

Fidelity in Hybrid Implementation Trials: Technical Guidance for Implementation Researchers

Diana Trutschel

Catherine Blatter
Thekla Brunkert

Michael Simon

10/20/22

Table of contents

Preface	3
Citation	3
Comparison to other tools	3
1 Introduction	4
1.1 Required R packages	4
1.2 Other considerations	5
2 Step-by-step simulation	6
2.1 Simulation experiment for one setting	7
2.1.1 Study design	8
2.1.2 Indication if there are deviations from perfect situations or not	9
2.1.3 Model parameter	10
2.1.4 Iterations of the simulation	12
2.1.5 Simulation experiment	12
2.2 Comparison of several Fidelity patterns for a parallel design	13
3 Detailed statistical information	18
3.1 Explanation of simulation experiment and provided corresponding function	18
3.1.1 Determining the design matrix regarding the chosen design	18
3.1.2 Sampling data of a cluster randomized trial with a given design	19
3.1.3 Effect estimation from the data	22
3.2 Determine different design matrices	23
3.2.1 Parallel cluster randomized trials	23
3.2.2 Stepped wedge cluster randomized trial	23
3.3 Determine different fidelity patterns	24
3.3.1 Slow increase of fidelity using exponential function	25
3.3.2 Linear increase of fidelity using linear function	27
3.3.3 Fast increase of fidelity using logarithmic function	28
References	30

Preface

This online tutorial serves as a hands-on guide for researchers to replicate the simulation described in

[our paper].

This supplementary material was designed with regard to support applied researchers (*not* statisticians) to adapt the simulation with required parameters for their own study. Consequently, the content is split in the following chapters:

- Chapter 1 gives an overview of the technical requirements
- Chapter 2 describes step-by-step how to conduct the simulation for a specific research study, based on parallel or stepped wedge CRTs, furthermore show the comparison of several fidelity patterns
- Chapter 3 optionally provides additional statistical information, but does *not* need to be consulted to be able to run the simulation

Citation

If you find this work helpful, please cite both the online tutorial and the paper by the reference of the original paper as in:

[our paper]

Comparison to other tools

1 Introduction

The following technical setup is needed to pursue a local simulation on your computer:

- A local installation of the R statistical programming language (R Core Team 2022) is required.
The current version can be downloaded from <https://cran.r-studio.org>.
- In addition, we recommend the use of RStudio Desktop (an integrated development environment) which provides an excellent user interface for working with R.
It is available from here: <https://www.rstudio.com/products/rstudio/download/>.

Both R and RStudio (as well as further used R packages) are free software.

To follow this tutorial, we assume basic familiarity with the use of the R language.

1.1 Required R packages

The following R packages are needed to run the simulation:

- `{fidelitysim}` (Trutschel and Blatter 2022) - holds packaged code for the simulation experiment

It uses and builds upon:

- `{samplingDataCRT}` v1.0 (Trutschel and Treutler 2017) - for sampling data matrices
- `{lme4}` (Bates et al. 2015) - for linear model estimation
- `{ggplot2}` (Wickham 2016) - for visualizing the results

You can install the necessary packages by:

```
# to install {fidelitysim} we need the {remotes}-package
install.packages("remotes")

# then install {fidelitysim} from GitHub
remotes::install_github("INS-Basel/fidelitysim")
```

The other packages are installed from CRAN:

```
# install packages from CRAN
install.packages(c("samplingDataCRT", "lme4", "ggplot2"))
```

Consequently, to start the calculation the package needs to be loaded by:

```
library(fidelitysim)
```

1.2 Other considerations

The local simulation can - depending on the number of clusters - take up to 10-12 hours when the recommended number of iterations of 10,000 is used.

