

Validation, calibration, etc

R code

Arnald Puy

Contents

```
# PRELIMINARY FUNCTIONS #####

# Load the packages
sensobol::load_packages(c("data.table", "tidyverse", "openxlsx", "tm", "stringr",
                           "pdftools", "tidytext", "scales", "cowplot", "lsa",
                           "LSAfun", "ggrepel", "tidyquant", "parallel", "benchmarkme"))

## Loading required package: parallel

# 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(0.2, "cm"),
          legend.key.height = unit(0.2, "cm"),
          legend.title = element_text(size = 9))
}

# FUNCTIONS TO CLEAN THE TEXT #####

# Function to remove words from text
removeWords <- function(str, stopwords) {
  x <- unlist(strsplit(str, " "))
  paste(x[!x %in% stopwords], collapse = " ")
}
```

```

# Function to remove punctuation, citations, numbers, stopwords in english,
# bring to lowercase and strip whitespace, and especial characters, etc...
clear_text <- function(x, stem = TRUE) {

  y <- tolower(x)
  y <- str_replace_all(y, "[[:punct:]]", " ") # Remove punctuation characters
  y <- tm::removeNumbers(y)
  y <- tm::removeWords(y, stopwords::stopwords(language = "en"))
  y <- str_remove_all(y, "[^\\da-zA-Z ]") # Remove all non-alphanumerical
  y <- gsub("\\s[A-Za-z](?= )", " ", y, perl = TRUE) # Remove isolated letters
  #y <- tm::stripWhitespace(y)
  y <- str_squish(y)

  if (stem == TRUE) {
    y <- stemDocument(y) # Stem the document and keep only the root of the word
  }

  return(y)
}

# READ DATA #####

dt <- data.table(read.xlsx("final.dt.xlsx"))
dt[, keywords.large:= tolower(keywords.large)]
dt[, abstract.cleaned:= clear_text(abstract.large, stem = FALSE)]

# TERMS TO SEARCH #####

keywords <- c("validation", "verification", "calibration", "confirmation", "evaluation",
             "benchmarking")
keywords.stemmed <- stemDocument(keywords)

# keywords.stemmed <- c(
#   "calibration", "verification", "validation", "evaluation",
#   "calibrations", "verifications", "validations", "evaluations",
#   "calibrated", "verified", "validated", "evaluated",
#   "calibrating", "verifying", "validating", "evaluating",
#   "calibrative", "verificative", "validative", "evaluative",
#   "calibratively", "verificatively", "validatively", "evaluatively"
# )

# SEARCHING FOR TERMS #####

# Check which papers include keywords in abstract, keywords or title -----
selected_cols <- c("title", "abstract", "keywords")
#selected_cols <- c("title.large", "abstract.large", "keywords.large")

```

```

out <- list()

for(i in 1:length(keywords.stemmed)) {

  out[[i]] <- dt[, lapply(.SD, function(x)
    str_detect(x, keywords.stemmed[i])), .SDcols = (selected_cols)]
}

# ARRANGE DATA #####

names(out) <- keywords.stemmed

valid.dt <- lapply(out, function(x) rowSums(x, na.rm = TRUE) > 0L) %>%
  do.call(cbind, .) %>%
  data.table() %>%
  .[, any.column:= rowSums(.SD) > 0]

full.dt <- cbind(dt, valid.dt)
full.dt.cols <- data.frame(full.dt[, .SD, .SDcols = keywords.stemmed])

# DESCRIPTIVE STATISTICS #####

full.dt[, lapply(.SD, sum), .SDcols = keywords.stemmed]

##    valid verif calibr confirm evalu benchmark
##    <int> <int>  <int>   <int> <int>      <int>
## 1:   536    76   743     74  1065        86

# Count number of papers with mentions to validation, validation + verification,
# validation + calibration, etc -----

# Function to count the number of TRUE values shared between columns
count_shared_true <- function(data, cols) {
  sum(rowSums(data[, cols]) == length(cols))
}

# Create an empty list to store results
results_list <- list()

# Loop through all combinations of columns and count shared TRUE values
for (size in 2:length(keywords.stemmed)) {

  for (cols_combination in combn(ncol(full.dt.cols), size, simplify = FALSE)) {

    shared_true_count <- count_shared_true(data = full.dt.cols, cols = cols_combination)
    col_names <- colnames(full.dt.cols)[cols_combination]
  }
}

