

# Fifty years of research have deepened uncertainties in global irrigation water use

R code of the multiverse analysis

Arnald Puy

## Contents

<b>1</b>	<b>Preliminary functions</b>	<b>2</b>
<b>2</b>	<b>The Multiverse Analysis</b>	<b>3</b>
2.1	The dataset . . . . .	3
2.2	Graphical representation of the multiverse . . . . .	10
2.3	The garden of forking paths . . . . .	18
2.4	Random forest model . . . . .	23
<b>3</b>	<b>Session information</b>	<b>39</b>

# 1 Preliminary functions

```
# PRELIMINARY FUNCTIONS #####

sensobol::load_packages(c("openxlsx", "data.table", "tidyverse", "cowplot",
                          "benchmarkme", "parallel", "wesanderson", "scales", "ncdf4",
                          "countrycode", "rworldmap", "sp", "doParallel", "here", "lme4",
                          "microbenchmark", "mgcv", "brms", "randomForest", "here",
                          "igraph", "ggraph", "gganimate", "magick",
                          "randomForestExplainer", "ggrepel"))

# 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.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))
}

# Select color palette -----

selected.palette <- "Darjeeling1"

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

# Source all .R files in the "functions" folder -----

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

## 2 The Multiverse Analysis

### 2.1 The dataset

```
# NAOMI DATASET #####

references.projected <- data.table(read.xlsx("./data/references_projection.xlsx")) %>%
  .[, focus:= "projected"]

references.current <- data.table(read.xlsx("./data/references_current.xlsx")) %>%
  .[, focus:= "current"]

references.full.dt <- rbind(references.projected, references.current) %>%
  .[, study:= paste(author, model, climate.scenario, sep = ".")]

# CLEAN THE DATASET #####

colnames_vector <- c("title", "author", "region")

# Remove leading and trailing spaces -----

references.full.dt[, (colnames_vector):= lapply(.SD, trimws), .SDcols = (colnames_vector)]
references.full.dt[, (colnames_vector):= lapply(.SD, str_squish), .SDcols = (colnames_vector)]

# Lowercaps -----

references.full.dt[, (colnames_vector):= lapply(.SD, tolower), .SDcols = (colnames_vector)]

# Remove multiple spaces -----

references.full.dt[, (colnames_vector):= lapply(.SD, function(x)
  gsub("\\s+", " ", x)), .SDcols = (colnames_vector)]

# Correct America -----

references.full.dt[, region:= ifelse(region == "america", "americas", region)]

# Extract the publication year -----

references.full.dt[, publication.date:= str_extract(author, "\\d{4}")] %>%
  .[, publication.date:= as.numeric(publication.date)]

# FEATURES OF THE DATASET #####

# Definition of target years -----
```

```
target_year <- c(2000, 2010, 2050, 2070, 2100)
```

```
# Name of different studies -----
```

```
sort(unique(references.full.dt[variable == "iww" & region == "global", title]))
```

```
## [1] "a global water scarcity assessment under shared socio-economic pathways - part 2: wat
## [2] "a pathway of global food supply adaptation in a world with increasingly constrained g
## [3] "a reservoir operation scheme for global river routing models"
## [4] "agricultural green and blue water consumption and its influence on the global water sy
## [5] "an integrated assessment of global and regional water demands for electricity generat
## [6] "an integrated model for the assessment of global water resources - part 2: application
## [7] "appraisal and assessment of world water resources"
## [8] "aquastat: fao's global information system on water and agriculture"
## [9] "bending the curve: toward global sustainability"
## [10] "cited in world resources 1990-1991, p. 172"
## [11] "climate change impacts on irrigation water requirements: effects of mitigation, 1990-2
## [12] "climate impacts on global irrigation requirements under 19 gcms, simulated with a veg
## [13] "climate mitigation policy implications for global irrigation water demand"
## [14] "climate policy implications for agricultural water demand"
## [15] "future long-term changes in global water resources driven by socio-economic and clima
## [16] "global and regional evaluation of energy for water"
## [17] "global hydrological cycles and world water resources,"
## [18] "global impacts of conversions from natural to agricultural ecosystems on water resourc
## [19] "global irrigation characteristics and effects simulated by fully coupled land surface
## [20] "global irrigation water demand: variability and uncertainties arising from agricultur
## [21] "global modeling of irrigation water requirements"
## [22] "global modeling of withdrawal, allocation and consumptive use of surface water and gro
## [23] "global monthly sectoral water use for 2010-2100 at 0.5° resolution across alternative
## [24] "global water demand and supply projections"
## [25] "globwat - a global water balance model to assess water use in irrigated agriculture"
## [26] "green and blue water accounting in the ganges and nile basins: implications for food a
## [27] "high-resolution modeling of human and climate impacts on global water resources"
## [28] "how can we cope with the water resources situation by the year 2050?"
## [29] "human appropriation of renewable fresh water"
## [30] "impact of climate forcing uncertainty and human water use on global and continental w
## [31] "implementation and evaluation of irrigation techniques in the community land model"
## [32] "incorporating anthropogenic water regulation modules into a land surface model"
## [33] "incorporation of groundwater pumping in a global land surface model with the represen
## [34] "integrated crop water management might sustainably halve the global food gap"
## [35] "isimip database"
## [36] "long-term global water projections using six socioeconomic scenarios in an integrated a
## [37] "lpjml4 - a dynamic global vegetation model with managed land - part 2: model evaluati
## [38] "modelling global water stress of the recent past: on the relative importance of trends
## [39] "multimodel projections and uncertainties of irrigation water demand under climate cha
## [40] "pcr-globwb 2: a 5 arcmin global hydrological and water resources model"
## [41] "physical impacts of climate change on water resources"
```

```
## [42] "present-day irrigation mitigates heat extremes"
## [43] "projecting irrigation water requirements across multiple socio-economic development f
## [44] "projection of future world water resources under sres scenarios: water withdrawal"
## [45] "quantifying global agricultural water appropriation with data derived from earth obser
## [46] "recent global cropland water consumption constrained by observations"
## [47] "reconciling irrigated food production with environmental flows for sustainable develop
## [48] "reconstructing 20th century global hydrography: a contribution to the global terrestri
## [49] "sustainability of global water use: past reconstruction and future projections"
## [50] "the land-water-energy-nexus: biophysical and economic consequences"
## [51] "the state of the world's land and water resources for food and agriculture"
## [52] "the united nations world water development report 2014: water and energy"
## [53] "the world's water, 2000-2001: the biennial report on freshwater resources"
## [54] "united nations world water development report 2020: water and climate change"
## [55] "water 2050. moving toward a sustainable vision for the earth's fresh water"
## [56] "water and sustainability. global pattern and long-range problems"
## [57] "water savings potentials of irrigation systems: global simulation of processes and li
## [58] "water scarcity in the twenty-first century"
## [59] "water sector assumptions for the shared socioeconomic pathways in an integrated model
## [60] "world agriculture towards 2030/2050: the 2012 revision"
## [61] "world agriculture towards 2030/2055"
## [62] "world resources 1992-93. a guide to the global environment"
## [63] "world water demand and supply, 1990 to 2025: scenarios and issues"
## [64] "world water in 2025 - global modeling and scenario analysis for the world commission o
## [65] "world water resources and their future"
```

```
# Number of data points -----
```

```
nrow(references.full.dt[variable == "iww" & region == "global"])
```

```
## [1] 1624
```

```
# Number of different studies per variable -----
```

```
references.full.dt[region == "global", unique(title), variable] %>%
  .[, .N, variable]
```

```
##   variable  N
## 1:      iww 65
## 2:      tww 20
## 3:      iwc 20
## 4:      twc  4
## 5:      iwr  2
```

```
# Number of data points for each target year -----
```

```
references.full.dt[variable == "iww" & region == "global" &
  estimation.year %in% target_year, .N, estimation.year]
```

```
##   estimation.year  N
## 1:           2000 65
```

```
## 2:          2070 148
## 3:          2100 121
## 4:          2010  97
## 5:          2050 152
```

```
# Number of unique studies estimating for each target year -----
```

```
references.full.dt[variable == "iww" & region == "global" &
                    estimation.year %in% target_year, unique(title), estimation.year] %>%
  .[, .N, estimation.year]
```

```
##      estimation.year  N
## 1:          2000  24
## 2:          2070   5
## 3:          2100   5
## 4:          2010  11
## 5:          2050  16
```

```
# Number of data points for every targeted year -----
```

```
references.full.dt[variable == "iww" & region == "global", .N, estimation.year] %>%
  .[order(estimation.year)]
```

```
##      estimation.year  N
## 1:          1900   3
## 2:          1910   2
## 3:          1920   2
## 4:          1930   2
## 5:          1940   4
## 6:          1950   4
## 7:          1960   7
## 8:          1970   5
## 9:          1975  22
##10:          1980  29
##11:          1983   1
##12:          1985  33
##13:          1986   1
##14:          1988   1
##15:          1990  29
##16:          1993   2
##17:          1994   3
##18:          1995  40
##19:          1996   2
##20:          2000  65
##21:          2002   1
##22:          2003   1
##23:          2004   1
##24:          2005  34
##25:          2006   2
```

```
## 26:          2007    1
## 27:          2008    1
## 28:          2010   97
## 29:          2015    9
## 30:          2020  121
## 31:          2021    1
## 32:          2025   16
## 33:          2030  112
## 34:          2035    7
## 35:          2040  123
## 36:          2050  152
## 37:          2055    6
## 38:          2060  112
## 39:          2065    7
## 40:          2070  148
## 41:          2075    6
## 42:          2080  127
## 43:          2090  109
## 44:          2095   14
## 45:          2099   38
## 46:          2100  121
##      estimation.year    N
```

```
# Number of data points for year 2000 or later years -----
```

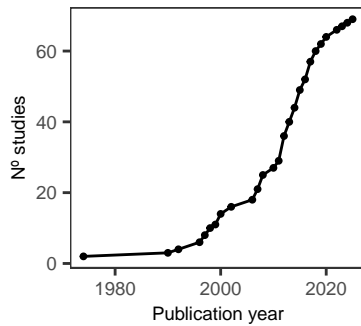
```
references.full.dt[variable == "iww" & region == "global", .N, estimation.year] %>%
  .[estimation.year >= 2000] %>%
  .[, N] %>%
  sum(.)
```

```
## [1] 1432
```

```
# Cumulative sum of published studies -----
```

```
cumulative.iww <- references.full.dt[, .(title, publication.date, variable)] %>%
  .[variable == "iww"] %>%
  .[!duplicated()] %>%
  setorder(., publication.date) %>%
  .[, .N, publication.date] %>%
  .[, cumulative_sum := cumsum(N)] %>%
  ggplot(., aes(publication.date, cumulative_sum)) +
  geom_line() +
  scale_x_continuous(breaks = breaks_pretty(n = 3)) +
  geom_point(size = 0.7) +
  theme_AP() +
  labs(x = "Publication year", y = "N° studies")
```

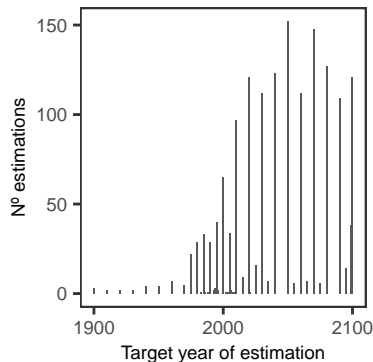
```
cumulative.iww
```



*# DISTRIBUTION OF DATA POINTS THROUGH YEARS @#####*

