Demonstrating functional mediation

John J. Dziak

2020-09-01

Introduction

The funmediation package provides software for exploring functional mediation in a sample of intensive longitudinal data from multiple individuals. It assumes a dichotomous non-time-varying randomized treatment or exposure X, a distal outcome Y observed at one time point, and a time-varying mediator $M(t)$. The mediator is assumed to be measured as intensive longitudinal data. The research question is whether it can be shown that the effect of X on Y is mediated by the process $M(t)$. This situation was first studied by Lindquist (2012, JASA). $M(t)$ can be numerical or binary, and Y can also be either numerical or binary; X is assumed to be binary.

Notation

The mediation model is fit in stages. First, the effect of X on M is fit behind the scenes as a time-varying effects (varying coefficients) model using the `tvem` package in R. The assumed mean model is $E(M_i(t)) = \alpha_0(t) + \alpha_X(t)X_i$ for numerical $M(t)$, or $\text{logit}^{-1}(E(M_i(t))) = \alpha_0(t) + \alpha_X(t)X_i$ for binary $M(t)$. This is fit as a marginal model: that is, without random effects but with a sandwich covariance estimate to handle within-subject correlation, as in working-independence generalized estimating equations.

Next, the effect of X and M on Y is modeled as a scalar-on-function functional regression using the `pfr` function in the `refund` R package. The assumed mean model is $E(Y_i) = \beta_0 + \beta_X X_i + \int \beta_M(t)M_i(t)dt$ for numerical Y, or $\text{logit}^{-1}(E(Y_i)) = \beta_0 + \beta_X X_i + \int \beta_M(t)M_i(t)dt$ for binary Y.

The indirect effect is calculated as the integral $\int \alpha_X(t)\beta_M(t)$. Because these functions are actually only estimated on a grid of points, the integral is actually approximated as a weighted average of the cross-products of the estimates. We do not have a standard error formula for this indirect effect, so we obtain a bootstrap confidence interval using the `boot` package in R.

Including Covariates

Covariates can be included in predicting $M(t)$ and Y. For convenience of notation, we assume a single subject-level covariate S which predicts $M(t)$ and Y and a single time-varying covariate $Z(t)$ which also predicts $M(t)$.

Example

Getting ready to run the example

First we load the required packages.

```
library(tvem)
#> Loading required package: mgcv
#> Loading required package: nlme
```

```
#> This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
library(refund)
library(boot)
#>
#> Attaching package: 'boot'
#> The following object is masked from 'package:refund':
#>
#>     cd4
library(funmediation)
```

We then simulate data. The `info1` object will contain not only the simulated dataset, but the true values of the simulated parameters, including the indirect effect.

```
set.seed(123);
info1 <- simulate_funmediation_example();
the_data <- info1$dataset;
```

We can use the `head` and `summary` function in R, in order to look at the first few lines of the data

```
print(head(the_data));
#>    subject_id    t X           M       Y
#> 2           1 0.02 0 -1.60907573 0.35016
#> 3           1 0.03 0 -0.09779362 0.35016
#> 5           1 0.05 0  5.74371316 0.35016
#> 10          1 0.10 0  5.02605496 0.35016
#> 16          1 0.16 0  2.63183913 0.35016
#> 23          1 0.23 0  2.75861375 0.35016
```

We can also look at univariate descriptive statistics:

```
summary(the_data);
#>    subject_id          t               X                M
#>  Min.   :  1.0   Min.   :0.010   Min.   :0.0000   Min.   :-7.7347
#>  1st Qu.: 62.5   1st Qu.:0.250   1st Qu.:0.0000   1st Qu.:-0.9597
#>  Median :125.0   Median :0.500   Median :0.0000   Median : 0.4164
#>  Mean   :125.1   Mean   :0.502   Mean   :0.4888   Mean   : 0.4251
#>  3rd Qu.:187.0   3rd Qu.:0.750   3rd Qu.:1.0000   3rd Qu.: 1.7823
#>  Max.   :250.0   Max.   :1.000   Max.   :1.0000   Max.   : 8.8885
#>        Y
#>  Min.   :-2.2391
#>  1st Qu.:-0.3291
#>  Median : 0.2254
#>  Mean   : 0.2667
#>  3rd Qu.: 0.8695
#>  Max.   : 2.5118
```

It would be reasonable to do other descriptive analyses also, but in order to demonstrate the function we proceed directly to the main analysis.

