

A short `fscaret` package introduction with examples

Jakub Szlek (j.szlek@uj.edu.pl)

October 7, 2016

1 Installation

As it is in case of `caret`, the `fscaret` uses large number of R packages but it loads them when needed. To fully take advantage of the package it is recommended to install it both with dependent and suggested packages. Install `fscaret` with the command

```
> install.packages("fscaret", dependencies = c("Depends", "Suggests"))
```

from R console.

Be advised! Running above code would install all possible packages (in some cases more than 240!), but it is necessary to fully benefit from `fscaret`. If you wish to use only specific algorithms, check which parameter from `funcRegPred` corresponds to which package.

In the second case install `fscaret` with the command

```
> install.packages("fscaret", dependencies = c("Depends"))
```

2 Overview

In general `fscaret` is a wrapper module. It uses the engine of `caret` to build models and to get the variable ranking from them. When models are build package tries to draw variable importance from them directly or indirectly. The raw feature ranking would be worthless, since the results in this form cannot be compared.

That is why within `fscaret` the scaling process was introduced. Developed models are used to get prediction errors (RMSE and MSE). Finally the output is produced. It contains the data frame of variable importance, errors for all build models and preprocessed data set if the `preprocessData = TRUE`

when calling the main `fscaret()` function. Also is possible to retrieve the original models build with `train()` function of `caret`, to do this you should set `saveModel=TRUE` in the call of `fscaret` function.

In summary the whole feature ranking process can be divided into:

1. User provides input data sets and a few settings
2. Models are build
3. Variable rankings are draw out of the models
4. Generalization error is calculated for each model
5. Variable rankings are scaled according to generalization error
6. The results are gathered in tables

3 Input

3.1 Data set format

Be advised that `fscaret` assumes that data sets are in MISO format (multiple input single output). The example of such (with header) is:

Input_no1	Input_no2	Input_no3	...	Output
2	5.1	32.06	...	1.02
5	1.21	2.06	...	7.2

For more information on reading files in R, please write `?read.csv` in R console. If `fscaret()` function is switched to `classPred=TRUE`, the output must be in binary format (0/1).

3.2 An example

There are plenty of methods to introduce data sets into R. The best way is to read file (preasumably csv with `tab` as column separator) as follows:

1. Select file name

```
> basename_file <- "My_database"  
> file_name <- paste(basename_file, ".csv", sep="")
```

2. Read data file into matrix

```
> matrixTrain <- read.csv(file_name,header=TRUE,sep="\t",  
+ strip.white = TRUE, na.strings = c("NA",""))
```

3. Put loaded matrix into `data.frame`

```
> matrixTrain <- as.data.frame(matrixTrain)
```

Be advised to use `header=TRUE` when you have data set with column names as first row and `header=FALSE` when there are no column names.

Starting from `fscaret` version 0.9 setting `header` is fixed to `TRUE`.

The last step is obligatory to introduce data into `fscaret` functions as it checks if the data presented is in `data.frame` format.

4 Function `fscaret()`

4.1 Settings

All the settings are documented in Reference manual of `fscaret` <http://cran.r-project.org/web/packages/fscaret/fscaret.pdf>. Here we will concentrate only on a few valuable ones.

- `installReqPckg` The default setting is `FALSE`, but if set to `TRUE` prior to calculations it installs all packages from the sections ‘Depends’ and ‘Suggests’ of DESCRIPTION. Please be advised to be logged as root (admin) if you want to install packages for all users.
- `preprocessData` The default setting is `FALSE`, but if set to `TRUE` prior to calculations it performs the data preprocessing, which in short is realized in two steps:
 1. Check for near zero variance predictors and flag as near zero if:
 - the percentage of unique values is less than 20
 - the ratio of the most frequent to the second most frequent value is greater than 20,
 2. Check for susceptibility to multicollinearity
 - Calculate correlation matrix
 - Find variables with correlation 0.9 or more and delete them
- `regPred` Default option is `TRUE` and so the regression models are applied
- `classPred` Default option is `FALSE` and if set `classPred=TRUE` remember to set `regPred=FALSE`

