

THE

The TagDust2 Benchmark

MANUAL

Timo Lassmann



11 SEPTEMBER 2014

Contents

Contents	ii
1 Introduction	1
2 Figure 1:	3
3 Figure 2-5: Comparison to BTRIM, cutadapt and fastx_barcode_clipper.	5
3.1 Simreads	5
Simulation without linkers:	5
Simulation with linkers:	6
3.2 Notes on BTRIM, FastX and Cutadapt used in the benchmark	6
Program Versions:	6
3.3 Command line options used in the benchmark	6
TagDust	6
fastx_barcode_splitter.pl	7
Cutadapt	7
fastx_barcode_splitter.pl after cutadapt	7
Btrim	7
3.4 Evaluation	7
Notes on evalres	8
3.5 Plotting	8
File: plotting.R	8
3.6 File: run.mk	10
File: barread.sh	11
File: 5barread3.sh	14
4 Figure 6	19
4.1 File: process_umi_data.sh	19
4.2 File: make_umi_plots.R	23
5 Table 1: Comparison to Illumina's bcl2fastq.	25
5.1 Run the default pipeline on the validation data containing barcodes:	25
5.2 Run the pipeline without de-multiplexing and use TagDust2.	26
5.3 Run TagDust2	26
5.4 Comparison	27
6 Table 2: Summary of Single Cell extracted reads.	29
7 Additional files:	31
7.1 File: Figure1.tex:	31

Chapter 1

Introduction

The purpose of this document is to provide a complete description and the supporting code to reproduce the figures from the TagDust2 paper.

Typing **make** at the top level directory runs a C parser to extract bits of code embedded in this document. Among these is an additional file **run.mk**. Running:

```
bash-3.2$ make -f run.mk
```

will start re-running the benchmark to make Figures 1-5.

The other figures and table require downloading large datasets which are not included in this package.

The following chapters describe the code for Figures 1-6 and Table 1 and 2.

Chapter 2

Figure 1:

The figure is a drawing made in tikz ¹. The code is listed [here](#).

To make the pdf version simply:

```
bash-3.2$ pdflatex Figure1.tex
```

¹<http://sourceforge.net/projects/pgf/>

Chapter 3

Figure 2-5: Comparison to BTRIM, cutadapt and fastx_barcode_clipper.

The purpose to these benchmarks is to compare the accuracy and number of reads extracted by running different extraction programs.

All benchmarks are carried out using the following logic and programs

- 1) generate dataset using the program **simreads**
- 2) run program X,Y and Z
- 3) compare results using the program **evalres**

3.1 Simreads

SIMreads is a simple C program distributed alongside TagDust2 to simulate raw NGS reads, including barcodes, linkers and adaptors. The syntax is fairly specific to the types of benchmarks conducted in the TagDust2 paper but perhaps of some general use.

Apart from simulating the reads, *simreads* will also generate a number of auxiliary files needed by the de-multiplex programs used in the benchmark. These include:

- TagDust2 architecture files (*XXX_tagdust_arch.txt*)
- BTRIM pattern file (*XXX_btrim_pattern.txt*)
- FastX barcode file (*XXX_fastxbarcodefile.txt*)

Here are the command line options used:

Simulation without linkers:

Here I simulate NGS sequences containing a barcode at the 5' end followed by the read we wish to extract:

```
error=$(../../src/simreads $barcodes -seed 42 -sim_barnum ${numbar[$c]}  
-sim_readlen 20 -sim_readlen_mod 0 -sim_numseq 100000 -sim_endloss 0 -  
sim_random_frac 0.1 -o $name -sim_error_rate ${array[$j]} 2>&1 )
```

The important options include:

- **\$barcodes** - points to a specific barcode file generated by the program `design_edit_metric_tags.py` from the EditTag package¹. To ensure reproducibility I distribute TagDust2 with a selection of barcode files found in the `dev/` directory.

¹Faircloth BC, Glenn TC. 2012. Not all sequence tags are created equal: Designing and validating sequence identification tags robust to indels. PLoS ONE 7(8): e42543. doi: 10.1371/journal.pone.0042543

- **sim_error_rate** - the sequencer error rate. To generate the paper figures I vary the error rate from 1 to 3 percent in 0.5% increments.
- **sim_barnum** - specifies the number of barcodes used from the files pointed at by **\$barcodes**.
- **sim_readlen** - specifies the length of the reads.
- **sim_numseq** - the number of sequences simulated.
- **sim_random_frac** - the fraction of sequences which are completely random. I.e. these should not be extracted.

Simulation with linkers:

Here I simulate NGS sequences as before but add 5' and 3' linkers:

```
error=$(../../src/simreads $barcodes -sim_5seq agggaggacgatgcgg -
sim_3seq gtgtcagtcacttccagcgg -seed 42 -sim_barnum ${numbar[$c]} -
sim_readlen 20 -sim_readlen_mod 0 -sim_numseq 100000 -sim_endloss 0 -
sim_random_frac 0.1 -o $name -sim_error_rate ${array[$j]} 2>&1 )
```

The important options include:

- **sim_5seq** - sequence appended to the 5' end (agggaggacgatgcgg)
- **sim_3seq** - sequence appended to the 3' end (gtgtcagtcacttccagcgg)

3.2 Notes on BTRIM, FastX and Cutadapt used in the benchmark

All programs used for the benchmark are distributed alongside TagDust2 distribution in `/reproducibiliy/bin`.

Program Versions:

BTRIM:

btrim64 - downloaded from:

<http://graphics.med.yale.edu/trim/>

btrim64 28-Jul-2014 15:56 35K

(version unknown)

cutadapt:

Version 1.3

fastx_barcode_splitter.pl

fastx_0.0.13

3.3 Command line options used in the benchmark

TagDust

TagDust is run in two different modes:

- 1) Basic mode - only one architecture is given.
- 2) Auto-detect mode - all architectures used in the **entire** benchmark are given and TagDust2 has to figure out which one to use.

Basic command

```
error=$( ../../src/tagdust -t 80 -seed 42 $name -arch $arch -o $outfile 2>&1 )
```

- **\$arch** - name of the file containing the architecture used in a particular test.
- **\$name** - name of the input file.
- **\$outfile** - name of the output file(s).

Auto-detect architecture

```
error=$( ../../src/tagdust -t 80 -seed 42 $name -arch all_arch.txt -o $outfile 2>&1 )
```

Here TagDust2 will search for the appropriate architecture in the file **all_arch.txt** no matter what the input is. Essentially this is how TagDust2 should be used in production environments. The user or operator specifies all the possible architectures based on the library preparation protocols used and TagDust2 automatically figures out which one is appropriate for each set of raw output files.

fastx_barcode_splitter.pl

```
error=$( cat $name | ../bin/fastx_barcode_splitter.pl --bcfile $arch --prefix $outfile -bol 2>&1)
```

The **\$arch** variable contains a link to the file containing the barcodes used to simulate the sequences.

Cutadapt

Cutadapt is run first, then the output is de-multiplexed by **fastx_barcode_splitter.pl**.

```
error=$( ../bin/cutadapt --discard-untrimmed -n 2 --front agggaggacgatgcgg --adapter=gtgtcagtcacttccagcgg $name -o $outfile 2>&1)
```

fastx_barcode_splitter.pl after cutadapt

Here is the command to de-multiplex the output of cutadapt.

```
error=$( cat $cutadaptout | ../bin/fastx_barcode_splitter.pl --bcfile $arch --prefix $outfile -bol 2>&1)
```

Btrim

```
error=$( ../bin/btrim64 -p $arch -t $name -o $outfile -l 18 -B 2>&1)
```

The file **\$arch** contains the barcode sequences and the linkers.

3.4 Evaluation

During the simulation **simreads** adds the following labels to all reads:

- 1) **READ** or **RANDOM** - indicating whether the sequence is a random background sequence or not.
- 2) **SEQ** the actual read sequence before applying errors.
- 3) **RBC** the barcode sequence used for this read.
- 4) **BARNUM** the index of the barcode sequence.

