V: 27/02/2023

4. Software

I would also explain here that the pkg not only covers the methods proposed in their thesis but also TTE, MAP,... and that the paper is currently under review.

In the following chapter, we introduce the R-package `NCC`, which allows for simulation and analysis of flexible platform trials with non-concurrent controls. All analysis methods discussed in Chapter 2 are implemented in this package, along with functions for data generation and wrapper functions for running simulation studies.

In Section 4.1 we present an overview of the software for simulating adaptive clinical trials available to date.

In Section 4.2 we describe how the package is structured, which functions it contains, and their respective input and output values.

Lastly, Section 4.3 shows specific examples how to use the `NCC` R-package to generate and analyze platform trial data and run simulation studies.

4.1. Related Software

move to CH1

~~**PENDING - change wording!**~~

In complex designs and especially in platform trials, the use of software to design the trials and to investigate their operating characteristics via simulations has become paramount [5]. Examples of commercial statistical software for that purpose are FACTS [21] and EAST [22]. FACTS (the "Fixed and Adaptive Clinical Trial Simulator"), is a commercial software package developed by Berry Consultants to facilitate the design and statistical analysis of clinical trials. EAST is also a commercial software, particularly devoted to the design, simulation, and monitoring of adaptive, group-sequential and fixed sample size trials. There have also been proposed open-source R packages. OCTOPUS [23] is an R package which has been developed with the objective of supporting in simulating platform trial designs. MAMS [24] implements the design of multi-arm multi-stage trials with normal, binary, ordinal or time-to-event endpoints within the group-sequential framework. gsDesign [25] and rpact [26] design groups sequential and confirmatory adaptive trial designs and help to describe their properties. SIMPLE [27] (SIMulating PLatform trials Efficiently) is a modular software developed for simulating a wide variety of platform trial designs.

However, to the best of our knowledge, there is no open software available implementing the different methodologies that have recently been proposed for the analysis including non-concurrent controls. Moreover, there is a need to provide statistical tools that allow for assessing the properties of the methods when utilising non-concurrent controls and risks of bias in the estimates under a range of situations, including time trends.

you could extend a bit this part

4.2. Software Description

We present the R-package `NCC` [28], which was developed for assessing the operating characteristics of analysis methods that utilize non-concurrent controls in the analysis of platform trials. The `NCC` package provides functions to simulate platform trials with continuous or binary endpoints, as well as functions to analyse the trial data using various approaches, allowing for the incorporation of NCC data. It can be installed either from CRAN (Comprehensive R Archive Network) or Github using the following commands:

```
> devtools::install_github("pavlakrotka/NCC")
> install.packages("NCC")
> library(NCC)
```

The package comes with an accompanying website with background explanations and short tutorials: `https://pavlakrotka.github.io/NCC/`.

The `NCC` package focuses on trials with continuous or binary endpoints and consists of 34 functions. Some of the functions are implemented for continuous endpoints and some for binary endpoints. The functions with the suffix `_cont` refer to functions for simulation and analysis of trials with continuous endpoints, while `_bin` refers to binary endpoints. The `NCC` functions can be grouped into three main categories according to their functionality: data simulation, analysis, and visualization and wrappers. See Table 4.1 for a summary of the main functions. Figure 4.1 outlines the package structure.

The functions `datasim_cont()` and `datasim_bin()` refer to the simulation of patient data from a platform trial; functions such as `fixmodel_cont()`, `mixmodel_cont()` or `splines_cont()` are functions devoted to comparing the efficacy of an experimental treatment versus control using concurrent and non-concurrent controls in trials with continuous endpoints. Finally, the functions `plot_trial()` and `sim_study_par()` are intended to visualise the generated trials and perform simulation studies, respectively.

Most functions in the `NCC` package use common arguments. The main arguments are briefly described in Table 4.2, together with the functions which rely on them.

