Documentatie R-scripts

Ten behoeve van data-analyse-project voor DANS, door Emil Bode

Table of Contents

[Helper functions and scripts 3](#_Toc499886581)

[Help.R 3](#_Toc499886582)

[Libinstandload 3](#_Toc499886583)

[ReadForAnalysisFromTotal 3](#_Toc499886584)

[readNARCIScla 3](#_Toc499886585)

[VSNU-pdf.R 4](#_Toc499886586)

[Small ad-hoc files 5](#_Toc499886587)

[AddToVSNUTelling.R 5](#_Toc499886588)

[VSNU-telling.R 5](#_Toc499886589)

[Countdesciption.R 5](#_Toc499886590)

[Check instellingsproefschriften.R 6](#_Toc499886591)

[Background 6](#_Toc499886592)

[Description 6](#_Toc499886593)

[Inner working 6](#_Toc499886594)

[In- and output 6](#_Toc499886595)

[Variants 6](#_Toc499886596)

[DARIAHplot.R and DARIAH-timeline.xlsx 8](#_Toc499886597)

[General description 8](#_Toc499886598)

[Input 8](#_Toc499886599)

[OAI Harvest New Records.R 10](#_Toc499886600)

[Andere globale variabelen, mogelijk pas in latere stappen ingevuld: 12](#_Toc499886601)

[Tijdelijke variabelen 13](#_Toc499886602)

[Uiteindelijke bestanden die worden gebruikt, met aanduiding. 14](#_Toc499886603)

[Opmerkingen 15](#_Toc499886604)

[JustGo 15](#_Toc499886605)

[Start 15](#_Toc499886606)

[harvIDs 15](#_Toc499886607)

[procIDs 15](#_Toc499886608)

[ShowIDsumm 16](#_Toc499886609)

[harvNewRecs 16](#_Toc499886610)

[SummariseRecords 17](#_Toc499886611)

# Helper functions and scripts

Functions are declared in Help.R

## Help.R

### Libinstandload

#### Description

Tries to load requested packages, and installs them if loading fails

#### Arguments

Packages: the needed packages, given as a character vector

Order: Does the order of loading matter? If T, might be a bit slower

Quiet: Suppress standard loading messages? Has no influence on newly installed libraries

### ReadForAnalysisFromTotal

#### Description

Reads the total set of all publications, and assign this to the global variable “Total”. Returns either zero, or a summarizing count. Further options means subsetting selected data

#### Arguments

Summarize: Should a summarizing count be returned? F for return 0, “Only” to just give a summary, without touching the global variable “Total”, otherwise for both setting Total and returning a summary

FilePath: Where to find the set. Auto means the newest file in werkset matching search pattern

ForceReload: The default, False, first checks whether a dataset Total exists, and if the value in Paths$WerksetTotal is the same as the filename that would be loaded. If that is the case, it is assumed that Total is up-to-date. If T, the file is loaded again (which might take a while). Note that this function relies on outside code to assign its output to Total

Silent: Suppress loading messages

DropCols: Schould any columns be dropped?

KeepCols: Give a list of columns to be kept, NULL means all. Overwrites DropCols

KeepSets: Keep only members of certain sets? NULL means keep all records. Note that the collection of all sets does not guarantee to give all records, but NULL does.

KeepMulti: Possibility to give multiple filtering criteria, in the form of a named list, with the names being search fields, with the allowed values.

#### Notes

The script “ReadAnalysisFromTotal.R” automatically assigned a value to TotalVals as well. This script is now deprecated, and exits with an error

### readNARCIScla

#### Description

Reads a pdf containing the NARCIS classification schema

#### Arguments

FilePath: the path to the pdf-file

## VSNU-pdf.R

#### Description

A function to load the results by the VSNU, reading 2 pdf-files and formatting them as two dataframes.