Here is an example of a simulated read:


```

17 ggplot(m, aes(x = simerror,y = value ,color=Program, fill=factor(Program))
   ) + geom_line() +facet_grid(scales="free", variable ~ barcodes ) +
   scale_x_continuous(breaks=c(0.01,0.015,0.02,0.025,0.03), labels =
   expression("1","1.5","2","2.5","3")) + xlab("Simulated Error Rate (%)")
   ) + scale_y_continuous(labels=yaxis_label_format)+ ylab("")
18 ggsave("Figure2.pdf",width=8,height=4,dpi = 300)
19
20 mat = read.table("barread_6nt_4r.tsv",header =T,sep="\t")
21 mat$barcodes = paste(mat$barcodes, "Barcodes")
22 mat$barcodes = factor(mat$barcodes, levels=c('8 Barcodes','24 Barcodes','
   48 Barcodes'))
23 s = subset(mat,mat$Program != "tagdustallarch")
24 s$Program = factor(s$Program, levels=c('tagdust','fastx','btrim'))
25 m = melt(s, id=c("Program","simerror","barcodes"),measure.vars = c("Recall
   ", "Precision"))
26 ggplot(m, aes(x = simerror,y = value ,color=Program, fill=factor(Program))
   ) + geom_line() +facet_grid(scales="free", variable ~ barcodes ) +
   scale_x_continuous(breaks=c(0.01,0.015,0.02,0.025,0.03), labels =
   expression("1","1.5","2","2.5","3")) + xlab("Simulated Error Rate (%)")
   ) + scale_y_continuous(labels=yaxis_label_format)+ ylab("")
27 ggsave("Figure3.pdf",width=8,height=4,dpi = 300)
28
29 mat = read.table("5barread3_4nt_4r.tsv",header =T,sep="\t")
30 mat$barcodes = paste(mat$barcodes, "Barcodes")
31 mat$barcodes = factor(mat$barcodes, levels=c('8 Barcodes','24 Barcodes','
   48 Barcodes'))
32 s = subset(mat,mat$Program != "tagdustallarch")
33 s$Program = factor(s$Program, levels=c('tagdust','fastx','btrim'))
34 m = melt(s, id=c("Program","simerror","barcodes"),measure.vars = c("Recall
   ", "Precision"))
35 ggplot(m, aes(x = simerror,y = value ,color=Program, fill=factor(Program))
   ) + geom_line() +facet_grid(scales="free", variable ~ barcodes ) +
   scale_x_continuous(breaks=c(0.01,0.015,0.02,0.025,0.03), labels =
   expression("1","1.5","2","2.5","3")) + xlab("Simulated Error Rate (%)")
   ) + scale_y_continuous(labels=yaxis_label_format) + ylab("")
36 ggsave("Figure4.pdf",width=8,height=4,dpi = 300)
37
38
39 mat = read.table("5barread3_6nt_4r.tsv",header =T,sep="\t")
40 mat$barcodes = paste(mat$barcodes, "Barcodes")
41 mat$barcodes = factor(mat$barcodes, levels=c('8 Barcodes','24 Barcodes','
   48 Barcodes'))
42 s = subset(mat,mat$Program != "tagdustallarch")
43 s$Program = factor(s$Program, levels=c('tagdust','fastx','btrim'))
44 m = melt(s, id=c("Program","simerror","barcodes"),measure.vars = c("Recall
   ", "Precision"))
45 ggplot(m, aes(x = simerror,y = value ,color=Program, fill=factor(Program))
   ) + geom_line() +facet_grid(scales="free", variable ~ barcodes ) +
   scale_x_continuous(breaks=c(0.01,0.015,0.02,0.025,0.03), labels =
   expression("1","1.5","2","2.5","3")) + xlab("Simulated Error Rate (%)")
   ) + scale_y_continuous(labels=yaxis_label_format)+ ylab("")
46 ggsave("Figure5.pdf",width=8,height=4,dpi = 300)

```

3.6 File: run.mk

Runs all the scripts to make the paper figures 2-5. At first all the test datasets are generated once and their corresponding TagDust2 architecture files are collected. The file containing all architectures is then used to see whether TagDust2 can automatically detect the correct architectures when the actual benchmark is run later on.

```
1
2 .PHONY: message benchmark rplot
3
4 benchmark: Figure1.pdf Figure2.pdf
5     @echo Done
6
7 Figure1.pdf: Figure1.tex
8     pdflatex $<;
9
10 Figure2.pdf: barread_6nt_4r.tsv barread_4nt_4r.tsv 5barread3_4nt_4r.tsv 5
11     barread3_6nt_4r.tsv
12     R --slave --vanilla < plotting.R
13     rm Rplots.pdf
14
15 barread_4nt_4r.tsv: barread_results.txt
16     cat barread_results.txt | grep _4nt_ | awk 'BEGIN{print "Program\
17     tsimerror\tbarcodes\tRecall\tSpecificity\tPrecision\tKappa"} {if(NR
18     >0){x = split($$1,a,"_");printf "%s\t%f\t%f\t%f\t%f\t%f\t%f\n" , a
19     [1],a[2],a[3],$$2,$$3,$$4,$$5 }}' > barread_4nt_4r.tsv
20
21 5barread3_4nt_4r.tsv: 5barread3_results.txt
22     cat 5barread3_results.txt | grep _4nt_ | awk 'BEGIN{print "Program\
23     tsimerror\tbarcodes\tRecall\tSpecificity\tPrecision\tKappa"} {if(NR
24     >0){x = split($$1,a,"_");printf "%s\t%f\t%f\t%f\t%f\t%f\t%f\n" , a
25     [1],a[2],a[3],$$2,$$3,$$4,$$5 }}' > 5barread3_4nt_4r.tsv
26
27 barread_6nt_4r.tsv: barread_results.txt
28     cat barread_results.txt | grep _6nt_ | awk 'BEGIN{print "Program\
29     tsimerror\tbarcodes\tRecall\tSpecificity\tPrecision\tKappa"} {if(NR
30     >0){x = split($$1,a,"_");printf "%s\t%f\t%f\t%f\t%f\t%f\t%f\n" , a
31     [1],a[2],a[3],$$2,$$3,$$4,$$5 }}' > barread_6nt_4r.tsv
32
33 5barread3_6nt_4r.tsv: 5barread3_results.txt
34     cat 5barread3_results.txt | grep _6nt_ | awk 'BEGIN{print "Program\
35     tsimerror\tbarcodes\tRecall\tSpecificity\tPrecision\tKappa"} {if(NR
36     >0){x = split($$1,a,"_");printf "%s\t%f\t%f\t%f\t%f\t%f\t%f\n" , a
37     [1],a[2],a[3],$$2,$$3,$$4,$$5 }}' > 5barread3_6nt_4r.tsv
38
39 barread_results.txt: all_arch.txt
40     ./barread.sh -b ../../dev/EDITTAG_4nt_ed_2.txt -r
41     ./barread.sh -b ../../dev/EDITTAG_6nt_ed_3.txt -r
42
43 5barread3_results.txt: all_arch.txt
44     ./5barread3.sh -b ../../dev/EDITTAG_4nt_ed_2.txt -r
45     ./5barread3.sh -b ../../dev/EDITTAG_6nt_ed_3.txt -r
46
47 all_arch.txt:
```

```

36 ./barread.sh -b ../../dev/EDITTAG_4nt_ed_2.txt
37 ./barread.sh -b ../../dev/EDITTAG_6nt_ed_3.txt
38 ./5barread3.sh -b ../../dev/EDITTAG_4nt_ed_2.txt
39 ./5barread3.sh -b ../../dev/EDITTAG_6nt_ed_3.txt
40 cat *tagdust_arch.txt | sort | uniq > all_arch.txt
41
42 all: message
43
44 message:
45 @echo To reproduce the figures run "make -f run.mk benchmark"

```

File: barread.sh

Runs a benchmark consisting of simulating sequences with a barcode followed by the reads. There are two options:

- b - should point to a file generated by *design_edit_metric_tags.py*
- r - toggle to run the benchmark. If not used, simreads will simulate reads and write TagDust architecture files but will not run the actual benchmark.

```

1  #!/bin/bash
2
3  echo "Running tagdust benchmarks:";
4
5  barcodes=
6  run=0
7
8  function usage()
9  {
10 cat <<EOF
11 usage: $0 -b <barcode file> -r
12 EOF
13 exit 1;
14 }
15
16 while getopts b:r opt
17 do
18 case ${opt} in
19 b) barcodes=${OPTARG};;
20 r) run=1;;
21 *) usage;;
22 esac
23 done
24
25 if [ "${barcodes}" = "" ]; then usage; fi

```

Below are the ranges of error rates and number of barcodes used:

```

1 array=(0.01 0.015 0.02 0.025 0.03)
2 len=${#array[*]}
3
4 numbar=(8 24 48)
5 lenbar=${#numbar[*]}

