Modeling complex longitudinal data in a quick and easy way

```
library(Mmcsd)
library(simstudy)
#> Warning: package 'simstudy' was built under R version 4.1.3
library(kableExtra)
#> Warning: package 'kableExtra' was built under R version 4.1.3
library(tidyverse)
#> Warning: package 'tidyverse' was built under R version 4.1.3
#> Warning: package 'ggplot2' was built under R version 4.1.3
#> Warning: package 'tibble' was built under R version 4.1.3
#> Warning: package 'tidyr' was built under R version 4.1.3
#> Warning: package 'readr' was built under R version 4.1.3
#> Warning: package 'purrr' was built under R version 4.1.3
#> Warning: package 'dplyr' was built under R version 4.1.3
#> Warning: package 'stringr' was built under R version 4.1.3
#> Warning: package 'forcats' was built under R version 4.1.3
#> Warning: package 'lubridate' was built under R version 4.1.3
#> -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
#> v dplyr     1.1.0     v readr     2.1.4
#> v forcats   1.0.0     v stringr   1.5.0
#> v ggplot2   3.4.1     v tibble    3.2.0
#> v lubridate 1.9.2     v tidyr     1.3.0
#> v purrr     1.0.1
#> -- Conflicts ------------------------------------------ tidyverse_conflicts() --
#> x dplyr::filter()     masks stats::filter()
#> x dplyr::group_rows() masks kableExtra::group_rows()
#> x dplyr::lag()        masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

Complex longitudinal data and the package

A survey is a method of collecting information from an audience with the aim of learning more about them. What you learn can be used to make better business decisions or inform research.

Among the various sampling methods, the package focuses on modeling complex longitudinal data.

Longitudinal observations consist of repeated measurements on the same units over a number of occasions, with fixed or varying time spells between the occasions. Complex sampling plans are usually the result of a combination of SRS with and without replacement with other sampling plan(s), such as: Stratified sampling; Conglomerate sampling; e Sampling with unequal probabilities; among others

Mmcsd is a package to deal with complex longitudinal survey data. It combines longitudinal methodology models with complex sampling design. It fits fixed and random effects models and covariance structured models so far. It also provides tools to perform statistical tests considering these specifications.

Functional programming

First, we emphasize that the package used the concepts of functional programming during its development, aiming to optimize the search for errors and reduce the execution time of each function.

We will follow the following tripod:

- Small functions
  - Practice of separating the algorithm into increasingly simple functions that will naturally reduce the size of each function
- Function and purpose
  - Functions must have a single purpose, that is, it is not enough to simply divide a function because of its size, when partitioning a function into smaller ones, the focus must be precisely on the unique output parameter.
- Encapsulation
  - The function available for use is the call of other minor functions

Package Avaliable Functions

1 - Defining

The package has the following functions available to users currently

- **mmcsd**
  - Estimate the fixed effects of the model, also known as **Beta** parameters of the regression,taking into account the sampling plan of the research, and also estimating the covariance matrix of the model considering the estimates of **Beta**
- **cov_mmcsd**
  - This function is responsible for performing the modeling of the model's covariance matrix through the use of covariance structures.
- **sigmaThetaExpr_viewer**
  - Knowing the difficulty of visualizing the covariance structure, especially when the user chooses to determine his own structure. This function was developed,that allows the user to view the provided structure even before it is evaluated, that is, through mathematics symbolic.

2 - How to call it

The following commands are also available using the **help()** function. For a better understanding of each function, we will explain each input parameter.

- We recommend using the functions in the following sequence :

1. *mmcsd*
2. *sigmaThetaExpr_viewer*
3. *cov_mmcsd*

- **mmcsd**

