Fitting time-varying effects models

John J. Dziak

2021-01-29

In this example we simulate a longitudinal dataset and fit a simple time-varying coefficient model to it using the `tvem` package. We show that this can be done for either a continuous or a binary outcome variable. Time-varying coefficient models are discussed further by Tan, Shiyko, Li, Li and Dierker (2012). They are an application of the varying-coefficient models approach of Hastie and Tibshirani (1993) to intensive longitudinal data.

Before running the examples, first install and load the `tvem` package.
A `.zip` or `.tar.gz` file containing the package is available at
https://github.com/dziakj1/TvemPackage_development, and it can then be used with the `install.packages()` function in R code, `Packages > Install Package(s)` from Local Files in the R graphical user interface, or `Tools > Install Packages` in the RStudio application, to install the package. The package is also available at https://github.com/dziakj1/tvem for use with the `install_github` function. We also intend to submit this package to CRAN, from where it would then be able to be installed directly. Of course, if you are viewing this guide from within R using the `vignette()` function, then the package is already installed.

After you install the package on your system, you can then load it as usual with the `library()` function.

```
library(tvem)
#> Loading required package: mgcv
#> Loading required package: nlme
#> This is mgcv 1.8-31. For overview type 'help("mgcv-package")'.
```

Example with a continuous outcome variable

The `tvem` package has a function for simulating a dataset. It is good to start by specifying a random seed.

```
set.seed(123)
the_data <- simulate_tvem_example()
```

Exploring the dataset

When analyzing any dataset, it is important to examine it descriptively first.

```
print(head(the_data))
#>   subject_id time  x1  x2   y
#> 1          1 0.00 5.3 5.8 0.9
#> 2          1 0.05 6.4 5.3 0.0
#> 3          1 0.10 5.4 5.5  NA
#> 4          1 0.15 5.7 3.2  NA
#> 5          1 0.20 5.8 2.8  NA
#> 6          1 0.25 8.7 3.2 1.9
print(summary(the_data))
#>    subject_id          time             x1              x2
#>  Min.   :  1.00   Min.   :0.00   Min.   : 0.00   Min.   : 0.000
#>  1st Qu.: 75.75   1st Qu.:1.75   1st Qu.: 3.60   1st Qu.: 1.900
#>  Median :150.50   Median :3.50   Median : 5.00   Median : 3.300
#>  Mean   :150.50   Mean   :3.50   Mean   : 5.02   Mean   : 3.326
#>  3rd Qu.:225.25   3rd Qu.:5.25   3rd Qu.: 6.40   3rd Qu.: 4.700
#>  Max.   :300.00   Max.   :7.00   Max.   :10.00   Max.   :10.000
#>
#>        y
#>  Min.   :0.000
#>  1st Qu.:1.000
#>  Median :2.200
#>  Mean   :2.308
#>  3rd Qu.:3.400
#>  Max.   :9.600
#>  NA's   :29647
```

The dataset is in long form (one row per observation, with multiple observation times for each participant). There are 300 participants. The observation time ranges from 0 (thought of as the beginning of an intensive study) to 7 (imagined as the end of the study seven days later). There is a single response variable y, and two predictor variables (covariates), x_1 and x_2. In the context of an intensive longitudinal study with human participants, these variables might be ratings of different feelings, symptoms, or behaviors, reported a few times per day at random times, during seven days following an event (such as the beginning of an intervention, treatment, lifestyle change, etc.). The values of the covariates and the response vary over time within subject. The analyst wishes to find out whether their means change systematically over time, whether they are interrelated, and whether this relationship, if it exists, changes over time.

Plotting average change over time

One easy thing to do is to investigate whether and how the response changes over time on average. This is simply curve fitting, similar to polynomial regression, but can be fit using the TVEM function, in an approach sometimes called 'intercept-only TVEM.' This approach uses a spline function to approximate the average change in y over time.

By default, the tvem function will fit a penalized B-spline (de Boor, 1972), which is called a "P-spline" in the terminology of Eilers and Marx (1996). This approach uses an automatic tuning penalty to choose the level of smoothness versus flexibility of the fitted function. It is similar, though not identical, to the P-splines used in the Methodology Center's `%TVEM` macro for the SAS programming language. The P-splines used by the `%TVEM` macro are penalized truncated power splines (Li et al., 2017; see Ruppert, Wand & Carroll, 2003).

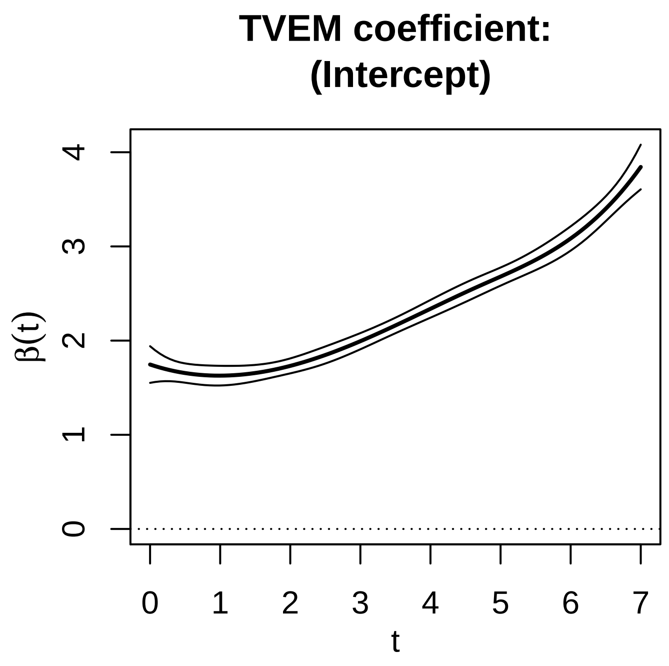
```
model1 <- tvem(data=the_data,
               formula=y~1,
               id=subject_id,
               time=time)
```

You also have the option to turn off the penalty and control the smoothness yourself, by specifying the number of interior knots, here 2.

```
model1_2knots_unpenalized <- tvem(data=the_data,
               formula=y~1,
               id=subject_id,
               num_knots=2,
               penalize=FALSE,
               time=time)
```

