Islas Canarias
Del 15 al 19 de noviembre de 2021

VALIDACIÓN DE FICHEROS ADMINISTRATIVOS DE CARÁCTER ECONÓMICO

Matias Ledesma Sereno
EUSTAT
m-ledesma@eustat.eus

Ander Iparraguirre García
EUSTAT
a-iparraguirregarcia@eustat.eus

PALABRAS CLAVE
Validación, Edición, Validate R-package

1. INTRODUCCIÓN

El proceso de validación es el paso posterior a la recogida del dato. Ocasionalmente, dicho dato suele contener alteraciones que desvirtúan la calidad de las observaciones. El proceso por el cual se aumenta la calidad del dato se conoce como "data cleaning".

Generalmente, los datos brutos pueden almacenar una amplia variedad de errores (errores tipográficos, errores de incoherencia, errores de formatos…). Este desajuste en los ficheros puede producir una distorsión en el análisis, por ello es de suma importancia verificar la validez de los ficheros.

2. OBJETIVOS

El objetivo de este trabajo ha sido establecer un proceso estándar de validación para ficheros de carácter económico, bien de encuesta o provenientes de registros administrativos. Para ello se han utilizado el software y estándares desarrollados bajo el grupo ValiDat Integration[1] de ESSnet de Eurostat.

EUSTAT recoge distintos ficheros administrativos de carácter económico para la producción estadística. En este caso el fichero empleado ha sido el del Impuesto de sociedades del 2018.

3. METODOLOGÍA

"La validación consiste en verificar si un valor o una combinación de valores es aceptable en un conjunto de combinaciones" (Di Zio, M. et al, 2018). La validación nace de una lógica binaria donde se acepta o se rechaza el dato dadas las reglas previamente

[1] https://ec.europa.eu/eurostat/cros/content/essnet-validat-integration

designadas. El aumento de la calidad del fichero suele estar sujeta a una implementación elaborada del proceso de validación, a una limpieza de datos y a una imputación óptima.

El proceso de validación consiste en un proceso de verificación. Trata de hacer suposiciones y verificar si el dato es correcto (Van der Loo & De Jonge, 2018). En el proceso de la validación, valga la redundancia, se valida o invalida un conjunto de datos que pertenecen a un registro.

Se ha empleado el software de R del repositorio de "Data cleaning" de Mark Van der Loo y Edwin de Jonge. Hay varios paquetes, pero el principal es el paquete "Validate" y hace uso de otras librerías que realizan tareas auxiliares Localización de Errores, Imputación, Validación de reglas….

Para elaborar un sólido proceso de validación es necesario tener un conjunto de reglas definidas que permitan aumentar la calidad del fichero. Hay que tener en cuenta todos los fallos posibles que pueden tener los datos en cada variable (formato, rango, condiciones simples…).

3.1 REGLAS DE VALIDACIÓN

Las reglas usadas en este proceso de validación siguen las normas de carácter contable. Estas reglas han sido elaboradas por el Área de Estadísticas Estructurales Económicas de EUSTAT. El proceso de validación cuenta con 112 reglas básicas que buscan aumentar la calidad del fichero. De estas 112, algunas de las reglas son de coherencia jerárquica entre los campos y otras de verificación de los valores positivos o negativos. El proceso de validación de los formatos de cada campo se ha realizado en la carga automática de los ficheros en la base de datos.

3.2 VALIDATETOOLS

Se ha utilizado el paquete de "Validatetools"[2] para la validación de las reglas. Mediante esta librería se busca que las reglas sean factibles, coherentes entre si y no redundantes.

