# word2vec with R

Brussels

Jan Wijffels

BNOSAC - jwijffels@bnosac.be

bnosac

# Overzicht

1. word2vec
2. example
3. disambiguation + UMAP
4. pretrained embeddings
5. BNOSAC R consultancy

bnosac

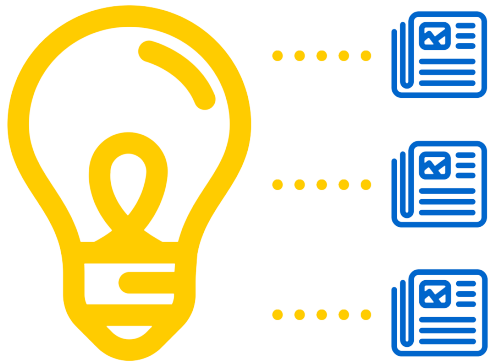
open analytical helpers

# word2vec

# word2vec

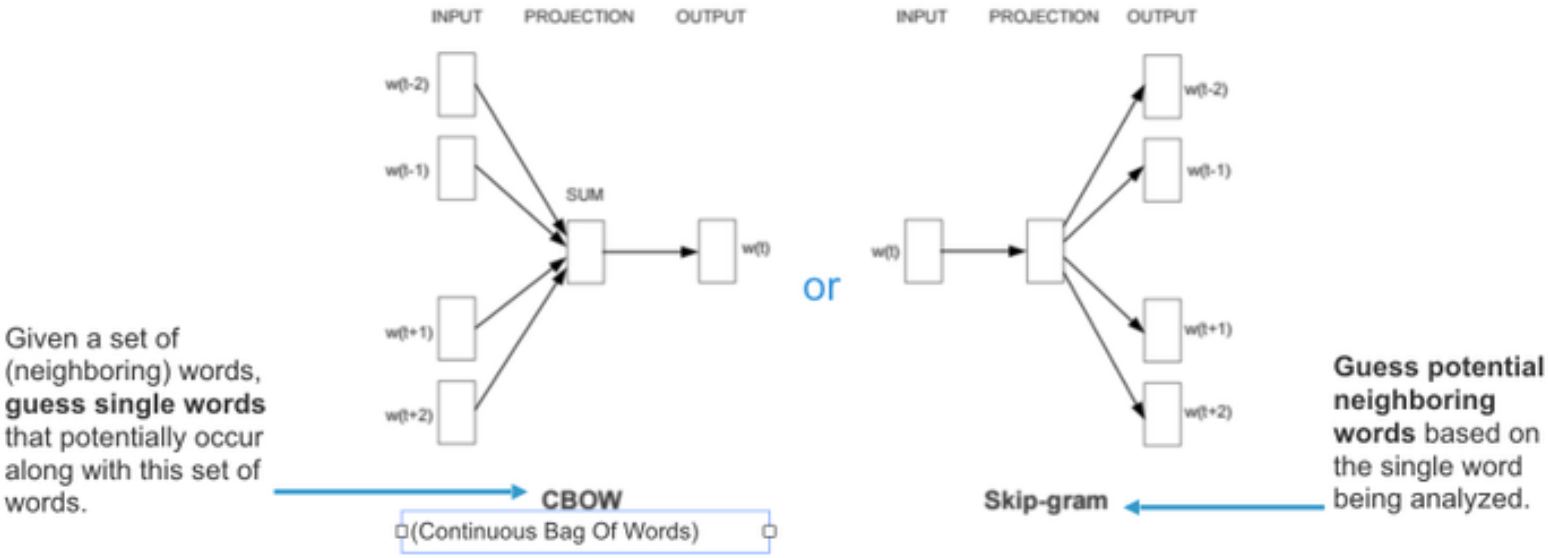
- word2vec is an algorithm by Mikolov et al. (2013) which allows to **map tokens** (words / subwords / letters / multi-word expressions) **to numbers** which are called embeddings
  - these numbers can be used to find similarities between the tokens (words / subwords / letters / multi-word expressions) or in downstream NLP tasks
  - the algorithm is explained in the paper at `https://arxiv.org/abs/1310.4546`
- For R users you can either compute word vectors by using one of the following R packages which are on CRAN:
  - R package **word2vec** (implementation of the original word2vec algorithm)
  - R package **text2vec** (implementation of the Glove algorithm)
  - R package **irlba** (implementation of Latent Semantic Analysis)
  - R package **fastTextR** (implementation of subword embeddings)
  - R package **ruimtehol** (generic embedding technique using Starspace)
  - R package **golgotha** (BERT-like embeddings)

