Tutorial for GORIC(A) evidence aggregation

Rebecca M. Kuiper

15 August 2024

Contents

This is a tutorial for performing GORIC(A) evidence aggregation using the `evSyn` function from the `restriktor` package. With GORIC(A) evidence aggregation, one aggregates / combines / synthesizes evidence / support for one or more central theories obtained from multiple (diverse / heterogeneous) studies using the GORIC(A).

Two examples using real data are provided. The first example is a step-by-step guide that illustrates GORIC(A) evidence aggregation for a central theory regarding a single relationship, using multiple heterogeneous/diverse studies.

In the second example, GORIC(A) evidence aggregation is applied to several sets of multiple hypotheses that reflect central theories. Here, data from the research by Berkey and colleagues (1998) is used, which is available in the `metafor` library.

In each primary study, a central theory is reflected by a (statistical) hypothesis. This hypothesis may differ per primary study, as long as it reflects the central theory. The last section gives an overview of how to specify (study-specific) sets of (study-specific) hypotheses in the `evSyn` function.

The `evSyn` function currently accepts the following objects as input:
- a list of vectors with (standardized) parameter estimates;
- a list of vectors with log-likelihood values together with a list of vectors with penalty values;
- a list of vectors with GORIC(A) weights;
- a list of vectors with GORIC(A) values;
as will become clear in the second example.

Three types of approaches to evidence aggregation are possible: “equal”, “added”, and “average”. The latter should be used when doing a multiverse analysis and, thus, when the aim is to average results from multiple analysis on one data set (due to different operationalization choices). The other two can be used when aggregating results from multiple studies.
In case of an equal-evidence approach, evidence from N = 5 studies with n = 100 observations (with support for the central theory) is the same as obtaining evidence from 1 study with n = 5 * 100 = 500 observations (with support for the central theory) – as in a meta-analysis. In the added-evidence approach, the evidence

aggregated from N = 5 studies with n = 100 observations (with support for the central theory) is stronger than if the data came from a single study with n = 5 * 100 = 500 observations (with support for the central theory).
For ease, this tutorial only shows code for the added approach, but one can easily adjust it to the equal approach by specifying "type = 'equal' " in the `evSyn` function.

Notes:
- In the examples, the evidence from 4 and 5 primary studies are aggregated. It is of course also possible to perform GORIC(A) evidence aggregation on many more studies.
- For (more) information regarding interpreting the GORIC(A) output, see 'Guidelines_output_GORIC' (https://github.com/rebeccakuiper/Tutorials).

Example 1: Relationship previous experience and trust

This example uses data from four studies that investigate whether buyer's trust is affected by previous experiences between a buyer and a seller. The studies investigated this relationship with different statistical models; namely, with linear, probit, and three-level logistic regressions models (which make the parameter estimates incomparable).

The central theory is that previous experience is positively related to trust. The (study-specific) hypothesis that reflect this central theory is $H_+ : \beta_{past} > 0$, in all four studies.

In each study, the GORICA weights denote the support for the hypothesis of interest and thus the study-specific support for the central theory. Using GORICA evidence aggregation, the combined/aggregated/synthesized support/evidence for the central theory over all four diverse studies is obtained. Next, the corresponding R code is given.

Data preparation

Load the data and inspect it.

```
dat <- read.table('data/Kuiper2012estimates.txt',
                  header = TRUE, sep = '\t')
dat
```

```
               study year estimate    se
1   Batenburg et al. 2003    0.090 0.029
2   Buskens and Raub 2002    0.140 0.054
3 Buskens and Weesie 2000    1.090 0.093
4     Buskens et al. 2010    1.781 0.179
```

The data consists of four rows and four columns. Each row represents a study. The columns are: `study`: the authors of the study, `year`: the year the study was published, `estimate`: the estimate of β_{past}, `se`: the standard error of the estimate.

Only the estimates and their standard errors (or better, their variances) are needed to perform GORICA evidence aggregation. Below, you find code to load the `evSyn` function from the `restriktor` package, which is needed to perform the analysis. You can find also code to show the documentation for the function in the help-tab.

```
## First, install the package, if you have not done this already:
if (!require("restriktor")) install.packages("restriktor")

## Then, load the packages:
library(restriktor) # for the goric function

# If you want to use restriktor from github:
```

```r
#if (!require("devtools")) install.packages("devtools")
#library(devtools)
#install_github("LeonardV/restriktor")
#library(restriktor) # for goric function

# print docs in the help-tab to view arguments and explanations for the function
#?evSyn
```

Data for evSyn()

The `evSyn` function requires some pre-processing of the data. Namely, it needs the estimates and their variances as a list of named vectors and as a list of (unnamed) covariance matrices, respectively. Each element in the list must represent a single study.

```r
# set estimates as named vectors and store in list.
estimates <- lapply(dat$estimate, setNames, "beta_past")

# store variances (i.e., the square of the standard errors)
# in list of matrices.
covmats <- lapply(dat$se^2, as.matrix)
```

Hypotheses

One also needs to specify the (study-specific) theory-based hypothesis to be evaluated, using the name/label of the parameters. The expectation is that positive experiences with the seller in the past will have a positive effect on the buyer's trust. In this example, the same theory-based hypothesis can be used for all studies:

```r
Hpos <- 'beta_past > 0'
```

This hypothesis is evaluated against its complement; which equals $'beta\_past < 0'$, in this case. Notably, when the hypothesis and its complement obtain equal support, then this would indicate support for the boundary, that is, the equality-constrained hypothesis $'beta\_past = 0'$.

