TECHNICAL REPORT NO. 2

HANDLING PLANNED AND UNPLANNED MISSING DATA

THE PROJECT IS FINANCED BY THE CANADIAN INSTITUTES OF HEALTH RESEARCH (CIHR)

Executive Summary

To decrease data collection costs and participants' fatigue, researchers often reduce the length of a survey with planned missingness (PM). PM is a useful technique to reduce a survey's length without compromising its validity. The present report introduces two state-of-the-art methods to handle planned (and unplanned) missing data in the COVID-19 Canada project - estimation by full information maximum likelihood (FIML) and multiple imputation (MI). With FIML, parameters are estimated by taking into account the patterns of missing values. With MI, a number of complete datasets (usually at least 50) are produced by estimating missing values, conditional on all the variables included in the imputation model. Statistical analyses are then performed on these complete datasets and pooled to obtain single parameters estimates and adequate standard errors. Researchers can select a set of auxiliary variables that are strongly correlated with important variables or explain why some values are missing to improve the prediction in both FIML and MI. When analyzing data from the project, it is recommended to use FIML or MI to handle missingness. This report contains some applied examples for both techniques utilizing the research project's dataset and provide suggestions for further reading.

Definitions of key concepts

Concepts	Definitions
Planned Missingness	Measurement design technique in which some items are only presented to subsets of participants, to reduce the length and cost of the data collection process.
Full Information Maximum Likelihood (FIML)	Method for estimating parameters with the likelihood function by using the raw data instead of a sample or population covariance matrix.
Multiple Imputation (MI)	Process by which missing values are replaced with multiple estimated values conditional upon the observed data and incorporating random noise.

Technical Concerns

To reduce the survey's length and cost while including as many items as possible, the *COVID-19 Canada* project used a survey design in which subsets of items were presented to each participant. This report presents:

- Techniques to mitigate the effect of missing values.
- Ways to account for planned and unplanned missing data.

Recommendations

Use Full Information Maximum Likelihood (FIML) or Multiple Imputation (MI) when dealing with the COVID-19 research project's data.
Keep the number of variables limited for the imputation (around 30, never over 100):

a) Select variables that are moderately to strongly correlated (but not necessarily causally related) to the variables of interest.

b) Select variables that can predict why some data is unobserved (e.g., a variable indicating the planned missingness questionnaire version).

Handling planned and unplanned missing data

Every study can be affected by missing data. Missing data can result from unplanned missingness for example when participants do not respond to some items due to inattention, refusal to provide responses to certain questions, decided to quit a longitudinal research project or due to technical malfunctions in the survey (e.g., an item was not presented or a response was not recorded). Missing data especially plague longitudinal studies and long surveys, where there can be attrition, fatigue, or both. Missing data can invalidate results from statistical analyses (Enders, 2010). However, in some cases, missing data might be planned. Indeed, researchers may decide that some questions in a survey will not be presented to each participant as a mean to reduce its length. This reduces costs and strain on participants; statistical techniques are then used to mitigate the impact of missing values on parameter estimates to make sure the data collection process yields valid information. Voluntary introduction of missing values in the data collection process is referred to as "Planned missingness" (Rhemtulla & Little, 2012). The present report aims to explain the planned missingness technique as well as to provide some practical advice on effectively using data collected with PM. Specifically, this report includes some real examples from the COVID-19 Canada research project's data with FIML and MI in popular R packages. Thus, we aim to equip researchers and students to understand planned and unplanned missingness, and how to deal with missing values.

Planned Missingness

Planned missingness is the voluntary introduction of missing values by researchers (Enders, 2010; Little & Rhemtulla, 2013; Rhemtulla & Little, 2012). In such research designs, some measurements/assessments are not performed for randomly selected subsamples. For instance, each participant may be asked to provide answers to two thirds of the items from a scale. In this type of design, scales that are not of central importance for the research question are separated into sets. For example, in a three-form design, items are divided into a core set presented to all participants (X), and three sets (A, B, and C) that are only presented to two-thirds of the sample. As a rule, demographics and items central to the research question or that can explain why some values are missing (i.e., are related to missingness or dropout such as reading speed or conscientiousness, Graham et al., 2006) are included in X. Other items are assigned to sets A, B, or C.

When data collection is underway, participants are randomly assigned to one of the three versions of the questionnaire. Each version of the questionnaire contains block X and the items from two blocks - AB, AC, or BC (Little & Rhemtulla, 2013; Rhemtulla & Little, 2012). In a longitudinal study, in each wave participants are assigned to a version of the questionnaire at random and independently of their assignment in the previous waves of data collection. According to Rhemtulla and Little (2012), planned missingness should be used with samples of at least 375 participants.

How to Analyze Data With Planned Missingness?

Planned missingness can be readily used with FIML and MI, even if unplanned missingness is present in the dataset (Enders, 2010; Rhemtulla & Little, 2012). When dealing with missing values, best practices recommend going through three phases: 1) a preparation phase for examining missingness, 2) a decision phase during which choices are made about how to perform the analysis and which variables to include, and 3) an analysis phase in which the analysis to answer the research question is performed. Below, we outline and explain each phase.

Preparation Phase

When preparing the dataset for analysis, one should examine and report unplanned missingness (Enders, 2010). Rhemtulla and Little (2012) suggest to first perform a single imputation on data that is missing due to planned missingness. Then, researchers can analyze the data to characterize the unplanned missingness, such as investigating the patterns of missingness and performing Little's MCAR test (Enders, 2010). These analyses can suggest whether the missing data mechanism is Missing Completely at Random (i.e., there is no underlying factor that explains why some data is unobserved) or if missingness can be predicted with the data (i.e., Missing at Random) or not (i.e., Missing Not at Random). Note, however, that FIML and MI are currently considered as best practice methods for dealing with missing data, no matter the underlying missingness mechanism (Enders, 2010). As recommended, the proportion of missingness and its location should be reported.

Using Full Information Maximum Likelihood and Multiple Imputation

When using FIML or MI, it is best to not only include variables related to the research question. Specifically, to facilitate the analysis, it is strongly suggested to keep the variables of interest as well as auxiliary (or instrumental) variables, that are not used for analysis but are strongly correlated and can predict the missing values in of the variables of interest (Enders, 2010). Furthermore, variables explaining why some values are not observed (i.e., the variable indicating in which planned missingness version of questionnaire each participant was assigned) must be included in the missing data procedure to provide unbiased estimates (Graham, 2009). Auxiliary variables will help refining both FIML and MI estimates, as they allow for better prediction. It is important to note that these techniques are not causal but predictive and postdictive, variables can be imputed using data from subsequent waves of data collection (Honaker et al., 2011). For instance, in a longitudinal design, a variable measured at time 3 can be used to estimate missing values at time 1 and 2.

