Deduper

Rachel Richardson

November 1, 2018

Runs quickly (minutes), can be run on the srun of talapas. Run best with bash script, it calls the python script and adjusts values accordingly. Counts duplicates and non-duplicates, does not count reads that don’t match UMI list.

Deduper program instructions:

Program takes single-end reads with optional UMI file. Duplicates can be kept in a file.duplicate.out.sam format if desired. UMI file must be seperated by new line characters. Program should be run as specified by below sample:

./dedup.script filewithreads.sam UMI.txt keep

NOTE: UMI.txt and keep are optional entries; keep indicates that duplicates should be kept.Non-entry or incorrect entries will result in discarded duplicates.

bash script:

```
#!/usr/bin/env bash
#SBATCH --partition=short       ### Partition (like a queue in PBS)
#SBATCH --job-name=RRPS7          ### Job Name
#SBATCH --time=0-20:01:00       ### Wall clock 0ime limit in Days-HH:MM:SS
#SBATCH --nodes=1               ### Number of nodes needed for the job
#SBATCH --ntasks-per-node=28     ### Number of tasks to be launched per Node
#SBATCH --mail-user=rarichardson92@gmail.com
#SBATCH --mail-type=BEGIN,END,FAIL

# Don't forget to load modules in bash, easybuild, prl, python/3.6.0 before running co
de!

ml purge
ml load easybuild prl python/3.6.0

# $1 should be the name of the SAM file to be processed
# $2 should be the UMI file to be processed
# if duplicates are desired for the script,

# ./dedup.script file.sam Umi.txt


# Sorts file, removes headers and saves as file.process.sam

#grep "^@" $1 > headers.sam
#To be included if you need the headers for any reason (optional in samtools, figure t
hat they can be optional here)

echo ""
echo "Deduper program instructions:"
echo ""
echo "Program takes single-end reads with optional UMI file."
echo "Duplicates can be kept in a file.duplicate.out.sam format if desired."
echo "UMI file must be seperated by new line characters."
echo "Program should be run as specified by below sample:"
echo ""
echo ""
echo "./dedup.script filewithreads.sam UMI.txt keep"
echo ""
echo ""
echo "NOTE: UMI.txt and keep are optional entries; keep indicates that duplicates shou
ld be kept."
echo "Non-entry or incorrect entries will result in discarded duplicates."
echo ""
echo ""

grep -v "^@" $1 | sed -E "s/([0-9]+):([ATCG]+)/\1\t\2/" | sort -k2,2d -k4,4 -k5,5n | s
ed -E "s/([0-9]+)\t([ATCG]+)/\1:\2/" > $1.process.sam
```

```
paired="`cut -f 9 "$1" | grep -P [0-9] | sort -rn | uniq | head -n 1`"

if [ $paired -gt 0 ]; then
    echo "Invalid input. File contains paired-end reads, program is only suitable for
single-end reads."
    exit 128
fi

UMI=$2

if [ "$UMI" = "" ]; then UMI="None"; fi

echo "UMI file: "$UMI

rvswin="`cut -f 6 "$1" | grep -o -P [0-9]+N | grep -o -P [0-9]+ | awk 'BEGIN{i=o} $0>i
{i=$0} END{print i+100}'`"

echo "SAM file sorted for processing. (Barcode, Chromosome/linkage group, Position)"
if [ "$3" = "keep" ]; then
    echo "Duplicates will be kept."
fi
echo "Reverse window is set to: "$rvswin
echo ""

if [ "$3" = "keep" ]; then
    time ./Richardson_deduper.py -f $1 -u $UMI -sizef 100 -sizer $rvswin -keep True
else
    time ./Richardson_deduper.py -f $1 -u $UMI -sizef 100 -sizer $rvswin
fi

rm $1.process.sam
```

