One in four water modelling papers is in an echo chamber

R code

Arnald Puy

Contents

1 Preliminary

1.1 Load the packages

```
# PRELIMINARY FUNCTIONS ########################################################

# Load the packages ------------------------------------------------------------

sensobol::load_packages(c("sensobol", "data.table", "tidyverse", "janitor",
                          "igraph", "ggraph", "tidygraph", "cowplot", "viridis",
                          "wesanderson", "parallel", "doParallel", "tm", "scales",
                          "ggforce", "here", "benchmarkme"))

# Custom theme for plots -------------------------------------------------------

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.text = element_text(size = 7.3),
          axis.title = element_text(size = 10),
          legend.key.width = unit(0.4, "cm"),
          legend.key.height = unit(0.4, "cm"),
          legend.key.spacing.y = unit(0, "lines"),
          legend.box.spacing = unit(0, "pt"),
          legend.title = element_text(size = 7.3),
          axis.text.x = element_text(size = 7),
          axis.text.y = element_text(size = 7),
          axis.title.x = element_text(size = 7.3),
          axis.title.y = element_text(size = 7.3),
          plot.title = element_text(size = 8),
          strip.text.x = element_text(size = 7.4),
          strip.text.y = element_text(size = 7.4))
}
```

1.2 Load the functions

```
# SOURCE ALL R FUNCTIONS NEEDED FOR THE STUDY ##################################

r_functions <- list.files(path = here("functions"), pattern = "\\.R$", full.names = TRUE)
lapply(r_functions, source)
```

2 Retrieve and clean Dimensions data

```
# DIMENSIONS DATA ##############################################################

# Vector with the name of the models -------------------------------------------

water.models <- c("WaterGAP", "PCR-GLOBWB", "LPJmL", "CLM4.5", "DBHM",
                  "TOPMODEL", "H08", "JULES-W1", "MPI-HM", "VIC", "SWAT",
                  "GR4J", "HYPE", "HBV", "MATSIRO", "SACRAMENTO", "MHM",
                  "CWatM", "ORCHIDEE")

# Programatically retrieve the datasets ----------------------------------------

dt <- list()

for (i in 1:length(water.models)) {

  dt[[i]] <- fread(paste(water.models[[i]], ".csv", sep = ""), skip = 1) %>%
    clean_names() %>%
    data.table()

}

# Name the slots and flatten the data ------------------------------------------

names(dt) <- water.models
dt.water <- rbindlist(dt, idcol = "Model")

# Retrieve WOS data to check titles --------------------------------------------

wos.dt <- fread("final.dt.csv")
wos.titles <- wos.dt[Model %in% water.models]

# REMOVE DUPLICATED REFERENCES #################################################

# Number of papers in more than one model --------------------------------------

n_occur <- data.frame(table(dt.water$publication_id))
papers_repeated <- data.table(n_occur[n_occur$Freq > 1,])
length(papers_repeated$Var1) # number of repeated papers
```

```
## [1] 2323
```

```
# Fraction of repeated papers over the total -----------------------------------

length(papers_repeated$Var1) / nrow(dt.water)
```

```
## [1] 0.07791903
```

```
# How many papers are repeated twice, three times, etc... ----------------------

papers_repeated[, .(N.repeated.papers = .N), Freq]
```

```
##    Freq N.repeated.papers
## 1:    2              1798
## 2:    4               106
## 3:    6                18
## 4:    3               348
## 5:    5                38
## 6:    8                 5
## 7:    7                 6
## 8:    9                 1
## 9:   11                 3
```

```
# Extract which papers are repeated for which model ----------------------------

dt.sample.repeated <- dt.water[publication_id %in% papers_repeated$Var1] %>%
  .[, .(publication_id, Model, title, source_title_anthology_title)] %>%
  .[order(publication_id)]

dt.sample.repeated
```

```
##       publication_id      Model
##    1: pub.1000120678   TOPMODEL
##    2: pub.1000120678 SACRAMENTO
##    3: pub.1000226548   WaterGAP
##    4: pub.1000226548 PCR-GLOBWB
##    5: pub.1000226548        HBV
##   ---
## 5482: pub.1167654662 PCR-GLOBWB
## 5483: pub.1167736853 PCR-GLOBWB
## 5484: pub.1167736853        MHM
## 5485: pub.1167835489     CLM4.5
## 5486: pub.1167835489   TOPMODEL
##
##    1:                                        Temporal dynamics of model parameter sensitivit
##    2:                                        Temporal dynamics of model parameter sensitivit
##    3:                                                                              Multiscale
##    4:                                                                              Multiscale
##    5:                                                                              Multiscale
##   ---
## 5482: Scenario setup and forcing data for impact model evaluation and impact attribution wit
## 5483:         Tradeoffs Between Temporal and Spatial Pattern Calibration and Their Impacts o
## 5484:         Tradeoffs Between Temporal and Spatial Pattern Calibration and Their Impacts o
## 5485:                                                                 Development of inter-gr
## 5486:                                                                 Development of inter-gr
##          source_title_anthology_title
```

```
##    1:        Water Resources Research
##    2:        Water Resources Research
##    3:     Journal of Hydrometeorology
##    4:     Journal of Hydrometeorology
##    5:     Journal of Hydrometeorology
##   ---
## 5482: Geoscientific Model Development
## 5483:        Water Resources Research
## 5484:        Water Resources Research
## 5485: Geoscientific Model Development
## 5486: Geoscientific Model Development
```

```r
# Randomly retrieve only one of the repeated studies per model -----------------

set.seed(6)
dt.no.repeated <- dt.sample.repeated[,.SD[sample(.N, min(1,.N))], publication_id]

# Setkey to filter and retrieve ------------------------------------------------

res <- setkey(dt.water, publication_id, Model) %>%
  .[J(dt.no.repeated$publication_id, dt.no.repeated$Model)]

# Make the dataset without repeated papers across models -----------------------

final.dt <- rbind(res, dt.water[!publication_id %in% papers_repeated$Var1])

# Check which papers do not have cited bibliography metadata and exclude them --

final.dt <- final.dt[, empty_cited_references:= grepl("^\\s*$", cited_references)] %>%
  .[empty_cited_references == FALSE] %>%
  # Filter dataset to ensure all titles use a water model
  .[tolower(.$title) %in% wos.titles$title.large] %>%
  setnames(., "authors", "from.authors")

# EXTRACT REFERENCES ###########################################################

setnames(final.dt, c("Model", "publication_id"), c("citing_model", "citing_id"))

column_names <- c("authors", "author_id", "source", "year", "volume", "issue",
                  "pagination", "to.doi", "publication_id", "times_cited")

direct.citation <- final.dt %>%
  .[, .(citing_id, cited_references, citing_model, title, doi, pub_year, from.authors)] %>%
  separate_rows(cited_references, sep = ";(?=\\[)") %>%
  separate(col = cited_references, into = column_names, sep = "\\|") %>%
  data.table() %>%
  setnames(., "authors", "to.authors")
```

```
## Warning: Expected 10 pieces. Additional pieces discarded in 16 rows [8733, 9603, 9819,
## 13673, 26551, 28818, 30722, 42918, 76098, 81210, 81628, 83529, 98793, 100483,
## 104944, 106015].
```

```
# ARRANGE DATA FOR NETWORK CITATION ANALYSIS ###################################

# Create a directed graph from the dataset ------------------------------------

edges <- data.table(from = direct.citation$citing_id,
                    to = direct.citation$publication_id,
                    from.title = direct.citation$title,
                    from.model = direct.citation$citing_model,
                    year = direct.citation$year,
                    n.citations = direct.citation$times_cited,
                    to.doi = direct.citation$to.doi,
                    to.authors = direct.citation$to.authors)

# Merge data from citing papers with data from cited papers -------------------

network.dt <- merge(edges, final.dt[, .(citing_id, doi, pub_year, times_cited, from.authors)],
                    by.x = "from", by.y = "citing_id")

# Change column names to clarify ----------------------------------------------

colnames(network.dt)
```

```
##  [1] "from"         "to"           "from.title"   "from.model"   "year"
##  [6] "n.citations"  "to.doi"       "to.authors"   "doi"          "pub_year"
## [11] "times_cited"  "from.authors"
```

```
new_colnames <- c("from", "to", "from.title", "from.model", "to.year", "to.n.citations",
                  "to.doi", "to.authors", "from.doi", "from.year", "from.n.citations", "from.au

setnames(network.dt, colnames(network.dt), new_colnames)

# Reorder columns -------------------------------------------------------------

setcolorder(network.dt, c("from", "to", "from.year", "to.year", "from.authors",
                          "to.authors", "from.n.citations", "to.n.citations",
                          "from.doi", "to.doi", "from.model"))

# Remove square brackets from the to.authors column ---------------------------

network.dt[, to.authors:= gsub("\\[|\\]", "", to.authors)]

# Identify the model of the cited paper ---------------------------------------

tmp <- network.dt[, .(from.model, from)] %>%
  unique()
```

```r
setkey(tmp, "from")

# Define parallel computing ------------------------------------------------

cl <- makeCluster(floor(detectCores() * 0.75))
registerDoParallel(cl)

# Search in parallel
to.model <- foreach(i=1:nrow(network.dt),
                    .packages = "data.table",
                    .combine = "c") %dopar%
  {
    tmp[network.dt[[i, "to"]]]$from.model
  }

# Stop parallel cluster
stopCluster(cl)

# Add vector of model names ------------------------------------------------

network.dt[, to.model:= to.model]

# Turn some columns into numeric -------------------------------------------

columns_to_modify <- grep("citation", names(network.dt), value = TRUE)
network.dt[, (columns_to_modify):= lapply(.SD, as.numeric), .SDcols = columns_to_modify]
```

```
## Warning in lapply(.SD, as.numeric): NAs introduced by coercion
```

```r
# Retrieve only the citations to a paper using a model --------------------

network.dt.final <- network.dt[!is.na(to.model)] %>%
  .[, .(from, to, from.year, to.year, from.doi, to.doi, from.model, to.model,
        from.n.citations, to.n.citations, from.authors, to.authors)]

# Export datasets ----------------------------------------------------------

fwrite(network.dt, "network.dt.csv")
fwrite(network.dt.final, "network.dt.final.csv")
```

