Of course, I’m paraphrasing Dirk’s fifteenth post in the *rarely rational R rambling* series: [#15: Tidyverse and data.table, sitting side by side … (Part 1)](http://dirk.eddelbuettel.com/blog/2018/01/21/#015_tidyverse_and_datatable_part_1). I very much liked it, because, although I’m a happy tidyverse user, I’m always trying not to be tied into that *verse* too much by replicating certain tasks with other tools (and languages) as an exercise. In this article, I’m going to repeat Dirk’s exercise in base R.

First of all, I would like to clean up the tidyverse version a little, because the original was distributed in chunks and was a little bit too verbose. We can also avoid using lubridate, because readr already parses the end\_date column as a date (and that’s why it is significantly slower, among other reasons). This is how I would do it:

## Getting the polls

library(tidyverse)

library(zoo)

polls\_2016 <- read\_tsv(url("http://elections.huffingtonpost.com/pollster/api/v2/questions/16-US-Pres-GE%20TrumpvClinton/poll-responses-clean.tsv"))

## Wrangling the polls

polls\_2016 <- polls\_2016 %>%

filter(sample\_subpopulation %in% c("Adults","Likely Voters","Registered Voters")) %>%

right\_join(data.frame(end\_date = seq.Date(min(.$end\_date), max(.$end\_date), by="days")), by="end\_date")

## Average the polls

rolling\_average <- polls\_2016 %>%

group\_by(end\_date) %>%

summarise(Clinton = mean(Clinton), Trump = mean(Trump)) %>%

mutate(Clinton.Margin = Clinton-Trump,

Clinton.Avg = rollapply(Clinton.Margin,width=14,

FUN=function(x){mean(x, na.rm=TRUE)},

by=1, partial=TRUE, fill=NA, align="right"))

ggplot(rolling\_average) +

geom\_line(aes(x=end\_date, y=Clinton.Avg), col="blue") +

geom\_point(aes(x=end\_date, y=Clinton.Margin))

which, by the way, has exactly the very same number of lines of code than the data.table version:

## Getting the polls

library(data.table)

library(zoo)

library(ggplot2)

pollsDT <- fread("http://elections.huffingtonpost.com/pollster/api/v2/questions/16-US-Pres-GE%20TrumpvClinton/poll-responses-clean.tsv")

## Wrangling the polls

pollsDT <- pollsDT[sample\_subpopulation %in% c("Adults","Likely Voters","Registered Voters"), ]

pollsDT[, end\_date := as.IDate(end\_date)]

pollsDT <- pollsDT[ data.table(end\_date = seq(min(pollsDT[,end\_date]),

max(pollsDT[,end\_date]), by="days")), on="end\_date"]

## Average the polls

pollsDT <- pollsDT[, .(Clinton=mean(Clinton), Trump=mean(Trump)), by=end\_date]

pollsDT[, Clinton.Margin := Clinton-Trump]

pollsDT[, Clinton.Avg := rollapply(Clinton.Margin, width=14,

FUN=function(x){mean(x, na.rm=TRUE)},

by=1, partial=TRUE, fill=NA, align="right")]

ggplot(pollsDT) +

geom\_line(aes(x=end\_date, y=Clinton.Avg), col="blue") +

geom\_point(aes(x=end\_date, y=Clinton.Margin))

Let’s translate this into base R. It is easier to start from the data.table version, mainly because filtering and assigning have a similar look and feel. Unsurprisingly, we have base::merge for the merge operation and stats::aggregate for the aggregation phase. base::as.Date works just fine for these dates and utils::read.csv has the only drawback that you have to specify the separator. Without further ado, this is my version in base R:

## Getting the polls

library(zoo)

pollsB <- read.csv(url("http://elections.huffingtonpost.com/pollster/api/v2/questions/16-US-Pres-GE%20TrumpvClinton/poll-responses-clean.tsv"), sep="\t")

## Wrangling the polls

pollsB <- pollsB[pollsB$sample\_subpopulation %in% c("Adults","Likely Voters","Registered Voters"), ]

pollsB$end\_date <- base::as.Date(pollsB$end\_date)

endDate <- data.frame(end\_date = seq.Date(min(pollsB$end\_date), max(pollsB$end\_date), by="days"))

pollsB <- merge(pollsB, endDate, by="end\_date", all=TRUE)