```
plot.bar <- references.full.dt[variable == "iww" & region == "global", .N, estimation.year] %>%
  ggplot(., aes(estimation.year, N)) +
  geom_bar(stat = "identity") +
  scale_x_continuous(breaks = breaks_pretty(n = 3)) +
  labs(x = "Target year of estimation", y = "N° estimations") +
  theme_AP()
```

plot.bar



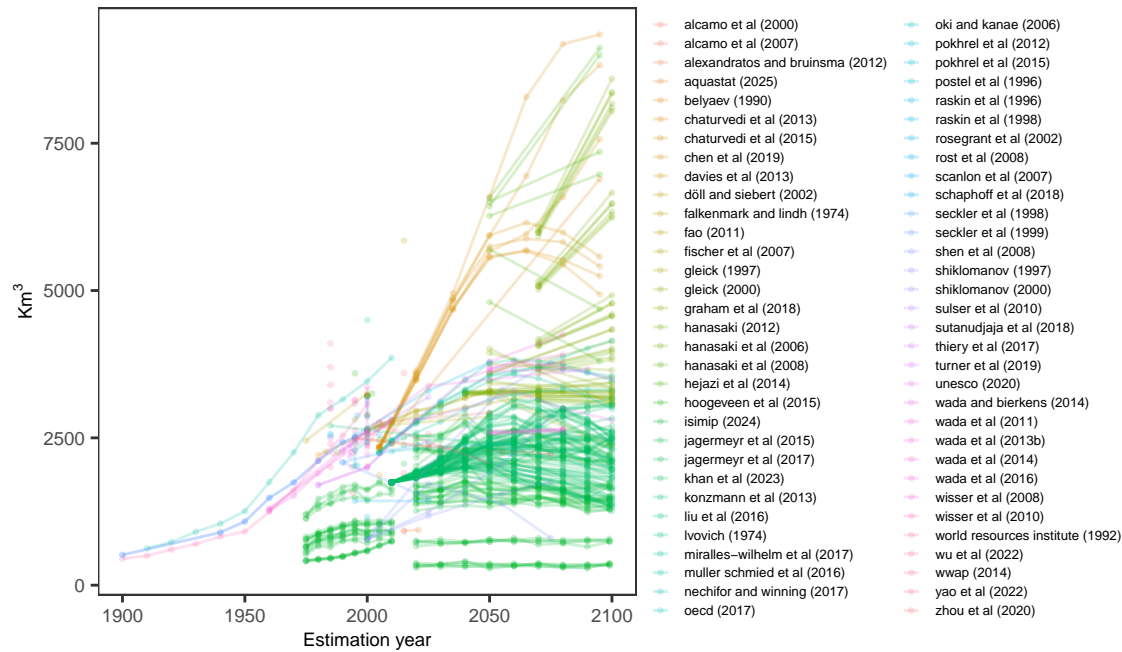
*# PLOT ALL ESTIMATIONS #####*

```
def.alpha <- 0.2

plot.iww <- references.full.dt[variable == "iww" & region == "global"] %>%
  .[, .(author, study, estimation.year, value)] %>%
  na.omit() %>%
  ggplot(., aes(estimation.year, value, color = author, group = study)) +
  geom_point(alpha = def.alpha, size = 0.5) +
  labs(x = "Estimation year", y = bquote("Km"^3)) +
  scale_color_discrete(name = "") +
  geom_line(alpha = def.alpha) +
  theme_AP() +
  guides(color = guide_legend(ncol = 2)) +
  theme(legend.text = element_text(size = 5.2),
        legend.key.width = unit(0.25, "cm"),
        legend.key.height = unit(0.25, "cm"))
```



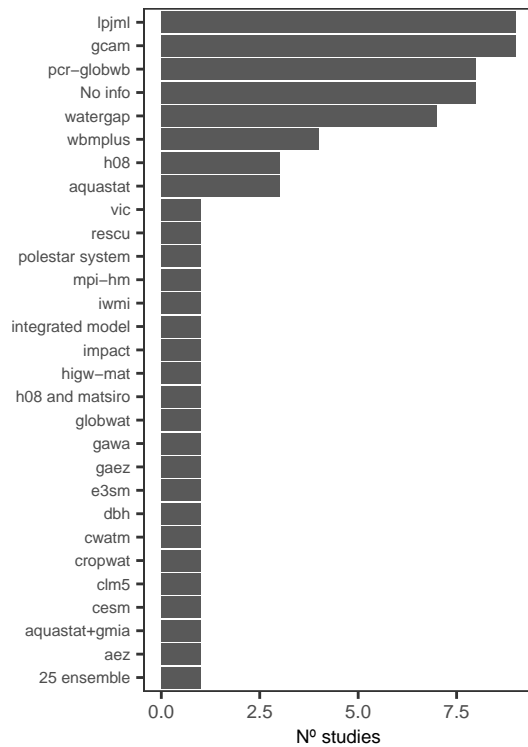
```
plot.iww
```



```
# PLOT NUMBER OF UNIQUE STUDIES PER MODEL #####
```

```
plot.models <- references.full.dt[variable == "iww" & region == "global"] %>%
  .[, .(title, doi, model)] %>%
  .[, model := tolower(model)] %>%
  .[, unique(doi), model] %>%
  .[, model := gsub("(?i)watergap\\s*\\d*\\.?.\\d*", "watergap", model, perl = TRUE)] %>%
  .[, .N, model] %>%
  .[, model := ifelse(is.na(model), "No info", model)] %>%
  ggplot(. , aes(reorder(model, N), N)) +
  geom_bar(stat = "identity") +
  labs(x = "", y = "N° studies") +
  coord_flip() +
  theme_AP() +
  theme(axis.text.y = element_text(size = 5.5))
```

```
plot.models
```



## 2.2 Graphical representation of the multiverse

```
# PLOT EXAMPLES TO ILLUSTRATE APPROACH #####

# Set seed for reproducibility -----

set.seed(123)

# Create datasets for different SD trends -----

data_increasing <- data.frame(
  period = rep(c("1990-2000", "2000-2010", "2010-2020"), times = c(5, 7, 4)),
  value = c(rnorm(5, mean = 5, sd = 0.3), # Low SD
            rnorm(7, mean = 7, sd = 0.8), # Medium SD
            rnorm(4, mean = 6, sd = 1.5)), # High SD
  target_year = 2000
)

data_decreasing <- data.frame(
  period = rep(c("1980-2000", "2000-2020"), times = c(5, 7)),
  value = c(rnorm(5, mean = 5, sd = 1.5), # High SD
            rnorm(7, mean = 7, sd = 0.8)), # Medium
  target_year = 2010
)

data_invertedV <- data.frame(
```

```

period = rep(c("1990-2000", "2000-2010", "2010-2020"), times = c(5, 7, 4)),
value = c(rnorm(5, mean = 5, sd = 0.4), # Low SD
          rnorm(7, mean = 7, sd = 1.4), # High SD (peak in the middle)
          rnorm(4, mean = 5, sd = 0.4)), # Low SD again
target_year = 2070
)

# Function to compute SD and create a ggplot -----

create_plot <- function(data, title) {
  sd_values <- data %>%
    group_by(period) %>%
    summarize(sd_value = sd(value) + 3)

  ggplot(data, aes(x = period, y = value)) +
    geom_point(size = 1) +
    geom_point(data = sd_values, aes(x = period, y = sd_value), color = "red", size = 1.5) +
    geom_line(data = sd_values, aes(x = period, y = sd_value, group = 1), color = "red", linewidth = 1) +
    theme_AP() +
    theme(axis.text.x = element_text(size = 5.35),
          plot.margin = unit(c(0.1, 0.1, 0, 0.1), "cm")) +
    scale_y_continuous(breaks = breaks_pretty(n = 3)) +
    scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
    labs(x = "", y = "Value") +
    annotate("text", x = 0.1 + 0.5, y = max(data$value),
             label = unique(data$target_year), hjust = 0, vjust = 1,
             size = 2)
}

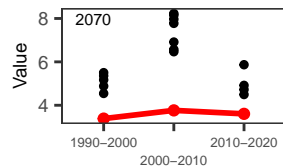
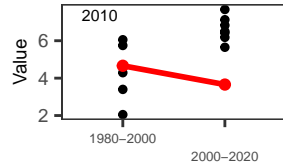
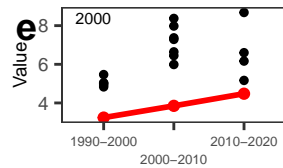
# Generate the three plots -----

p1 <- create_plot(data_increasing)
p2 <- create_plot(data_decreasing)
p3 <- create_plot(data_invertedV)

# Merge using plot_grid -----

plot.examples.trends.data <- plot_grid(p1, p2, p3, ncol = 1, labels = c("e", "", ""))
plot.examples.trends.data

```



```
# GRAPHICAL REPRESENTATION OF THE GARDEN OF FORKING PATHS #####

# Define size of nodes -----

size.nodes <- 1.5

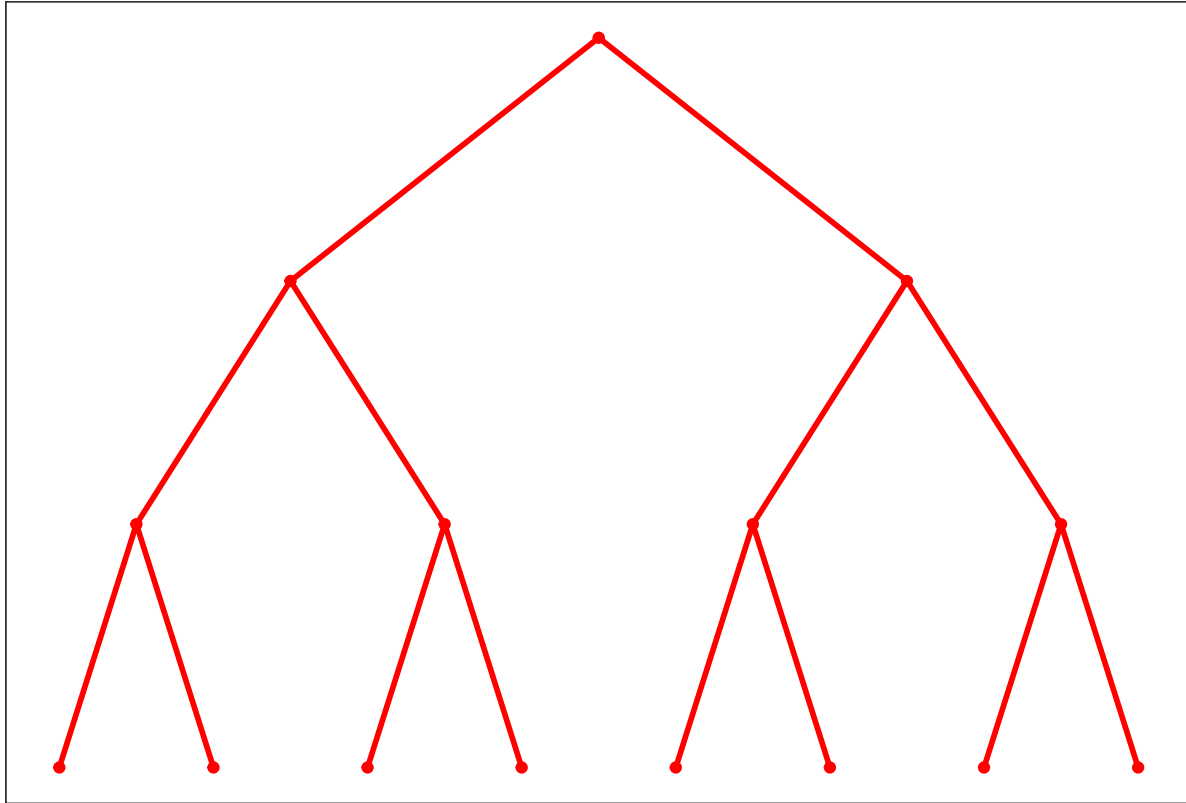
# Create a balanced binary tree with height 3 -----

tree <- make_tree(15, children = 2, mode = "out")

# Create a tree plot with all edges highlighted in red -----

all.paths <- ggraph(tree, layout = "dendrogram") +
  geom_edge_link(color = "red", width = 1) +
  geom_node_point(size = size.nodes, color = "red") +
  theme_AP() +
  labs(x = "", y = "") +
  theme(legend.position = "none",
        axis.ticks = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank())

all.paths
```



```

# Create a tree plot with only one analytical path highlighted -----
# Define the path to highlight (from root to a specific node) -----
highlight_nodes <- c(1, 2, 5, 11) # Path: 1 → 2 → 5 → 11

highlight_edges <- apply(cbind(head(highlight_nodes, -1),
                                tail(highlight_nodes, -1)), 1, function(x)
                          paste(x, collapse = "-"))

