# Custom scripts for miRToolsGallery

[Step 1: R code for extracting article information from PubMed 2](#_Toc500755220)

[Step 2: Perl code for parsed the XML output by Step 1 3](#_Toc500755221)

[Step 3: Perl code for constructing the network of tools base on Step 2 4](#_Toc500755222)

[Step 4: R code for PageRank the tools 5](#_Toc500755223)

## Step 1: R code for extracting article information from PubMed

# install.packages("rentrez")

library('rentrez')

##########################################################

# functions

##########################################################

## make dir

make.dir<-function(path){

 if(file.exists(path)){

 # print(paste(path," is exists!"))

 return (F)

 } else{

 dir.create(path)

 return (T)

 }

}

isEmptyFile <- function( file ){

 if( file.info(file)$size < 5){

 return (T)

 } else{

 return (F)

 }

}

## get references or cited paper PMID for input literature

get.refs.and.citedin <- function(PMID){

 tryCatch({

 hox\_data <- entrez\_link(db="pubmed", id = PMID, dbfrom="pubmed")

 },

 warning = function(w) {

 print (paste(PMID,"->","Warning!\n",sep=""))

 Sys.sleep(abs(rnorm(1)) + 10)

 hox\_data <- entrez\_link(db="pubmed", id = PMID, dbfrom="pubmed")

 }, error = function(e) {

 print (paste(PMID,"->","Errors!\n",sep=""))

 Sys.sleep(abs(rnorm(1)) + 15)

 hox\_data <- entrez\_link(db="pubmed", id = PMID, dbfrom="pubmed")

 }, finally = {

 data <- list(citedin = hox\_data$links$pubmed\_pubmed\_citedin , refs = hox\_data$links$pubmed\_pubmed\_ref)

 }

 )

 return (data)

}

get.info.xml <- function(PMID){

 data <- entrez\_fetch(db = "pubmed",

 id = PMID,

 rettype = "xml")

 return (data)

}

# This R script could be running on different platform

# use place to choose the system

# place : Linux or Windows

place = "Linux"

if(place != "Linux"){

 output.path = "path\\to\\output\\folder"

 input.PMID.file = "path\\to\\PMID\\list\\file"

 separator = "\\"

} else{

 output.path = "path/to/output/folder"

 input.PMID.file = "path/to/PMID/list/file"

 separator = "/"

}

data <- read.table(file = input.PMID.file, header = F, sep="\t")

PMIDS <- unique(data$V1)

# showing the version we used

sink(paste(output.path,separator,"DB\_build\_version\_and\_Messages.log", sep=""),append = FALSE, type=c("output", "message"))

entrez\_db\_summary("pubmed")

###############################################

# download all the artical in the same xml

###############################################

count = length(PMIDS)

output.all.in.one.file = paste(output.path,separator,"all.in.one.", count, ".xml", sep="")

if(!file.exists(output.all.in.one.file)){

 step = 200

 start <- seq(1, ceiling(count/step)\*step,by=step)

 end <- seq(step,floor(count/step)\*step, by=step)

 end <- c(end, count)

 xml.str = ""

 for(i in c(1:length(start))){

 subset.PMIDS = PMIDS[start[i]:end[i]]

 data <- entrez\_fetch(db = "pubmed",

 id = subset.PMIDS,

 rettype = "xml")

 xml.str <- paste(xml.str,data,sep = "\n")

 }

 write(xml.str, file = output.all.in.one.file )

}

for( PMID in PMIDS){

 dir.path <- paste(output.path,separator,"citations\_references",separator,PMID,sep="")

 if(make.dir(dir.path)){

 }else{

 print(paste(dir.path," is exists!"))

 next

 }

 print(dir.path)

 refs.citedin <- get.refs.and.citedin(PMID)

 # write references

 Sys.sleep(abs(rnorm(1)) + abs(rnorm(1)) + 5)

 refs.file = paste(dir.path,separator,PMID,".refs",sep="")

 write(refs.citedin$refs, file = refs.file)

 # write citedin

 citedin.file = paste(dir.path,separator,PMID,".citedin",sep="")

 write(refs.citedin$citedin, file = citedin.file)

}

sink()

