Effect modification in RRMM: NMA Simuation

Chris Rose, Norwegian Institute of Public health

2022-05-10

This report was generated on Tue May 10 12:52:40 2022 using git revision 8812683.

## Introduction

This document describes a simulation study to investigate the likely effect of the estimated effect modification on NMA estimates.

The following packages are used, and we set the random seed to ensure reproducibility (at least under the environment used, which is described in full in the Appendix).

library(tidyverse)  
#> ── Attaching packages ────────────────────────────────────────────────────────────────────────────── tidyverse 1.2.1 ──  
#> ✔ ggplot2 3.1.0 ✔ purrr 0.3.1   
#> ✔ tibble 3.1.6 ✔ dplyr 0.8.0.1  
#> ✔ tidyr 0.8.3 ✔ stringr 1.4.0   
#> ✔ readr 1.3.1 ✔ forcats 0.4.0  
#> ── Conflicts ───────────────────────────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
#> ✖ dplyr::filter() masks stats::filter()  
#> ✖ dplyr::lag() masks stats::lag()  
library(netmeta)  
#> Loading required package: meta  
#> Loading 'meta' package (version 4.18-0).  
#> Type 'help(meta)' for a brief overview.  
#> Loading 'netmeta' package (version 1.3-0).  
#> Type 'help("netmeta-package")' for a brief overview.  
library(knitr)  
import::from(magrittr, "%>%")  
set.seed(1234)

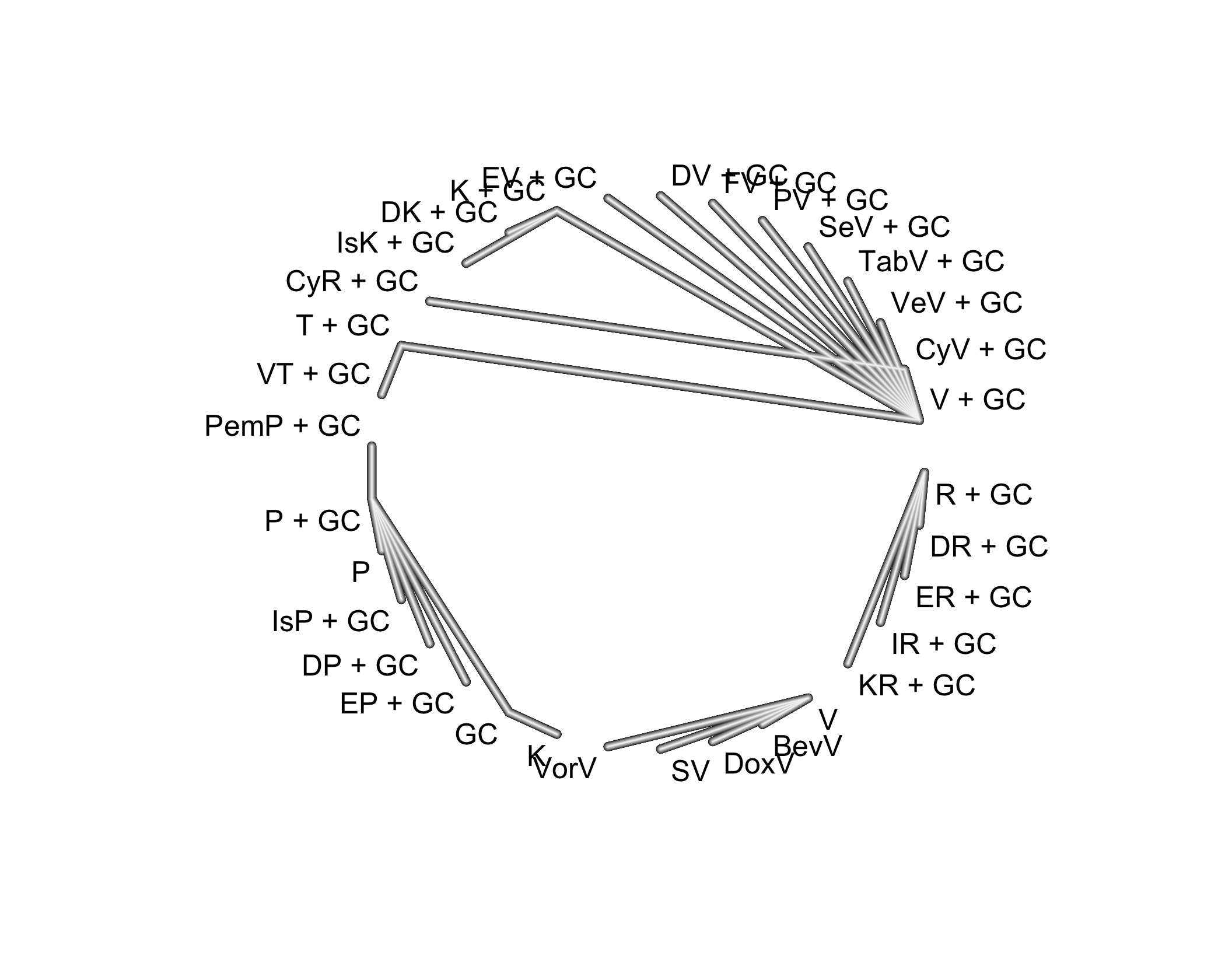
## Network topology and treatment components

