A space travel into the Multiverse

Cognitive Science Arena

Giulia Calignano Filippo Gambarota

Department of Developmental Psychology and Socialization, University of Padova

2025-05-02

Exploratory Multiverse Analysis (EMA)

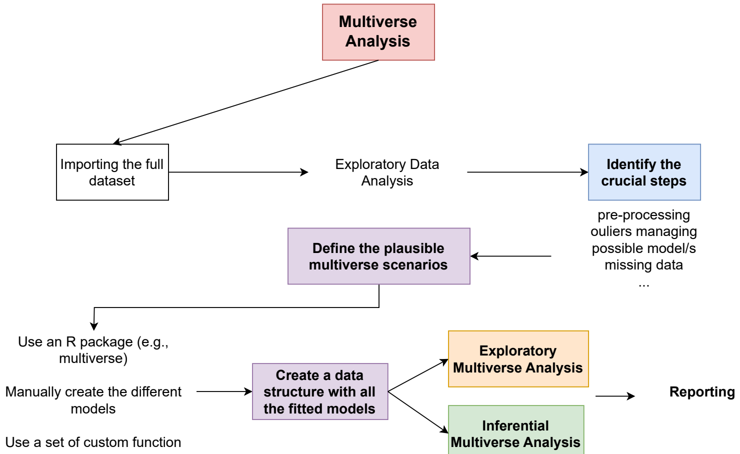
An example, Statistics and Math Anxiety

McCaughey et al. (2022) explored the relationship between self-efficacy anxiety sensitivity and perfectionism would be related to math/statistics anxiety controlling for gender, university program, and education level.

We used the dataset available at https://osf.io/nzhq6.

We are going to do crazy stuff with this dataset that are not related to the original paper and research question! :)

The big picture



Multiverse Analysis
Importing the full dataset
Exploratory Data Analysis
Identify the crucial steps
pre-processing
ouliers managing
possible model/s
missing data
...
Define the plausible multiverse scenarios
Use an R package (e.g., multiverse)
Manually create the different models
Use a set of custom function
Create a data structure with all the fitted models
Exploratory Multiverse Analysis
Inferential Multiverse Analysis
Reporting

Importing

We did a little bit of pre-processing. The `ms_anxiety.rds` file contains the cleaned version of the original dataset.

```
dat <- readRDS(here("data/ms_anxiety.rds"))
vars <- names(dat)
ys <- vars[grepl("^stat.anx|^math", vars)]
ys
##  [1] "stat.anx.tc"   "stat.anx.i"    "stat.anx.ah"   "stat.anx.ws"
##  [5] "stat.anx.fst"  "stat.anx.sc"   "math.anx"      "stat.anx.TOT"
##  [9] "stat.anx.ANX"  "stat.anx.FEEL"

xs <- vars[!vars %in% ys]
xs
## [1] "self.efficacy"   "asi"                "frost.com"
## [4] "frost.da"        "faculty"            "program.type"
## [7] "gender.category"
```

Exploring

Let's see the type of variables of the dataset:

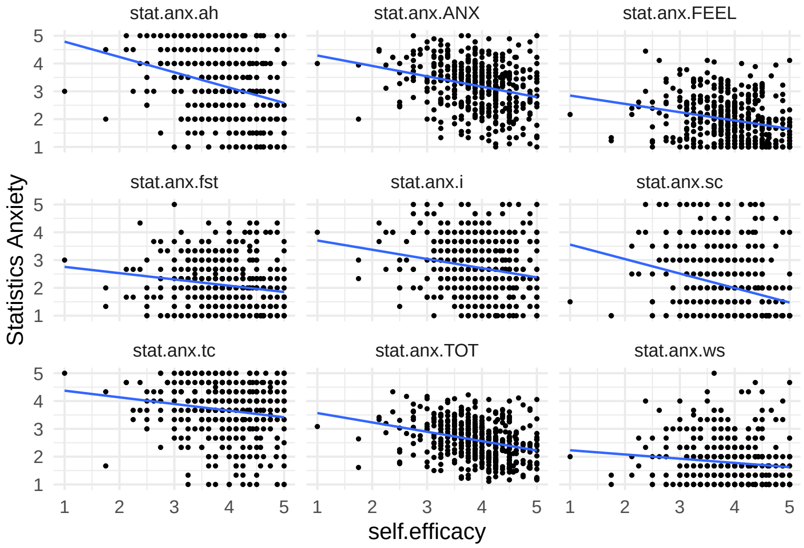
```
sapply(dat[ys], class)
##   stat.anx.tc    stat.anx.i   stat.anx.ah   stat.anx.ws  stat.anx.fst
##     "numeric"     "numeric"     "numeric"     "numeric"     "numeric"
##   stat.anx.sc      math.anx  stat.anx.TOT  stat.anx.ANX stat.anx.FEEL
##     "numeric"     "numeric"     "numeric"     "numeric"     "numeric"
sapply(dat[xs], class)
##   self.efficacy             asi       frost.com        frost.da
##       "numeric"       "numeric"       "numeric"       "numeric"
##         faculty    program.type gender.category
##        "factor"        "factor"        "factor"
```

Main research questions

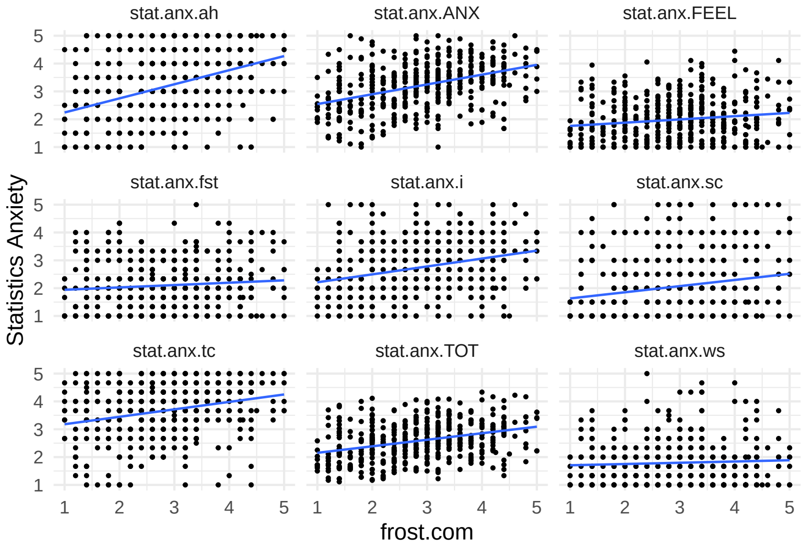
The main idea of the authors is predicting **math** and **statistics** anxiety with self-efficacy and perfectionism. In particular they pre-registered (see https://osf.io/b3g7s):

1. self-efficacy will be negatively related to math/statistics anxiety
2. anxiety sensitivity will be positively related to math/statistics anxiety.
3. self-critical perfectionism will be positively related to math/statistics anxiety.
4. the relationships described above will remain when statistically adjusting for gender, university program (arts vs. science) and student status (undergraduate vs. graduate).