```

If the benchmark should be run remove all temporary files left over from a previous run:

```

1  if [[ $run -eq 1 ]]; then
2  make --silent clean
3  fi
4
5  for (( c=0; c < $lenbar; c+=1 )); do
6  j=0
7
8  while [ $j -lt $len ]
9  do
10 name="barread${array[$j]}_${numbar[$c]}.fq"
11
12 ## Simulation without linkers:
13
14 status=$?
15 if [[ $status -eq 0 ]]; then
16 printf "%20s%10s\n"    sim SUCCESS;
17 else
18 printf "%20s%10s\n"    sim FAILED;
19 printf "with ERROR $status and Message:\n\n$error\n\n";
20 exit 1;
21 fi
22
23 if [[ $run -eq 1 ]]; then
24
25 suffix="_tagdust_arch.txt"
26 arch=$name$suffix
27 suffix="_tagdust";
28 outfile=$name$suffix
29
30 ## Basic command
31
32 status=$?
33 if [[ $status -eq 0 ]]; then
34 printf "%20s%10s\n"    tagdust SUCCESS;
35 else
36 printf "%20s%10s\n"    tagdust FAILED;
37 printf "with ERROR $status and Message:\n\n$error\n\n";
38 exit 1;
39 fi
40
41 suffix="_tagdustarchsel";
42 outfile=$name$suffix
43
44 ## Auto-detect architecture
45 status=$?
46 if [[ $status -eq 0 ]]; then
47 printf "%20s%10s\n"    tagdust SUCCESS;
48 else
49 printf "%20s%10s\n"    tagdust FAILED;
50 printf "with ERROR $status and Message:\n\n$error\n\n";
51 exit 1;
52 fi
53
54 suffix="_fastxbarcodefile.txt"
55 arch=$name$suffix

```

```

56 suffix="_fastx";
57 outfile=$name$suffix
58
59 ## fastx_barcode_splitter.pl
60 status=$?
61 if [[ $status -eq 0 ]]; then
62 printf "%20s%10s\n" fastx_split SUCCESS;
63 else
64 printf "%20s%10s\n" fastx_split FAILED;
65 printf "with ERROR $status and Message:\n\n$error\n\n";
66 exit 1;
67 fi
68
69 suffix="_tagdust_BC_";
70 outfile=$name$suffix
71
72 error=$( ../../src/evalres -name tagdust_${array[$j]}_${numbar[$c]}
    _$barcodes $outfile*.fq -o barread -sim_random_frac 0.1 -sim_numseq
    100000 2>&1)
73 status=$?
74 if [[ $status -eq 0 ]]; then
75 printf "%20s%10s\n" eval_run SUCCESS;
76 else
77 printf "%20s%10s\n" eval_run FAILED;
78 printf "with ERROR $status and Message:\n\n$error\n\n";
79 exit 1;
80 fi
81
82 suffix="_tagdustarchsel_BC_";
83 outfile=$name$suffix
84
85 error=$( ../../src/evalres -name tagdustallarch_${array[$j]}_${numbar[$c]}
    _$barcodes $outfile*.fq -o barread -sim_random_frac 0.1 -sim_numseq
    100000 2>&1)
86 status=$?
87 if [[ $status -eq 0 ]]; then
88 printf "%20s%10s\n" eval_run SUCCESS;
89 else
90 printf "%20s%10s\n" eval_run FAILED;
91 printf "with ERROR $status and Message:\n\n$error\n\n";
92 exit 1;
93 fi
94
95 suffix="_fastxBC";
96 outfile=$name$suffix
97
98 error=$( ../../src/evalres -name fastx_${array[$j]}_${numbar[$c]}
    _$barcodes $outfile* -o barread -sim_random_frac 0.1 -sim_numseq
    100000 2>&1)
99 status=$?
100 if [[ $status -eq 0 ]]; then
101 printf "%20s%10s\n" eval_run SUCCESS;
102 else
103 printf "%20s%10s\n" eval_run FAILED;
104 printf "with ERROR $status and Message:\n\n$error\n\n";

```

```

105 exit 1;
106 fi
107 fi
108
109 let j++
110 let myrun++
111 done
112
113 done
114
115 if [[ $run -eq 1 ]]; then
116 make --silent clean
117 fi

```

File: 5barread3.sh

```

1  #!/bin/bash
2
3  echo "Running tagdust benchmarks:";
4
5  barcodes=
6  run=0
7
8  function usage()
9  {
10 cat <<EOF
11 usage: $0 -b <barcode file> -r
12 EOF
13 exit 1;
14 }
15
16 while getopts b:r opt
17 do
18 case ${opt} in
19 b) barcodes=${OPTARG};;
20 r) run=1;;
21 *) usage;;
22 esac
23 done
24
25 if [ "${barcodes}" = "" ]; then usage; fi

```

Below are the ranges of error rates and number of barcodes used:

```

1
2 array=(0.01 0.015 0.02 0.025 0.03)
3 len=${#array[*]}
4
5 numbar=(8 24 48)
6 lenbar=${#numbar[*]}

```

If the benchmark should be run remove all temporary files left over from a previous run:

```

1
2 if [[ $run -eq 1 ]]; then
3 make --silent clean

```



```

4  fi
5
6  #barcodes="EDITTAG_4nt_ed_2.txt"
7
8  for (( c=0; c < $lenbar; c+=1 )); do
9
10 j=0
11
12 while [ $j -lt $len ]
13 do
14
15 name="5barread3${array[$j]}_${numbar[$c]}.fq"
16
17 ## Simulation with linkers:
18
19 status=$?
20 if [[ $status -eq 0 ]]; then
21 printf "%20s%10s\n"    sim SUCCESS;
22 else
23 printf "%20s%10s\n"    sim FAILED;
24 printf "with ERROR $status and Message:\n\n$error\n\n";
25 exit 1;
26 fi
27
28 if [[ $run -eq 1 ]]; then
29 suffix="_tagdust_arch.txt"
30 arch=$name$suffix
31 suffix="_tagdust";
32 outfile=$name$suffix
33
34 ## Basic command
35 status=$?
36 if [[ $status -eq 0 ]]; then
37 printf "%20s%10s\n"    tagdust SUCCESS;
38 else
39 printf "%20s%10s\n"    tagdust FAILED;
40 printf "with ERROR $status and Message:\n\n$error\n\n";
41 exit 1;
42 fi
43
44
45 suffix="_tagdustarchsel";
46 outfile=$name$suffix
47
48 ## Auto-detect architecture
49 status=$?
50 if [[ $status -eq 0 ]]; then
51 printf "%20s%10s\n"    tagdust SUCCESS;
52 else
53 printf "%20s%10s\n"    tagdust FAILED;
54 printf "with ERROR $status and Message:\n\n$error\n\n";
55 exit 1;
56 fi
57
58 suffix="_btrim_pattern.txt"

```

```

59 arch=$name$suffix
60 suffix="_btrim";
61 outfile=$name$suffix
62
63 ## Btrim
64 status=$?
65 if [[ $status -eq 0 ]]; then
66 printf "%20s%10s\n" btrim SUCCESS;
67 else
68 printf "%20s%10s\n" btrim FAILED;
69 printf "with ERROR $status and Message:\n\n$error\n\n";
70 exit 1;
71 fi
72
73 suffix="_cutadadapt.fq";
74 outfile=$name$suffix
75
76 ## Cutadapt
77 status=$?
78 if [[ $status -eq 0 ]]; then
79 printf "%20s%10s\n" cutadapt SUCCESS;
80 else
81 printf "%20s%10s\n" cutadapt FAILED;
82 printf "with ERROR $status and Message:\n\n$error\n\n";
83 exit 1;
84 fi
85
86
87 suffix="_cutadadapt.fq";
88 cutadaptout=$name$suffix
89 suffix="_fastxbarcodefile.txt"
90 arch=$name$suffix
91 suffix="_fastx";
92 outfile=$name$suffix
93
94 ## fastx_barcode_splitter.pl after cutadapt
95
96 status=$?
97 if [[ $status -eq 0 ]]; then
98 printf "%20s%10s\n" fastx_split SUCCESS;
99 else
100 printf "%20s%10s\n" fastx_split FAILED;
101 printf "with ERROR $status and Message:\n\n$error\n\n";
102 exit 1;
103 fi
104
105 suffix="_tagdust_BC_";
106 outfile=$name$suffix
107
108 error=$( ../../src/evalres -name tagdust_${array[$j]}_${numbar[$c]}
        _$barcodes $outfile*.fq -o 5barread3 -sim_random_frac 0.1 -sim_numseq
        100000 2>&1)
109 status=$?
110 if [[ $status -eq 0 ]]; then
111 printf "%20s%10s\n" eval_run SUCCESS;

