```
#!/usr/bin/env bash
# polap-bash-oatk-ont-sidekicks-stage0.sh
# Stage-0 prefilters ONLY for ONT organelle assembly:
#   trim (porechop_abi), scrub (yacrd), lenfilt (filtlong/seqkit),
#   HMM mt-bait, plastid drop, duplex-only, rare-mask (KMC), correction (Canu/Ratatosk).
#
# Outputs:
#   OUT/pre/reads.pre.fq   (or .fa/.fq.gz depending on tools used)
#   OUT/pre/stage0.report.tsv
#
# Notes:
#   - Does NOT do HPC or assembly. Feed reads.pre.fq into Stage-1/2 wrapper.

set -euo pipefail

: "${POLAP_LOG_LEVEL:=1}" # 0=quiet,1=info,2=verbose
log() {
  local l=$1
  shift
  [[ $POLAP_LOG_LEVEL -ge $l ]] && echo "$@" >&2
}
die() {
  echo "[ERR]" "$@" >&2
  exit 1
}
need() { command -v "$1" >/dev/null 2>&1 || die "missing dependency: $1"; }

# ---------------- CLI defaults ----------------
READS=""
OUT=""
THREADS=32
OATKDB=""
CLADE="magnoliopsida"
HMMBIN="hmmannot"
NHMMSCAN="nhmmscan"

DO_TRIM=0
DO_SCRUB=0
DO_LENFILT=0
LEN_MIN=5000
KEEP_PCT=70
DO_MT_BAIT=0
DO_DROP_PLASTID=0
DO_DUPLEX_ONLY=0
DUPLEX_TAG="duplex"
DO_RARE_MASK=0
RARE_K=31
RARE_MAX=1
DO_CORRECT=0
SHORT_READS=""

```

```
usage() {
  cat <<EOF
Usage:
  $(basename "$0") --reads READS.fq.gz --out OUTDIR --oatkdb /path/OatkDB [options]

Required:
  --reads FILE          ONT reads (fa/fq[.gz])
  --out   DIR           output directory
  --oatkdb DIR          OatkDB root (must contain v20230921)

General:
  --threads INT         (default ${THREADS})
  --clade NAME          fam prefix under v20230921 (default ${CLADE})
  --hmm-bin PATH        hmmannot/hmm_annotation (default ${HMMBIN})
  --nhmmscan PATH       nhmmscan path (default ${NHMMSCAN})
  -v|--verbose          verbose
  -q|--quiet            quiet

Stage-0 prefilters (all optional):
  --trim                porechop_abi
  --scrub               minimap2 ava-ont + yacrd
  --lenfilt             filtlong (preferred) or seqkit
  --len-min INT         min length (default ${LEN_MIN})
  --keep-pct INT        filtlong keep percent (default ${KEEP_PCT})
  --mt-bait             keep reads hitting mito fam (HMM)
  --drop-plastid        drop reads hitting plastid fam (HMM)
  --duplex-only         keep reads with tag (default '${DUPLEX_TAG}')
  --duplex-tag STR      header tag (default '${DUPLEX_TAG}')
  --rare-mask           KMC rare-kmer masking (heavy placeholder)
  --rare-k INT          (default ${RARE_K})
  --rare-max INT        (default ${RARE_MAX})
  --correct-canu        Canu correction-only
  --correct-ratatosk    Ratatosk (needs --short-reads)
  --short-reads FILE    short reads for Ratatosk
EOF
}

# --------------- parse args -------------------
while [[ $# -gt 0 ]]; do
  case "$1" in
  --reads)
    READS="$2"
    shift 2
    ;;
  --out)
    OUT="$2"
    shift 2
    ;;
  --oatkdb)
    OATKDB="$2"
    shift 2
```

```
    ;;
  --threads)
    THREADS="$2"
    shift 2
    ;;
  --clade)
    CLADE="$2"
    shift 2
    ;;
  --hmm-bin)
    HMMBIN="$2"
    shift 2
    ;;
  --nhmmscan)
    NHMMSCAN="$2"
    shift 2
    ;;
  -v | --verbose)
    POLAP_LOG_LEVEL=2
    shift
    ;;
  -q | --quiet)
    POLAP_LOG_LEVEL=0
    shift
    ;;
  --trim)
