Revisions: New search string for systematic search

```r
library(tidyverse)
```

```
## Warning: package 'stringr' was built under R version 4.2.3
```

```r
library(readxl)
```

Titles screening

We first want to remove all studies we have looked at in our previous search.

Load data sheets.

```r
# previous literature search
google <- read_csv("google.csv")
scopus <- read_csv("scopus.csv")
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```r
# Literature
included <- read_csv("../data/cleaned_before_revisions.csv") %>%
  # control conditions only
  filter(condition == "control") %>%
  group_by(paperID) %>%
  summarize(reference = unique(reference)) %>%
  arrange(paperID) %>%
  ungroup()
excluded <- read_csv("excluded_before_revisions.csv")

# revisions literature search
revisions_google <- read_csv("revisions_scholar.csv", col_types = cols(.default = col_character()))
revisions_scopus <- read_csv("revisions_scopus.csv", col_types = cols(.default = col_character()))

options(width = 10000)

revisions_scopus  %>% slice(1) %>% select(Abstract) %>% print(width = Inf)
```

```
## # A tibble: 1 x 1
##   Abstract
##   <chr>
## 1 False news articles pose a serious challenge in today's information landscape, impacting public op
```

```r
revisions_scopus  %>% slice(1) %>% select(Abstract) %>% pull
```

```
## [1] "False news articles pose a serious challenge in today's information landscape, impacting public
```

Modify string format.

```r
# little helper function
make_strings_more_compatible <- function (string) {
  string <- tolower(string)   # no more capital letters
  string <- trimws(string)    # no more white spaces at the end
}

# edit strings
google <- google %>%
  mutate(across(Title, make_strings_more_compatible))

scopus <- scopus %>%
  mutate(across(Title, make_strings_more_compatible))

included <- included %>%
  mutate(across(reference, make_strings_more_compatible))

excluded <- excluded %>%
  mutate(across(reference, make_strings_more_compatible))

revisions_google <- revisions_google %>%
  mutate(across(Title, make_strings_more_compatible))

revisions_scopus <- revisions_scopus %>%
  mutate(across(Title, make_strings_more_compatible))
```

Identify duplicates between revisions and prior search

```r
# extract prior search results
prior_search_doi <- c(scopus$DOI, google$DOI )
prior_search_titles <- c(scopus$Title, google$Title)

# identify duplicates from literature searches
revisions_google <- revisions_google %>%
  mutate(duplicate_by_DOI = ifelse((DOI %in% prior_search_doi) & !is.na(DOI), TRUE, FALSE),
         duplicate_by_Title = ifelse((Title %in% prior_search_titles), TRUE, FALSE),
         duplicate_by_either = ifelse((duplicate_by_DOI | duplicate_by_Title) == TRUE, TRUE, FALSE)
         )

# Overview of different matching methods
revisions_google %>% summarise(duplicate_by_DOI = sum(duplicate_by_DOI),
                     duplicate_by_Title = sum(duplicate_by_Title),
                     duplicate_by_either = sum(duplicate_by_either))
```

```
## # A tibble: 1 x 3
##   duplicate_by_DOI duplicate_by_Title duplicate_by_either
##              <int>              <int>               <int>
## 1              179                343                 353
```

```r
# store the duplicates with prior search
duplicates_prior_google <- sum(revisions_google$duplicate_by_either)
```

```r
# identify duplicates from literature searches
revisions_scopus <- revisions_scopus %>%
  mutate(duplicate_by_DOI = ifelse((DOI %in% prior_search_doi) & !is.na(DOI), TRUE, FALSE),
         duplicate_by_Title = ifelse((Title %in% prior_search_titles), TRUE, FALSE),
         duplicate_by_either = ifelse((duplicate_by_DOI | duplicate_by_Title) == TRUE, TRUE, FALSE)
         )

# Overview of different matching methods
revisions_scopus %>% summarise(duplicate_by_DOI = sum(duplicate_by_DOI),
                     duplicate_by_Title = sum(duplicate_by_Title),
                     duplicate_by_either = sum(duplicate_by_either))
```

```
## # A tibble: 1 x 3
##   duplicate_by_DOI duplicate_by_Title duplicate_by_either
##              <int>              <int>               <int>
## 1              607                637                 655
```

```
# store the duplicates with prior search
duplicates_prior_scopus <- sum(revisions_scopus$duplicate_by_either)
```

