LaF benchmarks

D.J. van der Laan

2011-11-06

1 Introduction

LaF is a package for R for working with large ASCII files in R. The manual vignette contains an discription of the functionality provided. In this vignette the performance of the LaF package is compared to that of the built in R routines for reading comma separated files (`read.table`) and fixed width files (`read.fwf`).

2 The test files

In total four files are generated. Two in fixed width format and two in comma separated format. For the comma seperated file the following ten lines are repeated until the required amount of lines are obtained:

```
1,M,1.45,Rotterdam
2,F,12.00,Amsterdam
3,,.22,Berlin
8,,.24,Berlin
,M,22,Paris
10,F,54321,London
4,F,12345,London
5,M,,Copenhagen
6,M,-12.1,
7,F,-1,Oslo
```

For the fixed width file, the following ten lines are repeated until the required amount of lines are obtained:

```
 1M 1.45Rotterdam
 2F12.00Amsterdam
 3  .22 Berlin
 8  .24 Berlin
  M22   Paris
10F54321London
```

```
4F12345London
5M     Copenhagen
6M-12.1
7F   -1Oslo
```

For each of the two formats two files are generated. One small one that is used when reading the complete dataset into memory and a large one that is used for all other operation. The small files consist of 100 000 rows the large files of 10 000 000 rows.

3 Reading complete files

In the following tests a complete file of 100 000 rows is read completely into memory.

3.1 Fixed width

3.1.1 LaF

```
> system.time({
+     laf <- laf_open_fwf(filename = filesmallfwf,
+         column_types = c("integer", "categorical",
+             "double", "string"), column_widths = c(2,
+             1, 5, 10))
+     tst <- laf[, ]
+ })

   user  system elapsed
  0.264   0.008   0.272
```

3.1.2 read.fwf

```
> system.time({
+     tst <- read.fwf(file = filesmallfwf,
+         widths = c(2, 1, 5, 10), comment.char = "",
+         colClasses = c("integer", "factor",
+             "numeric", "character"))
+ })

   user  system elapsed
  2.536   0.864   3.709
```

3.2 Separated

3.2.1 LaF

```
> system.time({
+     laf <- laf_open_csv(filename = filesmallcsv,
+         column_types = c("integer", "categorical",
+             "double", "string"))
+     tst <- laf[, ]
+ })

   user  system elapsed
  0.188   0.004   0.188
```

3.2.2 read.table

```
> system.time({
+     tst <- read.table(file = filesmallcsv,
+         sep = ",", comment.char = "", quote = "",
+         colClasses = c("integer", "factor",
+             "numeric", "character"))
+ })

   user  system elapsed
  0.132   0.000   0.132
```

4 Blockwise processing

Blockwise processing of files (reading and processing files in blocks or chunks that fit into memory). In the following tests the sum of the third column in the file is calculated using blockwise processing.

4.1 Fixed width

4.1.1 LaF

```
> calc_sum_laf <- function(block, result) {
+     if (is.null(result))
+         result <- 0
+     result + sum(block$V3, na.rm = TRUE)
+ }
> system.time({
+     laf <- laf_open_fwf(filename = filelargefwf,
+         column_types = c("integer", "categorical",
+             "double", "string"), column_widths = c(2,
+             1, 5, 10))
```

```
+     sm <- process_blocks(laf, calc_sum_laf)
+ })

   user  system elapsed
  8.693   0.048   8.747
```

The previous code can be made faster by using the `columns` argument of `process_blocks`:

```
> system.time({
+     sm <- process_blocks(laf, calc_sum_laf,
+         columns = 3)
+ })

   user  system elapsed
  1.748   0.036   1.782
```

An other option is to first read the complete column into memory (if that fits) and then work with the column in memory:

```
> system.time({
+     sm <- mean(laf[, 3], na.rm = TRUE)
+ })

   user  system elapsed
  1.804   0.400   2.206
```

4.1.2 `read.fwf`

The following code shows how a file can be processed in blocks using `read.table` and `read.fwf`. First, a connection to the file is made and opened. When, `read.table` is passed an open connection, it starts reading the specified number of lines (`n`) and does not close the connection after reading. The `try` block is needed in case the previous call to `read.table` stopped reading exactly at the end of the file. Checking for the end-of-file is unfortunately not possible in R (as far as I know). An other solution would be to use a combination of `readLines` and `read.table`. However, this was found to be much slower. Therefore, the solution below was choosen. It is used in most examples with `read.table` and `read.fwf` in the following sections.

```
> calc_sum_r_fwf <- function(filename) {
+     result <- 0
+     con <- file(filename, "rt")
+     while (TRUE) {
+         block <- data.frame()
+         try({
```

```
+             block <- read.fwf(file = con,
+                 n = 5000, widths = c(2, 1,
+                   5, 10), comment.char = "",
+                 colClasses = c("NULL", "NULL",
+                   "numeric", "NULL"))
+             result <- result + sum(block[,
+                 1], na.rm = TRUE)
+         })
+         if (nrow(block) < 5000)
+             break
+     }
+     close(con)
+     return(result)
+ }
> system.time({
+     sm <- calc_sum_r_fwf(filelargefwf)
+ })
```

```
   user  system elapsed
247.683  91.957 339.809
```