Python script:

```
#!/usr/bin/env python3
#SBATCH --partition=long       ### Partition (like a queue in PBS)
#SBATCH --job-name=RRPS7          ### Job Name
#SBATCH --time=1-20:01:00       ### Wall clock 0ime limit in Days-HH:MM:SS
#SBATCH --nodes=1               ### Number of nodes needed for the job
#SBATCH --ntasks-per-node=28     ### Number of tasks to be launched per Node
#SBATCH --mail-user=rarichardson92@gmail.com
#SBATCH --mail-type=BEGIN,END,FAIL
# Don't forget to load modules in bash, easybuild, prl, python/3.6.0 before running co
de!
import argparse
import re
def getarguments():
    parser=argparse.ArgumentParser(description = "Removes PCR duplicates from single-e
nd SAM files (UMI/randomer, chrom/linkage group, base position). Requires sorted SAM f
iles, optional file with line seperated UMIs. Outputs file with first occuring read b
y UMI and start position. Designed to be piped in from bash program for appropriate so
rting/size determination.")
    parser.add_argument("-f", "--file", help = "Defines name and path of sam file to u
se in program. Required, must be a string. Designed as input from dedup, which sorts a
nd saves file as file.sorted.sam. Rename file in this format if running without dedup.
script.", required = True, type = str)
    parser.add_argument("-u", "--umi", help = "Defines name and path of UMI file to us
e in program. Optional, must be a string.", required = False, type = str)
    parser.add_argument("-sizef", help = "Defines size of forward window to use in pro
gram. Required, must be an integer. Set at 100 in dedup.script (Accounts for all soft
clipping possible when sorted per UMI).", required = True, type = int)
    parser.add_argument("-sizer", help = "Defines of reverse window to use in progra
m. Required, must be an integer. Determined by file contents in dedupt.script when use
d together.", required = True, type = int)
    parser.add_argument("-keep", help = "Boolean. Determines if duplicates are kept i
n an out file. Default is False.", required = False, default = False, type = bool)
    parser.add_argument("-p", "--paired", help = "Paired end file boolean. Currently n
ot supported by this program.", required = False, default = False, type = bool)
    return parser.parse_args()
def fwdclip(CIGAR, start):
    '''For forward strands, adds soft clipping to start position'''
    if "S" in CIGAR:
        softclip = CIGAR.split("S")
        if str.isdigit(softclip[0]) == True:
            start = start - int(softclip[0])
    return(start)
def rvsclip(CIGAR, start):
    '''For reverse strands, adds all relevant cigar notation to start point'''
    Addlist = re.findall(r'[0-9]*[MNPD]', CIGAR)
    Cliplist = re.findall(r'[0-9]*S$', CIGAR)
    Totaladd=0
    for found in Addlist:
```

```
        Totaladd += int(found[:-1])
    if len(Cliplist) == 1:
        Totaladd += int(Cliplist[0][:-1])
    start = start + Totaladd - 1
    # -1, since original start position takes up a slot
    return(start)
def windowcheck(window, newline):
    '''Checks if line to add to window is a duplicate of anything in the window'''
    dup = False
    if newline in window:
            dup = True
            if newline[0] in dupcount:
                dupcount[newline[0]]+=1
            else:
                dupcount[newline[0]]=1
    return(dup)
def UMIcheck(UMIs, adderUMI):
    '''Determines if UMI of SAM file is in UMI dictionary is applicable'''
    continueflag = False
    if adderUMI in UMIs:
        continueflag = True
    return(continueflag)
args=getarguments()
file=str(args.file)
UMIfile=str(args.umi)
sizef=int(args.sizef)
sizer=int(args.sizer)
keep=bool(args.keep)
p=bool(args.paired)
if p == True:
    raise Exception('Paired-end input not supported.')
UMIs = set()
dupcount={}
goodcount = 0
if UMIfile != "None":
    with open(UMIfile, "r") as UMIf:
        for line in UMIf:
            UMIs.add(line.strip('\n'))
if keep == True:
    bad = open(file[:-4]+"_duplicate.sam", "w")
with open(file+".process.sam", "r") as fh, open(file[:-4]+"_deduped.sam", "w") as goo
d:
    fwdwin = []
    rvswin = []
    #Sets up window listss
    breakflag = False
    #flag for loop break if UMIs are not good at EOF
    for line in fh:
        full=line.strip('\n')
```

```
        #grabs sam record with stripped newline
        adder=full.split('\t')
        #splits full into adder, a list from full by SAM fields
        if UMIfile != "None":
            checkflag = UMIcheck(UMIs, adder[0][-8:])
            #check if new umi is in dictionary
            while checkflag == False:
            #will reset above variables if new UMI isn't valid,
                try:
                    full=next(fh).strip('\n')
                except StopIteration:
                    breakflag = True
                    break
                    #will break loop at EOF, flag breaks out of second loop
                adder=full.split('\t')
                checkflag = UMIcheck(UMIs, adder[0][-8:])
        if breakflag == True:
            break
        #if flagged true, EOF. Discontinue loop, current line is invalid UMI.
        start=int(adder[3])
        #sets start value to where the pos field is located
        if int(adder[1]) & 16 != 16:
        #Adds soft clipping for forward strands
            start=fwdclip(adder[5], start)
            adder=[ adder[0][-8:], adder[2], start]
            duplicate = windowcheck(fwdwin, adder)
            if duplicate == False and len(fwdwin) < sizef: #replace 3 with desired win
dow size
                good.write(full+"\n")
                fwdwin = fwdwin + [adder]
                goodcount+=1
            elif duplicate == False:
                good.write(full+"\n")
                fwdwin = fwdwin[1:] + [adder]
                goodcount+=1
            elif keep == True:
                bad.write(full+"\n")
        else:
            start = rvsclip(adder[5], start)
            adder=[ adder[0][-8:], adder[2], start ]
            duplicate = windowcheck(rvswin, adder)
            if duplicate == False and len(rvswin) < sizer: #replace 3 with desired win
dow size
                good.write(full+"\n")
                rvswin = rvswin + [adder]
                goodcount+=1
            elif duplicate == False:
                good.write(full+"\n")
                rvswin = rvswin[1:] + [adder]
```

```
                goodcount+=1
            elif keep == True:
                bad.write(full+"\n")
if keep == True:
    bad.close()
print()
print("Number of duplicates per UMI/randomer:")
for key, value in dupcount.items():
    print(key, value, sep = '\t')
print()
print("Total duplicates: "+str(sum(dupcount.values())))
print("Remaining reads: "+str(goodcount))
print()
print("Program time elapsed:")
```