The implied mean model is $E(y|t) = \beta_0(t)$ where t is time in days. After fitting a model, you can print a summary and plot the estimated coefficient.

```
print(model1_2knots_unpenalized)
#> =======================================================
#> Time-Varying Effects Modeling (TVEM) Function Output
#> =======================================================
#> Response variable:   y
#> Response outcome distribution type: gaussian
#> Time interval:   0 to 7
#> Number of subjects:  300
#> Effects specified as time-varying:  (Intercept)
#> You can use the plot function to view the plots.
#> =======================================================
#> Back-end model fitted in mgcv::bam function:
#> Method REML
#> Formula:
#> y ~ s(time, bs = "ps", by = NA, pc = 0, k = 6, fx = TRUE)
#> Pseudolikelihood AIC: 46234.19
#> Pseudolikelihood BIC: 46260.11
#> Note: Used listwise deletion for missing data.
#> =======================================================
plot(model1_2knots_unpenalized)
```



The plot shows the estimated coefficient function, and approximate estimates for 95% pointwise confidence intervals (not corrected for potential multiple comparisons) for the value of the function at each time.

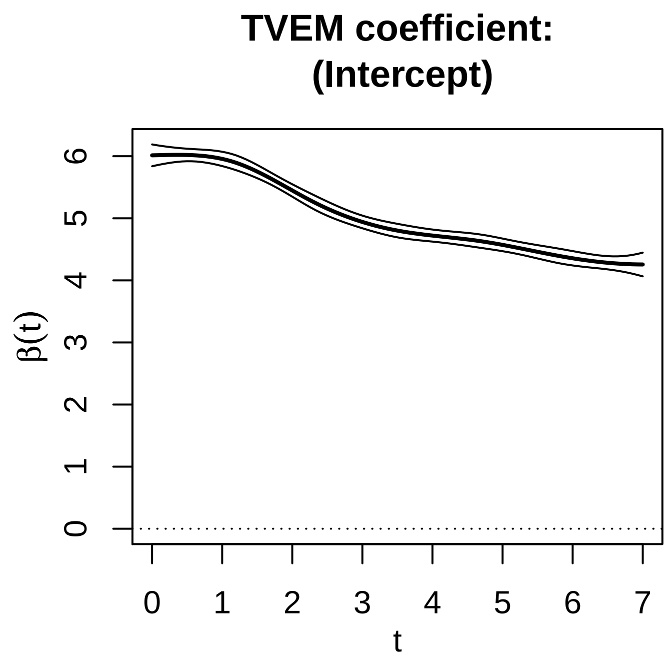
We do not estimate random effects in the current version of this package. Instead, we use a (possibly penalized) form of generalized estimating equations with working independence, and adjust the standard errors for within-subject correlation using a sandwich formula.

Using the `select_tvem` function

It is a very good idea to examine how the covariate means change over time. That is, we should fit intercept-only TVEMs for x_1 and x_2, not just y.

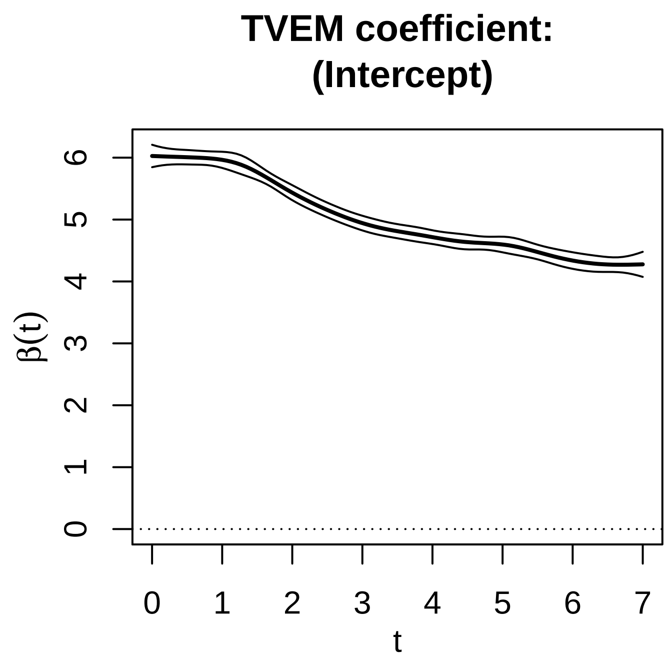
While doing this, we can take the opportunity to additionally explore yet another way to fit a model using the `tvem` function, by using the `select_tvem` function to choose the number of interior knots by a pseudolikelihood equivalent to an AIC or BIC criterion. "Pseudolikelihood" here means that the information criterion does not take within-subject correlation into account because we are fittin a marginal model agnostic to the exact correlation structure. The code below will fit the model with 0 to 5 interior knots, record the fit criterion for each potential choice, and select the one which gives the lowest (best) value of the criterion.

```
model2 <- select_tvem(data=the_data,
                      formula = x1~1,
                      id=subject_id,
                      time=time,
                      max_knots=5)
#>      knots       ic
#> [1,]     0 177522.9
#> [2,]     1 177475.0
#> [3,]     2 177453.0
#> [4,]     3 177452.0
#> [5,]     4 177448.3
#> [6,]     5 177446.4
#> [1] "Selected 5 interior knots."
print(model2)
#> ====================================================
#> Time-Varying Effects Modeling (TVEM) Function Output
#> ====================================================
#> Response variable:   x1
#> Response outcome distribution type: gaussian
#> Time interval:   0 to 7
#> Number of subjects:  300
#> Effects specified as time-varying:  (Intercept)
#> You can use the plot function to view the plots.
#> ====================================================
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> x1 ~ s(time, bs = "ps", by = NA, pc = 0, k = 9, fx = FALSE)
#> Pseudolikelihood AIC: 177446.42
#> Pseudolikelihood BIC: 177476.67
#> Model selection table for number of interior knots:
#>      knots       ic
#> [1,]     0 177522.9
#> [2,]     1 177475.0
#> [3,]     2 177453.0
#> [4,]     3 177452.0
#> [5,]     4 177448.3
#> [6,]     5 177446.4
#> ====================================================
plot(model2)
```



