# Multivariate Meta-Analysis for Longevity and Reproduction

Daniel Noble & Fay Frost

```
# VCV matrix. Let's set up the multivariate meta-analysis model. We first need to create the VCV sampli

# V <- metafor::vcalc(vi=v, cluster = trial, subgroup = Experiment.code, type = outcome, data = data_lo

# First, lets capture the outcome covariance
   V_1 <- make_VCV_matrix(data = data_long_final, cluster = "trial",  V = "v", rho = 0.5)

# Using this matrix, we can now capture the shared control. here, we now just feed in the V matrix we j
   V <- metaAidR::make_VCV_matrix(data = data_long_final, matrix = V_1, cluster = "shared_control",  V =

# Export V matrix for checking
  write.csv(V, here("Output", "tables", "V.csv"))

# Check that this is set up correctly. Note that there are warnings about non-positive definite matrix.
  V[1:15, 1:15]
```

```
##             1          2          3          4          5          6          7
## 1  0.07246655 0.03859268 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2  0.03859268 0.08221142 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 3  0.00000000 0.00000000 0.25297242 0.09279856 0.10601670 0.00000000 0.09661859
## 4  0.00000000 0.00000000 0.09279856 0.13616618 0.00000000 0.06744434 0.00000000
## 5  0.00000000 0.00000000 0.10601670 0.00000000 0.17771963 0.07705105 0.08098261
## 6  0.00000000 0.00000000 0.00000000 0.06744434 0.07705105 0.13362317 0.00000000
## 7  0.00000000 0.00000000 0.09661859 0.00000000 0.08098261 0.00000000 0.14760742
## 8  0.00000000 0.00000000 0.00000000 0.06740419 0.00000000 0.06677181 0.07017887
## 9  0.00000000 0.00000000 0.09186886 0.00000000 0.07700155 0.00000000 0.07017555
## 10 0.00000000 0.00000000 0.00000000 0.06737150 0.00000000 0.06673943 0.00000000
## 11 0.00000000 0.00000000 0.09211847 0.00000000 0.07721076 0.00000000 0.07036622
## 12 0.00000000 0.00000000 0.00000000 0.06778847 0.00000000 0.06715248 0.00000000
## 13 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 14 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 15 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
##             8          9         10         11         12         13         14
## 1  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 2  0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
## 3  0.00000000 0.09186886 0.00000000 0.09211847 0.00000000 0.00000000 0.00000000
## 4  0.06740419 0.00000000 0.06737150 0.00000000 0.06778847 0.00000000 0.00000000
## 5  0.00000000 0.07700155 0.00000000 0.07721076 0.00000000 0.00000000 0.00000000
## 6  0.06677181 0.00000000 0.06673943 0.00000000 0.06715248 0.00000000 0.00000000
## 7  0.07017887 0.07017555 0.00000000 0.07036622 0.00000000 0.00000000 0.00000000
## 8  0.13346411 0.00000000 0.06669970 0.00000000 0.06711250 0.00000000 0.00000000
## 9  0.00000000 0.13345151 0.06669655 0.06690705 0.00000000 0.00000000 0.00000000
## 10 0.06669970 0.06669655 0.13333470 