Add duplicates between the two revisions literature searches

```
# identify duplicates from literature searches
revisions_google <- revisions_google %>%
  mutate(duplicate_by_DOI = ifelse((DOI %in% revisions_scopus$DOI) & !is.na(DOI), TRUE, duplicate_by_DOI
         duplicate_by_Title = ifelse((Title %in% revisions_scopus$Title), TRUE, duplicate_by_Title),
         duplicate_by_either = ifelse((duplicate_by_DOI | duplicate_by_Title) == TRUE, TRUE, duplicate_
         )

# Overview of different matching methods
revisions_google %>% summarise(duplicate_by_DOI = sum(duplicate_by_DOI),
                     duplicate_by_Title = sum(duplicate_by_Title),
                     duplicate_by_either = sum(duplicate_by_either))
```

```
## # A tibble: 1 x 3
##   duplicate_by_DOI duplicate_by_Title duplicate_by_either
##              <int>              <int>               <int>
## 1              229                419                 438
```

```
# store duplicates between searches (after removing) prior search results
duplicates_revision <- sum(revisions_google$duplicate_by_either) - duplicates_prior_google
```

We remove all duplicates and store the remaining results in a combined data frame.

```
# make a common search data frame
revisions_search <- bind_rows(revisions_google %>%
                                select(DOI, Title, Abstract, duplicate_by_either) %>%
                                rename(duplicate = duplicate_by_either),
                              revisions_scopus %>%
                                select (DOI, Title, Abstract, duplicate_by_either)  %>%
                                rename(duplicate = duplicate_by_either)
                              )

# remove duplicates
revisions_search <- revisions_search %>%
  filter(duplicate == FALSE) %>%
  select(-duplicate)

# make title screening data
titles_screening <- revisions_search %>%
  # make a column for coder decisions
  mutate(include = "")
```

Our search for revisions yielded 980 entries for the google scholar search and 1799 entries for the scopus search. After removing duplicates with prior search results (353 in the google search, 655 in the scopus search), and removing 85 remaining duplicates between the two searches, we are left with 1686 new titles to screen.

Export Title screening data

```
write_csv(titles_screening, "revisions_titles_screening.csv")
```

Abstract screening

We read in the two independently conducted title screening spreadsheets.

```
titles_sacha <- read_csv("revisions_titles_screening_sacha.csv") %>%
  mutate(across(include, make_strings_more_compatible)) %>%
  rename(include_sacha = include)
```

```
## Rows: 1686 Columns: 3
## -- Column specification ------------------------------------------------------------------------------------
## Delimiter: ","
## chr (3): DOI, Title, include
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
titles_jan <- read_csv("revisions_titles_screening_jan.csv") %>%
  mutate(across(include, make_strings_more_compatible)) %>%
  rename(include_jan = include)
```

```
## Rows: 1686 Columns: 3
## -- Column specification ------------------------------------------------------------------------------------
## Delimiter: ","
## chr (3): DOI, Title, include
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

We then merge this to the originial data, this time adding abstracts.

```
revisions_search <- bind_cols(revisions_search,
                              titles_sacha %>%
                                select(include_sacha),
                              titles_jan %>%
                                select(include_jan))
```

Code (dis)agreements. Include all entries where at least one of the coders suggested abstract screening

```
revisions_search <- revisions_search %>%
  mutate(agreement = ifelse(include_sacha == include_jan, "agreement", "disagreement"),
         # code abstract screening
         abstract_screening = ifelse(include_jan == "yes" | include_sacha == "yes", TRUE, FALSE)
         )
```

Inspect.

```
revisions_search %>%
  summarize(across(starts_with("include"), ~ sum(. == "yes")),
            abstract_screening = sum(abstract_screening),
            disagreement = sum(agreement == "disagreement")
            )
```

```
## # A tibble: 1 x 4
##   include_sacha include_jan abstract_screening disagreement
##           <int>       <int>              <int>        <int>
## 1           111         267                287          196
```

Remove already included titles

So far we have removed duplicates with the previous search. But we also have added articles that didn't come up in the previous search to our data base. Before going into the abstract, and then full text assessment phase, we want to exclude potential entries of the current, second search, that might already be in our data base.

We also excluded some studies that we came across, but not in the systematic search. We also want to make sure to remove these, before going into the more detailed phase.