#### Arguments

Pdfs: a character vector of length2, with the expected pdf-files

#### Called by

* AddToVSNUTelling.R
* Plotjes-sept.R

No further documentation

# Small ad-hoc files

## AddToVSNUTelling.R

#### Description

Adds the new info from the KUOZ numbers to the totals of the VSNU, at least tries to do so

#### Status

The totals don’t match, so this is only partially finished

## VSNU-telling.R

#### Assumptions

A globally defined variable TotalVals,

#### Description

Writes two csv files with a summary of results

#### Status

Not being kept up-to-date, no full doc

## Countdesciption.R

#### Description

Counts how many publications have a filled description, output is in console

# Check instellingsproefschriften.R

### Background

This script has 2 purposes: Seeing how many doctoral theses are in NARCIS that are given out by other institutes then universities, and checking if there are copies of these theses that are given out by universities.

But in the future, this script could be extended to check for more duplicates.

### Description

The script reads the total file, selects the theses from these, and compares the titles of the set of the these by universities, with those by non-unviersties. Comparison is done by matching the fill titles, as well as splitting titles in two in comparing these, to make sure that *SomeTitle in English [Een titel in het Nederlands]* matches both *SomeTitle in English* and *Een titel in het Nederlands*.

The results are written to a csv file, and a plot is produced showing the relative number of theses, and the matching rate.

For matching, a score is given as well, where 50% is set at the lowest point in the matching distribution. This gives quite good results, where most of the scores fall in either <20% (match), or >80% (no match), but no upper bound.

### Inner working

At the core, the stringdist package is used to calculate differences between different titles. At the start, all titles are split in 2 parts, based on the first occurrence of any of the following characters: (:.[= This is not perfect, as titles may consist of more than 2 parts, but is sufficient for now.

Then the distances between the full titles are calculated, and the shortest is saved. In the next steps, different combinations of parts and full titles are checked, and if a better match is found, it is stored.

When calculating the distance, a correction is used to compensate for stringlength and method used, First, a method correction is added to the stringdist, and then the distance is multiplied by the square root of the (stringlength – correction), for both strings (so multiplied twice, so that correction ~ average stringlength, hence the sqrt)

### In- and output

Script reads the totalfile as given by ReadForAnalysisfromTotal, and bases its division on the BronCat.

Possible adjustable parameters ae the method used for distance calculation, the q parameter if some distance metrics are used (not used for lcs), and charcorr, the correction factor used to ‘punish’ short titles, as they seems to be overly matched if charcorr=0, and finally the added values for the different permutations.

Output is the csv-file, where original and match can be compared along with a score. The file is sorted on the score (most matching first)

A plot is generated and stored as well

### Variants

This script can be extended to match other publications as well, although for now it is too slow to check all possible permutations.

Possible adaptions include:

* Matching a set with itself, while of course removing the ‘same value’ comparisons.
* Checking for other fields than title alone
* Looking for ways to gain speed

# DARIAHplot.R and DARIAH-timeline.xlsx

### General description

Makes a visualization of different entities, and how they evolve over time, based on an excelsheet with parameters. As those two are interwoven, documentation is combined as well. Note that a lot of documentation is also supplied in the excelsheet.

### Input

See excelsheet, the path is supplied in the start of the script.

### Output

The plot is shown in RStudio, but the definite version is saved on the path given in the excelsheet.

### Portability

I tried to make the code as portable as possible, but this is not fully tested. To run on another system, the first should be deleted, and the path to the excelsheet fully supplied.

Furthermore, note that the function libinstandload is defined *if it does not exist yet* (and in the standard version it is defined at the call to SetLocal). This means care should be taken when changes are made, check whether both versions are still the same.

Finally, the package extrafont is quite OS-dependent. You can specify a path to look for fonts in the excelsheet, but no testing is done, and some fonts can’t be rendered well.

Ten eerste: Er is één script die alleen via terminal gerund kan worden. In andere gevallen runt hij wel, maar geeft hij een foutmelding.

