在Linux服务器上做样例回归并交接运行说明

完整操作教程 · 参考固定 · 往返恢复验收

固定一份已知输入和验收规则，用ID与数值比较替代只看文件名，并保留失败、差异和环境记录。

运行边界：示例代码需按安装版本和真实项目输入复核；未提供真实生物数据，语法或小样本检查不代表科学结果和性能验收。

迁移流程最后需要回答：相同输入在Linux上是否产生符合预期的结果。最有用的起点是一份小而能覆盖关键分支的固定样例，以及原系统留下的验收结果。不要把整个旧结果目录当标准答案，也不要用“输出文件都在”代替内容比较。

先定义能判断的回归样例

保留样例输入校验值、代码版本、参考发布、参数和随机种子。输入应覆盖典型样本与必要边界，如空结果、少量记录或特殊ID，但不要用真实敏感数据做公开测试。软件版本改变时先说明方法差异，不能悄悄换参考后仍称完全同一流程。

结果类型	比较方式	注意事项
计数与ID表	键集合、行数、列定义及整数值	先对齐ID，不只比较行顺序
浮点统计	约定绝对或相对容差	容差要有方法依据
图与二进制	核心数据、尺寸、结构与必要元数据	时间戳和编码差异可能改变字节
随机算法	固定种子并检查合理统计	跨库或硬件仍可能有差异

用明确键比较小表格

下例只演示sample,value两列CSV。保存为compare_table.py，执行python compare_table.py expected.csv actual.csv。它拒绝空表、重复ID及非有限数，并忽略不影响含义的行顺序；两种容差只是这个数值小例子的设置，不是所有生信结果的通用标准。

```
import csv, math, sys

def load(path):
    with open(path, encoding="utf-8", newline="") as handle:
        rows = list(csv.DictReader(handle))
    assert rows
    result = {}
    for row in rows:
        key = row["sample"].strip()
        value = float(row["value"])
        assert key and key not in result and math.isfinite(value)
        result[key] = value
    return result

expected, actual = (load(path) for path in sys.argv[1:3])
assert expected.keys() == actual.keys(), "sample IDs differ"
for key in expected:
    assert math.isclose(expected[key], actual[key], rel_tol=1e-8, abs_tol=1e-10), key
print("regression_ok", len(expected))
```

例如expected包含A,2和B,3，actual按B再A排列仍可通过；若B缺失或数值变为4，应明确失败。正式流程根据输出单位改写规则，整数计数通常不能用浮点容差放过，基因或细胞ID也不能随意去重后再比较。

在新目录从头执行

在Linux新环境里用一条完整命令运行样例，将标准输出、错误、退出码和新结果放在本轮目录，禁止复用旧成功文件。先跑正常样例，再跑一个已知应失败的输入，确认脚本会非零退出，不会继续发布不完整结果。

遇差异按顺序查输入内容、参数、参考、软件版本和数值实现。不要先调大容差直到通过，也不随意删除不同的记录。如果差异来自合理的方法升级，单独保存新的验收依据与评审结论，不覆盖旧基线。

给回归检查本身留一个已知失败样例，例如删除一行或改动一个数值，确认检查器确实拒绝。只运行正常表格，即使比较代码漏查了关键内容也可能一直显示通过，无法证明验收规则有效。

把运行说明交给另一位成员

说明至少列出环境恢复、项目根目录、启动命令、输入格式、输出清单、验收方式与常见错误。请另一位成员按文档在新会话复跑；若还需作者临时口头补步骤，就继续补齐入口和配置。

回归通过只证明已覆盖样例的行为。正式批次前另测代表规模、工作盘和资源上限；迁出时保留代码、输入清单、版本与必要结果。将运行状态和科学验收分开记录，才能知道失败发生在迁移链路还是分析方法本身。

先定义比较规则

科研软件升级前后，先明确哪些字段必须完全相同，哪些浮点量允许容差，哪些只比较统计趋势。按行比较 CSV 会把顺序改变误判为差异，也可能因浮点格式化误报；应先按稳定记录 ID 对齐，再比较字段。没有唯一 ID 时，先定义组合键并检查重复。脚本要对缺行、多行、重复 ID、额外列和缺失值给出明确失败信息，不能静默取第一条或自动删除。

浮点容差要结合数据尺度，不能所有列用一个宽松阈值。Python `math.isclose` 同时涉及相对和绝对容差；接近零的数值通常要考虑绝对容差。离散标签、整数计数、序列 ID 和结构索引应精确比较。对连续变量，依据数值精度与实验可接受变化事先定规则，并输出最大绝对差、最大相对差和超差记录数。

生成能审阅的差异报告

不要只打印 PASS 或 FAIL。报告应含输入文件校验和、记录总数、匹配数、缺失/多余 ID、重复 ID、逐字段差异、容差和脚本版本。失败时列出有限数量的代表行和字段，完整差异另存文件。大表使用流式读取或分块比较，避免全部装入内存；同时确保排序与键匹配不会改变业务含义。

数值单位、缺失值、科学计数法和精度也需明确。字符串 NA、空单元格、NaN 和真实零不是同一种值；读取 CSV 时指定编码、分隔符、列类型和缺失规则。若软件输出顺序不确定，比较器可按 ID 对齐并用声明过的容差，但不能为了通过门禁过滤重要差异。