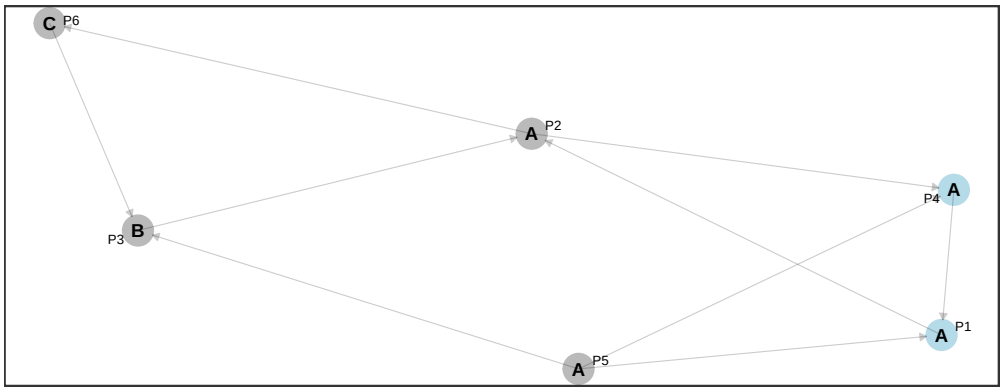
3 Identify echo chambers

3.1 Illustration of a strict and a structural echo chamber

```r
# ILLUSTRATIONS OF ECHO CHAMBERS ###############################################
################################################################################
```

```r
# ILLUSTRATION OF STRICT ECHO CHAMBER ##########################################

# Define nodes ----------------------------------------------------------------

nodes <- tibble(name = paste0("P", 1:6), model = c("A", "A", "B", "A", "A", "C"))

# Define edges: from = citing paper, to = cited paper -------------------------

edges <- tibble(from = c("P1", "P6", "P4", "P5", "P3", "P5", "P5", "P2", "P2"),
                to = c("P2", "P3", "P1", "P1", "P2", "P3", "P4", "P6", "P4")) %>%
  mutate(from = match(from, nodes$name),
         to = match(to, nodes$name)) %>%
  data.table()

# Create the graph ------------------------------------------------------------

graph <- tbl_graph(nodes = nodes, edges = edges, directed = TRUE)

# Add onto the graph ----------------------------------------------------------

graph <- graph %>%
  activate(edges) %>%
  mutate(from.model = .N()$model[from],
         to.model = .N()$model[to],
         same_model = from.model == to.model)

# Extract edges ---------------------------------------------------------------

edges_df <- graph %E>%
  as_tibble() %>%
  data.table()

# Compute same-model citation proportions per node (outgoing) -----------------

edges.df.outgoing <- edges_df[, .(
  same_model_outgoing = sum(same_model, na.rm = TRUE),
  prop_same_model_outgoing = sum(same_model, na.rm = TRUE) / .N), .(node = to)]

# Compute same-model citation proportions per node (incoming) -----------------

edges.df.incoming <- edges_df[, .(
  same_model_incoming = sum(same_model, na.rm = TRUE),
  prop_same_model_incoming = sum(same_model, na.rm = TRUE) / .N), .(node = from)]

# Merge summaries -------------------------------------------------------------

merged_model_citation_stats <- merge(edges.df.outgoing, edges.df.incoming,
```

```
                                    by = "node", all = TRUE)

# Add node index to graph and merge node stats back into graph -----------------

graph <- graph %>%
  activate(nodes) %>%
  mutate(node = row_number()) %>%
  left_join(merged_model_citation_stats, by = "node") %>%
  mutate(strict_echo_chamber = prop_same_model_outgoing == 1 & prop_same_model_incoming == 1)

# Plot -------------------------------------------------------------------------

set.seed(1)
example_strict <- ggraph(graph , layout = "igraph", algorithm = "nicely") +
  geom_node_point(aes(color = strict_echo_chamber), alpha = 0.9, size = 5) +
  geom_edge_link(alpha = 0.2, edge_width = 0.2,
                 arrow = arrow(length = unit(1, "mm"), type = "closed"),
                 end_cap = circle(2, "mm"),
                 show.legend = FALSE) +
  scale_color_manual(values = c("TRUE" = "lightblue", "FALSE" = "grey70")) +
  geom_node_text(aes(label = name), size = 1.8, repel = TRUE) +
  geom_node_text(aes(label = model), size = 2.5, fontface = "bold", color = "black") +
  labs(x = "", y = "") +
  theme_AP() +
  guides(color = "none") +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank())

example_strict
```




```
# # ILLUSTRATION OF STRUCTURAL ECHO CHAMBER ###################################

# Create a realistic cluster-based citation network ----------------------------
```

```r
set.seed(42)

# Simulate nodes ---------------------------------------------------------------

models <- rep(c("A", "B", "C"), times = c(10, 8, 7))
nodes <- tibble(name = paste0("P", seq_along(models)), model = models)

# Simulate intra- and inter-model citations ------------------------------------

edges <- tibble(from = integer(), to = integer())

# Strong intra-model links -----------------------------------------------------

for (m in unique(models)) {
  group <- which(models == m)
  for (i in group) {
    # cite 2-3 others within the same model
    targets <- sample(setdiff(group, i), size = sample(2:3, 1))
    edges <- bind_rows(edges, tibble(from = i, to = targets))
  }
}

# Cross-model citations --------------------------------------------------------

cross_citations <- tibble(
  from = sample(1:length(models), 10),
  to   = sample(1:length(models), 10)
) %>%
  filter(models[from] != models[to])  # ensure cross-model

edges <- bind_rows(edges, cross_citations)

# Build the graph --------------------------------------------------------------

graph <- tbl_graph(nodes = nodes, edges = edges, directed = TRUE)
model_vec <- graph %N>% pull(model)

graph <- graph %>%
  activate(nodes) %>%
  mutate(node_id = row_number())

# Calculate homophily ----------------------------------------------------------

node_ids <- graph %N>% pull(node_id)

first_order_h <- map_dbl(node_ids, function(i) {
  n1 <- ego(as.igraph(graph), order = 1, nodes = i, mode = "all")[[1]]
```