Output:

```
[rrichard@n034 barney]$  ./dedup.script Dataset1.sam STL96.txt keep


Note that the EASYBUILD tree is NOT supported by RACS.

Please contact user at sydes@uoregon.edu for questions regarding this tree.



Note that the PRL tree is NOT supported by RACS.

Please contact user at sameer@cs.uoregon.edu for questions regarding this tree.


Deduper program instructions:

Program takes single-end reads with optional UMI file.
Duplicates can be kept in a file.duplicate.out.sam format if desired.
UMI file must be seperated by new line characters.
Program should be run as specified by below sample:


./dedup.script filewithreads.sam UMI.txt keep


NOTE: UMI.txt and keep are optional entries; keep indicates that duplicates should be
kept.Non-entry or incorrect entries will result in discarded duplicates.


UMI file: STL96.txt
SAM file sorted for processing. (Barcode, Chromosome/linkage group, Position)
Duplicates will be kept.
Reverse window is set to: 189093


Number of duplicates per UMI/randomer:
AACGCCAT        3485
AAGGTACG        4680
AATTCCGG        1900
ACACAGAG        3020
ACACTCAG        1963
ACACTGTG        4522
ACAGGACA        6128
ACCTGTAG        3349
ACGAAGGT        7433
ACGACTTG        5152
ACGTCAAC        2076
ACGTCATG        4276
```

