medley: Predictive Modeling with Missing Data

useR! 2025, Duke University

Jason Bryer, Ph.D.

August 9, 2025

Agenda

1. Motivation for this package.
2. Exploring patterns of missing data.
3. Overview of some common approaches of training predictive models with missing data.
4. The `medley` approach to predictive modeling.
5. Compare the results across the various modeling approaches.

This research was developed under grant R305A210269 from the U.S. Department of Education. However, the contents do not necessarily represent the policy of the U.S. Department of Education, and you should not assume endorsement by the Federal Government.

Motivating Example

The Diagnostic Assessment and Achievement of College Skills (DAACS; www.daacs.net) is a suite of technological and social supports designed to optimize student learning. It asseses college readiness in a *no-stakes* environment.

Students complete **assessments** in **self-regulated learning, writing, mathematics, and reading comprehension**. They are then provided with immediate feedback in terms of one, two, and three dots (developing, emerging, and mastering, respectively) and receive customized strategies and resources based upon their results.

One of our primary research question is whether **can DAACS improve the accuracy of predicting student success** above what institutions already know?

Problem is that although students were expected to complete DAACS as part of orientation, many students did not attend orientation, and then some students did not complete all four assessments.

Data Source

Data for this study was collected as part of a large scale randomized control trial.

Online institution of predominately adult learners.

Most students have some prior college experience and transfer in many credits.

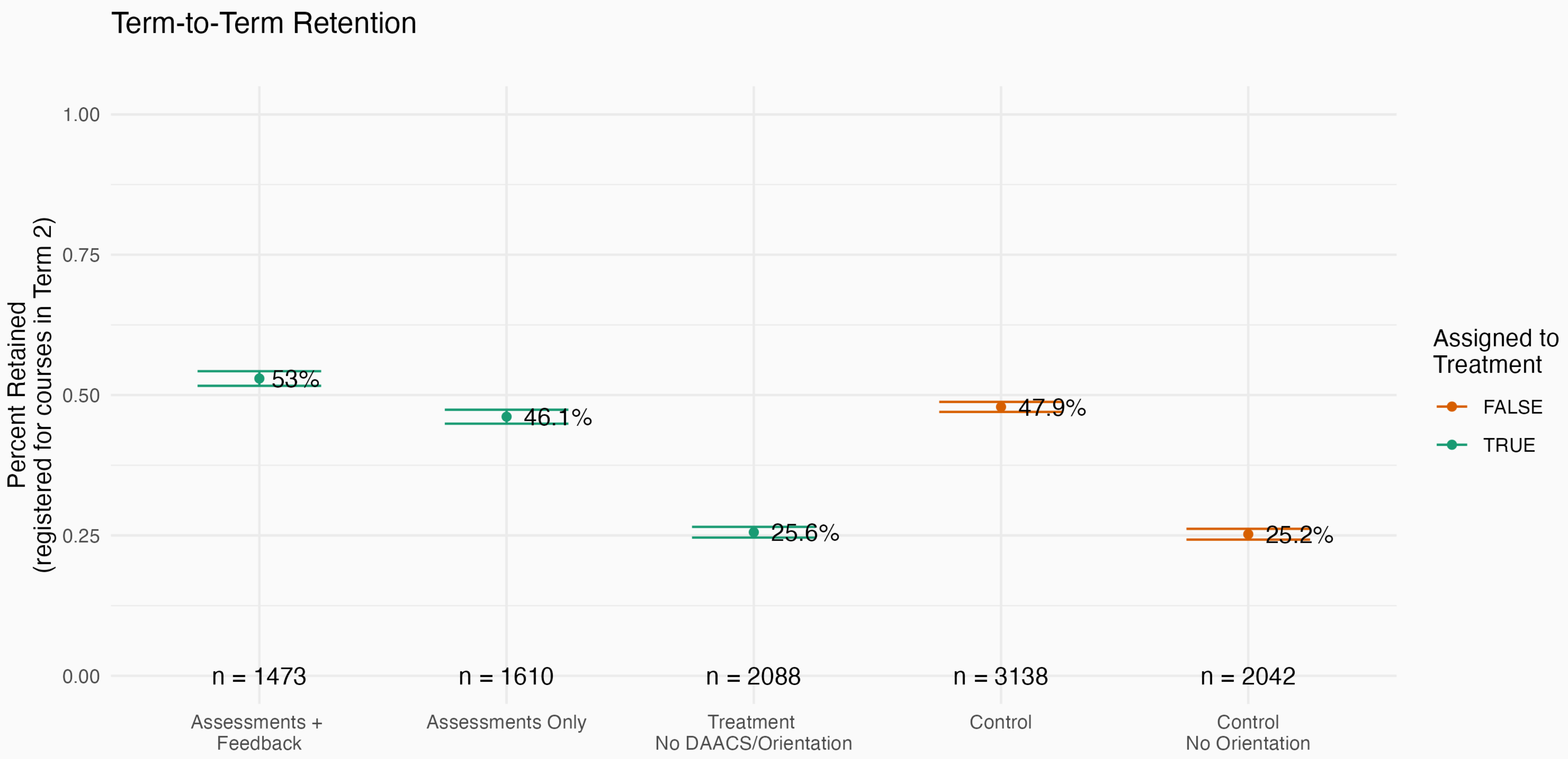
Our outcome measure of interest is **retained** which is `TRUE` if a student either graduates or returns for a second term.