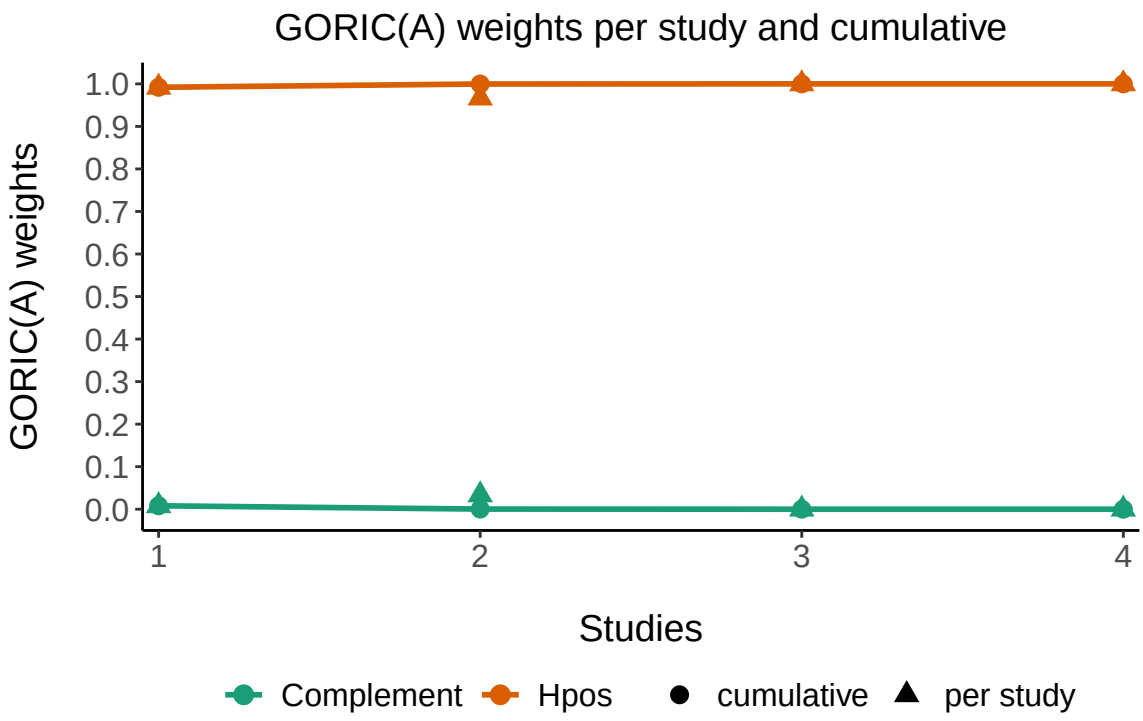
GORICA Evidence Synthesis

Using the created lists and hypothesis, run `evSyn` and obtain the results for GORICA Evidence Synthesis:

```r
evSyn_trust <- evSyn(estimates, VCOV = covmats,
                     hypotheses = list(Hpos = Hpos),
                     # type = 'added', # default
                     comparison = 'complement')

evSyn_trust
```

```
restriktor (0.5-80): added Evidence Synthesis results:

Final GORICA weights:
      Hpos  Complement
     1.000       0.000
---

Ratio final GORICA weights:
            vs. Hpos  vs. Complement
Hpos           1.000        7.54e+54
Complement     0.000           1.000
```

```r
#summary(evSyn_trust)
plot(evSyn_trust)
```




```r
round(evSyn_trust$Final_ratio_GORICA_weights, 2)
```

```
            vs. Hpos vs. Complement
Hpos               1   7.535163e+54
Complement         0   1.000000e+00
```

Here, one can see that the final/overall aggregated GORICA weights imply that H_+ (Hpos) has full support. Often a ratio of GORICA weights helps in interpreting the results. Such a ratio represents the relative aggregated support for one hypothesis (or better: central theory) over the other. It is presented in a $M \times M$ matrix, where M is the number of hypotheses in the set, where the diagonals represent the aggregated support of a hypothesis with itself, which is always one. To quantify the support for the central theory: The central theory H_+ has 7.5e+54 (infinite) times more support that its complement, that is, all other possible theories. The conclusion, thus, would be that positive past experiences with a seller have a positive effect on the buyer's trust in that seller.

