An integrated framework in R for textual sentiment time series aggregation and prediction

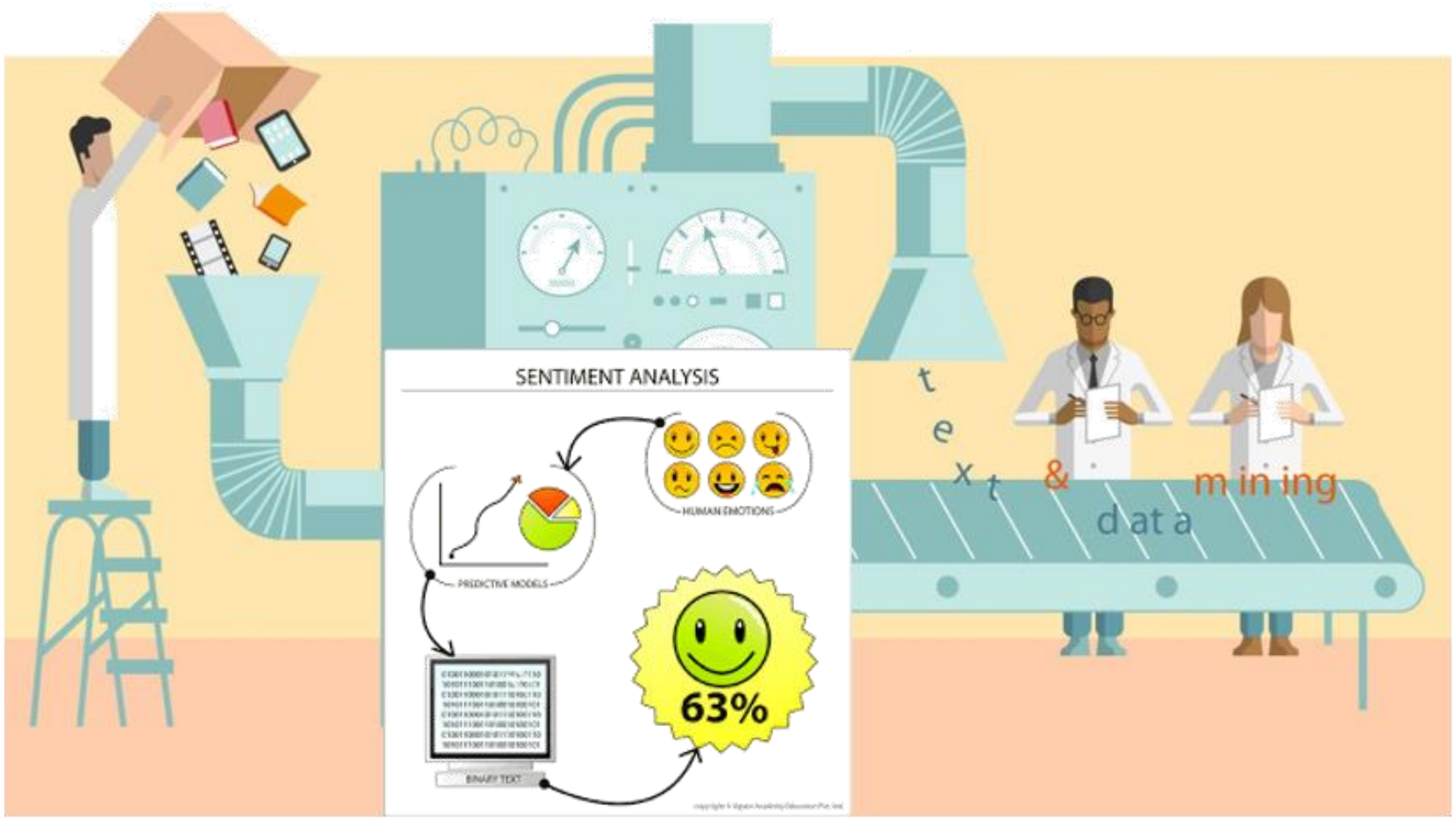
Ardia, D. , Bluteau, K., Borms, S. and Boudt, K. (2017). “The R Package *sentometrics* to Compute, Aggregate and Predict with Textual Sentiment”. Available at SSRN: http://dx.doi.org/10.2139/ssrn.3067734.

‘sentometrics’ repository: https://github.com/sborms/sentometrics.

Project website: https://www.sentometrics.com.

Text mining...