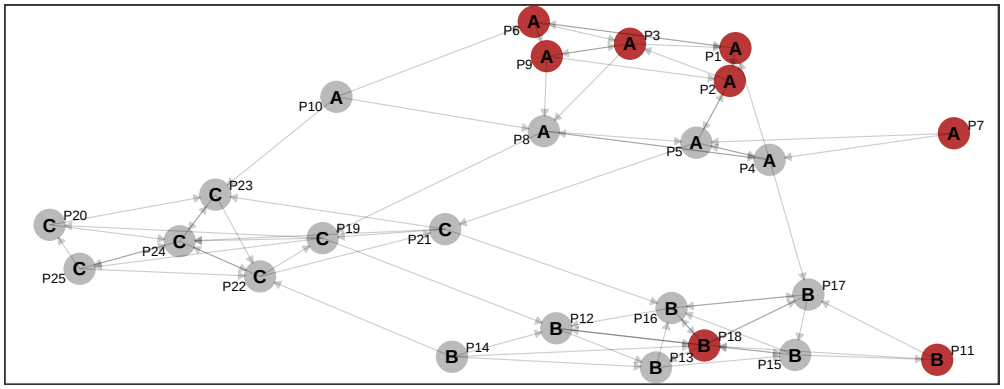
```
  n1 <- setdiff(n1, i)
  if (length(n1) == 0) return(NA_real_)
  max(table(model_vec[n1])) / length(n1)
})

second_order_h <- map_dbl(node_ids, function(i) {
  n1 <- ego(as.igraph(graph), order = 1, nodes = i, mode = "all")[[1]]
  n2 <- unique(unlist(ego(as.igraph(graph), order = 1, nodes = n1, mode = "all")))
  n2 <- setdiff(n2, c(i, n1))
  if (length(n2) == 0) return(NA_real_)
  max(table(model_vec[n2])) / length(n2)
})

# Add onto the graph ------------------------------------------------------------

graph <- graph %>%
  mutate(first_order_homophily = first_order_h,
         second_order_homophily = second_order_h,
         structural_echo_chamber = first_order_homophily > 0.9 &
           second_order_homophily > 0.5)

# Print the nodes with first and second order homophily

graph %N>%
  data.frame()
```

```
##    name model node_id first_order_homophily second_order_homophily
## 1    P1     A       1             1.0000000              0.8333333
## 2    P2     A       2             1.0000000              0.8000000
## 3    P3     A       3             1.0000000              0.7500000
## 4    P4     A       4             0.8000000              0.4166667
## 5    P5     A       5             0.8000000              0.4000000
## 6    P6     A       6             1.0000000              0.7500000
## 7    P7     A       7             1.0000000              0.6000000
## 8    P8     A       8             0.8333333              0.5000000
## 9    P9     A       9             1.0000000              0.8000000
## 10  P10     A      10             0.6666667              0.5000000
## 11  P11     B      11             1.0000000              0.8000000
## 12  P12     B      12             0.8333333              0.6250000
## 13  P13     B      13             1.0000000              0.5000000
## 14  P14     B      14             0.7500000              0.5555556
## 15  P15     B      15             1.0000000              0.5000000
## 16  P16     B      16             0.8333333              0.5000000
## 17  P17     B      17             0.8333333              0.5714286
## 18  P18     B      18             1.0000000              0.6000000
## 19  P19     C      19             0.7142857              0.4545455
## 20  P20     C      20             1.0000000              0.4000000
```