Het runnen kan/moet als volgt gebeuren:

* Open terminal
* Type ulimit –s 32768. De maximaal beschikbare stack wordt daarmee op 32 MB gezet (Standaard en te weinig is bij mijn Mac 8 MB) Andere waarde zou ook kunnen, maar ik weet niet wat de ondergrens is. Bovengrens is 65532 (net iets onder 64 MB), geen idee wat dit verder uitmaakt.
* “R” opent R. De vorige workspace wordt automatisch hersteld. Stond hier veel in, dan kan het lijken alsof R/Terminal blijft hangen.
* Maak evt. je workspace geheel leeg met rm(list=ls())
* source('~/surfdrive/Documents/NARCIS-publicaties-analyse/R-code/OAI Harvest New Records.R')

Verdere opmerking: Het script stopt als files niet op volgorde staan. Dit kan ook gebeuren als er tijdens een harvest een kloksynchronisatie plaatsvindt. Om dit te voorkomen kun je het best synchroniseren voor het begin van een harvest.

Als je toch in terminal zit kan dat als volgt:

Sudo ntpd

Mochten er nog meer fouten zijn, dan kun je de hele boel uitschakelen met

sudo systemsetup -setusingnetworktime off

Vergeet hem niet na afloop weer aan te zetten!

Mocht er toch iets misgaan dan kun je de modified-time van een file als volgt aanpassen:

* Open terminal
* Typ touch -mt 201712312359.59 Full/Path/Filename.xml
* Formaat is [CC]YYMMDDhhmm[.ss]

# OAI Harvest New Records.R

Doel van dit bestand: Het opnieuw binnenhalen (‘harvesten’) van records uit NARCIS of een andere bron via de OAI-PMH interface, en deze omzetten naar een datastructuur waar R mee verder kan werken.

Dit gebeurt in verschillende stappen, hier onder elke omschreven. Verder houdt het systeem bij elke stap bij of het environment sane is, en ruimte evt. oude restanten op.

Sommige stappen zijn verder onderverdeeld in SubSteps. Dit alles zodat je het script kunt onderbreken en opnieuw opstarten, dit werkt dan als een hervatting.

Hervatting kan op elk moment met je environment nog vanuit ditzelfde bestand. Zijn er ondertussen andere programma’s actief geweest, dan kun je het best je environment leeg maken. In dat geval wordt wel hervat op basis van de bestanden op schijf. Bij sommige stappen wordt de hoofdstap opnieuw begonnen, bij anderen wordt fijnmaziger gekeken.

Globale parameters, in het begin opgegeven:

* Paths (list): Variabelen met paden waar bestanden te vinden zijn. Zie bij elke stap
* Params: list. Opgegeven parameters:
  + Cla: Het NARCIS-classificatieschema, opgeslagen voor snelle referentie. Wordt geconstrueerd door help-script ClasSchema.R
  + Summarize: Een lijst met welke samenvattingen getoond moeten worden.
    - ‘Ids’ in lijst: de stap showIDsumm wordt uitgevoerd
    - ‘Records’ in lijst: stap SummariseRecords wordt uitgevoerd
    - Als deze er niet in staan kunnen bovenstaande stappen onder sommige condities toch uitgevoerd worden
    - Merk op dat als deze parameter NULL of c() is, deze ook niet wordt weergegeven
  + reCalcIDs: T of F. Als alle Ids binnen zijn gehaald, en er is nog een resultatenbestand, moet dit dan opnieuw berekend?
  + Oldfdate: Een garantiedatum van de laatste dataset. Ids sinds die datum worden uitgebreid verwerkt, van voor deze datum wordt er vanuit gegaan dat deze kloppen met de records. Zie ook stap ‘procIDs’
  + Harv. Met welk formaat geharvest moet worden: dc of didlmods
  + Urls. De parameters die aan de OAI-PMH interface meegegeven worden
  + Nfiles. De grootte om bestanden samen te vatten. Nfiles xml-bestanden worden tot één pakketje verwerkt
  + Filesize. Het aantal verwachte records in één xml-file, gebruikt voor controledoeleinden
  + Tempvarnames. Een lijstje met namen die in environment voor mogen komen op tijdelijke basis. Worden bij het begin van elke stap weggegooid.
    - Hiernaast is elke naam waar ‘temp’ in zit toegestaan
  + Reclsnames: namen van XML-rootnames, ter controle
  + Dateform: hoe een datum er uitziet
  + Subdfs: welke waarden multi-valued zijn, en dus een eigen dataframe krijgen
  + RegExErrors: de reguliere expressies die worden gebruikt om fouten op te sporen. Merk op dat je deze niet zomaar kunt aanvullen, er zitten in de code ook weer dependencies, zoals de vraag welke regex-engine ze mag interpreteren, of wat voor replacement er gebruikt wordt.
  + ResumeRecordParse. Voor debugging, begin bij het parsen van records bij dit nummer (default=afhankelijk van files). Zie stap parseRecords
  + ResumeRecordMerge. Idem als vorige, maar nu voor de MergeStep