```

```

    # Append results to the list
    results_list <- c(results_list, list(data.table(combination = paste(col_names, collapse = "
  }
}

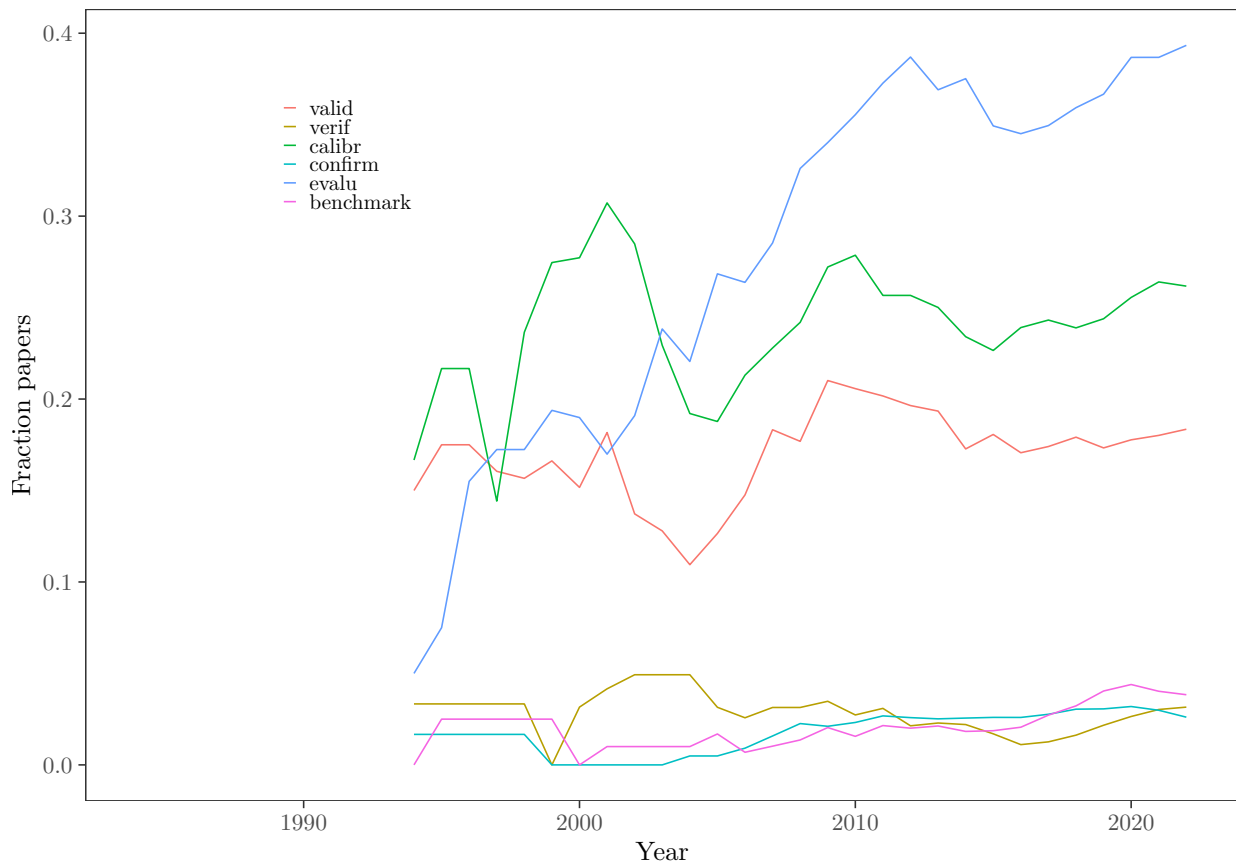
# Combine the list of data.tables into a single data.table
comb_dt <- rbindlist(results_list)

# PLOTS #####

plot.keywords.time <- merge(full.dt, full.dt[, .(total.papers = .N), year], by = "year") %>%
  melt(., measure.vars = keywords.stemmed) %>%
  .[, sum(value, na.rm = TRUE), .(variable, year, total.papers)] %>%
  .[, fraction:= V1 / total.papers] %>%
  ggplot(., aes(year, fraction, color = variable)) +
  scale_color_discrete(name = "") +
  geom_ma(ma_fun = SMA, n = 5, lty = 1) +
  theme_AP() +
  theme(legend.position = c(0.23, 0.85)) +
  labs(x = "Year", y = "Fraction papers")

plot.keywords.time

