

Irrigation's real impact on global water and food security

R code

Arnald Puy

Contents

1	Abstract corpus	6
2	Full text corpus	6
3	Policy corpus	8
4	Split datasets for analysis	9
5	Aquastat data	15
6	Session information	17

```

# PRELIMINARY FUNCTIONS #####

sensobol::load_packages(c("openxlsx", "data.table", "tidyverse", "bibliometrix",
                          "igraph", "ggraph", "cowplot", "tidygraph", "benchmarkme",
                          "parallel", "wesanderson"))

# Create custom theme
theme_AP <- function() {
  theme_bw() +
    theme(panel.grid.major = element_blank(),
          panel.grid.minor = element_blank(),
          legend.background = element_rect(fill = "transparent",
                                            color = NA),
          legend.key = element_rect(fill = "transparent",
                                     color = NA),
          strip.background = element_rect(fill = "white"),
          legend.margin = margin(0.5, 0.1, 0.1, 0.1),
          legend.box.margin = margin(0.2, -4, -7, -7),
          plot.margin = margin(3, 4, 0, 4),
          legend.text = element_text(size = 8),
          axis.title = element_text(size = 10),
          legend.key.width = unit(0.4, "cm"),
          legend.key.height = unit(0.4, "cm"),
          legend.title = element_text(size = 9))
}

```

```

# CREATION OF VECTORS WITH NAMES #####

database <- c("wos", "scopus", "dimensions")
topic <- c("water", "food")

# Create all possible combinations
combinations <- expand.grid(database = database, topic = topic)

# Combine the vectors with an underscore
file.name <- paste(combinations$database, "dt", combinations$topic, sep = "_")

# READ IN THE DATA #####

# Loop to create the file names -----
for (i in 1:length(file.name)) {

  database.type <- str_extract(file.name, "^(wos|scopus|dimensions)")

  if(isTRUE(database.type[i] == "wos")) {

    file.name[i] <- paste(file.name[i], "bib", sep = ".")

  } else {

    file.name[i] <- paste(file.name[i], "csv", sep = ".")

  }

}

# vector with new column names -----

new_colnames <- c("doi", "authors", "year", "title", "journal", "abstract", "database")
to_lower <- c("authors", "title", "journal", "abstract")

# Loop to read in the datasets -----

out <- list()

for (i in 1:length(file.name)) {

  database.type <- str_extract(file.name[i], "^(wos|scopus|dimensions)")

  if(isTRUE(database.type == "wos")) {

    out[[i]] <- convert2df(file = file.name[i],
                          dbsource = "wos",

```

```

        format = "bibtex") %>%
data.table() %>%
.[, .(DI, AU, PY, TI, SO, AB)] %>%
.[, database:= "wos"]

} else if (isTRUE(database.type == "dimensions")) {

  out[[i]] <- fread(file.name[i], skip = 1) %>%
    .[, .(DOI, Authors, PubYear, Title, `Source title`, Abstract)] %>%
    .[, database:= "dimensions"]

} else if(isTRUE(database.type == "scopus")) {

  out[[i]] <- fread(file.name[i]) %>%
    .[, .(DOI, Authors, Year, Title, `Source title`, Abstract)] %>%
    .[, database:= "scopus"]
}

setnames(out[[i]], colnames(out[[i]]), new_colnames) %>%
.[, (to_lower):= lapply(.SD, tolower), .SDcols = (to_lower)] %>%
.[, abstract:= sub("references.*", "", abstract)]

}

```

```

##
## Converting your wos collection into a bibliographic dataframe
##
##
## Warning:
## In your file, some mandatory metadata are missing. Bibliometrix functions may not work properly
##
## Please, take a look at the vignettes:
## - 'Data Importing and Converting' (https://www.bibliometrix.org/vignettes/Data-Importing-and-Converting)
## - 'A brief introduction to bibliometrix' (https://www.bibliometrix.org/vignettes/Introduction-to-Bibliometrix)
##
##
## Missing fields:  C1 CR
## Done!
##
##
## Converting your wos collection into a bibliographic dataframe
##
##
## Warning:
## In your file, some mandatory metadata are missing. Bibliometrix functions may not work properly
##
## Please, take a look at the vignettes:

```

```
## - 'Data Importing and Converting' (https://www.bibliometrix.org/vignettes/Data-Importing-and-Converting)
## - 'A brief introduction to bibliometrix' (https://www.bibliometrix.org/vignettes/Introduction-to-Bibliometrix)
##
##
## Missing fields: C1 CR
## Done!
```

```
names(out) <- combinations$topic

# CLEAN THE DATASETS #####

# Arrange -----

dt <- rbindlist(out, idcol = "topic")

tmp <- split(dt, list(dt$topic, dt$database))

cols_to_merge_by <- c("doi", "year", "title", "journal", "abstract")

dt.water <- merge(merge(tmp$water.dimensions, tmp$water.scopus, by = cols_to_merge_by,
  all = TRUE), tmp$water.wos, by = cols_to_merge_by,
  all = TRUE)

dt.food <- merge(merge(tmp$food.dimensions, tmp$food.scopus, by = cols_to_merge_by,
  all = TRUE), tmp$food.wos, by = cols_to_merge_by,
  all = TRUE)

# Filter out duplicated studies by doi -----

tmp.list <- list(dt.water, dt.food)
duplicated.dois <- final.dt <- list()

for (i in 1:length(tmp.list)) {

  duplicated.dois[[i]] <- duplicated(tmp.list[[i]]$doi, incomparables = NA, na.rm = TRUE)
  final.dt[[i]] <- tmp.list[[i]][!duplicated.dois[[i]][, location.belief.system := "abstract"]]

}

names(final.dt) <- topic

# Check if there is any duplicated doi -----

any(duplicated(final.dt$food$doi, na.rm = TRUE, incomparables = NA))

## [1] FALSE

# Export to xlsx -----
```

```
for (i in names(final.dt)) {

  write.xlsx(final.dt[[i]][, .(doi, year, title, abstract, location.belief.system)],
            paste("final.dt", names(final.dt[i]), "xlsx", sep = "."))
}
```

1 Abstract corpus

```
final.dt.water.screened <- data.table(read.xlsx("final.dt.water_screened.xlsx"))
final.dt.food.screened <- data.table(read.xlsx("final.dt.food_screened.xlsx"))
screened.dt <- list(final.dt.water.screened, final.dt.food.screened)
names(screened.dt) <- c("water", "food")

lapply(screened.dt, function(x) x[, .N, screening])
```

```
## $water
##   screening      N
##   <char> <int>
## 1:      F    168
## 2:      T    163
##
## $food
##   screening      N
##   <char> <int>
## 1:      F    465
## 2:      T     39
```

```
# Export for close-reading only the references that do include
# the belief system in the abstract -----

for (i in names(screened.dt)) {

  screened.dt[[i]][screening == "T"] %>%
    unique(., by = "title") %>%
    .[, .(doi, title, year)] %>%
    write.xlsx(., paste("abstract.corpus", i, "xlsx", sep = "."))
}
```