- `myTimeLimit` Time limit in seconds for single model development, be advised that some models need as time to be build, if the option is omitted, the standard 24-hours time limit is applied. This function is off on non-Unix like systems.
- `Used.funcRegPred` Vector of regression models to be used, for all available models please enter `Used.funcRegPred="all"`, the listing of functions is:

```
> library(fscaret)
> data(funcRegPred)
> funcRegPred

 [1] "ANFIS"           "avNNet"           "bag"
 [4] "bagEarth"       "bayesglm"        "bdk"
 [7] "blackboost"     "Boruta"          "bstLs"
[10] "bstSm"          "bstTree"         "cforest"
[13] "ctree"          "ctree2"          "cubist"
[16] "DENFIS"        "dnn"             "earth"
[19] "elm"           "enet"            "evtree"
[22] "extraTrees"    "FIR.DM"          "foba"
[25] "FS.HGD"        "gam"             "gamboost"
[28] "gamLoess"      "gamSpline"       "gaussprLinear"
[31] "gaussprPoly"   "gaussprRadial"   "gbm"
[34] "gcvEarth"      "GFS.FR.MOGAL"    "GFS.LT.RS"
[37] "GFS.Thrift"    "glm"             "glmboost"
[40] "glmnet"        "glmStepAIC"      "HYFIS"
[43] "icr"           "kernelpls"       "kkn"
[46] "knn"           "krlsPoly"        "krlsRadial"
[49] "lars"          "lars2"           "lasso"
[52] "leapBackward"  "leapForward"     "leapSeq"
[55] "lm"           "lmStepAIC"       "logicBag"
[58] "logreg"        "M5"              "M5Rules"
[61] "mlp"          "mlpWeightDecay"  "neuralnet"
[64] "nnet"         "nodeHarvest"     "parRF"
[67] "partDSA"      "pcaNNet"         "pcr"
[70] "penalized"    "pls"             "plsRglm"
[73] "ppr"          "qrf"             "qrnn"
[76] "relaxo"       "rf"              "ridge"
[79] "rknn"         "rknnBel"        "rlm"
[82] "rpart"        "rpart2"          "RRF"
[85] "RRFglobal"    "rvmLinear"       "rvmPoly"
[88] "rvmRadial"    "SBC"             "simpls"
[91] "spls"         "superpc"         "svmBoundrangeString"
[94] "svmExpoString" "svmLinear"       "svmPoly"
[97] "svmRadial"    "svmRadialCost"  "svmSpectrumString"
[100] "treebag"     "widekernelpls"  "WM"
[103] "xyf"
```

- `Used.funcClassPred` Vector of classification models to be used, for all available models please enter `Used.funcClassPred="all"`, the listing of functions is:

```
> library(fscaret)
> data(funcClassPred)
> funcClassPred
```

```
[1] "ada"           "bagFDA"           "brnn"
[4] "C5.0"         "C5.0Cost"        "C5.0Rules"
[7] "C5.0Tree"     "CSimca"          "fda"
[10] "FH.GBML"     "FRBCS.CHI"      "FRBCS.W"
[13] "GFS.GCCL"    "gpls"           "hda"
[16] "hdda"        "J48"            "JRip"
[19] "lda"         "lda2"           "Linda"
[22] "LMT"         "LogitBoost"     "lssvmLinear"
[25] "lssvmPoly"   "lssvmRadial"    "lvq"
[28] "mda"         "Mlda"           "multinom"
[31] "nb"          "oblique.tree"   "OneR"
[34] "ORFlog"      "ORFpls"         "ORFridge"
[37] "ORFsvm"      "pam"            "PART"
[40] "pda"         "pda2"           "PenalizedLDA"
[43] "plr"         "protoclass"     "qda"
[46] "QdaCov"      "rbf"            "rda"
[49] "rFerns"     "RFlda"          "rocc"
[52] "rpartCost"   "rrlda"          "RSimca"
[55] "sda"         "sddaLDA"        "sddaQDA"
[58] "SLAVE"       "slda"           "smda"
[61] "sparseLDA"   "stepLDA"        "stepQDA"
[64] "svmRadialWeights" "vbmpRadial"    "avNNet"
[67] "bag"         "bagEarth"       "bayesglm"
[70] "bdk"         "blackboost"     "Boruta"
[73] "bstLs"       "bstSm"          "bstTree"
[76] "cforest"    "ctree"          "ctree2"
[79] "dmn"         "earth"          "elm"
[82] "evtree"     "extraTrees"     "gam"
[85] "gamboost"   "gamLoess"       "gamSpline"
[88] "gaussprLinear" "gaussprPoly"   "gaussprRadial"
[91] "gbm"        "gcvEarth"       "glm"
[94] "glmboost"   "glmnet"         "glmStepAIC"
[97] "kernelpls"  "kknn"           "knn"
[100] "logicBag"   "logreg"         "mlp"
[103] "mlpWeightDecay" "nnet"          "nodeHarvest"
[106] "parRF"      "partDSA"        "pcaNNet"
[109] "pls"        "plsRglm"        "rf"
[112] "rknn"       "rknnBel"        "rpart"
[115] "rpart2"     "RRF"            "RRFglobal"
[118] "simpls"     "splS"           "svmBoundrangeString"
[121] "svmExpoString" "svmLinear"      "svmPoly"
[124] "svmRadial"  "svmRadialCost" "svmSpectrumString"
[127] "treebag"   "widekernelpls" "xyf"
```

- `no.cores` The default setting is `NULL` as to maximize the CPU utilization and to use all available cores.