Example 2: Comparing surgical and non-surgical treatments for periodontal disease

Background Information & Data

In the study by Berkey and colleagues (1998), surgical and non-surgical treatments for medium-severity periodontal disease are compared in five trials (i.e., 5 studies). Two outcomes reflect severity: attachment level (AL) and probing depth (PD) one year after the treatment. Higher attachment levels and lower probing

depth are indicators of less severity, and are thus preferred.

The effect sizes to be aggregated are the (raw) mean differences, where non-surgical treatment is the reference category. The outcomes are calculated such that a positive value indicates that surgery was more effective in reducing severity than non-surgical treatment. In summary, a positive estimate indicates effectiveness of surgery in either increasing the attachment level or decreasing the probing depth.

First, install and call the `metafor` library to load data from Berkey (1998) and the `restriktor` library to load the evSyn() function. If needed, it is possible to view the description of the data set and function with the `?` operator or the `help()` command.

```
# To install the packages in R:
#if (!require("restriktor")) install.packages("restriktor")

# To install restriktor from github:
# if (!require("devtools")) install.packages("devtools")
# library(devtools)
# install_github("LeonardV/restriktor")
library(restriktor)
#
# print docs in the help-tab to view arguments and explanations for the function
#?evSyn

# Data
library(metafor)
dat <- dat.berkey1998
#
# print docs in the help-tab to view explanations for the data set
#?dat.berkey1998
```

Let us inspect the `data.frame`.

```
View(dat)
```

Each `trial` is one study into the comparison of surgical and non-surgical treatments for severity of periodontal disease one year after treatment. Every study contains two rows in the `dat` data.frame, one for probing depth (PD) and one for attachment level (AL).

To prepare the data in such a way that it can be used as input for evSyn(), store the estimates and covariance matrices in a list where each element of the list represents a single study.

```
# Store estimates in list of named vectors
estimates <- lapply(seq(1, nrow(dat), 2), function(i)
  setNames(c(dat$yi[i:(i+1)]), c('PD', 'AL')))
# Note that these names will be used in specifying the hypotheses.

# Store (co-)variances in list of (unnamed) variance-covariance matrices.
covmats <- lapply(seq(1, nrow(dat), 2), function(i)
  as.matrix(dat[i:(i+1), c('v1i', 'v2i')]))
# Note that it does not matter what the names/labels are;
# for the estimates it does matter.
```

Hypotheses

Below, you can find three possible (hypothetical) sets of central theories (and the corresponding hypotheses that reflect those).

Set 1

One may hypothesize from theory that surgery has a positive effect on probing depth, but a negative effect on attachment level. In each study, this can be reflected by: $H_{1.1} : \delta_{AL} < 0;\ \delta_{PD} > 0$, where δ denotes the difference in means between surgical and non-surgical treatments.
There may also be reason to believe that surgery is too invasive, and thus non-surgery decreases severity on both outcomes compared to surgery: $H_{1.2} : \delta_{AL} < 0;\ \delta_{PD} < 0$.

Set 2
Alternatively, it can be the case that there is theory regarding the magnitude of the effect size and one wants to compare it to some cut-off value for that specific effect size type (e.g., when using Cohen's d or Hedges' g). In this example, there might be theory regarding the size of the mean differences stating: $H_2 : |\delta_{AL}| > 0.3;\ |\delta_{PD}| > 0.3$. This states that the difference between the treatments on both outcomes is larger than 0.3.

Set 3
As a final example, it might be expected from theory that the absolute size of the effect for AL is smaller than for PD. In this example, the comparison of effect sizes can be represented by: $H_3 : |\delta_{AL}| < |\delta_{PD}|$. This states that the absolute effect of surgery on attachment level is smaller than its absolute effect on probing depth.

Next, you can specify the (study-specific) hypotheses, where you now need the variables names or assigned names/labels of the estimates instead of the (Greek) parameter names:

```
# Set 1
H1.1 <- "AL < 0; PD > 0"
H1.2 <- "AL < 0; PD < 0"
#
# Set 2
H2 <- "abs(AL) > 0.3; abs(PD) > 0.3"
#
# Set 3
H3 <- "abs(AL) < abs(PD)"
```