```
mmcsd=function(formula, waves, ids, weights, stratum, cluster, data, sigma = "identity")
```

formula: A formula type struct. Represents the shape of the explanatory variables

waves: A dataframe column or an array. Represents the waves(rounds) of the longitudinal data

ids: A dataframe column or an array. Represents the id of each variable

data: A dataframe or tibble. It must be the structure that contains the data

sigma: The sigma argument can be: **identity**, **exchangeable**, **autorregressive** or a **custom square matrix**.

- Complex plane inputs

  weights: A dataframe column or an array. Represents the weight of each variable, if this has been used by the researcher in the complex sampling method

  stratum: A dataframe column or an array. Represents the data stratification, if this has been used by the researcher in the complex sampling method

  cluster: A dataframe column or an array. Represents the data clustering, if this has been used by the researcher in the complex sampling method

- **sigmaThetaExpr_viewer**

```
sigmaThetaExpr_viewer=function(sigmaThetaExpr, numWaves = NULL)
```

sigmaThetaExpr: A character with the covariance structure type or a list of expressions. The user has the following options available: **UCM**, **AR1**, **ARMA11**, **ARH1**, **Heterogeneous UCM**, **Heterogeneous Toeplitz**, **Unstructured** and **Predependence Order 1**

numWaves: An integer with the size of the square matrix to be printed.

- **cov_mmcsd**

```
cov_mmcsd=function(fit, fittingType, sigmaThetaExpr, optimParams)
```

fit: A fit model with class 'mmcsd' (The object returned by the mmcsd function)

fittingType:

sigmaThetaExpr: A character with the covariance structure type or a list of expressions. We recommend using the *sigmaThetaExpr_viewer* function first, which facilitates the visualization and choice of the model

optimParams: A list with configuration for optim function. 'Par' is required.

Examples

British Household Panel Survey - BHPS

BHPS is a panel-type longitudinal household survey and had its first round in 1991. sample was selected by a two-stage stratified sampling design with conglomeration by postal sectors. These postal sectors were selected in a systematic and considering selection probabilities proportional to its size, or that is, the number of households in the same postal sector. Then, in each of selected postal sectors, three households were chosen at random and within from each household, all individuals were surveyed (Vieira, 2012).

First, the **mmcsd()** function was used, where first, using the *formula* type, the response variable and its explanatory variables were defined, after which the complex sampling plan used in the research was defined

```r
fit <- mmcsd(
  score ~ wave + ageg + ecacg + qualifg,
  waves = wave, ids = id,
  weights = weight, stratum = strata, cluster = cluster,
  data = example_data, sigma = "exchangeable"
)
```

After that, the **sigmaThetaExpr_viewer()** function was used to visualize the covariance structure, where the UCM and the $AR1$ type were the options selected

```r
sigmaThetaExpr_viewer("UCM", 5)
#> \begin{table}
#>
#> \caption{\label{tab:unnamed-chunk-6}sigma2u sigma2v}
#> \centering
#> \begin{tabular}[t]{l|c|c|c|c|c}
#> \hline
#>   & T1 & T2 & T3 & T4 & T5\\
#> \hline
#> T1 & sigma2u + sigma2v & sigma2u & sigma2u & sigma2u & sigma2u\\
#> \hline
#> T2 & sigma2u & sigma2u + sigma2v & sigma2u & sigma2u & sigma2u\\
#> \hline
#> T3 & sigma2u & sigma2u & sigma2u + sigma2v & sigma2u & sigma2u\\
#> \hline
#> T4 & sigma2u & sigma2u & sigma2u & sigma2u + sigma2v & sigma2u\\
#> \hline
#> T5 & sigma2u & sigma2u & sigma2u & sigma2u & sigma2u + sigma2v\\
#> \hline
#> \end{tabular}
#> \end{table}
```

