-seq 阶段：

- `rnaseq_qc`：samples.yaml、fastq pairs、clean fastq、qc metrics。
- `rnaseq_align`：clean fastq、genome index、BAM 非空、alignment rate。
- `rnaseq_quant`：BAM、GTF/GFF、featureCounts、count matrix、FPKM/TPM。
- `rnaseq_de`：count matrix、samples.yaml、comparisons、DE results、significant gene counts。
- `rnaseq_sample_qc`：count/VST matrix、sample table、PCA、correlation。
- `rnaseq_go_enrich` / `rnaseq_kegg_enrich` / `rnaseq_enrich`：DE results、背景注释、ID overlap。
- `rnaseq_fgsea`：DE ranks、gene2ko、rank-background overlap。
- `rnaseq_wgcna`：expression/count matrix、sample table、module count、soft threshold。
- `rnaseq_candidate_family`：DE results、annotation sources、candidate family ranking。

自动修复与策略：

- 有错误模式匹配，如 missing input、memory、dependency failed、tool not found、参数格式等。
- 有 runtime repair codes，例如 manifest recovery、protein stop marker clean、Pfam fallback、GFF overlap report。
- 对 RNA-seq 也有特定审计函数，例如 `_audit_rnaseq_quant`、`_audit_rnaseq_fgsea`、`_audit_rnaseq_wgcna`。

3.7 Stage capabilities 与 ArtifactResolver

核心文件：

- `gfa_core/mas/stage_capabilities.py`
- `gfa_core/mas/artifact_resolver.py`

设计意义：

- 用机器可读方式描述每个 stage 消耗什么 artifact、产出什么 artifact、有哪些 preflight/postflight/repair。
- ArtifactResolver 从 manifest、final 文件、input 文件、canonical 路径中找候选输入。
- 每个候选有 kind、source、role、species、family、score、reason。
- 解析结果不只是“找到了文件”，还会保留候选和决策轨迹，方便 UI 解释。

这是本项目比外部 pipeline 更复杂也更强的地方。外部 pipeline 是固定目录固定脚本；本项目要在自然语言、多物种、多阶段、历史 manifest、scope 上下文之间解析正确输入。

3.8 Celery 队列和资源调度

核心文件：

- `gfa_core/mas/stage_catalog.py`
- `tasks.py`
- `celery_app.py`
- `start_celery_7880.sh`
- `gfa_core/mas/resource_governor.py`

当前设计：

- 队列：`default`、`heavy`、`medium`、`light`、`rnaseq`。
- RNA-seq 阶段统一进入 `rnaseq` 队列。
- RNA-seq 默认 CPU：
  - `rnaseq_qc`：default 8，max 10。
  - `rnaseq_align`：default 8，max 10。
  - `rnaseq_quant`：default 6，max 8。
  - `rnaseq_sample_qc/de/wgcna/enrich/fgsea/candidate_family`：default 4，max 4。
- `start_celery_7880.sh` 默认启动 `rnaseq` worker 并发 2。
- ResourceGovernor 默认总 CPU token 20，最大 running tasks 11。

- 任务执行前调用 `ResourceLease.wait()`，资源不足则进入 `WAITING_RESOURCE` 状态。

当前短板：

- 已有并行执行和 CPU 限流，但还不是按机器性能自动推导。
- 资源模型主要是 CPU token 和最大任务数，没有显式 RAM token、磁盘 IO token、单样本线程 vs 样本并行的拆分策略。
- 外部 pipeline 的 `ALIGN_PARALLEL_JOBS + HISAT2_THREADS_PER_SAMPLE` 这种二维线程模型值得吸收。

4. 与 BioScriptHub/RNAseq_pipeline 横向比较

4.1 总体定位

维度	本项目	BioScriptHub/RNAseq_pipeline	判断
项目定位	多智能体基因家族分析系统，RNA-seq 是其中一条证据链	独立 RNA-seq 命令行流水线	本项目更大、更复杂；外部更聚焦
使用入口	自然语言、前端 API、单 stage 执行、Celery	`run_all.sh` 和一组 shell/R 脚本	外部一键体验更清晰
执行范式	Agent 规划 + stage registry + manifest + diagnosis	固定顺序 shell pipeline	本项目灵活，外部稳定直观
结果定位	面向基因家族研究和候选家族发现	面向标准 RNA-seq 报告	两者目标不同

