```
#!/usr/bin/env Rscript

################################################################################
# This file is part of polap.
#
# polap is free software: you can redistribute it and/or modify it under the
# terms of the GNU General Public License as published by the Free Software
# Foundation, either version 3 of the License, or (at your option) any later
# version.
#
# polap is distributed in the hope that it will be useful, but WITHOUT ANY
# WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR
# A PARTICULAR PURPOSE. See the GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along with
# polap. If not, see <https://www.gnu.org/licenses/>.
################################################################################

# polaplib/run-polap-r-pairs.R
# Check: 2025-06-17

################################################################################
# NOTE: this is the first of the two read-selection scripts.
# This script selects reads using some of minimap2 output files.
# We have multiple scripts that selects reads.
# This is the 2nd one.
#
# 1. run-polap-pairs.R -> the first version used in v0.2.6
# 2. run-polap-r-pairs.R -> used in test-reads in oga
# 3. run-polap-r-bridge.R -> used in test-reads in oga
# 4. run-polap-r-select-reads-polap.R -> the 2nd version not used
# 5. run-polap-r-select-reads-ptgaul.R -> a slim version of the 2nd used in dga
# 6. run-polap-r-directional.R -> used by an older version of dga
#    or _run_polap_original-directional-reads
#
# Subcommand test-reads uses this script.
#
# Used by:
# function _run_polap_test-reads { # selects reads mapped on a genome assembly
#
# See Also:
# run-polap-pairs.R
# run-polap-r-pairs.R
# run-polap-r-bridge.R
# run-polap-r-select-reads-polap.R
# run-polap-r-select-reads-ptgaul.R
# run-polap-r-directional.R
# run-polap-function-oga.sh
# run-polap-function-dga.sh
#
# Check: 2025-06-17
```

```
################################################################################

################################################################################
# This script selects reads using minimap2 alignments of the reads on the seed
# contigs. It tries to use two main different approaches; one is due to ptGAUL,
# another is devised by Polap. Polap’s read selection is more stringent in a
# way that it selects reads that are mapped on two seed contigs and those that
# are mapped completely within a contig.
# This has some benefit or better assembly for a smaller dataset; i.e., 10x.
# If reads are enough, ptGAUL method seems to work great.
#
# Check: 2025-06-16
################################################################################

# name: select long reads
#
# synopsis:
#   run-polap-pairs.R mt.contig.name-1 contig.tab ${MTSEEDSDIR} $SINGLE_MIN $SINGLE_MIN
#
# requirement: executes Flye
# flye --nano-raw "$LR3K" --out-dir "${_arg_outdir}" \
#   --threads "${_arg_threads}" \
#   --stop-after contigger \
#   --asm-coverage 30 \
#   --genome-size "$EXPECTED_GENOME_SIZE"
#
# input:
#   1. mt.contig.name-1 - contig or edge numbers
#   2. contig.tab - minimap2 output modified
#   3. seeds directory for output
#   4. pair minimum length V11: 3000 for MT, 1000 for PT
#   5. bridge minimum length V7: depends: 3000 or 5000
#   6. single minimum length V11: 3000 for MT, 0 or 1000 for PT
#
# output:
#   single.names and <contig1-contig2.name> files in the output directory
#
# MTSEEDSDIR="${_arg_outdir}"/60-mt-${STEP4}/o${MR}/seeds
#   run-polap-pairs.R mt.contig.name-1 contig.tab ${MTSEEDSDIR} $PAIR_MIN $BRIDGE_MIN $SINGLE_MIN
#   run-polap-pairs.R mt.contig.name-1 contig.tab ${MTSEEDSDIR} $PAIR_MIN $BRIDGE_MIN

suppressPackageStartupMessages(library("optparse"))
suppressPackageStartupMessages(library("dplyr"))
suppressPackageStartupMessages(library("readr"))
suppressPackageStartupMessages(library("purrr"))
suppressPackageStartupMessages(library("tidyr"))
suppressPackageStartupMessages(library("ggplot2"))

debug ← Sys.getenv("_POLAP_DEBUG", unset = "0")

parser ← OptionParser()
parser ← add_option(parser, c("−t", "−−table"),
  action = "store",
  help = "minimap2 PAF",
  metavar = "<FILE>"
)
parser ← add_option(parser, c("−m", "−−mtcontigname"),
  action = "store",
  help = "mt.contig.name−1",
  metavar = "<FILE>"
)
parser ← add_option(parser, c("−o", "−−out"),
  action = "store",
  help = "output directory"
)
parser ← add_option(parser, c("−r", "−−pair−min"),
  type = "integer",
  default = 3000,
  help = "Minimum length of pair−mapping alignment",
  metavar = "number"
)
parser ← add_option(parser, c("−w", "−−single−min"),
  type = "integer",
  default = 3000,
  help = "Minimum length of single−mapping alignment",
  metavar = "number"
)
parser ← add_option(parser, c("−x", "−−bridge−min"),
  type = "integer",
  default = 3000,
  help = "Minimum length of bridging reads",
  metavar = "number"
)
parser ← add_option(parser, c("−−intra−base−ratio"),
  type = "double",
  default = 0.7,
  help = "Minimum of V10/V11",
  metavar = "number"
)
parser ← add_option(parser, c("−−intra−read−ratio"),
  type = "double",
  default = 0.7,
  help = "Minimum of (V4−V3)/V2",
  metavar = "number"
)
parser ← add_option(parser, c("−−inter−base−ratio"),
  type = "double",
  default = 0.7,
  help = "Minimum of V10/V11",
  metavar = "number"
)
parser ← add_option(parser, c("−−create−ptgaul"),
  action = "store_true",
  default = FALSE, help = "will create ptgaul.names"
)
parser ← add_option(parser, c("−−create−single"),
  action = "store_true",
  default = FALSE, help = "will create single.names"
)
parser ← add_option(parser, c("−−create−pair"),
  action = "store_true",
  default = FALSE, help = "will create pair.names"
)
parser ← add_option(parser, c("−−create−combined"),
  action = "store_true",
  default = FALSE, help = "will create combined.names"
)
parser ← add_option(parser, c("−−all"),
  action = "store_true",
  default = FALSE, help = "will create all the 4 names files."