The latest development version can be downloaded from Github (will be on CRAN soon).

```
remotes::isntall_github('jbryer/medley')
```

```
data(daacs, package = 'medley')
(null_accuracy <- mean(daacs$retained))
#> [1] 0.5616997
```

Study Outcomes



See Bryer, Andrade, Cleary, Lui, Franklin, & Akhmedjanova, (in press) for more details.

Missing Data Pattern

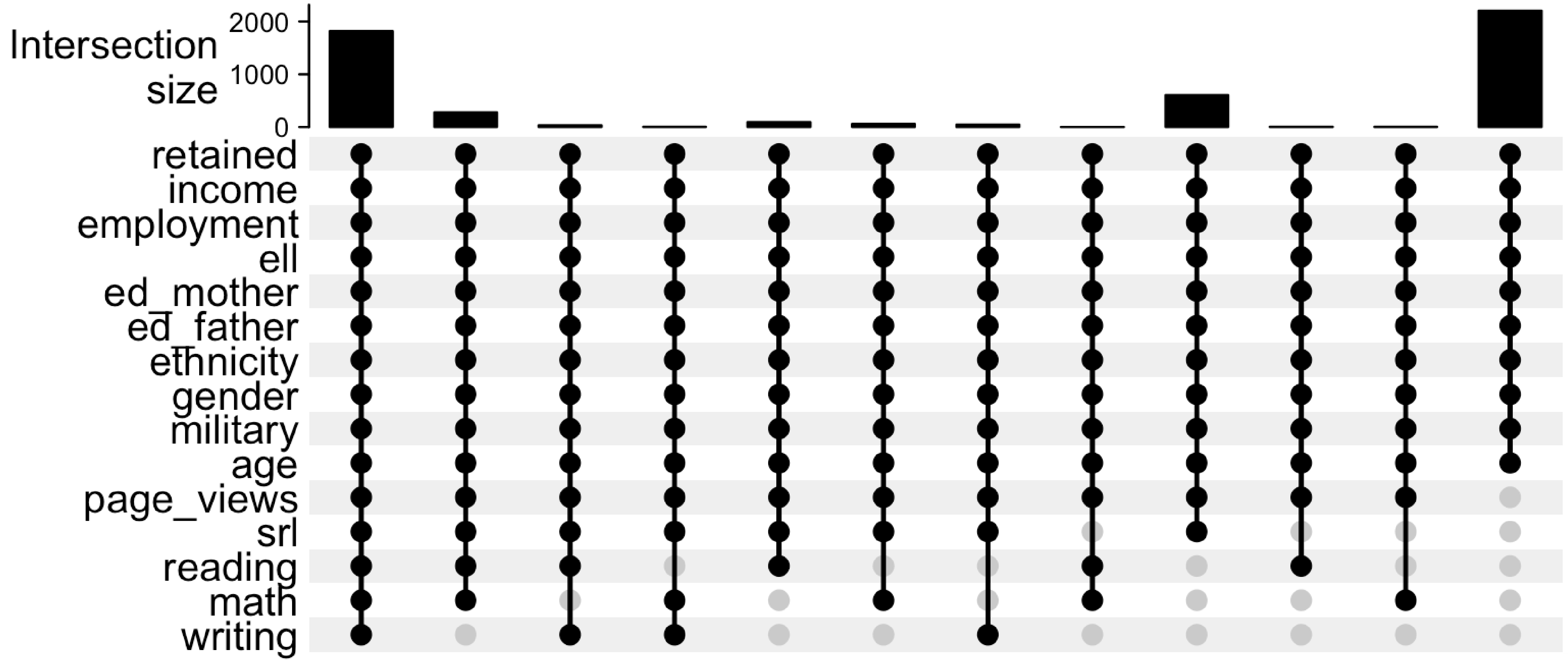


Figure created using the `ComplexHeatmap` package.