## Average the polls

pollsB <- aggregate(cbind(Clinton, Trump) ~ end\_date, data=pollsB, mean, na.action=na.pass)

pollsB$Clinton.Margin <- pollsB$Clinton - pollsB$Trump

pollsB$Clinton.Avg <- rollapply(pollsB$Clinton.Margin, width=14,

FUN=function(x){mean(x, na.rm=TRUE)},

by=1, partial=TRUE, fill=NA, align="right")

plot(pollsB$end\_date, pollsB$Clinton.Margin, pch=16)

lines(pollsB$end\_date, pollsB$Clinton.Avg, col="blue", lwd=2)

which is the shortest one! Finally, let’s repeat the benchmark too:

library(microbenchmark)

url <- "http://elections.huffingtonpost.com/pollster/api/v2/questions/16-US-Pres-GE%20TrumpvClinton/poll-responses-clean.tsv"

file <- "/tmp/poll-responses-clean.tsv"

download.file(url, destfile=file, quiet=TRUE)

res <- microbenchmark(tidy=suppressMessages(readr::read\_tsv(file)),

dt=data.table::fread(file, showProgress=FALSE),

base=read.csv(file, sep="\t"))

res

## Unit: milliseconds

## expr min lq mean median uq max neval

## tidy 13.877036 15.127885 18.549393 15.861311 17.813541 202.389391 100

## dt 4.084022 4.505943 5.152799 4.845193 5.652579 7.736563 100

## base 29.029366 30.437742 32.518009 31.449916 33.600937 45.104599 100

Base R is clearly the slowest option for the reading phase. Or, one might say, both readr and data.table have done a great job in improving things! Let’s take a look at the processing part now:

tvin <- suppressMessages(readr::read\_tsv(file))

dtin <- data.table::fread(file, showProgress=FALSE)

bsin <- read.csv(file, sep="\t")

library(tidyverse)

library(data.table)

library(zoo)

transformTV <- function(polls\_2016) {

polls\_2016 <- polls\_2016 %>%

filter(sample\_subpopulation %in% c("Adults","Likely Voters","Registered Voters")) %>%

right\_join(data.frame(end\_date = seq.Date(min(.$end\_date), max(.$end\_date), by="days")), by="end\_date")

rolling\_average <- polls\_2016 %>%

group\_by(end\_date) %>%

summarise(Clinton = mean(Clinton), Trump = mean(Trump)) %>%

mutate(Clinton.Margin = Clinton-Trump,

Clinton.Avg = rollapply(Clinton.Margin,width=14,

FUN=function(x){mean(x, na.rm=TRUE)},

by=1, partial=TRUE, fill=NA, align="right"))

}

transformDT <- function(dtin) {

pollsDT <- copy(dtin) ## extra work to protect from reference semantics for benchmark

pollsDT <- pollsDT[sample\_subpopulation %in% c("Adults","Likely Voters","Registered Voters"), ]

pollsDT[, end\_date := as.IDate(end\_date)]

pollsDT <- pollsDT[ data.table(end\_date = seq(min(pollsDT[,end\_date]),

max(pollsDT[,end\_date]), by="days")), on="end\_date"]

pollsDT <- pollsDT[, .(Clinton=mean(Clinton), Trump=mean(Trump)), by=end\_date]

pollsDT[, Clinton.Margin := Clinton-Trump]

pollsDT[, Clinton.Avg := rollapply(Clinton.Margin, width=14,

FUN=function(x){mean(x, na.rm=TRUE)},

by=1, partial=TRUE, fill=NA, align="right")]

}

transformBS <- function(pollsB) {

pollsB <- pollsB[pollsB$sample\_subpopulation %in% c("Adults","Likely Voters","Registered Voters"), ]

pollsB$end\_date <- base::as.Date(pollsB$end\_date)

endDate <- data.frame(end\_date = seq.Date(min(pollsB$end\_date), max(pollsB$end\_date), by="days"))

pollsB <- merge(pollsB, endDate, by="end\_date", all=TRUE)

pollsB <- aggregate(cbind(Clinton, Trump) ~ end\_date, data=pollsB, mean, na.action=na.pass)

pollsB$Clinton.Margin <- pollsB$Clinton - pollsB$Trump

pollsB$Clinton.Avg <- rollapply(pollsB$Clinton.Margin, width=14,

FUN=function(x){mean(x, na.rm=TRUE)},

by=1, partial=TRUE, fill=NA, align="right")

}

res <- microbenchmark(tidy=transformTV(tvin),

dt=transformDT(dtin),

base=transformBS(bsin))

res

## Unit: milliseconds

## expr min lq mean median uq max neval

## tidy 20.68435 22.58603 26.67459 24.56170 27.85844 84.55077 100

## dt 17.25547 18.88340 21.43256 20.24450 22.26448 41.65252 100

## base 28.39796 30.93722 34.94262 32.97987 34.98222 109.14005 100