```

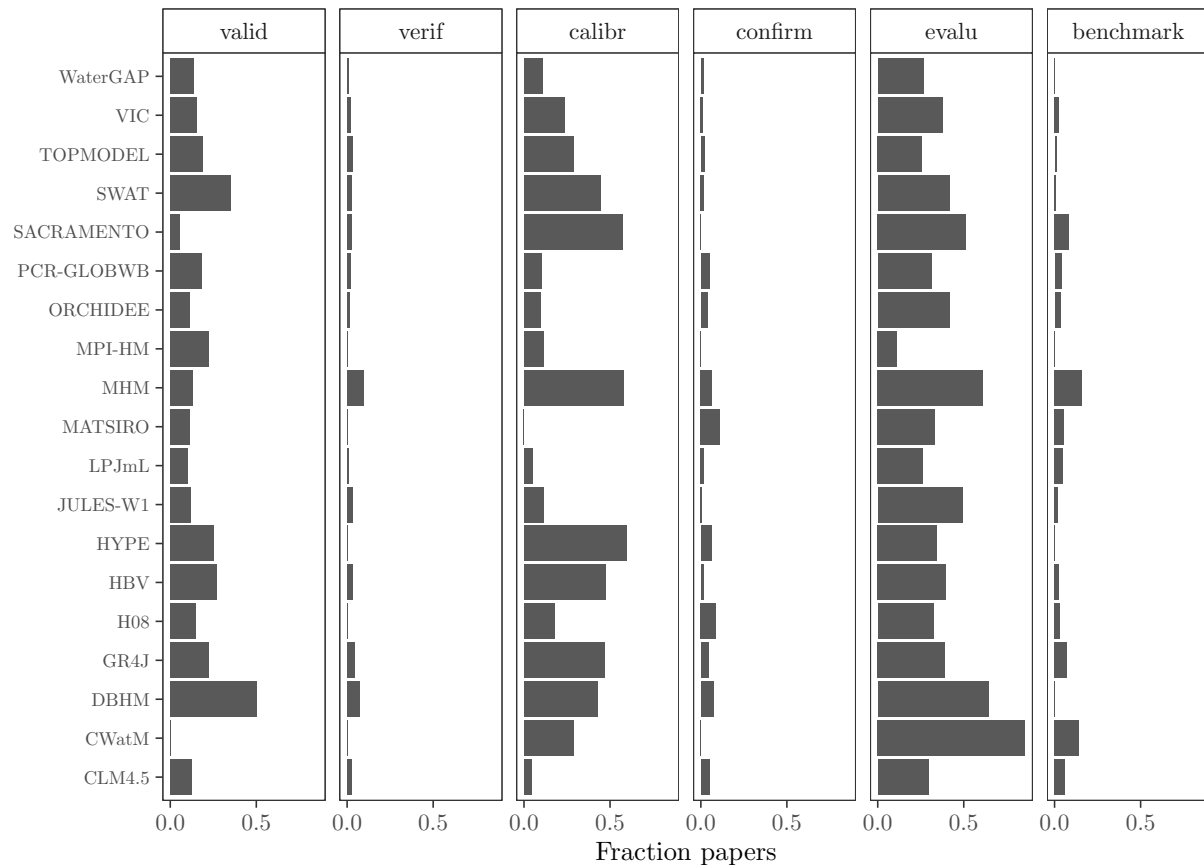


```

plot.keyword.per.model <- merge(full.dt, full.dt[, .(total.papers = .N), Model], by = "Model")
melt(., measure.vars = keywords.stemmed) %>%
.[, sum(value, na.rm = TRUE), .(variable, Model, total.papers)] %>%
.[, fraction:= V1 / total.papers] %>%
ggplot(., aes(Model, fraction)) +
geom_bar(stat = "identity", position = position_dodge(2)) +
coord_flip() +
facet_wrap(~variable, ncol = 6) +
scale_y_continuous(breaks = pretty_breaks(n = 3)) +
theme_AP() +
labs(x = "", y = "Fraction papers") +
theme(axis.text.y = element_text(size = 7))

