Last couple of days adding functionality for performing repeated cross-validation to cvms and groupdata2. In this quick post I will show an example.

In cross-validation, we split our training set into a number (often denoted “k”) of groups called folds. We repeatedly train our machine learning model on k-1 folds and test it on the last fold, such that each fold becomes test set once. Then we average the results and celebrate with food and music.

The benefits of using groupdata2 to create the folds are 1) that it allows us to balance the ratios of our output classes (or simply a categorical column, if we are working with linear regression instead of classification), and 2) that it allows us to keep all observations with a specific ID (e.g. participant/user ID) in the same fold to avoid leakage between the folds.

The benefit of cvms is that it trains all the models and outputs a tibble (data frame) with results, predictions, model coefficients, and other sweet stuff, which is easy to add to a report or do further analyses on. It even allows us to cross-validate multiple model formulas at once to quickly compare them and select the best model.

**Repeated Cross-validation**

In repeated cross-validation we simply repeat this process a couple of times, training the model on more combinations of our training set observations. The more combinations, the less one bad split of the data would impact our evaluation of the model.

For each repetition, we evaluate our model as we would have in regular cross-validation. Then we average the results from the repetitions and go back to food and music.

**groupdata2**

As stated, the role of groupdata2 is to create the folds. Normally it creates one column in the dataset called “.folds”, which contains a fold identifier for each observation (e.g. 1,1,2,2,3,3,1,1,3,3,2,2). In repeated cross-validation it simply creates multiple of such fold columns (“.folds\_1”, “.folds\_2”, etc.). It also makes sure they are unique, so we actually train on different subsets.

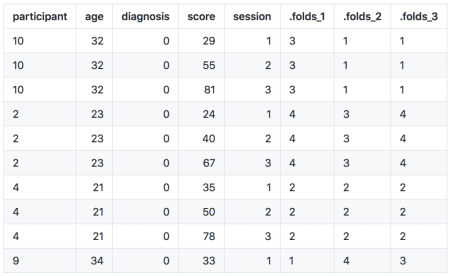
# Install groupdata2 and cvms

# Attach packages  
library(cvms) # cross\_validate()  
library(groupdata2) # fold()  
library(knitr) # kable()  
library(dplyr) # %>%

# Set seed for reproducibility  
set.seed(7)

# Fold data  
# Create 3 fold columns  
# cat\_col is the categorical column to balance between folds  
# id\_col is the column with IDs. Observations with the same ID will be put in the same fold.  
# num\_fold\_cols determines the number of fold columns, and thereby the number of repetitions.  
data <- fold(data, k = 4, cat\_col = 'diagnosis', id\_col = 'participant', num\_fold\_cols = 3)

# Show first 15 rows of data  
data %>% head(10) %>% kable()

[](https://i2.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/data_subset.png)

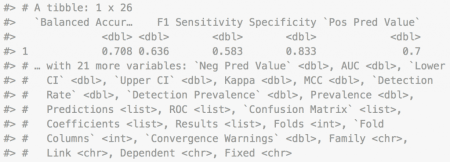
Data Subset with 3 Fold Columns

**cvms**

In the cross\_validate function, we specify our model formula for a logistic regression that classifies diagnosis. cvms currently supports linear regression and logistic regression, including mixed effects modelling. In the fold\_cols (previously called folds\_col), we specify the fold column names.

CV <- cross\_validate(data, "diagnosis~score",  
fold\_cols = c('.folds\_1','.folds\_2','.folds\_3'),  
family='binomial',  
REML = FALSE)

# Show results  
CV

[](https://i1.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/Screen-Shot-2019-05-05-at-14.06.27.png)

Output tibble

Due to the number of metrics and useful information, it helps to break up the output into parts:

CV %>% select(1:7) %>% kable()

[Repeated CV metrics 1](https://i0.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/Screen-Shot-2019-05-05-at-14.06.38.png)

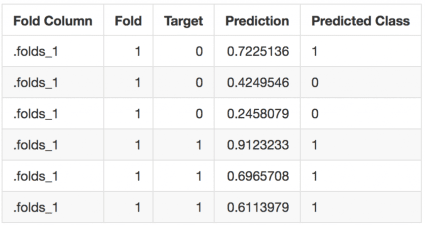
Evaluation metrics (subset 1)

CV %>% select(8:14) %>% kable()

[Repeated CV metrics 2](https://i0.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/Screen-Shot-2019-05-05-at-14.06.49.png)

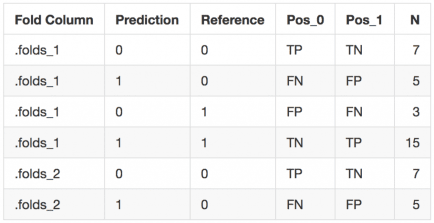
Evaluation metrics (subset 2)

CV$Predictions[[1]] %>% head() %>% kable()

[](https://i1.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/Screen-Shot-2019-05-05-at-14.06.58.png)

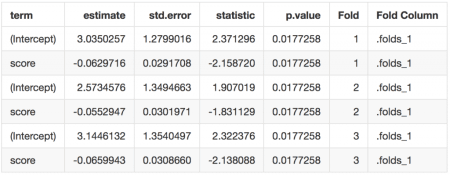
Nested predictions (subset)

CV$`Confusion Matrix`[[1]] %>% head() %>% kable()

[](https://i2.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/Screen-Shot-2019-05-05-at-14.07.08.png)

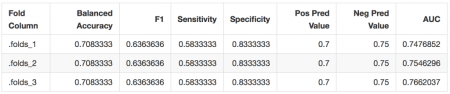
Nested confusion matrices (subset)

CV$Coefficients[[1]] %>% head() %>% kable()

[](https://i1.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/Screen-Shot-2019-05-05-at-14.07.17.png)

Nested model coefficients (subset)

CV$Results[[1]] %>% select(1:8) %>% kable()

[](https://i0.wp.com/ludvigolsen.dk/wp-content/uploads/2019/05/Screen-Shot-2019-05-05-at-14.07.51.png)

Nested results per fold column (subset)

We could have trained multiple models at once by simply adding more model formulas. That would add rows to the output, making it easy to compare the models.

The linear regression version has different evaluation metrics. These are listed in the help page at ?cross\_validate.