It is best to restrict the number of variables in the imputation model for several reasons. With too many variables, there are higher chances that the imputation fails because of

multicollinearity between predictors, but also due to other computational problems[1] (Graham, 2009; van Buuren, 2018). Some authors suggest including a maximum 100 variables (Graham, 2009). Van Buuren (2018) makes a more conservative suggestion to use only 15 to 25 variables in total. In short, it is preferable to only include a subset of variables in the procedure rather than the entirety of the dataset.

Full Information Maximum Likelihood

FIML can be used directly in the statistical analysis model. It is extremely convenient, as it requires only the variables necessary for the analytical model and an addition of a single line of code or a check box. FIML can be used with Amos, R, SAS, etc. When FIML is used, parameters are estimated by finding the set of values which maximizes the likelihood of having produced the observed data (Allison, 2012; Enders, 2010). This can be done with or without the use of auxiliary variables. To include auxiliary variables in Structural Equation Models (SEM), covariances are added between the auxiliary variables (entered as manifest variables) and the error terms of other manifest variables (Graham, 2003). In multilevel models, auxiliary variables can be added to the analysis by coding their values as if it was an additional measurement of the dependent variable (Allison, 2012).

FIML handles missingness on the outcome variables well (Graham, 2003; van Buuren, 2018), but it is not without limitations. Multivariate normality needs to be assumed to use FIML. FIML can handle some deviation from normality, but if the normality assumption is too heavily violated, it is preferable not to use this method. In some cases, it could also be advised to use a robust FIML estimator, which is available in some statistical packages (e.g., R, Mplus). Moreover, with too many missing values in the independent variables, FIML might reduce the statistical power, as cases with missingness in the predictors are discarded (note that this is not always the case when performing SEM with latent variables). To avoid this, researchers can include auxiliary variables when fitting their models. Furthermore, FIML can sometimes be problematic in some cases as some goodness of fit indices become unavailable when there is missing data. For example, in SPSS Amos for SEM, SRMR, modification indices, standard errors and bootstrapping become unavailable, even if other parameters (i.e., relation between variables) are estimated with FIML. Also, some statistical packages have not yet implemented FIML for some analyses when there is missing data on the predictors (this was the case for SAS for generalized linear models when data is missing on the predictors). A final caveat is that, while the inclusion of auxiliary variables can help to provide accurate estimates, it can also worsen a model's fit and complexify its convergence and identification. For those situations, researchers may decide to use multiple imputation instead of FIML.

Multiple Imputation

MI is a process in which missing values are replaced with multiple new values predicted with other available data to which some random variation is added (Enders, 2010). This is the

[1] As an example, while preparing this report, the author had many computational difficulties trying to perform MI on a portion of the project's dataset (three variables over nine waves of data collection). Too many variables were included in the imputation model because the effect of time was allowed to vary between participants. After letting the process run for two days straight, R returned an error because it could not allocate sufficient memory (in this case, a consecutive memory array of 2.5 GB). Because of how the imputation software was designed, it was impossible to retrieve the results. This anecdote illustrates the current computational limitations on handling missing data. We expect that, with a smaller dataset (i.e., less participants), we could have allowed time to vary between participants.

specificity of MI; multiple slightly different complete copies of the dataset are produced to account for the uncertainty inherent to data (Enders, 2010; van Buuren, 2018). For many MI software, the default number of imputed datasets is set to five, but there are no statistical advantages to have a small number of datasets. The recommended number of imputed datasets is at least 50 (Enders, 2010). The proportion of missing values should not be used to determine the number of imputed datasets (Madley-Dowd et al., 2019).

The MI procedure can give unbiased estimates for large proportions of missing data. Some simulation studies even found that it can perform well with 90% of missing values (Madley-Dowd et al., 2019). With SPSS, the MI procedure is based on the Expectation-Maximization algorithm (Enders, 2010). For R, commonly used packages to impute data are Amelia (Honaker et al., 2011) and mice (van Buuren & Groothuis-Oudshoorn, 2011). The imputed datasets can be used with other statistical procedures, even those using FIML, such as structural equation modelling with Amos or the lavaan package in R (Rosseel, 2012).

In comparison to FIML, MI needs to be performed separately from the analyses, and might cause other issues, especially on the computational aspect. Specifying the MI model (i.e., selecting variables to include, specifying predictors if needed) can be relatively tedious and performing the calculations can be quite heavy computationally and time consuming. Similarly, the analysis performed after the MI procedure can be lengthy, as it is repeated for each imputed dataset and the parameter estimates are then pooled. Furthermore, pooling the estimates can be complex or not available for some analyses and some statistical software. For instance, in SPSS, factorial ANOVA and ANCOVA procedures do not support the automization process to generate pooled results.

Concluding Remarks on FIML and MI

In sum, FIML works by estimating the parameters that fit well to the observed data, while MI creates multiple estimations for each missing value by using the observed data as predictors. Both of these methods are at the state of the art when dealing with missing data and should thus be used whether there is planned or unplanned missingness. With both methods, auxiliary variables should be chosen with care in order to obtain the best estimates. Furthermore, there are other choices that need to be made, for instance to decide whether missing values should be imputed at the item or at the scale level. FIML and MI have each their own advantages and drawbacks; thus, the researcher should choose the method depending on the research question, the planned analyses and the available data.

Method

Data was extracted from a large Canadian survey (N = 3617) which asked a representative sample of adults to complete an in-depth questionnaire on multiple occasions over several months. At the time this report was compiled, 10 waves of data collection were available. For more methodological details, please consult our technical report (de la Sablonnière et al., 2020). A subset of items was selected for the purpose of this report. As an example, we only kept items on sleep, emotional state, and several sociodemographic variables. Those variables were selected for two main reasons. First, variables on sleep, loneliness, nervousness and angriness were related to our research question about the relation between emotions and sleep quality in the COVID-19 pandemic context. Second, the other emotional state items and demographic were expected to be related to our variables of interest and would thus act as good auxiliary variables. Complete examples of handling missingness with FIML and MI procedures are described in the following sections. All the code is available in the GitHub repository for this project at https://github.com/mcarondiotte/handling-missing-data-covid19.