2 Step-by-step simulation

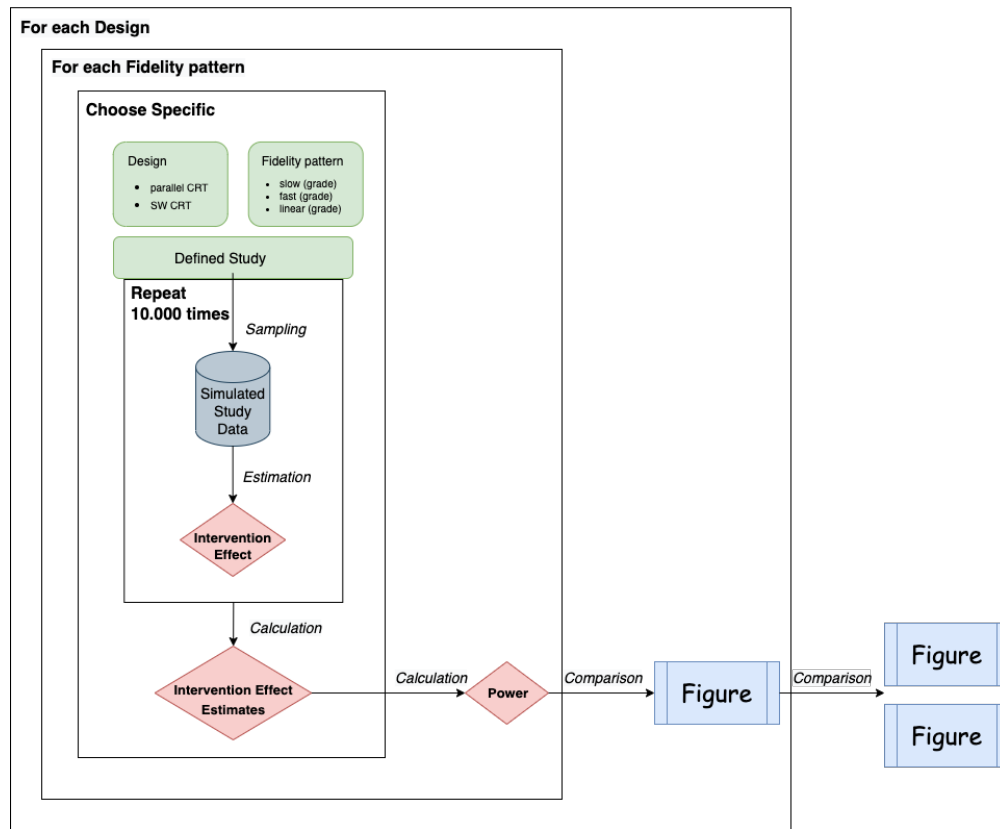


Figure 2.1: Workflow diagramme of the simulation

In the following the tutorial is separated in two parts, showing the use for:

1. Simulation experiment for one setting by choosing between
 - Parallel or Stepped wedge designs
 - Perfect versus Implementation error situations
2. Simulation experiments to comparison of several Fidelity patterns

2.1 Simulation experiment for one setting

According the [Workflow diagramme of the simulation] corresponds to inner box, here we show how to evaluate the performance of a cluster randomized trial with one specific setting of study design and probably existing specific implementation error (fidelity pattern).

Within each simulation experiment steps of data sampling and effect estimation, provided by function within the R-package {samplingDataCRT}, were repeated. For explanation see [Detailed statistical information](#). After repeating the steps of the simulation the obtained effect estimates can be evaluated for the performance of the model. The provided function `performanceMeas()` of the R-package {fidelitysim} calculates the following performance measures of a simulation:

- Bias: mean deviation of the estimates from the true value of the parameter of interest (intervention effect) is an indicator of accuracy
- Coverage: measurement used to control the Type I error rate for testing the null hypothesis (H_0) of no effect
- Power: proportion of simulation samples in which the H_0 of no effect is rejected at a significance level of α when H_0 is false (is related to the empirical Type II error rate).

To simplify the process of the whole simulation experiment the R-function `simulation()` is provided within the R-package {fidelitysim}. The function includes the steps: data sampling for a specific design and existing implementation error, effect estimation, performance measures calculation. Hence, the input of the function is the specification parameter:

- number of repeats for the simulation
- chosen design (including study design, number of cluster, time points and individuals per cluster)
- model parameters (effect, variances, ...)
- two design matrices which indicates whether deviation from perfect situations are assumed or not

and the output are:

- effect estimates
- performance measures

The usage is shown here for a parallel design trial, where the difference of the method application regarding the parameter setting for other design is explained. Be sure necessary packages are loaded:

```
# load necessary packages
library(fidelitysim)
```

```
library(samplingDataCRT)
#library(ggplot2)
```

One setting includes the specification of :

1. Study design
2. Indication if there are deviations from perfect situations or not
3. Model parameter
4. Iterations of the simulation

and finally,

5. the Simulation experiment can be performed.

2.1.1 Study design

Based on the study setup, the following parameters for the study design need to be defined:

- No. of clusters (e.g hospitals)
- No. of individuals per cluster (and time point) as the cluster size
- No. of time points the cluster were followed
- Design type controls if individuals will be followed over time (cross-sectional or longitudinal): cross-sectional type, which indicates trials where individuals within the cluster can change over time points

A design matrix according the study design has to be determined by the function `designMatrix()`. Here, two arguments are specific for the different types of studies. If the `design` argument is set 'parallel', then the parameter `nSW` indicates here the number of clusters are being control group. If choosing `design = "SWD"` (which is also set by default), the parameter `nSW` indicates here the number of cluster switches per time point from control to intervention group.

Then we determine the corresponding design matrix for a parallel cluster randomized study with:

- 6 cluster with a
- cluster size of 10 measured at
- 7 time points, where
- 3 clusters in each, the control as well as in the intervention group, are included

as follows:


```
## Design matrix ##

I <- 6 # number of clusters
J <- 10 # number of individuals per cluster (used later in simulation-step)
K <- 7 # number of time points
Sw <- 1 # number of cluster switches per time point can be manually set

# design matrix for parallel design
(designMat_prll <- designMatrix(nC = I, nT = K, nSw = round(I / 2), design = "parallel"))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0	0	0	0	0	0	0
[2,]	0	0	0	0	0	0	0
[3,]	0	0	0	0	0	0	0
[4,]	1	1	1	1	1	1	1
[5,]	1	1	1	1	1	1	1
[6,]	1	1	1	1	1	1	1

2.1.2 Indication if there are deviations from perfect situations or not

Further, in the function for the simulation by specifying a second design matrix **X.A**, we can indicate if there are deviations from the perfect situation present or not. Since we simulation perfect situation with no deviation, both arguments for the design matrices **X** and **X.A** are the same within the latter used R-function `simulation()` (see [Simulation experiment](#)). If deviations from perfect situation should be simulated, a second design matrix to which indicates the pattern of fidelity according to this study design has to be provided as well.

To consider the deviation from perfect implementation of an intervention (100% fidelity) we add fractional values to estimate the degree of the deviation and its effect on the study effects. Therefor 3 functions in within the R-package `{fidelitysim}` can be used to specify several fidelity patterns. For explanation of all kind of fidelity patters see [Determine different fidelity patterns](#). The conducted fidelity patterns can then be pushed to the design matrix with the R-package `{samplingDataCRT}`, see for more explanation [Fidelity patterns].

Using the same parallel design trial example and assume a linear increase of fidelity from 40% to 80% from time point 1 to the end of trial (time point K) 3 arguments are specified:

```
### Fidelity parameters ###

# Fidelity at the begin
Fid.T1 <- 0.4
```

```

# Fidelity at the end
Fid.End <- 0.8

# slope for linear function
m <- (Fid.T1 - Fid.End) / (1 - (K - 1))

# model linear increase
res.lin <- find.Fidelity.linear(time.points = K, Fid.End, Fid.T1)

```

Which is then supplied to create a design matrix that reflects the deviation from perfect implementation by fractional values for each time point and cluster (find the other two functions for slow and fast increase here [Determine different fidelity patterns](#)):

```

# design matrix of a linear fidelity pattern
(fidelMat_prll <- implemMatrix.parallel(
  nC = I, nT = K, nSw = round(I / 2),
  pattern = res.lin[, "Fidelity.Prozent"] / 100
))

```

```

      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]  0.0 0.00 0.00  0.0 0.00 0.00  0.0
[2,]  0.0 0.00 0.00  0.0 0.00 0.00  0.0
[3,]  0.0 0.00 0.00  0.0 0.00 0.00  0.0
[4,]  0.4 0.47 0.53  0.6 0.67 0.73  0.8
[5,]  0.4 0.47 0.53  0.6 0.67 0.73  0.8
[6,]  0.4 0.47 0.53  0.6 0.67 0.73  0.8

```

This matrix reflecting implementation error is then used in the simulation as a reference matrix (note the difference in arguments `X = designMat_prll` and `X.A = fidelMat_prll` within the latter used R-function `simulation()` in section [Simulation experiment](#), which did not differ in the *perfect* simulation).

2.1.3 Model parameter

Data for a cluster randomized trial is sampled from a model provided by the package `{samplingDataCRT}`. Therefore, and in addition to the design parameters, we have to define the following model parameters as well:

- baseline mean of the outcome of interest (e.g. mean quality of life score) of 10
- intervention effect (the change of scores after intervention) of 1

- kind of time trend (effect of each time point during followed time points)
- a between clusters and error variance regarding the outcome of interest that results in an inter-cluster correlation coefficient (ICC)

Define an hypothetical trial with:

- baseline mean of the outcome of interest of 10
- intervention effect of 1
- no time trend
- a between clusters and error variance that results in an ICC of 0.001

as follows:

```
## Model parameter ##

mu.0 <- 10 # Baseline mean of the outcome of interest
theta <- 1 # Intervention effect
betas <- rep(0, K - 1) # no time trend, but could be included

# variability within or error variance (H&H sigma)
sigma.1 <- 2
# variability within clusters, if longitudinal data
sigma.2 <- NULL
# between clusters variability (H&H tau)
sigma.3 <- sigma.1 * sqrt(0.001 / (1 - 0.001))

# resulting ICC
(ICC <- sigma.3^2 / (sigma.3^2 + sigma.1^2))
```

[1] 0.001

Choosing a longitudinal instead of a cross-sectional design for the study demand the specification of a third variance (see for more detail into [Sampling data of a cluster randomized trial with a given design](#)).

DIANA - can we say to recommend these parameters as default values? what does mu.0 mean? How do I (non-statistician) know which values to choose? From a practical point of view I would expect to have a comment on this.

2.1.4 Iterations of the simulation

In addition to the parameters set above, we need to define the number of iterations for the simulation. Its a number how often the sampling and estimation should be repeated to finally calculate the performance measures. The literature recommends between 10,000 and 100,000 repeats for valid results, but this comes at a time cost. To simplify this, we use $n = 1,000$ here.

```
## Number of repeats within the simulation ##  
anzSim <- 10
```

2.1.5 Simulation experiment

Since we set all necessary arguments for the usage of the provided simulation function:

1. Study design
2. Model parameter
3. Indication if there are deviations from perfect situations or not
4. Iterations of the simulation

we can start the simulation by using the provided function `simulation` in package `{fidelitysim}` as follows:

```
# linear increase of fidelity  
res.Simu.parallel.lin <- simulation(  
  anzSim = anzSim, #Simulation parameter  
  type = "cross-sec", K = K, J = J, I = I, #design paramter  
  sigma.1 = sigma.1, sigma.3 = sigma.3,  
  mu.0 = mu.0, theta = theta, betas = betas, #model parameters  
  X = designMat_prll, X.A = fidelMat_prll #design matrices  
)
```

Repeating the simulation 10 times for this design a computational time of 0.03 min is needed.

The output of the simulation provides a total of $n = 22$ results, among the following performance parameters (see also Introduction of this Chapter):

- Bias
- Coverage
- Power

Hence, the mean estimate of the intervention effect from all iterations and the power of the design can be accessed like this:

```
# mean estimated intervention effect
res.Simu.parallel.lin["intervention Mean ."]

intervention Mean .
      0.5754364

# estimated power
(res.Simu.parallel.lin["Power.Intervention"])

Power.Intervention
      0.7

Power.Intervention
      0.7
```

We obtain a power of 0.7 to detect an intervention effect of 1 within an parallel study with 6 clusters followed 7 time points and a cluster size of 10 when an linear increase of the fidelity from 0.4 % to 0.8 is achieved after implementation of the intervention until end of the study.

2.2 Comparison of several Fidelity patterns for a parallel design

See again also for fidelity patterns conduction in [Fidelity patterns]. To compare the effect of different degrees of increasing fidelity after implementation against perfect trials, a repetition the simulation experiment for different degrees of fidelity increase but same settings has to be conducted.

We take again the example from above: cross-sectional parallel trial with ten clusters (5 of them getting the intervention, 5 of them not), 7 time points and ten individuals within each cluster and time point. We also assume a fidelity of 0.4 at the beginning and 0.8 at the end of trial for the intervention group. Here , we investigate on total 7 different fidelity pattern increases, 3 fast increases, 1 linear and 3 slow increases for this design setting.