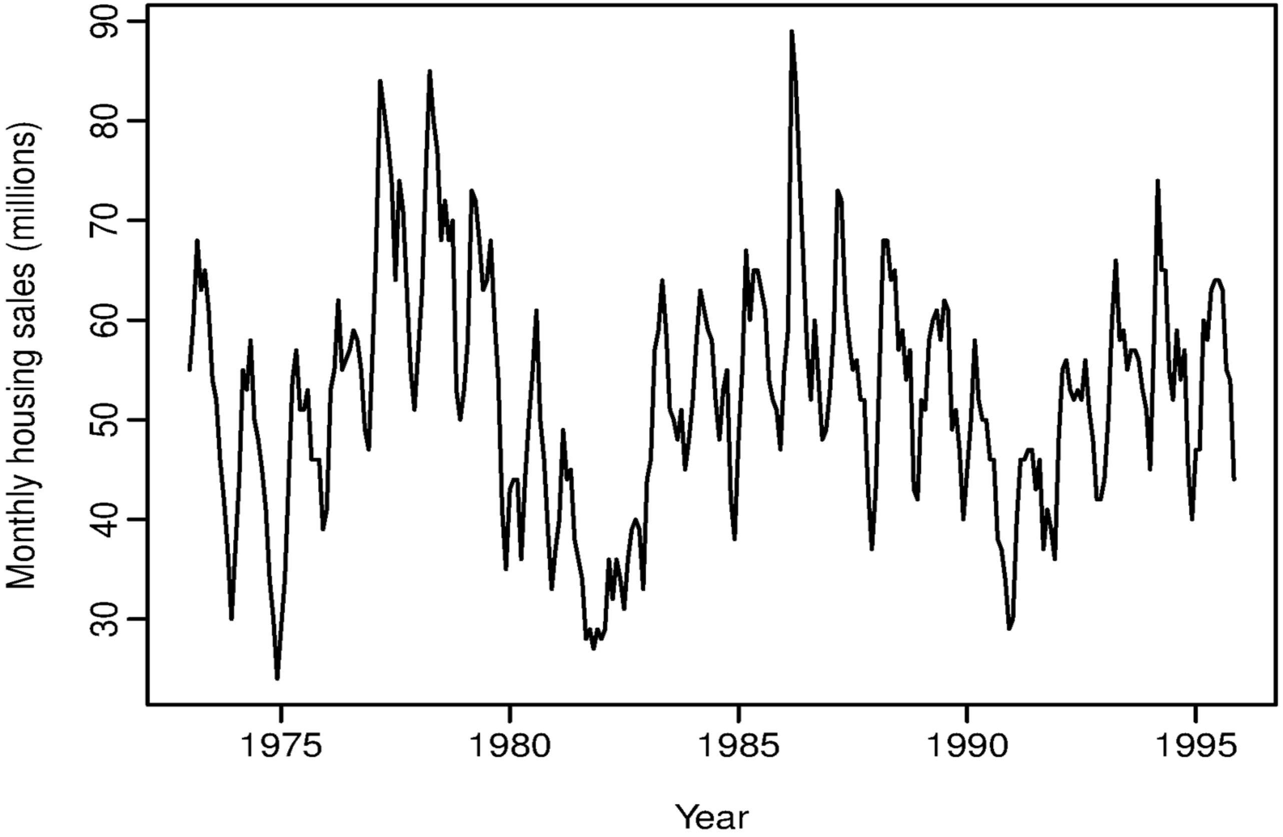
... is the process of distilling actionable insights from text.



Our focus is on textual sentiment analysis.

Time series econometrics...

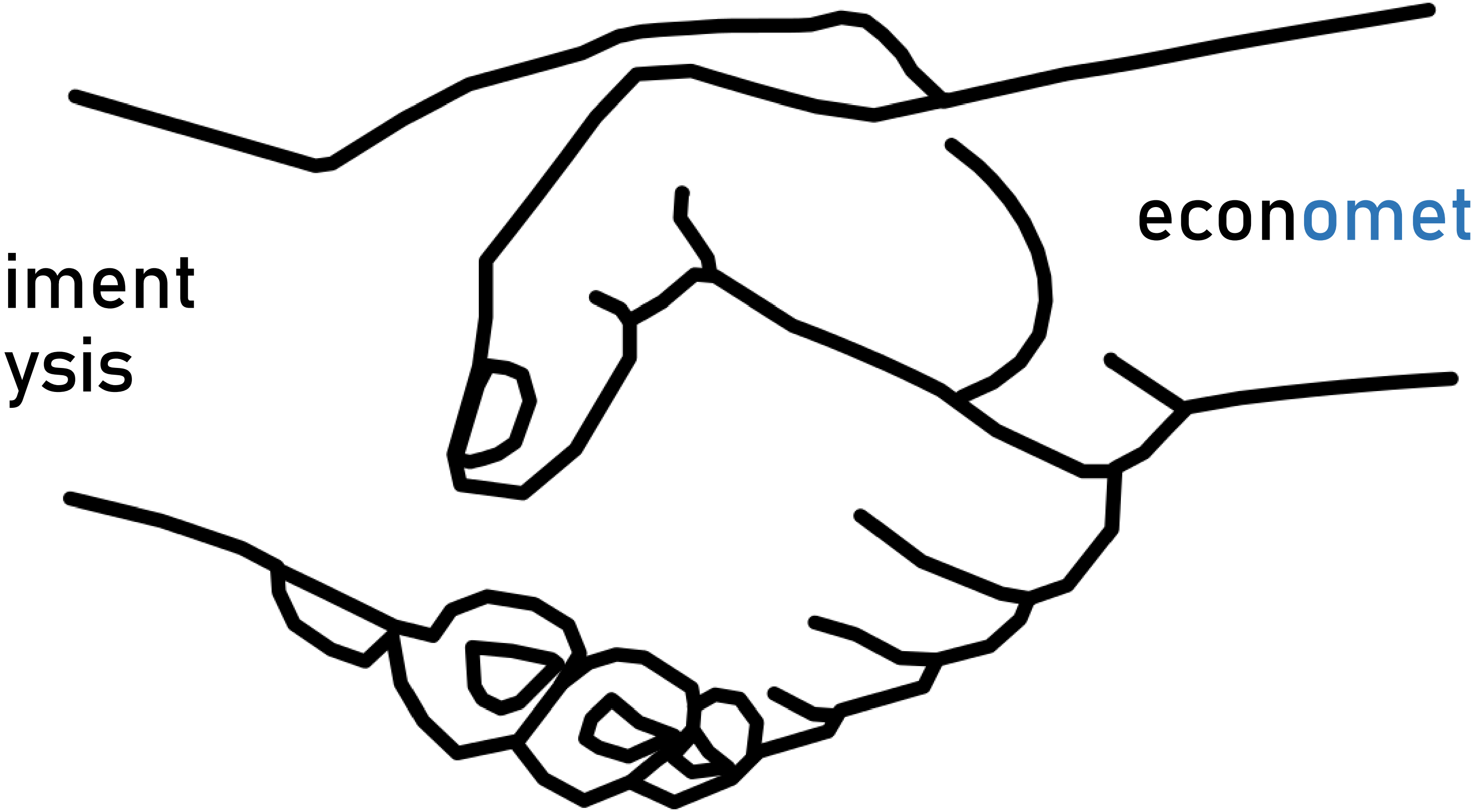
... is the analysis of quantitative time series data typically in an economic context.