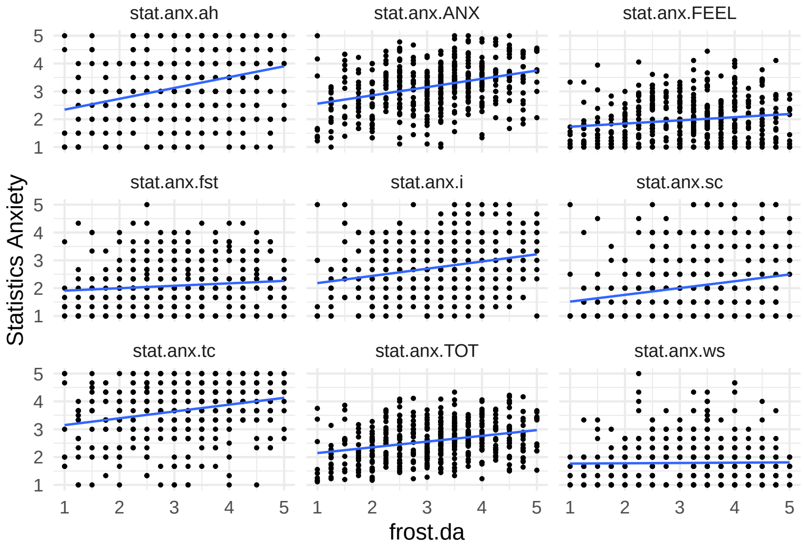
Exploring the relationships



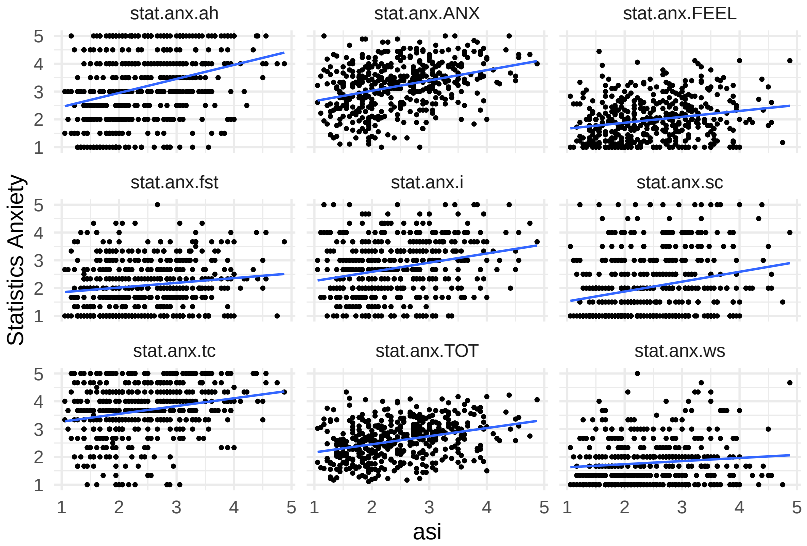
Exploring the relationships



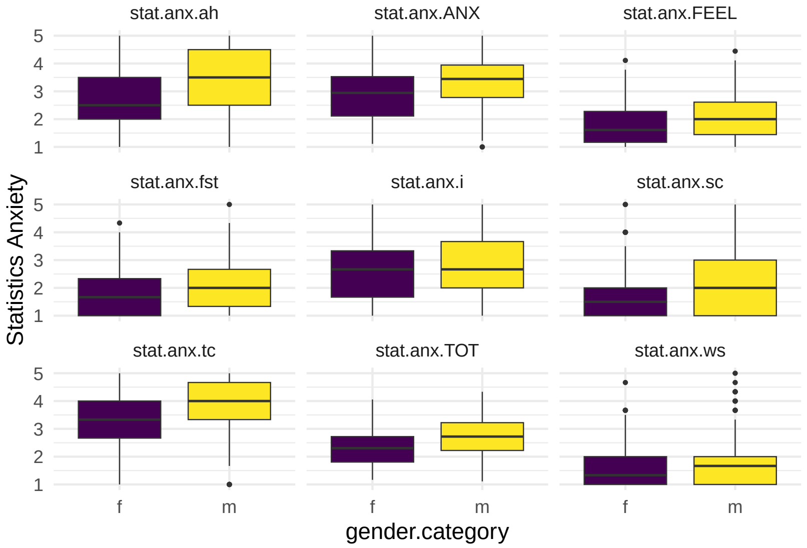
Exploring the relationships



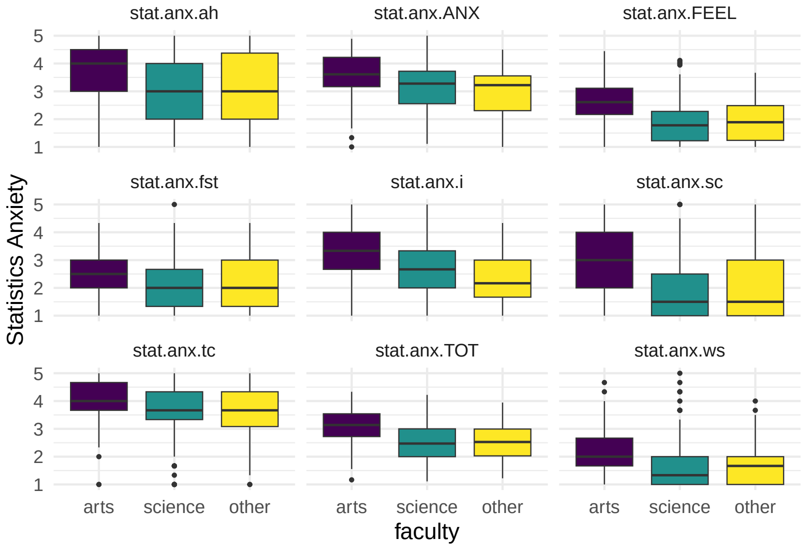
Exploring the relationships



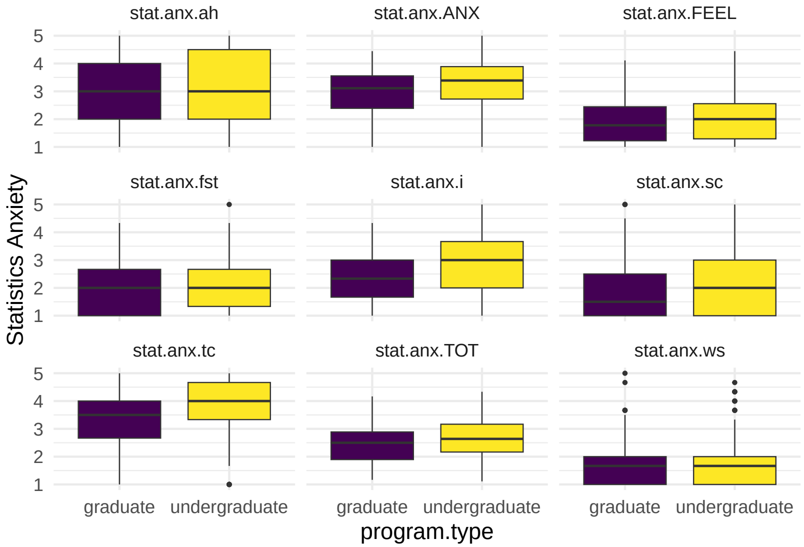
Exploring the relationships



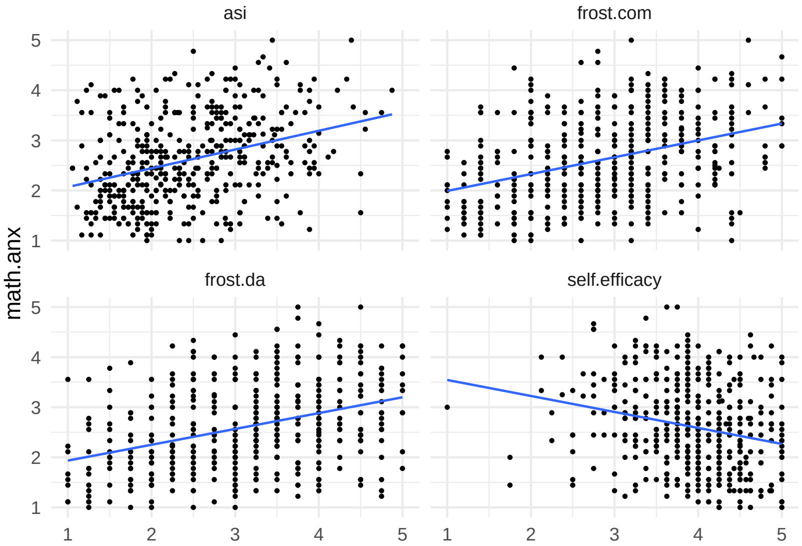
Exploring the relationships



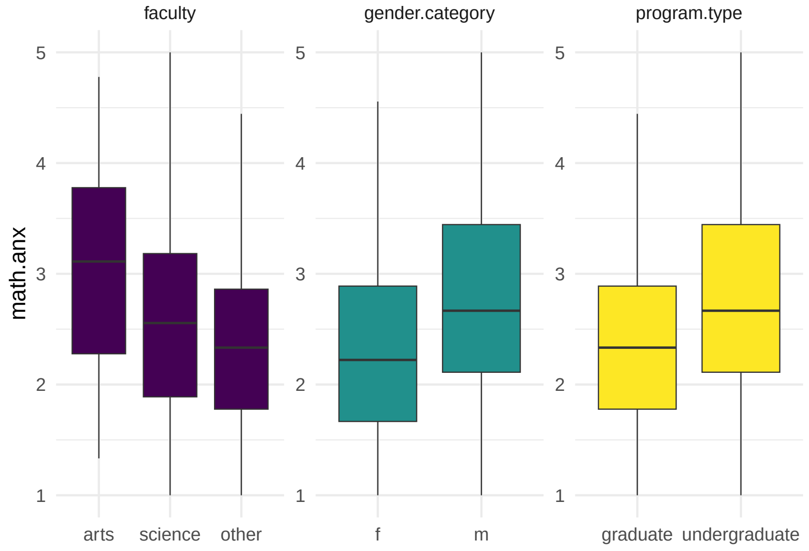
Exploring the relationships



Exploring the relationships



Exploring the relationships



Selecting a sub-sample

For the purpose of the example, we select a subsample of the dataset to increase the variability and simulate a more uncertain scenario with a lower sample size.

```r
set.seed(9386)
N <- 200
selected <- sample(1:nrow(dat), size = N, replace = FALSE)
dat <- dat[selected, ]
```

Data structure for specifications

When conducting a multiverse in R (or in whatever language) the data structure is very important.

- how to create and organize the different models?
- how to easily extract all the informations such as coefficients, standard errors, p-values, etc.
- ...

An R `list` is probably the best

A (named) `list` is flexible, easy to index and can be accesed by other functions to extract information and create other list.

A `list` in R can be easily transformed into a `data.frame` for other models, plots, tables, etc.

You can use the `*apply` family (`sapply`, `lapply`, etc.) to compute complex operations on lists.

An R `list` is probably the best

For example, assuming that I have some regression models within a named list:

```r
fit1 <- lm(math.anx ~ gender.category + asi, data = dat)
fit2 <- lm(math.anx ~ gender.category + asi + faculty, data = dat)
fit3 <- lm(math.anx ~ gender.category + faculty, data = dat)
fit4 <- lm(math.anx ~ gender.category + asi + program.type, data = dat)

# ... and other thousands of (plausible) models :)

mods <- list(fit1, fit2, fit3, fit4)
names(mods) <- paste0("mod", 1:length(mods))
```

An R `list` is probably the best

Then, I want to extract all the `asi` coefficients and put into a data.frame:

```
get_coef <- function(x, coef = NULL){
  x <- broom::tidy(x, conf.int = TRUE)
  if(!is.null(coef)){
    filter(x, term %in% coef)
  } else{
    x
  }
}

get_coef(mods$mod1, "asi")
```

```
# A tibble: 1 x 7
  term  estimate std.error statistic     p.value conf.low conf.high
  <chr>    <dbl>     <dbl>     <dbl>       <dbl>    <dbl>     <dbl>
1 asi      0.383    0.0748      5.11 0.000000745    0.235     0.530
```

An R `list` is probably the best

With `lapply` (or `purrr::map()`) and combining the results, you can easily create a nice dataframe with your coefficients:

```
lapply(mods, get_coef, "asi") |>
  dplyr::bind_rows(.id = "mod")
```

```
# A tibble: 3 x 8
  mod   term  estimate std.error statistic  p.value conf.low conf.high
  <chr> <chr>    <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 mod1  asi      0.383    0.0748      5.11 7.45e-7    0.235     0.530
2 mod2  asi      0.386    0.0741      5.21 4.88e-7    0.240     0.532
3 mod4  asi      0.410    0.0728      5.64 5.89e-8    0.267     0.554
```

Creating the specifications

There are multiple ways of creating the specifications in practice.
You can do it from scratch:

```r
mod1 <- lm(y ~ x1 + x2)
mod2 <- lm(y ~ log(x1) + log(x2))
mod3 <- lm(y ~ x1 + x2) # removing outliers

mods <- list(mod1 = mod1, mod2 = mod2, mod3 = mod3)
# ...
```

Creating the specifications

The `multiverse` R Package and the related paper (Götz et al., 2024) provides a very flexible and complex syntax to define different specifications.