Ejemplo del código de R para la validación de las reglas

```
# Iniciar los paquetes necesarios
> library(validate)
> library(validatetools)

# Abrir el document .txt de las reglas
> rules <- validator(.file = rules.txt)

# Verificar el .txt con reglas
> is_infeasible(rules)
#>  FALSE

# Detectar reglas no factibles
> detect_infeasible_rules(rules)
#>

# Eliminar las reglas que generan conflicto en el set de reglas
> make_feasible(rules)
```

[2] https://github.com/data-cleaning/validatetools

```
#> No infeasibility found, returning original rule set
#> Object of class 'validator' with 112 elements:
#>  V001: IS_ACT_NO_CORR == IS_ACT_INMOV_INT +
#> IS_ACT_INMOV_MAT + IS_ACT_INVR_INMO +
#> IS_ACT_INVR_LAR + IS_ACT_INVR_FINAN_LAR +
#> IS_ACT_IMPU_DIFER + IS_ACT_DEU_COMER
#> …
```

En nuestro caso, las reglas han sido las que habitualmente son utilizadas por el Área de Estadísticas Estructurales Económicas de EUSTAT y por ello no contienen ningún error, pero este debería de ser un proceso previo a aplicar para cualquier conjunto de reglas.

3.3 VALIDATE

El proceso de validación consiste en un proceso de verificación. Trata de hacer suposiciones y verificar si el dato es correcto (Van der Loo & De Jonge, 2018). En el proceso de validación, valga la redundancia, se valida o invalida un conjunto de datos que pertenecen a un registro.

El paso de validación es el paso previo a la identificación del error y a la limpieza de datos. Ya que con el diagnóstico elaborado se puede identificar que variables desajustan el proceso de validación. A posteriori, y una vez se tienen identificados las posiciones de los fallos se puede plantear el uso de distintos métodos de imputación y/o limpieza para aumentar la calidad del fichero.

El paquete "Validate"[3] dispone de unas opciones (voptions) que permiten realizar la tarea de verificación más al detalle[4].

- "na.value (NA, TRUE, FALSE) funciona para identificar los valores NA como NA, TRUE o FALSE en el proceso de validación se utiliza el parámetro na.value = NA, TRUE, FALSE."
- "raise ("none", "error", "all") controla si los métodos de confrontación capturan o aumentan excepciones."
- "lin.eq.eps ("valor numérico") es útil para aumentar o disminuir las igualdades en el proceso de validación."
- "reset() tiene la opción de resetear las opciones a valores predeterminados."

El proceso de análisis es simple gracias a esta librería. Solo se debe abrir el fichero en el que se desea ejecutar la verificación junto con el set de reglas indicado. Dicho set de reglas debe tener el nombre exacto de las variables a las que aplica. Este set de reglas guardado en formato .txt será ejecutado con la función "validator"[5] (validator(.file = rules.txt)). La función "confront"[6] es la que se encarga de realizar el proceso de verificación. Dicha función genera un objeto S4 del cual se pueden extraer múltiples observaciones (values(), summary(), plot(), violating()…).

```
# Iniciamos el paquete necesario para procesar la validación
```

[3] Validate: https://github.com/data-cleaning/validate
[4] Voptions pag 80: https://cran.r-project.org/web/packages/validate/validate.pdf
[5] Función validator: https://github.com/data-cleaning/validate/blob/master/pkg/R/validator.R
[6] Función confront: https://github.com/data-cleaning/validate/blob/master/pkg/R/confrontation.R

```
> library(validate)

# Se abren y leen los documentos indicados
> rules <- “path_directory/rules.txt”
> rules <- validator(.file = rules)
> data <- “path_directory/data.csv”
> data <- read.csv(data,header = TRUE, sep = ";")

# Se usa la función confront la cual se vinculan las reglas y los datos
# Se generan dos objetos S4 con el fin de compararlos según distinto nivel de tolerancia
> out_main <- confront(data, rules, key="IS_NIF")
> out_main_2 <- confront(data, rules, key="IS_NIF", lin.eq.eps = 2)
> df <- data.frame(pass = c(sum(summary(out_main)$passes),
>                           sum(summary(out_main_2)$passes)),
>                  fails = c(sum(summary(out_main)$fails),
>                            sum(summary(out_main_2)$fails)))
> rownames(df) <- c("lin.eq.eps = 0", "lin.eq.eps = 2")
> print(df)
>#                    pass   fails
># lin.eq.eps = 0   6337274  281407
># lin.eq.eps = 2   6441343  177338
```