Argument	Description	Functions
`num_arms`	Number of treatment arms in the trial	`datasim_bin()`, `datasim_cont()`
`n_arm`	Sample size per experimental treatment arm	`datasim_bin()`, `datasim_cont()`
`d`	Timings of adding new arms in terms of number of patients recruited to the trial	`datasim_bin()`, `datasim_cont()`
`p0`	Response in the control arm for platform trials with binary endpoints	`datasim_bin()`
`mu0`	Response in the control arm for platform trials with continuous endpoints	`datasim_cont()`
`OR`	Odds ratios for each treatment arm compared to control	`datasim_bin()`
`theta`	Treatment effects for each treatment arm	`datasim_cont()`
`sigma`	Standard deviation of the responses	`datasim_cont()`
`lambda`	Strength of time trend in each arm	`datasim_bin()`, `datasim_cont()`
`trend`	Time trend pattern	`datasim_bin()`, `datasim_cont()`

`data`	Trial data, e.g. generated with the `datasim_*()` functions	`fixmodel_bin()`, `fixmodel_cont()`, `fixmodel_cal_bin()`, `fixmodel_cal_cont()`, `mixmodel_cont()`, `mixmodel_cal_cont()`, `mixmodel_AR1_cont()`, `mixmodel_AR1_cal_cont()`, `splines_cont()`, `splines_cal_cont()`, `sepmodel_bin()`, `sepmodel_cont()`, `poolmodel_bin()`, `poolmodel_cont()`
`arm`	Treatment arm under study to perform inference on	`fixmodel_bin()`, `fixmodel_cont()`, `fixmodel_cal_bin()`, `fixmodel_cal_cont()`, `mixmodel_cont()`, `mixmodel_cal_cont()`, `mixmodel_AR1_cont()`, `mixmodel_AR1_cal_cont()`, `splines_cont()`, `splines_cal_cont()`, `sepmodel_bin()`, `sepmodel_cont()`, `poolmodel_bin()`, `poolmodel_cont()`
`alpha`	Significance level	`fixmodel_bin()`, `fixmodel_cont()`, `fixmodel_cal_bin()`, `fixmodel_cal_cont()`, `mixmodel_cont()`, `mixmodel_cal_cont()`, `mixmodel_AR1_cont()`, `mixmodel_AR1_cal_cont()`, `splines_cont()`, `splines_cal_cont()`, `sepmodel_bin()`, `sepmodel_cont()`, `poolmodel_bin()`, `poolmodel_cont()`
`ncc`	Whether to include NCC data into the analysis	`fixmodel_bin()`, `fixmodel_cont()`, `fixmodel_cal_bin()`, `fixmodel_cal_cont()`, `mixmodel_cont()`, `mixmodel_cal_cont()`, `mixmodel_AR1_cont()`, `mixmodel_AR1_cal_cont()`, `splines_cont()`, `splines_cal_cont()`, `sepmodel_bin()`, `sepmodel_cont()`, `poolmodel_bin()`, `poolmodel_cont()`
`unit_size`	Number of patients per calendar time unit	`fixmodel_cal_bin()`, `fixmodel_cal_cont()`, `mixmodel_cal_cont()`, `mixmodel_AR1_cal_cont()`, `splines_cal_cont()`
`bs_degree`	Degree of the polynomial spline	`splines_cont()`, `splines_cal_cont()`

Table 4.2.: Main input arguments together with a short description and functions included in this article using these arguments. Detailed explanations can be found at `https://pavlakrotka.github.io/NCC/`.

As this thesis focuses on platform trials with continuous endpoints, we will mainly describe the functions for this type of endpoints, however, a detailed explanation regarding the functions for binary endpoints can be found on the package website.