Creating the specifications

For this example we can use some custom functions, in particular the `create_multi()` function. There are no wrong solutions if the results is correct.

```r
devtools::load_all()

slog <- function(x) {
  if(any(x == 0)){
    x <- x + 1
  }
  log(x)
}

multi <- create_multi(
  math.anx ~ asi + faculty + stat.anx.TOT, # full model formula
  focal = "asi", # focal predictor, never removed
  nfuns = c("slog"), # functions for the numeric variables
  data = dat
)
```

Creating the specifications

```
$X
       fun            x    type focal .id_fun .id_x
1 identity          asi numeric  TRUE       1     1
2 identity      faculty  factor FALSE       1     2
3 identity stat.anx.TOT numeric FALSE       1     3
5     slog stat.anx.TOT numeric FALSE       2     3
                 call
1                 asi
2             faculty
3        stat.anx.TOT
5 slog(stat.anx.TOT)

$calls
[1] "~ asi"
[2] "~ asi + faculty"
[3] "~ asi + stat.anx.TOT"
[4] "~ asi + slog(stat.anx.TOT)"
[5] "~ asi + faculty + stat.anx.TOT"
[6] "~ asi + faculty + slog(stat.anx.TOT)"
```

Creating the specifications

Whatever the method we used, we need:

- a list of models
- a way to easily extract the coefficients or other quantities
- a way to extract a summary of the specifications i.e. if a variable is included or not, the type of tranformation, etc.

Pay attention with interactions!

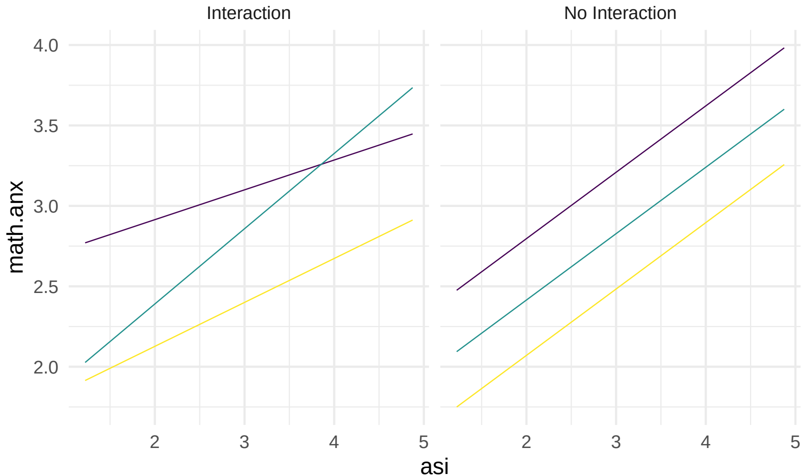
When an interaction is included in the model, the interpretation of the model coefficients completely change, especially if the interaction is consistent. You cannot compare a focal coefficients directly for models with and without interactions.

Let's assume that `asi` is the focal coefficient and we include in the multiverse these two models:

```r
fit_int <- lm(math.anx ~ faculty + asi + faculty:asi, data = dat)
fit_no_int <- lm(math.anx ~ faculty + asi, data = dat)
```

Pay attention with interactions!

The `asi` effect in one case is the overall effect (i.e., main effect) controlling for `faculty`. In the other case is the `asi` effect of the reference value.



Pay attention with interactions!

One should adjust the contrasts coding of factors and/or the centering of numeric variables.

```r
# sum to zero contrasts i.e. estimating the main effect of asi
update(fit_int, contrasts = list(faculty = contr.sum(3)))
```

```

Call:
lm(formula = math.anx ~ faculty + asi + faculty:asi, data = dat,
    contrasts = list(faculty = contr.sum(3)))

Coefficients:
 (Intercept)      faculty1      faculty2           asi  faculty1:asi
      1.8597        0.6839       -0.4048        0.3087       -0.1234
faculty2:asi
      0.1590
```

```r
# with emmeans
emmeans::emtrends(fit_int, ~1, var = "asi")
```

```
 1       asi.trend   SE  df lower.CL upper.CL
 overall     0.309 0.11 194   0.0925    0.525

Results are averaged over the levels of: faculty
Confidence level used: 0.95
```

Why exploring is important?

A multiverse analysis increase the complexity of the data analysis.
There is no longer a single dataset and result to discuss.

Let's create some scenarios :)

Firstly, we use variable transformations directly within the model formula. In this way it is easier to extract the conditions. Thus we define some wrappers:

```r
# safe version of log() with 0 variables
slog <- function(x){
  if(any(x == 0)){
    x <- x + 1
  }
  log(x)
}

# function factories, see https://adv-r.hadley.nz/function-factories.html
polyN <- function(degree = 1){
  function(x) poly(x, degree = degree)
}

poly2 <- polyN(2)
poly3 <- polyN(3)
```

Let's create some scenarios :)

More wrappers:

```r
cutN <- function(breaks){
  function(x){
    cut(x, breaks = breaks)
  }
}

cut2 <- cutN(2)
cut4 <- cutN(4)
```

Let’s create some scenarios :)

Then we can identify some univariate/multivariate outliers or some observations that we may consider removing for some reasons.

Let's create some scenarios :)

```r
focal <- "self.efficacy"

multi <- create_multi(
  math.anx ~ self.efficacy + faculty + asi + gender.category +
      program.type + frost.da,
  focal = focal,
  nfuns = c("slog", "cut2", "poly2"),
  data = dat
)
```

Let's fit the models

```r
# faster than before
get_coef <- function(x, coef = NULL){
  xs <- data.frame(summary(x)$coefficients)
  if(!is.null(coef)){
    xs <- xs[coef, ]
  }
  xs$param <- rownames(xs)
  return(xs)
}

fitl <- vector(mode = "list", length = length(multi$calls))

for(i in 1:length(multi$calls)){
  form <- paste0("math.anx", multi$calls[i])
  fitl[[i]] <- glm(form, family = gaussian(link = "identity"), data = dat)
}

resl <- lapply(fitl, get_coef, focal)
res <- bind_rows(resl, .id = "mod")
rownames(res) <- NULL
names(res) <- c("mod", "b", "se", "t", "p", "param")
```

Let's fit the models

Now we have a dataframe with all the model coefficients across the specifications. We can start our multiverse!

```r
head(res)
```

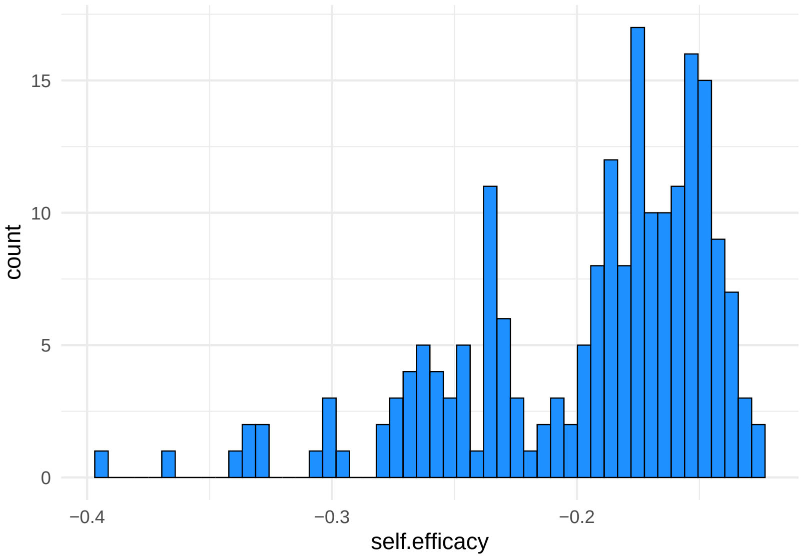
```
  mod          b         se         t            p         param
1   1 -0.3955213 0.10133460 -3.903122 0.0001301284 self.efficacy
2   2 -0.3315150 0.10439789 -3.175495 0.0017375474 self.efficacy
3   3 -0.3302943 0.09599808 -3.440634 0.0007084586 self.efficacy
4   4 -0.3041665 0.10324396 -2.946095 0.0036060375 self.efficacy
5   5 -0.3335836 0.09874191 -3.378339 0.0008788681 self.efficacy
6   6 -0.2266493 0.10148550 -2.233317 0.0266529679 self.efficacy
```

Exploratory tools

- Marginal/Conditional effects
- Vibration of Effects
- Specification Curve

Marginal/Conditional effects

Overall distribution of regression parameters:



Marginal/Conditional effects

We can combine the model results with a table created by all conditions with the custom `get_info_models()` function:

```r
info <- get_info_models(multi)
head(info)
```

```
# A tibble: 6 x 7
    mod x_self.efficacy x_faculty x_asi x_gender.category
  <int> <chr>           <chr>     <chr> <chr>
1     1 self.efficacy   <NA>      <NA>  <NA>
2     2 self.efficacy   faculty   <NA>  <NA>
3     3 self.efficacy   <NA>      asi   <NA>
4     4 self.efficacy   <NA>      <NA>  gender.category
5     5 self.efficacy   <NA>      <NA>  <NA>
6     6 self.efficacy   <NA>      <NA>  <NA>
# i 2 more variables: x_program.type <chr>, x_frost.da <chr>
```

Marginal/Conditional effects

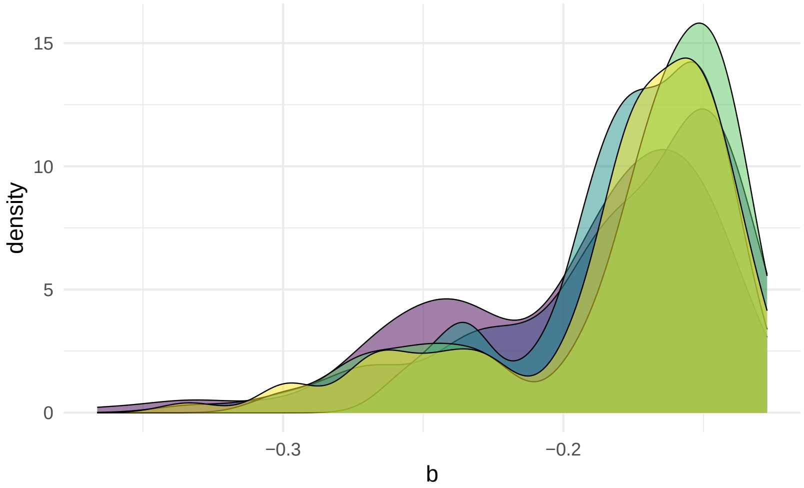
Then we can combine the `info` table with the coefficients table and we have all the important information.

```r
# same type
res$mod <- as.numeric(res$mod)
info$mod <- as.numeric(info$mod)
# merging the two tables
multi_res <- left_join(res, info, by = "mod")
head(multi_res)
```

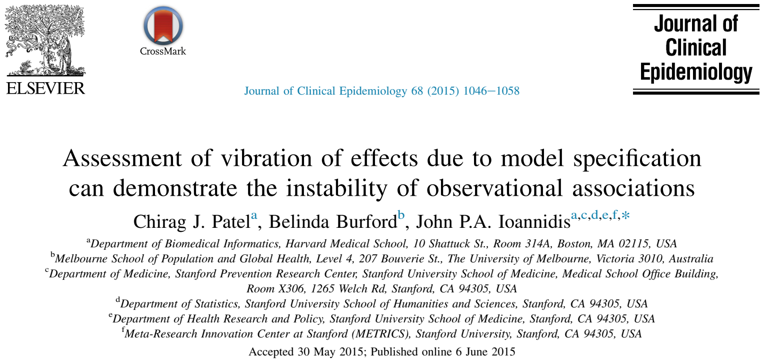
```
  mod          b         se         t            p         param
1   1 -0.3955213 0.10133460 -3.903122 0.0001301284 self.efficacy
2   2 -0.3315150 0.10439789 -3.175495 0.0017375474 self.efficacy
3   3 -0.3302943 0.09599808 -3.440634 0.0007084586 self.efficacy
4   4 -0.3041665 0.10324396 -2.946095 0.0036060375 self.efficacy
5   5 -0.3335836 0.09874191 -3.378339 0.0008788681 self.efficacy
6   6 -0.2266493 0.10148550 -2.233317 0.0266529679 self.efficacy
  x_self.efficacy x_faculty x_asi x_gender.category x_program.type
1   self.efficacy      <NA>  <NA>              <NA>           <NA>
2   self.efficacy   faculty  <NA>              <NA>           <NA>
3   self.efficacy      <NA>   asi              <NA>           <NA>
4   self.efficacy      <NA>  <NA>   gender.category           <NA>
5   self.efficacy      <NA>  <NA>              <NA>   program.type
6   self.efficacy      <NA>  <NA>              <NA>           <NA>
  x_frost.da
```

Marginal/Conditional effects

Finally we can plot also the distributions of parameters conditioned on the presence/absence of a particular other predictor:



Vibration of Effects (VoE) (Patel et al., 2015)



ELSEVIER

CrossMark

Journal of Clinical Epidemiology 68 (2015) 1046−1058

Journal of Clinical Epidemiology

Assessment of vibration of effects due to model specification can demonstrate the instability of observational associations

Chirag J. Patel[a], Belinda Burford[b], John P.A. Ioannidis[a,c,d,e,f,*]

[a]*Department of Biomedical Informatics, Harvard Medical School, 10 Shattuck St., Room 314A, Boston, MA 02115, USA*
[b]*Melbourne School of Population and Global Health, Level 4, 207 Bouverie St., The University of Melbourne, Victoria 3010, Australia*
[c]*Department of Medicine, Stanford Prevention Research Center, Stanford University School of Medicine, Medical School Office Building, Room X306, 1265 Welch Rd, Stanford, CA 94305, USA*
[d]*Department of Statistics, Stanford University School of Humanities and Sciences, Stanford, CA 94305, USA*
[e]*Department of Health Research and Policy, Stanford University School of Medicine, Stanford, CA 94305, USA*
[f]*Meta-Research Innovation Center at Stanford (METRICS), Stanford University, Stanford, CA 94305, USA*

Accepted 30 May 2015; Published online 6 June 2015

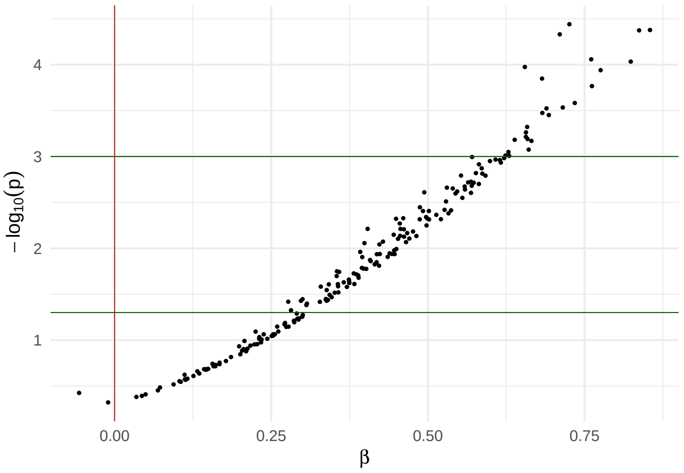
Vibration of Effects (VoE) (Patel et al., 2015)

The VoE is a statistical approach to evaluate the variability in effect estimates and p value due to different sources of variability (i.e., *vibrations*)

- **sampling** vibration: subsets of the full dataset
- **model** vibration: combinations of control variables
- **pre-processing** vibration: inclusio/exclusion criteria, outliers, etc.

Vulcano Plot

The Vulcano Plot is the graphical tool used in the VoE as a diagnostic tool.



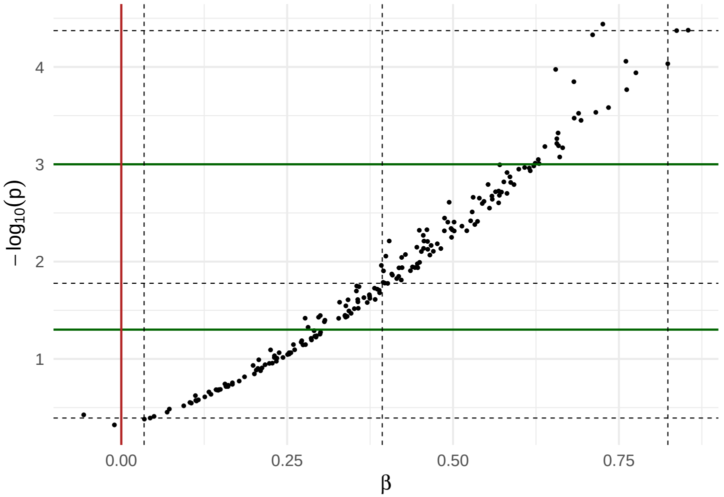
Vulcano Plot

The `x` axis is the effect size. Usually a regression coefficient of a *focal* parameter. Can be a raw or standardized regression coefficient or whatever effect size measure.

The `y` axis is the associated p-value transformed in $-\log_{10}(p)$ for better intepretation and visualization. Higher tranformed p values are smaller raw p values.

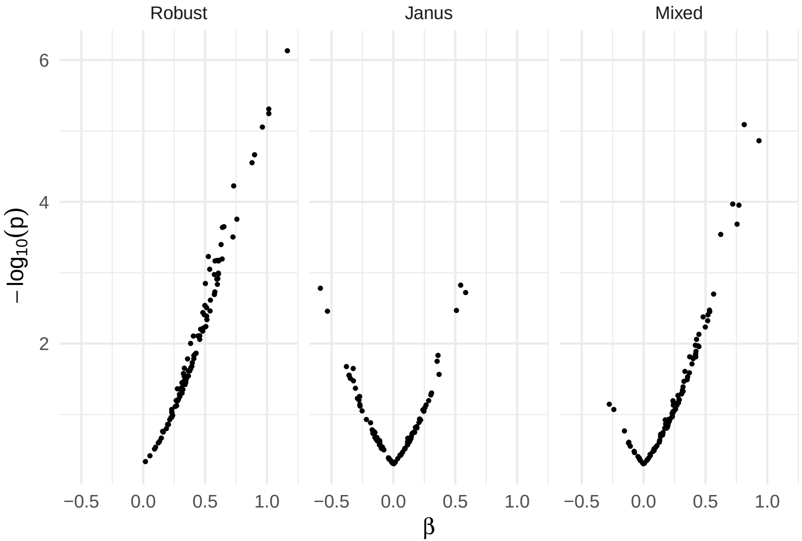
Vibration of Effects (VoE)

The authors proposed to summarise the VoE using the range of effect sizes and p values. In particular the difference between the 99^{th} and 1^{st} percentiles.



