Demonstrating functional mediation

John J. Dziak

2020-09-24

Introduction

The `funmediation` package fits a functional mediation model to a dataset consisting of intensive longitudinal data for a sample of individuals. For each individual i, the model assumes a dichotomous non-time-varying randomized treatment or exposure X_i, a distal outcome Y_i observed at one time point, and a time-varying mediator $M_i(t)$ observed repeatedly during the interval after X_i is observed and before Y_i is observed. The research question is whether the effect of X_i on Y_i is mediated by the process $M_i(t)$. This kind of model was first studied by Lindquist (2012). $M_i(t)$ can be numerical or binary, and Y_i can also be either numerical or binary. The treatment variable X_i is assumed to be binary.

Notation

The mediation model for `funmediation` is fit in stages. First, the effect of X_i on M_i is fit behind the scenes as a time-varying effects (longitudinal varying coefficients) model (see Hastie and Tibshirani, 1993; Tan et al., 2012) using the `tvem` package. The assumed mean model is $E(M_i(t)) = \alpha_0(t) + \alpha_X(t)X_i$ for numerical $M_i(t)$, or $\text{logit}^{-1}(E(M_i(t))) = \alpha_0(t) + \alpha_X(t)X_i$ for binary $M_i(t)$. This fits the marginal (population-averaged) mean: that is, without random effects but with a sandwich covariance estimate to handle within-subject correlation, as in working-independence generalized estimating equations. This part of the model is fit using the `tvem` R package.

Next, the effect of X_i and $M_i(t)$ on Y_i is modeled as a scalar-on-function functional regression (see Goldsmith et al., 2011) using the `pfr` function in the `refund` package. The assumed mean model is either $E(Y_i) = \beta_0 + \beta_X X_i + \int \beta_M(t)M_i(t)dt$ for numerical Y_i, or $\text{logit}^{-1}(E(Y_i)) = \beta_0 + \beta_X X_i + \int \beta_M(t)M_i(t)dt$ for binary Y_i. This part of the model is fit using the `pfr` function in the `refund` R package (see Goldsmith et al., 2011). Both the `tvem` and `refund` packages use the `mgcv` package (see Wood, 2017) for back-end calculations.

In this mediation model, $\alpha_X(t)$ and $\beta_M(t)$ both must be nonzero in order for an indirect effect (i.e., mediation) to exist. These two functions arise conceptually from two different kinds of functional regression; one represents a model with a concurrent function as a response (of which TVEM is a special case), while the other represents a model with a distal scalar response (see Ramsay and Silverman, 2005). However, intuitively the size of the indirect effect has to do with both of them combined. Here we operationally define the indirect effect here as the integral $\int \alpha_X(t)\beta_M(t)dt$. 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 obtain a bootstrap confidence interval for this quantity using the `boot` package.

Including Covariates

Covariates can be included in predicting $M_i(t)$ and Y_i. For example, suppose there is a covariate Z_i which has a time-varying relationship to $M_i(t)$. The TVEM model for the mediator can be expanded to $E(M_i(t)) = \alpha_0(t)+\alpha_X(t)X_i+\alpha_Z(t)Z_i$ for numerical $M_i(t)$, or $\text{logit}^{-1}(E(M_i(t))) = \alpha_0(t)+\alpha_X(t)X_i+\alpha_Z(t)Z_i$ for binary $M_i(t)$. It is possible for Z_i to have a time-varying effect even if the values of Z_i do not vary over time. That would mean that the correlation between the observation-level $M_i(t)$ and the subject-level Z_i is different for different values of t. It is currently not recommended to use a time-varying covariate alongside the time-varying mediator in this package.

It is also possible to assume that the relationship between Z_i and $M_i(t)$ does not depend on t, whether or not Z_i depends on t; in this case $\alpha_Z(t)$ can simply be written as α_Z. The package allows both time-varying-effects and time-invariant-effects covariates to be specified in predicting the mediator, using the `tve_covariates_on_mediator` and `tie_covariates_on_mediator` arguments respectively.

Alternatively, suppose that there is a subject-level covariate S_i which predicts Y_i. This can be added to the functional regression model by specifying $E(Y_i) = \beta_0 + \beta_X X_i + \beta_S S_i + \int \beta_M(t)M_i(t)dt$ for numerical Y_i, or $\text{logit}^{-1}(E(Y_i)) = \beta_0 + \beta_X X_i + \beta_S S_i + \int \beta_M(t)M_i(t)dt$ for binary Y_i. Such a covariate can be included using the `covariates_on_outcome` argument in the package. Currently, the package assumes a subject-level Y_i and does not support multiple functional coefficients in predicting the outcome; that is, the covariates in `covariates_on_outcome` cannot have time-varying values or effects.

Example with Numerical Outcomes

The following example shows how to simulate example data and then analyze it using the `funmediation` package.

Getting ready to run the example

First we load the required packages.

```r
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.

```r
set.seed(123)
simulation1 <- simulate_funmediation_example(nsub=500)
```

The `simulation1` object will contain not only the simulated dataset itself, but the true values of the simulated parameters, including the indirect effect.

```r
str(simulation1)
#> List of 8
#>  $ time_grid     : num [1:100] 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ...
#>  $ true_alpha_int: num [1:100] 0.1 0.141 0.173 0.2 0.224 ...
#>  $ true_alpha_X  : num [1:100] -0.0707 -0.1 -0.1225 -0.1414 -0.1581 ...
#>  $ true_beta_int : num 0
#>  $ true_beta_M   : num [1:100] 0.00503 0.0101 0.01523 0.02041 0.02564 ...
#>  $ true_beta_X   : num 0.2
#>  $ true_indirect : num -0.211
#>  $ dataset       :'data.frame':  20169 obs. of  5 variables:
#>   ..$ subject_id: int [1:20169] 1 1 1 1 1 1 1 1 1 1 ...
#>   ..$ t          : num [1:20169] 0.04 0.07 0.11 0.15 0.18 0.19 0.25 0.28 0.29 0.31 ...
#>   ..$ X          : int [1:20169] 0 0 0 0 0 0 0 0 0 0 ...
#>   ..$ M          : num [1:20169] 1.165 -0.653 -1.462 2.287 2.271 ...
#>   ..$ Y          : num [1:20169] -0.181 -0.181 -0.181 -0.181 -0.181 ...
```

We need the simulated dataset as a data.frame object.