Exploration of the Missing Data and Their Patterns

Before analyzing the data, we explore and characterize the missingness in our sample. The exploration of the missingness inherent to the project's data was done in two steps. First, we sought to quantify the missingness and its patterns. Then, we determined if some variables could predict why some participants did not answer the surveys (i.e., if there was a mechanism behind the missing data). Data can be loaded in R using the following code:

```
data <- read.csv("data/sleep.csv") # load data
head(data)

summary(data[, c("p1xq1a_1", "p1xq3a_1")])
```

This code snippet reads a CSV file using the **read.csv()** function, which takes the (direct or relative) path to the data file as input[2]. Then, the data matrix is stored in a **data.frame** object named data. Note that other functions can be used to read data files or to import data from other file types such as *SPSS, SAS, STATA*, or *Mplus* (see *rio's* **import()** or the foreign package). Then, we make sure the data are loaded correctly, we call the **head(data)** (which takes the data frame object as input) to print the first few lines of data. The data contained in the data frame object can be accessed by many ways. To access a particular column (or variable) of the data frame (for instance, the *age* variable), we can use **data$p1xq1a_1**. The **data.frame** objects support matrix notation (row/column); we can also access columns with **data[, "p1xq1a_1"]**. Multiple columns can be accessed by passing a vector (**c()**) of column names, **data[,c("p1xq1a_1", "p1xq3a_1")]**, or even their index (i.e., the column number), **data[,c(24,27)]**. We can also subset rows in the data frame by indicating their number in the first part of the matrix notation. For example, **data[1,]** selects the first row, **data[c(2,42),]** selects second and 42nd rows for instance, and data**[2:42,]** subsets every row from 2 to 42 (the same applies for columns). In the code example, we finally print summary statistics (mean, median, minimum, maximum, 1st and 3rd quartiles and number of missing values) for *age and number of people in the household* with R's **summary()** function. For further guidance on how to use R, we refer the reader to resources such as *An Introduction to R* (Venables et al., 2021) and other learning materials that are easily accessible online.

Patterns of Missing Data

We begin by quantifying the number of participants for whom we have no valid data at any data collection wave (i.e., participants who did not respond to a questionnaire, were excluded because they were too quick to answer the whole survey or failed the attention checks). Table 1 shows the number of missing values and Table 2 shows the percentage of participants for

[2] In this example, relative path is used, as we indicate a path from the current working directory. When working with R projects in R studio, the working directory is the project folder. From there, we go into the data folder, which contains the sleep.csv data file.

which we have no valid data. Figure 1 shows the patterns of missingness for the participation in each wave of data collection; there are 437 missingness patterns on wave participation. In total, 1288 (35.6%) participants missed at least 50% of the waves. There was 41.7% of missing values due to missed assessments or participant removal (i.e., proportion of complete surveys missed).

Table 1

Number (and Percentage) of Missing Participants for Each Assessment

	1	2	3	4	5	6	7	8	9	10
Frequency (%)	0 (0.00)	1402 (38.77)	1328 (36.73)	1399 (38.69)	1551 (42.89)	1591 (44.00)	1640 (45.35)	1799 (49.75)	1854 (51.27)	1814 (50.17)

Table 2

Number (and Percentage) of Participants in Function of the Number of Assessments Missed

	Number of waves missed									
	1	2	3	4	5	6	7	8	9	10
Frequency (%)	615 (17.00)	555 (15.35)	368 (10.18)	277 (7.66)	272 (7.52)	241 (6.66)	276 (7.63)	259 (7.16)	283 (7.83)	470 (13.00)

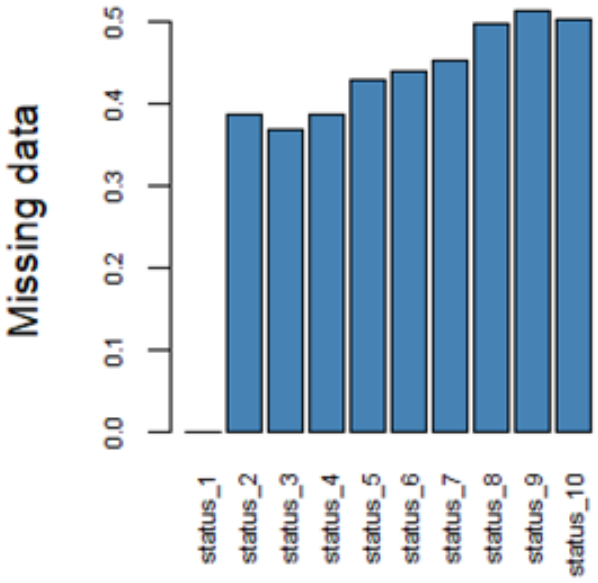

Figure 1. Patterns of missing data for the first 10 waves of data collection.

Having explored the missingness pertaining to the participation in each wave of data collection, we turn to the inspection of the missingness over our variables of interest and our auxiliary variables. As recommended by Rhemtulla and Little (2012), we could have performed a single imputation on data that was planned as missing to quantify the unplanned missingness. However, in this project, participants who replied to a questionnaire cannot have missing data due to the forced response method; unplanned missingness on items is thus out of the equation. In other words, participants who replied to a questionnaire cannot have missing data, due to the forced response method.

Predicting Unplanned Missingness

Because participants who did not respond to the survey at some time point could still respond to subsequent assessments, we can use Poisson regressions to examine whether sociodemographic variables predict the number of missed surveys[3]. The logic here is to find if participants from certain backgrounds are more likely to dropout from the survey. For instance, to assess if income is related to survey completion, we first compute one global income variable (**b21xq2**) from all the attempts to ask about the income (**b21xq2_7**, **b21xq2_8**, **b21xq2_9**, and **b21xq2_10**). Then, we compute a variable indicating if participants had responded to any of the waves that included the income question (**ans7_10**) and another indicating if their answer to this question was recorded (**ans_income**). We use the following code:

```
data$b21xq2 <- ifelse(is.na(data$b21xq2_7), data$b21xq2_8, data$b21xq2_7)
data$b21xq2 <- ifelse(is.na(data$b21xq2), data$b21xq2_9, data$b21xq2)
data$b21xq2 <- ifelse(is.na(data$b21xq2), data$b21xq2_10, data$b21xq2)

data$ans7_10 <- rowSums(mm[,7:10], na.rm = TRUE) > 0
data$ans_income <- ifelse(is.na(data$b21xq2), 0, 1)

addmargins(xtabs(~ ans7_10 + ans_income, data))
```

[3] We could also have used Little's MCAR test to determine if some missingness could be predicted by variables in our dataset. However, to our knowledge, the implementation of this test in the *BaylorEdPsych* package is limited to 50 variables (but a good implementation is found in *SPSS*).