How to Handle Missing Data?

1. Exclude observations or variables with missing data.
2. Impute missing data.

   a. Mean imputation.

   b. Multiple imputation.

3. Use methods that does not require complete data (e.g. `xgboost`).
4. Train multiple models using the available data (the `medley` approach).

For methods 1, 2, and 4 we will use both logistic regression and random forests with each of these approaches. Why random forest? See Fernandez-Delgado, Cernadas, Barro, and Amorim (2014).

Data Preparation

To perform the predictive modeling we will split the data into training (70%) and validation (30%) data sets.

```
set.seed(2112); train_rows <- sample(nrow(daacs), nrow(daacs) * 0.7)
daacs_train <- daacs[train_rows,]
daacs_valid <- daacs[-train_rows,]
```

```
str(daacs)
#> 'data.frame':    5154 obs. of  15 variables:
#>  $ retained  : logi  TRUE FALSE FALSE TRUE FALSE FALSE ...
#>  $ page_views: int  NA 120 NA 74 NA NA 29 NA NA NA ...
#>  $ srl       : num  NA 2.58 NA 2.75 NA ...
#>  $ math      : num  NA 0.5 NA 0.333 NA ...
#>  $ reading   : num  NA 0.944 NA 0.944 NA ...
#>  $ writing   : num  NA 0.722 NA 0.889 NA ...
#>  $ income    : int  4 3 4 6 6 5 8 9 8 4 ...
#>  $ employment: int  3 3 3 3 1 3 3 3 3 3 ...
#>  $ ell       : num  0 1 1 1 0 1 1 1 1 1 ...
#>  $ ed_mother : int  2 3 3 2 1 3 7 3 6 1 ...
#>  $ ed_father : int  2 4 3 3 1 3 7 6 4 1 ...
#>  $ ethnicity : Factor w/ 4 levels "Black or African American",..: 4 4 4 4 3 1 1 4 3 4 ...
#>  $ gender    : Factor w/ 2 levels "FEMALE","MALE": 2 1 1 1 2 1 2 2 2 1 ...
#>  $ military  : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
#>  $ age       : num  53.5 63.2 56.3 54.9 69.6 ...
```

Using Complete Data

First we need to subset our training data frame with variables where there are no missing values.

```
missingness <- apply(daacs_train, 2, FUN = function(x) { sum(is.na(x)) / length(x)})
daacs_train_complete <- daacs_train[,names(missingness)[missingness == 0]]
names(daacs_train_complete)
#>  [1] "retained"   "income"     "employment" "ell"        "ed_mother"
#>  [6] "ed_father"  "ethnicity"  "gender"     "military"   "age"
```

We will train a model using logistic regression...

```
lr_out <- glm(data = daacs_train_complete,
              formula = retained ~ .,
              family = binomial(link = 'logit'))
```

And random forests...

```
rf_out <- randomForest(formula = factor(retained) ~ .,
                       data = daacs_train_complete)
```

Using Complete Data Results

Logistic Regression

```
lr_predictions <- predict(lr_out,
                          newdata = daacs_valid,
                          type = 'response')
confusion_matrix(observed = daacs_valid$retained,
                 predicted = lr_predictions > 0.5)
#>                  predicted
#>   observed          FALSE           TRUE
#>      FALSE 295 (19.07%) 386 (24.95%)
#>       TRUE 216 (13.96%) 650 (42.02%)
#> Accuracy: 61.09%
#> Sensitivity: 43.32%
#> Specificity: 75.06%
```

Random Forest

```
rf_predictions <- predict(rf_out,
                          newdata = daacs_valid,
                          type = 'response')
confusion_matrix(observed = daacs_valid$retained,
                 predicted = rf_predictions)
#>                  predicted
#>   observed          FALSE           TRUE
#>      FALSE 279 (18.03%) 402 (25.99%)
#>       TRUE 251 (16.22%) 615 (39.75%)
#> Accuracy: 57.79%
#> Sensitivity: 40.97%
#> Specificity: 71.02%
```