Our focus is on aggregation, econometric modelling and prediction.

sentiment
analysis

econometrics

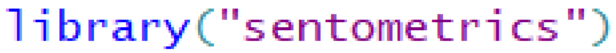


sentometrics

🔗 research

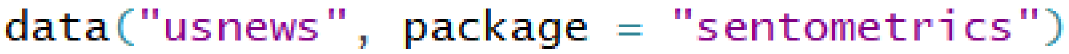
R package

Let’s go for a run with the R package ‘sentometrics’



```
library("sentometrics")
```

We have a built-in dataset of news articles between 1995 and 2014, from The Wall Street Journal and The Washington Post.



```
data("usnews", package = "sentometrics")
```

ID	DATE	TEXT	WSJ	WAPO	ECONOMY	NONECONOMY
1	1995-01-02	Full text 1	1	0	1	0
2	1995-01-05	Full text 2	0	1	1	0
…	…	…	…	…	…	…

Features: relevance/importance indicators & selectors.

Step 1

Massaging the corpus

Checking the requirements of the corpus.

```
corpusAll <- sento_corpus(usnews)
```

Subsetting the corpus, using the *quanteda* package.

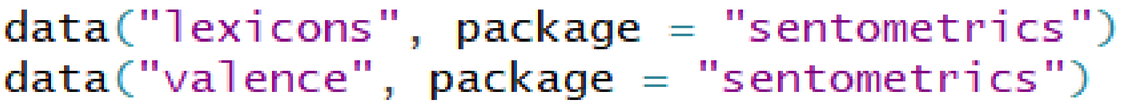
```
corpus <- quanteda::corpus_subset(corpusAll, date < "2014-10-01")
```

Adding features (for example: entities, topics, events).

```
regex <- c("\\bRepublic[s]?\\b|\\bDemocrat[s]?\\b|
             \\belection\\b|\\b[US|U.S.] [p|P]resident\\b|\\bwar\\b")
corpus <- add_features(corpus,
                       keywords = list(uncert = "uncertainty", uselect = regex),
                       do.binary = TRUE,
                       do.regex = c(FALSE, TRUE))
```

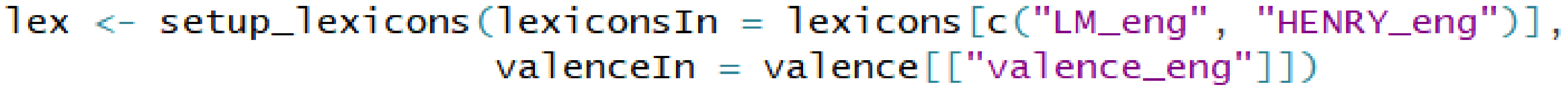
Pick the word lists for lexicon-based sentiment analysis

We have English, Dutch and French built-in word lists.



```
data("lexicons", package = "sentometrics")
data("valence", package = "sentometrics")
```

Prepare and check the lexicons.



```
lex <- setup_lexicons(lexiconsIn = lexicons[c("LM_eng", "HENRY_eng")],
                      valenceIn = valence[["valence_eng"]])
```