```r
sigmaThetaExpr_viewer("AR1", 5)
#> \begin{table}
#>
#> \caption{\label{tab:unnamed-chunk-7}sigma2u sigma2v gamma}
#> \centering
#> \begin{tabular}[t]{l|c|c|c|c|c}
#> \hline
#>   & T1 & T2 & T3 & T4 & T5\\
#> \hline
#> T1 & sigma2u + sigma2v & sigma2u + gamma\textasciicircum{}1 * sigma2v & sigma2u + gamma\textasciicir
#> \hline
#> T2 & sigma2u + gamma\textasciicircum{}1 * sigma2v & sigma2u + sigma2v & sigma2u + gamma\textasciicir
#> \hline
#> T3 & sigma2u + gamma\textasciicircum{}2 * sigma2v & sigma2u + gamma\textasciicircum{}1 * sigma2v & s
#> \hline
#> T4 & sigma2u + gamma\textasciicircum{}3 * sigma2v & sigma2u + gamma\textasciicircum{}2 * sigma2v & s
#> \hline
#> T5 & sigma2u + gamma\textasciicircum{}4 * sigma2v & sigma2u + gamma\textasciicircum{}3 * sigma2v & s
#> \hline
#> \end{tabular}
#> \end{table}
```

	RMR	AGFI	N_Params
UCM_structure	0.8795714	0.9225445	2
AR1_structure	0.5132369	0.9790654	3

Finally, the **fitTheta()** function was used to model the covariance matrix, using the UCM and the $AR1$ covariance structure, chosen from the **sigmaThetaExpr_viewer()** function.

```
fitTheta_ucm <- cov_mmcsd(fit,
  fittingType = "PML", sigmaThetaExpr = "UCM",
  optimParams = list(par = c(7, 5))
)


fitTheta_ar1 <- cov_mmcsd(fit,
  fittingType = "PML", sigmaThetaExpr = "AR1",
  optimParams = list(par = c(0.8, 1,0.8))
)
```

Therefore, in this real world problem example, the 2 models performed very well. It is possible to see that the $AR1$ model performed a little better, yet needs more optimization parameters, and then the choice of the best model should be decided by the researcher.

Using the UCM model as an example, this is the sigma matrix returned by the cov_mmcsd function