```
# Literature
included <- read_csv("../data/cleaned.csv") %>%
  # control conditions only
  filter(condition == "control") %>%
  group_by(paperID) %>%
  summarize(reference = unique(reference)) %>%
  arrange(paperID) %>%
  ungroup()
```

```
## Rows: 303 Columns: 60
## -- Column specification ------------------------------------------------------------------------
## Delimiter: ","
## chr (36): reference, ref, intervention_study, comments_intervention_selection, comments, peer_reviewe
## dbl (19): paperID, sampleID, newsID, n_news, n_news_pool, share_true_news, symetry, experimentID, co
## lgl  (5): news_source, perfect_symetry, treatment_intention, intervention, correlates
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
excluded <- read_csv("excluded.csv")
```

```
## Rows: 238 Columns: 6
## -- Column specification ------------------------------------------------------------------------
## Delimiter: ","
## chr (6): reference, screening, extent, Reason, Additional Reason, Additional remarks
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# make compatible
included <- included %>%
  mutate(across(reference, make_strings_more_compatible)) %>%
  select(paperID, reference) %>%
  mutate(included = "included")
```

```
excluded <- excluded %>%
  mutate(across(reference, make_strings_more_compatible)) %>%
  select(screening, extent, reference) %>%
  mutate(included = "excluded")

# Combine the `included` and `excluded` sheet into a single `studies` data frame.
studies <- bind_rows(included, excluded)
```

We write a function that allows us to identify overlaps between the literature searches.

The function below checks whether a reference in the second search has a matching title in previously included or excluded articles. It adds the variable(s) `duplicate_*` (by title and/or doi), which evaluates `TRUE` if reference appears in the current, second search, and `FALSE` otherwise. To be more certain that we don't miss anything, we search for matches via both the `DOI` and the `Title` column.

```
identify_appearance_in_searches <- function(data, compare_by = "DOI and Title") {

  if (str_detect(compare_by, "Title")) {

 data$duplicate_by_title <- sapply(data$reference, function(x) any(sapply(revisions_search$Title,
                                                                         grepl, x,
                                                                         fixed = TRUE)))
  }

  if (str_detect(compare_by, "DOI")) {

 data$duplicate_by_DOI <- sapply(data$reference, function(x) any(sapply(revisions_search$DOI,
                                                                         grepl, x,
                                                                         fixed = TRUE)))

  }

  if (str_detect(compare_by, "(?=.*DOI)(?=.*Title).*")) {
  data <- data %>%
    mutate(duplicate_by_any =
             ifelse(
               (duplicate_by_DOI | duplicate_by_title)  == TRUE,
                                    TRUE,
                                    FALSE)
           )
  }

 return(data)

}
```

We apply this function to the `studies` data frame that contains all studies we assessed.

```
studies <- identify_appearance_in_searches(studies)
```

Inspect.

```
studies %>%
  summarize(across(starts_with("duplicate"), ~ sum(., na.rm=TRUE))
           )
```

```
## # A tibble: 1 x 3
##   duplicate_by_title duplicate_by_DOI duplicate_by_any
##                <int>            <int>            <int>
## 1                139              123              153
```

Filter out duplicates

```
# filter out duplicates
duplicates_titles <- studies %>%
  filter(duplicate_by_title == TRUE) %>%
  select(reference) %>%
  pull()

duplicates_doi <- studies %>%
  filter(duplicate_by_DOI == TRUE) %>%
  select(reference) %>%
  pull()


identify_duplicates <- function(data) {
  # Apply grepl to each string in the "Title" column to check if it's contained in any string of the ex
  title_check <- sapply(data$Title, function(x) any(grepl(x, duplicates_titles, fixed = TRUE)))

  # Apply grepl to each string in the "DOI" column to check if it's contained in any string of the exte
  doi_check <- sapply(data$DOI, function(x) any(grepl(x, duplicates_doi, fixed = TRUE)))

  # Set "duplicate" to TRUE if either the title or the DOI matches
  data$duplicate <- title_check | doi_check

  # Return the modified dataset
  return(data)
}

revisions_search <- identify_duplicates(revisions_search)