# Assign default colors (black) to all edges and nodes -----

E(tree)$edge_color <- "black"
V(tree)$node_color <- "black"

# Extract edges from the tree and match with highlight_edges -----

edge_list <- apply(get.edgelist(tree), 1, function(x) paste(x, collapse = "-"))

## Warning: `get.edgelist()` was deprecated in igraph 2.0.0.
## i Please use `as_edgelist()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

```

```

E(tree)$edge_color[edge_list %in% highlight_edges] <- "red"

# Highlight the selected nodes in red -----

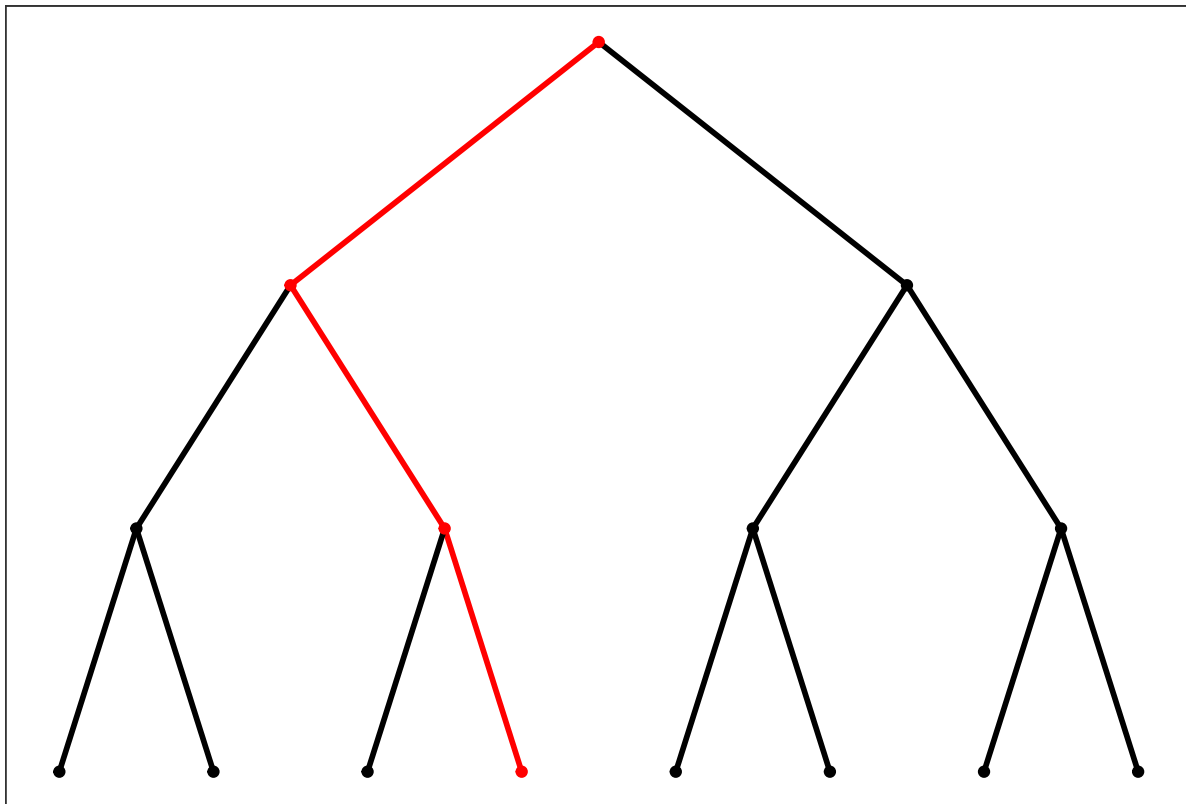
V(tree)$node_color[highlight_nodes] <- "red"

# Plot the tree with explicitly defined colors for both edges and nodes -----

one.path <- ggraph(tree, layout = "dendrogram") +
  geom_edge_link(aes(edge_color = edge_color), width = 1) + # Correct edge colors
  geom_node_point(aes(color = node_color), size = size.nodes) + # Correct node colors
  scale_edge_color_manual(values = c("black" = "black", "red" = "red")) + # Fix for edges
  scale_color_manual(values = c("black" = "black", "red" = "red")) + # Fix for nodes
  theme_AP() +
  labs(x = "", y = "") +
  theme(legend.position = "none",
        axis.ticks = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank())

one.path

```



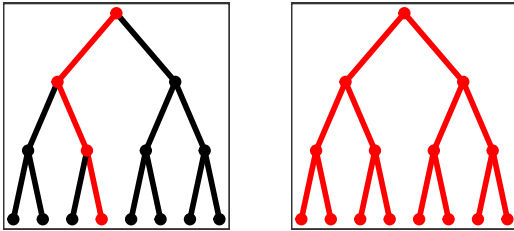
```

# MERGE FORKING PATHS #####

```

```
plot_grid(one.path, all.paths, ncol = 2, labels = c("a", ""))
```

**a**



```
# GRAPHICAL REPRESENTATION OF THE GARDEN OF FORKING PATHS #####
```

```
# Define size of nodes -----
```

```
size.nodes <- 1.5
```

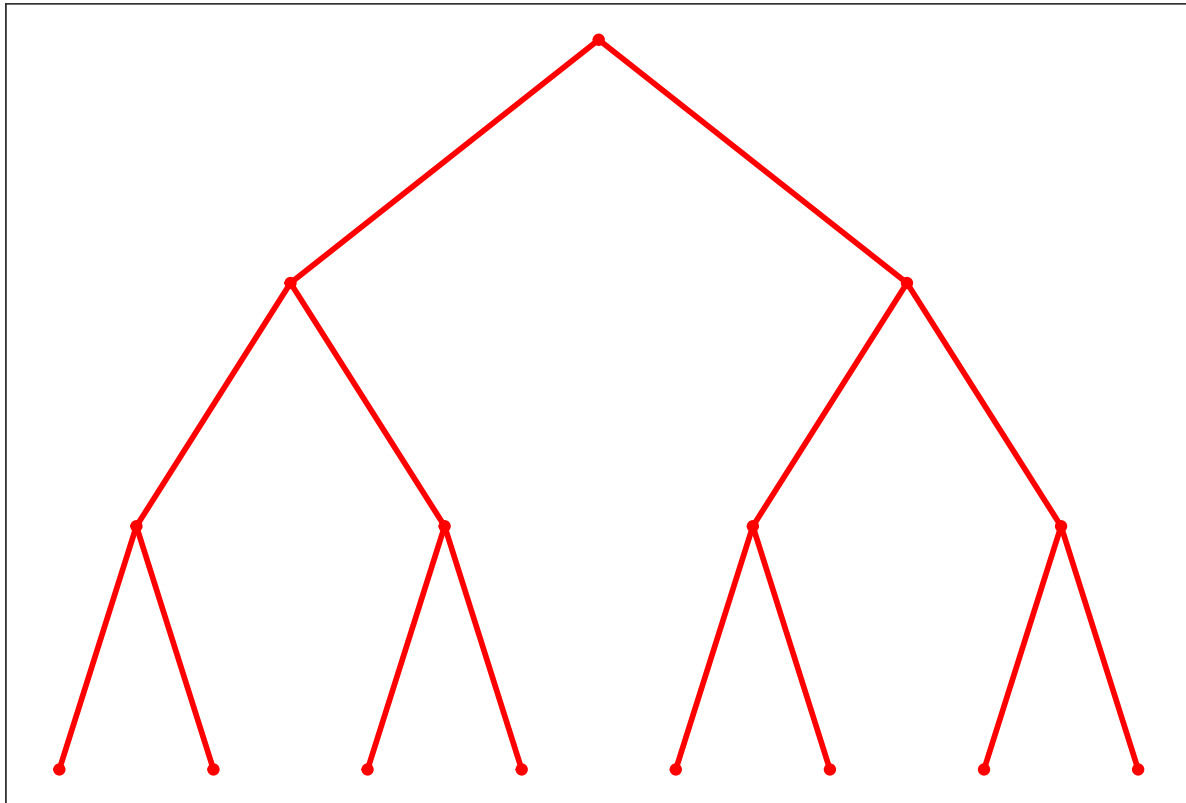
```
# Create a balanced binary tree with height 3 -----
```

```
tree <- make_tree(15, children = 2, mode = "out")
```

```
# Create a tree plot with all edges highlighted in red -----
```

```
all.paths <- ggraph(tree, layout = "dendrogram") +  
  geom_edge_link(color = "red", width = 1) +  
  geom_node_point(size = size.nodes, color = "red") +  
  theme_AP() +  
  labs(x = "", y = "") +  
  theme(legend.position = "none",  
        axis.ticks = element_blank(),  
        axis.text.x = element_blank(),  
        axis.text.y = element_blank())
```

```
all.paths
```



```

# Create a tree plot with only one analytical path highlighted -----
# Define the path to highlight (from root to a specific node) -----
highlight_nodes <- c(1, 2, 4, 8) # Path: 1 → 2 → 5 → 11

highlight_edges <- apply(cbind(head(highlight_nodes, -1),
                                tail(highlight_nodes, -1)), 1, function(x)
                          paste(x, collapse = "-"))

# Assign default colors (black) to all edges and nodes -----

E(tree)$edge_color <- "black"
V(tree)$node_color <- "black"

# Extract edges from the tree and match with highlight_edges -----

edge_list <- apply(get.edgelist(tree), 1, function(x) paste(x, collapse = "-"))
E(tree)$edge_color[edge_list %in% highlight_edges] <- "red"

# Highlight the selected nodes in red -_------

V(tree)$node_color[highlight_nodes] <- "red"

# Plot the tree with explicitly defined colors for both edges and nodes -----

```

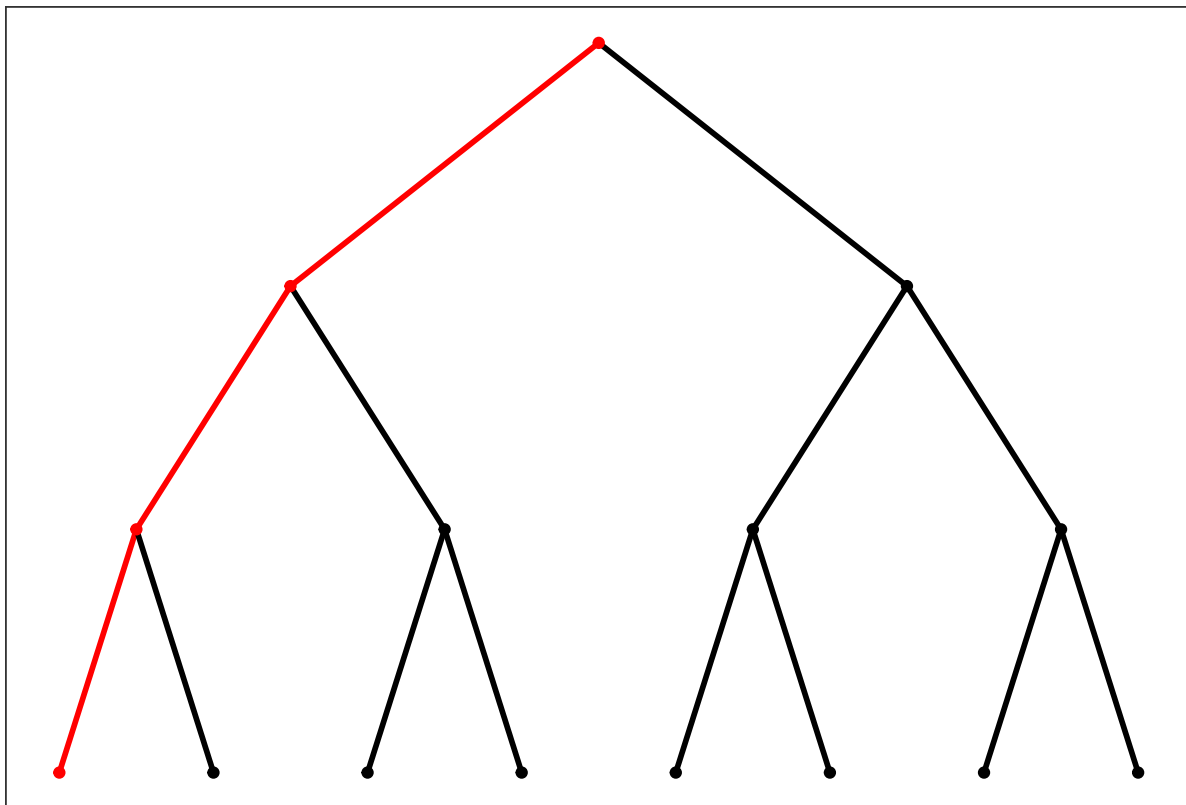