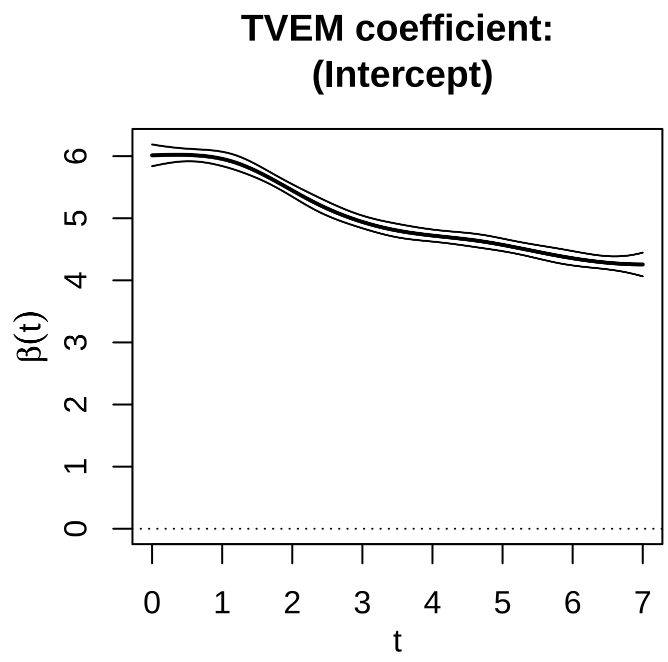
The highest number of interior knots in the range is selected. It might be reasonable to run the function again with a higher value of `max_knots`, to see whether there might be an even better fit that had not been found yet. Note that the individual knots and their locations do not have a special interpretation (as they do in some "changepoint" models); they are just a tool to make the fitted function more flexible than a simple polynomial would be. Let us try again with a maximum of ten interior knots.

```
model2_selected_knots <- select_tvem(data=the_data,
                     formula = x1~1,
                     id=subject_id,
                     time=time,
                     max_knots=10)
#>       knots       ic
#>  [1,]     0 177522.9
#>  [2,]     1 177475.0
#>  [3,]     2 177453.0
#>  [4,]     3 177452.0
#>  [5,]     4 177448.3
#>  [6,]     5 177446.4
#>  [7,]     6 177445.6
#>  [8,]     7 177445.5
#>  [9,]     8 177447.3
#> [10,]     9 177444.9
#> [11,]    10 177441.9
#> [1] "Selected 10 interior knots."
print(model2_selected_knots)
#> =======================================================
#> Time-Varying Effects Modeling (TVEM) Function Output
#> =======================================================
#> Response variable:   x1
#> Response outcome distribution type: gaussian
#> Time interval:   0 to 7
#> Number of subjects:  300
#> Effects specified as time-varying:  (Intercept)
#> You can use the plot function to view the plots.
#> ======================================================
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> x1 ~ s(time, bs = "ps", by = NA, pc = 0, k = 14, fx = FALSE)
#> Pseudolikelihood AIC: 177441.89
#> Pseudolikelihood BIC: 177481.16
#> Model selection table for number of interior knots:
#>       knots       ic
#>  [1,]     0 177522.9
#>  [2,]     1 177475.0
#>  [3,]     2 177453.0
#>  [4,]     3 177452.0
#>  [5,]     4 177448.3
#>  [6,]     5 177446.4
#>  [7,]     6 177445.6
#>  [8,]     7 177445.5
#>  [9,]     8 177447.3
#> [10,]     9 177444.9
#> [11,]    10 177441.9
#> ======================================================
plot(model2_selected_knots)
```



The fit continues improving with higher and higher numbers of knots, but the differences in fit are so small no further insight would be gained by trying more. The `use_bic` option can be set to `TRUE` in order to use a more parsimonious criterion that might be minimized at a smaller number of knots.

```
model2_selected_knots_bic <- select_tvem(data=the_data,
                    formula = x1~1,
                    id=subject_id,
                    use_bic=TRUE,
                    time=time,
                    max_knots=10)
#>       knots       ic
#>  [1,]     0 177541.3
#>  [2,]     1 177497.2
#>  [3,]     2 177478.9
#>  [4,]     3 177481.2
#>  [5,]     4 177477.1
#>  [6,]     5 177476.7
#>  [7,]     6 177477.7
#>  [8,]     7 177479.6
#>  [9,]     8 177481.4
#> [10,]     9 177480.7
#> [11,]    10 177481.2
#> [1] "Selected 5 interior knots."
print(model2_selected_knots_bic)
#> =======================================================
#> Time-Varying Effects Modeling (TVEM) Function Output
#> =======================================================
#> Response variable:   x1
#> Response outcome distribution type: gaussian
#> Time interval:   0 to 7
#> Number of subjects:  300
#> Effects specified as time-varying:  (Intercept)
#> You can use the plot function to view the plots.
#> =======================================================
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> x1 ~ s(time, bs = "ps", by = NA, pc = 0, k = 9, fx = FALSE)
#> Pseudolikelihood AIC: 177446.42
#> Pseudolikelihood BIC: 177476.67
#> Model selection table for number of interior knots:
#>       knots       ic
#>  [1,]     0 177541.3
#>  [2,]     1 177497.2
#>  [3,]     2 177478.9
#>  [4,]     3 177481.2
#>  [5,]     4 177477.1
#>  [6,]     5 177476.7
#>  [7,]     6 177477.7
#>  [8,]     7 177479.6
#>  [9,]     8 177481.4
#> [10,]     9 177480.7
#> [11,]    10 177481.2
#> =======================================================
plot(model2_selected_knots_bic)
```



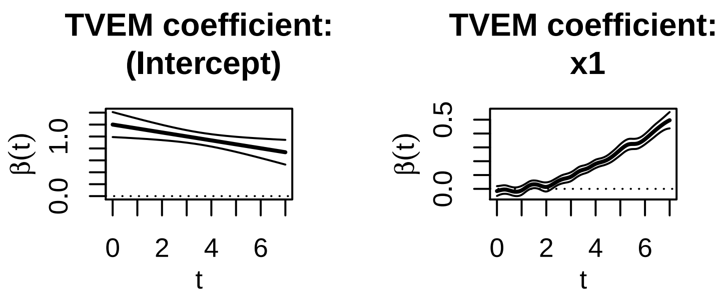
Now five knots have been selected.