## Step 2: Perl code for parsed the XML output by Step 1

#!/usr/bin/perl

use String::Util qw(trim);

my $path = "path\\to\\Step1\\output\\folder";

my $XML\_FILE = "path\\to\\Step1\\output\\folder\\all.in.one.xml";

my $COLLECTION\_FILE = "path\\to\\input\\folder\\INPUT.DatabaseCollection.txt";

my $BLACKLIST\_JOURNAL\_FILE = "path\\to\\input\\folder\\INPUT.JournalBlacklist.txt";

my $BLACKLIST\_YEAR\_FILE = "path\\to\\input\\folder\\INPUT.PMID2YEARBlacklist.txt";

my $output\_file = $XML\_FILE.".parsed.txt";

my $output4R = $XML\_FILE.".parsed.4.R.txt";

my ($tp1, $tp2) = initialHashs($COLLECTION\_FILE);

my %PMID2NAME = %{$tp1};

my %PMID2LINKS = %{$tp2};

my %LongName2ShortName = %{initialJournalHash($BLACKLIST\_JOURNAL\_FILE)};

my %PMID2YEAR = %{initialPMID2YearHash($BLACKLIST\_YEAR\_FILE)};

open OFILE , ">".$output\_file or die ("Sorry!\n");

open OFILE4R , ">".$output4R or die ("Sorry!\n");

my $xml = load\_total\_xml($XML\_FILE);

# remove all enter "\n"

 $xml =~ s/\n//g;

# parsed all articals

my @articals\_xml = ($xml =~ /<PubmedArticle>(.\*?)<\/PubmedArticle>/g);

my $num = @articals\_xml;

print OFILE4R "PMID\tName\tTitle\tYear\tJournal\tISSN\tCitation count\n";

foreach my $artical\_xml (@articals\_xml){

 my $PMID = get\_PMID($artical\_xml);

 my $citedin\_file = $path."\\".$PMID."\\".$PMID.".citedin";

 my $refs\_file = $path."\\".$PMID."\\".$PMID.".refs";

 my @citedin\_refs = @{get\_PMID\_list($citedin\_file)};

 my @refs = @{get\_PMID\_list($refs\_file)};

 my $title = get\_paper\_title($artical\_xml);

 my @keywords = @{get\_keywords($artical\_xml)};

 my $article = get\_article($artical\_xml);

 my @abstrcts = @{get\_abstract($article)};

 my $journal = get\_journal($article);

 my $journal\_info = get\_journal\_info($journal, $PMID, \%PMID2YEAR, \%LongName2ShortName);

 my $citation\_num = 0;

 if($citedin\_refs[0] ne ""){

 $citation\_num = @citedin\_refs;

 }

 my $name = $PMID2NAME{$PMID};

 if($name eq ""){

 $name = getToolName($title);

 }

 my $links = $PMID2LINKS{$PMID};

 print OFILE $PMID."\t".$name."\t".$title."\t".$links."\t".join(";",@keywords)."\t".join("<br>",@abstrcts)."\t".$journal\_info."\t".$citation\_num."\t".join(";",@citedin\_refs)."\t".join(";",@refs)."\n";

 print OFILE4R $PMID."\t".$name."\t".$title."\t".$journal\_info."\t".$citation\_num."\n";

}

print "There are ".$num." articals in the xml!\n";

close OFILE;

close OFILE4R;

print "Done!\n";

sub initialJournalHash{

 my $file = $\_[0];

 my %LongName2ShortName = ();

 open FILE , "<".$file or die ("Sorry!($file)\n");

 #remove header

 readline(FILE);

 while(<FILE>){

 my @temp = split /\t/ , $\_ ;

 my $LongName = trim($temp[1]);

 my $ShortName = trim($temp[0]);

 if(defined $LongName2ShortName{$LongName}){

 print "$LongName is duplicated!\n";

 }

 $LongName2ShortName{$LongName} = $ShortName;

 }

 close FILE;

 return (\%LongName2ShortName);

}

sub initialPMID2YearHash{

 my $file = $\_[0];

 my %PMID2YEAR = ();

 open FILE , "<".$file or die ("Sorry!($file)\n");

 readline(FILE);

 while(<FILE>){

 my @temp = split /\t/,trim($\_);

 my $pmid = $temp[0];

 my $year = $temp[1];

 $PMID2YEAR{$pmid} = $year;