From sentiment to time series: aggregation specs

Aggregation of the many sentiment scores…

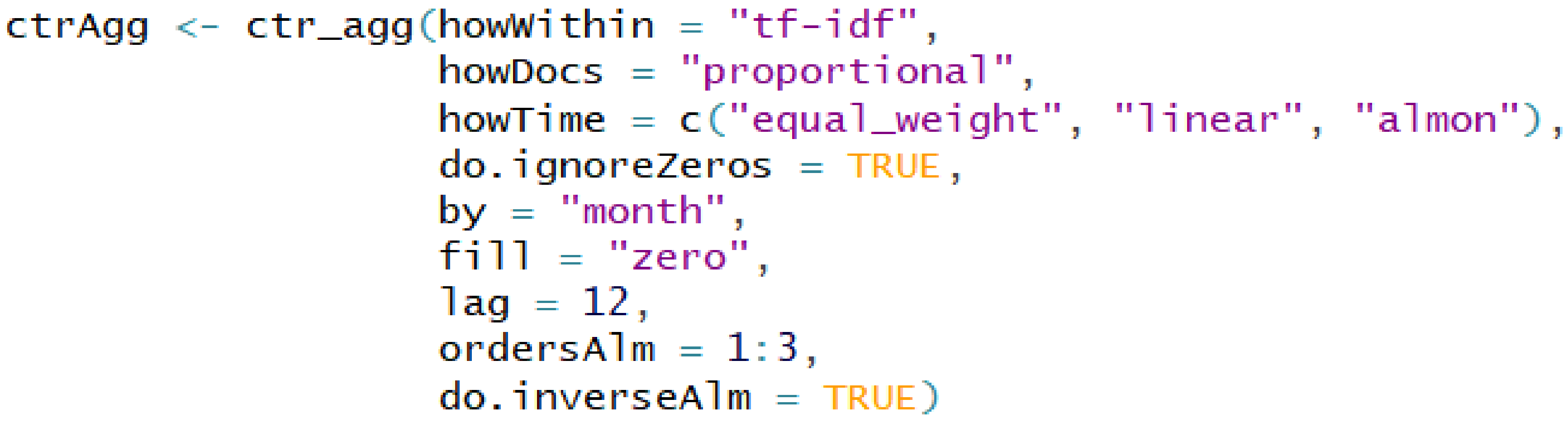
… within documents = document-level sentiment
… across documents = time series
… across time = *smoothed* time series

1 time series

… across lexicons, features and time aggregation schemes

P time series

One control function to define all of this.

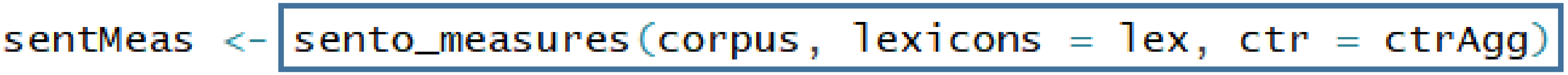


```
ctrAgg <- ctr_agg(howWithin = "tf-idf",
                  howDocs = "proportional",
                  howTime = c("equal_weight", "linear", "almon"),
                  do.ignoreZeros = TRUE,
                  by = "month",
                  fill = "zero",
                  lag = 12,
                  ordersAlm = 1:3,
                  do.inverseAlm = TRUE)
```

Steps 2 – 3

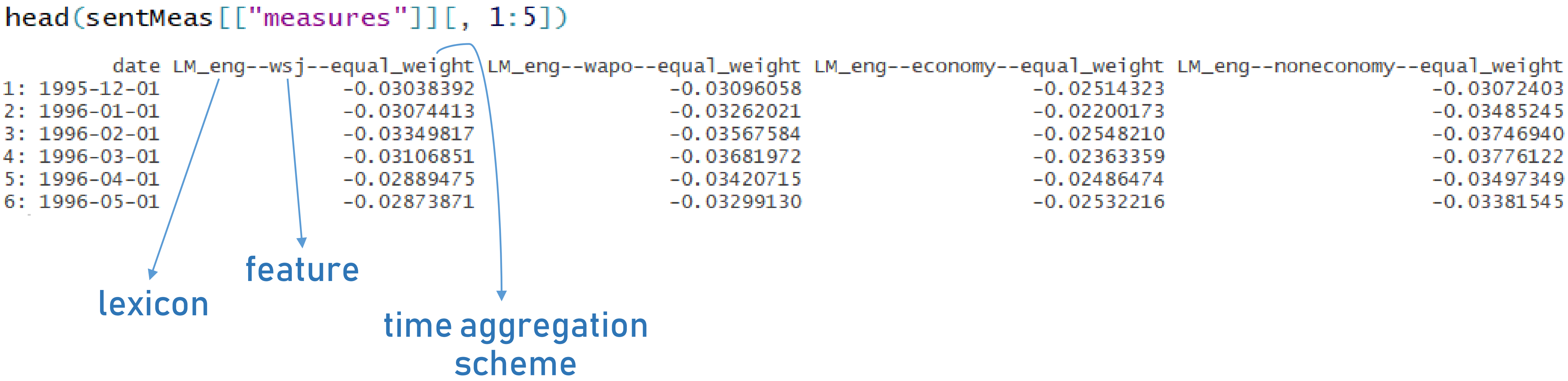
Ready to create some sentiment time series