4.2 输入与配置

维度	本项目	外部 pipeline	可借鉴点
样本输入	`samples.yaml`，可手写，也可由自然语言/前端生成后确认	`metadata.tsv`，也可从 FASTQ 名称自动生成	本项目不差；需要更低门槛的表格编辑 UI
对比输入	YAML 内 groups/comparisons 或自动 pairwise	`contrasts.csv/tsv`	可以在 UI 中提供“比较组表格”，生成 YAML
用户填写成本	YAML 对机器友好，对普通用户不如表格直观	TSV/CSV 对生信用户直观	本项目适合做弹窗表格 -> YAML/JSON 的转换
自动推断	依赖 Intent/Planning/ArtifactResolver	从 FASTQ 命名推断 sample/condition	外部的 FASTQ 命名推断可作为一个辅助导入器

结论：你的 YAML 设计不是差，而是“适合系统内部和高级用户”。为了开源，应增加普通用户层：上传 FASTQ 后自动生成样本表，用户在弹窗表格里确认，后端再转成完整 YAML。

4.3 Preflight

维度	本项目	外部 pipeline	判断
是否有 preflight	有，而且不止一层	有 `00_preflight_check.sh`	本项目不是缺 preflight
形式	Agent 内部结构化结果/API 返回/Diagnosis gate	独立脚本，输出 TSV handoff plan	外部更显式、更容易开源用户理解
检查内容	stage 状态、manifest、依赖、输入解析、scope、资源、诊断	config、metadata、contrasts、FASTQ、replicate、工具、reference	两者侧重点不同

维度	本项目	外部 pipeline	判断
产物	API JSON、task meta、diagnosis report、manifest 依赖	`logs/preflight_check.tsv`、`logs/pipeline_handoff_plan.tsv`	本项目应补一个正式 `rnaseq_preflight` artifact/report

建议：不是新增一个替代 scoped 的东西，而是把已有 preflight 固化成用户可见 artifact，例如：

- `{prefix}.rnaseq_preflight.manifest.json`
- `{prefix}.rnaseq_preflight_report.md`
- `{prefix}.rnaseq_preflight.inputs.tsv`
- `{prefix}.rnaseq_preflight.plan.tsv`

4.4 QC

维度	本项目	外部 pipeline	判断
原始质控	FastQC	FastQC + MultiQC	本项目缺 MultiQC 汇总
过滤工具	cutadapt	fastp	两者都常规，fastp 报告集成更方便
输出	clean FASTQ、FastQC、cutadapt log、MD report、manifest	clean FASTQ、fastp report、FastQC、MultiQC	可增加 MultiQC，不必改掉 cutadapt
并行	当前阶段内不突出样本并行	样本并行 + 单样本线程	外部并行拆分更直接

4.5 比对

维度	本项目	外部 pipeline	判断
index 选择	有物种锁定，防误用蛋白 index	config 指定/准备 reference	本项目更安全
中间文件	SAM -> BAM -> sorted BAM	HISAT2 pipe 到 samtools sort	外部更省磁盘
final BAM	sorted BAM	sorted BAM	二者最终一致
strandness/splice	当前不显式暴露	config 中有 HISAT2 RNA strandness、splice mode、intron length	外部参数更完整
BAM QC	解析 HISAT2 log	samtools flagstat/stats + MultiQC	本项目可补

4.6 定量

维度	本项目	外部 pipeline	判断
定量工具	StringTie + featureCounts	featureCounts	本项目输出更丰富
GFF3 处理	gffread + Python fallback 转 GTF	featureCounts attribute auto	本项目对格式转换更主动；外部对属性选择更稳
feature type	固定 exon	`FEATURE_TYPE` 可设，支持 auto 思路	本项目应显式化
group attribute	固定 gene_id	`GROUP_ATTRIBUTE=auto` 尝试 gene_id/gene/locus_tag/Parent/ID	外部对 NCBI GFF3 更鲁棒
downstream 稳定性	count matrix 标准化输出	count matrix 标准化输出	只要最终 gene_id 固定，auto 不会破坏后续

