RepeatMasker批量软屏蔽：固定重复库并验收FASTA

逐组装固定RepeatMasker、RMBlast和重复库，运行xsmall软屏蔽，并核对输入输出序列ID、长度和大写序列。

多组装运行RepeatMasker时，先固定重复库、搜索引擎和屏蔽策略，再比较结果。不同来源的Dfam、自建库或物种库不能只因名称相似就混用。本文按RMBlast和自定义FASTA库示范软屏蔽；输入已经带有的小写序列也会影响屏蔽比例，需先记录。

设置	本例做法	注意
输入	每个组装独立FASTA	检查唯一序列ID
重复库	固定库文件及指纹	记录来源与适用范围
引擎	RMBlast	保持版本和敏感度一致
屏蔽	-xsmall小写软屏蔽	不同于以N替换的硬屏蔽

先用一个组装跑通

GENOME和LIB填入真实路径，JOBS是RepeatMasker并行批次数。官方帮助说明并行对多序列或较长序列更有作用；很短的小组装未必能按JOBS线性提速，需以任务实际CPU利用率和耗时判断。RMBlast下每批次会启动多个计算线程；脚本中的CPU额度检查只是防止配置超出预算的门槛，不是操作系统级硬限额。先从1个任务开始，再按获配CPU和RAM实测扩展。每次运行使用新OUT目录。运行前抽查FASTA序列ID是否唯一、序列是否为空，并确认自建库来源、适用物种和分类说明可追溯；重复库更新或去冗余方法变化都应算新的分析批次。

```
set -euo pipefail
: "${GENOME:?■■■■■FASTA}"
: "${LIB:?■■■■■■■■FASTA}"
: "${JOBS:?■■RMBlast■■■■■}"
: "${CPU_ALLOWANCE:?■■■■■■■■CPU■■}"
: "${OUT:?■■■■■■■■}"
[[ "$JOBS" =~ ^[1-9][0-9]*$ ]] || exit 2
[[ "$CPU_ALLOWANCE" =~ ^[1-9][0-9]*$ ]] || exit 2
test "$JOBS" -le "$((CPU_ALLOWANCE/4))" || exit 2
test -s "$GENOME" && test -s "$LIB" || exit 2
test ! -e "$OUT" && test ! -L "$OUT" || exit 2
mkdir "$OUT"
cp "$GENOME" "$OUT/input.fa"
mkdir "$OUT/results"
RepeatMasker -v > "$OUT/version.txt" 2>&1
sha256sum "$GENOME" "$LIB" > "$OUT/inputs.sha256"
/usr/bin/time -v -o "$OUT/resources.txt" \
  RepeatMasker -engine rmblast -pa "$JOBS" -lib "$LIB" -xsmall \
  -dir "$OUT/results" "$OUT/input.fa" > "$OUT/run.log" 2>&1
for suffix in out tbl masked; do
  test -s "$OUT/results/input.fa.$suffix" || exit 1
done
python check_softmask.py "$OUT/input.fa" "$OUT/results/input.fa.masked" \
  > "$OUT/sequence-check.txt"
```

验收序列没被意外改写

脚本要求产生.out、.tbl和.masked文件，再比较输入与输出FASTA：序列ID、长度和大写后的序列必须一致。RepeatMasker的-xsmall把识别区域改成小写，因此比较大写序列可以验证软屏蔽位置变化，同时发现截断、丢失或非屏蔽内容变化。此检查不证明重复注释分类正确。out文件适合定位单条匹配及坐标，tbl用于查看分类汇总；同一基因组位置可能有重叠命中，因此命中长度之和不一定等于实际被软屏蔽的唯一碱基数。

```
import hashlib, sys
def fingerprints(path):
    result = {}
    name = None
    length = 0
    digest = None
    def finish():
        if name is not None:
            assert name not in result and length > 0
            result[name] = (length, digest.hexdigest())
    with open(path, "rb") as handle:
        for line in handle:
            line = line.strip()
            if not line:
                continue
            if line.startswith(b">"):
                finish()
                name = line[1:].split()[0].decode("utf-8")
                length = 0
                digest = hashlib.sha256()
            else:
                assert name is not None
                length += len(line)
                digest.update(line.upper())
    finish()
    assert result
    return result
before = fingerprints(sys.argv[1])
after = fingerprints(sys.argv[2])
assert before == after, "sequence IDs, lengths or uppercase sequence changed"
print("SOFTMASK_SEQUENCE_FINGERPRINTS_OK", len(before))
```

检查注释、比例和批次资源

查看.out中的重复类别、坐标和分数，结合.tbl分类汇总与.masked序列检查代表区域。软屏蔽比例应以唯一组装碱基为分母，并考虑输入已有小写区域；小写增多不自动代表识别更准确。若要报告新增屏蔽比例，先记录输入中原有小写状态，再从输出.masked序列统计小写碱基，不能把重叠命中长度直接相加。不同库的分类范围和不同引擎敏感度会改变out/tbl结果；跨组装比较需统一组装口径、库、参数和序列预处理，并把例外单独说明。若要输出GFF或供下游工具使用的格式，另行核对坐标、序列ID和转换版本。

每个组装独立目录，归档输入和库的校验值、RepeatMasker/RMBlast版本、完整命令、.out、.tbl、.masked、日志、运行时间和峰值RSS。多个任务并行会叠加索引、内存、CPU与磁盘I/O；先跑单样本和小并发，不能把-pa当作整机核数。失败时保留日志和输出状态，用新目录重跑。换库、引擎或敏感度选项应建立新批次，避免把不同方法的屏蔽比例直接比较。保留参数便于团队后续复核使用。

参考资料

RepeatMasker官方帮助：-pa、-engine、-lib和-xsmall 官方文档
RepeatMasker官方网页帮助：软屏蔽说明 官方文档