Vibration of Effects (VoE)

They identified three usual pattern for a Vulcano Plot:



Vibration of Effects (VoE)

The **Robust** plot suggests a stable pattern across specifications, with the majority if not the total being positive and significant.

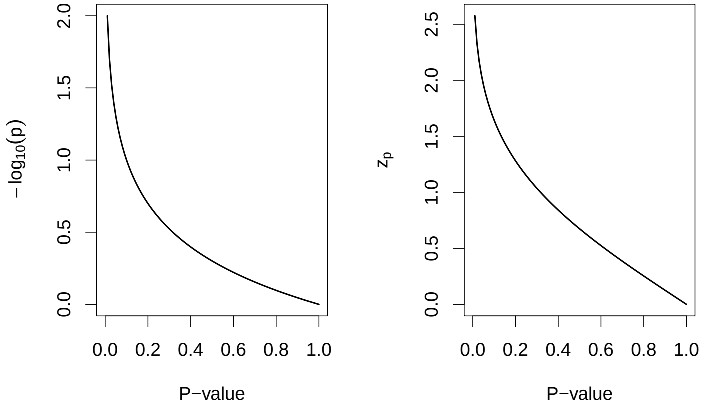
The **Janus**[1] plot suggests the worst scenario where in some conditions the effect is not only not significant but reversed.

The **Mixed** plot suggests a less robust effect with few effect size reversals in rare specifications.

[1] Fun fact: Janus comes from the Roman/Greek god with two faces :)

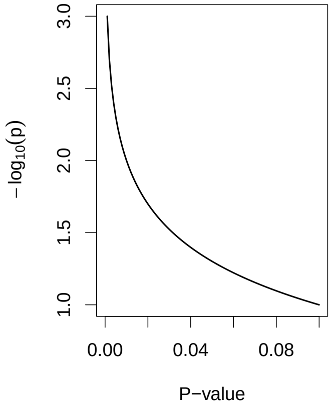
P-values transformation

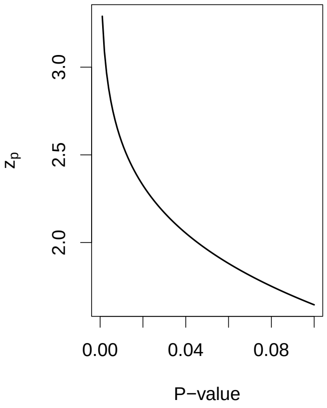
There are different ways to transform p-values to improve the interpretation and visualization.



P-values transformation

Values higher than ~1.3 (in $\log_{10}$) or ~2 (z transformation) are significant assuming the traditional $\alpha = 0.05$.



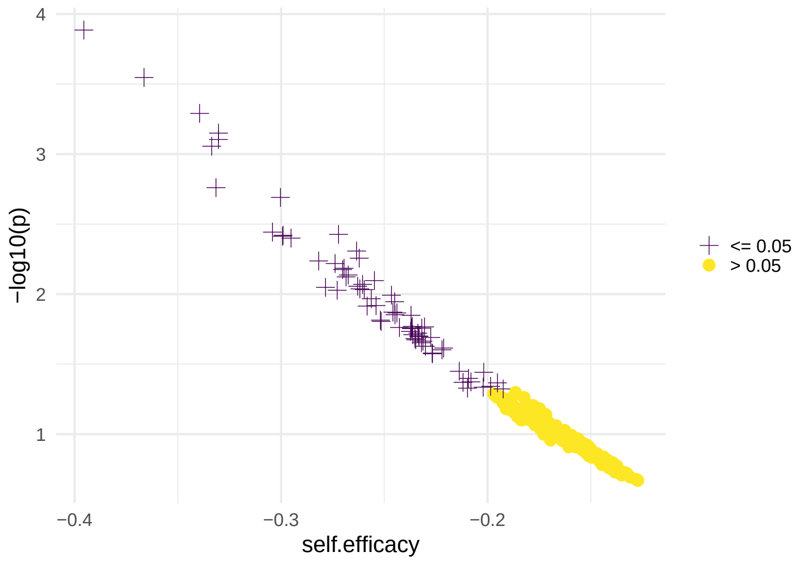


Vulcano plot with our data

We can create a basic version of the vulcano plot with our dataset:

```r
multi_res |>
    mutate(sign = ifelse(p <= 0.05, "<= 0.05", "> 0.05"),
           sign = factor(sign, levels = c("<= 0.05", "> 0.05"))) |>
    ggplot(aes(x = b, y = tp(p, "-log10"))) +
    geom_point(aes(shape = sign, color = sign), size = 5) +
    ylab("-log10(p)") +
    xlab(focal) +
    scale_shape_manual(values = c(3, 16)) +
    theme(legend.title = element_blank())
```

Vulcano plot with our data



Marginal/Conditional effects

A way to evaluate the impact of the multiverse scenario could be to use an ANOVA-style way of thinking. We can fit a regression model on the multiverse where the focal coefficient is the response variable and a series of dummy variables to code the inclusion/exclusion of a certain predictor.

Then we can estimate the % of explained variance of each predictor as an index of the impact in the multiverse results.

A more refined version of this approach can be found in Klau et al. (2023)

Decomposing the multiverse variance

We can create a dataset with dummy variables when a specific predictor is included or not. We are ignoring the transformations of the specific variable.

```
# A tibble: 6 x 6
       b faculty   asi gender.category program.type frost.da
   <dbl>   <dbl> <dbl>           <dbl>        <dbl>    <dbl>
1 -0.396       0     0               0            0        0
2 -0.332       1     0               0            0        0
3 -0.330       0     1               0            0        0
4 -0.304       0     0               1            0        0
5 -0.334       0     0               0            1        0
6 -0.227       0     0               0            0        1
```

Decomposing the multiverse variance

Then we can fit a linear regression and then evaluate the impact of including/excluding a predictor.

```r
fit <- lm(b ~ ., data = multi_fit)
summary(fit)
```

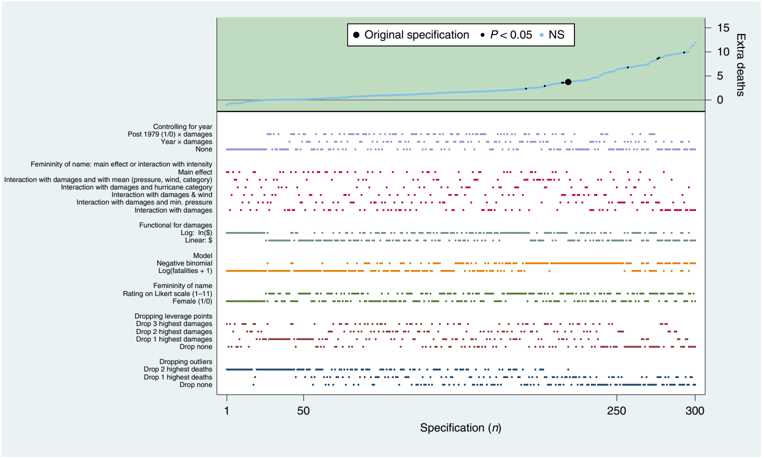
```
Call:
lm(formula = b ~ ., data = multi_fit)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.071822 -0.013166  0.002804  0.014675  0.037509 

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)    
(Intercept)     -0.323699   0.004734 -68.375   <2e-16 ***
faculty          0.027851   0.002733  10.190   <2e-16 ***
asi              0.002285   0.003417   0.669    0.504    
gender.category  0.037357   0.002733  13.668   <2e-16 ***
program.type     0.028599   0.002733  10.463   <2e-16 ***
frost.da         0.099365   0.003417  29.083   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Decomposing the multiverse variance

```
effectsize::eta_squared(fit, partial = FALSE)
```

Specification Curve (Simonsohn et al., 2020)

The specification curve is both an inferential and descriptive tool to summarise the results from a multiverse analysis.



Specification Curve as descriptive tool

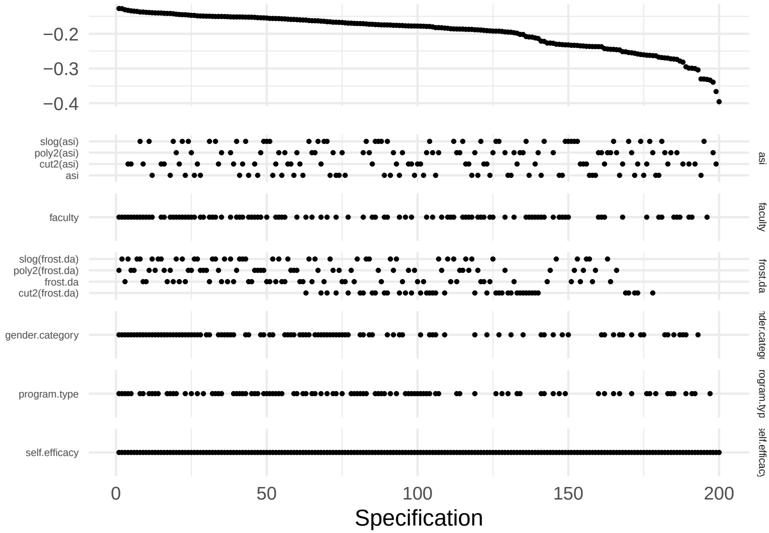
Basically from M specifications we extract the focal coefficient then:

- we sort the coefficients from the lowest to the highest and assign a progressive index
- we plot the index as a function of the coefficient value
- for each scenario we code the corresponding set of conditions/variables
- we combine the previous plot with a tile-plot (or similar) showing for each scenario the set of variables/choices

Specification with the dataset