0.00000000 0.06707996 0.00000000 0.00000000
## 11 0.00000000 0.06690705 0.00000000 0.13417766 0.06729167 0.00000000 0.00000000
## 12 0.06711250 0.00000000 0.06707996 0.06729167 0.13499025 0.00000000 0.00000000
## 13 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.13978248 0.07293282
## 14 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.07293282 0.15221354
## 15 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.06826003 0.00000000
##            15
## 1  0.00000000
## 2  0.00000000
## 3  0.00000000
## 4  0.00000000
## 5  0.00000000
## 6  0.00000000
## 7  0.00000000
## 8  0.00000000
## 9  0.00000000
## 10 0.00000000
## 11 0.00000000
## 12 0.00000000
## 13 0.06826003
## 14 0.00000000
## 15 0.13333379
```

```
  #corrplot(cov2cor(V)) # Takes a while so no need to run all the time

# Check of PD
  corpcor::is.positive.definite(V) # FALSE
```

```
## [1] FALSE
```

```
# Can bend it to make it PD
  V <- Matrix::nearPD(V)$mat

# Check of PD
  corpcor::is.positive.definite(V) # TRUE
```

```
## [1] TRUE
```

```
 # Multivariate model. Using the VCV matrix we can set up the model. We'll keep it simple for now

mv_mlma <- rma.mv(es ~ outcome - 1, V = V,
               random = list(~outcome - 1 | trial,
                             ~outcome - 1 | Species.latin),
               struc = "UN", data = data_long_final, test = "t", dfs = "contain")


## Lets try a model with the random effects structure from the univariate case. Not the full structure

# Phylogeny. Use reproduction because it should match with longevity in terms of species

 tree1 <- read.nexus(here("phylogeny", "all_reproduction_excHUM251_tree.nex"))
 tree_grafen = compute.brlen(tree1, method="Grafen", power=1)

classes <- read.csv(here("Data", "Species_classifications.CSV")) ## read in species classifications fro

data_long_final$Species.latin[which(data_long_final$Species.latin == "Marasmia exigua")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Matsumuratettix hieroglyphicus")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Mythimna roseilinea")]
```

```r
data_long_final$Species.latin[which(data_long_final$Species.latin == "Apis craccivora")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Cryptoleamus montrouzieri")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Asplanchna brightwelli")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Brennandania lambi")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Amblyseius alstoniae")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Siphoninus phyllyreae")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Proprioseiopsis asetus")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Parabemisia myrica")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Cirrospilus sp. near lyncus")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Anagyrus sp. nov. nr. sinope" )]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Monochamus leuconotus")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Ropalosiphum maidis")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Artemia fransiscana")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Blathyplectes curculionis")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "Menochilus sexmaculatus")]