2 Full text corpus

```
# LOAD IN DIMENSIONS DATASET (FULL TEXT) #####

# Load dataset of the full text -----

colnames.full.text <- c("doi", "year", "title", "journal", "topic")
```

```

keywords <- c("water", "irrigat")

# Function to load and preprocess data -----

load_and_preprocess_data <- function(file_path, topic) {
  fread(file_path, skip = 1)[, topic := topic]
}

# Load and preprocess water data -----

dimensions.full.text.water <- load_and_preprocess_data("dimensions_dt_full_text_water_2022_2023.csv", "water")

dimensions.full.text.food <- rbind(
  load_and_preprocess_data("dimensions_dt_full_text_food_2022.csv", "food"),
  load_and_preprocess_data("dimensions_dt_full_text_food_2023.csv", "food")
)

# Combine water and food data -----

result <- rbind(dimensions.full.text.water, dimensions.full.text.food) %>%
  .[, .(DOI, PubYear, Title, `Source title`, topic)] %>%
  setnames(., colnames(.), colnames.full.text) %>%
  # remove the references that are included in the dataset that
  # collects mentions in the abstracts
  .[!.$doi %in% final.dt.water.screened$doi]

# Create a logical condition for pattern matching using grepl -----

pattern_condition <- sapply(keywords, function(keyword)
  grepl(keyword, result$title, ignore.case = TRUE))

full.text.dt <- result[rowSums(pattern_condition) > 0]

# Sample just 50% for the analysis -----

# Create function
random_sample <- function(input_dt) {
  set.seed(123) # Set a seed for reproducibility
  sampled_dt <- input_dt[sample(.N, .N * 0.5), ]
  return(sampled_dt)
}

# sample
full.text.sampled <- full.text.dt[, random_sample(.SD), topic]

full.text.sampled[, .N, topic]

```

```
##      topic      N
##      <char> <int>
## 1:   water   4607
## 2:    food   8108

# Export -----

for (i in c("water", "food")) {

  full.text.sampled[topic == i] %>%
    write.xlsx(paste("full.text.corpus", i, ".xlsx", sep = "."))
}
```

3 Policy corpus

```
# LOAD IN DIMENSIONS DATASETS (POLICY TEXT) #####

dt.policy.water <- load_and_preprocess_data("dimensions_dt_policy.csv", "water")
dt.policy.food <- load_and_preprocess_data("dimensions_dt_policy_food.csv", "food")

dimensions.full.text.policy <- rbind(dt.policy.food, dt.policy.water) %>%
  .[, .(`Policy document ID`, PubYear, Title, `Publishing Organization`,
        `Sustainable Development Goals`, `Source Linkout`, topic)]

dimensions.full.text.policy[, .N, topic]

##      topic      N
##      <char> <int>
## 1:    food  10573
## 2:   water   3455

# Create a logical condition for pattern matching using grepl
pattern_condition_policy <- sapply(keywords, function(keyword)
  grepl(keyword, dimensions.full.text.policy$Title, ignore.case = TRUE))

# Combine conditions with OR using rowSums
matching.rows.policy <- dimensions.full.text.policy[rowSums(pattern_condition_policy) > 0]

matching.rows.policy[, .N, topic]

##      topic      N
##      <char> <int>
## 1:    food    750
## 2:   water    450

# Export -----

for (i in c("water", "food")) {
```



```

matching.rows.policy[topic == i] %>%
  write.xlsx(paste("policy.corpus", i, "xlsx", sep = "."))
}

```

4 Split datasets for analysis

```

# SPLIT THE DATASET INTO N FOR RESEARCH #####

# Function to split dataset in n chunks -----

split_dt_fun <- function(dt, num_parts) {

  split_dt <- list()

  # Calculate the number of rows in each part
  rows_per_part <- nrow(dt) %/% num_parts

  # Split the data.table into roughly equal parts
  for (i in 1:num_parts) {

    start_row <- (i - 1) * rows_per_part + 1
    end_row <- i * rows_per_part

    if (i == num_parts) {

      end_row <- nrow(dt)
    }
    split_dt[[i]] <- dt[start_row:end_row, ]
  }

  return(split_dt)
}

# Create the datasets for close reading -----

times.nanxin <- 4
times.arnald <- 2
nanxin <- paste(rep("nanxin", times.nanxin), 1:times.nanxin, sep = "")
arnald <- paste(rep("arnald", times.arnald), 1:times.arnald, sep = "")
names_surveyors <- c(arnald, nanxin, "seth", paste("student", 1:4, sep = ""))
n_surveyors <- length(names_surveyors)

full.text.corpus.water <- read.xlsx("full.text.corpus.water.xlsx")

survey.dt.split <- split_dt_fun(dt = full.text.corpus.water, num_parts = n_surveyors)