```

one.path2 <- ggraph(tree, layout = "dendrogram") +
  geom_edge_link(aes(edge_color = edge_color), width = 1) + # Correct edge colors
  geom_node_point(aes(color = node_color), size = size.nodes) + # Correct node colors
  scale_edge_color_manual(values = c("black" = "black", "red" = "red")) + # Fix for edges
  scale_color_manual(values = c("black" = "black", "red" = "red")) + # Fix for nodes
  theme_AP() +
  labs(x = "", y = "") +
  theme(legend.position = "none",
        axis.ticks = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank())

```

one.path2

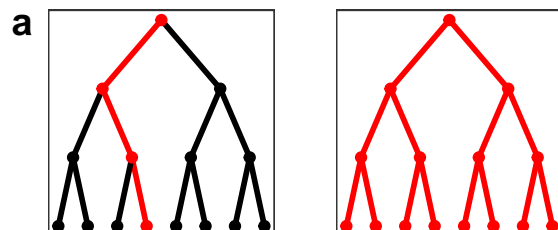


*# MERGE FORKING PATHS #####*

```

plot_grid(one.path, all.paths, ncol = 2, labels = c("a", ""))

```



## 2.3 The garden of forking paths

```
# DEFINE THE UNCERTAINTY SPACE #####

# Target year -----

## Defined above

# Target year interval -----

target_year_interval <- c("yes", "no")

# Interval publication -----

interval <- c(10, 15, 20)

# Metrics of study -----

metrics <- c("cv", "range", "sd", "var", "entropy", "iqr")

# Rolling windows -----

rolling_window_factor <- c(1, 0.5)

# Define the forking paths -----

forking_paths <- expand.grid(target_year = target_year,
                             target_year_interval = target_year_interval,
                             interval = interval,
                             rolling_window_factor = rolling_window_factor,
                             metric = metrics) %>%

  data.table()

# Number of simulations -----

nrow(forking_paths)

## [1] 360

# RUN MODEL #####

# Select only simulations at the global level of iww -----

dt <- references.full.dt[variable == "iww" & region == "global"]

# Run simulations -----

trend <- list()
```

```

for (i in 1:nrow(forking_paths)) {

  trend[[i]] <- forking_paths_fun(dt = dt,
                                target_year = forking_paths[[i, "target_year"]],
                                target_year_interval = forking_paths[[i, "target_year_interval"]],
                                interval = forking_paths[[i, "interval"]],
                                rolling_window_factor = forking_paths[[i, "rolling_window_factor"]],
                                metric = forking_paths[[i, "metric"]])
}

# ARRANGE DATA #####

output.dt <- lapply(trend, function(x) x[["results"]]) %>%
  do.call(rbind, .) %>%
  data.table() %>%
  setnames(., "V1", "trend") %>%
  .[, row:= .I]

final.dt <- cbind(forking_paths, output.dt)

# ARRANGE DATA PRIMARY #####

data.dt <- lapply(trend, function(x) x[["data"]]) %>%
  rbindlist(., idcol = "row") %>%
  merge(., final.dt, by = "row")

# Export simulations -----

fwrite(final.dt, "forking.paths.dataset.csv")
write.xlsx(final.dt, "forking.paths.dataset.xlsx")
fwrite(data.dt, "data.dt.csv")

# Print the fraction of simulations in each classification -----

final.dt %>%
  .[, .(total = .N), trend] %>%
  .[, fraction:= total / nrow(output.dt)] %>%
  print()

##          trend total  fraction
## 1:    Unstable   121 0.3361111
## 2:    Decrease    62 0.1722222
## 3:     Increase   141 0.3916667
## 4: single point    36 0.1000000

# Now remove all simulations that produced just one single point -----

final.dt <- final.dt[!trend == "single point"]

```

```

# Calculate how many forking paths lead to different results just
# by changing the metric (all the rest fixed) -----

final.dt %>%
  .[, .(target_year, target_year_interval, interval,
        rolling_window_factor, metric, trend)] %>%
  .[order(target_year, target_year_interval, interval, rolling_window_factor)] %>%
  split(., ceiling(seq_len(nrow(.)) / length(metrics))) %>%
  rbindlist(., idcol = "group") %>%
  .[, .(unique_trend_count = uniqueN(trend)), group] %>%
  .[, .N, unique_trend_count] %>%
  .[order(unique_trend_count)]

##      unique_trend_count  N
## 1:                      1 15
## 2:                      2 37
## 3:                      3  2

# Simulations that did not lead to a reduction in uncertainty -----

final.dt %>%
  .[, .(total = .N), trend] %>%
  .[, fraction:= total / nrow(output.dt)] %>%
  .[!trend == "Decrease"] %>%
  .[, sum(fraction)]

## [1] 0.7277778

# PLOTS FORKING PATHS EXAMPLES #####

plots.dt <- lapply(trend, function(x) x[["plot"]])

# Increasing trends -----

plots.increasing <- plot_grid(plots.dt[[7]] +
  geom_line(color = "red", group = 1),
  plots.dt[[11]] +
  geom_line(color = "red", group = 1),
  plots.dt[[278]] +
  geom_line(color = "red", group = 1),
  plots.dt[[226]] +
  geom_line(color = "red", group = 1), ncol = 1)

# Decreasing trend -----

plots.decreasing <- plot_grid(plots.dt[[4]] +
  geom_line(color = "darkgreen", group = 1) +
  geom_point(color = "darkgreen"),

```

```

plots.dt[[69]] +
  geom_line(color = "darkgreen", group = 1) +
  geom_point(color = "darkgreen"),
plots.dt[[100]] +
  geom_line(color = "darkgreen", group = 1) +
  geom_point(color = "darkgreen"),
plots.dt[[142]] +
  geom_line(color = "darkgreen", group = 1) +
  geom_point(color = "darkgreen"), ncol = 1)

# Random trend -----

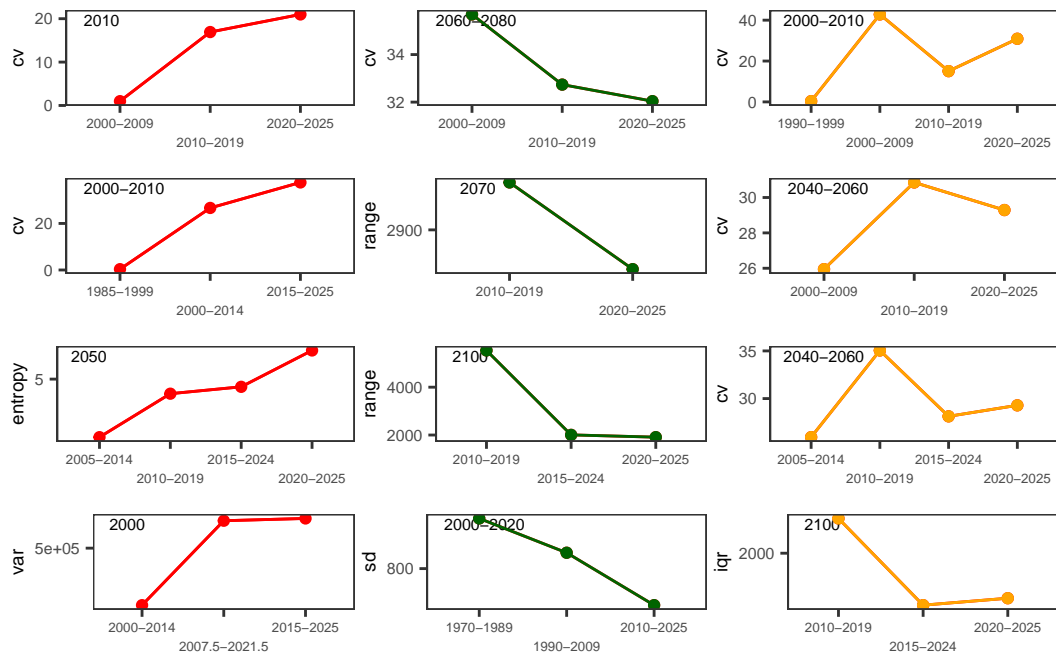
plots.random <- plot_grid(plots.dt[[1]] +
  geom_line(color = "orange", group = 1) +
  geom_point(color = "orange"),
plots.dt[[3]] +
  geom_line(color = "orange", group = 1) +
  geom_point(color = "orange"),
plots.dt[[33]] +
  geom_line(color = "orange", group = 1) +
  geom_point(color = "orange"),
plots.dt[[340]] +
  geom_line(color = "orange", group = 1) +
  geom_point(color = "orange"), ncol = 1)

# Merge -----

plots.examples.trends <- plot_grid(plots.increasing, plots.decreasing,
  plots.random, ncol = 3)

plots.examples.trends

```

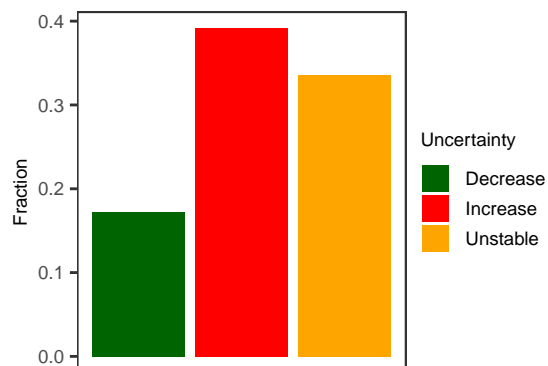


# PLOT RESULTS #####

```
selected_colors <- c("Increase" = "red", "Decrease" = "darkgreen", "Unstable" = "orange")
```

```
plot.fraction <- final.dt[, .(total = .N), trend] %>%
  .[, fraction:= total / nrow(output.dt)] %>%
  ggplot(. , aes(trend, fraction, fill = trend)) +
  geom_bar(stat = "identity") +
  labs(x = "", y = "Fraction") +
  scale_fill_manual(values = selected_colors, name = "Uncertainty") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_AP() +
  theme(axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        legend.position = "right")
```

plot.fraction



## 2.4 Random forest model

```
# RANDOM FOREST #####

# Convert categorical variables to factors -----

df <- data.frame(final.dt[trend != "single point"]) # Remove 5% observations
df$metric <- as.factor(df$metric)
df$trend <- as.factor(df$trend)
df$target_year_interval <- as.factor(df$target_year_interval)

# Train the model with weights on 2,000 random trees -----

rf_model <- randomForest(trend ~ target_year + target_year_interval + interval +
                        rolling_window_factor + metric,
                        data = df, importance = TRUE, ntree = 5000,
                        classwt = c(1.5, 1.5, 1), mtry = 3)

# Check model summary -----

print(rf_model)
```

```
##
## Call:
## randomForest(formula = trend ~ target_year + target_year_interval + interval + rolling_window_factor + metric,
##              data = df, importance = TRUE, ntree = 5000, classwt = c(1.5, 1.5, 1), mtry = 3)
##              Type of random forest: classification
##              Number of trees: 5000
## No. of variables tried at each split: 3
##
## OOB estimate of error rate: 15.12%
## Confusion matrix:
##      Decrease Increase Unstable class.error
## Decrease      51       8       3  0.1774194
## Increase       8      121      12  0.1418440
## Unstable       5       13     103  0.1487603
```