```
### several Slopes indicating the degree of increase ###
slope.seq<-round(exp(1)^(seq(-2,2,2)),2)
nr.sl<-length(slope.seq)

start_time <- Sys.time()
```

At first, perfect situation simulation (no deviation fro 100% implementation).

```
#####
# perfect implementation
# no individual or cluster miss
#####

#design matrix of perfect situation
X<-samplingDataCRT::designMatrix(nC=I, nT=K, nSw=round(I/2), design="parallel")
res<-fidelitysim::simulation(
  anzSim=anzSim, #Simulation parameter
  type="cross-sec", K=K,J=J,I=I, #design paramter
  sigma.1=sigma.1,sigma.3=sigma.3, #model parameters
  mu.0=mu.0, theta=theta,betas=betas,
  X=X, X.A=X #design matrices
)

#save results
res.Sim.diffFid<-as.data.frame(t(
  c(res,
    D="perfect",
    slope=0, sort=1, Fid.Begin=1, Fid.END=1)
  ))

```

Next for a slow increase (reflected by an exponential function):

```
###all the other patterns
res.Simu<-NULL

#exponential increase
for(sl in 1:nr.sl){#for each slope

  #Fidelity pattern
  res.exp<-fidelitysim::find.Fidelity.exp(time.points=K,
    Fid.End, Fid.T1,
    par.slope=slope.seq[sl])

  #new design matrix
  A1.exp <-fidelitysim::implemMatrix.parallel(nC=I, nT=K, nSw=round(I/2),
    pattern=res.exp[, "Fidelity.Prozent"]/100)

  #simulation experiment
  res<-fidelitysim::simulation(anzSim=anzSim,
    type="cross-sec", K=K,J=J,I=I,
    sigma.1=sigma.1,sigma.3=sigma.3,

```

```

        mu.0=mu.0, theta=theta,betas=betas,
        X=X, X.A=A1.exp
    )
#save results
res.Simu<-rbind(res.Simu,
               as.data.frame(t(c(
                   res,
                   D="exp", slope=slope.seq[s1], sort=2+nr.s1+(nr.s1-s1+1),
                   Fid.Begin=Fid.T1, Fid.END=Fid.END)))
    )
}

```

For a fast increase (reflected by a logarithmic function):

```

#logistic increase
for(s1 in 1:nr.s1){#for each slope
    #Fidelity pattern
    res.log<-fidelitysim::find.Fidelity.log(time.points=K,
                                           Fid.End, Fid.T1,
                                           par.slope=slope.seq[s1])

    #new design matrix
    A1.log <-fidelitysim::implemMatrix.parallel(nC=I, nT=K, nSw=round(I/2),
                                              pattern=res.log[, "Fidelity.Prozent"]/100)

    #simulation experiment
    res<-fidelitysim::simulation(anzSim=anzSim,
                                type="cross-sec", K=K,J=J,I=I,
                                sigma.1=sigma.1,sigma.3=sigma.3,
                                mu.0=mu.0, theta=theta,betas=betas,
                                X=X, X.A=A1.log)

    #save results
    res.Simu<-rbind(res.Simu,
                   as.data.frame(t(c(
                       res,
                       D="log", slope=slope.seq[s1], sort=1+s1
                       , Fid.Begin=Fid.T1, Fid.END=Fid.END)))
    )
}

```

For linear increase:

```

##linear increase
m<-(Fid.T1-Fid.End)/(1-(K-1))
#Fidelity pattern
res.lin<-fidelitysim::find.Fidelity.linear(time.points=K, Fid.End, Fid.T1)
#ne design matrix
A1.lin <-fidelitysim::implemMatrix.parallel(nC=I, nT=K, nSw=round(I/2),
                                           pattern=res.lin[, "Fidelity.Prozent"]/100)

#simulation experiment
res<-fidelitysim::simulation(anzSim=anzSim,
                             type="cross-sec", K=K, J=J, I=I,
                             sigma.1=sigma.1, sigma.3=sigma.3,
                             mu.0=mu.0, theta=theta, betas=betas,
                             X=X, X.A=A1.lin)