```
write_matex(fitTheta_ucm$sigmaTheta)
```

$$\begin{bmatrix} 12.30367943086 & 7.15878432824834 & 7.15878432824834 & 7.15878432824834 & 7.15878432824834 \\ 7.15878432824834 & 12.30367943086 & 7.15878432824834 & 7.15878432824834 & 7.15878432824834 \\ 7.15878432824834 & 7.15878432824834 & 12.30367943086 & 7.15878432824834 & 7.15878432824834 \\ 7.15878432824834 & 7.15878432824834 & 7.15878432824834 & 12.30367943086 & 7.15878432824834 \\ 7.15878432824834 & 7.15878432824834 & 7.15878432824834 & 7.15878432824834 & 12.30367943086 \end{bmatrix}$$

Using a longitudinal dataset simulated by the 'simstudy' package

In this second example, we used simulated data from the **defData()** and **genData()** functions, available in the **simstudy** package. It was considered that the explanatory variables follow a normal distribution.

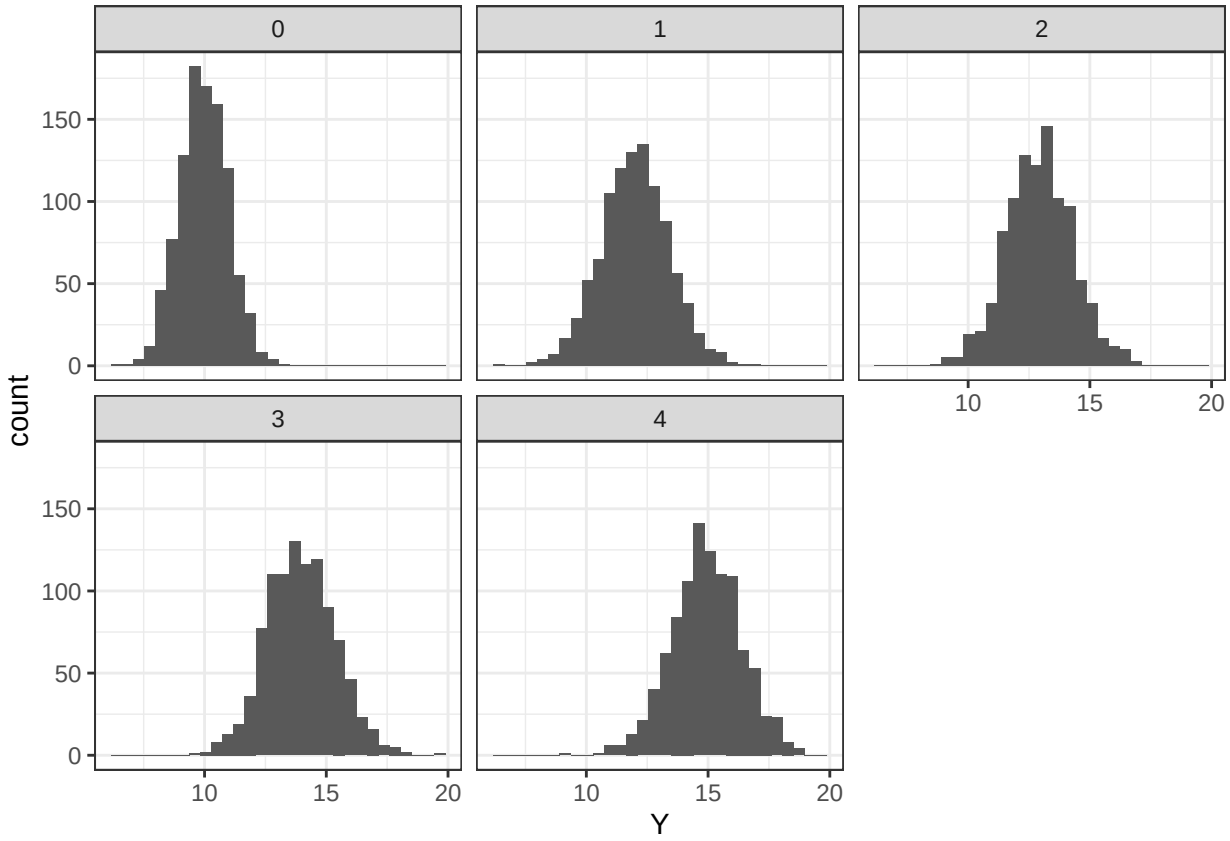
In both examples, a simple sample plan, and so, there is no need to define sampling plan

Linear formula population

First, the population was defined, where the normality of the observations was assumed.

```
set.seed(1108)

tdef <- defData(varname = "T", dist = "binary", formula = 0.5)
tdef <- defData(tdef, varname = "Y0", dist = "normal", formula = 10, variance = 1)
tdef <- defData(tdef, varname = "Y1", dist = "normal", formula = "Y0 + 2",
                variance = 1)
tdef <- defData(tdef, varname = "Y2", dist = "normal", formula = "Y0 + 3",
```

```
                variance = 1)
tdef <- defData(tdef, varname = "Y3", dist = "normal", formula = "Y0 + 4",
                variance = 1)
tdef <- defData(tdef, varname = "Y4", dist = "normal", formula = "Y0 + 5",
                variance = 1)



dtTrial <- genData(1000, tdef)

dtTime <- addPeriods(dtTrial, nPeriods = 5, idvars = "id", timevars = c("Y0", "Y1",
                                                                        "Y2", "Y3", "Y4"), timevarName =

dtTime %>%
  ggplot(aes(x = Y)) + geom_histogram() + facet_wrap(~period) + theme_bw()
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
fit=mmcsd(Y~period,ids = id,waves =  period, data=dtTime,sigma = 'exchangeable' )
```

Testing UCM and $AR1$ as covariance structures

