The R Package **sentometrics** to Compute, Aggregate and Predict with Textual Sentiment
Supplementary Appendix

January 2020

David Ardia
Keven Bluteau
Samuel Borms[*]
Kris Boudt

Efficiency of various lexicon–based sentiment analysis tools in R

This appendix provides an illustrative comparison of the computation time of various lexicon–based sentiment analysis tools in R. The core of the sentiment computation in the R package **sentometrics** (Ardia *et al.* 2020) is implemented in C++ through **Rcpp** (Eddelbuettel and Francois 2011). We compare the speed of our computation with the R packages **meanr** (Schmidt 2019), **SentimentAnalysis** (Feuerriegel and Pröllochs 2019), **syuzhet** (Jockers 2017), **quanteda** (Benoit *et al.* 2018), and **tidytext** (Silge and Robinson 2016). The first three of these five packages have proper sentiment functions. The **quanteda** and **tidytext** packages have no explicit sentiment computation function; it needs to be constructed first, based on their respective toolsets. This is an entry barrier for less–experienced programmers. The **SentimentAnalysis** package has the **tm** package as backend and uses internally a similar calculation as **tm**'s `tm_term_score()` function. The **sentimentr** package is not part of the exercise because it proved to be vastly slower than all others, which was anticipated as it aims to handle more difficult linguistic edge cases.

We perform two analyses. Sentiment is computed for 1000, 5000, 10000, 25000, 50000, 75000 and 100000 texts, and the average execution time in seconds across five repetitions, using the **microbenchmark** (Mersmann and Ulrich 2019) package, is shown in Table 1.

The first analysis (see Panel 1a) benchmarks these implementations with three approaches using the `compute_sentiment()` function from **sentometrics**: one without valence shifters, one with valence shifters integrated from a bigrams perspective, and one with valence shifters integrated from a clusters perspective. The number of threads for parallel computation is set to one where appropriate. All other algorithms are run with a version of the the Hu & Liu lexicon (about 6600 single words). The computations are counts–based and constructed so as to give the same output across all packages for a binary lexicon, if the tokenization is the same. For example, the **sentometrics** and **tidytext** implementations give identical results.

The **meanr** implementation comes out fastest because everything is written in the C programming language. Yet, it offers no flexibility to define the input lexicon nor the scale on which the scores are returned. On the other spectrum, amongst these approaches, the **SentimentAnalysis** and **suyzhet** packages are slowest. The latter package further does not offer the flexibility of adding different sentiment lexicons than those available in their package. **SentimentAnalysis** becomes exponentially

[*] Corresponding author, contact details: `borms_sam@hotmail.com`.

(a) Average execution time of the sentiment computation for one lexicon

	sentometrics							
Texts	*unigrams*	*bigrams*	*clusters*	**meanr**	**SentimentAnalysis**	**syuzhet**	**quanteda**	**tidytext**
1000	0.24	0.20	0.22	0.08	1.18	0.55	0.60	0.16
5000	0.87	0.87	0.91	0.34	5.26	1.99	1.74	0.60
10000	1.73	1.68	1.72	0.67	11.23	3.83	3.07	1.11
25000	4.41	4.21	4.40	1.71	26.88	9.07	7.19	2.83
50000	9.18	8.55	9.42	3.75	53.08	18.37	14.12	5.88
75000	13.62	13.49	13.44	5.06	78.44	27.13	20.37	8.48
100000	18.69	18.22	18.61	6.57	109.58	35.25	26.98	11.06

(b) Average execution time of the sentiment computation for nine lexicons

	sentometrics					**tidytext**	
Texts	*unigrams*	*unigrams*, feats.	*bigrams*	*clusters*	*clusters*, parallel	*unigrams*	*bigrams*
1000	0.26	0.24	0.27	0.26	0.22	0.21	0.66
5000	1.00	0.87	1.01	1.01	0.79	0.67	2.80
10000	1.96	1.68	1.98	1.97	1.54	1.27	5.68
25000	4.82	4.24	4.90	4.97	3.81	3.07	13.95
50000	9.96	8.71	10.13	10.02	7.85	6.03	28.00
75000	16.70	19.14	16.67	23.04	15.43	14.00	58.02
100000	32.40	23.66	23.80	36.41	30.86	14.02	64.73

Table 1: Average computation time (in seconds) of various lexicon–based sentiment tools in R. All implementations consider the Hu & Liu lexicon (Panel 1a), or the nine lexicons specified in the `lex` object defined in the main text of the vignette (Panel 1b). Some implementations do not integrate valence shifters (*unigrams*), others do from a bigrams perspective (*bigrams*) or from a clusters perspective (*clusters*).

slower as it suffers to manage the memory required for larger corpora. The **quanteda** package is fast, but slower than the **sentometrics** and **tidytext** implementations. The **tidytext** package is faster, particularly for the two largest corpus sizes.