4.7 样本 QC 与 DE

维度	本项目	外部 pipeline	判断
PCA	有，在 `rnaseq_sample_qc`	有，在 DESeq2 脚本/报告中	本项目不是没有，而是拆分阶段
样本相关性	有	有	二者都有
VST matrix	有，在 sample_qc	有，在 DESeq2 输出	本项目应在总报告中串起来
replicate policy	支持记录 violations，可放宽	外部要求条件至少 2 重复	外部更严格；本项目更宽容但要报告风险
DE 输出	per comparison DE_results/DE_genes/MA/volcano，支持 family foreground	all/sig/up/down、MA/volcano、summary	外部标准化报告更整齐
UpSet	当前未固化为标准 artifact	有	可借鉴
candidate gene ranking	当前未固化	有基础 score	可借鉴但扩展

4.8 富集、fgsea、WGCNA、候选

维度	本项目	外部 pipeline	判断
GO/KEGG ORA	强，支持 gene2go/gene2ko/OrgDb/植物 KEGG/pathview	有基础 enrichment	本项目更适合非模式植物
fgsea	有	无	本项目优势
WGCNA	有	无	本项目优势
candidate genes	尚未作为 DE 标准产物	有 `candidate_genes`	外部值得吸收
candidate family/domain	有 `rnaseq_candidate_family`	无	本项目特色

4.9 报告

维度	本项目	外部 pipeline	判断
stage 报告	每阶段 Markdown + manifest	各脚本输出 + HTML 总报告	本项目可追溯性强
MultiQC	暂无标准集成	有 `03_multiqc`	本项目应补
项目级 HTML	目前缺 RNA-seq 总报告收口	`14_report/RNAseq_analysis_report.html`	外部值得借鉴，但不要照搬结构和命名
报告目标	支持后续解读，但分散	一眼看完整 RNA-seq	本项目需要项目级 report assembler

建议本项目报告命名：

- `{prefix}.rnaseq_project_report.html`
- `{prefix}.rnaseq_project_report.md`
- `{prefix}.rnaseq_project_report.manifest.json`
- `{prefix}.rnaseq_multiqc_raw.html`
- `{prefix}.rnaseq_multiqc_clean.html`
- `{prefix}.rnaseq_multiqc_bam.html`

不要使用外部的 `14_report/RNAseq_analysis_report.html` 目录和命名。

4.10 Resume/断点续跑

维度	本项目	外部 pipeline	判断
规划层跳过	PlanningAgentV2 检查 manifest，已完成阶段可跳过	`RESUME_MODE` 下脚本跳过已有输出	本项目已有高层断点逻辑
执行层跳过	部分阶段会找已有 final/manifest，但并非统一 resume API	每个 shell 脚本有 resume skip	外部 stage 内 resume 更直接
诊断	DiagnosisAgent 可判断 manifest 缺失/失败/partial	preflight/check logs	本项目诊断更强
建议	固化统一 `--resume/--force/--from-stage` 语义	已有	本项目需要统一口径

断点续跑不是外部 pipeline 的"稀奇点"；你的规划智能体确实已经承担了高层断点。但开源用户也需要命令行可理解的低层 resume 语义。建议两层都保留：

- 高层：Agent 根据 manifest 和依赖自动跳过。
- 低层：每个 `run_rnaseq_*.py` 接受 `--resume/--force`，判断关键输出是否完整。

5. 关于用户前面关心的具体问题归纳

5.1 "RNA-seq 总控入口"

架构上已经能通过自然语言触发完整流程，但需要确认两点：

- IntentAgent 是否把"执行 RNA-seq 分析"稳定识别为完整 RNA-seq pipeline，而不是单个 `rnaseq_qc`。
- PlanningAgent 是否根据已有输入和用户目标自动补 `rnaseq_qc -> align -> quant -> sample_qc -> de -> enrich/fgsea/candidate`。

建议增加一个显式虚拟 stage：`rnaseq_pipeline`。它不直接执行脚本，只展开计划，并生成总 manifest/report。这样既保留自然语言入口，也给开源用户一个明确入口。

5.2 "preflight 和 scoped 的区别"

- preflight：执行前检查，回答"能不能跑、缺什么、会不会误选、是否需要确认"。
- scoped：分析范围，回答"跑哪几个物种、哪个家族、哪个成员集、target/reference/comparative 怎么定义"。
- scoped 不是替代 preflight；它是 preflight 判断输入和歧义时的重要上下文。

5.3 "降低用户填表成本"

当前 YAML 的优势：

- 结构清晰，适合系统内部传参。
- 可手写，适合高级用户。
- 可表达复杂字段，如 group、condition、comparison、batch、time、replicate policy。