This one simple function call gives you a wide number of different sentiment time series, or “measures”.



```
sentMeas <- sento_measures(corpus, lexicons = lex, ctr = ctrAgg)
```

The sentiment measures are represented as “lexicon—feature—smoothing”.



```
head(sentMeas[["measures"]][, 1:5])

         date LM_eng--wsj--equal_weight LM_eng--wapo--equal_weight LM_eng--economy--equal_weight LM_eng--noneconomy--equal_weight
1: 1995-12-01               -0.03038392                -0.03096058                   -0.02514323                      -0.03072403
2: 1996-01-01               -0.03074413                -0.03262021                   -0.02200173                      -0.03485245
3: 1996-02-01               -0.03349817                -0.03567584                   -0.02548210                      -0.03746940
4: 1996-03-01               -0.03106851                -0.03681972                   -0.02363359                      -0.03776122
5: 1996-04-01               -0.02889475                -0.03420715                   -0.02486474                      -0.03497349
6: 1996-05-01               -0.02873871                -0.03299130                   -0.02532216                      -0.03381545
```

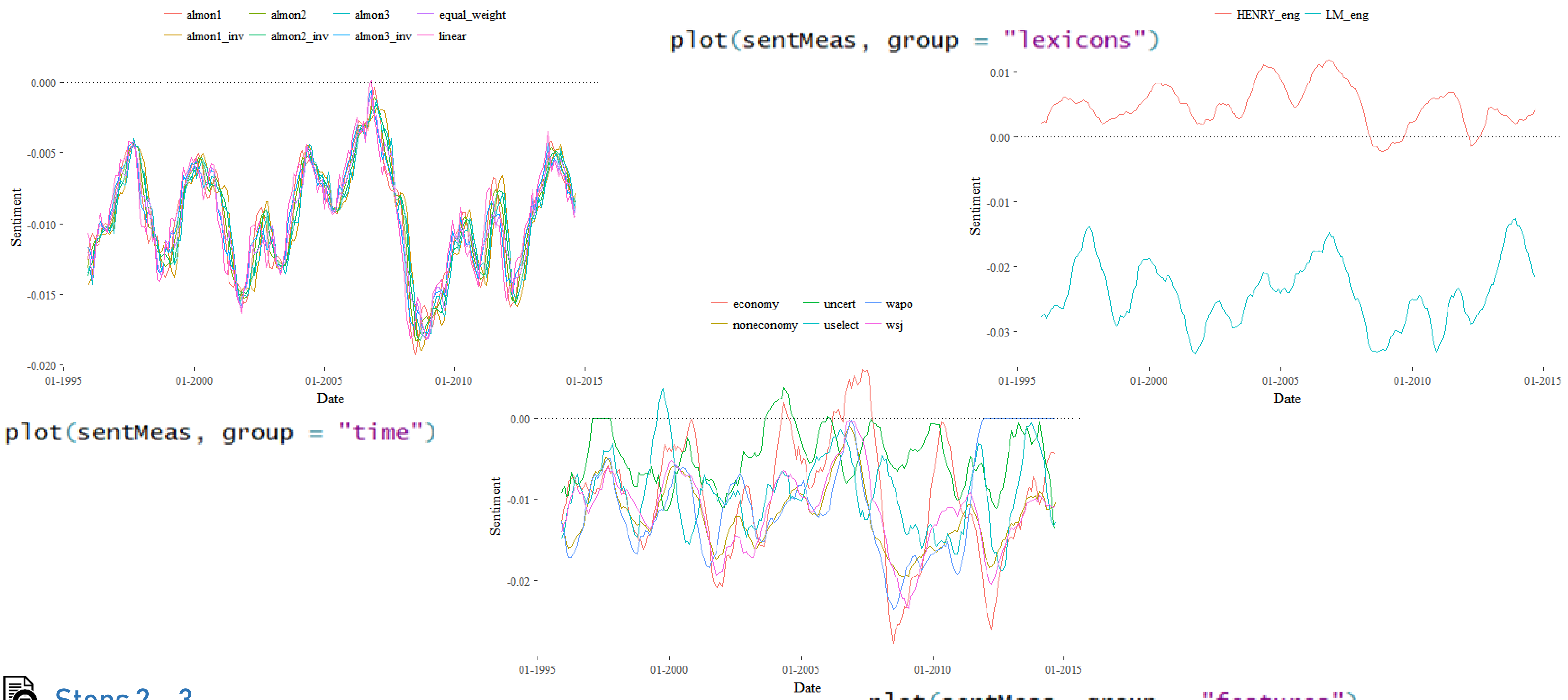
feature

lexicon

time aggregation scheme

Steps 2 – 3

Plotting across the three time series dimensions



```
plot(sentMeas, group = "time")
```