- Common applications of word vectors are:
  - Finding similar words
  - Use the vectors as input to predictive models to do specific NLP tasks (e.g. R package *udpipe* uses these to do dependency parsing, R package *sentencepiece* to encode subwords, you can use them with the *torch* R package for named entity recognition)
  - Text matching
  - Text summarisation alongside R package *textrank*
  - Performing text translations
  - Word sense disambguation



In these slides I cover R package **word2vec** of which I am the R package author. It wraps the excellent `https://github.com/maxoodf/word2vec` C++ library using Rcpp.

- Word vectors are a set of numbers assigned to each token - commonly applied to words
- R package word2vec allows to learn such word vectors on your own texts
  - Using Continuous Bag of Words (CBOW)
  - Skip-gram




- Just install the following packages from CRAN for showcasing word2vec

```
install.packages("word2vec")
install.packages("udpipe")
```

bnosac

open analytical helpers

# example

# Example

- Following example data are AirBnB reviews. We will work on French texts.

```
library(udpipe)
data(brussels_reviews, package = "udpipe")
x <- subset(brussels_reviews, language == "fr")
x <- tolower(x$feedback)
cat(x[1])
```

```
quelle excellent week end - merci a david pour sa confiance, merci a son appart d etre aussi chouette, merci au quartier d etre aussi c
```

- Train a word2vec model by passing on text or a .txt file (e.g. Wikipedia)

```
library(word2vec)
set.seed(123456789)
model <- word2vec(x = x, type = "cbow", dim = 15, iter = 20)
```

- Once you have a model, you can get the embeddings of all words using `as.matrix`. Which has for our case 15 columns as we specified `dim = 15`.
- Or get the embedding of only a set of words.

```
embedding <- as.matrix(model)
embedding <- predict(model, c("bus", "impeccable"), type = "embedding")
embedding
```

```
                [,1]       [,2]       [,3]     [,4]       [,5]      [,6]
bus        -0.2322512  0.1992204 -1.0628699 1.806145 -0.3539968 1.5626450
impeccable -1.3262277 -0.9322608 -0.8865705 0.560683 -0.5284746 0.5343688
                [,7]      [,8]       [,9]     [,10]      [,11]      [,12]
bus        -0.7310872 -2.025789  0.1661998 0.8345852 0.02255419  0.3226628
impeccable  0.1380508 -1.540417 -0.9030897 1.4599522 1.72766161 -0.4046724
               [,13]     [,14]      [,15]
bus        1.3417009 0.1721853 -0.8065062
impeccable 0.7241428 0.7217254  1.0835044
```

- And you can find similar words
  - We can see that ‘bus’ looks like ‘tram’ and ‘transport’
  - The French word ‘impeccable’ looks similar to ‘offre’, ‘propreté’ and the verb ‘correspondre’

```
lookslike <- predict(model, c("bus", "impeccable"), type = "nearest", top_n = 5)
lookslike
```

```
$bus
  term1      term2 similarity rank
1   bus       tram  0.9928071    1
2   bus    publics  0.9874816    2
3   bus  commerces  0.9858778    3
4   bus transports  0.9845581    4
5   bus  transport  0.9833308    5

$impeccable
       term1         term2 similarity rank
1 impeccable      terrasse  0.9871505    1
2 impeccable correspondait  0.9834139    2
3 impeccable      proprete  0.9808754    3
4 impeccable         offre  0.9806080    4
5 impeccable         salon  0.9782970    5
```

- We can write the embeddings to disk and read them back again

```
write.word2vec(model, "mymodel.bin")
```

```
[1] TRUE
```

```
model <- read.word2vec("mymodel.bin")
lookslike <- predict(model, c("jardin", "cafes"), type = "nearest", top_n = 5)
lookslike
```

```
$jardin
   term1      term2 similarity rank
1 jardin   chambres  0.9516298    1
2 jardin spacieuses  0.9375631    2
```