 }

 close FILE;

 return (\%PMID2YEAR);

}

sub initialHashs{

 my $file = $\_[0];

 my %PMID2name = ();

 my %PMID2links = ();

 open FILE , "<".$file or die ("Sorry!($file)\n");

 #remove header

 readline(FILE);

 while(<FILE>){

 my @temp = split /\t/ , trim($\_) ;

 my $PMID = trim($temp[0]);

 my $name = trim($temp[1]);

 my $links = trim($temp[3]);

 if(defined $PMID2name{$PMID}){

 print "$PMID is duplicated!\n";

 }

 $PMID2name{$PMID} = $name;

 $PMID2links{$PMID} = $links;

 }

 close FILE;

 return (\%PMID2name,\%PMID2links);

}

sub load\_total\_xml{

 my $file = $\_[0];

 open FILE , "<".$file or die ("Sorry can not open $file!\n");

 local $/=undef;

 my $xml = <FILE>;

 close FILE;

 return $xml;

}

sub get\_PMID{

 my $xml = $\_[0];

 my $PMID = "";

 if( $xml =~ /<PMID.\*?>(.\*?)<\/PMID>/g){

 $PMID =$1;

 }

 return $PMID;

}

sub get\_article{

 my $xml = $\_[0];

 my $article = "NULL";

 if( $xml =~ /<Article.\*?>(.\*)<\/Article>/g ){

 $article = $1;

 }

 return $article;

}

sub get\_journal{

 my $xml = $\_[0];

 my $journal = "NULL";

 if( $xml =~ /<Journal>(.\*)<\/Journal>/g ){

 $journal = $1;

 }

 return $journal;

}

sub get\_paper\_title{

 my $xml = $\_[0];

 my $title = "NULL";

 if( $xml =~ /<ArticleTitle>(.\*)<\/ArticleTitle>/g ){

 $title = $1;

 }

 return $title;

}

sub getToolName{

 my $title = $\_[0];

 my $name = "";

 my @temp = split /:|\-{2}/,$title ;

 if(@temp >= 2){

 $name = $temp[0];

 }

 return $name;

}

sub get\_abstract{

 my $xml = $\_[0];

 my @abstrcts ;

 my $list = "NULL";

 if( $xml =~ /<Abstract.\*?>(.\*)<\/Abstract>/g ){

 $list = $1;

 }

 while($list =~ /<AbstractText.\*?>(.\*?)<\/AbstractText>/g){

 push @abstrcts, $1;

 }

 return \@abstrcts;

}

sub get\_keywords{

 my $xml = $\_[0];

 my @keywords ;

 my $list = "NULL";

 if( $xml =~ /<KeywordList.\*?>(.\*)<\/KeywordList>/g ){

 $list = $1;

 }

 @keywords = ($list =~ /<Keyword.\*?>(.\*?)<\/Keyword>/g );

 return \@keywords;

}

sub get\_journal\_info{

 my $xml = $\_[0];

 my $pmid = $\_[1];

 my %PMID2YEAR = %{$\_[2]};

 my %LongName2ShortName = %{$\_[3]};

 my $ISSN = "NULL";

 my $title = "NULL";

 my $year = "NULL";

 if( $xml =~ /<ISSN.\*?>(.\*)<\/ISSN>/g ){

 $ISSN = $1;

 }

 if( $xml =~ /<Year>(.\*)<\/Year>/g ){

 $year = $1;

 }

 if( $xml =~ /<Title>(.\*)<\/Title>/g ){

 $title = $1;

 }

 if(defined $LongName2ShortName{$title}){

 $title = $LongName2ShortName{$title};

 }

 if( $year eq "NULL"){

 if(defined $PMID2YEAR{$pmid} ){

 $year = $PMID2YEAR{$pmid};

 } else{

 print "Please put $pmid into blacklist (INPUT.PMID2YEARBlacklist.txt) !\n";

 }

 }

 return $year."\t".$title."\t".$ISSN;

}

sub get\_PMID\_list{

 my $file = $\_[0];

 my %PMID ;

 open FILE , "<".$file or die ("Sorry can not open $file!\n");

 while(<FILE>){

 my $id = trim($\_);

 $PMID{$id} = $id;

 }

 close FILE;

 my @PMIDS = keys %PMID ;

 return \@PMIDS;