```

```

names(survey.dt.split) <- names_surveyors

# Export -----

for (i in 1:length(survey.dt.split)) {

  write.xlsx(survey.dt.split[[i]],
             file = paste0(names(survey.dt.split)[i], ".dt", ".xlsx"))

}

# CREATE VECTORS TO READ IN AND CLEAN THE DATASETS #####

tmp <- list()
names.files <- c("WORK", "NETWORK")
cols_of_interest <- c("title", "author", "claim", "citation")
files.abstract.water <- paste(paste("abstract.corpus.water2_", names.files, sep = ""), "xlsx",

# READ IN DATASETS AND TURN TO LOWERCAPS #####

for (i in 1:length(files.abstract.water)) {

  tmp[[i]] <- data.table(read.xlsx(files.abstract.water[i]))

  if (i == 1) {

    tmp[[i]][, title:= tolower(title)]

  } else {

    tmp[[i]][, (cols_of_interest):= lapply(.SD, tolower), .SDcols = (cols_of_interest)]
  }
}

names(tmp) <- names.files

# CLEAN DATASET #####

abstract.water.dt <- merge(tmp[[1]][claim.in.text == "F"], tmp[[2]], by = c("doi", "title"), a
abstract.water.dt[, claim.in.text:= ifelse(is.na(claim.in.text), "TRUE", "FALSE")]
abstract.water.dt[, c(cols_of_interest, "nature.claim"):= lapply(.SD, trimws), .SDcols = c(cols
abstract.water.dt[, year:= ifelse(is.na(year), as.numeric(gsub("\\D", "", abstract.water.dt$au

# PRELIMINARY ANALYSIS #####

a <- tmp$WORK[claim.in.text %in% c("T", "F")] %>%
  .[, .N, claim.in.text] %>%
  ggplot(., aes(claim.in.text, N)) +

```

```

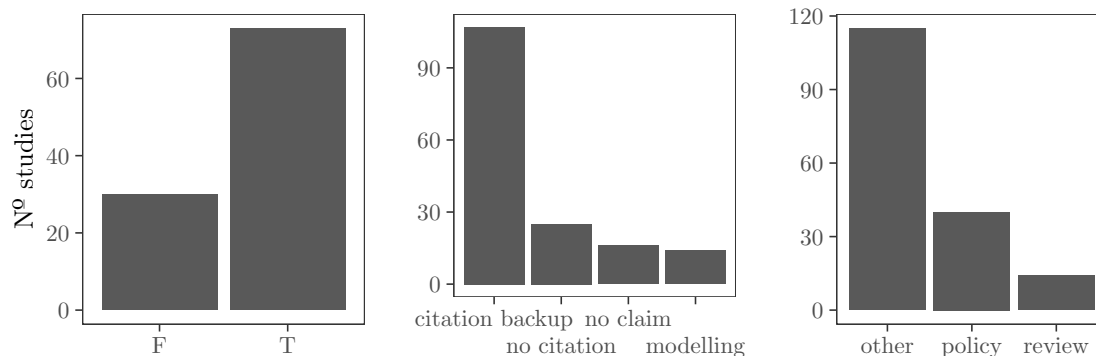
geom_bar(stat = "identity") +
labs(x = "", y = "Nº studies") +
theme_AP()

b <- tmp$NETWORK %>%
.[complete.cases(.$nature.claim), ] %>%
.[, nature.claim:= trimws(nature.claim)] %>%
.[, .N, nature.claim] %>%
ggplot(., aes(reorder(nature.claim, -N), N)) +
scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
geom_bar(stat = "identity") +
labs(x = "", y = "") +
theme_AP()

c <- tmp$NETWORK %>%
.[, .N, document.type] %>%
ggplot(., aes(reorder(document.type, -N), N)) +
geom_bar(stat = "identity") +
labs(x = "", y = "") +
theme_AP()

plot_grid(a, b, c, ncol = 3)