GORICA Evidence Synthesis

using Estimates and Covariance Matrix

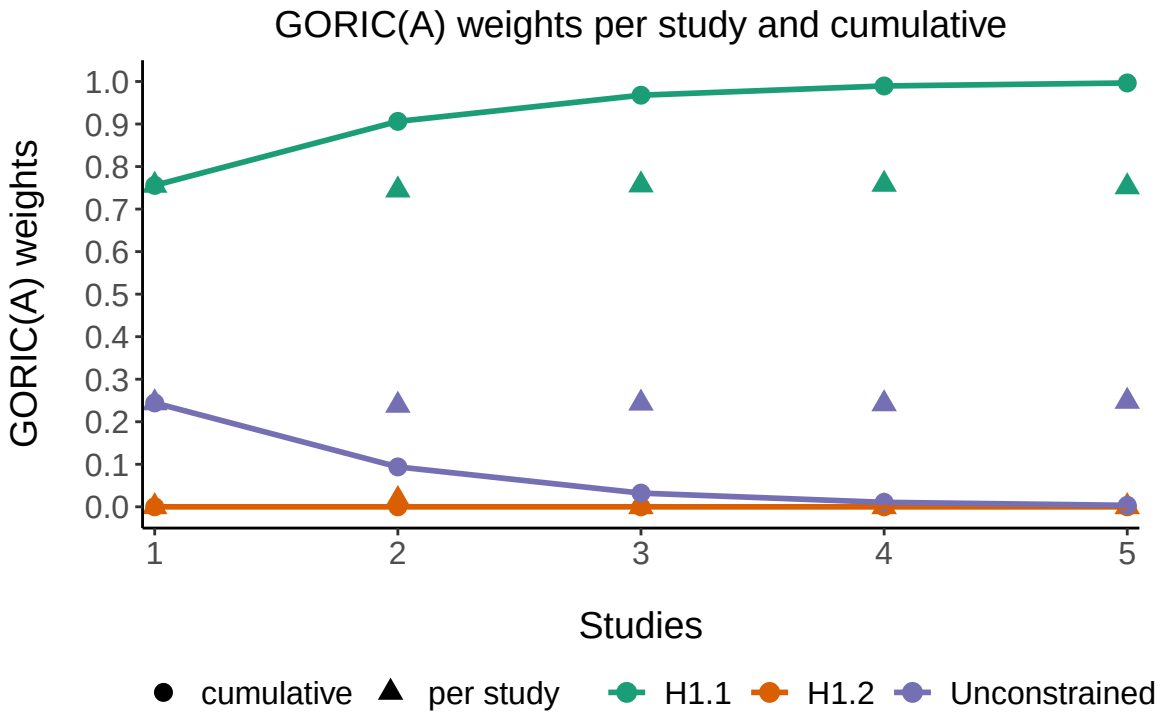
Set 1 In the first set, H1.1 and H1.2 are investigated together, as if these are the hypotheses of interest (and each study uses the same specification, that is, the same set of hypotheses is applied to each study).

```
results_Set1 <- evSyn(object = estimates, VCOV = covmats,
                      hypotheses = list(H1.1 = H1.1, H1.2 = H1.2)
                      #comparison = "unconstrained" # default
                      #type = "added" # default
                      )

results_Set1
```

```
restriktor (0.5-80): added Evidence Synthesis results:

Final GORICA weights:
        H1.1          H1.2  Unconstrained
       0.997         0.000          0.003
---

Ratio final GORICA weights:
              vs. H1.1  vs. H1.2  vs. Unconstrained
H1.1             1.000  4.95e+34            285.368
```

```
H1.2              0.000     1.000              0.000
Unconstrained     0.004  1.74e+32              1.000
```

```
# summary(results_Set1)
plot(results_Set1)
```



Note that the hypotheses are passed as strings within the `list` function, where you can also denote hypothesis names. Note further that the complement cannot be used yet as a safeguard hypothesis; for now, this option is restricted to single order-restricted / theory-based / informative hypothesis. However, if the goal is to compare informative hypotheses, one could just as well use the unconstrained.

When inspecting the Final ratios of GORICA weights:

```
round(results_Set1$Final_ratio_GORICA_weights, 2)
```

