R implementation

Sleiman Bassim, PhD

September 23, 2015

Loaded functions:

```r
#source("/media/Data/Dropbox/humanR/01funcs.R")
rm(list=ls())
#setwd("/media/Data/Dropbox/humanR/PD/")
#setwd("~/Dropbox/humanR/PD/")
###load("PD.Rdata", .GlobalEnv)
#lsos(pat="")
```

Load packages.

```r
pkgs <- c('xlsx','caret','leaps','glmnet','lattice',
          'latticeExtra', 'dplyr', 'tidyr')
lapply(pkgs, require, character.only = TRUE)
```

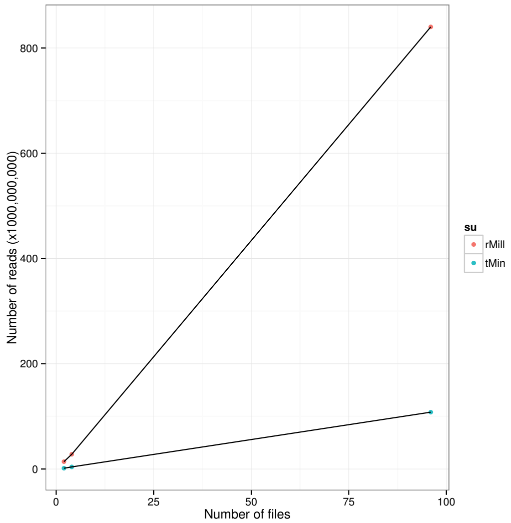
1 XSEDE benchmarking

Files are labeled either with BR for gills and GG for ganglia. These are the two tissues used in this project. There is 48 files for each GG and BR. R1 and R2 files denote reverse reads and forward reads. There is 24 R1 files and 24 R2 files for GG. The same applies for BR. The first 12 R1 and R2 in GG or BR are for starved oysters. The rest is for normally fed oysters. These are the conditions used in this project.

1.1 Quality control checks

Quality controls were done separately for each R1 and R2 samples.

```r
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 1)
dat <- gather(dat, su, count, 2:3)
dat

  file mMB  vMB core node    su count
1    2 175 1500   16    1 rMill  14.0
2    4 175 1500   16    1 rMill  28.0
3   96 201 1650   16    1 rMill 840.0
4    2 175 1500   16    1  tMin   1.5
5    4 175 1500   16    1  tMin   4.0
6   96 201 1650   16    1  tMin 108.0

ggplot(dat,
       aes(x = file,
           y = count,
           fill = su)) +
    geom_point(aes(color= su)) +
    geom_line(data = dat) +
    theme_bw() +
    labs(x = "Number of files",
         y = "Number of reads (x1000,000,000)")
```



1.2 Trimming data

Trimming can be done automatically in trinity. But trimming was also tested outside of trinity with trimmomatic. The tests show trinity is faster by two hours per sample.

```
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 2)
dat

  file rMill tMin  mMB   vMB core node
1    2    14    6 4300 20000   16    1
```

1.3 Counting reads

Half the samples were counted. Below is the time it takes to count R1 labeled files from GG samples.

```
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 3)
dat

  file rMill tMin mMB vMB core node
1   48   420   20   5 500   16    1
```

1.4 Merging samples

First, R1 and R2 files are always merged separately. Second, all GG and BR files are merged in a single fastq file. Third, all GG or BR files are merged in two separate fastq files.