```



```

# NETWORK ANALYSIS #####

# Arrange data -----
network.dt <- copy(tmp$NETWORK)
setnames(network.dt, c("author", "citation"), c("from", "to"))
network.dt <- network.dt[, .(from, to, document.type, nature.claim)]
cols_to_change <- colnames(network.dt)
network.dt[, (cols_to_change):= lapply(.SD, trimws), .SDcols = (cols_to_change)]

# only complete cases -----
network.dt.complete <- network.dt[complete.cases(network.dt$to), ]

# Transform to graph -----
citation_graph <- graph_from_data_frame(d = network.dt.complete, directed = TRUE)

```

```

# Calculate network metrics -----
edge_density(citation_graph)

## [1] 0.005838807

network_metrics <- data.table(node = V(citation_graph)$name,

                                # Degree of a node: The number of connections or
                                # edges linked to that node.
                                # It represents how well-connected or central a
                                # node is within the graph.
                                degree = degree(citation_graph, mode = "in"),

                                # Betweenness centrality of a node: Measures the
                                # extent to which a node lies on the shortest
                                # paths between all pairs of other nodes in the graph.
                                # Nodes with high betweenness centrality act as
                                # bridges or intermediaries, facilitating
                                # communication and information flow between other nodes.
                                betweenness = betweenness(citation_graph),

                                # Closeness centrality of a node: Measures how
                                # close a node is to all other nodes in the graph,
                                # taking into account the length of the shortest paths.
                                # Nodes with high closeness centrality are able to
                                # efficiently communicate or interact with other
                                # nodes in the graph.
                                closeness = closeness(citation_graph),
                                pagerank = page_rank(citation_graph)$vector
                                )

network_metrics[order(-degree)][1:5]

##           node degree betweenness closeness
##           <char>   <num>         <num>    <num>
## 1:           fao aquastat      13           0      NaN
## 2:           molden et al 2007    4          10 0.3333333
## 3: world water assessment programme 2009    2           4 0.3333333
## 4:           world bank 2007    2           7 1.0000000
## 5: world water assessment programme 2018    2           2 0.3333333
##           pagerank
##           <num>
## 1: 0.079718516
## 2: 0.020085501
## 3: 0.010586794
## 4: 0.023540448
## 5: 0.007253915

```

```
network_metrics[order(-betweenness)][1:5]
```

```
##           node degree betweenness closeness pagerank
##           <char>  <num>         <num>         <num>         <num>
## 1:    molden et al 2007         4          10 0.3333333 0.02008550
## 2:    world bank 2007          2           7 1.0000000 0.02354045
## 3: aivazidou et al 2016         1           6 0.1666667 0.01008686
## 4: united nations 2009         1           6 0.3333333 0.01249487
## 5:   oki and kanae 2006         1           6 1.0000000 0.02099371
```

```
network_metrics[order(-closeness)][1:5]
```

```
##           node degree betweenness closeness pagerank
##           <char>  <num>         <num>         <num>         <num>
## 1:    sharma and irmak 2012      0           0           1 0.003921035
## 2:    world bank 2007           2           7           1 0.023540448
## 3:    brajovic et al 2015      0           0           1 0.003921035
## 4:    rivers et al 2015      0           0           1 0.003921035
## 5: hafeez and khalid awan 2022  0           0           1 0.003921035
```

```
# PLOT NETWORK #####
```

```
# Retrieve a vector with the node names -----
```

```
graph <- tidygraph::as_tbl_graph(network.dt.complete, directed = TRUE)
vec.names <- graph %>%
  activate(nodes) %>%
  pull() %>%
  data.table(name = .)
```

```
# Merge with info from the network.dt -----
```

```
vec.nature.claim <- merge(vec.names, unique(network.dt[, .(from, nature.claim)]),
  by.x = "name", by.y = "from", all.x = TRUE)
```

```
# Merge with the correct order -----
```

```
order_indices <- match(vec.names$name, vec.nature.claim$name)
final.vec.nature.claim <- vec.nature.claim[order_indices, ] %>%
  .[, nature.claim]
```

```
# Attach to the graph -----
```

```
graph <- graph %>%
  activate(nodes) %>%
  mutate(nature.claim = final.vec.nature.claim)
```

```
# Plot network -----
```

```

ggraph(graph, layout = "igraph", algorithm = "nicely") +
  geom_edge_link(arrow = arrow(length = unit(1.5, 'mm')),
                end_cap = circle(1, "mm")) +
  geom_node_point(size = 2, aes(color = nature.claim)) +
  geom_node_text(aes(label = name), repel = TRUE, size = 2.1) +
  labs(x = "", y = "") +
  scale_color_discrete(name = "") +
  theme_AP() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "top")