```r
spec_data <- multi_res |>
    # sorting
    arrange(desc(b)) |>
    # index with the order
    mutate(spec = 1:n())

top <- spec_data |>
    # confidence intervals
    mutate(lb = b - se * 2,
           ub = b + se * 2) |>
    ggplot(aes(x = spec, y = b)) +
    geom_point() +
    theme(axis.text.x = element_blank(),
          axis.title.x = element_blank(),
          axis.title.y = element_blank())

bottom <- spec_data |>
    pivot_longer(starts_with("x_")) |>
    drop_na() |>
    mutate(name = gsub("x_", "", name)) |>
    ggplot(aes(x = spec, y = value)) +
    geom_point() +
    theme(axis.title.y = element_blank(),
          strip.text.y = element_text(size = 9),
          axis.text.y = element_text(size = 9)) +
    xlab("Specification") +
    facet_grid(name~., scales = "free")
```

Specification with the dataset



Other descriptive tools

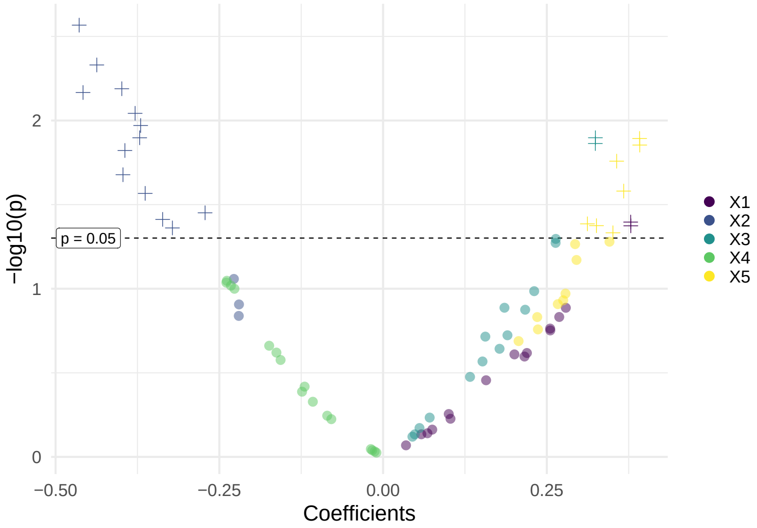
In general, any descriptive statistics can be useful. The main points in a multiverse description are:

- the range of the estimated effects
- the impact of the choices
- the impact on the conclusions (e.g., statistical significance)

Can the EMA be misleading?

Let's have a look to another example

We have a multiverse with 31 scenarios, 50 observations and 5 predictors, this is the vulcano plot. **What do you think?**



Let's have a look to another example[2]

From the previous multiverse it is clear that something is going on. Some of the coefficients are significant and other not. There is also a little bit of Janus effect.

But, the previous example was a simulated multiverse where all the coefficients $\beta_j = 0$ (the null hypothesis is true). **All the significant scenarios are false positives (type-1 error)!**

[2]Thanks to Livio Finos for the insightful example

Let's have a look to another example[2]

From the previous multiverse it is clear that something is going on. Some of the coefficients are significant and other not. There is also a little bit of Janus effect.

But, the previous example was a simulated multiverse where all the coefficients $\beta_j = 0$ (the null hypothesis is true). **All the significant scenarios are false positives (type-1 error)!**

[2]Thanks to Livio Finos for the insightful example

Why? multiple testing problem!

- A multiverse can be considered as a **multiple testing problem** because we are testing a set of hypotheses with the same dataset. The type-1 error rate (α) need to be controlled otherwise the actual level is higher than the nominal level.
- We can demonstrate this with a simple simulation. We simulate k variables and a one-sample t-test for each variable. The ground truth is that we have $\mu_1, \mu_2, \dots, \mu_k = 0$ thus H_0 is true.
- We repeat the simulation B times and we count how many times $p \leq \alpha$ for at least one of the k tests. This is our estimated type-1 error rate.

Why? multiple testing problem!

```r
k <- 10  # number of variables
n <- 100 # number of observations
R <- 0 + diag(1 - 0, k) # correlation matrix
B <- 1e3

PM <- matrix(NA, B, k)

for(i in 1:B){
    X <- MASS::mvrnorm(n, rep(0, k), R)
    p <- apply(X, 2, function(x) t.test(x)$p.value)
    PM[i, ] <- p
}

# type-1 error for each variable, ignoring multiple testing
apply(PM, 2, function(x) mean(x <= 0.05))
```

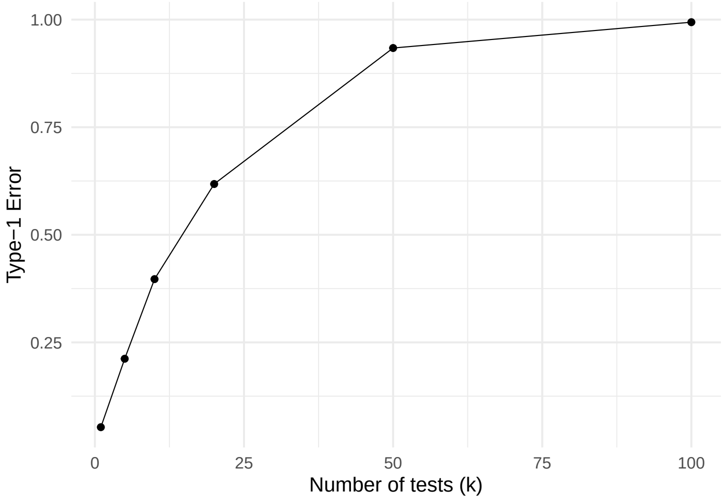
```
 [1] 0.048 0.045 0.055 0.052 0.052 0.061 0.049 0.049 0.056 0.059
```

```r
# type-1 error considering the k tests (should be alpha)
mean(apply(PM, 1, function(x) any(x <= 0.05)))
```

```
[1] 0.427
```

Why? multiple testing problem!

To have a better overview, we can repeat the simulation for different number of k. Quite scary right?



So what? No multiverse?

Exploring is fine and is quite important if not fundamental. But, when we explore the p-values thus the **inferential** results from the single scenarios, we are inflating the type-1 error and our **inferential** conclusions are no longer valid.

If we want an inferential answer from our multiverse (not always the case) we need a proper inferential framework. This is the role of the **inferential multiverse analaysis**.

So what? No multiverse?

Exploring is fine and is quite important if not fundamental. But, when we explore the p-values thus the **inferential** results from the single scenarios, we are inflating the type-1 error and our **inferential** conclusions are no longer valid.

If we want an inferential answer from our multiverse (not always the case) we need a proper inferential framework. This is the role of the **inferential multiverse analaysis**.

Inferential Multiverse Analysis (IMA)

Family-wise error rate (FWER)[3]

		H_0		
		False	**True**	**Tot**
Test	Rejected	**True Positive (S)**	**False Positive (V)**	R
	Not rejected	**False Negative (T)**	**True Negative (U)**	$m - R$
	Tot	m_1	m_0	m

The FWER is the probability of committing type-1 error thus P$(V > 0)$. Controlling the FWER (whatever the methods) keep P $\leq \alpha$.

There are different procedures for controlling the FWER, such as the Bonferroni or the Holm–Bonferroni method.

[3]Thanks to Anna Vesely for the amazing introduction to the multiple testing problem (see the slides)

Correcting the p-values

The main problem is that the number of tests in a multiverse can be quite large.

As an example, we simulated a series of tests with different effect size to show the impact on the type-1 error rate and the power.

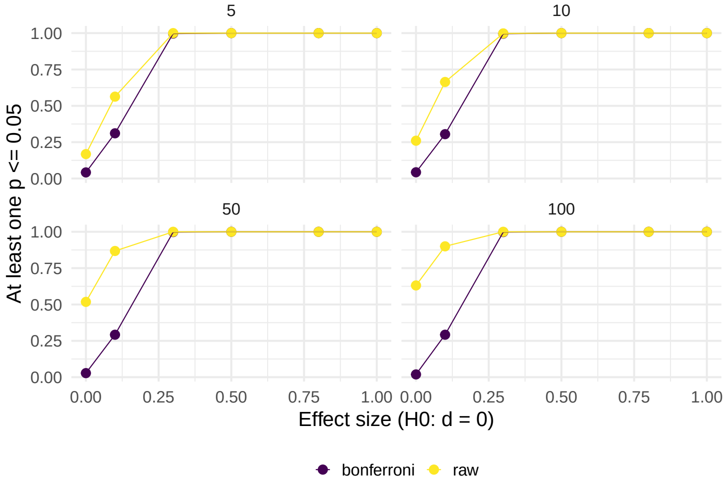
Correcting the p-values

The main problem is that the number of tests in a multiverse can be quite large.

As an example, we simulated a series of tests with different effect size to show the impact on the type-1 error rate and the power.

Correcting the p-values

Using a standard method (e.g., Bonferroni) clearly controls the type-1 error but reduces a lot the statistical power. At the same time, without correction the inflation is large.



Correlation between scenarios is (probably) large

The multiverse scenarios are computed on the same dataset thus the correlation between tests is probably medium-large. For example:

```r
x <- runif(100, 5, 10)
y <- x * 0.1 + rnorm(100)

fit1 <- lm(y ~ x)
fit2 <- lm(y ~ cut(x, breaks = 2))
fit3 <- lm(y ~ log(x))
fit4 <- lm(y ~ poly(x, 2))

pp <- sapply(list(fit1, fit2, fit3, fit4), predict)
round(cor(pp), 2)
```

```
     [,1] [,2] [,3] [,4]
[1,] 1.00 0.88 1.00 0.99
[2,] 0.88 1.00 0.88 0.88
[3,] 1.00 0.88 1.00 1.00
[4,] 0.99 0.88 1.00 1.00
```

A more powerful correction method[4]

The Bonferroni (and similar) methods assume that the tests are independent thus regardless the dependence structure the FWER is under control.

The permutation-based methods (maxT, minP, etc.) take into account the correlation structure providing FWER control under H_0 but a more powerful test under H_1.

[4]Goeman & Solari (2014)

Permutation testing in a nutshell

Permutation testing requires computing the distribution of the test statistics T where we know that H_0 is true.

We can force the null to be true permuting the data *removing* the assumed effect. We repeat this process a large number of times B.

Then we compare the observed test statistics T_1 with the distribution of permuted test statistics obtaining the permutation based p-value $p = \frac{\#(T_1 \geq \mathbf{T}_B)}{B}$[5]

[5] $\#$ is the count function.

Permutation testing in a nutshell

Permutation testing requires computing the distribution of the test statistics T where we know that H_0 is true.

We can force the null to be true permuting the data *removing* the assumed effect. We repeat this process a large number of times B.

Then we compare the observed test statistics T_1 with the distribution of permuted test statistics obtaining the permutation based p-value $p = \frac{\#(T_1 \geq \mathbf{T}_B)}{B}$[5]

[5] # is the count function.

Permutation testing in a nutshell

Permutation testing requires computing the distribution of the test statistics T where we know that H_0 is true.

We can force the null to be true permuting the data *removing* the assumed effect. We repeat this process a large number of times B.

Then we compare the observed test statistics T_1 with the distribution of permuted test statistics obtaining the permutation based p-value $p = \frac{\#(T_1 \geq \mathbf{T}_B)}{B}$[5]

[5] $\#$ is the count function.

Permutation testing in a nutshell

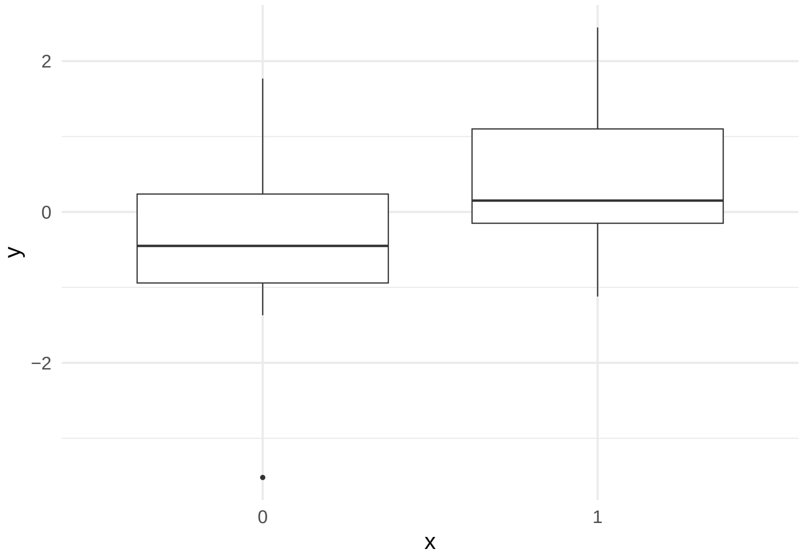
Let's make an example with a two-groups comparison:

```r
N <- 30
d <- 1 # effect size
x <- rep(c(0, 1), each = N/2) # dummy for the group
y <- rnorm(N, d * x, 1)
tapply(y, x, mean)
```

```
         0          1 
-0.3933720  0.4743739 
```

Permutation testing in a nutshell

Let's make an example with a two-groups comparison:



Permutation testing in a nutshell

We need to flip the group label thus removing the group effect.

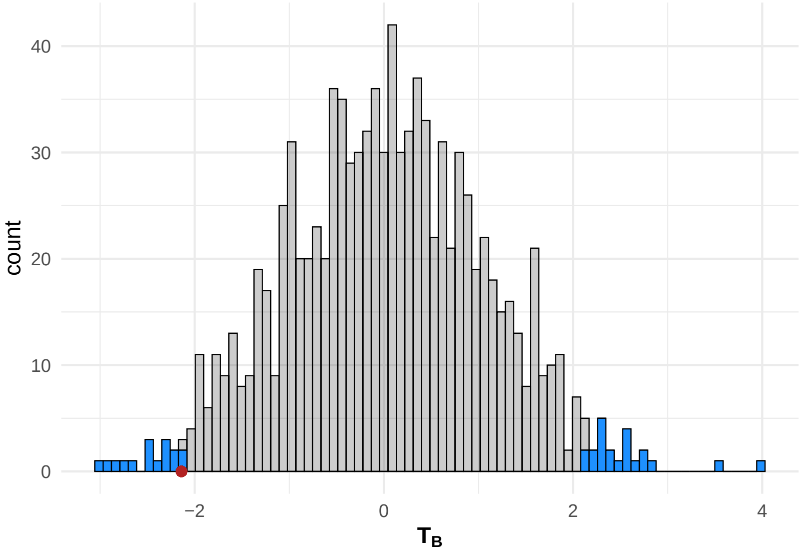
```r
B <- 1e3 # number of permutations
tp <- rep(NA, B)
tp[1] <- t.test(y ~ x)$statistic # first permutation always the observed data

sample(x) # shuffling the group label
```

```
 [1] 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0 1 1 0 1 0 0 0 1 1 1 1 1 1 0 1
```

```r
for(i in 2:B){
    xp <- sample(x)
    tp[i] <- unname(t.test(y ~ xp)$statistic)
}

mean(abs(tp) >= abs(tp[1]))
```

```
[1] 0.038
```

Permutation testing in a nutshell



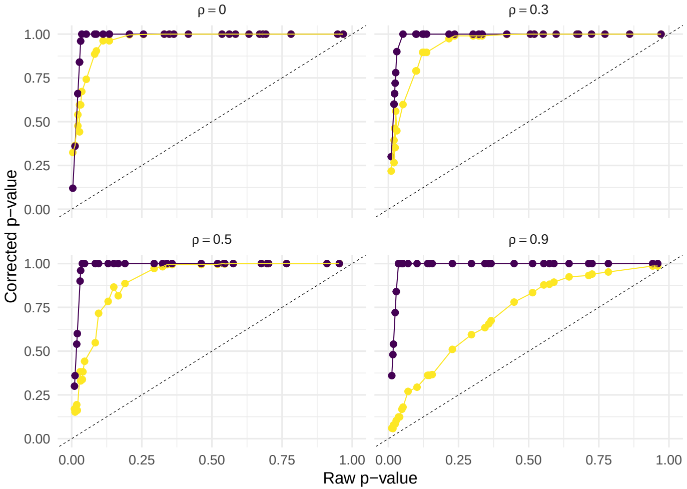
maxT procedure Westfall & Stanley Young (1993)

The maxT is a permutation-based method to control the FWER. With the method we can obtain:

- overall inference across M tests with *weak* control of FWER
- individually adjusted p-values for each test (i.e, *strong* FWER control)

maxT with correlated variables

Beyond the actual method and algorithm, the advantage of the maxT approach is taking into account the correlation between tests.



Inferential Methods

- Specification Curve
- Post-Selection Inference in Multiverse Analysis (PIMA)

Specification Curve

The specification curve (Simonsohn et al., 2020) is the first attempt to build an inferential framework for multiverse analysis.

- provides only *weak* control of type-1 error
- is not directly applicable to GLMs (only standard linear models, see Girardi et al., 2024)
- is computationally expensive

The PIMA recipe... (Girardi et al., 2024)

PIMA provides *weak* and *strong* type-1 error control with a powerful method based on permutations (maxT) and applicable to whatever GLM (Logistic, Poisson, etc.).

For constructing the inferential approach with PIMA we need:

- a flexible modelling framework: **Generalized Linear Models**
- a permutation-based inferential approach: **Flipscores**
- a permutation-based and powerful method for weak and strong FWER control: **maxT**

The core of PIMA, the flipscores method

- The formal part of the **flipscores** method is quite complex and beyond our scope and expertise. But a detailed description can be found in Hemerik et al. (2020) and Girardi et al. (2024).
- Essentially the **flipscores** method is an alternative way of doing inference for parameters of a GLM based on permutations.
- The idea is conceptually the same as the two-groups example, but can works for multiple regression models with covariates and interactions.

Intution of flipscores

This method can be extended to whatever GLM and to any number of predictors/confounders.

The actual permutation test is obtained flipping the sign of the scores/residuals thus obtaining the distribution under the null hypothesis of the test statistics.

Everthing is implemented into the `flipscores` package (Hemerik et al., 2020) and on CRAN https://cran.r-project.org/web/packages/flipscores/index.html.

flipscores package

With the `flipscores` function is very easy to fit a linear model with permutations-based p-values.