}

## Step 3: Perl code for constructing the network of tools base on Step 2

#!/usr/bin/perl

use String::Util qw(trim);

my $path = "path\\to\\Step1\\output\\folder";

my $PMID\_FILE = "path\\to\\input\\folder\\INPUT.PMID.txt";

my $TOOLS\_FILE = "path\\to\\input\\folder\\INPUT.DatabaseCollection20170909.txt";

my $PMID2YEAR\_FILE = "path\\to\\Step1\\output\\folder\\all.in.one.xml.parsed.4.R.txt";

my $output\_tool2pmids\_file = "path\\to\\Step2\\output\\folder\\tool2pmids\_and\_citedinpmids.txt";

my $output\_citedin\_file = "path\\to\\Step2\\output\\folder\\citedin.network.txt";

my $output\_refs\_file = "path\\to\\Step2\\output\\folder\\refs.network.txt";

my $node\_attribute\_file = "path\\to\\Step2\\output\\folder\\node.attribute.txt";

my $output\_paper\_network\_file = "path\\to\\Step2\\output\\folder\\refs.network.trimed.txt";

my $output\_tool\_paper\_citation\_network\_file = "path\\to\\Step2\\output\\folder\\citation.network.trimed.txt";

my @PMIDS = @{get\_PMID\_list($PMID\_FILE)};

print "There are ".@PMIDS." papers!\n" ;

my %TOOL\_PMID = ();

# one tool could have many publications

# so one tool could have several PMIDS

my %TOOL2PMIDS = ();

%TOOL2PMIDS = %{initHashTOOL2PMIDS($TOOLS\_FILE)};

%PMID2PMIDS = %{initHashPMID2PMIDS(\%TOOL2PMIDS)};

my %TOOL2CITEDINPMIDS = ();

my %PMID2YEAR = ();

%PMID2YEAR = %{initHashTOOLS2YEAR($PMID2YEAR\_FILE)};

foreach my $id (@PMIDS){

 $TOOL\_PMID{$id} = "TOOL";

}

my %OTHER\_PMID = ();

foreach my $tool (keys %TOOL2PMIDS){

# print "Parsing ".$tool."\n";

 my @PMIDS = @{$TOOL2PMIDS{$tool}};

 my ($tp1, $tp2) = get\_Tool\_PMIDS\_list($path,\@PMIDS);

 my %PMIDS\_CITEDIN\_TOTAL = %{$tp1};

 my %PMIDS\_REFS\_TOTAL = %{$tp2};

 my @PMIDS\_CITEDIN = keys %PMIDS\_CITEDIN\_TOTAL;

 my @PMIDS\_REFS = keys %PMIDS\_REFS\_TOTAL;

 $TOOL2CITEDINPMIDS{$tool} = \@PMIDS\_CITEDIN;

}

open OFILE\_CITEDIN , ">".$output\_citedin\_file or die ("Sorry!\n");

open OFILE\_REFS , ">".$output\_refs\_file or die ("Sorry!\n");

open OFILE\_NETWORK, ">".$output\_paper\_network\_file or die ("Sorry!\n");

open OFILE\_CITATION, ">".$output\_tool\_paper\_citation\_network\_file or die ("Sorry!\n");

%CITATIONS = ();

foreach my $id (@PMIDS){

# print "Parsing ".$id."\n";

 my $citedin\_file = $path."\\".$id."\\".$id.".citedin";

 my $refs\_file = $path."\\".$id."\\".$id.".refs";

 my @PMIDS\_CITEDIN = @{get\_PMID\_list($citedin\_file)};

 my @PMIDS\_REFS = @{get\_PMID\_list($refs\_file)};

 my $source = "";

 my $target = "";

 my $interaction = "";

 foreach my $pmid (@PMIDS\_CITEDIN){

 $source = $pmid;

 $target = $id ;

 if( ! defined $TOOL\_PMID{$pmid} ){

 $OTHER\_PMID{$pmid} = "OTHER";

 }

 if($pmid ne ""){

 my $link = "0";

 if(defined $TOOL\_PMID{$pmid} ){

 $link = "1";

 ######################################

 $tp = $source."\t".$link."\t".$target;

 $tp = renameEdge(\%PMID2PMIDS, $tp);

 if( ! defined $CITATIONS{$tp} ){

 $CITATIONS{$tp} = 1 ;