# Training parameters

Some note on important training arguments

- **dim**: dimensionality of the word vectors: usually more is better, but not always, especially with small data
- **type**: skip-gram (slower, better for infrequent words) vs cbow (fast)
- **window**: for skip-gram usually around 10, for cbow around 5
- **hs**: the training algorithm: hierarchical softmax (better for infrequent words) vs negative sampling (better for frequent words, better with low dimensional vectors)
- **sample**: sub-sampling of frequent words: can improve both accuracy and speed for large data sets (useful values are in range 0.001 to 0.00001)

bnosac

open analytical helpers

# disambiguation + UMAP

# Visualise semantic similarity

Below, we get adjectives and put them in a 2D plot by semantic similarity.

- Use R package *udpipe* to do Parts of Speech tagging and lemmatisation
- Build a word2vec model on the lemma + the parts of speech tag
- Perform dimensionality reduction using *UMAP*
- Visualise the adjectives in 2D using ggplot2

```
## Get some text
library(udpipe)
data(brussels_reviews, package = "udpipe")
x <- subset(brussels_reviews, language == "fr")
x <- data.frame(doc_id = x$id, text = x$feedback, stringsAsFactors = FALSE)

## Do Parts of Speech tagging + combine parts of speech tag with the lemma
anno <- udpipe(x, "french", trace = 10, parallel.cores = 1)
anno <- subset(anno, !is.na(lemma) & nchar(lemma) > 1 & !upos %in% "PUNCT")
anno$text <- sprintf("%s//%s", anno$lemma, anno$upos)
```

```
head(anno[, c("doc_id", "token", "upos", "lemma", "text")], n = 3)
```

```
    doc_id     token upos     lemma           text
1 47860059    Quelle  DET      quel      quel//DET
2 47860059 excellent  ADJ excellent excellent//ADJ
3 47860059      week NOUN   weekend  weekend//NOUN
```

```
## Paste the text together again with space as a separator
x <- paste.data.frame(anno, term = "text", group = "doc_id", collapse = " ")
str(x)
```

```
'data.frame':   500 obs. of  2 variables:
 $ doc_id: chr  "47860059" "34292252" "20224463" "3630095" ...
 $ text  : chr  "quel//DET excellent//ADJ weekend//NOUN end//NOUN merci//INTJ avoir//VERB David//PROPN pour//ADP son//DET confia"| __tr
```

- Build the word2vec model

```
model     <- word2vec(x = x$text, dim = 15, iter = 20, split = c(" ", ".\n?!"))
embedding <- as.matrix(model)
```