```r
library(flipscores)
fit <- flipscores(Sepal.Length ~ Petal.Width + Species, data = iris)
summary(fit)
```

```
Call:
flipscores(formula = Sepal.Length ~ Petal.Width + Species, data = iris)

Coefficients:
                   Estimate     Score Std. Error   z value Part. Cor
(Intercept)         4.78044 160.25913   13.50622  11.86558     0.979
Petal.Width         0.91690   5.64500    1.27732   4.41941     0.365
Speciesversicolor  -0.06025  -0.26260    1.00098  -0.26234    -0.022
Speciesvirginica   -0.05009  -0.09030    0.64372  -0.14028    -0.012
                  Pr(>|z|)
(Intercept)         0.0002 ***
Petal.Width         0.0002 ***
Speciesversicolor   0.7954
Speciesvirginica    0.9014
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for gaussian family taken to be 0.2313718)

    Null deviance: 102.17  on 149  degrees of freedom
Residual deviance:  33.78  on 146  degrees of freedom
AIC: 212.07

Number of Fisher Scoring iterations: 2
```

Intuition of PIMA

The idea of PIMA is to extend the `flipscores` method to M models (where M is the number of scenarios) and perform inference at the multiverse level.

Using the maxT approach we can combine the M tests into a single test with weak control of FWER. The global null hypothesis is:

$$\mathcal{H} = \bigcap_{m=1}^{M} \mathcal{H}_m : \beta_m = 0 \text{ for all } m = 1, \ldots, M.$$

In addition, we can correct the indidual p-values with strong FWER control using the maxT method.

The `pima` package

We are implementing everything into the `pima` package that is under development. You can try it but there could be bugs and breaking changes in the near future.

You can explore the package here https://github.com/livioivil/pima. The package mainly depends on `jointest` that is the actual package for combining multiple (correlated) tests and correcting them.

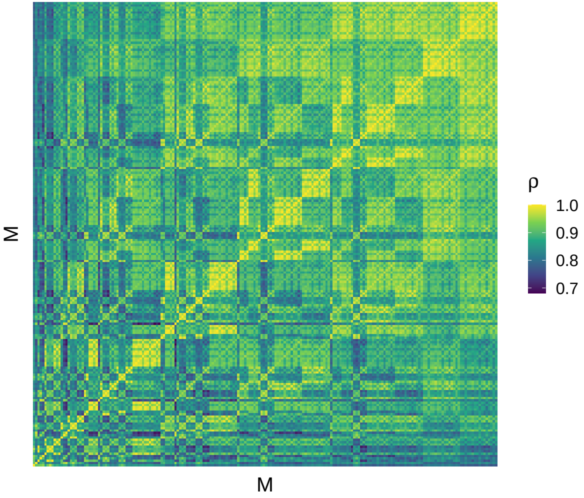
The `pima` package

The package has a main function called `pima::pima()` that takes a list of models (of class `glm`) and compute the global test and the correction for individual scenarios.

```r
# fitl is the list of fitted models
# tested_coeffs is the focal variable, other are confounders

library(pima)

# this is a bug/part to be improved, ignore
for(i in 1:length(fitl)){
    fitl[[i]]$call$formula <- as.formula(fitl[[i]]$formula)
}

multi_pima <- pima(fitl, tested_coeffs = "self.efficacy")
```

The `pima` results

The correlations between the scenarios is very high, the maxT method will be powerful!



The `pima` results

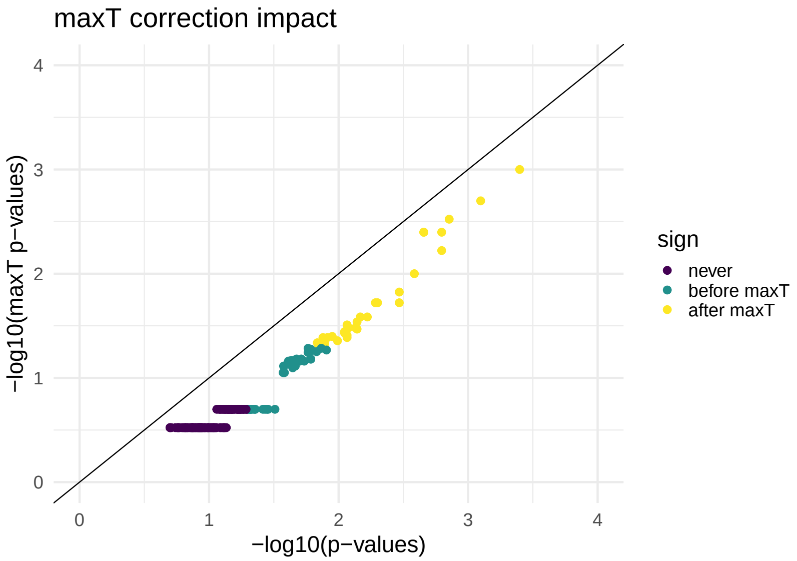
```r
# overall test, weak control FWER
summary(pima::global_tests(multi_pima))
```

```
    Model          Coeff Stat nTests     S     p
1 Overall self.efficacy maxT    200 30.26 4e-04
```

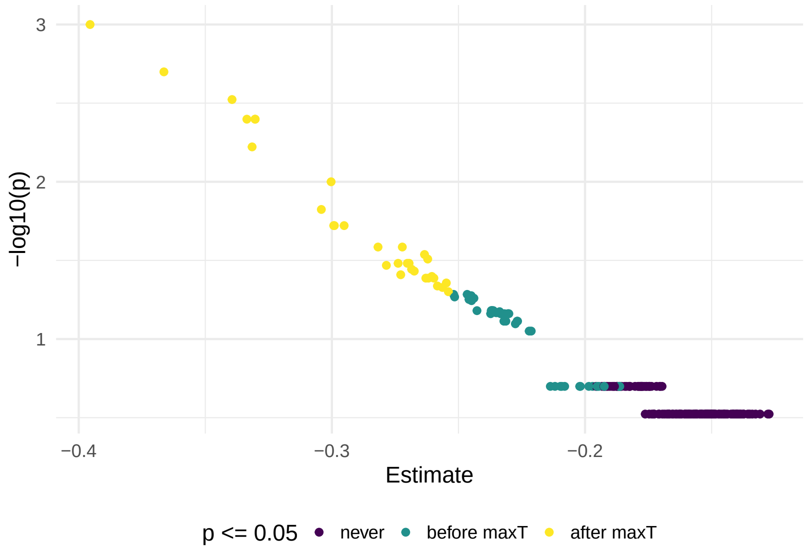
```r
# adjusted p values, strong control FWER
head(summary(multi_pima))
```

```
   Model .assign         Coeff   Estimate     Score Std. Error
1 Model1       1 self.efficacy -0.3955213 -30.26473   8.026511
2 Model2       1 self.efficacy -0.3315150 -23.30487   7.506267
3 Model3       1 self.efficacy -0.3302943 -24.84790   7.416916
4 Model4       1 self.efficacy -0.3041665 -21.44863   7.420203
5 Model5       1 self.efficacy -0.3335836 -24.92025   7.567967
6 Model6       1 self.efficacy -0.2266493 -15.42204   6.974630
    z value  Part. Cor      p p.adj.maxT
1 -3.770596 -0.2672905 0.0004      0.001
2 -3.104722 -0.2212023 0.0016      0.006
3 -3.350167 -0.2380860 0.0022      0.004
4 -2.890572 -0.2054240 0.0034      0.015
5 -3.292859 -0.2340133 0.0016      0.004
6 -2.211162 -0.1571404 0.0266      0.077
```

maxT correction



Improved vulcano plot



References

Girardi, P., Vesely, A., Lakens, D., Altoè, G., Pastore, M., Calcagnì, A., & Finos, L. (2024). Post-selection inference in multiverse analysis (PIMA): An inferential framework based on the sign flipping score test. *Psychometrika*, *89*, 542–568. https://doi.org/10.1007/s11336-024-09973-6

Goeman, J. J., & Solari, A. (2014). Multiple hypothesis testing in genomics. *Statistics in Medicine*, *33*, 1946–1978. https://doi.org/10.1002/sim.6082

Götz, M., Sarma, A., & O'Boyle, E. H. (2024). The multiverse of universes: A tutorial to plan, execute and interpret multiverses analyses using the r package multiverse. *International Journal of Psychology: Journal International de Psychologie*, *59*, 1003–1014. https://doi.org/10.1002/ijop.13229

Hemerik, J., Goeman, J. J., & Finos, L. (2020). Robust testing in generalized linear models by sign flipping score contributions. *Journal of the Royal Statistical Society. Series B, Statistical Methodology*, *82*, 841–864. https://doi.org/10.1111/rssb.12369

Klau, S., Felix, Patel, C. J., Ioannidis, J. P. A., Boulesteix, A.-L., & Hoffmann, S. (2023). Comparing the vibration of effects due to model, data pre-processing and sampling uncertainty on a large data set in personality psychology. *Meta-Psychology*, *7*. https://doi.org/10.15626/mp.2020.2556

References (cont.)

McCaughey, N. J., Hill, T. G., & Mackinnon, S. P. (2022). The association of self-efficacy, anxiety sensitivity, and perfectionism with statistics and math anxiety. *Personality Science*, *3*. https://doi.org/10.5964/ps.7091

Patel, C. J., Burford, B., & Ioannidis, J. P. A. (2015). Assessment of vibration of effects due to model specification can demonstrate the instability of observational associations. *Journal of Clinical Epidemiology*, *68*, 1046–1058. https://doi.org/10.1016/j.jclinepi.2015.05.029

Simonsohn, U., Simmons, J. P., & Nelson, L. D. (2020). Specification curve analysis. *Nature Human Behaviour*, *4*, 1208–1214. https://doi.org/10.1038/s41562-020-0912-z

Westfall, P. H., & Stanley Young, S. (1993). *Resampling-based multiple testing: Examples and methods for p-value adjustment*. John Wiley & Sons.