```
the_data <- simulation1$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
#> 1          1 0.04 0  1.1647240 -0.18087
#> 2          1 0.07 0 -0.6525125 -0.18087
#> 3          1 0.11 0 -1.4620807 -0.18087
#> 4          1 0.15 0  2.2871677 -0.18087
#> 5          1 0.18 0  2.2713836 -0.18087
#> 6          1 0.19 0  1.4810598 -0.18087
```

We can also look at univariate descriptive statistics:

```
summary(the_data)
#>    subject_id        t                X               M          
#>  Min.   :  1   Min.   :0.0100   Min.   :0.000   Min.   :-7.1295  
#>  1st Qu.:125   1st Qu.:0.2500   1st Qu.:0.000   1st Qu.:-0.9189  
#>  Median :251   Median :0.5000   Median :0.000   Median : 0.4338  
#>  Mean   :251   Mean   :0.5041   Mean   :0.472   Mean   : 0.4349  
#>  3rd Qu.:376   3rd Qu.:0.7600   3rd Qu.:1.000   3rd Qu.: 1.7870  
#>  Max.   :500   Max.   :1.0000   Max.   :1.000   Max.   : 7.7515  
#>        Y          
#>  Min.   :-2.9565  
#>  1st Qu.:-0.4219  
#>  Median : 0.2832  
#>  Mean   : 0.3322  
#>  3rd Qu.: 1.0521  
#>  Max.   : 3.7903
```

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 we call the functional mediation function. Only 10 bootstrap samples are used below for this quick illustration. At least a few hundred bootstraps are 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=10)
#> Ran original results.
#> Working on bootstrap results:
```

```
#> 1.2.3.4.5.6.7.8.9.10.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 warning message, “extreme order statistics used as endpoints,” occurs because too few bootstrap samples are used for the bootstrap confidence intervals to be valid; however, we ignore this here in order for the vignette example to run quickly.

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

```
print(model1)
#> ========================================================
#> Functional Regression Mediation Function Output
#> ========================================================
#> TREATMENT: X (Assumed Binary)
#> MEDIATOR: M (Assumed Numeric)
#> OUTCOME: Y (Assumed Numeric)
#> ========================================================
#> Indirect effect bootstrap estimate:
#> -0.248064
#> Indirect effect bootstrap confidence interval:
#> ... by normal method:
#> -0.3536 ,  -0.1425
#> ... by percentile method:
#> -0.3325 ,  -0.1519
#> Computation time:
#> Time difference of 30.18501 secs
#> ========================================================
#> Time-varying Effects Model Predicting MEDIATOR from TREATMENT:
#> Response variable:   MEDIATOR
#> Time interval:   0.01 to 1
#> Number of subjects:  500
#> 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: 84897.63
#> Pseudolikelihood BIC: 84937.51
#>
#> ========================================================
#> Functional Regression Model Predicting OUTCOME from MEDIATOR
#>   and TREATMENT:
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> OUTCOME ~ s(x = MEDIATOR.tmat, by = L.MEDIATOR) + TREATMENT
#>
```