```

plot.keyword.per.model

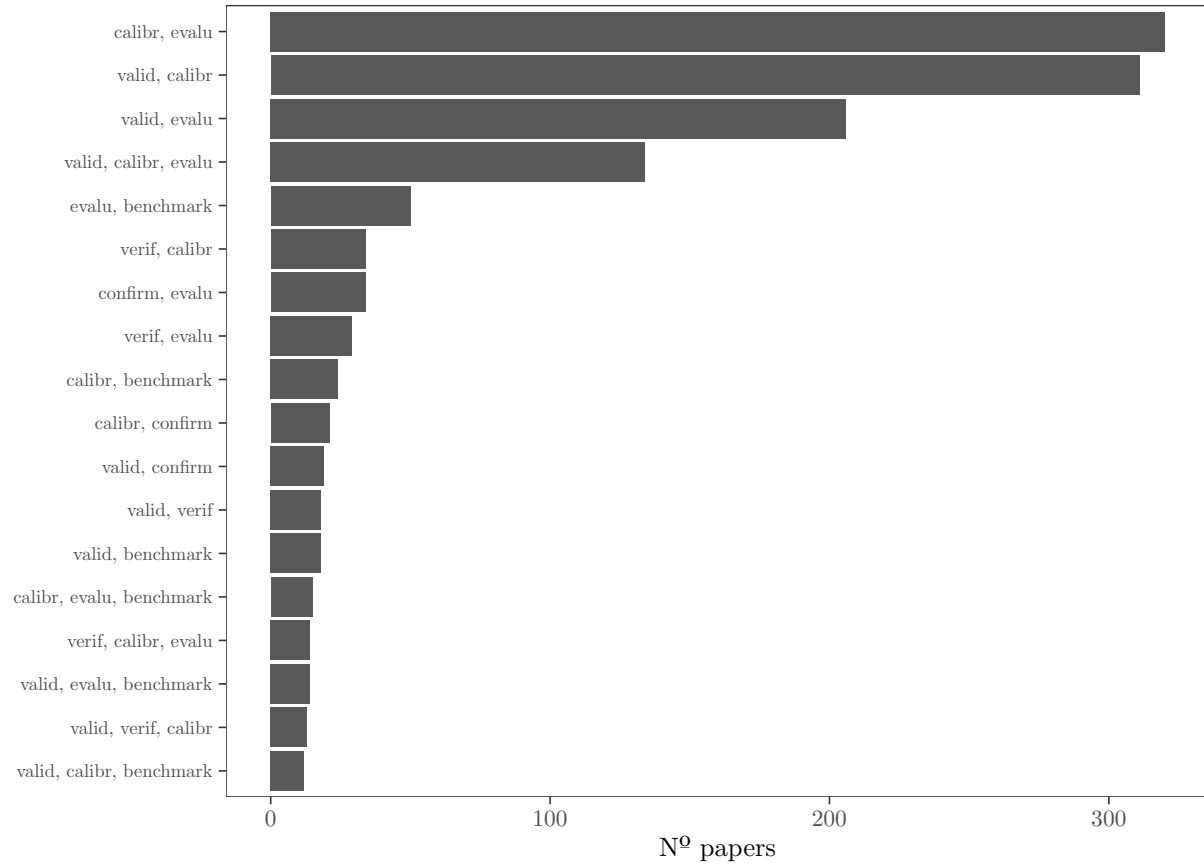


```

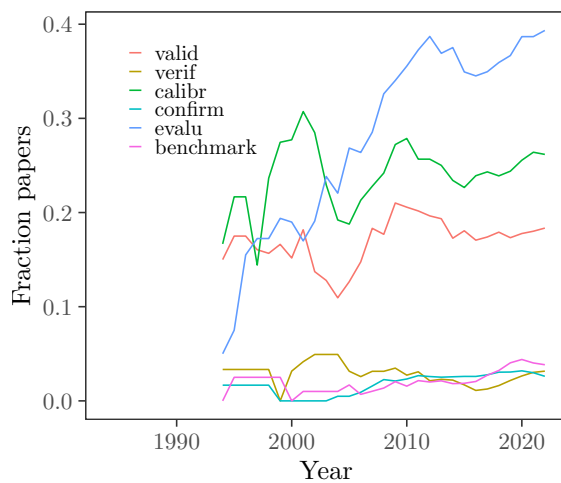
plot.keyword.comb <- comb_dt[N > 10] %>%
ggplot(., aes(reorder(combination, N), N)) +
geom_bar(stat = "identity") +
coord_flip() +
theme_AP() +
theme(axis.text.y = element_text(size = 7)) +
labs(x = "", y = "N° papers")