# inspect
revisions_search %>%
  filter(duplicate == TRUE) %>%
  select(Title)
```

```
## # A tibble: 151 x 1
##    Title
##    <chr>
##  1 psychological inoculation for credibility assessment, sharing intention, and discernment of misin
##  2 understanding effects of algorithmic vs. community label on perceived accuracy of hyper-partisan m
##  3 confidence-accuracy resolution in the misinformation paradigm is influenced by the availability of
##  4 emphasizing publishers does not effectively reduce susceptibility to misinformation on social medi
##  5 the effects of repeating false and misleading information on belief
##  6 effects of conspiracy thinking style, framing and political interest on accuracy of fake news reco
##  7 who’s in the crowd matters: cognitive factors and beliefs predict misinformation assessment accura
##  8 the effect of web add-on correction and narrative correction on belief in misinformation depending
##  9 what predicts people’s belief in covid-19 misinformation? a retrospective study using a nationwide
## 10 "fact-checking" videos reduce belief in but not the sharing of "fake news" on twitter
## # i 141 more rows
```

```
# remove duplicates
revisions_search <- revisions_search %>%
```

```r
  filter(duplicate == FALSE) %>%
  select(-c(duplicate, include_sacha, include_jan, agreement))
```

Make abstract screening data

```r
# make title screening data
abstract_screening <- revisions_search %>%
  filter(abstract_screening == TRUE) %>%
  select(-abstract_screening) %>%
  # make a column for coder decisions
  mutate(include = "")
```

Export Title screening data

```r
# Export data to CSV
write_csv(abstract_screening, "revisions_abstract_screening.csv")
```

Full text

We read in the two independently conducted abstract screening spreadsheets.

```r
abstracts_sacha <- read_csv("revisions_abstracts_sacha.csv") %>%
  mutate(across(include, make_strings_more_compatible)) %>%
  rename(include_sacha = include)
```

```
## Rows: 135 Columns: 4
## -- Column specification --------------------------------------------------------------------------
## Delimiter: ","
## chr (4): DOI, Title, Abstract, include
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
abstracts_jan <- read_csv("revisions_abstracts_jan.csv") %>%
  mutate(across(include, make_strings_more_compatible)) %>%
  rename(include_jan = include)
```

```
## Rows: 135 Columns: 4
## -- Column specification --------------------------------------------------------------------------
## Delimiter: ","
## chr (4): DOI, Title, Abstract, include
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

We then merge this to the orginial data.

```r
abstract_screening <- bind_cols(abstracts_sacha %>%
                                  select(include_sacha),
                                abstracts_jan %>%
                                  select(include_jan))
```

Code (dis)agreements. Include all entries where at least one of the coders suggested full text assessment

```r
abstract_screening <- abstract_screening %>%
  mutate(agreement = ifelse(include_sacha == include_jan, "agreement", "disagreement"),
         # code abstract screening
         full_text = ifelse(include_jan == "yes" | include_sacha == "yes", TRUE, FALSE)
         )
```

Inspect.

```
abstract_screening %>%
  summarize(across(starts_with("include"), ~ sum(. == "yes")),
            full_text = sum(full_text),
            disagreement = sum(agreement == "disagreement")
            )
```

```
## # A tibble: 1 x 4
##   include_sacha include_jan full_text disagreement
##           <int>       <int>     <int>        <int>
## 1            38          56        66           38
```

Inspect articles that Sacha wants included and Jan not

```
abstract_screening %>%
  filter(include_jan == "no" & include_sacha == "yes")
```

```
## # A tibble: 10 x 4
##    include_sacha include_jan agreement    full_text
##    <chr>         <chr>       <chr>        <lgl>
##  1 yes           no          disagreement TRUE
##  2 yes           no          disagreement TRUE
##  3 yes           no          disagreement TRUE
##  4 yes           no          disagreement TRUE
##  5 yes           no          disagreement TRUE
##  6 yes           no          disagreement TRUE
##  7 yes           no          disagreement TRUE
##  8 yes           no          disagreement TRUE
##  9 yes           no          disagreement TRUE
## 10 yes           no          disagreement TRUE
```

Make full text assessment data.

```
# make title screening data
full_text <- abstract_screening %>%
  filter(full_text == TRUE) %>%
  select(-full_text, -starts_with("include"), -agreement)
```

Export Title screening data

```
# Export data to CSV
write_csv(full_text, "revisions_full_text_assessment.csv")
```