    DO_TRIM=1
    shift
    ;;
  --scrub)
    DO_SCRUB=1
    shift
    ;;
  --lenfilt)
    DO_LENFILT=1
    shift
    ;;
  --len-min)
    LEN_MIN="$2"
    shift 2
    ;;
  --keep-pct)
    KEEP_PCT="$2"
    shift 2
    ;;
  --mt-bait)
    DO_MT_BAIT=1
    shift
    ;;
  --drop-plastid)
    DO_DROP_PLASTID=1
```

```
    shift
    ;;
  --duplex-only)
    DO_DUPLEX_ONLY=1
    shift
    ;;
  --duplex-tag)
    DUPLEX_TAG="$2"
    shift 2
    ;;
  --rare-mask)
    DO_RARE_MASK=1
    shift
    ;;
  --rare-k)
    RARE_K="$2"
    shift 2
    ;;
  --rare-max)
    RARE_MAX="$2"
    shift 2
    ;;
  --correct-canu)
    DO_CORRECT=1
    shift
    ;;
  --correct-ratatosk)
    DO_CORRECT=2
    shift
    ;;
  --short-reads)
    SHORT_READS="$2"
    shift 2
    ;;
  -h | --help)
    usage
    exit 0
    ;;
  *)
    echo "[ERR] unknown arg: $1" >&2
    usage
    exit 1
    ;;
  esac
done
[[ -z "$READS" || -z "$OUT" || -z "$OATKDB" ]] && {
  usage
  exit 1
}

# ---------------- prep & deps ------------------
```

```
READS="$(readlink −f "$READS")"
OUT="$(readlink −f "$OUT")"
OATKDB="$(readlink −f "$OATKDB")"
mkdir −p "$OUT/pre"
cd "$OUT/pre"

need seqkit
[[ $DO_TRIM −eq 1 ]] && need porechop_abi
[[ $DO_SCRUB −eq 1 ]] && {
  need minimap2
  need yacrd
}
[[ $DO_LENFILT −eq 1 ]] && { command −v filtlong >/dev/null 2>&1 || need seqkit; }
FAM_M="$OATKDB/v20230921/${CLADE}_mito.fam"
FAM_P="$OATKDB/v20230921/${CLADE}_pltd.fam"
[[ $DO_MT_BAIT −eq 0 || −s "$FAM_M" ]] || die "mito fam not found: $FAM_M"
[[ $DO_DROP_PLASTID −eq 0 || −s "$FAM_P" ]] || die "plastid fam not found: $FAM_P"
[[ $DO_MT_BAIT −eq 0 && $DO_DROP_PLASTID −eq 0 ]] || {
  command −v "$HMMBIN" >/dev/null 2>&1 || die "missing $HMMBIN"
  need "$NHMMSCAN"
}
[[ $DO_RARE_MASK −eq 1 ]] && {
  need kmc
  command −v kmc_tools >/dev/null 2>&1 || need kmc_tools
}
[[ $DO_CORRECT −eq 1 ]] && need canu
[[ $DO_CORRECT −eq 2 ]] && {
  need ratatosk || need Ratatosk || true
  [[ −n "$SHORT_READS" ]] || die "−−short−reads required"
}

# ---------------  Stage−0 run ------------------
CUR="$READS"

# trim
if [[ $DO_TRIM −eq 1 ]]; then
  log 1 "[trim] porechop_abi"
  porechop_abi −i "$CUR" −o reads.trim.fq −t "$THREADS"
  CUR="$PWD/reads.trim.fq"
fi
# scrub
if [[ $DO_SCRUB −eq 1 ]]; then
  log 1 "[scrub] minimap2 ava−ont + yacrd"
  minimap2 −x ava−ont −g 500 −t "$THREADS" "$CUR" "$CUR" >overlaps.paf
  yacrd −i overlaps.paf −o yacrd.tsv −c 4 −n −p 0.4
  awk '$3=="chimera"{next} $3=="unchanged"{print $1}' yacrd.tsv >keep.ids
  seqkit grep −f keep.ids "$CUR" −o reads.scrub.fq
  CUR="$PWD/reads.scrub.fq"
fi
# lenfilt
if [[ $DO_LENFILT −eq 1 ]]; then
```

```
  if command -v filtlong >/dev/null 2>&1; then
    log 1 "[lenfilt] filtlong −−min_length $LEN_MIN −−keep_percent $KEEP_PCT"
    filtlong --min_length "$LEN_MIN" --keep_percent "$KEEP_PCT" "$CUR" >reads.long.fq
    CUR="$PWD/reads.long.fq"
  else
    log 1 "[lenfilt] seqkit seq −m $LEN_MIN"
    seqkit seq -m "$LEN_MIN" "$CUR" -o reads.len.fq
    CUR="$PWD/reads.len.fq"
  fi
fi
# HMM mt-bait
if [[ $DO_MT_BAIT -eq 1 ]]; then
  log 1 "[mt−bait pre] $HMMBIN vs mito fam"