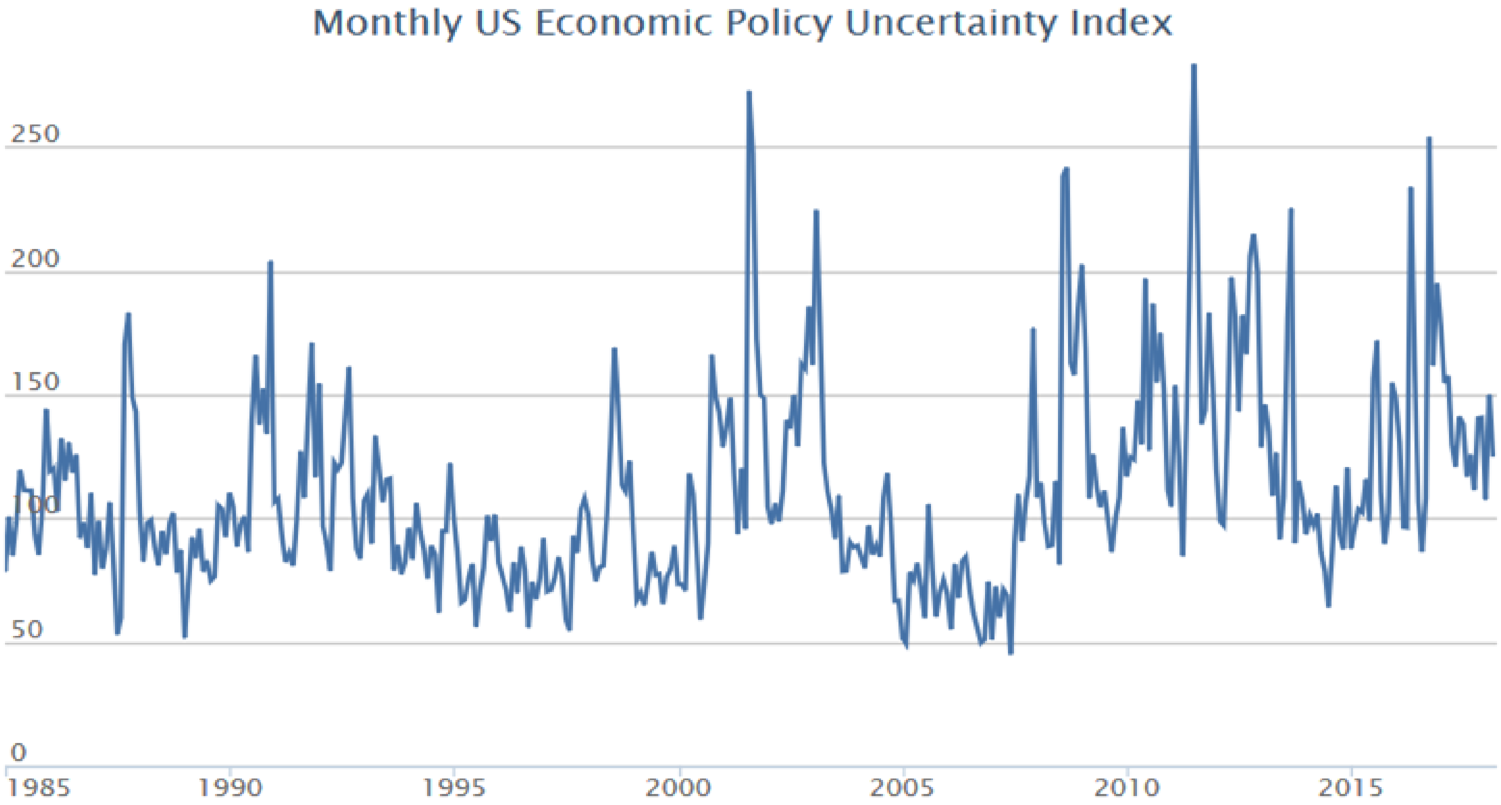
```
plot(sentMeas, group = "lexicons")
```

```
plot(sentMeas, group = "features")
```

Steps 2 – 3

We try to predict the monthly U.S. EPU index…

The Economic Policy Uncertainty (EPU) index is a partly news-based measure of policy-related economic uncertainty. It is served with the package as a dataset.