)
parser ← add_option(parser, c("−−outptgaul"),
  action = "store",
  default = "ptgaul.names",
  help = "ptgaul output file"
)
parser ← add_option(parser, c("−−outsingle"),
  action = "store",
  default = "single.names",
  help = "single output file"
)
parser ← add_option(parser, c("−−outpair"),
  action = "store",
  default = "pair.names",
  help = "pair output file"
)
parser ← add_option(parser, c("−−outcombined"),
  action = "store",
  default = "combined.names",
  help = "combined output file"
)

args1 ← parse_args(parser)

if (is_null(args1$table)) {
  s ← "bioprojects"
  o ← "PRJNA817235−Canavalia_ensiformis"

  input_dir0 ← file.path(".")
  input1 ← file.path(input_dir0, "contig.tab")
  input2 ← file.path(input_dir0, "mt.contig.name−1")
  output1 ← file.path(input_dir0, "02−reads")

  args1 ← parse_args(parser, args = c(
    "--table", input1,
    "--mtcontigname", input2,
    "-o", output1,
    "-r", 10000,
    "-w", 10000,
    "-x", 10000,
    "--all",
    "--intra-base-ratio", 0.7,
    "--intra-read-ratio", 0.7,
    "--inter-base-ratio", 0.7
  ))
}

# mt.contig.name-x
x1 ← as_tibble(read.table(args1$mtcontigname))

# https://lh3.github.io/minimap2/minimap2.html
# Li 2008 - "OUTPUT FORMAT: Minimap2 outputs mapping positions in the
# Pairwise mApping Format (PAF) by default. PAF is a TAB-delimited text format
# with each line consisting of at least 12 fields as are described in the
# following table:
#
# Col Type  Description
# 1 string  Query sequence name
# 2 int Query sequence length
# 3 int Query start coordinate (0-based)
# 4 int Query end coordinate (0-based)
# 5 char  âM-^@M-^X+âM-^@M-^Y if query/target on the same strand; âM-^@M-^X-âM-^@M-^Y if opposite
# 6 string  Target sequence name
# 7 int Target sequence length
# 8 int Target start coordinate on the original strand
# 9 int Target end coordinate on the original strand
# 10  int Number of matching bases in the mapping
# 11  int Number bases, including gaps, in the mapping
# 12  int Mapping quality (0-255 with 255 for missing)
#
# Read in the data with assigned column names
data ← read_tsv(args1$table,
  show_col_types = FALSE,
  col_names = c(
    "rname",
    "rlen",
    "rstart",
    "rend",
    "strand",
    "cname",
    "clen",
    "cstart",
    "cend",
    "match",
    "base"
  )
)
mtdir ← args1$out
pair_min ← as.numeric(args1$`pair-min`)
brigde_min ← as.numeric(args1$`bridge-min`)
single_min ← as.numeric(args1$`single-min`)
a ← as.numeric(args1$`intra-read-ratio`)
b ← as.numeric(args1$`intra-base-ratio`)
c ← as.numeric(args1$`inter-base-ratio`)
ptgaul_option_base ← paste("ptgaul",
  a,
  b,
  single_min,
  c,
  pair_min,
  brigde_min,
  sep = "-"
)
ptgaul_option_base ← paste0(ptgaul_option_base, ".names")

single_option_base ← paste("single",
  a,
  b,
  single_min,
  c,
  pair_min,
  brigde_min,
  sep = "-"
)
single_option_base ← paste0(single_option_base, ".names")

pair_option_base ← paste("pair",
  a,
  b,
  single_min,
  c,
  pair_min,
  brigde_min,
  sep = "-"
)
pair_option_base ← paste0(pair_option_base, ".names")

combined_option_base ← paste("combined",
  a,
  b,
  single_min,
  c,
  pair_min,
  brigde_min,
  sep = "-"
```

```
)
combined_option_base ← paste0(combined_option_base, ".names")