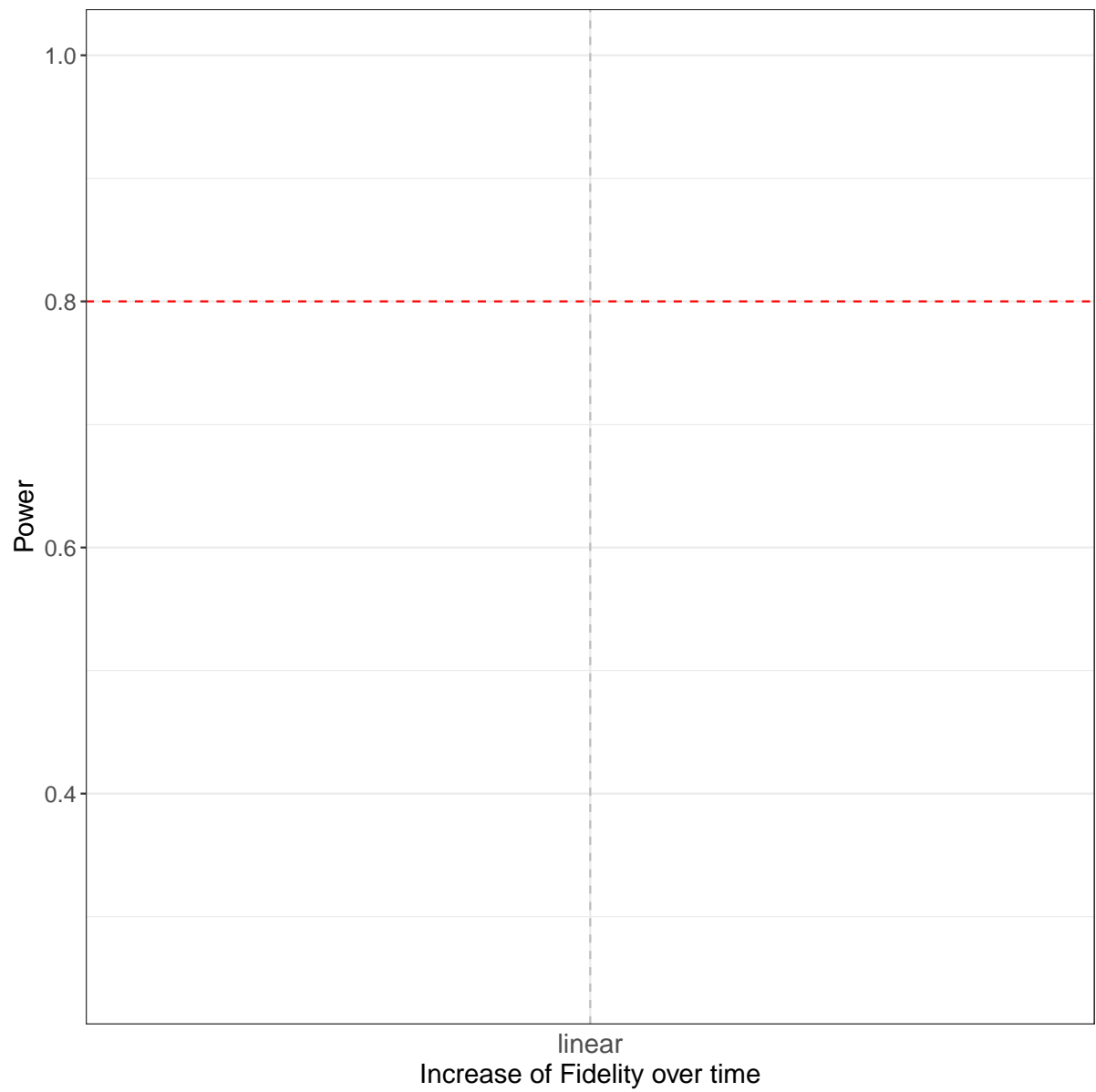
#save results
res.Simu<-rbind(res.Simu,
                as.data.frame(t(c(
                    res,
                    D="linear", slope=0, sort=2+nr.sl
                    , Fid.Begin=Fid.T1, Fid.END=Fid.End)))
                )

```

Needs time.3 minutes time.

The figure summarize the results of the several simulations for different fidelity patterns regarding power of the study.

Warning: Removed 8 rows containing missing values (geom_point).



3 Detailed statistical information

This chapter offers additional statistical information.

1. Steps within one simulation experiment
2. How to conduct design matrices
3. How to determine fidelity patterns and corresponding design matrices

3.1 Explanation of simulation experiment and provided corresponding function

For a simulation experiment like in [Tutorial] it is not necessary to understand the whole statistical modeling background, which is figured out here.

Each simulation step includes three necessary steps:

- Determining the design matrix regarding the specified design
- Sampling data
- Effect estimation from the data

and will be explained with more detail in the following subsections.

For the first 2 steps the provided R-package {samplingDataCRT} v1.0 (Trutschel and Treutler 2017) is needed, for the last {lme4}. How to install and load see [Introduction](#). The first steps is done once and the latter two are used repeatedly through one simulation experiment within each scenario setting specified in step 1.

3.1.1 Determining the design matrix regarding the chosen design

To specify a cluster randomized study the following parameter has to be determined:

- Number of clusters (e.g. hospitals, nursing homes, ...) obtained through the study
- Number of time points the clusters are followed
- Cluster size refers to the number of individuals obtained within one cluster
- Study design: parallel design or Stepped wedge design (cross-over as well possible)

- Study type: cross-sectional if individuals could be different within the cluster between time points or longitudinal if individuals are followed over time

Based on the example in the main article, we specify a hypothetical example including parameter settings for a reference setup of the simulation experiment: *a cross-sectional stepped wedge cluster randomized trial with 10 nursing homes, 6 time points and 10 individuals within each nursing home and time point* (see main article). All parameters can be adapted to an own practical example. The resulting reference design matrix can be create manually (by create a corresponding matrix) or by the provided R-function `designMatrix()`.

```
#####
#using the parameter setting of Table 1 in the article
#####

## Design ##
#####
K<-7 #number of time points
I<-6 #number of cluster
Sw<-1 #number of cluster switches per time point can be manually set
J<-10 #Subjects = Number of individuals per cluster

#design matrix of "SWD" with given setting
(X<-samplingDataCRT::designMatrix(nC=I, nT=K, nSw=Sw))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0	1	1	1	1	1	1
[2,]	0	0	1	1	1	1	1
[3,]	0	0	0	1	1	1	1
[4,]	0	0	0	0	1	1	1
[5,]	0	0	0	0	0	1	1
[6,]	0	0	0	0	0	0	1

3.1.2 Sampling data of a cluster randomized trial with a given design

To sample data for a cluster randomized trial using sampling from a **multivariate normal distribution** provided by the package `{samplingDataCRT}`, several model parameter has also be specified:

- Baseline mean of the outcome of interest
- Intervention effect (change/difference in mean outcome) by implementing the intervention)

- Time trend effects
- Variance within the multilevel data: between clusters, between individuals, within individuals which provide an estimate of the intra-cluster correlation coefficient (ICC) as a measure of dependencies within clusters.