The second analysis (see Panel 1b) compares the computation time with nine lexicons as input. The comparison is against the **tidytext** package, for a unigrams and a bigrams implementation. The lexicons are those in the `lex` object defined in the main text of the vignette. For the clusters approach, we also look at its parallelized version, using eight cores (see the '*clusters*, parallel' column). For the unigrams approach in **sentometrics**, we also assess the additional time it takes to spread out sentiment across features (see the '*unigrams*, feats.' column).

The **tidytext** package is, in general, faster for many lexicons as well. Differences are not large nonetheless, and running any **sentometrics** computation in parallel would make the speed differentials disappear. However, the bigrams calculation using **tidytext** is markedly slower. With **sentometrics**, the speed of the computation is comparable across all types of sentiment calculation. The **tidytext** framework thus copes more slowly with complexity.

Overall, the **sentometrics** package brings an off–the–shelf yet flexible sentiment calculator that is computationally efficient, being fast in itself, and independent as to the decision (how) to integrate valence shifters as well as (though to a smaller extent) the number of input lexicons.

Computational details

For the main computational details, we refer to the paper. The timings comparison can be replicated using the R script `run_timings.R`, available on the **sentometrics** GitHub repository (`https://github.com/sborms/sentometrics`) in the `appendix` folder. To generate the results, we have also used the packages **dplyr** version 0.8.3 (Wickham *et al.* 2019), **meanr** version 0.1.2 (Schmidt 2019), **microbenchmark** version 1.4.7 (Mersmann and Ulrich 2019), **SentimentAnalysis** version 1.3.3 (Feuerriegel and Pröllochs 2019), **syuzhet** version 1.0.4 (Jockers 2017), **tidytext** version 0.2.2 (Silge and Robinson 2016), and **tidyr** version 1.0.0 (Wickham and Henry 2019).

References

Ardia D, Bluteau K, Borms S, Boudt K (2020). "The R Package **sentometrics** to Compute, Aggregate and Predict with Textual Sentiment." doi:10.2139/ssrn.3067734. Working paper.

Benoit K, Watanabe K, Wang H, Nulty P, Obeng A, Müller S, Matsuo A (2018). "**quanteda**: An R Package for the Quantitative Analysis of Textual Data." *Journal of Open Source Software*, **3**(30), 774. doi:10.21105/joss.00774.

Eddelbuettel D, Francois R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Feuerriegel S, Pröllochs N (2019). **SentimentAnalysis***: Dictionary–Based Sentiment Analysis.* R Package Version 1.3.3, URL `https://CRAN.R-project.org/package=SentimentAnalysis`.

Jockers M (2017). **syuzhet***: Extract Sentiment and Plot Arcs from Text.* R Package Version 1.0.4, URL `https://CRAN.R-project.org/package=syuzhet`.

Mersmann O, Ulrich J (2019). **microbenchmark***: Accurate Timing Functions.* R Package Version 1.4.7, URL `https://CRAN.R-project.org/package=microbenchmark`.

Schmidt D (2019). **meanr***: Sentiment Analysis Scorer.* R Package Version 0.1.2, URL `https://CRAN.R-project.org/package=meanr`.

Silge J, Robinson D (2016). "**tidytext**: Text Mining and Analysis Using Tidy Data Principles in R." *Journal of Open Source Software*, **1**(3). doi:10.21105/joss.00037.

Wickham H, François R, Henry L, Müller K (2019). **dplyr***: A Grammar of Data Manipulation.* R package version 0.8.3, URL `https://CRAN.R-project.org/package=dplyr`.

Wickham H, Henry L (2019). **tidyr***: Tidy Messy Data.* R package version 1.0.0, URL `https://CRAN.R-project.org/package=tidyr`.