```
#> Estimated degrees of freedom:
#> 2  total = 4
#>
#> REML score: 742.266
#> Scalar terms in functional regression model:
#> (Intercept)   TREATMENT
#>  0.16499534 -0.07192656
#> =======================================================
#> Parametric model for Predicting OUTCOME from TREATMENT
#>   ignoring MEDIATOR:
#>
#> Call:
#> glm(formula = glm_formula)
#>
#> Deviance Residuals:
#>     Min       1Q   Median       3Q      Max
#> -3.4298  -0.7068   0.0100   0.7185   3.6232
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.47332    0.06671   7.096 4.45e-12 ***
#> TREATMENT   -0.30617    0.09730  -3.147  0.00175 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 1.179171)
#>
#>     Null deviance: 598.90  on 499  degrees of freedom
#> Residual deviance: 587.23  on 498  degrees of freedom
#> AIC: 1505.3
#>
#> Number of Fisher Scoring iterations: 2
#>
#> =======================================================
```

The first part of the output above summarizes the estimated indirect effect and its bootstrap confidence interval. The estimate for the indirect effect $\int \alpha_X(t)\beta_M(t)$ is given as -0.25. Choosing the wider confidence interval in order to be conservative, a 95% confidence interval extends from about -0.35 to -0.14. (This interval would not actually be reliable in practice because of the low number of bootstrap samples, which could be remedied by choosing a higher `nboot`). Thus, there appears to be a significant negative indirect effect of X on Y mediated through $M(t)$.

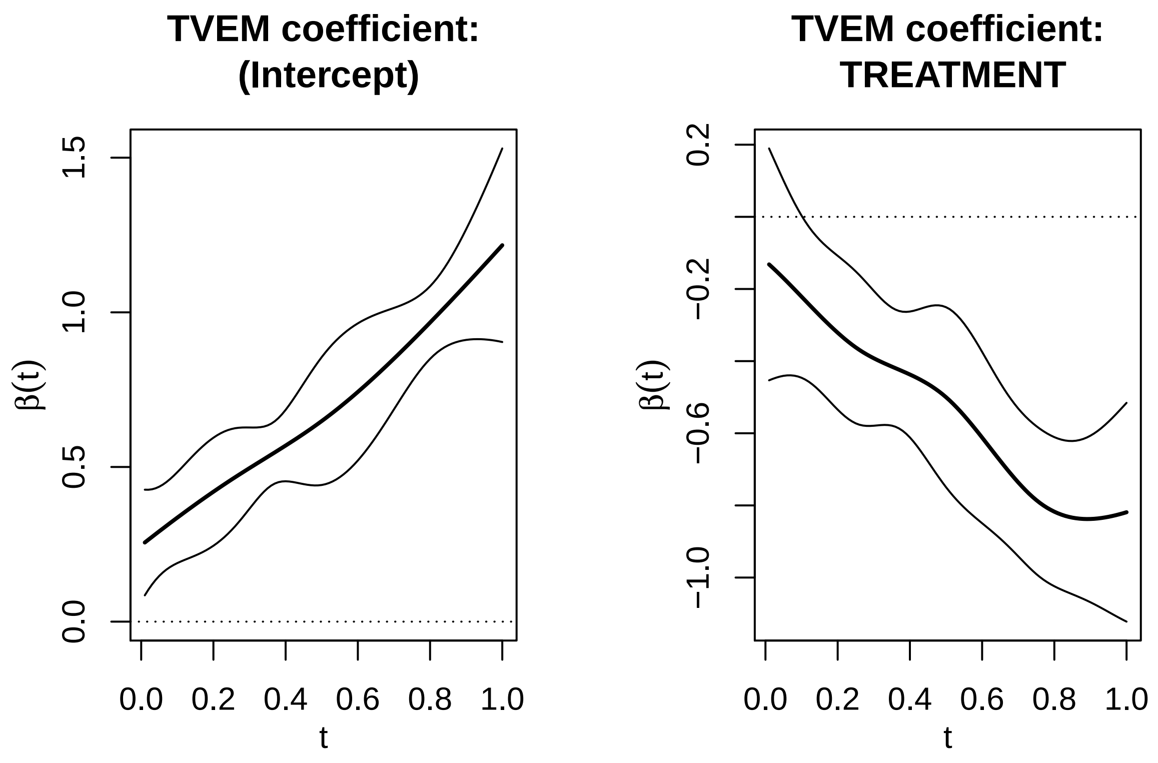
The second part of the printed output summarizes the TVEM model used to predict $M(t)$ from X. It only provides summary information on the model that was fit. The time-varying coefficients are not actually printed, because they are functions rather than single numbers. They can be plotted using the `plot` function, as described later. They are also stored in the `model1` output object, as described later.

The third part summarizes the scalar-on-function functional regression model used to predict Y from X and $M(t)$. This model involves scalar (non-time-varying) coefficients for the intercept and the direct effect of X, and a functional (time-varying) coefficient for the effect of (t). The scalar coefficients are printed, but as before, the functional coefficient needs to be plotted using the `plot` function.

The fourth section of the printed output simply presents an estimate of the total effect of X on Y ignoring $M(t)$, obtained using the `glm` function. X is shown to have a total negative effect on Y, with an estimated coefficient of `-0.30617` and p-value of `0.00175`. This suggests that the $X = 1$ group has statistically significantly lower average Y than the $X = 0$ group, regardless of whether or not this has anything to do with M.

The `plot` function plots the results. Three kinds of plots are available, represented by the options `tvem`, `pfr`, and `coef`. The "tvem" plot shows the estimates for $\alpha_0(t)$ and $\alpha_1(t)$.

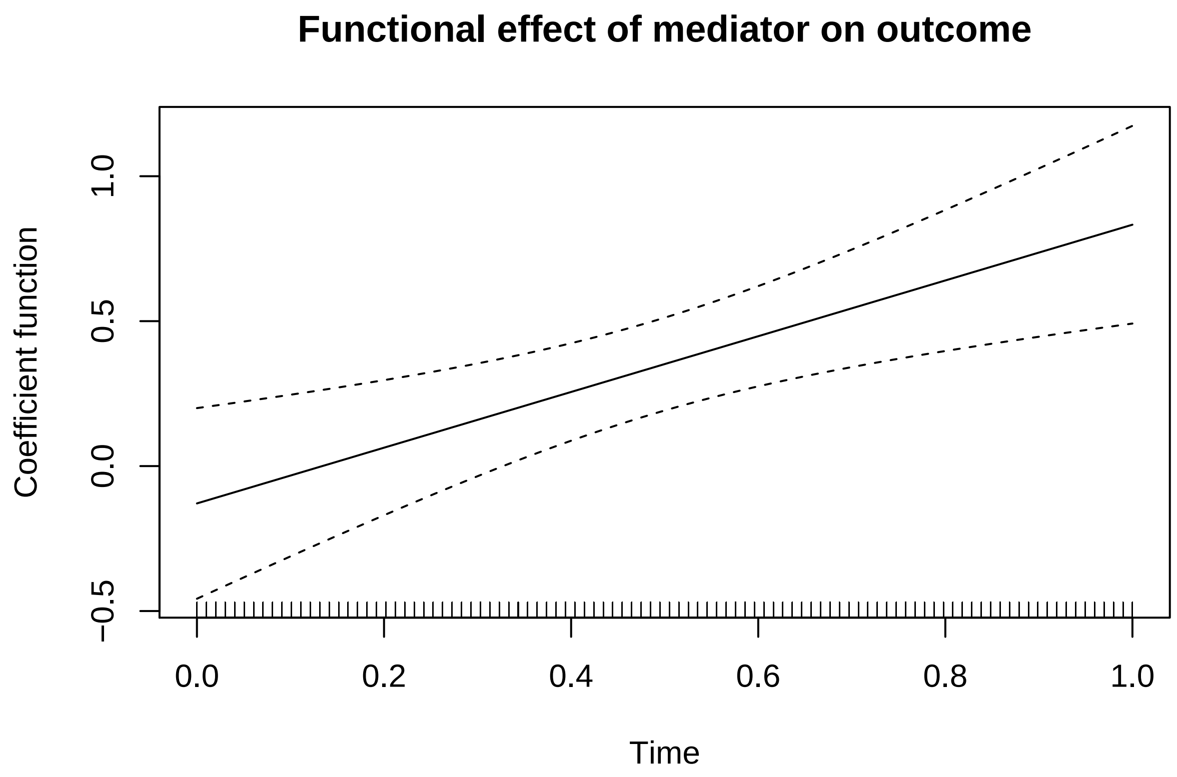
```
plot(model1, what_plot="tvem")
```



The intercept plot appears to be significantly above zero and significantly increasing. It can be interpreted as the mean $M(t)$ for the control $(X = 0)$ group because $E(M(t)|X = 0) = \alpha_0(t) + \alpha_X(t) \times 0 = \alpha_0(t)$. The treatment effect plot is generally nonzero and decreasing. It can be interpreted as $E(M(t)|X = 1) - E(M(t)|X = 0)$, a kind of time-specific treatment effect on the mediator, and it suggests that as time goes on the $X = 1$ group tends to become lower in average $M(t)$ than the $X = 0$ group.

The `pfr` plot shows the estimate for $\beta_M(t)$.

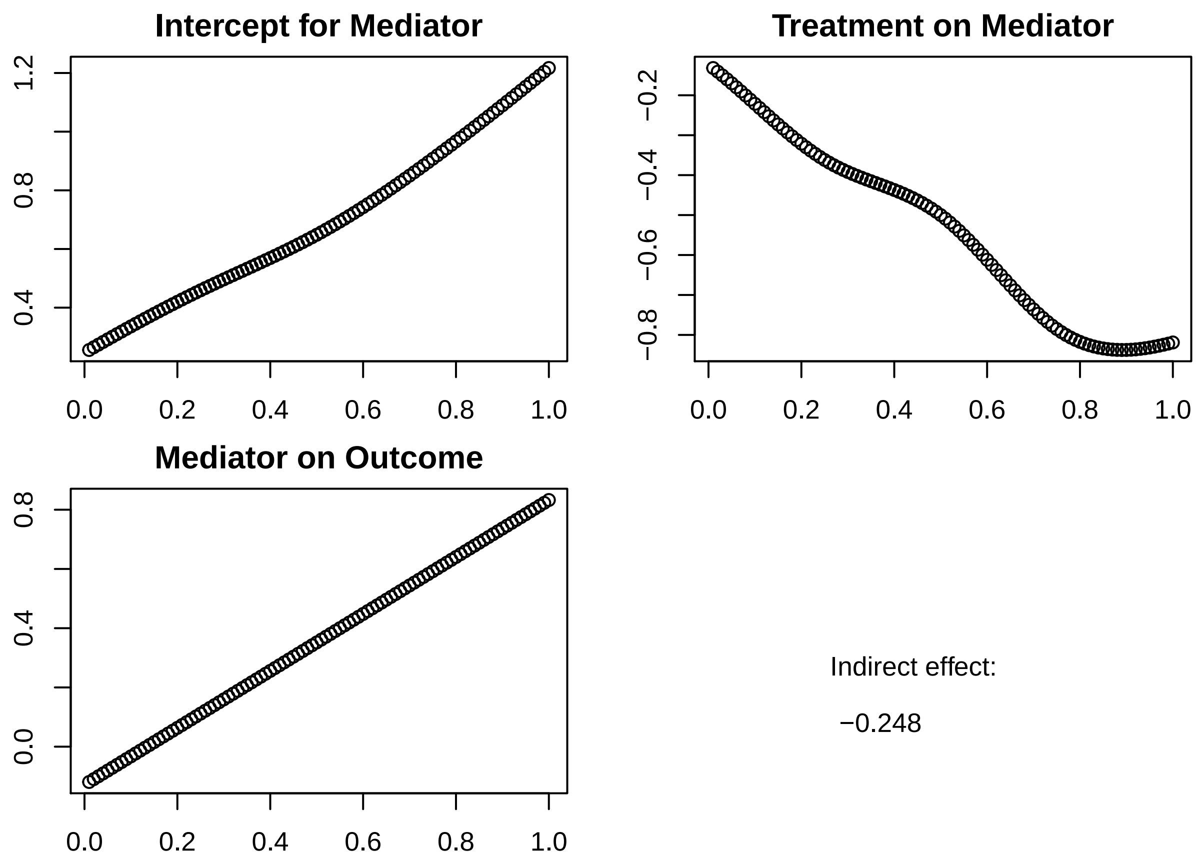
```
plot(model1, what_plot="pfr")
```



The function tends to be nonzero at least for higher values of time, and generally seems to be steadily increasing. One possible interpretation for this is that individuals who increase more (or decrease less) on average over time on the mediator M tend to have a higher value on the outcome Y.

Last, the `coef` plot summarizes all of the most important coefficients estimated in the model. The upper left and upper right panes show the estimates of $\alpha_0(t)$ and $\alpha_X(t)$ from the model of the effect of X on $M(t)$. The lower left shows the estimate of $\beta_M(t)$ from the model of the effect of $M(t)$ on Y given X. In the lower right panel, the estimate of the indirect is printed (not plotted, because it is a single number).

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



Because we have already looked at the relevant plots before using the other two options, there is nothing new to interpret in the `coef` plot.

Exploring Information in the Output Objects

The output model created, here called `model1`, also contains useful information both on the model fit to the original data, and on the bootstrapping results.

- `model1$original_results$time_grid` is the grid of time points on which the coefficient functions were estimated.
- `model1$original_results$alpha_int_estimate`, `model1$original_results$alpha_int_se`, `model1$original_results$alpha_X_estimate`, and `model1$original_results$alpha_X_se` are the functional coefficients and pointwise standard errors for the intercept and treatment in the TVEM model, at each point of the time grid.
- `model1$original_results$beta_int_estimate`, `model1$original_results$beta_int_se`, `model1$original_results$beta_X_estimate` and `model1$original_results$beta_X_se` are the estimates and standard errors for the scalar coefficients representing the intercept and the direct treatment effect in the scalar-outcome functional linear model.
- `model1$original_results$beta_M_estimate` and `model1$original_results$beta_M_se` are the estimates and pointwise standard errors representing the effect of the time-varying mediator at each point of the time grid.
- `beta_M_pvalue` is the p-value for an overall test of whether the mediator is significantly associated with the outcome, i.e., whether $\beta_M(t)$ can be shown not to be all zeroes; see the documentation for the `pfr` function in the `refund` package for more information.

- `model1$original_results$tau_int_estimate`, together with `$tau_int_se`, `$tau_X_estimate`, and `$tau_X_se`, represent the estimates of the intercept and effect coefficient in the model for the direct effect of X on Y. Thus, `model1$original_results$tau_X_estimate` is the estimated direct effect. Also, `model1$original_results$tau_X_pvalue` is the p-value for testing whether this effect is zero.
- `model1$original_results$indirect_effect_estimate` gives the estimated value of $\int \alpha_X(t)\beta_M(t)$ obtained without bootstrapping.
- `model1$original_results$tvem_XM_details`, `$funreg_MY_details`, and `$direct_effect_details` give the fitted model objects from `tvem`, `pfr` and `glm`. They might be useful for plots or diagnostics but might contain more technical detail than some users are interested in.

Including a Covariate

It is possible to run the model with one or more additional covariates. To demonstrate this, the `simulate_mediation_example` has an option `make_covariate_S` to create an extra subject-level covariate. The data is generated in such a way that S is not actually related to X, M or Y, but the analyst is assumed not to know this – that is, the simulated true value of its coefficient is zero.