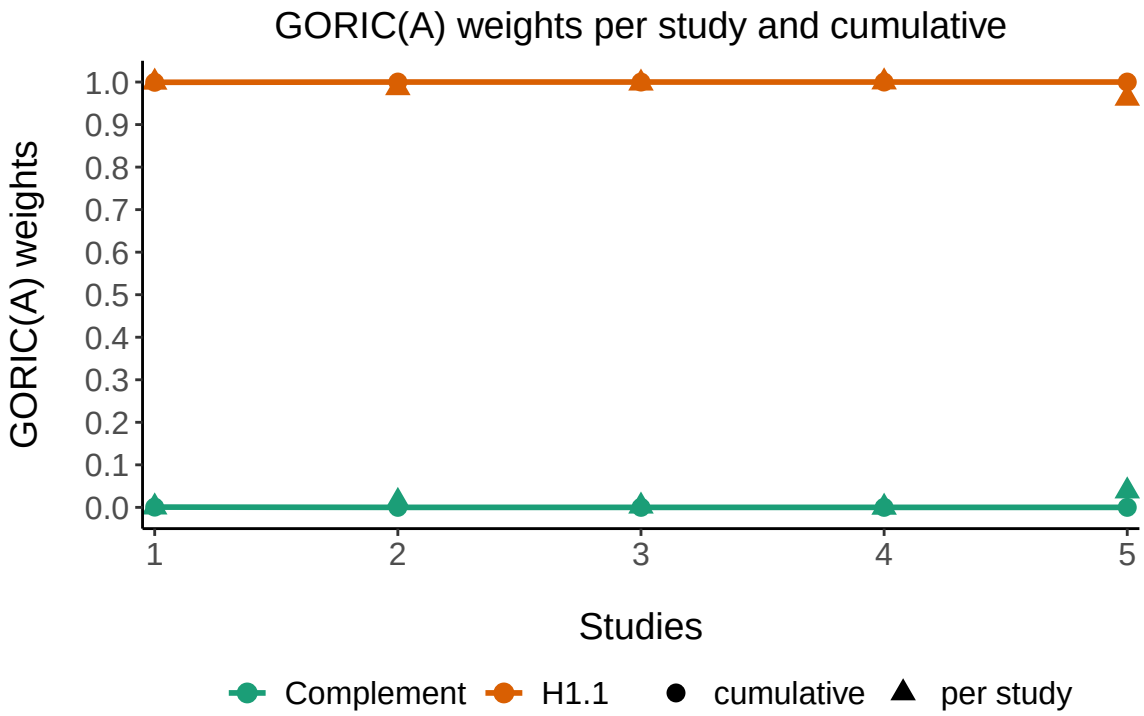
```
              vs. H1.1     vs. H1.2 vs. Unconstrained
H1.1                 1 4.953573e+34            285.37
H1.2                 0 1.000000e+00              0.00
Unconstrained        0 1.735855e+32              1.00
```

one can see that $H_{1.1}$ is not weak, since the GORICA weights ratio with the unconstrained is larger than 1. Furthermore, that it is convincingly preferred over its alternative $H_{1.2}$: $H_{1.1}$ is practically an infinite amount of times more likely than $H_{1.2}$, given the data. Therefore, it can be concluded that surgical performance has a positive effect on probing depth, but a negative effect on attachment level.

Now, you may also want to investigate $H_{1.1}$ against its complement, which may be helpful for future research:

```
results_Set1_H1c <- evSyn(estimates, VCOV = covmats,
                      hypotheses = list(H1.1 = H1.1),
                      comparison = 'complement')
```

```
#results_Set1_H1c
# summary(results_Set1_H1c)
plot(results_Set1_H1c)
```



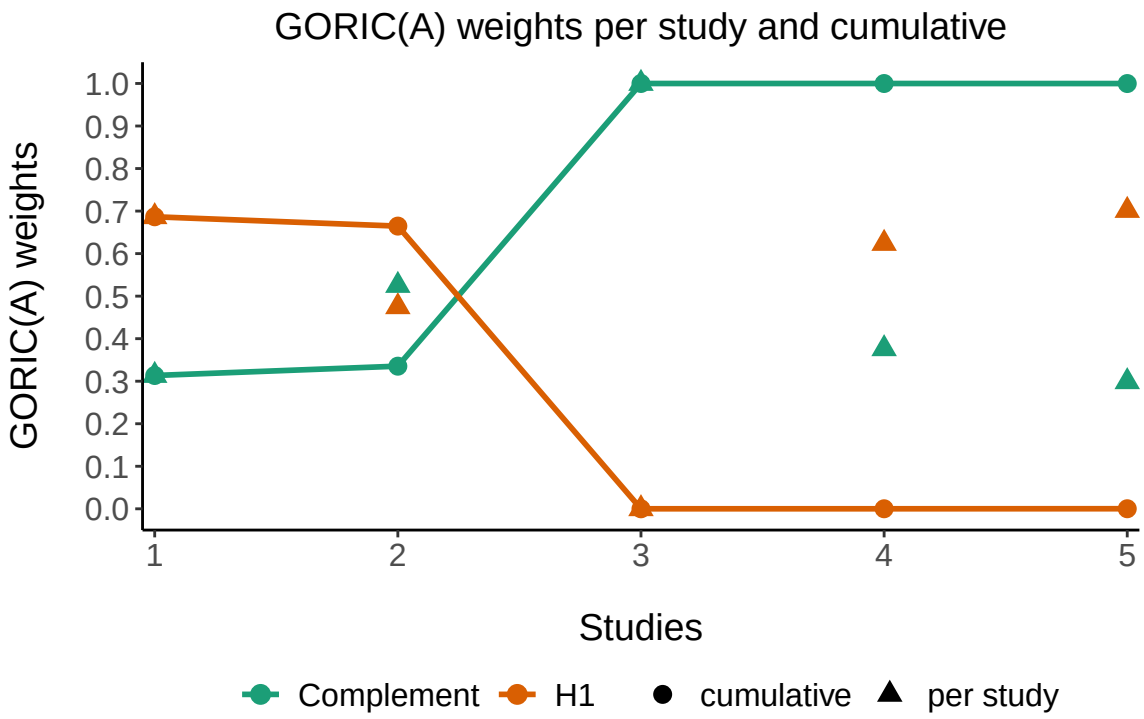

```
round(results_Set1_H1c$Final_ratio_GORICA_weights, 2)
```

```
            vs. H1.1 vs. Complement
H1.1               1   2.798726e+14
Complement         0   1.000000e+00
```

One can see that $H1.1$ is also much more likely than its complement, that is, all other possible hypotheses / theories.