4.2.1. Data simulation

Platform trials with a continuous outcome are simulated using the function `datasim_cont()`, as follows:

```
> datasim_cont(num_arms, n_arm, d, period_blocks = 2, mu0 = 0,
               theta, lambda, sigma, trend, N_peak, n_wave,
               full = FALSE, check = TRUE)
```

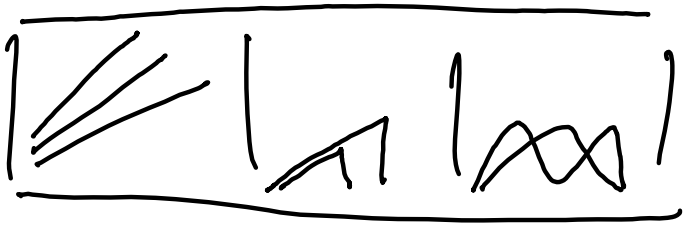
The arguments are, among others, the number of experimental treatment arms (`num_arms`), as well as their sample size (`n_arm`), timings of entering the trial in terms of patients

Figure 4.1.: Scheme of the `NCC` package functions by functionality. ~~Note that~~ Auxiliary functions for data generation are omitted in this figure.

Function	Description	Functionality
`datasim_cont()`	Simulates trials with continuous endpoints	Data simulation
`datasim_bin()`	Simulates trials with binary endpoints	Data simulation
`get_ss_matrix()`	Computes sample sizes per arm and period	Data simulation
`linear_trend()`	Generates a linear time trend	Data simulation
`sw_trend()`	Generates a step-wise time trend	Data simulation
`inv_u_trend()`	Generates a inverted-u time trend	Data simulation
`seasonal_trend()`	Generates a seasonal time trend	Data simulation
`fixmodel_bin()`	Performs analysis using a regression model adjusting for periods for binary data	Data analysis
`fixmodel_cont()`	Performs analysis using a regression model adjusting for periods for continuous data	Data analysis
`fixmodel_cal_bin()`	Performs analysis using a regression model adjusting for calendar times for binary data	Data analysis
`fixmodel_cal_cont()`	Performs analysis using a regression model adjusting for calendar times for continuous data	Data analysis
`poolmodel_bin()`	Performs pooled analysis for binary data	Data analysis
`poolmodel_cont()`	Performs pooled analysis for continuous data	Data analysis
`sepmodel_bin()`	Performs separate analysis for binary data	Data analysis
`sepmodel_cont()`	Performs separate analysis for continuous data	Data analysis
`mixmodel_cont()`	Performs analysis using a mixed model adjusting for periods as a random factor for continuous data	Data analysis
`mixmodel_cal_cont()`	Performs analysis using a mixed model adjusting for calendar times as a random factor for continuous data	Data analysis
`mixmodel_AR1_cont()`	Performs analysis using a mixed model adjusting for periods as a random factor with AR1 correlation structure for continuous data	Data analysis
`mixmodel_AR1_cal_cont()`	Performs analysis using a mixed model adjusting for calendar times with AR1 correlation structure as a random factor for continuous data	Data analysis
`splines_cont()`	Performs analysis using regression splines with knots placed according to periods for continuous data	Data analysis
`splines_cal_cont()`	Performs analysis using regression splines with knots placed according to calendar times for continuous data	Data analysis
`plot_trial()`	Visualizes the simulated trial over time	Data visualization
`sim_study_par()`	Performs a simulation study with given scenarios	Wrapper function

Table 4.1.: Main functions of the `NCC` package with a short description.

already enrolled in the trial to this point (`d`) and treatment effects (`theta`). The control mean is given by the argument `mu0` and has a default value of 0. The standard deviation of the responses is specified by the input argument `sigma`. It is assumed that the sample sizes in each experimental arm are equal. Patients are assigned to the arms according to block randomization using an allocation ratio of 1:1....:1 in each period. The function allows to simulate trial data in the presence of time trends of different patterns and strengths, which can also be specified in the input arguments. The time trend pattern can be specified by means of the argument `trend`, choosing from the options linear, stepwise, inverted-u with a peak at time `N_peak` and seasonal with `n_wave` cycles, while the strength of the trend is indicated by the argument `lambda`. The argument `full` specifies if the output is given in the form of a data frame (if `full=FALSE`) or a list (if `full=TRUE`). Finally, `check` is an indicator of whether the input parameters are checked to ensure that they are correctly specified. If `check=TRUE`, the function returns helpful error messages in case of wrong input.