```

```

112 else
113 printf "%20s%10s\n" eval_run FAILED;
114 printf "with ERROR $status and Message:\n\n$error\n\n";
115 exit 1;
116 fi
117
118
119 suffix="_tagdustarchsel_BC_";
120 outfile=$name$suffix
121
122 error=$( ../../src/evalres -name tagdustallarch_${array[$j]}_${numbar[$c]}
    _$barcodes $outfile*.fq -o 5barread3 -sim_random_frac 0.1 -sim_numseq
    100000 2>&1)
123 status=$?
124 if [[ $status -eq 0 ]]; then
125 printf "%20s%10s\n" eval_run SUCCESS;
126 else
127 printf "%20s%10s\n" eval_run FAILED;
128 printf "with ERROR $status and Message:\n\n$error\n\n";
129 exit 1;
130 fi
131
132 suffix="_btrim";
133 outfile=$name$suffix
134
135 error=$( ../../src/evalres -name btrim_${array[$j]}_${numbar[$c]}
    _$barcodes $outfile* -o 5barread3 -sim_random_frac 0.1 -sim_numseq
    100000 2>&1)
136 status=$?
137 if [[ $status -eq 0 ]]; then
138 printf "%20s%10s\n" eval_run SUCCESS;
139 else
140 printf "%20s%10s\n" eval_run FAILED;
141 printf "with ERROR $status and Message:\n\n$error\n\n";
142 exit 1;
143 fi
144
145 suffix="_fastxBC";
146 outfile=$name$suffix
147
148
149 error=$( ../../src/evalres -name fastx_${array[$j]}_${numbar[$c]}
    _$barcodes $outfile* -o 5barread3 -sim_random_frac 0.1 -sim_numseq
    100000 2>&1)
150 status=$?
151 if [[ $status -eq 0 ]]; then
152 printf "%20s%10s\n" eval_run SUCCESS;
153 else
154 printf "%20s%10s\n" eval_run FAILED;
155 printf "with ERROR $status and Message:\n\n$error\n\n";
156 exit 1;
157 fi
158
159 fi;
160

```

```
161 let j++
162 let myrun++
163 done
164
165 done
166
167 if [[ $run -eq 1 ]]; then
168     make --silent clean
169 fi
```

Chapter 4

Figure 6

Here we test the ability of TagDust2 to work with complicated read architectures including fingerprints. We use the following raw data files from the paper “Counting absolute numbers of molecules using unique molecular identifiers.”¹:

- ERR048988.fastq.bz2
- ERR048990.fastq.bz2
- ERR048992.fastq.bz2
- ERR048994.fastq.bz2
- ERR048989.fastq.bz2
- ERR048991.fastq.bz2
- ERR048993.fastq.bz2

To extract the reads we use the following read architecture file:

```
bash-3.2$ cat arch.txt
tagdust -t 80 -1 F:NNN -2 S:T -3 F:NNNN -4 S:T -5 F:NNN -6 B:GACTT -7 S:
      GGGG -8 R:N -start 1 -end 25
```

indicating that we expect a 10 nucleotide fingerprint sequence (F:) separated by thymidines (S:) followed by a 5 nucleotide barcode (B:), 4 guanosines and finally the mappable read (R:).

The commands to process the fastq files are:

4.1 File: process_umi_data.sh

```
1 tagdust -arch arch.txt -t 80 ERR048988.fastq -o 15
   c_repeat1_lane1_extracted
2 tagdust -arch arch.txt -t 80 ERR048990.fastq -o 25
   c_repeat1_lane1_extracted
3 tagdust -arch arch.txt -t 80 ERR048992.fastq -o
   RNA_corresponding_to_1000_cells_lane1_extracted
4 tagdust -arch arch.txt -t 80 ERR048994.fastq -o
   RNA_corresponding_to_10_cells_lane1_extracted
5 tagdust -arch arch.txt -t 80 ERR048989.fastq -o 15
   c_repeat2_lane1_extracted
6 tagdust -arch arch.txt -t 80 ERR048991.fastq -o 25
   c_repeat1_lane2_extracted
```

¹Teemu Kivioja, Anna VÃ¤hÃ¤drÃ¤utio, Kasper Karlsson, Martin Bonke, Martin Enge, Sten Linnarsson, and Jussi Taipale. Counting absolute numbers of molecules using unique molecular identifiers. Nature methods, 9(1):72–74, January 2012.

```

7 tagdust -arch arch.txt -t 80 ERR048993.fastq -o
  RNA_corresponding_to_100_cells_lane1_extracted

```

Note that we rename the files to better understand which fastq file belongs to which experiment.

The next step is to map the sequences using bwa²:

```

1
2 for file in ./*
3     do
4         if [ -f $file ]; then
5             if [[ $file =~ _extracted_BC_GACTT.fq$ ]]; then
6                 if [ -f $$file.sorted.bam ]; then
7                     echo "$file already processed"
8                 else
9                     echo "working on $file"
10                    sed -e 's/[ \t]//g' $file
11                    > tmp.fq
12                    bwa aln -t 80
13                    $GENOME_DIR tmp.fq |
14                    bwa samse $GENOME_DIR
15                    - tmp.fq | samtools
16                    view -Su - | samtools
17                    sort - $file.sorted
18                fi
19            fi
20        fi
21    done

```

\$GENOME_DIR should point to a BWA index of the Drosophila melanogaster (BDGP5) gene sequences (downloaded from <http://www.biomart.org>).

Now we extract reads mapping to same location, and sort them by fingerprint ID, chromosome and position.

```

1 samtools view -F 16 -q 20 15c_repeat1_lane1_extracted_BC_GACTT.fq.sorted.
  bam | awk '{x = split($1,a,";"); b = substr(a[2],4);print b , $3 , $4 }'
  | sort -S 16G -k 2,2 -k 3,3n -k 1,1n | uniq -c > 15
  c_repeat1_lane1_extracted_processed.csv
2
3 samtools view -F 16 -q 20
  RNA_corresponding_to_1000_cells_lane1_extracted_BC_GACTT.fq.sorted.bam
  | awk '{x = split($1,a,";"); b = substr(a[2],4);print b , $3 , $4 }'|
  sort -S 16G -k 2,2 -k 3,3n -k 1,1n | uniq -c >
  RNA_corresponding_to_1000_cells_lane1_extracted_processed.csv
4
5
6 samtools view -F 16 -q 20 15c_repeat2_lane1_extracted_BC_GACTT.fq.sorted.
  bam | awk '{x = split($1,a,";"); b = substr(a[2],4);print b , $3 , $4
  }'| sort -S 16G -k 2,2 -k 3,3n -k 1,1n | uniq -c > 15
  c_repeat2_lane1_extracted_processed.csv &
7
8 samtools view -F 16 -q 20
  RNA_corresponding_to_100_cells_lane1_extracted_BC_GACTT.fq.sorted.bam |

```

²Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. Bioinformatics, 25:1754-60. [PMID: 19451168]

```

    awk '{x = split($1,a,";"); b = substr(a[2],4);print b , $3 , $4 }' |
    sort -S 16G -k 2,2 -k 3,3n -k 1,1n | uniq -c >
    RNA_corresponding_to_100_cells_lane1_extracted_processed.csv
9
10
11 samtools view -F 16 -q 20 25c_repeat1_lane1_extracted_BC_GACTT.fq.sorted.
    bam | awk '{x = split($1,a,";"); b = substr(a[2],4);print b , $3 , $4 }' |
    | sort -S 16G -k 2,2 -k 3,3n -k 1,1n | uniq -c > 25
    c_repeat1_lane1_extracted_processed.csv
12
13 samtools view -F 16 -q 20
    RNA_corresponding_to_10_cells_lane1_extracted_BC_GACTT.fq.sorted.bam |
    awk '{x = split($1,a,";"); b = substr(a[2],4);print b , $3 , $4 }' |
    sort -S 16G -k 2,2 -k 3,3n -k 1,1n | uniq -c >
    RNA_corresponding_to_10_cells_lane1_extracted_processed.csv
14
15 samtools view -F 16 -q 20 25c_repeat1_lane2_extracted_BC_GACTT.fq.sorted.
    bam | awk '{x = split($1,a,";"); b = substr(a[2],4);print b , $3 , $4
    }' | sort -S 16G -k 2,2 -k 3,3n -k 1,1n | uniq -c > 25
    c_repeat1_lane2_extracted_processed.csv

```

Next we use the `groupBy` tool from the `bedtool` package ³ to either:

- sum or
- count

reads mapping to the same location with the same fingerprint.

```

1
2 cat 15c_repeat1_lane1_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t
    %s\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o count > 15
    c_repeat1_lane1_UMI.csv
3 cat 15c_repeat1_lane1_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t
    %s\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o sum > 15
    c_repeat1_lane1_NOUMI.csv
4
5 cat 15c_repeat2_lane1_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t
    %s\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o count > 15
    c_repeat2_lane1_UMI.csv
6 cat 15c_repeat2_lane1_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t
    %s\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o sum > 15
    c_repeat2_lane1_NOUMI.csv
7
8
9 cat 25c_repeat1_lane1_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t
    %s\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o count > 25
    c_repeat1_lane1_UMI.csv
10 cat 25c_repeat1_lane1_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t
    %s\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o sum > 25
    c_repeat1_lane1_NOUMI.csv
11
12 cat 25c_repeat1_lane2_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t
    %s\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o count > 25
    c_repeat1_lane2_UMI.csv