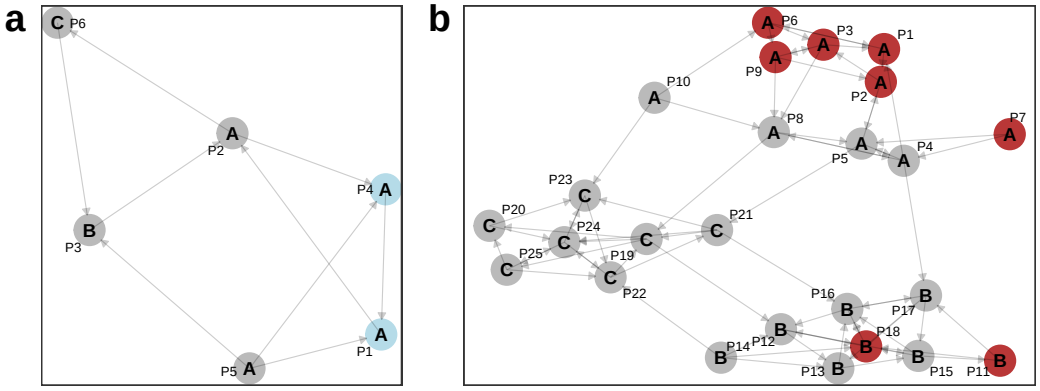
```
## 21  P21     C      21             0.6666667             0.4615385
## 22  P22     C      22             0.8333333             0.5000000
## 23  P23     C      23             0.8000000             0.4285714
## 24  P24     C      24             1.0000000             0.5000000
## 25  P25     C      25             1.0000000             0.4000000
##    structural_echo_chamber
## 1                     TRUE
## 2                     TRUE
## 3                     TRUE
## 4                    FALSE
## 5                    FALSE
## 6                     TRUE
## 7                     TRUE
## 8                    FALSE
## 9                     TRUE
## 10                   FALSE
## 11                    TRUE
## 12                   FALSE
## 13                   FALSE
## 14                   FALSE
## 15                   FALSE
## 16                   FALSE
## 17                   FALSE
## 18                    TRUE
## 19                   FALSE
## 20                   FALSE
## 21                   FALSE
## 22                   FALSE
## 23                   FALSE
## 24                   FALSE
## 25                   FALSE
```

```r
# Plot ------------------------------------------------------------------------

set.seed(1)

example_structural <- ggraph(graph , layout = "igraph", algorithm = "nicely") +
  geom_node_point(aes(color = structural_echo_chamber), alpha = 0.9, size = 5) +
  geom_edge_link(edge_width = 0.2, alpha = 0.2,
                 arrow = arrow(length = unit(1, "mm"), type = "closed"),
                 end_cap = circle(2, "mm"), show.legend = FALSE) +
  scale_color_manual(values = c("TRUE" = "firebrick", "FALSE" = "grey70")) +
  geom_node_text(aes(label = name), size = 1.8, repel = TRUE) +
  geom_node_text(aes(label = model), size = 2.5, fontface = "bold", color = "black") +
  labs(x = "", y = "") +
  theme_AP() +
  guides(color = "none") +
  theme(axis.text.y = element_blank(),
```

```
        axis.ticks.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank())

example_structural
```




```
plot.all.examples <- plot_grid(example_strict, example_structural, ncol = 2,
                               rel_widths = c(0.4, 0.6), labels = "auto")

plot.all.examples
```




```
# CREATE GRAPH ########################################################################

# Create graph from citation edge list ----------------------------------------

graph <- as_tbl_graph(network.dt.final, directed = TRUE) %>%
  mutate(year = coalesce(
           as.integer(network.dt.final$from.year[match(name, network.dt.final$from)]),
           as.integer(network.dt.final$to.year[match(name, network.dt.final$to)])),
         model = coalesce(
           network.dt.final$from.model[match(name, network.dt.final$from)],
           network.dt.final$to.model[match(name, network.dt.final$to)]),
         doi = coalesce(
           network.dt.final$from.doi[match(name, network.dt.final$from)],
           network.dt.final$to.doi[match(name, network.dt.final$to)]),
```

```
         total_citations = coalesce(
           network.dt.final$from.n.citations[match(name, network.dt.final$from)],
           network.dt.final$to.n.citations[match(name, network.dt.final$to)]),
         indegree = centrality_degree(mode = "in"),
         outdegree = centrality_degree(mode = "out"),
         betweenness = centrality_betweenness(directed = TRUE))

# Extract node and edge data ---------------------------------------------------

nodes_df <- graph %N>%
  as_tibble() %>%
  data.table()

edges_df <- graph %E>%
  as_tibble() %>%
  data.table() %>%
  .[, same_model:= from.model == to.model]

# Compute proportion of same-model citations (outgoing) ------------------------

edges.df.outgoing <- edges_df[, .(same_model_outgoing = sum(same_model, na.rm = TRUE),
                                  prop_same_model_outgoing = sum(same_model, na.rm = TRUE) / .N),
                              from.doi]

# Compute proportion of same-model citations (ingoing) -------------------------

edges.df.incoming <- edges_df[, .(same_model_incoming = sum(same_model, na.rm = TRUE),
                                  prop_same_model_incoming = sum(same_model, na.rm = TRUE) / .N),
                              to.doi]

# ADD ONTO THE GRAPH ###########################################################

graph <- graph %>%
  activate(nodes) %>%
  left_join(edges.df.outgoing, by = c("doi" = "from.doi")) %>%
  left_join(edges.df.incoming, by = c("doi" = "to.doi"))

# IDENTIFY ECHO CHAMBERS #######################################################

# Identify echo chambers and structural echo chambers --------------------------

# Strict definition of echo chamber: a paper is in an echo chamber if it only
# cites papers using the same model and is cited by papers using the same model
# ------------------------------------------------------------------------------

echo_chamber_dt <- graph %>%
  activate(nodes) %>%
```

```r
  data.frame() %>%
  data.table() %>%
  .[, strict_echo_chamber:= prop_same_model_outgoing >= 0.9 & prop_same_model_incoming >= 0.9]
  .[, .(doi, strict_echo_chamber)]

# MERGE BACK ONTO THE GRAPH ##########################################################

graph <- graph %>%
  left_join(echo_chamber_dt, by = "doi")

# CALCULATE STRUCTURAL ECHO CHAMBER ##################################################

# Structural echo chamber: paper whose first-order neighborhood (directly connected papers)
# is composed of more than 90% papers using the same model, and whose second-order
# neighborhood (neighbors of neighbors, excluding direct ones) contains more than 50%
# papers using the same model----------------------------------------------------

graph <- graph %>%
  activate(nodes) %>%
  mutate(node_id = row_number(),
         model_vec = list(V(as.igraph(graph))$model)) %>%

  # Compute first-order homophily ------------------------------------------
# ------------------------------------------------------------------------------

  mutate(first_order_homophily = map_dbl(node_id, function(i) {

      neighbors_1 <- ego(as.igraph(graph), order = 1, nodes = i, mode = "all")[[1]]
      neighbors_1 <- setdiff(neighbors_1, i)  # remove self just in case

      if (length(neighbors_1) == 0) return(NA_real_)

      models <- V(as.igraph(graph))$model[neighbors_1]
      max(table(models)) / length(models)

    }),

    # Compute second-order homophily -----------------------------------------
    # ------------------------------------------------------------------------

    second_order_homophily = map_dbl(node_id, function(i) {

      # 1st-order neighbors ----------------------------------------------

      neighbors_1 <- ego(as.igraph(graph), order = 1, nodes = i, mode = "all")[[1]]

      # 2nd-order neighbors (neighbors of 1st-order) ---------------------------
```

```
      neighbors_2 <- unique(unlist(ego(as.igraph(graph), order = 1,
                                       nodes = neighbors_1, mode = "all")))

      # Remove self and 1st-order neighbors to get true second-order only ------

      neighbors_2 <- setdiff(neighbors_2, c(i, neighbors_1))

      if (length(neighbors_2) == 0) return(NA_real_)

      models <- V(as.igraph(graph))$model[neighbors_2]
      max(table(models)) / length(models)

    }),

    # We compute: if a node does not have enough second-order structure,
    # we consider it does not fit into a structural echo chamber ---------------

    structural_echo_chamber = if_else((first_order_homophily > 0.9 &
                                         second_order_homophily > 0.5),
      TRUE, FALSE, missing = FALSE
    )
  )

# ADD FEATURES TO EDGES ########################################################

graph <- graph %>%
  activate(edges) %>%
  mutate(edge_color = .N()$structural_echo_chamber[to],
         from.authors = strsplit(from.authors, ";\\s*"),
         from.authors = lapply(from.authors, to_last_first_initial_fun),
         to.authors = strsplit(to.authors, ";\\s*"),
         to.authors = lapply(to.authors, to_last_first_initial_fun),
         shared_author = mapply(function(from, to) {
           length(intersect(from, to)) > 0}, from.authors, to.authors),
         citation_type = fifelse(
           from.model == to.model & shared_author, "Self (model + author)",
           fifelse(from.model == to.model & !shared_author, "Self (model only)",
                   fifelse(from.model != to.model & shared_author, "Cross (author only)",
                           "Cross (none)"))))

# CREATE FINAL NODE AND EDGE DATASET AND EXPORT ################################

final_nodes_dataset <- graph %N>%
  as_tibble() %>%
  data.table()

final_edges_dataset <- graph %E>%
```

```
  as_tibble() %>%
  data.table()

fwrite(final_nodes_dataset, "./datasets/final_nodes_dataset.csv")
fwrite(final_edges_dataset, "./datasets/final_edges_dataset.csv")
```

