cooc.R Des outils de calcul pour la sémantique historique

MODE D'EMPLOI (simplifié)

Nature et finalité du script

Le script cooc.R est constitué par une quinzaine de fonctions R, de taille et d'importance variées.
J'ai écrit ces fonctions dans le cadre de mes recherches, en fonction des besoins. Elles sont donc nées dans un contexte particulier.
Leur finalité principale est de permettre des analyses statistiques sur des textes inclus dans des bases de données textuelles open-cwb, en utilisant méthodiquement les ressources de la bibliothèque rcqp (Desgraupes et Loiseau), grâce à laquelle le logiciel R peut communiquer directement avec ces bases. Cette bibliothèque, en grande partie constituée de C compilé, travaille avec une extrême rapidité, ce qui permet d'obtenir des temps de réponse brefs, même pour des requêtes à la fois complexes et volumineuses.
Ces bases de données sont toutes constituées de textes historiques, et les calculs statistiques visent avant tout à mettre en œuvre des procédures d'analyse sémantique historique. Au surplus, ces textes sont surtout des textes latins, qui ont été tagués-lemmatisés avec tree-tagger selon les conventions adoptées dans le cadre du groupe et du projet OMNIA (www.glossaria.eu), et avec les paramètres créés par ce groupe. Tout cela pour dire que ces fonctions ne visaient pas, à l'origine, une finalité universelle. Il apparaît cependant, à l'expérience, qu'elles permettent de traiter directement des textes français, allemand et anglais (et certainement d'autres langues) moyennant de minimes ajustements.
Il faut cependant souligner que les limites autant que les possibilités de choix que ces fonctions comportent résultent de choix personnels ; des développements appropriés permettraient d'introduire des possibilités de paramétrage plus larges et plus variées.

Trois catégories d'opérations

Ces fonctions se répartissent nettement en trois groupes :
* un groupe a pour fin de montrer l'évolution de la fréquence relative, dans un corpus daté donné, de tel ou tel lemme, forme, bilemme ou expression quelconque.
* un autre groupe est centré sur les cooccurrences : il s'agit d'abord d'établir la liste des cooccurrents les plus significatifs autour d'un pivot (lemme) ; à partir de là, on prend en compte (selon deux méthodes différentes) les relations entre ces cooccurrents, pour construire un lexicogramme, c'est-à-dire un graphique qui visualise l'articulation générale de toutes ces relations ; quelques autres fonctions annexes permettent de comparer des listes de cooccurrents ou de rechercher des évolutions à l'intérieur de cette population.
* enfin, quelques fonctions-outils, destinées à identifier des lemmes ou des formes (recherche floue) ou à préparer des objets analysables par d'autres bibliothèques (pour le moment, zipfR).

Premier groupe : évolutions

---

freqlem(corp, lm, trnch=20, attr="", val="", date)

Fournit un graphique de l'évolution de la fréquence d'un lemme dans le corpus, en découpant celui-ci en tranches égales, seule manière de procéder qui permette de comparer correctement les effectifs ; l'opération est très simple pour le corpus entier, mais se complique lorsqu'on choisit un sous-corpus, car dans ce cas il faut créer un index auxiliaire permettant d'obtenir effectivement des tranches égales, quelle que soit la répartition concrète du sous-corpus à l'intérieur du corpus.
NB la courbe de tendance est obtenue par la fonction `lowess()`, avec les paramètres par défaut ; on peut modifier ceux-ci, soit en les inscrivant dans le script, soit en les faisant apparaître parmi les paramètres de la ligne de commande.

En entrée :
corp : nom du corpus (en capitales)
lm : lemme-pivot
trnch : nombre de tranches dans le corpus
attr : choix éventuel d'un attribut permettant de sélectionner un sous-corpus
val : valeur de l'attribut choisi
date : nom exact de l'attribut servant d'index chronologique

En sortie :
↠ la fonction produit un graphique de type histogramme complété par une courbe de tendance
la fonction ne renvoie pas d'objet, mais crée, dans le workspace, deux vecteurs :
↠ EFTR : les effectifs pour chaque tranche
↠ LB1 : la valeur de la date du centre de chaque tranche (colonne de l'histogramme)
on peut récupérer ces deux objets, sinon ils sont écrasés lors de l'emploi suivant de la fonction.