```
fitTheta_ucm <- cov_mmcsd(fit,
                      fittingType = "PML", sigmaThetaExpr = "UCM",
                      optimParams = list(par = c(1, 1))
```

	RMR	AGFI	N_Params
UCM_structure	0.2254495	0.8775235	2
AR1_structure	0.2245428	0.8641051	3

```
)

fitTheta_ar1 <- cov_mmcsd(fit,
                          fittingType = "PML", sigmaThetaExpr = "AR1",
                          optimParams = list(par = c(1, 1, -0.5))
)
```

Therefore, in a population of constant form, the 2 models performed very well. It is possible to see that the UCM model performed a little better, and needs fewer optimization parameters, and thus becomes the most suitable model for the situation.

So this is the sigma matrix returned by the cov_mmcsd function using the UCM model

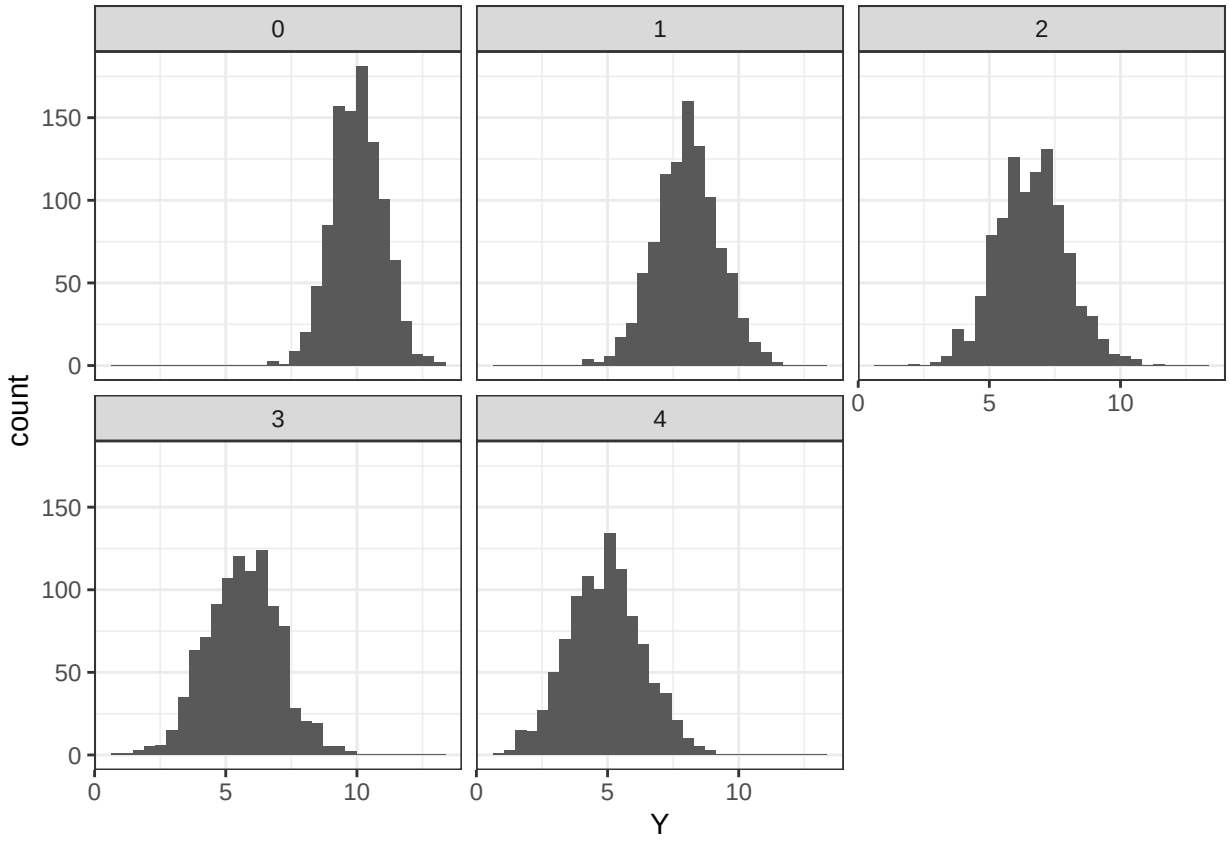
```
write_matex(fitTheta_ucm$sigmaTheta)
```

$$\begin{bmatrix} 1.84073917063395 & 0.904314641185721 & 0.904314641185721 & 0.904314641185721 & 0.904314641185721 \\ 0.904314641185721 & 1.84073917063395 & 0.904314641185721 & 0.904314641185721 & 0.904314641185721 \\ 0.904314641185721 & 0.904314641185721 & 1.84073917063395 & 0.904314641185721 & 0.904314641185721 \\ 0.904314641185721 & 0.904314641185721 & 0.904314641185721 & 1.84073917063395 & 0.904314641185721 \\ 0.904314641185721 & 0.904314641185721 & 0.904314641185721 & 0.904314641185721 & 1.84073917063395 \end{bmatrix}$$

Memoryless population