By default, the function returns the simulated trial data in the form of a data frame containing the following columns:

- `j` - patient recruitment index
- `response` - response for patient j
- `treatment` - indicator of the treatment patient j was allocated in
- `period` - indicator of the period in which patient j was recruited in

This dataset can be further analysed using the implemented analysis functions.

Simulation of binary endpoints is performed analogously, using the function `datasim_bin()`, which only differs in the indication of the control response (argument `p0`) and the treatment effects (argument `OR`), which are specified in terms of the odds ratio.

4.2.2. Analysis approaches

The main frequentist analysis approaches for continuous data implemented in the `NCC` package are the fixed and mixed effects, as well as spline regression models with adjustment either based on the periods or calendar times. The arguments common to all analysis functions in the `NCC` package are the data frame with the trial data, consisting of columns named "j", "response", "treatment" and "period" (`data`), the indicator of the experimental treatment arm that should be compared to the control group (`arm`) and the significance level (`alpha`).

To analyze the data using the frequentist fixed effects model from equation (2.5), one can use the `fixmodel_cont()` function as follows:

```
> fixmodel_cont(data, arm, alpha = 0.025, ...)
```

The function `mixmodel_cont()` permits to analyse the data using the mixed effects model described by equation (2.7) by means of:

```
> mixmodel_cont(data, arm, alpha = 0.025, ...)
```

To fit the spline regression model from equation (2.12), one can use the function `splines_cont()` with the following syntax:

```
> splines_cont(data, arm, alpha = 0.025, bs_degree = 3, ...)
```

where the parameter `bs_degree` indicates the degree of used the polynomial spline, with a default value of 3 for a cubic spline.

The above presented functions adjust for the time trend by means of the periods. Analogous functions for calendar time adjustment, indicated by the extension `_cal` (e.g. `fixmodel_cal_cont()`), are also implemented in the package.

The functions perform the respective analysis of the given dataset to compare the efficacy of a specific treatment against control, thus testing the null hypothesis for the treatment effect of the arm under study $H_0 : \theta_{\texttt{arm}} = 0$ against the one-sided alternative

$H_1 : \theta_{\texttt{arm}} > 0$. To test H_0, by default all trial data until the evaluated treatment arm leaves the trial are taken into account (i.e., also including data from unfinished arms that joined the platform up to the final analysis of the given treatment arm).

The output of the analysis functions is in the form of a list, containing the one-sided p-value, estimated treatment effect, 95% confidence interval, an indicator of whether the null hypothesis was rejected or not, and the fitted model. Functions for spline regression additionally output the position of the inner knots in terms of patient index.

4.2.3. Trial data visualization and wrapper functions

The package also includes functions to visualise the platform trial data and wrapper functions for performing simulation studies under different scenarios.

The visualization function `plot_trial()` uses as argument a vector with treatment indicators ordered by time (`treatments`) and outputs a plot of the trial progress over time, as we will illustrate in Section 4.3. The main wrapper function is `sim_study_par()`, which permits to efficiently run simulation studies using parallel computing. The code is parallelized on replication level, i.e. replications of one scenario are distributed over the available cores. Using this function requires creating a data frame with the desired simulation scenarios beforehand, which is then used as input to the function (argument `scenarios`) as follows:

```
> sim_study_par(nsim, scenarios, arms,
                models = c("fixmodel", "sepmodel", "poolmodel"),
                endpoint, perc_cores = 0.9)
```

where the remaining arguments specify how many times each scenario is to be replicated (`nsim`), the treatment arms that will be evaluated (`arms`), the considered analysis approaches (`models`), the indication of endpoint (`endpoint`) and the (approximate) percentage of available cores that should be used for the simulations (`perc_cores`). The output of `sim_study_par` is a data frame with all considered scenarios and corresponding results, that is, the probability to reject the null hypothesis, the bias, and the mean squared error (MSE) of the treatment effect estimates for each evaluated treatment arm and each considered analysis method.