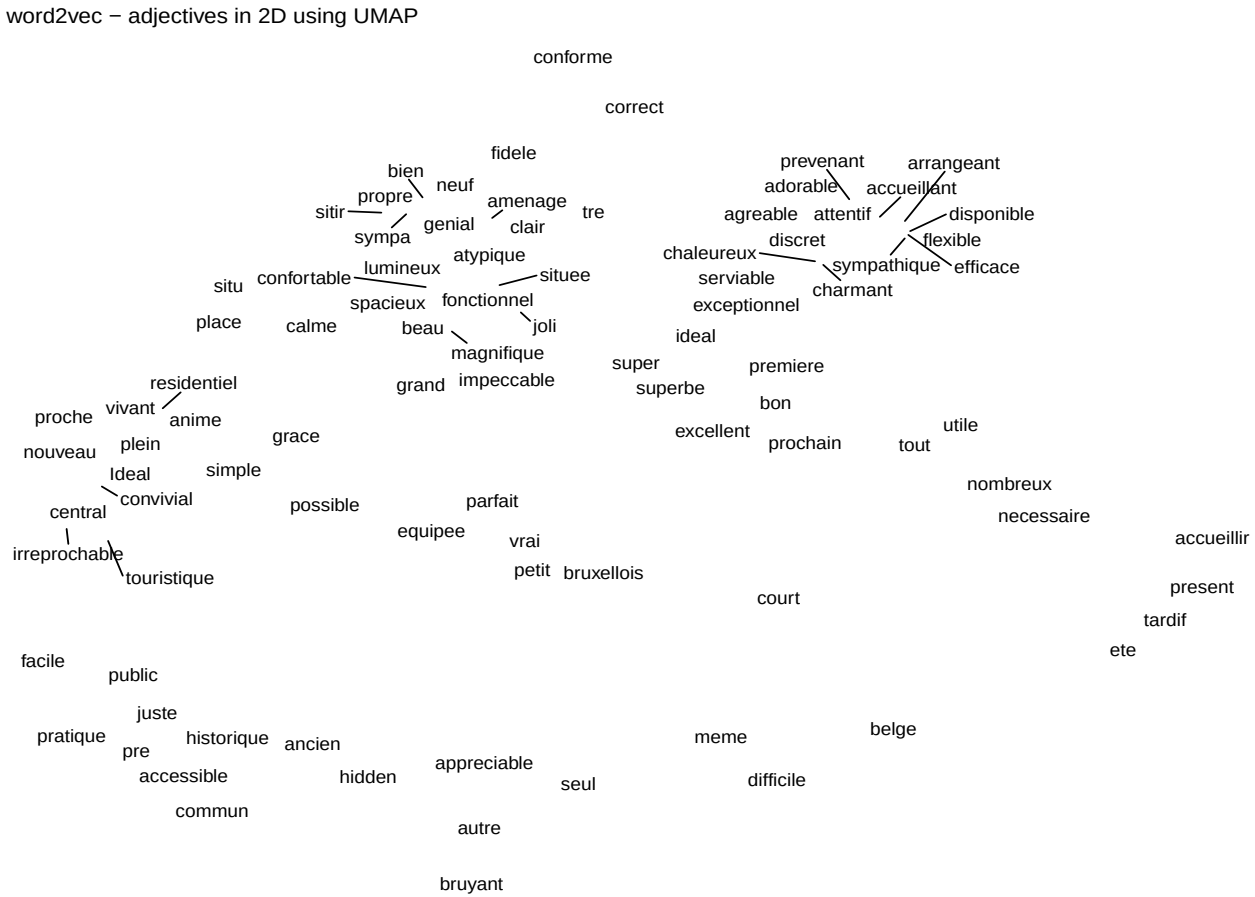
- Perform dimension reduction using UMAP, which maps embeddings in 2D

```
library(uwot)
viz <- umap(embedding, n_neighbors = 15, n_threads = 2)
rownames(viz) <- rownames(embedding)
head(viz, n = 10)
```

```
                      [,1]       [,2]
restos//NOUN     -3.0105790  0.4404782
internet//NOUN   -0.8594446 -2.4782318
absolument//ADV   1.2092939 -1.8778233
retenir//VERB     1.0604332  0.4839083
parler//VERB      1.0412366 -2.2243408
situ//VERB       -0.5221492  2.1066781
vacance//NOUN     0.7579063 -2.0998448
hotesse//NOUN     0.9480190  1.5384178
convivial//ADJ   -2.9796938  0.4346302
residentiel//ADJ -2.6128312  0.9984343
```

# Visualise

```
library(ggplot2)
library(ggrepel)
df  <- data.frame(word = gsub("//.+", "", rownames(viz)),
                  upos = gsub(".+//", "", rownames(viz)),
                  x = viz[, 1], y = viz[, 2],
                  stringsAsFactors = FALSE)
df  <- subset(df, upos %in% c("ADJ"))
ggplot(df, aes(x = x, y = y, label = word)) +
  geom_text_repel() + theme_void() +
  labs(title = "word2vec - adjectives in 2D using UMAP")
```