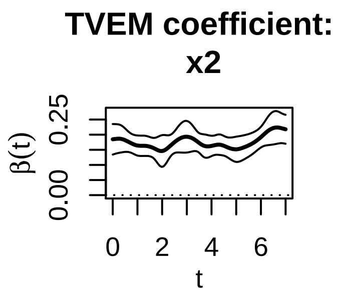
It would be good to repeat this analysis to plot the mean of x_2 over time in the same way.

Time-varying effects of covariates

Next, we fit a TVEM model with covariates. We allow both x_1 and x_2 to potentially have "time-varying effects" (regression relationships with the response that change over time, that is, a potential interaction between time and the covariate, specified without the assumption of linearity). The implied mean model is $E(y|t, x_1(t), x_2(t)) = \beta_0(t) + \beta_1(t)x_1(t) + \beta_2(t)x_2(t)$ where t is time in days.

```
model3 <- tvem(data=the_data,
               formula=y~x1+x2,
               id=subject_id,
               time=time)
print(model3)
#> ========================================================
#> Time-Varying Effects Modeling (TVEM) Function Output
#> ========================================================
#> Response variable:   y
#> Response outcome distribution type: gaussian
#> Time interval:    0 to 7
#> Number of subjects:  300
#> Effects specified as time-varying:  (Intercept), x1, x2
#> You can use the plot function to view the plots.
#> ========================================================
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> y ~ x1 + x2 + s(time, bs = "ps", by = NA, pc = 0, k = 24, fx = FALSE) +
#>     s(time, bs = "ps", by = x1, pc = 0, m = c(2, 1), k = 24,
#>         fx = FALSE) + s(time, bs = "ps", by = x2, pc = 0, m = c(2,
#>     1), k = 24, fx = FALSE)
#> Pseudolikelihood AIC: 44272.03
#> Pseudolikelihood BIC: 44373.9
#> Note: Used listwise deletion for missing data.
#> ======================================================
plot(model3)
```





The output includes the formula which is sent to the back-end calculation package, the `mgcv` package by Simon Wood, which is described in depth in Wood (2017). It is not necessary to interpret the pieces of the formula in order to understand the output; it is mainly supplied for advanced users and potential debugging use. In summary, it says that y depends on x_1 and x_2 as parametric terms, on a set of spline terms based on time, on a set of spline terms based on time multiplied by x_1, and on a set of spline terms based on time multiplied by x_2. Further details on interpreting the formula can be found in the `mgcv` documentation.

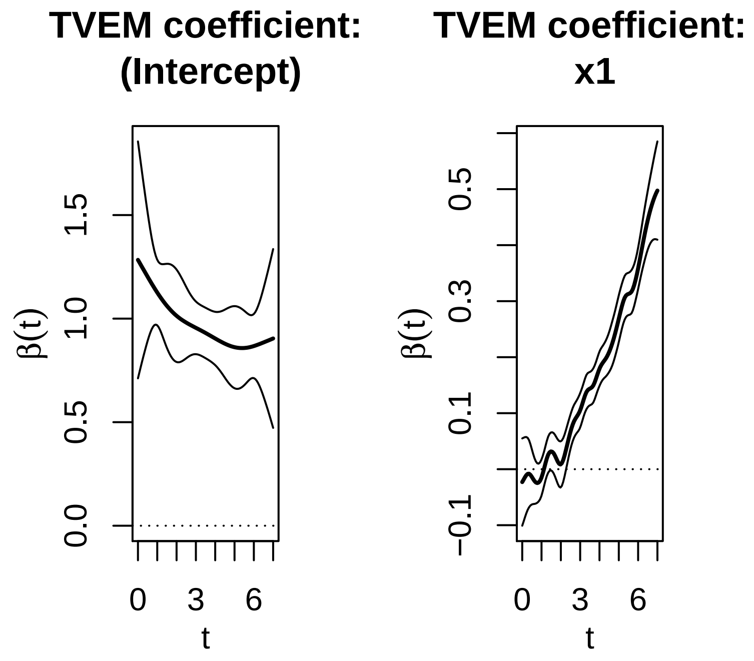
Holding x_1 and x_2 constant, the mean of y seems to decline over time. The penalty function causes the the relationship to be estimated as linear because there is no clear evidence of nonlinearity. From the results, x_1 appears to have an increasingly positive relationship with y over time. That is, x_1 and y are more strongly positively related as time goes on.

x_2 also seems to predict y (because the curve is not near zero), but the strength of the relationship does not change over time. That is, at any given time the predictive relationship between x_2 and y seems to be about equally strong. Thus, we could reasonably fit a similar but simpler model, but with x_2 having a non-time-varying effect even though it has time-varying values.

Time-invariant effects of covariates

The following code fits a model where x_2 is assumed to have a time-invariant effect but x_1 is allowed to have a time-varying effect.

```
model4 <- tvem(data=the_data,
               formula=y~x1,
               invar_effect=~x2,
               id=subject_id,
               time=time)
print(model4)
#> =======================================================
#> Time-Varying Effects Modeling (TVEM) Function Output
#> =======================================================
#> Response variable:   y
#> Response outcome distribution type: gaussian
#> Time interval:   0 to 7
#> Number of subjects:  300
#> Effects specified as time-varying:  (Intercept), x1
#> You can use the plot function to view the plots.
#> =======================================================
#> Effects specified as non-time-varying:
#>     estimate standard_error
#> x2 0.1753651    0.008207847
#> =======================================================
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> y ~ x1 + x2 + s(time, bs = "ps", by = NA, pc = 0, k = 24, fx = FALSE) +
#>     s(time, bs = "ps", by = x1, pc = 0, m = c(2, 1), k = 24,
#>         fx = FALSE)
#> Pseudolikelihood AIC: 44285.93
#> Pseudolikelihood BIC: 44370.6
#> Note: Used listwise deletion for missing data.
#> =======================================================
plot(model4)
```



The implied mean model is $E(y|t) = \beta_0(t) + \beta_1(t)x_1(t) + \beta_2 x_2(t)$. Notice that only covariates with time-varying effects are listed in the formula argument. The invar_effect argument is used for covariates with time-invariant effects. Also notice that a tilde (`~`) sign is needed before the list of covariates with time-invariant effects, because it is treated as the right-hand side of a formula. If there were multiple covariates with time-invariant effects, they would be listed as follows: `~x2+x3+x4` which again follows R style for the right-hand side of a formula.