4.3. Examples

4.3.1. How to analyze platform trial data utilising non-concurrent controls

Assume a platform trial with a shared control and three experimental treatment arms entering sequentially, where the second and third treatment arms enter after 100 and 250 patients are recruited to the trial, respectively. Furthermore, assume sample sizes of 100 and treatment effects of 0.25 in each experimental arm, as well as a standard deviation of the responses equal to 1. The mean response for all arms increases by 0.15, whenever a new arm is added to the trial. The data of this hypothetical trial can be simulated using the `datasim_cont()` function:

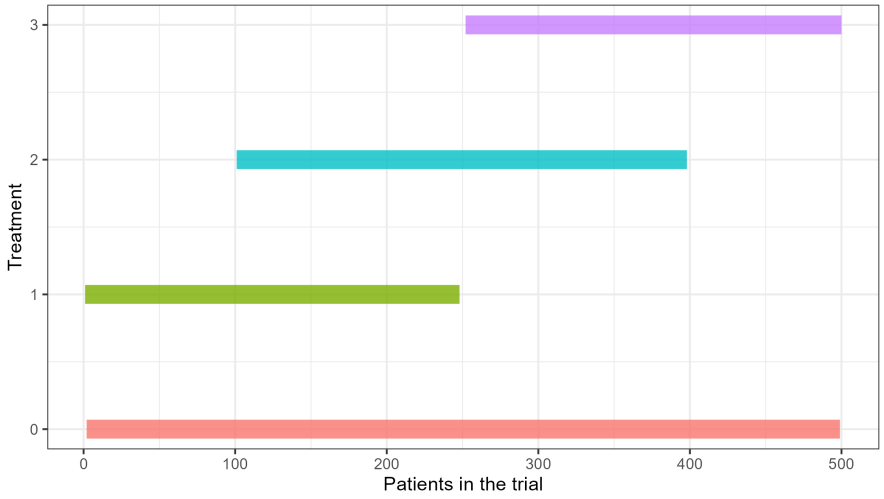


Figure 4.2.: Output of the function `plot_trial()`.

```
> set.seed(5)
> trial_data <- datasim_cont(num_arms = 3, n_arm = 100, d = c(0, 100, 250),
                             theta = rep(0.25, 3), lambda = rep(0.15, 4),
                             sigma = 1, trend = "stepwise_2")
```

The generated data is structured as follows:

```
> head(trial_data)
  j    response treatment period
1 1  0.78575816         1      1
2 2 -0.14030110         0      1
3 3 -0.36289414         0      1
4 4 -0.31737256         1      1
5 5  1.41623385         0      1
6 6 -0.04480078         0      1
```

where the patient index is given in the first column, followed by the continuous responses, the treatment arm indicator and finally the period allocation.

In order to illustrate the active treatment arms over time, we use the function `plot_trial()`, whose output is shown in Figure 4.2.

```
> plot_trial(trial_data$treatment)
```

The figure helps to display the entry and exit of arms into and out of the trial over time and to visualize the overlaps between arms more easily.

When the third arm ends, we want to evaluate its efficacy compared to the control using different frequentist model-based approaches. First, we consider a fixed regression

model with period adjustment for time trend (as in (2.5)), which is fitted using the `fixmodel_cont()` function:

```
> fixmodel_cont(trial_data, arm = 3, alpha = 0.025)
$p_val
[1] 0.01816827
$treat_effect
[1] 0.2790546
$lower_ci
[1] 0.01782669
$upper_ci
[1] 0.5402824
$reject_h0
[1] TRUE
```

← explain output
interpretation of results.

Analysis using a fixed regression model with calendar time adjustment (given by (2.6)) can be performed by means of the `fixmodel_cal_cont()` function. The length of the calendar time interval to adjust for (in terms of the recruited patients) is given by the argument `unit_size`.

```
> fixmodel_cal_cont(trial_data, arm = 3, unit_size = 25, alpha = 0.025)
$p_val
[1] 0.0218659
$treat_effect
[1] 0.2712733
$lower_ci
[1] 0.007656746
$upper_ci
[1] 0.5348899
$reject_h0
[1] TRUE
```