```
# View variable importance -----

dt_rf_model <- data.frame(importance(rf_model))
dt_rf_model
```

```
##              Decrease Increase Unstable MeanDecreaseAccuracy
## target_year      192.10345 101.61249 121.81205          212.985909
## target_year_interval 39.56988  72.08804  50.51787           90.089826
## interval          62.00356 178.18602 212.71774          249.543296
## rolling_window_factor 22.98758 -54.57240  46.36293           6.993587
## metric            86.21448 110.97814  13.99047          112.530444
##
##              MeanDecreaseGini
## target_year          68.79696
```

```

## target_year_interval      19.01072
## interval                  42.85123
## rolling_window_factor    15.11698
## metric                    52.82946

# Compute importance -----

importance_frame <- measure_importance(rf_model)
data <- importance_frame[importance_frame$no_of_trees > 0, ]

# Retrieve data -----

data_for_labels <- importance_frame[importance_frame$variable %in%
                                     important_variables(importance_frame, k = 10,
                                                         measures = c("mean_min_depth",
                                                                    "times_a_root",
                                                                    "no_of_nodes")),]

data_for_labels

##           variable mean_min_depth no_of_nodes accuracy_decrease
## 1           interval      1.0832      51900      0.170662146
## 2             metric      0.8888      82537      0.077324121
## 3 rolling_window_factor  2.3554      42483      0.003025196
## 4         target_year      0.6912     101960      0.150471000
## 5 target_year_interval  2.5932      50966      0.043589115
##  gini_decrease no_of_trees times_a_root p_value
## 1      42.85123      5000      1082      1
## 2      52.82946      5000      1567      0
## 3      15.11698      5000         3      1
## 4      68.79696      5000      2348      0
## 5      19.01072      5000         0      1

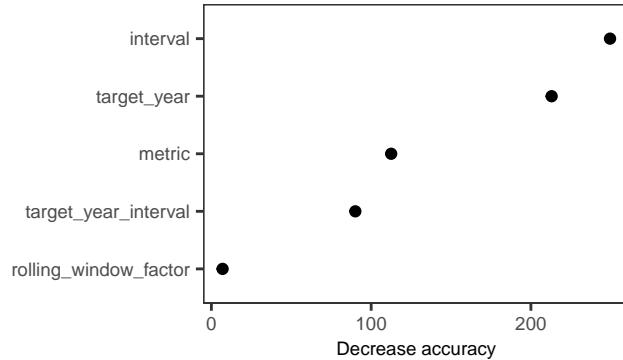
# Plot -----

plot.rf <- data.frame(importance(rf_model)) %>%
  rownames_to_column(., var = "factors") %>%
  data.table() %>%
  setnames(., c("MeanDecreaseAccuracy", "MeanDecreaseGini"),
           c("Accuracy", "Gini")) %>%
  melt(., measure.vars = c("Accuracy", "Gini")) %>%
  .[variable == "Accuracy"] %>%
  ggplot(., aes(reorder(factors, value), value)) +
  geom_point() +
  coord_flip() +
  scale_y_continuous(breaks = breaks_pretty(n = 3)) +
  labs(x = "", y = "Decrease accuracy") +
  theme_AP()

```



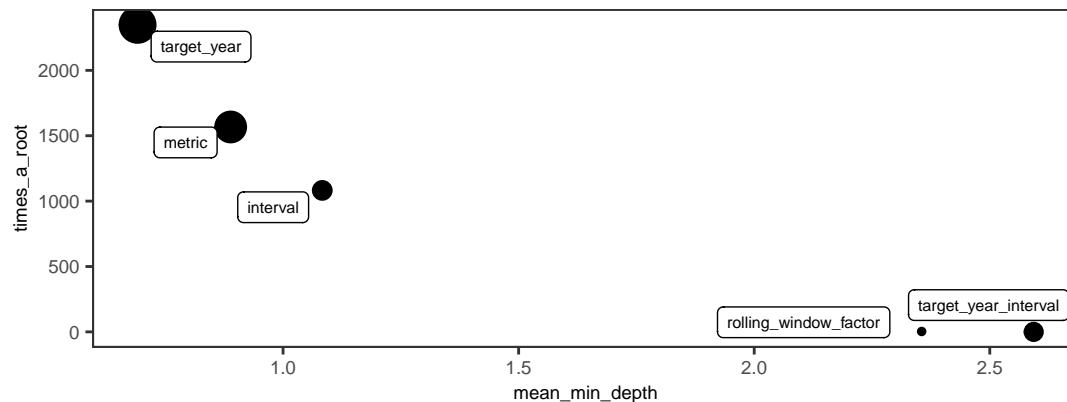
```
plot.rf
```



```
# MULTIWAY IMPORTANCE PLOT #####
```

```
plot.multiway <- ggplot(data, aes(x = mean_min_depth, y = times_a_root)) +
  geom_point(data = data_for_labels, aes(size = no_of_nodes)) +
  geom_label_repel(data = data_for_labels, aes(label = variable),
    show.legend = FALSE, size = 2) +
  scale_size_continuous(breaks = c(min(data$no_of_nodes),
    median(data$no_of_nodes),
    max(data$no_of_nodes)),
    label = label_number(accuracy = 10000)) +
  theme_AP() +
  theme(legend.position = "none")
```

```
plot.multiway
```



```
# # INTERACTIONS PLOT #####
```

```
# Redefine facet labels -----
```

```
supp.labs <- c("Decrease", "Increase", "Unstable")
names(supp.labs) <- paste("probability_", supp.labs, sep = "")
```

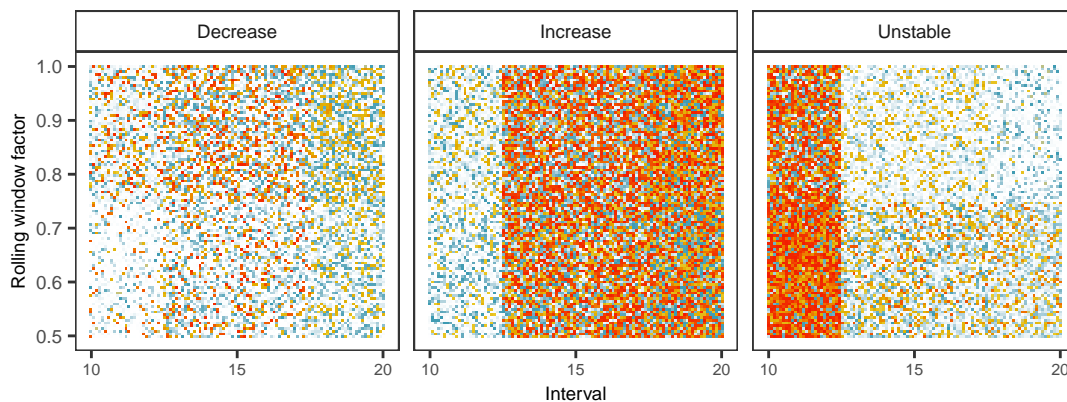
```
# Plot -----
```

```
plot.predict1 <- plot_predict_interaction(rf_model, df, "interval", "rolling_window_factor") +
  theme_AP() +
  scale_fill_gradientn(colours = c("white", wes_palette("Zissou1")),
    name = "probability",
    breaks = c(0, 0.5, 1)) +
  theme(plot.title = element_blank()) +
  labs(x = "Interval", y = "Rolling window factor") +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  facet_grid(~variable, labeller = labeller(variable = supp.labs)) +
  theme(legend.position = "none",
    axis.text.x = element_text(size = 6.1))
```

## Scale for fill is already present.

## Adding another scale for fill, which will replace the existing scale.

```
plot.predict1
```



*# Now on different combinations*

```
plot.predict2 <- plot_predict_interaction(rf_model, df, "target_year", "interval") +
  theme_AP() +
  scale_fill_gradientn(colours = c("white", wes_palette("Zissou1")),
    name = "probability",
    breaks = c(0, 0.5, 1)) +
  theme(plot.title = element_blank()) +
  labs(x = "Target year", y = "Interval") +
  scale_x_continuous(breaks = pretty_breaks(n = 3)) +
  facet_grid(~variable, labeller = labeller(variable = supp.labs)) +
  theme(legend.position = "none",
    axis.text.x = element_text(size = 6.1))
```

## Scale for fill is already present.

## Adding another scale for fill, which will replace the existing scale.

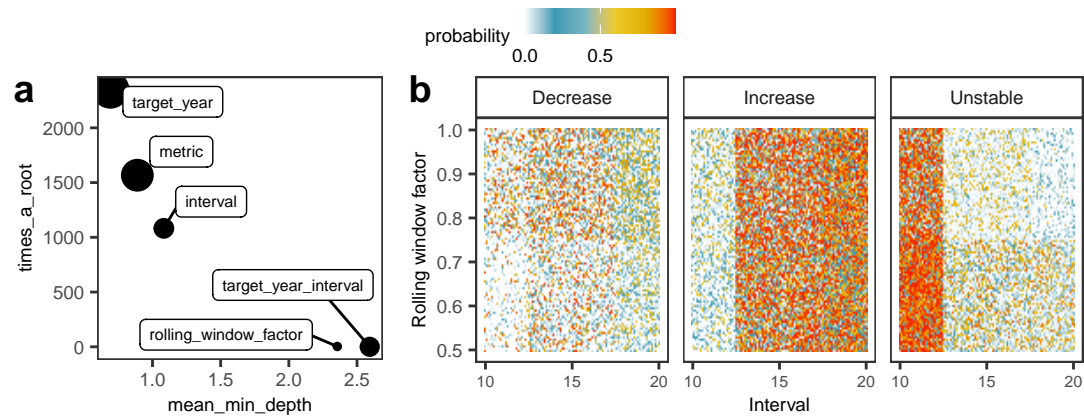
*# MERGE RANDOM FOREST PLOTS #####*

```
legend <- get_legend_fun(plot.predict1 + theme(legend.position = "top"))
bottom <- plot_grid(plot.multiway, plot.predict1, ncol = 2, rel_widths = c(0.36, 0.64),
```

```

labels = "auto")
plot_grid(legend, bottom, rel_heights = c(0.13, 0.87), ncol = 1)

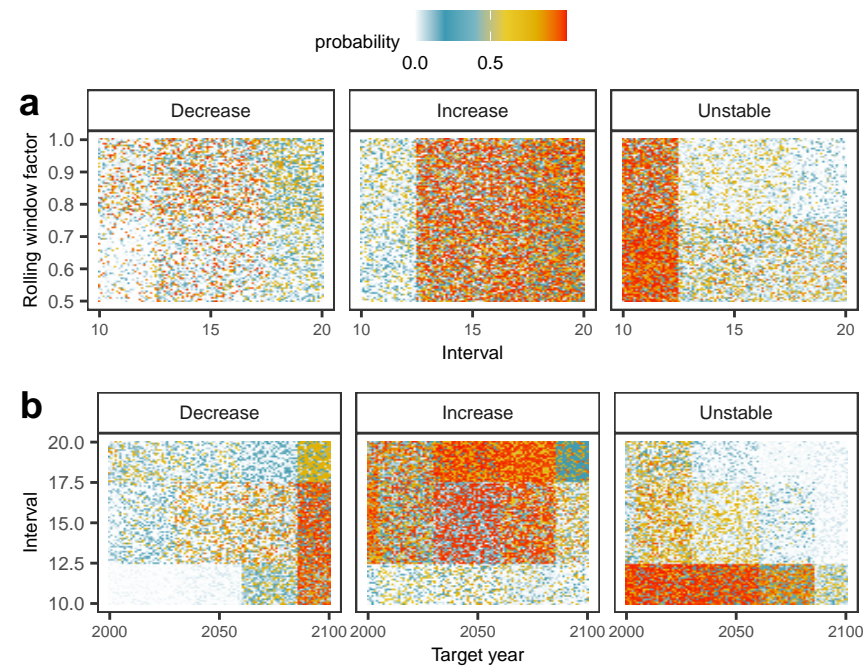
```



```

bottom <- plot_grid(plot.predict1, plot.predict2, ncol = 1, labels = "auto")
plot_grid(legend, bottom, rel_heights = c(0.1, 0.9), ncol = 1)