```

³Quinlan, Aaron R., and Ira M. Hall. "BEDTools: a flexible suite of utilities for comparing genomic features." *Bioinformatics* 26.6 (2010): 841-842.

```

13 cat 25c_repeat1_lane2_extracted_processed.csv | awk '{printf "%s\t%s\t%s\t%s\t\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o sum > 25
    c_repeat1_lane2_NOUMI.csv
14
15
16 cat RNA_corresponding_to_1000_cells_lane1_extracted_processed.csv | awk '{
    printf "%s\t%s\t%s\t%s\t\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o count
    > RNA_corresponding_to_1000_cells_UMI.csv
17 cat RNA_corresponding_to_1000_cells_lane1_extracted_processed.csv | awk '{
    printf "%s\t%s\t%s\t%s\t\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o sum >
    RNA_corresponding_to_1000_cells_NOUMI.csv
18
19 cat RNA_corresponding_to_100_cells_lane1_extracted_processed.csv | awk '{
    printf "%s\t%s\t%s\t%s\t\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o count
    > RNA_corresponding_to_100_cells_UMI.csv
20 cat RNA_corresponding_to_100_cells_lane1_extracted_processed.csv | awk '{
    printf "%s\t%s\t%s\t%s\t\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o sum >
    RNA_corresponding_to_100_cells_NOUMI.csv
21
22 cat RNA_corresponding_to_10_cells_lane1_extracted_processed.csv | awk '{
    printf "%s\t%s\t%s\t%s\t\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o count
    > RNA_corresponding_to_10_cells_UMI.csv
23 cat RNA_corresponding_to_10_cells_lane1_extracted_processed.csv | awk '{
    printf "%s\t%s\t%s\t%s\t\n", $1, $2, $3, $4}' | groupBy -g 3 -c 1 -o sum >
    RNA_corresponding_to_10_cells_NOUMI.csv

```

Here we combine the results from replicas.

```

1
2 cat 15c_repeat1_lane1_NOUMI.csv 15c_repeat2_lane1_NOUMI.csv | sort -k 1,1
    | groupBy -g 1 -c 2 -o sum > 15c_combined_noumi.csv
3 cat 15c_repeat1_lane1_UMI.csv 15c_repeat2_lane1_UMI.csv | sort -k 1,1 |
    groupBy -g 1 -c 2 -o sum > 15c_combined_UMI.csv
4
5 cat 25c_repeat1_lane1_NOUMI.csv 25c_repeat1_lane2_NOUMI.csv | sort -k 1,1
    | groupBy -g 1 -c 2 -o sum > 25c_combined_noumi.csv
6 cat 25c_repeat1_lane1_UMI.csv 25c_repeat1_lane2_UMI.csv | sort -k 1,1 |
    groupBy -g 1 -c 2 -o sum > 25c_combined_UMI.csv

```

We now join the processed files either using UMIs or not for easy plotting with R:

```

1
2 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat1_lane1_UMI.csv 15
    c_repeat2_lane1_UMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_umi.csv
3
4 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 25c_repeat1_lane1_NOUMI.csv 25
    c_repeat1_lane2_NOUMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_25c_noumi.csv
5
6 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 25c_repeat1_lane1_UMI.csv 25
    c_repeat1_lane2_UMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_25c_umi.csv
7
8 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat1_lane1_NOUMI.csv 25
    c_repeat1_lane1_NOUMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >

```



```

9      comp_15c_repeat1_25c_repeat1_noumi.csv
10 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat1_lane1_UMI.csv 25
    c_repeat1_lane1_UMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_repeat1_25c_repeat1_umi.csv
11
12 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat2_lane1_NOUMI.csv 25
    c_repeat1_lane2_NOUMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_repeat2_25c_repeat1_lane2_noumi.csv
13
14 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat2_lane1_UMI.csv 25
    c_repeat1_lane2_UMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_repeat2_25c_repeat1_lane2_umi.csv
15
16 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat1_lane1_NOUMI.csv 25
    c_repeat1_lane2_NOUMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_repeat1_25c_repeat1_lane2_noumi.csv
17
18 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat1_lane1_UMI.csv 25
    c_repeat1_lane2_UMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_repeat1_25c_repeat1_lane2_umi.csv
19
20 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat2_lane1_NOUMI.csv 25
    c_repeat1_lane1_NOUMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_repeat2_25c_repeat1_lane1_noumi.csv
21
22 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_repeat2_lane1_UMI.csv 25
    c_repeat1_lane1_UMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' >
    comp_15c_repeat2_25c_repeat1_lane1_umi.csv
23
24 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_combined_noumi.csv 25
    c_combined_noumi.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' > 15
    _25_combined_noumi.csv
25 join -a1 -a2 -1 1 -2 1 -o 0 1.2 2.2 -e "0" 15c_combined_UMI.csv 25
    c_combined_UMI.csv | awk '{printf "%s\t%d\t%d\n", $1, $2, $3 }' > 15
    _25_combined_UMI.csv

```

4.2 File: make_umi_plots.R

Here are the commands used to generate panel one and two of Figure 6:

```

library(ggplot2)

mat = read.table("15_25_combined_noumi.csv", header = F);
cor = cor(mat[,2], mat[,3]);
frame=ggplot(mat, aes(mat[,2], mat[,3]))
frame + geom_point(colour="black", size=1, shape=20, alpha=0.3) + labs(x="15
cycles ", y="25 cycles", title="without UMI") + annotate("text", label =
paste("R=", round(cor, 6), sep=""), x = min(mat[,2])+10, y = max(mat
[,3]) / 10, size = 6, colour = "red") + scale_x_log10() + scale_y_log10
()+theme(axis.text=element_text(size=12), axis.title=element_text(size
=14, face="bold"), title=element_text(size=18, face="bold"));
ggsave("15_25_combined_noumi.pdf")
mat = read.table("15_25_combined_UMI.csv", header = F);

```

```

cor = cor(mat[,2],mat[,3]);
frame=ggplot(mat,aes(mat[,2],mat[,3]))
frame + geom_point(colour="black",size=1,shape=20,alpha=0.3) + labs(x="15
cycles",y="25 cycles",title="with UMI") + annotate("text", label =paste
("R=", round (cor,6), sep=""), x = min(mat[,2])+10, y = max(mat[,3]) /
10, size = 6, colour = "red") +scale_x_log10() + scale_y_log10()+theme(
axis.text=element_text(size=12), axis.title=element_text(size=14,face="
bold"),title=element_text(size=18,face="bold"));
ggsave("15_25_combined_UMI.pdf")

```

Chapter 5

Table 1: Comparison to Illumina's bcl2fastq.

To run this benchmark some understanding on how to toggle the de-multiplexing options in the CASAVA pipeline are required. A great tutorial on this was written by Brant Faircloth¹.

It is necessary to install the bcl2fastq package² (bcl2fastq-1.8.4.tar.gz) and follow these steps:

5.1 Run the default pipeline on the validation data containing barcodes:

1) modify:

By default only one PhiX sample and one human sample are used for the validation. Modify:

```
./share/bcl2fastq-1.8.4/examples/Validation/110120_P20_0993_A805CKABXX/  
Data/Intensities/BaseCalls/SampleSheet.csv
```

NOTE: there are two validation datasets included but only 110120_P20_0993_A805CKABXX contains multiplexed samples.

so that all samples of lane 1 are de-multiplexed.

2) run:

```
bash-3.2$ configureValidation.pl \  
--source-dir ../share/bcl2fastq-1.8.4/examples/Validation/110120  
_P20_0993_A805CKABXX \  
--output-dir ./ValidationMultiplexed_casava
```

3) run base-caller with *default* de-multiplexing:

```
bash-3.2$ cd ValidationMultiplexed_casava  
bash-3.2$ make
```

At this point we should have the CASAVA de-multiplexed fastq files.

¹<https://gist.github.com/brantfaircloth/3125885>

²http://support.illumina.com/downloads/bcl2fastq_conversion_software_184.html

5.2 Run the pipeline without de-multiplexing and use TagDust2.

1) modify configuration makefile:

share/bcl2fastq-1.8.4/examples/Validation/110120_P20_0993_A805CKABXX/Unaligned/ValidationConfig.mk
from:

```
CONFIGURE_BCL_TO_FASTQ_PARAMS := --use-bases-mask 'y76n*,I6n,y76n*'
```

to:

```
CONFIGURE_BCL_TO_FASTQ_PARAMS := --use-bases-mask 'y76n*,Y6n,y76n*'
```

I.e. read two corresponding to the 6 nucleotide index is basecalled and written to a separate file alongside the actual paired-end reads.

2) modify sample sheet:

/share/bcl2fastq-1.8.4/examples/Validation/110120_P20_0993_A805CKABXX/Data/Intensities/BaseCalls/SampleSheet.csv
to:

```
FCID,Lane,SampleID,SampleRef,Index,Description,Control,Recipe,Operator
A805CKABXX,1,not_demultiplexed,,,Test bcl conversion,N,D109LACXX,BCF,
    testbclconv
#A805CKABXX,1,AR005,human,ACAGTG,Cypress,Y,101+7,CB,Demo
#A805CKABXX,1,AR008,human,ACTTGA,Cypress,Y,101+7,CB,Demo
#A805CKABXX,1,PhiX,phix,TTAGGC,Cypress,Y,101+7,CB,Demo
#A805CKABXX,1,,unknown,Undetermined,Ignored clusters with unmatched
    barcodes for lane 1,N,101+7,CB,
```

Note that only lane 1 is included in the test.