A mixed effect model with period adjustment that assumes uncorrelated random effects (see equation (2.7)) is implemented in the `mixmodel_cont()` function:

```
> mixmodel_cont(trial_data, arm = 3, ci = TRUE, alpha = 0.025)
$p_val
[1] 0.00231647
$treat_effect
[1] 0.3392225
$lower_ci
[1] 0.1059671
$upper_ci
[1] 0.572476
$reject_h0
[1] TRUE
```

Finally, if we want to adjust for time using cubic splines (see equation (2.12)), we can do so by means of the `splines_cont()` function:

```
> splines_cont(trial_data, arm = 3, bs_degree = 3, alpha = 0.025)
$p_val
[1] 0.01686
$treat_effect
[1] 0.2795447
$lower_ci
[1] 0.02160482
$upper_ci
[1] 0.5374847
$reject_h0
[1] TRUE
$knots
[1] 100 250 400
```

In the output of each analysis function, the first element of the list is the p-value (`p_val`) corresponding to testing the null hypothesis $H_0 : \theta_3 = 0$, followed by the estimated treatment effect (`treat_effect`) and the respective lower and upper confidence limits (`lower_ci`, `upper_ci`). The list also includes a binary indicator of `(p_val < alpha)`, i.e., whether the null hypothesis can be rejected on the specified significance level (`reject_h0`). In the considered case, the null hypothesis is rejected by all models, which implies that treatment arm 3 is efficacious. The output of the spline regression also includes the positions of the inner knots, which are in this case placed to the beginning of each period. Furthermore, each output includes the respective fitted regression model (`model`), which is here omitted for simplicity. However, the fitted model can be further analysed using the conventional R functions for generalized linear models, such as `summary(fixmodel_cont(data = trial_data, arm = 3)$model)`.

4.3.2. How to run a simulation study

Next, we consider the design of a platform trial with four experimental treatment arms entering sequentially. Aiming to assess the robustness of analysis methods that utilise non-concurrent controls in the presence of time trends, we want to perform a simulation study using the `NCC` package. For this, we first create a data frame with the desired scenarios that contains all the parameters needed for data generation and analysis.

```
> lambda_values <- rep(seq(-0.15, 0.15, length.out = 9), 2)
> sim_scenarios <- data.frame(num_arms = 4,
                              n_arm = 250,
                              d1 = 250*0,
                              d2 = 250*1,
                              d3 = 250*2,
```

```
d4 = 250*3,
period_blocks = 2,
mu0 = 0,
sigma = 1,
theta1 = 0,
theta2 = 0,
theta3 = 0,
theta4 = 0,
lambda0 = lambda_values,
lambda1 = lambda_values,
lambda2 = lambda_values,
lambda3 = lambda_values,
lambda4 = lambda_values,
trend = c(rep("linear", 9), rep("stepwise_2", 9)),
alpha = 0.025,
ncc = TRUE)
```

We assume here that the null hypothesis holds for all experimental arms (`theta1=...=theta4=0`). We vary the strength (`lambda`) and pattern (`trend`) of the time trend, in order to investigate their impact on the type I error rate, bias and mean squared error (MSE) of the treatment effect estimates. Note, however, that the time trend is equal across arms.

We use the function `sim_study_par()` to perform a simulation study with the created scenarios. Here, we evaluate the 4th experimental treatment arm using the regression model with period adjustment and compare its operating characteristics to the separate approach, where only CC data is used for the analysis and the pooled analysis, which naively pools CC and NCC data without further adjustments. Each scenario will be replicated 1000 times.

```
> sim_results <- sim_study_par(nsim = 1000,
                               scenarios = sim_scenarios,
                               arms = 4,
                               models = c("fixmodel", "sepmodel", "poolmodel"),
                               endpoint = "cont")
```

The function reports the system time after each scenario finishes in order to track the progress of the simulations.

The resulting data frame contains the considered scenarios and simulation results. The results include the probability of rejecting the null hypothesis, bias and MSE of the treatment effect estimates. We can now visualize the performance of the considered analysis methods with respect to the strength and pattern of the time trend. Figure 4.3 depicts the type 1 error, bias and MSE with respect to the strength of the time trend and according to the pattern of the time trend. The results show that the pooled analysis leads to inflation of the type I error rate in the presence of positive time trend and its deflation if there are negative time trends. The separate approach and regression model both control the type I error rate and yield unbiased treatment effect estimates.