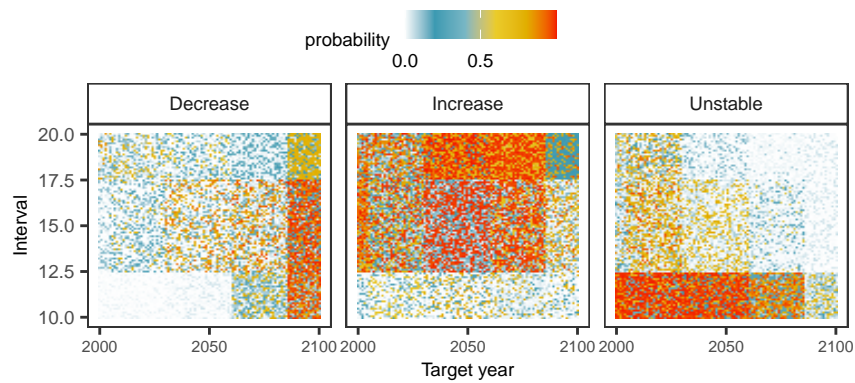
```



```

##
plotp1 <- plot_grid(legend, plot.predict2, ncol = 1, rel_heights = c(0.15, 0.85))
plotp1

```



```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 4.3.3
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.3.3
```

```
# Fit a decision tree model-
```

```
tree_model <- rpart(trend ~ target_year + target_year_interval + interval
  + rolling_window_factor + metric,
  data = final.dt, method = "class")
```

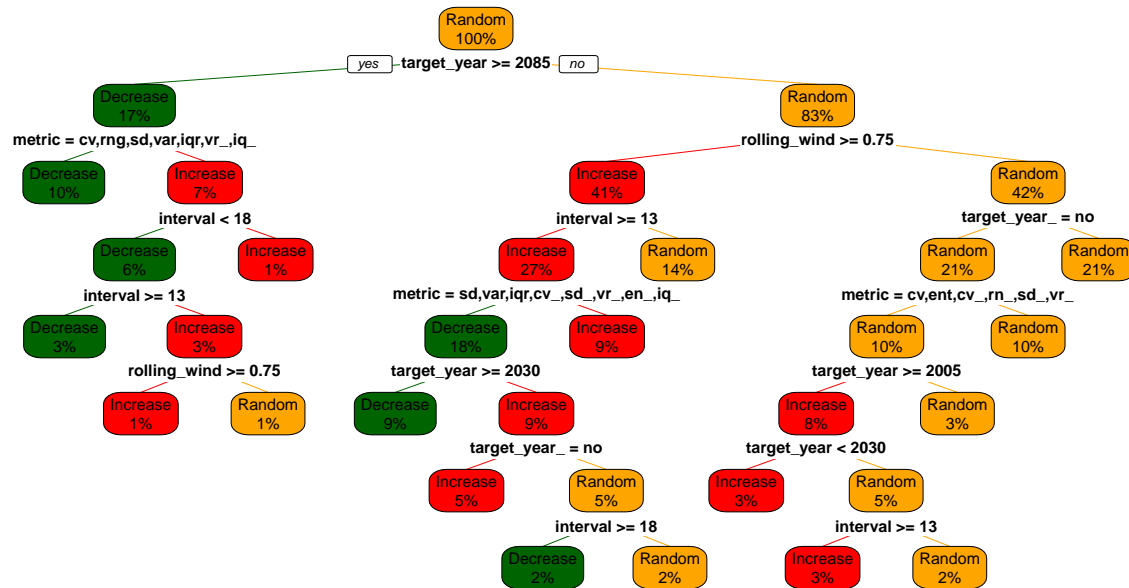
```
# Plot the tree
```

```
# Fit a decision tree model
```

```
tree_model <- rpart(trend ~ target_year + target_year_interval + interval +
  rolling_window_factor + metric,
  data = final.dt, method = "class")
```

```
heat.tree <- function(tree, low.is.green = FALSE, ...) { # dots args passed to prp
  y <- tree$frame$yval
  cols <- ifelse(y == 1, "darkgreen", ifelse(y == 2, "red", "orange"))
  prp(tree, branch.col = cols, box.col = cols, ...)
}
```

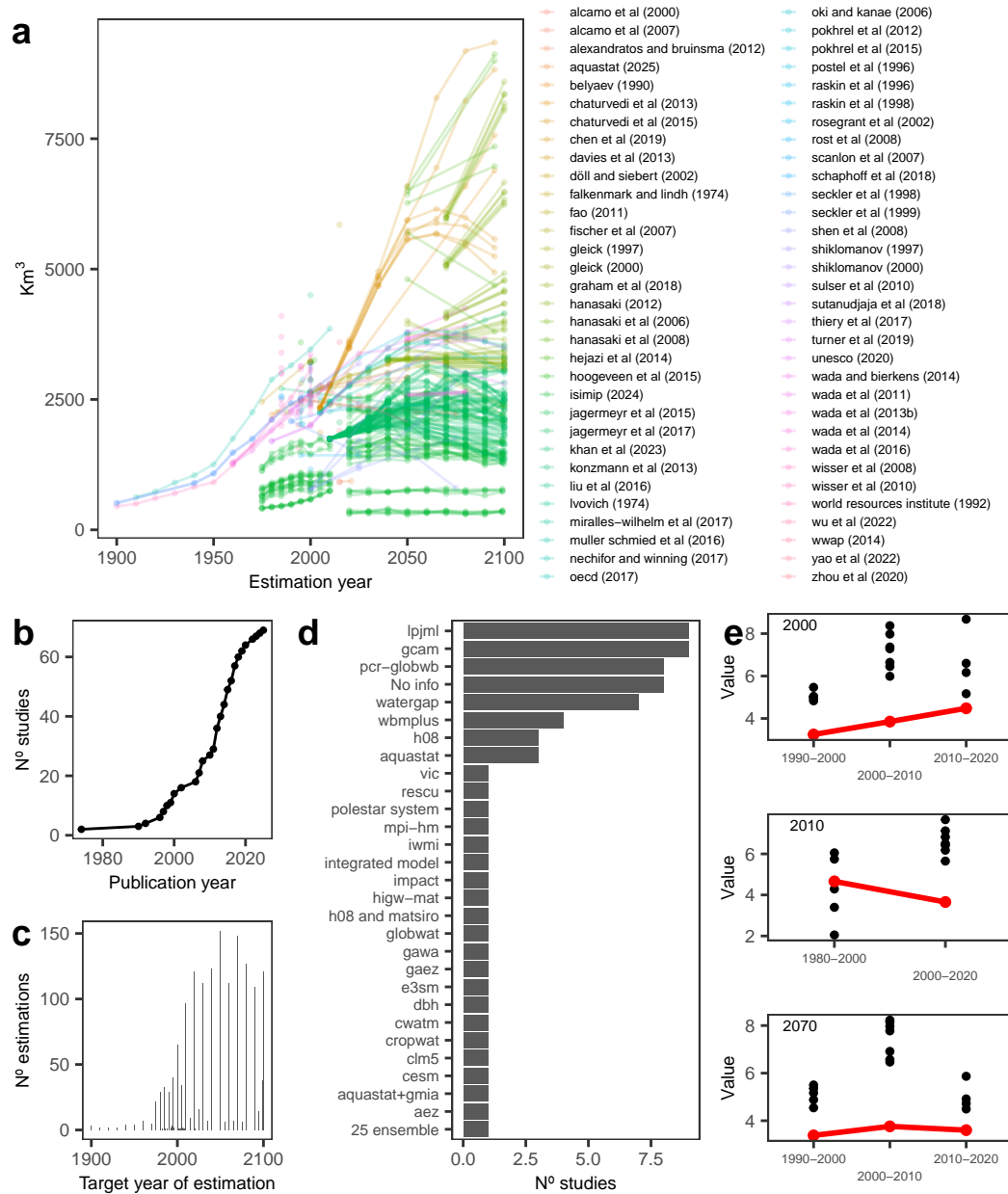
```
heat.tree(tree_model, type = 2, extra = 100, gap = 0, fallen.leaves = FALSE,
  tweak = 1.4, clip.right.labs = FALSE)
```



```

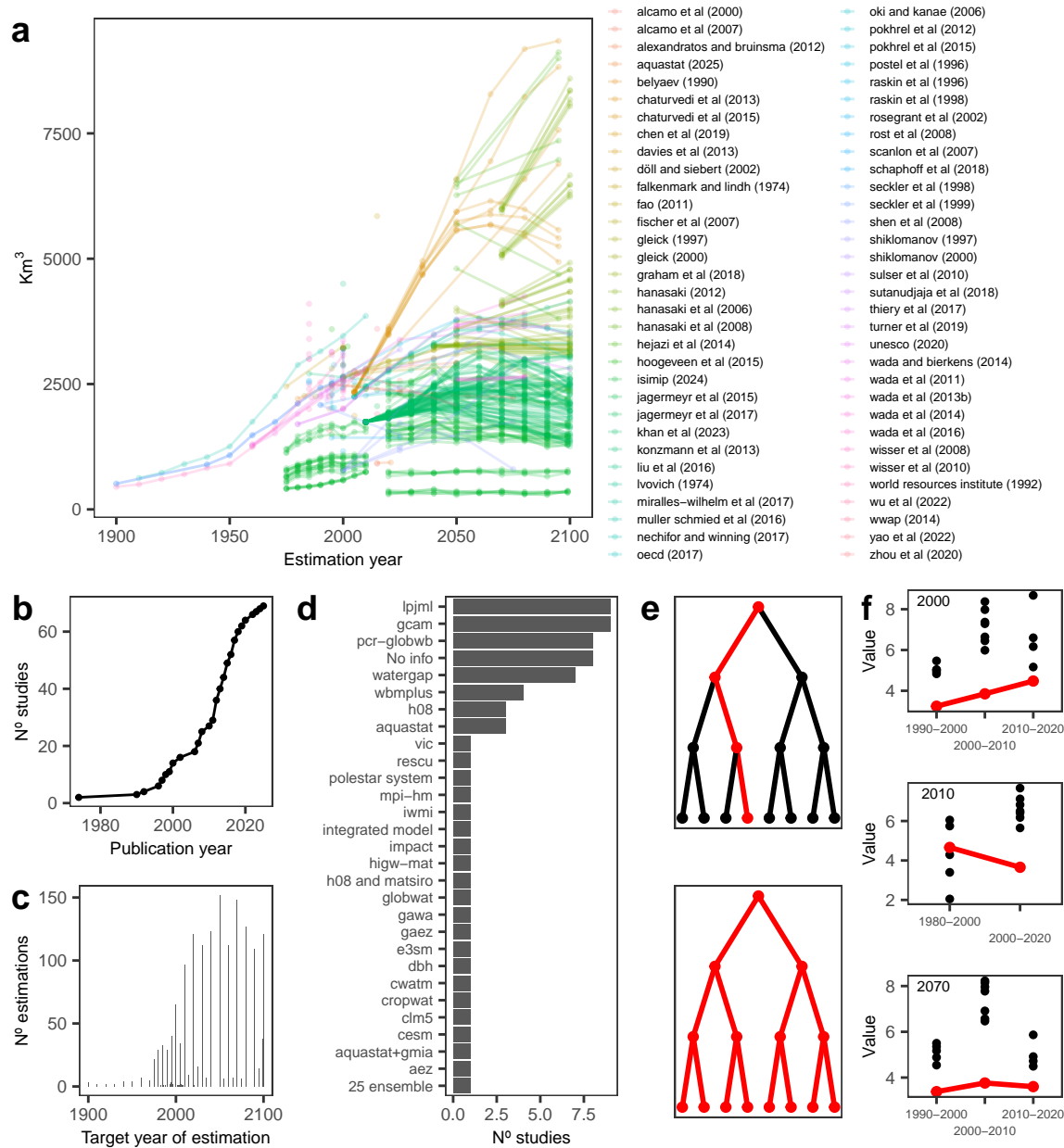
left <- plot_grid(cumulative.iww, plot.bar, ncol = 1, labels = c("b", "c"))
bottom <- plot_grid(left, plot.models, ncol = 2, labels = c("", "d"), rel_widths = c(0.4, 0.6))
bottom.right <- plot_grid(bottom, plot.examples.trends.data, ncol = 2, rel_widths = c(0.7, 0.3))
plot_grid(plot.iww, bottom.right, ncol = 1, rel_heights = c(0.5, 0.5), labels = c("a", ""))

```



# MERGE SA #####