3) run the pipeline again

```
bash-3.2$ configureValidation.pl \
    --source-dir ../share/bcl2fastq-1.8.4/examples/Validation/110120
    _P20_0993_A805CKABXX \
    --output-dir ./ValidationMultiplexed_casava
```

Now we should have these three fastq files:

```
not_demultiplexed_NoIndex_L001_R1_001.fastq.gz
not_demultiplexed_NoIndex_L001_R2_001.fastq.gz
not_demultiplexed_NoIndex_L001_R3_001.fastq.gz
```

5.3 Run TagDust2

Here is the TagDust2 command:

```
bash-3.2$ tagdust -arch casava_arch.txt
    not_demultiplexed_NoIndex_L001_R1_001.fastq.gz
    not_demultiplexed_NoIndex_L001_R2_001.fastq.gz
    not_demultiplexed_NoIndex_L001_R3_001.fastq.gz -t 80 -o demulti_large
```

Here are the contents of the architecture file `casava_arch.txt`:

```
tagdust -1 B:ACAGTG,ACTTGA,TTAGGC
tagdust -1 R:N
```

The first line corresponds to the architecture we wish to use on read number two containing the indices. The second line corresponds to *blank* architecture which results in the whole read to be extracted by TagDust2.

TagDust2 will:

- 1) scan each input file separately with both architectures
- 2) determine that file:
 - one is a read (R:N)
 - two is contains indices (B:ACAGTG,ACTTGA,TTAGGC)
 - three is another read (R:N)
- 3) scan all reads with the *correct* architectures
- 4) extract reads in which the index could be unambiguously identified.

5.4 Comparison

To count the number of extracted reads (CASAVA):

```
bash-3.2$ zcat Sample_AR005/AR005_ACAGTG_L001_R1_001.fastq.gz | wc -l
5890936

bash-3.2$ zcat Sample_AR008/AR008_ACTTGA_L001_R1_001.fastq.gz | wc -l
6072832

bash-3.2$ zcat Sample_PhiX/PhiX_TTAGGC_L001_R1_001.fastq.gz | wc -l
196416
```

and the number of reads extracted by TagDust2:

```
bash-3.2$ wc -l demulti_large_*READ1.fq
6035392 demulti_large_BC_ACAGTG_READ1.fq
6175672 demulti_large_BC_ACTTGA_READ1.fq
205952 demulti_large_BC_TTAGGC_READ1.fq
```

NOTE: the numbers above must be divided by 4 because each sequence corresponds to 4 lines in a fastq file.

To get an idea whether the reads additionally extracted by TagDust2 are of poor quality or not I aligned all reads to the human genome (hg38) / PhiX genome (NC 001422.1) using BWA-MEM³:

```
bash-3.2$ nice bwa mem -t 16 /GROUP/REFERENCE/GRCh38/GRCh38.genome.fa
demulti_large_BC_ACAGTG_READ1.fq demulti_large_BC_ACAGTG_READ2.fq |
samtools view -Sb - > tagdust_ACAGTG.bam

bash-3.2$ nice bwa mem -t 16 /GROUP/REFERENCE/GRCh38/GRCh38.genome.fa
demulti_large_BC_ACTTGA_READ1.fq demulti_large_BC_ACTTGA_READ2.fq |
samtools view -Sb - > tagdust_ACTTGA.bam
```

³Li, H.: Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. arXiv preprint arXiv:1303.3997 (2013)

```

bash-3.2$ nice bwa mem -t 16 phix.fa demulti_large_BC_TTAGGC_READ1.fq
demulti_large_BC_TTAGGC_READ2.fq | samtools view -Sb - >
tagdust_TTAGGC.bam

bash-3.2$ nice bwa mem -t 16 /GROUP/REFERENCE/GRCh38/GRCh38.genome.fa
Sample_AR005/AR005_ACAGTG_L001_R1_001.fastq.gz Sample_AR005/
AR005_ACAGTG_L001_R2_001.fastq.gz | samtools view -Sb - > casava_ACAGTG
.bam

bash-3.2$ nice bwa mem -t 16 /GROUP/REFERENCE/GRCh38/GRCh38.genome.fa
Sample_AR008/AR008_ACTTGA_L001_R1_001.fastq.gz Sample_AR008/
AR008_ACTTGA_L001_R2_001.fastq.gz | samtools view -Sb - > casava_ACTTGA
.bam

bash-3.2$ nice bwa mem -t 16 phix.fa Sample_PhiX/PhiX_TTAGGC_L001_R1_001.
fastq.gz Sample_PhiX/PhiX_TTAGGC_L001_R2_001.fastq.gz | samtools view
-Sb - > casava_TTAGGC.bam

```

To count the number of correctly mapped and paired reads (excluding secondary and supplementary alignments):

```

bash-3.2$ samtools view -c -f 1 -F 2316 casava_ACAGTG.bam
bash-3.2$ samtools view -c -f 1 -F 2316 casava_ACTTGA.bam
bash-3.2$ samtools view -c -f 1 -F 2316 casava_TTAGGC.bam
bash-3.2$ samtools view -c -f 1 -F 2316 tagdust_ACAGTG.bam
bash-3.2$ samtools view -c -f 1 -F 2316 tagdust_ACTTGA.bam
bash-3.2$ samtools view -c -f 1 -F 2316 tagdust_TTAGGC.bam

```

Chapter 6

Table 2: Summary of Single Cell extracted reads.

Here we test how TagDust2 can handle datasets with a large number of indices / barcodes. We downloaded datafiles from ¹:

- Run00020_L3_1_110110_GA2X_00020_FC.fq.bz2
- Run00021_L3_1_110114_GA2X_00021_FC.fq.bz2
- Run00020_L5_1_110110_GA2X_00020_FC.fq.bz2
- Run00021_L6_1_110114_GA2X_00021_FC.fq.bz2
- Run00021_L1_1_110114_GA2X_00021_FC.fq.bz2

To extract the reads we use the following read architecture file:

```
bash-3.2$ cat arch.txt
tagdust -1 O:N -2 B:TTTAGG,ATTCCA,GCTCAA,CATCCC,TTGGAC,CTGTGT,GGACAT,
CAAAGT,AAGCGG,AATAAA,GAGGAG,GGTACA,AGCGAG,GTGCGT,ATTTCG,AGGACT,GCCCTC,
TCGTAA,CCAGAC,TATGTA,ACAATA,ATGCTT,AGTTTA,CACAAG,ATCAAC,TAGTCG,TAGAGA,
GTCCCG,TACTTC,AAAGTT,TAAGGG,GTGCGC,AAGTAC,GATCTT,TTAACT,GCGAAT,CCGCTA,
TGAAGC,ATACAG,CTTCTG,GAGATC,CCGACG,CTCCAT,AAAACG,TAGCAT,TCGGGT,GTGGTA,
CCTAGA,GGGTTT,ATGGCG,TTTATA,AACGCC,GGCTGC,GCTGTG,AGATGG,GTAATG,AGGGTC,
ATCTCT,GCCTAG,TCAAAG,CATGAT,TGTGCG,GCAGGA,TCTACC,AGTCGT,CGTGGC,GCGTCC,
GAACGC,ACTTAT,TGGATG,TATTGT,ACGTTG,GAATTA,CCATCT,TGATCA,CGTATT,CGGCAG,
GACACT,TTCCGC,CTCGCA,GTATAC,TGTAC,TCGCGA,ACGAGC,ACACCC,CGCTTG,TGCAAT,
CAACAA,CTGAAA,AACCTA,ACCTGA,TCACTT,GGGCGA,CGCACC,CGAGTA,CCTTTC -3 S:GGG
-4 R:N -t 80
```

Here are the TagDust commands:

```
bash-3.2$ tagdust Run00020_L3_1_110110_GA2X_00020_FC.fq -arch arch.txt -t
80 -o Run00020_L3_1_110110_GA2X_00020_FC_extracted.fq
bash-3.2$ tagdust Run00020_L5_1_110110_GA2X_00020_FC.fq -arch arch.txt -t
80 -o Run00020_L5_1_110110_GA2X_00020_FC_extracted.fq
bash-3.2$ tagdust Run00021_L1_1_110114_GA2X_00021_FC.fq -arch arch.txt -t
80 -o Run00021_L1_1_110114_GA2X_00021_FC_extracted.fq
bash-3.2$ tagdust Run00021_L3_1_110114_GA2X_00021_FC.fq -arch arch.txt -t
80 -o Run00021_L3_1_110114_GA2X_00021_FC_extracted.fq
bash-3.2$ tagdust Run00021_L6_1_110114_GA2X_00021_FC.fq -arch arch.txt -t
80 -o Run00021_L6_1_110114_GA2X_00021_FC_extracted.fq
```

¹Islam, S., Kjällquist, U., Moliner, A., Zajac, P., Fan, J.-B., Lönnerberg, P., Linnarsson, S.: Characterization of the single-cell transcriptional landscape by highly multiplex rna-seq. *Genome research* 21(7), 1160–1167 (2011)

All resulting reads were aligned to the mouse genome (GRCm38.p2) using a RNAseq pipeline. The latter is included in the TagDust package (`./reproducibility/rnaseq_pipeline_4tagdust_paper-0.8.tar.gz`).