## Andere globale variabelen, mogelijk pas in latere stappen ingevuld:

* IDfiles: Bestanden waarin de opnieuw geharveste Ids worden opgeslagen, gesorteerd op datum. In HarvIDs wordt gekeken welke de nieuwste is, en op basis hiervan wordt hervat (als nodig). Verder wordt in stap ProcIDs een ketting geconstrueerd, om te controleren op juistheid en compleetheid.
* newIDs: Een overzicht met Ids die zijn geharvest, maar nog niet in de oude dataset zaten.
* NewTotal: het eindresultaat
* oldIDs: Een overzicht met de Ids uit de oude dataset. Deze wordt ingelezen vanuit bestand, de aanname is dat deze matcht met de TotalSet. In kolom ‘inNew’ is ingevuld of en hoe deze terugkomt in de nieuwe set Ids, dit gebeurt in stap ‘procIDs’
* Recfiles: overzicht van de xml-bestanden
* Total\_Del: Totaaloverzicht met bestanden die in de nieuwe inlezing staan aangegeven als deleted.
* Total\_Kept: De records uit de oude dataset die meegenomen worden naar de nieuwe.
* Total\_New: Dataset met nieuw ingelezen records, in simpel data.frame-format. Dus zonder multi-valued fields
* Errors: Een lijst met fouten die we zijn tegengekomen:
  + Count: totaaltelling
  + UpToStep: Tot waar zijn de fouten geregistreerd?
  + Longdelays: Zijn er xml-bestanden waarbij het verschil tussen request en antwoord erg groot is (>30 s)? Merk op dat zelfde info voor volledige harvest wordt opgeslagen onder Errors$RecSumm
  + MultiIDs: Zijn er Ids die vaker terugkomen bij het opvragen? Dit kan wel als een record tijdens een harvest wordt geupdatet, dus kan bij erg laat frame terugkomen
  + RecSumm: Overzicht van fouten bij stap ‘SummariseRecords’
  + Recparse: Overzicht van fouten bij stap ‘ParseRecords’
  + SummaryOfRecParse: Samenvatting van vorige
* MetaOut: Een overzicht met gegevens van Total\_new. Nu niet heel consequent of intensief gebruikt.
* Resdate: De datum vanaf wanneer nieuwe records binnengehaald moeten worden. Is gelijk aan de eerste datum in NewIDs, minus 1 dag.
* Total\_New\_Sets: Idem als Total\_New, maar als lijst dataframes voor multi-valued fields

## Tijdelijke variabelen

Deze staan in Params$tempvarnames, sommige zijn wat langer in gebruik:

* Rawfile: een direct ingelezen xml-bestand, als tekst.
* Rawlen: de lengte van vorige (volgens functie nchar())
* Recdf en Deleteddf: Dataframes met tussentijdse resultaten
* ParsDelfiles: Bestandenlijst met samengevatte deleted records.
* ParsRecfiles: Bestandenlijst met samengevatte records.
* Colinnames: een lijstje met namen. In de stap ParseRecords worden de xml-bestanden omgezet in dataframes, die later in MergeRecords weer worden omegzet in een iets vastere structuur. In de variabele colinnames zijn de kolomnamen van de tussen-dataframes vastgelegd, om het mogelijk te maken later veranderingen door te voeren zonder allerlei waarden in de code zelf aan te hoeven passen.
* colsMand: Kolommen die verplicht aanwezig moeten zijn, error als deze ontbreken.
* CummTotals. Een getallenlijstje met cummulatief het aantal records in de xml-bestanden.
* CorrErrors. In de stap SummariseRecords worden fouten gedetecteerd en opgeslagen. In de stap ParseRecords worden deze weer ingelezen en gebruikt om de boel weer te verbeteren. De fouten worden eerst in deze variabele opgeslagen.