```
plot.examples.trends.data <- plot_grid(p1, p2, p3, ncol = 1)
left <- plot_grid(cumulative.iww, plot.bar, ncol = 1, labels = c("b", "c"))
forking.paths <- plot_grid(one.path, all.paths, ncol = 1)
bottom <- plot_grid(left, plot.models, forking.paths, ncol = 3, labels = c("", "d", "e"),
  rel_widths = c(0.33, 0.4, 0.26))
bottom.right <- plot_grid(bottom, plot.examples.trends.data, ncol = 2, rel_widths = c(0.8, 0.2),
  labels = c("", "f"))
plot_grid(plot.iww, bottom.right, ncol = 1, rel_heights = c(0.5, 0.5), labels = c("a", ""))
```



# SENSITIVITY ANALYSIS PLOT BY FACET #####

```
plot.sa.facet <- final.dt %>%
  melt(., measure.vars = c("target_year", "target_year_interval", "interval",
    "rolling_window_factor", "metric")) %>%
  .[, .N, .(variable, value, trend)] %>%
  .[, value := gsub("_normalized", "_n", value)] %>%
  ggplot(., aes(value, N, fill = trend)) +
  scale_fill_manual(values = selected_colors, name = "Uncertainty") +
  geom_bar(stat = "identity", position = position_dodge(0.5)) +
  facet_wrap(~variable, scale = "free", ncol = 1) +
  labs(x = "", y = "N° simulations") +
  scale_y_continuous(breaks = breaks_pretty(n = 3)) +
```

```

theme_AP() +
coord_flip() +
theme(legend.position = "none",
      axis.text.y = element_text(size = 5.5),
      plot.margin = unit(c(0.05, 0, 0, 0.05), "cm"))

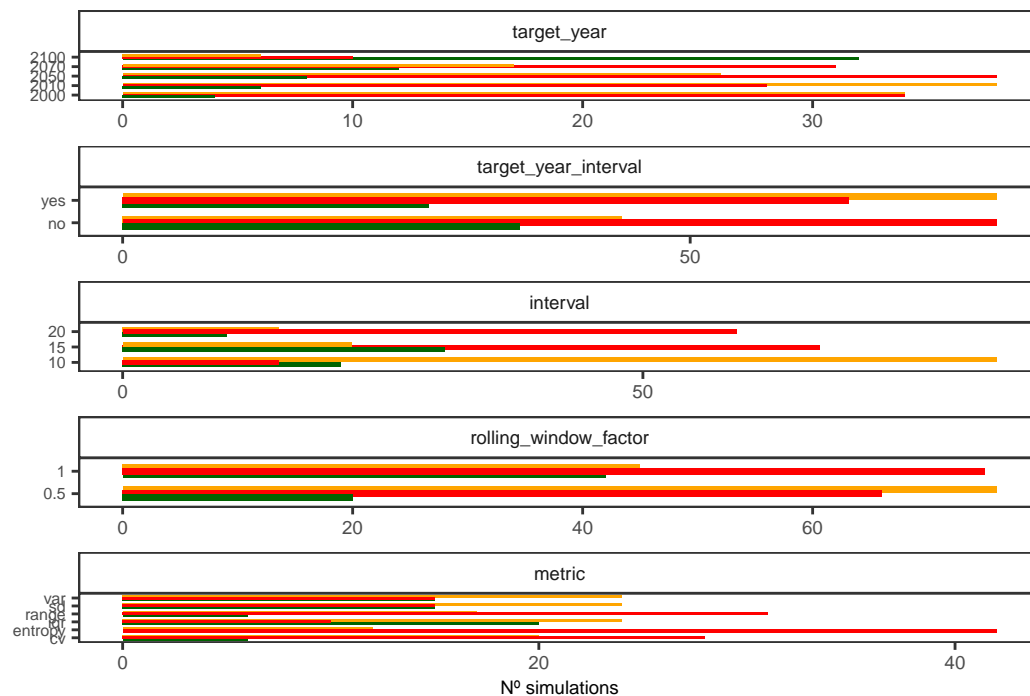
```

```

## Warning in melt.data.table(., measure.vars = c("target_year",
## "target_year_interval", : 'measure.vars' [target_year, target_year_interval,
## interval, rolling_window_factor, ...] are not all of the same type. By order of
## hierarchy, the molten data value column will be of type 'character'. All
## measure variables not of type 'character' will be coerced too. Check DETAILS in
## ?melt.data.table for more on coercion.

```

```
plot.sa.facet
```



```
# MERGE SENSITIVITY ANALYSIS PLOTS #####
```

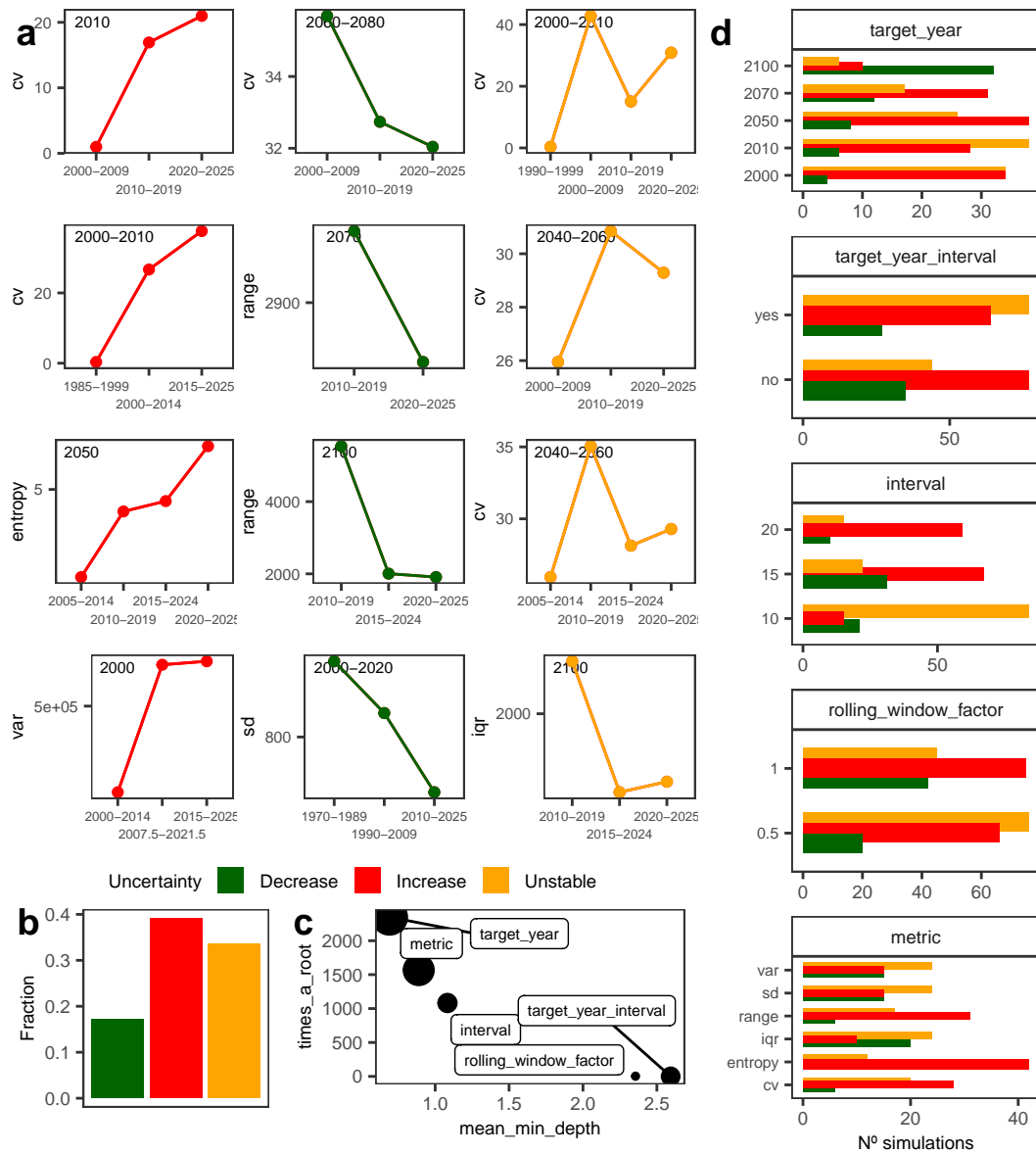
```

legend <- get_legend_fun(plot.fraction + theme(legend.position = "top"))
top <- plot_grid(plot.fraction + theme(legend.position = "none"), plot.multiway, ncol = 2,
                 rel_widths = c(0.4, 0.6), labels = c("b", "c"))
top.with.legend <- plot_grid(legend, top, rel_heights = c(0.1, 0.9), ncol = 1)

left <- plot_grid(plots.examples.trends, top.with.legend, ncol = 1,
                 rel_heights = c(0.75, 0.25))
plot_grid(left, plot.sa.facet, ncol = 2, rel_widths = c(0.67, 0.33),
          labels = c("a", "d"))

```





# CHECK METRIC EFFECT #####

```
data.dt <- lapply(trend, function(x) x[["data"]]) %>%
  rbindlist(., idcol = "row") %>%
  .[order(publication_period)] %>%
  .[publication_period == "2020-2029", publication_period := "2020-2025"]
```

# Plot trends based on metric 1 -----

```
row.id <- 4
```

```
final.dt[row %in% seq(row.id, nrow(forking_paths), 60)]
```

```
## target_year target_year_interval interval rolling_window_factor metric
## 1: 2070 yes 10 1 cv
```

```
## 2:      2070      yes      10      1      range
## 3:      2070      yes      10      1      sd
## 4:      2070      yes      10      1      var
## 5:      2070      yes      10      1      entropy
## 6:      2070      yes      10      1      iqr
##      trend row
## 1: Decrease    4
## 2: Unstable   64
## 3: Unstable  124
## 4: Unstable  184
## 5: Increase  244
## 6: Unstable  304
```

```
tmp <- data.dt[row == row.id] %>%
  merge(., final.dt, by = "row")
metrics <- c("cv", "sd", "entropy")

results <- lapply(metrics, function(m)
  tmp %>%
    .[, calculate_uncertainty_fun(data = .SD, metric = m), .(publication_period, period_midp)] %>%
    .[order(publication_period)] %>%
    .[!V1 == 0] %>%
    .[, trend:= lapply(.SD, check_order_fun), .SDcols = "V1"] %>%
    .[, publication_period:= gsub("-", "-", publication_period)] %>%
    .[publication_period == "2020-2029", publication_period:= "2020-2025"]
)

names(results) <- metrics

data.trend <- lapply(results, function(x)
  x[, scaled:= scale_to_range_fun(data = .SD, column = "V1",
                                ref_data = tmp, ref_column = "value")] %>%
  lapply(., data.table) %>%
  rbindlist(., idcol = "metric")

p1 <- tmp %>%
  .[!is.na(publication_period)] %>%
  .[, publication_period:= gsub("-", "-", publication_period)] %>%
  .[publication_period == "2020-2029", publication_period:= "2020-2025"] %>%
  ggplot(., aes(publication_period, value)) +
  geom_point(size = 0.3) +
  geom_line(data = data.trend, aes(x = publication_period,
                                   y = scaled, lty = metric, group = metric,
                                   color = trend),
            linewidth = 0.5) +
  scale_color_manual(values = c("Increase" = "red", "Decrease" = "darkgreen",
                                "Unstable" = "orange")) +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
```