3.4 ERROR LOCATE

Es tan importante evaluar las reglas de validación como identificar en que variable o variables se encuentra dichos errores. Esta tarea se puede realizar utilizando el paquete auxiliar “errorlocate”. La función “locate_errors”[7] llama a la función “fh_localizer”. Dicho algoritmo tiene el propósito de encontrar el menor número de cambios en variables para que se cumplan el mayor número de reglas siguiendo el método de Fellegi-Holt (Van der Loo & De Jonge, 2018, 159). Esta función genera un objeto S4 del cual se puede extraer la posición de los ítems con variables que se detectan como fallidas. Cabe destacar, que “locate_errors” permite asignar pesos a las variables evaluadas para dar prioridad en la detección de errores.

```
# Dataframe de ejemplo
> example_df <- data.frame(IS_ACT_NO_CORR=37, IS_ACT_INMOV_INT = -5,
> IS_ACT_INMOV_MAT = 8, IS_ACT_INVR_INMO = 3, IS_ACT_INVR_LAR = 4,
> IS_ACT_INVR_FINAN_LAR = 6, IS_ACT_IMPU_DIFER = 1, IS_ACT_DEU_COMER = 0)

# Reglas V1 y V2
> rules <- validator(IS_ACT_NO_CORR == IS_ACT_INMOV_INT + IS_ACT_INMOV_MAT
> + IS_ACT_INVR_INMO + IS_ACT_INVR_LAR + IS_ACT_INVR_FINAN_LAR +
> IS_ACT_IMPU_DIFER + IS_ACT_DEU_COMER, IS_ACT_INMOV_INT >= 0)

# Se aplica la función que detecta errores
# Parámetro opcional weight = c(8,7,6,5,4,3,2,1)
> locate_errors(example_df, rules, Ncpus = detectCores())$errors
>#  IS_ACT_NO_CORR IS_ACT_INMOV_INT IS_ACT_INMOV_MAT
>#           FALSE             TRUE            FALSE
>#  IS_ACT_INVR_INMO IS_ACT_INVR_LAR IS_ACT_INVR_FINAN_LAR
>#             FALSE           FALSE                 FALSE
>#  IS_ACT_IMPU_DIFER IS_ACT_DEU_COMER
>#              FALSE            FALSE
```

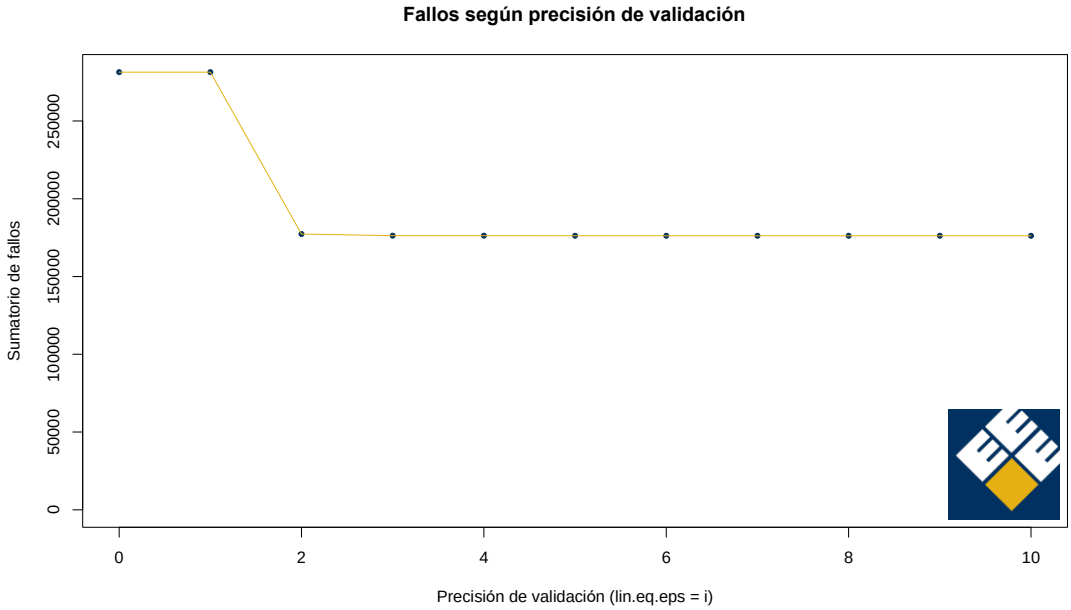
[7] Función locate_errors: https://github.com/data-cleaning/errorlocate/blob/master/R/locate-errors.R