Running a Functional Mediation Model

Now call the functional mediation function. Only 20 bootstrap samples are used below for the illustration. At least a few hundred bootstraps is recommended in practice in order to increase precision and power.

```
model1 <- funmediation(data=the_data,
                               treatment=X,
```

```
                            mediator=M,
                            outcome=Y,
                            id=subject_id,
                            time=t,
                            nboot=20);
#> Ran original results.
#> Working on bootstrap results:
#> 1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20.Done bootstrapping.
#> Warning in norm.inter(t, (1 + c(conf, -conf))/2): extreme order statistics used
#> as endpoints
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
```

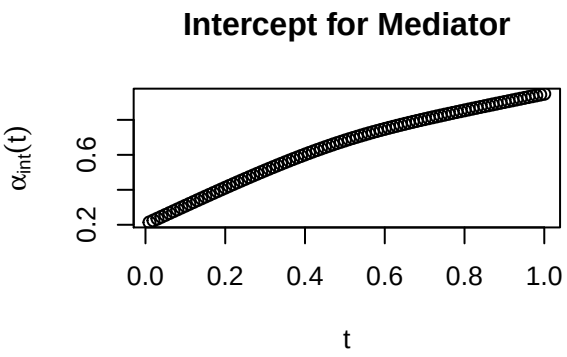
The `print` function gives an initial overview of the results.

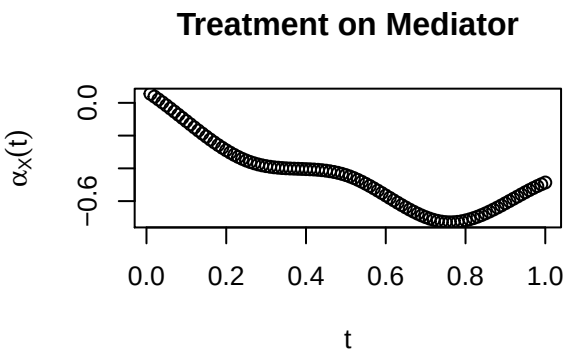
```
print(model1);
#> ======================================================
#> Functional Regression Mediation Function Output
#> ======================================================
#> Indirect effect bootstrap estimate:
#> -0.1047045
#> Indirect effect bootstrap confidence interval:
#> ... by normal method:
#> -0.193 ,  -0.0164
#> ... by percentile method:
#> -0.1379 ,  0.0142
#> Computation time:
#> Time difference of 36.19227 secs
#> ======================================================
#> TVEM Model for Predicting Mediator from Treatment:
#> Response variable:   mediator
#> Time interval:   0.01 to 1
#> Number of subjects:  250
#> Effects specified as time-varying:  (Intercept), treatment
#> You can use the plot function to view their plots.
#>
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> mediator ~ treatment + s(time, bs = "ps", by = NA, pc = 0, k = 7,
#>     fx = FALSE) + s(time, bs = "ps", by = treatment, pc = 0,
#>     m = c(2, 1), k = 7, fx = FALSE)
#> Pseudolikelihood AIC: 42181.28
#> Pseudolikelihood BIC: 42213.8
#>
#> ======================================================
#> Functional Mediation Model for Predicting Mediator from Treatment:
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> wide_outcome ~ s(x = wide_mediator.tmat, by = L.wide_mediator) +
#>     wide_treatment
#>
```