# 0. Check preconditions
# rstart is not greater than rend.
# match is not greater than base.
result ← data %>%
  summarize(all_rows_meet_condition = all(rstart ≤ rend & match ≤ base))

# Check the result
stopifnot(result$all_rows_meet_condition)
stopifnot(pair_min ≥ 0, brigde_min ≥ 0, single_min ≥ 0)

# 1. ptGAUL: we use ptGAUL for the case of single edge reference
if (args1$`create-ptgaul` || args1$all) {
  ptgaul_file ← file.path(mtdir, args1$outptgaul)
  ptgaul_mapped_reads ← data |>
    filter(match / base > args1$`intra-base-ratio`, base > single_min) |>
    select(rname) |>
    distinct(rname)

  ptgaul_mapped_reads |>
    write.table(
      ptgaul_file,
      row.names = FALSE,
      col.names = FALSE,
      quote = FALSE
    )
  ptgaul_option_file ← file.path(mtdir, ptgaul_option_base)
  print(ptgaul_file)
  file.copy(ptgaul_file, ptgaul_option_file)
  # file.copy(ptgaul_file, ptgaul_option_file, showWarnings = FALSE)
}

# 2. intra-contig mapping
if (args1$`create-single` || args1$all) {
  single_file ← file.path(mtdir, args1$outsingle)
  intra_contig_mapped_reads ← data |>
    filter(
      (rend - rstart) / rlen > args1$`intra-read-ratio`,
      match / base > args1$`intra-base-ratio`,
      base > single_min
    ) |>
    select(rname) |>
    distinct(rname)

  intra_contig_mapped_reads |>
    write.table(single_file, row.names = FALSE, col.names = FALSE, quote = FALSE)

  single_option_file ← file.path(mtdir, single_option_base)
  print(single_file)
  file.copy(single_file, single_option_file)
  # file.copy(single_file, single_option_file, showWarnings = FALSE)
}

if (nrow(x1) > 1) {
  y ← t(combn(x1$V1, 2))

  orders ← function(y) {
    stopifnot(length(y) ≡ 2)
    z ← as.numeric(gsub("\\D", "", y))
    bname ← paste0(z[1], "_", z[2])
    oname ← paste0(mtdir, "/", bname, ".name")
    y1 ← data |>
      filter(
        cname ≡ y[1],
        match / base > args1$`inter-base-ratio`,
        base > pair_min,
        rlen > brigde_min
      ) |>
      select(rname) |>
      distinct(rname)
    y2 ← data |>
      filter(
        cname ≡ y[2],
        match / base > args1$`inter-base-ratio`,
        base > pair_min,
        rlen > brigde_min
      ) |>
      select(rname) |>
      distinct(rname)
    intersect(y1, y2) |>
      write.table(oname, row.names = FALSE, col.names = FALSE, quote = FALSE)
  }


  # 3. inter-contig mapping: individual pair
  if (args1$all) {
    apply(y, 1, orders)
  }

  # 4. inter-contig mapping
  #
  # Filter data based on the given criteria

  if (args1$`create-pair` || args1$all) {
    inter_contig_mapped_reads ← data |>
      # Calculate the match/base ratio
      mutate(match_base_ratio = match / base) |>
      # Filter for match/base > 0.7 and base > 3000
      filter(
        match_base_ratio > args1$`inter-base-ratio`,
        base > pair_min,
        rlen > brigde_min
      ) |>
      # Count the number of contigs for each read
      group_by(rname) |>
      # Keep only those reads that are mapped to at least two contigs
      filter(n_distinct(cname) ≥ 2) |>
      # Select unique reads
      distinct(rname) |>
      ungroup()

    pair_file ← file.path(mtdir, args1$outpair)
    inter_contig_mapped_reads |>
      write.table(
        pair_file,
        row.names = FALSE,
        col.names = FALSE,
        quote = FALSE
      )
    pair_option_file ← file.path(mtdir, pair_option_base)
    print(pair_file)
    file.copy(pair_file, pair_option_file)
    # file.copy(pair_file, pair_option_file, showWarnings = FALSE)
  }

  # 5. combined: single + pair
  if (args1$`create-combined` || args1$all) {
    combined_mapped_reads ← bind_rows(
      inter_contig_mapped_reads,
      intra_contig_mapped_reads
    ) |>
      distinct(rname, .keep_all = TRUE)

    combined_file ← file.path(mtdir, args1$outcombined)
    combined_mapped_reads |>
      write.table(
        combined_file,
        row.names = FALSE,
        col.names = FALSE,
        quote = FALSE
      )
    combined_option_file ← file.path(mtdir, combined_option_base)
    print(combined_file)
    file.copy(combined_file, combined_option_file)
    # file.copy(combined_file, combined_option_file, showWarnings = FALSE)
  }
}
```