4. RESULTADOS

El gráfico número 1 indica que ampliando el margen de tolerancia se permite aceptar más datos. A medida que se aumenta el parámetro "lin.eq.eps = i" los fallos identificados en el fichero decrecen y se estabilizan. Como norma general, los datos de actividades contables se suelen almacenar con dos decimales. En este caso los datos con los que se trabaja son números enteros. Por lo tanto, se asume cierta perdida de calidad del fichero tras el redondeo.

En el *gráfico 1* se puede ver la evolución de la validación según se va cambiando el parámetro anterior. Cuando la validación es un proceso de tolerancia 0 (lin.eq.eps = 0), los fallos totales ascienden a 281.407 en 59.098 registros y 112 reglas. Sin embargo, cuando se amplía tolerancia hasta 2, los fallos disminuyen en más de 100.000 unidades (siendo un total de 177.338 errores con la nueva tolerancia).

Gráfico 1



La multiplicación de los items por el número de reglas nos da como resultado la cantidad de pruebas de validación por items ejecutadas en dicho fichero.

$$n_{validación*items} = n_{items} * n_{reglas}$$

En este caso se han producido 6.618.976 validaciones. Para obtener la calidad bruta de los datos según el número de verificaciones se tienen en cuenta los registros que superan cada regla anteriormente designada. En este caso, el número de datos que supera el proceso de validación es de 95,74% cuando "lin.eq.eps = 0". Sin embargo, la frecuencia relativa aumenta a 97,32% mientras el parametro "lin.eq.eps = 2" cumple lo siguiente.

$$calidad\_bruta\_datos = \frac{\sum_{i=1}^{n} n_{reglas\_aceptados_i}}{n_{pruebas}} * 100$$

Las reglas que fallan con mayor regularidad son aquellas que consisten en sumas de distintas variables del fichero (80,27 % de los fallos pertenecen a este tipo de reglas cuando la tolerancia es igual a 0). Cuando la tolerancia se iguala a 2 la frecuencia relativa se disminuye hasta el 68,69%. Como dato curioso, el 99,997 % de los valores admitidos nuevamente pertenecen a reglas que contienen sumas de distintas variables de carácter contable.

La regla que tiene como resultados la variable "Otros Gastos de Explotación" ("IS_PYG_GAS_EXPL" equivalente a V98) es la que más fallos acumula. En el 82,9 % de registros no supera esta. De cerca le acompaña la regla que tiene como resultado la variable "Importe neto de la Cifra de Negocios" (IS_PYG_CIFRANEG, V78). La cual le sucede con un 78,67% de ítems. Tal y como se puede apreciar, el apartado contable de "Perdidas y Ganancias" es el que más fallos acumula por ítem.

Tabla 1

N	% de fallos
V98	82,90 %
V78	78,67 %
V103	55,64 %
V84	52,49 %
V106	32,67 %

Las reglas V106, V9 y V31 reducen sustancialmente su numero de errores cuando se procede de lin.eq.eps = 0 a lin.eq.eps = 2. La regla 106 pasa de tener 32,67 % de ítems con errores a 1,62 %. La regla 9 disminuye de un 23,8 % de ítems con error a tan solo un 0,1 %. Para la regla número 31 la disminución también es significativa. Ya que se pasa de un 25,68 % a un 0,08 %. Por otro lado, también se encuentran reglas que tras este parámetro (lin.eq.eps = 2) eliminan los errores identificados. Estas reglas son aquellas que tienen como resultado las variables de; activo total, patrimonio neto, pasivo no corriente y resultado del ejercicio procedente de operaciones continuadas (*Tabla 2*).