```

```
plot.keyword.comb
```



```
plot.keywords.time
```



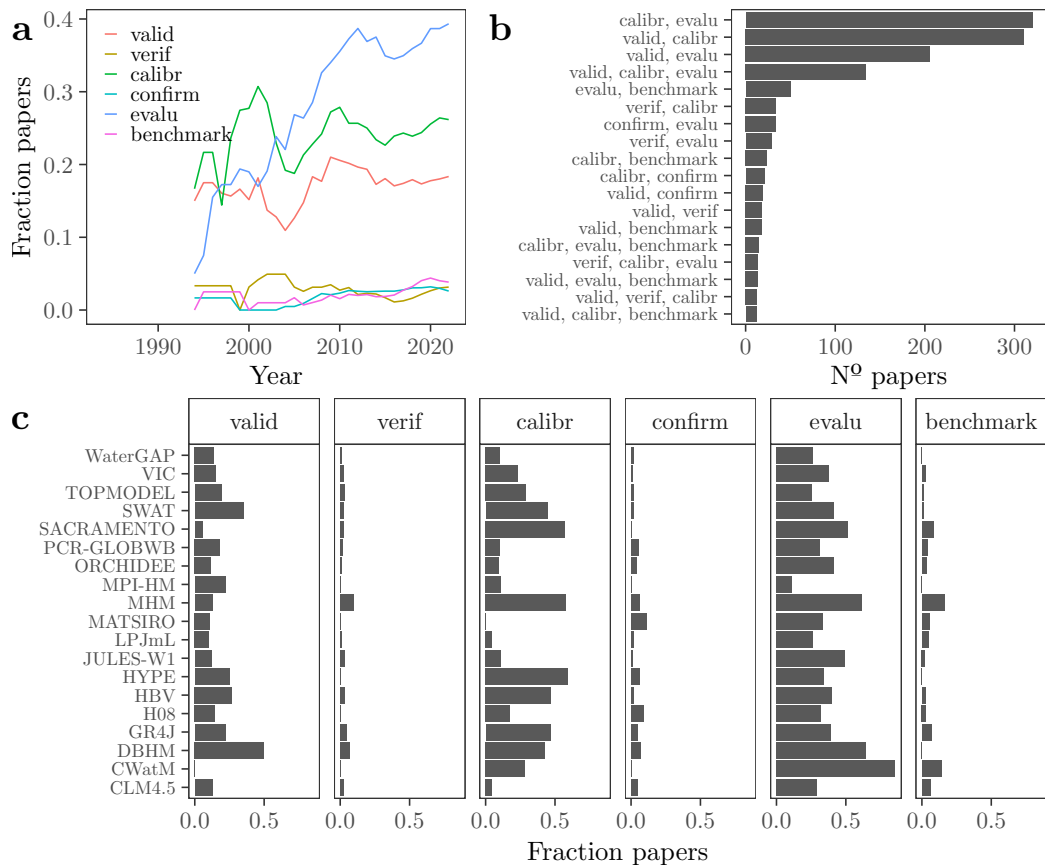
```
# MERGE DESCRIPTIVE PLOTS #####
```

```
top <- plot_grid(plot.keywords.time, plot.keyword.comb, ncol = 2, labels = "auto",
  rel_widths = c(0.45, 0.55))
```

```
plot.merged <- plot_grid(top, plot.keyword.per.model, ncol = 1,
```

```
labels = c("", "c"), rel_heights = c(0.45, 0.55))
```

```
plot.merged
```



```
# TOKEN ANALYSIS #####

# Create function -----
tokenize_fun <- function(dt, word, keywords, N.tokens) {

  # Create long dataset
  dt <- melt(dt, measure.vars = keywords)
  output <- dt[variable == word & value == TRUE]

  # Token analysis -----
  # We count the co-occurrences of words without taking into account their order
  # within the n-token
  token.analysis <- output %>%
    unnest_tokens(bigram, abstract.cleaned, token = "ngrams", n = N.tokens) %>%
    separate(bigram, into = c("word1", "word2"), sep = " ") %>%
    data.table() %>%
    .[, `:=`(word1= pmin(word1, word2), word2 = pmax(word1, word2))] %>%
    count(word1, word2, sort = TRUE) %>%
    unite(., col = "bigram", c("word1", "word2"), sep = " ") %>%