3.2 Citations in papers in structural echo chambers vs those that are not

```
# ARE PAPERS IN STRUCTURAL ECHO CHAMBERS LESS CITED? ############################

final_nodes_dataset[, .(mean_citations = mean(total_citations),
                        sd = sd(total_citations),
                        median = median(total_citations)), structural_echo_chamber]
```

```
##    structural_echo_chamber mean_citations        sd median
##                     <lgcl>          <num>     <num>  <num>
## 1:                   FALSE       61.43515 118.90758     24
## 2:                    TRUE       37.34677  58.49826     19
```

```
wilcox.test(log10(total_citations + 1) ~ structural_echo_chamber,
            data = final_nodes_dataset, alternative = "two.sided")
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  log10(total_citations + 1) by structural_echo_chamber
## W = 524058, p-value = 4.075e-05
## alternative hypothesis: true location shift is not equal to 0
```

```
# Check effect size ------------------------------------------------------------
# On average, a randomly selected paper inside an echo chamber has a XX-XX
# probability of receiving less citations than a randomly selected
# paper not in the echo chamber.
effsize::cliff.delta(total_citations ~ structural_echo_chamber,
                     data = final_nodes_dataset)
```

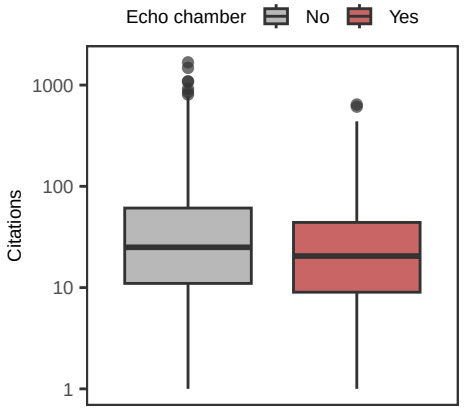
```
##
## Cliff's Delta
##
## delta estimate: 0.1129107 (negligible)
## 95 percent confidence interval:
##     lower     upper
## 0.0604537 0.1647457
```

```
# Distribution of citations to papers in and outside echo chamber --------------

boxplots.echo.chamber <- ggplot(final_nodes_dataset,
                                aes(structural_echo_chamber, total_citations)) +
  geom_boxplot(aes(fill = structural_echo_chamber), alpha = 0.7) +
```

```
  scale_y_log10() +
  scale_fill_manual(values = c("TRUE" = "firebrick", "FALSE" = "gray60"),
                    labels = c("TRUE" = "Yes", "FALSE" = "No"),
                    name = "Echo chamber") +
  labs(x = "", y = "Citations") +
  theme_AP() +
  theme(legend.position = "top",
        axis.ticks.x = element_blank(),
        axis.text.x = element_blank())

boxplots.echo.chamber
```

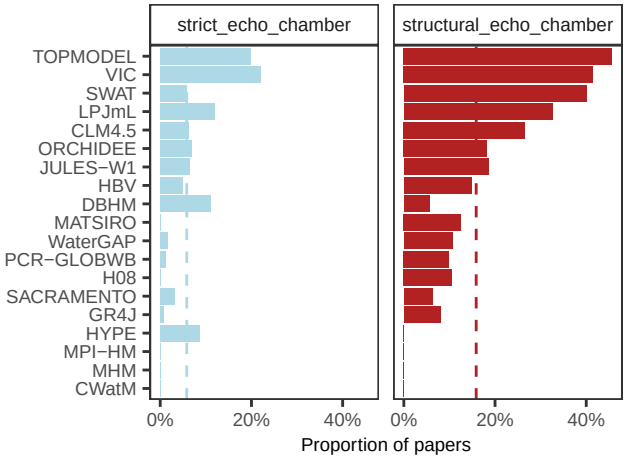
```
## Warning in scale_y_log10(): log-10 transformation introduced infinite values.
```

```
## Warning: Removed 58 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



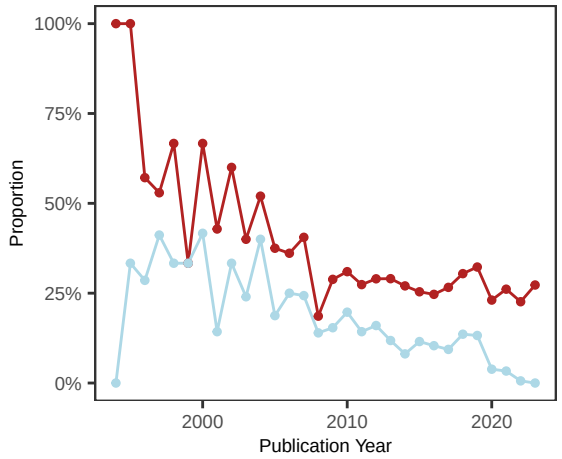

```
# PLOT ##########################################################################

total_papers_by_model <- final_nodes_dataset[, .(total_papers = .N), model]

# Fraction of papers in strict and structural echo chambers ---------------------

# Compute mean values

mean_echo_dt <- final_nodes_dataset %>%
  melt(., measure.vars = c("strict_echo_chamber", "structural_echo_chamber")) %>%
  .[, sum(value, na.rm = TRUE), .(model, variable)] %>%
  merge(., total_papers_by_model, by = "model") %>%
  .[, prop:= V1 / total_papers] %>%
  .[, .(mean_value = mean(prop)), variable]

plot.echo.chambers <- final_nodes_dataset %>%
  melt(., measure.vars = c("strict_echo_chamber", "structural_echo_chamber")) %>%
  .[, sum(value, na.rm = TRUE), .(model, variable)] %>%
  merge(., total_papers_by_model, by = "model") %>%
```

```
  .[, prop:= V1 / total_papers] %>%
  ggplot(., aes(reorder(model, prop), prop, fill = variable)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("strict_echo_chamber" = "lightblue",
                               "structural_echo_chamber" = "firebrick")) +
  facet_wrap(~variable) +
  geom_hline(data = mean_echo_dt,
             aes(yintercept = mean_value, color = variable),
             linetype = "dashed", linewidth = 0.5) +
  scale_color_manual(values = c("strict_echo_chamber" = "lightblue",
                               "structural_echo_chamber" = "firebrick")) +
  labs(x = "", y = "Proportion of papers") +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1),
                     breaks = breaks_pretty(n = 3)) +
  coord_flip() +
  theme_AP() +
  theme(legend.position = "none")