```

```

## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` in the `default_aes` field and elsewhere instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

```



```

# AQUASTAT DATA #####

# Read in dataset -----

aquastat <- data.table(read.xlsx("aquastat_dt.xlsx"))
aquastat.dt <- aquastat[Variable == "Agricultural water withdrawal as % of total water withdrawal", .(Area, Year, Value, Symbol)] %>%
  .[Year == 2020] %>%
  .[, Symbol := ifelse(Symbol == "E", "estimated", "inputed")]

# Count proportion of estimated and inputed values
# E: Estimate either calculated as sum or identify (yield) from
# official values or from an AQUASTAT estimation.
# I: Imputed (carry forward, vertical imputation, linear interpolation)

# Calculate mean and median -----

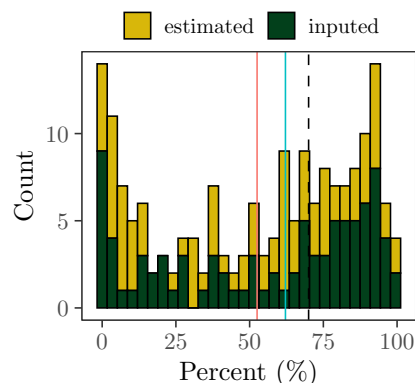
mean.and.median <- aquastat.dt[, .(mean = mean(Value), median = median(Value))] %>%
  melt(., measure.vars = colnames(.))

# Plot histogram -----

a <- ggplot(aquastat.dt, aes(Value)) +
  geom_histogram(color = "black", aes(fill = Symbol)) +
  geom_vline(data = mean.and.median, aes(xintercept = value, color = variable),
    show.legend = FALSE) +
  geom_vline(xintercept = 70, lty = 2) +
  scale_fill_manual(name = "",
    values = wes_palette(name = "Cavalcanti1", 5)) +
  theme_AP() +
  labs(x = "Percent (\\%)", y = "Count") +
  theme(legend.position = "top")
a

```

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.