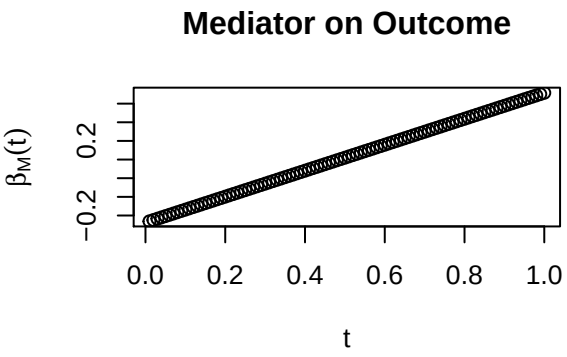
```
#> Estimated degrees of freedom:
#> 2  total = 4
#>
#> REML score: 332.6466
#> ======================================================
#> Scalar terms:
#>    (Intercept) wide_treatment
#>     0.15405471     0.08210633
#> ======================================================
#> Parametric model for Predicting Outcome from Treatment and Mediator:
#>
#> Call:
#> glm(formula = glm_formula)
#>
#> Deviance Residuals:
#>      Min        1Q    Median        3Q       Max
#> -2.50315  -0.59245  -0.04527   0.60490   2.24758
#>
#> Coefficients:
#>                 Estimate Std. Error t value Pr(>|t|)
#> (Intercept)    0.2640562  0.0802042   3.292  0.00114 **
#> wide_treatment 0.0001341  0.1148120   0.001  0.99907
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 0.8233878)
#>
#>     Null deviance: 204.2  on 249  degrees of freedom
#> Residual deviance: 204.2  on 248  degrees of freedom
#> AIC: 664.88
#>
#> Number of Fisher Scoring iterations: 2
#>
#> ======================================================
```

The `plot` function plots the results. Three kinds of plots are available, represented by the options `coef`, `tvem`, and `pfr`.

```
plot(model1, what_plot="coef");
```