Chapter 7

Additional files:

7.1 File: Figure1.tex:

```
1
2 \documentclass{article}
3 \usepackage[margin=0.2cm, left=0.0cm, paperwidth=18.5cm, paperheight=17cm
  ]{geometry}
4 \usepackage{tikz}
5 \usetikzlibrary{shapes}
6 \usetikzlibrary{calc, backgrounds}
7 \usetikzlibrary{automata}
8 \usetikzlibrary{arrows, positioning, calc, matrix}
9 \usetikzlibrary{decorations.pathreplacing, shapes.multipart}
10
11 \tikzstyle{Dstate}=[shape=circle, draw=black!50, fill=black!10]
12 \tikzstyle{Istate}=[shape=diamond, draw=black!50, fill=black!10]
13 \tikzstyle{Mstate}=[shape=rectangle, draw=black!50, fill=black!10]
14
15 \tikzstyle{empty}=[shape=circle]
16
17 \tikzstyle{lightedge}=[->, dotted, thick]
18 \tikzstyle{mainstate}=[state, thick]
19 \tikzstyle{mainedge}=[->, thick]
20
21 %\renewcommand{\chaptername}{}
22 %\renewcommand{\thechapter}{}
23
24 \definecolor{black}{RGB}{0,0,0}
25 \definecolor{darkgrey}{RGB}{64,64,64}
26 \definecolor{grey}{RGB}{127,127,127}
27 \definecolor{lightgrey}{RGB}{230,230,230}
28
29 % scheme 1
30 \definecolor{winered}{RGB}{158,16,0}
31 \definecolor{lightred}{RGB}{199,53,42}
32 \definecolor{brown}{RGB}{158,95,0}
33 \definecolor{orange}{RGB}{235,141,0}
34
35 %scheme2
36
37 \definecolor{darkblue}{RGB}{19,48,182}
```

```

38
39 \definecolor{blue}{RGB}{36,89,158}
40 \definecolor{browngreen}{RGB}{82,75,19}
41 \definecolor{lightbrown}{RGB}{158,126,36}
42
43 %scheme3
44
45 \definecolor{green}{RGB}{52,132,23}
46
47
48 \definecolor{lightgreen}{RGB}{82,209,36}
49 \definecolor{lightbrowngreen}{RGB}{125,133,23}
50 \definecolor{yellowgreen}{RGB}{198,209,36}
51
52 %scheme4
53
54
55 \definecolor{paper}{RGB}{240,238,183}
56 \definecolor{organicgrey}{RGB}{163,162,124}
57
58
59 \definecolor{metal}{RGB}{75,81,82}
60 \begin{document}
61 \begin{tikzpicture}[]
62   \tiny
63
64
65   \node[shape=circle,draw=black,thick,font =\Large] at (-3,6.5) {\bf 1};
66   \node[shape=circle,draw=black,thick,font =\Large] at (-3,0) {\bf 2};
67   \node[shape=circle,draw=black,thick,font =\Large] at (-3,-6) {\bf 3};
68   \node[shape=circle,draw=black,thick,font =\Large] at (-3,-8) {\bf 4};
69
70   \node[style={align=center},draw = black, thick,font=\large] at (6,6.5)
71     { \tt tagdust2 -1 0:N -2 B:GTA,AAC -3 R:N -4 L:CCTTAA reads.fq} ;
72
73   \path[shade,top color=gray!5,bottom color=blue!10!white] (2.2,6.2)--
74     (3.2,6.2) -- (0.8,4.6) -- (-0.7,4.6) -- cycle;
75
76   \path[shade,top color=gray!5,bottom color=blue!10!white] (4,6.2)--
77     (6,6.2) -- (4.5,4.6) -- (1.6,4.6) -- cycle;
78
79   \path[shade,top color=gray!5,bottom color=blue!10!white] (6.8,6.2)--
80     (7.5,6.2) -- (6.7,4.6) -- (5.3,4.6) -- cycle;
81
82   \path[shade,top color=gray!5,bottom color=blue!10!white] (8.3,6.2)--
83     (10.2,6.2) -- (13.4,4.6) -- (7.5,4.6) -- cycle;
84
85   \draw[
86     -triangle 90,
87     line width=2mm,
88     postaction={draw, line width=0.5cm, shorten >=0.5cm, -}
89   ] (6,6) -- node[right=1,font =\Large] {\bf HMM construction} (6,5) ;

```

```

88
89
90
91
92 \path[shade,top color=gray!5,bottom color=blue!10!white] (-0.65,-6.25)--
    (12.25,-6.25) -- (12.5,-8.15) -- (-0.5,-8.25) -- (-0.65,-6.25);
93
94 \path[top color=winered!80!white,bottom color=winered!30!white]
    (-1.75,-6.25)-- (-0.65,-6.25) -- (6,-7.7) -- (2.5,-7.7) --
    (-1.75,-6.25);
95
96 \draw[rounded corners=4pt,draw=lightgrey, fill=lightgrey] (-0.75,4.5)
    rectangle (0.75,-4);
97
98 \draw[rounded corners=4pt,draw=lightred, fill=lightred] (1.5,4.5)
    rectangle (4.5,-4);
99
100
101 \draw[rounded corners=4pt,draw=blue, fill=blue] (5.25,4.5) rectangle
    (6.75,-4);
102
103 \draw[rounded corners=4pt,draw=lightgrey, fill=lightgrey] (7.5,4.5)
    rectangle (13.5,-4);
104
105 \node at (0,4) [font=\Large,style={align=center}] () {\bf Opt}};
106
107 \node at (3,4) [font=\Large,style={align=center}] () {\bf Barcode}};
108
109 % \node at (3,2+1.75) [font=\Large,style={align=center}] () {\bf 1}};
110
111 % \node at (3,-2+ 1.75) [font=\Large,style={align=center}] () {\bf
    2}};
112 \node at (6,4) [font=\Large,style={align=center}] () {\bf Read}};
113
114 \node at (10.5,4) [font=\Large,style={align=center}] () {\bf Linker
    }};
115
116 \draw[
117     -triangle 90,
118     line width=2mm,
119     postaction={draw, line width=0.5cm, shorten >=0.5cm, -}
120 ] (6,-4.5) -- node[right=1,font =\Large] {\bf Decoding}} (6,-5.5) ;
121
122 \node[draw=white, thick,font=\Large] at (6,-6) {\bf {\color{lightgrey}C
    }}{\color{lightred}GTA}}{\color{blue}GGGGAACCCCGCCTGTTTACCAAAAACATCA}}{\color{lightgrey}CCTTA}} ;
123
124 \draw[
125     -triangle 90,
126     line width=2mm,
127     postaction={draw, line width=0.5cm, shorten >=0.5cm, -}
128 ] (6,-6.5) -- node[right=1,font =\Large] {\bf Output}} (6,-7.5) ;
129
130 \node[style={align=left},text width=27em, thick,font=\Large] at (2.8,-8)
    {\bf $>$READ X, {\color{lightred}Barcode GTA} } ;