- `missData` This option handles the missing data. Possible values are:
 - `missData="delRow"` - for deletion of observations (rows) with missing values,
 - `missData="delCol"` - for deletion of attributes (columns) with missing values,
 - `missData="meanCol"` - for imputing mean to missing values,
 - `missData=NULL` - no action is taken.
- `supress.output` Default option is `FALSE`, but it is sometimes justified to suppress the output of intermediate functions and focus on ranking predictions.
- `saveModel` Default option is `FALSE` as some models have large size, and therefore saving all obtained would lead to 100-500 MB RData files. Keep in mind that loading such large objects into R would require a lot of RAM, e.g. 140MB RData file consumes about 1.5GB of RAM. On the other hand one may want to utilize developed models. To export a model from a result of `fscaret()` function, e.g. `myFS` object:

```
> my_res_foba <- myFS$VarImp$model$foba
> my_res_foba <- structure(my_res_foba,class="train")
```

4.2 Regression problems - an example

A simple example of regression problem utilizing the data provided in the `fscaret`:

```
> library(fscaret)
> data(dataset.train)
> data(dataset.test)
> trainDF <- dataset.train
> testDF <- dataset.test
> myFS<-fscaret(trainDF, testDF, myTimeLimit = 5, preprocessData=TRUE,
+             Used.funcRegPred=c("pcr", "pls"), with.labels=TRUE,
+             supress.output=TRUE, no.cores=1)
> myRES_tab <- myFS$VarImp$matrixVarImp.MSE[1:10,]
> myRES_tab <- subset(myRES_tab, select=c("pcr", "pls", "SUM%", "ImpGrad", "Input_no"))
> myRES_rawMSE <- myFS$VarImp$rawMSE
> myRES_PPlabels <- myFS$PPlabels
```

4.3 Classification problems - an example

An example of classification problem utilizing the data `data(Pima.te)` in the MASS:

```

> library(MASS)
> # make testing set
> data(Pima.te)
> Pima.te[,8] <- as.numeric(Pima.te[,8])-1
> myDF <- Pima.te
> myFS.class<-fscaret(myDF, myDF, myTimeLimit = 5, preprocessData=FALSE,
+                    with.labels=TRUE, classPred=TRUE,regPred=FALSE,
+                    Used.funcClassPred=c("knn","rpart"), supress.output=TRUE, no.cores=1)
> myRES.class_tab <- myFS.class$VarImp$matrixVarImp.MeasureError
> myRES.class_tab <- subset(myRES.class_tab, select=c("knn","rpart","SUM%","ImpGrad","Input_no"))
> myRES.class_rawError <- myFS.class$VarImp$rawMeasureError

```

5 Output

For regression problems, as it was stated previously there are three lists of outputs.

1. Feature ranking and generalization errors for models:

```

> # Print out the Variable importance results for MSE scaling
> print(myRES_tab)

```

	pcr	pls	SUM%	ImpGrad	Input_no
1	5.862841e+01	5.227714e+01	1.000000e+02	0.000000	4
2	1.567799e+01	2.741963e+01	3.885975e+01	61.140253	5
3	1.916511e+01	1.995465e+01	3.527303e+01	9.229896	22
4	2.519981e-01	3.161112e-01	5.122461e-01	98.547769	23
5	1.872058e-02	3.079973e-02	4.465089e-02	91.283313	2
6	2.880832e-04	1.324904e-03	1.454379e-03	96.742777	13
7	5.880416e-04	2.781880e-04	7.810516e-04	46.296556	9
8	7.190168e-05	6.894892e-05	1.270005e-04	83.739807	1
9	1.570926e-06	2.697715e-06	3.848898e-06	96.969384	17
10	1.081909e-06	1.078743e-06	1.948191e-06	49.383143	21

2. Raw RMSE/MSE errors for each model

```

> # Print out the generalization error for models
> print(myRES_rawMSE)

```

	pcr	pls
1	716.6597	671.8195

3. Reduced data frame of inputs after preprocessing

```

> # Print out the reduced number of inputs after preprocessing
> print(myRES_PPlabels)

```

Orig.Input.No		Labels
1	1	Balaban.index
2	2	Dreiding.energy
3	3	Fused.aromatic.ring.count
4	4	Hyper.wiener.index
5	5	Szeged.index
6	6	Ring.count.of.atom
7	7	pI
8	8	Quaternary_structure
9	9	PLGA_Mw
10	10	La_to_Gly
11	11	PVA_conc_inner_phase
12	12	PVA_conc_outer_phase
13	13	PVA_Mw
14	14	Inner_phase_volume
15	15	Encaps_rate
16	16	PLGA_conc
17	17	PLGA_to_Placticizer
18	18	diss_pH
19	19	diss_add
20	20	Prod_method
21	22	Asymmetric.atom.count.1
22	23	Hyper.wiener.index.1
23	24	Szeged.index.1
24	25	count
25	26	pH_14_logd
26	27	bpKa2
27	29	Cyclomatic.number.2

As one can see in the example there were only two models used "pcr","pls", to use all available models please set option `Used.funcRegPred="all"`. The results can be presented on a bar plot (see Figure 1). Then the arbitrary feature reduction can be applied.