```

theme_AP() +
labs(x = "", y = bquote("Km"^3), linetype = NULL) +
theme(axis.text.x = element_text(size = 6.3),
      axis.text.y = element_text(size = 6.3),
      axis.title.y = element_text(size = 6.5),
      plot.margin = unit(c(0.05, 0.05, 0, 0.05), "cm"),
      legend.position = "top") +
annotate("text", x = 0.1 + 0.5, y = max(tmp$value),
        label = unique(tmp$target_year), hjust = 0, vjust = 1,
        size = 2)

p.withoutline <- tmp %>%
  .[!is.na(publication_period)] %>%
  ggplot(., aes(publication_period, value)) +
  geom_point(size = 0.3) +
  geom_line(data = data.trend, aes(x = publication_period,
                                   y = scaled, group = metric,
                                   color = trend),
            linewidth = 0.5) +
  scale_color_manual(values = c("Increase" = "red", "Decrease" = "darkgreen",
                                "Unstable" = "orange"),
                    name = "Uncertainty") +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_AP() +
labs(x = "", y = bquote("Km"^3), linetype = NULL) +
theme(axis.text.x = element_text(size = 6.3),
      axis.text.y = element_text(size = 6.3),
      axis.title.y = element_text(size = 6.5),
      plot.margin = unit(c(0.05, 0.05, 0, 0.05), "cm")) +
annotate("text", x = 0.1 + 0.5, y = max(tmp$value),
        label = unique(tmp$target_year), hjust = 0, vjust = 1,
        size = 2)

# Plot trends based on metric 2 -----

row.id <- 7

final.dt[row %in% seq(row.id, nrow(forking_paths), 60)]

##   target_year target_year_interval interval rolling_window_factor metric
## 1:      2010                no      10                1         cv
## 2:      2010                no      10                1        range
## 3:      2010                no      10                1          sd
## 4:      2010                no      10                1          var
## 5:      2010                no      10                1    entropy
## 6:      2010                no      10                1          iqr

```

```

##          trend row
## 1: Increase    7
## 2: Increase   67
## 3: Unstable  127
## 4: Unstable  187
## 5: Unstable  247
## 6: Unstable  307

tmp <- data.dt[row == row.id] %>%
  merge(., final.dt, by = "row")
metrics <- c("range", "cv", "sd")

results <- lapply(metrics, function(m)
  tmp %>%
    .[, calculate_uncertainty_fun(data = .SD, metric = m), .(publication_period, period_midpoint)] %>%
    .[order(publication_period)] %>%
    .[!V1 == 0] %>%
    .[, trend:= lapply(.SD, check_order_fun), .SDcols = "V1"] %>%
    .[, publication_period:= gsub("-", "-", publication_period)] %>%
    .[publication_period == "2020-2029", publication_period:= "2020-2025"]
)

names(results) <- metrics

data.trend <- lapply(results, function(x)
  x[, scaled:= scale_to_range_fun(data = .SD, column = "V1", ref_data = tmp,
    ref_column = "value")] %>%
  lapply(., data.table) %>%
  rbindlist(., idcol = "metric")

p2 <- tmp %>%
  .[!is.na(publication_period)] %>%
  .[, publication_period:= gsub("-", "-", publication_period)] %>%
  .[publication_period == "2020-2029", publication_period:= "2020-2025"] %>%
  ggplot(., aes(publication_period, value)) +
  geom_point(size = 0.3) +
  geom_line(data = data.trend, aes(x = publication_period,
    y = scaled, color = trend,
    group = metric, lty = metric),
    linewidth = 0.5) +
  scale_color_manual(values = c("Increase" = "red", "Decrease" = "darkgreen",
    "Unstable" = "orange")) +
  guides(color = "none") +
  labs(x = "", y = "", linetype = NULL) +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_AP() +
  theme(axis.text.x = element_text(size = 6.3),

```

```

axis.text.y = element_text(size = 6.3),
axis.title.y = element_text(size = 6.5),
plot.margin = unit(c(0.05, 0.05, 0, 0.05), "cm"),
legend.position = "top") +
annotate("text", x = 0.1 + 0.5, y = max(tmp$value),
         label = unique(tmp$target_year), hjust = 0, vjust = 1,
         size = 2)

# plot trends based on metrics 3 -----

row.id <- 40

final.dt[row %in% seq(row.id, nrow(forking_paths), 60)]

##   target_year target_year_interval interval rolling_window_factor metric
## 1:         2100                   no         10                0.5    cv
## 2:         2100                   no         10                0.5  range
## 3:         2100                   no         10                0.5    sd
## 4:         2100                   no         10                0.5   var
## 5:         2100                   no         10                0.5 entropy
## 6:         2100                   no         10                0.5   iqr
##      trend row
## 1: Unstable  40
## 2: Decrease 100
## 3: Decrease 160
## 4: Decrease 220
## 5: Unstable 280
## 6: Unstable 340

tmp <- data.dt[row == row.id] %>%
  merge(., final.dt, by = "row")
metrics <- c("range", "var", "iqr")

results <- lapply(metrics, function(m)
  tmp %>%
    .[, calculate_uncertainty_fun(data = .SD, metric = m), .(publication_period, period_midpoint)] %>%
    .[order(publication_period)] %>%
    .[!V1 == 0] %>%
    .[, trend:= lapply(.SD, check_order_fun), .SDcols = "V1"] %>%
    .[, publication_period:= gsub("-", "-", publication_period)] %>%
    .[publication_period == "2020-2029", publication_period:= "2020-2025"]
)

names(results) <- metrics

data.trend <- lapply(results, function(x)
  x[, scaled:= scale_to_range_fun(data = .SD, column = "V1", ref_data = tmp,

```

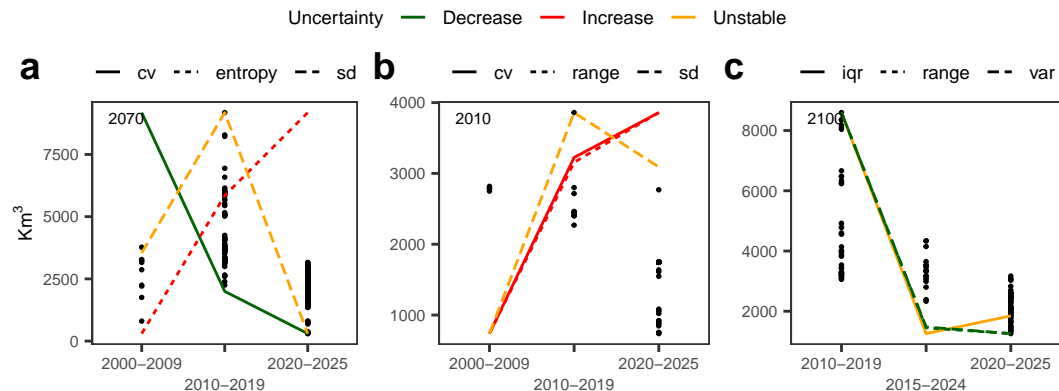
```

                                ref_column = "value"]]) %>%
lapply(., data.table) %>%
rbindlist(., idcol = "metric")

p3 <- tmp %>%
  .[!is.na(publication_period)] %>%
  .[, publication_period:= gsub("-", "", publication_period)] %>%
  .[publication_period == "2020-2029", publication_period:= "2020-2025"] %>%
  ggplot(., aes(publication_period, value)) +
  geom_point(size = 0.3) +
  geom_line(data = data.trend, aes(x = publication_period,
                                   y = scaled, color = trend,
                                   group = metric, lty = metric),
            linewidth = 0.5) +
  scale_color_manual(values = c("Increase" = "red", "Decrease" = "darkgreen",
                                "Unstable" = "orange")) +
  guides(color = "none") +
  labs(x = "", y = "", linetype = NULL) +
  scale_x_discrete(guide = guide_axis(n.dodge = 2)) +
  theme_AP() +
  theme(axis.text.x = element_text(size = 6.3),
        axis.text.y = element_text(size = 6.3),
        axis.title.y = element_text(size = 6.5),
        plot.margin = unit(c(0.05, 0.05, 0, 0.05), "cm"),
        legend.position = "top") +
  annotate("text", x = 0.1 + 0.5, y = max(tmp$value),
          label = unique(tmp$target_year), hjust = 0, vjust = 1,
          size = 2)

da <- get_legend_fun(p.withoutline + theme(legend.position = "top"))
di <- plot_grid(p1 + guides(color = "none"), p2,p3, ncol = 3, labels = "auto")
plot_grid(da, di, rel_heights = c(0.1, 0.9), ncol = 1)

```



### 3 Session information

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

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29)
## 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] randomForest_4.7-1.2 brms_2.22.0      Rcpp_1.0.13-1
##  [4] mgcv_1.9-1           nlme_3.1-166     microbenchmark_1.5.0
##  [7] lme4_1.1-35.5        Matrix_1.6-5     here_1.0.1
## [10] doParallel_1.0.17    iterators_1.0.14 foreach_1.5.2
## [13] rworldmap_1.3-8      sp_2.1-4         countrycode_1.6.0
## [16] ncd4_1.23            scales_1.3.0     wesanderson_0.3.7
## [19] benchmarkme_1.0.8    cowplot_1.1.3    lubridate_1.9.3
## [22] forcats_1.0.0        stringr_1.5.1    dplyr_1.1.4
## [25] purrr_1.0.2          readr_2.1.5      tidyr_1.3.1
## [28] tibble_3.2.1         ggplot2_3.5.1    tidyverse_2.0.0
## [31] data.table_1.16.2    openxlsx_4.2.7.1
##
## loaded via a namespace (and not attached):
##  [1] Rdpack_2.6.2          rlang_1.1.4       magrittr_2.0.3
##  [4] matrixStats_1.4.1     compiler_4.3.3    loo_2.8.0
##  [7] vctrs_0.6.5           maps_3.4.2.1      crayon_1.5.3
## [10] pkgconfig_2.0.3       fastmap_1.2.0     backports_1.5.0
## [13] labeling_0.4.3        utf8_1.2.4        rmarkdown_2.29
## [16] tzdb_0.4.0            nloptr_2.1.1      tinytex_0.54
## [19] xfun_0.49             terra_1.7-78      R6_2.5.1
## [22] stringi_1.8.4         boot_1.3-31       estimability_1.5.1
## [25] knitr_1.49            fields_16.3        bayesplot_1.11.1
## [28] splines_4.3.3         timechange_0.3.0  tidyselect_1.2.1
```

```
## [31] rstudioapi_0.17.1      abind_1.4-8            yaml_2.3.10
## [34] codetools_0.2-20       lattice_0.22-6         withr_3.0.2
## [37] bridgesampling_1.1-2   benchmarkmeData_1.0.4 posterior_1.6.0
## [40] coda_0.19-4.1          evaluate_1.0.1         RcppParallel_5.1.9
## [43] zip_2.3.1              pillar_1.9.0           tensorA_0.36.2.1
## [46] checkmate_2.3.2        distributional_0.5.0    generics_0.1.3
## [49] rprojroot_2.0.4        hms_1.1.3              rstantools_2.4.0
## [52] munsell_0.5.1          minqa_1.2.8            sensobol_1.1.5
## [55] xtable_1.8-4           glue_1.8.0             emmeans_1.10.5
## [58] tools_4.3.3            mvtnorm_1.3-2          dotCall64_1.2
## [61] grid_4.3.3             rbibutils_2.3          colorspace_2.1-1
## [64] raster_3.6-30          cli_3.6.3              spam_2.11-0
## [67] fansi_1.0.6            viridisLite_0.4.2      Brobdingnag_1.2-9
## [70] gtable_0.3.6           digest_0.6.37          farver_2.1.2
## [73] htmltools_0.5.8.1      lifecycle_1.0.4        httr_1.4.7
## [76] MASS_7.3-60.0.1
```

```
## 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
```