plot.echo.chambers
```

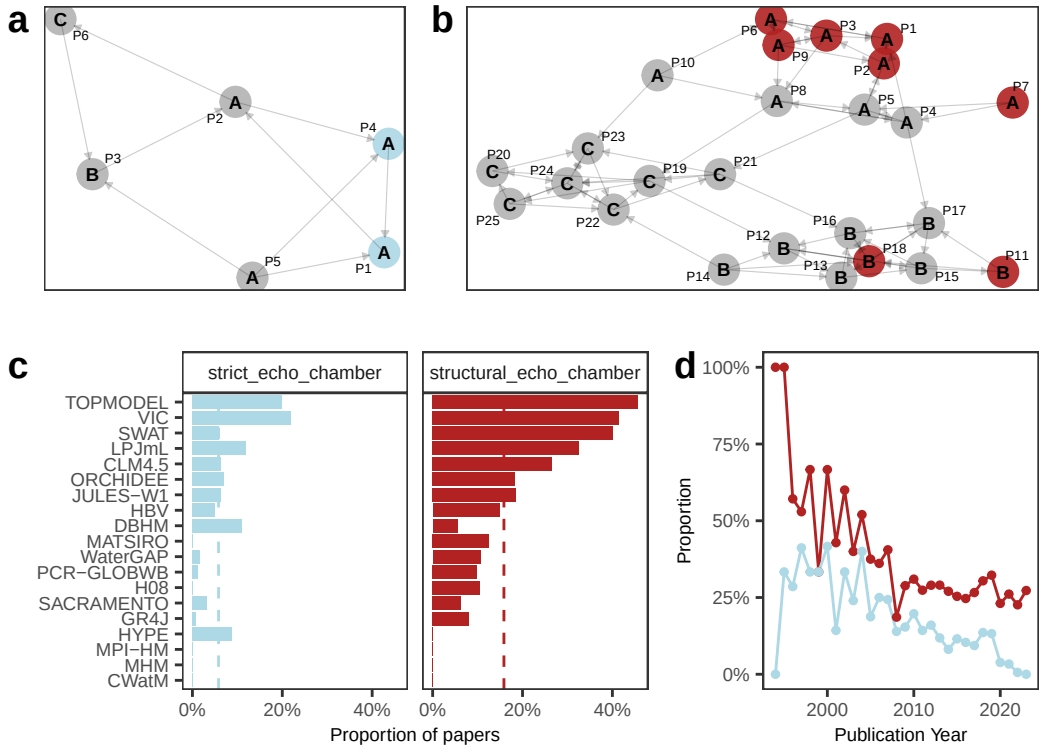



```
# PLOT ########################################################################

# Fraction of papers in structural echo chambers over time ---------------------

plot.echo.time <- final_nodes_dataset[, .( total_papers = .N,
                          structural_papers = sum(structural_echo_chamber, na.rm = TRUE),
                          strict_papers = sum(strict_echo_chamber, na.rm = TRUE)), year] %>%
  .[, prop_structural:= structural_papers / total_papers] %>%
  .[, prop_strict:= strict_papers / total_papers] %>%
  melt(., measure.vars = c("prop_structural", "prop_strict")) %>%
  ggplot(., aes(year, value, color = variable)) +
  scale_color_manual(values = c("prop_strict" = "lightblue",
                               "prop_structural" = "firebrick")) +
  geom_line() +
```

```
  geom_point(size = 1) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  labs(x = "Publication Year", y = "Proportion") +
  theme_AP() +
  theme(legend.position = "none")

plot.echo.time
```

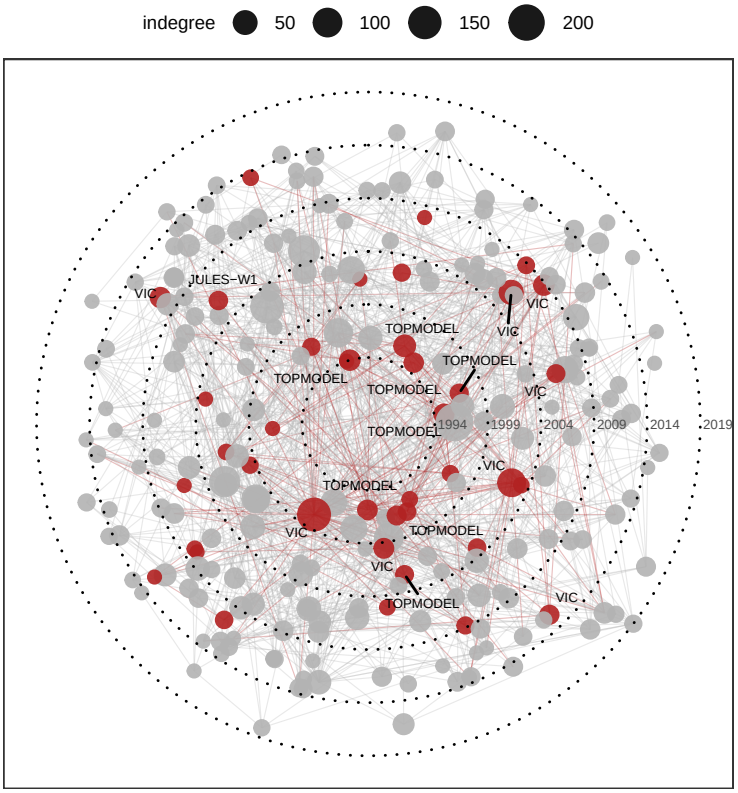



```
# MERGE PLOTS ######################################################################

bottom <- plot_grid(plot.echo.chambers, plot.echo.time, ncol = 2,
                    rel_widths = c(0.63, 0.37), labels = c("c", "d"))
plot_grid(plot.all.examples, bottom, ncol = 1, rel_heights = c(0.45, 0.55))
```



```r
# PLOT IN POLAR COORDINATES ##########################################################

# Filter and prepare polar layout ------------------------------------------

graph_polar <- graph %>%
  activate(nodes) %>%
  filter(indegree >= 10) %>%
  mutate(radius = rescale(year, to = c(0.2, 1)),
         angle = runif(n(), min = 0, max = 2 * pi),
         x = radius * cos(angle),
         y = radius * sin(angle))

# Create manual layout by pulling x and y from node data -----------------------

layout_polar <- graph_polar %N>% as_tibble()

# Choose years for labeling concentric rings -----------------------------------

years_to_label <- seq(min(layout_polar$year, na.rm = TRUE),
                      max(layout_polar$year, na.rm = TRUE),
                      by = 5)

year_radius <- rescale(years_to_label, to = c(0.2, 1))

# Plot -------------------------------------------------------------------------

polar.plot <- ggraph(graph_polar, layout = "manual", x = layout_polar$x, y = layout_polar$y) +
  geom_edge_link(aes(color = edge_color), alpha = 0.3, edge_width = 0.2, show.legend = FALSE) +
  geom_node_point(aes(size = indegree, color = structural_echo_chamber), alpha = 0.9) +
  scale_edge_color_manual(values = c("TRUE" = "firebrick", "FALSE" = "grey70"), guide = "none") +
  scale_color_manual(values = c("TRUE" = "firebrick", "FALSE" = "grey70")) +
  geom_node_text(aes(label = ifelse(indegree > 15 & structural_echo_chamber == TRUE, model, "")),

  # Year rings ------------------------------------------------

geom_circle(data = tibble(r = year_radius, year = years_to_label),
            aes(x0 = 0, y0 = 0, r = r),
            color = "black", linetype = "dotted") +