```
simulation_with_covariate <- simulate_funmediation_example(nsub=500,
                                                           make_covariate_S=TRUE);
data_with_covariate <- simulation_with_covariate$dataset;
```

To include the covariate S_i with an assumption of a possibly time-varying effect $\alpha_S(t)$ on $M(t)$ (recall that the covariate can have a time-varying effect even though it cannot have a time-varying value itself), you can use code like:

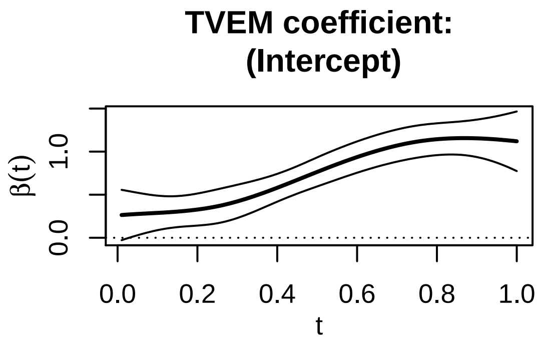
```
model_with_tve_covariate <- funmediation(data=data_with_covariate,
                        treatment=X,
                        mediator=M,
                        outcome=Y,
                        tve_covariates_on_mediator = ~S,
                        covariates_on_outcome = ~S,
                        id=subject_id,
                        time=t,
                        nboot=10)
#> Ran original results.
#> Working on bootstrap results:
#> 1.2.3.4.5.6.7.8.9.10.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
```

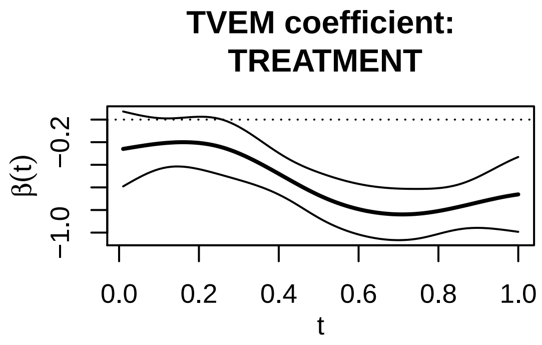
Notice that the tilde (~) character needs to be included for the covariates. This is done so that R will allow multiple covariates to be listed in the style of the right side of a formula, e.g., `~S1 + S2`.

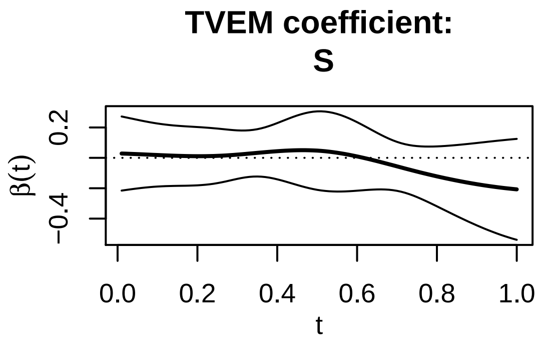
The covariates in `tve_covariates_on_mediator` and `covariates_on_outcome` need to be subject-level and should not have time-varying values; the function does not currently support models with time-varying variables other than the mediator. However, the relationship of the covariates to the mediator can be specified as either time-varying or non-time-varying. The relationship of the covariate to the outcome, in contrast, is assumed not to be time varying, at least in the current version of the package. That is, $M(t)$ is the only time-varying predictor in the functional regression predicting the outcome.

The results of the model fit can be viewed in the usual way:

```
print(model_with_tve_covariate)
#> =======================================================
#> Functional Regression Mediation Function Output
#> =======================================================
#> TREATMENT: X (Assumed Binary)
#> MEDIATOR: M (Assumed Numeric)
#> OUTCOME: Y (Assumed Numeric)
#> =======================================================
#> Indirect effect bootstrap estimate:
#> -0.2944332
#> Indirect effect bootstrap confidence interval:
#> ... by normal method:
#> -0.3681 ,  -0.2207
#> ... by percentile method:
#> -0.3694 ,  -0.2542
#> Computation time:
#> Time difference of 28.61007 secs
#> =======================================================
#> Time-varying Effects Model Predicting MEDIATOR from TREATMENT:
#> Response variable:   MEDIATOR
#> Time interval:   0.01 to 1
#> Number of subjects:  500
#> Effects specified as time-varying:  (Intercept), TREATMENT, S
#> You can use the plot function to view their plots.
#>
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> MEDIATOR ~ TREATMENT + S + 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) + s(time, bs = "ps", by = S,
#>     pc = 0, m = c(2, 1), k = 7, fx = FALSE)
#> Pseudolikelihood AIC: 83986.37
#> Pseudolikelihood BIC: 84047.99
#>
#> =======================================================
#> Functional Regression Model Predicting OUTCOME from MEDIATOR
#>   and TREATMENT:
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> OUTCOME ~ s(x = MEDIATOR.tmat, by = L.MEDIATOR) + TREATMENT +
#>     S
#>
#> Estimated degrees of freedom:
#> 2  total = 5
#>
#> REML score: 729.0973
#> Scalar terms in functional regression model:
#>  (Intercept)    TREATMENT            S
#> -0.025787118  0.112576286 -0.001643264
#> =======================================================
#> Parametric model for Predicting OUTCOME from TREATMENT
#>   ignoring MEDIATOR:
#>
#> Call:
#> glm(formula = glm_formula)
#>
#> Deviance Residuals:
#>      Min        1Q    Median        3Q       Max
#> -3.02647  -0.72182  -0.00592   0.67548   2.95742
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.36962    0.08331   4.437 1.12e-05 ***
#> TREATMENT   -0.16672    0.09601  -1.737   0.0831 .
#> S           -0.02827    0.09593  -0.295   0.7684
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 1.149205)
#>
#>     Null deviance: 574.74  on 499  degrees of freedom
#> Residual deviance: 571.16  on 497  degrees of freedom
#> AIC: 1493.5
#>
#> Number of Fisher Scoring iterations: 2
#>
#> =======================================================
plot(model_with_tve_covariate, what_plot="tvem")
```







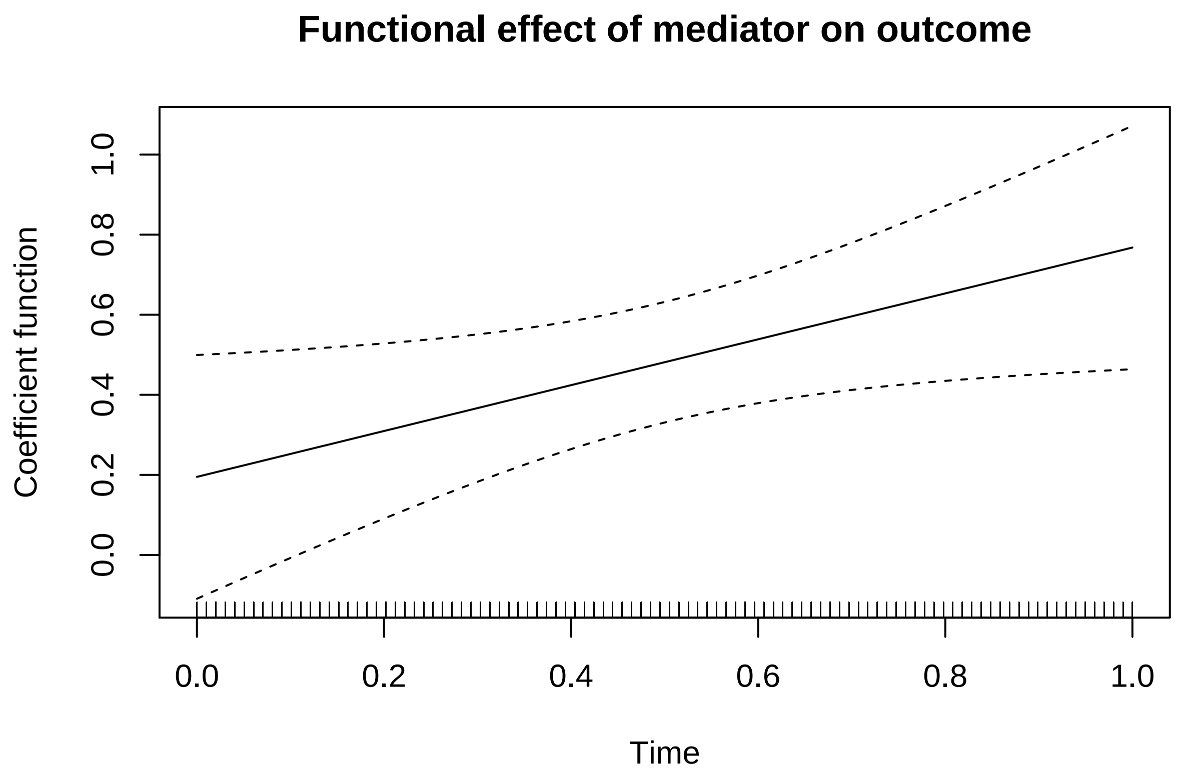
Notice that we now get a time-varying effects plot which includes a coefficient for the effect of S also, in addition to the coefficients for the effect of X. Note that the "effect" of S may not be an effect in the causal sense if S is not randomized.

In this example $\alpha_S(t)$ is not significantly nonzero at any time and does not change over time.

Alternatively, by specifying `tie_covariates_on_mediator` instead of `tve_covariates_on_mediator`, the covariate can be assumed to have time-invariant effects rather than time-varying effects.

To include the covariate with an assumption of a non-time-varying effect on M, you can use code like:

```
model_with_tie_covariate <- funmediation(data=data_with_covariate,
                         treatment=X,
                         mediator=M,
                         outcome=Y,
                         tie_covariates_on_mediator = ~S,
                         covariates_on_outcome = ~S,
                         id=subject_id,
                         time=t,
                         nboot=10)
#> Ran original results.
#> Working on bootstrap results:
#> 1.2.3.4.5.6.7.8.9.10.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(model_with_tie_covariate)
#> =======================================================
#> Functional Regression Mediation Function Output
#> ==========================================================
#> TREATMENT: X (Assumed Binary)
#> MEDIATOR: M (Assumed Numeric)
#> OUTCOME: Y (Assumed Numeric)
#> ==========================================================
#> Indirect effect bootstrap estimate:
#> -0.2689879
#> Indirect effect bootstrap confidence interval:
#> ... by normal method:
#> -0.3432 ,  -0.1948
#> ... by percentile method:
#> -0.3726 ,  -0.2741
#> Computation time:
#> Time difference of 26.67535 secs
#> ==========================================================
#> Time-varying Effects Model Predicting MEDIATOR from TREATMENT:
#> Response variable:   MEDIATOR
#> Time interval:   0.01 to 1
#> Number of subjects:  500
#> Effects specified as time-varying:  (Intercept), TREATMENT
#> You can use the plot function to view their plots.
#>
#> Effects specified as non-time-varying:
#>      estimate standard_error
#> S -0.03008131     0.05790221
#>
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> MEDIATOR ~ TREATMENT + S + 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: 83993.2
#> Pseudolikelihood BIC: 84042.46
#>
#> ==========================================================
#> Functional Regression Model Predicting OUTCOME from MEDIATOR
#>   and TREATMENT:
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
#> OUTCOME ~ s(x = MEDIATOR.tmat, by = L.MEDIATOR) + TREATMENT +
#>     S
#>
#> Estimated degrees of freedom:
#> 2  total = 5
#>
#> REML score: 729.0973
#> Scalar terms in functional regression model:
#>  (Intercept)    TREATMENT            S
#> -0.025787118  0.112576286 -0.001643264
#> =======================================================
#> Parametric model for Predicting OUTCOME from TREATMENT
#>   ignoring MEDIATOR:
#>
#> Call:
#> glm(formula = glm_formula)
#>
#> Deviance Residuals:
#>      Min        1Q    Median        3Q       Max
#> -3.02647  -0.72182  -0.00592   0.67548   2.95742
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.36962    0.08331   4.437 1.12e-05 ***
#> TREATMENT   -0.16672    0.09601  -1.737   0.0831 .
#> S           -0.02827    0.09593  -0.295   0.7684
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 1.149205)
#>
#>     Null deviance: 574.74  on 499  degrees of freedom
#> Residual deviance: 571.16  on 497  degrees of freedom
#> AIC: 1493.5
#>
#> Number of Fisher Scoring iterations: 2
#>
#> =======================================================
plot(model_with_tie_covariate, what_plot="pfr")
```



Now there are numerical estimates for $\alpha_S = -0.03008$ and $\beta_S = -0.00164$ but no plot for $\alpha_S(t)$ anymore. Again, S is not significant here either as a predictor of the mediator or of the outcome, because the coefficients are not large relative to their standard errors.

Using a Binary Mediator

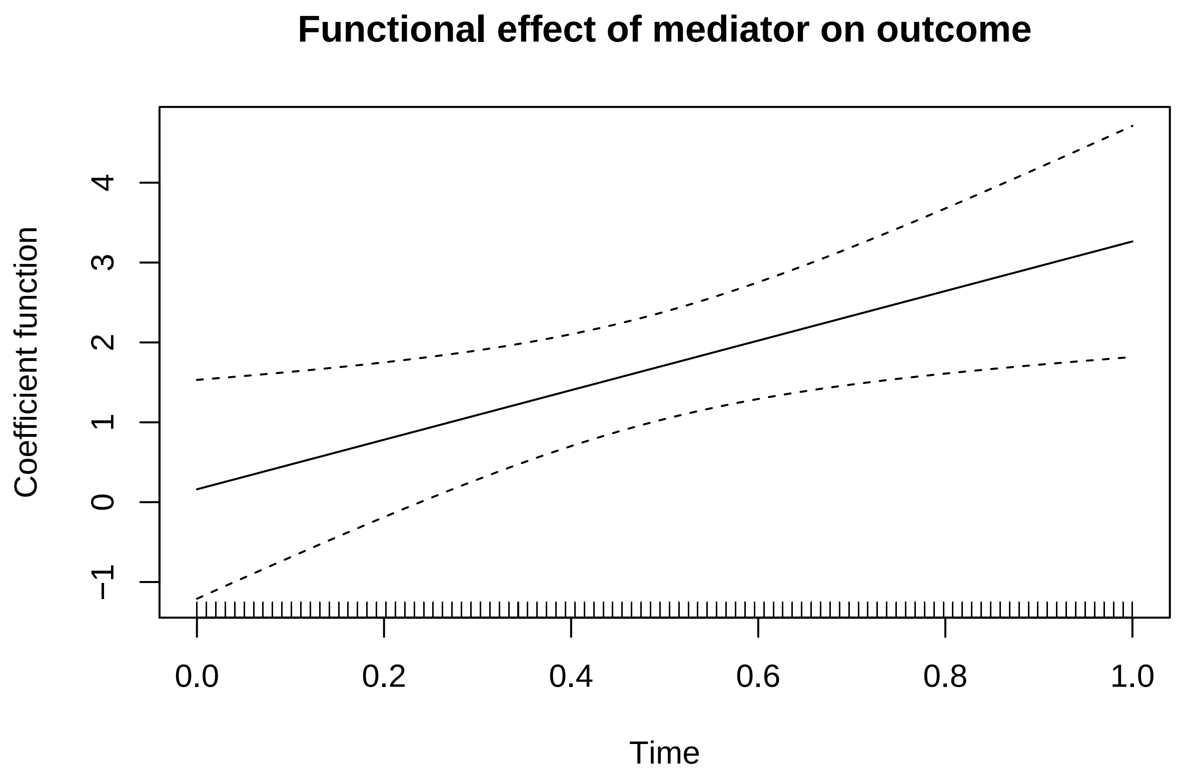
If desired, you can use a binary mediator instead of a numerical mediator, and can specify that the TVEM part of the model use the logistic link function. The easiest way to demonstrate this for purposes of this vignette (although it usually would not be advisable for real data because it involves loss of information) is to dichotomize M in the simulated dataset and use that as our binary mediator example.