For classification problems, two lists of outputs.

1. Feature ranking and errors (F-measure) for models:

```
> # Print out the Variable importance results for F-measure scaling
> print(myRES.class_tab)
```

2. Raw F-measures for each model

```
> # Print out the generalization error for models
> print(myRES.class_rawError)
```

As one can see in the example there were only two models used "knn","rpart", to use all available models please set option `Used.funcClassPred="all"`. The results can be presented on a bar plot as the previous ones. Then the arbitrary feature reduction can be applied.

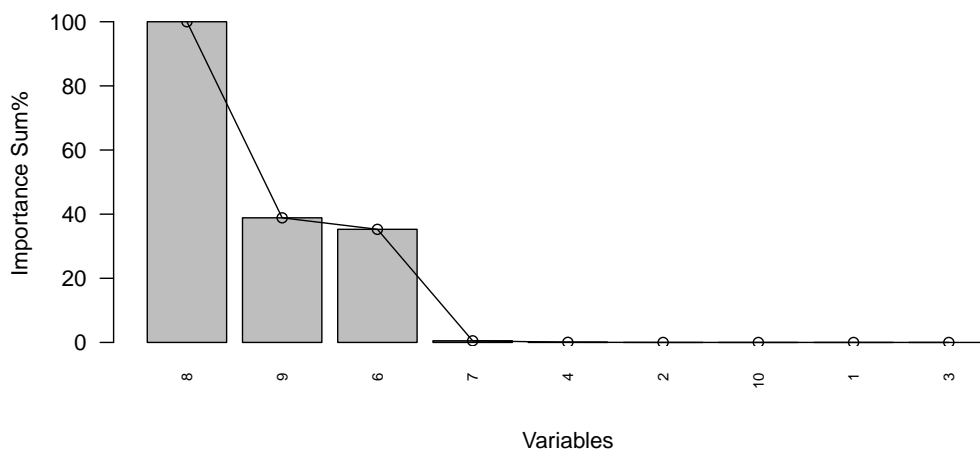


Figure 1: A sum of feature ranking of models trained and tested on `dataset.train`, two models were used "pcr", "pls".

6 Known issues

1. In some cases during model development stage users can encounter "caught segfault" errors. It is highly dependent on the input data and the model. The nature of the error prevents function `fscaret()` returning proper results, therefore no scaling of variable importance is done, and no summary of feature ranking is presented.

The way around is to exclude the troublesome method from calculations. If you encounter an odd behaviour of your working script, e.g. results of an object `myFS` in `VarImp` is an empty `list()`, search for "segfault" in a Rout file. In the example given below "partDSA" is the trouble maker. Then run once again computations.

```
> library(fscaret)
> myFuncRegPred <- funcRegPred[which(funcRegPred!="partDSA")]
> print(funcRegPred)
> myFS<-fscaret(trainDF, testDF, myTimeLimit = 12*60*60, preprocessData=TRUE,regPred=TRUE,
+             Used.funcRegPred=myFuncRegPred, with.labels=TRUE,
+             suppress.output=TRUE, no.cores=NULL, saveModel=FALSE)
```

7 Acknowledgments

This work was funded by Poland-Singapore bilateral cooperation project no 2/3/POL-SIN/2012.

8 References

1. Szlek J, Paclawski A, Lau R, Jachowicz R, Mendyk A. Heuristic modeling of macromolecule release from PLGA microspheres. *International Journal of Nanomedicine*. 2013;8(1); 4601 - 4611. [link to webpage](#)
2. Szlek, J., Paclawski, A., Lau, R., Jachowicz, R., Mendyk, A. Heuristic modeling of macromolecules release from PLGA microspheres. Conference proceedings. Gdansk, May 24-25, 2013.[Abstract book](#)
3. Paclawski A, Szlek J, Lau R, Jachowicz R, Mendyk A. Empirical modeling of the fine particle fraction for carrier-based pulmonary delivery formulations. *International Journal of Nanomedicine*. 2015;10(1); 801 - 810. [link to webpage](#)