data_long_final$Species.latin[which(data_long_final$Species.latin == "unknown (Tominic)")]

### specify classifications from map
data_long_final$Class <- classes$class[match(data_long_final$Species.latin, classes$species_latin)]

 ############################################## End of Fay's amendments ################################

# Now we need to prune this tree to the species in this long data
       tree_checks <- tree_checks(data_long_final, tree1, dataCol = "Species.latin", type = "check") # 

# Prune tree
        tree2 <- tree_checks(data_long_final, tree1, dataCol = "Species.latin", type = "prune")

# Now, use Fay's code to create R matrix. Not yet working because we need to sort out the species in th
   tree_grafen <- compute.brlen(tree2, method="Grafen", power=1)
 phylo_matrix <- vcv(tree_grafen, cor=TRUE, model="Brownian")

# Now fit the model with phylogeny. That seems to work just fine!
mv_mlma2 <- rma.mv(es ~ outcome - 1, V = V,
               random = list(~outcome - 1 | trial,
                             ~outcome - 1 | Species.latin),
               R= list(Species.latin = phylo_matrix),
               struc = "UN", data = data_long_final, test = "t", dfs = "contain")
```

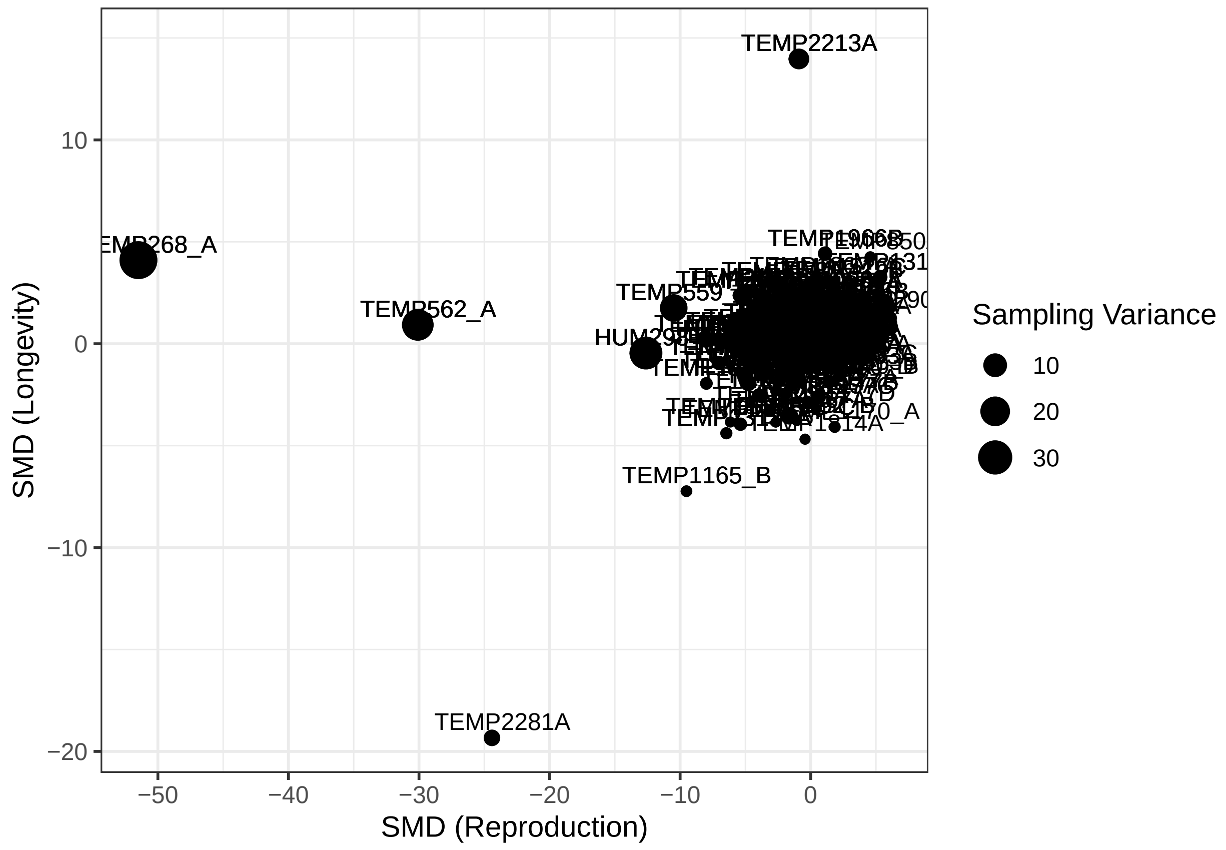
```r
# We'll now run the multivariate model with non-linear terms as was done in the univariate cases.
rerun2=FALSE
if(rerun2){
  mv_mlma_4 <- rma.mv(es ~ -1 + outcome + outcome:poly(c_treattemp, degree=3, raw=TRUE), V = V,
               random = list(~outcome - 1 | trial,        # This would be equivalent to an obs level ra
                             ~outcome - 1 | Paper.code),  # This should estimate a study level random e
               struc = "UN", data = data_long_final, test = "t", dfs = "contain")
  saveRDS(mv_mlma_4, here("output", "models", "mv_mlma_4.rds"))
} else {
  mv_mlma_4 <- readRDS(here("output", "models", "mv_mlma_4.rds"))
}

# We can get confidence intervals by profiling the liklihood
  cis <- confint(mv_mlma_4, rho = 1)
  write.csv(cis, here("output", "tables", "mv_mlma_4_cis.csv"))
```

```
## Warning: Extra arguments ('optional', 'stringsAsFactors') disregarded.
```

```
# Lets explore the among study correlation.
data  %>%  group_by(Experiment.code)  %>% summarise(es_repro = mean(es_reproduction), es_long = mean(es
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `summarise()` has grouped output by 'Experiment.code'. You can override using
## the `.groups` argument.
```


TEMP2213A
EMP268_A
TEMP562_A
TEMP559
HUM29
TEMP1966B
TEMP1165_B
TEMP2281A
SMD (Longevity)
SMD (Reproduction)
Sampling Variance