6 Session information

```
# SESSION INFORMATION #####
```

```
sessionInfo()
```

```
## R version 4.3.2 (2023-10-31)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS Sonoma 14.2.1
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/lib/libRlapack.dylib;
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## time zone: Europe/London
## tzcode source: internal
##
## attached base packages:
## [1] parallel stats graphics grDevices utils datasets methods
## [8] base
##
## other attached packages:
## [1] wesanderson_0.3.6 benchmarkme_1.0.8 tidygraph_1.3.0 cowplot_1.1.1
## [5] ggraph_2.1.0 igraph_1.6.0 bibliometrix_4.0.1 lubridate_1.9.2
## [9] forcats_1.0.0 stringr_1.5.1 dplyr_1.1.4 purrr_1.0.2
## [13] readr_2.1.4 tidyr_1.3.0 tibble_3.2.1 ggplot2_3.4.4
## [17] tidyverse_2.0.0 data.table_1.14.99 openxlsx_4.2.5.2
##
## loaded via a namespace (and not attached):
## [1] Rdpack_2.6 gridExtra_2.3 readxl_1.4.2
## [4] rlang_1.1.3 magrittr_2.0.3 tidytext_0.4.1
## [7] compiler_4.3.2 vctrs_0.6.5 crayon_1.5.2
## [10] pkgconfig_2.0.3 fastmap_1.1.1 ellipsis_0.3.2
## [13] labeling_0.4.3 utf8_1.2.4 promises_1.2.0.1
## [16] rmarkdown_2.21 tzdb_0.3.0 tinytex_0.45
## [19] bit_4.0.5 xfun_0.39 jsonlite_1.8.4
## [22] flashClust_1.01-2 highr_0.10 SnowballC_0.7.1
## [25] later_1.3.0 tweenr_2.0.2 cluster_2.1.4
## [28] R6_2.5.1 stringi_1.8.3 RColorBrewer_1.1-3
## [31] cellranger_1.1.0 estimability_1.4.1 iterators_1.0.14
## [34] Rcpp_1.0.12 knitr_1.42 filehash_2.4-5
## [37] httpuv_1.6.9 rentrez_1.2.3 Matrix_1.6-1.1
## [40] timechange_0.2.0 tidyselect_1.2.0 viridis_0.6.4
## [43] rstudioapi_0.15.0 stringdist_0.9.10 pubmedR_0.0.3
## [46] yaml_2.3.7 codetools_0.2-19 doParallel_1.0.17
```

```
## [49] lattice_0.21-9      plyr_1.8.8           shiny_1.7.4
## [52] withr_3.0.0         benchmarkmeData_1.0.4 coda_0.19-4
## [55] evaluate_0.20       polyclip_1.10-6     zip_2.3.0
## [58] pillar_1.9.0        janeaustenr_1.0.0   foreach_1.5.2
## [61] DT_0.27            plotly_4.10.1       generics_0.1.3
## [64] vroom_1.6.1         hms_1.1.3           munsell_0.5.0
## [67] scales_1.3.0        sensobol_1.1.4      xtable_1.8-4
## [70] leaps_3.1           glue_1.7.0          tikzDevice_0.12.4
## [73] emmeans_1.8.5       scatterplot3d_0.3-43 lazyeval_0.2.2
## [76] tools_4.3.2         tokenizers_0.3.0    mvtnorm_1.1-3
## [79] graphlayouts_1.0.2  XML_3.99-0.14       grid_4.3.2
## [82] rbibutils_2.2.16    rscopus_0.6.6       colorspace_2.1-0
## [85] dimensionsR_0.0.3   ggforce_0.4.1       bibliometrixData_0.3.0
## [88] cli_3.6.2           fansi_1.0.6          viridisLite_0.4.2
## [91] gtable_0.3.4        digest_0.6.34       ggrepel_0.9.5
## [94] FactoMineR_2.8      htmlwidgets_1.6.2   farver_2.1.1
## [97] htmltools_0.5.5     factoextra_1.0.7    lifecycle_1.0.4
## [100] httr_1.4.5          multcompView_0.1-9  mime_0.12
## [103] bit64_4.0.5         MASS_7.3-60
```

```
## Return the machine CPU
```

```
cat("Machine:      "); print(get_cpu())$model_name)
```

```
## Machine:
```

```
## [1] "Apple M1 Max"
```

```
## Return number of true cores
```

```
cat("Num cores:    "); print(detectCores(logical = FALSE))
```

```
## Num cores:
```

```
## [1] 10
```

```
## Return number of threads
```

```
cat("Num threads: "); print(detectCores(logical = FALSE))
```

```
## Num threads:
```

```
## [1] 10
```