Tabla 2 resumen por regla del número de aciertos y fallos según el nivel de tolerancia (0 o 2) cuando fail_0 > 1 y fail_2 = 0

Regla	pass_0	pass_2	fail_0	fail_2	fail_diff
V25	48494	59098	10604	0	0
V26	44634	59098	14464	0	0
V27	59095	59098	3	0	0
V29	58357	59091	734	0	0
V45	58090	59091	1001	0	0
V112	55587	59098	3511	0	0

No obstante, no todas las reglas tienen un aumento de validación importante al aumentar la tolerancia. Tales son los casos de las reglas; V98, V78, V84 y V103. Las reglas anteriormente citadas, tan solo tienen un leve mejoria en la que se aceptan; 14, 21, 16 y 0 items por norma.

Las 6 ultimas reglas del *gráfico 2* desaparecen en el *grafico 3* (salvo la V30). Esto puede suceder porque dichas reglas consiguen aceptar items que se pueden ver condicionados por el redondeo.

La cantidad de registros que cumplen todas las reglas no supera el 10% cuando su tolerancia es igual a 0 (*gráfico 4*). Es decir, aquellos registros que cuentan con una coherencia contable impecable son una minoría. A pesar de incrementar el margen de confianza en la verificación, la mejora no es sustancial (*gráfico 5*). La conclusión de esta lectura es que pocos registros no tienen ningún fallo contable. A modo de conclusión, la norma es que tener fallos contables es habitual incluso cuando la tolerancia es igual a 2.

El diagnostico de validación permite entender cuales son las reglas que más influyen en la calidad del fichero. Las variables de dichas reglas, deben de ser limpiadas o imputadas para mejorar la condición del fichero. No obstante, este proceso permite focalizar los recursos en aquellas variables de las reglas que más errores contienen.