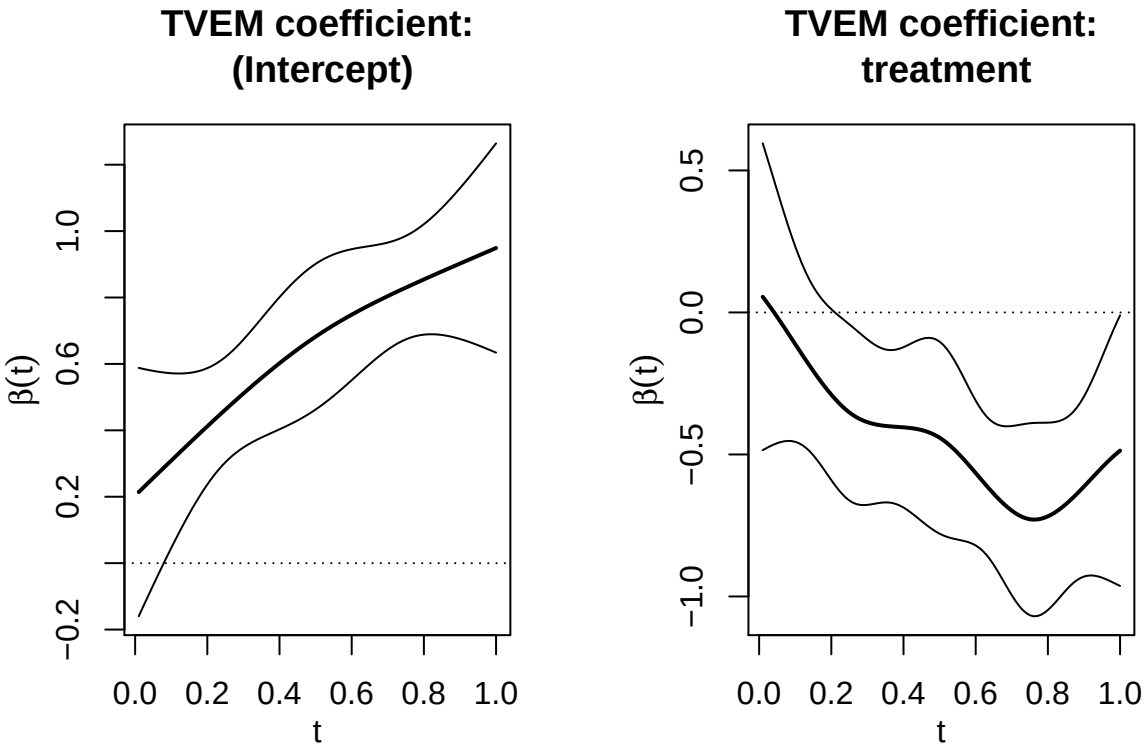




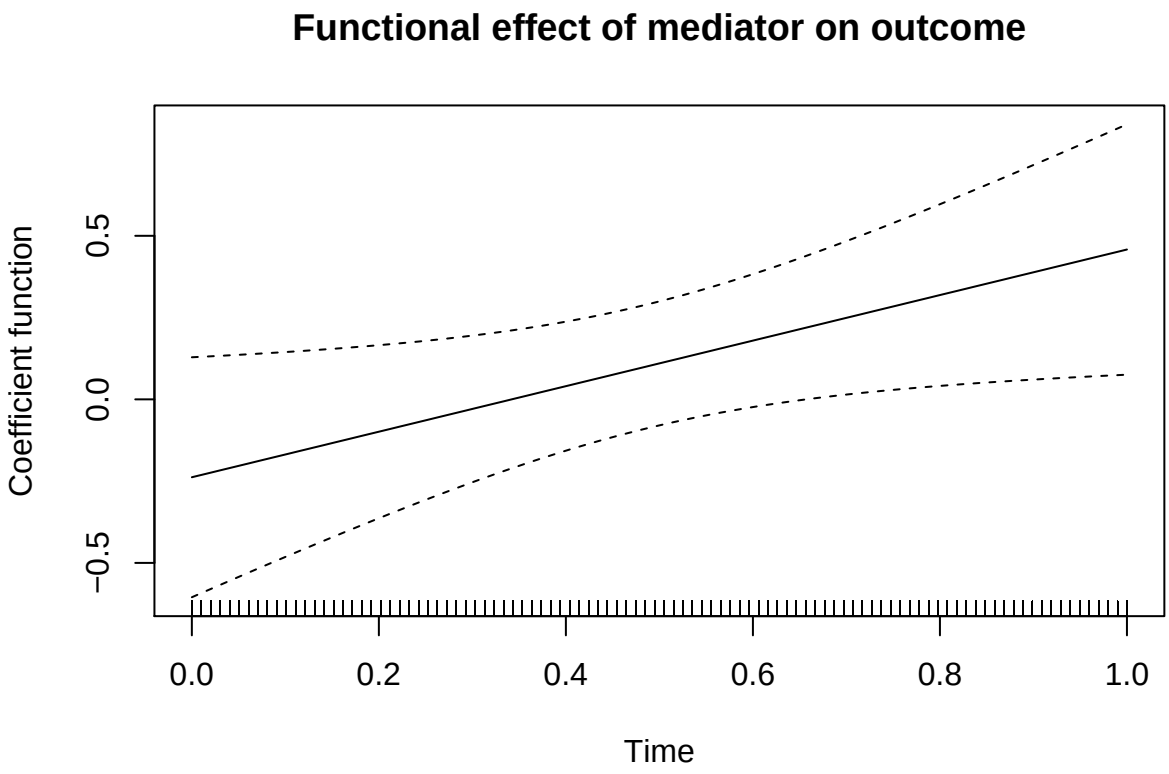


Mediator intercept:
0.15
Indirect effect:
−0.1

```
plot(model1, what_plot="tvem");
```




```
plot(model1, what_plot="pfr");
```