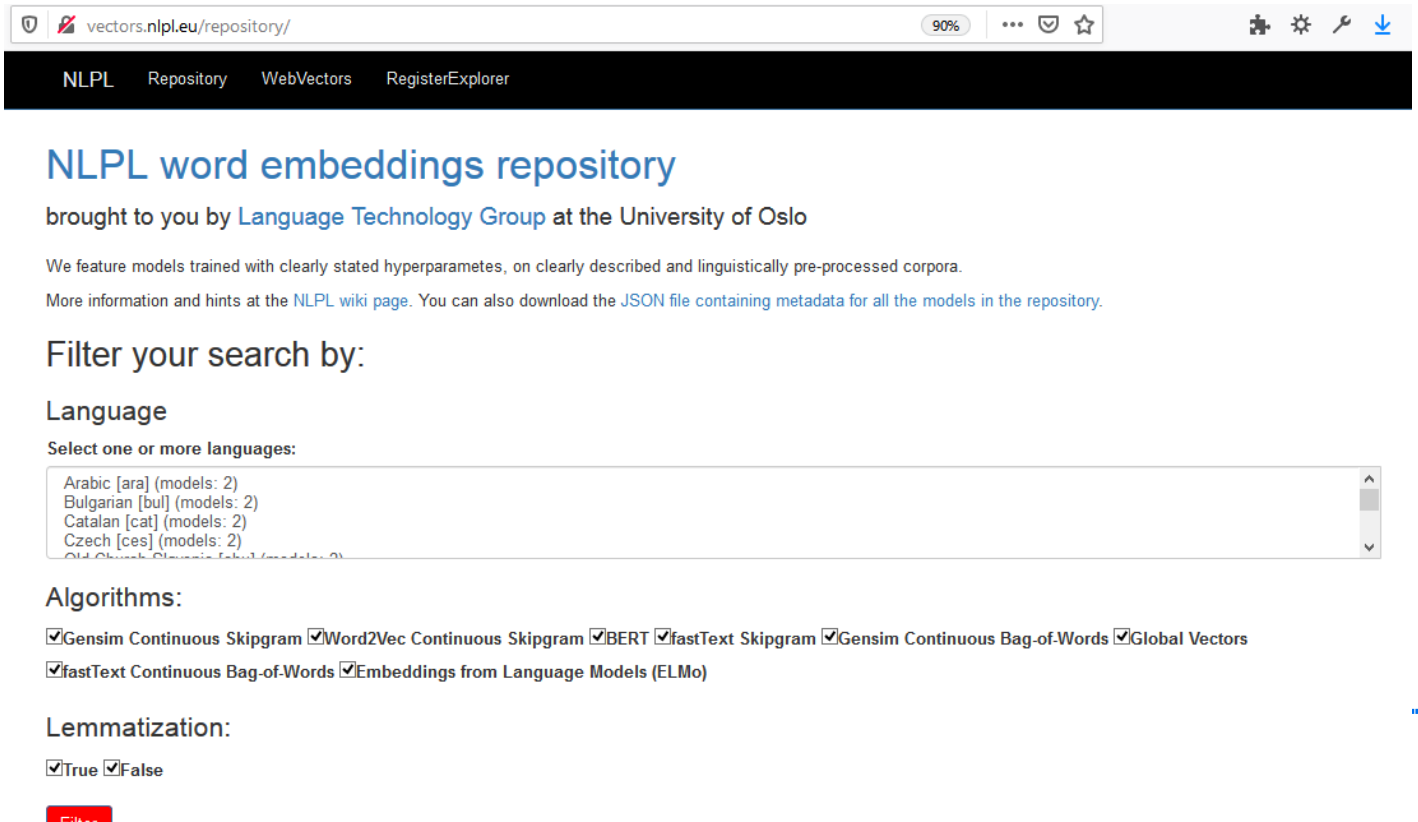
bnosac

open analytical helpers

# pretrained embeddings

# Pretrained word vectors

- Common pretrained word vectors which I use are available at
  - `http://vectors.nlpl.eu/repository`
  - `https://nlp.h-its.org/bpemb` - usefull alongside R package *sentencepiece*
  - `https://github.com/maxoodf/word2vec#basic-usage`
- Compatible are wordvectors trained with traditional word2vec and gensim
- It is important that when you load these external ones, you take the binary file and you set `normalize = TRUE`



# Example, loading model downloaded from https://github.com/maxoodf/word2vec#basic-usage

```
model <- read.word2vec("cb_ns_500_10.w2v", normalize = TRUE)
```