---

freqlem2(corp, lm, trnch=20, attr="", val="", date, D=0, F=Inf)

Fonction analogue à la précédente, mais avec anamorphose ; la fonction simple `freqlem()` renvoie en fait un simple histogramme, avec des colonnes de même largeur, cette largeur correspondant aux effectifs, qui sont identiques ; dans ce cas, les abscisses correspondent au numéro d'ordre des tokens du corpus. Avec la fonction `freqlem2()`, on calcule les mêmes effectifs, mais on place les bornes des colonnes sur un axe chronologique, ce qui donne le plus souvent des colonnes de largeurs inégales ; l'avantage est de procurer une une représentation de l'évolution bien plus conforme à l'habitude, puisque les abscisses gomment en quelque sorte le fait que les textes ne sont pas répartis de manière homogène sur toute la durée concernée, tout en conservant le principe des tranches d'effectif égal.
Accessoirement, cette fonction permet de choisir une borne de départ et/ou une borne de fin différentes des extrémités du corpus ; les textes non datés (i.e. codés 999 ou 9999) sont automatiquement éliminés.

En entrée :
corp : nom du corpus (en capitales)
lm : lemme-pivot
trnch : nombre de tranches dans le corpus
attr : choix éventuel d'un attribut permettant de sélectionner un sous-corpus
val : valeur de l'attribut choisi
date : nom exact de l'attribut servant d'index chronologique
D / F : choix d'un début et d'une fin (cpos)

En sortie :
↠ la fonction produit un graphique.
la fonction ne renvoie pas d'objet, mais crée, dans le workspace, deux vecteurs :
↠ EFTR : les effectifs pour chaque tranche
↠ LB1 : la valeur de la date du centre de chaque tranche (colonne de l'histogramme)
on peut récupérer ces deux objets, sinon ils sont écrasés lors de l'emploi suivant de la fonction.

NB la courbe de tendance est obtenue par la fonction lowess(), avec les paramètres par défaut ; on peut modifier ceux-ci, soit en les inscrivant dans le script, soit en les faisant apparaître parmi les paramètres de la ligne de commande.

---

freqfrm(corp, frm, trnch=20, attr="", val="")

forme simplifiée de `freqlem()`, où l'on remplace `lm` par `frm` ; permet d'examiner rapidement telle ou telle forme d'un lemme.

---

freqexp(corp, exp, trnch=25, attr="", lim=200)

Examen de l'évolution chronologique d'une expression quelconque ; *cette expression doit être écrite selon les règles du CQL* ; l'affichage est double : d'un côté un graphique du type `freqlem()`, et de l'autre une matrice incluant les contextes contenant l'expression ; le nombre de contextes pouvant être très important, on le limite à « `lim` », un choix aléatoire est alors effectué pour donner un échantillon « représentatif ».
NB : on peut faire appel, dans l'expression, à des macros CQP, si celles-ci ont été lancées au départ par le fichier .cqprc

En entrée :
corp : nom du corpus (en capitales)
exp : expression (en CQL)
trnch : nombre de tranches dans le corpus
attr : choix de l'attribut à afficher
lim : nombre maximal de lignes à afficher

En sortie :
- le graphique de l'évolution
- affichage des contextes (en nombre inférieur à `lim`)
- un objet de type `matrix` contenant ces contextes (en totalité)

---

freqcooc(corp, lm1, lm2, dis=5, coeff="dice", trnch=10, D=0, F=Inf, date, selec=1)

Fonction qui permet l'affichage chronologique de deux lemmes, dénombrés par tranches égales du corpus, et le nombre de leurs cooccurrences dans ces mêmes tranches, plus le coefficient de significativité de ce dernier effectif. Toutes ces valeurs sont indiquées en indices (moyenne 100).
Ce graphique offre ainsi la possibilité de lire l'évolution relative des deux lemmes ainsi que l'évolution de la significativité de leurs cooccurrences : il s'agit donc d'un outil destiné à examiner l'évolution du lien entre deux lemmes.
Ce graphique doit être interprété avec attention : les courbes étant de simples indices, une cooccurrence globale très significative ne sera pas différente d'une autre insignifiante, il faut considérer de près les effectifs.