```

```

data.table()

# Vector to retrieve only the bigrams with uncertainty or sensit
vec <- token.analysis[, str_detect(bigram, word)]

# Final dataset
output.dt <- token.analysis[vec]

# Plot the q0 words most commonly
# associated with uncertainty and sensit -----
plot.token <- output.dt %>%
  .[, sum(n), bigram] %>%
  .[order(-V1)] %>%
  .[, head(.SD, 10)] %>%
  ggplot(. , aes(reorder(bigram, V1, sum), V1)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_y_continuous(breaks = pretty_breaks(n = 3)) +
  theme_AP() +
  labs(y = "$n$", x = "") +
  ggtitle(word) +
  theme(legend.position = "none",
        plot.title = element_text(size = 11),
        axis.text.y = element_text(size = 7))

# Arrange and output -----

out <- list(output.dt, plot.token)
names(out) <- c("data", "token")

return(out)
}

# RUN TOKEN ANALYSIS #####

N.tokens <- 2
token.dt <- list()

for (j in keywords.stemmed) {
  token.dt[[j]] <- tokenize_fun(dt = full.dt, word = j,
                                keywords = keywords.stemmed,
                                N.tokens = N.tokens)
}

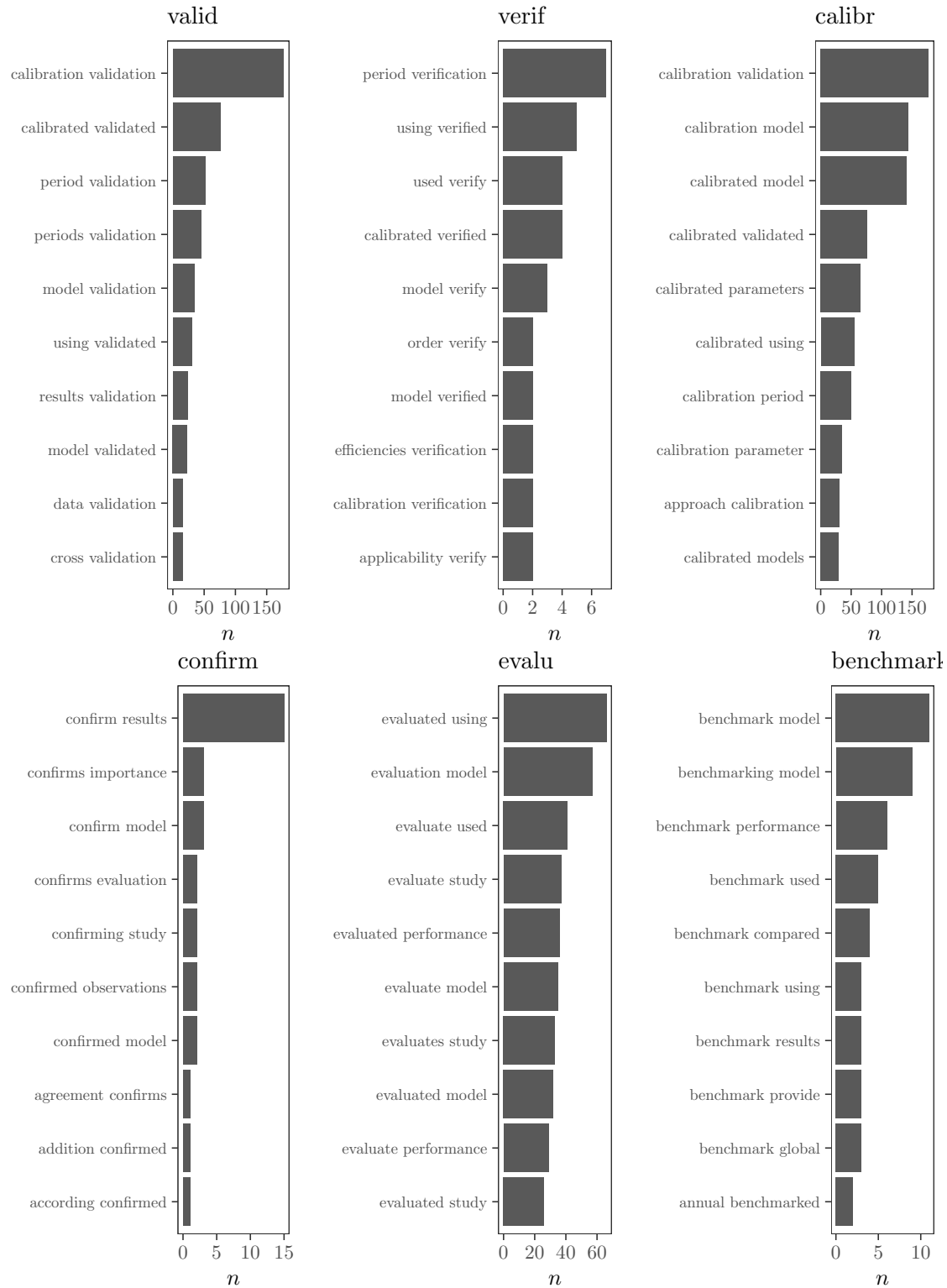
```