当前 YAML 的复杂点：

- 普通用户不直观看出样本表缺了哪列。
- 文件路径、样本名、condition、comparison 混在一个嵌套结构里。
- 出错时用户需要理解 YAML 缩进和 schema。

建议 UI：

- FASTQ 上传后自动识别 R1/R2。
- 弹窗表格列：sample_id、condition、replicate、fastq_1、fastq_2、batch、time。
- 对比组表格列：comparison、numerator、denominator。
- 用户确认后保存两份：
  - 人可读：`samples.user.tsv`、`contrasts.user.tsv`

    - 系统读：`samples.yaml`

这样不是否定 YAML，而是把 YAML 作为编译产物。

5.4 “MultiQC 和项目级 HTML 总报告”

应该做。目标不是复制外部仓库，而是符合常规 RNA-seq 习惯：

- MultiQC 汇总 raw/clean FastQC、cutadapt/fastp、HISAT2、samtools、featureCounts。
- 项目级 HTML 汇总：
    - 项目与样本信息；
    - 运行参数和环境；
    - QC 概览；
    - 比对率和 BAM QC；
    - featureCounts assigned 率；
    - PCA/样本相关性；
    - DE summary；
    - enrichment/fgsea；
    - WGCNA；
    - Top Candidate Genes；
    - Top Candidate Families/Domains；
    - artifacts index。

建议实现为本项目自己的 `run_rnaseq_report.py`，读取各阶段 manifest 和 final artifacts，而不是按外部目录硬编码。

5.5 “样本并行数 + 单样本线程数”

本项目已有 Celery 队列和 ResourceGovernor，并不是没有并行。但它解决的是“多个任务/阶段之间如何并发”，外部 pipeline 的参数解决的是“一个 RNA-seq stage 内多个样本如何并发、每个样本分配几个线程”。

建议补一个资源策略函数：

- 读取 `lscpu`、内存、磁盘空间。
- 估算 total threads。
- 按 stage 类型分配：
    - QC：多样本并行高，单样本线程低。
    - align：样本并行中等，单样本 HISAT2 线程中等。
    - sort：考虑 samtools sort 内存。
    - quant：featureCounts 通常一个全局线程参数。
    - R 阶段：CPU 不宜太高，内存更关键。
- 输出到 manifest：`requested_cpu`、`effective_cpu`、`sample_parallel_jobs`、`threads_per_sample`、`memory_policy`。

5.6 “featureCounts auto 会不会打乱后续分析”

不会必然打乱，但必须做成“试探阶段内部逻辑”，不能让后续看到多个版本的 count matrix。

安全实现：

1. 默认先试 `-t exon -g gene_id`。
2. 如果 featureCounts 失败或 Assigned 率低于阈值，再尝试候选：
    - feature type：`exon/CDS/gene`
    - group attribute：`gene_id/gene/locus_tag/Parent/ID`
3. 选择 Assigned 率最高且 gene 数合理的组合作为正式输出。
4. 只生成一个标准 `{prefix}.count_matrix.txt`。
5. manifest 记录全部尝试和最终选择。
6. `rnaseq_de/enrich/candidate` 只消费正式 count matrix。

5.7 “PCA 你是不是看漏了”

是的，PCA 在 `rnaseq_sample_qc`，并且相关图、VST、样本相关性热图都有。更准确的修订不是"新增 PCA"，而是"总报告收口 PCA/相关性，并让 DE 报告引用 sample_qc artifacts"。

5.8 "strandness、splice_mode、max_intronlen、feature_type"

当前本项目：

- `rnaseq_align` 没有显式暴露 HISAT2 RNA strandness、known splice site、min/max intron length。
- `rnaseq_quant` 固定 `featureCounts -t exon -g gene_id`，没有显式 `feature_type/group_attribute/strandness`。

这些参数意义：

- `strandness`：文库链特异性。错设会显著影响 featureCounts assigned reads，甚至方向相反。
- `splice_mode`：是否使用注释中的 splice sites 指导 HISAT2。对真核 RNA-seq 比对质量有影响。
- `max_intronlen`：最大内含子长度。物种差异很大，植物/动物/真菌可能不同，影响跨剪接比对。
- `feature_type`：计数哪些 feature，常见 exon/CDS/gene。GFF3 结构不同会影响 assigned rate。
- `group_attribute`：按哪个属性归并 reads 到基因。NCBI GFF3 不一定有标准 `gene_id`。