En entrée :
corp : nom du corpus en capitales
lm1 / lm2 : deux lemmes
dis : nombre de tokens à considérer de part et d'autre du pivot
coeff : coefficient de significativité utilisé (choix : dice/poiss/daille/pmi/hyperg/ms)
selec : paramètre utilisable uniquement si l'on choisit le coefficient de dice ; par défaut 1, une valeur supérieure rend le filtre plus sélectif, une valeur inférieure moins sélectif (faire des essais)
trnch : nombre de tranches égales dans le corpus
D / F : possibilité de définir un début et/ou une fin différents des extrémités
date : attribut à utiliser comme index chronologique

En sortie :
- un graphique affichant les courbes brutes et les différentes courbes de tendance
- un `data.frame` à 9 colonnes comportant tous les effectifs et les valeurs des divers coefficients de significativité.

`stablem(corp, lm, trnch="", mod="R")`

Fonction expérimentale destinée à examiner la stabilité de la moyenne et de la variance de la distribution des effectifs d'un lemme considéré sur n tranches successives ; on commence par deux tranches, et l'on augmente d'une unité jusqu'à couvrir le corpus entier ; on peut procéder de deux manières : soit on considère les tranches dans l'ordre naturel, soit on les considère dans un ordre aléatoire ; dans ce cas, la fonction `sample()` donnant des suites toujours différentes, on obtiendrait des courbes toutes différentes ; pour remédier à cet inconvénient, on effectue un pseudo-boostrap, c'est-à-dire que l'on calcule la même chose 100 fois et que l'on affiche la moyenne des résultats.
En pratique, cette fonction exploratoire doit être mise en œuvre plusieurs fois, en faisant varier fortement les paramètres. On observera couramment que l'évolution de la variance cumulée n'est pas du tout la même selon la largeur des tranches.

En entrée :
`corp` : nom du corpus en capitales
`lm` : lemme
`trnch` : nombre de tranches
`mod` : choix entre « R » (ordre réel) et « A » (ordre aléatoire avec bootstrap)

En sortie :
↬ un graphique composite, figurant 1. l'évolution de la fréquence par tranches (graphique `freqlem()`), 2. un histogramme des fréquences par tranche, 3. un graphique d'évolution de la moyenne cumulée, 4. un graphique d'évolution de la variance cumulée.
↬ un objet de type `list` comportant deux vecteurs : les effectifs par tranches, la variance cumulée par tranches successives cumulées.

==========

Deuxième groupe : cooccurrences

`coocA(corp, piv, poscooc="QLF SUB VBE", dis=5, coeff="dice", selec=1, nbs=30, D=0, F=Inf, attr="", val="")`

Fonction principale permettant d'établir la liste des cooccurrents les plus « significatifs » d'un lemme-pivot, dans un corpus donné. Cette fonction est conçue pour faciliter deux types de choix : d'une part des choix portant la manière de dénombrer les cooccurrents et de calculer la significativité de ces effectifs, et d'autre part des choix permettant de ne prendre en compte qu'un sous-corpus.
Le choix final des cooccurrents retenus s'effectue en utilisant un des coefficients suivants :
* Dice-Soerensen (par défaut)
* Poisson-Stirling
* Daille
* Pointwise Mutual Information
* Hypergéométrique
* Minimum Sensitivity
(documentation et formules : www.collocations.de ; auteur : Stefan EVERT)
NB : l'expérience tend à montrer que les filtres les plus efficaces sont, dans l'ordre, Dice, MS et Hypergéométrique ; il faut toujours en essayer plusieurs).