Mean Imputation

```r
daacs_complete_mean <- daacs                                             # Create a copy of the data.frame
for(i in 2:ncol(daacs_complete_mean)) {                                  # Loop through all the variables,
    missing_rows <- is.na(daacs_complete_mean[,i])                       # find the rows with missing
    if(sum(missing_rows) > 0) {                                          # values and replace with mean.
        daacs_complete_mean[missing_rows, i] <- mean(daacs_complete_mean[,i], na.rm = TRUE)
    }
}

daacs_train_complete_mean <- daacs_complete_mean[train_rows,]            # Split into training and
daacs_valid_complete_mean <- daacs_complete_mean[-train_rows,]           # validation data.frames

mean_lr_out <- glm(formula = retained ~ .,                               # Logistic regression
                   data = daacs_train_complete_mean,
                   family = binomial(link = logit))
mean_lr_predictions <- predict(mean_lr_out,                              # Random forest
                               newdata = daacs_valid_complete_mean,
                               type = 'response')

mean_rf_out <- randomForest(formula = factor(retained) ~ .,              # Logistic regression predictions
                            data = daacs_train_complete_mean)
mean_rf_predictions <- predict(mean_rf_out,                              # Random forest predictions
                               newdata = daacs_valid_complete_mean,
                               type = 'response')
```

Mean Imputation Results

Logistic Regression

```
confusion_matrix(observed = daacs_valid_complete_mean$r
                 predicted = mean_lr_predictions > 0.5)
#>                 predicted
#>   observed        FALSE         TRUE
#>      FALSE 292 (18.88%) 389 (25.15%)
#>       TRUE 212 (13.70%) 654 (42.28%)
#> Accuracy: 61.15%
#> Sensitivity: 42.88%
#> Specificity: 75.52%
```

Random Forests

```
confusion_matrix(observed = daacs_valid_complete_mean$r
                 predicted = mean_rf_predictions)
#>                 predicted
#>   observed        FALSE         TRUE
#>      FALSE 316 (20.43%) 365 (23.59%)
#>       TRUE 178 (11.51%) 688 (44.47%)
#> Accuracy: 64.9%
#> Sensitivity: 46.4%
#> Specificity: 79.45%
```

Multiple Imputation

We will use the `mice` package to do multiple imputation.

```
mice_out <- mice::mice(daacs[,-1], M = 5, seed = 2112, printFlag = FALSE)
daacs_complete_mice <- cbind(retained = daacs$retained, mice::complete(mice_out))

daacs_train_complete_mice <- daacs_complete_mice[train_rows,]
daacs_valid_complete_mice <- daacs_complete_mice[-train_rows,]
```

```
mice_lr_out <- glm(formula = retained ~ .,
                   data = daacs_train_complete_mice,
                   family = binomial(link = logit))
mice_lr_predictions <- predict(mice_lr_out,
                               newdata = daacs_valid_complete_mice,
                               type = 'response')

mice_rf_out <- randomForest(formula = factor(retained) ~ .,
                            data = daacs_train_complete_mice)
mice_rf_predictions <- predict(mice_rf_out,
                               newdata = daacs_valid_complete_mice,
                               type = 'response')
```

Multiple Imputation Results

Logistic Regression

```
confusion_matrix(observed = daacs_valid_complete_mice$r
                 predicted = mice_lr_predictions > 0.5)
#>                predicted
#>   observed          FALSE          TRUE
#>      FALSE 297 (19.20%) 384 (24.82%)
#>       TRUE 213 (13.77%) 653 (42.21%)
#> Accuracy: 61.41%
#> Sensitivity: 43.61%
#> Specificity: 75.4%
```

Random Forests

```
confusion_matrix(observed = daacs_valid_complete_mice$r
                 predicted = mice_rf_predictions)
#>                predicted
#>   observed          FALSE          TRUE
#>      FALSE 278 (17.97%) 403 (26.05%)
#>       TRUE 207 (13.38%) 659 (42.60%)
#> Accuracy: 60.57%
#> Sensitivity: 40.82%
#> Specificity: 76.1%
```

XGBoost

```r
library(xgboost)
options(na.action='na.pass')
train_data <- Matrix::sparse.model.matrix(retained ~ ., data = daacs_train)
xg_out <- xgboost(data = train_data,
                  label = daacs_train$retained == TRUE,
                  nrounds = 100,
                  objective = "binary:logistic")
xg_predicted <- predict(xg_out, Matrix::sparse.model.matrix(retained ~ ., data = daacs_valid))
```

Results

```r
confusion_matrix(observed = daacs_valid$retained,
                 predicted = xg_predicted > 0.5)
#>                 predicted
#>   observed         FALSE          TRUE
#>      FALSE 348 (22.50%) 333 (21.53%)
#>       TRUE 248 (16.03%) 618 (39.95%)
#> Accuracy: 62.44%
#> Sensitivity: 51.1%
#> Specificity: 71.36%
```

Medley Approach

The goal of the `medley` package is to provide a framework for training models based upon the patterns of missing data.