```
tdef <- defData(varname = "T", dist = "binary", formula = 0.5)
tdef <- defData(tdef, varname = "Y0", dist = "normal", formula = 10, variance = 1)
tdef <- defData(tdef, varname = "Y1", dist = "normal", formula = "Y0 * 0.7 + 1",
                variance = 1)
tdef <- defData(tdef, varname = "Y2", dist = "normal", formula = "Y1 * 0.7 + 1",
                variance = 1)
tdef <- defData(tdef, varname = "Y3", dist = "normal", formula = "Y2 * 0.7 + 1",
                variance = 1)
tdef <- defData(tdef, varname = "Y4", dist = "normal", formula = "Y3 * 0.7 + 1",
                variance = 1)



dtTrial <- genData(1000, tdef)

dtTime <- addPeriods(dtTrial, nPeriods = 5, idvars = "id", timevars = c("Y0", "Y1",
                                                                        "Y2", "Y3", "Y4"), timevarName =


dtTime %>%
  ggplot(aes(x = Y)) + geom_histogram() + facet_wrap(~period) + theme_bw()
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

	RMR	AGFI	N_Params
UCM_structure	0.5025443	0.6810002	2
AR1_structure	0.3669071	0.8567735	3




```
fit=mmcsd(Y~period,ids = id,waves =  period, data=dtTime,sigma = 'exchangeable' )
```

Testing UCM and $AR1$ as covariance structures

```
fitTheta_ucm <- cov_mmcsd(fit,
                      fittingType = "PML", sigmaThetaExpr = "UCM",
                      optimParams = list(par = c(1, 1))
)

fitTheta_ar1 <- cov_mmcsd(fit,
                      fittingType = "PML", sigmaThetaExpr = "AR1",
                      optimParams = list(par = c(1, 1, -0.5))
)
```

In this example, where the population has the memoryless property, the $AR1$ model has a performance considerably superior to the UCM model, and thus becomes the most suitable model for this type of population. This can be explained by the fact that the population, when generated, has an autoregressive structure. This can be observed by the formula argument in the 'tdef' function

So this is the sigma matrix returned by the cov_mmcsd function using the $AR1$ model

```
write_matex(fitTheta_ar1$sigmaTheta)
```

$$\begin{bmatrix} 1.70340275041982 & 1.03860931019733 & 0.627164720478811 & 0.372519231871133 & 0.214917633773547 \\ 1.03860931019733 & 1.70340275041982 & 1.03860931019733 & 0.627164720478811 & 0.372519231871133 \\ 0.627164720478811 & 1.03860931019733 & 1.70340275041982 & 1.03860931019733 & 0.627164720478811 \\ 0.372519231871133 & 0.627164720478811 & 1.03860931019733 & 1.70340275041982 & 1.03860931019733 \\ 0.214917633773547 & 0.372519231871133 & 0.627164720478811 & 1.03860931019733 & 1.70340275041982 \end{bmatrix}$$