  # Year labels -----------------------------------------------

geom_text(data = tibble(r = year_radius, year = years_to_label),
          aes(x = r, y = 0, label = year),
          hjust = -0.1, vjust = 0.5, size = 1.8, color = "grey30") +
  coord_fixed() +
  scale_size(range = c(2, 6)) +
  theme_AP() +
```

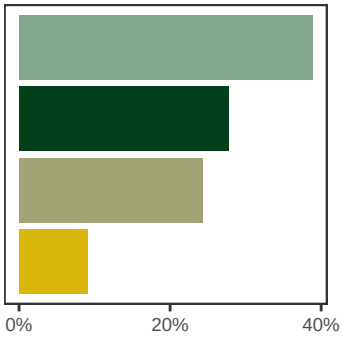
```
  labs(x = "", y = "") +
  guides(color = "none") +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank(),
        legend.position = "top")

polar.plot
```



4 Authorship study

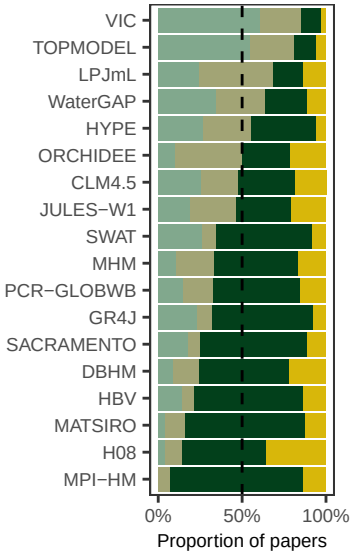
```
##############################################################################
##############################################################################

# CO-AUTHORSHIP STUDY ##########################################################

# STRONGEST FORM OF SELF-CITATION:
# Same model and shared author in the citing and cited papers: self (model + author)

# MODEL HOMOPHILY WITHOUT PERSONAL OVERLAP:
# Same model and different author in the citing and cited papers: self (model only)

# CROSSING MODEL BOUNDARIES BUT SHARED AUTHORSHIP:
```

```
# Different model and shared author in the citing and cited papers: cross (author only)

# PUREST FORM OF EXTERNAL ENGAGEMENT: CITATION CROSSES MODELS AND NO AUTHORS ARE SHARED
# Different model and different author in the citing and cited papers: cross (none)

# Bar plot of total citations by citation type ---------------------------------

selected_colors <- "Cavalcanti1"

plot.bars <- final_edges_dataset[, .(prop = .N / nrow(final_edges_dataset)),
                                 citation_type] %>%
  ggplot(aes(x = reorder(citation_type, prop), prop, fill = citation_type)) +
  geom_col(show.legend = FALSE) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1),
                     breaks = breaks_pretty(n = 3)) +
  scale_fill_manual(values = wes_palette(selected_colors)) +
  labs(x = "", y = "") +
  coord_flip() +
  theme_AP() +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank())

plot.bars
```




```
# Aggregate by from.model and citation_type ------------------------------------

tmp <- final_edges_dataset[, .N, .(from.model, citation_type)] %>%
  .[, total := sum(N), from.model] %>%
  .[, prop := N / total]

tmp2 <- tmp[citation_type %in% c("Self (model only)", "Self (model + author)"),
                                        .(order_val = sum(prop)), from.model]

plot.bars.model <- merge(tmp, tmp2, by = "from.model") %>%
  .[, from.model:= reorder(from.model, order_val)] %>%
  ggplot(., aes(from.model, prop, fill = citation_type)) +
  geom_col() +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1),
```

```
                      breaks = breaks_pretty(n = 3)) +
  labs(x = "", y = "Proportion of papers", fill = "Citation Type") +
  scale_fill_manual(values = wes_palette(selected_colors)) +
  coord_flip() +
  geom_hline(yintercept = 0.5, linetype = 2) +
  theme_AP() +
  theme(legend.position = "none") +
  guides(fill = guide_legend(nrow = 2))

plot.bars.model
```

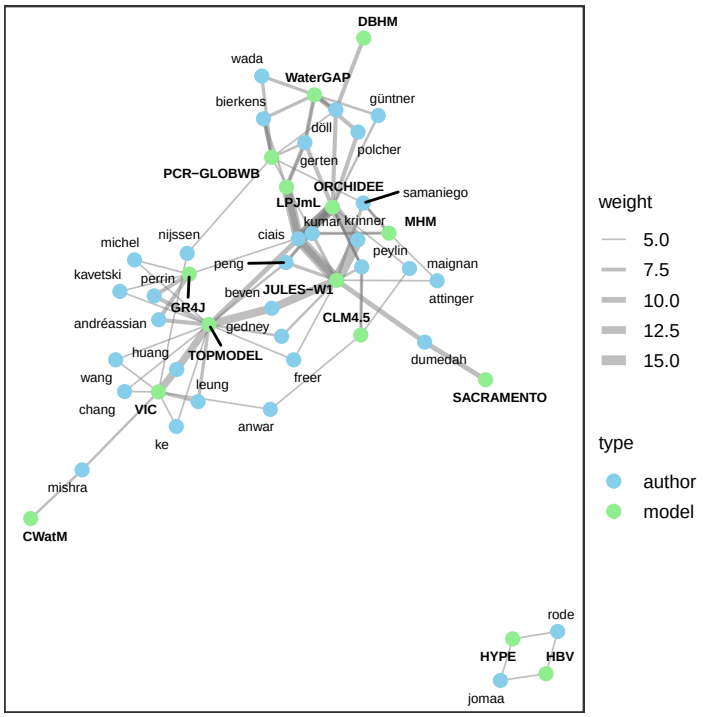



```
# IDENTIFY BRIDGING AUTHORS ##########################################################

cross_author_only <- final_edges_dataset[citation_type == "Cross (author only)"]

cross_author_only[, shared_authors:= mapply(function(from, to) {
  shared <- intersect(from, to)
  if (length(shared) > 0) paste(shared, collapse = "; ") else NA_character_
}, from.authors, to.authors)]

# Flatten the shared authors into one long vector ------------------------------

bridging_authors <- cross_author_only[, .(author = unlist(strsplit(shared_authors, ";\\s*"))),
                                      .(from.model, to.model)]

# Count how often each bridging author appears ---------------------------------

author_counts <- bridging_authors[, .N, author] %>%
  .[order(-N)]

head(author_counts, 10)
```

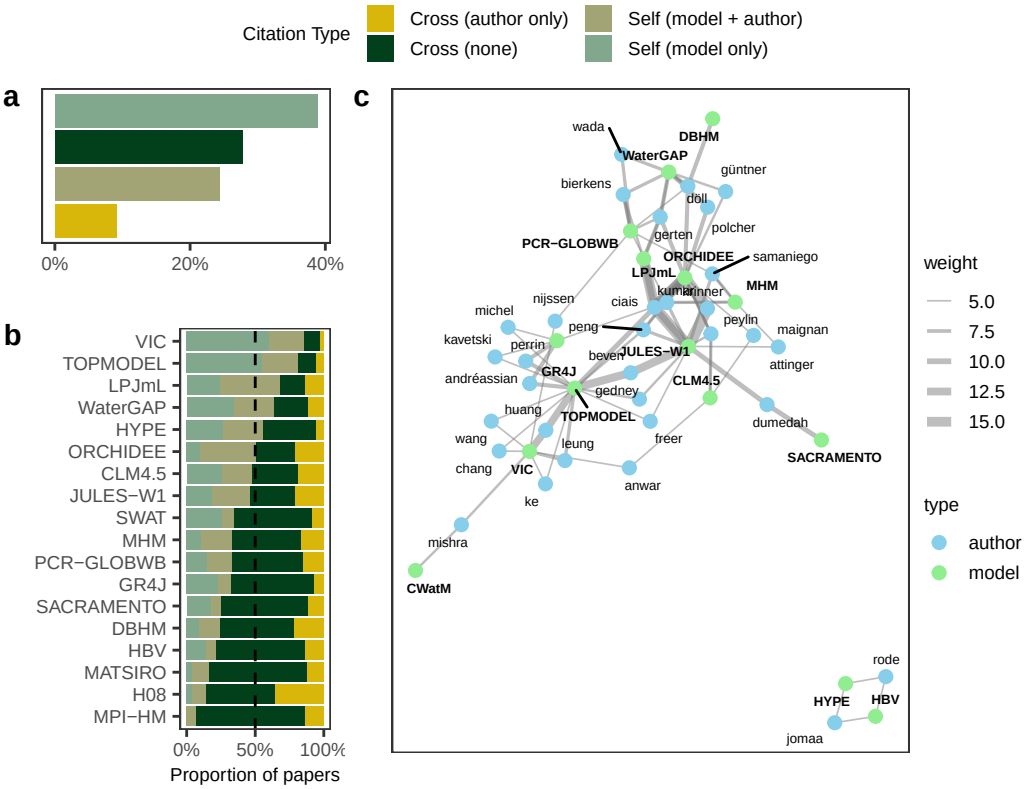
```
##          author     N
##          <char> <int>
##  1:     ciais,p    80
##  2:     kumar,r    44
##  3: samaniego,l    44
##  4:  hanasaki,n    35
##  5:    gerten,d    33
##  6:      döll,p    28
##  7:    peylin,p    23
##  8:     huang,m    20
##  9:   krinner,g    20
## 10:      peng,s    20
```