In the example here we set the mean outcome to 10 (e.g. the mean population value of measured quality of life), the intervention effect which aimed to change quality of life of 1, and no time trends. With a given between cluster variance of `sigma.3` and a error variance of `sigma.1`, the resulting ICC is `ICC`.

```
## Model parameter ##
#####
mu.0<- 10           # Baseline mean of the outcome of interest
theta <- 1           # intervention effect
betas<-rep(0, K-1)   # no Time trend, but could be included

# variability within or error variance (H&H sigma)
sigma.1<-2
# variability within clusters, if longitudinal data
sigma.2<-NULL
# between clusters variability (H&H tau)
sigma.3<-sigma.1*sqrt(0.001/(1-0.001))

#resulting ICC
(ICC<-sigma.3^2/(sigma.3^2+sigma.1^2))
```

```
[1] 0.001
```

To note, by choosing **longitudinal or cross-sectional design**, we need to specify in the first case 3, in the second only two **variances** regarding using a **three- instead of a two-level hierarchical experiment** with the following meaning:

- three-level design (longitudinal data):
 - between clusters variability
 - within cluster (or between individuals) variability
 - within individuals (or error) variability
- two-level design (cross-sectional data):
 - between clusters variability [σ put this reference here Hussey and Hughes 2007]
 - within cluster (or error) variability [τ put this reference here Hussey and Hughes 2007]

A complete data set for a special design with a given setup can be sampled by the given function R-function {sampleData()}. Therefore, the complete data design matrix and the covariance-variance matrix for the data given the design are also needed to be specified, which is provided by the functions R-function {completeDataDesignMatrix()} and R-function {CovMat.Design()}. The complete data design matrix has the size of (Number of cluster x cluster size x Number of time points) rows and (Number of model parameters) columns. To sample the data from for a cluster randomized trial, it has additionally be specified, if the individuals are followed over time (longitudinal design) or not (cross-sectional design).

```
#complete data design matrix
D<-samplingDataCRT::completeDataDesignMatrix(J, X)
(dim(D))
```

```
[1] 420    8
```

```
#covariance-variance matrix for the data given the design
V<-samplingDataCRT::CovMat.Design(K, J, I, sigma.1=sigma.1, sigma.3=sigma.3)
dim(V)
```

```
[1] 420 420
```

```
#corresponding fixed effects in linear mixed model
parameters<-c(mu.0, betas, theta)

#sample complete data given the setup
# study design type = cross-sectional
type<-"cross-sec"
sample.data<-samplingDataCRT::sampleData(type = type, K=K,J=J,I=I, D=D, V=V, parameters=pa
```

To validate the number of observations provided by the sampling method a summary of the data can be conducted.

```
dim(sample.data)
```

```
[1] 420    5
```

```
#show the number of observations within the SWD
xtabs(~cluster+measurement, data=sample.data)
```

```
      measurement
cluster 1  2  3  4  5  6  7
1  10 10 10 10 10 10 10
2  10 10 10 10 10 10 10
3  10 10 10 10 10 10 10
4  10 10 10 10 10 10 10
5  10 10 10 10 10 10 10
6  10 10 10 10 10 10 10
```

3.1.3 Effect estimation from the data

The sampled data can then be analyzed by a **linear mixed model** with the function R-function{lmer()} of the package R-package {lme4}, hence the parameter of the model will be estimated.

```
#analysis of the two-level data by a linear mixe model
lme4::lmer(val~intervention+measurement + (1|cluster), data=sample.data)
```

```
Linear mixed model fit by REML ['lmerMod']
Formula: val ~ intervention + measurement + (1 | cluster)
Data: sample.data
REML criterion at convergence: 1412.276
Random effects:
Groups   Name             Std.Dev.
cluster (Intercept) 0.2524
Residual                  1.2812
Number of obs: 420, groups: cluster, 6
Fixed Effects:
(Intercept) intervention measurement2 measurement3 measurement4
10.165903      1.111540      -0.051636      -0.127315      -0.363441
measurement5 measurement6 measurement7
-0.119169      -0.628540      -0.008076
```

This process of sampling data and effect estimation from the data will be used repeatedly through the simulation experiment within each scenario setting.

3.2 Determine different design matrices

The function `designMatrix` of the R-package `{samplingDataCRT}` provides the design matrix of cluster randomized trials with several study designs, given the number of clusters, time points and cluster size assumed for the trial. We show here for two examples:

- parallel
- SWD

3.2.1 Parallel cluster randomized trials

Within the function the `design` argument is set 'parallel', and the parameter `nSw` indicates the number of clusters are being control group.

```
## Design paramter ##
#####
I <- 6 # number of clusters
J <- 10 # number of individuals per cluster (used later in simulation-step)
K <- 7 # number of time points
Sw <- round(I / 2) # number of cluster within the control group

# design matrix for parallel design
(designMat_prll <- samplingDataCRT::designMatrix(nC = I, nT = K, nSw = round(I / 2), design = "parallel"))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0	0	0	0	0	0	0
[2,]	0	0	0	0	0	0	0
[3,]	0	0	0	0	0	0	0
[4,]	1	1	1	1	1	1	1
[5,]	1	1	1	1	1	1	1
[6,]	1	1	1	1	1	1	1

3.2.2 Stepped wedge cluster randomized trial

Whereas the design parameters can be the same as in the parallel study the design matrix differs accordingly when changing the argument `design = "SWD"` (which is also set by default, so it has not to be specified). The parameter 'nSw' indicates here the number of cluster switches per time point from control to intervention group.