In the output, there is no longer a plot for the coefficient of x_2 as a function of time, because β_2 is considered constant over time even if x_2 varies. Instead, the estimate and estimated standard error of β_2 are given as text in the output from print.

The seeming oscillations in confidence interval width in the plot do not have any particular interpretation; they are just an artifact of the locations of the knots. Also, the confidence intervals shown are approximate pointwise confidence intervals, much like those in the Methodology Center's `%TVEM` SAS macro. They are not simultaneous confidence bands, so they do not directly correct for multiple comparisons.

The main takeaway message from the analysis is the increasing $\beta_1(t)$ over time, suggesting an increasing association between x_1 and y, that is, some kind of interaction between t and x_1 in predicting y.

Example with a binary outcome variable

This example will be similar to the previous one, but with a binary response variable. The `tvem` library's simulation function will simulate binary y if requested by an optional argument.

```
set.seed(123)
the_data <- simulate_tvem_example(simulate_binary=TRUE)
```

The simulated dataset is similar to the previous example, but with binary y (0=no, 1=yes) generated from a logistic model.

Plotting the average log odds over time

We can plot the expected log odds over time, using a time-varying-intercept-only logistic model. The model assumes $\text{logit}(E(Y|t)) = \beta_0(t)$. The binary outcome is specified using the family argument as in R's `glm` function.

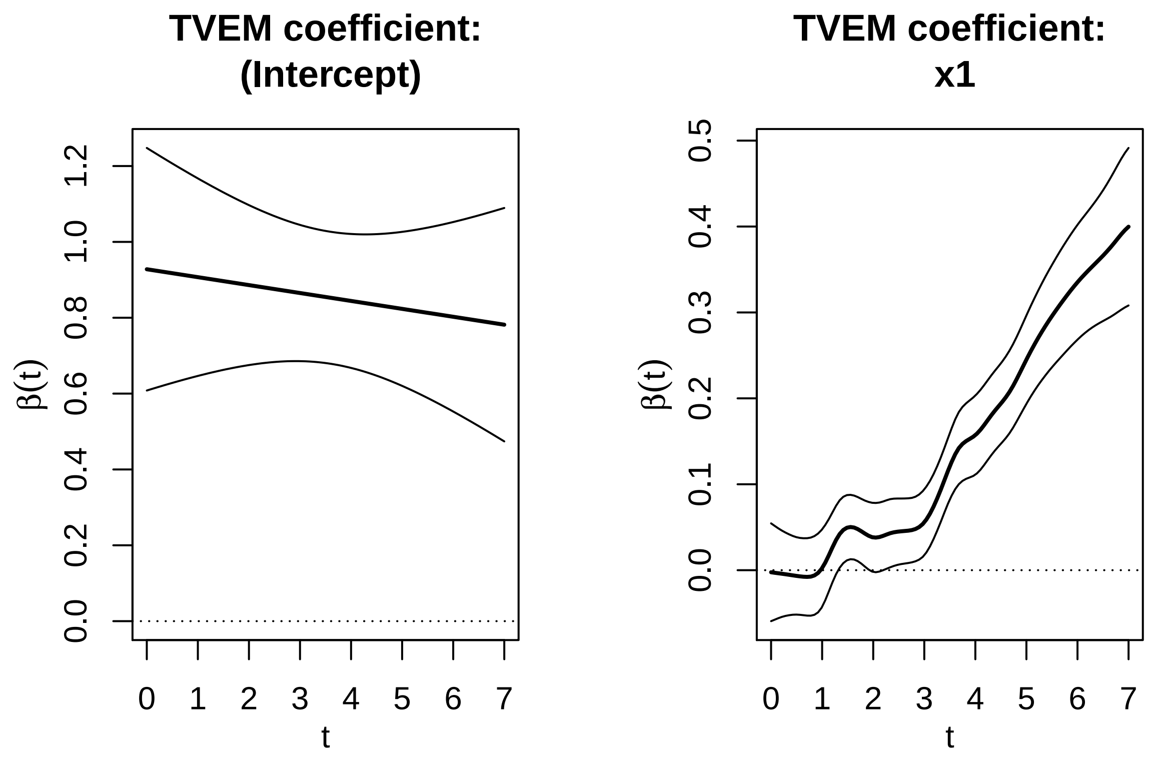
```
model_binary1 <- tvem(data=the_data,
                      formula=y~1,
                      family=binomial(),
                      id=subject_id,
                      time=time)
```

As before, you can use the default option of an automatic penalty function, or you could specify the number of knots, or you could use automatic selection for the number of knots without a penalty.

Including covariates

A model with covariates is similar to the previous example.