We consider a fixed network topology, specifically the one for HR for PFS. Note that the network is disconnected (see figure below), and we address this using component NMA.

net\_data <- read.csv("pfs.csv")  
  
# Create a wrapper to perform a component NMA that suppresses warnings that  
# occur due to the way that the strings are formatted in the data.  
cnma <- function(x) suppressWarnings(  
 x %>% netmeta::discomb(TE, seTE, treat1, treat2, studlab, data = .,  
 reference.group = "R + GC",  
 sep.comps = "+",  
 comb.fixed = FALSE, comb.random = TRUE))  
  
# Make a disconnected component NMA object that can be used as a template  
# in subsequent code.  
net <- net\_data %>% cnma()

The following figure shows the network topology:

suppressWarnings(  
 net\_data %>%   
 discomb(TE, seTE, treat1, treat2, studlab, data = .) %>%   
 netgraph()  
)



## Simulating networks of evidence

To simulate a network of evidence, we need to simulate log HRs (effect sizes). We do this by sampling component-specific effects (cf. relative treatment effects) and then using these to compute relative treatment effects as contrasts, using the matrix C, as defined in the paper on CNMA by Rücker et al. The distribution used is based on the estimated component effects in the CNMA of the PFS data. This gives log HRs with similar distribution to that in the PFS data.

sim\_es <- function(x = net) { # Example component NMA object.  
 component\_effects <- rnorm(length(x$comps),   
 mean = mean(x$Comp.fixed),  
 sd = sd(x$Comp.fixed))  
 es <- as.matrix(dist(x$C.matrix %\*% component\_effects))  
 es[lower.tri(es)] <- -es[lower.tri(es)]  
 es  
}

To simulate a network of evidence, we also need to define the variance of the random effects (i.e., heterogeneity), and we use an assumed value based on the NMAs performed for the HTA.

os\_s2re <- 0.115 # Estimated random effects variance.

We also need to parameterize the distribution of effect modification. The estimates used are from the meta-analysis of relative hazard ratio (RHR) for refractory status for PFS, which corresponds to the worst-case scenario (i.e., largest effect modification). The estimated values (a point estimate and a 95% CI) are on a ratio scale, so we take logs to define the mean and standard error on the log HR scale. Similarly, we define the worst-case RHR for LOT (also from PFS).

pfs\_em <- list(mean = log(1.31), se = (log(1.47)-log(1.16))/(2\*1.96))  
lot\_em <- list(mean = log(1.19), se = (log(1.29)-log(1.09))/(2\*1.96))

The function simulate\_data() simulates a network of evidence that has the same topology as, and effect sizes and standard errors that are similar in distribution to, the PFS network. The returned data frame has two columns for effect size, cTE and eTE, for control and experimental, respectively. The control effect sizes are not subject to effect modification, while the experimental effect sizes are. The variable nl specifies the number of levels of the effect modification variable, which is 2 for refractory status, and can be larger for LOT. Because RHR is defined relative to the previous level, effect modification “compounds” with each subsequent level, which is equivalent to being cumulative on the log scale. We assume that there is no effect modification for the first level of the effect modifier variable, and then that modification applies and compounds with each subsequent level.

simulate\_data <- function(es = sim\_es(), # True effect sizes.  
 x = net, # Example component NMA object.  
 em = pfs\_em, # Effect modification parameters.  
 nl = 2, # Levels of EM variable.  
 s2re = os\_s2re, # Random effect \*variance\*.  
 unif\_dir = FALSE # Uniform direction?  
 ) {  
 n <- length(x$trts)  
  
 # Allocate a data frame using the template.  
 sim\_data <- data.frame(studlab = x$studlab,  
 treat1 = x$treat1, treat2 = x$treat2,  
 cTE = NA, eTE = NA, seTE = NA)  
   
 # Model SE as lognormal, based on the actual data.  
 sim\_data$seTE <- rlnorm(length(x$seTE),  
 mean = mean(log(x$seTE)),   
 sd = sd(log(x$seTE)))  
   