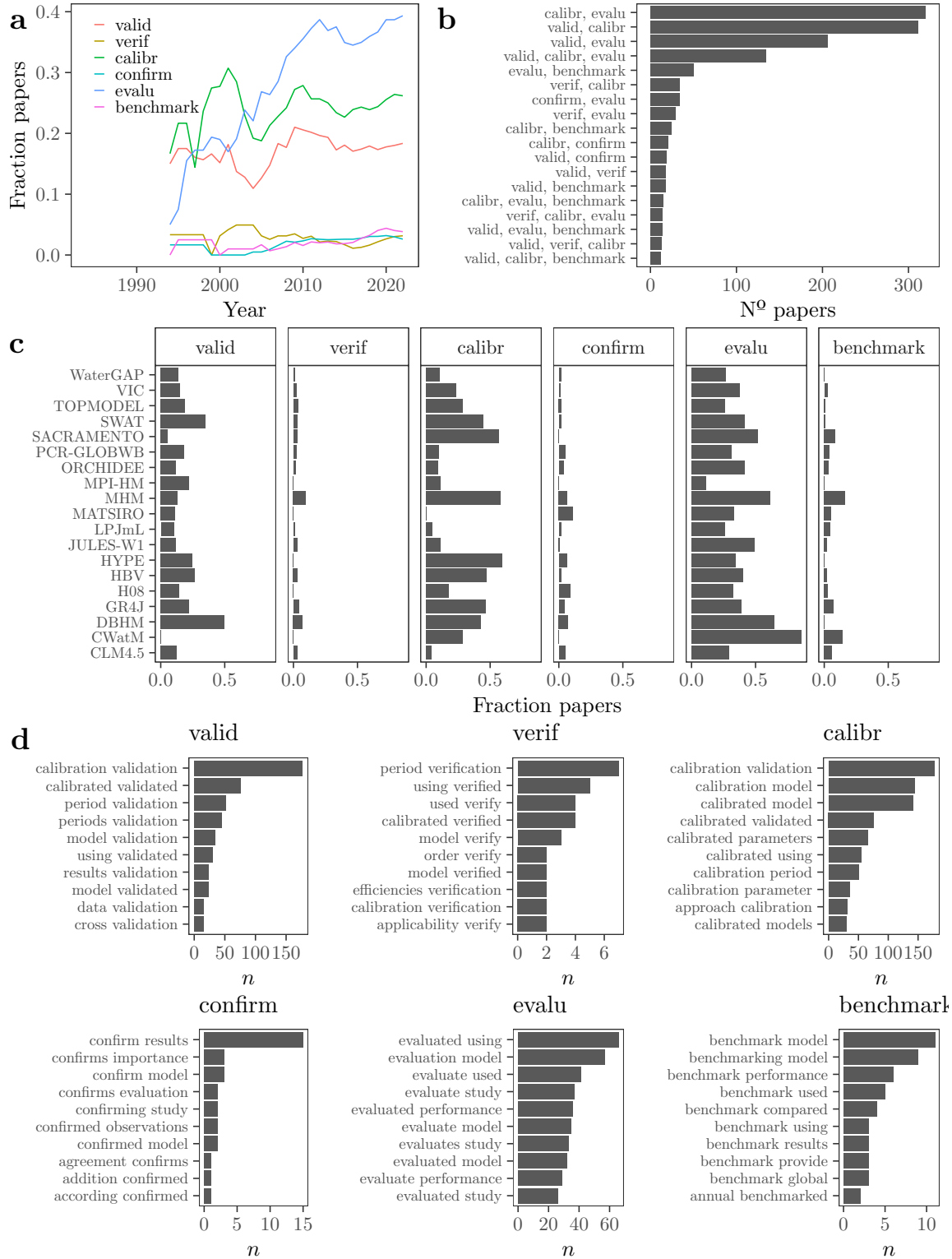
```
# PLOT RESULTS #####

plot.tokens <- plot_grid(token.dt$valid[[2]], token.dt$verif[[2]], token.dt$calibr[[2]],
                        token.dt$confirm[[2]], token.dt$evalu[[2]], token.dt$benchmark[[2]],
                        ncol = 3)

plot.tokens
```



```
plot_grid(plot.merged, plot.tokens, ncol = 1, labels = c("", "d"),
          rel_heights = c(0.57, 0.43))
```



```
# 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] benchmarkme_1.0.8 tidyquant_1.0.7
## [3] quantmod_0.4.25 TTR_0.24.3
## [5] PerformanceAnalytics_2.0.4 xts_0.13.1
## [7] zoo_1.8-12 ggrepel_0.9.5
## [9] LSAfun_0.6.3 rgl_1.1.3
## [11] lsa_0.73.3 SnowballC_0.7.1
## [13] cowplot_1.1.1 scales_1.3.0
## [15] tidytext_0.4.1 pdftools_3.3.3
## [17] tm_0.7-11 NLP_0.2-1
## [19] openxlsx_4.2.5.2 lubridate_1.9.2
## [21] forcats_1.0.0 stringr_1.5.1
## [23] dplyr_1.1.4 purrr_1.0.2
## [25] readr_2.1.4 tidyr_1.3.0
## [27] tibble_3.2.1 ggplot2_3.4.4
## [29] tidyverse_2.0.0 data.table_1.14.99
##
## loaded via a namespace (and not attached):
## [1] tidyselect_1.2.0 filehash_2.4-5 farver_2.1.1
## [4] fastmap_1.1.1 janeaustenr_1.0.0 digest_0.6.34
## [7] timechange_0.2.0 lifecycle_1.0.4 qpdf_1.3.2
## [10] tokenizers_0.3.0 magrittr_2.0.3 compiler_4.3.2
## [13] rlang_1.1.3 tools_4.3.2 utf8_1.2.4
## [16] sensobol_1.1.4 yaml_2.3.7 knitr_1.42
## [19] askpass_1.1 labeling_0.4.3 stopwords_2.3
```