Interpreting the Output

Exploring Information in the Output Objects

Including Covariates

Binary Outcomes

```
set.seed(123);
info2 <- simulate_funmediation_example(simulate_binary_Y=TRUE);
the_data_binary <- info2$dataset;
head(the_data_binary)
#>    subject_id    t X           M Y
#> 3           1 0.03 0 -0.09779362 1
#> 8           1 0.08 0  4.82621071 1
#> 9           1 0.09 0  5.45629031 1
#> 15          1 0.15 0  1.86080077 1
#> 16          1 0.16 0  2.63183913 1
#> 21          1 0.21 0  1.62314539 1
```

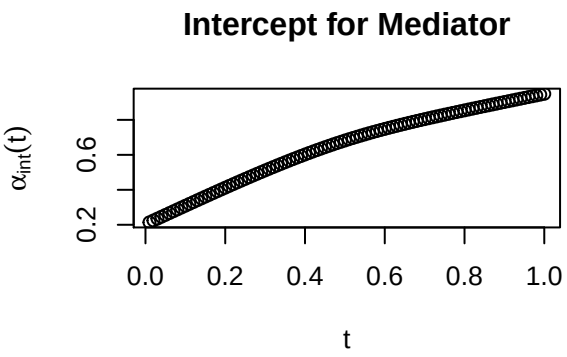
```
model2 <- funmediation(data=the_data_binary,
                           treatment=X,
                           mediator=M,
                           outcome=Y,
                           id=subject_id,
                           time=t,
```

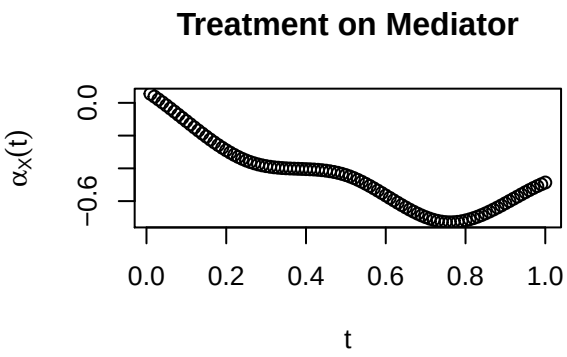
```
                            binary_outcome=TRUE,
                            nboot=20);
#> Ran original results.
#> Working on bootstrap results:
#> 1.2.3.4.5.6.7.8.9.10.11.12.13.14.15.16.17.18.19.20.Done bootstrapping.
#> Warning in norm.inter(t, (1 + c(conf, -conf))/2): extreme order statistics used
#> as endpoints
#> Warning in norm.inter(t, alpha): extreme order statistics used as endpoints
```