## Uiteindelijke bestanden die worden gebruikt, met aanduiding.

Codes: R=wordt alleen ingelezen, RW= lezen en schrijven, W= wordt alleen geschreven

I= Intermediate, tussenresultaat voor bu e.d., O= Output, eindresultaat

Directories voor XML-bestanden en ID-bestanden worden automatisch aangemaakt (W, I)

* IDlist.rds: lijst met oude IDS. **Moet matchen met TotalSet.rds** (R)
* TotalSet.rds: de oude set records. (R)
* Map ..Ids: Overzicht van ID-bestanden (RW, I)
* Map dc/didlmods: de rauwe gedownloade xml-bestanden, met daarin als het goed is alle info. Nog niet verwerkt. (RW, I)
* Map chunkdc: Verder verwerkte ‘chunks’ van xml-bestanden (normaal per 50) (RW,I)
* oldIDs (temp) (*timetamp).*rds: Overzicht met de oude Ids (van IDlist.rds), met extra info of de records opnieuw in de set zitten. (RW,I)
* FileList IDFs

## Opmerkingen

Aandacht is nodig voor de verwerking van tijden, het gaat soms fout met tijdzones. Vooral readRDS gedraagt zich vreemd: identieke calls, met identiek ingestelde Sys.unsetenv(‘TZ’), levert verschillende resultaten.

De verschillende stappen:

# JustGo

Standaardinstelling, bedoeld om te kijkenwaar we gebleven zijn en dit te hervatten. Stelt geen eisen aan environment. Pakt alle bestanden uit summary-directory met (temp), en vult op basis hier van de variabelen oldIDs, IDfiles, newIDs en Errors. Verder laadt hij de oude dataset, en bekijkt vanaf wanneer records geharvest moeten worden.

# Start

Bedoeld voor een hele nieuwe start. Gooit oude environment weg, en begint met nieuwe harvest.

# harvIDs

Kijkt tot hoe ver we zijn gekomen met Ids harvesten, en hervat dit.

* Stap 1 is kijken op de harde schijf, wat de nieuwste file is. Bij een tie worden de requestTokens op een rijtje gezet, en als er geen match is dat de laatste file.
* In lastResToken wordt de ‘resumptionToken’ opgeslagen waarmee we moeten hervatten, of ‘Endfile’ als we alles gehad hebben.
* Daarna wordt de OAI-PMH interface aangeroepen, deze werkt volautomatisch. Om voortgang bij te houden kun je het beste in Finder/verkenner kijken hoe en wanneer er files bijkomen.

Deze stap kan vanuit ‘JustGo’ worden overgeslagen

Als er nieuwe Ids zijn binnengehaald is de volgende stap procIDs, anders hangt het van de waarden in Params af of we naar ‘procIDs’, ‘showIDsumm’ of gelijk ‘harvNewRecords’ gaan.

# procIDs

Deze stap maakt een lijst met wat meer gegevens over de IDfiles, en vergelijkt met de oude Ids

* Eerst aanmaak van schone lege variabelen
* Dan een loop die alle files leest en info invult en controleert (dubbel om elke 100 even een melding te geven). Merk op dat binnen deze loop ‘NoRecords’ nog het totaal aantal in een file is, dus ‘echte’ records en deleted. Op het eind van de loop wordt dit gecorigeerd
* Daarna wordt de volgorde bepaald: de ketting van EndRequestToken en RequestTokens wordt gevolgd, en gecontroleerd
* Dan worden de relatief nieuwe Ids die zijn binnengehaald vergeleken met de oudere Ids: ze worden gematcht, en aan de hand hiervan wordt bij oldIDs de nieuwe status ingevuld (plus verwijzing naar nieuwe file), en degene die er nog niet in stonden worden bij newIDs opgeslagen.
  + Deze loop gaat best langzaam, daarom is ook wat aandacht besteed aan optimalisatie. tempCacheVals slaat de wijzigingen in oldIDs eerst op, en deze wordt periodiek weggeschreven (per ong. 10000 records). Op dit moment gaat de meeste tijd nog naar de substrings uit tekst trekken. Een melding wordt gegeven als de cache wordt weggeschreven.
* Voor de oudere Ids wordt gekeken of de aantallen kloppen, tbv de snelheid. De eerste en laatste worden met oldIDs gematcht, als het gaat hiertussen klopt wordt aangenomen dat de rest ook klopt.