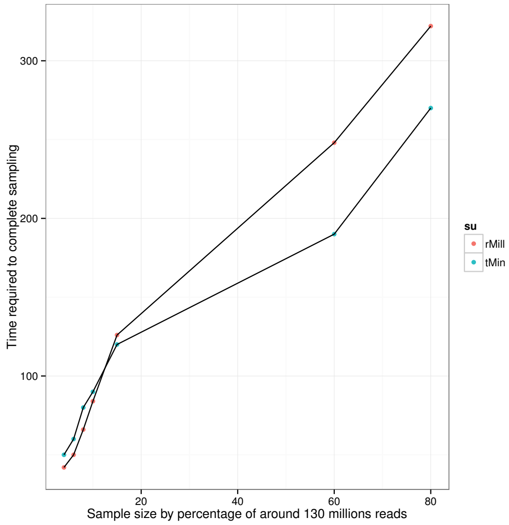
```
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 4)
dat

  file rMill tMin mMB vMB core node merge
1   96   840  666   4 300   16    1   all
2   48   414  240   4  42   16    1    GG
3   48   428  270   4  42   16    1    BR
```

1.5 Sampling

Randomly sampling 80% and 60% of reads is done only on merged GG and BR fastq files. The file that contains both GG and BR is not sampled. Sampling jobs at 80% failed when running on GORDON normal (native) cluster. These jobs are now running on GORDON virtual memory (Vsmp).

```
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 5)
```

```
# first row is the failed GORDON test of 80% sampling
# normal settings
dat <- dat[-1, ]
dat

  file percentage rMill tMin   mMB   vGB core node    pbs
2   48         80   322  270 85000 85000  256    1   vsmp
3   48         60   248  190 64000 64000   16    1 native
4   48         15   126  120 31000 31000   16    1 native
5   48         10    84   90 31000 21000   16    1 native
6   48          8    66   80 17000 17000   16    1 native
7   48          6    50   60 12000 12000   16    1 native
8   48          4    42   50 10000 10000   16    1 native

dat <- gather(dat, su, count, 3:4)
ggplot(dat,
       aes(x = percentage,
           y = count,
           fill = su)) +
    theme_bw() +
    geom_point(aes(color = su)) +
    geom_line(data = dat) +
    labs(x = 'Sample size by percentage of around 130 millions reads',
         y = 'Time required to complete sampling')
```



1.6 File size

The size of each file is relative to its state either being compressed gzip or flat. The difference between compressed and flat is 4 folds. All R1 BR and GG have 110 GB before compression and the size is reduced to 33 GB after compression. It will take almost 1 hour to decompress this amount of data. So it is best to keep a flat version of each file to speed up server jobs and avoid the 48 walltime termination on jobs that exceed this limit. At this stage, that is after sampling GG and BR at 80% and 60% and merging the corresponding files the total sum of disk size occupied by the flat files is almost 500 GB.

Table 1: Disk size

File	Size one file R1	Total (R1+R2)+(BR+GG)
60% reads	20 GB	80 GB
80% reads	24 GB	94 GB
100% reads	30 GB	115 GB
All reads	57 GB	115 GB
Reads by sample*	1 GB	115 GB

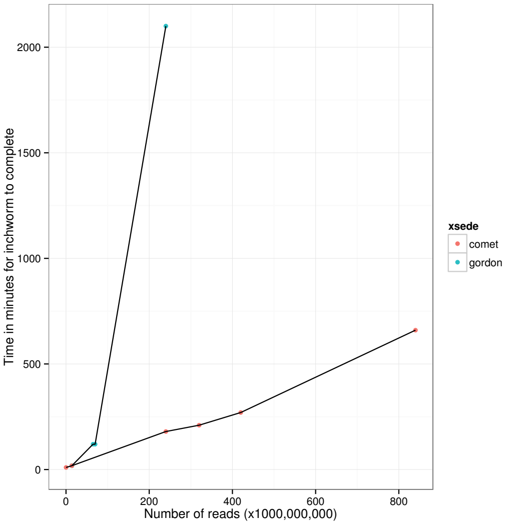
*Raw files generated by the sequencing platform separated by biological sample, condition, and tissue.

1.7 Butterfly: Final phase in transcriptome assembly

Butterfly is the final phase of running trinity on raw reads sequencing data. On a single sample, which includes one R1 file and one R2 file, butterfly can complete 50% of the analysis in 2 hours with 1 node and 64 cores at 900 GB. However, with little over 240 million reads, butterfly completes only 3% of the mapping of contigs in 1 hour with 1 node and 64 cores at 900 GB.