En entrée :
`corp` : nom du corpus en capitales
`piv` : lemme-pivot
`poscooc` : sélection des lemmes par les POS retenus ; le choix doit être effectué en fonction des POS retenus par chaque jeu de paramètres (liste à écrire en capitales et en séparant les pos par de simples espaces)
`dis` : nombre de tokens à prendre en considération de part et d'autre du pivot
`coeff` : coefficient de significativité permettant de classer les cooccurrents (dice/poiss/daille/pmi/hyperg/ms)
`selec` : paramètre accessoire si l'on choisit le coefficient de dice : 1 par défaut, un coefficient supérieur rend le filtre plus sélectif, un coefficient inférieur moins sélectif
`nbs` : nombre de cooccurrents retenus
`D / F` : bornes inférieure et supérieure autres que les extrémités du corpus

attr / val : paire attribut-valeur permettant de définir un sous-corpus

En sortie :
- affichage de la liste des cooccurrents les plus significatifs en fonction des choix de départ
- un objet de type `list` et de classe « COOCA », comportant à la fois tous les choix de départ et tous les résultats intermédiaires et finaux.

---

coocB(tc, cex=.7, marg=4, prox="dice", selec=1)

Cette fonction a pour finalité de calculer les cooccurrences du second ordre, entre les cooccurrents du premier ordre (précédemment calculés) ; selon deux méthodes successivement : soit dans le corpus entier, soit uniquement dans les contextes du pivot. Les graphiques résultent d'analyses en composantes principales de tableaux carrés symétriques comportant les coefficients de dice entre toutes les paires de cooccurrents retenus ; l'analyse porte sur les colonnes, on affiche le graphique des axes 1-2. On ajoute sur les graphiques des segments reliant les points les plus proches en fonction du coefficient de dice ou de spearman ; le coefficient de dice relie des points qui apparaissent dans les mêmes contextes (cooccurrents), le coefficient de spearman des points qui ont le même profil, c'est-à-dire apparaissent dans des contextes de même type, mais pas les mêmes, il s'agit donc plus de termes apparentés ou synonymes.
NB : pour rendre les graphiques lisibles, on utilise une procédure ad hoc qui fait disparaître toutes les superpositions d'étiquettes, en repoussant progressivement ces étiquettes vers l'extérieur du graphique, à partir du centre ; selon la configuration des nuages de points, cette procédure peut aboutir à un graphique sensiblement différent du résultat brut de l'ACP.

En entrée :
tc : nom de l'objet de classe « COOCA » créé par `cooca()`
cex : taille des caractères sur les graphiques
marg : nombre de tokens pris en compte dans le corpus (équivalent de `dis` pour `coocA()`)
prox : coefficient choisi pour les segments reliant, sur les graphiques, des lemmes « voisins » ; choix : dice (= coefficient de significativité de la cooccurrence, déjà calculé) / spearman (= corrélation linéaire entre les colonnes du tableau des coefficients)
selec : paramètre accessoire appliqué au calcul du coefficient de dice que l'on calcule entre les cooccurrents de premier ordre.

En sortie :
- deux graphiques (lexicogrammes) représentant les relations entre cooccurrents du premier ordre, dans le corpus entier d'une part, dans les contextes du pivot d'autre part
- dans la console, la liste des paires les plus fréquentes dans les contextes (sans pondération)
- un objet de type `list` et de classe « COOCB », contenant les paramètres de départ, les résultats intermédiaires et finaux.

---

coocE(tc, trnch=3, cex=.8)

Fonction d'analyse chronologique des résultats de `coocA()` répartis en tranches ; graphique de type Bertin-Cibois ; on considère donc uniquement les cooccurrents sélectionnés sur l'ensemble du corpus, et l'on examine leurs fréquences dans les diverses tranches.
On part d'un tableau complet des cooccurrents, et on le découpe en `trnch` tranches égales ; d'où résulte un tableau à `trnch` colonnes et `nbs` lignes (nombre de cooccurrents) ; on calcule les écarts à l'indépendance et l'on réordonne ce tableau des écarts par un simple calculs des moyennes réciproques sur le colonnes (la première ligne correspond au lemme le plus représenté dans la première tranche, la dernière ligne à celui qui est le plus représenté dans la dernière tranche). Ce tableau est affiché sur la console et traduit en graphique par la méthode Bertin-Cibois. On crée d'autre part un graphique de type `freqlem()`, sur lequel sont indiquées les bornes entre tranches.