ACTGTCAG 2332
ACTGTGAC 4328
AGACACTC 4041
AGAGGAGA 5074
AGCATCGT 2434
AGCATGGA 3319
AGCTACCA 4178
AGCTCTAG 1315
AGGACAAC 3324
AGGACATG 5313
AGGTTGCT 4711
AGTCGAGA 3173
AGTGCTGT 3071
ATAAGCGG 2201
ATCCATGG 2544
ATCGAACC 6387
ATCGCGTA 2950
ATCGTTGG 3614
CAACGATC 3258
CAACGTTG 4125
CAACTGGT 3269
CAAGTCGT 2661
CACACACA 2378
CAGTACTG 2928
CATCAGCA 2030
CATCGTTC 3857
CCAAGGTT 5733
CCTAGCTT 4072
CGATTACG 3980
CGCCTATT 4485
CGTTCCAT 2712
CGTTGGAT 3700
CTACGTTC 4215
CTACTCGT 2022
CTAGAGGA 3403
CTAGGAAG 2424
CTAGGTAC 3397
CTCAGTCT 4989
CTGACTGA 2639
CTGAGTGT 5340
CTGATGTG 4252
CTGTTCAC 2907
CTTCGTTG 3499
GAACAGGT 5441
GAAGACCA 4698
GAAGTGCA 2856
GACATGAG 4849
GAGAAGAG 5146
GAGAAGTC 5456

```
GATCCTAG        2560
GATGTCGT        3061
GCCGATAT        5283
GCCGATTA        4818
GCGGTATT        5190
GGAATTGG        4393
GGATAACG        4570
GGCCTAAT        2811
GGCGTATT        7288
GTCTTGTC        3738
GTGATGAG        3783
GTGATGTC        5528
GTGTACTG        3107
GTGTAGTC        3527
GTTCACCT        6498
GTTCTGCT        4188
GTTGTCGA        3037
TACGAACC        5968
TAGCAAGG        2536
TAGCTAGC        994
TAGGTTCG        3685
TATAGCGC        1461
TCAGGACT        5196
TCCACATC        4344
TCGACTTC        5609
TCGTAGGT        5088
TCGTCATC        4398
TGAGACTC        3957
TGAGAGTG        4203
TGAGTGAG        2774
TGCTTGGA        3114
TGGAGTAG        3145
TGTGTGTG        4518
TTCGCCTA        3249
TTCGTTCG        2372

Total duplicates: 369005
Remaining reads: 635985

Program time elapsed:

real    0m9.749s
user    0m9.213s
sys     0m0.298s
[rrichard@n034 barney]$  ./dedup.script Dataset2.sam STL96.txt keep


Note that the EASYBUILD tree is NOT supported by RACS.
```

```
Please contact user at sydes@uoregon.edu for questions regarding this tree.



Note that the PRL tree is NOT supported by RACS.

Please contact user at sameer@cs.uoregon.edu for questions regarding this tree.


Deduper program instructions:

Program takes single-end reads with optional UMI file.
Duplicates can be kept in a file.duplicate.out.sam format if desired.
UMI file must be seperated by new line characters.
Program should be run as specified by below sample:


./dedup.script filewithreads.sam UMI.txt keep


NOTE: UMI.txt and keep are optional entries; keep indicates that duplicates should be
kept.
Non-entry or incorrect entries will result in discarded duplicates.


UMI file: STL96.txt
SAM file sorted for processing. (Barcode, Chromosome/linkage group, Position)
Duplicates will be kept.
Reverse window is set to: 191192


Number of duplicates per UMI/randomer:
AACGCCAT        6105
AAGGTACG        8091
AATTCCGG        3384
ACACAGAG        5322
ACACTCAG        3494
ACACTGTG        7671
ACAGGACA        10394
ACCTGTAG        5646
ACGAAGGT        11988
ACGACTTG        8639
ACGTCAAC        3832
ACGTCATG        7452
ACTGTCAG        4165
ACTGTGAC        7313
AGACACTC        7104
AGAGGAGA        8666
AGCATCGT        4310
```