We find that 2570 (71.1%) participants had responded at least to one survey between wave 7 and wave 10, and that, among those participants, only 70 (2.7%) had not reported their income. Then, we fit a Poisson regression using the **glm()** function if to see if the income bracket reported by the participants is associated to the number of surveys completed.

```
fit.income <- glm(n_miss ~ b2_1xq2, data = data, family = ''poisson'')
summary(fit.income)
```

Among individuals who did respond to the income item, we found no association between income and the number of surveys completed, B = 0.00, p = .945. To explore further the association between income and missingness, we regressed the binary indicator of response to income on the number of waves completed. We find that the number of waves completed varies with the response to the income item. Among participants who responded between waves 7 and 10, those who reported their income had fewer uncompleted surveys, B = -0.71, p < .001. As the choice to disclose one's income is related to the disclosure of one's income in surveys, this response indicator could be included when handling missing data (Enders, 2010; Graham, 2009).

We performed Poisson regressions to predict the number of missing waves with other demographic variables. Older participants had fewer missing waves, B = -0.01, p < .001, and women had more missing waves as compared to men, B = -0.01, p < .001. Participants living with more people, B = 0.06, p < .001, with more children under the age of 18, B = 0.08, p < .001, and with children under the age of six, B = 0.14, p < .001, were more likely to miss survey waves. Furthermore, education seemed to play a role, the less educated, the more missing waves. The same can be said of work status, type of employment, company size, native language, political affiliation and religion, as some categories were associated with lower response to survey than others. Political orientation (left/right) is not associated with missingness, B = 0.00, p = .061. In sum, almost all of the sociodemographic variables seem to be associated with the number of uncompleted assessments, which is expected for large datasets. Indeed, with large datasets, trivial effects (such as B = 0.01) are easily identified as "significant". As such, it could be best to conclude that, if there is an effect of those variables, it is very small for most of them. A similar analysis could be done by predicting patterns of response to waves (instead of the number) with the same variables, for instance with a sequence analysis.

Another exploratory analysis was used to determine if sociodemographic variables are related to the type of missingness. It is useful to investigate whether some variables are associated with missing data, or with exclusion due to failing the attention checks or completing the survey too quickly. To this end, we regress age, gender, and income on binary indicators of sources of missingness (e.g., not completed the survey, exclusion due to failing the attention check, exclusion due to completing the survey too quickly) using mixed-effects logistic regressions (wave number was specified as a level-1 variable; random intercepts were specified for each participant). Regarding the lack of response to surveys, older participants are more likely to complete a questionnaire, B = -0.01, p < .001; women are more likely than men to complete surveys, B = -0.22, p < .001. No effect of age, gender or income was detected for finishing the survey in under four minutes or failing attention checks.

Concrete Examples in R

We now turn to how to use FIML and MI in practice. Statistical packages include procedures to deal easily with missing data. However, for some applications of FIML, some data manipulations are needed. In what follows, we present and explain code snippets to analyze the project's data by handling missing data with FIML and MI.

Full Information Maximum Likelihood

FIML estimates the parameters (i.e., the results of the analysis) by using all the observed data. As such, this procedure requires to be set up each time an analysis is performed. To utilize the full potential of FIML, auxiliary parameters should be properly included in the model (Allison, 2012; Graham, 2003). Depending upon the analysis used, there are multiple ways to specify a model. Some methods are detailed in what follows, but readers can find more FIML specification in the *Recommended Readings* section (Allison, 2012; Graham, 2003). We first demonstrate the use of FIML in R with the ***lavaan*** package (Rosseel, 2012) for two types of models: regression and growth curve modelling. Then, we show how to apply FIML with multilevel regressions (also called mixed-effects models) using the ***lme4*** package (Bates et al., 2015).

Regression

Once the package and data are loaded, we specify the model as a string variable using the ***lavaan*** syntax. In this example, we regress *loneliness* (**lonely_1**), *nervousness* (**nervous_1**) and *angriness* (**angry_1**) scores (i.e., three continuous predictors) on the *valence of the last dream* (**dreamval_1**). In this first model, we do not use FIML nor auxiliary variables:

```
library(lavaan) # first we load the lavaan package
model <- 'mintosleep_1 ~ lonely_1 + nervous_1 + angry_1' # specify the
regression model
fit <- sem(model, data = data, fixed.x = FALSE) # fit model
summary(fit, fit.measures = TRUE, rsquare = TRUE, standardize = TRUE) #
print model summary
```

In the above code, we first specify the regression model. Then, we use the **sem()** function to fit the model to the data. Specifying **fixed.x = FALSE** allows to easily (without typing it all in the model string) estimate the means, variances, and the covariances on all exogenous (independent) variables instead of fixing them on the sample's values, which is required to perform FIML. The call to the **summary()** function prints the analysis results (along with the regression coefficients, variances, covariances, and meanstructures). Specifying **fit.measures**

= TRUE retrieves the model's fit, rsquare = TRUE shows the R^2, and standardize = TRUE shows the standardized regression coefficients). To handle missing data with FIML, we run the sem() function, simply adding missing = "FIML" in the function's parameters.

```
model <- '
  # regression
  dreamval_1 ~ lonely_1 + nervous_1 + angry_1
'
fit <- sem(model,
           data = data,
           missing = ''FIML'',
           fixed.x = FALSE)
summary(fit, fit.measures = TRUE, rsquare = TRUE, standardized = TRUE)
```

To improve the parameter estimation, we can use auxiliary variables. Adding auxiliary variables to a regression model can be done by using the saturated correlates approach by including covariances between it and all the other variables in the model (Allison, 2012; Graham, 2003). In this example, we add the *number of minutes taken to fall asleep* (mintosleep_1) as an auxiliary variable. Thus, we change the model string:

```
model.aux <- '
  # regressions
  dreamval_1 ~ lonely_1 + nervous_1 + angry_1
  #covariances
  lonely_1 ~~ mintosleep_1
  nervous_1 ~~ mintosleep_1
  angry_1 ~~ mintosleep_1
  dreamval_1 ~~ mintosleep_1
'
fit <- sem(model.aux,
           data = data,
           missing = ''FIML'',
           fixed.x = FALSE)
summary(fit, fit.measures = TRUE, rsquare = TRUE, standardized = TRUE)
```