```
set.seed(123)
data_with_covariate$binary_M <- 1*(data_with_covariate$M > mean(data_with_covariate$M))
head(data_with_covariate)
#>   subject_id    t X          M        Y S binary_M
#> 1          1 0.01 1 -2.9541303 -0.44206 0        0
#> 2          1 0.05 1 -0.5834873 -0.44206 0        0
#> 3          1 0.06 1  0.6919501 -0.44206 0        1
#> 4          1 0.08 1  0.4411048 -0.44206 0        1
#> 5          1 0.15 1 -1.2007641 -0.44206 0        0
#> 6          1 0.17 1 -1.0369571 -0.44206 0        0
```

Now we can use the code:

```
model_with_binary_M <- funmediation(data=data_with_covariate,
                             treatment=X,
                             mediator=binary_M,
                              outcome=Y,
                              id=subject_id,
                              time=t,
                              binary_mediator=TRUE,
                              nboot=10)
#> Ran original results.
#> Working on bootstrap results:
#> 1.2.3.4.5.6.7.8.9.10.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(model_with_binary_M);
#> ==========================================================
#> Functional Regression Mediation Function Output
#> ==========================================================
#> TREATMENT: X (Assumed Binary)
#> MEDIATOR: binary_M (Assumed Binary)
#> OUTCOME: Y (Assumed Numeric)
#> ==========================================================
#> Indirect effect bootstrap estimate:
#> -0.2190034
#> Indirect effect bootstrap confidence interval:
#> ... by normal method:
#> -0.3038 ,  -0.1342
#> ... by percentile method:
#> -0.2506 ,  -0.1298
#> Computation time:
#> Time difference of 35.62272 secs
#> ==========================================================
#> Time-varying Effects Model Predicting MEDIATOR from TREATMENT:
#> Response variable:   MEDIATOR
#> Time interval:   0.01 to 1
#> Number of subjects:  500
#> 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: 27100.7
#> Pseudolikelihood BIC: 27134.79
#>
#> ==========================================================
#> Functional Regression Model Predicting OUTCOME from MEDIATOR
#>   and TREATMENT:
#>
#> Family: gaussian
#> Link function: identity
#>
#> Formula:
```

```
#> OUTCOME ~ s(x = MEDIATOR.tmat, by = L.MEDIATOR) + TREATMENT
#>
#> Estimated degrees of freedom:
#> 2  total = 4
#>
#> REML score: 731.6248
#> Scalar terms in functional regression model:
#> (Intercept)   TREATMENT
#> -0.62912300  0.03910025
#> =========================================================
#> Parametric model for Predicting OUTCOME from TREATMENT
#>   ignoring MEDIATOR:
#>
#> Call:
#> glm(formula = glm_formula)
#>
#> Deviance Residuals:
#>      Min        1Q    Median        3Q       Max
#> -3.01293  -0.71639  -0.01174   0.68195   2.97096
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  0.35608    0.06942   5.129 4.18e-07 ***
#> TREATMENT   -0.16720    0.09591  -1.743   0.0819 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 1.147098)
#>
#>     Null deviance: 574.74  on 499  degrees of freedom
#> Residual deviance: 571.25  on 498  degrees of freedom
#> AIC: 1491.6
#>
#> Number of Fisher Scoring iterations: 2
#>
#> =========================================================
plot(model_with_binary_M,type="coef")
```



Note that if you only included `binary_M` as the mediator without specifying the `binary_mediator` argument, the TVEM model would be fit using an identity link (hence a linear model at any given time t) as for a normally distributed outcome, even though in this instance `binary_M` happens to consist of zeroes and ones and no other numbers. The function would not detect that `binary_M` was binary. The identity link is probably not the best approach but might be an interesting comparison. If the mediator is binary, it is probably better to use `binary_M=TRUE` in order to specify that the α coefficients are to be interpreted on the logit (logistic regression) scale. A future version of this package might allow log link functions as an alternative to logit link functions, but this is currently not available.

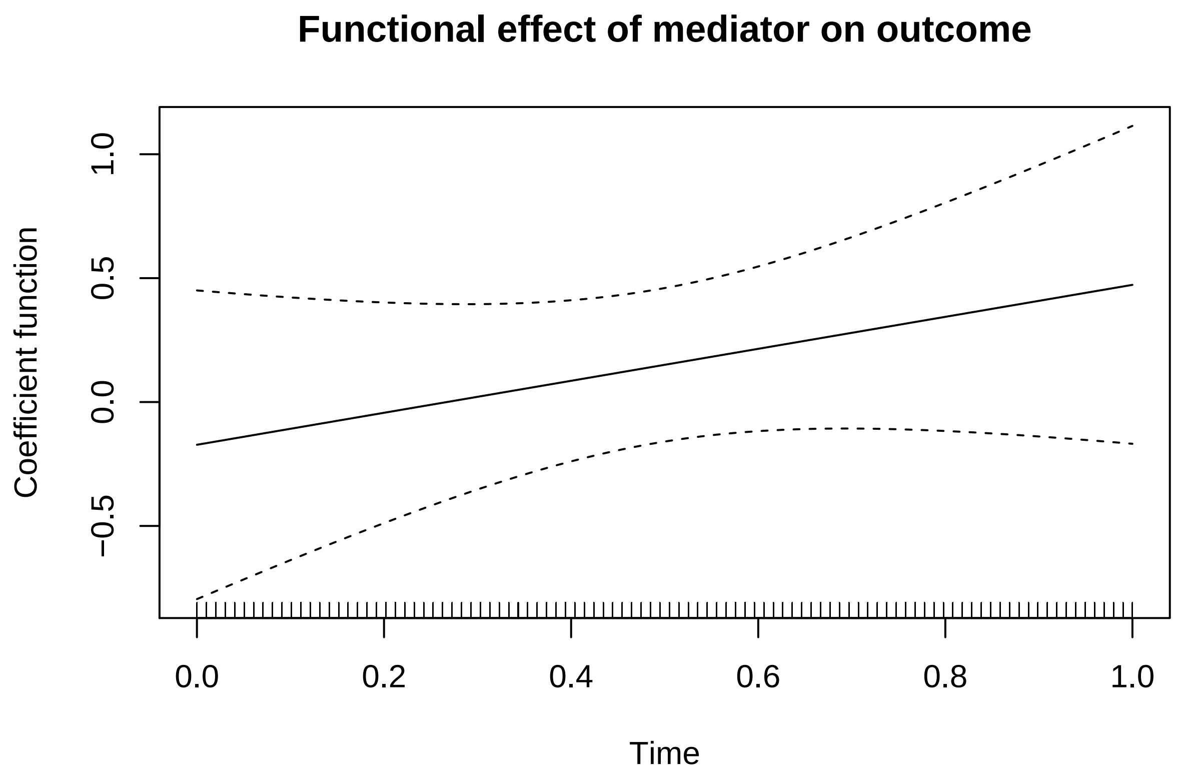
Example with a Binary Outcome

Instead of, or in addition to, using a binary mediator, you can also use a binary outcome. In this case, the scalar-on-function part of the model (the β_0, β_X and $\beta_M(t)$ functions) would be fit using a logistic link function. To demonstrate this, first simulate a practice dataset with a binary outcome, using the `simulate_binary_Y` option in `simulate_funmediation_example`. We set this option to true (on) now, although it was false (off) by default. (Note that we do not currently have a corresponding `simulate_binary_M` option in this function, as simulating realistic binary data trajectories is somewhat more difficult.)