 }

 ######################################

 }

 $tp = $source."\t".$link."\t".$target;

 $tp = renameEdge(\%PMID2PMIDS, $tp);

 print OFILE\_CITEDIN $tp."\n";

 }

 }

 foreach my $pmid (@PMIDS\_REFS){

 $source = $id;

 $target = $pmid ;

 if( ! defined $TOOL\_PMID{$pmid}){

 $OTHER\_PMID{$pmid} = "OTHER";

 }

 if($pmid ne ""){

 my $link = "0";

 if(defined $TOOL\_PMID{$pmid} ){

 $link = "1";

 $tp = $source."\t".$link."\t".$target;

 $tp = renameEdge(\%PMID2PMIDS, $tp);

 print OFILE\_NETWORK $tp."\n";

 ######################################

 $tp = $source."\t".$link."\t".$target;

 $tp = renameEdge(\%PMID2PMIDS, $tp);

 if( ! defined $CITATIONS{$tp} ){

 $CITATIONS{$tp} = 1 ;

 }

 ######################################

 }

 $tp = $source."\t".$link."\t".$target;

 $tp = renameEdge(\%PMID2PMIDS, $tp);

 print OFILE\_REFS $tp."\n";

 }

 }

}

close OFILE\_CITEDIN;

close OFILE\_REFS;

close OFILE\_NETWORK;

foreach my $citation( keys %CITATIONS ){

 print OFILE\_CITATION $citation."\n";

}

close OFILE\_CITATION;

saveTOOL2PMIDS($output\_tool2pmids\_file,\%TOOL2PMIDS,\%TOOL2CITEDINPMIDS, \%PMID2YEAR);

open FILE , ">". $node\_attribute\_file or die ("Sorry!\n");

foreach my $id (keys %TOOL\_PMID ){

 print FILE $id."\t".$TOOL\_PMID{$id}."\n";

}

foreach my $id (keys %OTHER\_PMID ){

 print FILE $id."\t".$OTHER\_PMID{$id}."\n";

}

close FILE;

print "Done!\n";

sub renameEdge{

 my $PMID2PMIDS = $\_[0];

 my $edge = $\_[1];

 my @temp = split /\t/, $edge ;

 my $source = "";

 if(! defined $PMID2PMIDS{$temp[0]} ){

 $source = $temp[0];

 } else {

 $source = $PMID2PMIDS{$temp[0]};

 }

 my $link = $temp[1];

 my $target = $PMID2PMIDS{$temp[2]};

 if(! defined $PMID2PMIDS{$temp[2]} ){

 $target = $temp[0];

 } else {

 $target = $PMID2PMIDS{$temp[2]};

 }

 my $newedge = "".$source."\t".$link."\t".$target;

 return ($newedge);

}

sub initHashPMID2PMIDS{

 my $TOOL2PMIDS = $\_[0];

 my %PMID2PMIDS = ();

 foreach my $toolname ( keys %$TOOL2PMIDS ){

 my $temp = $TOOL2PMIDS{$toolname} ;

 my $name = join(";", @$temp);

 foreach my $pmid ( @$temp ){

 if( defined $PMID2PMIDS{$pmid} ){

 print $pmid." is duplicated!\n" ;

 }

 $PMID2PMIDS{$pmid} = $name;

 #print $pmid." -> ". $name."\n";

 }

 }

 return (\%PMID2PMIDS);

}

sub get\_Tool\_PMIDS\_list{

 my $path = $\_[0];

 my @PMIDS = @{$\_[1]};

 my %PMIDS\_CITEDIN\_TOTAL = ();

 my %PMIDS\_REFS\_TOTAL = ();

 foreach my $id (@PMIDS){

 my $citedin\_file = $path."\\".$id."\\".$id.".citedin";

 my $refs\_file = $path."\\".$id."\\".$id.".refs";

 my @PMIDS\_CITEDIN = @{get\_PMID\_list($citedin\_file)};

 my @PMIDS\_REFS = @{get\_PMID\_list($refs\_file)};

 foreach my $tp (@PMIDS\_CITEDIN){

 $PMIDS\_CITEDIN\_TOTAL{$tp} = $tp;