Set 2 In the second set, H_2 is the supposed central theory / hypothesis of interest, which will be denoted as H1 because that would be what you probably use in practice when evaluating this set. H1 states that the mean difference between surgery and non-surgery is larger than 0.3 for both outcomes. Note that the same hypothesis specification can be used for all studies.

```
results_Set2 <- evSyn(estimates, VCOV = covmats,
                      hypotheses = list(H1 = H2),
                      comparison = 'complement')

#results_Set2
# summary(results_Set2)
plot(results_Set2)
```



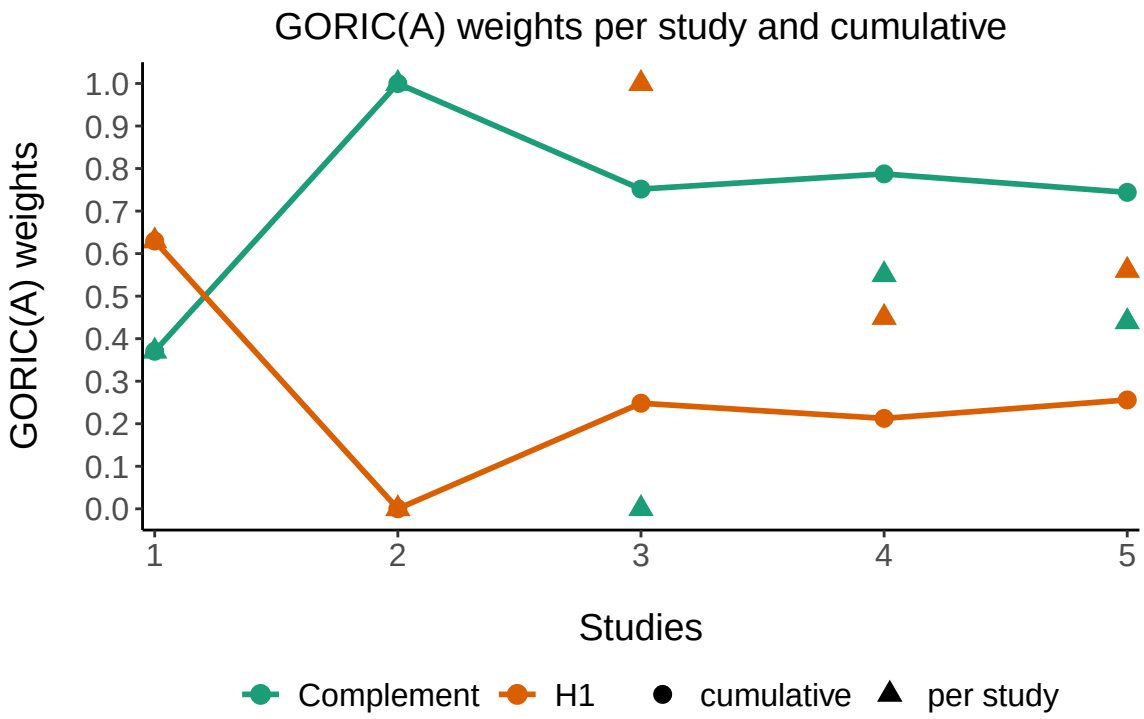

```
round(results_Set2$Final_ratio_GORICA_weights, 2)
```

```
            vs. H1 vs. Complement
H1            1.00              0
Complement 6374.84              1
```

It can be concluded that H1 (i.e., H_2) is a weak hypothesis (or better: central theory) as it is many times less likely than its complement. Hence, it is very unlikely that the absolute difference between the groups on both outcomes is larger than 0.3.
From the Study-specific results (available via the summary() function) and the plot, it can be seen that the studies show mixed results: most preferring H1 or the boundary of H1 and its complement, while Study 3 clearly prefers the complement of H1. When aggregating the evidence, this results (in this case) in strong support for the complement of H1.

