把Notebook任务迁到服务器并记录环境与输出

完整操作教程 · 参考固定 · 往返恢复验收

清除交互会话的隐含状态，用固定内核从头执行Notebook，保存新输出、环境和结果检查记录。

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。

Notebook迁到服务器后“有时能跑、有时报变量不存在”，常见原因是本地会话保留了之前执行的对象。迁移验收应从新内核按顺序执行全部单元，而不是打开文件后看到旧图就认为成功。原始Notebook、输入、实际执行产物应分别保留。

先移除路径与交互前提

记录Python版本、主要科学包、外部命令、参考发布和样本清单。不要复制Windows的venv到Linux；在目标系统按已验证依赖重建。Notebook中的盘符、桌面目录、手工上传文件和依赖上一个单元未保存对象的操作，都要改成明确输入。

下面是一段可用于路径验收的小单元：Notebook放在项目根目录，data/counts.csv含sample,value表头；RUN_DIR由启动脚本指定为本轮新目录。它只展示显式路径和基本表格条件，不替代真实生信统计检查。

```
import csv, math, os, sys
from pathlib import Path
print(sys.executable)
root = Path.cwd()
with (root / "data/counts.csv").open(encoding="utf-8", newline="") as handle:
    rows = list(csv.DictReader(handle))
assert rows and all(row["sample"] for row in rows)
assert len({row["sample"] for row in rows}) == len(rows)
values = [float(row["value"]) for row in rows]
assert all(math.isfinite(value) for value in values)
out = Path(os.environ["RUN_DIR"])
assert out.is_dir()
with (out / "summary.txt").open("x", encoding="utf-8") as handle:
    handle.write(f"rows={len(rows)} total={sum(values)}\n")
```

正式分析应把非有限数、缺失值、特征ID和样本对应等检查写进业务代码。路径错误先查工作目录和挂载，不用临时切目录让部分单元偶然成功。输入如果允许为空或重复，必须依据研究设计显式修改规则。

用nbconvert从头执行

前提是目标环境已有nbconvert、所需内核及全部分析依赖。先用kernelspec list确认内核名，KERNEL填实际已验证名称，ENV_DIR是提供执行工具的环境。下面从项目根目录运行analysis.ipynb，OUT为新的结果目录绝对路径。

```
set -eu
: "${ENV_DIR:?■■■■■■■■■■■}"
: "${KERNEL:?■■■■■■■■■■■■■}"
: "${OUT:?■■■■■■■■■■■■■}"
test -r analysis.ipynb && test ! -e "$OUT" || exit 2
mkdir "$OUT"
export RUN_DIR="$OUT"
"$ENV_DIR/bin/jupyter" kernelspec list > "$OUT/kernels.txt"
"$ENV_DIR/bin/python" -m pip freeze > "$OUT/packages.txt"
"$ENV_DIR/bin/jupyter" nbconvert --to notebook --execute analysis.ipynb \
  --ExecutePreprocessor.kernel_name="$KERNEL" \
  --ExecutePreprocessor.timeout=600 \
  --output=executed --output-dir="$OUT" \
  > "$OUT/execution.log" 2>&1
test -s "$OUT/executed.ipynb" && test -s "$OUT/summary.txt" || exit 2
```

600秒是本例单元执行超时，不是整个项目的时限或推荐值，应按任务修改。默认遇单元异常即失败，不添加allow-errors把错误混进“成功”结果，也不使用inplace覆盖原稿。服务环境的包清单不一定等于内核环境，Notebook内还需保存实际解释器及关键包版本。

检查	看什么	失败时处理
执行顺序	新内核全部单元连续通过	清除对旧会话对象的依赖
输入对应	样本、维度、参考版本	查缺失挂载或旧数据
输出内容	新结果可读，数量与数值合理	文件存在不能单独算通过
资源边界	代表规模的RAM、时间和工作盘	小样例不外推整批容量