Gráfico 2

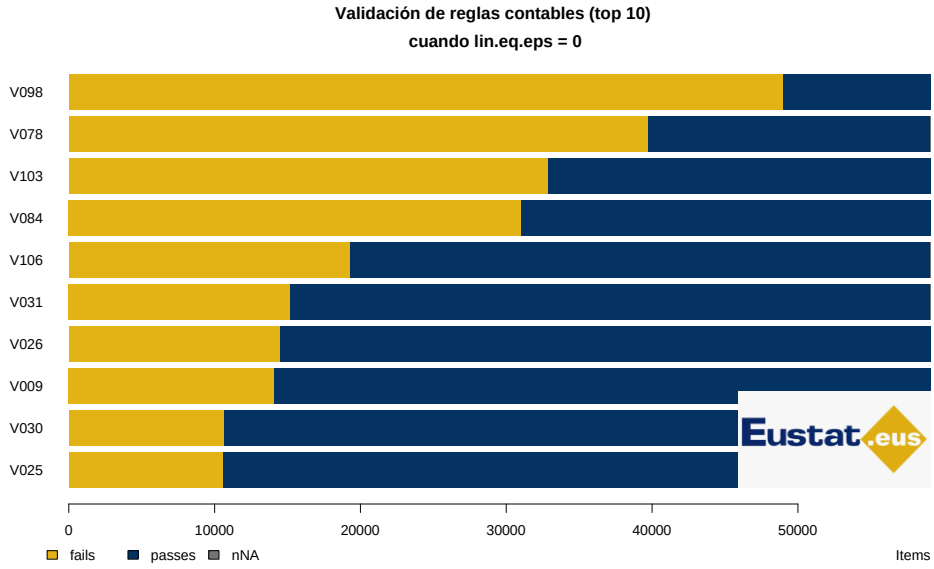


Gráfico 3

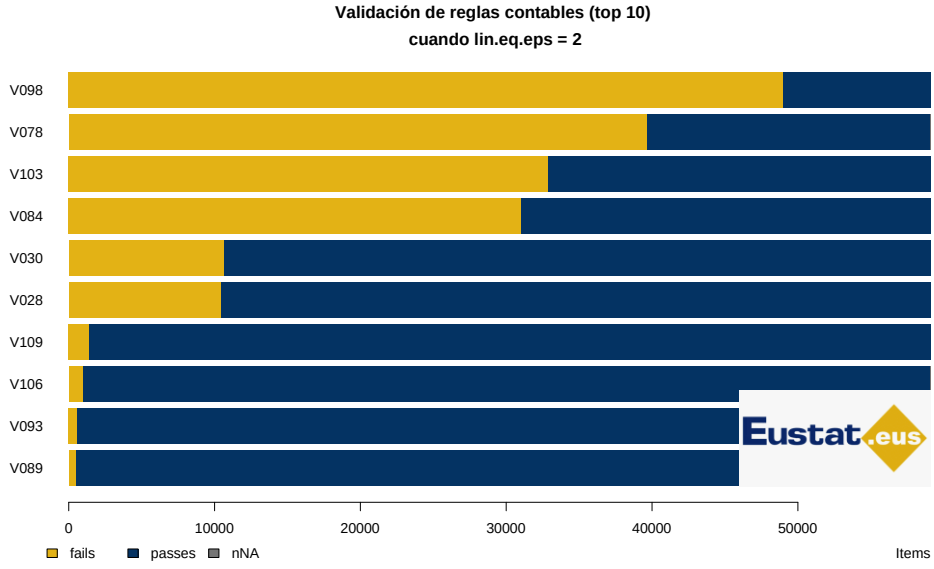


Gráfico 4

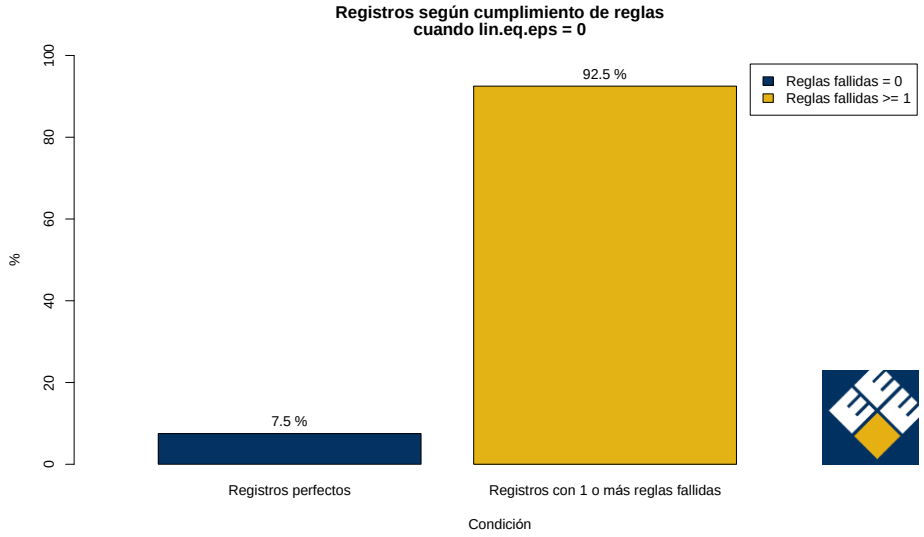
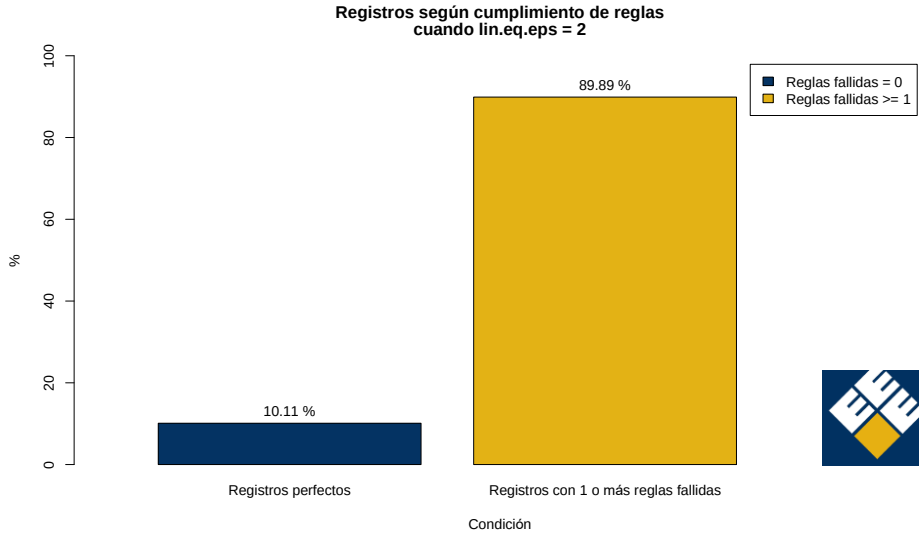