# ShowIDsumm

We gaan samenvatten wat we net hebben opgehaald, en hoe dat zich verhoudt tot de oude records. Deze stap is ook bedoeld om zelf wat in rond te struinen.

* In cnt wordt een telling gemaakt van zowel oude als nieuwe Ids, per repository gekeken hoeveel van elke status er nu zijn. Uitgesplitst per repo, wel/niet dataset, wel/niet beschikbaar in dc-formaat, en wat de status dus is.
* Er wordt een plotje geprint van alle nieuwe Ids, hoe die zich verhouden met de oude dataset. Warnings over non-finite of missing values mogen genegeerd. Merk wel op dat de datumrange handmatig ingevuld is.
* Dan een plotje met wat er met de oude Ids gebeurd is. Ook hier mogen missing values-warnings genegeerd worden. Ook hier handmatige datumrange. En verder betekent een waarde ‘updated’ dat er een nieuwe timestamp is meegegeven, niet per se dat de inhoud ook is geupdatet.
* Tot slot een aantal waarden:
  + Het eerste tijdstip van een nieuw record,
  + Het eerste tijdstip van een oud record met veranderde status (deze zouden indicatoren moeten zijn voor vanaf wanneer we moeten harvesten, deze worden ook gebruikt als basis voor resdate
  + Een telling met hoeveel datasets en publicaties welke status hebben

# harvNewRecs

In deze stap harvesten we de nieuwe records.

* Eerst wordt gekeken of we ergens kunnen hervatten omdat er al files zijn. In dat geval zoeken we de resumptiontoken op en hervatten hiervandaar weer.
* Zo niet, dan beginnen we met harvesten vanaf de datum *resdate*, die eerder is uitgerekend.
* Slotregels zijn kijken wat de volgende stap gaat worden.

De lokale variabel newRecords houdt bij of er geharvest is. Zo ja, dan moet iig de stap SummariseRecords worden uitgevoerd

# SummariseRecords

Deze stap is bedoeld om te kijken wat we binnengehaald hebben aan records. **Is uitgebreider dan de naam doet vermoeden.**

Verder zitten er de nodige optimalisatiestappen in tbv de snelheid.

In deze stap worden de variabelen RecFiles en DelFiles met info gevuld, en de Errorslijst speelt een belangrijke rol, omdat hierin alle nodige aanpassingen in de xml-bestanden worden opgeslagen.

* Eerst doen we weer een environmentcheck, en kijken of we moeten initialiseren.
* Daarna loopen we over alle files. Ten behoeve van de snelheid schrijven we niet alle resultaten direct weg, maar slaan we deze op in een cache-variabele: tempCacheRecs.
  + In tempstrcount wordt geteld hoeveel fouten er zijn in het xml-bestand. Voor elk type fout wordt gekeken waar deze precies staat, wat er fout is, en met welke waarde we de fout zouden kunnen vervangen. Dit is deels om wat mooiere waarden te krijgen, maar bovendien om er voor te zorgen dat we valide xml-bestanden krijgen.
    - Merk op dat de waarde idx wordt geteld per fout, nadat eerder fouten zijn verwijderd. Voorbeeld (X is een fout): aaXbbbXcXX geeft indices 3, 6, 7, 7 als X wordt vervangen door “”.
    - Merk ook op dat de gewijzigde xml-file niet wordt opgeslagen, maar de inhoud van de Error-list in de volgende stap wordt gebruikt
  + Daarna wordt er informatie uit de xml-getrokken:
    - ReqDelay: Verschil tussen request-timestamp en moment van laatste wijziging xml-bestand.