The same can be done with two auxiliary variables (or more). We add another set of covariances to include the other auxiliary variables, here the *evaluation of one's sleep quality* (sleepqual_1):

```
model.aux <- '
  # regressions
  dreamval_1 ~ lonely_1 + nervous_1 + angry_1
  # covariances
  lonely_1 ~~ mintosleep_1
  nervous_1 ~~ mintosleep_1
  angry_1 ~~ mintosleep_1
  lonely_1 ~~ sleepqual_1
  nervous_1 ~~ sleepqual_1
  angry_1 ~~ sleepqual_1
  dreamval_1 ~~ mintosleep_1
  dreamval_1 ~~ sleepqual_1
  mintosleep_1 ~~ sleepqual_1
'
fit <- sem(model.aux,
           data = data,
           missing = ''FIML'',
           fixed.x = FALSE)
summary(fit, fit.measures = TRUE, rsquare = TRUE, standardized = TRUE)
```

Growth Curve Modelling

With growth curve modelling, the goal is to examine how an outcome varies over time, and how variables influence this variation. As the project's data was collected from the same participants over time, growth curve modelling is well suited to analyze the data. For growth curves, a model with two latent variables, random intercept (i) and random slope (s) is specified. Then, we can include predictors (i.e., in our example, age and gender) for those two latent variables, as well as time varying predictors (regressed on each measurement). As with regression analysis with *lavaan*, missing values can be estimated using FIML. Means, variances, and covariances of the independent variables need to be estimated. Auxiliary variables can be specified as in a regression model, by treating them as dependent variables. The following model string describes such a model:

```
model.growth <- '
  # define intercept (i) and slope (s)
    i =~ 1*dreamval_1 + 1*dreamval_2 + 1*dreamval_3 + 1*dreamval_3 +
1*dreamval_5 + 1*dreamval_6 + 1*dreamval_7 + 1*dreamval_8 + 1*dreamval_9
    s =~ 0*dreamval_1 + 1*dreamval_2 + 2*dreamval_3 + 3*dreamval_3 +
4*dreamval_5 + 5*dreamval_6 + 6*dreamval_7 + 7*dreamval_8 + 8*dreamval_9
  # regressions
    i ~ gender + age
    s ~ gender + age
  # time-varying covariates
    dreamval_1 ~ lonely_1 + nervous_1 + angry_1
```

```
    dreamval_2 ~ lonely_2 + nervous_2 + angry_2
    dreamval_3 ~ lonely_3 + nervous_3 + angry_3
    dreamval_4 ~ lonely_4 + nervous_4 + angry_4
    dreamval_5 ~ lonely_5 + nervous_5 + angry_5
    dreamval_6 ~ lonely_6 + nervous_6 + angry_6
    dreamval_7 ~ lonely_7 + nervous_7 + angry_7
    dreamval_8 ~ lonely_8 + nervous_8 + angry_8
    dreamval_9 ~ lonely_9 + nervous_9 + angry_9
  # covariances
    dreamval_1 ~~ sleepqual_1
    dreamval_1 ~~ sleepqual_2
    dreamval_1 ~~ sleepqual_3
    dreamval_1 ~~ sleepqual_4
    dreamval_1 ~~ sleepqual_5
    dreamval_1 ~~ sleepqual_6
    dreamval_1 ~~ sleepqual_7
    dreamval_1 ~~ sleepqual_8
    dreamval_1 ~~ sleepqual_9
    dreamval_2 ~~ sleepqual_2
    dreamval_2 ~~ sleepqual_3
    dreamval_2 ~~ sleepqual_4
    dreamval_2 ~~ sleepqual_5
    dreamval_2 ~~ sleepqual_6
    dreamval_2 ~~ sleepqual_7
    dreamval_2 ~~ sleepqual_8
    dreamval_2 ~~ sleepqual_9
    dreamval_3 ~~ sleepqual_3
    dreamval_3 ~~ sleepqual_4
    dreamval_3 ~~ sleepqual_5
    dreamval_3 ~~ sleepqual_6
    dreamval_3 ~~ sleepqual_7
    dreamval_3 ~~ sleepqual_8
    dreamval_3 ~~ sleepqual_9
    dreamval_4 ~~ sleepqual_4
    dreamval_4 ~~ sleepqual_5
    dreamval_4 ~~ sleepqual_6
    dreamval_4 ~~ sleepqual_7
    dreamval_4 ~~ sleepqual_8
    dreamval_4 ~~ sleepqual_9
    dreamval_5 ~~ sleepqual_5
    dreamval_5 ~~ sleepqual_6
    dreamval_5 ~~ sleepqual_7
    dreamval_5 ~~ sleepqual_8
    dreamval_5 ~~ sleepqual_9
    dreamval_6 ~~ sleepqual_6
    dreamval_6 ~~ sleepqual_7
```

```
    dreamval_6 ~~ sleepqual_8
    dreamval_6 ~~ sleepqual_9
    dreamval_7 ~~ sleepqual_7
    dreamval_7 ~~ sleepqual_8
    dreamval_7 ~~ sleepqual_9
    dreamval_8 ~~ sleepqual_8
    dreamval_8 ~~ sleepqual_9
    dreamval_9 ~~ sleepqual_9
'
```

Then, we can fit this model to the data with the **growth()** function using the same parameters as with regression models (this can be quite time consuming).

```
fit.growth <- growth(model.growth,
                     data = data,
                     missing = ''FIML'',
                     fixed.x = FALSE)
summary(fit.growth,
        fit.measures = TRUE,
        rsquare = TRUE,
        standardized = TRUE)
```

Multilevel Regression (or Mixed-Effects Models)

Similar to growth curve modelling, multilevel regression allows to model how an outcome varies over time and the variables which associate with this change. Hierarchical regression is particularly robust to missing data in the dependent variable. However, some adjustments need to be done when missingness is present in the independent variables and when we want to include an auxiliary variable. Specifically, we need to treat the auxiliary variable as another measurement of the dependent variable, and to create an index (*D*), which distinguishes the outcome from the auxiliary (Allison, 2012).

In R, we can use the ***lme4*** and *nlme* packages to fit multilevel regressions. For these packages, the data frame must be in the long format, such that each row represents one measurement occasion (i.e., one wave) for one participant. The following code transforms the project's data from the wide to the long format (other packages such as *tidyr* and *reshape2* can also be used for this task):

```
# Reshape the dataframe from wide to long
data_long <- reshape(data,
                     idvar = ''id'',
                     varying = 5:157,
                     direction = ''long'',
                     timevar = ''wave'',
                     sep = ''_'')
row.names(data_long) <- 1:nrow(data_long) # fix the row numbers
```

The **reshape()** function transform data from wide to long and from long to wide (**idvar** = "id" refers to the column containing the participants' unique identifier, **varying** = **5:157** specify which columns are to be transposed, **direction** = "**long**" indicate that we want a long data frame, **timevar** = "**wave**" indicates that the time of measure variable will be named "wave", and **sep** = "_" tells the function that variables names and their measurement occasion are separated by an underscore). Then, we fix the rows numbers.