Be default, an observation will be used in the model with the most number of predictor varaibles.

```
get_variable_sets(data = daacs, formula = retained ~ ., min_set_size = 0.1)
#> [[1]]
#> retained ~ page_views + srl + math + reading + writing + income +
#>     employment + ell + ed_mother + ed_father + ethnicity + gender +
#>     military + age
#>
#> [[2]]
#> retained ~ page_views + srl + income + employment + ell + ed_mother +
#>     ed_father + ethnicity + gender + military + age
#>
#> [[3]]
#> retained ~ income + employment + ell + ed_mother + ed_father +
#>     ethnicity + gender + military + age
```

medley

Parameters for the `medley()` function:

- `data` - The dataset.
- `formula` - Formula. This will typically only include the dependent variable, that is `Y ~ .`.
- `method` - The actual modeling function. The default is `stats::glm`.
- `var_sets` - A list of formulas used for the various models. Typically the `get_varaible_sets()` function will provide reasonable defaults.
- `min_set_size` - The minimum (as a percentage) size of any dataset to train a model. The default is 10%.
- `exclusive_membership` - If `TRUE`, any observation will be used in exactly one model.
- `...` - Other parameters passed to the `method` function. Default is `FALSE`.

The object returned by `medley()` contains the following elements:

- `n_models` - The number of models estimated.
- `formulas` - A list of the formulas used for each model.
- `models` - A list containing the model output for each model. In this example this would contain the results of the `glm` function call.
- `data` - The full data set used to train the models.
- `model_observations` - A data frame indicating which models each observation was used in. The rows correspond to the rows in `data` and the columns correspond to the model.

Medley Trainig and Predicting

Training

You can use `medley` like most modeling functions in R (i.e. there are `data` and `formula` parameters).

```
medley_lr_out <- medley(
    data = daacs_train,
    formula = retained ~ .,
    method = glm,
    family = binomial(link = logit))
```

```
medley_rf_out <- medley(
    data = daacs_train,
    formula = retained ~ .,
    method = randomForest)
```

Predicting

Also like most modeling functions, the `predict()` function works as expected.

```
medley_lr_predictions <- predict(
    medley_lr_out,
    newdata = daacs_valid,
    type = 'response')
```

```
medley_rf_predictions <- predict(
    medley_rf_out,
    newdata = daacs_valid,
    type = "response") == 2
```

Medley Results

Logistic Regression

```
confusion_matrix(observed = daacs_valid$retained,
                 predicted = medley_lr_predictions > 0.
#>                 predicted
#>   observed         FALSE          TRUE
#>      FALSE 305 (19.72%) 376 (24.31%)
#>       TRUE 175 (11.31%) 691 (44.67%)
#> Accuracy: 64.38%
#> Sensitivity: 44.79%
#> Specificity: 79.79%
```

Random Forests

```
confusion_matrix(observed = daacs_valid$retained,
                 predicted = medley_rf_predictions )
#>                 predicted
#>   observed         FALSE          TRUE
#>      FALSE 317 (20.49%) 364 (23.53%)
#>       TRUE 194 (12.54%) 672 (43.44%)
#> Accuracy: 63.93%
#> Sensitivity: 46.55%
#> Specificity: 77.6%
```

Medley Summary

The `summary` function will provide some insight into what Medley is doing. Here, we trained three models.

```
summary(medley_lr_out)
#>    Model    n  Success
#> m1     1 1268 70.97792
#> m2     2  788 58.24873
#> m3     3 1551 43.19794
#>
#> m1 retained ~ page_views + srl + math + reading + writing + income + employment + ell + ed_mother + ed_father + et
#> m2                           retained ~ page_views + srl + income + employment + ell + ed_mother + ed_father + et
#> m3                                                retained ~ income + employment + ell + ed_mother + ed_father + et
```

You can get further details about the individual model results using the `models` element on the returned object.