Set 3 In the final set, the hypothesis of interest is H_3, H1 in this set, which states that the negative effect of surgery on attachment level is smaller than the positive effect on probing depth. Also here, the same hypothesis specification can be used for all studies.

```
results_Set3 <- evSyn(estimates, VCOV = covmats,
                      hypotheses = list(H1 = H3),
                      comparison = 'complement')

#results_Set3
# summary(results_Set3)
plot(results_Set3)
```




```
round(results_Set3$Final_ratio_GORICA_weights, 2)
```

```
           vs. H1 vs. Complement
H1           1.00           0.34
Complement   2.91           1.00
```

The output shows that the complement of H1 is 2.91 times more likely than H1 (i.e., H_3) itself.
From the Study-specific results (available via the summary() function) and the plot, it can be seen that the studies show mixed results. When aggregating the evidence, this results (in this case) in support for the complement of H1.

using GORIC(A) Values or GORIC(A) Weights

Estimates and their variance-covariance matrices are not necessary the perform evidence synthesis. The `evSyn` function also works with GORIC(A) values or GORIC(A) weights; as well as with log-likelihood and penalty values (which is shown in the next subsection).

The GORIC(A) values can be used as input by creating a vector with the c() command; e.g., c(gorica_H1, gorica_H2, gorica_Hu). Below, these are subtracted from a goric object.

```
# goric objects needed to obtain the GORICA values from
gorica_Study1 <- goric(estimates[[1]], VCOV = covmats[[1]],
                       hypotheses = list(H1.1=H1.1, H1.2=H1.2))
gorica_Study2 <- goric(estimates[[2]], VCOV = covmats[[2]],
                       hypotheses = list(H1.1=H1.1, H1.2=H1.2))
gorica_Study3 <- goric(estimates[[3]], VCOV = covmats[[3]],
                       hypotheses = list(H1.1=H1.1, H1.2=H1.2))
gorica_Study4 <- goric(estimates[[4]], VCOV = covmats[[4]],
```

```
                        hypotheses = list(H1.1=H1.1, H1.2=H1.2))
gorica_Study5 <- goric(estimates[[5]], VCOV = covmats[[5]],
                        hypotheses = list(H1.1=H1.1, H1.2=H1.2))

# list of GORICA values for each of the five studies
GORICA <- list(gorica_Study1$result[,4], gorica_Study2$result[,4],
               gorica_Study3$result[,4], gorica_Study4$result[,4],
               gorica_Study5$result[,4])
# GORICA

# GORICA evidence synthesis
results_Set1_gorica <- evSyn(GORICA)
#
#results_Set1_gorica
# summary(results_Set1_gorica)
#plot(results_Set1_gorica)
round(results_Set1_gorica$Final_ratio_GORICA_weights, 2)
```

```
   vs. H1       vs. H2 vs. H3
H1      1 4.953573e+34 285.37
H2      0 1.000000e+00   0.00
H3      0 1.735855e+32   1.00
```

The results lead us to the same output as above of course.

In cases where hypotheses names may be confusing, as it is here, it is recommendable that the user utilize the `hypo_names` function argument to make the results more accessible. In this example, the function would be:

```
results_Set1_gorica <- evSyn(GORICA, hypo_names = c("H1.1", "H1.2", "Hu"))
#
#results_Set1_gorica
# summary(results_Set1_gorica)
#plot(results_Set1_gorica)
round(results_Set1_gorica$Final_ratio_GORICA_weights, 2)
```

```
     vs. H1.1     vs. H1.2 vs. Hu
H1.1        1 4.953573e+34 285.37
H1.2        0 1.000000e+00   0.00
Hu          0 1.735855e+32   1.00
```

Notice that, when using GORIC(A) values, GORIC(A) weights, or log-likelihood and penalty values as input, you do not need to specify the hypotheses in `evSyn`. That is because the hypotheses were provided previously when calculating the values/weights with `goric`. Nevertheless, to avoid confusion, `hypo_names` can be used for a better labeling in the output.

using GORIC(A) weights Similarly, the GORIC(A) weights can be used:

```
# list of GORICA weights for each of the five studies
GORICAweights <- list(gorica_Study1$result[,7], gorica_Study2$result[,7],
               gorica_Study3$result[,7], gorica_Study4$result[,7],
               gorica_Study5$result[,7])
# GORICAweights

# GORICA evidence synthesis
results_Set1_gw <- evSyn(GORICAweights)
#
```

```
#results_Set1_gw
#summary(results_Set1_gw)
#plot(results_Set1_gw)
round(results_Set1_gw$Final_ratio_GORICA_weights, 2)
```

```
   vs. H1       vs. H2 vs. H3
H1      1 4.953573e+34 285.37
H2      0 1.000000e+00   0.00
H3      0 1.735855e+32   1.00
```

using Log-likelihood and Penalty Values

One can also perform evidence aggregation using log-likelihood and penalty values.