We can now begin to modify our data frame to include the auxiliary variable. To do so, we first split the data frame into two data frames containing all the same variables, with one exception: the outcome variable (here the valence of the dream, **dreamval**) is included only in one data frame, and the auxiliary variables are included only in the other data frame. We create two data frames to make it easier to work with our real outcome variable and our auxiliary variable. We then create an indicator **D** in each part to distinguish the outcome from the auxiliary variable (0 being the outcome, 1 the auxiliary). In our example, we use the sleep quality item (**sleepqual**) as an auxiliary variable. Finally, the two data frames are merged using the **rbind()** function. This is done in the following code:

```
aux <- data_long[,c(''id'',''wave'',''gender'',''sleepqual'')] # create
first data frame (with the auxiliary variable)
aux$D <- 1 # create the indicator specifying that this variable is auxi-
liary
names(aux)[4] <- 'dreamval' # rename the auxiliary variable so that they
are treated as the outcome variable
data_long.aux <- data_long[,c(''id'',''wave'',''gender'',''dreamval'')] #
create the second data frame (with the dependent variable)
data_long.aux$D <- 0 # create the indicator specifying that this variable
is not auxiliary
data_long.aux <- rbind(data_long.aux, aux) # merge the two data frames
```

Finally, we can specify and fit the model to our data using the *lme4* package. However, this package does not present *p*-values (by choice, see Bates, 2006). To get *p*-values, we load the *lmerTest* package. Then, we fit our model by specifying its formula. For the analysis to be valid and use the auxiliary variable, we need to include the *D* variable in the model, as well as its interaction with each other variable (this is done with a colon in the formula syntax, or an asterisk for *lmer*). Finally, we fit the model using an unstructured covariance matrix (Allison, 2012), which is why we chose to use *lme4*. This model is fit using the **lmer()** function.

```
library(lme4)
library(lmerTest)
fit_lme.aux <- lmer(dreamval ~ D + gender + wave + gender:wave + D:gender
+ D:wave + D:gender:wave + (1 | id), data = data_long.aux)
summary(fit_lme.aux)
```

To interpret the results, examine the main effects (i.e., not the interaction terms).

Multiple Imputation

MI produces multiple estimates for each missing value by using the observed data as predictors. Contrary to FIML, MI does not need to be set up each time an analysis is performed. In fact, if all variables needed for the analyses (as well as auxiliary variables) are included in the imputation model, MI could be done once for a given set of analyses. Note that MI can be done on single items or on a composite score (i.e., mean on a scale).

It can be useful to perform some operations (i.e., computing scores, centering values) on the data frame before running the imputation. For instance, when an interaction effect is tested, the interaction term has to be computed beforehand (Enders, 2010). In the following code snippet, we centre and compute an interaction between T1 *nervousness* and *anger*. We store the new centered variables in two new columns (**nervous_1.c** and **angry_1.c**) and the computed interaction term in a column named **inter**.

```
data$nervous_1.c <- scale(data$nervous_1.c, centre = TRUE, scale = FALSE)
data$angry_1.c <- scale(data$angry_1.c, centre = TRUE, scale = FALSE)
data$inter <- data$nervous_1.c * data$angry_1.c
```

The **scale()** function is used to perform the centering. Crucially, the **centre = TRUE** indicate that we want the result to be mean-centered, while the **scale = FALSE** means that we do not want the result to be divided by the standard deviation (i.e., we do not want to compute the z-score).

Using the "mice" Package

The *mice* package (van Buuren & Groothuis-Oudshoorn, 2011) is useful for dealing with missing data, because of its many features. For instance, *mice* allows to include a predictor in the imputation model without imputing its values (or even remove it entirely from the imputation model). To do so, we perform an empty imputation using the **mice()** function. By specifying only one imputed datasets (m) and no maximum number of iterations (**maxit**), the function returns an almost empty *mice* imputated data frame object, which we stores in an object called **ini**. The "**predictorMatrix**" part of this object represents the predictor matrix, which defines which variables are imputed and which variables are used as predictors. Rows indicate the predicted variables while the columns indicate the predictors. For any row, a 1 in a column indicate that it is predicted by the variable which name is in the column; while a 0 indicate that it is not predicted by it. This is an example of a predictor matrix:

	age	gender	angry
age	0	0	0
gender	0	0	0
angry	1	1	0

In this example, *age* and *gender* are not predicted by any variable, but *angry* is predicted by *age* and *gender* (but not itself). By default, all variables are used as predictors and are predicted if they have missing values. For instance, if we want to use only age as a predictor (i.e., we are not interested in imputing its values), we need to set its entire row to 0. Similarly, if we want to set a specific predictor (e.g., *age*) for a specific variable (e.g., *angry_1*), we would change the value of the intersecting cell to 1. This is done in the following code:

```
# Load packages
library(mice)
library(miceadds)
# Set up the predictor matrix
ini <- mice(data, m = 1, maxit = 0)
ini$predictorMatrix[,"age"] <- 0 # age is not predicted
ini$predictorMatrix["angry_1","age"] <- 1 # age predicts angry_1
```

These operations give us a custom predictor matrix in the object **ini$predictorMatrix**. For more information on multilevel imputation, see van Buuren (2018) and the *micemd* package documentation (Audigier & Resche-Rigon, 2019).

Then, we can launch the imputation process using again the **mice()** function. We set the number of imputed datasets (**m**) to 50, the maximum number of iterations (**maxit**) to 35, and give the random number generator (**seed**) a seed to ensure reproducibility. We also specify a predictor matrix by passing our custom predictor matrix (**ini$predictorMatrix**) to the **predictorMatrix** parameter. The multiply imputed datasets are stored in a new object called **imp**.

```
imp <- mice(data,
            m = 50,
            maxit = 35,
            predictorMatrix = ini$predictorMatrix,
            seed = 23109)
# Assess convergence
print(imp)
plot(imp)
```

When the multiple imputation is finished, we need to assess convergence, as *mice* uses an interated procedure to estimates the parameters of the chained equations. This is done by inspecting (with the **print()** function) and by verifying that all chains are converging towards the same (credible) values using **plot()**.