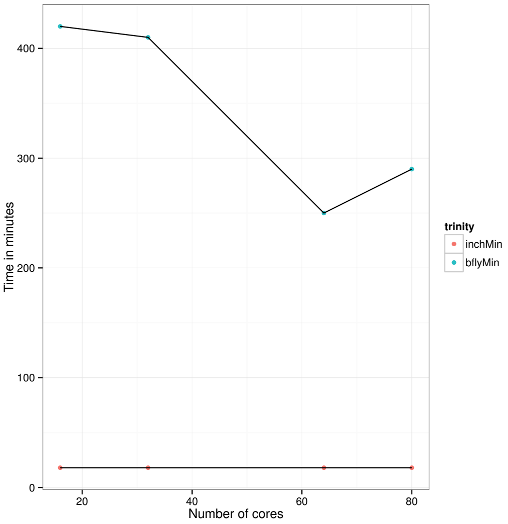
↰ GG 60% was used here for testing butterfly

```
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 6)
dat

   node score file inchMin bflyMin mGB maskedMin rMill   task  xsede
1     1    16    2      18     420  11        60  14.0   test gordon
2     2    32    2      18     410  11        60  14.0   test gordon
3     5    80    2      18     290  48       120  14.0   test gordon
4     1    64    2      18     250 900      1000  14.0   test  comet
5     1    64    2      10       0 900      1000   0.1 sample  comet
6     1    64   48     210       0 900      1000 320.0    80p  comet
7     1    64   48     180       0 900      1000 240.0    60p  comet
8     1    64   48     270       0 900      1000 420.0   100p  comet
9     1    64   96     660       0 900      1000 840.0    all  comet
10    1   256   48    2100       0 486       900 240.0   vsmp gordon
11    1   256   10     120       0 363       900  70.0   vsmp gordon
12    1   256   10     120       0 363       900  65.0   vsmp gordon

ggplot(dat,
       aes(x = rMill,
           y = inchMin,
           fill = xsede)) +
    geom_point(aes(color = xsede)) +
    geom_line(data = dat) +
    theme_bw() +
    labs(x= 'Number of reads (x1000,000,000)',
         y = 'Time in minutes for inchworm to complete')
```



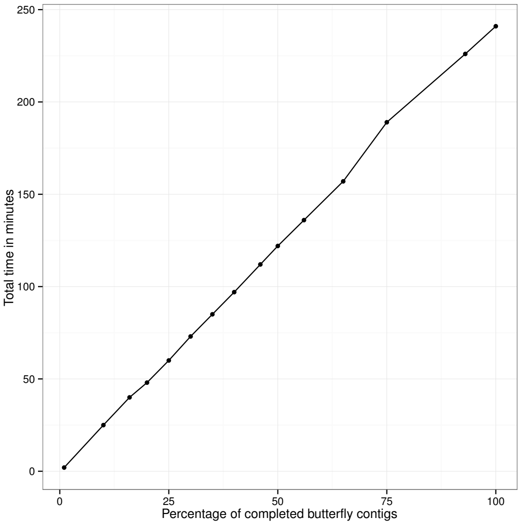
Completion time for trinity, all phases, on 1 sample, that is one R1 and one R2.

```
dat <- dat[1:4, ]
dat <- gather(dat, trinity, count, 4:5)
ggplot(dat,
       aes(x = score,
           y = count,
           fill = trinity)) +
    geom_point(aes(color = trinity)) +
    geom_line(data = dat) +
    theme_bw() +
    labs(x = "Number of cores",
         y = "Time in minutes")
```



Percentage of completion of butterfly.

```
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 8)
ggplot(dat,
       aes(x = completed,
           y = tMin)) +
    theme_bw() +
    geom_point(data = dat) +
    geom_line(data = dat) +
    labs(x = 'Percentage of completed butterfly contigs',
         y = 'Total time in minutes')
```