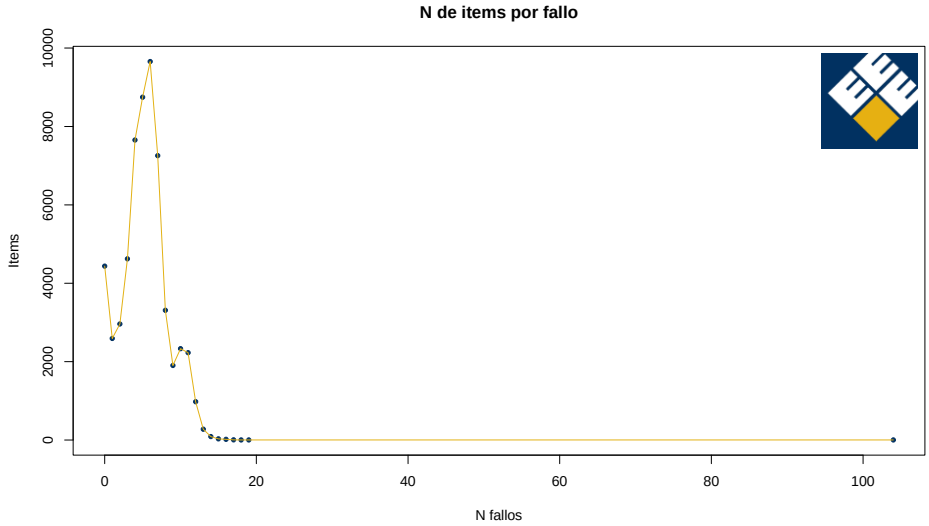
Gráfico 5



El fichero reescalado del Impuesto de sociedades de 2018[8] cuenta con 314.406 errores identificados cuando summary (locate_errors(data, rules, Ncpus = 6, timeout = 10))[2]. Lo cual representa una calidad total del fichero del 95,21 %. Este dato indica que la calidad del fichero es muy elevada. No obstante, hay registros que concentran dichos fallos y es poco habitual hallar resgistros sin ningún fallo (7,51 %). Por otro lado, acumular 6 errores es la moda (16,34 %). Sin embargo, se puede encontrar un registro anómalo con 104 fallos como se aprecia en el *gráfico 6*.

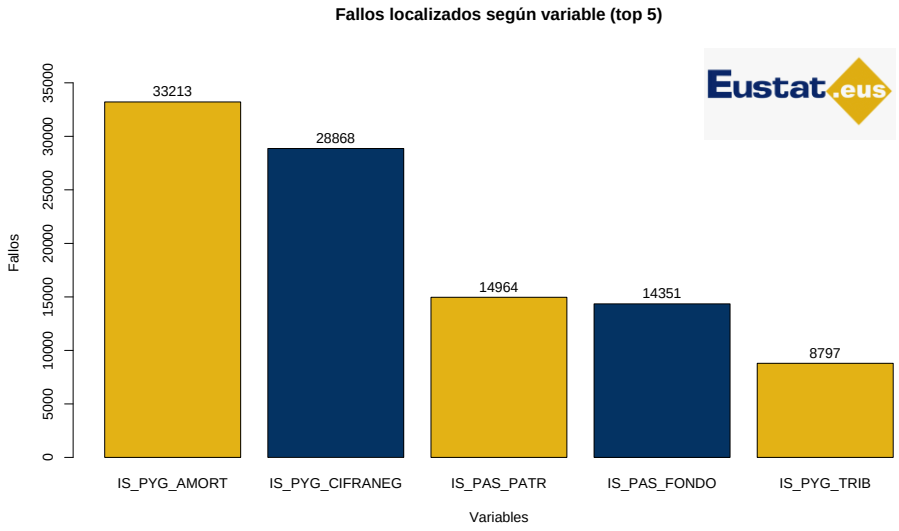
[8] Reescalado debido al error "This might be indication that these column(s) should be rescaled.
(the problem because otherwise numerically unstable)". Por ello se han reescalado los valores (data/1e6).