 }

 foreach my $tp (@PMIDS\_REFS){

 $PMIDS\_REFS\_TOTAL{$tp} = $tp;

 }

 }

 return (\%PMIDS\_CITEDIN\_TOTAL,\%PMIDS\_REFS\_TOTAL);

}

sub get\_PMID\_list{

 my $file = $\_[0];

 my %PMID = () ;

 open FILE , "<".$file or die ("Sorry can not open $file!(fun get\_PMID\_list)\n");

 while(<FILE>){

 my $id = trim($\_);

 if($id eq ""){

# print "pmid is null string!\n";

 } else{

 $PMID{$id} = $id;

 }

 }

 close FILE;

 my @PMIDS = keys %PMID ;

 return \@PMIDS;

}

sub initHashTOOLS2YEAR{

 my $file = $\_[0];

 my %PMID2YEAR = ();

 open FILE , "<".$file or die ("Sorry!($file)\n");

 readline(FILE);

 while(<FILE>){

 my @temp = split /\t/,trim($\_);

 my $pmid = trim($temp[0]);

 my $year = trim($temp[3]);

 $PMID2YEAR{$pmid} = $year;

 }

 close FILE;

 return (\%PMID2YEAR);

}

sub initHashTOOL2PMIDS{

 my $file = $\_[0];

 my %TOOL2PMIDS = ();

 open FILE , "<".$file or die ("Sorry can not open $file!(fun initHashTOOL2PMIDS)\n");

 # remove headers

 readline(FILE);

 while(<FILE>){

 my @temp = split /\t/,$\_;

 my $PMID = trim($temp[0]);

 my $TOOL = trim($temp[1]);

 if( $TOOL eq ""){ print $PMID."tool title is null!\n" };

 if( defined $TOOL2PMIDS{$TOOL} ){

 push $TOOL2PMIDS{$TOOL}, $PMID;

 } else{

 my @tp;

 push @tp, $PMID;

 $TOOL2PMIDS{$TOOL} = \@tp;

 }

 }

 close FILE;

 return (\%TOOL2PMIDS);

}

sub saveTOOL2PMIDS{

 my $file = $\_[0];

 my %TOOL2PMIDS = %{$\_[1]};

 my %TOOL2CITEDINPMIDS = %{$\_[2]};

 my %PMID2YEAR = %{$\_[3]};

 open FILE, ">".$file or die ("Sorry can not write $file!\n");

 # print headers

 print FILE "Tool Name\tLatest Publication Year\tPublication Count\tPMIDS\tTotal Citation Count\tCitedin PMIDS\n";

 my @tools = keys %TOOL2PMIDS;

 foreach $tool (@tools){

 my @temp = sort(@{$TOOL2PMIDS{$tool}});

# print "-->". join ("---",@temp) . ">".$temp[-1]."\n";

 my $max\_pmid = $temp[-1] ;

 my $latest\_pub\_year = $PMID2YEAR{$max\_pmid};

 print FILE $tool."\t".$latest\_pub\_year."\t".@{$TOOL2PMIDS{$tool}}."\t". join(";",@{$TOOL2PMIDS{$tool}})."\t".

 @{$TOOL2CITEDINPMIDS{$tool}}."\t".join(";", @{$TOOL2CITEDINPMIDS{$tool}})."\n";

 }

 close FILE;

 print "$file have been saved!\n";

}

## Step 4: R code for PageRank the tools

# install.packages('igraph')

# install.packages('reshape')

library(igraph)

library(reshape)

network.files <- c('citedin.network.txt', 'refs.network.trimed.txt','refs.network.txt')

path = "path\\to\\Step2\\output\\folder"

for( file in network.files){

 network.file <- paste(path, file, sep="")

 output.file <- paste(path,file,".PageRanked.csv",sep="")

 network <- read.table(network.file, header = F, sep="\t")

 network <- data.frame(source.node = network$V1, target.node = network$V3)

 vector.network <- as.vector(t(network))

 g <- make\_graph(as.character(vector.network) , directed = T)

 pagerank.score <- page\_rank(g)$vector

 pagerank.score <- data.frame(node = names(pagerank.score), score = pagerank.score)

 pagerank.score <- pagerank.score[order(pagerank.score$score, decreasing=T),]

 write.csv(pagerank.score, file = output.file)

}