```
## [22] htmlwidgets_1.6.2      curl_5.0.0             xml2_1.3.3
## [25] withr_3.0.0            grid_4.3.2             fansi_1.0.6
## [28] colorspace_2.1-0       iterators_1.0.14       tinytex_0.45
## [31] cli_3.6.2              rmarkdown_2.21         generics_0.1.3
## [34] tikzDevice_0.12.4      rstudioapi_0.15.0     httr_1.4.5
## [37] tzdb_0.3.0             base64enc_0.1-3        vctrs_0.6.5
## [40] Matrix_1.6-1.1         jsonlite_1.8.4         slam_0.1-50
## [43] hms_1.1.3              foreach_1.5.2          glue_1.7.0
## [46] benchmarkmeData_1.0.4  codetools_0.2-19      Quandl_2.11.0
## [49] stringi_1.8.3          gtable_0.3.4           quadprog_1.5-8
## [52] munsell_0.5.0          pillar_1.9.0           htmltools_0.5.5
## [55] R6_2.5.1               Rdpack_2.6             doParallel_1.0.17
## [58] evaluate_0.20          lattice_0.21-9         highr_0.10
## [61] rbibutils_2.2.16       Rcpp_1.0.12            zip_2.3.0
## [64] xfun_0.39              pkgconfig_2.0.3
```

```
## 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
```