```
## Design matrix ##
```

```
I <- 6 # number of clusters
```

```
J <- 10 # number of individuals per cluster (used later in simulation-step)
```

```
K <- 7 # number of time points
```

```
Sw <- 1 # number of cluster switches per time point can be manually set
```

```
# design matrix of "SWD"
```

```
(designMat_SWD <- samplingDataCRT::designMatrix(nC = I, nT = K, nSw = Sw, design = "SWD"))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0	1	1	1	1	1	1
[2,]	0	0	1	1	1	1	1
[3,]	0	0	0	1	1	1	1
[4,]	0	0	0	0	1	1	1
[5,]	0	0	0	0	0	1	1
[6,]	0	0	0	0	0	0	1

3.3 Determine different fidelity patterns

Fidelity refers to the degree to which an intervention was implemented as it was prescribed or intended. We aim to include different patterns of how fidelity might increase over time to estimate the respective effects on power of the study. To describe hypothetical fidelity patterns of increasing fidelity (slow, linear, fast) different mathematical functions (i.e. logistic, linear and exponential curves) are implemented. By considering different values for the slope parameter we can cover a range of fidelity patterns. The slope parameter ranges from $(0, \infty)$, where a slope parameter near to 0 indicates a increase far away from a linear (fast increase upper left corner, slow increase right bottom corner curve) and a great slope parameter near to linear (See figure Figure 3.1).

```
library(ggplot2)
```

```
library(gridExtra)
```

```
#study design
```

```
#####
```

```
#number of measurement
```

```
K<-7
```

```
#points of time afterintervention
```

```
T.points<-K-1
```



```

#parameter Fidelity specification
#####
Fid.T1<-0.2
Fid.End<-1

####set several slopes of increasing fidelity
#####
slope.seq<-round(exp(1)^(seq(-2,2,1)),2)
nr.sl<-length(slope.seq)

####fidelity patterns determined by several slopes within slow and fast increase
#####
res.plot.Patterns<-NULL
for(sl in slope.seq){
  res<-fidelitysim::find.Fidelity.log(time.points=T.points, Fid.End, Fid.T1, par.slope=sl)
  res<-data.frame(res, FUN="log", slope=sl)
  res.plot.Patterns<-rbind(res.plot.Patterns, res)
  res<-fidelitysim::find.Fidelity.exp(time.points=T.points, Fid.End, Fid.T1, par.slope=sl)
  res<-data.frame(res, FUN="exp", slope=sl)
  res.plot.Patterns<-rbind(res.plot.Patterns, res)
}

#fidelity pattern for linear increase
#####
res.lin<-fidelitysim::find.Fidelity.linear(time.points=T.points, Fid.End, Fid.T1)
res.plot.Patterns<-rbind(res.plot.Patterns, data.frame(res.lin, FUN="linear", slope=1))

```

For our calculation within the simulation we use then these determined fractional values of intervention effects to define the degree of deviation from 100% implementation within the design matrix. Technically 3 functions in the R-package {fidelitysim} provides the implementation of the different patterns of fidelity. To create a corresponding design matrix using the fidelity patterns also a function is provided for each design.

3.3.1 Slow increase of fidelity using exponential function

Here we show an example how to determine the design matrix for a cluster randomized **parallel design** study (with same design as above) with existing implementation error, where fidelity starts with **40% after implementation** and reach after a slow increase **80% at the end** of the study. A great slope parameter is chosen which reflect a **slow increase more closed to the linear** increase.