AGCATGGA 5633
AGCTACCA 7008
AGCTCTAG 2556
AGGACAAC 5760
AGGACATG 8755
AGGTTGCT 8219
AGTCGAGA 5473
AGTGCTGT 5221
ATAAGCGG 3788
ATCCATGG 4437
ATCGAACC 10842
ATCGCGTA 4967
ATCGTTGG 6492
CAACGATC 5205
CAACGTTG 6963
CAACTGGT 5471
CAAGTCGT 4595
CACACACA 4410
CAGTACTG 4943
CATCAGCA 3428
CATCGTTC 6641
CCAAGGTT 9796
CCTAGCTT 6908
CGATTACG 6896
CGCCTATT 7169
CGTTCCAT 4518
CGTTGGAT 6382
CTACGTTC 7149
CTACTCGT 3598
CTAGAGGA 5879
CTAGGAAG 4218
CTAGGTAC 5579
CTCAGTCT 8267
CTGACTGA 4423
CTGAGTGT 8654
CTGATGTG 7338
CTGTTCAC 5178
CTTCGTTG 6047
GAACAGGT 8953
GAAGACCA 7566
GAAGTGCA 5086
GACATGAG 8476
GAGAAGAG 9008
GAGAAGTC 9219
GATCCTAG 4297
GATGTCGT 5257
GCCGATAT 9041
GCCGATTA 8118
GCGGTATT 8629

```
GGAATTGG        7484
GGATAACG        7710
GGCCTAAT        4975
GGCGTATT        12132
GTCTTGTC        6345
GTGATGAG        6236
GTGATGTC        9172
GTGTACTG        5255
GTGTAGTC        6092
GTTCACCT        10802
GTTCTGCT        7369
GTTGTCGA        5157
TACGAACC        9903
TAGCAAGG        4368
TAGCTAGC        1652
TAGGTTCG        6050
TATAGCGC        2614
TCAGGACT        9112
TCCACATC        7341
TCGACTTC        9738
TCGTAGGT        8573
TCGTCATC        7645
TGAGACTC        6772
TGAGAGTG        7146
TGAGTGAG        4737
TGCTTGGA        5422
TGGAGTAG        5359
TGTGTGTG        7547
TTCGCCTA        5692
TTCGTTCG        4170

Total duplicates: 628602
Remaining reads: 744188

Program time elapsed:

real    0m12.998s
user    0m12.205s
sys     0m0.491s
[rrichard@n034 barney]$  ./dedup.script Dataset3.sam STL96.txt keep


Note that the EASYBUILD tree is NOT supported by RACS.

Please contact user at sydes@uoregon.edu for questions regarding this tree.



Note that the PRL tree is NOT supported by RACS.
```

```
Please contact user at sameer@cs.uoregon.edu for questions regarding this tree.


Deduper program instructions:

Program takes single-end reads with optional UMI file.
Duplicates can be kept in a file.duplicate.out.sam format if desired.
UMI file must be seperated by new line characters.
Program should be run as specified by below sample:


./dedup.script filewithreads.sam UMI.txt keep


NOTE: UMI.txt and keep are optional entries; keep indicates that duplicates should be
kept.
Non-entry or incorrect entries will result in discarded duplicates.


UMI file: STL96.txt
SAM file sorted for processing. (Barcode, Chromosome/linkage group, Position)
Duplicates will be kept.
Reverse window is set to: 550620


Number of duplicates per UMI/randomer:
AACGCCAT        15509
AAGGTACG        20755
AATTCCGG        7870
ACACAGAG        13815
ACACTCAG        8699
ACACTGTG        20144
ACAGGACA        27432
ACCTGTAG        13891
ACGAAGGT        32128
ACGACTTG        22570
ACGTCAAC        9652
ACGTCATG        19845
ACTGTCAG        10166
ACTGTGAC        19761
AGACACTC        18393
AGAGGAGA        21515
AGCATCGT        10698
AGCATGGA        14311
AGCTACCA        17925
AGCTCTAG        5803
AGGACAAC        14459
AGGACATG        23961
```

AGGTTGCT	22196
AGTCGAGA	13840
AGTGCTGT	12912
ATAAGCGG	9393
ATCCATGG	11187
ATCGAACC	29083
ATCGCGTA	12641
ATCGTTGG	16422
CAACGATC	13764
CAACGTTG	18042
CAACTGGT	13991
CAAGTCGT	10919
CACACACA	10257
CAGTACTG	12455
CATCAGCA	7722
CATCGTTC	17256
CCAAGGTT	26135
CCTAGCTT	17236
CGATTACG	17471
CGCCTATT	18449
CGTTCCAT	11178
CGTTGGAT	16443
CTACGTTC	18405
CTACTCGT	8126
CTAGAGGA	14947
CTAGGAAG	10476
CTAGGTAC	14598
CTCAGTCT	22162
CTGACTGA	11398
CTGAGTGT	22967
CTGATGTG	18511
CTGTTCAC	13077
CTTCGTTG	15277
GAACAGGT	23364
GAAGACCA	20098
GAAGTGCA	13085
GACATGAG	22679
GAGAAGAG	23328
GAGAAGTC	24448
GATCCTAG	10545
GATGTCGT	13297
GCCGATAT	23473
GCCGATTA	21457
GCGGTATT	22579
GGAATTGG	19059
GGATAACG	20521
GGCCTAAT	12622
GGCGTATT	32264
GTCTTGTC	16670

```
GTGATGAG        17078
GTGATGTC        24332
GTGTACTG        12993
GTGTAGTC        15442
GTTCACCT        28595
GTTCTGCT        19312
GTTGTCGA        13042
TACGAACC        25889
TAGCAAGG        10678
TAGCTAGC        3881
TAGGTTCG        15818
TATAGCGC        5957
TCAGGACT        24007
TCCACATC        17997
TCGACTTC        25738
TCGTAGGT        22438
TCGTCATC        19705
TGAGACTC        17619
TGAGAGTG        18113
TGAGTGAG        11696
TGCTTGGA        13077
TGGAGTAG        13633
TGTGTGTG        19140
TTCGCCTA        14425
TTCGTTCG        10235

Total duplicates: 1618567
Remaining reads: 4053644

Program time elapsed:

real    1m2.942s
user    0m59.997s
sys     0m1.894s
```