建议：这些参数不要强迫普通用户都填。UI 用高级参数区，默认 auto；manifest 必须记录最终值。

5.9 "候选基因排序要不要做"

建议做，而且分两层。

第一层：gene-level candidate ranking，放进 `rnaseq_de` 或新增轻量后处理：

- `{comparison}.candidate_genes_ranked.tsv`
- `{comparison}.top50_candidate_genes.tsv`

建议字段：

- `gene_id`
- `comparison`
- `regulation`
- `baseMean`
- `log2FoldChange`
- `padj`
- `candidate_score`
- `normalized_mean`
- `multi_comparison_hits`
- `family_member`
- `go_terms`
- `kegg_terms`
- `annotation`

基础分数：

```
candidate_score =
abs(log2FC) * -log10(padj)
+ multi_comparison_bonus
+ family_member_bonus
+ enrichment_hit_bonus
```

注意：报告中必须写清楚 candidate score 是优先级排序，不是新的统计显著性。

第二层：family/domain-level candidate ranking，继续使用并强化 `rnaseq_candidate_family`。

总报告中增加：

- `Top Candidate Genes`

- `Top Candidate Families/Domains`

这会形成闭环：DE 强度用于挑单基因，结构域/家族排序用于决定下一步跑哪个基因家族 identify、phylogeny、motif、domain、expression。

6. 建议的开源化修订优先级

P0：报告和用户可见收口

1. 新增 `run_rnaseq_report.py`。
2. 汇总各阶段 manifest，生成 `{prefix}.rnaseq_project_report.html/md/manifest.json`。
3. 接入 MultiQC，至少 raw/clean FastQC、HISAT2、samtools、featureCounts。
4. 在报告中明确链接各阶段 artifacts。

P1：配置和 preflight 产品化

1. 增加 `rnaseq_pipeline` 虚拟总控 stage。
2. 把现有 preflight 固化为 `rnaseq_preflight` artifact。
3. 增加样本表 UI：表格确认后生成 YAML。
4. 导出 `samples.user.tsv` 和 `contrasts.user.tsv` 方便开源用户理解。

P2：资源调度升级

1. 在 ResourceGovernor 保持现有 CPU token 基础上，增加机器探测。
2. RNA-seq stage 内增加 `sample_parallel_jobs` 和 `threads_per_sample`。
3. 增加内存/磁盘空间检查，尤其 HISAT2/samtools sort/WGCNA。
4. manifest 写入资源分配决策。

P3：定量鲁棒性

1. featureCounts 支持 `--feature-type auto/exon/CDS/gene`。
2. 支持 `--group-attribute auto/gene_id/gene/locus_tag/Parent/ID`。
3. 以 Assigned 率和 gene 数选择最终 count matrix。
4. 把 featureCounts summary 标准化进入 manifest 和项目报告。

P4：磁盘与 resume

1. HISAT2 直接 pipe 到 samtools sort，减少 SAM/BAM 中间文件。
2. 每个 RNA-seq stage 统一 `--resume/--force`。
3. 高层 Agent 继续基于 manifest 跳过，低层脚本基于关键输出完整性跳过。

P5：候选模块闭环

1. `rnaseq_de` 增加 candidate gene ranking。
2. `rnaseq_candidate_family` 继续负责 family/domain ranking。
3. 项目 HTML 报告并列展示 Top Candidate Genes 和 Top Candidate Families/Domains。
4. 后续可让 DiscoveryAgent 根据候选家族自动建议 `identify -> phylogeny -> motif/domain -> expression`。

7. 本次比较的核心判断

BioScriptHub/RNAseq_pipeline 对本项目最有启发的不是“分析能力更强”，而是“RNA-seq 工程体验更完整”：

- 更像开源用户熟悉的一键 RNA-seq pipeline；
- 报告更集中；
- MultiQC 更符合常规；
- preflight 是显式产物；
- resume 和并行参数更直观；
- featureCounts 对 GFF3 的容错更务实。

本项目的优势则在：

- 多智能体自然语言入口；
- manifest 与 artifact resolver；
- scope-aware 多物种/基因家族上下文；
- 诊断和自动修复；
- Celery 资源队列；
- 植物/非模式物种 enrichment；
- fgsea、WGCNA；
- 候选家族/结构域排序。

因此后续修订方向不是复制外部 pipeline，而是把它的“RNA-seq 工程收口经验”吸收到本项目现有多智能体和基因家族框架里。