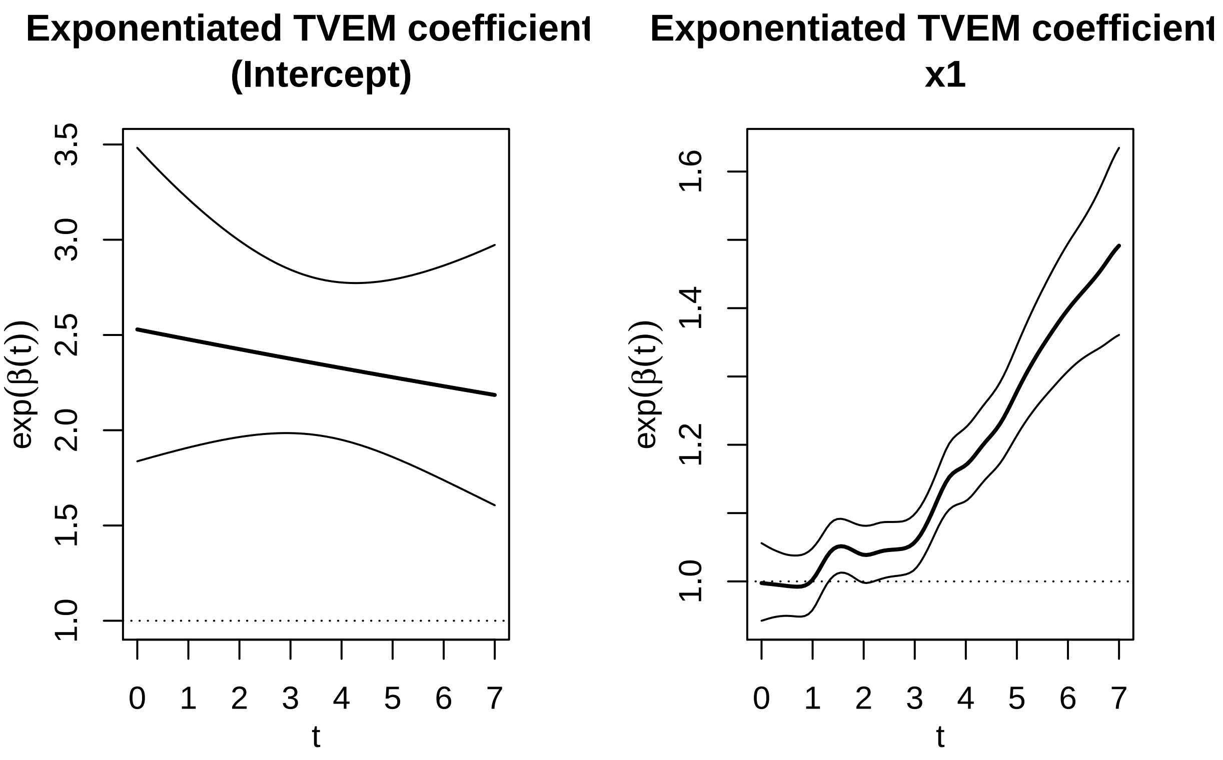
```
model_binary2 <- tvem(data=the_data,
                      formula=y~x1,
                      invar_effect=~x2,
                      id=subject_id,
                      family=binomial(),
                      time=time)
print(model_binary2)
#> =======================================================
#> Time-Varying Effects Modeling (TVEM) Function Output
#> =======================================================
#> Response variable:   y
#> Response outcome distribution type: binomial
#> Time interval:   0 to 7
#> Number of subjects:  300
#> Effects specified as time-varying:  (Intercept), x1
#> You can use the plot function to view the plots.
#> =======================================================
#> Effects specified as non-time-varying:
#>     estimate standard_error
#> x2 0.2074037     0.01582357
#> =======================================================
#> Back-end model fitted in mgcv::bam function:
#> Method fREML
#> Formula:
#> y ~ x1 + x2 + s(time, bs = "ps", by = NA, pc = 0, k = 24, fx = FALSE) +
#>     s(time, bs = "ps", by = x1, pc = 0, m = c(2, 1), k = 24,
#>         fx = FALSE)
#> Pseudolikelihood AIC: 8609.09
#> Pseudolikelihood BIC: 8657.7
#> Note: Used listwise deletion for missing data.
#> =======================================================
plot(model_binary2)
```



The implied mean model is $\text{logit}(E(y|t)) = \beta_0(t) + \beta_1(t)x_1(t) + \beta_2 x_2(t)$. In this simulated example, the relationship of x_1 with y appears to change over time, becoming more strong and positive as in the previous example.

With logistic TVEM, we can additionally plot exponentiated coefficients to get odds and odds ratios. In the plots below, the exponentiated intercept function shows the estimated odds of $Y = 1$ at a given time, assuming $x_1 = x_2 = 0$. The exponentiated beta function for x_1 shows the estimated odds ratio for $Y = 1$ given a 1-unit increase in x_1 at a given time.

```
plot(model_binary2, exponentiate=TRUE)
```



Although we do not present an example, Poisson count outcomes with a log link can be modeled similarly by specifying `family=poisson()` instead of `family=binomial()`. However, for overdispersed or zero-inflated counts, the Poisson distribution might not be appropriate. Currently, the only distributions supported by the `tvem` package are normal with identity link function for continuous numerical responses, binomial with logistic link for binary responses, and Poisson with log link for count responses.

Analysis with Sampling Weights

In some studies, it is helpful to weight participants within the sample, so that some participants have more influence on the final estimates than others. This is often done to counteract some characteristic of the study design or implementation which could cause bias if not properly adjusted for. For example, if people from a particular group were more likely to be included in the sample than other people, certain kinds of statistics for the overall population can still be estimated in an unbiased way, but they might require that these oversampled people be weighted downwards.