遇到执行后没有预期文件，先核对单元的实际工作目录和RUN_DIR，不把另一个旧目录中的同名文件搬过来凑验收。若内核在执行中消失，保留日志并查进程或平台的终止原因；只有确认是单元超时，才讨论调整该项限制，不用无限等待掩盖失败。

保留执行记录再扩展任务

核对executed.ipynb的错误输出、关键表格及图形，并与迁移前已知样例比较。浮点与随机算法按预先约定容差和种子检查；看到旧缓存图不能当新结果。超时先定位是慢单元、死循环、外部服务还是资源等待，再调整限制。

长任务将参数、输入和输出移入清楚的脚本或工作流，Notebook保留解释与展示。记录本轮命令、代码校验值、退出状态和结果验收，后续失败能回到具体单元或阶段，而不是依靠不断重启全部服务器。

把环境、输入与Notebook身份一起固化

迁移前给原Notebook、输入文件和依赖说明生成校验值；迁移后确认服务器上的副本与预期一致。记录Python和Jupyter版本、kernel spec指向的解释器、关键包版本、外部二进制路径、参考库发布号及环境变量。pip freeze只记录当前Python环境，不能覆盖操作系统库、外部命令或内核选择；若项目已有锁文件或容器，优先使用并另存构建摘要。不要将带令牌、个人目录或敏感元数据的Notebook输出公开。

从失败单元定位，而非盲目拉长超时

Notebook执行失败时看执行日志中的单元序号、异常栈、stderr和最后一个成功输出。NameError通常指向执行顺序或隐含状态，FileNotFoundError指向工作目录或挂载，Kernel died再查OOM、段错误或平台回收；单元超时才按任务特点调整timeout。保留失败的执行版，修复后新建输出目录重跑，原始Notebook不覆盖。若外部命令在单元内运行，还需记录命令退出码，不要只检查该单元没有异常显示。

定义一次可复核的输出验收

至少确认所有预期单元均有新执行计数且无error输出，输出文件的修改时间属于本轮、文件能被独立读取，并核对行数、样本ID、数值范围或已知小样例。图形也应确认来自新数据。对随机算法按预先制定的容差比较，不能要求浮点结果逐字节相同；出现差异时先排查随机种子、线程和库版本，再讨论科学意义。将验收摘要写入结果目录，方便合作者知道“执行成功”的定义。

什么时候把Notebook迁成脚本或流程

适合交互探索的图表和解释可保留Notebook；重复运行、长时间占用资源、需要断点恢复或多人协作的步骤应提取为参数化脚本或工作流。明确输入输出合同、退出码、日志、检查点和可重跑规则后，再由Notebook调用或展示结果。迁移并不要求把所有内容改写成复杂平台；关键是去掉依赖人工执行顺序，并让一次干净启动可重建所需输出。

给每个Notebook阶段加输入输出清单

迁移前按单元或逻辑阶段列出输入文件、生成文件、依赖软件和计算量，区分一次性交互探索与必须重跑的正式步骤。重要阶段写清输出路径和覆盖策略；如输出已存在，明确是拒绝覆盖、校验后复用还是新建版本。若Notebook顺序依赖隐藏状态，将前置变量写入显式函数参数或配置文件，避免靠“先运行某单元”口头传承。

怎样判断新输出不是旧结果

建立每轮唯一的RUN_DIR，并在开始时确保该目录不存在；结束后检查关键文件的修改时间在本轮范围、散列已写入清单、输出可独立打开。可将原始输入和结果通过manifest记录相对路径与散列，不把旧目录软链接进新结果目录。迁移前后的随机结果不一定逐字相同，应该比较行列、样本顺序、统计摘要和预先设定的容差；若样本ID缺失，即使图形看起来相似也不应通过。

无显示器服务器上的绘图处理

绘图代码迁移到无图形界面节点时，检查所用绘图库是否要求交互后端。设置非交互式后端并把图保存为文件，Notebook输出仍可嵌入图像。区分渲染失败与上游数据为空：先在图前输出数据维度和过滤后记录数，再检查图片文件存在、可读且时间戳为本次执行。若图像字体或布局与本地不同，记录字体和软件版本，不把视觉细节差异误判为统计结果变化。