http://www.policyuncertainty.com

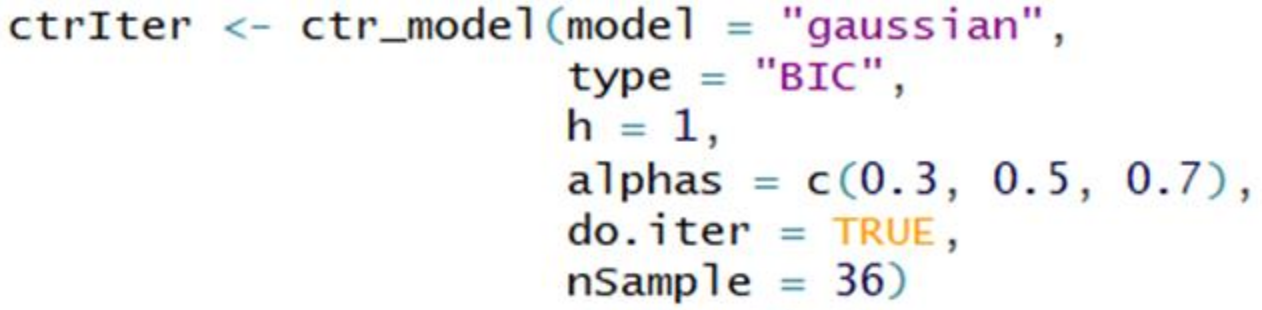
Steps 4 – 5

… using elastic net regularization

We propose to use the elastic net regression (relying on *glmnet*),which balances between the LASSO and Ridge regressions through an α parameter. The large number and collinearity of the sentiment measures motivate this choice.

$$\underbrace{y_{u+h}}_{\text{target}} = \delta + \underbrace{\gamma^\top \boldsymbol{x}_u}_{\text{other explanatory variables}} + \underbrace{\beta_1 s_u^1 + \ldots + \beta_p s_u^p + \ldots + \beta_P s_u^P}_{\text{sentiment}} + \epsilon_{u+h}$$

A straightforward control function defines the model setup.

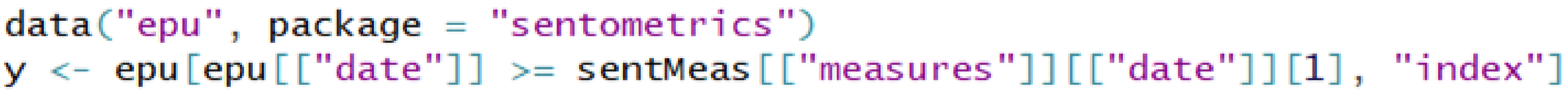


```
ctrIter <- ctr_model(model = "gaussian",
                     type = "BIC",
                     h = 1,
                     alphas = c(0.3, 0.5, 0.7),
                     do.iter = TRUE,
                     nSample = 36)
```

Steps 4 – 5

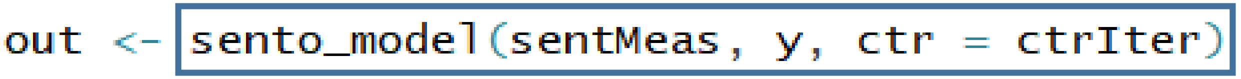
Ready to run the prediction model iteratively

Load the data.



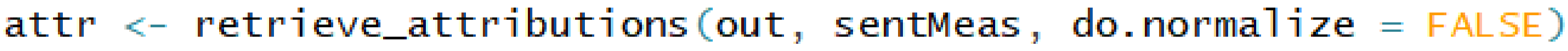
```
data("epu", package = "sentometrics")
y <- epu[epu[["date"]] >= sentMeas[["measures"]][["date"]][1], "index"]
```

Running the out-of-sample prediction analysis is easy.



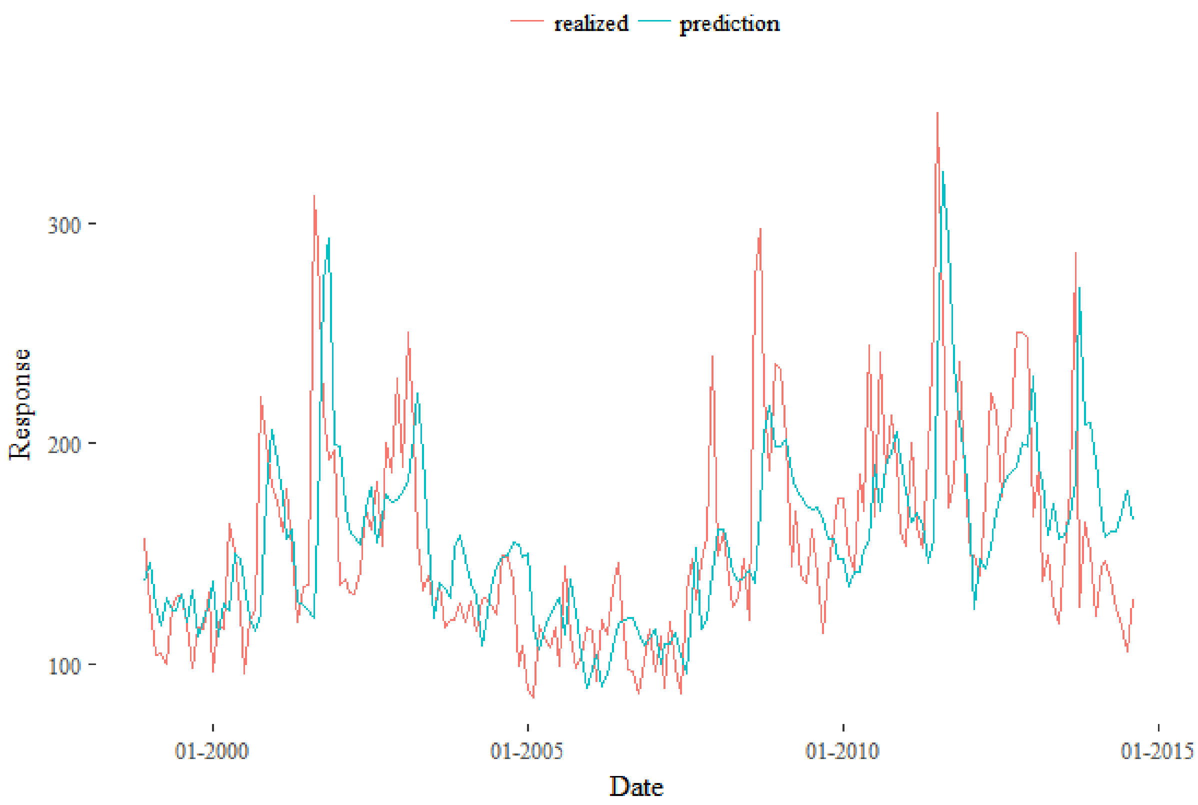
```
out <- sento_model(sentMeas, y, ctr = ctrIter)
```

