**The problem**: We have data, and we need to create models (xgboost, random forest, regression, etc). Each one of them has its constraints regarding data types.  
Many *strange* errors appear when we are creating models just because of data format.

The new version of funModeling 1.9.3 (Oct 2019) aimed to provide quick and clean assistance on this.

Based on some *messy* data, we want to run a random forest, so before getting some weird errors, we can check…

Example 1:

# install.packages("funModeling")

library(funModeling)

library(tidyverse)

# Load data

data=read\_delim("<https://raw.githubusercontent.com/pablo14/data-integrity/master/messy_data.txt>", delim = ';')

# Call the function:

integ\_mod\_1=data\_integrity\_model(data = data, model\_name = "randomForest")

# Any errors?

integ\_mod\_1

##

## ✖ {NA detected} num\_vessels\_flour, thal, gender

## ✖ {Character detected} gender, has\_heart\_disease

## ✖ {One unique value} constant

Regardless the "one unique value", the other errors need to be solved in order to create a random forest.

Alghoritms have their own data type restrictions, and their own error messages making the execution a hard debugging task… data\_integrity\_model will alert with a common error message about such errors.

**Introduction**

data\_integrity\_model is built on top of data\_integrity function.

It checks:

* NA
* Data types (allow non-numeric? allow character?)
* High cardinality
* One unique value

**Supported models **

It takes the metadata from a table that is pre-loaded with funModeling

head(metadata\_models)

## # A tibble: 6 x 6

## name allow\_NA max\_unique allow\_factor allow\_character only\_numeric

##

## 1 randomForest FALSE 53 TRUE FALSE FALSE

## 2 xgboost TRUE Inf FALSE FALSE TRUE

## 3 num\_no\_na FALSE Inf FALSE FALSE TRUE

## 4 no\_na FALSE Inf TRUE TRUE TRUE

## 5 kmeans FALSE Inf TRUE TRUE TRUE

## 6 hclust FALSE Inf TRUE TRUE TRUE

The idea is anyone can add the most popular models or some configuration that is not there.  
There are some redundancies, but the purpose is to focus on the model, not the needed metadata.  
This way we don’t think in no NA in random forest, we just write randomForest.

Some custom configurations:

* no\_na: no NA variables.
* num\_no\_na: numeric with no NA (for example, useful when doing deep learning).

**Embed in a data flow on production **

Many people ask for typical questions when interviewing candidates. I like these ones: *"How do you deal with new data?"* or *"What are the considerations you have when you do a deploy?"*

Based on our first example:

integ\_mod\_1

##

## ✖ {NA detected} num\_vessels\_flour, thal, gender

## ✖ {Character detected} gender, has\_heart\_disease

## ✖ {One unique value} constant

We can check:

integ\_mod\_1$data\_ok

## [1] FALSE

data\_ok is a flag useful to stop a process raising an error if anything goes wrong.

**More examples **

Example 2:

On mtcars data frame, check if there is any variable with NA:

di2=data\_integrity\_model(data = mtcars, model\_name = "no\_na")

# Check:

di2

## ✔ Data model integrity ok!

Good to go?

di2$data\_ok

## [1] TRUE

Example 3:

data\_integrity\_model(data = heart\_disease, model\_name = "pca")

##

## ✖ {NA detected} num\_vessels\_flour, thal

## ✖ {Non-numeric detected} gender, chest\_pain, fasting\_blood\_sugar, resting\_electro, thal, exter\_angina, has\_heart\_disease

Example 4:

data\_integrity\_model(data = iris, model\_name = "kmeans")

##

## ✖ {Non-numeric detected} Species

**Any suggestions?**

How about time series? I took them as: numeric with no na (model\_name = num\_no\_na). You can add any new model by updating the table metadata\_models.

And that’s it.