从代表性raw文件试跑规划MaxQuant批次与磁盘

完整操作教程 · 参考固定 · 往返恢复验收

MaxQuant处理很多RAW文件前，先核对发行版、系统和原始数据类型，再用代表批次测内存、时间与工作盘。本篇固定MaxQuant2.9.0.0的Linux命令行DDA入口，从同版本GUI确认过的mqpar.x

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。

MaxQuant处理很多RAW文件前，先核对发行版、系统和原始数据类型，再用代表批次测内存、时间与工作盘。本篇固定MaxQuant2.9.0.0的Linux命令行DDA入口，从同版本GUI确认过的mqpar.xml开始，避免仅复制参数文件后继续使用旧服务器路径。

先验证实际平台和参数路径

当前官方Linux安装说明采用.NET8，并建议相应SDK至少8.0.302；不能沿用旧版Mono说明解释所有新版本。保存dotnet --info及MaxQuant帮助，完成该版本与RAW类型的小例。能启动DLL并不证明厂商格式读取、全部依赖或分析方法都已经可用，Ubuntu等支持范围也需按发行说明核对。

记录项	本例准备	验收
程序	固定MaxQuant DLL与整套发行包	与生成mqpar的版本相符
RAW	单个代表批次、明确实验/分组/组分	路径和文件类型可读取
检索	FASTA、酶切、修饰和FDR方案	指纹及参数保持一致
输出	已确认combined/txt位置	summary、evidence与proteinGroups
资源	numThreads与实际阶段峰值	不是核数越多越快

先检查XML中的原始文件清单

本例检查mqpar的filePaths结构，输出实际RAW文件数和总字节。它不修改实验设计，也不猜测另一个版本的XML字段；若模板结构不同，回同版本GUI与官方说明确认。FASTA、预测库和输出目录同样应逐项核对，不能只检查RAW路径。

```
from pathlib import Path
import xml.etree.ElementTree as ET
root=ET.parse("mqpar.xml").getroot()
values=[x.text for x in root.findall("./filePaths/string")]
assert values and all(values)
paths=[Path(x) for x in values]
assert all(p.is_absolute() and p.is_file() and p.stat().st_size>0 for p in paths)
assert len({p.resolve() for p in paths})==len(paths)
threads=root.findtext("numThreads")
assert threads and int(threads)>0
print("RAW files",len(paths),"bytes",sum(p.stat().st_size for p in
paths),"numThreads",threads)
```

检查任务图，再运行代表批次

在/project/mq-pilot准备本次独立输入与mqpar，并在配置中确认输出为示例RESULTS位置。MQ_DLL填写已验收的绝对路径。dryrun显示作业步骤，不会证明RAW已读通或计算正确；正式运行结束仍需检查各阶段和实际输出，不以最后一行日志判断整批成功。

```
set -euo pipefail
: "${MQ_DLL:?■■■■MaxQuant/bin/MaxQuantCmd.dll■■■■}"
[[ "$MQ_DLL" = /* ]] || exit 2
test -s "$MQ_DLL"
cd /project/mq-pilot
test -s mqpar.xml
```

```
mkdir run-evidence-001
dotnet --info > run-evidence-001/dotnet-info.txt
dotnet "$MQ_DLL" --help > run-evidence-001/help.txt 2>&1
sha256sum mqpar.xml "$MQ_DLL" > run-evidence-001/inputs.sha256
dotnet "$MQ_DLL" mqpar.xml --dryrun > run-evidence-001/jobs.txt 2>&1
/usr/bin/time -v -o run-evidence-001/resources.txt \
  dotnet "$MQ_DLL" mqpar.xml > run-evidence-001/run.log 2>&1
RESULTS=/project/mq-pilot/combined/txt
test -s "$RESULTS/summary.txt"
test -s "$RESULTS/evidence.txt"
test -s "$RESULTS/proteinGroups.txt"
```

工作盘按共存文件演算

示例预算表需要用本项目试跑值替换：RAW六十GiB、索引/辅助文件二十五、combined四十、运行临时三十、保留失败及交接副本二十，共一百七十五GiB。它只展示加总口径，不是实测或通用倍数；运行高峰与终态目录大小分别测，配额及实际挂载点也应确认。

核对summary中的文件与实验列表、鉴定和定量统计，再看evidence/proteinGroups中的关键字段、反向/污染标识和缺失模式。表只有头部与有效鉴定不同，较多鉴定也不能单独证明质量。跨RAW匹配、LFQ、分组和FDR都按已批准方案解释，不为提高数字临时改规则。

先定位阶段，再决定扩大并发

记录检索、匹配、定量与写出阶段的时间和内存，numThreads在mqpar中设置，不能任意添加未支持的命令行threads参数。更多并行线程可能叠加RAM及磁盘压力；先测完整代表批次，再判断CPU、内存或I/O瓶颈，官方硬件指引不能代替本项目峰值。

为测资源选择代表RAW子集时，要明确它不是完整队列的最终科学结果。任意拆开全批再拼proteinGroups，可能改变跨文件匹配、LFQ和统计范围；需要拆批时先论证分析设计。保留mqpar、FASTA、RAW映射、版本与全部日志，失败恢复按该版本作业步骤核对，不从另一轮参数接着写旧结果。

保存实际RAW读取错误及缺失文件清单，不能仅凭XML格式正确就判定路径和数据有效。

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

Cox Lab当前Linux/.NET与命令行说明 · 来源资料

MaxQuant官方2.9.0.0发行入口 · 来源资料

MaxQuant实验与分组字段说明 · 来源资料

MaxQuant官方Linux CLI说明 · 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。