 # Dir. of effect modification is assumed constant within comparison.  
 em\_dir <- matrix(sign(rnorm(n \* n)), nrow = n, ncol = n)  
 em\_dir[lower.tri(em\_dir)] <- -em\_dir[lower.tri(em\_dir)]  
 rownames(em\_dir) <- rownames(es)  
 colnames(em\_dir) <- colnames(es)  
   
 # Simulate control and experimental effect sizes reported by  
 # the simulated studies.  
 for (i in 1:nrow(sim\_data)) {  
 t1 <- sim\_data$treat1[i] %>% as.character()  
 t2 <- sim\_data$treat2[i] %>% as.character()  
   
 # Simulate control data.  
 se <- rnorm(1, sd = sim\_data$seTE[i]) # Sampling error.  
 re <- rnorm(1, sd = sqrt(s2re)) # Random effect.  
 sim\_data$cTE[i] <- es[t1, t2] + se + re  
   
 # Simulate experimental data.  
 props <- runif(nl, min = 0, max = 1)  
 props <- props / sum(props) # Cat. dist. over levels.  
 direction <- if (unif\_dir) 1 else em\_dir[t1, t2]  
 e <- c(0, rnorm(nl - 1, mean = em$mean \* direction, sd = em$se))  
 e <- cumsum(e) # EM accumulates over level (on the log scale).  
 e <- props %\*% e # Average effect modification.  
 sim\_data$eTE[i] <- sim\_data$cTE[i] + e  
 }  
 sim\_data  
}

## Comparing estimates

We want to know whether effect modification likely leads to different conclusions. To assess this, we estimate treatment effects under no effect modification, and under modification, and then test that corresponding estimates are equal. Testing is performed using two-sided Z-tests, given the means and SEs of the estimates. We summarize the tests by counting how many are “significant” (p < 0.05). We restrict testing to one triangle of the matrix of estimates, to prevent double-counting.

compare\_estimates <- function(s) { # s given by simulate\_data().  
 # Fit CNMA models to the control and experimental data.  
 suppressWarnings(  
 net\_c <- s %>% discomb(cTE, seTE, treat1, treat2, studlab,  
 reference.group = "R + GC", data = .,)  
 )  
 suppressWarnings(  
 net\_e <- s %>% discomb(eTE, seTE, treat1, treat2, studlab,  
 reference.group = "R + GC", data = .,)  
 )  
 # Z-test of equality of estimates.  
 est\_c <- net\_c$TE.random  
 se\_c <- net\_c$seTE.random  
 est\_e <- net\_e$TE.random  
 se\_e <- net\_e$seTE.random  
 p <- 1 - pnorm(abs(est\_c - est\_e) / sqrt((se\_c^2) + (se\_e^2)))  
 # Return number of elements of upper triangle with "significant" p-values,  
 # and number of elements in the upper triangle.  
 sig <- p[upper.tri(p)] < 0.05  
 list(sig = sum(sig), n = length(sig))  
}

# Simulation studies

The function simulation\_study() runs a simulation study under specified conditions (combinations of effect modification and random effects).

simulation\_study <- function(n = 1000,  
 x = net,  
 em = pfs\_em, # Effect mod. parameters.  
 nl = 2, # N. levels of EM var.  
 s2re = os\_s2re, # Random effect \*variance\*.  
 unif\_dir = FALSE, # Uniform direction?  
 conditions = ""  
 ) {  
 f <- function() {  
 sim\_es(x = net) %>%   
 simulate\_data(es = ., x = net,  
 em = em,  
 nl = nl,  
 s2re = s2re,  
 unif\_dir = unif\_dir) %>%   
 compare\_estimates()  
 }  
 results <- replicate(n, f(), simplify = FALSE)  
   
 # Collect the results.  
 sig <- function(x) x$sig  
 n <- function(x) x$n  
 sig <- results %>% map(sig) %>% unlist() %>% sum()  
 n <- results %>% map(n) %>% unlist() %>% sum()  
   
 list(conditions = conditions, sig = sig, n = n)  
}

The full experiment is a set of simulation studies, including sensitivity analyses.

run\_experiment <- function(n = 1000) {  
 list(  
 # Non-uniform direction of effect modification for refractory status.  
 simulation\_study(n,  
 conditions = "Refr: EM [✓] Het. [✓]"),  
 simulation\_study(n, em = list(mean = 0, se = 0),  
 conditions = "Refr: EM [×] Het. [✓]"),  
   
 # Sanity check that absent both EM and heterogeneity, there  
 # are no differences. Note that no heterogeneity but  
 # effect modification is an unrealistic scenario, so we  
 # omit that.  
 simulation\_study(n, s2re = 0, em = list(mean = 0, se = 0),  
 conditions = "Refr: EM [×] Het. [×]"),  
   