  "$HMMBIN" -t "$THREADS" --nhmmscan "$NHMMSCAN" -o reads.mito.txt "$FAM_M" "$CUR"
  awk -v FS='[[:space:]]+' '$1!~/^#/{print $3}' reads.mito.txt | sort -u >mt.ids
  seqkit grep -f mt.ids "$CUR" -o reads.mt.fq
  CUR="$PWD/reads.mt.fq"
fi
# plastid drop
if [[ $DO_DROP_PLASTID -eq 1 ]]; then
  log 1 "[plastid pre] drop plastid reads"
  "$HMMBIN" -t "$THREADS" --nhmmscan "$NHMMSCAN" -o reads.pt.txt "$FAM_P" "$CUR"
  awk -v FS='[[:space:]]+' '$1!~/^#/{print $3}' reads.pt.txt | sort -u >pt.ids
  seqkit grep -v -f pt.ids "$CUR" -o reads.mt.noPT.fq
  CUR="$PWD/reads.mt.noPT.fq"
fi
# duplex-only
if [[ $DO_DUPLEX_ONLY -eq 1 ]]; then
  log 1 "[duplex] keep header tag: $DUPLEX_TAG"
  awk -v tag="$DUPLEX_TAG" 'BEGIN{OFS="\n"}
   NR%4==1{keep=index($0,tag)>0;h=$0}
   NR%4==2{s=$0}
   NR%4==3{p=$0}
   NR%4==0{q=$0; if(keep){print h,s,p,q}}' "$CUR" >reads.duplex.fq
  CUR="$PWD/reads.duplex.fq"
fi
# rare-mask (placeholder)
if [[ $DO_RARE_MASK -eq 1 ]]; then
  log 1 "[rare−k] KMC k=$RARE_K, <=$RARE_MAX"
  mkdir -p kmc_tmp
  kmc -k"$RARE_K" -t"$THREADS" -ci1 -cs100000000 "$CUR" kmc.db kmc_tmp/
  kmc_tools transform kmc.db dump -ci1 -cx"$RARE_MAX" rare_k.txt
  cp "$CUR" reads.mask.fq
  CUR="$PWD/reads.mask.fq"
fi
# correction
if [[ $DO_CORRECT -eq 1 ]]; then
  log 1 "[correct] Canu correction−only"
  mkdir -p corr
  canu -correct -p ontcorr -d corr genomeSize=500k -nanopore-raw "$CUR" 2>corr/canu.log || log 1 "[warn] canu failed?"
  if ls corr/ontcorr.correctedReads.fasta.gz >/dev/null 2>&1; then CUR="$PWD/corr/ontcorr.correctedReads.fasta.gz"; fi
```

```
307 elif [[ $DO_CORRECT -eq 2 ]]; then
308   log 1 "[correct] Ratatosk"
309   need ratatosk || need Ratatosk || true
310   [[ -n "$SHORT_READS" ]] || die "--short-reads required for Ratatosk"
311   ratatosk -s "$SHORT_READS" -l "$CUR" -o reads.rtk.fq -t "$THREADS" || log 1 "[warn] ratatosk failed?"
312   CUR="$PWD/reads.rtk.fq"
313 fi
314
315 # emit final
316 FINAL="$PWD/reads.pre.fq"
317 cp -f "$CUR" "$FINAL"
318
319 # report
320 {
321   echo -e "key\tvalue"
322   echo -e "reads_in\t$READS"
323   echo -e "out\t$OUT"
324   echo -e "reads_pre\t$FINAL"
325   echo -e "trim\t$([[ $DO_TRIM -eq 1 ]] && echo ON || echo OFF)"
326   echo -e "scrub\t$([[ $DO_SCRUB -eq 1 ]] && echo ON || echo OFF)"
327   echo -e "len_min\t$([[ $DO_LENFILT -eq 1 ]] && echo $LEN_MIN || echo .)"
328   echo -e "keep_pct\t$([[ $DO_LENFILT -eq 1 && $(
329     command -v filtlong >/dev/null
330     echo $?
331   ) -eq 0 ]] && echo $KEEP_PCT || echo .)"
332   echo -e "mt_bait_prefilter\t$([[ $DO_MT_BAIT -eq 1 ]] && echo ON || echo OFF)"
333   echo -e "drop_plastid_prefilter\t$([[ $DO_DROP_PLASTID -eq 1 ]] && echo ON || echo OFF)"
334   echo -e "duplex_only\t$([[ $DO_DUPLEX_ONLY -eq 1 ]] && echo $DUPLEX_TAG || echo OFF)"
335   echo -e "rare_mask\t$([[ $DO_RARE_MASK -eq 1 ]] && echo K=$RARE_K,max=$RARE_MAX || echo OFF)"
336   echo -e "corrector\t$([[ $DO_CORRECT -eq 1 ]] && echo CANU || ([[ $DO_CORRECT -eq 2 ]] && echo RATATOSK || echo OFF))"
337 } >"$OUT/pre/stage0.report.tsv"
338
339 log 1 "[Stage-0 done] Preprocessed reads: $FINAL"
```