```r
# Retrieve only the authors linked more than 5 times ----------------------------

author_model_links <- bridging_authors[, .N, .(author, from.model, to.model)] %>%
  .[order(-N)] %>%
  .[N >= 5]

# Create long edge list  ---------------------------------------------------------

edges_long <- melt(author_model_links, id.vars = c("author", "N"),
                   measure.vars = c("from.model", "to.model"),
                   value.name = "model")[, .(author, model, N)]

edges_long <- unique(edges_long)
edge_list <- edges_long[, .(from = author, to = model, weight = N)]

# Create tidygraph object --------------------------------------------------------

g_tbl <- tbl_graph(nodes = tibble(name = unique(c(edge_list$from, edge_list$to))),
                   edges = edge_list,
                   directed = FALSE) %>%
  mutate(type = if_else(name %in% author_model_links$author, "author", "model"),
         fontface = if_else(type == "model", "bold", "plain"),
         label = if_else(type == "author", sub(",.*", "", name), name))

# Plot ---------------------------------------------

# Set seed for reproducibility ----------------------

set.seed(10)

plot.network.authors <- ggraph(g_tbl, layout = "igraph", algorithm = "nicely") +
  geom_edge_link(aes(width = weight), alpha = 0.5, color = "gray50") +
  geom_node_point(aes(color = type), size = 2) +
  geom_node_text(aes(label = label, fontface = fontface), repel = TRUE, size = 1.8) +
  scale_color_manual(values = c("author" = "skyblue", "model" = "lightgreen")) +
```

```
  scale_edge_width(range = c(0.3, 2)) +
  theme_AP() +
  labs(x = "", y = "") +
  theme(axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank())

plot.network.authors
```




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

a <- plot_grid(plot.bars, plot.bars.model, ncol = 1, rel_heights = c(0.33, 0.67),
               labels = "auto", label_size = 11)

legend <- get_legend_fun(plot.bars.model + theme(legend.position = "top"))

bottom <- plot_grid(a, plot.network.authors, ncol = 2, rel_widths = c(0.335, 0.675),
                    labels = c("", "c"), label_size = 11)

plot_grid(legend, bottom, rel_heights = c(0.1, 0.9), ncol = 1)
```


Citation Type
Cross (author only)
Cross (none)
Self (model + author)
Self (model only)
a
0%
20%
40%
b
VIC
TOPMODEL
LPJmL
WaterGAP
HYPE
ORCHIDEE
CLM4.5
JULES−W1
SWAT
MHM
PCR−GLOBWB
GR4J
SACRAMENTO
DBHM
HBV
MATSIRO
H08
MPI−HM
0%
50%
100%
Proportion of papers
c
wada
DBHM
WaterGAP
güntner
bierkens
döll
polcher
gerten
PCR−GLOBWB
ORCHIDEE
samaniego
LPJmL
MHM
kumar
ciais
nijssen
michel
peng
peylin
maignan
kavetski
perrin
beven
JULES−W1
attinger
andréassian
GR4J
gedney
CLM4.5
huang
TOPMODEL
dumedah
wang
leung
freer
SACRAMENTO
chang
VIC
anwar
ke
mishra
CWatM
rode
HYPE
HBV
jomaa
weight
5.0
7.5
10.0
12.5
15.0
type
author
model

5 Session Information

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

sessionInfo()
```

```
## R version 4.3.3 (2024-02-29)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS 15.4.1
##
## 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 here_1.0.1        ggforce_0.4.2     scales_1.3.0
##  [5] tm_0.7-15         NLP_0.3-2         doParallel_1.0.17 iterators_1.0.14
##  [9] foreach_1.5.2     wesanderson_0.3.7 viridis_0.6.5     viridisLite_0.4.2
## [13] cowplot_1.1.3     tidygraph_1.3.1   ggraph_2.2.1      igraph_2.1.1
## [17] janitor_2.2.0     lubridate_1.9.3   forcats_1.0.0     stringr_1.5.1
## [21] dplyr_1.1.4       purrr_1.0.2       readr_2.1.5       tidyr_1.3.1
## [25] tibble_3.2.1      ggplot2_3.5.1     tidyverse_2.0.0   data.table_1.16.2
## [29] sensobol_1.1.5
##
## loaded via a namespace (and not attached):
##  [1] benchmarkmeData_1.0.4 gtable_0.3.6          xfun_0.49
##  [4] ggrepel_0.9.6         lattice_0.22-6        tzdb_0.4.0
##  [7] vctrs_0.6.5           tools_4.3.3           Rdpack_2.6.2
## [10] generics_0.1.3        fansi_1.0.6           pkgconfig_2.0.3
## [13] Matrix_1.6-5          lifecycle_1.0.4       compiler_4.3.3
## [16] farver_2.1.2          munsell_0.5.1         graphlayouts_1.2.1
## [19] codetools_0.2-20      snakecase_0.11.1      htmltools_0.5.8.1
## [22] yaml_2.3.10           pillar_1.9.0          MASS_7.3-60.0.1
## [25] cachem_1.1.0          tidyselect_1.2.1      digest_0.6.37
## [28] slam_0.1-55           stringi_1.8.4         rprojroot_2.0.4
## [31] polyclip_1.10-7       fastmap_1.2.0         grid_4.3.3
## [34] colorspace_2.1-1      cli_3.6.3             magrittr_2.0.3
## [37] utf8_1.2.4            withr_3.0.2           timechange_0.3.0
```

```
## [40] httr_1.4.7            rmarkdown_2.29        gridExtra_2.3
## [43] hms_1.1.3             memoise_2.0.1         evaluate_1.0.1
## [46] knitr_1.49            rbibutils_2.3         rlang_1.1.4
## [49] Rcpp_1.0.13-1         glue_1.8.0            xml2_1.3.6
## [52] tweenr_2.0.3          rstudioapi_0.17.1     R6_2.5.1
```

```
## 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
```