Medley Model Summaries

	(1)		(2)		(3)	
(Intercept)	0.290	(0.742)	0.170	(0.751)	-0.245	(0.376)
page_views	-0.004	(0.003)	-0.002	(0.004)		
srl	-0.132	(0.159)	-0.190	(0.180)		
math	0.429	(0.382)				
reading	-0.474	(0.464)				
writing	0.153	(0.425)				
income	-0.003	(0.029)	0.046	(0.035)	0.062 *	(0.025)
employment	0.152	(0.087)	-0.190	(0.104)	-0.047	(0.072)
ell	0.034	(0.268)	0.436	(0.318)	0.311	(0.210)
ed_mother	0.056	(0.045)	-0.001	(0.053)	-0.036	(0.037)
ed_father	0.011	(0.044)	0.093	(0.052)	0.024	(0.036)
ethnicityHispanic	-0.241	(0.251)	0.438	(0.264)	0.123	(0.186)
ethnicityOther	-0.015	(0.265)	0.212	(0.284)	0.250	(0.201)
ethnicityWhite	0.053	(0.198)	0.348	(0.194)	0.069	(0.134)
genderMALE	0.179	(0.146)	-0.048	(0.173)	0.280 *	(0.120)
militaryTRUE	0.869 ***	(0.155)	0.643 ***	(0.178)	0.721 ***	(0.123)

Comparing Results

Method	Accuracy	Improvement
Observed data only logistic regression	61.09	4.92
Observed data only random forest	57.79	1.62
Mean imputed data set with logistic regression	61.15	4.98
Mean imputed data set with random forest	64.90	8.73
Mice imputed data set logistic regression	61.41	5.24
Mice imputed data set random forest	60.57	4.40
XGboost	62.44	6.27
Medley with logistic regression	64.38	8.21
Medley with random forest	63.93	7.76

Note: Improvement is the difference with the overall base retention rate of 56.17%.

Comparing Results (cont.)

		Predicted		
Model	Observed	FALSE	TRUE	Accuracy
Observed data only logistic regression	FALSE	295 (19.07%)	386 (24.95%)	
	TRUE	216 (13.96%)	650 (42.02%)	
				61.09%
Observed data only random forest	FALSE	279 (18.03%)	402 (25.99%)	
	TRUE	251 (16.22%)	615 (39.75%)	
				57.79%
Imputed data set logistic regression	FALSE	297 (19.20%)	384 (24.82%)	
	TRUE	213 (13.77%)	653 (42.21%)	
				61.41%
Imputed data set random forest	FALSE	278 (17.97%)	403 (26.05%)	
	TRUE	207 (13.38%)	659 (42.60%)	
				60.57%
XGboost	FALSE	348 (22.50%)	333 (21.53%)	
	TRUE	248 (16.03%)	618 (39.95%)	
				62.44%
Medley with logistic regression	FALSE	305 (19.72%)	376 (24.31%)	
	TRUE	175 (11.31%)	691 (44.67%)	
				64.38%
Medley with random forest	FALSE	317 (20.49%)	364 (23.53%)	
	TRUE	194 (12.54%)	672 (43.44%)	
				63.93%

Reusing Data

Recall that we have a number of variables where there was no missing data. We could potentially use all observations for the "base" model (i.e. the model we use when there is missing data in the other variables).

We set `exclusive_membership = FALSE` to allow observations to be used in multiple models.

```
medley_rf_out2 <- medley(data = daacs_train, formula = retained ~ ., method = randomForest,
                         exclusive_membership = FALSE)
medley_rf_predictions2 <- predict(medley_rf_out2, newdata = daacs_valid,type = "response")
confusion_matrix(observed = daacs_valid$retained, predicted = medley_rf_predictions2)
#>              predicted
#>    observed      FALSE          TRUE
#>       FALSE 0 (0.00%) 681 (44.02%)
#>        TRUE 0 (0.00%) 866 (55.98%)
#> Accuracy: 55.98%
#> Sensitivity: 0%
#> Specificity: 100%
```

In this case, it turns out to not help increase accuracy, in part because we know there is an interaction between missingness and some demographics.

Conclusion

- Training predictive models can be challenging when there is a large amount of systematicly missing data.

- The medley approach performs very well compared to other approaches.

    - The only method performing better was mean imputation, which honestly, makes me very unforgettable 🤢

Miscellaneous

- Downsampling - the `downsampling()` function is designed for situations where there is significant imbalance in you dependent variable.

- `recode_small_levels()` is a utility function that will find rarely used factor levels and recode them.

Thank you!

jason.bryer@cuny.edu

@jbryer

@jbryer@vis.social

github.com/jbryer/medley