To perform operations on the variables after the imputation phase, we need to transform the *mice* imputed object to a data frame, perform the operations and then convert the data frame back to a *mice* imputed object. The following code does so, creating a new object **imp2** to not overwrite the original **imp** object:

```
imp2 <- complete(imp, action = 'long', include = TRUE)
imp2$neg_affect <- rowMeans(imp2[,c('lonely_1','nervous_1','angry_1')],
na.rm = TRUE)
imp2 <- as.mids(imp2)
```

We use the **with()** function to perform our analysis (e.g., a regression model) on each imputed data frame and store the results in an object named fit. Finally, we pool the results using the **pool()** function and display the pooled results using **summary()**.

```
fit <- with(imp2, lm(dreamval_1 ~ lonely_1 + nervous_1 + angry_1))
summary(pool(fit))
```

Pooling procedures for many analyses have been implemented in the *mice* package (van Buuren & Groothuis-Oudshoorn, 2011). The package *miceadds* (Robitzsch, & Grund, 2021) also add numerous other functions to help in setting and analyzing multiply imputed data.

Using the "Amelia" Package

The package *Amelia* (Honaker et al., 2011) is another option to deal with missing values. The algorithm used by the *Amelia* package has been extensively tested and is fast (Enders, 2010). However, *Amelia* does not allow to control predictors and prediction, but can be easily used to perform multiple imputation on longitudinal and multilevel data.

We can launch the imputation process using the **amelia()** function. We set the number of imputed datasets (**m**) to 50, and specify ordinal (**ords**) and nominal (**noms**) variables, which names are passed as character vectors. Of importance for the imputation of longitudinal data, the "**ts**" parameter points to the variable identifying the time of measurement for each observation (here, we specify that, in the long version of the data frame, our column containing the time indicator is named "wave"; see the "Multilevel Regression" section). By default, all columns contained in the data frame passed to the function are used as predictors and are imputed. We can specify which variables will not be included in the imputation process by passing them as a character vector to the **idvars** parameter. We also give the random number

```
imp <- amelia(data_long,
              m = 50,
              ts = wave,
              ords = ords, # ords is a vector of column names
              noms = c(''gender''),
              idvars = c(''id''),
              parallel = 'snow', # «snow» for windows, ''multicore'' for
UNIX systems (Mac or Linux)
              ncpus = 4, # number of cores
              cl = parallel::makePSOCKcluster(4),
              p2s = 2)
```

generator (**seed**) a seed to ensure reproducibility. To speed up the process, the computations can be done in parallel on multiple cores (**see the parallel, ncpus, and cl parameters**). The multiply imputed datasets are stored in an object called imp.

To perform operations on the data frame after the MI process, we can call the **transform()** function. This function takes the multiply imputed datasets (**imp**) and a rule to generate a new variable, passed as a parameter. The new variable is created by assigning its name and

computation (e.g., new_var = old_var + 5) inside the function call (the column names used should be written without quotes). Here, we compute the mean of the negative affect with the values from item **lonely**, **nervous**, and **angry**:

```
imp <- transform(imp, neg_affect = rowMeans(cbind(lonely, nervous,
angry))).
```

To perform the analyses over the imputed datasets, we use the *Zelig* package (Choirat et al., 2020). To do so, we specify the model's formula, the source of data in the data parameter (here, **imp$imputations**), specify the model used (here "**ls**" for "least squares regression"). Please see the ***Zelig*** package's documentation for all available models (Choirat et al., 2020).

```
library(Zelig)
fit <- zelig(dreamval_1 ~ lonely + nervous + angry,
             data = imp$imputations,
             model = ``ls'',
             cite = FALSE)
summary(fit)
```

However, it is also relatively easy to loop over all the imputed data frames, store the results in an object and then combine them to obtain the results (see Heiss, 2018 for an example of melding regressions). To perform SEM or growth curves analysis with multiply imputed data, we can use the **sem.mi()** function of the semTools package (Jorgensen et al., 2021):

```
library(semTools)
model <- `dreamval ~ lonely + nervous + angry'
fit <- sem.mi(model,
              data = imp$imputations) # to work with mice, only use `imp'
summary(fit)
```

Finally, fitting multilevel models to multiply imputed data can be done with the merTools package (Knowles, & Frederick, 2020). In this model, we examine the change of the valence of the last dream over time, the influence of the negative affect (**neg_affect**, computed previously) and the interaction between those two variables (random slopes and intercepts are specified for each participants, as indicated by **"(wave|id)"** in the model formula):

```
library(merTools)
fit <- lmerModList(dreamval ~ wave + neg_affect + wave*neg_affect +
(wave|id),
                   data = imp$imputations) # to work with mice, only use
`imp'
summary(fit)
```

Conclusion

This report was designed to provide researchers and students using data from the COVID-19 Canada project with an understanding of planned and unplanned missing data as well as to equip them with methods to deal with missing values. FIML and MI are two validated state-of-the-art methods to handle missing data. Both methods allow obtaining unbiased estimates and retaining statistical power. These methods can be used with the project's data, for planned and unplanned missingness. Researchers using the data to address specific research questions must carefully choose the appropriate method, depending on the research question, the data and the planned statistical analyses. We hope the suggestions in this report are helpful to guide researchers and students in handling the missingness contained in the COVID-19 Canada project's data.

Collaborator

Mathieu Caron-Diotte, M. Sc.
Doctoral candidate
Department of Psychology, Université de Montréal

To cite this research report

Caron-Diotte, M., Dorfman, A., Pelletier-Dumas, M., Lacourse, É., Lina, J. M., Stolle, D., Taylor, D. M, & de la Sablonnière, R. (2021). *COVID-19 Canada: The end of the world as we know it? Technical Report No. 2. Handling planned and unplanned missing data.* Université de Montréal.

To visit our website

csdc-cecd.wixsite.com/covid19csi?lang=en

Recommended Readings

Allison, P. D. (2012). Handling missing data by maximum likelihood. *Statistics and Data Analysis* (Paper 312-2012). SAS Global Forum 2012. https://statisticalhorizons.com/wp-content/uploads/MissingDataByML.pdf

Enders, C. K. (2010). *Applied missing data analysis*. Guilford.