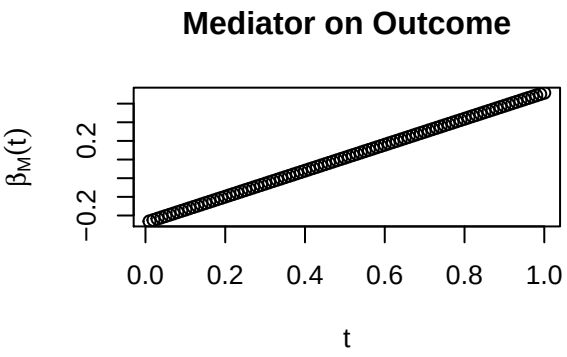
```
print(model2);
#> ======================================================
#> Functional Regression Mediation Function Output
#> ======================================================
#> Indirect effect bootstrap estimate:
#> -0.1334394
#> Indirect effect bootstrap confidence interval:
#> ... by normal method:
#> -0.3367 ,  0.0699
#> ... by percentile method:
#> -0.3892 ,  0.0049
#> Computation time:
#> Time difference of 36.59322 secs
#> ======================================================
#> TVEM Model for Predicting Mediator from Treatment:
#> Response variable:   mediator
#> Time interval:   0.01 to 1
#> Number of subjects:  250
#> Effects specified as time-varying:  (Intercept), treatment
#> You can use the plot function to view their plots.
#>
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> mediator ~ treatment + s(time, bs = "ps", by = NA, pc = 0, k = 7,
#>     fx = FALSE) + s(time, bs = "ps", by = treatment, pc = 0,
#>     m = c(2, 1), k = 7, fx = FALSE)
#> Pseudolikelihood AIC: 42052.91
#> Pseudolikelihood BIC: 42082.27
#>
#> ======================================================
#> Functional Mediation Model for Predicting Mediator from Treatment:
#>
#> Family: binomial
#> Link function: logit
#>
#> Formula:
#> wide_outcome ~ s(x = wide_mediator.tmat, by = L.wide_mediator) +
#>     wide_treatment
#>
#> Estimated degrees of freedom:
#> 2.77  total = 4.77
#>
#> REML score: 171.697
#> ======================================================
```

```
#> Scalar terms:
#>    (Intercept) wide_treatment
#>    -0.02471272     0.26851448
#> ========================================================
#> Parametric model for Predicting Outcome from Treatment and Mediator:
#>
#> Call:
#> glm(formula = glm_formula, family = binomial)
#>
#> Deviance Residuals:
#>    Min      1Q  Median      3Q     Max
#> -1.321  -1.272   1.041   1.086   1.086
#>
#> Coefficients:
#>                Estimate Std. Error z value Pr(>|z|)
#> (Intercept)      0.2196     0.1778   1.235    0.217
#> wide_treatment   0.1112     0.2556   0.435    0.663
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 341.94  on 249  degrees of freedom
#> Residual deviance: 341.75  on 248  degrees of freedom
#> AIC: 345.75
#>
#> Number of Fisher Scoring iterations: 4
#>
#> ========================================================
```

```
plot(model1, what_plot="coef");
```