验证比较器能发现错误

先用已知通过样例验证，再构造应失败输入：删一行、加重复 ID、修改离散标签、把数值改到容差内与容差外、打乱行顺序、改变列名。脚本应接受允许的行重排，同时拒绝缺记录和关键值变化。合成样例能验证程序逻辑，不代表科研阈值已经合理；阈值仍需研究者确定。

输出写入新的目标路径，避免覆盖原始结果。失败时返回非零状态，方便批处理识别；成功时同时生成摘要和审计信息。让调用者能区分数据不一致、格式错误、文件无法读取和脚本异常，而不是把所有情况统一显示为不通过。

升级环境时建立可复现基线

升级 Python 或科学计算库前，保存代表性输入、旧环境清单、参考结果和脚本版本。新环境使用锁定依赖运行同一工作流，记录硬件、操作系统、线程变量和随机种子。并行计算可能出现小幅浮点差异，应对敏感字段制定容差，而不是看到结果后临时放宽标准。

差异超出容差时，回到最小可复现样例逐步比较中间结果，定位依赖、算法、参数或数据变化。保留旧环境以便复现，不要只留下未经检查的最新版。结果表、比较报告、环境清单和输入校验和共同构成审核记录，帮助团队理解为何接受或回退软件升级。

比较脚本的资源规划

比较程序通常不需要 GPU，但大表会受 RAM、磁盘和读取吞吐限制。用代表性文件测处理速度、内存峰值和报告大小，再按实际行数估算。若比较重复运行，可减少重复解析和不必要副本；超过可用内存时考虑分块、排序归并或数据库索引。申请计算资源时说明输入规模、并发、持久化和数据敏感性，而不是笼统说是生信项目。

交接包应包含比较规则、容差依据、脚本版本、依赖环境、输入清单、报告格式和非敏感示例。这样组内成员能清楚脚本保证了什么、没有保证什么，也能在软件升级或结果变化时重复核对。

在正式科研门禁中加入人工审阅点

自动比较适合发现约定范围内的差异，不适合替研究者决定差异是否具有生物学意义。报告超差后，先检查是否为单位变化、舍入、缺失值编码或软件算法变化，再决定接受、回退或重新分析。阈值修改要留下原因、批准人和日期，避免为了让升级通过而逐步放宽到失去检测能力。

对最终使用的报告保留输入文件校验和、配置、脚本提交版本、运行环境和人工审阅结论。项目的重要结果若发生差异，应链接到相应 issue 或分析记录，并保存旧版与新版输出，确保决策过程可复核，而不是只留下绿色的通过标志。

把脚本设计成可重复运行的命令

比较工具接收输入路径、输出目录和容差参数时，校验参数类型和文件存在性，并在启动时打印清晰的运行配置。将标准输出摘要与错误日志分开，返回稳定的退出码。相同输入与配置重复执行时应生成相同判断结果，避免依赖当前工作目录、系统区域设置或 CSV 自动推断类型。

保存一个脱敏的示例输入和期望报告，供环境升级后快速复核工具行为。交接文档应说明字段定义、排序键、缺失值规则和容差单位；只给脚本本身，接手者仍无法判断 PASS 的含义或报告遗漏了哪些差异。

不要让容差掩盖上游问题

比较出现大量小幅差异时，先确认单位、排序键、随机种子、线程数和依赖版本，而不是立即调大 tolerance。上游输入顺序变化可能改变模型或抽样结果；参考数据库更新也可能产生真实内容变化。把可接受数值波动与数据合同变化分开处理，并把异常差异回溯到产生它的步骤。

对多文件流程，可逐阶段比较中间结果，找到首次偏离点。只比较最终汇总表很难定位问题，尤其当后续聚合把单个样本差异平均掉时。对高价值结论保留足够的中间审计摘要，但需遵循敏感数据存储政策。

形成可审计的接受或回退决定

每次版本比较结束后，记录结论是接受、回退还是需要进一步复核，以及决定依据、审阅人和日期。保留新旧环境文件和结果报告，防止环境更新后无法复现。若某些字段允许变化，应把字段名、容差和科研理由写入规则文件，而不是埋在脚本代码里。

定期用已知通过与已知失败样例复核比较程序，确保后续改动没有削弱检测能力。分析门禁的价值在于及时暴露差异和留存证据，最终科学判断仍由研究团队完成。

保留输入数据的来源与生成方式

参考 CSV 或 TSV 应记录生成程序、来源数据版本、筛选步骤和文件校验和。若新版结果与旧版不一致，先确认输入是否完全相同，再看依赖和代码变化。文件编码、换行、浮点小数位、列顺序和区域设置都可能影响解析；比较工具应将这些格式检查纳入报告，并在发现不支持的格式时明确停止。

涉及大型表格时，报告总行数、唯一 ID 数和每列类型概况，避免只比较抽样行就宣称整体一致。确需抽样检查时，固定抽样种子并记录抽样范围，同时保留全量统计。

版本变化要留书面决策

依赖升级导致数值差异时，记录采用新结果还是回退旧环境，以及对下游结论的影响。对关键差异由负责分析的研究者签字或在项目记录中确认，避免脚本维护者单独决定科学阈值。

配套练习包

本PDF对应的ZIP包含本文代码块、主题练习模板或合成数据、README和SHA256校验文件。请从同一教程页面下载练习包；先核对校验值，再在独立目录复现。

参考资料

Python：math.isclose容差定义 · 来源资料

Python：CSV读取规则 · 来源资料

Conda：环境恢复和平台条件 · 来源资料

操作建议：先在合成小样本上验证流程，再依据项目参考、标签合同、存储策略与获配资源验收真实数据。