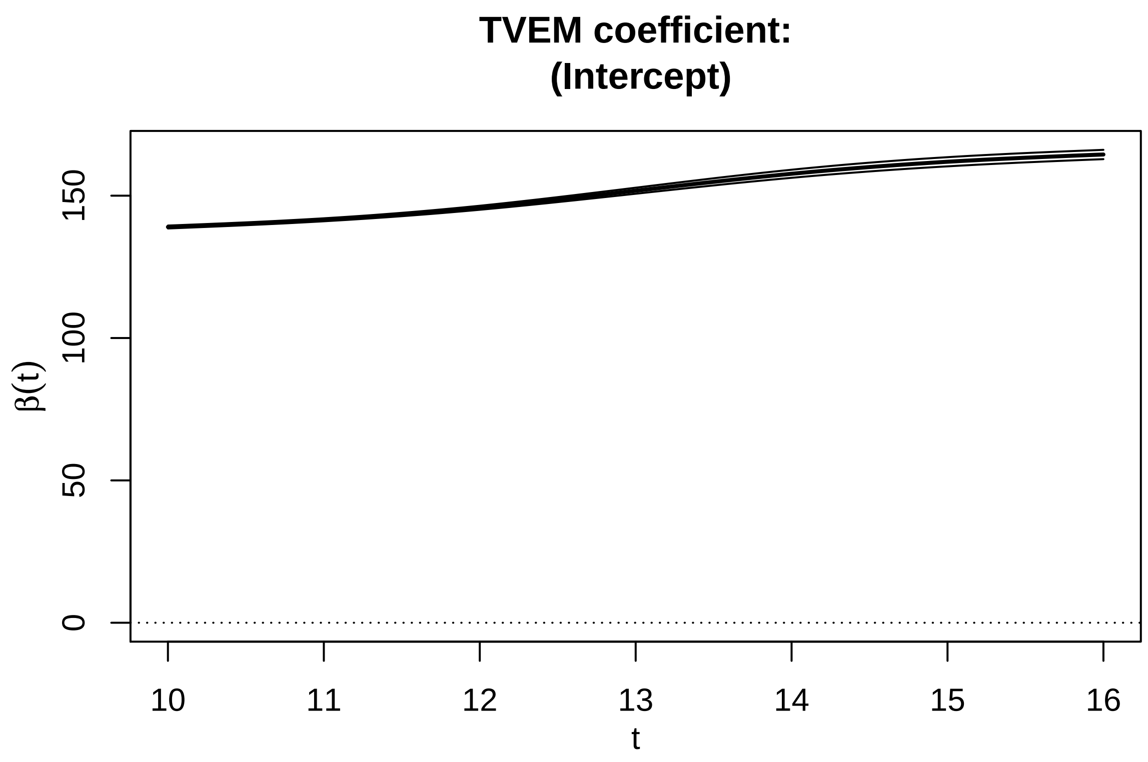
The `tvem` package can account for sampling weights, by multiplying each participant's contribution to the pseudolikelihood by that participant's sample weight. Ordinarily, the `tvem` function will automatically normalize the weights, which means adjusting them so that they have an average of 1. Without this adjustment, the standard errors might be calculated incorrectly because the number of subjects could be miscounted. For example, a participant with a weight of 100 should not really count as 100 separate participants for purposes of determining sample size. If you are in an unusual situation where you need to leave the weights as specified, use the option `normalize_weights=FALSE`. However, if you are in any doubt, leave `normalize_weights` at its default of `TRUE`.

As a simplistic example, suppose a researcher records the height in centimeters of children on their tenth birthday, and twice a year afterwards until their sixteenth birthday. Suppose that in the population of interest, 50% of the children were male and 50% were female. However, suppose girls were less likely to agree to be in the study. Therefore, in the final sample, about 2/3 of the participants were boys, thus twice as many boys as girls. The investigator consults with a statistician and decides to give each girl twice the sampling weight of a boy, so that the overall weighted results give equal attention to boys and girls in calculating overall estimates. The following code will simulate a fairly realistic dataset from such a study.

```r
set.seed(12345)
n <- 100
simulated_dataset <- NULL
for (this_id in 1:n) {
  age <- seq(10,16,by=.5)
  female <- rbinom(1,1,1/3)
  sample_weight <- ifelse(female==1,yes=2,no=1)
  if (female==1) {
    logistic_curve <- round(rnorm(1,0,1) + 135 +
                           runif(1,.9,1.1)*(160-135)/(1+exp(-.75*(age-13))) +
                           rnorm(length(age),0,.5),1)
  } else {
    logistic_curve <- round(rnorm(1,0,1) + 140 +
                           runif(1,.9,1.1)*(175-140)/(1+exp(-(age-13))) +
                           rnorm(length(age),0,.5),1)
  }
  simulated_dataset <- rbind(simulated_dataset,
                            data.frame(id=this_id,
                                       sample_weight=sample_weight,
                                       female=female,
                                       age=age,
                                       height=logistic_curve))
}
summary(simulated_dataset)
#>        id          sample_weight      female           age           height
#>  Min.   :  1.00   Min.   :1.00   Min.   :0.00   Min.   :10.0   Min.   :134.8
#>  1st Qu.: 25.75   1st Qu.:1.00   1st Qu.:0.00   1st Qu.:11.5   1st Qu.:143.6
#>  Median : 50.50   Median :1.00   Median :0.00   Median :13.0   Median :151.9
#>  Mean   : 50.50   Mean   :1.39   Mean   :0.39   Mean   :13.0   Mean   :153.5
#>  3rd Qu.: 75.25   3rd Qu.:2.00   3rd Qu.:1.00   3rd Qu.:14.5   3rd Qu.:162.5
#>  Max.   :100.00   Max.   :2.00   Max.   :1.00   Max.   :16.0   Max.   :176.6
```

The simplest TVEM that can be done with this data is an estimate of the average growth curve over time, averaging over individual differences and ignoring sex. In this case, the intercept is the only time-varying coefficient and becomes the estimate of the fitted mean conditional on time. The following code fits this model with and without weights. A lower number of knots than usual is used, because there are relatively few distinct values of time (age) in the dataset.

```r
tvem1_unweighted <- tvem(data=simulated_dataset,
                         formula=height~1,
                         id=id,
                         num_knots=5,
                         time=age)
tvem1_weighted <- tvem(data=simulated_dataset,
                         formula=height~1,
                         id=id,
                         num_knots=5,
                         time=age,
                         weights=sample_weight)
plot(tvem1_weighted)
```