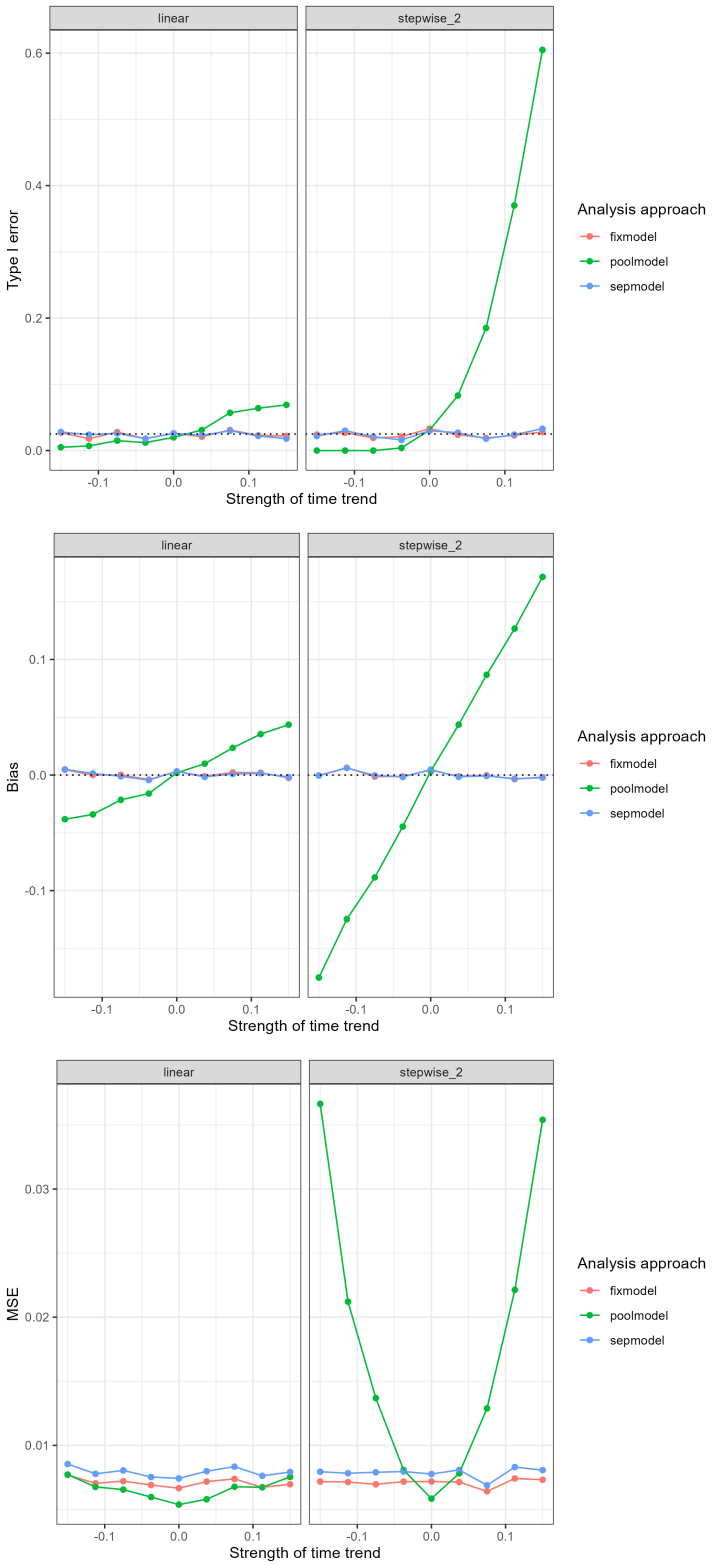


Figure 4.3.: Results of the toy simulation study. Type I error rate, bias and MSE of the treatment effect estimates for treatment arm 4 with respect to the strength of the time trend.