Here, these values (as vectors) are retrieved from the `goric` functions ran before. It is also possible to concatenate numerical values using `c()` and create two vectors: one containing the log-likelihood values of each study and one containing the penalties of each study. Then, you should make two lists of the study-specific vectors, which then should be fed into `evSyn`, in the `object` and `PT` arguments, respectively.

```
# Log-likelihood values
LL_Set1 <- list(gorica_Study1$result[,2], gorica_Study2$result[,2],
                gorica_Study3$result[,2], gorica_Study4$result[,2],
                gorica_Study5$result[,2])
# Penalty values
PT_Set1  <- list(gorica_Study1$result[,3], gorica_Study2$result[,3],
                 gorica_Study3$result[,3], gorica_Study4$result[,3],
                 gorica_Study5$result[,3])

# GORICA evidence synthesis
results_Set1_LLandPT <- evSyn(LL_Set1, PT = PT_Set1,
                              #type = "added", # Default
                              hypo_names = c("H1.1", "H1.2", "Hu"))
#
#results_Set1_LLandPT
# summary(results_Set1_LLandPT)
#plot(results_Set1_LLandPT)
round(results_Set1_LLandPT$Final_ratio_GORICA_weights, 2)
```

```
     vs. H1.1     vs. H1.2 vs. Hu
H1.1        1 4.953573e+34 285.37
H1.2        0 1.000000e+00   0.00
Hu          0 1.735855e+32   1.00
```

Once more, one can see that the results do not change when the function arguments changed (as it denotes the same input). Any of these methods is suitable for aggregating evidence over multiple studies.

As in the previous section, it is not necessary to specify the hypotheses in `evSyn`. However, to avoid confusion, `hypo_names` is used for a better labeling in the output.

Specifying sets of hypotheses in `evSyn`

In the examples above, the set of hypotheses (reflecting the central theories) are the same for all studies. This does not need to be the case. Note: The number of hypotheses per study-specific set is always the same, but the specification of the (study-specific) hypotheses does not need to be the same for all studies.

If the specification of hypotheses differ per study, then one needs to specify the sets of hypotheses per study.

Below, one can find examples of code to specify sets of hypotheses in the `evSyn` function in the case of having (for ease) 3 primary studies; for both the cases where there are two or 1 central theories (excluding the failsafe).
Below, you can find the following three blocks of examples:
- for having one set of 2 hypotheses for all 3 studies;
- for having 3 study-specific sets (containing 2 study-specific hypotheses);
including some incorrect specification;
- for having 3 study-specific sets of 1 hypothesis.

```
# If you have one set of hypotheses for all studies,
# then you should specify one list.
# You can use one of the following:
#
Hypo_studies <- list(H1, H2)
# names central theories: H1 and H2
#
Hypo_studies <- list(Hpos, Hneg)
# names central theories: H1 and H2
#
Hypo_studies <- list(Hpos = H1, Hneg = H2)
# names central theories: Hpos and Hneg
#
# Note that, in each study,
# the parameter estimate names should be the same.
```

```
# If you have study-specific sets for your 3 studies,
# then you should specify 3 lists in one list.
#
# Note that the order of hypotheses in the set matters,
# as well as the labeling of the central theories.
# The labeling of the hypotheses sets does not matter.
#
# You can use one of the following:
#
Hypo_studies <- list(list(H1_St1, H2_St1),
                     list(H1_St2, H2_St2),
                     list(H1_St3, H2_St3))
# names central theories: H1 and H2
#
Hypo_studies <- list(SetStudy1 = list(H1_St1, H2_St1),
                     SetStudy2 = list(H1_St2, H2_St2),
                     SetStudy3 = list(H1_St3, H2_St3))
# names central theories: H1 and H2
# Note: the names 'SetStudy.' will not be used.
#
Hypo_studies <- list(list(Hpos = H1_St1, Hneg = H2_St1),
                     list(Hpos = H1_St2, Hneg = H2_St2),
                     list(Hpos = H1_St3, Hneg = H2_St3))
# names central theories: Hpos and Hneg
#
Hypo_studies <- list(SetStudy1 = list(Hpos = H1_St1, Hneg = H2_St1),
                     SetStudy2 = list(Hpos = H1_St2, Hneg = H2_St2),
                     SetStudy3 = list(Hpos = H1_St3, Hneg = H2_St3))
# names central theories: Hpos and Hneg
# Note: the names 'SetStudy.' will not be used.
```

```
# INcorrect ways of specifying:
Hypo_studies <- list(list(Hpos = H1_St1, Hneg = H2_St1),
                     list(H1   = H1_St2, H2   = H2_St2),
                     list(Ha   = H1_St3, Hb   = H2_St3))
# Note: Each central theory should have the same labeling.
#
Hypo_studies <- list(list(Hpos = H1_St1, Hneg = H2_St1),
                     list(Hpos = H1_St2, Hneg = H2_St2),
                     list(Hneg = H1_St3, Hpos = H2_St3))
# Note: The order of hypotheses in the sets should be the same.
```

```
# Just to be sure,
# if there would be one central theory for all 3 studies:
#
Hypo_studies <- list(list(H1_St1),
                     list(H1_St2),
                     list(H1_St3))
# names central theories: H1
#
Hypo_studies <- list(list(Hpos = H1_St1),
                     list(Hpos = H1_St2)                    ,
                     list(Hpos = H1_St3))
# names central theories: Hpos
```