```

```

131 \node[style={align=left},text width=20em,draw=white, thick,font=\Large]
    at (2,-8.5) {\bf {\color{blue}GGGGAACCCCGCCTGTTTACCAAAAACATCA}}
    ;
132
133
134 \begin{scope}[shift={(-1.5,0)}];
135 \node[Dstate] (START) at (0,0){$$};
136 \end{scope}
137
138 \begin{scope}[shift={(0,1.5)}];
139 \node[Mstate] (01) at (0,0){\shortstack{ A\\ C\\ G\\ T } };
140 \end{scope}
141
142 \begin{scope}[shift={(2,2)}];
143 %\node[Dstate] (d1) at (0,1){D$};
144 \node[Istate] (i1) at (0,0){I$};
145 \node[Mstate] (m1) at (0,-1.2){\shortstack{ {\color{black!40}A}\\ {\color{black!40}C}\\ {\bf G}\\ {\color{black!40}T} } };
146
147
148
149 \node[Dstate] (d2) at (1,1){D$};
150 \node[Istate] (i2) at (1,0){I$};
151 \node[Mstate] (m2) at (1,-1.2){\shortstack{ {\color{black!40}A}\\ {\color{black!40}C}\\ {\color{black!40}G}\\ {\bf T} } };
152
153
154 %\node[Dstate] (d3) at (2,1){D$};
155 %\node[Istate] (i3) at (2,-0){I$};
156 \node[Mstate] (m3) at (2,-1.2){\shortstack{ {\bf A}\\ {\color{black!40}C}\\ {\color{black!40}G}\\ {\color{black!40}T} } };
157 \end{scope}
158
159
160 \begin{scope}[shift={(2,-2)}];
161
162 %\node[Dstate] (d4) at (0,1){D$};
163 \node[Istate] (i4) at (0,0){I$};
164 \node[Mstate] (m4) at (0,-1.2){\shortstack{ {\bf A}\\ {\color{black!40}C}\\ {\color{black!40}G}\\ {\color{black!40}T} } };
165
166
167
168 \node[Dstate] (d5) at (1,1){D$};
169 \node[Istate] (i5) at (1,0){I$};
170 \node[Mstate] (m5) at (1,-1.2){\shortstack{ {\bf A}\\ {\color{black!40}C}\\ {\color{black!40}G}\\ {\color{black!40}T} } };
171
172
173 %\node[Dstate] (d6) at (2,1){D$};
174 %\node[Istate] (i6) at (2,-0){I$};
175 \node[Mstate] (m6) at (2,-1.2){\shortstack{ {\color{black!40}A}\\ {\bf C}\\ {\color{black!40}G}\\ {\color{black!40}T} } };
176 \end{scope}
177

```

```

178
179
180 \begin{scope}[shift={(6,0)}];
181
182
183
184 \node[Mstate] (R1) at (0,0){\shortstack{ A\\ C \\ G \\ T} };
185 \end{scope}
186
187
188
189 \begin{scope}[shift={(8,2)}];
190
191 %CCTTAAGG
192 %\node[Dstate] (dl1) at (0,1){$D$};
193 \node[Istate] (il1) at (0,0){$I$};
194 \node[Mstate] (ml1) at (0,-1.2){\shortstack{ {\color{black!40}A}\\ {\color{black!40}G}\\ {\color{black!40}T}} };
195
196
197
198 \node[Dstate] (dl2) at (1,1){$D$};
199 \node[Istate] (il2) at (1,0){$I$};
200 \node[Mstate] (ml2) at (1,-1.2){\shortstack{ {\color{black!40}A}\\ C\\ {\color{black!40}G}\\ {\color{black!40}T}} };
201
202
203 \node[Dstate] (dl3) at (2,1){$D$};
204 \node[Istate] (il3) at (2,-0){$I$};
205 \node[Mstate] (ml3) at (2,-1.2){\shortstack{ {\color{black!40}A}\\ {\color{black!40}C}\\ {\color{black!40}G}\\ {\color{black!40}T}} };
206
207 \node[Dstate] (dl4) at (3,1){$D$};
208 \node[Istate] (il4) at (3,0){$I$};
209 \node[Mstate] (ml4) at (3,-1.2){\shortstack{ {\color{black!40}A}\\ {\color{black!40}C}\\ {\color{black!40}G}\\ {\color{black!40}T}} };
210
211 \node[Dstate] (dl5) at (4,1){$D$};
212 \node[Istate] (il5) at (4,0){$I$};
213 \node[Mstate] (ml5) at (4,-1.2){\shortstack{ {\bf A}\\ C\\ {\color{black!40}G}\\ {\color{black!40}T}} };
214 %\node[Dstate] (dl6) at (5,1){$D$};
215 %\node[Istate] (il6) at (5,-0){$I$};
216 \node[Mstate] (ml6) at (5,-1.2){\shortstack{ {\bf A } \\ !40}C}\\ {\color{black!40}G}\\ {\color{black!40}T}} };
217 \end{scope}
218
219 \begin{scope}[shift={(14,-2)}];
220 \node[Dstate] (END) at (0,0){$E$};
221 \end{scope}
222
223 \path
224
225 (START) edge [lightedge] (O1)
226

```

```

227 (START) edge [lightedge] (m1)
228
229 (START) edge [lightedge] (m4)
230
231 (O1) edge [lightedge, loop above] ()
232 (R1) edge [lightedge, loop above] ()
233
234 (O1)edge [lightedge] (m1)
235 (O1)edge [lightedge] (m4)
236 (R1)edge [lightedge] (m11)
237
238 (m11.south)edge [lightedge] (END)
239 (m12.south)edge [lightedge] (END)
240 (m13.south)edge [lightedge] (END)
241 (m14.south)edge [lightedge] (END)
242 (m15.south)edge [lightedge] (END)
243 (m16.south)edge [lightedge] (END)
244 (R1) edge [lightedge](END)
245
246 (m3)edge [lightedge] (R1)
247 (m6)edge [lightedge] (R1)
248
249 (m1) edge [lightedge] (m2)
250 (m1) edge [lightedge] (d2)
251 (m1) edge [lightedge] (i1)
252
253 (i1) edge [lightedge,loop above] ()
254 (i1) edge [lightedge] (m2)
255 (i1) edge [lightedge, loop above] ()
256 %(d1) edge [lightedge,->] (d2)
257 %(d1) edge [lightedge,->] (m2)
258
259 (m2) edge [lightedge] (m3)
260 %(m2) edge [lightedge] (d3)
261 (m2) edge [lightedge] (i2)
262
263 (i2) edge [lightedge,loop above] ()
264 (i2) edge [lightedge] (m3)
265 (i2) edge [lightedge, loop above] ()
266 %(d2) edge [lightedge,->] (d3)
267 (d2) edge [lightedge,->] (m3)
268
269
270 %(m3) edge [lightedge] (i3)
271
272 %(i3) edge [lightedge,loop above] ()
273
274
275
276 (m4) edge [lightedge] (m5)
277 (m4) edge [lightedge] (d5)
278 (m4) edge [lightedge] (i4)
279
280 (i4) edge [lightedge,loop above] ()
281 (i4) edge [lightedge] (m5)

```

```

282 (i4) edge [lightedge, loop above] ()
283 %(d4) edge [lightedge,->] (d5)
284 %(d4) edge [lightedge,->] (m5)
285
286 (m5) edge [lightedge] (m6)
287 %(m5) edge [lightedge] (d6)
288 (m5) edge [lightedge] (i5)
289
290 (i5) edge [lightedge, loop above] ()
291 (i5) edge [lightedge] (m6)
292 (i5) edge [lightedge, loop above] ()
293 %(d5) edge [lightedge,->] (d6)
294 (d5) edge [lightedge,->] (m6)
295
296
297 %(m6) edge [lightedge] (i6)
298
299 %(i6) edge [lightedge, loop above] ()
300
301
302
303 (m11) edge [lightedge] (m12)
304 (m11) edge [lightedge] (d12)
305 (m11) edge [lightedge] (i11)
306
307 (i11) edge [lightedge, loop above] ()
308 (i11) edge [lightedge] (m12)
309 (i11) edge [lightedge, loop above] ()
310 %(d11) edge [lightedge,->] (d12)
311 %(d11) edge [lightedge,->] (m12)
312
313 (m12) edge [lightedge] (m13)
314 (m12) edge [lightedge] (d13)
315 (m12) edge [lightedge] (i12)
316
317 (i12) edge [lightedge, loop above] ()
318 (i12) edge [lightedge] (m13)
319 (i12) edge [lightedge, loop above] ()
320 (d12) edge [lightedge,->] (d13)
321 (d12) edge [lightedge,->] (m13)
322
323
324 (m13) edge [lightedge] (m14)
325 (m13) edge [lightedge] (d14)
326 (m13) edge [lightedge] (i13)
327
328 (i13) edge [lightedge, loop above] ()
329 (i13) edge [lightedge] (m14)
330 (i13) edge [lightedge, loop above] ()
331 (d13) edge [lightedge,->] (d14)
332 (d13) edge [lightedge,->] (m14)
333
334
335 (m14) edge [lightedge] (m15)
336 (m14) edge [lightedge] (d15)

```

```

337 (m14) edge [lightedge] (i14)
338
339 (i14) edge [lightedge,loop above] ()
340 (i14) edge [lightedge] (m15)
341 (i14) edge [lightedge, loop above] ()
342 (d14) edge [lightedge,->] (d15)
343 (d14) edge [lightedge,->] (m15)
344
345
346
347 (m15) edge [lightedge] (m16)
348 %(m15) edge [lightedge] (d16)
349 (m15) edge [lightedge] (i15)
350
351 (i15) edge [lightedge,loop above] ()
352 (i15) edge [lightedge] (m16)
353 (i15) edge [lightedge, loop above] ()
354 %(d15) edge [lightedge,->] (d16)
355 (d15) edge [lightedge,->] (m16)
356 ;
357
358 \end{tikzpicture}
359
360 \end{document}

```