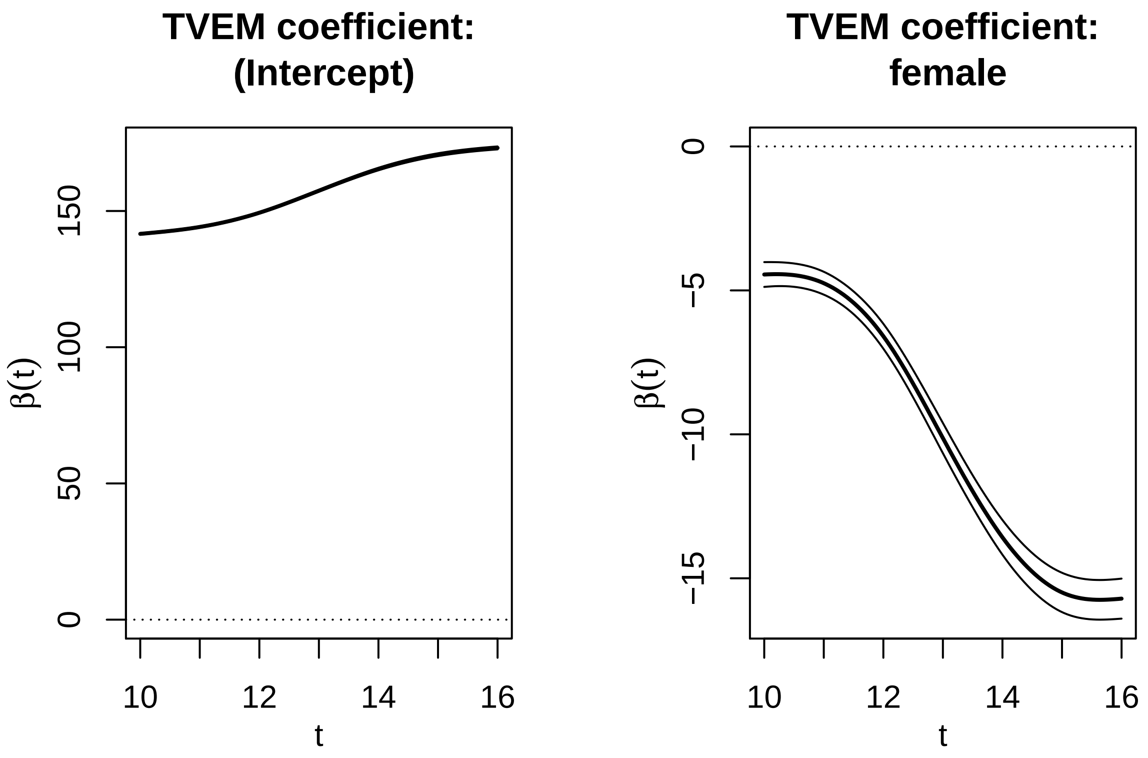
The plots of the two models look superficially similar so only one is shown above. However, a close examination of the fitted results shows that the weighted dataset leads to an estimated average height of about 0-3 centimeters shorter than the unweighted dataset, depending on age.

```
print(summary((tvem1_unweighted$grid_fitted_coefficients$`(Intercept)`$estimate)))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   139.8   144.3   153.5   153.5   162.6   167.1
print(summary((tvem1_weighted$grid_fitted_coefficients$`(Intercept)`$estimate)))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   139.0   143.4   151.7   151.7   160.1   164.5
```

This is because the weighted analysis attempts to generalize to a population with equal numbers of girls and boys, while the unweighted analysis attempts to generalize to a population of mostly boys. Boys are taller than girls on average, during their teenage years.

This example is not particularly compelling as an illustration of the need for weights, because if weights depend only on sex then conditioning on sex in the model will the weights unnecessary. Once sex (dummy-coded as 1 for female and 0 for male) is represented as part of the estimated mean model, the weight becomes unnecessary. Note that sex has a time-varying effect on height even if sex itself remains constant, because the difference in height between males and females depends on age.

```
tvem2_unweighted <- tvem(data=simulated_dataset,
                         formula=height~female,
                         id=id,
                         num_knots=5,
                         time=age)
tvem2_weighted <- tvem(data=simulated_dataset,
                       formula=height~female,
                       id=id,
                       num_knots=5,
                       time=age,
                       weights=sample_weight)
plot(tvem2_weighted)
```



```
print(summary((tvem2_unweighted$grid_fitted_coefficients$`(Intercept)`$estimate)))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   141.6   146.3   157.4   157.4   168.4   173.1
print(summary((tvem2_weighted$grid_fitted_coefficients$`(Intercept)`$estimate)))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   141.6   146.3   157.4   157.4   168.4   173.2
```

The plot at left shows the estimated expected height by age for participants with `female==0` (males), and the plot at right shows the signed difference between groups (very roughly speaking, the effect on height of being female).

However, in a more realistic example the probabilities of entering the sample might depend on factors such as city of residence, racial or ethnic background, economic class, etc., which are not necessarily of interest as covariates in our model. In this case, sampling weights may become more important. If you are in doubt, don't try to construct or use weights by yourself. However, if you are in a situation that requires weights (as in analyzing a large-scale social or economic survey), the `weights` argument allows you to use them in `tvem`.

References

- de Boor, C. (1972). On calculating with B-splines. Journal of Approximation Theory, 6: 50–62.

- Eilers, P. H. C., & Marx, B. D. (1996). Flexible smoothing with B-splines and penalties. Statistical Science, 11: 89-121.
- Hastie, T, & Tibshirani, R. (1993). Varying-coefficient models. Journal of the Royal Statistical Socety, B, 55:757-796.
- Li, R., Dziak, J. J., Tan, X., Huang, L., Wagner, A. T., & Yang, J. (2017). TVEM (time-varying effect model) SAS macro users' guide (Version 3.1.1). University Park: The Methodology Center, Penn State. Available online at [https://www.methodology.psu.edu/downloads/tvem/])(https://www.methodology.psu.edu/downloads/tvem/) and archived at https://github.com/dziakj1/MethodologyCenterTVEMmacros and https://scholarsphere.psu.edu/collections/v41687m23q.
- Ruppert, D., Wand, M. P., & Carroll, R. J. (2003). Semiparametric regression. Cambridge: Cambridge University Press.
- 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. (2017). Generalized Additive Models: An Introduction with R, 2nd edition. Chapman and Hall/CRC.

Acknowledgements

This software was developed by John Dziak at the Methodology Center at Pennsylvania State University, under the supervision of Donna Coffman of Temple University. We thank Patricia Berglund, Danielle Cabel, and Yajnaseni Chakraborti for their very valuable help with testing the package and documentation.

The development of this software was part of a research project supported by National Institutes of Health grants P50 DA039838 from the National Institute of Drug Abuse and 1R01 CA229542-01 from the National Cancer Institute and the NIH Office of Behavioral and Social Science Research. Content is solely the responsibility of the authors and does not necessarily represent the official views of the funding institutions mentioned above.