En entrée :
tc : objet de classe « COOCA » créé par `coocA()`
trnch : nombre de tranches chronologiques
cex : taille des caractères sur le graphique

En sortie :
↬ un tableau des effectifs par tranche pour chaque coocccurrent.
↬ un graphique de Bertin-Cibois représentant ce même tableau
↬ un graphique d'évolution de la fréquence du lemme-pivot, avec indication des bornes des tranches
↬ un objet de type `list` et de classe « COOCC », comportant les résultats des calculs (tableau des fréquences et tableau des écarts à l'indépendance).

`coocC(x, y)`

Fonction de comparaison brute des listes de cooccurrents (munis de leur coefficient) ; affiche trois groupes : a) les cooccurrents présents uniquement dans x, b) ceux que l'on trouve dans x et dans y, c) ceux que l'on trouve uniquement dans y.
Fonction de simple mise en ordre, permettant en particulier d'examiner le taux de recouvrement des deux listes ; ce taux varie de 0 (aucun recouvrement) à `nbs` (recouvrement complet) ; très efficace pour comparer des tranches successives ou des tranches avec le corpus entier.

En entrée :
`x / y` : objets de classe « COOCA » créés par coocA()

En sortie :
↬ un objet de type `matrix` formé par les trois listes superposées.

`coocK2(corp, attr, lm1, lm2, dis=8, lim=200)`
`coocK3(corp, attr, lm1, lm2, lm3, dis=8, lim=200)`

Fonctions de recherche des paires et des triplets de lemmes, présents dans n'importe quel ordre (peut être obtenu par de simples macros CQP), et d'affichage des contextes (20 ou 24 tokens), accompagnés d'une indication d'attribut ; si le nombre de contextes dépasse une limite fixée, on procède à un choix aléatoire.

En entrée :
`corp` : nom du corpus en capitales
`attr` : attribut à afficher à côté des contextes
`lm1 lm2 lm3` : nom des lemmes
`dis` : distances entre les lemmes dans les contextes
`lim` : limite du nombre de lignes affichées

En sortie :
↬ affichage de la liste des contextes sélectionnés
↬ un objet de type `matrix` comportant ces mêmes résultats (liste complète)

==========

Troisième groupe : fonctions utilitaires

`flou(corp, type, larg=0, mot)`

Fonction destinée à rechercher, dans un corpus, les lemmes ou les formes qui « ressemblent » le plus à un un lemme ou à une forme donnés ; la fonction établit la liste complète des lemmes ou formes du corpus et utilise la fonction `agrep()` (procédure tre-agrep) de recherche floue par permutations pour identifier les objets les plus voisins ; la fonction renvoie une liste plus ou moins longue selon la valeur du paramètre larg (entre 0 et 3, en général).

En entrée :
`corp` : nom du corpus en capitales
`type` : word ou lemma

larg : coefficient d'approximation (0 à 3, rarement davantage)
mot : chaîne de caractères dont on cherche les voisins

En sortie :
↬ un objet de type `data.frame` comportant 3 ou 4 colonnes : pour les lemmes, les lemmes identifiés, leur effectif et leur POS ; pour les formes, on ajoute le lemme correspondant.

VB0(corp, attr, num=0)

Fonction de création d'un objet comportant tous les tokens d'un corpus munis de leurs attributs de position (cpos, word, lemma, POS) et d'un attribut de structure (au choix)

En entrée :
corp : nom du corpus en capitales
attr : choix d'un attribut de structure (pour tri éventuel ultérieur)
num : indiquer si l'attribut est numérique (1) ou non (0)

En sortie :
↬ un objet de type `list` et de classe « VBASE », à 7 colonnes :
* cpos
* id du lemma
* id du word
* id du POS
* lemma
* word
* attribut

VBZ(base, D="", F="", val="", hz="")

Fonction de création des objets du format approprié à leur utilisation par les fonctions de la bibliothèque zipfR ; permet de sélectionner un sous-corpus par diverses méthodes : *en choisissant des bornes de début et de fin, * en sélectionnant une valeur de l'attribut précédemment choisi, * en procédant à un tirage aléatoire selon une distribution de Poisson.
NB `VB0()` et `VBZ()` consomment beaucoup de mémoire, un débordement peut facilement se produire sur une machine disposant de moins de 4 Go de RAM.