- Which words are similar to fries or money

```
predict(model, newdata = c("fries", "money"), type = "nearest", top_n = 5)
```

```
$fries
  term1         term2 similarity rank
1 fries       burgers  0.7641346    1
2 fries cheeseburgers  0.7636056    2
3 fries  cheeseburger  0.7570285    3
4 fries    hamburgers  0.7546136    4
5 fries      coleslaw  0.7540344    5

$money
  term1     term2 similarity rank
1 money     funds  0.8281102    1
2 money      cash  0.8158758    2
3 money    monies  0.7874741    3
4 money      sums  0.7648080    4
5 money taxpayers  0.7553093    5
```

# Word analogies

- Classical example: king - man + woman = queen

```
wv <- predict(model, newdata = c("king", "man", "woman"), type = "embedding")
wv <- wv["king", ] - wv["man", ] + wv["woman", ]
predict(model, newdata = wv, type = "nearest", top_n = 3)
```

```
      term similarity rank
1     king  0.9479475    1
2    queen  0.7680065    2
3 princess  0.7155131    3
```

- What could Belgium look like if we had a government or Belgium without a government. YES! Intelligent :). Clearly Belgium is a failed democracy.

```
wv <- predict(model, newdata = c("belgium", "government"), type = "embedding")
predict(model, newdata = wv["belgium", ] + wv["government", ], type = "nearest", top_n = 2)
```

```
         term similarity rank
1 netherlands  0.9337973    1
2     germany  0.9305047    2
```

```
predict(model, newdata = wv["belgium", ] - wv["government", ], type = "nearest", top_n = 1)
```

```
     term similarity rank
1 belgium  0.9759384    1
```

BREAKING THE BLACK BOX

# How Machines Learn to Be Racist

Word vectors are just numbers, you can prove anything with it.

- Here we prove that black + white is blue

```
wv <- predict(model, newdata = c("black", "white"), type = "embedding")
wv <- wv["black", ] + wv["white", ]
predict(model, newdata = wv, type = "nearest", top_n = 3)
```

```
     term similarity rank
1    blue  0.9792663    1
2  purple  0.9520039    2
3 colored  0.9480994    3
```

- Here we prove something white who is not a person but includes racism is black. You'll certainly love this in the US.

```
wv <- predict(model, newdata = c("black", "white", "racism", "person"), type = "embedding")
wv <- wv["white", ] - wv["person", ] + wv["racism", ]
predict(model, newdata = wv, type = "nearest", top_n = 10)
```

```
              term similarity rank
1            black  0.9480463    1
2           racial  0.8962515    2
3           racist  0.8518659    3
4  segregationists  0.8304701    4
5          bigotry  0.8055548    5
6       racialized  0.8053641    6
7          racists  0.8034531    7
8         racially  0.8023036    8
9       dixiecrats  0.8008670    9
10      homophobia  0.7886864   10
```