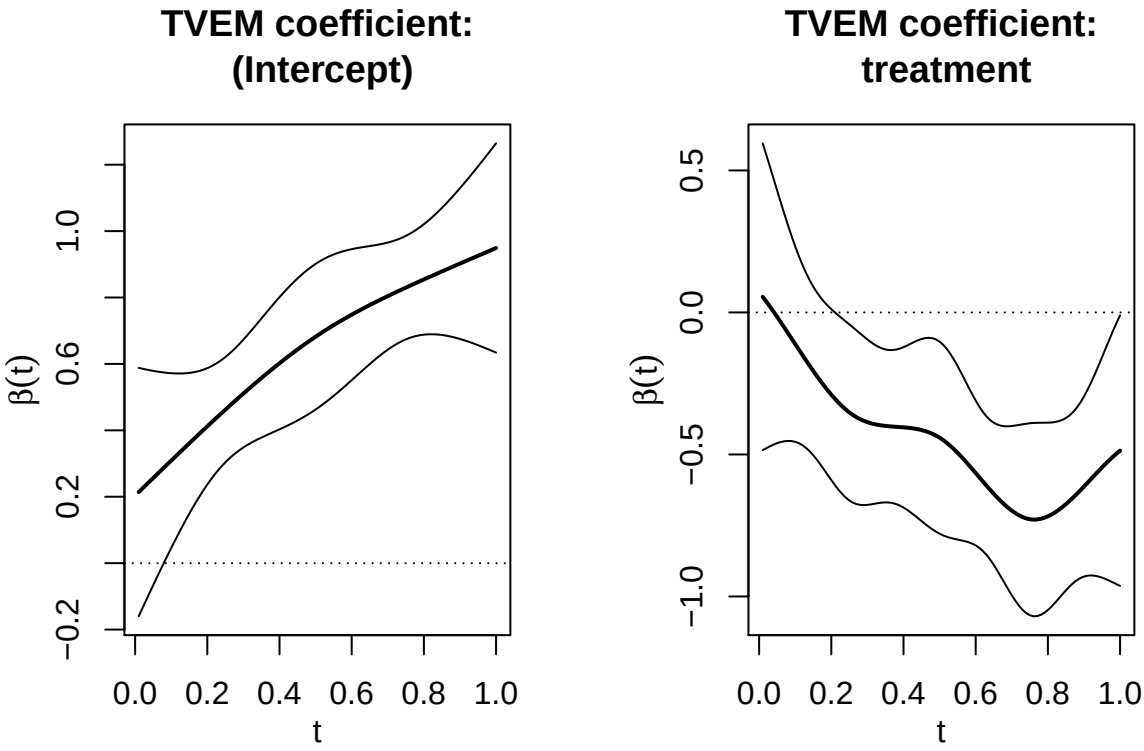




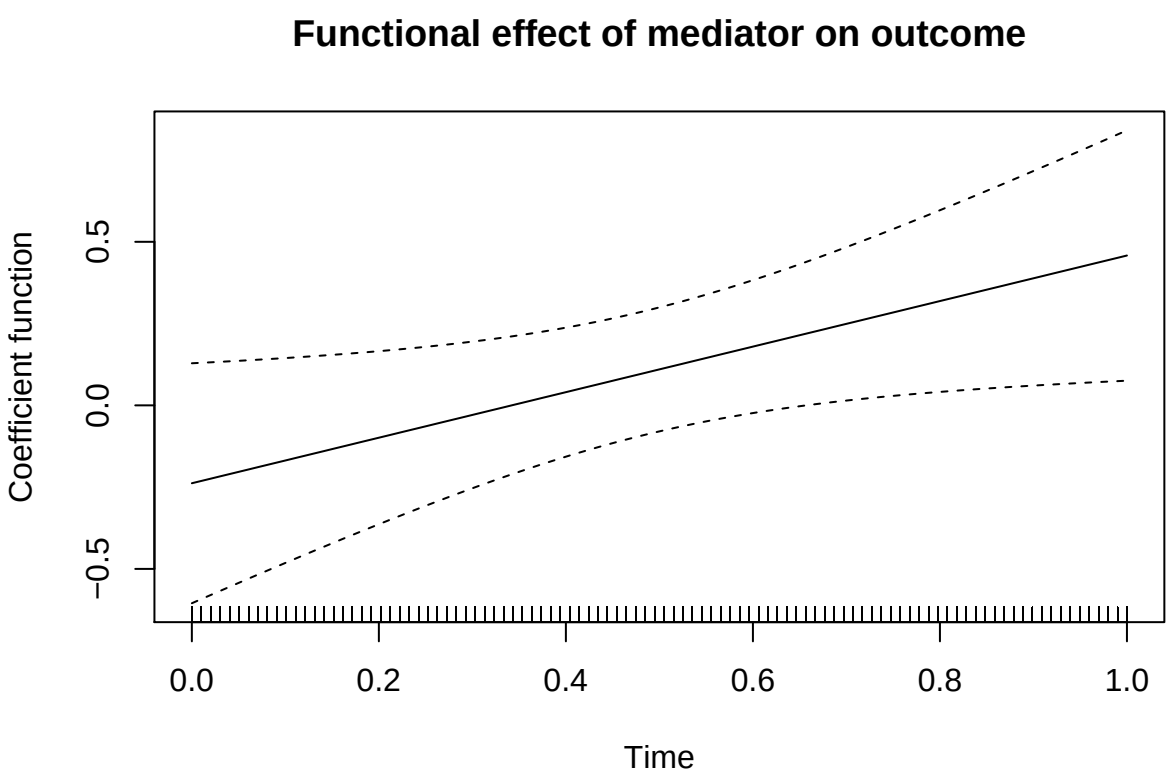


Mediator intercept:
0.15
Indirect effect:
−0.1

```
plot(model1, what_plot="tvem");
```




```
plot(model1, what_plot="pfr");
```



Binary Mediators

Discussion

Acknowledgements

The pfr function was developed by Jonathan Gellar, Mathew W. McLean, Jeff Goldsmith, and Fabian Scheipl, and the refund package was developed and maintained by Julia Wrobel and others