4.2 Separated

4.2.1 LaF

```
> system.time({
+     laf <- laf_open_csv(filename = filelargecsv,
+         column_types = c("integer", "categorical",
+             "double", "string"))
+     sm <- process_blocks(laf, calc_sum_laf)
+ })
```

```
   user  system elapsed
  8.609   0.048   8.664
```

```
> system.time({
+     sm <- process_blocks(laf, calc_sum_laf,
+         columns = 3)
+ })
```

```
   user  system elapsed
  2.224   0.064   2.285
```

4.2.2 read.table

```
> calc_sum_r_csv <- function(filename) {
+     result <- 0
```

```
+     con <- file(filename, "rt")
+     while (TRUE) {
+         block <- data.frame()
+         try({
+             block <- read.table(file = con,
+                 sep = ",", nrows = 5000,
+                 comment.char = "", quote = "",
+                 colClasses = c("NULL", "NULL",
+                   "numeric", "NULL"))
+             result <- result + sum(block[,
+                 1], na.rm = TRUE)
+         })
+         if (nrow(block) < 5000)
+             break
+     }
+     close(con)
+     return(result)
+ }
> system.time({
+     sm <- calc_sum_r_csv(filelargecsv)
+ })
```

```
   user  system elapsed
 10.125   0.036  10.160
```

5 Reading subset

In the tests below all data belonging to the municipality of 'Rotterdam' is read.

5.1 Fixed width

5.1.1 LaF

```
> system.time({
+     laf <- laf_open_fwf(filename = filelargefwf,
+         column_types = c("integer", "categorical",
+             "double", "string"), column_widths = c(2,
+             1, 5, 10))
+     d <- laf[laf$V4[] == "Rotterdam ", ]
+     print(nrow(d))
+ })
```

```
[1] 1000000
   user  system elapsed
```

```
  4.184   2.897   7.089
```

5.1.2 read.fwf

```
> system.time({
+     d <- data.frame()
+     con <- file(filelargefwf, "rt")
+     while (TRUE) {
+         block <- data.frame()
+         try({
+             block <- read.fwf(file = con,
+                 n = 5000, widths = c(2, 1,
+                   5, 10), comment.char = "",
+                 colClasses = c("integer",
+                   "factor", "numeric", "character"))
+             d <- rbind(d, block[block[, 4] ==
+                 "Rotterdam ", ])
+         })
+         if (nrow(block) < 5000)
+             break
+     }
+     close(con)
+     print(nrow(d))
+ })
```

```
[1] 1000000
    user   system  elapsed
2099.623   92.869 2193.452
```

The example above takes a very long time. One of the reasons is that we have a growing `data.frame` which is slow and memory inefficient. A faster solution would be to first allocate the `data.frame` before reading. Unfortunately, the end size of the `data.frame` is usually not known beforehand. One could first calculate the end size using code similar to that used in section 4, or one could guess the size. As an optimal example, using the usually unknown end size of `d`, the following result is obtained (we will use `d` from the previous example):

```
> system.time({
+     con <- file(filelargefwf, "rt")
+     i <- 1
+     while (TRUE) {
+         block <- data.frame()
+         try({
+             block <- read.fwf(file = con,
```

```
+                 n = 5000, widths = c(2, 1,
+                   5, 10), comment.char = "",
+                 colClasses = c("integer",
+                   "factor", "numeric", "character"))
+             sel <- block[, 4] == "Rotterdam "
+             d[seq_len(sum(sel)) + i - 1,
+                 ] <- block[sel, ]
+             i <- i + sum(sel)
+         })
+         if (nrow(block) < 5000)
+             break
+     }
+     close(con)
+     print(nrow(d))
+ })

[1] 1000000
   user  system elapsed
619.803  87.858 708.058
```

5.2 Separated

5.2.1 LaF

```
> system.time({
+     laf <- laf_open_csv(filename = filelargecsv,
+         column_types = c("integer", "categorical",
+             "double", "string"))
+     d <- laf[laf$V4[] == "Rotterdam", ]
+     print(nrow(d))
+ })

[1] 1000000
   user  system elapsed
  5.308   0.448   5.758
```

5.2.2 read.table

```
> system.time({
+     d <- data.frame()
+     con <- file(filelargecsv, "rt")
+     while (TRUE) {
+         block <- data.frame()
+         try({
+             block <- read.table(file = con,
```

```
+                 sep = ",", nrows = 5000,
+                 comment.char = "", quote = "",
+                 colClasses = c("integer",
+                   "factor", "numeric", "character"))
+             d <- rbind(d, block[block[, 4] ==
+                 "Rotterdam", ])
+         })
+         if (nrow(block) < 5000)
+             break
+     }
+     close(con)
+     print(nrow(d))
+ })
```

```
[1] 1000000
    user   system  elapsed
1699.907    2.800 1703.378
```

Using the usually unknown end size of `d` (see the discussion for `read.fwf`):

```
> system.time({
+     con <- file(filelargecsv, "rt")
+     i <- 1
+     while (TRUE) {
+         block <- data.frame()
+         try({
+             block <- read.table(file = con,
+                 sep = ",", nrows = 5000,
+                 comment.char = "", quote = "",
+                 colClasses = c("integer",
+                   "factor", "numeric", "character"))
+             sel <- block[, 4] == "Rotterdam"
+             d[seq_len(sum(sel)) + i - 1,
+                 ] <- block[sel, ]
+             i <- i + sum(sel)
+         })
+         if (nrow(block) < 5000)
+             break
+     }
+     close(con)
+     print(nrow(d))
+ })
```

```
[1] 1000000
   user  system elapsed
337.953   0.028 338.151
```