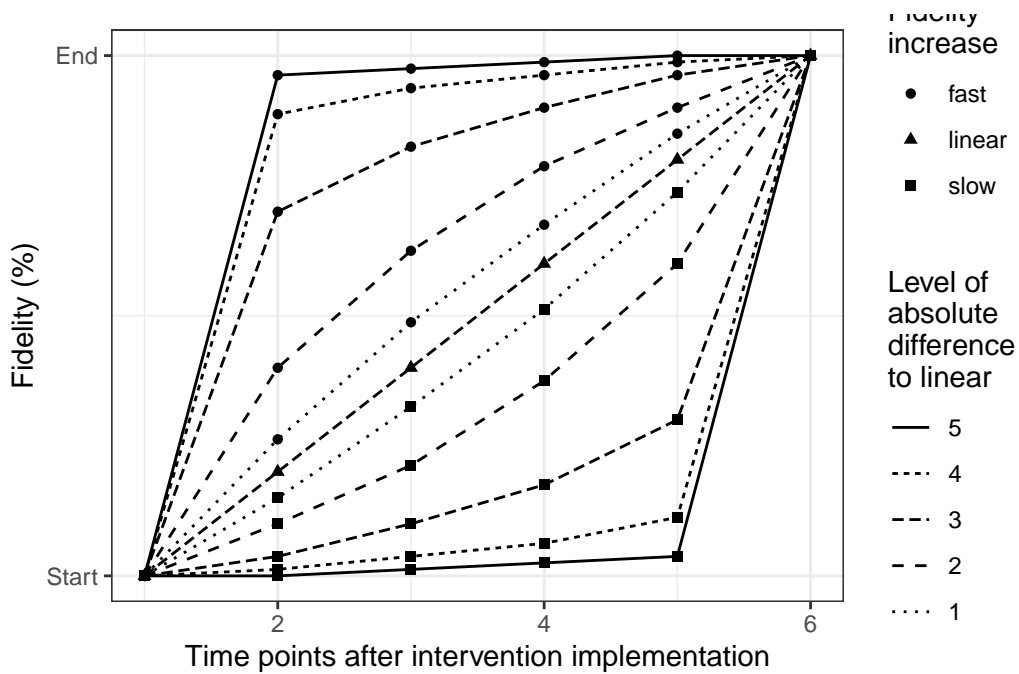


Figure 3.1: Patterns of a fidelity increase (fast, linear or slow) over 6 times points

```
#study design
#####
I <- 6 # number of clusters
J <- 10 # number of individuals per cluster (used later in simulation-step)
K <- 7 # number of time points
Sw<- round(I/2) # number of cluster within the control group

#parameter Fidelity specification
#####
Fid.T1<-0.4
Fid.End<-0.8

#parameter tunes the slope for the log and exp functions
slope.seq<-5

#exponential function to determine slow increase
(res.exp<-fidelitysim::find.Fidelity.exp(time.points=K, Fid.End, Fid.T1, par.slope=slope.s
```

```
time Fidelity.Prozent
[1,] 1 40
```

[2,]	2	44
[3,]	3	49
[4,]	4	55
[5,]	5	62
[6,]	6	70
[7,]	7	80

```
#determine corresponding design matrix
(A1.exp <-fidelitiesim::implemMatrix.parallel(nC=I, nT=K, nSw=Sw,
                                             pattern=res.exp[, "Fidelity.Prozent"]/100))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0.0	0.00	0.00	0.00	0.00	0.0	0.0
[2,]	0.0	0.00	0.00	0.00	0.00	0.0	0.0
[3,]	0.0	0.00	0.00	0.00	0.00	0.0	0.0
[4,]	0.4	0.44	0.49	0.55	0.62	0.7	0.8
[5,]	0.4	0.44	0.49	0.55	0.62	0.7	0.8
[6,]	0.4	0.44	0.49	0.55	0.62	0.7	0.8

3.3.2 Linear increase of fidelity using linear function

Here we show an example how to determine the design matrix for a cluster randomized **parallel design** study (with same design as above) with existing implementation error, where fidelity starts with **40% after implementation** and reach after a **linear increase 80% at the end** of the study.

```
#parameter Fidelity specification
#####
Fid.T1<-0.4
Fid.End<-0.8

#slope for linear function
m<-(Fid.T1-Fid.End)/(1-(K-1))
#linear increase
(res.lin<-fidelitiesim::find.Fidelity.linear(time.points=K, Fid.End, Fid.T1))
```

	time	Fidelity.Prozent
[1,]	1	40

[2,]	2	47
[3,]	3	53
[4,]	4	60
[5,]	5	67
[6,]	6	73
[7,]	7	80

```
# design matrix of a learning implementation pattern, linear
(A1.lin <-fidelitysim::implemMatrix.parallel(nC=I, nT=K, nSw=round(I/2),
      pattern=res.lin[, "Fidelity.Prozent"]/100))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0.0	0.00	0.00	0.0	0.00	0.00	0.0
[2,]	0.0	0.00	0.00	0.0	0.00	0.00	0.0
[3,]	0.0	0.00	0.00	0.0	0.00	0.00	0.0
[4,]	0.4	0.47	0.53	0.6	0.67	0.73	0.8
[5,]	0.4	0.47	0.53	0.6	0.67	0.73	0.8
[6,]	0.4	0.47	0.53	0.6	0.67	0.73	0.8

3.3.3 Fast increase of fidelity using logarithmic function

Here we show an example how to determine the design matrix for a cluster randomized **stepped wedge design** study (with same design as above) with existing implementation error, where fidelity starts with **20% after implementation** and reach after a fast increase **100% at the end** of the study. A small slope parameter is chosen for determining the fidelity curve, which reflect a **very fast increase**.

```
Sw <- 1 # number of cluster switches per time point can be manually set

#parameter Fidelity specification
#####
Fid.T1<-0.2
Fid.End<-1

#parameter tunes the slope for the log and exp functions
slope.seq<-0.2

#logistic function to determine fast increase
(res.log<-fidelitysim::find.Fidelity.log(time.points=K-1, Fid.End, Fid.T1, par.slope=slope
```

	time	Fidelity.Prozent
[1,]	1	20
[2,]	2	95
[3,]	3	97
[4,]	4	98
[5,]	5	99
[6,]	6	100

```
(A1.log <-samplingDataCRT::implemMatrix.SWD(nC=I, nT=K, nSw=Sw,
                                             pattern=res.log[, "Fidelity.Prozent"]/100))
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	0	0.2	0.95	0.97	0.98	0.99	1.00
[2,]	0	0.0	0.20	0.95	0.97	0.98	0.99
[3,]	0	0.0	0.00	0.20	0.95	0.97	0.98
[4,]	0	0.0	0.00	0.00	0.20	0.95	0.97
[5,]	0	0.0	0.00	0.00	0.00	0.20	0.95
[6,]	0	0.0	0.00	0.00	0.00	0.00	0.20

References

- Bates, Douglas, Martin Mächler, Ben Bolker, and Steve Walker. 2015. “Fitting Linear Mixed-Effects Models Using lme4.” *Journal of Statistical Software* 67 (1): 1–48. <https://doi.org/10.18637/jss.v067.i01>.
- R Core Team. 2022. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. <https://www.R-project.org/>.
- Trutschel, Diana, and Catherine Blatter. 2022. *fidelitysim: Simulations Experiments for Cluster Randomized Trials Having Fidelity Patterns for Implementation Science Research*.
- Trutschel, Diana, and Hendrik Treutler. 2017. *samplingDataCRT: Sampling Data Within Different Study Designs for Cluster Randomized Trials*. <https://CRAN.R-project.org/package=samplingDataCRT>.
- Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. <https://ggplot2.tidyverse.org>.