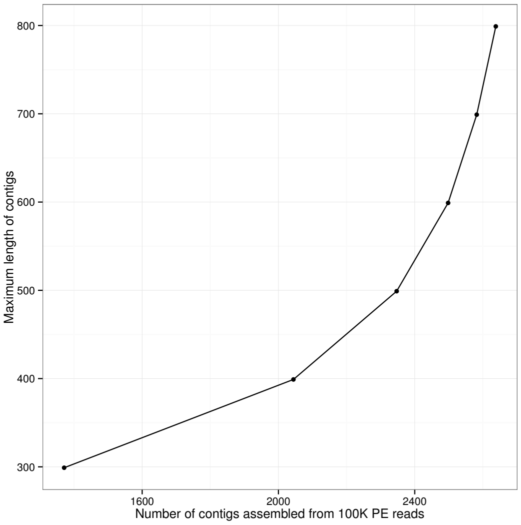


1.8 Assembly length of contigs

What is the size of the assembled contigs if we randomly sample 100 K reads from each R1 and R2 of one sample? From a total of 200 K reads (R1 and R2) we can assemble 2842 contigs at a full size of 1,160,388 base.

↰ GG 11 was used for sampling

```
dat <- read.xlsx("./data/xsede.xlsx", sheetIndex = 9)
ggplot(dat,
       aes(x = count,
           y = length)) +
    theme_bw() +
    geom_point(data = dat) +
    geom_line(data = dat) +
    labs(x = "Number of contigs assembled from 100K PE reads",
         y = "Maximum length of contigs")
```



2 System Information

The version number of R and packages loaded for generating the vignette were:

```
###save(list=ls(pattern=".*|.*"),file="PD.Rdata")
sessionInfo()

R version 3.2.1 (2015-06-18)
Platform: x86_64-unknown-linux-gnu (64-bit)
Running under: elementary OS Luna

locale:
 [1] LC_CTYPE=en_US.UTF-8          LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8           LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8       LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8          LC_NAME=en_US.UTF-8
 [9] LC_ADDRESS=en_US.UTF-8        LC_TELEPHONE=en_US.UTF-8
[11] LC_MEASUREMENT=en_US.UTF-8    LC_IDENTIFICATION=en_US.UTF-8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
 [1] tidyr_0.2.0          dplyr_0.4.2          latticeExtra_0.6-26
 [4] RColorBrewer_1.1-2   glmnet_2.0-2         foreach_1.4.2
 [7] Matrix_1.2-1         leaps_2.9            caret_6.0-47
[10] ggplot2_1.0.1        lattice_0.20-31      xlsx_0.5.7
[13] xlsxjars_0.6.1       rJava_0.9-6          knitr_1.10.5
[16] RevoUtilsMath_3.2.1

loaded via a namespace (and not attached):
 [1] Rcpp_0.11.6          compiler_3.2.1       formatR_1.2
 [4] nloptr_1.0.4         plyr_1.8.3           highr_0.5
 [7] iterators_1.0.7      tools_3.2.1          digest_0.6.8
[10] lme4_1.1-8           evaluate_0.7         nlme_3.1-121
[13] gtable_0.1.2         mgcv_1.8-6           DBI_0.3.1
[16] parallel_3.2.1       brglm_0.5-9          SparseM_1.6
[19] proto_0.3-10         BradleyTerry2_1.0-6 stringr_1.0.0
[22] gtools_3.5.0         grid_3.2.1           nnet_7.3-10
[25] R6_2.0.1             minqa_1.2.4          reshape2_1.4.1
[28] car_2.0-25           magrittr_1.5         scales_0.2.5
[31] codetools_0.2-11     MASS_7.3-41          splines_3.2.1
[34] assertthat_0.1       pbkrtest_0.4-2       colorspace_1.2-6
[37] labeling_0.3         quantreg_5.11        stringi_0.5-5
[40] lazyeval_0.1.10      munsell_0.4.2
```