```
set.seed(123)
simulation_with_binary_Y <- simulate_funmediation_example(simulate_binary_Y=TRUE, nsub=500)
data_with_binary_Y <- simulation_with_binary_Y$dataset
head(data_with_binary_Y)
#>   subject_id    t X          M Y
#> 1          1 0.01 0 -0.6512057 1
#> 2          1 0.02 0  1.3865730 1
#> 3          1 0.05 0 -0.5086171 1
#> 4           1 0.09 0 -1.4154640 1
#> 5           1 0.10 0 -1.2526154 1
#> 6           1 0.12 0  0.9287293 1
```

Now fit the model, using the new dataset with binary Y, and using the `binary_outcome` option. By default, `binary_outcome` is false, but here we set it to true in order to use this approach. The `binary_outcome` option indicates that the β coefficients are to be interpreted as logistic regression coefficients, similar to the `binary_mediator` option which does the same for the α coefficients.

```
model_with_binary_Y <- funmediation(data=data_with_binary_Y,
                                    treatment=X,
                                    mediator=M,
                                    outcome=Y,
                                    id=subject_id,
                                    time=t,
                                    binary_outcome=TRUE,
                                    nboot=10)
#> Ran original results.
#> Working on bootstrap results:
#> 1.2.3.4.5.6.7.8.9.10.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(model_with_binary_Y);
#> =======================================================
#> Functional Regression Mediation Function Output
#> =======================================================
#> TREATMENT: X (Assumed Binary)
#> MEDIATOR: M (Assumed Numeric)
#> OUTCOME: Y (Assumed Binary)
#> =======================================================
#> Indirect effect bootstrap estimate:
#> -0.1519281
#> Indirect effect bootstrap confidence interval:
#> ... by normal method:
#> -0.2643 ,  -0.0396
#> ... by percentile method:
#> -0.1732 ,  0.0089
#> Computation time:
#> Time difference of 39.64473 secs
#> =======================================================
#> Time-varying Effects Model Predicting MEDIATOR from TREATMENT:
#> Response variable:   MEDIATOR
#> Time interval:   0.01 to 1
#> Number of subjects:  500
#> 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: 84969.94
#> Pseudolikelihood BIC: 85003.97
#>
#> ==========================================================
#> Functional Regression Model Predicting OUTCOME from MEDIATOR
#>   and TREATMENT:
#>
#> Family: binomial
#> Link function: logit
#>
#> Formula:
#> OUTCOME ~ s(x = MEDIATOR.tmat, by = L.MEDIATOR) + TREATMENT
#>
#> Estimated degrees of freedom:
#> 2  total = 4
#>
#> REML score: 347.7077
#> Scalar terms in functional regression model:
#> (Intercept)   TREATMENT
#> -0.01912531  0.20302530
#> ==========================================================
#> Parametric model for Predicting OUTCOME from TREATMENT
#>   ignoring MEDIATOR:
#>
#> Call:
#> glm(formula = glm_formula, family = binomial)
#>
#> Deviance Residuals:
#>    Min      1Q  Median      3Q     Max
#> -1.269  -1.232   1.088   1.123   1.123
#>
#> Coefficients:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   0.1285     0.1231   1.044    0.297
#> TREATMENT     0.0851     0.1799   0.473    0.636
#>
#> (Dispersion parameter for binomial family taken to be 1)
#>
#>     Null deviance: 689.62  on 499  degrees of freedom
#> Residual deviance: 689.39  on 498  degrees of freedom
#> AIC: 693.39
#>
#> Number of Fisher Scoring iterations: 3
#>
#> ==========================================================
plot(model_with_binary_Y,type="coef")
```



Future work relative to this package may involve comparing potential significance tests for different parts of the model, in terms of power and coverage, as well as elucidating the assumptions required to estimate each part of the model causally.

References

- Goldsmith, J., Bobb, J., Crainiceanu, C., Caffo, B., and Reich, D. (2011). Penalized functional regression. Journal of Computational and Graphical Statistics, 20(4), 830-851.
- Hastie, T, Tibshirani, R. (1993). Varying-coefficient models. Journal of the Royal Statistical Socety, B, 55:757-796.
- Lindquist, M. A. (2012). Functional Causal Mediation Analysis With an Application to Brain Connectivity. Journal of the American Statistical Association, 107: 1297-1309.
- Ramsay, J. O., & Silverman, B. W. (2005). Functional data analysis (2nd ed.). New York: Springer.
- Tan, X., Shiyko, M. P., Li, R., Li, Y., & Dierker, L. (2012). A time-varying effect model for intensive longitudinal data. Psychological Methods, 17: 61-77.
- Wood, S.N. (2017) Generalized Additive Models: an introduction with R (2nd edition), CRC.

Acknowledgements

The `pfr` function was developed by Jonathan Gellar, Mathew W. McLean, Jeff Goldsmith, and Fabian Scheipl, the `boot` package was written by Angelo Canty and Brian Ripley, the `mgcv` package was developed by Simon N. Wood, and the `refund` package was developed and maintained by Julia Wrobel and others.