团队协作时冻结输入版本

多人共同编辑Notebook时，迁移运行前指定一个提交版本或只读副本，记录作者修改时间和代码校验值。执行时使用固定样本表及输入清单；如果数据更新，创建新版本目录并明确哪个Notebook版本对应哪批输入。将日志和结果放入项目可访问的受控路径，避免个人家目录权限使导师或合作者无法复核。敏感数据不应通过把Notebook复制到公共目录来解决权限问题，应由管理员按项目成员配置最小访问权限。

nbconvert执行环境中常见的路径问题

nbconvert的工作目录可能与Notebook所在目录、调用命令所在目录不同。执行前打印Path.cwd()、Notebook绝对路径、输入绝对路径和RUN_DIR，并验证这些位置属于本轮预期项目。读取配置时尽量使用相对项目根目录的路径或显式传入参数，不依赖登录Shell的当前目录。若Notebook调用R、samtools或其他外部程序，确认该进程的PATH、环境变量和工作目录与交互式会话一致。

保留环境锁定与数据访问说明

执行后的Notebook之外，保存依赖锁定文件、系统工具版本、数据目录挂载说明和必要环境变量名。环境变量的秘密值不要写进Notebook或日志；在平台的安全配置中提供，并记录变量是否存在即可。若Notebook需要许可证、参考数据库或外部服务，提前确认目标服务器具备合法访问条件。迁移复现的目标应是团队按说明重建运行环境，而不是把某位研究者的个人会话完整复制到服务器。

建立Notebook迁移前后的验收清单

迁移前记录Notebook版本、输入数据校验、执行顺序依赖、预期文件和关键摘要；迁移后记录服务器路径、内核解释器、环境锁定、完整执行日志、新输出目录和验收人。清单中为每个分析阶段写一个可判定条件，如样本数量一致、矩阵维度符合预期、图表时间戳为本轮、文件可重新读入。不能判定的项目列为未验收，而不是用“Notebook运行成功”概括整条流程。

Notebook输出中容易泄露的信息

执行后的Notebook可能保存表格、图、文件路径、完整命令输出和访问令牌。对外分享前检查输出单元、环境变量打印、隐藏元数据和绝对路径；清理敏感输出时另存分享副本，不破坏内部可复核的受控原件。涉及患者或受限测序数据，应按机构审批的数据处理流程安排服务器访问与结果存储。可复现不要求公开原始数据，公开合成例子和代码即可说明方法。

计算任务长时间运行时的状态设计

长Notebook若需要数小时以上运行，应在关键阶段写入可验证的中间结果和完成标记，并记录输入散列与参数。重新启动时先验证检查点身份和完整性再继续，不能仅凭文件存在就跳过阶段。对不支持安全续跑的单元，保留独立脚本和日志，在新目录重跑。将检查点写到持久项目盘并制定保留周期，避免节点重启或临时盘清理导致结果丢失。

Notebook复现还要考虑外部服务和随机性

有些单元访问数据库、网络API、共享参考库或随机数生成器。记录服务端/参考数据版本、访问时间和随机种子；若外部服务会更新，尽量保存符合许可要求的输入响应或使用本地固定版本。不同硬件和线程下随机算法可能产生轻微差异，预先定义比较的统计摘要和容差，不因图形像素不同判失败，也不忽略样本顺序或大幅数值变化。

迁移验收记录的责任分工

Notebook作者负责说明科学输入、参数及预期检查，运行环境维护者负责解释内核、依赖、挂载和配额，执行者负责记录本次命令、日志与输出。将未解决的问题标成待验收，不要求导师在服务器终端里追着排查。清晰分工可让问题返回具体责任点，减少重复运行和无效资源申请。

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

nbconvert：Notebook执行与遇错行为 · 来源资料

Python：虚拟环境迁移需重建 · 来源资料

Jupyter Server：内核与服务架构 · 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。