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based Structural Equation Models. *Structural Equation Modeling: A Multidisciplinary Journal, 10*(1), 80-100. https://doi.org/10.1207/S15328007SEM1001_4

Graham, J. W. (2009). Missing data analysis: Making it work in the real world. *Annual Review of Psychology, 60*(1), 549–576. https://doi.org/10.1146/annurev.psych.58.110405.085530

Little, T. D., & Rhemtulla, M. (2013). Planned missing data designs for developmental researchers. *Child Development Perspectives, 7*(4), 199–204. https://doi.org/10.1111/cdep.12043

van Buuren, S. (2018). *Flexible imputation of missing data* (2nd ed.). CRC. https://stefvanbuuren.name/fimd/

References

Allison, P. D. (2012). Handling missing data by maximum likelihood. *Statistics* and Data Analysis (Paper 312-2012). SAS Global Forum 2012. https://statisticalhorizons.com/wp-content/uploads/MissingDataByML.pdf

Audigier, V., & Resche-Rigon, M. (2019). *micemd: Multiple Imputation by Chained Equations with Multilevel Data* (Version 1.6.0) [R package]. https://CRAN.R-project.org/package=micemd

Bates, D. (2006, May 19). *[R] lmer, p-values and all that*. https://stat.ethz.ch/pipermail/r-help/2006-May/094765.html

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting linear mixed-effects models using lme4. *Journal of Statistical Software, 67*(1), 1–48. https://doi.org/10.18637/jss.v067.i01

Choirat, C., Honaker, J., Imai, K., King, G., Lau, O. (2020). Zelig: Everyone's statistical software (Version 5.1.7). https://zeligproject.org/.

de la Sablonnière, R., Dorfman, A. R., Lina, J.-M., Pelletier-Dumas, M., Stolle, D., Taylor, D. M., Benoît, Z., Boulanger, A., Caron-Diotte, M., Mérineau, S., & Nadeau, A. (2020). *COVID-19 Canada: The end of the world as we know it? Technical report: Presenting the COVID-19 Canada Survey*. Université de Montréal. https://csdc-cecd.wixsite.com/covid19csi/resultats

Enders, C. K. (2010). *Applied missing data analysis*. Guilford.

Graham, J. W. (2003). Adding missing-data-relevant variables to FIML-based Structural Equation Models. Structural *Equation Modeling: A Multidisciplinary Journal, 10*(1), 80–100. https://doi.org/10.1207/S15328007SEM1001_4

Graham, J. W. (2009). Missing data analysis: Making it work in the real world. *Annual Review of Psychology, 60*(1), 549–576. https://doi.org/10.1146/annurev.psych.58.110405.085530

Graham, J. W., Taylor, B. J., Olchowski, A. E., & Cumsille, P. E. (2006). Planned missing data designs in psychological research. *Psychological Methods, 11*(4), 323–343. https://doi.org/10.1037/1082-989X.11.4.323

Heiss, A. (2018, March 7). *Meld regression output from multiple imputations with tidyverse*. https://www.andrewheiss.com/blog/2018/03/07/amelia-tidy-melding/

Honaker, J., King, G., & Blackwell, M. (2011). Amelia II: A program for missing data. *Journal of Statistical Software, 45*(7). https://doi.org/10.18637/jss.v045.i07

Jorgensen, T. D., Pornprasertmanit, S., Schoemann, A. M., & Rosseel, Y. (2021). semTools: Useful tools for structural equation modeling (Version 0.5-5) [R package]. https://CRAN.R-project.org/package=semTools

Little, T. D., & Rhemtulla, M. (2013). Planned missing data designs for developmental researchers. *Child Development Perspectives, 7*(4), 199–204. https://doi.org/10.1111/cdep.12043

Madley-Dowd, P., Hughes, R., Tilling, K., & Heron, J. (2019). The proportion of missing data should not be used to guide decisions on multiple imputation. *Journal of Clinical Epidemiology, 110*, 63–73. https://doi.org/10.1016/j.jclinepi.2019.02.016

Rhemtulla, M., & Little, T. D. (2012). Planned missing data designs for research in cognitive development. *Journal of Cognition and Development, 13*(4), 425–438. https://doi.org/10.1080/15248372.2012.717340

Robitzsch, A., & Grund, S. (2021). miceadds: Some additional multiple imputation functions, especially for 'mice' (Version 3.11-6) [R package]. https://CRAN.R-project.org/package=miceadds.

Rosseel, Y. (2012). lavaan: An R package for structural equation modeling. *Journal of Statistical Software, 48*(2), 1–36. https://doi.org/10.18637/jss.v048.i02

van Buuren, S. (2018). *Flexible imputation of missing data* (2nd ed.). CRC. https://stefvanbuuren.name/fimd/

van Buuren, S., & Groothuis-Oudshoorn, K. (2011). mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software, 45*(3). https://doi.org/10.18637/jss.v045.i03

Venables, W. N., Smith, D. N., & R Core Team. (2021). *An Introduction to R (4.1.0)*. R Core Team. https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

RESEARCH TEAM

Principal investigator

Roxane de la Sablonnière, Ph.D.
Full professor
Department of Psychology
University of Montreal

Director and founder of the Social Change and Identity Laboratory since 2005, Roxane studies the challenges people face when they are exposed to dramatic social change, such as the colonization that affected Canada's Aboriginal peoples or immigration.

Team members

Anna Dorfman, Ph.D.
Assistant Professor
Department of Psychology
Bar Ilan University

A behavioural researcher Interested in decision-making processes, Anna focuses on the interactions between emotions, cognitions and behaviours in order to understand how individuals react when faced with difficult social situations.

Eric Lacourse, Ph.D.
Full Professor, Senior methodologist
Department of Sociology
University of Montreal

Éric Lacourse is a full professor in the Department of Sociology at the University of Montreal. He is currently responsible for the bi-disciplinary baccalaureate in psychology and sociology and formerly director of the microprogram in social statistics.

Jean-Marc Lina, Ph.D.
Professor
Department of Electrical Engineering
École de technologie supérieure de Montréal

Jean-Marc is the founder and director of the PhysNum laboratory, as well as a researcher at the Center for Advanced Research in Sleep Medecine of the Hôpital du Sacré-Cœur. He studies the dynamics of complex systems including rhythms in social psychology.

Mathieu Pelletier-Dumas, Ph.D.
Senior research advisor
Department of Psychology
University of Montreal

A social psychology researcher in the Social Change and Identity Laboratory, Mathieu is interested in the profound changes that people face (social and personal changes), in identity, and in negative behaviours (discrimination, prejudice, disruptive behaviours in video games).

Dietlind Stolle, Ph.D.
James McGill Professor
Department of Political Science
McGill University

Dietlind has directed the Centre for the Study of Democratic Citizenship. She is an expert on trust, social capital, ethnic diversity, attitudinal democratic backsliding and new forms of political participation.

Partners

CÉCD CSDC
CENTRE POUR L'ÉTUDE DE LA CITOYENNETÉ DÉMOCRATIQUE
CENTRE FOR THE STUDY OF DEMOCRATIC CITIZENSHIP

McGill UNIVERSITY

SSHRC CRSH

Social Change and Identity Laboratory

CIHR IRSC

ÉTS Le génie pour l'industrie
ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
Université du Québec

Fonds de recherche Société et culture Québec

MITACS

Université de Montréal