 # Sensitivity analysis: uniform direction of effect  
 # modification for refractory status.  
 simulation\_study(n,  
 unif\_dir = TRUE,  
 conditions = "Refr: EM [✓] Het. [✓] (Unif.)"),  
 simulation\_study(n, em = list(mean = 0, se = 0),  
 unif\_dir = TRUE,  
 conditions = "Refr: EM [×] Het. [✓] (Unif.)"),  
   
 # Non-uniform direction of effect modification for LOT.  
 # Note that we don't run the simulations with heterogeneity  
 # alone, because this is identical to the corresponding simulations  
 # above.  
 simulation\_study(n, em = lot\_em, nl = 4,  
 conditions = "LOT: EM [✓] Het. [✓]"),  
   
 # Sensitivity analysis: uniform direction of effect modification  
 # for LOT.  
 simulation\_study(n, em = lot\_em, nl = 4,  
 unif\_dir = TRUE,  
 conditions = "LOT: EM [✓] Het. [✓] (Unif.)")  
 ) %>% map(as\_tibble) %>% bind\_rows()  
}  
  
run\_experiment(n = 1000) %>% kable()

| conditions | sig | n |
| --- | --- | --- |
| Refr: EM [✓] Het. [✓] | 1841 | 528000 |
| Refr: EM [×] Het. [✓] | 0 | 528000 |
| Refr: EM [×] Het. [×] | 0 | 528000 |
| Refr: EM [✓] Het. [✓] (Unif.) | 821 | 528000 |
| Refr: EM [×] Het. [✓] (Unif.) | 0 | 528000 |
| LOT: EM [✓] Het. [✓] | 23643 | 528000 |
| LOT: EM [✓] Het. [✓] (Unif.) | 12987 | 528000 |

## Appendix: System, R version, and package details

The following shows a summary of the system, R version, and packages used to perform the simulation.

#> R version 3.5.2 (2018-12-20)  
#> Platform: x86\_64-apple-darwin15.6.0 (64-bit)  
#> Running under: OS X 12.2.1  
#>   
#> Matrix products: default  
#> LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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  
#>   
#> attached base packages:  
#> [1] stats graphics grDevices utils datasets methods base   
#>   
#> other attached packages:  
#> [1] knitr\_1.33 netmeta\_1.3-0 meta\_4.18-0 forcats\_0.4.0 stringr\_1.4.0 dplyr\_0.8.0.1 purrr\_0.3.1   
#> [8] readr\_1.3.1 tidyr\_0.8.3 tibble\_3.1.6 ggplot2\_3.1.0 tidyverse\_1.2.1  
#>   
#> loaded via a namespace (and not attached):  
#> [1] Rcpp\_1.0.0 lubridate\_1.7.4 lattice\_0.20-38 assertthat\_0.2.0 digest\_0.6.18 utf8\_1.1.4   
#> [7] R6\_2.4.0 cellranger\_1.1.0 plyr\_1.8.4 backports\_1.1.3 magic\_1.5-9 evaluate\_0.13   
#> [13] highr\_0.7 httr\_1.4.0 pillar\_1.6.4 rlang\_0.4.12 lazyeval\_0.2.1 readxl\_1.3.0   
#> [19] rstudioapi\_0.9.0 minqa\_1.2.4 nloptr\_1.2.1 Matrix\_1.2-15 rmarkdown\_2.10 splines\_3.5.2   
#> [25] lme4\_1.1-21 munsell\_0.5.0 broom\_0.5.1 compiler\_3.5.2 modelr\_0.1.4 xfun\_0.25   
#> [31] pkgconfig\_2.0.2 htmltools\_0.3.6 tidyselect\_0.2.5 fansi\_0.4.0 withr\_2.1.2 crayon\_1.3.4   
#> [37] MASS\_7.3-51.1 grid\_3.5.2 nlme\_3.1-137 jsonlite\_1.6 gtable\_0.2.0 lifecycle\_1.0.1   
#> [43] magrittr\_1.5 metafor\_2.1-0 scales\_1.0.0 cli\_1.0.1 stringi\_1.3.1 xml2\_1.2.0   
#> [49] ellipsis\_0.3.2 generics\_0.0.2 vctrs\_0.3.8 boot\_1.3-20 tools\_3.5.2 import\_1.1.0   
#> [55] CompQuadForm\_1.4.3 glue\_1.3.0 hms\_0.4.2 abind\_1.4-5 yaml\_2.2.0 colorspace\_1.4-0   
#> [61] rvest\_0.3.2 haven\_2.1.0