Gráfico 6



El *gráfico 7* muestra las 5 variables de los datos que más errores almacenan. Las mismas suponen un 31,87 % del total de fallos. La variable amortización del inmovilizado (IS_PYG_AMORT) está presente en las reglas V103 y V106, las cuales están presentes en el top 5[9]. La segunda variable que más errores acumula (IS_PYG_CIFRANEG) está relacionada, con la regla V78, la cual es la segunda que más errores almacena cuando "lin.eq.eps = 0 y 2". El vínculo entra reglas fallidas t variables fallidas que pertenecen a esas reglas es apreciable.

Gráfico 7



[9] Ver gráfico 2 y 3

5. CONCLUSIONES

Se ha elaborado un programa en R para la estandarización de la validación con reglas fijas pero modificables para ficheros de carácter económico. El proceso pasa las fases básicas de la validación y produce un informe con el resumen de las reglas, validaciones efectuadas localización de errores y un indicador de la calidad del fichero.

Se aprecia que la calidad del fichero es elevada (95,21 %). No obstante, los registros sin ningún error son minoría (7,51 %). Es decir, en general se trabaja con un fichero de alta calidad, pero en el que hay muchas empresas que incluyen datos que son un tanto incoherentes.

Finalmente, las herramientas utilizadas facilitan mucho la tarea de validación ya que es muy intuitivo y dispone de amplias posibilidades. Aun así, se han encontrado problemas ajenos a la librearía como la necesidad de reescalar el fichero.

6. BIBLIOGRAFÍA

- Bantilan, N. (2020). pandera: Statistical Data Validation of Pandas Dataframes. Accessed: 30 Aug 2021. URL:

  https://conference.scipy.org/proceedings/scipy2020/pdfs/niels_bantilan.pdf

- Di Zio, M., Fursova, N., Gelsema, T., Gießing, S., Guarnera, U., Petrauskienė, J., & Walsdorfer, K. (2018). Methodology for data validation 2.0. Essnet Validat Foundation, Brussels, Belgium, 1-85. Accessed: 13 Sept 2021.

  https://ec.europa.eu/eurostat/cros/system/files/ess_handbook_methodology_for_data_validation_v2.0_-_rev2018_0.pdf

- Jeff Reback, jbrockmendel, Wes McKinney, Joris Van den Bossche, Tom Augspurger, Phillip Cloud, Simon Hawkins, gfyoung, Sinhrks, Matthew Roeschke, Adam Klein, Terji Petersen, Jeff Tratner, Chang She, William Ayd, Patrick Hoefler, Shahar Naveh, Marc Garcia, Jeremy Schendel, … Kaiqi Dong. (2021). pandas-dev/pandas: Pandas 1.3.2 (v1.3.2). Zenodo. Accessed: 30 Aug 2021.

  https://doi.org/10.5281/zenodo.5203279

- Van der Loo, M. (2021) The Data Validation Cookbook version 1.0.4. Accessed: 30 Aug 2021. URL:

  https://data-cleaning.github.io/validate

- Van der Loo, M. (2021). The Data Validation Cookbook. Cran R Project. Accessed: 30 Aug 2021.

  https://cran.r-project.org/web/packages/validate/vignettes/cookbook.html

- Van der Loo, M. & de Jonge, E. (2020). Data Validation Infrastructure for R. Journal of Statistical Software, Accepted for publication. Accessed: 30 Aug 2021

  https://arxiv.org/abs/1912.09759

- Van der Loo, M., & De Jonge, E. (2018). Statistical data cleaning with applications in R. John Wiley & Sons.
- Van der Loo, M. P. J., & de Jonge, E. (2021). Data Validation Infrastructure for R. Journal of Statistical Software, 97(10), 1–31. Accessed: 4 Oct 2021.

  https://doi.org/10.18637/jss.v097.i10