We call “attribution” the decomposition of the prediction into one of the underlying sentiment time series dimensions.

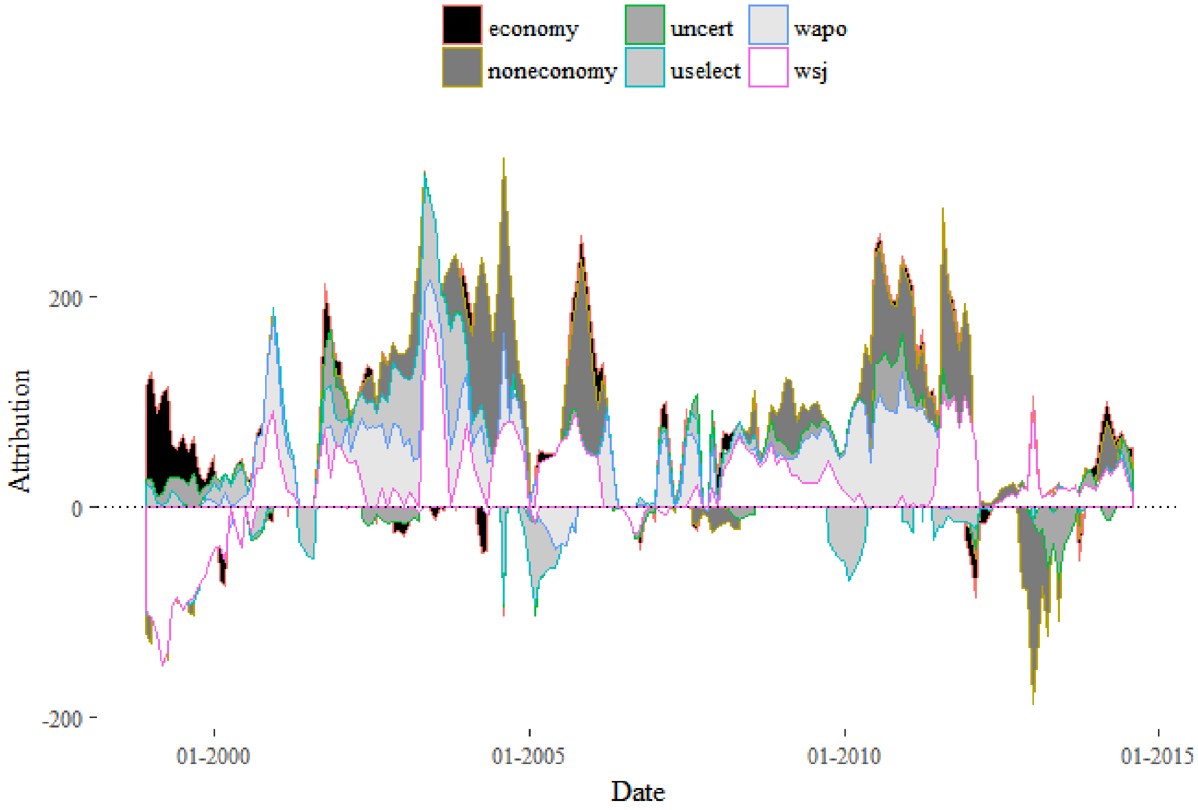


```
attr <- retrieve_attributions(out, sentMeas, do.normalize = FALSE)
```

Visualizing the out-of-sample prediction and attribution



```
plot(out)
```



```
plot_attributions(attr, group = "features")
```

Steps 4 – 5

Next steps

The package already offers quite some flexibility to develop sentiment time series.

Improvements along:

Faster and more complex sentiment analysis;
Interfaces to more types of models;
More flexible aggregation and modelling.

Purpose?

Become the go-to package for embedding textual sentiment into the prediction of other variables!

If you want to help out, get in touch!