bnosac

open analytical helpers

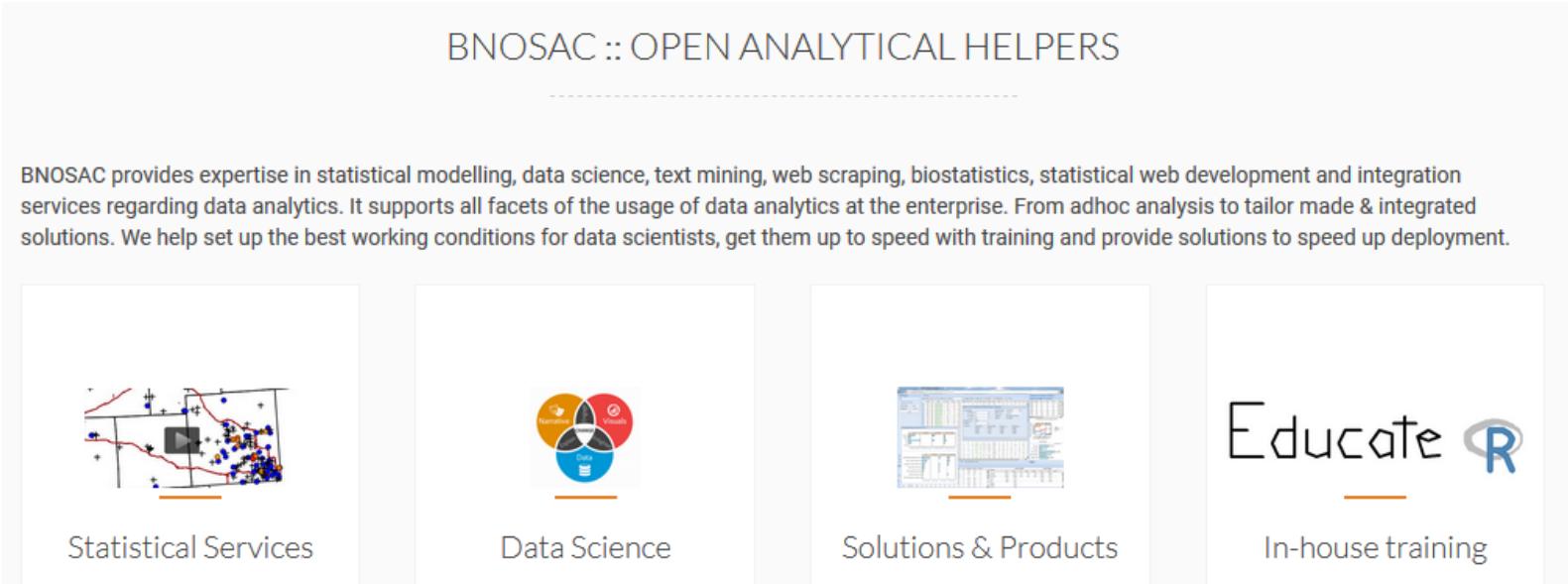
# BNOSAC R consultancy

www.bnosac.be

# www.bnosac.be: open source analytics experts

Providing consultancy services in **open source analytical engineering**

- Support for R / Oracle R Enterprise / Microsoft R / PostgreSQL / Python / ExtJS / Hadoop / …
- Expertise in predictive data mining, biostatistics, geostats, R + Python programming, GUI building, artificial intelligence, process automation, analytical web development
- R/Python implementations, application maintenance & training/consulting
- RStudio Server Pro / Shiny Pro / RStudio Connect reseller
- Hosting CRAN at www.datatailor.be
- Contributing to the R community with R packages / R training



# R support by BNOSAC

## WE SUPPORT R USAGE AT YOUR ENTERPRISE

BNOSAC gives support for all facets of the usage of R at your enterprise. We help set up the best working conditions for R users, get them up to speed with training and provide solutions to speed up deployment. For small startups to big corporations or governments.


RStudio Pro & Shiny Pro reseller

Oracle R Enterprise support


Continuous integration & deployment for R users


Inhouse R training & package development

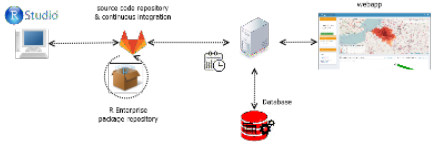
# Training by BNOSAC: www.bnosac.be/training

R Training

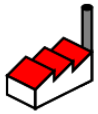
Data viz &
data munging

**2a. Common data manipulation & Programming in R**
**2b. Visualisation with R**
**2c. Reporting with R**
**2d. Data connectivity with R**

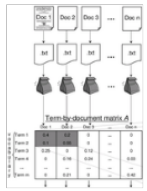
**1. R for starters**




Deploying



**4a. Creating R packages and R repositories**
**4b. git/SVN + continuous integration with R**
**4c. Managing R processes**
**4d. Integration of R in web applications**
**4e. Shiny**
**4f. Big data analytics with R (Spark, HAWQ & PL/R)**

Analytics

**3a. Statistical Machine learning with R**
**3b. Text mining with R**
**3c. Applied Spatial Analysis with R**
**3d. Computer Vision with R and Python**



bnosac
open analytical helpers

# www.bnosac.be

Need support, send a message at www.bnosac.be/index.php/contact/get-in-touch

## OFFICE ADDRESS

The bnosac office is located in

- la Lustrerie Business Center - http://www.lalustrerie.be
- Paleizenstraat 153, 1030 Brussels, Belgium
- close to Gare du Nord in Brussels
- email: info [at] bnosac [dot] be