En entrée :
base : un objet de classe « VBASE » créé par la fonction `VB0()`
D /F : sélection d'un sous-corpus par bornes de début et de fin
val : valeur éventuelle de l'attribut pour création d'un sous-corpus
hz : coefficient lambda de la fonction `rpois()`, permettant de sélectionner un sous-ensemble par tirage au hasard (selon une distribution de Poisson) ; la valeur de `hz` détermine le caractère plus ou moins sélectif du tri, procéder par essais successifs (plus la valeur est faible, plus le résultat est limité ; en gros, ce coefficient représente la proportion que l'on souhaite sélectionner, faire des essais entre 0.1 et 1)

En sortie :
↬ un objet de type `list` à 8 éléments :
* cposd : cpos de départ
* cposf : cpos de fin
* Ltabtfl : objet tfl (lemmes) [tfl : type frequency list]
* Ltabvgc : objet vgc (lemmes) [vgc : vocabulary growth curve]
* Ltabspc : objet spc (lemmes) [spc : frequency spectrum]
* Ftabtfl : objet tfl (formes)
* Ftabvgc : objet vgc (formes)
* Ftabspc : objet spc (formes)

Remarques sur la manière de procéder.

1. Il est indispensable de connaître les éléments principaux du langage CQL pour pouvoir utiliser correctement ces fonctions ; il est hautement préférable de connaître le langage R pour pouvoir lire les fonctions, seule manière de comprendre réellement ce qu'elles font.

2. Si l'on est amené à utiliser le plus souvent tel ou tel corpus, on aura avantage à inscrire le nom de ce corpus dans les lignes de commande, à titre de valeur par défaut, on gagnera ainsi beaucoup de temps ; d'une manière générale, les valeurs par défaut peuvent être modifiées, et divers paramètres, sans valeur par défaut, peuvent en recevoir une très facilement.

3. Ces fonctions comportent quelques messages d'erreur destinés à répondre à des erreurs en entrée ; mais de nombreux cas plus ou moins exotiques ne sont pas prévus, et les fonctions comportent certainement de nombreuses bogues, qui peuvent générer elles aussi des messages d'erreur bruts, à peu près incompréhensibles.

4. Ces fonctions sont destinées à l'exploration ; elles n'ont aucune valeur probatoire, il faut toujours se demander ce que l'on voit sur une liste ou dans un graphique, et revenir autant que nécessaire aux textes eux-mêmes. Surtout, on ne saurait trop insister sur l'absolue nécessité de procéder, pour chaque opération, à de nombreux essais en faisant varier – fortement – les paramètres ; dans la majorité des cas, on tirera plus d'informations de la comparaison des résultats que des résultats eux-mêmes ; en particulier, selon que les résultats sont stables (varient peu si les paramètres varient) ou au contraire varient plus ou moins radicalement en fonction des variations de tel ou tel paramètre.

5. Un usage, même rapide, de ces fonctions permettra de s'apercevoir des lacunes et des insuffisances qu'elles comportent ; dans la plupart des cas, ces fonctions devraient être étendues, d'autres paramètres prévus ; surtout, d'autres outils d'analyse sémantiques devraient être développés, susceptibles, d'une part, de mieux saisir et faire apparaître des champs et des réseaux et, d'autre part, de mieux faire apparaître les évolutions des structures de ces champs. Je place toutes ces fonctions dans le domaine public, dans l'espoir de les voir corrigées, enrichies et développées.

6. On observera notamment l'intérêt d'employer les fonctions `coocA()` et `coocB()` sur des tranches successives d'un corpus pour faire apparaître les évolutions d'un réseau lexical, méthode qui permet en même temps de mieux cerner, globalement, ce réseau lui-même.

AG 10.2014